STATISTICAL MACHINE LEARNING FOR COMPLEX CLASSIFICATION PROBLEMS

Mingyuan Zhang

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2024

Dedicated to my parents

ACKNOWLEDGEMENT

Firstly, I would like to express my deepest gratitude to my advisor, Shivani Agarwal, for her invaluable guidance, support, and encouragement throughout the course of my doctoral research. Her expertise and mentorship have been instrumental in shaping this thesis.

Secondly, I am thankful to the members of my thesis committee—Shivani Agarwal, Weijie Su, Jacob Gardner, Surbhi Goel, and Ambuj Tewari—for their insightful feedback and constructive criticism.

I would like to acknowledge the contributions of my excellent collaborators: Shivani Agarwal, Sheng Gao, Hangfeng He, Jane Lee, Qiang Ning, Harish G. Ramaswamy, Dan Roth, and Hua Wang. I am grateful to Britton Carnevali, Zack Ives, Boon Thau Loo, and Mayur Naik, who have helped me navigate logistics during my Ph.D. journey.

I would also like to thank my friends—Alejandro, Bowen, Ding, Duc, Hangfeng, Haoyu, Jiani, Li, Muhao, Qing, Rongguang, Shaoru, Shenghan, Xiangzhuo, Xin, Xingran, Xuanyu, Yan, Yao, Yi, Zebang, Zihan, and Ziyang—who have provided support and encouragement during the past few years.

Lastly, I am indebted to my family, especially my parents and wife, for their unwavering love, understanding, and patience during this journey. Their constant support has been a source of strength and motivation. I would like to thank my mother and father for loving me unconditionally.

ABSTRACT

STATISTICAL MACHINE LEARNING FOR COMPLEX CLASSIFICATION PROBLEMS

Mingyuan Zhang

Shivani Agarwal

Classification is a fundamental problem in statistical machine learning. It seeks to classify instances into several classes. Binary and multiclass classification settings are the most basic and well-studied settings. Yet, real-world classification problems often involve additional complexities. In this thesis, we focus on complex classification problems in statistical machine learning by exploring three dimensions of complexity: 1) Complex label space; 2) Complex learning setting; 3) Complex performance measure. These complexities pose significant challenges in real-world applications and necessitate the development of novel methodologies that can effectively handle such issues. The goal of my thesis research is to study these complex classification problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

Classification problems represent a fundamental and prevalent challenge in the field of machine learning, where the objective is to assign one or more predefined labels (categories) to a given input instance. These problems arise in a diverse array of domains, such as natural language processing, computer vision, healthcare, and finance, among others. Machine learning algorithms employed to solve classification tasks have evolved over time, from traditional methods like decision trees, logistic regression, and support vector machines, to newer approaches involving deep learning.

Despite significant advancements, classification problems continue to pose unique challenges, particularly when dealing with complex scenarios involving multi-label classification (Zhang and Zhou, 2014), noisy or partial labels (Natarajan et al., 2013; van Rooyen and Williamson, 2017), and non-decomposable performance measures (Narasimhan et al., 2014, 2015). Developing good and efficient algorithms that can effectively tackle these complexities is crucial for enhancing the applicability of machine learning algorithms in real-world settings and addressing emerging challenges across various domains. In particular, it is desirable to design *Bayes consistent* algorithms, meaning that as the size of the training sample grows, the performance (which can be gain or loss) of the learned classifier converges to the *Bayes optimal performance*, which is the best possible performance for a given classification problem. Bayes consistency is an important property for a learning algorithm because it guarantees that the learned classifier's performance will improve as more data is collected, eventually reaching the optimal performance that is theoretically achievable.

In this thesis, we study complex classification problems in statistical machine learning by exploring three dimensions of complexity:

- **Complex label space.** This involves a class of machine learning problems where the goal is to predict structured output objects, rather than simple categorical values as in binary or multiclass classification. In other words, the label space is composed of complex interdependent structures rather than single values. These problems are called *structured prediction problems*, and often involve modeling the relationships and dependencies between the elements of the output structure, which makes them more challenging than traditional prediction tasks. Examples of structured prediction problems include multi-label classification, sequence labeling, parsing, image segmentation, and many others. Methods proposed to address structured prediction problems often aim to capture and exploit the dependencies and relationships within the output structures to make more accurate and coherent predictions.

- **Complex learning setting.** This refers to scenarios in machine learning that involve additional challenges or complexities beyond traditional supervised learning tasks. These complexities can arise due to various factors, such as the nature of the data, the learning environment, or the desired output. Some examples of complex learning settings include but are not limited to noisy labels, missing or partial labels, imbalanced data, semi-supervised learning, weakly supervised learning, active learning, transfer learning, online learning, multi-task learning, and reinforcement learning. These complex learning settings often require specialized algorithms, or adaptations of existing methods to effectively address the challenges they present and achieve satisfactory performance.

- **Complex performance measure.** Complex performance measures are also called *non-decomposable* performance measures. In contrast to the standard 0-1loss or more general cost-sensitive losses (which are *linear* functions of the confusion matrix of a classifier), non-decomposable performance measures are general (usually *nonlinear*) functions of the confusion matrix. Such performance measures take complex forms, and cannot be expressed as the expectation or sum of a loss on individual examples. These performance measures often involve intricate calculations that account for dependencies or interactions between different aspects of the predictions. They are particularly relevant in scenarios where traditional performance measures (0-1loss or cost-sensitive losses) may not adequately capture the nuances of the problem at hand. Examples of non-decomposable performance measures include F_1 score, Jaccard measure, H-mean, G-mean, Q-mean, area under the ROC curve (AUC-ROC), area under the Precision-Recall curve (AUC-PR), and others. Designing algorithms to learn good classifiers for non-decomposable performance measures can be challenging, as they often have to account for the dependencies and interactions between the predictions of instances.

The objective of this thesis is to investigate classification problems encompassing one or more of the aforementioned complexities. We adopt a principled and rigorous approach, employing mathematical formalisms and theoretical analyses to develop algorithms tailored to address these challenges. Additionally, we offer theoretical guarantees for the devised algorithms, demonstrating their Bayes consistency properties to ensure optimal performance as the amount of training data increases. We believe that this study will enhance our understanding of complex classification problems in machine learning and foster the development of more principled algorithms to effectively tackle these challenges.

1.1. Organization

Our research was generally organized into the following stages:

1. **Literature review:** We conducted a literature review in the field of complex classification problems, focusing on the three dimensions of complexity mentioned earlier. This review covers both traditional and more recent methods, along with their strengths and weaknesses. It also identifies existing gaps and open research questions in the field.

2. **Problem formulation:** Based on the literature review, we selected specific complex classification problems that warrant further investigation. We then rigorously defined these problems and formulated them in mathematical terms. This provides a solid foundation for the subsequent development of algorithms and theoretical analyses.

3. **Algorithm development:** For each of the selected problems, we proposed new algorithms or adapted existing methods to address the complexities involved. This may involve designing novel learning techniques or integrating modifications to existing algorithms to make them more suitable for the specific challenges associated with the problem.

4. **Theoretical analysis:** We conducted a thorough theoretical analysis of the proposed algorithms, focusing on their Bayes consistency properties. This involves proving the convergence of the learned classifiers' performance to the Bayes optimal performance as the size of the training sample increases. We also analyzed other aspects of the algorithms, such as their computational complexity, sample complexity, or generalization bounds.

5. **Empirical evaluation:** We performed an empirical evaluation of the proposed algorithms on both synthetic and real-world datasets, comparing their performance with existing methods. This helps confirm our theoretical findings and demonstrate the practical utility of the devised algorithms in addressing complex classification problems.

6. **Documentation and dissemination:** Finally, we documented our findings, detailing the algorithms, theoretical analyses, and empirical results in a well-structured thesis. Along the way, we disseminated our research through publications in relevant conferences and engaged with the broader research community to promote further advancements in the field.

In the rest of this chapter, we will give an overview of the three complexities and our work. This will serve as a roadmap for the rest of the thesis. In Figure 1.1, we show the works along with their positions with respect to the three complexities.

In particular, Chapter 2 is our published work *Convex Calibrated Surrogates for the Multi-Label F-Measure*. Chapter 3 is our work *Multi-Label Learning for Multiple Performance Measures without Re-training*. Chapter 4 is our published work *Learning from Noisy Labels with No Change to the Training Process*. Chapter 5 is our work *Multiclass Learning from Noisy Labels Using Weighted Losses*. Chapter 6 is our work *Consistent Multi-Label Learning from Noisy Labels*. Chapter 7 is our published work *Multiclass Learning from Noisy Labels for Non-decomposable Performance Measures*. Chapter 8 summarizes this thesis.

1.2. Overview

Complex Label Space

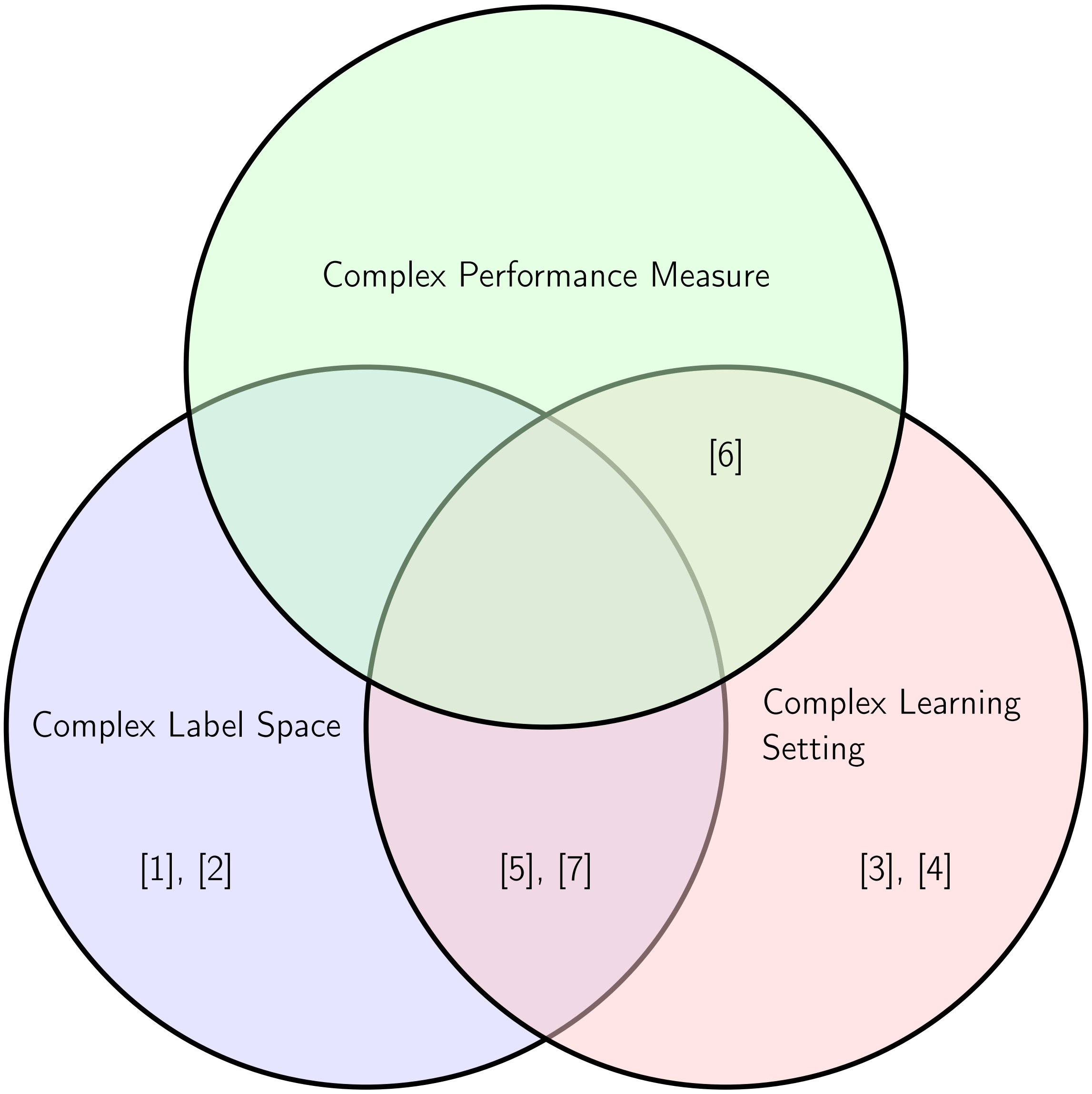

Figure 1.1: Overview of this thesis. [1] Chapter 2: Convex Calibrated Surrogates for the Multi-Label F-Measure (published); [2] Chapter 3: Multi-Label Learning for Multiple Performance Measures without Re-training; [3] Chapter 4: Learning from Noisy Labels with No Change to the Training Process (published); [4] Chapter 5: Multiclass Learning from Noisy Labels Using Weighted Losses; [5] Chapter 6: Consistent Multi-Label Learning from Noisy Labels; [6] Chapter 7: Multiclass Learning from Noisy Labels for Non-decomposable Performance Measures (published); [7] Foreseeing the Benefits of Incidental Supervision (published in collaboration with Hangfeng He, Qiang Ning and Dan Roth; not included in this thesis).

The concept of complex label space in machine learning arises from the need to handle problems with structured output objects rather than simple categorical values as in binary or multiclass classification. In these problems, the label space is composed of intricate, interdependent structures, making the prediction task more challenging than traditional classification tasks. This class of machine learning problems is often referred to as structured prediction problems. Multi-label classification is a prominent subclass of such problems. In multi-label classification, each instance can be associated with multiple labels (tags) simultaneously. This contrasts with the binary or multiclass classification, where each instance is assigned to a single class. Multi-label classification has applications in text categorization, image annotation, and many others (Zhang and Zhou, 2014). Due to the complex nature of multi-label classification, a number of performance measures have been proposed to evaluate multi-label classifiers. They include Hamming loss, subset 0-1loss (subset accuracy), precision, recall, F_1-measure (Dembczynski et al., 2010b; Wu and Zhou, 2017; Menon et al., 2019).

Complex Learning Setting

Complex learning settings in machine learning refer to scenarios that involve additional challenges or complexities beyond traditional supervised learning tasks. These complexities can arise due to various factors, such as the nature of the data, the learning environment, or the desired output. The background of complex learning settings can be traced back to the need to address a diverse range of real-world problems, which often do not fit neatly into the framework of standard supervised learning. Learning from noisy labels is an important family of problems with complex learning settings (Natarajan et al., 2013; van Rooyen and Williamson, 2017). In real-world datasets, labels can often be noisy, incorrect, or ambiguous. When learning from noisy labels, the provided labels in the training data may not always be reliable, potentially leading to a decline in the performance of the machine learning model. This unreliability can arise from various factors, such as errors in human labeling, misalignment between labels and corresponding data, or noise introduced during the data collection process. Learning from such data requires algorithms that can handle and account for label noise while still building accurate and robust models. It has become a rapidly growing area of interest in recent years (Frénay and Verleysen, 2014; Song et al., 2023; Han et al., 2020).

Complex Performance Measure

Complex performance measures, also known as non-decomposable performance measures, play a crucial role in evaluating the effectiveness of machine learning algorithms, particularly in scenarios where standard performance measures such as 0-1 loss or cost-sensitive losses may not adequately capture the nuances of a given problem. Such performance measures are useful in information retrieval, class imbalance settings, and others.

Complex Label Space: Convex Calibrated Surrogates for the Multi-Label F-Measure (Chapter 2; Zhang et al. (2020))

The F-measure is a widely used performance measure for multi-label classification, where multiple labels can be active in an instance simultaneously (e.g. in image tagging, multiple tags can be active in any image). In particular, the F-measure explicitly balances recall (fraction of active labels predicted to be active) and precision (fraction of labels predicted to be active that are actually so), both of which are important in evaluating the overall performance of a multi-label classifier. As with most discrete prediction problems, however, directly optimizing the F-measure is computationally hard. In this work, we explore the question of designing convex surrogate losses that are *calibrated* for the F-measure – specifically, that have the property that minimizing the surrogate loss yields (in the limit of sufficient data) a Bayes optimal multi-label classifier for the F-measure. We show that the F-measure for an s-label problem, when viewed as a $2^s \times 2^s$ loss matrix, has rank at most $s^2 + 1$, and apply a result of Ramaswamy et al. (2014) to design a family of convex calibrated surrogates for the F-measure. The resulting surrogate risk minimization algorithms can be viewed as decomposing the multi-label F-measure learning problem into $s^2 + 1$ binary class probability estimation problems. We also provide a quantitative regret transfer bound for our surrogates, which allows any regret guarantees for the binary problems to be transferred to regret guarantees for the overall F-measure problem, and discuss a connection with the algorithm of Dembczynski et al. (2013). Our experiments confirm our theoretical findings.

Complex Label Space: Multi-Label Learning for Multiple Performance Measures without Re-training (Chapter 3)

Unlike binary or multiclass classification, where 0-1loss is the standard performance measure, there is no such canonical performance measure for multi-label classification, where multiple labels can be active in an instance simultaneously (e.g., in image tagging, multiple tags can be active in any image). Various performance measures have been proposed to assess multi-label classifiers. These include Hamming loss, subset 0-1loss (subset accuracy), precision, recall, and F_1-measure. It has been observed that different algorithms tend to perform variably across different performance measures and there has been progress in understanding the reasons behind these performance variations, identifying which algorithms excel under specific performance measures, and finding connections between different performance measures. Still, there is a lack of principled understanding of whether it is possible to design a multi-label learning algorithm such that it can optimize several performance measures at the same time, meaning that it is not needed to train for different performance measures separately. In this work, we study this problem by utilizing the theory of convex calibrated surrogates. We first show that it is possible to design one convex calibrated surrogate with respect to several performance measures so that one can train using the surrogate once and then apply different post-processing functions to optimize different performance measures. Then we show how to optimize Hamming loss, precision, recall and Top@k using a learned scoring function for F_β-measure. Finally, we provide a regret transfer bound for our method to show it is Bayes consistent.

Complex Learning Setting: Learning from Noisy Labels with No Change to the Training Process (Chapter 4; Zhang et al. (2021))

There has been much interest in recent years in developing learning algorithms that can learn accurate classifiers from data with noisy labels. A widely-studied noise model is that of *class-conditional noise* (CCN), wherein a label y is flipped to a label $\widetilde{y}$ with some associated noise probability that depends on both y and $\widetilde{y}$. In the multiclass setting, all previously proposed algorithms under the CCN model involve changing the training process, by introducing a 'noise-correction' to the surrogate loss to be minimized over the noisy training examples. In this work, we show that this is really unnecessary: one can simply perform class probability estimation (CPE) on the noisy examples, e.g. using a standard (multiclass) logistic regression algorithm, and then apply noise-correction only in the final prediction step. This means that the training algorithm itself does not need any change, and one can simply use standard off-the-shelf implementations with no modification to the code for training. Our approach can handle general multiclass loss matrices, including the usual 0-1 loss but also other losses such as those used for ordinal regression problems. We also provide a quantitative regret transfer bound, which bounds the target regret on the true distribution in terms of the CPE regret on the noisy distribution; in doing so, we extend the notion of strong properness introduced for binary losses by Agarwal (2014) to the multiclass case. Our bound suggests that the sample complexity of learning under CCN increases as the noise matrix approaches singularity. We also provide fixes and potential improvements for noise estimation methods that involve computing anchor points. Our experiments confirm our theoretical findings.

Complex Learning Setting: Multiclass Learning from Noisy Labels Using Weighted Losses (Chapter 5)

In many machine learning applications, the labels provided with the training data can be noisy. As a result, there has been growing interest in recent years in developing learning algorithms capable of learning good classifiers from data with noisy labels. While a number of Bayes consistent algorithms have been proposed for multiclass learning with noisy labels, there are still several unresolved questions in this area. For example, most of the noise-corrected algorithms proposed so far require the use of smooth surrogate losses that can estimate class probabilities (such as the multiclass logistic loss), effectively ruling out hinge-type losses used in support vector type algorithms. So, an open question is whether one can design noise-corrected multiclass algorithms that allow for such losses. In this work, we close this question by solving a more general open problem. Specifically, we show how to design a surrogate loss for a general multiclass loss $\mathbf{L}$ by taking a weighted combination of surrogate losses for the standard 0-1 loss. Our method works with both smooth surrogate losses and non-smooth surrogate losses, allowing for margin-based losses such as those used in various formulations of multiclass support vector type algorithms. In addition, the proposed method preserves the convexity of the underlying surrogate loss, a desirable property to allow for efficient optimization. We also provide theoretical results to show the proposed method is Bayes consistent, and provide an estimation error bound. Then, we apply the proposed method to extend the weighted loss method proposed in Natarajan et al. (2013) for binary learning from noisy labels to multiclass learning from noisy labels. Finally, we also apply the proposed method to solve problems in multiclass learning with a reject option; in doing so, we recover several results of Cao et al. (2022).

Complex Label Space and Learning Setting: Foreseeing the Benefits of Incidental Supervision (He et al. (2021); not included in this thesis)

In a collaborative empirical study (He et al., 2021), we explore the possibility of quantifying the benefits of various types of *incidental signals* (such as noisy labels, partial/missing labels, knowledge-based constraints, cross-domain and cross-task annotations) for a specific target task within a single framework, without conducting combinatorial experiments. We propose a unified PAC-Bayesian motivated informativeness measure that characterizes the uncertainty reduction provided by incidental supervision signals.

Complex Label Space and Learning Setting: Consistent Multi-Label Learning from Noisy Labels (Chapter 6)

In many applications of machine learning, the training data comes with noisy labels; this issue is even more pronounced in multi-label problems, where multiple labels/tags can be active in an instance simultaneously. In recent years, many consistent noise-corrected algorithms have been designed for binary and multiclass learning under class-conditional noise (CCN) and other noise models; however, relatively few consistent algorithms exist for multi-label learning, and those that do are under the very simple independent flipping noise (IFN) model. In this work, we develop three consistent noise-corrected multi-label learning algorithms: *Noise-Corrected Plug-in* (NCPLUG) algorithm for Hamming loss under IFN; *Noise-Corrected Exact F-measure Plug-in* (NCEFP) algorithm for multi-label F_1-measure under general CCN; and *Noise-Corrected Output Coding* (NCOC) algorithm for general low-rank multi-label losses under general CCN. We provide quantitative regret transfer bounds for all three algorithms to establish their consistency. We also propose a new family of structured multi-label noise models that we term *Similar-Tag Switching Noise* (STSN) models; STSN models are a special case of CCN that require fewer parameters and enable fast computation, and moreover, unlike IFN, they also capture some correlations among tags. Our experiments confirm the effectiveness of our algorithms in correcting for multi-label noise.

Complex Performance Measure and Learning Setting: Multiclass Learning from Noisy Labels for Non-decomposable Performance Measures (Chapter 7; Zhang and Agarwal (2024))

There has been much interest in recent years in learning good classifiers from data with noisy labels. Most work on learning from noisy labels has focused on standard loss-based performance measures. However, many machine learning problems require using *non-decomposable* performance measures which cannot be expressed as the expectation or sum of a loss on individual examples; these include for example the H-mean, Q-mean and G-mean in class imbalance settings, and the Micro F_1 in information retrieval. In this work, we design algorithms to learn from noisy labels for two broad classes of multiclass non-decomposable performance measures, namely, monotonic convex and ratio-of-linear, which encompass all the above examples. Our work builds on the Frank-Wolfe and Bisection based methods of Narasimhan et al. (2015). In both cases, we develop noise-corrected versions of the algorithms under the widely studied family of class-conditional noise models. We provide regret (excess risk) bounds for our algorithms, establishing that even though they are trained on noisy data, they are Bayes consistent in the sense that their performance converges to the optimal performance w.r.t. the clean (non-noisy) distribution. Our experiments demonstrate the effectiveness of our algorithms in handling label noise.

Additional Work During Ph.D.: Bayes Consistency vs. H-Consistency: The Interplay between Surrogate Loss Functions and the Scoring Function Class (Zhang and Agarwal (2020); not included in this thesis)

A fundamental question in multiclass classification concerns understanding the consistency properties of surrogate risk minimization algorithms, which minimize a (often convex) surrogate to the multiclass 0-1 loss. In particular, the framework of calibrated surrogates has played an important role in analyzing *Bayes consistency* of such algorithms, i.e. in studying convergence to a Bayes optimal classifier (Zhang, 2004a,b; Tewari and Bartlett, 2007). However, follow-up work has suggested this framework can be of limited value when studying *$\mathcal{H}$-consistency*; in particular, concerns have been raised that even when the data comes from an underlying linear model, minimizing certain convex calibrated surrogates over linear scoring functions fails to recover the true model (Long and Servedio, 2013). In this work, we investigate this apparent conundrum. We find that while some calibrated surrogates can indeed fail to provide $\mathcal{H}$-consistency when minimized over a natural-looking but naïvely chosen scoring function class $\mathcal{F}$, the situation can potentially be remedied by minimizing them over a more carefully chosen class of scoring functions $\mathcal{F}$. In particular, for the popular one-vs-all hinge and logistic surrogates, both of which are calibrated (and therefore provide Bayes consistency) under realizable models, but were previously shown to pose problems for realizable $\mathcal{H}$-consistency, we derive a form of scoring function class $\mathcal{F}$ that enables $\mathcal{H}$-consistency. When $\mathcal{H}$ is the class of linear models, the class $\mathcal{F}$ consists of certain piecewise linear scoring functions that are characterized by the same number of parameters as in the linear case, and minimization over which can be performed using an adaptation of the min-pooling idea from neural network training. Our experiments confirm that the one-vs-all surrogates, when trained over this class of *nonlinear* scoring functions $\mathcal{F}$, yield better *linear* multiclass classifiers than when trained over standard linear scoring functions.

1.3. List of Publications Related to This Thesis/Work Done During Ph.D.

- Mingyuan Zhang · Shivani Agarwal

  Multiclass Learning from Noisy Labels for Non-decomposable Performance Measures

  In Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (**AISTATS**), 2024.

- Hangfeng He · Mingyuan Zhang · Qiang Ning · Dan Roth

  Foreseeing the Benefits of Incidental Supervision

  In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (**EMNLP**), 2021.

- Mingyuan Zhang · Jane Lee · Shivani Agarwal

  Learning from Noisy Labels with No Change to the Training Process

  In Proceedings of the 38th International Conference on Machine Learning (**ICML**), 2021.

- Mingyuan Zhang · Shivani Agarwal

  Bayes Consistency vs. H-Consistency: The Interplay between Surrogate Loss Functions and the Scoring Function Class

  In Advances in Neural Information Processing Systems (**NeurIPS**), 2020. **Spotlight paper.**

- Mingyuan Zhang · Harish Guruprasad Ramaswamy · Shivani Agarwal

  Convex Calibrated Surrogates for the Multi-Label F-Measure

  In Proceedings of the 37th International Conference on Machine Learning (**ICML**), 2020.

Beginning with the next chapter, we present our work in detail. Each chapter is designed to be self-contained, allowing it to be read independently of the others.

CHAPTER 2

COMPLEX LABEL SPACE: CONVEX CALIBRATED SURROGATES FOR THE MULTI-LABEL F-MEASURE

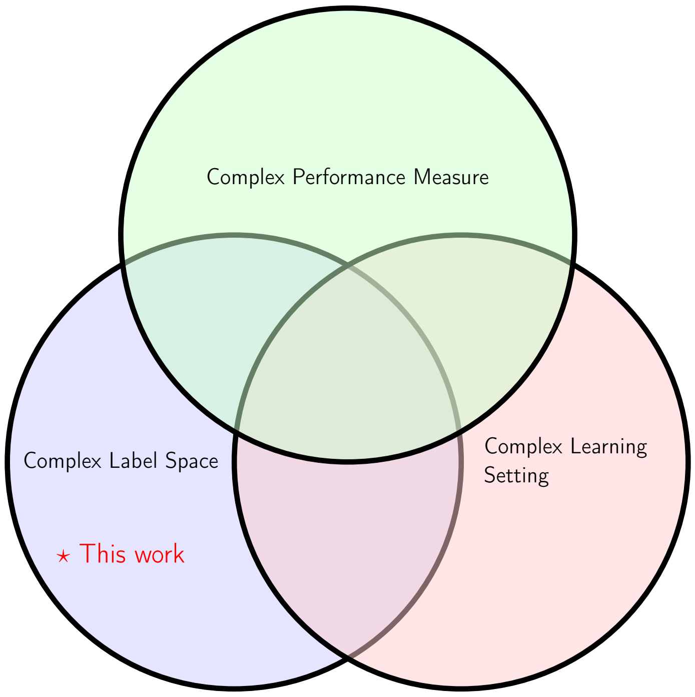

Figure 2.1: Position of *Convex Calibrated Surrogates for the Multi-Label F-Measure* in the thesis.

This chapter was previously published as Mingyuan Zhang, Harish Guruprasad Ramaswamy, and Shivani Agarwal. Convex calibrated surrogates for the multi-label f-measure. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 11246–11255. PMLR, 2020. As the sole first author, I developed most of the results (both theoretical and experimental) in this chapter.

In this chapter, we start our discussion of the first dimension of complexities: complex label spaces. We focus on multi-label classification problems and show how to design consistent algorithms for the multi-label F-measure using convex calibrated surrogates.

2.1. Background of Complex Label Space

In machine learning, the notion of complex label space emerges from the requirement to tackle problems involving structured output objects instead of categorical values in binary or multiclass classification. In such problems, the label space consists of interdependent structures, which makes the prediction task more demanding compared to traditional classification tasks. These types of machine learning problems are known as structured prediction problems.

The background of structured predictions can be traced back to the emergence of various machine learning problems that require a more sophisticated output structure than just a single label. These tasks include:

Multi-label classification: In multi-label classification, each instance can be associated with multiple labels (tags) simultaneously. A good example is that of image tagging, where several tags (such as `sky`, `sand`, `water`) can be active in the same image. This contrasts with the traditional multiclass classification, where each instance is assigned to a single class. Multi-label classification also has applications in text categorization and others.

Sequence labeling: In sequence labeling tasks, the goal is to assign a label to each element in a sequence while considering the dependencies between the elements. Examples of sequence labeling tasks include part-of-speech tagging and named entity recognition in natural language processing.

Parsing: Parsing (also known as syntax analysis, or syntactic analysis) is the process of analyzing and assigning syntactic structures to sentences or sequences of symbols in natural language processing, computer languages, or bioinformatics. For example, syntactic parsing in computational linguistics involves determining the grammatical structure of a sentence and assigning a parse tree to represent the hierarchical relationships among the words.

Image segmentation: In computer vision, image segmentation is the process of partitioning an image into semantically meaningful regions. This task involves assigning labels for individual pixels or regions, taking into account the spatial dependencies between them.

As the need to handle structured prediction problems increased, researchers began developing methods that could effectively model the relationships and dependencies between the elements of the output structure. Early methods, such as conditional random fields (Lafferty et al., 2001), max-margin markov networks (Taskar et al., 2003; Bartlett et al., 2004; Collins et al., 2008), and structured support vector machines (Tsochantaridis et al., 2005; Finley and Joachims, 2008), were proposed to address structured prediction tasks, and these methods paved the way for more advanced techniques in later years.

The study of structured prediction problems continues to be an active area of research as new challenges emerge and the need for more sophisticated models and algorithms grows. Recent advancements in deep learning, such as convolutional neural networks, recurrent neural networks, and transformers, have also contributed significantly to tackling problems with complex label spaces.

2.2. Introduction

2.2.1. Background and Our Contributions

The F_β-measure is a widely used performance measure for multi-label classification (MLC) problems. In particular, in an MLC problem, multiple labels can be active in an instance simultaneously; a good example is that of image tagging, where several tags (such as `sky`, `sand`, `water`) can be active in the same image. In such problems, when evaluating the performance of a classifier on a particular instance, it is important to balance the *recall* of the classifier on the given instance, i.e. the fraction of active labels for that instance that are correctly predicted as such, and the *precision* of the classifier on the instance, i.e. the fraction of labels predicted to be active for that instance that are actually so. The F_β-measure accomplishes this by taking the (possibly weighted) harmonic mean of these two quantities.

Unfortunately, as with most discrete prediction problems, optimizing the F_β-measure directly during training is computationally hard. Consequently, one generally settles for some form of approximation. One approach is to simply treat the labels as independent, and train a separate binary classifier for each label; this is sometimes referred to as the *binary relevance* (BR) approach. Of course, this ignores the fact that labels can have correlations among them (e.g. `sky` and `cloud` may be more likely to co-occur than `sky` and `computer`). Several other approaches have been proposed in recent years (Dembczynski et al., 2013; Koyejo et al., 2015; Wu and Zhou, 2017; Pillai et al., 2017).

In this work, we turn to the theory of convex calibrated surrogate losses – which has yielded convex risk minimization algorithms for several other discrete prediction problems in recent years (Bartlett et al., 2006; Zhang, 2004b; Tewari and Bartlett, 2007; Steinwart, 2007; Duchi et al., 2010; Gao and Zhou, 2013; Ramaswamy et al., 2014, 2015) – to design principled surrogate risk minimization algorithms for the multi-label F_β-measure. In particular, for an MLC problem with s tags, the total number of possible labelings of an instance is 2^s (each tag can be active or inactive). Viewing the F_β-measure as (one minus) a $2^s \times 2^s$ loss matrix, we show that this matrix has rank at most s^2+1, and apply the results of Ramaswamy et al. (2014) to design an output coding scheme that reduces the F_β learning problem to a set of s^2+1 binary class probability estimation (CPE) problems. By using a suitable binary surrogate risk minimization algorithm (such as binary logistic regression) for these binary problems, we effectively construct a (s^2+1)-dimensional convex calibrated surrogate loss for the F_β-measure. We also give a quantitative regret transfer bound for the constructed surrogate, which allows us to transfer any regret guarantees for the binary subproblems to guarantees on F_β-regret for the overall MLC problem. In particular, this means that using a consistent learner for the binary problems yields a consistent learner for the MLC problem (whose F_β-regret goes to zero as the training sample size increases).

Our algorithm is related to the plug-in algorithm of Dembczynski et al. (2013), which also estimates s^2+1 statistics of the underlying distribution. Dembczynski et al. (2013) estimate these statistics by reducing the F_β maximization problem to s multiclass CPE problems, each with at most $s+1$ classes (plus one binary CPE problem); we do so by reducing the problem to s^2+1 binary CPE problems. As we show, both algorithms effectively estimate the same s^2+1 statistics, and indeed, both perform similarly in experiments. Interestingly, the algorithm of Dembczynski et al. (2013), while motivated primarily by the plug-in approach, can also be viewed as minimizing a certain convex calibrated surrogate loss (different from ours); conversely, our algorithm, while motivated primarily by the convex calibrated surrogates approach, can also be viewed as a plug-in algorithm. Our study brings out interesting connections between the two approaches; in addition, to the best our knowledge, our analysis is the first to provide a quantitative regret transfer bound for calibrated surrogates for the F_β-measure.

2.2.2. Notation

For an integer n, we denote by $[n]$ the set of integers $\{1, \ldots, n\}$, and by Δ_n the probability simplex $\{\mathbf{p} \in \mathbb{R}_+^n : \sum_{y=1}^n p_y = 1\}$. For a vector $\mathbf{a}$, we denote by $\|\mathbf{a}\|_p$ the L_p norm of $\mathbf{a}$, and by a_j the j-indexed entry of $\mathbf{a}$. For a matrix $\mathbf{A}$, we denote by $\|\mathbf{A}\|_p$ the induced p-norm of $\mathbf{A}$, and by $\mathbf{a}_y$ the y-indexed column vector of $\mathbf{A}$. a_{yj} (or $a_{y,j}$) is the j-indexed entry of $\mathbf{a}_y$. Indicator function is $\mathbf{1}(\cdot)$.

2.2.3. Related Work

There has been much work on multi-label learning, learning with the F_β-measure, and convex calibrated surrogates. Below we briefly discuss work that is most related to our study. For detailed surveys on multi-label learning, we refer the reader to Zhang and Zhou (2014) and Pillai et al. (2017).

Bayes optimal multi-label classifiers. In an elegant study, Dembczynski et al. (2011) studied in detail the form of a Bayes optimal multi-label classifier for the F_1-measure. In particular, they showed that, for an s-label MLC problem, given a certain set of s^2+1 statistics of the true conditional label distribution (distribution over 2^s labelings), one can compute a Bayes optimal classifier for the F_1-measure in $O(s^3)$ time. Their result extends to general F_β-measures. Bayes optimal classifiers have also been studied for other MLC performance measures, such as Hamming loss and subset 0-1 loss (Dembczynski et al., 2010a).

Consistent algorithms for multi-label learning. Dembczynski et al. (2013) extended and operationalized the results of Dembczynski et al. (2011) by providing a consistent plug-in MLC algorithm for the F_β-measure. Specifically, they showed that the s^2+1 statistics of the conditional label distribution needed to compute a Bayes optimal classifier can be estimated via s multiclass CPE problems, each with at most $s+1$ classes, plus one binary CPE problem; the statistics estimated by solving these CPE problems can then be plugged into the $O(s^3)$-time procedure of Dembczynski et al. (2011) to produce a consistent plug-in algorithm termed the *exact F-measure plug-in* (EFP) algorithm. Consistent learning algorithms have also been studied for other multi-label performance measures (Gao and Zhou, 2013; Koyejo et al., 2015).[1] The simple approach of learning an independent binary classifier for each of the s labels, known as *binary relevance* (BR), is known to yield a consistent algorithm for the Hamming loss; it also yields a consistent algorithm for the F_β-measure under the assumption of conditionally independent labels, but can be arbitrarily bad otherwise (Dembczynski et al., 2011).

Large-margin algorithms for multi-label learning. Several studies have considered large-margin algorithms for multi-label learning with the F_β-measure. These include the *reverse multi-label* (RML) and *sub-modular multi-label* (SML) algorithms of Petterson and Caetano (2010, 2011), which make use of the StructSVM framework (Tsochantaridis et al., 2005), and more recently, the *label-wise and instance-wise margin optimization* (LIMO) algorithm due to Wu and Zhou (2017), which aims to simultaneously optimize several different multi-label performance measures. The RML and SML algorithms were proven to be inconsistent for the F_β-measure and shown to be outperformed by the EFP algorithm by Dembczynski et al. (2013). We include a comparison with LIMO in our experiments.

Multivariate F_β-measure for binary classification. The F_β-measure is also used as a multivariate performance measure in binary classification tasks with significant class imbalance. This use of the F_β-measure is related to, but distinct from, the use of the F_β-measure in MLC problems. Several approaches have been proposed that aim to optimize the multivariate F_β-measure in binary classification (Joachims, 2005; Ye et al., 2012; Parambath et al., 2014).

Convex calibrated surrogates. Convex surrogate losses are frequently used in machine learn-

[1]Note that while the study of Koyejo et al. (2015) also includes the F_β-measure (among other performance measures), their study is in the context of what has been referred to as the 'expected utility maximization' (EUM) framework; in contrast, our study is in the context of what has been referred to as the 'decision-theoretic analysis' (DTA) framework. Their results are generally incomparable to ours. (In particular, under the EUM framework, Koyejo et al. (2015) showed that a thresholding approach leads to Bayes optimal performance; on the contrary, under the DTA framework, it was shown by Dembczynski et al. (2011) that a thresholding approach cannot be optimal for general distributions.)

ing to design computationally efficient learning algorithms. The notion of calibrated surrogate losses, which ensures that minimizing the surrogate loss can (in the limit of sufficient data) recover a Bayes optimal model for the target discrete loss, was initially studied in the context of binary classification (Bartlett et al., 2006; Zhang, 2004a) and multiclass 0-1 classification (Zhang, 2004b; Tewari and Bartlett, 2007). In recent years, calibrated surrogates have been designed for several more complex learning problems, including general multiclass problems and certain types of subset ranking and multi-label problems (Steinwart, 2007; Duchi et al., 2010; Gao and Zhou, 2013; Ramaswamy et al., 2013, 2014, 2015). In our work, we will make use of a result of Ramaswamy et al. (2014), who designed convex calibrated surrogates based on output coding for multiclass problems with low-rank loss matrices.

2.2.4. Organization

Section 2.3 gives preliminaries and background. Section 2.4 gives our convex calibrated surrogates for the F_β-measure; Section 2.5 provides a corresponding regret transfer bound. Section 2.6 discusses the relationship with the plug-in algorithm of Dembczynski et al. (2013). Section 2.7 provides experimental evaluations of our algorithm. Section 2.8 concludes this work.

2.3. Preliminaries and Background

2.3.1. Problem Setup

Multi-label classification (MLC). In an MLC problem, there is an instance space $\mathcal{X}$, and a set of s labels or 'tags' $\mathcal{L} = [s] := \{1, \ldots, s\}$ that can be associated with each instance in $\mathcal{X}$. For example, in image tagging, $\mathcal{X}$ is the set of possible images, and $\mathcal{L}$ is a set of s pre-defined tags (such as `sky`, `cloud`, `water` etc) that can be associated with each image. The learner is given a training sample $S = \{(x_1, \mathbf{y}_1), \ldots, (x_m, \mathbf{y}_m)\} \in (\mathcal{X} \times \{0,1\}^s)^m$, where the labeling $\mathbf{y}_i \in \{0,1\}^s$ indicates which of the s tags are active in instance x_i (specifically, $y_{ij} = 1$ denotes that tag j is active in instance x_i, and $y_{ij} = 0$ denotes it is inactive). The goal is to learn from these examples a multi-label classifier $\mathbf{h} : \mathcal{X} \to \{0,1\}^s$ which, given a new instance $x \in \mathcal{X}$, predicts which tags are active or inactive via $\mathbf{h}(x) \in \{0,1\}^s$.

F_β-measure. For any $\beta > 0$, the F_β-measure evaluates the quality of an MLC prediction as follows.

Given a true labeling $\mathbf{y} \in \{0,1\}^s$ and a predicted labeling $\widehat{\mathbf{y}} \in \{0,1\}^s$, the recall and precision are given by

$$\text{rec}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{\sum_{j=1}^{s} y_j \widehat{y}_j}{\|\mathbf{y}\|_1} \,; \quad \text{prec}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{\sum_{j=1}^{s} y_j \widehat{y}_j}{\|\widehat{\mathbf{y}}\|_1} \,.$$

In words, the recall measures the fraction of active tags that are predicted correctly, and the precision measures the fraction of tags predicted as active that are actually so. The F_β-measure balances these two quantities by taking their (weighted) harmonic mean:

$$\begin{aligned} F_\beta(\mathbf{y}, \widehat{\mathbf{y}}) &= \left(\left(\tfrac{\beta^2}{1+\beta^2}\right) \frac{1}{\text{rec}(\mathbf{y}, \widehat{\mathbf{y}})} + \left(\tfrac{1}{1+\beta^2}\right) \frac{1}{\text{prec}(\mathbf{y}, \widehat{\mathbf{y}})} \right)^{-1} \\ &= \frac{(1+\beta^2) \sum_{j=1}^{s} y_j \widehat{y}_j}{\beta^2 \|\mathbf{y}\|_1 + \|\widehat{\mathbf{y}}\|_1} \,. \end{aligned} \tag{2.1}$$

Clearly, $0 \leq F_\beta(\mathbf{y}, \widehat{\mathbf{y}}) \leq 1$. Higher values of the F_β-measure correspond to better quality predictions. We will take $\frac{0}{0} = 1$, so that when $\mathbf{y} = \widehat{\mathbf{y}} = \mathbf{0}$, we have $F_\beta(\mathbf{0}, \mathbf{0}) = 1$. The most commonly used instantiation is the F_1-measure, which weighs recall and precision equally; other commonly used variants include the F_2-measure, which weighs recall more heavily than precision, and the $F_{0.5}$-measure, which weighs precision more heavily than recall.

Learning goal. Assuming that training examples are drawn IID from some underlying probability distribution D on $\mathcal{X} \times \{0,1\}^s$, it is natural then to measure the quality of a multi-label classifier $\mathbf{h} : \mathcal{X} \to \{0,1\}^s$ by its F_β-*generalization accuracy*:[2]

$$\text{acc}_D^{F_\beta}[\,\mathbf{h}\,] \;=\; \mathbf{E}_{(x,\mathbf{y}) \sim D}[\,F_\beta(\mathbf{y}, \mathbf{h}(x))\,] \,.$$

The *Bayes* F_β-*accuracy* is then the highest possible value of the F_β-generalization accuracy for D:

$$\text{acc}_D^{F_\beta,*} \;=\; \sup_{\mathbf{h}:\mathcal{X} \to \{0,1\}^s} \text{acc}_D^{F_\beta}[\,\mathbf{h}\,] \,.$$

The F_β-*regret* of a multi-label classifier $\mathbf{h}$ is then the difference between the Bayes F_β-accuracy and

[2]Note that our focus is on *instance-averaged* F_β performance (Zhang and Zhou, 2014).

the F_β-accuracy of $\mathbf{h}$:

$$\mathrm{regret}_D^{F_\beta}[\,\mathbf{h}\,] \;=\; \mathrm{acc}_D^{F_\beta,*} - \mathrm{acc}_D^{F_\beta}[\,\mathbf{h}\,]\,.$$

Our goal will be to design *consistent* algorithms for the F_β-measure, i.e. algorithms whose F_β-regret converges (in probability) to zero as the number of training examples increases. In particular, since we cannot maximize the (discrete) F_β-measure directly, we would like to design consistent algorithms that maximize a concave surrogate performance measure – or equivalently, minimize a convex surrogate loss – instead. For this, we will turn to the theory of convex calibrated surrogates.

2.3.2. Convex Calibrated Surrogates for Multiclass Problems

Here we review the theory of convex calibrated surrogates for multiclass classification problems, and in particular, the result of Ramaswamy et al. (2014) for low-rank multiclass loss matrices that we will use in our work. We will apply the theory to the multi-label F_β-measure in Section 2.4.

Multiclass classification. Consider a standard multiclass (not multi-label) learning problem with instance space $\mathcal{X}$ and label space $\mathcal{Y} = [n]$ (i.e., n classes). Let $\mathbf{L} \in \mathbb{R}_+^{n\times n}$ be a loss matrix whose $(y, \widehat{y})$-th entry $\ell_{y,\widehat{y}} = \ell(y, \widehat{y})$ (for each $y, \widehat{y} \in [n]$) specifies the loss incurred on predicting $\widehat{y}$ when the true label is y (the 0-1 loss $\mathbf{L}^{\text{0-1}}$ is a special case with $\ell^{\text{0-1}}_{y,\widehat{y}} = \mathbf{1}(\widehat{y} \neq y)$). Then, given a training sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ with examples drawn IID from some underlying probability distribution D on $\mathcal{X} \times \mathcal{Y}$, the performance of a classifier $h : \mathcal{X} \to \mathcal{Y}$ is measured by its $\mathbf{L}$-generalization error $\mathrm{er}_D^{\mathbf{L}}[h] = \mathbf{E}_{(x,y)\sim D}[\,\ell_{y,h(x)}\,]$, or its $\mathbf{L}$-regret $\mathrm{regret}_D^{\mathbf{L}}[h] = \mathrm{er}_D^{\mathbf{L}}[h] - \mathrm{er}_D^{\mathbf{L},*}$, where $\mathrm{er}_D^{\mathbf{L},*} = \inf_{h:\mathcal{X}\to\mathcal{Y}} \mathrm{er}_D^{\mathbf{L}}[h]$ is the Bayes $\mathbf{L}$-error for D. A learning algorithm that maps training samples S to classifiers h_S is said to be (universally) $\mathbf{L}$-*consistent* if for all D and for $S \sim D^m$, $\mathrm{regret}_D^{\mathbf{L}}[h_S] \xrightarrow{P} 0$ as $m \to \infty$.

Surrogate risk minimization and calibrated surrogates. Since minimizing the discrete loss $\mathbf{L}$ directly is computationally hard, a common algorithmic framework is to minimize a surrogate loss $\psi : [n] \times \mathbb{R}^d \to \mathbb{R}_+$ for some suitable $d \in \mathbb{Z}_+$. In particular, given a multiclass training sample S

as above, one learns a d-dimensional 'scoring' function $\mathbf{f}_S : \mathcal{X} \to \mathbb{R}^d$ by solving

$$\min_{\mathbf{f}} \textstyle\sum_{i=1}^m \psi(y_i, \mathbf{f}(x_i))$$

over a suitably rich class of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^d$; and then returns $h_S = \text{decode} \circ \mathbf{f}_S$ for some suitable mapping decode : $\mathbb{R}^d \to [n]$. In practice, the surrogate ψ is often chosen to be convex in its second argument to enable efficient minimization. It is known that if the minimization is performed over a universal function class (with suitable regularization), then the resulting algorithm is universally *ψ-consistent*, i.e. that the ψ-regret converges to zero: $\text{regret}_D^{\psi}[\mathbf{f}_S] = \text{er}_D^{\psi}[\mathbf{f}_S] - \text{er}_D^{\psi,*} \xrightarrow{P} 0$ as $m \to \infty$ (where $\text{er}_D^{\psi}[\mathbf{f}] = \mathbf{E}_{(x,y)\sim D}[\,\psi(y, \mathbf{f}(x))\,]$ is the ψ-generalization error of $\mathbf{f}$ and $\text{er}_D^{\psi,*} = \inf_{\mathbf{f}:\mathcal{X}\to\mathbb{R}^d} \text{er}_D^{\psi}[\mathbf{f}]$ is the Bayes ψ-error). The surrogate ψ, together with the mapping decode, is said to be **L**-*calibrated* if this also implies **L**-consistency, i.e. if

$$\text{regret}_D^{\psi}[\mathbf{f}_S] \xrightarrow{P} 0 \implies \text{regret}_D^{\mathbf{L}}[\text{decode} \circ \mathbf{f}_S] \xrightarrow{P} 0\,.$$

Thus, given a target loss **L**, the task of designing an **L**-consistent algorithm reduces to designing a convex **L**-calibrated surrogate-mapping pair (ψ, decode); the resulting surrogate risk minimization algorithm (implemented in a universal function class with suitable regularization) is then universally **L**-consistent.

Result of Ramaswamy et al. (2014) for low-rank loss matrices. The result of Ramaswamy et al. (2014) effectively decomposes multiclass problems into a set of binary CPE problems; to describe the result, we will need the following definition for *binary* losses:

Definition 2.1 (Strictly proper composite binary losses (Reid and Williamson, 2010))**.** *A binary loss $\phi : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}_+$ is* strictly proper composite with underlying (invertible) link function $\gamma : [0,1] \to \mathbb{R}$ *if for all $q \in [0,1]$ and $u \neq \gamma(q) \in \mathbb{R}$:*

$$\mathbf{E}_{y\sim Bin^{\pm 1}(q)}\Big[\phi(y,u) - \phi(y,\gamma(q))\Big] \;>\; 0\,,$$

where $y \sim Bin^{\pm 1}(q)$ denotes a $\{\pm 1\}$-valued random variable that takes value $+1$ with probability q and value -1 with probability $1-q$.

Intuitively, minimizing a strictly proper composite binary loss allows one to recover accurate class probability estimates for binary CPE problems: the learned real-valued score is simply inverted via γ^{-1} (Reid and Williamson, 2010).

We can now state the result of Ramaswamy et al. (2014), which for multiclass loss matrices $\mathbf{L}$ of rank r, gives a family of r-dimensional convex $\mathbf{L}$-calibrated surrogates defined in terms of strictly proper composite binary losses as follows (result specialized here to the case of square loss matrices, and stated with a small change in normalization):

Theorem 2.1 (Ramaswamy et al. (2014))**.** *Let $\mathbf{L} \in \mathbb{R}_+^{n\times n}$ be a rank-r multiclass loss matrix, with $\ell_{y,\widehat{y}} = \mathbf{a}_y^\top \mathbf{b}_{\widehat{y}}$ for some $\mathbf{a}_1, \ldots, \mathbf{a}_n, \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^r$. Let $\phi : \{\pm 1\} \times \mathbb{R} \rightarrow \mathbb{R}_+$ be any strictly proper composite binary loss, with underlying link function $\gamma : [0,1] \rightarrow \mathbb{R}$. Define a multiclass surrogate $\psi : [n] \times \mathbb{R}^r \rightarrow \mathbb{R}_+$ and mapping* decode $: \mathbb{R}^r \rightarrow [n]$ *as follows:*

$$\psi(y, \mathbf{u}) \;=\; \sum_{j=1}^{r} \Big(\widetilde{a}_{yj}\phi(+1, u_j) + (1-\widetilde{a}_{yj})\phi(-1, u_j) \Big)$$

$$\text{decode}(\mathbf{u}) \;\in\; \operatorname*{argmin}_{\widehat{y}\in[n]} \sum_{j=1}^{r} \widetilde{b}_{\widehat{y}j}\gamma^{-1}(u_j) + c_{\widehat{y}}\,,$$

where

$$\begin{aligned}
\widetilde{a}_{yj} &= \frac{a_{yj} - a_{\min}}{a_{\max} - a_{\min}} \quad (\in [0,1]) \\
\widetilde{b}_{\widehat{y}j} &= (a_{\max} - a_{\min}) \cdot b_{\widehat{y}j} \\
c_{\widehat{y}} &= a_{\min} \textstyle\sum_{j=1}^{r} b_{\widehat{y}j} \\
a_{\min} &= \min_{y,j} a_{yj} \\
a_{\max} &= \max_{y,j} a_{yj}\,.
\end{aligned}$$

Then (ψ, decode) *is* $\mathbf{L}$*-calibrated.*

The above result effectively decomposes the multiclass problem into r binary CPE problems, where the labels for these CPE problems can themselves be given as probabilities in $[0,1]$ rather than binary values (see Ramaswamy et al. (2014) for details). For our purposes, we will use the standard binary logistic loss for the binary CPE problems, which is known to be strictly proper composite (see Section 2.4 below for more details).

2.4. Convex Calibrated Surrogates for F_β

In order to construct convex calibrated surrogates – and corresponding surrogate risk minimization algorithms – for the multi-label F_β-measure, we will start by viewing the multi-label learning problem as a giant multiclass classification problem with $n = 2^s$ classes (this is only for the purpose of analysis and derivation of the surrogates; as we will see, the actual algorithms we will obtain will require learning only $O(s^2)$ real-valued score functions). To this end, let us define the F_β-loss matrix $\mathbf{L}^{F_\beta} \in \mathbb{R}_+^{\{0,1\}^s \times \{0,1\}^s}$ as follows:

$$\ell^{F_\beta}_{\mathbf{y},\widehat{\mathbf{y}}} \;=\; 1 - F_\beta(\mathbf{y},\widehat{\mathbf{y}})\,.$$

$\mathbf{L}^{F_\beta}$ has low rank. We show here that (a slightly shifted version of) the above loss matrix has rank at most s^2+1.

Proposition 2.2. $\mathrm{rank}(\mathbf{L}^{F_\beta} - 1) \le s^2 + 1$.

Proof. We have,

$$\ell^{F_\beta}_{\mathbf{y},\widehat{\mathbf{y}}} - 1 \;=\; -F_\beta(\mathbf{y},\widehat{\mathbf{y}}) \;=\; -\frac{(1+\beta^2)\sum_{j=1}^s y_j\widehat{y}_j}{\beta^2\|\mathbf{y}\|_1 + \|\widehat{\mathbf{y}}\|_1}\,.$$

Stratifying over the $s+1$ different values of $\|\mathbf{y}\|_1 \in \{0,1\ldots,s\}$, we can write this as

$$\begin{aligned}
\ell^{F_\beta}_{\mathbf{y},\widehat{\mathbf{y}}} - 1 \;&=\; -\mathbf{1}(\|\mathbf{y}\|_1 = 0)\cdot\mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) \\
&\qquad - \sum_{k=1}^s \mathbf{1}(\|\mathbf{y}\|_1 = k)\cdot\frac{(1+\beta^2)\sum_{j=1}^s y_j\widehat{y}_j}{\beta^2 k + \|\widehat{\mathbf{y}}\|_1} \\
&=\; a_{\mathbf{y},0}\cdot b_{\widehat{\mathbf{y}},0} + \sum_{j=1}^s\sum_{k=1}^s a_{\mathbf{y},jk}\cdot b_{\widehat{\mathbf{y}},jk}\,,
\end{aligned}$$

where

$$\begin{aligned} a_{\mathbf{y},0} &= \mathbf{1}(\|\mathbf{y}\|_1 = 0) &\quad (2.2)\\ b_{\widehat{\mathbf{y}},0} &= -\mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) &\quad (2.3)\\ a_{\mathbf{y},jk} &= \mathbf{1}(\|\mathbf{y}\|_1 = k)\cdot y_j &\quad (2.4)\\ b_{\widehat{\mathbf{y}},jk} &= -\frac{(1+\beta^2)\cdot \widehat{y}_j}{\beta^2 k + \|\widehat{\mathbf{y}}\|_1}\,. &\quad (2.5)\end{aligned}$$

This proves the claim. $\square$

$\mathbf{L}^{F_\beta}$-calibrated surrogates. Given the above result, we can now apply Theorem 2.1 to construct a family of (s^2+1)-dimensional convex calibrated surrogate losses for $\mathbf{L}^{F_\beta}$.[3] Specifically, starting with any strictly proper composite binary loss $\phi : \{\pm1\}\times\mathbb{R}\to\mathbb{R}_+$ with underlying link function $\gamma : [0,1]\to\mathbb{R}$, we define a multiclass surrogate $\psi : \{0,1\}^s \times \mathbb{R}^{s^2+1}\to\mathbb{R}_+$ and mapping decode : $\mathbb{R}^{s^2+1}\to\{0,1\}^s$ as follows (where we denote $\mathbf{u} = \big(u_0, (u_{jk})_{j,k=1}^s\big)^\top \in \mathbb{R}^{s^2+1}$):

$$\begin{aligned} &\psi(\mathbf{y},\mathbf{u})\\ &= a_{\mathbf{y},0}\cdot\phi(+1,u_0) + (1-a_{\mathbf{y},0})\cdot\phi(-1,u_0)\\ &\quad + \sum_{j=1}^s\sum_{k=1}^s a_{\mathbf{y},jk}\cdot\phi(+1,u_{jk}) + (1-a_{\mathbf{y},jk})\cdot\phi(-1,u_{jk}) &\quad (2.6)\\ &\text{decode}(\mathbf{u})\\ &\in \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\{0,1\}^s}\ b_{\widehat{\mathbf{y}},0}\cdot\gamma^{-1}(u_0) + \sum_{j=1}^s\sum_{k=1}^s b_{\widehat{\mathbf{y}},jk}\cdot\gamma^{-1}(u_{jk})\,, &\quad (2.7)\end{aligned}$$

where $a_{\mathbf{y},0}, a_{\mathbf{y},jk}, b_{\widehat{\mathbf{y}},0}, b_{\widehat{\mathbf{y}},jk}$ are as defined in Eqs. (2.2-2.5). Then, by Theorem 2.1 and the proof of Proposition 2.2, it follows that (ψ,decode) is $\mathbf{L}^{F_\beta}$-calibrated.[4] Therefore, the resulting (ψ,decode)-based surrogate risk minimization algorithm, when implemented in a universal function class (with suitable regularization), is consistent for the F_β-measure. The algorithm is summarized in Algorithm 2.1. Note that since $a_{\mathbf{y},0}, a_{\mathbf{y},jk} \in \{0,1\}$, in this case minimizing the surrogate risk above

[3] Note that minimizing the $\mathbf{L}^{F_\beta}$-generalization error is equivalent to minimizing the $(\mathbf{L}^{F_\beta}-1)$-generalization error, and therefore a calibrated surrogate for $\mathbf{L}^{F_\beta}-1$ is also calibrated for $\mathbf{L}^{F_\beta}$.

[4] Note that when applying Theorem 2.1 here, we have $a_{\min}=0$ and $a_{\max}=1$, and therefore $\widetilde{\mathbf{a}}_{\mathbf{y}} = \mathbf{a}_{\mathbf{y}}$, $\widetilde{\mathbf{b}}_{\widehat{\mathbf{y}}} = \mathbf{b}_{\widehat{\mathbf{y}}}$, and $c_{\widehat{\mathbf{y}}} = 0$.

Algorithm 2.1 Surrogate risk minimization algorithm for multi-label F_β-measure

```
1: Input: Training sample $S = ((x_1, \mathbf{y}_1), \ldots, (x_m, \mathbf{y}_m)) \in (\mathcal{X} \times \{0,1\}^s)^m$
2: Parameters: (1) Strictly proper composite binary CPE loss $\phi : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}_+$; (2) Class $\mathcal{F}$
   of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^{s^2+1}$
3: Find $\mathbf{f}_S \in \text{argmin}_{\mathbf{f} \in \mathcal{F}} \sum_{i=1}^m \psi(\mathbf{y}_i, \mathbf{f}(x_i))$, where $\psi$ is as defined in Eq. (2.6)
4: Output: Multi-label classifier $\mathbf{h}_S = \text{decode} \circ \mathbf{f}_S$, where decode is as defined in Eq. (2.7) (see
   Appendix for efficient implementation of decode)
```

amounts to solving $s^2 + 1$ binary CPE problems with standard binary (non-probabilistic) labels.

Choice of strictly proper composite binary loss ϕ. As a specific instantiation, in our experiments, we will make use of the binary logistic loss $\phi_{\log} : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}_+$ given by

$$\phi_{\log}(y, u) \quad = \quad \ln(1 + e^{-yu}) \tag{2.8}$$

as the binary loss above; this is known to be strictly proper composite (Reid and Williamson, 2010), with underlying logit link function $\gamma_{\log} : [0,1] \to \mathbb{R}$ given by

$$\gamma_{\log}(p) \quad = \quad \ln\left(\frac{p}{1-p}\right). \tag{2.9}$$

Implementation of 'decode' mapping. The mapping decode : $\mathbb{R}^{s^2+1} \to \{0,1\}^s$ above can be implemented in $O(s^3)$ time using a procedure due to Dembczynski et al. (2011); details are provided in the Appendix for completeness. In particular, Dembczynski et al. (2011) show that if one knows the true conditional MLC distribution $p(\mathbf{y}|x)$, then one can use $s^2 + 1$ statistics of this distribution to construct a Bayes optimal classifier for the F_β-measure; they then provide a procedure to perform this computation in $O(s^3)$ time. As we discuss in greater detail in Section 2.6, our surrogate loss ψ can be viewed as computing estimates of the same $s^2 + 1$ statistics from the training sample S, and therefore our algorithm, which applies the 'decoding' procedure of Dembczynski et al. (2011) to these estimated quantities, can be viewed as effectively learning a form of 'plug-in' multi-label classifier for the F_β-measure.

2.5. Regret Transfer Bound

Above, we constructed a family of $\mathbf{L}^{F_\beta}$-calibrated surrogate-mapping pairs (ψ, decode) (Eqs. (2.6-2.7)), yielding a family of surrogate risk minimization algorithms for the F_β-measure (Algorithm 2.1). We now give a quantitative regret transfer bound showing that any guarantees on the surrogate ψ-regret also translate to guarantees on the target F_β-regret. Specifically, the surrogate loss ψ was defined in terms of a constituent strictly proper composite binary loss $\phi : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}_+$. We show that if the binary loss ϕ is *strongly* proper composite (a relatively mild condition satisfied by several common strictly proper composite binary losses, including the logistic loss), then for all models $\mathbf{f} : \mathcal{X} \to \mathbb{R}^{s^2+1}$, we can upper bound $\text{regret}_D^{F_\beta}[\text{decode} \circ \mathbf{f}]$, the target F_β-regret of the multi-label classifier given by $\mathbf{h}(x) = \text{decode}(\mathbf{f}(x))$, in terms of $\text{regret}_D^{\psi}[\mathbf{f}]$, the surrogate regret of $\mathbf{f}$. In order to prove the regret transfer bound, we will need the following definition:

Definition 2.2 (Strongly proper composite binary losses (Agarwal, 2014))**.** *Let* $\lambda > 0$*. A binary loss* $\phi : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}_+$ *is said to be* λ-strongly proper composite with underlying (invertible) link function $\gamma : [0,1] \to \mathbb{R}$ *if for all* $q \in [0,1]$*,* $u \in \mathbb{R}$*:*

$$\mathbf{E}_{y \sim Bin^{\pm 1}(q)}\Big[\phi(y,u) - \phi(y,\gamma(q))\Big] \;\geq\; \frac{\lambda}{2}\Big(\gamma^{-1}(u) - q\Big)^2 .$$

We note that the logistic loss (Eq. (2.8)) is known to be 4-strongly proper composite with underlying link given by the logit link (Eq. (2.9)) (Agarwal, 2014).

Additional notation. To prove our regret transfer bound, we will also need some additional notation. In particular, for each $\mathbf{y}, \widehat{\mathbf{y}} \in \{0,1\}^s$, we will define the vectors

$$\mathbf{a}_{\mathbf{y}} = \begin{pmatrix} a_{\mathbf{y},0} \\ a_{\mathbf{y},11} \\ \vdots \\ a_{\mathbf{y},ss} \end{pmatrix} \in \{0,1\}^{s^2+1} \tag{2.10}$$

$$\mathbf{b}_{\widehat{\mathbf{y}}} = \begin{pmatrix} b_{\widehat{\mathbf{y}},0} \\ b_{\widehat{\mathbf{y}},11} \\ \vdots \\ b_{\widehat{\mathbf{y}},ss} \end{pmatrix} \in \mathbb{R}^{s^2+1}, \tag{2.11}$$

where $a_{\mathbf{y},0}, a_{\mathbf{y},jk}, b_{\widehat{\mathbf{y}},0}, b_{\widehat{\mathbf{y}},jk}$ are as defined in Eqs. (2.2-2.5). Moreover, for each $x \in \mathcal{X}$, we will define

$$\mathbf{q}(x) = \mathbf{E}_{\mathbf{y}|x}[\mathbf{a}_{\mathbf{y}}] = \sum_{\mathbf{y}\in\{0,1\}^s} p(\mathbf{y}|x) \cdot \mathbf{a}_{\mathbf{y}} \ \in [0,1]^{s^2+1}. \tag{2.12}$$

Intuitively, the elements $q_0(x), (q_{jk}(x))_{j,k=1}^s$ of $\mathbf{q}(x)$ are the 'class probability functions' corresponding to the s^2+1 binary CPE problems effectively created by the surrogate loss ψ defined in Eq. (2.6). The function $\mathbf{f}_S : \mathcal{X} \to \mathbb{R}^{s^2+1}$ learned by minimizing ψ will be such that $\gamma^{-1}(\mathbf{f}_S(x))$ will serve as an estimate of $\mathbf{q}(x)$.

Regret transfer bound. We are now ready to state and prove the following regret transfer bound for the family of surrogate losses defined in the previous section:

Theorem 2.3. *Let $\phi : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}_+$ be a λ-strongly proper composite binary loss with underlying link function $\gamma : [0,1] \to \mathbb{R}$. Let (ψ, decode) be defined as in Eqs. (2.6-2.7). Then for all probability distributions D on $\mathcal{X} \times \{0,1\}^s$ and all $\mathbf{f} : \mathcal{X} \to \mathbb{R}^{s^2+1}$, we have*

$$\mathrm{regret}_D^{F_\beta}[\mathrm{decode} \circ \mathbf{f}] \ \le \ \frac{1+\beta^2}{\beta} \sqrt{\frac{2(\ln s + 1)}{\lambda} \cdot \mathrm{regret}_D^{\psi}[\mathbf{f}]}\,.$$

Proof. We have,

$$
\begin{aligned}
&\mathrm{regret}_D^{F_\beta}[\mathrm{decode}\circ\mathbf{f}] \\
&= \mathbf{E}_x\Big[\sum_{\mathbf{y}} p(\mathbf{y}|x)\cdot\Big(\ell^{F_\beta}_{\mathbf{y},\mathrm{decode}(\mathbf{f}(x))} - \min_{\widehat{\mathbf{y}}} \ell^{F_\beta}_{\mathbf{y},\widehat{\mathbf{y}}}\Big)\Big] \\
&= \mathbf{E}_x\Big[\sum_{\mathbf{y}} p(\mathbf{y}|x)\cdot\Big(\mathbf{a}_{\mathbf{y}}^\top\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \min_{\widehat{\mathbf{y}}} \mathbf{a}_{\mathbf{y}}^\top\mathbf{b}_{\widehat{\mathbf{y}}}\Big)\Big] \\
&= \mathbf{E}_x\Big[\mathbf{q}(x)^\top\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \min_{\widehat{\mathbf{y}}} \mathbf{q}(x)^\top\mathbf{b}_{\widehat{\mathbf{y}}}\Big] \\
&= \mathbf{E}_x\Big[\max_{\widehat{\mathbf{y}}}\ \mathbf{q}(x)^\top\Big(\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{\mathbf{y}}}\Big)\Big] \\
&\le \mathbf{E}_x\Big[\max_{\widehat{\mathbf{y}}}\ \Big(\mathbf{q}(x) - \gamma^{-1}(\mathbf{f}(x))\Big)^\top\Big(\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{\mathbf{y}}}\Big)\Big] \\
&\qquad\qquad \text{(since by the definition of decode,} \\
&\qquad\qquad -\gamma^{-1}(\mathbf{f}(x))^\top\Big(\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{\mathbf{y}}}\Big) \ge 0 \ \ \forall\, \widehat{\mathbf{y}}) \\
&\le \mathbf{E}_x\Big[\big\|\mathbf{q}(x) - \gamma^{-1}(\mathbf{f}(x))\big\|_2\cdot\max_{\widehat{\mathbf{y}}}\ \big\|\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{\mathbf{y}}}\big\|_2\Big] \\
&\qquad\qquad \text{(by the Cauchy-Schwarz inequality)} \\
&\le 2\max_{\widehat{\mathbf{y}}}\ \big\|\mathbf{b}_{\widehat{\mathbf{y}}}\big\|_2\cdot\mathbf{E}_x\Big[\big\|\mathbf{q}(x) - \gamma^{-1}(\mathbf{f}(x))\big\|_2\Big]\,. \qquad (2.13)
\end{aligned}
$$

Now, since ϕ is λ-strongly proper composite with link function γ, we have

$$
\begin{aligned}
&\mathbf{E}_x\Big[\big\|\mathbf{q}(x) - \gamma^{-1}(\mathbf{f}(x))\big\|_2^2\Big] \\
&= \mathbf{E}_x\Big[\Big(q_0(x) - \gamma^{-1}(f_0(x))\Big)^2 + \\
&\qquad\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}\Big(q_{jk}(x) - \gamma^{-1}(f_{jk}(x))\Big)^2\Big] \\
&\le \frac{2}{\lambda}\mathbf{E}_x\Big[\mathbf{E}_{y\sim\mathrm{Bin}^{\pm1}(q_0(x))}\Big[\phi(y, f_0(x)) - \phi(y,\gamma(q_0(x))\Big] + \\
&\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}\mathbf{E}_{y\sim\mathrm{Bin}^{\pm1}(q_{jk}(x))}\Big[\phi(y, f_{jk}(x)) - \phi(y,\gamma(q_{jk}(x))\Big]\Big] \\
&\qquad\qquad \text{(by } \lambda\text{-strong proper compositeness of } \phi) \\
&= \frac{2}{\lambda}\mathbf{E}_x\Big[\mathbf{E}_{\mathbf{y}|x}\Big[\psi(\mathbf{y},\mathbf{f}(x)) - \inf_{\mathbf{u}\in\mathbb{R}^{s^2+1}}\psi(\mathbf{y},\mathbf{u})\Big]\Big] \\
&= \frac{2}{\lambda}\mathrm{regret}_D^{\psi}[\mathbf{f}]\,. \qquad (2.14)
\end{aligned}
$$

Moreover, we have

$$\|\mathbf{b}_{\mathbf{0}}\| = 1\,,$$

and for $\widehat{\mathbf{y}} \neq \mathbf{0}$, we have

$$\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2^2 \;=\; (1+\beta^2)^2 \sum_{k=1}^{s} g_k(\|\widehat{\mathbf{y}}\|_1)\,,$$

where

$$g_k(t) = \frac{t}{(\beta^2 k + t)^2}\,.$$

It can be verified that $g_k(t)$ is maximized at $t^* = \beta^2 k$, yielding for each $\widehat{\mathbf{y}} \neq \mathbf{0}$,

$$\begin{aligned}
\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2^2 &\leq (1+\beta^2)^2 \sum_{k=1}^{s} g_k(\beta^2 k) \\
&= (1+\beta^2)^2 \sum_{k=1}^{s} \frac{1}{4\beta^2 k} \\
&\leq \frac{(1+\beta^2)^2}{4\beta^2}(\ln s + 1) \\
&\qquad\qquad \left(\text{since } \textstyle\sum_{k=1}^{s} \frac{1}{k} \leq \ln s + 1\right).
\end{aligned}$$

This gives

$$\max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \;\leq\; \frac{(1+\beta^2)}{2\beta}\sqrt{\ln s + 1}\,. \tag{2.15}$$

Combining Eqs. (2.13-2.15) and applying Jensen's inequality (to the convex function $g(z) = z^2$) proves the claim. $\square$

Remark. We note that Theorem 2.3 gives a self-contained proof that the surrogate-mapping pair (ψ, decode) defined in Eqs. (2.6-2.7) is $\mathbf{L}^{F_\beta}$-calibrated, since the result implies that for any sequence of models $\mathbf{f}_S$ learned from training samples $S \sim D^m$ of increasing size m,

$$\text{regret}_D^{\psi}[\mathbf{f}_S] \xrightarrow{P} 0 \implies \text{regret}_D^{F_\beta}[\text{decode} \circ \mathbf{f}_S] \xrightarrow{P} 0\,.$$

Nevertheless, since the design of our surrogate-mapping pair (ψ, decode) was based on the work of Ramaswamy et al. (2014), we chose to present their calibration result (Theorem 2.1) first. We also note that, while we have stated the above regret transfer bound for the F_β-measure, a similar bound also applies more generally to all multiclass problems with low-rank matrices as considered in Theorem 2.1, thus yielding a stronger (quantitative) result than Theorem 2.1 (Ramaswamy, 2015).

2.6. Relationship with Plug-in Algorithm of Dembczynski et al. (2013)

The plug-in algorithm of Dembczynski et al. (2013), termed *exact F-measure plug-in* (EFP), estimates the following statistics of the conditional label distribution $p(\mathbf{y}|x)$:

$$\begin{aligned}&\mathbf{P}(\|\mathbf{y}\|_1 = 0 \,|\, x)\\ &\mathbf{P}(\|\mathbf{y}\|_1 = k, y_j = 1 \,|\, x)\,,\ \mathbf{P}(y_j = 0 \,|\, x)\,, \quad j, k \in [s]\,.\end{aligned}$$

It formulates estimation of the first statistic above as a binary CPE problem (solved via binary logistic regression), and estimation of the remaining statistics as s multiclass CPE problems (one for each $j \in [s]$), each with $s+1$ classes (solved via multiclass logistic regression). In practice, since the label vectors $\mathbf{y}$ are typically sparse (only a small subset of the s labels are active in any instance), the effective number of classes for each of the s problems is much smaller than $s+1$, and Dembczynski et al. (2013) exploit this fact by considering the statistics $\mathbf{P}(\|\mathbf{y}\|_1 = k, y_j = 1 \,|\, x)$ only for small k (based on the maximum number of active labels in the training instances).

As the proof of Theorem 2.3 makes clear, our algorithm can be viewed as estimating the vector $\mathbf{q}(x) \in [0,1]^{s^2+1}$, with estimation of each component formulated as a binary CPE problem; in particular, having learned a score vector $\mathbf{f}_S : \mathcal{X} \to \mathbb{R}^{s^2+1}$, our algorithm yields $\gamma^{-1}(\mathbf{f}_S(x)) \in [0,1]^{s^2+1}$ as an estimate for $\mathbf{q}(x)$. A closer look reveals that $\mathbf{q}(x)$ captures essentially the same s^2+1 statistics as above:[5]

$$\begin{aligned} q_0(x) &= \mathbf{E}_{\mathbf{y}|x}[a_{\mathbf{y},0}] \;\;\, = \mathbf{P}(\|\mathbf{y}\|_1 = 0 \,|\, x) \\ q_{jk}(x) &= \mathbf{E}_{\mathbf{y}|x}[a_{\mathbf{y},jk}] = \mathbf{P}(\|\mathbf{y}\|_1 = k, y_j = 1 \,|\, x)\,, \quad j,k \in [s]\,. \end{aligned}$$

Thus, both algorithms effectively estimate the same statistics of the conditional label distribution $p(\mathbf{y}|x)$; indeed, these are precisely the statistics needed to compute a Bayes optimal multi-label classifier for the F_β-measure (Dembczynski et al., 2011). In practice, as with the EFP algorithm, our algorithm can also be implemented to estimate $q_{jk}(x)$ only for small values of k (i.e. values of k for which labelings $\mathbf{y}$ with $\|\mathbf{y}\|_1 = k$ are actually seen in the training data).

2.7. Experiments

We conducted two sets of experiments to evaluate our algorithm. In the first experiment, we generated synthetic data from a known distribution for which the Bayes optimal F_1-accuracy could be estimated, and tested the convergence of our algorithm to this optimal F_1 performance. In the second set of experiments, we compared the performance of our algorithm to that of other algorithms on various benchmark data sets. We summarize both sets of experiments below.

2.7.1. Synthetic Data: Convergence to Bayes Optimal F_1

In the first experiment, we tested the consistency behavior of our algorithm on a synthetic data set from a known distribution for which the Bayes optimal F_1 performance could be estimated. Specifically, we generated a multi-label data set with instances $\mathbf{x}$ in $\mathcal{X} = \mathbb{R}^{100}$ and $s = 6$ labels/tags (i.e., labelings $\mathbf{y}$ in $\{0,1\}^6$), such that the vector $\mathbf{q}(\mathbf{x}) \in [0,1]^{37}$ containing the $s^2+1 = 37$ statistics of the conditional label distribution $p(\mathbf{y}|x)$ needed to compute a Bayes optimal multi-label classifier for F_1 (see Eq. (2.12)) could be obtained from a linear function of $\mathbf{x}$. More precisely, we fixed a matrix $\mathbf{W} \in [0,1]^{37\times 100}$ with entries drawn uniformly at random from $[0,1]$; we checked that $\mathbf{W}$ has full row rank. We also fixed a vector $\boldsymbol{\alpha} \in [0.1,1]^{64}$ with entries drawn uniformly from $[0.1,1]$. To generate a data point $(\mathbf{x},\mathbf{y})$, we then did the following: we first sampled $\mathbf{p} \in \Delta_{64} \equiv \Delta_{\{0,1\}^6}$

[5]Note that for each $j \in [s]$, the $s+1$ probabilities $\mathbf{P}(\|\mathbf{y}\|_1 = k, y_j = 1 \,|\, x)$ $(k \in [s])$ and $\mathbf{P}(y_j = 0 \,|\, x)$ estimated by the j-th multiclass problem in EFP add up to 1, so the EFP algorithm effectively estimates a total of s^2+1 statistics.

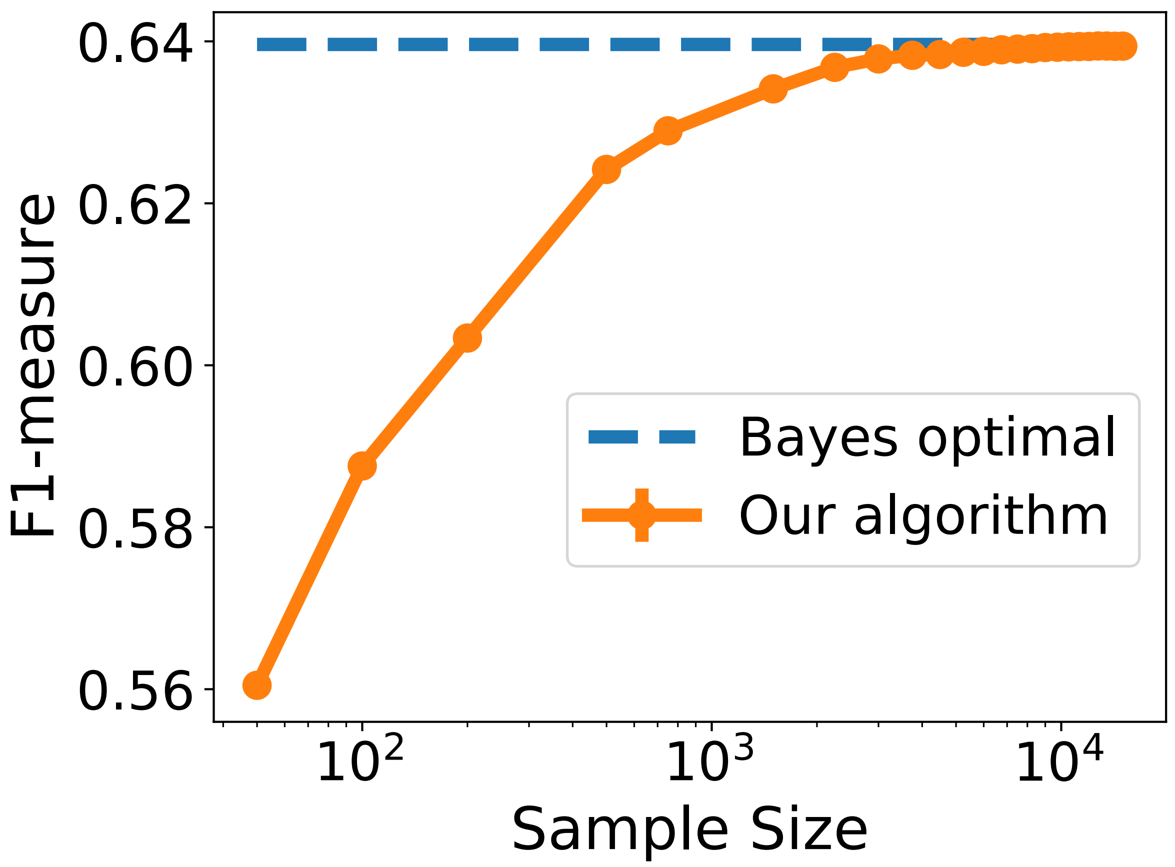

Figure 2.2: Convergence of our algorithm to Bayes optimal F_1 performance on synthetic multi-label data (see Section 2.7.1).

from Dirichlet($\boldsymbol{\alpha}$). We set $\mathbf{q} = \mathbf{E}_{\mathbf{y}\sim\mathbf{p}}[\mathbf{a}_\mathbf{y}] \in [0,1]^{37}$, where $\mathbf{a}_\mathbf{y} \in \{0,1\}^{37}$ is as defined in Eq. (2.10). We then took $\mathbf{x} = \mathbf{W}^\dagger \gamma_{\log}(\mathbf{q})$, and drew $\mathbf{y} \sim \mathbf{p}$ (here $\mathbf{W}^\dagger$ denotes the pseudo-inverse of $\mathbf{W}$). It can be verified that this gives $\mathbf{q}(\mathbf{x}) = \mathbf{q} = \gamma_{\log}^{-1}(\mathbf{W}\mathbf{x})$, and therefore, taking the function class $\mathcal{F}$ in our algorithm to be the class of linear functions (i.e., functions of the form $\mathbf{x} \mapsto \mathbf{V}\mathbf{x}$ for $\mathbf{V} \in \mathbb{R}^{37\times 100}$) suffices to learn a Bayes optimal multi-label classifier.

With the above settings, we used our algorithm (with logistic binary loss $\phi_{\log}$ and linear function class) to learn a multi-label classifier from increasingly large training samples drawn according to the above distribution, and measured the F_1 performance on a large test set of $15,000$ data points drawn from the same distribution. The results are shown in Figure 2.2. As can be seen, our algorithm indeed converges to a Bayes optimal classifier for F_1.

2.7.2. Real Data: Comparison with Other Algorithms

In the second set of experiments, we evaluated the performance of our algorithm on various benchmark multi-label data sets drawn from the Mulan repository.[6] Details of the data sets are provided in Table 2.1. All the data sets come with prescribed train/test splits. After training our models on the training set, we measure the *instance-averaged* F_1 performance on the test set (i.e., we compute the multi-label F_1-measure on each test example and take the average).

We compared with the following algorithms: EFP (Dembczynski et al., 2013), LIMO (label-wise version recommended for instance-averaged F_1) (Wu and Zhou, 2017), and BR (which treats the s labels as conditionally independent and trains s binary logistic regression classifiers, one for each label). All algorithms were trained to learn linear models. Regularization parameters (for regularized logistic regression in our algorithm, EFP, and BR; and for the margin-based objective in LIMO) were chosen by 5-fold cross-validation on the training set from $\{10^{-4}, \ldots, 10^3\}$ (for all algorithms, the parameter value maximizing average F_1-measure across the 5 folds was selected). For our algorithm and EFP, as discussed in Section 2.6, we generally implemented the algorithms to estimate only a small subset of the s^2+1 statistics in $\mathbf{q}(x)$ (only those corresponding to numbers of active labels seen in the training data); for the Birds data set, this resulted in poor performance for both algorithms, and so for this data set we trained both algorithms to perform a full estimation of all s^2+1 statistics.

The results are shown in Table 2.2 (the asterisks in the results for the Birds data set denote the full estimation of s^2+1 statistics for this data set, as discussed above). As expected, the performance of our algorithm is similar to that of EFP. BR, as expected, is generally a relatively weak baseline. LIMO is sometimes competitive, but since it aims to simultaneously optimize several multi-label performance measures, we do not expect it to outperform algorithms designed for a specific performance measure, and indeed this is borne out in our experiments.

[6]http://mulan.sourceforge.net/datasets-mlc.html

Table 2.1: Multi-label data sets used in experiments in Section 2.7.2.

Data set	**# train**	**# test**	**# labels**	**# features**
Scene	1211	1196	6	294
Yeast	1500	917	14	103
Birds	322	323	19	260
Medical	333	645	45	1449
Enron	1123	579	53	1001
Mediamill	30993	12914	101	120

Table 2.2: Comparison of F_1 performance of our algorithm with other MLC algorithms on various Mulan multi-label data sets. Higher values are better. See Section 2.7.2 for details and for an explanation of the asterisks for the Birds data set.

Data set	**Our algorithm**	**EFP**	**LIMO**	**BR**
Scene	**0.7445**	0.7426	0.6325	0.6009
Yeast	**0.6571**	0.6558	0.4914	0.6065
Birds	***0.5836**	*0.5293	0.5463	0.5510
Medical	0.7557	**0.7685**	0.7237	0.6507
Enron	0.5868	**0.6204**	0.5764	0.5455
Mediamill	**0.5642**	0.5600	0.5135	0.5229

2.8. Conclusion

We have provided a family of convex calibrated surrogate losses for the multi-label F_β-measure, together with a quantitative regret transfer bound. Our surrogates effectively decompose the F_β learning problem over s labels into (at most) $s^2 + 1$ binary class probability estimation (CPE) problems. The regret transfer bound allows us to transfer any regret guarantees on the binary CPE learners to regret guarantees on the overall F_β learner. Although motivated from a different viewpoint, like the EFP algorithm of Dembczynski et al. (2013), our algorithm can also be viewed as a type of 'plug-in' algorithm for the F_β-measure. While we have described the algorithm in the context of multi-label classification, the algorithm can also be used for binary sequence labeling tasks where the F_β-measure is useful.

CHAPTER 3

COMPLEX LABEL SPACE: MULTI-LABEL LEARNING FOR MULTIPLE PERFORMANCE MEASURES WITHOUT RE-TRAINING

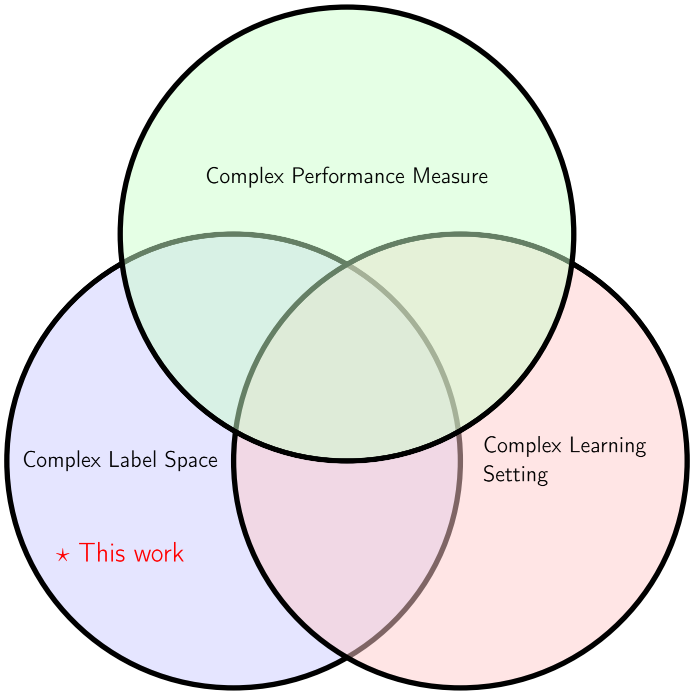

Figure 3.1: Position of *Multi-Label Learning for Multiple Performance Measures without Re-training* in the thesis.

In this chapter, we continue our discussion of multi-label classification problems. We show how to design consistent algorithms to optimize for several widely used multi-label performance measures simultaneously (i.e., without re-training).

3.1. Introduction

3.1.1. Background and Our Contributions

In contrast to binary or multiclass classification, where 0-1loss is the standard performance measure, there is no such canonical performance measure for multi-label classification. Owing to the complexity of multi-label classification, various performance measures have been proposed to assess multi-label classifiers. These include Hamming loss, subset 0-1loss (subset accuracy), precision, recall, and F_1-measure (Dembczynski et al., 2010b; Wu and Zhou, 2017; Menon et al., 2019).

It has been observed that different algorithms tend to perform variably across different performance measures. There has been progress in understanding the reasons behind these performance variations, identifying which algorithms excel under specific performance measures, and finding connections between different performance measures. This knowledge can be beneficial for multiple reasons:

Algorithm selection: By understanding the strengths and weaknesses of various algorithms with respect to different performance measures, practitioners can make more informed decisions when selecting an algorithm for a particular task, ultimately leading to better performance.

Algorithm optimization: Gaining insight into why certain algorithms perform well under specific performance measures can help researchers develop new methods or improve existing ones to optimize performance for different measures.

Theoretical advancements: Investigating the performance of algorithms across different performance measures contributes to the theoretical understanding of multi-label classification, paving the way for further developments in the field and inspiring novel approaches.

Performance benchmarking: A comprehensive understanding of algorithm performance with respect to various measures facilitates the establishment of benchmarking standards, enabling fair and meaningful comparisons between different algorithms.

In particular, Dembczynski et al. (2010b) showed that Hamming loss and subset 0-1loss could not be optimized at the same time. They performed a regret analysis showing quantitatively that a multi-label classifier intended to minimize the subset 0-1loss can become very poor in terms of Hamming loss, and vice versa. Wu and Zhou (2017) proposed a unified margin view to study eleven performance measures in multi-label classification. They defined label-wise margin and instance-wise margin, and showed that one can optimize different performance measures by maximizing label-wise margin or instance-wise margin. Menon et al. (2019) studied several multi-label learning algorithms from the perspective of reduction: how they reduce to some binary or multiclass problems. They focused on *precision@k* and *recall@k* performance measures, and studied five commonly used reductions. They explicated the underlying risks for these reductions, and showed they are either consistent with respect to either precision or recall. Furthermore, they showed that in general no reduction can be optimal for both precision and recall. Wu et al. (2018) made a counter-intuitive observation that when the label space was small, algorithms aiming to optimize Hamming loss often had better performance on the subset 0-1loss than the algorithms that optimize subset 0-1loss directly. This is inconsistent with Dembczynski et al. (2010b). As an attempt to fill this gap, Wu and Zhu (2020) analyzed the generalization bounds for the algorithms on various performance measures including Hamming loss, subset 0-1loss and ranking loss.

Still, there is a lack of principled understanding of whether it is possible to design a multi-label learning algorithm such that when given a pre-specified set of performance measures and a sufficiently large training sample, it can output a multi-label classifier whose performance converges to the corresponding Bayes optimal performance as the training size increases, for each of the performance measures in the set (with some tweaks to the learned classifier, but re-training is not allowed).

In this work, we study this problem by utilizing the theory of convex calibrated surrogates. We first show that it is possible to design one convex calibrated surrogate with respect to several performance measures so that one can train using the surrogate once and then apply different post-processing functions to optimize different performance measures. Then we show how to optimize Hamming loss, precision, recall and Top@k using a learned scoring function for F_β-measure. Finally, we provide a regret transfer bound for our method to show it is Bayes consistent.

Methodology. We study the problem by utilizing the theory of convex calibrated surrogates. Convex calibrated surrogates are a class of surrogate loss functions used in machine learning, particularly in classification problems. They are designed to approximate the target performance measures (typically discrete losses) while retaining desirable properties such as convexity and calibration. The notion of calibration ensures that minimizing the surrogate loss can (in the limit of a sufficiently large training sample) recover a Bayes optimal model for the target discrete loss. The purpose of using convex calibrated surrogate loss functions is to make the optimization problem more tractable and efficient while recovering a Bayes optimal model for the target performance measure. In recent years, the development of convex calibrated surrogates has expanded to address more intricate learning problems. In particular, we build solutions to the problem based on the following works.

- Ramaswamy and Agarwal (2012, 2016) introduced the notion of *convex calibration dimension* of a multiclass loss $\mathbf{L}$ that measures the smallest 'size' of the surrogate prediction space in which there exists a convex calibrated surrogate with respect to $\mathbf{L}$. Since in principle, multi-label classification problems can be treated as multiclass classification problems with huge label spaces, we can use these works to study upper bounds on the dimension of the surrogate prediction space in which it is possible to design convex calibrated surrogate with respect to several performance measures.

- Ramaswamy et al. (2014) studied consistency properties of output coding based methods for multiclass learning problems with a general loss matrix $\mathbf{L}$. In our previous work (Zhang et al., 2020), we have utilized the results of Ramaswamy et al. (2014) to design output coding based algorithms that are consistent with respect to F_β-measure. We can also use this work to study the commonality between output coding based methods for various multi-label performance measures.

- Papers that have studied and proposed consistent algorithms for commonly used multi-label performance measures are also helpful (Zhang and Zhou, 2014; Ramaswamy et al., 2013; Wydmuch et al., 2018; Menon et al., 2019; Dembczynski et al., 2013; Zhang et al., 2020; Yang and Koyejo, 2020).

3.1.2. Notation

For an integer n, we denote by $[n]$ the set of integers $\{1, \ldots, n\}$, and by Δ_n the probability simplex $\{\mathbf{p} \in \mathbb{R}_+^n : \sum_{y=1}^n p_y = 1\}$. For a vector $\mathbf{a}$, we denote by $\|\mathbf{a}\|_p$ the L_p norm of $\mathbf{a}$, and by a_j the j-indexed entry of $\mathbf{a}$. For a matrix $\mathbf{A}$, we denote by $\|\mathbf{A}\|_p$ the induced p-norm of $\mathbf{A}$, and by $\mathbf{a}_y$ the

y-indexed column vector of $\mathbf{A}$. $(\mathbf{a}_y)_j$ is the j-indexed entry of $\mathbf{a}_y$. Indicator function is $\mathbf{1}(\cdot)$.

3.1.3. Related Work

There has been much work on multi-label learning and convex calibrated surrogates. Below we briefly discuss work that is most related to our study. For detailed surveys on multi-label learning, we refer the reader to Zhang and Zhou (2014) and Pillai et al. (2017).

Bayes optimal multi-label classifiers and consistent algorithms. When one wants to design a good algorithm for a target performance measure ℓ in MLC, the standard goal will be to design *(Bayes) consistent* algorithms for ℓ, i.e., algorithms whose ℓ-regret converges (in probability) to zero as the number of training examples increases. For example, binary relevance is known to yield a consistent algorithm for Hamming loss (Zhang and Zhou, 2014). Ramaswamy et al. (2013) proposed convex calibrated surrogates for the precision measure. Wydmuch et al. (2018) also studied consistent algorithms for the precision measure. Menon et al. (2019) studied Bayes optimal predictions for the recall measure. Dembczynski et al. (2013); Zhang et al. (2020) proposed consistent algorithms for F_β-measure. Yang and Koyejo (2020) studied consistent algorithms for the Top@k accuracy.

Convex calibrated surrogates. Convex surrogate losses are frequently used in machine learning to design computationally efficient learning algorithms. The notion of calibrated surrogate losses, which ensures that minimizing the surrogate loss can (in the limit of sufficient data) recover a Bayes optimal model for the target discrete loss, was initially studied in the context of binary classification (Bartlett et al., 2006; Zhang, 2004a) and multiclass 0-1 classification (Zhang, 2004b; Tewari and Bartlett, 2007). In recent years, calibrated surrogates have been designed for several more complex learning problems, including general multiclass problems and certain types of subset ranking and multi-label problems (Steinwart, 2007; Duchi et al., 2010; Gao and Zhou, 2013; Ramaswamy et al., 2013, 2014, 2015).

3.1.4. Organization

Section 3.2 gives preliminaries and background. Section 3.3 gives a tighter upper bound for convex calibration dimension for multiple loss matrices. Section 3.4 shows how to optimize Hamming loss, precision, recall and Top@k using a learned scoring function for F_β-measure. Section 3.5 provides a regret transfer bound for our method to show it is Bayes consistent. Section 3.6 concludes this work.

3.2. Preliminaries and Background

3.2.1. Preliminaries

In an MLC problem, there is an instance space $\mathcal{X}$, and a set of s labels or tags $\mathcal{L} = [s] := \{1, \dots, s\}$ that can be associated with each instance in $\mathcal{X}$. For example, in image tagging, $\mathcal{X}$ is the set of possible images, and $\mathcal{L}$ is a set of s pre-defined tags (such as `sky`, `cloud`, `water` etc.) that can be associated with each image. The label space $\mathcal{Y}$ is $\{0,1\}^s$. The learner is given a training sample $S = ((x_1, \mathbf{y}_1), \dots, (x_m, \mathbf{y}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, where the labeling $\mathbf{y}_i \in \{0,1\}^s$ indicates which of the s tags are active in instance x_i (specifically, $y_{ij} = 1$ denotes that tag j is active in instance x_i, and $y_{ij} = 0$ denotes it is inactive). The goal is to learn from these examples a multi-label classifier $\mathbf{h} : \mathcal{X} \to \{0,1\}^s$ which, given a new instance $x \in \mathcal{X}$, predicts which tags are active or inactive via $\mathbf{h}(x) \in \{0,1\}^s$. The learned multi-label classifier $\mathbf{h}$ is evaluated by some performance measures. Below we give definitions of some commonly used performance measures in MLC.

Hamming loss. Given a true labeling $\mathbf{y} \in \{0,1\}^s$ and a predicted labeling $\widehat{\mathbf{y}} \in \{0,1\}^s$, Hamming loss $\mathrm{Ham} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is defined as

$$\mathrm{Ham}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{1}{s} \sum_{j=1}^{s} \mathbf{1}(y_j \neq \widehat{y}_j)\,. \tag{3.1}$$

Clearly, $0 \leq \mathrm{Ham}(\mathbf{y}, \widehat{\mathbf{y}}) \leq 1$. Lower values of Hamming loss correspond to better quality predictions. Hamming loss measures the fraction of the number of incorrectly predicted tags to the total number of tags.

Precision (gain). Given a true labeling $\mathbf{y} \in \{0,1\}^s$ and a predicted labeling $\widehat{\mathbf{y}} \in \{0,1\}^s$, precision $\mathrm{Prec} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is defined as

$$\mathrm{Prec}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{\sum_{j=1}^{s} y_j \widehat{y}_j}{\|\widehat{\mathbf{y}}\|_1} \,. \tag{3.2}$$

Clearly, $0 \le \mathrm{Prec}(\mathbf{y}, \widehat{\mathbf{y}}) \le 1$. Higher precision values correspond to better predictions. Precision measures the fraction of true active tags among predicted active tags.

Recall (gain). Given a true labeling $\mathbf{y} \in \{0,1\}^s$ and a predicted labeling $\widehat{\mathbf{y}} \in \{0,1\}^s$, recall $\mathrm{Rec} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is defined as

$$\mathrm{Rec}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{\sum_{j=1}^{s} y_j \widehat{y}_j}{\|\mathbf{y}\|_1} \,. \tag{3.3}$$

Clearly, $0 \le \mathrm{Rec}(\mathbf{y}, \widehat{\mathbf{y}}) \le 1$. Higher recall values correspond to better predictions. We take $\frac{0}{0} = 1$, so that when $\mathbf{y} = \widehat{\mathbf{y}} = \mathbf{0}$, we have $\mathrm{Rec}(\mathbf{0}, \mathbf{0}) = 1$. Recall measures the fraction of predicted active tags among true active tags.

***F_β*-measure (gain).** F_β-measure balances precision and recall by taking their (weighted) harmonic mean. Specifically, given a true labeling $\mathbf{y} \in \{0,1\}^s$ and a predicted labeling $\widehat{\mathbf{y}} \in \{0,1\}^s$, F_β-measure $F_\beta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is defined as

$$\begin{aligned} F_\beta(\mathbf{y}, \widehat{\mathbf{y}}) &= \left(\left(\frac{\beta^2}{1+\beta^2} \right) \frac{1}{\mathrm{Rec}(\mathbf{y}, \widehat{\mathbf{y}})} + \left(\frac{1}{1+\beta^2} \right) \frac{1}{\mathrm{Prec}(\mathbf{y}, \widehat{\mathbf{y}})} \right)^{-1} \\ &= \frac{(1+\beta^2) \sum_{j=1}^{s} y_j \widehat{y}_j}{\beta^2 \|\mathbf{y}\|_1 + \|\widehat{\mathbf{y}}\|_1} \,. \end{aligned} \tag{3.4}$$

Clearly, $0 \le F_\beta(\mathbf{y}, \widehat{\mathbf{y}}) \le 1$. Higher values of the F_β-measure correspond to better quality predictions. We take $\frac{0}{0} = 1$, so that when $\mathbf{y} = \widehat{\mathbf{y}} = \mathbf{0}$, we have $F_\beta(\mathbf{0}, \mathbf{0}) = 1$. The most commonly used instantiation is the F_1-measure, which weighs recall and precision equally; other commonly used variants include the F_2-measure, which weighs recall more heavily than precision, and the $F_{0.5}$-measure, which weighs precision more heavily than recall.

Top@k accuracy. Given a true labeling $\mathbf{y} \in \{0,1\}^s$ with $\|\mathbf{y}\|_1 = 1$ and a predicted labeling $\widehat{\mathbf{y}} \in \{0,1\}^s$, Top@$k$ accuracy Top@$k : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is defined as

$$\text{Top@}k(\mathbf{y}, \widehat{\mathbf{y}}) = \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = k) \cdot \sum_{j=1}^{s} y_j \widehat{y}_j \,. \tag{3.5}$$

Clearly, Top@$k(\mathbf{y}, \widehat{\mathbf{y}}) \in \{0,1\}$. Top@$k$ accuracy is often used to evaluate performance for challenging classification tasks such as computer vision because it can compensate for ambiguity in true labels. It allows the classifier to predict k active tags, and treat the prediction as correct if the predicted active tags include the true active tag. This performance measure has been studied in Yang and Koyejo (2020).

Generalization error and regret. Assume that training examples are drawn i.i.d. from some underlying probability distribution D on $\mathcal{X} \times \mathcal{Y}$, and consider the performance measure $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ (here, we assume ℓ is a loss; when the performance measure is a gain, one can simply negate it). Then it is natural to measure the quality of a multi-label classifier $\mathbf{h} : \mathcal{X} \to \mathcal{Y}$ by its *ℓ-generalization error*:

$$\text{er}_D^{\ell}[\mathbf{h}] = \mathbf{E}_{(x,\mathbf{y})\sim D}[\ell(\mathbf{y}, \mathbf{h}(x))] \,. \tag{3.6}$$

The *Bayes optimal ℓ-error* is then the lowest possible value of the ℓ-generalization error for D:

$$\text{er}_D^{\ell,*} = \inf_{\mathbf{h}:\mathcal{X}\to\mathcal{Y}} \text{er}_D^{\ell}[\mathbf{h}] \,. \tag{3.7}$$

The *ℓ-regret* of a multi-label classifier $\mathbf{h}$ is then the difference between the ℓ-generalization error of $\mathbf{h}$ and the Bayes ℓ-error:

$$\text{regret}_D^{\ell}[\mathbf{h}] = \text{er}_D^{\ell}[\mathbf{h}] - \text{er}_D^{\ell,*} \,. \tag{3.8}$$

Consistency. If a learning algorithm $\mathcal{A}$, when given a training sample $S = ((x_1, \mathbf{y}_1), \ldots, (x_m, \mathbf{y}_m)) \in$

$(\mathcal{X} \times \mathcal{Y})^m$ of size m, outputs a multi-label classifier $\mathbf{h}_S = \mathcal{A}(S)$ with the following properties:

$$\text{regret}_D^{\ell}[\mathbf{h}_S] \xrightarrow{P} 0 \quad \text{as} \quad m \to \infty\,, \tag{3.9}$$

then algorithm $\mathcal{A}$ is *Bayes consistent* for the given target loss ℓ.

3.2.2. Learning Goal

Instead of proposing different consistent algorithms for different performance measures, we want to understand whether one can design a multi-label learning algorithm capable of generating classifiers, such that when given a predetermined set of performance measures and an adequately large training set, the learned classifier's performance converges to the respective Bayes optimal performance as the training size increases, for each performance measure in the set (some post-processing to the learned classifier are allowed; however, re-training is not allowed).

Specifically, let $\mathcal{P} = \{\ell_1, ..., \ell_l\}$ be a set of l performance measures in MLC. Assume the underlying distribution on $\mathcal{X} \times \mathcal{Y}$ is D. We want to design a learning algorithm $\mathcal{A}$ such that when given a training sample $S = ((x_1, \mathbf{y}_1), \ldots, (x_m, \mathbf{y}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ of size m, the algorithm outputs a multi-label classifier $\mathbf{h}_S = \mathcal{A}(S)$ with the following properties:

$$\forall i = 1, 2, ..., l, \quad \text{regret}_D^{\ell_i}[\mathbf{h}_S^{(i)}] \xrightarrow{P} 0 \quad \text{as} \quad m \to \infty\,, \tag{3.10}$$

where $\mathbf{h}_S^{(i)}$ is obtained by applying post-processing to $\mathbf{h}_S$ (no training involved at this step).

Remark. In multiclass classification, one can do this for a set of cost-sensitive performance measures (including the commonly used 0-1loss). Indeed, for an n-class classification problem, one can simply learn an estimator $\widehat{\boldsymbol{\eta}}$ for the class probability function: $\boldsymbol{\eta} : \mathcal{X} \to \Delta_n$ where

$$\eta_i(x) = \mathbf{P}(Y = i|x)\,. \tag{3.11}$$

Then for any cost-sensitive loss $\mathbf{L} \in \mathbb{R}_+^{n\times n}$, it is well-known that an accurate estimator of $\boldsymbol{\eta}$ can produce a Bayes optimal classifier for $\mathbf{L}$ (for an instance x, choose a prediction that minimizes the expected loss under $\mathbf{L}$). Although, in principle, MLC can be viewed as a multiclass problem, the challenge is that the label space is too large (for MLC with s tags, the label space can be as large as $\{0,1\}^s$). Therefore, we need to design algorithms that are efficient for MLC problems.

3.2.3. Convex Calibrated Surrogates and Convex Calibration Dimension

Since the label space $\mathcal{Y}$ is fixed in MLC problems (or more generally, in multiclass classification problems), we can represent an MLC performance measure ℓ via a loss matrix $\mathbf{L}$, with entries $\ell_{\mathbf{y},\widehat{\mathbf{y}}}$ indicating the loss incurred on predicting $\widehat{\mathbf{y}}$ when the clean label is $\mathbf{y}$:

$$\ell_{\mathbf{y},\widehat{\mathbf{y}}} = \ell(\mathbf{y},\widehat{\mathbf{y}})\,.$$

Surrogate risk minimization and calibrated surrogates. Since minimizing the discrete loss $\mathbf{L}$ directly is computationally hard, a common algorithmic framework is to minimize a surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^d \to \mathbb{R}_+$ for some suitable $d \in \mathbb{Z}_+$. In particular, given a multiclass training sample S as above, one learns a d-dimensional 'scoring' function $\mathbf{f}_S : \mathcal{X} \to \mathbb{R}^d$ by solving

$$\min_{\mathbf{f}} \textstyle\sum_{i=1}^m \psi(\mathbf{y}_i, \mathbf{f}(x_i))$$

over a suitably rich class of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^d$; and then returns $\mathbf{h}_S = \text{decode} \circ \mathbf{f}_S$ for some suitable mapping decode : $\mathbb{R}^d \to \mathcal{Y}$. In practice, the surrogate ψ is often chosen to be convex in its second argument to enable efficient minimization. It is known that if the minimization is performed over a universal function class (with suitable regularization), then the resulting algorithm is universally *ψ-consistent*, i.e., that the ψ-regret converges to zero:

$$\text{regret}_D^{\psi}[\mathbf{f}_S] = \text{er}_D^{\psi}[\mathbf{f}_S] - \text{er}_D^{\psi,*} \xrightarrow{P} 0 \quad \text{as} \quad m \to \infty\,,$$

where $\text{er}_D^{\psi}[\mathbf{f}] = \mathbf{E}_{(x,\mathbf{y})\sim D}[\,\psi(\mathbf{y},\mathbf{f}(x))\,]$ is the ψ-generalization error of $\mathbf{f}$ and $\text{er}_D^{\psi,*} = \inf_{\mathbf{f}:\mathcal{X}\to\mathbb{R}^d} \text{er}_D^{\psi}[\mathbf{f}]$ is the Bayes ψ-error. The surrogate ψ, together with the mapping decode, is said to be $\mathbf{L}$-*calibrated* if this also implies $\mathbf{L}$-consistency, i.e., if

$$\text{regret}_D^{\psi}[\mathbf{f}_S] \xrightarrow{P} 0 \implies \text{regret}_D^{\mathbf{L}}[\text{decode} \circ \mathbf{f}_S] \xrightarrow{P} 0\,.$$

Thus, given a target loss $\mathbf{L}$, the task of designing an $\mathbf{L}$-consistent algorithm reduces to designing a convex $\mathbf{L}$-calibrated surrogate-mapping pair (ψ, decode); the resulting surrogate risk minimization algorithm (implemented in a universal function class with suitable regularization) is then universally $\mathbf{L}$-consistent.

Convex calibration dimension. To measure the smallest dimension of a prediction space in which it is possible to design a convex surrogate that is calibrated with respect to the loss matrix $\mathbf{L}$, Ramaswamy and Agarwal (2016) introduced the notion of *convex calibration dimension (CC dimension).*

Definition 3.1 (Convex calibration dimension; Definition 10 of Ramaswamy and Agarwal (2016))**.**

$$\text{CCdim}(\mathbf{L}) = \min\{d \in \mathbb{Z}_+ : \exists \textit{ convex } \mathbf{L}\textit{-calibrated surrogate } (\psi, \text{decode}) \textit{ acting on } \mathcal{C} \subseteq \mathbb{R}^d\}$$

More importantly, they showed that the CC dimension of a loss matrix $\mathbf{L}$ is upper bounded by its rank:

Proposition 3.1 (Corollary 13 of Ramaswamy and Agarwal (2016))**.**

$$\text{CCdim}(\mathbf{L}) \leq \text{rank}(\mathbf{L})\,.$$

This means that one could design a convex calibrated surrogate for $\mathbf{L}$ with at most $d = \text{rank}(\mathbf{L})$ dimension.

Convex calibration dimension for multiple loss matrices. Now, we want to understand the dimension needed to design a convex calibrated surrogate for multiple loss matrices at the same time. Without loss of generality, suppose there are two loss matrices $\mathbf{L}^{(1)}$ and $\mathbf{L}^{(2)}$. By Proposition 3.1, we can immediately get the following upper bound:

$$\mathrm{CCdim}(\mathbf{L}^{(1)}, \mathbf{L}^{(2)}) \leq \mathrm{rank}(\mathbf{L}^{(1)}) + \mathrm{rank}(\mathbf{L}^{(2)})\,. \tag{3.12}$$

Intuitively, one could 'stack' a convex calibrated surrogate for $\mathbf{L}^{(1)}$ and a convex calibrated surrogate for $\mathbf{L}^{(2)}$ together to create a convex calibrated surrogate for $\mathbf{L}^{(1)}$ and $\mathbf{L}^{(2)}$. Can we do better than this trivial upper bound? The answer is yes.

Below, we review a result of Ramaswamy et al. (2014), which we will use in the next section to show how to improve this upper bound.

Output coding (OC) method of Ramaswamy et al. (2014) for low-rank multiclass loss matrices. Output coding (OC) methods are a popular approach to solving multiclass learning problems. The idea is to reduce a multiclass classification problem into a set of binary classification problems via a 'code matrix'. Such approach includes the widely used one-vs-all and all-pairs methods (e.g., one-vs-all SVM and one-vs-all logistic regression), and the error-correcting output coding method of Dietterich and Bakiri (1995). Below, we review the output coding technique of Ramaswamy et al. (2014) for multiclass problems.

Given a multiclass loss matrix $\mathbf{L} \in \mathbb{R}_+^{n\times n}$ of rank r (which then can be factorized as $\mathbf{L} = \mathbf{A}^\top\mathbf{B} + \mathbf{1}\mathbf{t}^\top$ for $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{r\times n}, \mathbf{t} \in \mathbb{R}^n$, (i.e., $\ell_{y,\widehat{y}} = \mathbf{a}_y^\top \mathbf{b}_{\widehat{y}} + t_{\widehat{y}}$), where $\mathbf{1}$ is a vector of all ones), the method decomposes the multiclass problem into r binary CPE problems by 'encoding' labels into vectors in $[0,1]^r$ using a 'code matrix' $\mathbf{A}' = [\mathbf{a}'_y]_{y\in[n]} \in [0,1]^{r\times n}$, a shifted and scaled version of $\mathbf{A}$ (and vice versa). Then it solves the r binary CPE problems by minimizing a surrogate $\psi : [n] \times \mathcal{V}^r \to \mathbb{R}_+$ (here $\mathcal{V} \subseteq \mathbb{R}$ is an interval; see Definition 3.2 below), which is built from *strongly proper composite binary losses* ϕ with link function γ (see Definition 3.2 below):

$$\psi(y, \mathbf{u}) \;=\; \sum_{j=1}^{r} \Big((\mathbf{a}'_y)_j \phi(1, u_j) + (1 - (\mathbf{a}'_y)_j)\phi(0, u_j) \Big)\,. \tag{3.13}$$

Examples of ϕ include the widely used binary logistic loss and squared loss. It is known that minimizing such losses allows one to recover accurate class probability estimates for binary CPE problems (Reid and Williamson, 2010). Essentially, strongly proper composite binary loss ϕ allows us to learn a scoring function $u_j(x)$ for each j such that

$$\gamma^{-1}(u_j(x)) \approx \mathbf{E}_{y\sim \mathbf{P}(Y|X=x)}[(\mathbf{a}'_y)_j] \,.$$

Finally, the method applies decode : $\mathcal{V}^r \to \mathcal{Y}$ (by solving a linear optimization problem) to convert the binary class probability estimates into labels in $\mathcal{Y}$, as follows:

$$\text{decode}(\mathbf{u}) \;\in\; \operatorname*{argmin}_{\widehat{y}\in[n]} \sum_{j=1}^{r} (a_{\max} - a_{\min})(\mathbf{b}_{\widehat{y}})_j \gamma^{-1}(u_j) + t_{\widehat{y}} + a_{\min} \textstyle\sum_{j=1}^{r} (\mathbf{b}_{\widehat{y}})_j \,. \tag{3.14}$$

where $a_{\max}, a_{\min}$ depend on $\mathbf{A}$ (these are used to shift and scale $\mathbf{A}$ into $\mathbf{A}'$). The OC method is summarized in Algorithm 3.1.

As shown in Ramaswamy et al. (2014), Algorithm 3.1 is Bayes consistent for $\mathbf{L}$(with a sufficiently rich function class and suitable regularization).

Definition 3.2 (Strongly proper composite binary losses (Agarwal, 2014))**.** *Let $\mathcal{V} \subseteq \mathbb{R}$ be an interval. A binary loss $\phi : \{0,1\} \times \mathcal{V} \to \mathbb{R}_+$ is λ-*strongly proper composite with underlying (invertible) link function $\gamma : [0,1] \to \mathcal{V}$ *if for all $q \in [0,1]$ and $u \in \mathcal{V}$:*

$$\mathbf{E}_{y\sim \mathit{Bernoulli}(q)}\Big[\phi(y,u) - \phi(y,\gamma(q))\Big] \;\geq\; \frac{\lambda}{2}\Big(\gamma^{-1}(u) - q\Big)^2 ,$$

where $y \sim \mathit{Bernoulli}(q)$ denotes a Bernoulli random variable that takes value 1 with probability q and value 0 with probability $1-q$. One can recover class probability estimates by applying γ^{-1} to the learned real-valued scores u (Reid and Williamson, 2010).

3.3. A Tighter Upper Bound for Convex Calibration Dimension for Multiple Loss Matrices

In this section, we will use the result of Ramaswamy et al. (2014) for low-rank loss matrices to show that the convex calibration dimension for two loss matrices enjoys a tighter upper bound than Eq.

Algorithm 3.1 Output Coding (OC) Method for Multiclass Classification

```
1: Inputs:
   (1) Training sample, $S = ((x_1, y_1), \dots, (x_m, y_m)) \in (\mathcal{X} \times [n])^m$
   (2) Target loss $\mathbf{L} \in \mathbb{R}_+^{n\times n}$ factorized as $\mathbf{L} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ for some $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{r\times n}, \mathbf{t} \in \mathbb{R}^n$, (i.e., $\ell_{y,\widehat{y}} = \mathbf{a}_y^\top \mathbf{b}_{\widehat{y}} + t_{\widehat{y}}$), where $\mathbf{1}$ is a vector of all ones
2: Parameters:
   (1) Strongly proper composite binary loss $\phi : \{0,1\} \times \mathcal{V} \to \mathbb{R}_+$ with link function $\gamma : [0,1] \to \mathcal{V}$
   (2) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathcal{V}^r$
3: Define $a_{\min} = \min_{y,j} (\mathbf{a}_y)_j$ and $a_{\max} = \max_{y,j} (\mathbf{a}_y)_j$. Define $\mathbf{A}' \in [0,1]^{r\times n}$ with entries
   $$(\mathbf{a}'_y)_j = \frac{(\mathbf{a}_y)_j - a_{\min}}{a_{\max} - a_{\min}} \in [0,1]\,, \text{ where } \mathbf{a}'_y \text{ is the } y\text{-indexed column vector of } \mathbf{A}'$$
4: Define surrogate $\psi : [n] \times \mathcal{V}^r \to \mathbb{R}_+$ and decode $: \mathcal{V}^r \to \mathcal{Y}$ as follows:
   $$\psi(y, \mathbf{u}) \;=\; \sum_{j=1}^{r} \Big( (\mathbf{a}'_y)_j \phi(1, u_j) + (1 - (\mathbf{a}'_y)_j) \phi(0, u_j) \Big)\,,$$
   $$\text{decode}(\mathbf{u}) \;\in\; \operatorname*{argmin}_{\widehat{y}\in[n]} \sum_{j=1}^{r} (a_{\max} - a_{\min}) (\mathbf{b}_{\widehat{y}})_j \gamma^{-1}(u_j) + t_{\widehat{y}} + a_{\min} \textstyle\sum_{j=1}^{r} (\mathbf{b}_{\widehat{y}})_j$$
5: Compute $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f}\in\mathcal{F}} \sum_{i=1}^m \psi(y_i, \mathbf{f}(x_i))$, where $\psi$ is defined above
6: Output:
   Multiclass classifier $\widehat{h} = \text{decode} \circ \widehat{\mathbf{f}}$, where decode is defined above
```

(3.12). Our result can be easily generalized to a general number of loss matrices.

Let $\mathbf{L}^{(1)} \in \mathbb{R}_+^{n\times n}$ of rank r_1 be a multiclass loss matrix $\mathbf{L}^{(1)} = (\mathbf{A}^{(1)})^\top \mathbf{B}^{(1)} + \mathbf{1}(\mathbf{t}^{(1)})^\top$, with $\ell^{(1)}_{y,\widehat{y}} = (\mathbf{a}^{(1)}_y)^\top \mathbf{b}^{(1)}_{\widehat{y}} + t^{(1)}_{\widehat{y}}$ for some $\mathbf{a}^{(1)}_1, \dots, \mathbf{a}^{(1)}_n \in \mathbb{R}^{r_1}$, $\mathbf{b}^{(1)}_1, \dots, \mathbf{b}^{(1)}_n \in \mathbb{R}^{r_1}$ and $\mathbf{t}^{(1)} \in \mathbb{R}^{r_1}$. Let $\mathbf{L}^{(2)} \in \mathbb{R}_+^{n\times n}$ of rank r_2 be a multiclass loss matrix $\mathbf{L}^{(2)} = (\mathbf{A}^{(2)})^\top \mathbf{B}^{(2)} + \mathbf{1}(\mathbf{t}^{(2)})^\top$, with $\ell^{(2)}_{y,\widehat{y}} = (\mathbf{a}^{(2)}_y)^\top \mathbf{b}^{(2)}_{\widehat{y}} + t^{(2)}_{\widehat{y}}$ for some $\mathbf{a}^{(2)}_1, \dots, \mathbf{a}^{(2)}_n \in \mathbb{R}^{r_2}$, $\mathbf{b}^{(2)}_1, \dots, \mathbf{b}^{(2)}_n \in \mathbb{R}^{r_2}$ and $\mathbf{t}^{(2)} \in \mathbb{R}^{r_2}$.

Let

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^{(1)} \\ \mathbf{A}^{(2)} \end{bmatrix}$$

be a matrix by concatenating $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ vertically. Note that $\mathbf{A}$ has size $(r_1 + r_2) \times n$.

Let r denote the row rank of $\mathbf{A}$ (therefore, r is also rank of $\mathbf{A}$). Then there exist $\mathbf{A}' \in [0,1]^{r\times n}$, $\mathbf{S}^{(1)} \in \mathbb{R}^{r\times r_1}$, $\mathbf{S}^{(2)} \in \mathbb{R}^{r\times r_2}$, $\boldsymbol{\tau}^{(1)} \in \mathbb{R}^{r_1}$ and $\boldsymbol{\tau}^{(2)} \in \mathbb{R}^{r_2}$ such that

$$\mathbf{A}^{(1)} = (\mathbf{S}^{(1)})^\top \mathbf{A}' + \boldsymbol{\tau}^{(1)} \mathbf{1}_n^\top$$
$$\mathbf{A}^{(2)} = (\mathbf{S}^{(2)})^\top \mathbf{A}' + \boldsymbol{\tau}^{(2)} \mathbf{1}_n^\top$$

where $\mathbf{1}_n \in \mathbb{R}^n$ is a vector of all ones.

Let $\phi : \{0,1\} \times \mathcal{V} \to \mathbb{R}_+$ be a strongly proper composite binary loss with link function $\gamma : [0,1] \to \mathcal{V}$. Define a surrogate $\psi : [n] \times \mathcal{V}^r \to \mathbb{R}_+$ as:

$$\psi(y, \mathbf{u}) \;=\; \sum_{j=1}^{r} \Big((\mathbf{a}'_y)_j \phi(1, u_j) + (1 - (\mathbf{a}'_y)_j) \phi(0, u_j) \Big) \,. \tag{3.15}$$

Then, define $\text{decode}^{(1)} : \mathcal{V}^r \to \mathcal{Y}$ to optimize for loss $\mathbf{L}^{(1)}$ as:

$$\text{decode}^{(1)}(\mathbf{u}) \;\in\; \operatorname*{argmin}_{\widehat{y}\in[n]} \left\langle (\mathbf{S}^{(1)})^\top \boldsymbol{\gamma}^{-1}(\mathbf{u}) + \boldsymbol{\tau}^{(1)}, \mathbf{b}^{(1)}_{\widehat{y}} \right\rangle + t^{(1)}_{\widehat{y}} \,, \tag{3.16}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product, and $(\boldsymbol{\gamma}^{-1}(\mathbf{u}))_j = \gamma^{-1}(u_j)$.

Similarly, define $\text{decode}^{(2)} : \mathcal{V}^r \to \mathcal{Y}$ to optimize for loss $\mathbf{L}^{(2)}$ as:

$$\text{decode}^{(2)}(\mathbf{u}) \;\in\; \operatorname*{argmin}_{\widehat{y}\in[n]} \left\langle (\mathbf{S}^{(2)})^\top \boldsymbol{\gamma}^{-1}(\mathbf{u}) + \boldsymbol{\tau}^{(2)}, \mathbf{b}^{(2)}_{\widehat{y}} \right\rangle + t^{(2)}_{\widehat{y}} \,. \tag{3.17}$$

By Ramaswamy et al. (2014), $(\psi, \text{decode}^{(1)})$ is $\mathbf{L}^{(1)}$-calibrated, and $(\psi, \text{decode}^{(2)})$ is $\mathbf{L}^{(2)}$-calibrated. In Section 3.5, we will formally prove the consistency of the proposed approach via a regret transfer bound.

The proposed approach above can be adapted to show that one can design a convex calibrated surrogate for two loss matrices $\mathbf{L}^{(1)}$ and $\mathbf{L}^{(2)}$ at the same time with dimension equal to the rank of

$$\begin{bmatrix} (\mathbf{L}^{(1)})^\top \\ (\mathbf{L}^{(2)})^\top \end{bmatrix} \in \mathbb{R}_+^{2n\times n} .$$

We summarize this observation in a proposition below.

Proposition 3.2.

$$\mathrm{CCdim}(\mathbf{L}^{(1)}, \mathbf{L}^{(2)}) \leq \mathrm{rank}(\mathbf{L}^\top) = \mathrm{rank}(\mathbf{L})\,, \quad \mathbf{L}^\top = \begin{bmatrix} (\mathbf{L}^{(1)})^\top \\ (\mathbf{L}^{(2)})^\top \end{bmatrix} \in \mathbb{R}_+^{2n\times n} . \tag{3.18}$$

Proof. See the discussion above. □

Note that

$$\mathrm{rank}(\mathbf{L}) \leq \mathrm{rank}(\mathbf{L}^{(1)}) + \mathrm{rank}(\mathbf{L}^{(2)}) .$$

Therefore, the bound in Eq. (3.18) is tighter than Eq. (3.12).

3.4. Optimizing Multiple Performance Measures in MLC Without Re-training

In this section, we apply the approach proposed in Section 3.3 to show how to optimize multiple performance measures in MLC without re-training. Specifically, we show that once we have optimized F_β-measure, we can use the learned scoring function to optimize several other MLC performance measures by applying different decode functions.

3.4.1. Optimizing F_β-measure

Factorization of F_β-measure. F_β-measure (Eq. (3.4)) is a gain function, so we define the corresponding loss matrix $\mathbf{L}^{F_\beta}$ as

$$\ell^{F_\beta}_{\mathbf{y},\widehat{\mathbf{y}}} = 1 - F_\beta(\mathbf{y}, \widehat{\mathbf{y}}) . \tag{3.19}$$

Then we have the following factorization:

Proposition 3.3 (Factorization of $\mathbf{L}^{F_\beta}$)**.**

$$
\begin{aligned}
\ell^{F_\beta}_{\mathbf{y},\widehat{\mathbf{y}}} &= 1 - \frac{(1+\beta^2)\sum_{j=1}^{s} y_j \widehat{y}_j}{\beta^2\|\mathbf{y}\|_1 + \|\widehat{\mathbf{y}}\|_1} \\
&= 1 - \mathbf{1}(\|\mathbf{y}\|_1 = 0)\cdot\mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) \\
&\quad - \sum_{k=1}^{s} \mathbf{1}(\|\mathbf{y}\|_1 = k)\frac{(1+\beta^2)\sum_{j=1}^{s} y_j \widehat{y}_j}{\beta^2 k + \|\widehat{\mathbf{y}}\|_1} \\
&= 1 - \mathbf{1}(\|\mathbf{y}\|_1 = 0)\cdot\mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) \\
&\quad - \sum_{k=1}^{s}\sum_{j=1}^{s} \mathbf{1}(\|\mathbf{y}\|_1 = k)\cdot y_j \frac{(1+\beta^2)\widehat{y}_j}{\beta^2 k + \|\widehat{\mathbf{y}}\|_1} \\
&= t_{\widehat{\mathbf{y}}} + (\mathbf{a}_{\mathbf{y}})_0 \cdot (\mathbf{b}_{\widehat{\mathbf{y}}})_0 + \sum_{j=1}^{s}\sum_{k=1}^{s} (\mathbf{a}_{\mathbf{y}})_{jk}\cdot(\mathbf{b}_{\widehat{\mathbf{y}}})_{jk}\,, \qquad (3.20)
\end{aligned}
$$

where

$$t_{\widehat{\mathbf{y}}} = 1, \qquad (3.21)$$

$$(\mathbf{a}_{\mathbf{y}})_0 = \mathbf{1}(\|\mathbf{y}\|_1 = 0), \qquad (3.22)$$

$$(\mathbf{b}_{\widehat{\mathbf{y}}})_0 = -\mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) \qquad (3.23)$$

$$(\mathbf{a}_{\mathbf{y}})_{jk} = \mathbf{1}(\|\mathbf{y}\|_1 = k)\cdot y_j, \qquad (3.24)$$

$$(\mathbf{b}_{\widehat{\mathbf{y}}})_{jk} = -\frac{(1+\beta^2)\cdot\widehat{y}_j}{\beta^2 k + \|\widehat{\mathbf{y}}\|_1}\,. \qquad (3.25)$$

Then the $\mathbf{y}$-indexed column of $\mathbf{A}^{F_\beta}$ is a vector collecting $(\mathbf{a}_{\mathbf{y}})_0$ and $(\mathbf{a}_{\mathbf{y}})_{jk}$, the $\widehat{\mathbf{y}}$-indexed column of $\mathbf{B}^{F_\beta}$ is a vector collecting $(\mathbf{b}_{\widehat{\mathbf{y}}})_0$ and $(\mathbf{b}_{\widehat{\mathbf{y}}})_{jk}$, and $\mathbf{t}^{F_\beta} = \mathbf{1}$. This factorization has been shown in Zhang et al. (2020).

Optimizing F_β-measure. We can follow the Output Coding method (Algorithm 3.1) to optimize F_β-measure (Ramaswamy et al., 2014; Zhang et al., 2020). The scoring function $\mathbf{f}$ has dimension

s^2+1. The resulting surrogate loss is

$$\begin{aligned}\psi(\mathbf{y},\mathbf{u}) =&(\mathbf{a}_{\mathbf{y}})_0 \cdot \phi(1,u_0) + (1-(\mathbf{a}_{\mathbf{y}})_0)\cdot\phi(0,u_0)\\ &+\sum_{j=1}^{s}\sum_{k=1}^{s}(\mathbf{a}_{\mathbf{y}})_{jk}\cdot\phi(1,u_{jk}) + (1-(\mathbf{a}_{\mathbf{y}})_{jk})\cdot\phi(0,u_{jk})\end{aligned} \tag{3.26}$$

where $(\mathbf{a}_{\mathbf{y}})_0$ is defined in Eq. (3.22) and $(\mathbf{a}_{\mathbf{y}})_{jk}$ is defined in Eq. (3.24). The corresponding decode function is

$$\mathrm{decode}(\mathbf{f}(x)) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} t_{\widehat{\mathbf{y}}} + \gamma^{-1}(f_0(x))\cdot(\mathbf{b}_{\widehat{\mathbf{y}}})_0 + \sum_{j=1}^{s}\sum_{k=1}^{s}\gamma^{-1}(f_{jk}(x))\cdot(\mathbf{b}_{\widehat{\mathbf{y}}})_{jk}\,.$$

Note that

$$\gamma^{-1}(f_0(x)) \approx \mathbf{E}_{\mathbf{y}\sim\mathbf{P}(\mathbf{Y}|X=x)}[(\mathbf{a}_{\mathbf{y}})_0]\,, \tag{3.27}$$

$$\gamma^{-1}(f_{jk}(x)) \approx \mathbf{E}_{\mathbf{y}\sim\mathbf{P}(\mathbf{Y}|X=x)}[(\mathbf{a}_{\mathbf{y}})_{jk}]\,,\ j,k\in[s]\,. \tag{3.28}$$

3.4.2. Optimizing Hamming Loss Using Learned Scoring Function for F_β-measure

Factorization of Hamming loss. Hamming loss (Eq. (3.1)) is a loss function, so we define the corresponding loss matrix $\mathbf{L}^{\mathrm{Ham}}$ as

$$\ell^{\mathrm{Ham}}_{\mathbf{y},\widehat{\mathbf{y}}} = \mathrm{Ham}(\mathbf{y},\widehat{\mathbf{y}})\,. \tag{3.29}$$

Then we have the following factorization:

Proposition 3.4 (Factorization of $\mathbf{L}^{\text{Ham}}$)**.**

$$\begin{aligned}\ell^{\text{Ham}}_{\mathbf{y},\widehat{\mathbf{y}}} &= \frac{1}{s}\sum_{j=1}^{s}\mathbf{1}(\widehat{y}_j \neq y_j)\\ &= \frac{1}{s}\sum_{j=1}^{s}(1-2\widehat{y}_j)y_j + \frac{1}{s}\sum_{j=1}^{s}\widehat{y}_j\\ &= \frac{1}{s}\sum_{j=1}^{s}(1-2\widehat{y}_j)\mathbf{1}(y_j=1) + \frac{1}{s}\sum_{j=1}^{s}\widehat{y}_j\,. \end{aligned} \tag{3.30}$$

Optimizing Hamming loss using learned scoring function for F_β-measure. Based on this factorization, for a given instance x, if we know

$$\mathbf{E}_{\mathbf{y}\sim\mathbf{P}(\mathbf{Y}|X=x)}[\mathbf{1}(y_j=1)]\,,\, j\in[s]\,,$$

then we can choose an optimal prediction for x by choosing $\widehat{\mathbf{y}}$ that minimizes Eq. (3.30). The question reduces to how to get $\mathbf{E}_{\mathbf{y}\sim\mathbf{P}(\mathbf{Y}|X=x)}[\mathbf{1}(y_j=1)]$ from the learned scoring function $\mathbf{f}$ for F_β-measure.

Note that

$$\begin{aligned}\mathbf{E}_{\mathbf{y}\sim\mathbf{P}(\mathbf{Y}|X=x)}[\mathbf{1}(y_j=1)] &= \mathbf{P}(y_j=1|X=x)\\ &= \sum_{k=1}^{s}\mathbf{P}(\|\mathbf{y}\|_1=k, y_j=1|X=x)\\ &= \sum_{k=1}^{s}\mathbf{E}_{\mathbf{y}\sim\mathbf{P}(\mathbf{Y}|X=x)}[(\mathbf{a}_{\mathbf{y}})_{jk}] \quad \text{(by Eq. (3.24))}\\ &\approx \sum_{k=1}^{s}\gamma^{-1}(f_{jk}(x)) \quad \text{(by Eq. (3.28))}\,. \end{aligned}$$

So, with the learned scoring function $\mathbf{f}$ for F_β-measure, the decode function to optimize Hamming loss is

$$\operatorname{decode}(\mathbf{f}(x)) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \frac{1}{s}\sum_{j=1}^{s}(1-2\widehat{y}_j)\Big(\sum_{k=1}^{s}\gamma^{-1}(f_{jk}(x))\Big) + \frac{1}{s}\sum_{j=1}^{s}\widehat{y}_j\,. \tag{3.31}$$

Remark. Normally, we need an s-dimensional scoring function to optimize Hamming loss (Eq. (3.30)) and an (s^2+1)-dimensional scoring function to optimize F_β-measure (Eq. (3.20)), for a total of s^2+s+1. With our method, the combined loss matrix of $\mathbf{L}^{F_\beta}$ and $\mathbf{L}^{\text{Ham}}$ has rank no more than s^2+1, so we only need an (s^2+1)-dimensional scoring function to optimize Hamming loss and F_β-measure at the same time.

3.4.3. Optimizing Precision Using Learned Scoring Function for F_β-measure

Factorization of precision. Precision (Eq. (3.2)) is a gain function, so we define the corresponding loss matrix $\mathbf{L}^{\text{Prec}}$ as

$$\ell^{\text{Prec}}_{\mathbf{y},\widehat{\mathbf{y}}} = 1 - \text{Prec}(\mathbf{y}, \widehat{\mathbf{y}})\,. \tag{3.32}$$

Then we have the following factorization:

Proposition 3.5 (Factorization of $\mathbf{L}^{\text{Prec}}$)**.**

$$\begin{aligned}\ell^{\text{Prec}}_{\mathbf{y},\widehat{\mathbf{y}}} &= 1 - \frac{\sum_{j=1}^{s} y_j \widehat{y}_j}{\|\widehat{\mathbf{y}}\|_1} \\ &= 1 - \sum_{j=1}^{s} y_j \cdot \frac{\widehat{y}_j}{\|\widehat{\mathbf{y}}\|_1} \\ &= 1 - \sum_{j=1}^{s} \mathbf{1}(y_j = 1) \cdot \frac{\widehat{y}_j}{\|\widehat{\mathbf{y}}\|_1}\,. \end{aligned} \tag{3.33}$$

Optimizing precision using learned scoring function for F_β-measure. Based on this factorization, for a given instance x, if we know

$$\mathbf{E}_{\mathbf{y}\sim\mathbf{P}(\mathbf{Y}|X=x)}[\mathbf{1}(y_j = 1)]\,,\ j \in [s]\,,$$

then we can choose an optimal prediction for x by choosing $\widehat{\mathbf{y}}$ that minimizes Eq. (3.33). The question reduces to how to get them from the learned scoring function $\mathbf{f}$ for F_β-measure.

We recognize $\mathbf{1}(y_j = 1), j \in [s]$ are the same factors as in Hamming loss (Eq. (3.30)). So, with the learned scoring function $\mathbf{f}$ for F_β-measure, the decode function to optimize precision is

$$\operatorname{decode}(\mathbf{f}(x)) = \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \mathcal{Y}} 1 - \sum_{j=1}^{s} \Big(\sum_{k=1}^{s} \gamma^{-1}(f_{jk}(x)) \Big) \cdot \frac{\widehat{y}_j}{\|\widehat{\mathbf{y}}\|_1} \,.$$

Remark. Normally, we need an s-dimensional scoring function to optimize precision (Eq. (3.33)) and an $(s^2 + 1)$-dimensional scoring function to optimize F_β-measure (Eq. (3.20)), for a total of $s^2 + s + 1$. With our method, the combined loss matrix of $\mathbf{L}^{F_\beta}$ and $\mathbf{L}^{\text{Prec}}$ has rank no more than s^2+1, so we only need an (s^2+1)-dimensional scoring function to optimize precision and F_β-measure at the same time.

3.4.4. Optimizing Recall Using Learned Scoring Function for F_β-measure

Factorization of recall. Recall (Eq. (3.3)) is a gain function, so we define the corresponding loss matrix $\mathbf{L}^{\text{Rec}}$ as

$$\ell^{\text{Rec}}_{\mathbf{y},\widehat{\mathbf{y}}} = 1 - \operatorname{Rec}(\mathbf{y}, \widehat{\mathbf{y}}) \,. \tag{3.34}$$

Then we have the following factorization:

Proposition 3.6 (Factorization of $\mathbf{L}^{\text{Rec}}$).

$$\begin{aligned} \ell^{\text{Rec}}_{\mathbf{y},\widehat{\mathbf{y}}} &= 1 - \frac{\sum_{j=1}^{s} y_j \widehat{y}_j}{\|\mathbf{y}\|_1} \\ &= 1 - \sum_{j=1}^{s} \frac{y_j}{\|\mathbf{y}\|_1} \cdot \widehat{y}_j \,. \end{aligned} \tag{3.35}$$

Optimizing recall using learned scoring function for F_β-measure. Based on this factorization, for a given instance x, if we know

$$\mathbf{E}_{\mathbf{y} \sim \mathbf{P}(\mathbf{Y}|X=x)} \left[\frac{y_j}{\|\mathbf{y}\|_1} \right], \, j \in [s] \,,$$

then we can choose an optimal prediction for x by choosing $\widehat{\mathbf{y}}$ that minimizes Eq. (3.35). The question reduces to how to get them from the learned scoring function $\mathbf{f}$ for F_β-measure.

Note that

$$\begin{aligned} &1 - \sum_{j=1}^{s} \frac{y_j}{\|\mathbf{y}\|_1} \cdot \widehat{y}_j \\ &= 1 - \mathbf{1}(\|\mathbf{y}\|_1 = 0) \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{k=1}^{s} \sum_{j=1}^{s} \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j \frac{\widehat{y}_j}{k} . \end{aligned}$$

We recognize $\mathbf{1}(\|\mathbf{y}\|_1 = 0)$ and $\mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j$ are the same factors as in F_β-measure. So, with the learned scoring function $\mathbf{f}$ for F_β-measure, the decode function to optimize recall is

$$\operatorname{decode}(\mathbf{f}(x)) = \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \mathcal{Y}} 1 - \gamma^{-1}(f_0(x)) \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{k=1}^{s} \sum_{j=1}^{s} \gamma^{-1}(f_{jk}(x)) \frac{\widehat{y}_j}{k} .$$

Remark. Normally, we need an s-dimensional scoring function to optimize recall (Eq. (3.35)) and an (s^2+1)-dimensional scoring function to optimize F_β-measure (Eq. (3.20)), for a total of $s^2 + s + 1$. With our method, the combined loss matrix of $\mathbf{L}^{F_\beta}$ and $\mathbf{L}^{\text{Rec}}$ has rank no more than $s^2 + 1$, so we only need an (s^2+1)-dimensional scoring function to optimize recall and F_β-measure at the same time.

3.4.5. Optimizing Top@k Using Learned Scoring Function for F_β-measure

Factorization of Top@k. Top@k (Eq. (3.5)) is a accuracy function, so we define the corresponding loss matrix $\mathbf{L}^{\text{Top@}k}$ as

$$\ell_{\mathbf{y},\widehat{\mathbf{y}}}^{\text{Top@}k} = 1 - \text{Top@}k(\mathbf{y}, \widehat{\mathbf{y}}) . \tag{3.36}$$

Then we have the following factorization:

Proposition 3.7 (Factorization of $\mathbf{L}^{\mathrm{Top@}k}$)**.**

$$\begin{aligned}
\ell^{\mathrm{Top@}k}_{\mathbf{y},\widehat{\mathbf{y}}} &= 1 - \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = k) \cdot \sum_{j=1}^{s} y_j \widehat{y}_j \\
&= 1 - \sum_{j=1}^{s} y_j \cdot \widehat{y}_j \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = k) \\
&= 1 - \sum_{j=1}^{s} \mathbf{1}(y_j = 1) \cdot \widehat{y}_j \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = k)\,. \qquad (3.37)
\end{aligned}$$

Optimizing Top@k using learned scoring function for F_β-measure. Based on this factorization, for a given instance x, if we know

$$\mathbf{E}_{\mathbf{y}\sim\mathbf{P}(\mathbf{Y}|X=x)}[\mathbf{1}(y_j = 1)]\,,\; j \in [s]\,,$$

then we can choose an optimal prediction for x by choosing $\widehat{\mathbf{y}}$ that minimizes Eq. (3.37). The question reduces to how to get them from the learned scoring function $\mathbf{f}$ for F_β-measure.

We recognize $\mathbf{1}(y_j = 1), j \in [s]$ are the same factors as in Hamming loss (Eq. (3.30)). So, with the learned scoring function $\mathbf{f}$ for F_β-measure, the decode function to optimize Top@k is

$$\mathrm{decode}(\mathbf{f}(x)) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} 1 - \sum_{j=1}^{s} \left(\sum_{k=1}^{s} \gamma^{-1}(f_{jk}(x)) \right) \cdot \widehat{y}_j \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = k)\,.$$

Remark. Normally, we need an s-dimensional scoring function to optimize Top@k (Eq. (3.37)) and an (s^2+1)-dimensional scoring function to optimize F_β-measure (Eq. (3.20)), for a total of $s^2 + s + 1$. With our method, the combined loss matrix of $\mathbf{L}^{F_\beta}$ and $\mathbf{L}^{\mathrm{Top@}k}$ has rank no more than $s^2 + 1$, so we only need an (s^2+1)-dimensional scoring function to optimize Top@k and F_β-measure at the same time.

3.5. Consistency and Regret Transfer Bounds

In this section, we provide a regret transfer bound for the method described in Section 3.3 to optimize two performance measures at the same time. The regret transfer bound can be easily generalized to optimize multiple performance measures in MLC using a learned scoring function for F_β-measure as described in Section 3.4.

Theorem 3.8. *Assume the setup as described in Section 3.3. Let $\phi : \{0,1\} \times \mathcal{V} \to \mathbb{R}_+$ be a λ-strongly proper composite binary loss with link function $\gamma : [0,1] \to \mathcal{V}$ (Definition 3.2). Let ψ, $\mathrm{decode}^{(1)}$, and $\mathrm{decode}^{(2)}$ be defined as in Eqs. (3.15,3.16,3.17). Then for all probability distributions D on $\mathcal{X} \times [n]$ and all $\mathbf{f} : \mathcal{X} \to \mathcal{V}^r$, we have*

$$\mathrm{regret}_D^{\mathbf{L}^{(1)}}[\mathrm{decode}^{(1)} \circ \mathbf{f}] \leq 2 \max_{\widehat{y} \in [n]} \|\mathbf{b}_{\widehat{y}}^{(1)}\|_2 \cdot \|\mathbf{S}^{(1)}\|_2 \cdot \sqrt{\frac{2}{\lambda} \cdot \mathrm{regret}_D^{\psi}[\mathbf{f}]}\,,$$

and

$$\mathrm{regret}_D^{\mathbf{L}^{(2)}}[\mathrm{decode}^{(2)} \circ \mathbf{f}] \leq 2 \max_{\widehat{y} \in [n]} \|\mathbf{b}_{\widehat{y}}^{(2)}\|_2 \cdot \|\mathbf{S}^{(2)}\|_2 \cdot \sqrt{\frac{2}{\lambda} \cdot \mathrm{regret}_D^{\psi}[\mathbf{f}]}\,.$$

Remark. This theorem shows that $(\psi, \mathrm{decode}^{(1)})$ is $\mathbf{L}^{(1)}$-calibrated, and $(\psi, \mathrm{decode}^{(2)})$ is $\mathbf{L}^{(2)}$-calibrated.

Proof. We use $\langle \cdot, \cdot \rangle$ to denote the standard inner product. For $\mathbf{u} \in \mathcal{V}^r$, let $\boldsymbol{\gamma}^{-1}(\mathbf{u}) \in [0,1]^r$ be such that the i-indexed entry of $\boldsymbol{\gamma}^{-1}(\mathbf{u})$ is simply $\gamma^{-1}(u_i)$. Let $\boldsymbol{\eta} : \mathcal{X} \to \Delta_n$ be class probability function under the distribution D whose components are given by for each $y \in [n]$,

$$\eta_y(x) = \mathbf{P}(Y = y \,|\, X = x)\,.$$

We prove the regret transfer bound for $(\psi, \mathrm{decode}^{(1)})$. The second regret bound can be proven in a similar way. To simplify notations, below, we use decode to denote $\mathrm{decode}^{(1)}$, $\mathbf{L}$ to denote $\mathbf{L}^{(1)}$, $\mathbf{A}$ to denote $\mathbf{A}^{(1)}$, $\mathbf{B}$ to denote $\mathbf{B}^{(1)}$, $\mathbf{t}$ to denote $\mathbf{t}^{(1)}$, $\mathbf{S}$ to denote $\mathbf{S}^{(1)}$, and $\boldsymbol{\tau}$ to denote $\boldsymbol{\tau}^{(1)}$.

$$
\begin{aligned}
&\mathrm{regret}_D^{\mathbf{L}}[\mathrm{decode} \circ \mathbf{f}] \\
&= \mathbf{E}_x \left[\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathrm{decode}(\mathbf{f}(x))} \rangle - \min_{\widehat{y} \in [n]} \langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\widehat{y}} \rangle \right] \\
&= \mathbf{E}_x \left[\max_{\widehat{y}} \langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathrm{decode}(\mathbf{f}(x))} - \boldsymbol{\ell}_{\widehat{y}} \rangle \right] \\
&= \mathbf{E}_x \left[\max_{\widehat{y}} \langle \boldsymbol{\eta}(x), \mathbf{A}^\top \mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} + t_{\mathrm{decode}(\mathbf{f}(x))} \mathbf{1} - \mathbf{A}^\top \mathbf{b}_{\widehat{y}} - t_{\widehat{y}} \mathbf{1} \rangle \right] \\
&= \mathbf{E}_x \left[\max_{\widehat{y}} \langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{y}}) + (t_{\mathrm{decode}(\mathbf{f}(x))} - t_{\widehat{y}}) \mathbf{1} \rangle \right] \\
&= \mathbf{E}_x \left[\max_{\widehat{y}} \left[\langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{y}}) \rangle + (t_{\mathrm{decode}(\mathbf{f}(x))} - t_{\widehat{y}}) \right] \right] \\
&= \mathbf{E}_x \left[\max_{\widehat{y}} \left[\langle \boldsymbol{\eta}(x), (\mathbf{S}^\top \mathbf{A}' + \boldsymbol{\tau} \mathbf{1}_n^\top)^\top (\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{y}}) \rangle + (t_{\mathrm{decode}(\mathbf{f}(x))} - t_{\widehat{y}}) \right] \right] \\
&= \mathbf{E}_x \left[\max_{\widehat{y}} \left[\langle (\mathbf{S}^\top \mathbf{A}' + \boldsymbol{\tau} \mathbf{1}_n^\top) \boldsymbol{\eta}(x), (\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{y}}) \rangle + (t_{\mathrm{decode}(\mathbf{f}(x))} - t_{\widehat{y}}) \right] \right] \\
&\qquad \text{(by property of adjoint)} \\
&= \mathbf{E}_x \left[\max_{\widehat{y}} \left[\langle \mathbf{S}^\top \mathbf{A}' \boldsymbol{\eta}(x) + \boldsymbol{\tau}, (\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{y}}) \rangle + (t_{\mathrm{decode}(\mathbf{f}(x))} - t_{\widehat{y}}) \right] \right] \\
&\le \mathbf{E}_x \left[\max_{\widehat{y}} \left[\langle \mathbf{S}^\top \big(\mathbf{A}' \boldsymbol{\eta}(x) - \boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) \big), (\mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{y}}) \rangle \right] \right] \\
&\qquad \Big(\text{Since by Eq. (3.16), } \langle \mathbf{S}^\top \boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \boldsymbol{\tau}, \mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{y}} \rangle + (t_{\mathrm{decode}(\mathbf{f}(x))} - t_{\widehat{y}}) \le 0 \text{ for all } \widehat{y} \Big) \\
&\le \mathbf{E}_x \left[\| \mathbf{S}^\top \big(\mathbf{A}' \boldsymbol{\eta}(x) - \boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) \big) \|_2 \cdot \max_{\widehat{y}} \| \mathbf{b}_{\mathrm{decode}(\mathbf{f}(x))} - \mathbf{b}_{\widehat{y}} \|_2 \right] \\
&\qquad \text{(by the Cauchy-Schwarz inequality)} \\
&\le 2 \max_{\widehat{y}} \| \mathbf{b}_{\widehat{y}} \|_2 \cdot \| \mathbf{S} \|_2 \cdot \mathbf{E}_x \left[\| \mathbf{A}' \boldsymbol{\eta}(x) - \boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) \|_2 \right]. \qquad (3.38)
\end{aligned}
$$

Then,

$$
\begin{aligned}
&\mathbf{E}_x \left[\| \mathbf{A}' \boldsymbol{\eta}(x) - \boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) \|_2^2 \right] \\
&\quad = \mathbf{E}_x \left[\sum_{j=1}^{r} \big((\mathbf{A}' \boldsymbol{\eta}(x))_j - \gamma^{-1}(f_j(x)) \big)^2 \right]. \qquad (3.39)
\end{aligned}
$$

Note that $(\mathbf{A}'\boldsymbol{\eta}(x))_j \in [0,1]$ because $\mathbf{A}' \in [0,1]^{r\times b}$. Since ϕ is a λ-strongly proper composite binary loss with underlying link function γ, we have

$$\big((\mathbf{A}'\boldsymbol{\eta}(x))_j - \gamma^{-1}(f_j(x))\big)^2 \le \frac{2}{\lambda}\mathbf{E}_{y\sim\text{Bernoulli}\big((\mathbf{A}'\boldsymbol{\eta}(x))_j\big)}\left[\phi\Big(y, f_j(x)\Big) - \phi\Big(y, \gamma\big((\mathbf{A}'\boldsymbol{\eta}(x))_j\big)\Big)\right].$$

Then Eq. (3.39) becomes

$$\begin{aligned}
&\mathbf{E}_x\left[\|\mathbf{A}'\boldsymbol{\eta}(x) - \boldsymbol{\gamma}^{-1}(\mathbf{f}(x))\|_2^2\right] \\
&= \mathbf{E}_x\left[\sum_{j=1}^r \big((\mathbf{A}'\boldsymbol{\eta}(x))_j - \gamma^{-1}(f_j(x))\big)^2\right] \\
&\le \mathbf{E}_x\left[\frac{2}{\lambda}\sum_{j=1}^r \mathbf{E}_{y\sim\text{Bernoulli}\big((\mathbf{A}'\boldsymbol{\eta}(x))_j\big)}\left[\phi\Big(y, f_j(x)\Big) - \phi\Big(y, \gamma\big((\mathbf{A}'\boldsymbol{\eta}(x))_j\big)\Big)\right]\right] \\
&= \frac{2}{\lambda}\mathbf{E}_x\left[\mathbf{E}_{y|x\sim\boldsymbol{\eta}(x)}\left[\psi(y, \mathbf{f}(x)) - \inf_{\mathbf{u}\in\mathcal{V}^r}\psi(y,\mathbf{u})\right]\right] \\
&= \frac{2}{\lambda}\text{regret}_D^{\psi}[\mathbf{f}]\,. \qquad (3.40)
\end{aligned}$$

Combining Eqs. (3.38) and (3.40) and applying Jensen's inequality (to the convex function $x \mapsto x^2$) shows

$$\text{regret}_D^{\mathbf{L}}[\text{decode}\circ\mathbf{f}] \le 2\max_{\widehat{y}}\|\mathbf{b}_{\widehat{y}}\|_2 \cdot \|\mathbf{S}\|_2 \cdot \sqrt{\frac{2}{\lambda}\text{regret}_D^{\psi}[\mathbf{f}]}\,,$$

as claimed. □

3.6. Conclusion

In this work, we study whether it is possible and how to design a multi-label learning algorithm such that it can optimize several performance measures at the same time. We have proposed a method built on the theory of convex calibrated surrogates. Our method makes it possible to design one convex calibrated surrogate with respect to several performance measures so that one can train using the surrogate once and then apply different post-processing functions to optimize different performance measures. We have provided examples showing how to optimize Hamming loss, precision, recall and Top@k using a learned scoring function for F_β-measure. Last, we have provided a regret transfer bound for our method, establishing its (Bayes) consistency property.

CHAPTER 4

COMPLEX LEARNING SETTING: LEARNING FROM NOISY LABELS WITH NO CHANGE TO THE TRAINING PROCESS

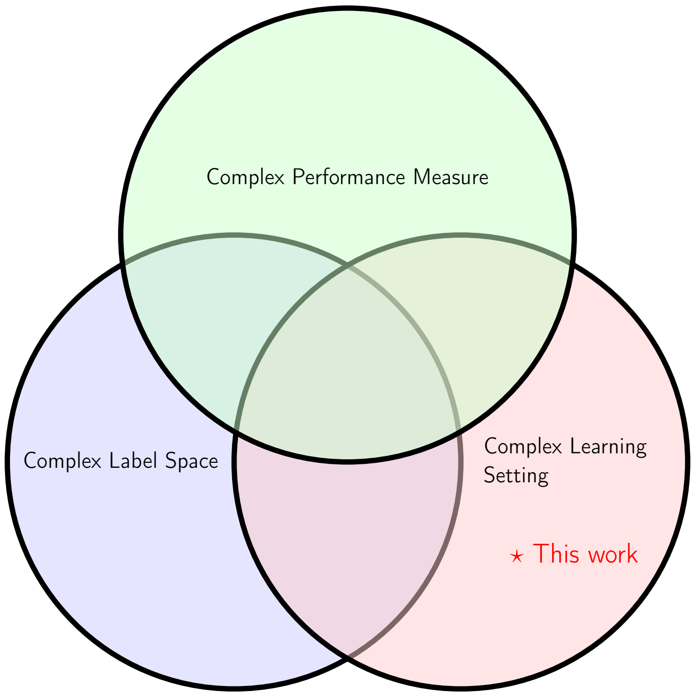

Figure 4.1: Position of *Learning from Noisy Labels with No Change to the Training Process* in the thesis.

This chapter was previously published as Mingyuan Zhang, Jane Lee, and Shivani Agarwal. Learning from noisy labels with no change to the training process. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12468–12478. PMLR, 2021. As the sole first author, I developed all the results (both theoretical and experimental) in this chapter.

In this chapter, we shift gears to begin discussing the second dimension of complexity: complex learning settings. We focus on multiclass learning from noisy labels and show how to design consistent noise-corrected algorithms to handle label noise without changing the training process.

4.1. Background of Complex Learning Setting

In machine learning, complex learning settings refer to scenarios that involve additional challenges or complexities beyond traditional supervised learning tasks. These complexities can arise due to various factors, such as the nature of the data, the learning environment, or the desired output. Addressing these challenges effectively often requires specialized algorithms or adaptations of existing methods. Examples of complex learning settings include:

Noisy labels: Noisy labels refer to situations in which some of the labels associated with instances in the training sample are incorrect or inaccurate. This can occur due to various reasons, such as human labeling errors, misalignments between labels and instances, or noise introduced during data collection. Learning from noisy labels is challenging because incorrect or inaccurate labels can adversely affect the performance of the machine learning model, leading to suboptimal predictions or increased generalization error.

Missing or partial labels: In some cases, labels might be unavailable or only partially provided for certain instances in the training data. This can happen when obtaining full labels is expensive, time-consuming, or when it is inherently difficult to provide complete label information for certain instances. In such cases, learning algorithms must make the most of the available labeled data while also leveraging the structure in the unlabeled or partially labeled data.

Imbalanced data: Imbalanced data refers to a situation in a classification problem where the distribution of classes in the dataset is significantly skewed, with some classes having much fewer instances than others. This imbalance can lead to difficulties in learning a classifier, as algorithms may become biased towards the majority classes and underperform on the minority classes. As a result, the classifier's performance may be poor on the instances with underrepresented classes, even if the overall accuracy appears to be high.

Semi-supervised learning: Semi-supervised learning is a machine learning paradigm that involves both labeled and unlabeled data during the training process. It lies between supervised learning, which relies solely on labeled data, and unsupervised learning, which uses only unlabeled data.

Semi-supervised learning aims to make the most of the available labeled data while also leveraging the structure present in the unlabeled data to improve the learning process and learn a better classifier.

Weakly supervised learning: Weakly supervised learning is a learning paradigm in which instances in the training data only have imprecise, or incomplete supervision signals. The supervision information is often considered *weak* because it is less accurate or less informative than the fully labeled data typically used in supervised learning settings. The main goal of weakly supervised learning is to make the most of the available weak supervision signals to learn a good model.

Active learning: In active learning, learning algorithms actively select the most informative instances from the available unlabeled data pool for labeling by an oracle or expert. The primary goal of active learning is to minimize the labeling effort required to achieve a desired level of performance by intelligently choosing the most informative instances to label.

Transfer learning: Transfer learning focuses on leveraging knowledge learned from one task or domain to improve the learning process in other (usually related) tasks or domains. The main goal of transfer learning is to reduce the amount of data or training time required to achieve good performance in the target task by exploiting the knowledge gained from the source task. It is closely related to feature learning and pretraining. Prominent examples of transfer learning include pretrained image classification models in computer vision and pretrained language models in natural language processing.

Online learning: In online learning, the model learns and adapts to new data points one at a time or in small batches as they become available. This is in contrast to the traditional batch learning setting, where the model is trained on a fixed dataset all at once.

Multi-task learning: Multi-task learning aims to improve the performance of multiple related tasks by learning them jointly, instead of training for each task independently. The main idea behind multi-task learning is to exploit the commonalities and relationships among the tasks to allow the model to learn shared knowledge, representations, or structures across the tasks.

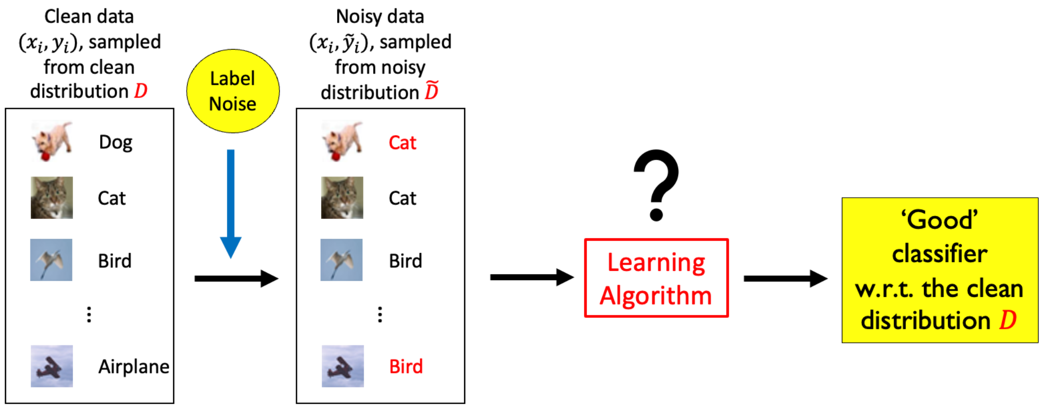
Clean data
(x_i, y_i), sampled from clean distribution D
Dog
Cat
Bird
Airplane
Label Noise
Noisy data
$(x_i, \tilde{y}_i)$, sampled from noisy distribution $\tilde{D}$
Cat
Cat
Bird
Bird
?
Learning Algorithm
'Good' classifier w.r.t. the clean distribution D

Figure 4.2: Learning from noisy labels. (Images in the left boxes are taken from the CIFAR-10 dataset (Krizhevsky and Hinton, 2009).)

Reinforcement learning: Reinforcement learning focuses on training agents to make decisions by interacting with an environment. In reinforcement learning, an agent learns to choose the best actions to take in different situations to maximize cumulative rewards. The learning process is guided by the feedback received from the environment in the form of rewards (or penalties) based on the agent's actions.

As real-world problems often involve complexities in many different aspects, by developing a deep understanding of complex learning settings, researchers can design more effective, robust, and scalable algorithms to tackle real-world challenges. It could also lead to advancements in model architecture design, representation learning, and pretrained models across various domains, ultimately benefiting a wide range of applications and industries.

4.2. Introduction

4.2.1. Background and Our Contributions

In many applications of machine learning, one receives noisy labels during training. This can happen for a variety of reasons, including human labeling errors, sensor measurement errors, distributed label collection via crowdsourcing, automatic label collection via internet crawling, and many others. Consequently, there has been much interest in recent years in developing learning algorithms that can learn accurate classifiers from data with noisy labels (Frénay and Verleysen, 2014; Song et al., 2023).

We focus here on the setting of *label-dependent noise*, where the (random) noise in a label depends on the label but not on the instance (the more general setting of *label- and instance-dependent noise* is also of interest (Menon et al., 2018; Cheng et al., 2020), but we do not focus on that here). An early example of label-dependent noise for binary classification that has been widely studied in the PAC learning literature is the *random classification noise* (RCN) model, in which a binary label y is flipped to the opposite label with a fixed probability $\gamma \in [0, \frac{1}{2})$ (Angluin and Laird, 1987; Bylander, 1994; Aslam and Decatur, 1996; Kearns, 1998; Blum and Mitchell, 1998; Cesa-Bianchi et al., 1999). More recently, Natarajan et al. (2013) generalized the RCN model and proposed the *class-conditional random label noise* (CCN) model for binary classification, in which flip probabilities for positive and negative labels can be different. This was then extended to the more general multiclass case, wherein a label y is flipped to a label $\widetilde{y}$ with some noise probability that depends on y and $\widetilde{y}$ (van Rooyen and Williamson, 2017; Patrini et al., 2017; Ghosh et al., 2017; Wang et al., 2018).

The primary challenge in learning from noisy labels is to design algorithms which, despite being given data with noisy labels as input, can learn accurate classifiers for the true, *clean* distribution (see Figure 4.2 for a summary). In particular, it is desirable to design algorithms which, when trained using a sufficiently rich function class, are statistically consistent for the clean distribution (i.e. that converge to a Bayes optimal classifier for the clean distribution). For the general multiclass CCN model, two such algorithms have been proposed: the *unbiased estimator* method of van Rooyen and Williamson (2017) (which builds on a method of Natarajan et al. (2013) for binary labels), and the *forward* method of Patrini et al. (2017) (the 'backward' method of Patrini et al. (2017) is the same as the unbiased estimators method). Both algorithms make use of the framework of surrogate loss minimization, and both require modifying the surrogate loss to correct for the noise. In practice, this means modifying the training algorithm.

In this work, we take a first-principles approach, and show that, for the general multiclass CCN model, one can design statistically consistent learning algorithms *without* modifying the training process. In particular, by examining the form of the Bayes optimal classifier for any target (cost-sensitive) multiclass loss, and the relation between the noisy and clean distributions over labels, we show that it suffices to simply implement a standard class probability estimation (CPE) algorithm (such as multiclass logistic regression) on the given noisy training data, and then apply a noise-corrected plug-in step at prediction time. For practitioners lacking expertise to modify the optimization process, or when retraining is a bottleneck, the post-processing step at prediction time can be easier to implement and use.

To establish consistency of our method (when trained with a sufficiently rich function class), we derive a quantitative regret transfer bound which shows that the target regret on the true, clean distribution can be upper bounded by the CPE regret on the noisy distribution. We also extend the notion of strong properness, defined for binary losses by Agarwal (2014), to multiclass surrogate losses; for CPE learners that minimize such surrogate losses (including for example the multiclass logistic/cross-entropy loss), we provide a regret bound in terms of the surrogate regret on the noisy distribution. Our bound suggests that as the noise matrix becomes closer to being singular, the sample size needed to achieve a given target performance level becomes larger.

In their basic forms, the methods of both van Rooyen and Williamson (2017) and Patrini et al. (2017), as well as our noise-corrected plug-in method, all assume that the noise flip probabilities are known. In practice, one may need to estimate the noise probabilities from the given noisy data. In recent years, a number of approaches have been proposed for estimating noise flip probabilities; these are generally based on identifying a small number of *anchor points* (instances that belong to a certain class with probability one). In particular, Patrini et al. (2017) proposed a noise estimation method based on anchor points, with the intent to provide an 'end-to-end' noise-estimation-and-learning method. Later, Yao et al. (2020) exploited the divide-and-conquer paradigm to propose another noise estimation method, also based on anchor points. However, it turns out that both methods do not always work correctly; we identify an error in their methods (specifically, the error is in the method for computing anchor points), and provide conditions on the noise under which the methods work or fail. We also propose an iterative noise estimation heuristic that aims to partly correct the error; while the heuristic is not guaranteed to converge or recover the correct noise probabilities, it works well in our experiments, sometimes outperforming the methods of Patrini et al. (2017) and Yao et al. (2020). Moreover, all three noise estimation methods require a CPE model to be learned from the noisy data, which in our case comes for free, with no further training required; thus our method also provides a more efficient 'end-to-end' solution.

Our experiments confirm that our noise-corrected plug-in method performs comparably to previous methods, while requiring no change to the training process.

Relationship with previous work in the binary case. As noted above, the works on learning from noisy labels in multiclass classification that are most closely related to ours are those of van Rooyen and Williamson (2017) and Patrini et al. (2017). In the special case of binary classification, two works are most directly relevant: those of Natarajan et al. (2013) and Menon et al. (2015). Natarajan et al. (2013) studied the CCN model for binary classification, and expressed the Bayes optimal classifier for the *noisy* distribution as a plug-in rule involving the *clean* class probability function (Lemma 7), and then used this to reduce the CCN learning problem to a cost-sensitive classification problem on the noisy data using classification-calibrated surrogate losses. In contrast, we express the Bayes optimal classifier for the *clean* distribution as a plug-in rule involving the *noisy* class probabilities, which can be estimated directly from the noisy data (we use strongly proper composite surrogate losses for this estimation). Menon et al. (2015) studied the more general *mutually contaminated distributions* (MCD) noise model for binary classification, and while they focused mostly on the balanced error (BER) and area under the ROC curve (AUC) metrics, they also used strongly proper composite (binary) surrogate losses, and applied their analysis to derive a regret transfer bound for the 0-1 error as well (Proposition 7). When specialized to the CCN model, their bound for binary classification with 0-1 loss can be viewed as a special case of our bound in Theorem 4.3 (our bound holds for multiclass classification with general losses).

4.2.2. Notation

For an integer n, we denote by $[n]$ the set of integers $\{1, \ldots, n\}$, and by Δ_n the probability simplex $\{\mathbf{p} \in \mathbb{R}_+^n : \sum_{y=1}^n p_y = 1\}$. For a vector $\mathbf{a}$, we denote by $\|\mathbf{a}\|_2$ the L_2 norm of $\mathbf{a}$. For a matrix $\mathbf{A}$, we denote by $\|\mathbf{A}\|_F$ the Frobenius norm of $\mathbf{A}$, by $\|\mathbf{A}\|_2$ the induced 2-norm of $\mathbf{A}$ (largest singular value of $\mathbf{A}$), and by $\mathbf{a}_y$ the y-th column vector of $\mathbf{A}$. We use $\mathbf{e}_y$ to denote a standard basis vector with y-th element 1.

4.2.3. Related Work

Noise models in learning from noisy labels. In learning from noisy labels, several noise models have been proposed and studied. In *random classification noise* (RCN) model, each label is flipped with a fixed probability $\rho \in [0, \frac{1}{2})$ (Angluin and Laird, 1987; Bylander, 1994; Kearns, 1998; Cesa-Bianchi et al., 1999; van Rooyen et al., 2015). A more general noise model is *class-conditional noise* (CCN), which says noisy labels are generated according to a fixed conditional distribution given the true class (Natarajan et al., 2013; Scott et al., 2013; Menon et al., 2015; Liu and Tao, 2016; van Rooyen and Williamson, 2017; Patrini et al., 2017). However, both RCN and CCN depend only on the labels. The most general label noise, *instance-dependent and label-dependent noise* (ILN), also depends on the instance (Menon et al., 2018; Cheng et al., 2020). In multi-label learning from noisy labels, *independent flipping noise* (IFN) model is commonly used, in which each label (tag) is independently flipped from active to non-active (or vice versa) with some probability (Kumar et al., 2020; Zhao and Gomes, 2021; Xie and Huang, 2023). Below we briefly discuss some developments in these fields and focus on works that are the most related to our study. For detailed surveys about learning from noisy labels, we refer the reader to Frénay and Verleysen (2014); Song et al. (2023); Han et al. (2020).

Binary learning from noisy labels. The initial studies focused on the RCN model and PAC-style guarantees (Angluin and Laird, 1987; Bylander, 1994; Aslam and Decatur, 1996; Kearns, 1998; Blum and Mitchell, 1998; Cesa-Bianchi et al., 1999). Some recent studies concerned about designing surrogate losses robust to RCN (Long and Servedio, 2010; van Rooyen et al., 2015; Ghosh et al., 2015). It was also mentioned in Menon et al. (2015) that for RCN, the noise rate is not needed for consistent predictions.

For CCN, Natarajan et al. (2013) and Menon et al. (2015) are the most related studies to ours and they assumed CCN is known. Natarajan et al. (2013) showed two ways of correcting surrogate losses by the noise rates so that minimizing the modified surrogates with noisy labels is consistent w.r.t. the true distribution. Menon et al. (2015) proposed to learn class probability estimation (CPE) models from noisy labels and then to apply a threshold depending on the noise rates. Other methods dealing with CCN include Stempfel and Ralaivola (2009); Scott et al. (2013); Scott (2015); Liu and Tao (2016); Patrini et al. (2016); Liu and Guo (2020). There are also results when noise rates are not known. Scott et al. (2013); Scott (2015); Menon et al. (2015); Liu and Tao (2016) proposed consistent estimators for noise rates. Liu and Guo (2020) used peer loss fcuntions.

For ILN, Menon et al. (2018) studied consistency properties with instance-dependent (but label-independent) noise, and a subclass of general ILN models which they termed as boundary consistent noise model. Cheng et al. (2020) studied bounded ILN models and proposed to use 'distilled' examples to learn from such noise.

Multiclass learning from noisy labels. Symmetric CCN here is the multiclass version of RCN in binary classification. Ghosh et al. (2017) proved a sufficient condition for a loss function to be robust to symmetric CCN. Wang et al. (2018) proposed an importance re-weighting method for symmetric CCN.

In an elegant study, van Rooyen and Williamson (2017) studied in detail learning from known CCN for multiclass problems. They generalized the unbiased estimator method in Natarajan et al. (2013) to correct surrogate losses using noise rates in the multiclass setting and provided upper and lower risk bounds. They also studied loss functions that are invariant to CCN and showed a method to construct such losses. Patrini et al. (2017) proposed two ways of modifying losses by the known CCN: forward and backward, and they showed the minimizer of the modified loss under the noisy distribution coincide with the minimizer of the original loss under the clean distribution. They also extended results in Menon et al. (2015) to estimate the noise when it is unknown.

4.2.4. Organization

After preliminaries in Section 4.3, we describe our noise-corrected plug-in method in Section 4.4. Section 4.5 gives regret transfer bounds; Section 4.6 discusses noise estimation. Section 4.7 summarizes our experiments. Section 4.8 concludes this work with a brief discussion.

4.3. Preliminaries

The problem of (multiclass) learning from noisy labels can be described as follows. There is an instance space $\mathcal{X}$, and a set of n class labels $\mathcal{Y}$, which we will take without loss of generality to be $\mathcal{Y} = [n]$. There is a (unknown) joint probability distribution D on $\mathcal{X} \times \mathcal{Y}$ from which labeled examples (X, Y) are drawn. In the standard (non-noisy) supervised learning setting, the learner would be given training examples drawn directly from D. When learning from noisy labels, however, the learner does not get clean labels Y; instead, the learner sees noisy examples $(X, \widetilde{Y})$, where $\widetilde{Y}$ denotes a noisy version of Y. In particular, the learner receives a noisy training sample $\widetilde{S} = ((x_1, \widetilde{y}_1), \ldots, (x_m, \widetilde{y}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, and the goal is to learn a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ that performs well with respect to the clean distribution D.

We consider here the *class-conditional* random label noise (CCN) model (Natarajan et al., 2013; van Rooyen and Williamson, 2017), wherein a label y is randomly flipped to a label $\widetilde{y}$ with some probability $\gamma_{y,\widetilde{y}}$ that depends on y and $\widetilde{y}$. In particular, the CCN model is characterized by a row-stochastic *noise matrix* $\mathbf{C} \in [0,1]^{n \times n}$ with entries $\gamma_{y,\widetilde{y}}$, such that for each $y, \widetilde{y} \in [n]$,

$$\mathbf{P}(\widetilde{Y} = \widetilde{y} \,|\, Y = y) = \gamma_{y,\widetilde{y}} \,.$$

The noisy training examples seen by the learner can therefore be viewed as being drawn IID from a 'noisy' distribution $\widetilde{D}$ on $\mathcal{X} \times \mathcal{Y}$, wherein an example (X, Y) is first drawn randomly according to D, and then noise is injected according to the noise matrix $\mathbf{C}$ to generate $(X, \widetilde{Y})$.

Thus, given a noisy training sample $\widetilde{S}$ drawn according to the noisy distribution $\widetilde{D}$ as above, the goal of the learner is to learn a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ that performs well under the clean distribution D. To measure performance, we consider a general multiclass loss matrix $\mathbf{L} \in \mathbb{R}_+^{n \times n}$, with entries $\ell_{y,\widehat{y}}$ indicating the loss incurred on predicting $\widehat{y}$ when the true label is y (the 0-1 loss $\mathbf{L}^{\text{0-1}}$ with $\ell^{\text{0-1}}_{y,\widehat{y}} = \mathbf{1}(\widehat{y} \neq y)$ is a special case). The performance of the classifier h is then measured by the

$\mathbf{L}$-generalization error or $\mathbf{L}$-risk under D:

$$\mathrm{er}_D^{\mathbf{L}}[h] = \mathbf{E}_{(X,Y)\sim D}\big[\ell_{Y,h(X)}\big]\,.$$

4.4. Noise-Corrected Plug-in Method

The approach we describe is conceptually very simple. We will denote by $\boldsymbol{\eta}, \widetilde{\boldsymbol{\eta}} : \mathcal{X} \to \Delta_n$ the (vector) class probability functions under the clean distribution D associated with clean labeled examples and the noisy distribution $\widetilde{D}$ associated with noisy examples, respectively, with components given by

$$\begin{aligned} \eta_y(x) &= \mathbf{P}(Y = y \,|\, X = x) \\ \widetilde{\eta}_y(x) &= \mathbf{P}(\widetilde{Y} = y \,|\, X = x) \end{aligned}$$

for each $y \in [n]$. It is easy to see that

$$\begin{aligned} \widetilde{\eta}_y(x) &= \sum_{y' \in [n]} \mathbf{P}(\widetilde{Y} = y \,|\, Y = y') \cdot \mathbf{P}(Y = y' \,|\, X = x) \\ &= \sum_{y' \in [n]} \gamma_{y',y} \cdot \eta_{y'}(x) \\ &= \mathbf{c}_y^\top \boldsymbol{\eta}(x)\,, \end{aligned}$$

which gives

$$\widetilde{\boldsymbol{\eta}}(x) = \mathbf{C}^\top \boldsymbol{\eta}(x)\,.$$

Therefore, provided $\mathbf{C}$ is invertible, we have

$$\boldsymbol{\eta}(x) \;=\; (\mathbf{C}^\top)^{-1} \widetilde{\boldsymbol{\eta}}(x)\,. \qquad (4.1)$$

This suggests that once we have an estimate of the *noisy* class probability function $\widetilde{\boldsymbol{\eta}}$, we may be able to 'de-noise' it to construct an estimate of the clean class probability function $\boldsymbol{\eta}$. This idea in its basic form can be problematic, since $\mathbf{C}^{-1}$ is not necessarily a stochastic matrix; in particular, $\mathbf{C}^\top$

generally maps probability vectors $\boldsymbol{\eta}(x)$ in the probability simplex Δ_n to noisy probability vectors $\widetilde{\boldsymbol{\eta}}(x)$ in a limited *subset* of the simplex Δ_n, and in general, an estimate of $\widetilde{\boldsymbol{\eta}}(x)$ could fall outside that subset, so that multiplying the estimate by $(\mathbf{C}^\top)^{-1}$ could then lead to an invalid 'estimate' of $\boldsymbol{\eta}(x)$ that falls outside Δ_n. Nevertheless, we get around this issue by never really needing to construct a fully valid estimate of $\boldsymbol{\eta}(x)$; instead, we simply use the above relation to derive a noise-corrected plug-in classifier that operates directly on estimates of the *noisy* class probabilities $\widetilde{\boldsymbol{\eta}}(x)$. Our regret transfer bounds in Section 4.5 will establish that this indeed leads to a correct learning approach.

We start by explaining our approach in the context of the multiclass 0-1 loss, and then describe the extension to general multiclass losses.

Multiclass 0-1 loss. As is well known, the Bayes optimal classifier for the multiclass 0-1 loss is given by

$$h_D^{\text{0-1},*}(x) \quad = \quad \operatorname*{argmax}_{y \in [n]} \, \eta_y(x) \, .$$

By Eq. (4.1), we can re-write this in terms of the *noisy* class probability function $\widetilde{\boldsymbol{\eta}}$ as follows:

$$\begin{aligned} h_D^{\text{0-1},*}(x) \quad &= \quad \operatorname*{argmax}_{y \in [n]} \, \big((\mathbf{C}^\top)^{-1} \widetilde{\boldsymbol{\eta}}(x)\big)_y \\ &=: \quad \text{plugin}_{\mathbf{C}}^{\text{0-1}}\big(\widetilde{\boldsymbol{\eta}}(x)\big) \, . \end{aligned}$$

Notably, this means that during training, we can simply construct a multiclass CPE model $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ for the *noisy* class probability function $\widetilde{\boldsymbol{\eta}}$, by running any standard multiclass CPE method (such as standard multiclass logistic regression) on the given *noisy* training examples, and then construct a noise-corrected classifier $\widehat{h} : \mathcal{X} \to \mathcal{Y}$ by applying the above noise-corrected plug-in step during prediction:

$$\widehat{h}(x) = \text{plugin}_{\mathbf{C}}^{\text{0-1}}\big(\widehat{\widetilde{\boldsymbol{\eta}}}(x)\big) \, .$$

Multiclass cost-sensitive losses. More generally, consider any multiclass loss matrix $\mathbf{L} \in \mathbb{R}_+^{n \times n}$. The Bayes optimal classifier for $\mathbf{L}$ (which for any instance x, chooses a prediction that minimizes the expected loss under $\mathbf{L}$) is given by

$$h_D^{\mathbf{L},*}(x) \quad = \quad \operatorname*{argmin}_{y \in [n]} \, \boldsymbol{\eta}(x)^\top \boldsymbol{\ell}_y \, .$$

As for the 0-1 loss, by Eq. (4.1), we can re-write this in terms of the *noisy* class probability function $\widetilde{\boldsymbol{\eta}}$ as follows:

$$\begin{aligned} h_D^{\mathbf{L},*}(x) \quad &= \quad \operatorname*{argmin}_{y \in [n]} \, \widetilde{\boldsymbol{\eta}}(x)^\top \mathbf{C}^{-1} \boldsymbol{\ell}_y \\ &= \quad \operatorname*{argmin}_{y \in [n]} \, \widetilde{\boldsymbol{\eta}}(x)^\top \big(\mathbf{C}^{-1}\mathbf{L}\big)_y \\ &=: \quad \mathrm{plugin}^{\mathbf{L}}_{\mathbf{C}}\big(\widetilde{\boldsymbol{\eta}}(x)\big) \, . \end{aligned}$$

Again, this means that during training, we can use a standard multiclass CPE learner on the noisy examples to construct a CPE model $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ for the *noisy* class probability function $\widetilde{\boldsymbol{\eta}}$, and then construct a noise-corrected classifier $\widehat{h} : \mathcal{X} \to \mathcal{Y}$ by applying the above noise-corrected plug-in step during prediction:

$$\widehat{h}(x) = \mathrm{plugin}^{\mathbf{L}}_{\mathbf{C}}\big(\widehat{\widetilde{\boldsymbol{\eta}}}(x)\big) \, .$$

Note that one can pre-compute $\mathbf{C}^{-1}\mathbf{L}$, and so at prediction time, in order to implement $\mathrm{plugin}^{\mathbf{L}}_{\mathbf{C}}(\widehat{\widetilde{\boldsymbol{\eta}}}(x))$, one needs to compute n inner products (of the column vectors of $\mathbf{C}^{-1}\mathbf{L}$ with $\widehat{\widetilde{\boldsymbol{\eta}}}(x)$), for a total computational cost of $O(n^2)$.[7,8]

Our final algorithm is shown in Algorithm 4.1. An example of a CPE learner that minimizes a (strongly) proper composite multiclass surrogate loss is provided in Section 4.5.2. In settings where the noise matrix $\mathbf{C}$ is not known, one may need to estimate $\mathbf{C}$ from the noisy training sample itself; this is discussed in Section 4.6.

[7]Also note that if one has an implementation of the standard plug-in step $\mathrm{plugin}^{\mathbf{L}}(\cdot)$ (without noise correction) for general (cost-sensitive) loss matrices $\mathbf{L}$, one can simply use that implementation with loss $\widetilde{\mathbf{L}} = \mathbf{C}^{-1}\mathbf{L}$ (since $\mathrm{plugin}^{\mathbf{L}}_{\mathbf{C}}\big(\widehat{\widetilde{\boldsymbol{\eta}}}(x)\big) = \mathrm{plugin}^{\widetilde{\mathbf{L}}}\big(\widehat{\widetilde{\boldsymbol{\eta}}}(x)\big)$).

[8]This also applies to the 0-1 loss: one can simply pre-compute $\mathbf{C}^{-1}\mathbf{L}^{0\text{-}1}$, and then at prediction time, compute n inner products (of the column vectors of $\mathbf{C}^{-1}\mathbf{L}^{0\text{-}1}$ with $\widehat{\widetilde{\boldsymbol{\eta}}}(x)$) for a cost of $O(n^2)$ (and predict according to $\mathrm{argmin}_{y \in [n]} \, \widehat{\widetilde{\boldsymbol{\eta}}}(x)^\top (\mathbf{C}^{-1}\mathbf{L}^{0\text{-}1})_y$).

Algorithm 4.1 Noise-Corrected Plug-in Algorithm

```
1: Inputs:
   (1) Noisy training sample,
         $\widetilde{S} = ((x_1, \widetilde{y}_1), \ldots, (x_m, \widetilde{y}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
   (2) Target loss matrix $\mathbf{L} \in \mathbb{R}_+^{n \times n}$
   (3) (If known) Noise matrix $\mathbf{C} \in [0,1]^{n \times n}$
2: Run a standard CPE learner on $\widetilde{S}$:
         $\widehat{\widetilde{\boldsymbol{\eta}}} = \text{CPE-Learner}(\,\widetilde{S}\,)$
3: If $\mathbf{C}$ unknown: Construct estimate $\widehat{\mathbf{C}}$ (see Section 4.6)
4: Output:
         If $\mathbf{C}$ known:     $\widehat{h} = \text{plugin}^{\mathbf{L}}_{\mathbf{C}} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$
         If $\mathbf{C}$ unknown: $\widehat{h} = \text{plugin}^{\mathbf{L}}_{\widehat{\mathbf{C}}} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$
```

4.5. Regret Transfer Bounds and Consistency

In this section, we provide quantitative regret transfer bounds for our noise-corrected plug-in algorithm; these bounds also establish that if the noisy CPE method used in training is consistent (i.e., converges to the correct noisy class probabilities), then our approach is consistent for the target learning problem. We derive our results for the multiclass case with a general loss matrix $\mathbf{L}$; they can be specialized to the binary and/or 0-1 case as needed. In particular, define the $\mathbf{L}$-*regret* (or the *excess* $\mathbf{L}$-*risk*) of a classifier $h : \mathcal{X} \to \mathcal{Y}$ under the clean distribution D as follows:

$$\text{regret}^{\mathbf{L}}_D[h] \quad = \quad \text{er}^{\mathbf{L}}_D[h] - \inf_{h' : \mathcal{X} \to \mathcal{Y}} \text{er}^{\mathbf{L}}_D[h']\,. \tag{4.2}$$

Our goal is to upper bound this $\mathbf{L}$-regret for our learned classifier $\widehat{h}$; if this regret converges (in probability, over the random draw of the noisy training sample) to zero as the training sample size increases, then the algorithm is *(Bayes) consistent* for $\mathbf{L}$ under D.

In Section 4.5.1, we provide a general result upper bounding the target $\mathbf{L}$-regret of our learned classifier $\widehat{h} = \text{plugin}^{\mathbf{L}}_{\mathbf{C}} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$ (on the clean distribution D) in terms of the noisy CPE regret of $\widehat{\widetilde{\boldsymbol{\eta}}}$ (on the noisy distribution $\widetilde{D}$). In Section 4.5.2, we specialize our result to CPE methods that learn $\widehat{\widetilde{\boldsymbol{\eta}}}$ by minimizing a *strongly proper composite* surrogate loss (extending the notion of strong properness defined for binary losses by Agarwal (2014) to the multiclass case), and apply this result in particular to the multiclass logistic loss, which we show to be 1-strongly proper composite.

4.5.1. Regret Transfer Bound for General CPE Methods

We have the following result for our noise-corrected plug-in method using any CPE learner:

Theorem 4.1. *For any noisy CPE model $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ and resulting noise-corrected plug-in classifier $\widehat{h} = \text{plugin}^{\mathbf{L}}_{\mathbf{C}} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$, we have*

$$\begin{aligned} &\text{regret}^{\mathbf{L}}_D[\,\widehat{h}\,] \\ &\leq \;\; 2 \max_y \left\|\boldsymbol{\ell}_y\right\|_2 \cdot \left\|\mathbf{C}^{-1}\right\|_2 \cdot \mathbf{E}_X\Big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_2\Big]\,. \end{aligned}$$

Proof. We use $\langle\cdot,\cdot\rangle$ to denote the standard inner product.

$$\begin{aligned} &\text{regret}^{\mathbf{L}}_D[\,\widehat{h}\,] \\ &= \mathbf{E}_X\big[\langle \boldsymbol{\eta}(X), \boldsymbol{\ell}_{\widehat{h}(X)}\rangle - \min_{y\in[n]}\langle \boldsymbol{\eta}(X), \boldsymbol{\ell}_y\rangle\big] \\ &= \mathbf{E}_X\big[\max_{y\in[n]}\langle \boldsymbol{\eta}(X), \boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y\rangle\big] \\ &= \mathbf{E}_X\big[\max_{y\in[n]}\langle (\mathbf{C}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(X), \boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y\rangle\big] \\ &= \mathbf{E}_X\big[\max_{y\in[n]}\langle \widetilde{\boldsymbol{\eta}}(X), \mathbf{C}^{-1}(\boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y)\rangle\big] \quad \text{(by property of adjoint)} \\ &\leq \mathbf{E}_X\big[\max_{y\in[n]}\langle \widetilde{\boldsymbol{\eta}}(X) - \widehat{\widetilde{\boldsymbol{\eta}}}(X), \mathbf{C}^{-1}(\boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y)\rangle\big] \\ &\qquad\quad \text{(since by the definition of } \widehat{h}(X),\ \langle \widehat{\widetilde{\boldsymbol{\eta}}}(X), \mathbf{C}^{-1}(\boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y)\rangle \leq 0\,\forall y \in [n]) \\ &\leq \mathbf{E}_X\Big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_2 \cdot \left\|\mathbf{C}^{-1}\right\|_2 \cdot \max_{y\in[n]}\big\|\boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y\big\|_2\Big] \\ &\qquad\quad \text{(by Cauchy-Schwarz inequality)} \\ &\leq 2\max_{y\in[n]}\left\|\boldsymbol{\ell}_y\right\|_2 \cdot \left\|\mathbf{C}^{-1}\right\|_2 \cdot \mathbf{E}_X\Big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_2\Big] \end{aligned}$$

□

In other words, if the learned noisy CPE model $\widehat{\widetilde{\boldsymbol{\eta}}}$ is close to the correct noisy class probabilities $\widetilde{\boldsymbol{\eta}}$, in the sense that $\mathbf{E}_X\big[\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\|_2\big]$ is small, then the target **L**-regret of the noise-corrected plug-in classifier on the clean distribution D, $\text{regret}_D^{\mathbf{L}}[\,\widehat{h}\,]$, is also small. In particular, if the CPE learner is consistent for the noisy distribution in the sense that $\mathbf{E}_X\big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_2\big] \xrightarrow{P} 0$ (as the sample size increases), then the overall noise-corrected plug-in method is (Bayes) $\mathbf{L}$-consistent for the clean distribution D, in the sense that $\text{regret}_D^{\mathbf{L}}[\,\widehat{h}\,] \xrightarrow{P} 0$.

Note that the above bound depends on the noise matrix $\mathbf{C}$ through the term $\big\|\mathbf{C}^{-1}\big\|_2$. This is the largest singular value of $\mathbf{C}^{-1}$, or equivalently, the reciprocal of the smallest singular value of $\mathbf{C}$. Thus, as the noise matrix $\mathbf{C}$ approaches singularity, the bound becomes larger. This suggests that as $\mathbf{C}$ becomes closer to being singular, we may need a higher quality class probability approximation on the noisy distribution $\widetilde{D}$ (i.e. larger sample size) to reach the same level of $\mathbf{L}$-regret on the clean distribution D. As we will see in Section 4.7, our experiments also support this observation.

4.5.2. Regret Transfer Bound for CPE Methods Minimizing a Strongly Proper Composite Surrogate Loss

In practice, a popular approach for learning CPE models is to minimize a suitable (convex) surrogate loss, such as the multiclass logistic loss (this is also what we use in our experiments). We show that for a suitable class of such surrogate losses, the CPE regret can be further upper bounded in terms of the surrogate loss based regret.

Specifically, let $\psi : [n] \times \mathbb{R}^{n-1} \to \mathbb{R}_+$ be any surrogate loss that acts on $(n-1)$-dimensional 'score vectors' in $\mathbb{R}^{n-1}$, and let $\boldsymbol{\lambda} : \Delta_n \to \mathbb{R}^{n-1}$ be an invertible 'link' function.[9] Then ψ is said to be *strictly proper composite with link function* $\boldsymbol{\lambda}$ if for all $\mathbf{p} \in \Delta_n$ and $\mathbf{u} \in \mathbb{R}^{n-1}$, $\mathbf{u} \neq \boldsymbol{\lambda}(\mathbf{p})$:

$$\mathbf{E}_{Y\sim\mathbf{p}}\big[\psi(Y,\mathbf{u}) - \psi(Y,\boldsymbol{\lambda}(\mathbf{p}))\big] > 0\,.$$

It is well known that minimizing such a strictly proper composite surrogate loss over a suitably rich function class provides consistent class probability estimates (Williamson et al., 2016). For binary surrogates, Agarwal (2014) defined a stronger condition that allows the derivation of quantitative

[9]More generally, one can consider surrogate losses $\psi : [n] \times \mathcal{C} \to \mathbb{R}_+$ acting on score vectors in any convex set $\mathcal{C}$ that is in 1-to-1 correspondence with Δ_n, such as $\mathcal{C} = \{\mathbf{u} \in \mathbb{R}^n : \sum_{i=1}^n u_i = 0\}$. It is also common to consider 'over-parameterized' surrogate losses acting on $\mathcal{C} = \mathbb{R}^n$; e.g. see the discussion on the multiclass logistic surrogate loss toward the end of the section.

bounds in terms of the surrogate regret. Here we extend this notion to the multiclass case and apply it to obtain bounds for our noisy labels problem.[10]

Definition 4.1 (Strongly proper composite multiclass losses)**.** *Let $s > 0$. We say a multiclass surrogate loss $\psi : [n] \times \mathbb{R}^{n-1} \to \mathbb{R}_+$ is* s-strongly proper composite with (invertible) link function $\boldsymbol{\lambda} : \Delta_n \to \mathbb{R}^{n-1}$ *if for all $\mathbf{p} \in \Delta_n$ and $\mathbf{u} \in \mathbb{R}^{n-1}$:*

$$\mathbf{E}_{Y\sim\mathbf{p}}\big[\psi(Y,\mathbf{u}) - \psi(Y,\boldsymbol{\lambda}(\mathbf{p}))\big] \;\geq\; \frac{s}{2}\big\|\boldsymbol{\lambda}^{-1}(\mathbf{u}) - \mathbf{p}\big\|_2^2\,.$$

As a concrete example, consider the widely used multiclass logistic surrogate loss:

Example 4.1 (Multiclass logistic loss and link function)**.** *The multiclass logistic loss $\psi_{\text{mlog}} : [n] \times \mathbb{R}^{n-1} \to \mathbb{R}_+$ is defined as*

$$\psi_{\text{mlog}}(y,\mathbf{u}) = \begin{cases} -\ln\Big(\frac{\exp(u_y)}{1+\sum_{i=1}^{n-1}\exp(u_i)}\Big) & \text{if } y \in [n-1] \\[2ex] \ln\big(1+\sum_{i=1}^{n-1}\exp(u_i)\big) & \text{if } y = n\,. \end{cases}$$

The loss is often used with the invertible link function $\boldsymbol{\lambda}_{\text{mlog}} : \Delta_n \to \mathbb{R}^{n-1}$, which together with its inverse $\boldsymbol{\lambda}_{\text{mlog}}^{-1} : \mathbb{R}^{n-1} \to \Delta_n$ is given by

$$\boldsymbol{\lambda}_{\text{mlog}}(\mathbf{p}) = \begin{pmatrix} \ln(\frac{p_1}{p_n}) \\ \vdots \\ \ln(\frac{p_{n-1}}{p_n}) \end{pmatrix}; \quad \boldsymbol{\lambda}_{\text{mlog}}^{-1}(\mathbf{u}) = \begin{pmatrix} \frac{\exp(u_1)}{1+\sum_{i=1}^{n-1}\exp(u_i)} \\ \vdots \\ \frac{\exp(u_{n-1})}{1+\sum_{i=1}^{n-1}\exp(u_i)} \\ \frac{1}{1+\sum_{i=1}^{n-1}\exp(u_i)} \end{pmatrix}.$$

We note that the multiclass logistic loss above is often implemented in an 'over-parameterized' form, with score vectors in $\mathcal{C} = \mathbb{R}^n$ and the softmax function used for 'inverting' such score vectors to class

[10]Strong properness implies strict properness. Most commonly used strictly proper composite losses are also strongly proper composite, but the latter condition allows for stronger quantitative guarantees.

probabilities (indeed, softmax is a many-to-one mapping).[11] We have the following result showing that ψ_{mlog} is strongly proper composite:

Lemma 4.2. *The multiclass logistic loss* ψ_{mlog} *is* 1*-strongly proper composite with link function* $\boldsymbol{\lambda}_{\text{mlog}}$.

Proof. We will show for all $\mathbf{p} \in \Delta_n$ and $\mathbf{u} \in \mathbb{R}^{n-1}$,

$$\mathbf{E}_{Y\sim\mathbf{p}}\Big[\psi_{\text{mlog}}(Y,\mathbf{u}) - \psi_{\text{mlog}}(Y,\boldsymbol{\lambda}_{\text{mlog}}(\mathbf{p}))\Big] \geq \frac{1}{2}\big\|\boldsymbol{\lambda}_{\text{mlog}}^{-1}(\mathbf{u}) - \mathbf{p}\big\|_1^2 \geq \frac{1}{2}\big\|\boldsymbol{\lambda}_{\text{mlog}}^{-1}(\mathbf{u}) - \mathbf{p}\big\|_2^2 .$$

Fix $\mathbf{p} \in \Delta_n$ and $\mathbf{u} \in \mathbb{R}^{n-1}$. Then

$$\begin{aligned}
&\mathbf{E}_{Y\sim\mathbf{p}}\Big[\psi_{\text{mlog}}(Y,\mathbf{u}) - \psi_{\text{mlog}}(Y,\boldsymbol{\lambda}_{\text{mlog}}(\mathbf{p}))\Big]\\
&= -\sum_{i\in[n]} p_i \ln\big((\boldsymbol{\lambda}_{\text{mlog}}^{-1}(\mathbf{u}))_i\big) + \sum_{i\in[n]} p_i \ln(p_i)\\
&= \sum_{i\in[n]} p_i \ln\left(\frac{p_i}{(\boldsymbol{\lambda}_{\text{mlog}}^{-1}(\mathbf{u}))_i}\right)\\
&= D_{KL}(\mathbf{p}||\boldsymbol{\lambda}_{\text{mlog}}^{-1}(\mathbf{u})) \quad \text{by the definition of Kullback-Leibler divergence}\\
&\geq \frac{1}{2}\big\|\mathbf{p} - \boldsymbol{\lambda}_{\text{mlog}}^{-1}(\mathbf{u})\big\|_1^2 \quad \text{using Pinsker's inequality and properties of total variation distance}\\
&\geq \frac{1}{2}\big\|\mathbf{p} - \boldsymbol{\lambda}_{\text{mlog}}^{-1}(\mathbf{u})\big\|_2^2 .
\end{aligned}$$

□

A CPE learner minimizing a strongly proper composite surrogate loss (over noisy training examples) is shown in Algorithm 4.2. (Instantiating this with the multiclass logistic loss ψ_{mlog} above and the class of linear scoring functions leads to the multiclass linear logistic regression algorithm.)

In what follows, for a surrogate loss $\psi : [n] \times \mathbb{R}^{n-1} \rightarrow \mathbb{R}_+$, we will define the *$\psi$-generalization error*

[11] The over-parameterized multiclass logistic loss is also sometimes referred to as the cross-entropy loss.

of a scoring function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^{n-1}$ under $\widetilde{D}$ as

$$\mathrm{er}^{\psi}_{\widetilde{D}}[\mathbf{f}] \;=\; \mathbf{E}_{(X,Y)\sim\widetilde{D}}\big[\psi(Y, \mathbf{f}(X))\big]\,,$$

and the ψ-*regret* of $\mathbf{f}$ under $\widetilde{D}$ as

$$\mathrm{regret}^{\psi}_{\widetilde{D}}[\mathbf{f}] \;=\; \mathrm{er}^{\psi}_{\widetilde{D}}[\mathbf{f}] - \inf_{\mathbf{f}':\mathcal{X}\to\mathbb{R}^{n-1}} \mathrm{er}^{\psi}_{\widetilde{D}}[\mathbf{f}']\,.$$

Then we have the following regret transfer bound:

Theorem 4.3. *Let $s > 0$. Let $\psi : [n] \times \mathbb{R}^{n-1} \to \mathbb{R}_+$ be a s-strongly proper composite surrogate loss with (invertible) link function $\boldsymbol{\lambda} : \Delta_n \to \mathbb{R}^{n-1}$. For any scoring model $\widehat{\widetilde{\mathbf{f}}} : \mathcal{X} \to \mathbb{R}^{n-1}$ being used as a (noisy) CPE model via $\widehat{\widetilde{\boldsymbol{\eta}}}(x) = \boldsymbol{\lambda}^{-1}(\widehat{\widetilde{\mathbf{f}}}(x))$, and resulting noise-corrected plug-in classifier $\widehat{h} = \mathrm{plugin}^{\mathbf{L}}_{\mathbf{C}} \circ \big(\boldsymbol{\lambda}^{-1} \circ \widehat{\widetilde{\mathbf{f}}}\big)$, we have*

$$\mathrm{regret}^{\mathbf{L}}_{D}[\,\widehat{h}\,] \;\le\; 2\max_{y} \big\|\boldsymbol{\ell}_y\big\|_2 \cdot \big\|\mathbf{C}^{-1}\big\|_2 \cdot \sqrt{\frac{2}{s}\,\mathrm{regret}^{\psi}_{\widetilde{D}}[\,\widehat{\widetilde{\mathbf{f}}}\,]}\,.$$

Proof. By Theorem 4.1, we have

$$\mathrm{regret}^{\mathbf{L}}_{D}[\,\widehat{h}\,] \le 2\max_{y} \big\|\boldsymbol{\ell}_y\big\|_2 \cdot \big\|\mathbf{C}^{-1}\big\|_2 \cdot \mathbf{E}_X\Big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_2\Big]\,. \tag{4.3}$$

Then, since ψ is s-strongly proper composite with link function $\boldsymbol{\lambda}$, we have

$$\begin{aligned}
&\mathrm{regret}^{\psi}_{\widetilde{D}}[\,\widehat{\widetilde{\mathbf{f}}}\,] \\
&= \mathbf{E}_X\Big[\mathbf{E}_{Y|X\sim\widetilde{\boldsymbol{\eta}}(X)}\big[\psi(Y, \widehat{\widetilde{\mathbf{f}}}(X))\big] - \inf_{\mathbf{u}\in\mathbb{R}^{n-1}} \mathbf{E}_{Y|X\sim\widetilde{\boldsymbol{\eta}}(X)}\big[\psi(Y, \mathbf{u})\big]\Big] \\
&= \mathbf{E}_X\Big[\mathbf{E}_{Y|X\sim\widetilde{\boldsymbol{\eta}}(X)}\big[\psi(Y, \widehat{\widetilde{\mathbf{f}}}(X)) - \psi(Y, \boldsymbol{\lambda}(\widetilde{\boldsymbol{\eta}}(X)))\big]\Big] \\
&\qquad\qquad \text{(by definition of strongly proper composite multiclass loss)} \\
&\ge \mathbf{E}_X\Big[\frac{s}{2}\big\|\boldsymbol{\lambda}^{-1}(\widehat{\widetilde{\mathbf{f}}}(X)) - \widetilde{\boldsymbol{\eta}}(X)\big\|_2^2\Big] \\
&= \frac{s}{2}\mathbf{E}_X\Big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_2^2\Big]
\end{aligned} \tag{4.4}$$

Algorithm 4.2 CPE Learner Minimizing a Strongly Proper Composite Surrogate Loss (on Noisy Data)

```
1: Input: Noisy training sample,
       $\widetilde{S} = ((x_1, \widetilde{y}_1), \ldots, (x_m, \widetilde{y}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
2: Parameters:
   (1) Strongly proper composite loss $\psi : [n] \times \mathbb{R}^{n-1} \to \mathbb{R}_+$ with (invertible) link function $\boldsymbol{\lambda} : \Delta_n \to \mathbb{R}^{n-1}$;
   (2) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^{n-1}$
3: Compute $\widehat{\widetilde{\mathbf{f}}} \in \mathrm{argmin}_{\mathbf{f} \in \mathcal{F}} \sum_{i=1}^{m} \psi(\widetilde{y}_i, \mathbf{f}(x_i))$
4: Output: $\widehat{\widetilde{\boldsymbol{\eta}}} = \boldsymbol{\lambda}^{-1} \circ \widehat{\widetilde{\mathbf{f}}}$
```

Combining Eqs. (4.3, 4.4), and applying Jensen's inequality (to the convex function $x \mapsto x^2$) establishes the result. $\square$

Thus in particular, if the CPE learner in Algorithm 4.2 minimizes a strongly proper composite surrogate loss ψ over a universal function class $\mathcal{F}$ (with suitable regularization), thus ensuring that $\mathrm{regret}^{\psi}_{\widetilde{D}}[\widehat{\widetilde{\mathbf{f}}}] \xrightarrow{P} 0$, then we have that $\mathrm{regret}^{\mathbf{L}}_{D}[\widehat{h}] \xrightarrow{P} 0$ as desired.

4.5.3. Improved Regret Transfer Bound

Below, we show an improved regret transfer bound for the multiclass logistic loss ψ_{mlog} (Example 4.1) that depends on the noise matrix $\mathbf{C}$ through $\|(\mathbf{C}^\top)^{-1}\|_1$ instead of $\|\mathbf{C}^{-1}\|_2$ as shown in Theorem 4.3.[12]

We start by showing the following improved result for our noise-corrected plug-in method using any CPE learner (the improved bound depends on the noise matrix $\mathbf{C}$ through $\|(\mathbf{C}^\top)^{-1}\|_1$ instead of $\|\mathbf{C}^{-1}\|_2$ as shown in Theorem 4.1):

Theorem 4.4. *For any noisy CPE model $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ and resulting noise-corrected plug-in classifier $\widehat{h} = \mathrm{plugin}^{\mathbf{L}}_{\mathbf{C}} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$, we have*

$$\begin{aligned}&\mathrm{regret}^{\mathbf{L}}_{D}[\widehat{h}] \\ &\quad\le\; 2 \max_{y} \left\|\boldsymbol{\ell}_y\right\|_\infty \cdot \left\|(\mathbf{C}^\top)^{-1}\right\|_1 \cdot \mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\right\|_1\right].\end{aligned}$$

[12]This result is not in our conference paper (Zhang et al., 2021) on which this chapter is based.

Proof. We use $\langle \cdot, \cdot \rangle$ to denote the standard inner product.

$$
\begin{aligned}
&\mathrm{regret}_D^{\mathbf{L}}[\,\widehat{h}\,] \\
&= \mathbf{E}_X\big[\langle \boldsymbol{\eta}(X), \boldsymbol{\ell}_{\widehat{h}(X)}\rangle - \min_{y\in[n]}\langle \boldsymbol{\eta}(X), \boldsymbol{\ell}_y\rangle\big] \\
&= \mathbf{E}_X\big[\max_{y\in[n]}\langle \boldsymbol{\eta}(X), \boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y\rangle\big] \\
&= \mathbf{E}_X\big[\max_{y\in[n]}\langle (\mathbf{C}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(X), \boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y\rangle\big] \\
&= \mathbf{E}_X\big[\max_{y\in[n]}\langle \widetilde{\boldsymbol{\eta}}(X), \mathbf{C}^{-1}(\boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y)\rangle\big] \quad \text{(by property of adjoint)} \\
&\le \mathbf{E}_X\big[\max_{y\in[n]}\langle \widetilde{\boldsymbol{\eta}}(X) - \widehat{\widetilde{\boldsymbol{\eta}}}(X), \mathbf{C}^{-1}(\boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y)\rangle\big] \\
&\qquad\quad \text{(since by the definition of } \widehat{h}(X),\ \langle \widehat{\widetilde{\boldsymbol{\eta}}}(X), \mathbf{C}^{-1}(\boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y)\rangle \le 0\, \forall y \in [n]) \\
&\le \mathbf{E}_X\big[\max_{y\in[n]}\langle (\mathbf{C}^\top)^{-1}(\widetilde{\boldsymbol{\eta}}(X) - \widehat{\widetilde{\boldsymbol{\eta}}}(X)), \boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y\rangle\big] \\
&\le \mathbf{E}_X\Big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_1 \cdot \big\|(\mathbf{C}^\top)^{-1}\big\|_1 \cdot \max_{y\in[n]}\big\|\boldsymbol{\ell}_{\widehat{h}(X)} - \boldsymbol{\ell}_y\big\|_\infty\Big] \\
&\qquad\quad \text{(by Hölder inequality)} \\
&\le 2\max_{y\in[n]}\big\|\boldsymbol{\ell}_y\big\|_\infty \cdot \big\|(\mathbf{C}^\top)^{-1}\big\|_1 \cdot \mathbf{E}_X\Big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_1\Big]
\end{aligned}
$$

□

Theorem 4.5. *Let $\psi_{\mathrm{mlog}} : [n] \times \mathbb{R}^{n-1} \to \mathbb{R}_+$ be the multiclass logistic loss with (invertible) link function $\boldsymbol{\lambda}_{\mathrm{mlog}} : \Delta_n \to \mathbb{R}^{n-1}$ as in Example 4.1. For any scoring model $\widehat{\widetilde{\mathbf{f}}} : \mathcal{X} \to \mathbb{R}^{n-1}$ being used as a (noisy) CPE model via $\widehat{\widetilde{\boldsymbol{\eta}}}(x) = \boldsymbol{\lambda}_{\mathrm{mlog}}^{-1}(\widehat{\widetilde{\mathbf{f}}}(x))$, and resulting noise-corrected plug-in classifier $\widehat{h} = \mathrm{plugin}_{\mathbf{C}}^{\mathbf{L}} \circ \big(\boldsymbol{\lambda}_{\mathrm{mlog}}^{-1} \circ \widehat{\widetilde{\mathbf{f}}}\big)$, we have*

$$
\mathrm{regret}_D^{\mathbf{L}}[\,\widehat{h}\,] \le 2\max_y \big\|\boldsymbol{\ell}_y\big\|_\infty \cdot \big\|(\mathbf{C}^\top)^{-1}\big\|_1 \cdot \sqrt{2\,\mathrm{regret}_{\widetilde{D}}^{\psi_{\mathrm{mlog}}}[\,\widehat{\widetilde{\mathbf{f}}}\,]}\,.
$$

Proof. By Theorem 4.4, we have

$$
\mathrm{regret}_D^{\mathbf{L}}[\,\widehat{h}\,] \le 2\max_y \big\|\boldsymbol{\ell}_y\big\|_\infty \cdot \big\|(\mathbf{C}^\top)^{-1}\big\|_1 \cdot \mathbf{E}_X\Big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_1\Big]\,. \tag{4.5}
$$

Then we have

$$
\begin{aligned}
&\mathrm{regret}_{\widetilde{D}}^{\psi_{\mathrm{mlog}}}[\widehat{\widetilde{\mathbf{f}}}] \\
&= \mathbf{E}_X\Big[\mathbf{E}_{Y|X\sim\widetilde{\boldsymbol{\eta}}(X)}\big[\psi_{\mathrm{mlog}}(Y,\widehat{\widetilde{\mathbf{f}}}(X))\big] - \inf_{\mathbf{u}\in\mathbb{R}^{n-1}} \mathbf{E}_{Y|X\sim\widetilde{\boldsymbol{\eta}}(X)}\big[\psi_{\mathrm{mlog}}(Y,\mathbf{u})\big]\Big] \\
&= \mathbf{E}_X\Big[\mathbf{E}_{Y|X\sim\widetilde{\boldsymbol{\eta}}(X)}\big[\psi_{\mathrm{mlog}}(Y,\widehat{\widetilde{\mathbf{f}}}(X)) - \psi_{\mathrm{mlog}}(Y,\boldsymbol{\lambda}_{\mathrm{mlog}}(\widetilde{\boldsymbol{\eta}}(X)))\big]\Big] \\
&\qquad\qquad \text{(by definition of strongly proper composite multiclass loss)} \\
&\geq \mathbf{E}_X\Big[\frac{1}{2}\big\|\boldsymbol{\lambda}_{\mathrm{mlog}}^{-1}(\widehat{\widetilde{\mathbf{f}}}(X)) - \widetilde{\boldsymbol{\eta}}(X)\big\|_1^2\Big] \quad \text{(by Lemma 4.2)} \\
&= \frac{1}{2}\mathbf{E}_X\Big[\big\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\big\|_1^2\Big] \qquad (4.6)
\end{aligned}
$$

Combining Eqs. (4.5, 4.6), and applying Jensen's inequality (to the convex function $x \mapsto x^2$) establishes the result. □

4.6. Estimating the Noise Matrix $\mathbf{C}$

In settings where the noise matrix $\mathbf{C}$ is not known in advance, one may need to estimate $\mathbf{C}$ from the noisy training examples themselves. Most previous work on estimating the noise matrix assumes the existence of 'anchor points' (definition provided below), and relies on estimating these points accurately.[13] In particular, Menon et al. (2015) provided a method for estimating $\mathbf{C}$ using anchor points in the case of binary labels; Patrini et al. (2017) extended it to the multiclass setting, and later, Yao et al. (2020) proposed another noise estimation method also based on anchor points. Unfortunately, however, we show below that these methods do not work correctly for all noise matrices $\mathbf{C}$. In particular, in Section 4.6.1, we point out an error in the approach used to compute anchor points in the noise estimation methods of Patrini et al. and Yao et al., and provide sufficient and necessary conditions on $\mathbf{C}$ under which these methods do work correctly. Of course, when $\mathbf{C}$ is unknown, we may not know whether it satisfies these conditions, and so we may not be able to verify whether the estimation is correct. Building on the intuition developed from our analysis, in Section 4.6.2 we propose an iterative noise estimation heuristic that essentially tries to improve the estimation of anchor points, and that can be applied for any unknown $\mathbf{C}$; while it is not guaranteed

[13] There is also some recent work that aims to estimate $\mathbf{C}$ without identifying anchor points (Xia et al., 2019).

to converge or recover a correct estimate, in our experiments, it generally performs as well as, or improves upon, the methods of Patrini et al. and Yao et al. It remains an open question whether general noise matrices $\mathbf{C}$ can be estimated reliably using anchor points.

4.6.1. Conditions for Correctness of Noise Estimation Methods Based on Anchor Points

The methods of Patrini et al. (2017) and Yao et al. (2020) make the following assumption:

(A) (Anchor points) Under the clean distribution $D = (\mu, \boldsymbol{\eta})$, for every $y \in \mathcal{Y}$, there is a 'perfect' example $\bar{x}^y \in \mathcal{X}$ of class y (called an *anchor point* of class y) with marginal $\mu(\bar{x}^y) > 0$ and $\boldsymbol{\eta}(\bar{x}^y) = \mathbf{e}_y$.

Under this assumption, Patrini et al. observe that, for all $y, \widetilde{y} \in \mathcal{Y}$,

$$\widetilde{\eta}_{\widetilde{y}}(\bar{x}^y) = \left(\mathbf{C}^\top \boldsymbol{\eta}(\bar{x}^y)\right)_{\widetilde{y}} = \left(\mathbf{C}^\top \mathbf{e}_y\right)_{\widetilde{y}} = \gamma_{y,\widetilde{y}}\,.$$

Therefore, if one can identify such perfect examples/anchor points $\bar{x}^y$, and if the class probability estimates $\widehat{\widetilde{\boldsymbol{\eta}}}(x)$ are accurate, then one can estimate the noise rates via

$$\widehat{\gamma}_{y,\widetilde{y}} = \widehat{\widetilde{\eta}}_{\widetilde{y}}(\bar{x}^y) \quad \forall y, \widetilde{y} \in [n]\,.$$

As discussed previously, provided one has a sufficiently large training sample, accurate class probability estimates can be formed by minimizing a strongly proper composite surrogate over a suitably rich function class. The main step that is needed, therefore, is to identify the 'perfect' examples/anchor points $\bar{x}^y$ above.

Patrini et al. suggest identifying such anchor points by first estimating a CPE model $\widehat{\widetilde{\boldsymbol{\eta}}}$, and then taking a large collection of available instances $\mathcal{X}_{\text{train}} \subset \mathcal{X}$ drawn IID from the marginal μ (these could just be the training instances in $\widetilde{S}$ or could include other unlabeled instances as well), and estimating anchor points according to

$$\widehat{\bar{x}}^y \in \operatorname*{argmax}_{x \in \mathcal{X}_{\text{train}}} \widehat{\widetilde{\eta}}_y(x)$$

However, note that these anchor points should be chosen to maximize the *true* class probability $\eta_y(x)$, *not* the noisy class probability $\widetilde{\eta}_y(x)$! Therefore, the above method (also used by Yao et al., according to footnote 2 in their paper) is in general incorrect. Of course, we do have a relation between $\boldsymbol{\eta}$ and $\widetilde{\boldsymbol{\eta}}$ (Eq. (4.1)), but that relation involves $\mathbf{C}$; without knowledge of $\mathbf{C}$, we cannot in general use a noisy CPE model $\widehat{\widetilde{\boldsymbol{\eta}}}$ to find instances maximizing $\eta_y(x)$.

Nevertheless, surprisingly, Patrini et al. and Yao et al. did report some successful experiments with their methods. On investigating further, we identified a sufficient condition on the noise matrix $\mathbf{C}$ under which $\text{argmax}_x\, \widetilde{\eta}_y(x) = \text{argmax}_x\, \eta_y(x)$, and therefore, under which the above approach for estimating anchor points does work correctly, as well as a related necessary condition failing which the approach fails:

Theorem 4.6. *Suppose assumption (A) above holds.*

1. *If the noise matrix $\mathbf{C} = [\gamma_{y,\widetilde{y}}]$ satisfies the sufficient condition*

$$\gamma_{\widetilde{y},\widetilde{y}} \;>\; \gamma_{y,\widetilde{y}} \;\;\forall y \neq \widetilde{y}\,,$$

   then provided that $\mathcal{X}_{train}$ is a large enough sample (drawn IID from μ) and the noisy class probabilities $\widetilde{\boldsymbol{\eta}}(x)$ are modeled accurately, the anchor point estimation method of Patrini et al. (2017) described above works correctly.

2. *If $\mathbf{C}$ fails to satisfy the necessary condition*

$$\gamma_{\widetilde{y},\widetilde{y}} \;\geq\; \gamma_{y,\widetilde{y}} \;\;\forall y \neq \widetilde{y}\,,$$

   then the anchor point estimation method of Patrini et al. (2017) described above fails.

Proof. **Part 1 (Sufficiency).**

Suppose $\mathbf{C}$ satisfies the given sufficient condition, i.e. that

$$\gamma_{\widetilde{y},\widetilde{y}} > \gamma_{y,\widetilde{y}} \;\; \forall y \neq \widetilde{y}\,.$$

We will show that

$$\operatorname*{argmax}_x \eta_y(x) = \operatorname*{argmax}_x \widetilde{\eta}_y(x) \;\; \forall y \in [n]\,;$$

the claim will then follow.

Fix any class $y \in [n]$.

First, suppose $x' \in \operatorname{argmax}_x \eta_y(x)$. Then by assumption (A), it must be the case that $\eta_y(x') = 1$, i.e. that $\boldsymbol{\eta}(x') = \mathbf{e}_y$. This gives

$$\widetilde{\eta}_y(x') = (\mathbf{C}^\top \boldsymbol{\eta}(x'))_y = (\mathbf{C}^\top \mathbf{e}_y)_y = \gamma_{y,y}\,.$$

Now for any $x \in \mathcal{X}$, we have

$$\widetilde{\eta}_y(x) = (\mathbf{C}^\top \boldsymbol{\eta}(x))_y = \sum_{y'=1}^{n} \gamma_{y',y}\eta_{y'}(x) \leq \sum_{y'=1}^{n} \gamma_{y,y}\eta_{y'}(x) = \gamma_{y,y} = \widetilde{\eta}_y(x')\,.$$

Thus $x' \in \operatorname{argmax}_x \widetilde{\eta}_y(x)$. This establishes $\operatorname{argmax}_x \eta_y(x) \subseteq \operatorname{argmax}_x \widetilde{\eta}_y(x)$.

Conversely, suppose $x' \in \operatorname{argmax}_x \widetilde{\eta}_y(x) = \operatorname{argmax}_x (\mathbf{C}^\top \boldsymbol{\eta}(x))_y$. This means

$$\sum_{y'=1}^{n} \gamma_{y',y}\eta_{y'}(x') \geq \sum_{y'=1}^{n} \gamma_{y',y}\eta_{y'}(x) \quad \forall x \in \mathcal{X}\,.$$

By assumption (A), there exists $\bar{x}^y \in \mathcal{X}$ such that $\boldsymbol{\eta}(\bar{x}^y) = \mathbf{e}_y$. Applying the above inequality to $x = \bar{x}^y$, we have

$$\sum_{y'=1}^{n} \gamma_{y',y}\eta_{y'}(x') \geq \sum_{y'=1}^{n} \gamma_{y',y}\eta_{y'}(\bar{x}^y) = \gamma_{y,y}\,.$$

Moreover, we have

$$\sum_{y'=1}^{n} \gamma_{y',y}\eta_{y'}(x') \leq \gamma_{y,y} \,.$$

Combining the above two inequalities, we get

$$\sum_{y'=1}^{n} \gamma_{y',y}\eta_{y'}(x') = \gamma_{y,y} \,.$$

Since $\gamma_{y',y} < \gamma_{y,y}$ for all $y' \neq y$, this means we must have $\boldsymbol{\eta}(x') = \mathbf{e}_y$. Thus, $x' \in \operatorname{argmax}_x \eta_y(x)$. This establishes $\operatorname{argmax}_x \widetilde{\eta}_y(x) \subseteq \operatorname{argmax}_x \eta_y(x)$.

Part 2 (Necessity).

Suppose that $\mathbf{C}$ fails to satisfy the given necessary condition, i.e. that there exist $y \neq \widetilde{y}$ such that

$$\gamma_{\widetilde{y},\widetilde{y}} < \gamma_{y,\widetilde{y}} \,.$$

We will show that $\operatorname{argmax}_x \eta_{\widetilde{y}}(x) \neq \operatorname{argmax}_x \widetilde{\eta}_{\widetilde{y}}(x)$.

We give a proof by contradiction. In particular, let if possible $\operatorname{argmax}_x \eta_{\widetilde{y}}(x) = \operatorname{argmax}_x \widetilde{\eta}_{\widetilde{y}}(x) = \operatorname{argmax}_x (\mathbf{C}^\top \boldsymbol{\eta}(x))_{\widetilde{y}}$.

By assumption (A), there exists $\bar{x}^{\widetilde{y}} \in \mathcal{X}$ such that $\boldsymbol{\eta}(\bar{x}^{\widetilde{y}}) = \mathbf{e}_{\widetilde{y}}$, so this means $\bar{x}^{\widetilde{y}} \in \operatorname{argmax}_x \eta_{\widetilde{y}}(x) = \operatorname{argmax}_x \widetilde{\eta}_{\widetilde{y}}(x) = \operatorname{argmax}_x (\mathbf{C}^\top \boldsymbol{\eta}(x))_{\widetilde{y}}$. This means

$$\gamma_{\widetilde{y},\widetilde{y}} = \sum_{y'=1}^{n} \gamma_{y',\widetilde{y}}\eta_{y'}(\bar{x}^{\widetilde{y}}) \geq \sum_{y'=1}^{n} \gamma_{y',\widetilde{y}}\eta_{y'}(x) \quad \forall x \in \mathcal{X} \,.$$

But by assumption (A), we can also find $\bar{x}^y \in \mathcal{X}$ such that $\boldsymbol{\eta}(\bar{x}^y) = \mathbf{e}_y$. Applying the above inequality to $x = \bar{x}^y$ then gives

$$\gamma_{\widetilde{y},\widetilde{y}} \geq \sum_{y'=1}^{n} \gamma_{y',\widetilde{y}}\eta_{y'}(\bar{x}^y) = \gamma_{y,\widetilde{y}} \,,$$

contradicting our assumption. Therefore, we must have $\operatorname{argmax}_x \eta_{\widetilde{y}}(x) \neq \operatorname{argmax}_x \widetilde{\eta}_{\widetilde{y}}(x)$. $\square$

It is worth noting that the noise matrices in Patrini et al.'s study that were estimated correctly by their method all satisfy the sufficient condition above; for the one noise matrix in their study which did not satisfy the necessary condition above, their estimation method failed (see Section 4.7.2 for details). Similarly, all the noise matrices considered in Yao et al.'s study satisfy the sufficient condition above; in our experiments, for noise matrices that fail to satisfy the necessary condition above, Yao et al.'s method also fails.

We also note that, in Patrini et al.'s study, after learning a noisy CPE model $\widehat{\widetilde{\boldsymbol{\eta}}}$ and estimating $\mathbf{C}$, a different learning algorithm that minimizes a noise-corrected loss was then used to learn a classifier $\widehat{h}$. In our case, after learning $\widehat{\widetilde{\boldsymbol{\eta}}}$ and estimating $\mathbf{C}$, we can simply output the plug-in classifier $\widehat{h} = \text{plugin}^{\mathbf{L}}_{\mathbf{C}} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$, with no additional training required.

4.6.2. An Iterative Noise Estimation Heuristic

Based on the discussion above, we propose an alternative, iterative noise estimation heuristic that aims to improve anchor point estimation, wherein we start with an estimate of $\widehat{\mathbf{C}} = \mathbf{I}$ (no noise), and iteratively feed in the current estimate into a corrected version of Patrini et al.'s method to obtain an updated estimate. The approach is shown in Algorithm 4.3. The first iteration simply corresponds to Patrini et al.'s original method; therefore, if $\mathbf{C}$ satisfies the condition of Theorem 4.6, then the first iteration produces an accurate estimate. Unfortunately, the method is not guaranteed to converge or to produce an accurate estimate in general; nevertheless, in our experiments, we find this method performs as well as, or better than, the methods of Patrini et al. and Yao et al. It remains an open question whether general noise matrices $\mathbf{C}$ can be estimated reliably using anchor points.

4.7. Experiments

We conducted two sets of experiments to evaluate our noise-corrected plug-in algorithm. In the first set of experiments, we generated synthetic data, and tested the sample complexity behavior of our algorithm, using linear models, for a variety of different noise matrices $\mathbf{C}$ with increasing values of $\|\mathbf{C}^{-1}\|_2$. In the second set of experiments, we compared the performance of our noise correction method with those of van Rooyen and Williamson (2017) and Patrini et al. (2017), all

Algorithm 4.3 Iterative Noise Estimation Heuristic

```
1: Inputs:
   (1) CPE model $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ (for noisy distribution)
   (2) $\mathcal{X}_{\text{train}} \subset \mathcal{X}$
   (3) Maximum number of iterations $T$
2: Initialize: $\widehat{\mathbf{C}}^{(1)} = \big[\widehat{\gamma}^{(1)}_{y,\widetilde{y}}\big] \leftarrow \mathbf{I}$
3: For $t = 1, \ldots, T$:
4:     $\forall y \in \mathcal{Y}: \quad \widehat{x}^{y} \leftarrow \operatorname*{argmax}_{x \in \mathcal{X}_{\text{train}}} \Big( \big( (\widehat{\mathbf{C}}^{(t)})^\top \big)^{-1} \widehat{\widetilde{\boldsymbol{\eta}}}(x) \Big)_y$
5:     $\forall y, \widetilde{y} \in \mathcal{Y}: \quad \widehat{\gamma}^{(t+1)}_{y,\widetilde{y}} \leftarrow \widehat{\widetilde{\eta}}_{\widetilde{y}}(\widehat{x}^{y})$
6:     $\text{diff}^{(t)} \leftarrow \|\widehat{\mathbf{C}}^{(t+1)} - \widehat{\mathbf{C}}^{(t)}\|_F$
7: Output: $\widehat{\mathbf{C}}^{(t^*)}$, where $t^* = \operatorname*{argmin}_{t \in [T]} \text{diff}^{(t)}$
```

using neural network models, on two real benchmark data sets; in this set of experiments, we used noise matrices $\mathbf{C}$ constructed for these data sets by Patrini et al. (2017), closely following their experimental settings. We also compared the performance of our noise estimation method with those of Patrini et al. (2017) and Yao et al. (2020). In all cases, we used the multiclass logistic loss (unmodified in our case, and modified as needed by each of the other algorithms). We summarize both sets of experiments below. In all cases, training labels were flipped randomly according to the prescribed (invertible) noise matrix $\mathbf{C}$; performance of the learned models was then measured on a clean test set.

4.7.1. Synthetic Data: Sample Complexity Behavior

In order to test the sample complexity behavior of our algorithm, we generated synthetic data from a known distribution (from which we could draw increasingly large training samples as needed). Specifically, we constructed a 5-class problem over a 10-dimensional instance space $\mathcal{X} = [-1, 1]^{10}$ as follows. Instances $\mathbf{x}$ were generated uniformly at random from $\mathcal{X}$. The class probability function $\boldsymbol{\eta} : \mathcal{X} \to \Delta_5$ was set to $\eta_y(\mathbf{x}) = \frac{\exp(\mathbf{w}_y^\top \mathbf{x})}{\sum_{y'=1}^{5} \exp(\mathbf{w}_{y'}^\top \mathbf{x})}$ for some fixed weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_5 \in \mathbb{R}^{10}$ (the entries of the weight vectors were drawn IID from $\mathcal{N}(0, 1)$ and then scaled so that $\|\mathbf{w}_y\|_2 = 1$). Given an instance $\mathbf{x}$, a clean label y was drawn randomly according to $\boldsymbol{\eta}(\mathbf{x})$. For any prescribed (row-stochastic) noise matrix $\mathbf{C}$, training labels y were then stochastically flipped to a noisy label $\widetilde{y}$ according to the probabilities in the y-th row of $\mathbf{C}$.

We tested the sample complexity behavior of our algorithm, implemented to minimize the multiclass logistic loss over linear models, for a variety of noise matrices $\mathbf{C}$ with increasing values of $\|\mathbf{C}^{-1}\|_2$.[14][15] We ran the algorithm on increasingly large (noisy) training samples (up to 40,000 examples) and measured the performance on a large test set of 10,000 (clean) examples. The results are shown in Figure 4.3: The left plot in the figure shows results for the 0-1 loss (shown as accuracy); the right plot shows results for a different target loss, specifically, the ordinal regression loss $\mathbf{L}^{\text{ord}}$ defined as $\ell^{\text{ord}}_{y,\widehat{y}} = |\widehat{y} - y|$.[16] We see that, as suggested by our regret transfer bound, as $\|\mathbf{C}^{-1}\|_2$ increases (i.e. as the matrix $\mathbf{C}$ becomes closer to being singular), the sample size required to achieve a given level of performance generally increases.[17]

4.7.2. Real Data: Comparison with Other Algorithms

We conducted experiments on several real data sets. Here we describe experiments on two benchmark data sets, MNIST (Lecun et al., 1998) and CIFAR10 (Krizhevsky and Hinton, 2009), where we compared our algorithm with the *unbiased estimator* method of van Rooyen and Williamson (2017) and the *forward* method of Patrini et al. (2017), all using neural network models, and also tested the incorporation of noise estimation methods. These experiments were designed to closely mimic experiments of Patrini et al. (2017); we used code provided by the authors[18] and kept the neural network architectures and all parameters as given.[19]

[14]Although in Section 4.5.3, we provided an improve regret transfer bound for the multiclass logistic loss that depends on the noise matrix $\mathbf{C}$ through $\|(\mathbf{C}^\top)^{-1}\|_1$ instead of $\|\mathbf{C}^{-1}\|_2$, here we are only comparing the relative magnitude, and both norms can serve this purpose. The detailed descriptions here correspond to our conference paper (Zhang et al., 2021) on which this chapter is based.

[15]The implementation was in PyTorch (Paszke et al., 2019), and used the AdamW optimizer. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was halved at the end of every 5 epochs.

[16]Following Natarajan et al. (2013), for each noise matrix, we repeated each experiment 3 times with independent random corruptions of the training set using the same noise matrix; our results give the mean performance over the 3 runs.

[17]We note that for the synthetic data distribution described above, although the clean class probabilities $\boldsymbol{\eta}(\mathbf{x})$ take the form of a softmax-of-linear model, the noisy class probabilities $\widetilde{\boldsymbol{\eta}}(\mathbf{x})$ are not of this form. Therefore, even though the plots in Figure 4.3 seem to suggest our algorithm converges to the Bayes optimal performance, strictly speaking, this is not the case: The algorithm does appear to have learned a fairly accurate model for the noisy class probabilities $\widetilde{\boldsymbol{\eta}}(\mathbf{x})$, but it cannot express them exactly; in order to truly model them exactly, we would need to implement the algorithm using a richer function class. (We do not do this here since the difference in performance would be unnoticeable. We use richer function classes in the experiments with real data, where we employ neural network models.)

[18]https://github.com/giorgiop/loss-correction

[19]We note that for some parameters (e.g. batch size), there is a discrepancy between the settings used in the code and those mentioned in the paper; we used the settings in the code.

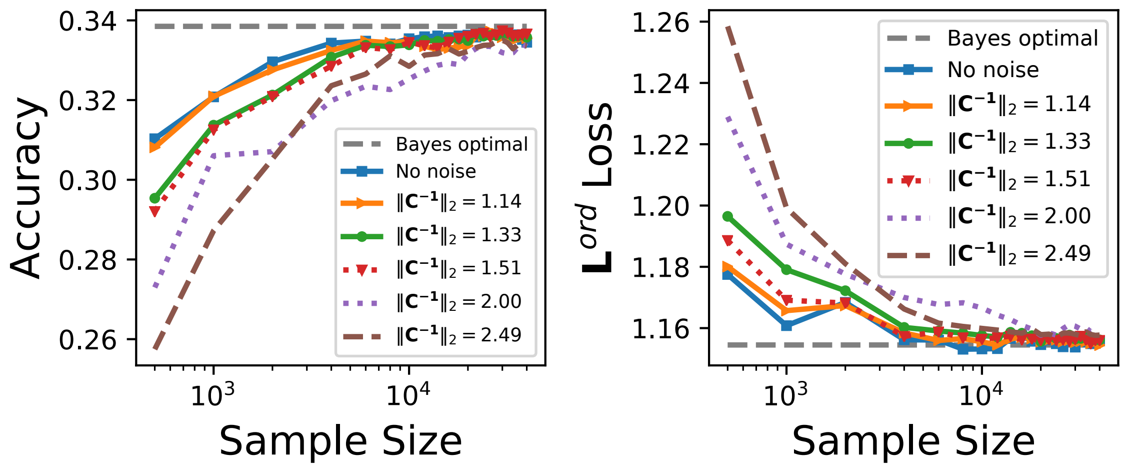

Figure 4.3: Sample complexity behavior of our algorithm on synthetic 5-class data for a variety of noise matrices $\mathbf{C}$ with increasing values of $\|\mathbf{C}^{-1}\|_2$. **Left:** 0-1 loss (shown as accuracy). **Right:** Ordinal regression loss $\mathbf{L}^{\text{ord}}$. As suggested by our regret bounds, as $\|\mathbf{C}^{-1}\|_2$ increases, the sample size needed to reach a given level of performance generally increases. See Section 4.7.1 for details.

Table 4.1: Details of MNIST and CIFAR10 data sets.

Data set	**# train**	**# test**	**# classes** (n)	**# features** (d)
MNIST	60,000	10,000	10	784
CIFAR10	50,000	10,000	10	3072

Both MNIST and CIFAR10 are 10-class data sets (see Table 4.1 for details of the data sets). The experiments used 0-1 loss (measured as accuracy). In both cases, experiments were conducted with clean data (no noise) and with 6 noise matrices $\mathbf{C}$. One of these, $\mathbf{C}^{\text{sym}(0.2)}$, was a symmetric noise matrix with the following structure: all diagonal entries γ_{yy} were set to $1-\gamma$, where $\gamma = 0.2$; all off-diagonal entries were set to $\frac{\gamma}{n-1}$ (here $n = 10$). For such symmetric noise matrices (with $\gamma < \frac{n-1}{n}$) and 0-1 loss, it is known that no noise correction is needed, and that standard algorithms designed to learn a good classifier for 0-1 loss (on the noisy data) work correctly (van Rooyen and Williamson, 2017; Ghosh et al., 2017). The other 5 noise matrices were asymmetric, and were artificially designed by Patrini et al. (2017) to simulate some of the possible structures of real label noise, where a label might be replaced with some probability γ by some other similar label, for example, `Cat` $\rightarrow$ `Dog`. For each of MNIST and CIFAR10, Patrini et al. specified a set of such 'label noise' transitions to create specific parametric noise matrices $\mathbf{C}^{\text{MNIST}(\gamma)}$ and $\mathbf{C}^{\text{CIFAR10}(\gamma)}$, and instantiated these with $\gamma = 0.2, 0.6$; we additionally included $\gamma = 0.45, 0.55, 0.65$. The noise matrices $\mathbf{C}^{\mathrm{sym}(0.2)}$ and $\mathbf{C}^{\mathrm{MNIST}(\gamma)}$, $\mathbf{C}^{\mathrm{CIFAR10}(\gamma)}$ for $\gamma < 0.5$ all satisfy the sufficient condition of Theorem 4.6; the matrices $\mathbf{C}^{\mathrm{MNIST}(\gamma)}$, $\mathbf{C}^{\mathrm{CIFAR10}(\gamma)}$ for $\gamma > 0.5$ fail to satisfy the necessary condition.

For MNIST, the asymmetric noise matrix $\mathbf{C}^{\mathrm{MNIST}(\gamma)}$ includes the following label noise transitions: $2 \to 7$, $3 \to 8$, $5 \leftrightarrow 6$, $7 \to 1$. Following Patrini et al. (2017), features were normalized to $[0, 1]$, and two fully connected hidden layers of size 128 were trained, with ReLU activation and dropout rate 0.2.[20]

For CIFAR10, the asymmetric noise matrix $\mathbf{C}^{\mathrm{CIFAR10}(\gamma)}$ includes the following label noise transitions: `Truck` $\to$ `Automobile`, `Bird` $\to$ `Airplane`, `Deer` $\to$ `Horse`, `Cat` $\leftrightarrow$ `Dog`. Again following Patrini et al. (2017), per-pixel mean subtraction and data augmentation were performed, and a 14-layer residual network (ResNet) (He et al., 2016) was trained.[21]

The results are summarized in Tables 4.2 and 4.3, respectively. For each algorithm, we implemented four versions: one with the noise matrix $\mathbf{C}$ known, and the other three with the noise matrix estimated using either the method of Patrini et al. (2017) (denoted $\widehat{\mathbf{C}}^{\mathrm{Patrini}}$), the Dual T method of Yao et al. (2020) ($\widehat{\mathbf{C}}^{\mathrm{DT}}$), or our iterative noise estimation heuristic ($\widehat{\mathbf{C}}^{\mathrm{iter}}$).[22] Several observations are in order. First, for the symmetric noise matrix $\mathbf{C}^{\mathrm{sym}(0.2)}$, standard logistic regression with no noise correction does well as expected; for heavy asymmetric noise ($\mathbf{C}^{\mathrm{MNIST}(\gamma)}$ and $\mathbf{C}^{\mathrm{CIFAR10}(\gamma)}$ for $\gamma > 0.5$), standard logistic regression without noise correction does not do well. Second, our noise-corrected plug-in method is comparable to the other noise-corrected methods, even though it requires no change to the training process. Third, our iterative noise estimation heuristic either performs similarly to the noise estimation methods of Patrini et al. and Yao et al., or in some cases (particularly $\mathbf{C}^{\mathrm{MNIST}(\gamma)}$ for $\gamma > 0.5$) significantly outperforms their methods. Finally, for the noise matrices that satisfy the sufficient condition of Theorem 4.6, all three noise estimation methods perform well; for the noise matrices that fail to satisfy the necessary condition, no method achieves

[20] Batch size was 32. AdaGrad (Duchi et al., 2010) was run for 40 epochs with default parameters.

[21] Batch size was 32. SGD was run for 120 epochs with momentum 0.9 and learning rate set to 0.1 initially and divided by 10 after 40 and 80 epochs; weight decay was 10^{-4}.

[22] Our iterative noise estimation heuristic was implemented with maximum number of iterations T set to 1000.

perfect estimation.

It is worth pointing out again that all three noise estimation methods (the methods of Patrini et al. (2017) and Yao et al. (2020), and our iterative method) make use of a noisy CPE model $\widehat{\widetilde{\boldsymbol{\eta}}}(x)$ learned from the noisy training data. Our noise-corrected plug-in algorithm makes use of this noisy CPE model directly, simply applying a noise-corrected plug-in step at prediction time, and does not need any further re-training; on the other hand, the other two noise correction methods above both need to further minimize a noise-corrected loss on the noisy data in order to learn a classifier.

4.8. Conclusion

We have provided a simple noise-corrected plug-in method for general multiclass class-conditional label noise (CCN) that requires no change to the training process. Noise correction takes place at prediction time, and after a one-time matrix inversion and multiplication step, requires $O(n^2)$ time per prediction, where n is the number of classes. For general loss matrices $\mathbf{L}$, this is the same computational cost that is needed for standard plug-in methods; for the 0-1 loss, it is a factor of n larger than the standard cost (for small to moderate n, this may still be a smaller cost overall as compared to the cost of modifying the training process). We have also provided quantitative regret transfer bounds for our method that quantify the effect of learning from noisy labels, as well as an iterative noise estimation heuristic.

One possible issue to be careful about is that accurate estimation of noisy class probabilities can potentially be challenging due to their typically higher variance (particularly with neural networks, which often exhibit high calibration errors (Guo et al., 2017; Rahimi et al., 2020)) – while we did not find this to be a significant concern in our experiments, it could possibly be an issue for certain types of data sets or noise. It remains an open question whether general noise matrices $\mathbf{C}$ can be estimated reliably using anchor points.

Table 4.2: Test accuracy (percentage) on MNIST data, shown as the mean (with standard error of the mean in parentheses) over 5 random trials. In each column, the best algorithm(s) using the known noise matrix **C** and the best algorithm(s) using each of the 3 noise estimation methods (Patrini et al., Dual T, and our iterative heuristic) are shown in bold font; among the latter, the best algorithm + noise estimation combination overall is further enclosed in asterisks. See Section 4.7.2 for details.

Algorithm	**No noise**	$\mathbf{C}^{\text{sym}(0.2)}$	$\mathbf{C}^{\text{MNIST}(0.2)}$	$\mathbf{C}^{\text{MNIST}(0.45)}$	$\mathbf{C}^{\text{MNIST}(0.55)}$	$\mathbf{C}^{\text{MNIST}(0.6)}$	$\mathbf{C}^{\text{MNIST}(0.65)}$
Logistic	**92.84** (0.14)	**92.00** (0.07)	91.58 (0.08)	80.80 (0.30)	58.27 (0.27)	52.08 (0.23)	49.86 (0.05)
Unbiased, **C**	92.76 (0.02)	91.98 (0.11)	**92.24** (0.08)	**89.75** (0.30)	**89.54** (0.11)	**90.72** (0.06)	**90.68** (0.08)
Forward, **C**	**92.84** (0.06)	91.69 (0.08)	92.00 (0.09)	84.52 (1.48)	82.14 (2.48)	87.47 (1.43)	89.57 (0.92)
Plug-in, **C**	**92.84** (0.14)	**92.00** (0.08)	92.05 (0.10)	87.57 (0.46)	87.70 (0.18)	89.31 (0.19)	89.23 (0.06)
Unbiased, $\widehat{\mathbf{C}}^{\text{Patrini}}$	92.63 (0.05)	91.45 (0.05)	91.94 (0.02)	**88.50** (0.28)	68.50 (3.41)	66.91 (2.11)	69.45 (0.94)
Forward, $\widehat{\mathbf{C}}^{\text{Patrini}}$	92.68 (0.10)	91.75 (0.05)	***92.20*** (0.09)	83.44 (1.90)	71.57 (2.18)	68.14 (3.01)	61.58 (2.22)
Plug-in, $\widehat{\mathbf{C}}^{\text{Patrini}}$	**92.84** (0.13)	**92.01** (0.01)	91.97 (0.09)	87.01 (0.54)	**73.17** (1.37)	**70.37** (1.09)	**69.64** (0.97)
Unbiased, $\widehat{\mathbf{C}}^{\text{DT}}$	92.16 (0.04)	91.29 (0.07)	91.50 (0.09)	85.79 (0.59)	60.30 (1.70)	53.72 (1.10)	52.21 (4.55)
Forward, $\widehat{\mathbf{C}}^{\text{DT}}$	92.37 (0.07)	91.46 (0.06)	91.80 (0.07)	80.77 (2.37)	62.84 (3.77)	**60.79** (2.94)	**58.15** (2.28)
Plug-in, $\widehat{\mathbf{C}}^{\text{DT}}$	**92.84** (0.13)	**91.92** (0.04)	**92.04** (0.08)	**86.51** (0.60)	**66.07** (2.62)	58.19 (0.43)	57.99 (3.13)
Unbiased, $\widehat{\mathbf{C}}^{\text{iter}}$	92.73 (0.07)	91.76 (0.08)	92.00 (0.07)	***89.51*** (0.19)	74.95 (0.22)	71.24 (3.52)	70.98 (3.44)
Forward, $\widehat{\mathbf{C}}^{\text{iter}}$	92.70 (0.07)	91.60 (0.17)	**92.16** (0.04)	85.22 (2.64)	***81.36*** (1.26)	***73.52*** (4.57)	***74.42*** (4.35)
Plug-in, $\widehat{\mathbf{C}}^{\text{iter}}$	**92.85** (0.13)	***92.03*** (0.04)	91.96 (0.09)	87.55 (0.46)	76.96 (0.13)	71.65 (3.26)	70.68 (3.19)

Table 4.3: Test accuracy (percentage) on CIFAR10 data, shown as the mean (with standard error of the mean in parentheses) over 5 random trials. In each column, the best algorithm(s) using the known noise matrix $\mathbf{C}$ and the best algorithm(s) using each of the 3 noise estimation methods (Patrini et al., Dual T, and our iterative heuristic) are shown in bold font; among the latter, the best algorithm + noise estimation combination overall is further enclosed in asterisks. See Section 4.7.2 for details.

Algorithm	**No noise**	$\mathbf{C}^{\text{sym}(0.2)}$	$\mathbf{C}^{\text{CIFAR10}(0.2)}$	$\mathbf{C}^{\text{CIFAR10}(0.45)}$	$\mathbf{C}^{\text{CIFAR10}(0.55)}$	$\mathbf{C}^{\text{CIFAR10}(0.6)}$	$\mathbf{C}^{\text{CIFAR10}(0.65)}$
Logistic	**89.76** (0.07)	85.26 (0.16)	86.95 (0.12)	76.56 (0.31)	63.78 (0.58)	58.1 (0.39)	54.79 (0.31)
Unbiased, $\mathbf{C}$	89.6 (0.08)	81.82 (0.27)	84.1 (0.14)	61.91 (2.32)	57.43 (3.81)	70.12 (2.43)	74.91 (1.99)
Forward, $\mathbf{C}$	89.6 (0.14)	**86.62** (0.13)	**88.7** (0.16)	**85.71** (0.2)	**85.12** (0.11)	**86.99** (0.03)	**87.12** (0.14)
Plug-in, $\mathbf{C}$	**89.76** (0.07)	85.26 (0.16)	87.46 (0.09)	82.71 (0.26)	81.8 (0.24)	83.5 (0.12)	84.01 (0.16)
Unbiased, $\widehat{\mathbf{C}}^{\text{Patrini}}$	89.54 (0.16)	82.27 (0.14)	86.08 (0.42)	69.45 (1.75)	67.15 (1.97)	69.77 (0.58)	69.94 (0.45)
Forward, $\widehat{\mathbf{C}}^{\text{Patrini}}$	89.66 (0.05)	**86.05** (0.23)	**88.11** (0.06)	**84.67** (0.48)	***78.78*** (1.14)	75.47 (0.36)	***74.48*** (0.48)
Plug-in, $\widehat{\mathbf{C}}^{\text{Patrini}}$	**89.76** (0.07)	85.29 (0.19)	87.39 (0.08)	82.46 (0.24)	78.02 (0.24)	**75.48** (0.19)	74.26 (0.3)
Unbiased, $\widehat{\mathbf{C}}^{\text{DT}}$	89.3 (0.16)	82.7 (0.12)	83.43 (0.23)	67.67 (2.05)	63.38 (0.54)	63.4 (1.05)	62.64 (1.16)
Forward, $\widehat{\mathbf{C}}^{\text{DT}}$	89.75 (0.21)	***86.93*** (0.08)	***88.32*** (0.1)	***84.72*** (0.42)	**75.98** (0.48)	69.85 (1.68)	61.89 (0.6)
Plug-in, $\widehat{\mathbf{C}}^{\text{DT}}$	**89.77** (0.07)	85.14 (0.19)	87.39 (0.1)	81.91 (0.32)	75.6 (0.29)	**71.21** (0.34)	**68.04** (0.7)
Unbiased, $\widehat{\mathbf{C}}^{\text{iter}}$	89.67 (0.06)	82.09 (0.14)	86.0 (0.19)	68.75 (2.68)	65.85 (1.23)	68.77 (1.1)	67.83 (1.65)
Forward, $\widehat{\mathbf{C}}^{\text{iter}}$	89.5 (0.07)	**85.92** (0.2)	**88.17** (0.03)	**84.09** (0.71)	77.64 (0.73)	***75.71*** (0.32)	**74.46** (0.59)
Plug-in, $\widehat{\mathbf{C}}^{\text{iter}}$	**89.76** (0.07)	85.27 (0.2)	87.4 (0.08)	82.41 (0.22)	**77.94** (0.19)	75.49 (0.17)	74.27 (0.29)

CHAPTER 5

COMPLEX LEARNING SETTING: MULTICLASS LEARNING FROM NOISY LABELS USING WEIGHTED LOSSES

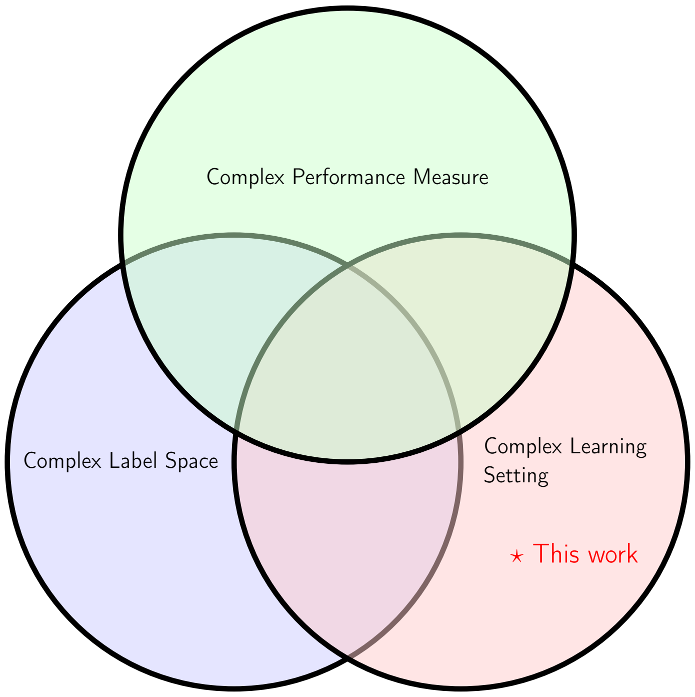

Figure 5.1: Position of *Multiclass Learning from Noisy Labels Using Weighted Losses* in the thesis.

This work was done in collaboration with Sheng Gao and Hua Wang, who were both Ph.D. students at Penn Wharton at the time of this project, under the supervision of Professor Shivani Agarwal. The three student authors contributed equally to this project. In particular, I participated in deriving theoretical results, explicitly showed that the multiclass weighted loss method recovers the binary weighted loss method as a special case, and connected the multiclass weighted loss method to some existing work in multiclass learning with a reject option.

In this chapter, we continue our discussion of multiclass learning from noisy labels. We show how to generalize the weighted loss method from binary to multiclass, and how to use the method to design consistent algorithms for problems in multiclass learning from noisy labels and multiclass learning with a reject option.

5.1. Introduction

5.1.1. Background and Our Contributions

In a famous study of learning from noisy labels under CCN in binary classification, Natarajan et al. (2013) proposed two methods: the method of *unbiased estimator* and the method of *weighted loss* (termed as method of label-dependent costs in their paper). Later, the unbiased estimator approach was extended to multiclass classification (van Rooyen and Williamson, 2017) to correct multiclass label noise. Meanwhile, Patrini et al. (2017) proposed the *forward* and *backward* approaches for learning from noisy labels under CCN in multiclass classification; it turns out the backward method is equivalent to the unbiased estimator method. The weighted loss method of Natarajan et al. (2013), which essentially minimizes a weighted surrogate loss where the weights are designed to correct label noise, however, has not yet been extended to multiclass classification; indeed, even in the standard (non-noisy) multiclass setting, it is not well-understood how to design suitable weighted surrogate losses for general cost-sensitive learning. We note that all the noise-correction methods above involve modifying the surrogate loss used in the training process. In Chapter 4, we have described our published work (Zhang et al., 2021) in which we have proposed a simple plug-in method to learn from noisy labels that does not modify the surrogate loss. It has some connections with the weighted loss method of Natarajan et al. (2013). Still, it remains unclear how to generalize the weighted loss method to multiclass classification for correcting label noise. Figure 5.2 depicts the relationships between these works and this work.

The weighted loss method for correcting label noise in binary classification not only works for smooth surrogate losses, but also works for the margin-based surrogate losses. In addition, it preserves convexity if the underlying surrogate loss is convex. This is in contrast to the method of unbiased estimator and the forward method: even if the underlying surrogate losses are convex, they may not preserve the convexity. For the method of unbiased estimator in binary classification, Natarajan et al. (2013) provided a simple condition to ensure convexity. When extending this method to multiclass classification, van Rooyen and Williamson (2017) generalized the condition in Natarajan et al. (2013) and showed that a large and useful class of loss functions remains convex

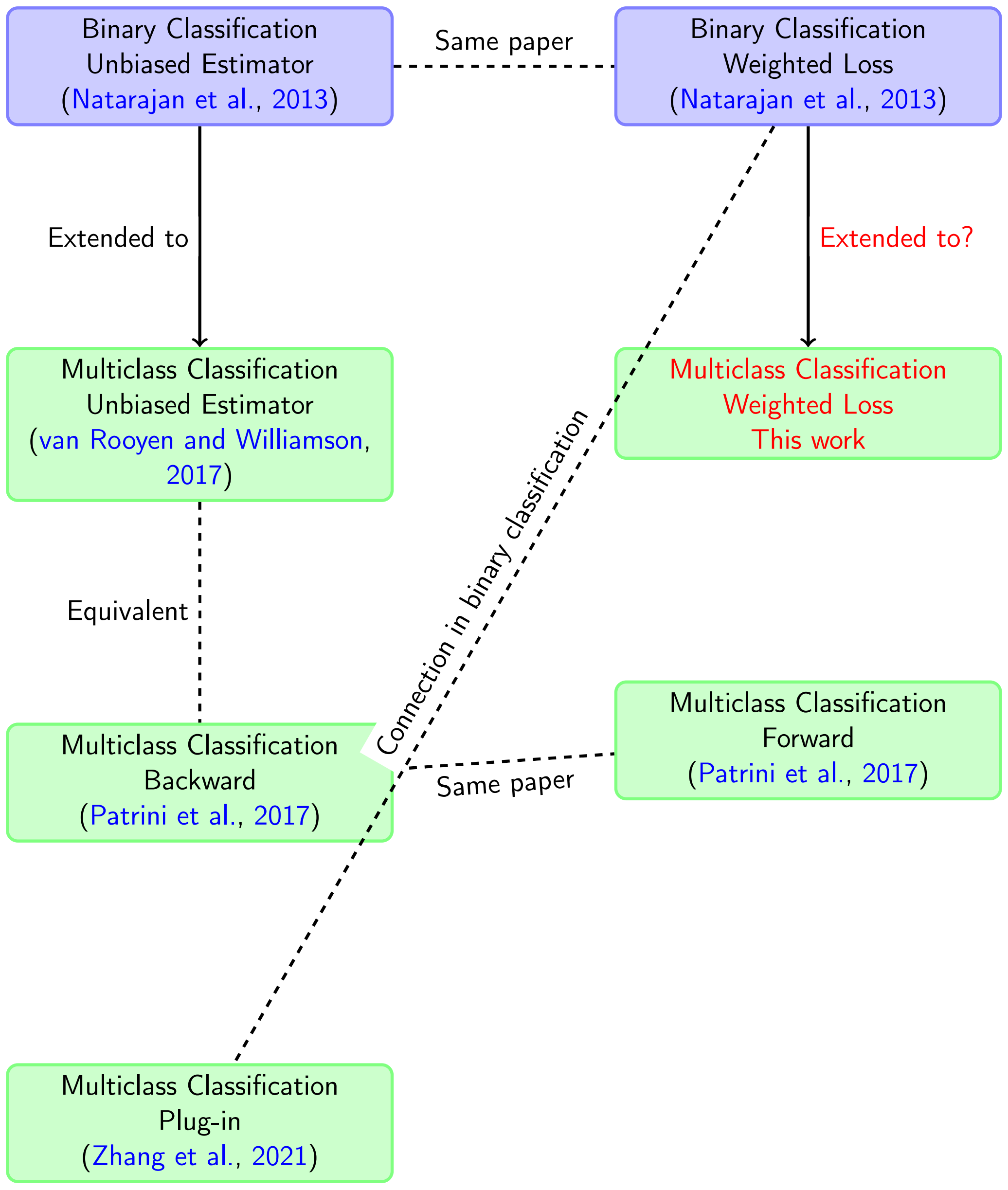

Figure 5.2: Relationships between existing works on learning from noisy labels under CCN in the literature and this work (highlighted in red).

after noise-correction. For the forward method, Patrini et al. (2017) did not discuss convexity, and the question is not trivial. The convexity of the forward corrected loss remains unclear.

Therefore, if it is possible to extend the weighted loss method to multiclass classification for correcting label noise, it is expected that the multiclass versions of the weighted loss should also work for the multiclass versions of margin-based losses (such as Crammer-Singer SVM (Crammer and Singer, 2001) and Weston-Watkins SVM (Weston and Watkins, 1999)), and preserve convexity when the underlying losses are convex.

The prevalence of multiclass problems across numerous domains highlights the importance of investigating the generalization of the weighted loss method to multiclass classification for correcting label noise. By achieving this generalization, it would be possible to develop a new family of algorithms specifically designed for learning from noisy labels in multiclass settings. This would not only provide practitioners with a broader range of tools to tackle noisy label problems, but also potentially improve the performance and robustness of existing methods.

In addition to the potential for creating novel noise-corrected algorithms, generalizing the weighted loss method to multiclass classification for correcting label noise could also lead to advancements in the broader field of multiclass classification. In particular, the weighted loss method of Natarajan et al. (2013) essentially involves techniques used for solving cost-sensitive learning problems in the standard (non-noisy) binary setting, and those problems have been well-studied (Scott, 2012). However, it remains unclear how to design suitable weighted surrogate losses for general cost-sensitive learning in the standard (non-noisy) multiclass setting, and how to design weights to correct label noise in the presence of noisy labels.

In this work, we close these open questions. Specifically, we first show how to design a surrogate loss for a general multiclass loss $\mathbf{L}$ by taking a weighted combination of surrogate losses for the standard 0-1 loss. Our method works with both smooth surrogate losses and non-smooth surrogate losses, allowing for margin-based losses such as those used in various formulations of multiclass support vector type algorithms. In addition, the proposed method preserves the convexity of the underlying surrogate loss, a desirable property to allow for efficient optimization. We also provide theoretical results to show the proposed method is Bayes consistent, and provide an estimation error bound. Then, we apply the proposed method to extend the weighted loss method proposed in Natarajan et al. (2013) for binary learning from noisy labels to multiclass learning from noisy labels; we achieve this by choosing appropriate weights designed to correct label noise. Finally, we also apply the proposed method to solve problems in multiclass learning with a reject option; in doing so, we recover several results of Cao et al. (2022). Figure 5.3 depicts our contributions.

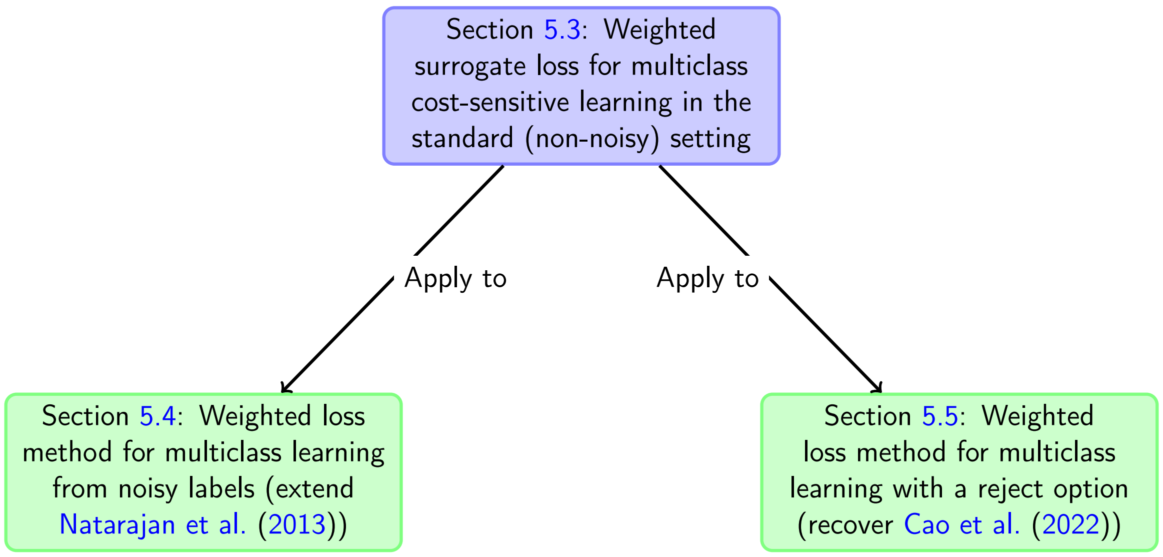
Section 5.3: Weighted surrogate loss for multiclass cost-sensitive learning in the standard (non-noisy) setting
Apply to
Apply to
Section 5.4: Weighted loss method for multiclass learning from noisy labels (extend Natarajan et al. (2013))
Section 5.5: Weighted loss method for multiclass learning with a reject option (recover Cao et al. (2022))

Figure 5.3: Our contributions.

Methodology. We study the problem by utilizing current studies on learning from noisy labels and the theory of convex calibrated surrogates. Convex calibrated surrogates are a class of surrogate loss functions used in machine learning, particularly in classification problems. They are designed to approximate the target performance measures (typically discrete losses) while retaining desirable properties such as convexity and calibration. The notion of calibration ensures that minimizing the surrogate loss can (in the limit of a sufficiently large training sample) recover a Bayes optimal model for the target discrete loss. In the context of learning from noisy labels, we want to ensure that minimizing the surrogate loss over the *noisy* training sample can (in the limit of a sufficiently large training sample) recover a Bayes optimal model for the target discrete loss under the *clean* distribution. In particular, we build solutions based on the following works.

- Natarajan et al. (2013); van Rooyen and Williamson (2017); Patrini et al. (2017); Zhang et al. (2021) proposed and analyzed different consistent algorithms to learn from noisy labels under the CCN model in both binary and multiclass settings.
- (Zhang, 2004a,b; Bartlett et al., 2006; Tewari and Bartlett, 2007; Steinwart, 2007; Ramaswamy et al., 2014; Ramaswamy and Agarwal, 2016) studied calibrated surrogate losses in binary and multiclass classification.
- Other works that utilized the theory of convex calibration surrogates to design algorithms for various binary/multiclass learning settings (other than learning from noisy labels) are also helpful. For example, Ramaswamy et al. (2018); Cao et al. (2022) studied *classification with rejection* settings in which a classifier refrains from making a prediction to avoid critical misclassification when encountering examples that are difficult to classify.

5.1.2. Notation

For an integer n, we denote by $[n]$ the set of integers $\{1, ..., n\}$, and by Δ_n the probability simplex $\{\mathbf{p} \in \mathbb{R}_+^n : \sum_{y=1}^n p_y = 1\}$. For a vector $\mathbf{a}$, we denote by $\|\mathbf{a}\|_2$ the L_2 norm of $\mathbf{a}$. For a matrix $\mathbf{A}$, we denote by $\|\mathbf{A}\|_F$ the Frobenius norm of $\mathbf{A}$, by $\|\mathbf{A}\|_p$ the induced p-norm of $\mathbf{A}$ ($\|\mathbf{A}\|_2$ is the largest singular value of $\mathbf{A}$), and by $\mathbf{a}_y$ the y-th column vector of $\mathbf{A}$. We use $\mathbf{e}_y$ to denote a standard basis vector with y-th element 1. $\mathbf{1}(\cdot)$ is the indicator function.

5.1.3. Related Work

Noise models in learning from noisy labels. In learning from noisy labels, several noise models have been proposed and studied. In *random classification noise* (RCN) model, each label is flipped with a fixed probability $\rho \in [0, \frac{1}{2})$ (Angluin and Laird, 1987; Bylander, 1994; Kearns, 1998; Cesa-Bianchi et al., 1999; van Rooyen et al., 2015). A more general noise model is *class-conditional noise* (CCN), which says noisy labels are generated according to a fixed conditional distribution given the true class (Natarajan et al., 2013; Scott et al., 2013; Menon et al., 2015; Liu and Tao, 2016; van Rooyen and Williamson, 2017; Patrini et al., 2017). However, both RCN and CCN depend only on the labels. The most general label noise, *instance-dependent and label-dependent noise*

(ILN), also depends on the instance (Menon et al., 2018; Cheng et al., 2020). In multi-label learning from noisy labels, *independent flipping noise* (IFN) model is commonly used, in which each label (tag) is independently flipped from active to non-active (or vice versa) with some probability (Kumar et al., 2020; Zhao and Gomes, 2021; Xie and Huang, 2023). Below we briefly discuss some developments in these fields and focus on works that are the most related to our study. For detailed surveys about learning from noisy labels, we refer the reader to Frénay and Verleysen (2014); Song et al. (2023); Han et al. (2020).

Binary learning from noisy labels. The initial studies focused on the RCN model and PAC-style guarantees (Angluin and Laird, 1987; Bylander, 1994; Aslam and Decatur, 1996; Kearns, 1998; Blum and Mitchell, 1998; Cesa-Bianchi et al., 1999). Some recent studies concerned about designing surrogate losses robust to RCN (Long and Servedio, 2010; van Rooyen et al., 2015; Ghosh et al., 2015). It was also mentioned in Menon et al. (2015) that for RCN, the noise rate is not needed for consistent predictions.

For CCN, Natarajan et al. (2013) and Menon et al. (2015) are the most related studies to ours and they assumed CCN is known. Natarajan et al. (2013) showed two ways of correcting surrogate losses by the noise rates so that minimizing the modified surrogates with noisy labels is consistent w.r.t. the true distribution. Menon et al. (2015) proposed to learn class probability estimation (CPE) models from noisy labels and then to apply a threshold depending on the noise rates. Other methods dealing with CCN include Stempfel and Ralaivola (2009); Scott et al. (2013); Scott (2015); Liu and Tao (2016); Patrini et al. (2016); Liu and Guo (2020). There are also results when noise rates are not known. Scott et al. (2013); Scott (2015); Menon et al. (2015); Liu and Tao (2016) proposed consistent estimators for noise rates. Liu and Guo (2020) used peer loss fcuntions.

For ILN, Menon et al. (2018) studied consistency properties with instance-dependent (but label-independent) noise, and a subclass of general ILN models which they termed as boundary consistent noise model. Cheng et al. (2020) studied bounded ILN models and proposed to use 'distilled' examples to learn from such noise.

Multiclass learning from noisy labels. Symmetric CCN here is the multiclass version of RCN in binary classification. Ghosh et al. (2017) proved a sufficient condition for a loss function to be robust to symmetric CCN. Wang et al. (2018) proposed an importance re-weighting method for symmetric CCN.

In an elegant study, van Rooyen and Williamson (2017) studied in detail learning from known CCN for multiclass problems. They generalized the unbiased estimator method in Natarajan et al. (2013) to correct surrogate losses using noise rates in the multiclass setting and provided upper and lower risk bounds. They also studied loss functions that are invariant to CCN and showed a method to construct such losses. Patrini et al. (2017) proposed two ways of modifying losses by the known CCN: forward and backward, and they showed the minimizer of the modified loss under the noisy distribution coincide with the minimizer of the original loss under the clean distribution. They also extended results in Menon et al. (2015) to estimate the noise when it is unknown. Zhang et al. (2021) proposed a simple plug-in method to learn from noisy labels that does not modify the surrogate loss.

Convex calibrated surrogates. Convex surrogate losses are frequently used in machine learning to design computationally efficient learning algorithms. The notion of calibrated surrogate losses, which ensures that minimizing the surrogate loss can (in the limit of sufficient data) recover a Bayes optimal model for the target discrete loss, was initially studied in the context of binary classification (Bartlett et al., 2006; Zhang, 2004a) and multiclass 0-1 classification (Zhang, 2004b; Tewari and Bartlett, 2007). In recent years, calibrated surrogates have been designed for several more complex learning problems, including general multiclass problems and certain types of subset ranking and multi-label problems (Steinwart, 2007; Duchi et al., 2010; Gao and Zhou, 2013; Ramaswamy et al., 2013, 2014, 2015).

Classification with a reject option. In some binary and multiclass learning settings, a classifier is required to refrain from making a prediction to avoid costly misclassification errors when encountering examples that are difficult to classify (i.e., when the confidence of classification of an example is low). Such settings are termed 'classification with a reject option', and have been studied in standard binary and multiclass classification problems (Chow, 1970; Yuan and Wegkamp, 2010; El-Yaniv and Wiener, 2010; Bartlett and Wegkamp, 2008; Cortes et al., 2016a,b; Geifman and El-Yaniv, 2017; Ramaswamy et al., 2018; Ni et al., 2019; Shen et al., 2020; Charoenphakdee et al., 2021; Cao et al., 2022).

5.1.4. Organization

Section 5.2 gives preliminaries and background. Section 5.3 describes our weighted surrogate loss for multiclass cost-sensitive learning settings. Section 5.4 applies our weighted surrogate loss to multiclass learning from noisy labels, extending the weighted loss method proposed in Natarajan et al. (2013) for binary learning from noisy labels to multiclass learning from noisy labels. Section 5.5 applies our weighted surrogate loss to multiclass learning with a reject option, recovering several results of Cao et al. (2022). Section 5.6 concludes this work.

5.2. Preliminaries and Background

5.2.1. Problem Setup: Multiclass Learning from Noisy Labels

We start with some necessary notations and definitions.

The problem of (multiclass) learning from noisy labels can be described as follows. There is an instance space $\mathcal{X}$, and a set of n class labels $\mathcal{Y}$, which we will take without loss of generality to be $\mathcal{Y} = [n]$. There is a (unknown) joint probability distribution D on $\mathcal{X} \times \mathcal{Y}$ from which labeled examples (X, Y) are drawn. In the standard (non-noisy) supervised learning setting, the learner would be given training examples drawn directly from D. When learning from noisy labels, however, the learner does not get clean labels Y; instead, the learner sees noisy examples $(X, \widetilde{Y})$, where $\widetilde{Y}$ denotes a noisy version of Y. In particular, the learner receives a noisy training sample $\widetilde{S} = ((x_1, \widetilde{y}_1), \ldots, (x_m, \widetilde{y}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, and the goal is to learn a classifier $h : \mathcal{X} \to \mathcal{Y}$ that performs well with respect to the clean distribution D.

We consider here the *class-conditional noise* (CCN) model (Natarajan et al., 2013; van Rooyen and Williamson, 2017), wherein a label y is randomly flipped to a label $\widetilde{y}$ with some probability $\gamma_{y,\widetilde{y}}$ that depends on y and $\widetilde{y}$. In particular, the CCN model is characterized by a row-stochastic *noise matrix* $\mathbf{C} \in [0,1]^{n\times n}$ with entries $c_{y,\widetilde{y}}$, such that for each $y, \widetilde{y} \in [n]$,

$$c_{y,\widetilde{y}} = \mathbf{P}(\widetilde{Y} = \widetilde{y} \,|\, Y = y) = \gamma_{y,\widetilde{y}}\,. \tag{5.1}$$

The noisy training examples seen by the learner can therefore be viewed as being drawn i.i.d. from a 'noisy' distribution $\widetilde{D}$ on $\mathcal{X} \times \mathcal{Y}$, wherein an example (X, Y) is first drawn randomly according to D, and then noise is injected according to the noise matrix $\mathbf{C}$ to generate $(X, \widetilde{Y})$.

Learning goal. Given a noisy training sample $\widetilde{S}$ drawn according to the noisy distribution $\widetilde{D}$ as above, the goal of the learner is to learn a classifier $h : \mathcal{X} \to \mathcal{Y}$ that performs well under the clean distribution D. To measure performance, we consider a general multiclass loss matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$, with entries $\ell_{y,\widehat{y}} = \ell(\widehat{y}, y)$ indicating the loss incurred on predicting $\widehat{y}$ when the true label is y (the 0-1 loss $\mathbf{L}^{0\text{-}1}$ with $\ell^{0\text{-}1}_{y,\widehat{y}} = \mathbf{1}(\widehat{y} \neq y)$ is a special case). The performance of the classifier h is then measured by the $\mathbf{L}$-*generalization error* (*risk for* $\mathbf{L}$) under D:

$$\mathrm{er}^{\mathbf{L}}_{D}[h] = \mathbf{E}_{(X,Y)\sim D}[\ell(h(X), Y)]\,. \tag{5.2}$$

The *Bayes* $\mathbf{L}$-*error* (*Bayes risk for* $\mathbf{L}$) under D is then the lowest possible value of the $\mathbf{L}$-generalization error under D:

$$\mathrm{er}^{\mathbf{L},*}_{D} = \inf_{h:\mathcal{X}\to\mathcal{Y}} \mathrm{er}^{\mathbf{L}}_{D}[h]\,. \tag{5.3}$$

The classifier that achieves Bayes $\mathbf{L}$-error is called *Bayes optimal classifier for* $\mathbf{L}$. The $\mathbf{L}$-*regret* (*excess risk for* $\mathbf{L}$) of a classifier h is the difference between the $\mathbf{L}$-generalization error of h and the Bayes $\mathbf{L}$-error:

$$\mathrm{regret}^{\mathbf{L}}_{D}[h] = \mathrm{er}^{\mathbf{L}}_{D}[h] - \mathrm{er}^{\mathbf{L},*}_{D}\,. \tag{5.4}$$

Our goal is to design *consistent* algorithms for a given loss $\mathbf{L}$: as the number of (noisy) training examples increases, algorithms $\mathcal{A}$ can output a classifier whose $\mathbf{L}$-regret converges to zero in probability:

$$\text{for all } \epsilon > 0, \quad \mathbf{P}_{\widetilde{S} \sim \widetilde{D}^m}\big(\text{regret}_D^{\mathbf{L}}[\mathcal{A}(\widetilde{S})] > \epsilon\big) \to 0 \quad \text{as} \quad m \to \infty \,. \tag{5.5}$$

Surrogate losses. Since it is computationally hard to optimize discrete losses $\mathbf{L}$, algorithms often learn a scoring function $\mathbf{f} : \mathcal{X} \to \mathcal{C}$ (where $\mathcal{C}$ is usually $\mathbb{R}^n$ or $\mathbb{R}^{n-1}$) by minimizing a surrogate loss $\psi : \mathcal{C} \times \mathcal{Y} \to \mathbb{R}_+$, where $\psi(\mathbf{u}, y)$ denotes the loss for the score $\mathbf{u}$ when the clean label is y. Then a decoding function decode : $\mathcal{C} \to \mathcal{Y}$ is applied to map scores to labels in $\mathcal{Y}$. For example, in multiclass classification problems, multiclass logistic regression loss, a composition of cross entropy loss (aka. log loss) and softmax, is often used as the surrogate loss, and argmax is the corresponding decoding function. Specifically, given the multiclass scoring function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^n$ with $f_1(x), ..., f_n(x)$ denoting elements of $\mathbf{f}(x)$, the softmax function mapping from $\mathbb{R}^n$ to Δ_n is defined as:

$$\mathbf{f}(x) \mapsto \begin{bmatrix} \exp(f_1(x))/\sum_{i=1}^n \exp(f_i(x)) \\ \vdots \\ \exp(f_n(x))/\sum_{i=1}^n \exp(f_i(x)) \end{bmatrix}, \tag{5.6}$$

and the cross entropy function mapping from $\Delta_n \times \mathcal{Y}$ to $\mathbb{R}$ is defined as

$$\boldsymbol{\eta}, y \mapsto -\log(\eta_y) \,. \tag{5.7}$$

The multiclass logistic regression loss is a mapping from $\mathbb{R}^n \times \mathcal{Y}$ to $\mathbb{R}$, a composition of cross entropy function and softmax function.

Similar to discrete losses, we can also define risk, Bayes risk, excess risk for a surrogate loss ψ as follows:

$$\mathrm{er}^{\psi}_{\widetilde{D}}[\mathbf{f}] = \mathbf{E}_{(X,\widetilde{Y})\sim\widetilde{D}}\left[\psi(\mathbf{f}(X),\widetilde{Y})\right], \tag{5.8}$$

$$\mathrm{er}^{\psi,*}_{\widetilde{D}} = \inf_{\mathbf{f}:\mathcal{X}\to\mathcal{C}} \mathrm{er}^{\psi}_{\widetilde{D}}[\mathbf{f}], \tag{5.9}$$

$$\mathrm{regret}^{\psi}_{\widetilde{D}}[\mathbf{f}] = \mathrm{er}^{\psi}_{\widetilde{D}}[\mathbf{f}] - \mathrm{er}^{\psi,*}_{\widetilde{D}}. \tag{5.10}$$

It is known that minimizing ψ over $\widetilde{S}$ is a consistent algorithm for ψ under $\widetilde{D}$. A desired property of a surrogate loss ψ is that consistency w.r.t. ψ under $\widetilde{D}$ can be transferred to consistency w.r.t. the target loss $\mathbf{L}$ under D. The notation of *calibration* exactly characterizes this property (Zhang, 2004a,b; Bartlett et al., 2006; Tewari and Bartlett, 2007; Ramaswamy et al., 2013, 2014). Convexity in $\mathbf{u}$ of a surrogate loss $\psi(\mathbf{u}, y)$ is also a typical desired property to have in order to enable fast optimization.

5.2.2. Weighted Loss Method for Binary Learning from Noisy Labels

In binary classification, CCN is characterized by the following noise rates:

$$\gamma_{+1,-1} = \mathbf{P}(\widetilde{Y} = -1|Y = +1) =: \rho_{+1} \tag{5.11}$$

$$\gamma_{-1,+1} = \mathbf{P}(\widetilde{Y} = +1|Y = -1) =: \rho_{-1} \tag{5.12}$$

with $\rho_{+1} + \rho_{-1} < 1$. Denote by $\eta(x)$ the *clean* class probability $\mathbf{P}(Y = +1|X = x)$, and $\widetilde{\eta}(x)$ the *noisy* class probability $\mathbf{P}(\widetilde{Y} = +1|X = x)$. It is well-known that the Bayes optimal classifier for 0-1loss $\mathbf{L}^{\text{0-1}}$ is given by thresholding $\eta(x)$ at 1/2.

Here, we summarize the weighted loss method of Natarajan et al. (2013) for correcting label noise in the binary case. The weighted loss method was built from two key observations. First, the Bayes optimal classifier for $\mathbf{L}^{\text{0-1}}$ under the noisy distribution $\widetilde{D}$, denoted by $\widetilde{h}^*$, uses a threshold other than 1/2. Second, $\widetilde{h}^*$ is the minimizer of a weighted 0-1loss on the noisy distribution $\widetilde{D}$. To see the first, note that

$$\widetilde{h}^*(x) = \operatorname{sign}(\widetilde{\eta}(x) - \frac{1}{2}) = \operatorname{sign}(\eta(x) - \frac{1/2 - \rho_{-1}}{1 - \rho_{+1} - \rho_{-1}})\,. \tag{5.13}$$

To see the second, for $\alpha \in (0,1)$, define α-weighted 0-1loss as

$$U_\alpha(t,y) = (1-\alpha)\mathbf{1}(y=+1)\mathbf{1}(t \le 0) + \alpha\mathbf{1}(y=-1)\mathbf{1}(t>0)\,. \tag{5.14}$$

It can be seen that $U_{\frac{1}{2}}$ corresponds to the classical 0-1loss. The following lemma from Scott (2012) shows that for α-weighted 0-1loss, the minimizer thresholds $\eta(x)$ at α.

Lemma 5.1 (Scott (2012), Lemma 8 in Natarajan et al. (2013).). *Define U_α-risk of f under distribution D as $\operatorname{er}_D^{U_\alpha}[f] = \mathbf{E}_{(X,Y)\sim D}\big[U_\alpha(f(X),Y)\big]$. Then, $f_\alpha^*(x) = \operatorname{sign}(\eta(x)-\alpha)$ minimizes U_α-risk under D.*

Consider the U_α-risk of f under the noisy distribution $\widetilde{D}$. Is there an $\alpha \in (0,1)$ so that the minimizer of U_α-risk under $\widetilde{D}$ is the Bayes optimal classifier h^* for $\mathbf{L}^{0\text{-}1}$ under D? If so, one can simply minimize the empirical U_α-risk over the noisy training sample to find a good classifier. This question is answered by the following theorem.

Theorem 5.2 (Theorem 9 in Natarajan et al. (2013).). *For the choices,*

$$\alpha^* = \frac{1-\rho_{+1}+\rho_{-1}}{2}, \quad \textit{and} \quad A = \frac{1-\rho_{+1}-\rho_{-1}}{2}\,, \tag{5.15}$$

there exists a constant B, independent of f, such that for all functions $f : \mathcal{X} \to \mathbb{R}$,

$$\operatorname{er}_{\widetilde{D}}^{U_{\alpha^*}}[f] = A \cdot \operatorname{er}_D^{\mathbf{L}^{0\text{-}1}}[\operatorname{sign}\circ f] + B\,. \tag{5.16}$$

An immediate result following this theorem is that the minimizer of U_{α^*}-risk under the noisy distribution $\widetilde{D}$ coincides with the Bayes optimal classifier of $\ell_{0\text{-}1}$ under the clean distribution D:

$$\operatorname*{argmin}_f \mathrm{er}_{\widetilde{D}}^{U_{\alpha^*}}[f] = \operatorname*{argmin}_f \mathrm{er}_D^{\mathbf{L}^{0\text{-}1}}[\mathrm{sign} \circ f] = x \mapsto \mathrm{sign}(\eta(x) - \frac{1}{2})\,. \tag{5.17}$$

More generally, for any surrogate loss function ψ with the decomposition

$$\psi(t, y) = \mathbf{1}(y = +1)\psi(t, +1) + \mathbf{1}(y = -1)\psi(t, -1)\,, \tag{5.18}$$

the authors defined ψ_α as α-weighted ψ, analogous to the 0-1loss case in Eq. (5.14), and defined the corresponding ψ_α-risk. The following theorem shows that minimizing ψ_α-risk w.r.t. the noisy distribution $\widetilde{D}$ leads to the minimization of $\ell_{0\text{-}1}$-risk w.r.t. the clean distribution D.

Theorem 5.3 (Theorem 11 in Natarajan et al. (2013).)**.** *Consider the empirical risk minimization problem with noisy labels over some function class $\mathcal{F}$:*

$$\widehat{f}_{\alpha^*} = \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \psi_{\alpha^*}(f(x_i), \widetilde{y}_i)\,, \tag{5.19}$$

where ψ_α is an α-weighted margin loss function of the form

$$\psi_\alpha(t, y) = (1 - \alpha)\mathbf{1}(y = +1)\phi(t) + \alpha\mathbf{1}(y = -1)\phi(-t) \tag{5.20}$$

and $\phi : \mathbb{R} \to [0, \infty)$ is a convex loss function with Lipschitz constant L such that it is classification calibrated (i.e., $\phi'(0) < 0$). Then, for the choices α^* and A in Eq. (5.15), there exists a nondecreasing function $\zeta_{\psi_{\alpha^*}}$ with $\zeta_{\psi_{\alpha^*}}(0) = 0$, such that the following bound holds with probability at least $1 - \delta$:

$$\mathrm{er}_D^{\mathbf{L}^{0\text{-}1}}[\mathrm{sign} \circ \widehat{f}_{\alpha^*}] - \mathrm{er}_D^{\mathbf{L}^{0\text{-}1},*} \leq \frac{1}{A}\zeta_{\psi_{\alpha^*}}\left(\min_{f \in \mathcal{F}} \mathrm{er}_{\widetilde{D}}^{\psi_{\alpha^*}}[f] - \mathrm{er}_{\widetilde{D}}^{\psi_{\alpha^*},*} + 4L \cdot \mathfrak{R}_m(\mathcal{F}) + 2\sqrt{\frac{\log(1/\delta)}{2m}}\right), \tag{5.21}$$

where $\mathfrak{R}_m(\mathcal{F})$ is the Rademacher complexity of the function class $\mathcal{F}$ (see Definition 5.7).

This theorem shows that minimizing the weighted loss ψ_α over the noisy training sample leads to a consistent algorithm for $\mathbf{L}^{0\text{-}1}$ under the clean distribution D. Moreover. From Eq. (5.20), ψ_α is convex (in t) if ϕ is convex. Thus, the empirical risk minimization above can be efficiently solved.

Our goal is to extend the above approach to multiclass learning from noisy labels. To do so, we first study the general question of designing weighted surrogate losses for general multiclass cost-sensitive learning in the standard (non-noisy) setting.

5.3. Weighted Surrogate Loss for Cost-Sensitive Multiclass Classification

In this section, we focus on the standard (non-noisy) multiclass classification setting and show how to design a weighted surrogate loss for a general multiclass loss matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$, using a surrogate loss $\psi : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ for $\mathbf{L}^{0\text{-}1}$. In addition, we show that if ψ is convex in the first parameter, then convexity is preserved in the weighted surrogate loss. Finally, we show that if ψ is classification-calibrated ($\mathbf{L}^{0\text{-}1}$-calibrated), then the weighted surrogate loss is $\mathbf{L}$-calibrated. Then, in Section 5.4 and Section 5.5, we will show how to apply weighted surrogate losses to solve problems in multiclass learning from noisy labels and multiclass learning with a reject option, respectively.

5.3.1. Objective

As mentioned earlier, the weighted loss method of Natarajan et al. (2013) essentially involves techniques used for solving cost-sensitive learning problems in the standard (non-noisy) binary setting (Lemma 5.1 and Theorem 5.2), and those problems have been well-studied (Scott, 2012). However, it is still unclear how to design suitable weighted surrogate losses for general cost-sensitive learning in the standard (non-noisy) multiclass setting. We formulate this problem below.

Specifically, given a general cost-sensitive multiclass loss $\mathbf{L}$ and a surrogate loss $\psi : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ for the multiclass 0-1loss, our goal is to find an $\mathbf{M}$-weighted surrogate loss $\psi^{\mathbf{M}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ of the form:

$$\psi^{\mathbf{M}}(\mathbf{u}, y) = \sum_{i=1}^{n} \sum_{j=1}^{n} m_{i,j} \mathbf{1}(y = i) \psi(\mathbf{u}, j) \tag{5.22}$$

for some $\mathbf{M} \in \mathbb{R}^{n\times n}$. Moreover, when the underlying surrogate loss ψ is classification calibrated (meaning that minimizing ψ leads to minimization of the 0-1loss), minimizing the weighted loss

$\psi^{\mathbf{M}}$ over the training sample should lead to a consistent algorithm for $\mathbf{L}$. In addition, when the underlying surrogate loss $\psi(\mathbf{u}, y)$ is convex in $\mathbf{u}$, the weighted loss $\psi^{\mathbf{M}}(\mathbf{u}, y)$ should be convex in $\mathbf{u}$ as well. Lastly, this method should work for both smooth surrogate losses (e.g., the multiclass logistic regression loss) and non-smooth surrogate losses (the multiclass versions of margin-based losses, e.g., Crammer-Singer SVM and Weston-Watkins SVM losses).

5.3.2. Weighted Surrogate Loss

Consider a general cost-sensitive multiclass loss $\mathbf{L}$. Let $\boldsymbol{\beta} \in \mathbb{R}^n$ and $\alpha > 0$. Define

$$\mathbf{M} = \boldsymbol{\beta} \cdot \mathbf{1}_n^\top - \alpha \mathbf{L}\,, \tag{5.23}$$

where $\mathbf{1}_n \in \mathbb{R}^n$ is a vector with all entries equal to one. Equivalently, we have

$$m_{y,\widehat{y}} = \beta_y - \alpha \ell_{y,\widehat{y}}\,. \tag{5.24}$$

It is clear that a prediction that minimizes $\mathbf{L}$ also maximizes $\mathbf{M}$, and vice versa.

Note that

$$m_{y,\widehat{y}} = \sum_{i=1}^{n}\sum_{j=1}^{n} m_{i,j}\mathbf{1}(y=i)\mathbf{1}(\widehat{y}=j) = \sum_{j=1}^{n} m_{y,j}\mathbf{1}(\widehat{y}=j) = \sum_{j=1}^{n} m_{y,j}(1-\mathbf{1}(\widehat{y}\neq j))\,. \tag{5.25}$$

Now, a prediction maximizing $\mathbf{M}$ is equivalent to a prediction minimizing

$$\sum_{j=1}^{n} m_{y,j}\mathbf{1}(\widehat{y}\neq j) - \sum_{j=1}^{n} m_{y,j}\,. \tag{5.26}$$

It is equivalent to a prediction minimizing

$$\sum_{j=1}^{n} m_{y,j}\mathbf{1}(\widehat{y}\neq j)\,. \tag{5.27}$$

Motivated by the decomposition above, we can similarly define a multiclass weighted loss $U^{\mathbf{M}}$:

$\mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ as

$$U^{\mathbf{M}}(\mathbf{u}, y) = \sum_{j=1}^{n} m_{y,j} \mathbf{1}(\operatorname{argmax}(\mathbf{u}) \neq j)\,. \tag{5.28}$$

This loss is an $\mathbf{M}$-weighted 0-1loss that mimics the behavior of the given cost-sensitive loss $\mathbf{L}$, and will help us construct $\mathbf{M}$-weighted surrogate losses later. As a special case, it recovers the binary version in Eq. (5.14), which we show below.

Proposition 5.4. *Consider binary cost-sensitive loss* $\mathbf{L}$ *of the form*

$$\mathbf{L} = \begin{array}{c c} & \begin{array}{cc} +1 & -1 \end{array} \\ \begin{array}{c} +1 \\ -1 \end{array} & \begin{bmatrix} 0 & 1-\alpha \\ \alpha & 0 \end{bmatrix} \end{array}.$$

Let $\eta(x)$ *be the class probability function* $\mathbf{P}(Y = +1|X = x)$*. Then* $h^*(x) = \operatorname{sign}(\eta(x) - \alpha)$ *is Bayes optimal for* $\mathbf{L}$*. Moreover, let*

$$\boldsymbol{\beta} = \begin{bmatrix} 1-\alpha \\ \alpha \end{bmatrix},$$

and define

$$\mathbf{M} = \begin{bmatrix} 1-\alpha & 1-\alpha \\ \alpha & \alpha \end{bmatrix} - \mathbf{L} = \begin{bmatrix} 1-\alpha & 0 \\ 0 & \alpha \end{bmatrix},$$

according to Eq. (5.23)*. Then* $U^{\mathbf{M}}$ *is equivalent to Eq.* (5.14)*.*

Proof.

$$\mathbf{L}^\top \begin{bmatrix} \eta(x) \\ 1-\eta(x) \end{bmatrix} = \begin{bmatrix} \alpha - \alpha\eta(x) \\ \eta(x) - \alpha\eta(x) \end{bmatrix},$$

so it is clear that

$$\min(\alpha - \alpha\eta(x), \eta(x) - \alpha\eta(x)) = \begin{cases} \alpha - \alpha\eta(x) & \text{if} \quad \eta(x) > \alpha \\ \eta(x) - \alpha\eta(x) & \text{if} \quad \eta(x) \leq \alpha \end{cases} .$$

Hence, $h^*(x) = \operatorname{sign}(\eta(x) - \alpha)$ is Bayes optimal for $\mathbf{L}$.

Define $t = u_{+1} - u_{-1}$, so

$$\mathbf{1}(\operatorname{argmax}(\mathbf{u}) = +1) = \mathbf{1}(t > 0)\,,$$
$$\mathbf{1}(\operatorname{argmax}(\mathbf{u}) = -1) = \mathbf{1}(t \leq 0)\,.$$

Then

$$\begin{aligned} U^{\mathbf{M}}(\mathbf{u}, y) &= (1-\alpha)\mathbf{1}(y = +1)\mathbf{1}(\operatorname{argmax}(\mathbf{u}) = -1) + \alpha\mathbf{1}(y = -1)\mathbf{1}(\operatorname{argmax}(\mathbf{u}) = +1) \\ &= (1-\alpha)\mathbf{1}(y = +1)\mathbf{1}(t \leq 0) + \alpha\mathbf{1}(y = -1)\mathbf{1}(t > 0)\,, \end{aligned}$$

which is Eq. (5.14). □

Remark. This proposition also recovers Lemma 5.1.

However, $U^{\mathbf{M}}$ is still discrete and, therefore, hard to optimize. To overcome this issue, we can replace $\mathbf{1}(\operatorname{argmax}(\mathbf{u}) \neq j)$ by a continuous surrogate function ψ for $\mathbf{L}^{0\text{-}1}$ as in Eq. (5.20).

Definition 5.1. *Let $\psi : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ be a surrogate loss for $\mathbf{L}^{0\text{-}1}$. We define the corresponding $\mathbf{M}$-weighted surrogate loss $\psi^{\mathbf{M}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ as*

$$\psi^{\mathbf{M}}(\mathbf{u}, y) = \sum_{j=1}^{n} m_{y,j}\psi(\mathbf{u}, j)\,. \tag{5.29}$$

ERM framework. We can use empirical risk minimization (ERM) framework to minimize the unbiased estimator of

$$\mathrm{er}_D^{\psi^{\mathbf{M}}}[\mathbf{f}] = \mathbf{E}_{(X,Y)\sim D}[\psi^{\mathbf{M}}(\mathbf{f}(X), Y)] \tag{5.30}$$

over a training sample $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ to learn a scoring function $\mathbf{f}$ in some function class $\mathcal{F}^n$, where $\mathcal{F} \subseteq (\mathcal{X} \to \mathbb{R})$, as

$$\widehat{\mathbf{f}} = \min_{\mathbf{f}\in\mathcal{F}^n} \frac{1}{m} \sum_{i=1}^{m} \psi^{\mathbf{M}}(\mathbf{f}(x_i), y_i)\,. \tag{5.31}$$

Then, the resulting classifier $h : \mathcal{X} \to [n]$ is $h = \mathrm{argmax} \circ \widehat{\mathbf{f}}$.

5.3.3. Convexity of $\psi^{\mathbf{M}}$

The following proposition shows that for non-negative $\mathbf{M}$, if the underlying surrogate loss $\psi(\mathbf{u}, y)$ is convex in $\mathbf{u}$, then the weighted loss $\psi^{\mathbf{M}}(\mathbf{u}, y)$ is convex in $\mathbf{u}$ as well.

Proposition 5.5. *Suppose $\psi(\mathbf{u}, y)$ is convex in $\mathbf{u}$, and $\mathbf{M}$ is a non-negative matrix, i.e., $m_{i,j} \geq 0$. Then $\psi^{\mathbf{M}}(\mathbf{u}, y)$ is convex in $\mathbf{u}$.*

Proof. This follows immediately since $\psi^{\mathbf{M}}(\mathbf{u}, y)$ is a linear combination of convex functions with non-negative coefficients. $\square$

5.3.4. Examples of Various Multiclass Surrogate Losses and Corresponding M-weighted Surrogate Losses

Below, we apply the method above to several commonly used multiclass surrogate losses. In the remark at the end, we mention calibration results of these multiclass surrogate losses.

Example 5.1 (Multiclass logistic regression surrogate loss (Zhang, 2004b))**.** *Suppose $\psi_{\mathrm{mlog}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ is the multiclass logistic regression surrogate loss, i.e.,*

$$\psi_{\mathrm{mlog}}(\mathbf{u}, y) = -u_y + \ln(\sum_{y'=1}^{n} \exp(u_{y'}))\,. \tag{5.32}$$

Then the corresponding $\mathbf{M}$*-weighted multiclass logistic regression surrogate loss is*

$$\begin{aligned}\psi^{\mathbf{M}}_{\mathrm{mlog}}(\mathbf{u}, y) &= \sum_{j=1}^{n} m_{y,j}\psi_{\mathrm{mlog}}(\mathbf{u}, j)\\ &= \sum_{j=1}^{n} m_{y,j}[-u_y + \ln(\sum_{y'=1}^{n}\exp(u_{y'}))]\\ &= -\sum_{j=1}^{n} m_{y,j}\cdot u_y + \ln(\sum_{y'=1}^{n}\exp(u_{y'}))\cdot\sum_{j=1}^{n} m_{y,j}\,. \qquad (5.33)\end{aligned}$$

Example 5.2 (One-vs-all logistic regression surrogate loss (Zhang, 2004b))**.** *Suppose* $\psi_{\mathrm{OvA,log}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ *is the one-vs-all logistic regression surrogate loss, i.e.,*

$$\psi_{\mathrm{OvA,log}}(\mathbf{u}, y) = \ln(1 + e^{-u_y}) + \sum_{y' \neq y}\ln(1 + e^{u_{y'}})\,. \qquad (5.34)$$

Then the corresponding $\mathbf{M}$*-weighted one-vs-all logistic regression surrogate loss is*

$$\begin{aligned}\psi^{\mathbf{M}}_{\mathrm{OvA,log}}(\mathbf{u}, y) &= \sum_{j=1}^{n} m_{y,j}\psi_{\mathrm{OvA,log}}(\mathbf{u}, j)\\ &= \sum_{j=1}^{n} m_{y,j}[\ln(1 + e^{-u_y}) + \sum_{y' \neq y}\ln(1 + e^{u_{y'}})]\,. \qquad (5.35)\end{aligned}$$

Example 5.3 (Crammer-Singer surrogate loss (Crammer and Singer, 2001))**.** *Suppose* $\psi_{\mathrm{CS}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ *is the Crammer-Singer surrogate loss, i.e.,*

$$\psi_{\mathrm{CS}}(\mathbf{u}, y) = \max_{y' \neq y}(1 - (u_y - u_{y'}))_+\,, \qquad (5.36)$$

where $(z)_+ = \max(0, z)$ *is hinge loss. Then the corresponding* $\mathbf{M}$*-weighted Crammer-Singer surro-*

gate loss is

$$\psi^{\mathbf{M}}_{\text{CS}}(\mathbf{u}, y) = \sum_{j=1}^{n} m_{y,j} \psi_{\text{CS}}(\mathbf{u}, j)$$
$$= \sum_{j=1}^{n} m_{y,j} \cdot \max_{y' \neq y} (1 - (u_y - u_{y'}))_+ \,. \tag{5.37}$$

Example 5.4 (One-vs-all hinge surrogate loss (Zhang, 2004b)). *Suppose $\psi_{\text{OvA,hinge}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ is the one-vs-all hinge surrogate loss, i.e.,*

$$\psi_{\text{OvA,hinge}}(\mathbf{u}, y) = (1 - u_y)_+ + \sum_{y' \neq y} (1 + u_{y'})_+ \,, \tag{5.38}$$

where $(z)_+ = \max(0, z)$ is hinge loss. Then the corresponding $\mathbf{M}$-weighted one-vs-all hinge surrogate loss is

$$\psi^{\mathbf{M}}_{\text{OvA,hinge}}(\mathbf{u}, y) = \sum_{j=1}^{n} m_{y,j} \psi_{\text{OvA,hinge}}(\mathbf{u}, j)$$
$$= \sum_{j=1}^{n} m_{y,j} [(1 - u_y)_+ + \sum_{y' \neq y} (1 + u_{y'})_+] \,. \tag{5.39}$$

Example 5.5 (Weston-Watkins surrogate loss (Weston and Watkins, 1999)). *Suppose $\psi_{\text{WW}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ is the Weston-Watkins surrogate loss, i.e.,*

$$\psi_{\text{WW}}(\mathbf{u}, y) = \sum_{y' \neq y} (1 - (u_y - u_{y'}))_+ \,, \tag{5.40}$$

where $(z)_+ = \max(0, z)$ is hinge loss. Then the corresponding $\mathbf{M}$-weighted Weston-Watkins surrogate *loss is*

$$\begin{aligned}\psi_{\mathrm{WW}}^{\mathbf{M}}(\mathbf{u}, y) &= \sum_{j=1}^{n} m_{y,j} \psi_{\mathrm{WW}}(\mathbf{u}, j) \\ &= \sum_{j=1}^{n} m_{y,j} \cdot \sum_{y' \neq y} (1 - (u_y - u_{y'}))_+ \,. \end{aligned} \tag{5.41}$$

Example 5.6 (Lee-Lin-Wahba surrogate loss (Lee et al., 2004))**.** *Suppose* $\psi_{\mathrm{LLW}} : \mathcal{C} \times \mathcal{Y} \to \mathbb{R}_+$ *is the Lee-Lin-Wahba surrogate loss, where* $\mathcal{C} = \{\mathbf{u} \in \mathbb{R}^n : \sum_{j=1}^n u_j = 0\}$*, i.e.,*

$$\psi_{\mathrm{LLW}}(\mathbf{u}, y) = \sum_{y' \neq y} (1 + u_{y'})_+ \,, \tag{5.42}$$

where $(z)_+ = \max(0, z)$ *is hinge loss. Then the corresponding* $\mathbf{M}$*-weighted Lee-Lin-Wahba surrogate loss is*

$$\begin{aligned}\psi_{\mathrm{LLW}}^{\mathbf{M}}(\mathbf{u}, y) &= \sum_{j=1}^{n} m_{y,j} \psi_{\mathrm{LLW}}(\mathbf{u}, j) \\ &= \sum_{j=1}^{n} m_{y,j} \cdot \sum_{y' \neq y} (1 + u_{y'})_+ \,. \end{aligned} \tag{5.43}$$

Example 5.7 (Generalized cross entropy surrogate loss (Zhang and Sabuncu, 2018; Cao et al., 2022))**.** *Suppose* $\psi_{\mathrm{gce},\gamma} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ *is the generalized cross entropy surrogate loss for* $\gamma \in (0, 1]$*, i.e.,*

$$\psi_{\mathrm{gce},\gamma}(\mathbf{u}, y) = \frac{1}{\gamma}\Big[1 - \big(\frac{e^{u_y}}{\sum_{i=1}^n e^{u_i}}\big)^{\gamma}\Big] \,. \tag{5.44}$$

Then the corresponding $\mathbf{M}$*-weighted generalized cross entropy surrogate loss is*

$$\begin{aligned}\psi_{\mathrm{gce},\gamma}^{\mathbf{M}}(\mathbf{u}, y) &= \sum_{j=1}^{n} m_{y,j} \psi_{\mathrm{gce},\gamma}(\mathbf{u}, j) \\ &= \sum_{j=1}^{n} m_{y,j} \cdot \frac{1}{\gamma}\Big[1 - \big(\frac{e^{u_y}}{\sum_{i=1}^n e^{u_i}}\big)^{\gamma}\Big] \,. \end{aligned} \tag{5.45}$$

Remark. Among the surrogate losses shown in the examples above, ψ_{mlog}, $\psi_{\text{OvA,log}}$ and ψ_{LLW} are universally calibrated for $\mathbf{L}^{0\text{-}1}$ (calibrated for all probability distributions), while ψ_{CS}, $\psi_{\text{OvA,hinge}}$ and ψ_{WW} are calibrated under the so-called 'dominant-label' condition (calibrated for probability distributions in which the conditional distributions $p(y|\mathbf{x})$ assign probability at least $\frac{1}{2}$ to one of the n classes) (Zhang, 2004b; Tewari and Bartlett, 2007). Cao et al. (2022) showed that the generalized cross entropy surrogate loss $\psi_{\text{gce},\gamma}$ is universally calibrated for $\mathbf{L}^{0\text{-}1}$ for any $\gamma \in (0, 1]$.

5.3.5. Calibration and Consistency

In this section, we show that if ψ is classification-calibrated ($\mathbf{L}^{0\text{-}1}$-calibrated), then the weighted surrogate loss is $\mathbf{L}$-calibrated. We also provide an estimation error bound for using ERM framework over a training sample. We start with some notations and definitions.

For surrogate prediction space $\mathcal{C}$ and surrogate loss $\psi : \mathcal{C}\times\mathcal{Y} \to \mathbb{R}_+$, we have the following definitions.

Definition 5.2 (Conditional risk). *Condition risk $L_\psi : \mathcal{C} \times \Delta_n \to \mathbb{R}_+$ is defined as*

$$L_\psi(\mathbf{u}, \mathbf{p}) = \mathbf{E}_{Y\sim\mathbf{p}}\Big[\psi(\mathbf{u}, Y)\Big] . \tag{5.46}$$

Definition 5.3 (Conditional Bayes risk). *Condition Bayes risk $H_\psi : \Delta_n \to \mathbb{R}_+$ is defined as*

$$H_\psi(\mathbf{p}) = \inf_{\mathbf{u}\in\mathcal{C}} L_\psi(\mathbf{u}, \mathbf{p}) . \tag{5.47}$$

Definition 5.4 (Conditional regret). *Condition Bayes risk $R_\psi : \mathcal{C} \times \Delta_n \to \mathbb{R}_+$ is defined as*

$$R_\psi(\mathbf{u}, \mathbf{p}) = L_\psi(\mathbf{u}, \mathbf{p}) - H_\psi(\mathbf{p}) . \tag{5.48}$$

Definition 5.5 (Multiclass classification calibration (Zhang, 2004b; Tewari and Bartlett, 2007; Steinwart, 2007)). *A multiclass surrogate loss $\psi : \mathcal{C} \times \mathcal{Y} \to \mathbb{R}_+$ is $\mathbf{L}^{0\text{-}1}$-calibrated (classification-*

calibrated) if

$$\forall \mathbf{p} \in \Delta_n : \quad \inf_{\mathbf{u} \in \mathcal{C} : \operatorname{argmax}_{\widehat{y}} u_{\widehat{y}} \notin \operatorname{argmin}_{\widehat{y}} (\boldsymbol{\ell}^{0\text{-}1}_{\widehat{y}})^\top \mathbf{p}} L_\psi(\mathbf{u}, \mathbf{p}) > \inf_{\mathbf{u} \in \mathcal{C}} L_\psi(\mathbf{u}, \mathbf{p}) = H_\psi(\mathbf{p}) , \tag{5.49}$$

where $\boldsymbol{\ell}^{0\text{-}1}_{\widehat{y}}$ is the $\widehat{y}$-column of $\mathbf{L}^{0\text{-}1}$. Or equivalently, if

$$\forall \mathbf{p} \in \Delta_n : \quad \inf_{\mathbf{u} \in \mathcal{C} : \operatorname{argmax}_{\widehat{y}} u_{\widehat{y}} \notin \operatorname{argmax}_{\widehat{y}} p_{\widehat{y}}} L_\psi(\mathbf{u}, \mathbf{p}) > \inf_{\mathbf{u} \in \mathcal{C}} L_\psi(\mathbf{u}, \mathbf{p}) = H_\psi(\mathbf{p}) . \tag{5.50}$$

We can also define calibration for a general multiclass loss $\mathbf{L}$.

Definition 5.6 (Multiclass $\mathbf{L}$ calibration)**.** *A multiclass surrogate loss $\psi : \mathcal{C} \times \mathcal{Y} \to \mathbb{R}_+$ is $\mathbf{L}$-calibrated if*

$$\forall \mathbf{p} \in \Delta_n : \quad \inf_{\mathbf{u} \in \mathcal{C} : \operatorname{argmax}_{\widehat{y}} u_{\widehat{y}} \notin \operatorname{argmin}_{\widehat{y}} (\boldsymbol{\ell}_{\widehat{y}})^\top \mathbf{p}} L_\psi(\mathbf{u}, \mathbf{p}) > \inf_{\mathbf{u} \in \mathcal{C}} L_\psi(\mathbf{u}, \mathbf{p}) = H_\psi(\mathbf{p}) , \tag{5.51}$$

where $\boldsymbol{\ell}_{\widehat{y}}$ is the $\widehat{y}$-column of $\mathbf{L}$.

Calibration result.

Theorem 5.6. *Let $\psi : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ be a surrogate loss. For a general multiclass loss $\mathbf{L}$, define $\mathbf{M} = \boldsymbol{\beta} \cdot \mathbf{1}_n^\top - \alpha \mathbf{L}$, $\alpha > 0$, where $\mathbf{1}_n \in \mathbb{R}^n$ is a vector with all entries equal to one. Let $\boldsymbol{\beta}$ be such that*

$$\beta_y = \alpha \cdot \max_j \ell_{y,j} .$$

Then,

$$m_{y,\widehat{y}} = \beta_y - \alpha \ell_{y,\widehat{y}} \geq 0 .$$

So all entries of $\mathbf{M}$ *are non-negative. Define the weighted surrogate loss* $\psi^{\mathbf{M}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ *as*

$$\psi^{\mathbf{M}}(\mathbf{u}, y) = \sum_{j=1}^{n} m_{y,j} \psi(\mathbf{u}, j) \,.$$

Then if ψ *is* $\mathbf{L}^{0\text{-}1}$*-calibrated, we have that* $\psi^{\mathbf{M}}$ *is* $\mathbf{L}$*-calibrated.*

Proof. We have the conditional risk for ψ as

$$L_{\psi}(\mathbf{u}, \mathbf{p}) = \mathbf{E}_{Y \sim \mathbf{p}}\Big[\psi(\mathbf{u}, Y)\Big] = \psi(\mathbf{u})^{\top}\mathbf{p} \,, \tag{5.52}$$

where $\psi(\mathbf{u}) = [\psi(\mathbf{u}, 1), ..., \psi(\mathbf{u}, n)]^{\top}$.

The conditional risk for $\psi^{\mathbf{M}}$ is

$$\begin{aligned} L_{\psi^{\mathbf{M}}}(\mathbf{u}, \mathbf{p}) &= \mathbf{E}_{Y \sim \mathbf{p}}\Big[\psi^{\mathbf{M}}(\mathbf{u}, Y)\Big] \\ &= \mathbf{E}_{Y \sim \mathbf{p}}\Big[\sum_{j=1}^{n} m_{Y,j}\psi(\mathbf{u}, j)\Big] \\ &= \sum_{j=1}^{n} \psi(\mathbf{u}, j) \mathbf{E}_{Y \sim \mathbf{p}}\Big[m_{Y,j}\Big] \\ &= \sum_{j=1}^{n} \psi(\mathbf{u}, j)(\mathbf{M}^{\top}\mathbf{p})_j \\ &= \psi(\mathbf{u})^{\top}(\mathbf{M}^{\top}\mathbf{p}) \,. \end{aligned} \tag{5.53}$$

Let $\omega_{\mathbf{M}}(\mathbf{p}) = \|\mathbf{M}^{\top}\mathbf{p}\|_1$ and $\nu_{\mathbf{M}}(\mathbf{p}) = \frac{\mathbf{M}^{\top}\mathbf{p}}{\|\mathbf{M}^{\top}\mathbf{p}\|_1}$. Since $\mathbf{M}$ is a non-negative matrix, $\nu_{\mathbf{M}}(\mathbf{p}) \in \Delta_n$. Moreover,

$$\mathbf{L} = \frac{\boldsymbol{\beta} \cdot \mathbf{1}_n^{\top} - \mathbf{M}}{\alpha} \,. \tag{5.54}$$

Then,

$$
\begin{aligned}
&\inf_{\mathbf{u}\in\mathcal{C}:\mathrm{argmax}_{\widehat{y}}\, u_{\widehat{y}}\notin \mathrm{argmin}_{\widehat{y}}(\boldsymbol{\ell}_{\widehat{y}})^\top \mathbf{p}} L_{\psi^{\mathbf{M}}}(\mathbf{u},\mathbf{p}) > \inf_{\mathbf{u}\in\mathcal{C}} L_{\psi^{\mathbf{M}}}(\mathbf{u},\mathbf{p})\\
\iff& \inf_{\mathbf{u}\in\mathcal{C}:\mathrm{argmax}_{\widehat{y}}\, u_{\widehat{y}}\notin \mathrm{argmin}_{\widehat{y}}(\boldsymbol{\ell}_{\widehat{y}})^\top \mathbf{p}} \psi(\mathbf{u})^\top(\mathbf{M}^\top\mathbf{p}) > \inf_{\mathbf{u}\in\mathcal{C}} \psi(\mathbf{u})^\top(\mathbf{M}^\top\mathbf{p})\\
\iff& \inf_{\mathbf{u}\in\mathcal{C}:\mathrm{argmax}_{\widehat{y}}\, u_{\widehat{y}}\notin \mathrm{argmin}_{\widehat{y}}(\boldsymbol{\ell}_{\widehat{y}})^\top \mathbf{p}} \psi(\mathbf{u})^\top(\omega_{\mathbf{M}}(\mathbf{p})\nu_{\mathbf{M}}(\mathbf{p})) > \inf_{\mathbf{u}\in\mathcal{C}} \psi(\mathbf{u})^\top(\omega_{\mathbf{M}}(\mathbf{p})\nu_{\mathbf{M}}(\mathbf{p}))\\
\iff& \inf_{\mathbf{u}\in\mathcal{C}:\mathrm{argmax}_{\widehat{y}}\, u_{\widehat{y}}\notin \mathrm{argmin}_{\widehat{y}}(\mathbf{L}^\top \mathbf{p})_{\widehat{y}}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p}) > \inf_{\mathbf{u}\in\mathcal{C}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p})\\
\iff& \inf_{\mathbf{u}\in\mathcal{C}:\mathrm{argmax}_{\widehat{y}}\, u_{\widehat{y}}\notin \mathrm{argmin}_{\widehat{y}}((\frac{\boldsymbol{\beta}\cdot\mathbf{1}_n^\top-\mathbf{M}}{\alpha})^\top \mathbf{p})_{\widehat{y}}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p}) > \inf_{\mathbf{u}\in\mathcal{C}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p})\\
\iff& \inf_{\mathbf{u}\in\mathcal{C}:\mathrm{argmax}_{\widehat{y}}\, u_{\widehat{y}}\notin \mathrm{argmin}_{\widehat{y}}(\frac{\mathbf{1}_n\cdot\boldsymbol{\beta}^\top-\mathbf{M}^\top}{\alpha} \mathbf{p})_{\widehat{y}}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p}) > \inf_{\mathbf{u}\in\mathcal{C}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p})\\
\iff& \inf_{\mathbf{u}\in\mathcal{C}:\mathrm{argmax}_{\widehat{y}}\, u_{\widehat{y}}\notin \mathrm{argmax}_{\widehat{y}}(\mathbf{M}^\top \mathbf{p})_{\widehat{y}}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p}) > \inf_{\mathbf{u}\in\mathcal{C}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p})\\
\iff& \inf_{\mathbf{u}\in\mathcal{C}:\mathrm{argmax}_{\widehat{y}}\, u_{\widehat{y}}\notin \mathrm{argmax}_{\widehat{y}}(\frac{\mathbf{M}^\top\mathbf{p}}{\|\mathbf{M}^\top\mathbf{p}\|_1})_{\widehat{y}}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p}) > \inf_{\mathbf{u}\in\mathcal{C}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p})\\
\iff& \inf_{\mathbf{u}\in\mathcal{C}:\mathrm{argmax}_{\widehat{y}}\, u_{\widehat{y}}\notin \mathrm{argmax}_{\widehat{y}}(\nu_{\mathbf{M}}(\mathbf{p}))_{\widehat{y}}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p}) > \inf_{\mathbf{u}\in\mathcal{C}} \psi(\mathbf{u})^\top\nu_{\mathbf{M}}(\mathbf{p})\,. \qquad (5.55)
\end{aligned}
$$

Since $\nu_{\mathbf{M}}(\mathbf{p}) \in \Delta_n$ and ψ is $\mathbf{L}^{0\text{-}1}$-calibrated, this shows $\psi^{\mathbf{M}}$ is $\mathbf{L}$-calibrated. □

Remark. In Theorem 5.6, surrogate losses ψ must be universally calibrated for $\mathbf{L}^{0\text{-}1}$ (calibrated for all probability distributions). Such surrogate losses include ψ_{mlog}, $\psi_{\text{OvA,log}}$, ψ_{LLW} and $\psi_{\text{gce},\gamma}$ (see Section 5.3.4).

Estimation error bound. Below, we provide an estimation error bound for scoring function $\widehat{\mathbf{f}}$ learned using ERM framework over a training sample.

Definition 5.7 (Rademacher complexity (Bartlett and Mendelson, 2002; Cao et al., 2022))**.** *Let $Z_1, ..., Z_m \in Z$ be m i.i.d. random variables drawn from a probability distribution μ, and $\mathcal{F} = \{f : Z \to \mathbb{R}\}$ be a class of measurable functions. Then the expected Rademacher complexity of function* *class $\mathcal{F}$ is given as follows:*

$$\mathfrak{R}_m(\mathcal{F}) = \mathbf{E}_{Z_1,\dots,Z_m\sim\mu}\mathbf{E}_{\boldsymbol{\sigma}}\Big[\sup_{f\in\mathcal{F}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i f(Z_i)\Big], \tag{5.56}$$

where $\sigma_1,\dots,\sigma_m$ are Rademacher random variables taking value from $\{-1,1\}$ uniformly.

Theorem 5.7. *Fix $\delta\in(0,1)$. Suppose scoring function $\mathbf{f}$ belongs to some function class $\mathcal{F}^n$, where each $f_i\in\mathcal{F}\subseteq(\mathcal{X}\to\mathbb{R})$. Then $\mathbf{f}\in\mathcal{F}^n\subseteq(\mathcal{X}\to\mathbb{R}^n)$. Let $\psi(\cdot,y)$ be ρ-Lipschitz continuous and is bounded by $C_\psi>0$ for all y. Let $M=\max_{y,j}|m_{y,j}|$. Assume that the identifiable condition holds, i.e., $\min_{\mathbf{f}\in\mathcal{F}^n}\mathrm{er}_D^{\psi^{\mathbf{M}}}[\mathbf{f}]=\mathrm{er}_D^{\psi^{\mathbf{M}},*}$. Then the following inequality holds with probability at least $1-\delta$:*

$$\mathrm{er}_D^{\psi^{\mathbf{M}}}[\widehat{\mathbf{f}}]-\mathrm{er}_D^{\psi^{\mathbf{M}},*}\le 4\sqrt{2}n^2M\rho\mathfrak{R}_m(\mathcal{F})+nMC_\psi\sqrt{\frac{2\ln(2/\delta)}{m}}. \tag{5.57}$$

Proof. From the conditions, it is easy to see that $\psi^{\mathbf{M}}(\cdot,y)$ is $(nM\rho)$-Lipschitz continuous and is bounded by nMC_ψ for all y. Then we can follow routine (Bartlett and Mendelson, 2002; Mohri et al., 2012; Cao et al., 2022) to show

$$\begin{aligned}\sup_{\mathbf{f}\in\mathcal{F}^n}|\mathrm{er}_D^{\psi^{\mathbf{M}}}[\mathbf{f}]-\mathrm{er}_S^{\psi^{\mathbf{M}}}[\mathbf{f}]| &\le 2\sqrt{2}nM\rho\sum_{j=1}^{n}\mathfrak{R}_m(\mathcal{F})+nMC_\psi\sqrt{\frac{\ln(2/\delta)}{2m}}\\ &=2\sqrt{2}n^2M\rho\mathfrak{R}_m(\mathcal{F})+nMC_\psi\sqrt{\frac{\ln(2/\delta)}{2m}},\end{aligned} \tag{5.58}$$

using McDiarmid's inequality (McDiarmid, 1989) and Talagrand's contraction lemma (Maurer, 2016), where $\mathrm{er}_S^{\psi^{\mathbf{M}}}[\mathbf{f}]=\frac{1}{m}\sum_{i=1}^{m}\psi^{\mathbf{M}}(\mathbf{f}(x_i),y_i)$.

Then for $\mathbf{f}^* = \operatorname{argmin}_{\mathbf{f}\in\mathcal{F}^n} \mathrm{er}_D^{\psi^{\mathbf{M}}}[\mathbf{f}]$, we have

$$\begin{aligned}
\mathrm{er}_D^{\psi^{\mathbf{M}}}[\widehat{\mathbf{f}}] - \min_{\mathbf{f}\in\mathcal{F}^n} \mathrm{er}_D^{\psi^{\mathbf{M}}}[\mathbf{f}] &= \mathrm{er}_D^{\psi^{\mathbf{M}}}[\widehat{\mathbf{f}}] - \mathrm{er}_D^{\psi^{\mathbf{M}},*} \\
&= \Big(\mathrm{er}_D^{\psi^{\mathbf{M}}}[\widehat{\mathbf{f}}] - \mathrm{er}_S^{\psi^{\mathbf{M}}}[\widehat{\mathbf{f}}]\Big) + \Big(\mathrm{er}_S^{\psi^{\mathbf{M}}}[\widehat{\mathbf{f}}] - \mathrm{er}_S^{\psi^{\mathbf{M}}}[\mathbf{f}^*]\Big) + \Big(\mathrm{er}_S^{\psi^{\mathbf{M}}}[\mathbf{f}^*] - \mathrm{er}_D^{\psi^{\mathbf{M}},*}\Big) \\
&\le \Big(\mathrm{er}_D^{\psi^{\mathbf{M}}}[\widehat{\mathbf{f}}] - \mathrm{er}_S^{\psi^{\mathbf{M}}}[\widehat{\mathbf{f}}]\Big) + \Big(\mathrm{er}_S^{\psi^{\mathbf{M}}}[\mathbf{f}^*] - \mathrm{er}_D^{\psi^{\mathbf{M}},*}\Big) \\
&\le 2 \sup_{\mathbf{f}\in\mathcal{F}^n} |\mathrm{er}_D^{\psi^{\mathbf{M}}}[\mathbf{f}] - \mathrm{er}_S^{\psi^{\mathbf{M}}}[\mathbf{f}]| \\
&= 4\sqrt{2}n^2 M\rho\mathfrak{R}_m(\mathcal{F}) + nMC_\psi\sqrt{\frac{2\ln(2/\delta)}{m}}\,. \qquad (5.59)
\end{aligned}$$

□

Regret transfer bound. It is known that $\psi^{\mathbf{M}}$ is $\mathbf{L}$-calibrated is equivalent to that $\psi^{\mathbf{M}}$ admits a regret transfer bound w.r.t. $\mathbf{L}$ (Zhang, 2004b; Tewari and Bartlett, 2007; Steinwart, 2007).

5.4. Weighted Loss Method for Multiclass Learning from Noisy Labels

In this section, we show how to apply weighted surrogate losses in Section 5.3 to learn from noisy labels in multiclass classification; we achieve this by choosing appropriate weights designed to correct label noise.

We will denote by $\boldsymbol{\eta}, \widetilde{\boldsymbol{\eta}} : \mathcal{X}\to\Delta_n$ the (vector) class probability functions under the clean distribution D associated with clean labeled examples and the noisy distribution $\widetilde{D}$ associated with noisy examples, respectively, with components given by

$$\begin{aligned}
\eta_y(x) &= \mathbf{P}(Y = y \,|\, X = x) \\
\widetilde{\eta}_y(x) &= \mathbf{P}(\widetilde{Y} = y \,|\, X = x)
\end{aligned}$$

for each $y \in [n]$. It is easy to see that

$$\begin{aligned}
\widetilde{\eta}_y(x) &= \sum_{y' \in [n]} \mathbf{P}(\widetilde{Y} = y \,|\, Y = y') \cdot \mathbf{P}(Y = y' \,|\, X = x) \\
&= \sum_{y' \in [n]} \gamma_{y',y} \cdot \eta_{y'}(x) \\
&= \mathbf{c}_y^\top \boldsymbol{\eta}(x) \,,
\end{aligned}$$

which gives

$$\widetilde{\boldsymbol{\eta}}(x) = \mathbf{C}^\top \boldsymbol{\eta}(x) \,.$$

Therefore, provided $\mathbf{C}$ is invertible, we have

$$\boldsymbol{\eta}(x) \;=\; (\mathbf{C}^\top)^{-1} \widetilde{\boldsymbol{\eta}}(x) \,. \tag{5.60}$$

Bayes optimal classifier for L. For any multiclass loss matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$, the Bayes optimal classifier for $\mathbf{L}$ (which for any instance x, chooses a prediction that minimizes the expected loss under $\mathbf{L}$) is given by

$$h_D^{\mathbf{L},*}(x) \;=\; \operatorname*{argmin}_{y \in [n]} \, (\mathbf{L}^\top \boldsymbol{\eta}(x))_y \,.$$

Weighted loss method to learn from noisy labels. We first show that we can convert a problem of multiclass learning from noisy labels into a problem of multiclass learning for a general loss $\mathbf{L}$.

Proposition 5.8. *Let $\mathbf{L} = \mathbf{C}^{-1}\mathbf{L}^{\text{0-1}}$. Then any Bayes optimal classifier for $\mathbf{L}$ w.r.t. $\widetilde{D}$ is also Bayes optimal for $\mathbf{L}^{\text{0-1}}$ w.r.t. D.*

Proof.

$$\begin{aligned}
h_{\widetilde{D}}^{\mathbf{L},*}(x) &= \operatorname*{argmin}_{y\in[n]} (\mathbf{L}^\top \widetilde{\boldsymbol{\eta}}(x)) \\
&= \operatorname*{argmin}_{y\in[n]} (\mathbf{L}^{0\text{-}1} (\mathbf{C}^\top)^{-1} \widetilde{\boldsymbol{\eta}}(x)) \\
&= \operatorname*{argmin}_{y\in[n]} (\mathbf{L}^{0\text{-}1} \boldsymbol{\eta}(x)) \quad \text{(by Eq. (5.60))} \\
&= h_D^{\mathbf{L}^{0\text{-}1},*}(x)\,. \qquad (5.61)
\end{aligned}$$

□

Let $\mathbf{L} = \mathbf{C}^{-1}\mathbf{L}^{0\text{-}1}$. Let $\psi : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ be a surrogate loss for $\mathbf{L}^{0\text{-}1}$. Define $\mathbf{M} = \boldsymbol{\beta} \cdot \mathbf{1}_n^\top - \alpha\mathbf{L}$ by choosing $\alpha > 0$ and $\boldsymbol{\beta}$ so that $\mathbf{M}$ is non-negative. Then we define the corresponding $\mathbf{M}$-weighted surrogate loss $\psi^{\mathbf{M}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ as

$$\psi^{\mathbf{M}}(\mathbf{u}, y) = \sum_{j=1}^{n} m_{y,j}\psi(\mathbf{u}, j)\,. \qquad (5.62)$$

By Theorem 5.6, if ψ is $\mathbf{L}^{0\text{-}1}$-calibrated, then $\psi^{\mathbf{M}}$ is $\mathbf{L}$-calibrated. Chaining with Proposition 5.8, it means that optimizing $\psi^{\mathbf{M}}$ over the noisy training sample $\widetilde{S}$ can output a classifier whose performance converges to Bayes optimal w.r.t. the clean distribution D as the size of noisy training sample goes to infinity. We formalize this result in a theorem below.

Theorem 5.9. *Let $\psi : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ be a surrogate loss. Let $\mathbf{C}$ be the noise matrix. Define $\mathbf{L} = \mathbf{C}^{-1}\mathbf{L}^{0\text{-}1}$, and define $\mathbf{M} = \boldsymbol{\beta} \cdot \mathbf{1}_n^\top - \alpha\mathbf{L}$, $\alpha > 0$, where $\mathbf{1}_n \in \mathbb{R}^n$ is a vector with all entries equal to one, and $\boldsymbol{\beta}$ is such that*

$$\beta_y = \alpha \cdot \max_j \ell_{y,j}\,.$$

Then,

$$m_{y,\widehat{y}} = \beta_y - \alpha \ell_{y,\widehat{y}} \geq 0\,.$$

So all entries of $\mathbf{M}$ *are non-negative. Define the weighted surrogate loss* $\psi^{\mathbf{M}} : \mathbb{R}^n \times \mathcal{Y} \to \mathbb{R}_+$ *as*

$$\psi^{\mathbf{M}}(\mathbf{u}, y) = \sum_{j=1}^{n} m_{y,j}\psi(\mathbf{u}, j)\,.$$

If ψ *is* $\mathbf{L}^{0\text{-}1}$*-calibrated, then there exists an invertible, nondecreasing transformation* $\xi : \mathbb{R}_+ \to \mathbb{R}_+$ *with* $\xi(0) = 0$ *such that for all clean probability distribution* D *on* $\mathcal{X} \times [n]$ *and the corresponding noisy probability distribution* $\widetilde{D}$ *obtained using the noise matrix* $\mathbf{C}$*, and all function* $\mathbf{f} : \mathcal{X} \to \mathbb{R}^n$*:*

$$\text{regret}_{D}^{\mathbf{L}^{0\text{-}1}}[\text{argmax} \circ \mathbf{f}] \leq \xi\Big(\text{regret}_{\widetilde{D}}^{\psi^{\mathbf{M}}}[\mathbf{f}]\Big)\,. \tag{5.63}$$

Proof. Since ψ is $\mathbf{L}^{0\text{-}1}$-calibrated, by Theorem 5.6, $\psi^{\mathbf{M}}$ is $\mathbf{L}$-calibrated. By Zhang (2004b); Tewari and Bartlett (2007); Steinwart (2007), there exists an invertible, nondecreasing transformation $\xi : \mathbb{R}_+ \to \mathbb{R}_+$ with $\xi(0) = 0$ such that for all clean probability distribution D on $\mathcal{X} \times [n]$ and the corresponding noisy probability distribution $\widetilde{D}$ obtained using the noise matrix $\mathbf{C}$, and all function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^n$:

$$\text{regret}_{\widetilde{D}}^{\mathbf{L}}[\text{argmax} \circ \mathbf{f}] \leq \xi\Big(\text{regret}_{\widetilde{D}}^{\psi^{\mathbf{M}}}[\mathbf{f}]\Big)\,. \tag{5.64}$$

Let $h : \mathcal{X} \to [n]$ be a classifier obtained from $\mathbf{f}$ by $h = \operatorname{argmax} \circ \mathbf{f}$. Then

$$\begin{aligned}
\mathrm{er}_{\widetilde{D}}^{\mathbf{L}}[h] &= \mathbf{E}_{(X,\widetilde{Y})\sim\widetilde{D}}\left[\ell_{\widetilde{Y},h(X)}\right] \\
&= \mathbf{E}_X \mathbf{E}_{\widetilde{Y}\sim\widetilde{\boldsymbol{\eta}}(X)}\left[\ell_{\widetilde{Y},h(X)}\right] \\
&= \mathbf{E}_X\left[(\mathbf{L}^\top \widetilde{\boldsymbol{\eta}}(X))_{h(X)}\right] \\
&= \mathbf{E}_X\left[(\mathbf{L}^{0\text{-}1}(\mathbf{C}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(X))_{h(X)}\right] \quad \text{(by Eq. (5.60))} \\
&= \mathbf{E}_X\left[(\mathbf{L}^{0\text{-}1}\boldsymbol{\eta}(X))_{h(X)}\right] \\
&= \mathbf{E}_X \mathbf{E}_{Y\sim\boldsymbol{\eta}(X)}\left[\ell_{Y,h(X)}\right] \\
&= \mathbf{E}_{(X,Y)\sim D}\left[\ell_{Y,h(X)}\right] \\
&= \mathrm{er}_{D}^{\mathbf{L}^{0\text{-}1}}[h]\,.
\end{aligned} \tag{5.65}$$

So, it follows that

$$\mathrm{regret}_{\widetilde{D}}^{\mathbf{L}}[h] = \mathrm{regret}_{D}^{\mathbf{L}^{0\text{-}1}}[h]\,. \tag{5.66}$$

And we have

$$\mathrm{regret}_{D}^{\mathbf{L}^{0\text{-}1}}[\operatorname{argmax}\circ\mathbf{f}] = \mathrm{regret}_{\widetilde{D}}^{\mathbf{L}}[\operatorname{argmax}\circ\mathbf{f}] \le \xi\left(\mathrm{regret}_{\widetilde{D}}^{\psi^{\mathbf{M}}}[\mathbf{f}]\right), \tag{5.67}$$

as claimed. □

In the example below, we apply the method above to learn from noisy labels in binary classification. Our method recovers results in Natarajan et al. (2013), showing that the method covers the weighted loss method in Natarajan et al. (2013) as a special case.

Example 5.8. *The noise matrix* $\mathbf{C}$ *is binary classification with labels* $-1, +1$ *can be written as* *follows:*

$$\mathbf{C} = \begin{array}{c} \\ +1 \\ -1 \end{array} \begin{array}{c} \begin{array}{cc} +1 & -1 \end{array} \\ \begin{bmatrix} 1-\rho_{+1} & \rho_{+1} \\ \rho_{-1} & 1-\rho_{-1} \end{bmatrix} \end{array},$$

Then, we have

$$\begin{aligned} \mathbf{L} &= \mathbf{C}^{-1}\mathbf{L}^{\text{0-1}} \\ &= \frac{1}{1-\rho_{+1}-\rho_{-1}} \begin{bmatrix} -\rho_{+1} & 1-\rho_{-1} \\ 1-\rho_{+1} & -\rho_{-1} \end{bmatrix}. \end{aligned}$$

Note that $1-\rho_{+1}-\rho_{-1} > 0$ *since* $\rho_{+1}+\rho_{-1} < 1$. *Define*

$$\mathbf{M} = \begin{bmatrix} \frac{1-\rho_{-1}}{2} & \frac{1-\rho_{-1}}{2} \\ \frac{1-\rho_{+1}}{2} & \frac{1-\rho_{+1}}{2} \end{bmatrix} - \frac{1-\rho_{+1}-\rho_{-1}}{2}\mathbf{L} = \begin{bmatrix} \frac{1-\rho_{-1}+\rho_{+1}}{2} & 0 \\ 0 & \frac{1-\rho_{+1}+\rho_{-1}}{2} \end{bmatrix}$$

according to Eq. (5.23)*. Clearly, entries of* $\mathbf{M}$ *are non-negative. Comparing with Proposition 5.4, Theorem 5.2 and Theorem 5.3, we recognize that this choice of* $\mathbf{M}$ *recovers the weighted loss method for binary learning from noisy labels with*

$$\alpha^* = \frac{1-\rho_{+1}+\rho_{-1}}{2}.$$

5.5. Weighted Loss Method for Multiclass Learning with a Reject Option

In some multiclass learning settings, a classifier is required to refrain from making a prediction to avoid costly misclassification errors when encountering examples that are difficult to classify (i.e., when the confidence of classification of an example is low). Such settings are termed 'classification with a reject option', and have been studied in standard binary and multiclass classification problems (Chow, 1970; Yuan and Wegkamp, 2010; El-Yaniv and Wiener, 2010; Bartlett and Wegkamp, 2008; Cortes et al., 2016a,b; Geifman and El-Yaniv, 2017; Ramaswamy et al., 2018; Ni et al., 2019;

Shen et al., 2020; Charoenphakdee et al., 2021; Cao et al., 2022).

In this section, we show how to apply weighted surrogate losses in Section 5.3 for multiclass learning with a reject option.

5.5.1. Learning with a Reject Option

The problem setting of learning with a reject option is an extension of the standard classification setting. There is an instance space $\mathcal{X}$, and a set of n class labels $\mathcal{Y}$, which we will take without loss of generality to be $\mathcal{Y} = [n]$. There is a (unknown) joint probability distribution D on $\mathcal{X} \times \mathcal{Y}$ from which labeled examples (X, Y) are drawn. The prediction space $\widehat{\mathcal{Y}}$, however, consists of labels in $\mathcal{Y}$ and a 'reject' option. Without loss of generality, we assume the reject option is $n+1$, so $\widehat{\mathcal{Y}} = \mathcal{Y} \cup \{n+1\} = [n+1]$. The goal is to learn a classifier $h : \mathcal{X} \to [n+1]$ the performs well w.r.t. the performance measure. The commonly used performance measure in learning with a reject option is the 0-1-c loss, defined as follows:

$$\ell^{\text{0-1-c}}(y, \widehat{y}) = \begin{cases} c, & \widehat{y} = n+1 \\ \mathbf{1}(\widehat{y} \neq y), & \widehat{y} \in [n] \end{cases}, \tag{5.68}$$

where $c \in (0,1)$. That is, the cost of abstention should be higher than that of a correct prediction, but should be lower than that of a wrong prediction. We will denote by $\boldsymbol{\eta} : \mathcal{X} \to \Delta_n$ the (vector) class probability functions under distribution D, with components given by

$$\eta_y(x) \quad = \quad \mathbf{P}(Y = y \,|\, X = x)\,,$$

for each $y \in [n]$. It is well-known that Chow's rule (Chow, 1970) characterizes the optimal solution of this problem.

Definition 5.8 (Chow's rule). *A classifier $h : \mathcal{X} \to [n+1]$ is Bayes optimal if and only if the* *following condition holds almost surely:*

$$h(x) = \begin{cases} n+1, & \max_{y\in[n]} \eta_y(x) \le 1-c \\ \operatorname{argmax}_{y\in[n]} \eta_y(x), & \textit{otherwise} \end{cases} . \tag{5.69}$$

An interpretation of Chow's rule is that an optimal classifier should refrain from making a prediction for an instance if its most confident prediction is still not confident enough. We can see that $1-c$ is a threshold, and can be defined according to the nature of the problem at hand.

5.5.2. Weighted Loss Method for Learning with a Reject Option

In this section, we apply weighted surrogate losses for multiclass learning with a reject option. We start by writing the loss function $\ell^{\text{0-1-c}}$ (Eq. (5.68)) in a matrix form. Define $\mathbf{L} \in \mathbb{R}^{n\times(n+1)}$ as follows:

$$\ell_{y,\widehat{y}} = \ell^{\text{0-1-c}}(y, \widehat{y}) . \tag{5.70}$$

Let $\mathbf{1}_n \in \mathbb{R}^n$ be a vector with all entries equal to 1. Then we can write $\mathbf{L}$ as

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}^{\text{0-1}} & c \cdot \mathbf{1}_n \end{bmatrix} . \tag{5.71}$$

The Bayes optimal classifier for $\mathbf{L}$ (which for any instance x, chooses a prediction that minimizes the expected loss under $\mathbf{L}$) is given by

$$\begin{aligned} h_D^{\mathbf{L},*}(x) &= \operatorname*{argmin}_{y\in[n+1]} (\mathbf{L}^\top \boldsymbol{\eta}(x))_y \\ &= \begin{cases} \operatorname{argmin}_{y\in[n]} ((\mathbf{L}^{\text{0-1}})^\top \boldsymbol{\eta}(x))_y , & \min_{y'\in[n]} ((\mathbf{L}^{\text{0-1}})^\top \boldsymbol{\eta}(x))_{y'} < c\mathbf{1}_n^\top \boldsymbol{\eta}(x) = c \\ n+1 , & \text{otherwise} \end{cases} , \\ &= \begin{cases} \operatorname{argmax}_{y\in[n]} \eta_y(x) , & \max_{y'\in[n]} \eta_{y'}(x) > 1-c \\ n+1 , & \text{otherwise} \end{cases} . \end{aligned} \tag{5.72}$$

We can see that $h_D^{\mathbf{L},*}$ satisfies Chow's rule (Definition 5.8).

Let $\mathbf{L}$ be the cost-sensitive loss defined in Eq. (5.70). We apply weighted surrogate losses described in Section 5.3 to $\mathbf{L}$.

According to Eq. (5.23). we define $\mathbf{M} \in \mathbb{R}_+^{n\times(n+1)}$ as

$$\begin{aligned}\mathbf{M} &= \mathbf{1}_n \mathbf{1}_{n+1}^\top - \mathbf{L} \\ &= \begin{bmatrix} \mathbf{I}_n & (1-c)\cdot\mathbf{1}_n \end{bmatrix}, \end{aligned} \tag{5.73}$$

where $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix, and $\mathbf{1}_n \in \mathbb{R}^n$ is a vector whose entries are all 1.

Let $\psi : \mathbb{R}^{n+1} \times \mathcal{Y} \to \mathbb{R}_+$ be a surrogate loss. Then we define the $\mathbf{M}$-weighted surrogate loss $\psi^{\mathbf{M}} : \mathbb{R}^{n+1} \times \mathcal{Y} \to \mathbb{R}_+$ as

$$\begin{aligned}\psi^{\mathbf{M}}(\mathbf{u}, y) &= \sum_{j=1}^{n+1} m_{y,j} \cdot \psi(\mathbf{u}, j) \\ &= \psi(\mathbf{u}, y) + (1-c)\psi(\mathbf{u}, n+1)\,. \end{aligned} \tag{5.74}$$

Remark. The weighted loss $\psi^{\mathbf{M}}$ is exactly the surrogate loss proposed in Cao et al. (2022) (Definition 5), showing that the weighted loss method covers the method proposed in Cao et al. (2022) as a special case in learning with a reject option.

Consistency and estimation error bound. With Theorem 5.6, if $\psi : \mathbb{R}^{n+1} \times \mathcal{Y} \to \mathbb{R}_+$ is classification-calibrated, then $\psi^{\mathbf{M}}$ is consistent for $\mathbf{L}$. This result is consistent with Theorem 4 in Cao et al. (2022). In addition, with Theorem 5.7, we can also recover the estimation error bound shown in Theorem 3 of Cao et al. (2022).

5.6. Conclusion

In this work, we have designed a surrogate loss for a general multiclass loss $\mathbf{L}$ by taking a weighted combination of surrogate losses for the standard 0-1 loss, closing an open problem. Our method works with both smooth surrogate losses and non-smooth surrogate losses, allowing for margin-based losses. The proposed method also preserves the convexity of the underlying surrogate loss. We have provided theoretical results to show the proposed method is Bayes consistent, and provided an estimation error bound. Moreover, we have applied the proposed method to extend the weighted loss method proposed in Natarajan et al. (2013) for binary learning from noisy labels to multiclass learning from noisy labels. Finally, we have also applied the proposed method to solve problems in multiclass learning with a reject option, recovering several results of Cao et al. (2022).

CHAPTER 6

COMPLEX LABEL SPACE AND COMPLEX LEARNING SETTING: CONSISTENT MULTI-LABEL LEARNING FROM NOISY LABELS

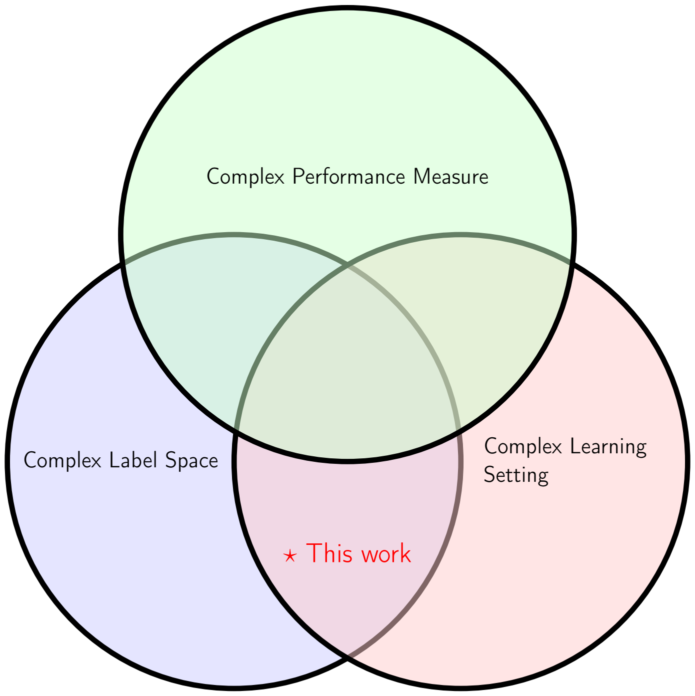

Figure 6.1: Position of *Consistent Multi-Label Learning from Noisy Labels* in the thesis.

In this chapter, we move to begin our discussion of the intersection between complex label spaces and complex label settings. In particular, we show how to design consistent noise-corrected algorithms for multi-label learning from noisy labels.

6.1. Background of Complex Label Space and Learning Setting

In previous chapters, we have seen both complex label spaces and complex learning settings. In real world, there are problems that pertain both complexities. Such problems require tackling both complexities, necessitating a broad toolbox of methods and algorithms. Here are a few examples of real-world machine learning problems that involve both complex label spaces and complex learning settings:

Automated Medical Diagnosis: In healthcare, developing machine learning models for medical diagnosis can involve predicting multiple related diseases for a patient based on their symptoms, medical history, and test results. This is a multi-label classification problem as a patient could have multiple diseases at the same time. In addition, the learning setting is also complex due to the presence of noisy labels (diagnosis errors or inconsistencies across different doctors), missing labels (unavailable test results), and imbalanced data (rare diseases have fewer instances in the dataset).

Natural Language Processing: In natural language processing, sequence labeling tasks such as named entity recognition (NER) and part-of-speech (POS) tagging involve assigning a label to each word in a sentence, which is a complex label space scenario. Moreover, these tasks often involve transfer learning, where models pre-trained on large corpora are fine-tuned for the specific sequence labeling task.

Multi-modal Emotion Recognition: In this task, the goal is to recognize the emotional state of a person using multi-modal data such as text, audio, and video. This task involves a complex label space if the goal is to predict multiple emotion labels for each instance. It could also involve complex learning settings such as semi-supervised learning (when there is a lack of fully labeled data), and transfer learning (when knowledge is transferred from one modality to another).

Self-driving Cars: Autonomous vehicles need to interpret sensory input (like camera feeds, LIDAR, and RADAR data) to identify and classify objects around them (cars, pedestrians, bicycles, etc.). This is a complex label space problem, particularly in the case of image segmentation tasks that assign labels to each pixel in the image. The learning setting is also complex, involving online learning (constantly adapting to new data), reinforcement learning (learning to make decisions based on rewards), and transfer learning (applying pre-trained models to new tasks).

Recommendation Systems: Modern recommendation systems aim to provide personalized recommendations by predicting a list of items (like movies, songs, products) that a user would like, which represents a complex label space. In addition, these systems often operate in complex learning settings. For instance, they may use semi-supervised learning when labeled data (user feedback) is scarce, active learning to intelligently decide which items to recommend to collect valuable feedback,

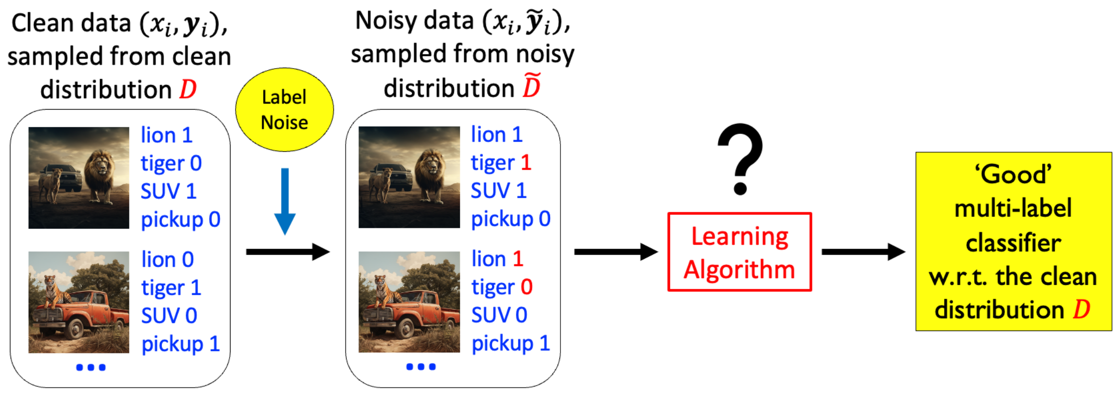
Clean data $(x_i, \mathbf{y}_i)$, sampled from clean distribution D
lion 1
tiger 0
SUV 1
pickup 0
lion 0
tiger 1
SUV 0
pickup 1
Label Noise
Noisy data $(x_i, \tilde{\mathbf{y}}_i)$, sampled from noisy distribution $\tilde{D}$
lion 1
tiger 1
SUV 1
pickup 0
lion 1
tiger 0
SUV 0
pickup 1
?
Learning Algorithm
'Good' multi-label classifier w.r.t. the clean distribution D

Figure 6.2: Multi-label learning from noisy labels. (Images in the left boxes were generated with the help of Midjourney.)

and online learning to adapt to the evolving tastes of users and the introduction of new items.

6.2. Introduction

6.2.1. Background and Our Contributions

In many applications of machine learning, accurate labels are difficult or expensive to obtain; therefore, in practice, one often receives noisy labels. This problem is even more pronounced in multi-label classification (MLC) settings, where multiple labels/tags can be active in an instance simultaneously. In recent years, there has been much interest in developing learning algorithms that can learn good classifiers from data with noisy labels (Frénay and Verleysen, 2014; Han et al., 2020; Song et al., 2023). While there has been much work in this area for binary and multiclass problems, there has been relatively limited work on multi-label learning from noisy labels. In this work, we develop principled noise-corrected multi-label learning algorithms for a variety of performance measures under the general class-conditional noise (CCN) model.

The key challenge in learning from noisy labels is to develop algorithms that can produce accurate classifiers for the true/clean distribution despite noisy labels; in particular, a desirable goal is that the algorithms should be *(Bayes) consistent*, meaning that as the size of the (noisy) training sample increases, the performance of the learned classifier converges to the Bayes optimal performance under the clean (non-noisy) distribution (see Figure 6.2). For binary and multiclass learning, many such consistent algorithms have been designed under CCN and related noise models (Natarajan et al., 2013; Scott et al., 2013; Scott, 2015; Menon et al., 2015; Liu and Tao, 2016; Patrini et al., 2016; Ghosh et al., 2017; van Rooyen and Williamson, 2017; Patrini et al., 2017; Natarajan et al., 2017; Wang et al., 2018; Xia et al., 2019; Liu and Guo, 2020; Zhang et al., 2021; Li et al., 2021; Zhang and Agarwal, 2024). However, for multi-label learning, only a few consistent noise-corrected algorithms have been designed, and the consistency guarantees that do currently exist are for specific performance measures under the relatively simple independent flipping noise (IFN) model (a special case of CCN) that fails to capture correlations among tags (Kumar et al., 2020; Xie and Huang, 2023). In this work, we develop provably Bayes consistent noise-corrected multi-label algorithms for a broad family of multi-label performance measures under the general CCN model.

Our contributions include the following (see also Figure 6.3):

1. Algorithms. We provide the following three consistent noise-corrected multi-label algorithms, all of which work by identifying a small set of what we call 'Bayes-sufficient' statistics for the target loss and estimating these reliably from the given noisy training sample:

- *Noise-Corrected Plug-in* (NCPLUG) algorithm for Hamming loss under IFN;
- *Noise-Corrected Exact F-measure Plug-in* (NCEFP) algorithm for multi-label F_1-measure under general CCN;
- *Noise-Corrected Output Coding* (NCOC) algorithm for general low-rank multi-label losses under general CCN.

2. Regret transfer bounds and consistency. For all these algorithms, we provide quantitative regret transfer bounds to establish consistency. The bounds suggest that as the amount of label noise increases, the (noisy) sample size needed to reach a given level of performance generally increases.

3. Similar-Tag Switching Noise (STSN) model. While our NCEFP and NCOC algorithms are provably consistent under general CCN models, in multi-label settings, general CCN models involve extremely large noise matrices that make computation prohibitive. We propose a new family of

Noise model	Multi-label losses	Noise-corrected algorithms
Symmetric IFN	Hamming	Kumar et al. (2020)
IFN	Hamming, Ranking	CCMN (Xie and Huang, 2023)
IFN	Hamming	NCPLUG (this work)
CCN	F_1-measure	NCEFP (this work)
CCN	General low-rank $\mathbf{L}$	NCOC (this work)

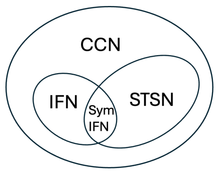

Figure 6.3: Summary of consistent noise-corrected multi-label algorithms and associated noise models.

structured multi-label noise models that we term *Similar-Tag Switching Noise* (STSN) models. STSN models are a special case of CCN that require fewer parameters and enable fast computation for NCEFP and NCOC; moreover, unlike IFN models, they also capture some correlations among tags.

4. Experimental validation. Finally, we evaluate our algorithms on both synthetic and real data.

6.2.2. Notation

For an integer n, we denote by $[n]$ the set of integers $\{1, \dots, n\}$, and by Δ_n the probability simplex $\{\mathbf{p} \in \mathbb{R}_+^n : \sum_{y=1}^n p_y = 1\}$. For a vector $\mathbf{a}$, we denote by $\|\mathbf{a}\|_p$ the p-norm of $\mathbf{a}$, and by a_j the j-indexed entry of $\mathbf{a}$. For a matrix $\mathbf{A}$, we denote by $\|\mathbf{A}\|_p$ the induced p-norm of $\mathbf{A}$, by $\mathbf{a}_y$ the y-indexed column vector of $\mathbf{A}$, and by $a_{j,y}$ the (j, y)-indexed element of $\mathbf{A}$. We use $\mathbf{1}(\cdot)$ to denote the indicator function and $\xrightarrow{p}$ to denote convergence in probability.

6.2.3. Related Work

Below we briefly review some works that are most closely related to our study.

• **Consistent algorithms for standard (non-noisy) multi-label learning.** Bayes optimal multi-label classifiers and consistent algorithms for MLC performance measures, including Hamming loss and F_1-measure, have been studied by Dembczynski et al. (2010a, 2011); Gao and Zhou (2013); Dembczynski et al. (2013); Menon et al. (2019); Zhang et al. (2020); Wu and Zhu (2020); Wu et al. (2021) and others. These works do not deal with noisy labels. A detailed survey on multi-label learning can be found in Zhang and Zhou (2014).

- **Consistent algorithms for binary/multiclass learning from noisy labels for CCN models.** Many consistent noise-corrected algorithms have been designed for binary/multiclass learning (Natarajan et al., 2013; Scott et al., 2013; Scott, 2015; Menon et al., 2015; Liu and Tao, 2016; Patrini et al., 2016; Ghosh et al., 2017; van Rooyen and Williamson, 2017; Patrini et al., 2017; Natarajan et al., 2017; Wang et al., 2018; Xia et al., 2019; Liu and Guo, 2020; Zhang et al., 2021; Li et al., 2021; Zhang and Agarwal, 2024). But when applied to multi-label problems in a straightforward way (by treating each label vector as a class), these algorithms need exponential (in the number of tags) number of parameters and suffer from slow computation. In essence, they are not designed for multi-label problems.

- **Multi-label learning from noisy labels.** Two types of noise models have been studied: statistical and non-statistical. For statistical noise models, Kumar et al. (2020) studied a special 'symmetric' case of IFN and focused on loss functions satisfying certain conditions (e.g., Hamming loss). Xie and Huang (2023) showed consistent algorithms for Hamming and Ranking losses under IFN. Li et al. (2022) proposed a way to estimate noise matrices under IFN. For non-statistical noise models, partial multi-label learning (PML) is a prominent example, where for each instance, its noisy label contains all active tags from the clean label, as well as some non-active tags. Some algorithms have been proposed to deal with PML (Xie and Huang, 2018; Fang and Zhang, 2019; Wang et al., 2019; Sun et al., 2019; Xie and Huang, 2020); however it is unclear whether a Bayes optimal classifier can be recovered under PML. Other empirical studies of multi-label learning from noisy labels include (Veit et al., 2017; Hu et al., 2018; Bai et al., 2020; Zhao and Gomes, 2021).

6.2.4. Organization

After preliminaries and background in Section 6.3, we provide intuition for our algorithms in Section 6.4, followed by our three noise-corrected algorithms in Section 6.5. Section 6.6 gives regret transfer bounds for our algorithms. Section 6.7 describes the STSN model. Section 6.8 summarizes our experiments. Finally, Section 6.9 concludes this work.

Multi-label classification (MLC) with noisy labels. In an MLC problem, there is an instance space $\mathcal{X}$, and a set of s tags $\mathcal{T} = [s] := \{1, \ldots, s\}$ that can be associated with each instance in $\mathcal{X}$. For example, in image tagging, $\mathcal{X}$ is the set of possible images, and $\mathcal{T}$ is a set of s pre-defined tags (such as `sky`, `cloud`, `water` etc.) that can be associated with each image. The label space $\mathcal{Y} \subseteq \{0,1\}^s$ consists of label vectors $\mathbf{y} \in \{0,1\}^s$ that indicate which of the s tags are active (specifically, $y_j = 1$ denotes that tag j is active, and $y_j = 0$ denotes it is inactive). Let $|\mathcal{Y}|$ denote the size of $\mathcal{Y}$.

There is a (unknown) joint probability distribution D on $\mathcal{X} \times \mathcal{Y}$ from which labeled examples $(X, \mathbf{Y})$ are drawn. In the standard (non-noisy) MLC problem, the learner would be given training examples drawn directly from D. However, when learning from noisy labels, the learner instead sees only noisy examples $(X, \widetilde{\mathbf{Y}})$, where $\widetilde{\mathbf{Y}}$ is a noisy version of $\mathbf{Y}$. Given a noisy training sample $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, the goal is to learn a multi-label classifier $\mathbf{h} : \mathcal{X} \rightarrow \mathcal{Y}$ that performs well with respect to the clean distribution D.

Class-conditional noise (CCN). The label noise models we consider here belong to the well-known CCN model that has been widely studied in binary and multiclass learning from noisy labels (Natarajan et al., 2013; van Rooyen and Williamson, 2017; Patrini et al., 2017), in which a label $\mathbf{y}$ is randomly flipped to a label $\widetilde{\mathbf{y}}$ with some probability $c_{\mathbf{y},\widetilde{\mathbf{y}}}$ that depends on $\mathbf{y}$ and $\widetilde{\mathbf{y}}$, but not on the features. Specifically, the CCN model is characterized by a row-stochastic noise matrix $\mathbf{C} \in [0,1]^{|\mathcal{Y}| \times |\mathcal{Y}|}$ with entries $c_{\mathbf{y},\widetilde{\mathbf{y}}}$, such that for each $\mathbf{y}, \widetilde{\mathbf{y}} \in \mathcal{Y}$, $c_{\mathbf{y},\widetilde{\mathbf{y}}} = \mathbf{P}(\widetilde{\mathbf{Y}} = \widetilde{\mathbf{y}} \,|\, \mathbf{Y} = \mathbf{y})$. The noisy training examples can therefore be viewed as being drawn i.i.d. from a 'noisy' distribution $\widetilde{D}$ on $\mathcal{X} \times \mathcal{Y}$: an example $(X, \mathbf{Y})$ is first drawn randomly according to D, and then noise is injected according to the noise matrix $\mathbf{C}$ to produce $(X, \widetilde{\mathbf{Y}})$. In MLC, $|\mathcal{Y}|$ can be as large as 2^s; therefore, a fully general noise matrix $\mathbf{C}$ requires too many parameters (exponential in s). This necessitates considering more structured noise models well-suited to MLC problems.

Independent flipping noise (IFN). So far, most previous work has considered only the very simple multi-label IFN model (a special case of CCN) where each tag is flipped independently from active to inactive or vice versa; this involves only $2s$ parameters defined as $c_{0,1}^{(j)} = \mathbf{P}(\widetilde{Y}_j = 1|Y_j = 0)$ and $c_{1,0}^{(j)} = \mathbf{P}(\widetilde{Y}_j = 0|Y_j = 1)$, $\forall j \in [s]$.

Multi-label performance measures/loss matrices $\mathbf{L}$. We will consider multi-label loss matrices of the form $\mathbf{L} \in \mathbb{R}_+^{|\mathcal{Y}|\times|\mathcal{Y}|}$, with entries $\ell_{\mathbf{y},\widehat{\mathbf{y}}}$ indicating the loss incurred on predicting $\widehat{\mathbf{y}}$ when the clean label is $\mathbf{y}$. Two specific examples we will use throughout the paper are the following:

- **(Normalized) Hamming loss $\mathbf{L}^{\mathbf{Ham}}$:** $\ell_{\mathbf{y},\widehat{\mathbf{y}}}^{\mathrm{Ham}} = \frac{1}{s}\sum_{j=1}^{s} \mathbf{1}(\widehat{y}_j \neq y_j)\,,$ (6.1)

- **F_1-measure $\mathbf{L}^{F_1}$ (specified as a loss (Dembczynski et al., 2013; Zhang et al., 2020)):**

$$\ell_{\mathbf{y},\widehat{\mathbf{y}}}^{F_1} = 1 - \frac{2\sum_{j=1}^{s} y_j \widehat{y}_j}{\|\mathbf{y}\|_1 + \|\widehat{\mathbf{y}}\|_1}\,,\ \text{where we take } \frac{0}{0} = 1\,. \tag{6.2}$$

L-generalization error, L-regret, and Bayes consistency. Given a multi-label loss matrix $\mathbf{L}$, the $\mathbf{L}$-*generalization* error of a multi-label classifier $\mathbf{h} : \mathcal{X}\to\mathcal{Y}$ under the clean distribution D is defined as $\mathrm{er}_D^{\mathbf{L}}[\mathbf{h}] = \mathbf{E}_{(X,\mathbf{Y})\sim D}[\ell_{\mathbf{Y},\mathbf{h}(X)}]$, its $\mathbf{L}$-*regret* is defined as $\mathrm{regret}_D^{\mathbf{L}}[\mathbf{h}] = \mathrm{er}_D^{\mathbf{L}}[\mathbf{h}] - \inf_{\mathbf{h}':\mathcal{X}\to\mathcal{Y}} \mathrm{er}_D^{\mathbf{L}}[\mathbf{h}']$. We will say a noise-corrected algorithm that maps a noisy training sample $\widetilde{S}$ to a classifier $\widehat{\mathbf{h}}$ is *Bayes consistent* for $\mathbf{L}$ under D if $\mathrm{regret}_D^{\mathbf{L}}[\widehat{\mathbf{h}}] \xrightarrow{p} 0$ as the noisy sample size $m \to \infty$.

Multi-label class probability functions. We will denote by $\boldsymbol{\eta}, \widetilde{\boldsymbol{\eta}} : \mathcal{X}\to\Delta_{|\mathcal{Y}|}$ the multi-label class probability functions associated with the clean distribution D and the noisy distribution $\widetilde{D}$, respectively, defined as $\eta_{\mathbf{y}}(x) = \mathbf{P}(\mathbf{Y} = \mathbf{y}|X = x)$ and $\widetilde{\eta}_{\mathbf{y}}(x) = \mathbf{P}(\widetilde{\mathbf{Y}} = \mathbf{y}|X = x)$.

Binary and multiclass class probability estimation (CPE), and logistic losses. Our multi-label algorithms will involve solving various binary and multiclass CPE sub-problems, which in turn involve estimating the class probability functions associated with the corresponding binary/multi-class problems. For binary CPE problems, we will use the binary logistic loss $\phi_{\log} : \{0,1\} \times \mathbb{R}\to\mathbb{R}_+$ and associated inverse link function $\gamma_{\log}^{-1} : \mathbb{R}\to[0,1]$ defined by $\phi_{\log}(y,u) = \ln(1 + e^{-(2y-1)u})$ and $\gamma_{\log}^{-1}(u) = \frac{1}{1+\exp(-u)}$, respectively; similarly for n-class CPE problems, we will use the multiclass logistic loss $\phi_{\mathrm{mlog}} : [n] \times \mathbb{R}^{n-1} \to \mathbb{R}_+$ and associated inverse link function $\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1} : \mathbb{R}^{n-1} \to \Delta_n$

defined by $\phi_{\text{mlog}}(y, \mathbf{u}) = -\ln\big(\frac{\exp(u_y)}{1+\sum_{i=1}^{n-1}\exp(u_i)}\big)$ if $y \in [n-1]$ and $\ln\big(1+\sum_{i=1}^{n-1}\exp(u_i)\big)$ if $y = n$, and $(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(\mathbf{u}))_y = \frac{\exp(u_y)}{1+\sum_{i=1}^{n-1}\exp(u_i)}$ if $y \in [n-1]$ and $\frac{1}{1+\sum_{i=1}^{n-1}\exp(u_i)}$ if $y = n$, respectively.

6.4. Key Ideas and Intuition

Bayes optimal classifier for L and 'Bayes-sufficient' statistics $\mathbf{q}(x)$. Given a multi-label loss matrix $\mathbf{L}$, the Bayes optimal classifier for $\mathbf{L}$ under the clean distribution D (i.e., the classifier with smallest $\mathbf{L}$-generalization error under D) is given by

$$\mathbf{h}^*(x) \in \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \mathcal{Y}} \boldsymbol{\eta}(x)^\top \boldsymbol{\ell}_{\widehat{\mathbf{y}}}\,.$$

Our goal will be to construct an approximation to $\mathbf{h}^*$ from $\widetilde{S}$. There are two main challenges: (1) for multi-label problems, $\boldsymbol{\eta}(x) \in \Delta_{|\mathcal{Y}|}$ is potentially a very large vector; (2) we have access to only the noisy training sample $\widetilde{S}$. In order to overcome these challenges, the key ideas in all our algorithms will be to (1) identify a small set of statistics $\mathbf{q}(x)$ of the class probability vector $\boldsymbol{\eta}(x)$ – which we will refer to as 'Bayes-sufficient' statistics for $\mathbf{L}$ – that suffice to construct the Bayes optimal classifier for $\mathbf{L}$; and (2) estimate the statistics $\mathbf{q}(x)$ reliably from the given noisy training sample $\widetilde{S}$.

6.5. Algorithms

6.5.1. Hamming Loss under IFN: NCPLUG Algorithm

Let us start with the simplest case: Hamming loss under the independent flipping noise (IFN) model. This is also the main setting for which consistent noise-corrected algorithms have previously been developed (Kumar et al., 2020; Xie and Huang, 2023). Under this setting, the loss and noise model both involve independent components for the s tags, and the problem reduces to solving s independent binary noisy label problems, one for each tag; indeed, the CCMN algorithm of Xie and Huang (2023) essentially solves each of these binary problems using the binary unbiased estimator method of Natarajan et al. (2013). Our *Noise-Corrected Plug-in* (NCPLUG) algorithm will also solve s independent binary problems, but will do so in a way that illustrates in this simple setting the essence of the approach that we will build on for more complex settings later.

Under the Hamming loss, the Bayes optimal classifier requires only the s Bayes-sufficient statistics

$$q_j(x) = \mathbf{P}(Y_j = 1|x) \quad \forall j \in [s]\,.$$

Indeed, the Bayes optimal classifier for $\mathbf{L}^{\mathrm{Ham}}$ can be written as

$$h_j^*(x) = \mathbf{1}(q_j(x) \geq \tfrac{1}{2}) \quad \forall j \in [s]\,.$$

The key idea then is to estimate the statistics $q_j(x)$, associated with the clean distribution D, reliably from the noisy training sample $\widetilde{S}$. For this, we use a very simple approach. In particular, we first apply a standard binary CPE learner to $\widetilde{S}$ to obtain estimates of the statistics $q'_j(x) = \mathbf{P}(\widetilde{Y}_j = 1|x)$ associated with the noisy distribution $\widetilde{D}$. Next, under the IFN model, assuming $c_{0,1}^{(j)} + c_{1,0}^{(j)} < 1$, we have $q_j(x)$ and $q'_j(x)$ are related by $q'_j(x) = (1 - c_{1,0}^{(j)}) \cdot q_j(x) + c_{0,1}^{(j)} \cdot (1 - q_j(x))$, or equivalently $q_j(x) = \frac{q'_j(x) - c_{0,1}^{(j)}}{1 - c_{0,1}^{(j)} - c_{1,0}^{(j)}}, \forall j \in [s]$. Therefore, given $\widehat{q}'_j(x)$ estimated from $\widetilde{S}$, the multi-label classifier output by our NCPLUG algorithm is given by

$$\widehat{h}_j(x) = \mathbf{1}\Big(\frac{\widehat{q}'_j(x) - c_{0,1}^{(j)}}{1 - c_{0,1}^{(j)} - c_{1,0}^{(j)}} \geq \tfrac{1}{2}\Big) \quad \forall j \in [s]\,.$$

Estimating $\mathbf{q}'(x)$. Our implementation of NCPLUG uses a binary logistic loss minimizer for the CPE learner. In particular, we first learn a vector of s real-valued functions $\widehat{\mathbf{f}} : \mathcal{X} \to \mathbb{R}^s$ by minimizing the s-dimensional convex surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^s \to \mathbb{R}_+$ defined as

$$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^{s} \phi_{\log}(y_j, u_j)$$

over the noisy training sample $\widetilde{S}$. Specifically, $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$ for a suitable class of real-valued vector functions $\mathcal{F} \subseteq \{\mathbf{f} : \mathcal{X} \to \mathbb{R}^s\}$. The estimated statistics are then given by $\widehat{q}'_j(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$. Detailed pseudocode is in Section B.1.

6.5.2. F_1-measure under General CCN: NCEFP Algorithm

Next, we consider multi-label learning with F_1-measure under general class-conditional noise (CCN). In this section, we will build on the Exact F-measure Plug-in (EFP) algorithm of Dembczynski et al. (2013), which is consistent for multi-label F_1-measure in the non-noisy setting. We will develop a noise-corrected version of this algorithm that we will call the *Noise-Corrected Exact F-measure Plug-in* (NCEFP) algorithm. Again, our approach will be to reliably estimate suitable Bayes-sufficient statistics $\mathbf{q}(x)$ associated with the clean distribution D from the noisy training sample.

As shown in Dembczynski et al. (2013), for the multi-label F_1-measure, the following s^2+1 statistics are Bayes-sufficient:

$$q_0(x) = \mathbf{P}(\|\mathbf{Y}\|_1 = 0|x)\,; \qquad q_{jk}(x) = \mathbf{P}(Y_j = 1, \|\mathbf{Y}\|_1 = k|x) \quad \forall j,k \in [s]\,.$$

In particular, under the F_1-measure, the Bayes optimal classifier is given by

$$\mathbf{h}^*(x) \in \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \Big\{1 - q_0(x)\cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{j=1}^{s}\sum_{k=1}^{s} q_{jk}(x)\cdot \frac{2\cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1}\Big\}\,.$$

For the standard (non-noisy) setting, Dembczynski et al. (Dembczynski et al., 2013) showed how to estimate the s^2 statistics $\{q_{jk}(x) : j,k \in [s]\}$ by solving s multiclass ($(s+1)$-class) CPE problems, and statistic $q_0(x)$ by solving a binary CPE problem. Here we develop noise-corrected versions of these procedures for estimating these statistics from the noisy training sample.

Let us first define a matrix $\mathbf{A} \in [0,1]^{(s^2+1)\times|\mathcal{Y}|}$ as follows:

$$a_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)\,; \qquad a_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k)\cdot y_j \quad \forall j,k \in [s]\,.$$

Then it can be seen that the Bayes-sufficient statistics above can be written as $\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x)$. Under the general CCN model, the clean class probability function $\boldsymbol{\eta}(x)$ is related to the noisy class probability function $\widetilde{\boldsymbol{\eta}}(x)$ via $\widetilde{\boldsymbol{\eta}}(x) = \mathbf{C}^\top\boldsymbol{\eta}(x)$. Therefore, if $\mathbf{C}$ is invertible, then the desired statistics $\mathbf{q}(x)$ can be written in terms of $\widetilde{\boldsymbol{\eta}}(x)$ as $\mathbf{q}(x) = \mathbf{A}(\mathbf{C}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x) = \widetilde{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)$, where $\widetilde{\mathbf{A}} = \mathbf{A}(\mathbf{C}^\top)^{-1}$.

Now unlike the standard (non-noisy) setting, where the statistics $\mathbf{q}(x)$ expressed directly in terms of $\boldsymbol{\eta}(x)$ naturally decomposed into a set of multiclass (and one binary) CPE problems, these statistics expressed in terms of $\widetilde{\boldsymbol{\eta}}(x)$ no longer naturally decompose this way. Nevertheless, we will show how to estimate these statistics from the noisy training sample by solving s suitably weighted multiclass CPE problems together with a suitably weighted binary CPE problem. To do so, we will use a shifted and scaled matrix $\widetilde{\mathbf{A}}'$ to estimate related statistics $\mathbf{q}'(x)$ and then factor back in the scaling and shifting when using the estimated statistics to make a final prediction.[23] Towards this, define $\widetilde{a}_{\min} = \min(\min_{\mathbf{y}} \widetilde{a}_{0,\mathbf{y}}, \min_{\mathbf{y},jk} \widetilde{a}_{jk,\mathbf{y}})$ and $\widetilde{a}_{\max} = \max(\max_{\mathbf{y}} \widetilde{a}_{0,\mathbf{y}}, \max_{\mathbf{y},jk} \widetilde{a}_{jk,\mathbf{y}})$, and let the entries of $\widetilde{\mathbf{A}}' \in [0,1]^{(s^2+1)\times|\mathcal{Y}|}$ be defined as

$$\widetilde{a}'_{0,\mathbf{y}} = \frac{\widetilde{a}_{0,\mathbf{y}} - \widetilde{a}_{\min}}{\widetilde{a}_{\max} - \widetilde{a}_{\min}} \in [0,1]\,; \qquad \widetilde{a}'_{jk,\mathbf{y}} = \frac{\widetilde{a}_{jk,\mathbf{y}} - \widetilde{a}_{\min}}{s\cdot(\widetilde{a}_{\max} - \widetilde{a}_{\min})} \in [0,1] \quad \forall j,k \in [s]\,.$$

It can be verified that $\sum_{k=1}^{s} \widetilde{a}'_{jk,\mathbf{y}} \le 1$ for all $j \in [s], \mathbf{y} \in \mathcal{Y}$. Next, define $\mathbf{q}'(x) = \widetilde{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x)$. Then, for each $j \in [s]$, we set up a weighted multiclass ($(s+1)$-class) CPE problem with weights $(\widetilde{a}'_{j1,\mathbf{y}}, ..., \widetilde{a}'_{js,\mathbf{y}}, (1 - \sum_{k=1}^{s} \widetilde{a}'_{jk,\mathbf{y}}))$ to estimate the statistics $q'_{j1}(x), ..., q'_{js}(x)$, and a weighted binary CPE problem with weights $(\widetilde{a}'_{0,\mathbf{y}}, (1 - \widetilde{a}'_{0,\mathbf{y}}))$ to estimate $q'_0(x)$. Finally, given $\widehat{\mathbf{q}}'(x)$ estimated in this way from the noisy training sample, our NCEFP algorithm outputs the multi-label classifier[24]

$$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \Big\{ 1 - [(\widetilde{a}_{\max} - \widetilde{a}_{\min})\cdot \widehat{q}'_0(x) + \widetilde{a}_{\min}]\cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{j=1}^{s}\sum_{k=1}^{s} [s\cdot(\widetilde{a}_{\max} - \widetilde{a}_{\min})\cdot \widehat{q}'_{jk}(x) + \widetilde{a}_{\min}]\cdot \frac{2\cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1} \Big\}\,.$$

Estimating $\mathbf{q}'(x)$. Our implementation of NCEFP uses weighted multiclass and binary logistic loss minimizers for the weighted CPE learners. In particular, we first learn a vector of s^2+1 real-valued functions $\widehat{\mathbf{f}} : \mathcal{X} \to \mathbb{R}^{s^2+1}$ by minimizing the (s^2+1)-dimensional convex surrogate loss

[23] Entries of $\widetilde{\mathbf{A}}$ cannot be used directly as they may be negative and/or not add up to one.

[24] The combinatorial optimization problem involved in producing $\widehat{\mathbf{h}}(x)$ has a similar functional form as its non-noisy counterpart, and for $\mathcal{Y} = \{0,1\}^s$, it can be solved in order $O(s^3)$ time using a procedure of Dembczynski et al. (2011) (if the label vectors are sparse with at most K nonzero entries each, then the optimization can be solved in order $O(sK^2)$ time; a special case of this scenario is discussed in Section 6.7 and Section B.3).

$\psi : \mathcal{Y} \times \mathbb{R}^{s^2+1} \to \mathbb{R}_+$ defined as

$$\psi(\mathbf{y}, \mathbf{u}) = \widetilde{a}'_{0,\mathbf{y}} \cdot \phi_{\log}(1, u_0) + (1 - \widetilde{a}'_{0,\mathbf{y}}) \cdot \phi_{\log}(0, u_0) + \\ \sum_{j=1}^{s} \big[\sum_{k=1}^{s} \widetilde{a}'_{jk,\mathbf{y}} \phi_{\text{mlog}}(k, (u_{j1}, ..., u_{js})) + (1 - \sum_{k=1}^{s} \widetilde{a}'_{jk,\mathbf{y}}) \phi_{\text{mlog}}(s+1, (u_{j1}, ..., u_{js})) \big]$$

over the noisy sample $\widetilde{S}$. Specifically, $\widehat{\mathbf{f}} \in \text{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$ for a suitable class of real-valued vector functions $\mathcal{F} \subseteq \{\mathbf{f} : \mathcal{X} \to \mathbb{R}^{s^2+1}\}$. The estimated statistics are then given by $\widehat{q}'_0(x) = \gamma_{\log}^{-1}(\widehat{f}_0(x))$ and $\widehat{q}'_{jk}(x) = \big(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(\widehat{f}_{j1}(x), ..., \widehat{f}_{js}(x))\big)_k$. Detailed pseudocode is in Section B.1.

6.5.3. General Low-rank Multi-label Losses under General CCN: NCOC Algorithm

We now consider multi-label learning with a general low-rank loss matrix (encompassing the Hamming loss and F_1-measure as special cases) under general class-conditional noise (CCN). In this section, we will build on the Output Coding (OC) algorithm of Zhang et al. (2020) which was developed for the multi-label F_1-measure in the standard (non-noisy) setting and was shown to be consistent for that setting. The approach applies more broadly to low-rank loss matrices in general, and we will develop a noise-corrected version of this algorithm for the general setting that we will call the *Noise-Corrected Output Coding* (NCOC) algorithm. Again, our approach will be to reliably estimate suitable Bayes-sufficient statistics $\mathbf{q}(x)$ associated with the clean distribution D from the noisy training sample. In the special cases of Hamming loss and F_1-measure, these statistics are the same as those discussed in Section 6.5.1 and Section 6.5.2, but the estimation procedures will be different.

Output coding is a general term that refers to the solution of multiclass or multi-label problems by decomposing them into a set of binary prediction problems (Dietterich and Bakiri, 1995; Allwein et al., 2000; Ramaswamy et al., 2014). The OC algorithm of Zhang et al. (2020) breaks down a multi-label prediction problem with a low-rank loss into a small number of weighted binary CPE problems. In particular, consider a multi-label loss matrix $\mathbf{L}$ that can be written as $\mathbf{L} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ for some $\mathbf{A} \in [0,1]^{r \times |\mathcal{Y}|}, \mathbf{B} \in \mathbb{R}^{r \times |\mathcal{Y}|}, \mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ (so that $\text{rank}(\mathbf{L}) \leq r + 1$). Then it turns out that the r-dimensional vector statistic $\mathbf{q}(x)$ defined as

$$\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x)$$

is Bayes-sufficient for $\mathbf{L}$. Indeed, the Bayes optimal classifier for $\mathbf{L}$ can be written as

$$\mathbf{h}^*(x) \in \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \boldsymbol{\ell}_{\widehat{\mathbf{y}}}^\top \boldsymbol{\eta}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \mathbf{b}_{\widehat{\mathbf{y}}}^\top (\mathbf{A}\boldsymbol{\eta}(x)) + t_{\widehat{\mathbf{y}}} = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \mathbf{b}_{\widehat{\mathbf{y}}}^\top \mathbf{q}(x) + t_{\widehat{\mathbf{y}}}\,.$$

In the standard (non-noisy) setting, the OC algorithm of Zhang et al. (2020) estimates these statistics $\mathbf{q}(x)$ by decomposing the multi-label problem into r weighted binary CPE problems. Again, we will develop noise-corrected versions of these procedures to estimate the statistics from the noisy training sample.

As before, under the general CCN model, we have $\widetilde{\boldsymbol{\eta}}(x) = \mathbf{C}^\top \boldsymbol{\eta}(x)$. Therefore, if $\mathbf{C}$ is invertible, then $\mathbf{q}(x)$ can be written in terms of $\widetilde{\boldsymbol{\eta}}(x)$ as $\mathbf{q}(x) = \widetilde{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)$, where $\widetilde{\mathbf{A}} = \mathbf{A}(\mathbf{C}^\top)^{-1}$. Again, in order to set up suitably weighted binary CPE problems that can allow us to estimate these statistics from the noisy training sample, we will use a shifted and scaled matrix $\widetilde{\mathbf{A}}'$ to estimate related statistics $\mathbf{q}'(x)$, and then factor back in the scaling and shifting when making a final prediction. Towards this, define $\widetilde{a}_{\min} = \min_{\mathbf{y},j} \widetilde{a}_{j,\mathbf{y}}$ and $\widetilde{a}_{\max} = \max_{\mathbf{y},j} \widetilde{a}_{j,\mathbf{y}}$, and define the entries of $\widetilde{\mathbf{A}}' \in [0,1]^{r\times|\mathcal{Y}|}$ as

$$\widetilde{a}'_{j,\mathbf{y}} = \frac{\widetilde{a}_{j,\mathbf{y}} - \widetilde{a}_{\min}}{\widetilde{a}_{\max} - \widetilde{a}_{\min}} \in [0,1] \quad \forall j \in [r]\,.$$

We note that scaling for the terms $\widetilde{a}'_{j,\mathbf{y}}$ is different from that used for the NCEFP algorithm in Section 6.5.2, as now we are decomposing the problem into r binary problems rather than multiclass. Next, define $\mathbf{q}'(x) = \widetilde{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x)$. Then, for each $j \in [r]$, we set up a weighted binary CPE problem with weights $(\widetilde{a}'_{j,\mathbf{y}}, (1-\widetilde{a}'_{j,\mathbf{y}}))$ to estimate $q'_j(x)$. Finally, given estimated statistics $\widehat{\mathbf{q}}'(x)$ estimated in this way from the noisy training sample, our NCOC algorithm outputs the multi-label classifier

$$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \left\{ t_{\widehat{\mathbf{y}}} + \sum_{j=1}^r [(\widetilde{a}_{\max} - \widetilde{a}_{\min}) \cdot \widehat{q}'_j(x) + \widetilde{a}_{\min}] \cdot b_{j,\widehat{\mathbf{y}}} \right\}.$$

Estimating $\mathbf{q}'(x)$. Our implementation of NCOC uses weighted binary logistic loss minimizers for the weighted CPE learners. In particular, we first learn a vector of r real-valued functions $\widehat{\mathbf{f}} : \mathcal{X} \to \mathbb{R}^r$ by minimizing the r-dimensional convex surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^r \to \mathbb{R}_+$ defined as

$$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^{r} \left(\widetilde{a}'_{j,\mathbf{y}} \phi_{\log}(1, u_j) + (1 - \widetilde{a}'_{j,\mathbf{y}}) \phi_{\log}(0, u_j) \right)$$

over $\widetilde{S}$. Specifically, $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$ for a suitable class of functions $\mathcal{F} \subseteq \{\mathbf{f} : \mathcal{X} \to \mathbb{R}^r\}$. The estimated statistics are then given by $\widehat{q}'_j(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$. Detailed pseudocode is in Section B.1.

The above approach can be applied to any low-rank loss matrix that can be written in the form described above, including both Hamming loss and F_1-measure as discussed below.

Example 6.1 **(Low-rank decomposition for $\mathbf{L}^{\mathbf{Ham}}$).** *The Hamming loss in Eq.* (6.1) *can be written as*

$$\ell^{\text{Ham}}_{\mathbf{y},\widehat{\mathbf{y}}} = \frac{1}{s} \sum_{j=1}^{s} \mathbf{1}(\widehat{y}_j \neq y_j) = \sum_{j=1}^{s} \frac{1 - 2\widehat{y}_j}{s} y_j + \sum_{j=1}^{s} \frac{\widehat{y}_j}{s} \,. \tag{6.3}$$

In other words, we have $\mathbf{L}^{\text{Ham}} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ where $\mathbf{A} \in [0,1]^{s \times |\mathcal{Y}|}$ with $a_{j,\mathbf{y}} = y_j$, $\mathbf{B} \in \mathbb{R}^{s \times |\mathcal{Y}|}$ with $b_{j,\widehat{\mathbf{y}}} = \frac{1-2\widehat{y}_j}{s}$, and $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ with $t_{\widehat{\mathbf{y}}} = \sum_{j=1}^{s} \frac{\widehat{y}_j}{s} = \frac{1}{s}\|\widehat{\mathbf{y}}\|_1$. Thus, for Hamming loss, $r = s$ and the statistics $\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x) \in [0,1]^s$ turn out to be the same as in Section 6.5.1.

Example 6.2 **(Low-rank decomposition for $\mathbf{L}^{F_1}$).** *The F_1-measure loss in Eq.* (6.2) *can be written as*

$$\ell^{F_1}_{\mathbf{y},\widehat{\mathbf{y}}} = 1 - \mathbf{1}(\|\mathbf{y}\|_1 = 0) \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{j=1}^{s} \sum_{k=1}^{s} \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j \cdot \frac{2 \cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1} \,. \tag{6.4}$$

In other words, we have $\mathbf{L}^{F_1} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ where $\mathbf{A} \in [0,1]^{(s^2+1) \times |\mathcal{Y}|}$ with $a_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)$ and $a_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j$, $\mathbf{B} \in \mathbb{R}^{(s^2+1) \times |\mathcal{Y}|}$ with $b_{0,\widehat{\mathbf{y}}} = -\mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0)$ and $b_{jk,\widehat{\mathbf{y}}} = -\frac{2 \cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1}$, and $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ with $t_{\widehat{\mathbf{y}}} = 1$. Thus, for F_1-measure, $r = s^2 + 1$ and the statistics $\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x) \in [0,1]^{s^2+1}$

turn out to be the same as in Section 6.5.2.

Remark on computation for NCEFP and NCOC, and fast NCOC-Ham-IFN algorithm. We note that the NCEFP and NCOC algorithms above both require storing the noise matrix $\mathbf{C}$ and computing $(\mathbf{C}^\top)^{-1}$. For a general multi-label noise matrix $\mathbf{C}$, this can be prohibitively expensive. Therefore, these algorithms are practical when either the number of tags s is small or the noise matrix $\mathbf{C}$ is suitably structured. We describe one such structure, namely the STSN model, in Section 6.7, that enables fast computation. We also note that under the previously well-studied IFN model, noise matrices $\mathbf{C}$ – even though relatively simple with few parameters – are (to our knowledge) expensive to invert. For the special case of Hamming loss under IFN, in Section B.1.4, we present an alternative faster noise-corrected output coding algorithm – that we call NCOC-Ham-IFN – that decomposes the problem of estimating statistics $\mathbf{q}(x)$ into a different set of s binary CPE problems obtained using a different coding matrix $\widetilde{\mathbf{A}}''$ that does not require inverting $\mathbf{C}^\top$.

6.6. Regret Transfer Bounds and Consistency

Below we provide quantitative regret transfer bounds for each of the three algorithms above that upper bound the target $\mathbf{L}$-regret of the learned classifier $\widehat{\mathbf{h}}$ under the clean distribution D in terms of the surrogate ψ*-regret* (defined below) of the associated real-valued vector function $\widehat{\mathbf{f}}$ obtained by minimizing the corresponding convex surrogate loss ψ (defined for each algorithm in the corresponding section above) under the noisy distribution $\widetilde{D}$. In each case, if the surrogate loss ψ is minimized over a suitably rich function class, then ψ-regret under $\widetilde{D}$ converges in probability to 0 as $m \to \infty$. Therefore, this also implies Bayes consistency of these algorithms for the target loss $\mathbf{L}$ under D.

ψ-generalization error and ψ-regret. For any positive integer r, an r-dimensional surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^r \to \mathbb{R}_+$ and vector-valued function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^r$, define the ψ*-generalization error* of $\mathbf{f}$ under the noisy distribution $\widetilde{D}$ as $\mathrm{er}^{\psi}_{\widetilde{D}}[\mathbf{f}] = \mathbf{E}_{(X,\widetilde{\mathbf{Y}})\sim\widetilde{D}}[\psi(\widetilde{\mathbf{Y}}, \mathbf{f}(X))]$, and its ψ-regret of under $\widetilde{D}$ as $\mathrm{regret}^{\psi}_{\widetilde{D}}[\mathbf{f}] = \mathrm{er}^{\psi}_{\widetilde{D}}[\mathbf{f}] - \inf_{\mathbf{f}' : \mathcal{X} \to \mathbb{R}^r} \mathrm{er}^{\psi}_{\widetilde{D}}[\mathbf{f}']$.

Theorem 6.1 (Regret bound for NCPLUG). *Consider Hamming loss $\mathbf{L}^{\mathrm{Ham}}$ (Eq. (6.1)) under*

IFN model. Assume $c^{(j)}_{0,1} + c^{(j)}_{1,0} < 1$ for all $j \in [s]$. Let D be any distribution on $\mathcal{X} \times \mathcal{Y}$ with corresponding noisy distribution $\widetilde{D}$. Suppose NCPLUG (Section 6.5.1) is run with noisy training sample $\widetilde{S}$ (in which examples are sampled i.i.d. from $\widetilde{D}$), and let $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ be as defined in Section 6.5.1. Then we have

$$\mathrm{regret}^{\mathbf{L}^{\mathrm{Ham}}}_{D}[\widehat{\mathbf{h}}] \leq \frac{1}{\sqrt{s}} \max_{i} \frac{1}{1 - c^{(i)}_{0,1} - c^{(i)}_{1,0}} \sqrt{2\mathrm{regret}^{\psi}_{\widetilde{D}}[\widehat{\mathbf{f}}]}\,.$$

Proof. See Section B.2.1. □

Theorem 6.2 (Regret bound for NCEFP). *Consider F-measure $\mathbf{L}^{F_1}$ (Eq. (6.2)) under the general CCN model. Assume noise matrix $\mathbf{C}$ is invertible. Let D be any distribution on $\mathcal{X} \times \mathcal{Y}$ with corresponding noisy distribution $\widetilde{D}$. Suppose NCEFP (Section 6.5.2) is run with noisy training sample $\widetilde{S}$ (in which examples are sampled i.i.d. from $\widetilde{D}$). Let $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ be as defined in Section 6.5.2, and let $\mathbf{A} \in [0,1]^{(s^2+1)\times|\mathcal{Y}|}$, $\mathbf{B} \in \mathbb{R}^{(s^2+1)\times|\mathcal{Y}|}$, and $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ be as defined in Example 6.2. Then we have*

$$\mathrm{regret}^{\mathbf{L}^{F_1}}_{D}[\widehat{\mathbf{h}}] \leq 4s \max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \|\mathbf{A}\|_1 \|(\mathbf{C}^\top)^{-1}\|_1 \sqrt{2\mathrm{regret}^{\psi}_{\widetilde{D}}[\widehat{\mathbf{f}}]}\,.$$

Proof. See Section B.2.2. □

Theorem 6.3 (Regret bound for NCOC). *Consider a general low-rank loss matrix $\mathbf{L}$ written as $\mathbf{L} = \mathbf{A}^\top\mathbf{B} + \mathbf{1}\mathbf{t}^\top$ for some $\mathbf{A} \in [0,1]^{r\times|\mathcal{Y}|}, \mathbf{B} \in \mathbb{R}^{r\times|\mathcal{Y}|}, \mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$, under the general CCN model. Assume noise matrix $\mathbf{C}$ is invertible. Let D be any distribution on $\mathcal{X} \times \mathcal{Y}$ with corresponding noisy distribution $\widetilde{D}$. Suppose NCOC (Section 6.5.3) is run with noisy training sample $\widetilde{S}$ (in which examples are sampled i.i.d. from $\widetilde{D}$), and let $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ be as defined in Section 6.5.3. Then we have*

$$\mathrm{regret}^{\mathbf{L}}_{D}[\widehat{\mathbf{h}}] \leq 2 \max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \|\mathbf{A}\|_1 \|(\mathbf{C}^\top)^{-1}\|_1 \sqrt{2\mathrm{regret}^{\psi}_{\widetilde{D}}[\widehat{\mathbf{f}}]}\,.$$

Proof. See Section B.2.3. □

The quantity $\|(\mathbf{C}^\top)^{-1}\|_1$ can be viewed as capturing the amount of label noise in $\mathbf{C}$. The bounds therefore suggest that as the amount of label noise increases (larger $\|(\mathbf{C}^\top)^{-1}\|_1$), the sample size needed to reach a given level of performance generally increases.

In Section B.2, we provide proofs of more general versions of the above theorems that allow one to use an estimated noise matrix $\widehat{\mathbf{C}}$ when the true noise matrix $\mathbf{C}$ may be unknown.

6.7. Similar-Tag Switching Noise (STSN) model

As noted earlier, general CCN models can require too many (up to order $O(4^s)$) parameters and make computation prohibitive. Here we propose a new family of structured multi-label noise models that we term *Similar-Tag Switching Noise* (STSN) models; STSN models are a special case of CCN that require fewer parameters and enable fast computation, and moreover, unlike IFN, they also capture some correlations among tags. The idea behind STSN models is that tags are partitioned into several groups, each of which contains similar/related tags, and independently within each group, an active tag can be switched with another tag in the group (i.e., similar tags can be switched with each other).

Specifically, let the set of s tags $\mathcal{T} = [s]$ be partitioned into K ($K \leq s$) groups of tags $G_1, ..., G_K$, such that the tags within any group are similar/related to each other (for example, G_1 could contain tags `lion`, `tiger`; G_2 could contain tags `river`, `lake`; etc.). We will assume that within each group, at most one tag is active in any label vector $\mathbf{y}$; this gives $|\mathcal{Y}| = \prod_{k=1}^{K}(1+|G_k|) \ll 2^s$, and $\|\mathbf{y}\|_1 \leq K$ for all $\mathbf{y} \in \mathcal{Y}$ (indeed, this is in line with many real multi-label datasets in which labels are very sparse). The STSN model involves K noise parameters: $\sigma_k \in [0,1]$ for $k \in [K]$. Specifically, for a label $\mathbf{y} \in \mathcal{Y} \subseteq \{0,1\}^s$, let $\mathbf{y}_{G_k} \in \{0,1\}^{|G_k|}$ denote the sub-label restricted to tags in G_k. Recall that $\|\mathbf{y}_{G_k}\| \leq 1$. For groups G_k with $|G_k| \geq 2$, the noise process within G_k is as follows: if $\mathbf{y}_{G_k} = \mathbf{0}$, then $\widetilde{\mathbf{y}}_{G_k} = \mathbf{0}$; if $\mathbf{y}_{G_k} \neq \mathbf{0}$, then with probability σ_k, $\mathbf{y}_{G_k}$ is changed to $\widetilde{\mathbf{y}}_{G_k}$ by switching its (only) active tag with one of the remaining $|G_k|-1$ non-active tags chosen uniformly at random, and with probability $1-\sigma_k$, $\widetilde{\mathbf{y}}_{G_k}$ is the same as $\mathbf{y}_{G_k}$. For groups G_k with $|G_k| = 1$, $\mathbf{y}_{G_k}$ is flipped to the opposite with probability σ_k. The above noise process is applied independently to each group. (For the special case when there are $K = s$ groups each of size one, the STSN model reduces to the symmetric IFN model studied by Kumar et al. (2020), but in the general case, it can capture much richer structure.) Section B.3 summarizes various small changes/simplifications to our algorithms under STSN.

Example 6.3. *Consider a dataset of images that need to be annotated with various object tags. The tags can be grouped into categories like animals, vehicles, plants, and many others. In this case, it is more likely to confuse between tags within the same group (e.g., confusing a `Lion` for a `Tiger`) rather than confusing tags from different groups (e.g., confusing a `Lion` for a `Car`). Specifically, for a given image with `Lion` being an active tag in the true/clean label, after the noise process, `Tiger` may become an active tag in the noisy label instead; but `Lion` cannot be switched to `Car`.*

Example 6.4 (Noise matrix calculation)**.** *Consider $\mathcal{T} = \{1,2,3\}$ with groups $G_1 = \{1,2\}$ and $G_2 = \{3\}$. Let $\sigma_1 = 0.2$ and $\sigma_2 = 0.1$. Then $\mathbf{C}^{(1)}$ and $\mathbf{C}^{(2)}$ can be calculated as below. Then, for example, $\mathbf{P}(\widetilde{\mathbf{Y}} = [1,0,1] \,|\, \mathbf{Y} = [0,1,0]) = \mathbf{P}(\widetilde{\mathbf{Y}}_{G_1} = [1,0] \,|\, \mathbf{Y}_{G_1} = [0,1]) \cdot \mathbf{P}(\widetilde{\mathbf{Y}}_{G_2} = [1] \,|\, \mathbf{Y}_{G_2} = [0]) = 0.02$. Other entries of $\mathbf{C}$ can be calculated similarly.*

$$\mathbf{C}^{(1)} = \begin{array}{c|ccc} & 00 & 01 & 10 \\ \hline 00 & 1 & 0 & 0 \\ 01 & 0 & 0.8 & 0.2 \\ 10 & 0 & 0.2 & 0.8 \end{array}, \quad \mathbf{C}^{(2)} = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0.9 & 0.1 \\ 1 & 0.1 & 0.9 \end{array}.$$

6.8. Experiments

6.8.1. Synthetic Data: Sample Complexity Behavior

We tested the sample complexity behavior of our algorithms.

F_1-measure under STSN. We generated a multi-label dataset with instances x in $\mathcal{X} = \mathbb{R}^{100}$ and $s = 10$ tags partitioned into $K = 5$ groups $\mathcal{G} = \{\{1,2,3\},\{4,5,6\},\{7,8\},\{9\},\{10\}\}$, so $|\mathcal{Y}| = 192$ and $\|\mathbf{y}\|_1 \le K = 5 \;\forall \mathbf{y} \in \mathcal{Y}$ (details in Section B.4). We then added label noise using 4 single-parameter STSN noise matrices respecting this partition: specifically, we let $\sigma_1, ..., \sigma_5 = \sigma$, and

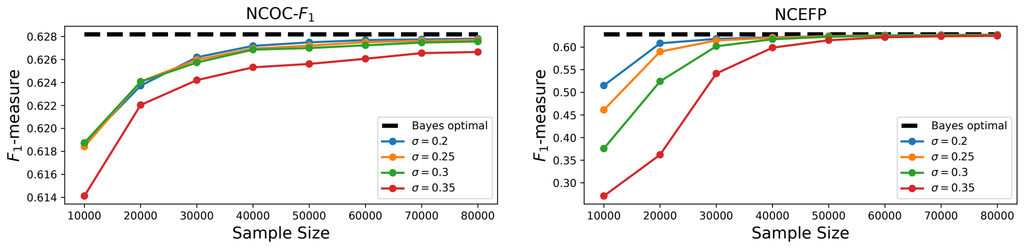

Figure 6.4: Sample complexity behavior of NCOC-F_1 and NCEFP on synthetic multi-label data for 4 noise parameters under the STSN model. Performance measure is F_1-measure (specified as a gain). For both algorithms, as noise parameter σ increases, the sample size needed to reach a given level of performance generally increases.

chose 4 values of σ: $0.2, 0.25, 0.3, 0.35$. We ran NCOC-F_1 and NCEFP (with a linear function class) to learn multi-label classifiers from increasingly large noisy training samples generated in this way, and measured the F_1-measure on a test set of $10,000$ clean data points. The results are shown in Figure 6.4. We see that, as suggested by our regret bounds, as noise parameter σ increases, the sample size needed to reach a given level of performance generally increases.

Hamming loss under IFN. We generated a multi-label dataset with instances x in $\mathcal{X} = \mathbb{R}^{100}$ and $s = 8$ tags (details in Section B.4). We then added label noise using 3 sets of IFN noise rates: specifically, we let $c^{(j)}_{1,0} = c_{1,0}$ and $c^{(j)}_{0,1} = c_{0,1}$ for all j, and chose 3 pairs of values of $(c_{0,1}, c_{1,0})$: $(0.1, 0.05), (0.15, 0.2), (0.3, 0.3)$. We ran all algorithms (with a linear function class) to learn multi-label classifiers from increasingly large noisy training samples generated in this way, and measured the Hamming loss on a test set of $10,000$ clean data points. The results are shown in Figure 6.5. We see that, as suggested by our regret bounds, as the overall noise increases, the sample size needed to reach a given level of performance generally increases.

6.8.2. Real Data: Comparison with Other Methods

Next, we evaluated our algorithms on two real multi-label datasets: Mediamill and Multi-MNIST (Snoek et al., 2006; Sun, 2019).

Mediamill dataset. The original Mediamill dataset has 30,993 training examples and 12,914 test examples with 101 tags, and the images have been processed into 120 features (Snoek et al., 2006). We selected a subset of 17 tags that contains naturally groupable tags, and divided them into 7

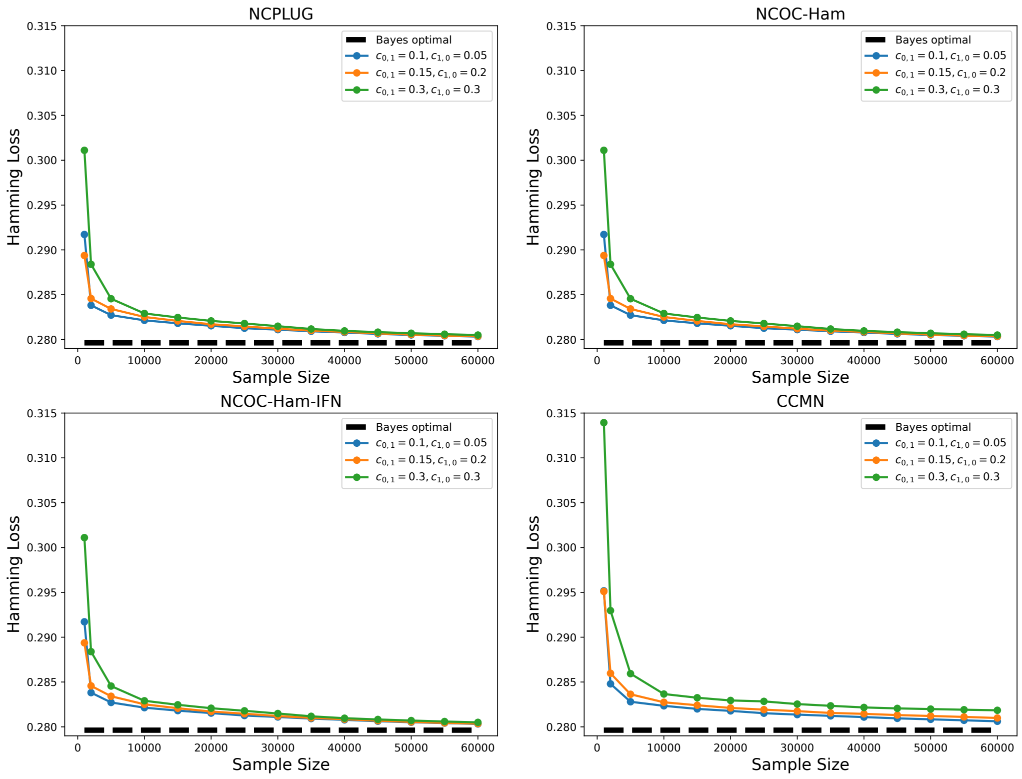

Figure 6.5: Sample complexity behavior of NCPLUG, NCOC-Ham, NCOC-Ham-IFN, and CCMN on synthetic multi-label data for 3 pairs of noise parameters under the IFN model. Performance measure is Hamming loss. For NCPLUG, NCOC-Ham and NCOC-Ham-IFN algorithms, as the overall noise increases, the sample size needed to reach a given level of performance generally increases.

groups: { {`Duo-anchor`, `Anchor`}, {`People`, `People marching`, `People walking`}, {`Split screen`, `Screen`}, {`Sky`, `Cloud`}, {`Religious leader`, `Monologue`}, {`Court`, `Meeting`}, {`Tower`, `Government building`, `Urban`, `Building`} }. (See also Figure 6.6 for visual impressions.) For consistency with the STSN model assumption, we removed instances that have more than one active tag in any group; we also removed instances that do not have any active tag among the 17 tags. Our modified Mediamill dataset has 20,141 training examples and 7,737 test examples with 17 tags.

Hamming loss under IFN. For IFN, we let $c_{1,0}^{(j)} = c_{1,0}$ and $c_{0,1}^{(j)} = c_{0,1}$ for all j, and chose noise parameters of the form $(c_{0,1}, c_{1,0})$. We compared our NCPLUG and NCOC-Ham-IFN algorithms with CCMN (Xie and Huang, 2023) and basic OC-Ham/BR (Zhang and Zhou, 2014). All algorithms were trained to learn linear models with regularization (details in Section B.4). The results are

Figure 6.6: Examples of the selected 17 tags in Mediamill dataset. Pictures were taken from Snoek et al. (2006).

shown in Table 6.1. As seen, our NCPLUG and NCOC-Ham-IFN algorithms generally outperform other baselines.

Table 6.1: Hamming loss on (modified) Mediamill data with IFN model (*lower* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter $(c_{0,1}, c_{1,0})$	**Noise level (%)**	**NCPLUG**	**NCOC-Ham-IFN**	**CCMN**	**OC-Ham/BR**
(0.2,0.1)	97.34	7.74±0.0	7.74±0.0	7.91±0.09	**7.66±0.0**
(0.1,0.2)	85.62	**7.73±0.0**	**7.73±0.0**	7.77±0.0	7.98±0.0
(0.4,0.15)	99.96	**7.79±0.0**	**7.79±0.0**	8.09±0.07	11.75±0.01
(0.15,0.4)	96.17	**7.8±0.0**	**7.8±0.0**	8.03±0.07	8.32±0.0
(0.35,0.2)	99.86	**7.8±0.0**	**7.8±0.0**	8.17±0.07	8.65±0.01
(0.2,0.35)	98.32	**7.82±0.0**	**7.82±0.0**	8.11±0.07	8.23±0.0
(0.45,0.25)	99.99	**7.86±0.04**	**7.86±0.04**	8.28±0.04	15.44±0.04
(0.25,0.45)	99.48	**7.9±0.0**	**7.9±0.0**	8.29±0.02	8.33±0.0

Hamming loss and F_1-measure under STSN. For STSN, we used single-parameter noise matrices respecting the partition into $K = 7$ groups described above, with $\sigma_1, ..., \sigma_7 = \sigma$. We compared our NCOC-Ham algorithm with basic OC-Ham/BR (Zhang and Zhou, 2014), as well as our NCOC-F_1 and NCEFP algorithms with basic OC-F_1 (Zhang et al., 2020) and EFP (Dembczynski et al., 2013). All algorithms were trained to learn linear models with regularization (details in Section B.4). The results are shown in Table 6.2 and Table 6.3. Again, our noise-corrected algorithms generally outperform other baselines.

Multi-MNIST dataset. We started with the TripleMNIST dataset (each image contains 3 digits;

Table 6.2: Hamming loss on (modified) Mediamill data with STSN model (*lower* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \tilde{\mathbf{y}}_i)$.

Noise parameter (σ)	**Noise level (%)**	**NCOC-Ham**	**OC-Ham/BR**
0.05	6.61	**7.65±0.0**	7.72±0.0
0.1	13.47	**7.66±0.0**	7.84±0.0
0.15	19.98	**7.68±0.0**	7.99±0.0
0.2	26.31	**7.68±0.0**	8.09±0.0
0.25	32.36	**7.69±0.0**	8.18±0.0
0.3	38.18	**7.68±0.0**	8.24±0.0
0.35	43.52	**7.65±0.0**	8.29±0.0
0.4	49.01	**7.65±0.0**	8.31±0.0
0.45	53.97	**7.64±0.0**	8.33±0.0
0.55	63.67	**7.99±0.03**	8.35±0.0
0.6	68.21	**7.75±0.0**	8.37±0.0
0.65	72.48	**7.89±0.01**	8.4±0.0
0.7	76.87	**7.8±0.08**	8.47±0.0

see Figure 6.7) in the Multi-MNIST repository and applied the following data processing steps.

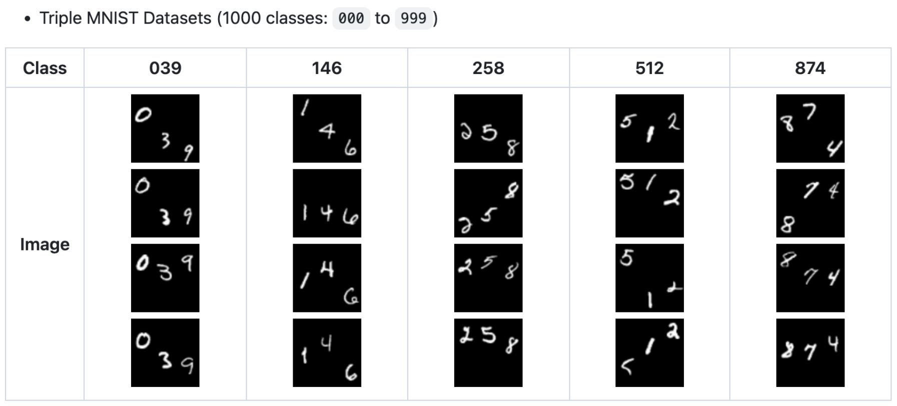

Figure 6.7: Some examples in TripleMNIST. Pictures were taken from https://github.com/shaohua0116/MultiDigitMNIST.

The original TripleMNIST has 1,000 images for each of 1,000 classes (000, 001, ..., 999), with feature dimension $84 \times 84 = 7,056$. We sampled 100 images for each class. For each instance x, we created a label vector $\mathbf{y} \in \{0,1\}^{10}$ in which $y_j = 1$ if digit $j-1$ is present in x, and 0 otherwise. Motivated by the noise model in Patrini et al. (2017), we divided the 10 tags into 5 groups: { {2, 7, 1}, {5, 6}, {3, 8}, {0, 9}, {4} }. For consistency with the STSN model assumption, we removed instances that have more than one active tag in any group. We did not change the features in this process.

Table 6.3: F_1-measure on (modified) Mediamill data with STSN model (*higher* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter (σ)	**Noise level (%)**	**NCEFP**	**NCOC-F_1**	**EFP**	**OC-F_1**
0.05	6.61	42.11±0.03	**42.5±0.02**	42.01±0.02	41.87±0.02
0.1	13.47	42.29±0.03	**42.86±0.02**	41.84±0.03	41.76±0.04
0.15	19.98	42.33±0.03	**42.88±0.01**	41.7±0.02	41.61±0.04
0.2	26.31	42.22±0.02	**42.95±0.02**	41.42±0.01	41.44±0.03
0.25	32.36	42.0±0.02	**43.06±0.03**	41.28±0.03	41.19±0.05
0.3	38.18	41.36±0.03	**43.03±0.03**	41.01±0.02	40.96±0.03
0.35	43.52	40.78±0.03	**43.15±0.04**	40.94±0.02	40.85±0.03
0.4	49.01	39.04±0.09	**43.1±0.05**	40.81±0.03	40.71±0.03
0.45	53.97	32.67±0.05	**41.56±0.38**	40.48±0.04	40.27±0.04
0.55	63.67	32.23±0.1	**41.53±0.39**	7.43±0.03	7.47±0.03
0.6	68.21	31.81±0.03	**42.83±0.06**	6.57±0.03	6.57±0.03
0.65	72.48	18.36±0.12	**37.15±0.26**	6.59±0.01	6.57±0.01
0.70	76.87	26.93±0.1	**43.19±0.4**	6.48±0.02	6.44±0.01

Our modified Multi-MNIST dataset has 68,800 examples with 10 tags. Since the original data did not come with prescribed train/test splits, we split the data into training and test sets with ratio 8 : 2. So we ended up with 55,040 training examples and 13,760 test examples.

Hamming loss under IFN. For IFN, we let $c_{1,0}^{(j)} = c_{1,0}$ and $c_{0,1}^{(j)} = c_{0,1}$ for all j, and chose noise parameters of the form $(c_{0,1}, c_{1,0})$. We compared our NCPLUG and NCOC-Ham-IFN algorithms with CCMN (Xie and Huang, 2023) and basic OC-Ham/BR (Zhang and Zhou, 2014). The results are shown in Table 6.4. As seen, our NCPLUG and NCOC-Ham-IFN algorithms often outperform other baselines.

Table 6.4: Hamming loss on (modified) Multi-MNIST data with IFN model (*lower* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter $(c_{0,1}, c_{1,0})$	**Noise level (%)**	**NCPLUG**	**NCOC-Ham-IFN**	**CCMN**	**OC-Ham/BR**
(0.2,0.1)	85.52	6.15±0.05	**6.14±0.23**	10.83±0.42	7.1±0.26
(0.1,0.2)	74.18	**6.74±0.08**	6.8±0.09	9.9±0.31	6.87±0.14
(0.4,0.15)	98.44	16.42±1.49	**15.72±1.4**	17.57±0.81	27.04±0.95
(0.15,0.4)	91.89	12.82±0.29	**12.72±0.44**	16.04±0.65	14.17±0.26
(0.35,0.2)	97.58	**15.24±1.17**	15.76±1.22	17.33±0.81	20.16±1.6
(0.2,0.35)	93.65	13.38±0.54	13.3±0.52	16.89±0.46	**12.75±0.27**
(0.45,0.25)	99.40	22.58±1.37	22.75±1.33	**22.53±1.17**	36.0±0.53
(0.25,0.45)	97.43	20.01±0.24	20.36±0.17	21.92±0.7	**18.01±0.18**

Hamming loss and F_1-measure under STSN. For STSN model, we used single-parameter noise matrices respecting the partition into $K = 5$ groups described above, with $\sigma_1, ..., \sigma_5 = \sigma$. We compared our NCOC-Ham algorithm with basic OC-Ham/BR (Zhang and Zhou, 2014), as well as our NCOC-F_1 and NCEFP algorithms with basic OC-F_1 (Zhang et al., 2020) and EFP (Dembczynski et al., 2013). The results are shown in Table 6.5 and Table 6.6 Again, our noise-corrected algorithms often outperform other baselines.

Table 6.5: Hamming loss on (modified) Multi-MNIST data with STSN model (*lower* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter (σ)	**Noise level (%)**	**NCOC-Ham**	**OC-Ham/BR**
0.1	28.96	**4.83±0.13**	6.13±0.22
0.2	51.11	**9.11±0.75**	10.28±0.54
0.3	68.14	16.9±1.21	**16.21±0.35**
0.4	80.23	28.06±1.45	**22.5±0.21**
0.6	94.29	**27.38±1.62**	34.19±0.05
0.7	97.35	**24.6±0.29**	40.17±0.19

Table 6.6: F_1-measure on (modified) Multi-MNIST data with STSN model (*higher* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter (σ)	**Noise level (%)**	**NCEFP**	**NCOC-F_1**	**EFP**	**OC-F_1**
0.1	28.96	**91.14±0.16**	90.13±0.18	83.77±0.24	78.33±0.37
0.2	51.11	**84.98±1.24**	84.95±0.52	74.16±0.23	69.61±0.35
0.3	68.14	73.55±1.47	**75.74±0.35**	64.61±0.77	60.71±0.6
0.4	80.23	53.64±0.81	51.48±1.83	**53.87±0.6**	50.97±0.79
0.6	94.29	**43.82±1.72**	41.93±2.02	31.82±0.24	32.57±0.47
0.7	97.35	40.91±0.74	**45.75±2.0**	22.52±0.14	22.88±0.26

6.9. Conclusion

We have developed three consistent noise-corrected multi-label learning algorithms (NCPLUG, NCEFP, and NCOC), encompassing a variety of multi-label performance measures and general class-conditional noise (CCN) models. We have provided quantitative regret transfer bounds for all three algorithms to establish their consistency. We have also proposed a new family of structured multi-label noise models that we term similar-tag switching noise (STSN) models; STSN models are a special case of CCN that require fewer parameters and enable fast computation, and moreover, unlike IFN, they also capture some correlations among tags. Our experiments have confirmed the effectiveness of our algorithms in correcting for multi-label noise. Future work includes developing ways to estimate STSN models from noisy data, and exploring the design of other structured noise models that could be suitable for multi-label settings.

CHAPTER 7

COMPLEX PERFORMANCE MEASURE AND COMPLEX LEARNING SETTING: MULTICLASS LEARNING FROM NOISY LABELS FOR NON-DECOMPOSABLE PERFORMANCE MEASURES

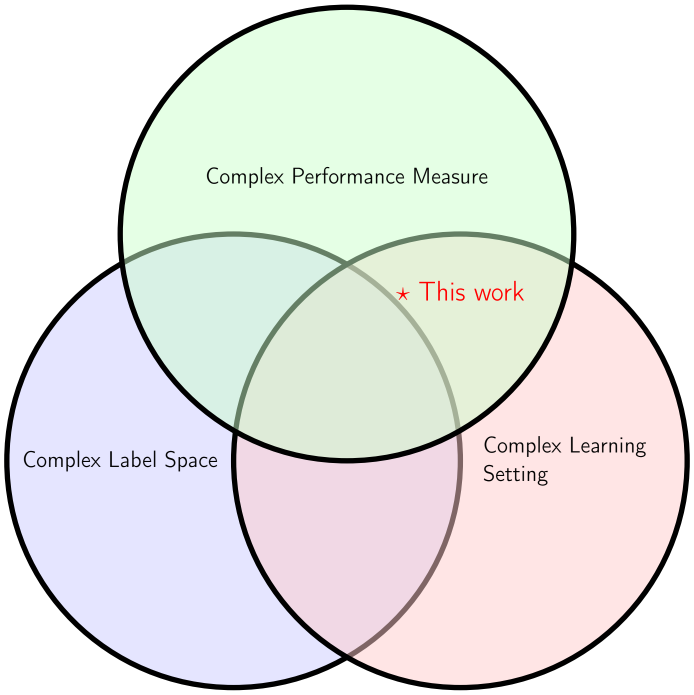

Figure 7.1: Position of *Multiclass Learning from Noisy Labels for Non-decomposable Performance Measures* in the thesis.

This chapter was previously published as Mingyuan Zhang and Shivani Agarwal. Multiclass learning from noisy labels for non-decomposable performance measures. In *International Conference on Artificial Intelligence and Statistics 2024, AISTATS 2024*, volume 238 of *Proceedings of Machine Learning Research*, pages 2170–2178. PMLR, 2024. As the sole first author, I developed all the results (both theoretical and experimental) in this chapter.

In this chapter, we start our discussion of the third dimension of complexities: complex performance measures. We focus on the intersection between complex performance measures and complex learning settings. We show how to design consistent noise-corrected algorithms for multiclass non-decomposable performance measures in the presence of label noise.

7.1. Background of Complex Performance Measure

Complex performance measures, also known as *non-decomposable* performance measures, play a crucial role in evaluating the performance of machine learning models in scenarios where standard performance measures like the 0-1loss or cost-sensitive losses are insufficient to capture the nuances of the problem at hand. These measures are typically *nonlinear* functions of the confusion matrix of a classifier, which represents the predictions of a classifier compared to the ground truth labels.

The complexity of these performance measures arises from their intricate calculations, which require taking into account dependencies or interactions between different aspects of the predictions. Unlike the 0-1loss or cost-sensitive losses that can be expressed as the expectation or sum of losses on individual examples, non-decomposable measures *cannot* be expressed as the expectation or sum of losses on individual examples.

Several non-decomposable performance measures are commonly used in machine learning:

Micro F_1 score: The Micro F_1 score combines precision and recall into a single metric, providing a balanced evaluation of a classifier's performance. It is a widely used metric in information retrieval.

Jaccard measure: Also known as the intersection over union (IoU), the Jaccard measure assesses the similarity between the predicted and true sets by computing their overlapping area divided by their union. It is commonly used in tasks like image segmentation or object detection.

H-mean, G-mean, and Q-mean: The H-mean , G-mean , and Q-mean are often used in imbalanced classification problems to evaluate a classifier's performance.

Area under the ROC curve (AUC-ROC): The AUC-ROC measures the trade-off between true positive rate and false positive rate across different classification thresholds. It provides a comprehensive evaluation of a classifier's performance across all possible decision boundaries.

Area under the Precision-Recall curve (AUC-PR): The AUC-PR summarizes the trade-off between precision and recall of a classifier by computing the area under the precision-recall curve.

Designing machine learning algorithms to learn classifiers that optimize these non-decomposable performance measures can be challenging due to the need to account for dependencies and interactions between the predictions of individual instances. Specialized techniques and optimization approaches are often required.

7.2. Background of Complex Performance Measure and Learning Setting

We have seen both complex performance measures and complex learning settings. In real world, there are problems that pertain both complexities. Such problems require tackling both complexities, necessitating a broad toolbox of methods and algorithms. Here are a few examples of real-world machine learning problems that involve both complex performance measures and complex learning settings:

Information Retrieval: In the field of Information Retrieval (IR), machine learning models are trained to rank documents based on their relevance to a query. This is a complex learning setting because the labels (relevance scores) are often noisy, and the relevance of a document can depend on the relevance of other documents. Furthermore, the performance of IR systems is often evaluated using complex measures like micro and macro F_1 measures, which cannot be decomposed into a sum of losses over individual instances.

Medical Diagnosis: Machine learning in healthcare often involves complex learning settings and performance measures. Healthcare datasets often suffer from class imbalance (where some diseases are more common than others), noisy labels (due to inconsistencies in diagnostic criteria), and missing data (due to incomplete patient records). Further, performance measures like the area under the ROC curve (AUC-ROC) are often used, as it is essential to balance the trade-off between false positives and false negatives. For example, in cancer detection, failing to detect a positive case (false negative) can be detrimental, but so can wrongly diagnosing a patient with cancer (false positive).

Object Detection in Images: Object detection tasks in computer vision often involve both complex learning settings and performance measures. The datasets used can be noisy and imbalanced, and semi-supervised or active learning techniques can be necessary when labeled data is scarce or expensive to obtain. Moreover, the performance of object detectors is commonly evaluated using the Mean Average Precision or the Intersection over Union, which are complex measures.

These examples illustrate the need for machine learning algorithms and systems that can handle both complex performance measures and complex learning settings.

7.3. Introduction

7.3.1. Background and Our Contributions

In many machine learning problems, the labels provided with the training data may be noisy. This can happen due to a variety of reasons, such as sensor measurement errors, human labeling errors, and data collection errors among others. Therefore, there has been much interest in recent years in learning good classifiers from data with noisy labels (Frénay and Verleysen, 2014; Song et al., 2023; Han et al., 2020). Most work has focused on learning from noisy labels for standard loss-based performance measures; these include both the 0-1loss and more general cost-sensitive losses, all of which are linear functions of the confusion matrix of a classifier. However, many machine learning problems require using *non-decomposable* performance measures which cannot be expressed as the expectation or sum of a loss on individual examples; these are general nonlinear functions of the confusion matrix, and include for example the H-mean, Q-mean and G-mean in class imbalance settings (Sun et al., 2006; Kennedy et al., 2009; Lawrence et al., 2012; Wang and Yao, 2012), and the Micro F_1 in information retrieval (Manning et al., 2008; Kim et al., 2013). In this work, we design algorithms to learn from noisy labels for two broad classes of multiclass non-decomposable performance measures, namely, monotonic convex and ratio-of-linear, which encompass all the above examples.

The main challenge in learning from noisy labels is to design algorithms which, given training data with noisy labels, can still learn accurate classifiers w.r.t. the clean/true distribution for a given target performance measure. For loss-based (linear) performance measures, previous works have

Table 7.1: Position of Our Work Relative to Previous Work on Consistent Learning Under CCN Model

Performance Measures	**Standard (Non-noisy) Setting**	**Noisy Setting Under CCN Model**
Loss-based (linear)	Many algorithms including surrogate risk minimization algorithms	Many noise-corrected algorithms (Natarajan et al., 2013; van Rooyen and Williamson, 2017; Patrini et al., 2017; Zhang et al., 2021)
Monotonic convex	Frank-Wolfe based method (Narasimhan et al., 2015)	This work
Ratio-of-linear	Bisection based method (Narasimhan et al., 2015)	This work

designed Bayes consistent algorithms so that, when given sufficient noisy training data, their performance converges to the Bayes optimal performance w.r.t. the clean distribution (Natarajan et al., 2013; Scott et al., 2013; Scott, 2015; Menon et al., 2015; Liu and Tao, 2016; Patrini et al., 2016; Ghosh et al., 2017; van Rooyen and Williamson, 2017; Patrini et al., 2017; Natarajan et al., 2017; Wang et al., 2018; Liu and Guo, 2020; Zhang et al., 2021; Li et al., 2021). In this work, we provide similarly Bayes consistent noise-corrected algorithms for multiclass monotonic convex and ratio-of-linear performance measures, under the widely studied family of class-conditional noise (CCN) models. Our work builds on the Frank-Wolfe and Bisection based methods of Narasimhan et al. (2015), which were proposed for the standard (non-noisy) setting. Table 7.1 summarizes the position of our work relative to other consistent algorithms under the CCN model.

Our key contributions include the following:

- **Algorithms:** We develop noise-corrected versions of the Frank-Wolfe and Bisection based algorithms for the families of monotonic convex and ratio-of-linear performance measures, respectively.

- **Theory:** While the noise corrections we introduce are fairly intuitive, establishing the correctness of the resulting algorithms is not trivial. We provide regret (excess risk) bounds for our algorithms, establishing that even though they are trained on noisy data, they are

Bayes consistent in the sense that their performance converges to the optimal performance w.r.t. the clean (non-noisy) distribution. The bounds quantify the effect of label noise on the sample complexity. We also provide extended regret bounds that quantify the effect of using an estimated noise matrix.

- **Empirical validations:** We provide results of experiments on synthetic data verifying the sample complexity behavior of our algorithms, and also on real data comparing with previous baselines.

7.3.2. Notation

For an integer n, we denote by $[n]$ the set of integers $\{1, \ldots, n\}$, and by Δ_n the probability simplex $\{\mathbf{p} \in \mathbb{R}_+^n : \sum_{y=1}^n p_y = 1\}$. For a vector $\mathbf{a}$, we denote by $\|\mathbf{a}\|_p$ the p-norm of $\mathbf{a}$, and by a_j the j-th entry of $\mathbf{a}$. For a matrix $\mathbf{A}$, we denote by $\|\mathbf{A}\|_p$ the induced matrix p-norm of $\mathbf{A}$, and by $\mathbf{a}_j$ the j-th column vector of $\mathbf{A}$. We use $A_{i,j}$ to denote the (i, j)-th entry of $\mathbf{A}$. In addition, we use $\|\mathbf{A}\|_{\text{vec},p} = (\sum_{i,j} |A_{i,j}^p|)^{1/p}$ for the matrix analogue of the vector p-norm.[25] For matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$, we define $\langle \mathbf{A}, \mathbf{B}\rangle = \sum_{i,j} A_{i,j}B_{i,j}$. The indicator function is $\mathbf{1}(\cdot)$.

7.3.3. Related Work

Consistent algorithms for binary/multiclass classification for non-decomposable performance measures in the standard (non-noisy) setting. Most work in this category has focused on binary classification, for a variety of performance measures, including F-measure (Ye et al., 2012), the arithmetic mean of the true positive and true negative rates (AM) (Menon et al., 2013), ratio-of-linear performance measures (Koyejo et al., 2014; Bao and Sugiyama, 2020), and monotonic performance measures (Narasimhan et al., 2014). Dembczynski et al. (2017) revisited consistency analysis in binary classification for non-decomposable performance measures for two distinct settings and notions of consistency (Population Utility and Expected Test Utility). For multiclass classification, Narasimhan et al. (2015) developed a general framework for designing provably consistent algorithms for monotonic convex and ratio-of-linear performance measures; an extended version of this work also studies such performance measures in constrained learning settings (Narasimhan et al.,

[25]Note that $\|\mathbf{A}\|_1$ and $\|\mathbf{A}\|_\infty$ in Narasimhan et al. (2015) are $\|\mathbf{A}\|_{\text{vec},1}$ and $\|\mathbf{A}\|_{\text{vec},\infty}$ in our notations. We choose to follow conventional definitions of the matrix norm in the literature instead.

2022). Parambath et al. (2014); Koyejo et al. (2015); Natarajan et al. (2016) also designed algorithms for some multiclass non-decomposable performance measures. All of these works designed algorithms for standard (non-noisy) settings. Our methods, which build on Narasimhan et al. (2015), are designed to correct for noisy labels for monotonic convex and ratio-of-linear performance measures, with provable consistency guarantees.

Consistent algorithms for binary/multiclass learning from noisy labels for the 0-1or cost-sensitive losses. For the CCN model in binary classification, many consistent algorithms have been proposed and analyzed (Natarajan et al., 2013; Scott et al., 2013; Menon et al., 2015; Liu and Tao, 2016; Patrini et al., 2016; Liu and Guo, 2020). Scott et al. (2013); Scott (2015); Menon et al. (2015); Liu and Tao (2016) also proposed consistent estimators for noise rates when they are not known (additional assumptions required). Scott et al. (2013); Menon et al. (2015) studied the more general mutually contaminated distributions (MCD) noise model for binary classification, and proposed consistent algorithms. Natarajan et al. (2017) studied cost-sensitive loss functions. Progress has also been made in instance-dependent and label-dependent noise (ILN) model (Menon et al., 2018; Cheng et al., 2020). For the multiclass CCN model, Ghosh et al. (2017); van Rooyen and Williamson (2017); Patrini et al. (2017); Wang et al. (2018); Zhang et al. (2021); Li et al. (2021) proposed consistent algorithms. All the methods above are designed to handle noisy labels for loss-based performance measures; our work, on the other hand, focuses on non-decomposable performance measures.

Consistent algorithms for binary learning from noisy labels for non-decomposable performance measures. The method in Scott et al. (2013) focused on the minmax error. Menon et al. (2015) focused mostly on the balanced error (BER) and area under the ROC curve (AUC) metrics. Both studied the MCD model (which includes CCN model). All these results are for binary classification. Our proposed algorithms, under the CCN model, are designed for monotonic convex and ratio-of-linear performance measures in both binary and multiclass classification settings.

Performance measures in multi-label classification and structured prediction. There is also a line of work studying performance measures (e.g., Hamming loss and F-measure) in multi-label classification and structured prediction problems (Zhang and Zhou, 2014; Li et al., 2016; Wang et al., 2017; Zhang et al., 2020), but those are distinct from (albeit related to) non-decomposable performance measures in multiclass classification settings as considered in this work.

7.3.4. Organization

After preliminaries and background in Section 7.4, we describe our noise-corrected algorithms for two broad classes of non-decomposable performance measures (monotonic convex and ratio-of-linear) in Section 7.5 and Section 7.3.3, respectively. Section 7.7 provides consistency guarantees for our algorithms in the form of regret bounds. Section 7.8 summarizes our experiments. Section 7.9 concludes this work.

7.4. Preliminaries and Background

Multiclass learning from noisy labels. Let $\mathcal{X}$ be the instance space, and $\mathcal{Y}$ be the label space. Without loss of generality, we assume $\mathcal{Y} = [n] = \{1, ..., n\}$. There is an unknown distribution D over $\mathcal{X} \times \mathcal{Y}$. In a standard multiclass learning problem, the learner is given labeled examples (X, Y) drawn from D. However, when learning from noisy labels, the learner is only given noisy examples $(X, \widetilde{Y})$, where $\widetilde{Y}$ is the corresponding noisy label for Y. The learner's goal is to learn a classifier using the *noisy* training sample, so that its performance is good w.r.t. the *clean* distribution.

We consider the *class-conditional noise* (CCN) model (Natarajan et al., 2013; van Rooyen and Williamson, 2017; Patrini et al., 2017), in which a label $Y = y$ is switched by the noise process to $\widetilde{Y} = \widetilde{y}$ with probability $\mathbf{P}(\widetilde{Y} = \widetilde{y}|Y = y)$ that only depends on y (and not on x). This noise can be fully described by a column stochastic matrix.

Definition 7.1 (Class-conditional noise matrix)**.** *The class-conditional noise matrix,* $\mathbf{T} \in [0, 1]^{n \times n}$, *is column stochastic with entries* $T_{i,j} = \mathbf{P}(\widetilde{Y} = i|Y = j)$.

We assume $\mathbf{T}$ is invertible. In practice, $\mathbf{T}$ often needs to be estimated; several methods have been developed to estimate $\mathbf{T}$ from the noisy sample (Xia et al., 2019; Yao et al., 2020; Li et al., 2021). Our algorithms and theoretical guarantees work with both known $\mathbf{T}$ and estimated $\widehat{\mathbf{T}}$.

We can then view the noisy training examples as being drawn i.i.d. from a *noisy* distribution $\widetilde{D}$ on $\mathcal{X} \times \mathcal{Y}$. Specifically, to generate $(X, \widetilde{Y})$, an example (X, Y) is firstly drawn according to D, and then Y is switched to $\widetilde{Y}$ according to noise matrix $\mathbf{T}$.

Non-decomposable performance measure. To measure the performance of a classifier $h : \mathcal{X} \to [n]$, or more generally, a *randomized classifier* $h : \mathcal{X} \to \Delta_n$ (which for a given instance x, predicts a label y according to the probability specified by $h(x)$), we consider performance measures that are general functions of confusion matrices.

Definition 7.2 (Confusion matrix)**.** *The confusion matrix of a (possibly randomized) classifier h w.r.t. a distribution D, denoted by $\mathbf{C}^D[h]$, has entries $C^D_{i,j}[h] = \mathbf{P}_{(X,Y)\sim D, Y'\sim h(X)}(Y = i, Y' = j)$, where $Y' \sim h(X)$ denotes a random draw of label from distribution $h(X)$ when h is randomized.*

Definition 7.3 (Performance measure)**.** *For any function $\psi : \mathbb{R}^{n\times n} \to \mathbb{R}_+$, define the ψ-performance measure of h w.r.t. D as*

$$\Psi^{\psi}_{D}[h] = \psi(\mathbf{C}^D[h])\,.$$

We adopt the convention that lower *values of Ψ correspond to* better *performance.*

The following shows this formulation of performance measure includes the common loss-based performance measures (e.g., the 0-1loss and cost-sensitive losses).

Example 7.1 (**L**-performance measures)**.** *Consider a multiclass loss matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$, where $L_{y,\widehat{y}}$ is the loss incurred for predicting $\widehat{y}$ when the true class is y. Then for a deterministic classifier h,*

$$\Psi^{\mathbf{L}}_{D}[h] = \mathbf{E}_{(X,Y)\sim D}\Big[L_{Y,h(X)}\Big] = \langle \mathbf{L}, \mathbf{C}^D[h]\rangle\,.$$

In fact, loss-based performance measures are linear functions of confusion matrices. For nonlinear ψ, ψ-performance measures are *non-decomposable*, i.e., they cannot be expressed as the expected loss on a new example drawn from D. Common examples of such non-decomposable performance measures include Micro F_1 in information retrieval (Manning et al., 2008; Kim et al., 2013), H-mean, Q-mean and G-mean in class imbalance settings (Kennedy et al., 2009; Lawrence et al., 2012; Sun et al., 2006; Wang and Yao, 2012), and others.[26]

Learning goal. Given a noisy training sample $\widetilde{S}$ drawn according to the noisy distribution $\widetilde{D}$, the goal of the learner is to learn a (randomized) classifier $h : \mathcal{X} \to \Delta_n$ that performs well w.r.t. D for a pre-specified ψ-performance measure. In particular, we want the performance of h to converge (in probability) to *Bayes optimal ψ-performance* as the training sample size increases. Below we define Bayes optimal ψ-performance as the optimal value over *feasible confusion matrices.*

Definition 7.4 (Feasible confusion matrices)**.** Feasible confusion matrices *w.r.t. D are all possible confusion matrices achieved by randomized classifiers. Define $\mathcal{C}_D$ as the set of feasible confusion matrices w.r.t. D as*

$$\mathcal{C}_D = \{\mathbf{C}^D[h] : h : \mathcal{X} \to \Delta_n\}\,.$$

We note that $\mathcal{C}_D$ is a convex set (Narasimhan et al. (2015)).

Definition 7.5 (Bayes optimal ψ-performance)**.** *For any function $\psi : \mathbb{R}^{n\times n} \to \mathbb{R}_+$, define the Bayes optimal ψ-performance w.r.t. D as*

$$\Psi_D^{\psi,*} = \inf_{h:\mathcal{X}\to\Delta_n} \Psi_D^{\psi}[h] = \inf_{h:\mathcal{X}\to\Delta_n} \psi(\mathbf{C}^D[h]) = \inf_{\mathbf{C}\in\mathcal{C}_D} \psi(\mathbf{C})\,.$$

In the following sections, we focus on two broad classes of non-decomposable performance measures, namely *monotonic convex* and *ratio-of-linear*. The former includes H-mean, Q-mean and G-mean, and the latter includes Micro F_1.

[26] See Table 1 of Narasimhan et al. (2015).

7.5. Monotonic Convex Performance Measures

Our work develops noise-corrected versions of the algorithms of Narasimhan et al. (2015). Below, we describe two key operations on which the algorithms in Narasimhan et al. (2015) are built; we then describe our noise-corrected algorithm for monotonic convex performance measures. We will show how we use the noise matrix $\mathbf{T}$ to correct the two operations to learn from noisy labels. We note that the noise correction operations work with estimated $\widehat{\mathbf{T}}$ as well. We start with the definition and some examples of monotonic convex performance measures.

Definition 7.6 (Monotonic convex performance measures)**.** *A performance measure* $\psi : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ *is* monotonic convex *if for any confusion matrix* $\mathbf{C}$*,* $\psi(\mathbf{C})$ *is convex in* $\mathbf{C}$*, and monotonically (strictly) decreasing in* $C_{i,i}$ *and non-decreasing in* $C_{i,j}$ *for* $i \neq j$*.*

Example 7.2 (H-mean, Q-mean and G-mean, all in loss forms)**.** *H-mean:*

$$\psi(\mathbf{C}) = 1 - n\Big(\sum_{i=1}^{n} \frac{\sum_{j=1}^{n} C_{i,j}}{C_{i,i}}\Big)^{-1}.$$

Q-mean:

$$\psi(\mathbf{C}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \Big(1 - \frac{C_{i,i}}{\sum_{j=1}^{n} C_{i,j}}\Big)^2}.$$

G-mean:

$$\psi(\mathbf{C}) = 1 - \Big(\prod_{i=1}^{n} \frac{C_{i,i}}{\sum_{j=1}^{n} C_{i,j}}\Big)^{\frac{1}{n}}.$$

Next, we sketch the idea behind the algorithms in Narasimhan et al. (2015), and show how to introduce noise corrections to learn from noisy labels. We first define the *class probability function.*

Definition 7.7 (Class probability function, class probability for short)**.** *For* D*, the class probability function* $\boldsymbol{\eta} : \mathcal{X} \to \Delta_n$ *is defined as* $\eta_y(X) = \mathbf{P}(Y = y|X)$ *for* $y \in [n]$*. Similarly for* $\widetilde{D}$*, we define* $\widetilde{\boldsymbol{\eta}} : \mathcal{X} \to \Delta_n$ *as* $\widetilde{\eta}_{\widetilde{y}}(X) = \mathbf{P}(\widetilde{Y} = \widetilde{y}|X)$ *for* $\widetilde{y} \in [n]$*.*

Idea behind algorithms in the standard (non-noisy) setting (Narasimhan et al., 2015).

The algorithmic framework optimizes the non-decomposable performance measure ψ of interest through an iterative approach (based on the Frank-Wolfe method for the monotonic convex case, and based on the bisection method for the ratio-of-linear case; details later), which in each iteration t, approximates the target performance measure ψ by a linear loss-based performance measure $\mathbf{L}^t$. Each iteration involves two key operations: OP1 and OP2. **OP1** involves finding an optimal classifier for the current linear approximation $\mathbf{L}^t$. This is done by using a class probability estimator (CPE) $\widehat{\boldsymbol{\eta}}$ learned from the (clean) training sample, and then defining classifier $\widehat{g}^t : \mathcal{X} \to [n]$ as $\widehat{g}^t(x) = \text{argmin}_{y\in[n]} \widehat{\boldsymbol{\eta}}(x)^\top \boldsymbol{\ell}^t_y$. **OP2** involves estimating $\mathbf{C}^D[\widehat{g}^t]$, the confusion matrix of $\widehat{g}^t$ w.r.t. D, by $\widehat{\mathbf{C}}^S[\widehat{g}^t]$, the empirical confusion matrix of $\widehat{g}^t$ w.r.t. sample $S = ((x_i, y_i))_{i=1}^m \sim D^m$ defined below:

$$\widehat{\mathbf{C}}^S_{j,k}[\widehat{g}^t] = \frac{1}{m}\sum_{i=1}^m \mathbf{1}(y_i = j, \widehat{g}^t(x_i) = k)\,. \tag{7.1}$$

Note that $\widehat{\mathbf{C}}^S[\widehat{g}^t]$ converges to $\mathbf{C}^D[\widehat{g}^t]$ as m increases. (More specifically, to facilitate consistency analysis, the iterative algorithms split the training sample S into S_1 and S_2. S_1 is used to learn a CPE model $\widehat{\boldsymbol{\eta}}$, and in each iterative step t, S_2 is used to calculate $\widehat{\mathbf{C}}^{S_2}[\widehat{g}^t]$ via OP2.)

Noise-corrected algorithm for monotonic convex performance measures. We are now ready to describe our approach. In learning from noisy labels, the algorithm only sees noisy sample $\widetilde{S}$. Our approach is to introduce noise corrections to both OP1 and OP2, so the modified algorithm can still output a good classifier w.r.t. the clean distribution D.

Noise-corrected OP1. Recall OP1 involves finding an optimal classifier for a loss-based performance measure $\mathbf{L}^t$ w.r.t. D. To do so with a noisy sample, we propose to find an optimal classifier for a noise-corrected loss-based performance measure $(\mathbf{L}^t)' = (\mathbf{T}^\top)^{-1}\mathbf{L}^t$ w.r.t. $\widetilde{D}$ according to the following proposition.

Proposition 7.1. *Let $\mathbf{L}' = (\mathbf{T}^\top)^{-1}\mathbf{L}$. Then any Bayes optimal classifier for $\mathbf{L}'$-performance w.r.t. $\widetilde{D}$ is also Bayes optimal for $\mathbf{L}$-performance w.r.t. D.*

Proof. Let h^* be a Bayes optimal classifier for $(\mathbf{T}^\top)^{-1}\mathbf{L}$-performance w.r.t. $\widetilde{D}$. So

$$\begin{aligned}\inf_{\widetilde{\mathbf{C}} \in \mathcal{C}_{\widetilde{D}}} \langle (\mathbf{T}^\top)^{-1}\mathbf{L}, \widetilde{\mathbf{C}} \rangle &= \langle (\mathbf{T}^\top)^{-1}\mathbf{L}, \mathbf{C}^{\widetilde{D}}[h^*] \rangle \\ &= \langle \mathbf{L}, \mathbf{T}^{-1}\mathbf{C}^{\widetilde{D}}[h^*] \rangle \\ &= \langle \mathbf{L}, \mathbf{C}^{D}[h^*] \rangle \,,\end{aligned}$$

where we have used properties of the adjoint in the second "=".

Note that for any $\mathbf{C} \in \mathcal{C}_D$, we have

$$\begin{aligned}\langle \mathbf{L}, \mathbf{C} \rangle &= \langle (\mathbf{T}^\top)^{-1}\mathbf{L}, \mathbf{T}\mathbf{C} \rangle \\ &\geq \inf_{\widetilde{\mathbf{C}} \in \mathcal{C}_{\widetilde{D}}} \langle (\mathbf{T}^\top)^{-1}\mathbf{L}, \widetilde{\mathbf{C}} \rangle \\ &= \langle \mathbf{L}, \mathbf{C}^{D}[h^*] \rangle \,.\end{aligned}$$

So h^* is also Bayes optimal for $\mathbf{L}$-performance w.r.t. D, i.e., $\langle \mathbf{L}, \mathbf{C}^D[h^*] \rangle = \Psi_D^{\mathbf{L},*}$. $\square$

This idea has also been used in multiclass noisy label settings with $\mathbf{L}$-performance (van Rooyen and Williamson, 2017; Zhang et al., 2021).

Noise-corrected OP2. Recall OP2 is to estimate $\mathbf{C}^D[\widehat{g}^t]$. We need to do so with noisy sample $\widetilde{S}$. We first observe a relation between clean confusion matrix $\mathbf{C}^D[\widehat{g}^t]$ and noisy confusion matrix $\mathbf{C}^{\widetilde{D}}[\widehat{g}^t]$ under noise matrix $\mathbf{T}$.

Proposition 7.2. *For a given classifier h, the relation between* clean *confusion matrix $\mathbf{C}^D$ and* noisy *confusion matrix $\mathbf{C}^{\widetilde{D}}$ under CCN matrix $\mathbf{T}$ is $\mathbf{C}^{\widetilde{D}}[h] = \mathbf{T}\mathbf{C}^D[h]$.*

Proof.

$$\begin{aligned}
C^{\widetilde{D}}_{i,j}[h] &= \mathbf{P}(\widetilde{Y} = i, h(X) = j) \\
&= \mathbf{P}(\widetilde{Y} = i | h(X) = j)\mathbf{P}(h(X) = j) \\
&= \sum_{k\in[n]} \mathbf{P}(\widetilde{Y} = i, Y = k | h(X) = j)\mathbf{P}(h(X) = j) \\
&= \sum_{k\in[n]} \mathbf{P}(\widetilde{Y} = i | Y = k)\mathbf{P}(Y = k | h(X) = j)\mathbf{P}(h(X) = j) \\
&= \sum_{k\in[n]} T_{i,k} \cdot \mathbf{P}(Y = k | h(X) = j)\mathbf{P}(h(X) = j) \\
&= \sum_{k\in[n]} T_{i,k} \cdot \mathbf{P}(Y = k, h(X) = j) \\
&= \sum_{k\in[n]} T_{i,k} \cdot C^{D}_{kj}[h] \,.
\end{aligned}$$

□

So we propose to estimate $\mathbf{C}^D[\widehat{g}^t]$ by $\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[\widehat{g}^t]$. In Section 7.7, we will show this gives a consistent estimate, i.e., $\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[\widehat{g}^t]$ converges to $\mathbf{C}^D[\widehat{g}^t]$ as the size of $\widetilde{S}$ increases.

We can now incorporate the noise-corrected OP1 and OP2 into the iterative algorithm based on Frank-Wolfe method (Frank and Wolfe, 1956; Narasimhan et al., 2015). The noise-corrected algorithm is summarized in Algorithm 7.1. This algorithm applies to monotonic convex performance measures ψ, such as H-mean, Q-mean and G-mean. It seeks to solve $\min_{\mathbf{C}\in\mathcal{C}_D} \psi(\mathbf{C})$ with the noisy sample $\widetilde{S}$. Note that the form $\nabla\psi(\cdot)$ in Line 7 comes from the form of Bayes optimal classifier for monotonic convex performance measures in the standard (non-noisy) setting (Theorem 13 of Narasimhan et al. (2015)). Specifically, Algorithm 7.1 maintains $\mathbf{C}^t$ implicitly via h^t. At each step t, it applies noise-corrected OP1 and OP2 to construct a loss matrix $(\mathbf{L}^t)'$ and solve a linear minimization problem, and to compute an empirical confusion matrix. The final randomized classifier h^T is a convex combination of all the classifiers $h^0, h^1, ..., h^{T-1}$. In Section 7.7, we will formally prove the noise-corrected algorithm is consistent.

Algorithm 7.1 Noise-Corrected Frank-Wolfe (NCFW) Based Algorithm for Monotonic Convex Performance Measures (See Section 7.5 for details.)

```
1: Input: 1) Performance measure $\psi : [0,1]^{n\times n} \to \mathbb{R}_+$ that is convex over $\mathcal{C}_D$; 2) Noisy training
   sample $\widetilde{S} = ((x_i, \widetilde{y}_i))_{i=1}^m \in (\mathcal{X}\times\mathcal{Y})^m$; 3) Noise matrix $\mathbf{T}$ (or estimated noise matrix $\widehat{\mathbf{T}}$)
2: Parameter: Number of iterative steps $T \in \mathbb{N}$
3: Split $\widetilde{S}$ into $\widetilde{S}_1$ and $\widetilde{S}_2$, each with size $\frac{m}{2}$
4: Run a CPE learner on $\widetilde{S}_1$: $\widehat{\widetilde{\boldsymbol{\eta}}} = \text{CPE}(\widetilde{S}_1)$
5: Initialize: $h^0 : \mathcal{X} \to \Delta_n, \quad \mathbf{C}^0 = \widehat{\mathbf{C}}^{\widetilde{S}_2}[h^0]$
6: for $t = 1$ to $T$ do
7:     Calculate noise-corrected loss-based performance measure $(\mathbf{L}^t)' = (\mathbf{T}^\top)^{-1}\nabla\psi(\mathbf{T}^{-1}\mathbf{C}^{t-1})$
8:     Obtain $\widehat{g}^t = x \mapsto \text{argmin}_{y\in[n]} \widehat{\widetilde{\boldsymbol{\eta}}}(x)^\top(\boldsymbol{\ell}_y^t)'$ and update $h^t = (1-\frac{2}{t+1})h^{t-1} + \frac{2}{t+1}\widehat{g}^t$
9:     Calculate $\boldsymbol{\Gamma}^t = \widehat{\mathbf{C}}^{\widetilde{S}_2}[\widehat{g}^t]$ and update $\mathbf{C}^t = (1-\frac{2}{t+1})\mathbf{C}^{t-1} + \frac{2}{t+1}\boldsymbol{\Gamma}^t$
10: end for
11: Output: $h^T$
```

7.6. Ratio-of-linear Performance Measures

We now move to the next family of non-decomposable performance measures, namely ratio-of-linear performance measures. We start with the definition and an example. Then we will show how to use the noise-corrected OP1 and OP2 described in Section 7.5 to build an algorithm to learn from noisy labels for ratio-of-linear performance measures. We will also provide another view of the algorithm from the perspective of correcting the performance measure ψ.

Definition 7.8 (Ratio-of-linear performance measures). *A performance measure $\psi : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ is* ratio-of-linear *if there are $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ such that for any confusion matrix $\mathbf{C}$, $\langle \mathbf{B}, \mathbf{C}\rangle > 0$ and $\psi(\mathbf{C}) = \frac{\langle \mathbf{A}, \mathbf{C}\rangle}{\langle \mathbf{B}, \mathbf{C}\rangle}$.*

Example 7.3 (Micro F_1 in loss form). *Micro F_1: $\psi(\mathbf{C}) = 1 - \frac{2\sum_{i=2}^n C_{i,i}}{2-\sum_{i=1}^n C_{1,i}-\sum_{i=1}^n C_{i,1}}$.*

Noise-corrected algorithm for ratio-of-linear performance measures. The iterative algorithm based on Bisection method (Lemaréchal, 2006; Narasimhan et al., 2015) follows broadly a similar idea as described in Section 7.5, so we can use the same noise-corrected OP1 and OP2 to modify the algorithm. The noise-corrected algorithm is summarized in Algorithm 7.2. This algorithm applies to ratio-of-linear performance measures ψ, such as Micro F_1. It uses a binary search

Algorithm 7.2 Noise-Corrected Bisection (NCBS) Based Algorithm for Ratio-of-linear Performance Measures (See Section 7.6 for details.)

```
1:  Input: 1) Performance measure $\psi(\mathbf{C}) = \frac{\langle \mathbf{A},\mathbf{C}\rangle}{\langle \mathbf{B},\mathbf{C}\rangle}$ with $\mathbf{A},\mathbf{B} \in \mathbb{R}^{n\times n}$; 2) Noisy training sample $\widetilde{S} = ((x_i, \widetilde{y}_i))_{i=1}^m \in (\mathcal{X}\times\mathcal{Y})^m$; 3) Noise matrix $\mathbf{T}$ (or estimated noise matrix $\widehat{\mathbf{T}}$)
2:  Parameter: Number of iterative steps $T \in \mathbb{N}$
3:  Split $\widetilde{S}$ into $\widetilde{S}_1$ and $\widetilde{S}_2$, each with size $\frac{m}{2}$
4:  Run a CPE learner on $\widetilde{S}_1$: $\widehat{\widetilde{\boldsymbol{\eta}}} = \text{CPE}(\widetilde{S}_1)$
5:  Initialize: $h^0 : \mathcal{X} \to [n], \quad \alpha^0 = 0, \quad \beta^0 = 1$
6:  for $t = 1$ to $T$ do
7:      Calculate noise-corrected loss $(\mathbf{L}^t)' = (\mathbf{T}^\top)^{-1}(\mathbf{A} - \gamma^t\mathbf{B})$ where $\gamma^t = (\alpha^{t-1} + \beta^{t-1})/2$
8:      Obtain $\widehat{g}^t = x \mapsto \text{argmin}_{y\in[n]}\, \widehat{\widetilde{\boldsymbol{\eta}}}(x)^\top(\boldsymbol{\ell}_y^t)'$ and calculate $\boldsymbol{\Gamma}^t = \widehat{\mathbf{C}}^{\widetilde{S}_2}[\widehat{g}^t]$
9:      if $\psi(\mathbf{T}^{-1}\boldsymbol{\Gamma}^t) \le \gamma^t$ then $\alpha^t = \alpha^{t-1}, \beta^t = \gamma^t, h^t = \widehat{g}^t$ else $\alpha^t = \gamma^t, \beta^t = \beta^{t-1}, h^t = h^{t-1}$
10: end for
11: Output: $h^T$
```

approach to find the minimum value of $\min_{\mathbf{C}\in\mathcal{C}_D} \psi(\mathbf{C})$. Note that the form $\mathbf{A} - \gamma\mathbf{B}$ in Line 7 comes from the form of Bayes optimal classifier for ratio-of–linear performance measures in the standard (non-noisy) setting (Theorem 11 of Narasimhan et al. (2015)). Again, Algorithm 7.2 maintains $\mathbf{C}^t$ implicitly via h^t. At each step t, it applies noise-corrected OP1 and OP2 to construct a loss matrix $(\mathbf{L}^t)'$ and solve a linear minimization problem, and to compute an empirical confusion matrix. The final classifier h^T is deterministic. In Section 7.7, we will formally prove the noise-corrected algorithm is consistent.

We also offer another view of Algorithm 7.2 from the perspective of correcting ψ. In particular, we show that one can construct a noise-corrected performance measure $\widetilde{\psi}$, which is also ratio-of-linear. Then one can simply optimize $\widetilde{\psi}$ using a noisy sample to learn a classifier h, and the learned h will also be optimal for the original performance measure ψ w.r.t. the clean distribution D.

Theorem 7.3 (Form of Bayes optimal classifier for ratio-of-linear ψ by correcting ψ)**.** *Consider ratio-of-linear performance measure* $\psi(\mathbf{C}) = \frac{\langle \mathbf{A},\mathbf{C}\rangle}{\langle \mathbf{B},\mathbf{C}\rangle}$ *with* $\langle \mathbf{B}, \mathbf{C}\rangle > 0\ \forall \mathbf{C} \in \mathcal{C}_D$. *Define* noise-corrected *performance measure* $\widetilde{\psi} : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ *by* $\widetilde{\psi} = \psi \circ \mathbf{T}^{-1}$. *Then* $\widetilde{\psi}(\widetilde{\mathbf{C}}) = \frac{\langle (\mathbf{T}^\top)^{-1}\mathbf{A},\widetilde{\mathbf{C}}\rangle}{\langle (\mathbf{T}^\top)^{-1}\mathbf{B},\widetilde{\mathbf{C}}\rangle}$ *with* $\langle (\mathbf{T}^\top)^{-1}\mathbf{B}, \widetilde{\mathbf{C}}\rangle > 0$ *for all* $\widetilde{\mathbf{C}} \in \mathcal{C}_{\widetilde{D}}$. *Moreover, any Bayes optimal classifier for $\widetilde{\psi}$-performance w.r.t. $\widetilde{D}$ is also Bayes optimal for ψ-performance w.r.t. D.*

Proof. By property of adjoint, we have

$$\widetilde{\psi}(\widetilde{\mathbf{C}}) = \psi \circ \mathbf{T}^{-1}(\widetilde{\mathbf{C}}) = \psi\Big(\mathbf{T}^{-1}\widetilde{\mathbf{C}}\Big) = \frac{\langle \mathbf{A}, \mathbf{T}^{-1}\widetilde{\mathbf{C}}\rangle}{\langle \mathbf{B}, \mathbf{T}^{-1}\widetilde{\mathbf{C}}\rangle} = \frac{\langle (\mathbf{T}^{\top})^{-1}\mathbf{A}, \widetilde{\mathbf{C}}\rangle}{\langle (\mathbf{T}^{\top})^{-1}\mathbf{B}, \widetilde{\mathbf{C}}\rangle}\,.$$

For all $\widetilde{\mathbf{C}} \in \mathcal{C}_{\widetilde{D}}$, there exists $\mathbf{C} \in \mathcal{C}_D$ such that $\widetilde{\mathbf{C}} = \mathbf{T}\mathbf{C}$. So,

$$\langle (\mathbf{T}^{\top})^{-1}\mathbf{B}, \widetilde{\mathbf{C}}\rangle = \langle \mathbf{B}, \mathbf{C}\rangle > 0\,.$$

This shows $\langle (\mathbf{T}^{\top})^{-1}\mathbf{B}, \widetilde{\mathbf{C}}\rangle > 0$ for all $\widetilde{\mathbf{C}} \in \mathcal{C}_{\widetilde{D}}$.

Let h^* be a Bayes optimal classifier for $\widetilde{\psi}$-performance w.r.t. $\widetilde{D}$ (the existence of such a classifier is guaranteed by Theorem 11 of Narasimhan et al. (2015)). So,

$$\begin{aligned}
\inf_{\widetilde{C} \in \mathcal{C}_{\widetilde{D}}} \widetilde{\psi}(\widetilde{\mathbf{C}}) &= \widetilde{\psi}(\mathbf{C}^{\widetilde{D}}[h^*]) \\
&= \frac{\langle (\mathbf{T}^{\top})^{-1}\mathbf{A}, \mathbf{C}^{\widetilde{D}}[h^*]\rangle}{\langle (\mathbf{T}^{\top})^{-1}\mathbf{B}, \mathbf{C}^{\widetilde{D}}[h^*]\rangle} \\
&= \frac{\langle (\mathbf{T}^{\top})^{-1}\mathbf{A}, \mathbf{T}\mathbf{C}^{D}[h^*]\rangle}{\langle (\mathbf{T}^{\top})^{-1}\mathbf{B}, \mathbf{T}\mathbf{C}^{D}[h^*]\rangle} \\
&= \frac{\langle \mathbf{A}, \mathbf{C}^{D}[h^*]\rangle}{\langle \mathbf{B}, \mathbf{C}^{D}[h^*]\rangle}\,.
\end{aligned}$$

For all $\mathbf{C} \in \mathcal{C}_D$,

$$\begin{aligned}
\psi(\mathbf{C}) &= \frac{\langle \mathbf{A}, \mathbf{C}\rangle}{\langle \mathbf{B}, \mathbf{C}\rangle} \\
&\geq \inf_{\mathbf{C}' \in \mathcal{C}_D} \frac{\langle \mathbf{A}, \mathbf{C}'\rangle}{\langle \mathbf{B}, \mathbf{C}'\rangle} \\
&= \inf_{\mathbf{C}' \in \mathcal{C}_D} \frac{\langle (\mathbf{T}^{\top})^{-1}\mathbf{A}, \mathbf{T}\mathbf{C}'\rangle}{\langle (\mathbf{T}^{\top})^{-1}\mathbf{B}, \mathbf{T}\mathbf{C}'\rangle} \\
&= \inf_{\widetilde{\mathbf{C}} \in \mathcal{C}_{\widetilde{D}}} \widetilde{\psi}(\widetilde{\mathbf{C}}) \\
&= \frac{\langle \mathbf{A}, \mathbf{C}^{D}[h^*]\rangle}{\langle \mathbf{B}, \mathbf{C}^{D}[h^*]\rangle}\,.
\end{aligned}$$

This shows h^* is also Bayes optimal for ψ-performance w.r.t. D. $\square$

Therefore, one can view Algorithm 7.2 as finding the Bayes optimal classifier for $\widetilde{\psi}$ w.r.t. the noisy distribution $\widetilde{D}$, which in turn is also Bayes optimal for ψ w.r.t. the clean distribution D. This view is reminiscent of the Unbiased Estimator approach in van Rooyen and Williamson (2017) and Backward method in Patrini et al. (2017), in which one optimizes noise-corrected surrogate losses using a noisy sample to learn classifiers that are optimal w.r.t. the clean distribution.

7.7. Consistency and Regret Bounds

In this section, we derive quantitative regret bounds for our noise-corrected algorithms. Our results show that when the CPE learner used in the algorithms is consistent (i.e., it converges to the noisy class probabilities), then the noise-corrected algorithms are consistent, i.e., they can output classifiers whose ψ-performance converges to the Bayes optimal ψ-performance w.r.t. D as the size of the noisy training sample $\widetilde{S}$ increases. In addition, we provide regret bounds for our algorithms when estimated $\widehat{\mathbf{T}}$ is used instead of $\mathbf{T}$. To start, we formally define what it means for a learning algorithm to be ψ-consistent when learning from noisy labels.

Definition 7.9 (ψ-regret)**.** *For any function $\psi : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ and classifier $h : \mathcal{X} \to \Delta_n$, define ψ-regret of h w.r.t. D as the difference between ψ-performance of h and the Bayes optimal ψ-performance:* $\text{regret}^{\psi}_{D}[h] = \Psi^{\psi}_{D}[h] - \Psi^{\psi,*}_{D}$.

Definition 7.10 (ψ-consistent algorithm when learning from noisy labels)**.** *For $\psi : \mathbb{R}^{n\times n} \to \mathbb{R}_+$, we say a multiclass algorithm $\mathcal{A} : \cup_{m=1}^{\infty} \widetilde{D}^m \to (\mathcal{X} \to \Delta_n)$, which given a noisy sample $\widetilde{S}$ of size m outputs a (randomized) classifier $\mathcal{A}(\widetilde{S})$, is* consistent *for ψ w.r.t. D if for all $\epsilon > 0$:*

$$\mathbf{P}_{\widetilde{S}\sim\widetilde{D}^m}\big(\text{regret}^{\psi}_{D}[\mathcal{A}(\widetilde{S})] > \epsilon\big) \to 0 \quad \textit{as} \quad m \to \infty\,.$$

7.7.1. Regret Bounds with Known $\mathbf{T}$

Below, we provide guarantees for the noise-corrected OP1 and OP2 (Lemma 7.4 and Lemma 7.5). They are then used in deriving the regret bounds.

Lemma 7.4 (Guarantee for noise-corrected OP1)**.** *Let $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ be the CPE model learned from* *noisy sample $\widetilde{S}$. Then*

$$\begin{aligned}&\mathbf{E}_X\left[\left\|\mathbf{T}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\boldsymbol{\eta}(X)\right\|_1\right]\\&\quad\leq\left\|\mathbf{T}^{-1}\right\|_1\cdot\mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\widetilde{\boldsymbol{\eta}}(X)\right\|_1\right].\end{aligned}$$

Proof.

$$\mathbf{E}_X\left[\left\|\mathbf{T}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\boldsymbol{\eta}(X)\right\|_1\right]=\mathbf{E}_X\left[\left\|\mathbf{T}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\mathbf{T}^{-1}\widetilde{\boldsymbol{\eta}}(X)\right\|_1\right]\leq\left\|\mathbf{T}^{-1}\right\|_1\cdot\mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\widetilde{\boldsymbol{\eta}}(X)\right\|_1\right].$$

□

Lemma 7.5 (Guarantee for noise-corrected OP2)**.** *For $h:\mathcal{X}\to\Delta_n$, let $\widehat{\mathbf{C}}^{\widetilde{S}}[h]$ be the empirical confusion matrix w.r.t. noisy sample $\widetilde{S}$ (computed similarly as $\widehat{\mathbf{C}}^{S}[h]$ in Eq. (7.1)). Then*

$$\begin{aligned}&\left\|\mathbf{C}^D[h]-\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\mathrm{vec},\infty}\\&\quad\leq n\left\|\mathbf{T}^{-1}\right\|_1\cdot\left\|\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\mathrm{vec},\infty}.\end{aligned}$$

Proof.

$$\begin{aligned}\left\|\mathbf{C}^D[h]-\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\mathrm{vec},\infty}&=\left\|\mathbf{T}^{-1}\mathbf{C}^{\widetilde{D}}[h]-\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\mathrm{vec},\infty}\\&\leq\left\|\mathbf{T}^{-1}\mathbf{C}^{\widetilde{D}}[h]-\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_1\\&\leq\left\|\mathbf{T}^{-1}\right\|_1\cdot\left\|\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_1\\&\leq n\left\|\mathbf{T}^{-1}\right\|_1\cdot\left\|\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\mathrm{vec},\infty}.\end{aligned}$$

□

Notes for Lemma 7.4 and Lemma 7.5: In Lemma 7.4, $\mathbf{T}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(x)$ might be viewed as an estimate for $\boldsymbol{\eta}(x)$. If the CPE model used is consistent (i.e., $\mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\widetilde{\boldsymbol{\eta}}(X)\right\|_1\right]\to 0$ as the sample size increases), then this estimation is consistent as well. Similarly, in Lemma 7.5, $\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]$ might be viewed as an estimate for $\mathbf{C}^D[h]$. Because the confusion matrix estimator in Eq. (7.1) is consistent (i.e., $\|\mathbf{C}^{\widetilde{D}}[h] - \widehat{\mathbf{C}}^{\widetilde{S}}[h]\|_{\text{vec},\infty} \to 0$ as the sample size increases) as shown in Lemma 15 of Narasimhan et al. (2015), this estimation is also consistent. $\|\mathbf{T}^{-1}\|_1$ might be viewed as a constant capturing the overall amount of label noise.

Theorem 7.6 (ψ-regret bound for Algorithm 7.1)**.** *Let $\psi : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ be monotonic convex over $\mathcal{C}_D$, and L-Lipschitz and β-smooth w.r.t. L_1 norm.*[27] *Noisy sample $\widetilde{S} = ((x_i, \widetilde{y}_i))_{i=1}^m \in (\mathcal{X} \times [n])^m$ is drawn randomly from $\widetilde{D}^m$. Let $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ be the CPE model learned from $\widetilde{S}_1$ as in Algorithm 7.1. Then for $\delta \in (0, 1]$, with probability at least $1 - \delta$ (over $\widetilde{S} \sim \widetilde{D}^m$), we have*

$$\begin{aligned}\text{regret}_D^{\psi}[h^T] \le 4L\|\mathbf{T}^{-1}\|_1 \mathbf{E}_X\left[\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\|_1\right] + \frac{8\beta}{T+2} \\ + 4\sqrt{2}\beta n^3 C\|\mathbf{T}^{-1}\|_1 \sqrt{\frac{n^2\log(n)\log(m) + \log(n^2/\delta)}{m}},\end{aligned}$$

where $C > 0$ is a distribution-independent constant.

Theorem 7.7 (ψ-regret bound for Algorithm 7.2)**.** *Let $\psi(\mathbf{C}) = \frac{\langle \mathbf{A}, \mathbf{C}\rangle}{\langle \mathbf{B}, \mathbf{C}\rangle}$ for $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ with $\min_{\mathbf{C}\in\mathcal{C}_D} \langle \mathbf{B}, \mathbf{C}\rangle \ge b$ for some $b > 0$. Noisy sample $\widetilde{S} = ((x_i, \widetilde{y}_i))_{i=1}^m \in (\mathcal{X} \times [n])^m$ is drawn randomly from $\widetilde{D}^m$. Let $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ be the CPE model learned from $\widetilde{S}_1$ as in Algorithm 7.2. Then for $\delta \in (0, 1]$, with probability at least $1 - \delta$ (over $\widetilde{S} \sim \widetilde{D}^m$), we have*

$$\begin{aligned}\text{regret}_D^{\psi}[h^T] \le 2\tau\|\mathbf{T}^{-1}\|_1 \mathbf{E}_X\left[\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\|_1\right] + 2^{-T} \\ + 2\sqrt{2}\tau n C\|\mathbf{T}^{-1}\|_1 \sqrt{\frac{n^2\log(n)\log(m) + \log(n^2/\delta)}{m}},\end{aligned}$$

where $\tau = \frac{1}{b}\left(\|\mathbf{A}\|_{\text{vec},1} + \|\mathbf{B}\|_{\text{vec},1}\right)$ and $C > 0$ is a distribution-independent constant.

In particular, using a strongly/strictly proper composite surrogate loss (e.g., multiclass logistic regression loss/cross entropy loss with softmax function) over a universal function class (with suitable regularization) to learn a CPE model ensures a consistent noisy class probability estimation, i.e., $\mathbf{E}_X\left[\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\|_1\right] \to 0$ as the sample size increases (Agarwal, 2014; Williamson et al., 2016;

[27] A function ψ is β-smooth if its gradient is β-Lipschitz.

Zhang et al., 2021). This leads to the convergence of the regret to zero as $m \to \infty, T \to \infty$. Also, as the amount of label noise (captured by $\left\|\mathbf{T}^{-1}\right\|_1$) increases, the bounds get larger; one might therefore need a larger noisy sample size to achieve the same level of ψ-regret w.r.t. D. Our synthetic experiments also confirm this sample complexity behavior.

Notes for our proof of Theorem 7.6: Our proof of Theorem 7.6 uses Lemma 14, Lemma 15 and Theorem 16 in Narasimhan et al. (2015), along with their proofs. We include a proposition below that summarizes the key aspects of Lemma 14, Lemma 15 and Theorem 16 in Narasimhan et al. (2015) that we make use of (with slight modification in order for it to be consistent with our notations).

Proposition 7.8 (ψ-regret bound of Frank-Wolfe based algorithm in the non-noisy setting; Theorem 16 in Narasimhan et al. (2015))**.** *Let $\psi : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ be monotonic convex over $\mathcal{C}_D$, and L-Lipschitz and β-smooth w.r.t. L_1 norm. Let clean sample $S = ((x_i, y_i))_{i=1}^m \in (\mathcal{X} \times [n])^m$ be drawn randomly from D^m. Let $\widehat{\boldsymbol{\eta}} : \mathcal{X} \to \Delta_n$ be the CPE model learned from S_1 as in the Frank-Wolfe based algorithm, and $h_S^{\mathrm{FW}} : \mathcal{X} \to \Delta_n$ be the classifier returned after T iterations. Let $\delta \in (0, 1]$. Then with probability $\geq 1 - \delta$ (over $S \sim D^m$),*

$$\begin{aligned}
\mathrm{regret}_D^{\psi}[h_S^{\mathrm{FW}}] &\leq 4L\mathbf{E}_X\left[\left\|\widehat{\boldsymbol{\eta}}(X) - \boldsymbol{\eta}(X)\right\|_1\right] + \frac{8\beta}{T+2} \\
&+ 4\beta n^2 \sup_{h\in\mathcal{H}_{\widehat{\boldsymbol{\eta}}}} \left\|\mathbf{C}^D[h] - \widehat{\mathbf{C}}^{S_2}[h]\right\|_{\mathrm{vec},\infty} \\
&\leq 4L\mathbf{E}_X\left[\left\|\widehat{\boldsymbol{\eta}}(X) - \boldsymbol{\eta}(X)\right\|_1\right] + \frac{8\beta}{T+2} \\
&+ 4\sqrt{2}\beta n^2 C\sqrt{\frac{n^2\log(n)\log(m) + \log(n^2/\delta)}{m}},
\end{aligned}$$

where $C > 0$ is a distribution-independent constant, and

$$\mathcal{H}_{\widehat{\boldsymbol{\eta}}} = \{h : \mathcal{X} \to [n], h(x) = \operatorname*{argmin}_{y\in[n]} \widehat{\boldsymbol{\eta}}(x)^\top \boldsymbol{\ell}_y, \mathbf{L} \in \mathbb{R}^{n\times n}\}. \qquad (7.2)$$

The second '$\leq$' was obtained by Lemma 15 of Narasimhan et al. (2015) and $|S_2| = m/2$.

Proof of Theorem 7.6.

Proof. In Algorithm 7.1, we only have noisy sample $\widetilde{S}$ that is split into $\widetilde{S}_1$ and $\widetilde{S}_2$. We implicitly estimate $\boldsymbol{\eta}$ by $\mathbf{T}^{-1} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$, where $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ is a CPE model learned from $\widetilde{S}_1$. Lemma 7.4 shows an additional factor of $\left\|\mathbf{T}^{-1}\right\|_1$ as a price paid to learn from a noisy sample instead of a clean one. For a classifier h, we estimate $\mathbf{C}^D[h]$ by $\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}_2}[h]$, where $\widehat{\mathbf{C}}^{\widetilde{S}_2}[h]$ is the empirical confusion matrix learned from $\widetilde{S}_2$. Lemma 7.5 shows an additional factor of $n\left\|\mathbf{T}^{-1}\right\|_1$ as a cost to learn from a noisy sample instead of a clean one. Note that $\mathcal{H}_{\widehat{\boldsymbol{\eta}}} = \mathcal{H}_{\widehat{\widetilde{\boldsymbol{\eta}}}}$ for $\widehat{\boldsymbol{\eta}} = \mathbf{T}^{-1} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$. Chaining this reasoning with Proposition 7.8 establishes the claim. $\square$

Notes for our proof of Theorem 7.7: Our proof of Theorem 7.7 uses Lemma 14, Lemma 15 and Theorem 17 in Narasimhan et al. (2015), along with their proofs. We include a proposition below that summarizes the key aspects of Lemma 14, Lemma 15 and Theorem 17 in Narasimhan et al. (2015) that we make use of (with slight modification in order for it to be consistent with our notations).

Proposition 7.9 (ψ-regret bound for bisection based algorithm in the non-noisy setting; Theorem 17 in Narasimhan et al. (2015))**.** *Let $\psi(\mathbf{C}) = \frac{\langle \mathbf{A}, \mathbf{C}\rangle}{\langle \mathbf{B}, \mathbf{C}\rangle}$ for $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ with $\min_{\mathbf{C}\in\mathcal{C}_D} \langle \mathbf{B}, \mathbf{C}\rangle \geq b$ for some $b > 0$. Let clean sample $S = ((x_i, y_i))_{i=1}^m \in (\mathcal{X} \times [n])^m$ be drawn randomly from D^m. Let $\widehat{\boldsymbol{\eta}} : \mathcal{X} \to \Delta_n$ be the CPE model learned from S_1 as in the bisection based algorithm, and $h_S^{\mathrm{BS}} : \mathcal{X} \to \Delta_n$ be the classifier returned after T iterations. Let $\delta \in (0, 1]$. Then with probability $\geq 1 - \delta$ (over $S \sim D^m$),*

$$
\begin{aligned}
\mathrm{regret}_D^{\psi}[h_S^{\mathrm{BS}}] &\leq 2\tau \mathbf{E}_X\left[\left\|\widehat{\boldsymbol{\eta}}(X) - \boldsymbol{\eta}(X)\right\|_1\right] + 2^{-T} \\
&\quad + 2\tau \sup_{h\in\mathcal{H}_{\widehat{\boldsymbol{\eta}}}} \left\|\mathbf{C}^D[h] - \widehat{\mathbf{C}}^{S_2}[h]\right\|_{\mathrm{vec},\infty} \\
&\leq 2\tau \mathbf{E}_X\left[\left\|\widehat{\boldsymbol{\eta}}(X) - \boldsymbol{\eta}(X)\right\|_1\right] + 2^{-T} \\
&\quad + 2\sqrt{2}C\tau\sqrt{\frac{n^2\log(n)\log(m) + \log(n^2/\delta)}{m}},
\end{aligned}
$$

where $\tau = \frac{1}{b}\big(\|\mathbf{A}\|_{\mathrm{vec},1} + \|\mathbf{B}\|_{\mathrm{vec},1}\big)$*,* $C > 0$ *is a distribution-independent constant, and* $\mathcal{H}_{\widehat{\boldsymbol{\eta}}} = \{h : \mathcal{X} \to [n], h(x) = \mathrm{argmin}_{y\in[n]}\, \widehat{\boldsymbol{\eta}}(x)^\top \boldsymbol{\ell}_y, \mathbf{L} \in \mathbb{R}^{n\times n}\}$*. The second '*$\leq$*' was obtained by Lemma 15 of Narasimhan et al. (2015) and* $|S_2| = m/2$*.*

Proof of Theorem 7.7.

Proof. In Algorithm 7.2, we only have noisy sample $\widetilde{S}$ that is split into $\widetilde{S}_1$ and $\widetilde{S}_2$. We implicitly estimate $\boldsymbol{\eta}$ by $\mathbf{T}^{-1} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$, where $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ is a CPE model learned from $\widetilde{S}_1$. Lemma 7.4 shows an additional factor of $\left\|\mathbf{T}^{-1}\right\|_1$ as a price paid to learn from a noisy sample instead of a clean one. For a classifier h, we estimate $\mathbf{C}^D[h]$ by $\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}_2}[h]$, where $\widehat{\mathbf{C}}^{\widetilde{S}_2}[h]$ is the empirical confusion matrix learned from $\widetilde{S}_2$. Lemma 7.5 shows an additional factor of $n\left\|\mathbf{T}^{-1}\right\|_1$ as a cost to learn from a noisy sample instead of a clean one. Note that $\mathcal{H}_{\widehat{\boldsymbol{\eta}}} = \mathcal{H}_{\widehat{\widetilde{\boldsymbol{\eta}}}}$ for $\widehat{\boldsymbol{\eta}} = \mathbf{T}^{-1} \circ \widehat{\widetilde{\boldsymbol{\eta}}}$. Chaining this reasoning with Proposition 7.9 establishes the claim. □

7.7.2. Regret Bounds with Estimated $\widehat{\mathbf{T}}$

When noise matrix $\mathbf{T}$ is not known, one may need to use estimated $\widehat{\mathbf{T}}$. Several methods have been developed to estimate $\mathbf{T}$ from the noisy sample (Xia et al., 2019; Yao et al., 2020; Li et al., 2021). Below, we provide regret bounds for our noise-corrected algorithms when estimated $\widehat{\mathbf{T}}$ is used. They involve an additional factor $\left\|\widehat{\mathbf{T}}^{-1} - \mathbf{T}^{-1}\right\|_1$ that quantifies the quality of the estimated $\widehat{\mathbf{T}}$.

Lemma 7.10 (Guarantee for noise-corrected OP1 with estimated $\widehat{\mathbf{T}}$)**.** *Let* $\widehat{\widetilde{\boldsymbol{\eta}}} : \mathcal{X} \to \Delta_n$ *be the CPE model learned from noisy sample* $\widetilde{S}$*. Let* $\widehat{\mathbf{T}}$ *be an estimate of* $\mathbf{T}$*. Then*

$$\begin{aligned}
&\mathbf{E}_X\left[\left\|\widehat{\mathbf{T}}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \boldsymbol{\eta}(X)\right\|_1\right] \\
&\leq \left\|\mathbf{T}^{-1}\right\|_1 \cdot \mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\right\|_1\right] + \left\|\widehat{\mathbf{T}}^{-1} - \mathbf{T}^{-1}\right\|_1 .
\end{aligned}$$

Proof.

$$\begin{aligned}
&\mathbf{E}_X\left[\left\|\widehat{\mathbf{T}}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\boldsymbol{\eta}(X)\right\|_1\right]\\
&=\mathbf{E}_X\left[\left\|\widehat{\mathbf{T}}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\mathbf{T}^{-1}\widetilde{\boldsymbol{\eta}}(X)\right\|_1\right]\\
&=\mathbf{E}_X\left[\left\|\widehat{\mathbf{T}}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\mathbf{T}^{-1}\widetilde{\boldsymbol{\eta}}(X)+\mathbf{T}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\mathbf{T}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)\right\|_1\right]\\
&\le\mathbf{E}_X\left[\left\|\mathbf{T}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\mathbf{T}^{-1}\widetilde{\boldsymbol{\eta}}(X)\right\|_1+\left\|\widehat{\mathbf{T}}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\mathbf{T}^{-1}\widehat{\widetilde{\boldsymbol{\eta}}}(X)\right\|_1\right]\\
&\le\left\|\mathbf{T}^{-1}\right\|_1\cdot\mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\widetilde{\boldsymbol{\eta}}(X)\right\|_1\right]+\left\|\widehat{\mathbf{T}}^{-1}-\mathbf{T}^{-1}\right\|_1\cdot\mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X)\right\|_1\right]\\
&\le\left\|\mathbf{T}^{-1}\right\|_1\cdot\mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\widetilde{\boldsymbol{\eta}}(X)\right\|_1\right]+\left\|\widehat{\mathbf{T}}^{-1}-\mathbf{T}^{-1}\right\|_1.
\end{aligned}$$

□

Lemma 7.11 (Guarantee for noise-corrected OP2 with estimated $\widehat{\mathbf{T}}$)**.** *For $h:\mathcal{X}\to\Delta_n$, let $\widehat{\mathbf{C}}^{\widetilde{S}}[h]$ be the empirical confusion matrix w.r.t. noisy sample $\widetilde{S}$ (computed similarly as $\widehat{\mathbf{C}}^{S}[h]$ in Eq. (7.1)). Let $\widehat{\mathbf{T}}$ be an estimate of $\mathbf{T}$. Then*

$$\begin{aligned}
&\left\|\mathbf{C}^D[h]-\widehat{\mathbf{T}}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\text{vec},\infty}\\
&\le n\left\|\mathbf{T}^{-1}\right\|_1\cdot\left\|\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\text{vec},\infty}+\left\|\widehat{\mathbf{T}}^{-1}-\mathbf{T}^{-1}\right\|_1.
\end{aligned}$$

Proof.

$$
\begin{aligned}
&\left\|\mathbf{C}^{D}[h]-\widehat{\mathbf{T}}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\text{vec},\infty} \\
&=\left\|\mathbf{T}^{-1}\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{T}}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\text{vec},\infty} \\
&=\left\|\mathbf{T}^{-1}\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{T}}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]+\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]-\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\text{vec},\infty} \\
&\leq\left\|\mathbf{T}^{-1}\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{T}}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]+\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]-\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{1} \\
&\leq\left\|\mathbf{T}^{-1}\mathbf{C}^{\widetilde{D}}[h]-\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{1}+\left\|\mathbf{T}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]-\widehat{\mathbf{T}}^{-1}\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{1} \\
&\leq\left\|\mathbf{T}^{-1}\right\|_{1}\cdot\left\|\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{1}+\left\|(\mathbf{T}^{-1}-\widehat{\mathbf{T}}^{-1})\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{1} \\
&\leq n\left\|\mathbf{T}^{-1}\right\|_{1}\cdot\left\|\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\text{vec},\infty}+\left\|\mathbf{T}^{-1}-\widehat{\mathbf{T}}^{-1}\right\|_{1}\cdot\left\|\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{1} \\
&\leq n\left\|\mathbf{T}^{-1}\right\|_{1}\cdot\left\|\mathbf{C}^{\widetilde{D}}[h]-\widehat{\mathbf{C}}^{\widetilde{S}}[h]\right\|_{\text{vec},\infty}+\left\|\mathbf{T}^{-1}-\widehat{\mathbf{T}}^{-1}\right\|_{1}.
\end{aligned}
$$

□

Theorem 7.12 (ψ-regret bound for Algorithm 7.1 with estimated $\widehat{\mathbf{T}}$)**.** *Let ψ, $\widetilde{S}$ and $\widehat{\widetilde{\boldsymbol{\eta}}}$ be specified as in Theorem 7.6. Let $\widehat{\mathbf{T}}$ be an estimate of $\mathbf{T}$. Then for $\delta \in (0,1]$, with probability at least $1-\delta$ (over $\widetilde{S} \sim \widetilde{D}^m$), we have*

$$
\begin{aligned}
\text{regret}_D^{\psi}[h^T] \leq\ & 4L\left\|\mathbf{T}^{-1}\right\|_1 \mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X)-\widetilde{\boldsymbol{\eta}}(X)\right\|_1\right]+\frac{8\beta}{T+2} \\
&+4\sqrt{2}\beta n^3 C\left\|\mathbf{T}^{-1}\right\|_1\sqrt{\frac{n^2\log(n)\log(m)+\log(n^2/\delta)}{m}} \\
&+(4L+4\beta n^2)\left\|\widehat{\mathbf{T}}^{-1}-\mathbf{T}^{-1}\right\|_1,
\end{aligned}
$$

where $C>0$ is a distribution-independent constant.

Proof. Similar as in the proof of Theorem 7.6, chaining Lemma 7.10 and Lemma 7.11 with Proposition 7.8 establishes the claim. □

Theorem 7.13 (ψ-regret bound for Algorithm 7.2 with estimated $\widehat{\mathbf{T}}$)**.** *Let ψ, $\widetilde{S}$ and $\widehat{\widetilde{\boldsymbol{\eta}}}$ be specified as in Theorem 7.7. Let $\widehat{\mathbf{T}}$ be an estimate of $\mathbf{T}$. Then for $\delta \in (0,1]$, with probability at least $1-\delta$*

(over $\widetilde{S} \sim \widetilde{D}^m$), we have

$$\begin{aligned}\text{regret}_D^{\psi}[h^T] \leq 2\tau \left\|\mathbf{T}^{-1}\right\|_1 \mathbf{E}_X\left[\left\|\widehat{\widetilde{\boldsymbol{\eta}}}(X) - \widetilde{\boldsymbol{\eta}}(X)\right\|_1\right] + 2^{-T} \\ + 2\sqrt{2}\tau n C \left\|\mathbf{T}^{-1}\right\|_1 \sqrt{\frac{n^2 \log(n)\log(m) + \log(n^2/\delta)}{m}} \\ + 4\tau \left\|\widehat{\mathbf{T}}^{-1} - \mathbf{T}^{-1}\right\|_1,\end{aligned}$$

where $\tau = \frac{1}{b}\left(\|\mathbf{A}\|_{\text{vec},1} + \|\mathbf{B}\|_{\text{vec},1}\right)$ and $C > 0$ is a distribution-independent constant.

Proof. Similar as in the proof of Theorem 7.7, chaining Lemma 7.10 and Lemma 7.11 with Proposition 7.9 establishes the claim. □

7.8. Experiments

We conducted two sets of experiments. In the first set of experiments, we generated synthetic data and tested the sample complexity behavior of our algorithms. In the second set of experiments, we used real data and compared our algorithms with other algorithms. Our code is available at https://github.com/moshimowang/noisy-labels-non-decomposable.

7.8.1. Sample Complexity Behavior

We tested the sample complexity behavior of our algorithm on synthetic data generated from a known distribution.

Specifically, we constructed a 3-class problem over a 2-dimensional instance space $\mathcal{X} = \mathbb{R}^2$ as follows. Instances $\mathbf{x}$ were generated according to a fixed Gaussian mixture distribution. The class probability function $\boldsymbol{\eta} : \mathcal{X} \to \Delta_3$ was $\eta_y(\mathbf{x}) = \frac{\exp(\mathbf{w}_y^\top \mathbf{x} + b_y)}{\sum_{y'=1}^{3} \exp(\mathbf{w}_{y'}^\top \mathbf{x} + b_{y'})}$ for some fixed weight vectors $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \in \mathbb{R}^2$ and bias terms $b_1, b_2, b_3 \in \mathbb{R}$. Given an instance $\mathbf{x}$, a clean label y was drawn randomly according to $\boldsymbol{\eta}(\mathbf{x})$. Then y was flipped to a noisy label $\widetilde{y}$ according to the probabilities in the y-th column of $\mathbf{T}$, where $\mathbf{T}$ is a prescribed column stochastic noise matrix.

We generated noise matrices by choosing a noise level $\sigma \in [0, 1]$ and setting diagonal entries of $\mathbf{T}$ to $1 - \sigma$ and off-diagonal entries of $\mathbf{T}$ to $\frac{\sigma}{2}$. We tested the sample complexity behavior of our algorithms

Table 7.2: Details of Data Sets Used in Section 7.8

Data set	**# instances**	**# classes**	**# features**
vehicle	846	4	18
pageblocks	5,473	5	10
satimage	6,435	6	36
covtype	581,012	7	14
abalone	4,177	12	8

for a variety of noise matrices $\mathbf{T}$ with increasing values of noise level $\sigma = 0.1, 0.2, 0.3, 0.4, 0.6$. The corresponding values of $\|\mathbf{T}^{-1}\|_1$ were also increasing. The non-decomposable performance measures were Q-mean and Micro F_1. We applied Algorithm 7.1 for Q-mean with $T = 5000$ and Algorithm 7.2 for Micro F_1 with $T = 200$. In both algorithms, the CPE learner was implemented by minimizing the multiclass logistic regression loss (aka. cross entropy loss with softmax function) over linear functions. We ran the algorithms on noisy training samples with increasing sizes ($10^2, 10^3$, 10^4, 10^5), and measured the performance on a clean test set of 10^5 examples. The results are shown in Figure 7.2. The top plot shows results for Q-mean. The bottom plot shows results for Micro F_1. We see that, as suggested by our regret bounds, as $\|\mathbf{T}^{-1}\|_1$ increases (i.e., more noise), the sample size required to achieve a given level of performance generally increases.

7.8.2. Comparison with Other Algorithms

We conducted experiments on several real data sets taken from UCI Machine Learning Repository (Dua and Graff, 2017). Details of the data sets are in Table 7.2. We compared our noise-corrected algorithms (NCFW and NCBS) with the baseline Frank-Wolfe (FW) and Bisection (BS) based methods of Narasimhan et al. (2015, 2022) that were designed for the standard (non-noisy) learning setting, as well as various previously proposed noise-corrected versions of multiclass logistic regression (NCLR-Backward (van Rooyen and Williamson, 2017; Patrini et al., 2017), NCLR-Forward (Patrini et al., 2017), and NCLR-Plug-in (Zhang et al., 2021)). We used the authors' implementations for FW and BS.[28] To ensure a fair comparison, we also implemented our algorithms in the same framework. Different variants of NCLR were implemented based on Patrini et al. (2017).[29] A linear function class is used in all algorithms. For NCFW, NCBS, FW, and BS, the linear model

[28]https://github.com/shivtavker/constrained-classification.
[29]https://github.com/giorgiop/loss-correction.

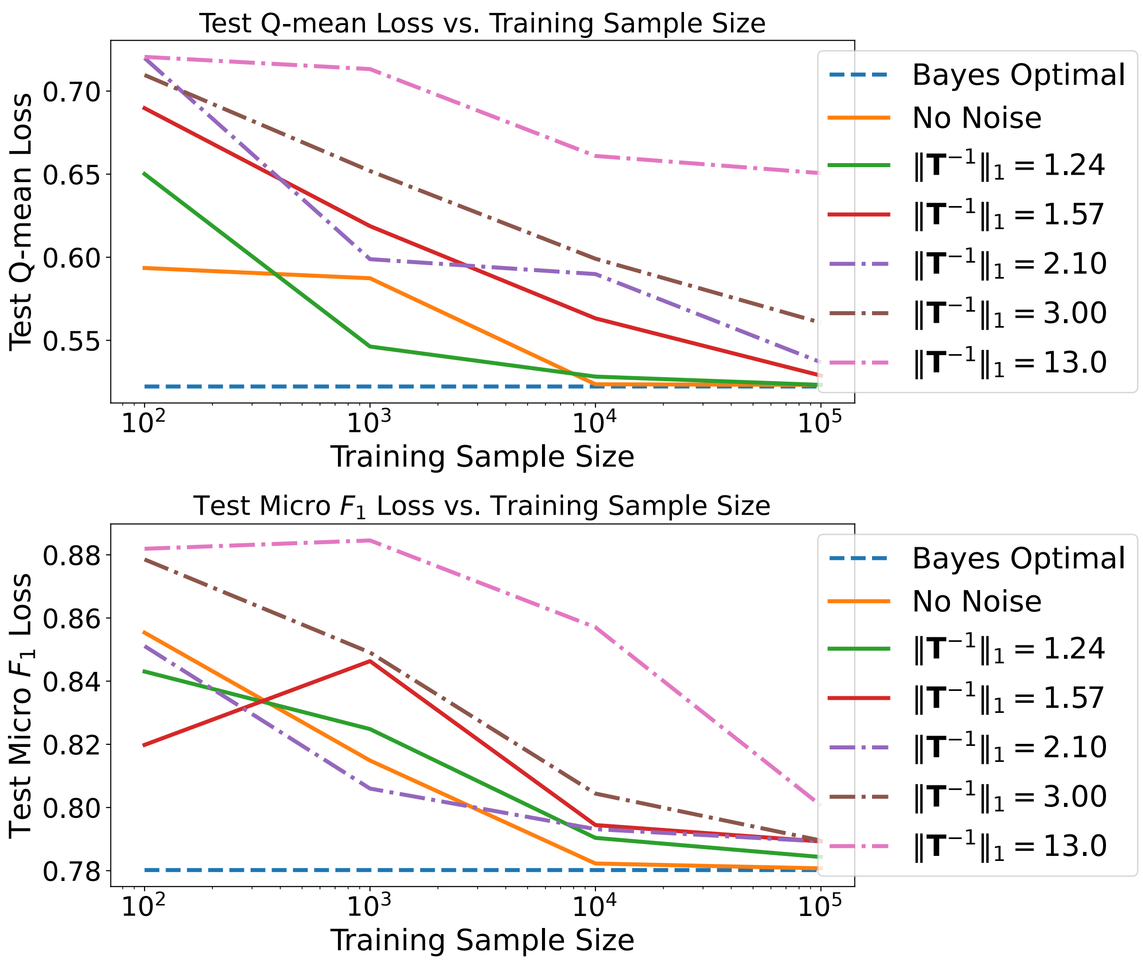

Figure 7.2: Sample Complexity Behavior of Our Noise-corrected Algorithms NCFW (top) and NCBS (bottom)

CPE learner was implemented using scikit-learn (Pedregosa et al., 2011). The different variants of NCLR, implemented in TensorFlow (Abadi et al., 2015), also used a linear function class. In all cases, regularization parameters were chosen by cross-validation.

To generate noise matrices $\mathbf{T}$, we chose a noise level $\sigma \in [0, 1]$, set diagonal entries of $\mathbf{T}$ to $1 - \sigma$, and set off-diagonal entries uniformly at random from $[0, 1]$ so that each column of $\mathbf{T}$ sums to 1. This makes sure that on average, 100σ percent of clean labels were flipped to other labels, i.e., $\sigma \approx \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}(y_i \neq \widetilde{y}_i)$. Therefore, higher value of σ means a higher noise level. We generated 4 noise matrices with $\sigma = 0.1, 0.2, 0.3, 0.4$ according to this process. Training labels were flipped randomly according to the prescribed noise matrix $\mathbf{T}$.

We ran FW and NCFW for $T = 5000$ iterative steps, and ran BS and NCBS for $T = 200$ iterative steps. Performance of the learned model was then measured on a clean test set. The results are summarized in Table 7.3 (for H-mean loss) and Table 7.4 (for Micro F_1 loss), shown as the mean (with standard error of the mean in parentheses) over 5 random $7:3$ train-test splits. Higher σ is a high noise level. For each data set and each noise level, the best performance is shown in bold font. The results for G-mean loss and Q-mean loss can be found in Table 7.5 and Table 7.6. As expected, in most cases, NCFW and NCBS outperform FW and BS, respectively, and they outperform variants of noise-corrected multiclass logistic regression as well.

7.9. Conclusion

We have provided the first known noise-corrected algorithms, NCFW and NCBS, for multiclass monotonic convex and ratio-of-linear performance measures under general class-conditional noise models. We have also provided regret bounds for our algorithms showing that they are consistent w.r.t. the clean data distribution, and quantifying the effect of noise on their sample complexity. Our experiments have demonstrated the effectiveness of our algorithms in handling label noise. For settings where the noise matrix $\mathbf{T}$ may be unknown, approaches for estimating $\mathbf{T}$ have been proposed in the literature. These can be combined with our algorithms where needed, and we have also provided regret bounds for our algorithms when estimated $\widehat{\mathbf{T}}$ is used.

Table 7.3: Comparison with Other Algorithms for H-mean Loss

Data sets	**Algorithms**	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$	$\sigma = 0.4$
vehicle	FW	0.255 (0.005)	0.267 (0.011)	0.309 (0.007)	0.373 (0.010)
	NCFW	**0.254 (0.007)**	**0.266 (0.015)**	**0.307 (0.008)**	**0.338 (0.013)**
	NCLR-Backward	0.482 (0.024)	0.573 (0.020)	0.512 (0.033)	0.508 (0.021)
	NCLR-Forward	0.512 (0.035)	0.563 (0.021)	0.570 (0.011)	0.563 (0.029)
	NCLR-Plug-in	0.515 (0.016)	0.567 (0.010)	0.517 (0.028)	0.540 (0.015)
pageblocks	FW	0.380 (0.041)	0.286 (0.011)	0.633 (0.097)	0.627 (0.066)
	NCFW	**0.269 (0.017)**	**0.253 (0.006)**	**0.535 (0.019)**	**0.528 (0.034)**
	NCLR-Backward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Forward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Plug-in	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	0.926 (0.066)
satimage	FW	0.188 (0.005)	0.224 (0.005)	0.247 (0.006)	0.340 (0.006)
	NCFW	**0.186 (0.005)**	**0.222 (0.006)**	**0.230 (0.005)**	**0.300 (0.005)**
	NCLR-Backward	0.556 (0.023)	0.630 (0.026)	0.685 (0.043)	0.960 (0.012)
	NCLR-Forward	0.542 (0.020)	0.522 (0.011)	0.612 (0.030)	0.877 (0.017)
	NCLR-Plug-in	0.679 (0.049)	0.793 (0.023)	0.854 (0.031)	0.902 (0.021)
covtype	FW	0.569 (0.001)	0.591 (0.001)	0.771 (0.010)	0.741 (0.009)
	NCFW	**0.525 (0.001)**	**0.569 (0.001)**	**0.606 (0.002)**	**0.706 (0.004)**
	NCLR-Backward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Forward	0.995 (0.001)	0.987 (0.002)	0.980 (0.002)	0.963 (0.005)
	NCLR-Plug-in	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
abalone	FW	0.806 (0.014)	0.799 (0.006)	0.812 (0.006)	**0.801 (0.010)**
	NCFW	**0.797 (0.008)**	**0.795 (0.006)**	**0.804 (0.008)**	0.814 (0.010)
	NCLR-Backward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Forward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Plug-in	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)

Table 7.4: Comparison with Other Algorithms for Micro F_1 Loss

Data sets	**Algorithms**	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$	$\sigma = 0.4$
vehicle	BS	0.268 (0.007)	0.307 (0.006)	0.323 (0.013)	0.388 (0.012)
	NCBS	**0.264 (0.008)**	**0.299 (0.006)**	**0.314 (0.011)**	**0.346 (0.009)**
	NCLR-Backward	0.435 (0.020)	0.524 (0.006)	0.508 (0.027)	0.494 (0.028)
	NCLR-Forward	0.470 (0.031)	0.483 (0.020)	0.548 (0.016)	0.553 (0.030)
	NCLR-Plug-in	0.494 (0.022)	0.488 (0.023)	0.491 (0.025)	0.521 (0.015)
pageblocks	BS	**0.231 (0.009)**	0.323 (0.011)	0.862 (0.005)	0.899 (0.006)
	NCBS	0.251 (0.008)	**0.261 (0.010)**	**0.320 (0.006)**	**0.404 (0.020)**
	NCLR-Backward	0.515 (0.050)	0.457 (0.055)	0.756 (0.079)	0.510 (0.048)
	NCLR-Forward	0.823 (0.083)	0.880 (0.048)	0.743 (0.113)	0.832 (0.093)
	NCLR-Plug-in	0.609 (0.096)	0.595 (0.107)	0.568 (0.051)	0.795 (0.045)
satimage	BS	0.219 (0.004)	0.224 (0.002)	0.242 (0.002)	0.313 (0.003)
	NCBS	0.219 (0.004)	0.220 (0.003)	0.236 (0.002)	0.292 (0.002)
	NCLR-Backward	**0.215 (0.004)**	0.222 (0.003)	0.227 (0.004)	0.231 (0.003)
	NCLR-Forward	0.217 (0.003)	**0.214 (0.002)**	**0.213 (0.003)**	**0.221 (0.003)**
	NCLR-Plug-in	0.234 (0.002)	0.236 (0.003)	0.255 (0.002)	0.300 (0.003)
covtype	BS	**0.361 (0.000)**	0.355 (0.000)	**0.362 (0.000)**	**0.362 (0.000)**
	NCBS	**0.361 (0.000)**	**0.352 (0.000)**	**0.362 (0.000)**	**0.362 (0.000)**
	NCLR-Backward	0.384 (0.000)	0.385 (0.000)	0.388 (0.000)	0.390 (0.001)
	NCLR-Forward	0.384 (0.000)	0.380 (0.000)	0.381 (0.001)	0.382 (0.001)
	NCLR-Plug-in	0.398 (0.000)	0.396 (0.001)	0.397 (0.000)	0.397 (0.000)
abalone	BS	0.731 (0.007)	0.746 (0.005)	0.746 (0.003)	**0.750 (0.002)**
	NCBS	**0.729 (0.007)**	**0.743 (0.005)**	**0.740 (0.003)**	0.754 (0.001)
	NCLR-Backward	0.787 (0.005)	0.789 (0.007)	0.797 (0.010)	0.793 (0.011)
	NCLR-Forward	0.774 (0.007)	0.806 (0.004)	0.783 (0.003)	0.794 (0.009)
	NCLR-Plug-in	0.789 (0.005)	0.789 (0.009)	0.803 (0.007)	0.799 (0.010)

Table 7.5: Comparison with Other Algorithms for G-mean Loss

Data sets	**Algorithms**	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$	$\sigma = 0.4$
vehicle	FW	**0.230 (0.007)**	0.249 (0.011)	**0.287 (0.009)**	0.349 (0.011)
	NCFW	0.231 (0.008)	**0.247 (0.012)**	0.289 (0.009)	**0.312 (0.012)**
	NCLR-Backward	0.432 (0.017)	0.519 (0.013)	0.488 (0.034)	0.469 (0.025)
	NCLR-Forward	0.463 (0.030)	0.496 (0.015)	0.523 (0.012)	0.524 (0.025)
	NCLR-Plug-in	0.477 (0.016)	0.503 (0.012)	0.486 (0.021)	0.502 (0.013)
pageblocks	FW	0.331 (0.026)	0.225 (0.009)	0.552 (0.081)	0.563 (0.046)
	NCFW	**0.236 (0.040)**	**0.167 (0.010)**	**0.503 (0.125)**	**0.382 (0.018)**
	NCLR-Backward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Forward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Plug-in	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	0.898 (0.091)
satimage	FW	0.181 (0.004)	0.214 (0.004)	0.225 (0.005)	0.310 (0.005)
	NCFW	**0.180 (0.005)**	**0.213 (0.004)**	**0.212 (0.005)**	**0.272 (0.005)**
	NCLR-Backward	0.368 (0.010)	0.400 (0.013)	0.428 (0.020)	0.684 (0.072)
	NCLR-Forward	0.362 (0.009)	0.351 (0.005)	0.385 (0.011)	0.524 (0.014)
	NCLR-Plug-in	0.439 (0.023)	0.486 (0.012)	0.550 (0.023)	0.638 (0.019)
covtype	FW	0.570 (0.001)	0.577 (0.001)	0.685 (0.004)	0.690 (0.005)
	NCFW	**0.515 (0.001)**	**0.516 (0.001)**	**0.546 (0.001)**	**0.612 (0.004)**
	NCLR-Backward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Forward	0.908 (0.021)	0.875 (0.003)	0.869 (0.002)	0.847 (0.004)
	NCLR-Plug-in	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
abalone	FW	0.779 (0.008)	0.775 (0.002)	0.786 (0.005)	**0.778 (0.005)**
	NCFW	**0.776 (0.007)**	**0.766 (0.001)**	**0.781 (0.007)**	0.784 (0.005)
	NCLR-Backward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Forward	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
	NCLR-Plug-in	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)

Table 7.6: Comparison with Other Algorithms for Q-mean Loss

Data sets	**Algorithms**	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$	$\sigma = 0.4$
vehicle	FW	**0.268 (0.008)**	**0.287 (0.012)**	0.313 (0.009)	0.367 (0.006)
	NCFW	0.276 (0.005)	0.293 (0.009)	**0.310 (0.010)**	**0.334 (0.011)**
	NCLR-Backward	0.449 (0.011)	0.518 (0.010)	0.495 (0.031)	0.480 (0.022)
	NCLR-Forward	0.473 (0.023)	0.501 (0.011)	0.527 (0.011)	0.527 (0.023)
	NCLR-Plug-in	0.486 (0.016)	0.504 (0.011)	0.489 (0.021)	0.507 (0.013)
pageblocks	FW	0.352 (0.023)	0.276 (0.005)	0.499 (0.021)	0.547 (0.030)
	NCFW	**0.267 (0.016)**	**0.201 (0.004)**	**0.426 (0.032)**	**0.466 (0.036)**
	NCLR-Backward	0.684 (0.022)	0.683 (0.036)	0.776 (0.033)	0.715 (0.031)
	NCLR-Forward	0.850 (0.024)	0.800 (0.039)	0.812 (0.042)	0.861 (0.019)
	NCLR-Plug-in	0.695 (0.039)	0.641 (0.045)	0.681 (0.026)	0.677 (0.055)
satimage	FW	**0.197 (0.005)**	**0.228 (0.006)**	**0.246 (0.006)**	0.323 (0.006)
	NCFW	0.199 (0.006)	0.232 (0.005)	0.252 (0.007)	**0.312 (0.005)**
	NCLR-Backward	0.390 (0.004)	0.403 (0.007)	0.412 (0.005)	0.443 (0.002)
	NCLR-Forward	0.386 (0.005)	0.380 (0.003)	0.393 (0.004)	0.425 (0.001)
	NCLR-Plug-in	0.425 (0.004)	0.435 (0.003)	0.465 (0.003)	0.528 (0.006)
covtype	FW	0.567 (0.001)	0.570 (0.001)	0.639 (0.001)	0.653 (0.001)
	NCFW	**0.546 (0.001)**	**0.544 (0.001)**	**0.624 (0.000)**	**0.623 (0.001)**
	NCLR-Backward	0.736 (0.001)	0.744 (0.001)	0.752 (0.001)	0.768 (0.001)
	NCLR-Forward	0.729 (0.001)	0.734 (0.001)	0.732 (0.001)	0.732 (0.001)
	NCLR-Plug-in	0.799 (0.000)	0.802 (0.000)	0.813 (0.000)	0.813 (0.000)
abalone	FW	0.754 (0.005)	0.762 (0.003)	**0.763 (0.005)**	**0.767 (0.003)**
	NCFW	**0.753 (0.005)**	**0.760 (0.003)**	0.770 (0.006)	0.775 (0.004)
	NCLR-Backward	0.910 (0.004)	0.919 (0.004)	0.917 (0.006)	0.910 (0.010)
	NCLR-Forward	0.892 (0.008)	0.921 (0.004)	0.902 (0.004)	0.910 (0.006)
	NCLR-Plug-in	0.907 (0.006)	0.915 (0.007)	0.911 (0.005)	0.908 (0.006)

CHAPTER 8

SUMMARY

This thesis has explored complex classification problems in statistical machine learning, focusing on three aspects: complex label space, complex learning setting, and complex performance measure. The thesis aimed to develop principled, mathematically sound algorithms with theoretical guarantees for handling these complexities, ultimately improving the learning algorithm's performance with increasing data volumes. In particular, we have developed new principled algorithms for all the above settings, together with mathematical performance guarantees as well as empirical demonstration of their effectiveness. Our study could significantly improve the reliability, accuracy, and adaptability of machine learning algorithms, increasing their efficacy and applicability in real-world scenarios. This research also holds the potential to inspire novel methodologies in machine learning, fostering further exploration and development in this rapidly evolving field.

APPENDIX A

SUPPLEMENTAL MATERIAL FOR CHAPTER 2

A.1. Implementation of 'decode'

To solve the combinatorial optimization problem involved in the mapping decode : $\mathbb{R}^{s^2+1} \to \{0,1\}^s$ as defined in Eq. (2.7) efficiently, we make use of an $O(s^3)$-time procedure due to Dembczynski et al. (2011). Specifically, Dembczynski et al. (2011) gave a procedure that, given a certain set of s^2+1 statistics of the true conditional distribution $p(\mathbf{y}|x)$ at a point $x \in \mathcal{X}$, computes in $O(s^3)$ time a Bayes optimal multi-label prediction $h^*(x) \in \{0,1\}^s$ at that point with respect to the F_1-measure by solving a similar combinatorial optimization problem (the approach generalizes easily to the F_β-measure for general β). Our algorithm (Algorithm 2.1) can be viewed as effectively estimating the same s^2+1 statistics from the training sample S; in particular, once a scoring function $\mathbf{f}_S : \mathcal{X} \to \mathbb{R}^{s^2+1}$ is learned by minimizing our surrogate loss ψ, the estimated statistics at a point $x \in \mathcal{X}$ are given by $\gamma^{-1}(\mathbf{f}_S(x))$ (where γ^{-1} is the inverse of the link function $\gamma : [0,1] \to \mathbb{R}$ associated with the strictly proper composite binary loss ϕ used in our surrogate, and is applied element-wise to $\mathbf{f}_S(x)$). Our 'decode' mapping effectively corresponds to estimating a Bayes optimal prediction at x using these estimated statistics; we can therefore apply the procedure of Dembczynski et al. (2011) to these estimated statistics.

The implementation below is described for a general input vector $\mathbf{u} \in \mathbb{R}^{s^2+1}$ (see Eq. (2.7)); in our F_β learning algorithm, to make a prediction at $x \in \mathcal{X}$, it would be applied to $\mathbf{u} = \mathbf{f}_S(x)$. The overall idea is that the combinatorial search over $\widehat{\mathbf{y}} \in \{0,1\}^s$ is stratified over the $s+1$ sets $\widehat{\mathcal{Y}}_l = \{\widehat{\mathbf{y}} \in \{0,1\}^s : \|\widehat{\mathbf{y}}\|_1 = l\}$, $l \in \{0,1,\ldots,s\}$; to find an optimal element $\widehat{\mathbf{y}}^{l,*}$ within each set $\widehat{\mathcal{Y}}_l$, one need only solve a problem of the form $\widehat{\mathbf{y}}^{l,*} \in \operatorname{argmin}_{\widehat{\mathbf{y}} \in \widehat{\mathcal{Y}}_l} \sum_{j=1}^{s} \widehat{y}_j T_{jl}$ for certain numbers T_{jl}, which can be done simply by finding the smallest l numbers among $\{T_{jl} : j \in [s]\}$ and setting the corresponding l entries of $\widehat{\mathbf{y}}^{l,*}$ to 1 (and remaining entries to 0). Solving these $s+1$ subproblems and picking the best solution among them takes a total of $O(s^2 \ln(s))$ time; computing the s^2 numbers

Algorithm A.1 Decode

```
1:  Input: Vector $\mathbf{u} = (u_0, (u_{jk})_{j,k=1}^s)^\top \in \mathbb{R}^{s^2+1}$
2:  Parameters: Link function $\gamma : [0,1] \to \mathbb{R}$
3:  Define matrices $\mathbf{Q} \in [0,1]^{s\times s}$ and $\mathbf{V} \in \mathbb{R}^{s\times s}$ as follows:
        $$Q_{jk} = \gamma^{-1}(u_{jk})$$
        $$V_{kl} = \frac{-(1+\beta)^2}{\beta^2 k + l}$$
4:  Compute $\mathbf{T} = \mathbf{QV}$    // matrix multiplication, $O(s^3)$ time
5:  For $l = 1 \dots s$:                 // for loop takes total $O(s^2 \ln(s))$ time
6:      Find the $l$ smallest numbers among $\{T_{jl} : j \in [s]\}$; call the corresponding indices $j_1^l, \dots, j_l^l$
7:      Define $\widehat{\mathbf{y}}^{l,*} \in \{0,1\}^s$ as follows:
            $$\widehat{y}_j^{l,*} = \begin{cases} 1 & \text{if } j \in \{j_1^l, \dots, j_l^l\} \\ 0 & \text{otherwise.} \end{cases}$$    // this solves $\widehat{\mathbf{y}}^{l,*} \in \operatorname{argmin}_{\widehat{\mathbf{y}} \in \widehat{\mathcal{Y}}_l} \sum_{j=1}^s \widehat{y}_j T_{jl}$
8:      Set $z_l^* = \sum_{j=1}^s \widehat{y}_j^{l,*} T_{jl}$
9:  End for
10: Pick $\widehat{\mathbf{y}}^* \in \{0,1\}^s$ as follows:
        $$\widehat{\mathbf{y}}^* \in \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \{\mathbf{0}, \widehat{\mathbf{y}}^{1,*}, \dots, \widehat{\mathbf{y}}^{s,*}\}} -\mathbf{1}(\widehat{\mathbf{y}} = \mathbf{0}) \cdot \gamma^{-1}(u_0) + \mathbf{1}(\widehat{\mathbf{y}} \neq \mathbf{0}) \cdot z_{\|\widehat{\mathbf{y}}\|_1}^*$$
11: Output: $\widehat{\mathbf{y}}^* \in \{0,1\}^s$
```

T_{jl} involves a matrix multiplication that takes a total of $O(s^3)$ time.[30]

[30]One could in principle use faster matrix multiplication methods that take $o(s^3)$ time, but in practice, this would be helpful for only extremely large values of s.

APPENDIX B

SUPPLEMENTAL MATERIAL FOR CHAPTER 6

B.1. Supplement to Section 6.5

B.1.1. Detailed pseudocode for NCPLUG algorithm

See Algorithm B.1.

Algorithm B.1 Noise-Corrected Plug-in (NCPLUG) for Hamming loss under IFN

```
1: Inputs:
   (1) Noisy training sample, $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
   (2) Noise rates $c^{(j)}_{0,1}, c^{(j)}_{1,0} \quad \forall j \in [s]$
2: Parameters:
   (1) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^s$
3: Compute $\widehat{\mathbf{f}} \in \text{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$, where $\psi$ is defined as
       $$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^s \phi_{\log}(y_j, u_j)$$
4: Output:
   Multi-label classifier
       $$\widehat{h}_j(x) = \mathbf{1}\Big(\frac{\gamma^{-1}_{\log}(\widehat{f}_j(x)) - c^{(j)}_{0,1}}{1 - c^{(j)}_{0,1} - c^{(j)}_{1,0}} \geq \frac{1}{2}\Big) \quad \forall j \in [s]$$
```

B.1.2. Detailed pseudocode for NCEFP algorithm

See Algorithm B.2.

B.1.3. Detailed pseudocode for NCOC algorithm

See Algorithm B.3.

B.1.4. NCOC-Ham-IFN algorithm

As noted in Section 6.5.3, under the IFN model, noise matrices $\mathbf{C}$ are (to our knowledge) computationally expensive to invert, which makes it difficult to run the NCOC algorithm for such noise matrices in practice. For the special case of Hamming loss under the IFN model, we present an

Algorithm B.2 Noise-Corrected Exact F-measure Plug-in (NCEFP) for F_1-measure

```
1: Inputs:
   (1) Noisy training sample, $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
   (2) Noise matrix $\mathbf{C} \in [0,1]^{|\mathcal{Y}| \times |\mathcal{Y}|}$
2: Parameters:
   (1) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^{s^2+1}$
3: Let $\mathbf{A} \in [0,1]^{(s^2+1) \times |\mathcal{Y}|}$ be
   $$a_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)\,; \qquad a_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j \quad \forall j, k \in [s]$$
4: Let $\widetilde{\mathbf{A}} = \mathbf{A}(\mathbf{C}^\top)^{-1}$, and define $\widetilde{a}_{\min} = \min(\min_{\mathbf{y}} \widetilde{a}_{0,\mathbf{y}}, \min_{\mathbf{y},jk} \widetilde{a}_{jk,\mathbf{y}})$ and $\widetilde{a}_{\max} = \max(\max_{\mathbf{y}} \widetilde{a}_{0,\mathbf{y}}, \max_{\mathbf{y},jk} \widetilde{a}_{jk,\mathbf{y}})$
5: Construct $\widetilde{\mathbf{A}}' \in [0,1]^{(s^2+1) \times |\mathcal{Y}|}$ by shifting and scaling $\widetilde{\mathbf{A}}$ as
   $$\widetilde{a}'_{0,\mathbf{y}} = \frac{\widetilde{a}_{0,\mathbf{y}} - \widetilde{a}_{\min}}{\widetilde{a}_{\max} - \widetilde{a}_{\min}} \in [0,1]\,; \qquad \widetilde{a}'_{jk,\mathbf{y}} = \frac{\widetilde{a}_{jk,\mathbf{y}} - \widetilde{a}_{\min}}{s \cdot (\widetilde{a}_{\max} - \widetilde{a}_{\min})} \in [0,1] \quad \forall j, k \in [s]$$
6: Compute $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$, where $\psi$ is as defined as
   $$\psi(\mathbf{y}, \mathbf{u}) = \widetilde{a}'_{0,\mathbf{y}} \cdot \phi_{\log}(1, u_0) + (1 - \widetilde{a}'_{0,\mathbf{y}}) \cdot \phi_{\log}(0, u_0) + \sum_{j=1}^s \Big[ \sum_{k=1}^s \widetilde{a}'_{jk,\mathbf{y}} \phi_{\mathrm{mlog}}(k, (u_{j1}, ..., u_{js})) + (1 - \sum_{k=1}^s \widetilde{a}'_{jk,\mathbf{y}}) \phi_{\mathrm{mlog}}(s+1, (u_{j1}, ..., u_{js})) \Big]$$
7: Output:
   Multi-label classifier
   $$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \mathcal{Y}} \Big\{ 1 - [(\widetilde{a}_{\max} - \widetilde{a}_{\min}) \cdot \widehat{q}'_0(x) + \widetilde{a}_{\min}] \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{j=1}^s \sum_{k=1}^s [s \cdot (\widetilde{a}_{\max} - \widetilde{a}_{\min}) \cdot \widehat{q}'_{jk}(x) + \widetilde{a}_{\min}] \cdot \frac{2 \cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1} \Big\}$$
   where $\widehat{q}'_0(x) = \gamma_{\log}^{-1}(\widehat{f}_0(x))$ and $\widehat{q}'_{jk}(x) = \big(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(\widehat{f}_{j1}(x), ..., \widehat{f}_{js}(x))\big)_k$
```

Algorithm B.3 Noise-Corrected Output Coding (NCOC)

```
1: Inputs:
   (1) Noisy training sample, $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
   (2) Target loss $\mathbf{L} \in \mathbb{R}_+^{|\mathcal{Y}|\times|\mathcal{Y}|}$ factorized as $\mathbf{L} = \mathbf{A}^\top\mathbf{B} + \mathbf{1}\mathbf{t}^\top$ for some $\mathbf{A} \in [0,1]^{r\times|\mathcal{Y}|}, \mathbf{B} \in \mathbb{R}^{r\times|\mathcal{Y}|}, \mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$
   (3) Noise matrix $\mathbf{C} \in [0,1]^{|\mathcal{Y}|\times|\mathcal{Y}|}$
2: Parameters:
   (1) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^r$
3: Let $\widetilde{\mathbf{A}} = \mathbf{A}(\mathbf{C}^\top)^{-1}$, and define $\widetilde{a}_{\min} = \min_{\mathbf{y},j} \widetilde{a}_{j,\mathbf{y}}$ and $\widetilde{a}_{\max} = \max_{\mathbf{y},j} \widetilde{a}_{j,\mathbf{y}}$
4: Construct $\widetilde{\mathbf{A}}' \in [0,1]^{r\times|\mathcal{Y}|}$ by shifting and scaling $\widetilde{\mathbf{A}}$ as
       $$\widetilde{a}'_{j,\mathbf{y}} = \frac{\widetilde{a}_{j,\mathbf{y}} - \widetilde{a}_{\min}}{\widetilde{a}_{\max} - \widetilde{a}_{\min}} \in [0,1] \quad \forall j \in [r]$$
5: Compute $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f}\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$, where $\psi$ is defined as
       $$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^r \left(\widetilde{a}'_{j,\mathbf{y}}\phi_{\log}(1, u_j) + (1 - \widetilde{a}'_{j,\mathbf{y}})\phi_{\log}(0, u_j)\right)$$
6: Output:
   Multi-label classifier
       $$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \left\{t_{\widehat{\mathbf{y}}} + \sum_{j=1}^r [(\widetilde{a}_{\max} - \widetilde{a}_{\min}) \cdot \gamma_{\log}^{-1}(\widehat{f}_j(x)) + \widetilde{a}_{\min}] \cdot b_{j,\widehat{\mathbf{y}}}\right\}$$
```

Algorithm B.4 NCOC-Ham-IFN for Hamming loss under IFN

```
1: Inputs:
   (1) Noisy training sample, $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
   (2) Noise rates $c^{(j)}_{0,1}, c^{(j)}_{1,0} \quad \forall j \in [s]$
2: Parameters:
   (1) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^s$
3: Let $\mathbf{c}_{0,1} = [c^{(1)}_{0,1}, ..., c^{(s)}_{0,1}]^\top \in [0,1)^s$, and define a diagonal matrix $\boldsymbol{\Lambda}$ of size $s \times s$ such that its $i$-th
   diagonal element is $\frac{1}{1 - c^{(i)}_{0,1} - c^{(i)}_{1,0}}$
4: Let $\widetilde{\widetilde{\mathbf{A}}} = \boldsymbol{\Lambda}(\mathbf{A} - \mathbf{c}_{0,1}\mathbf{1}^\top)$, and define $\widetilde{\widetilde{a}}_{\min} = \min_{\mathbf{y},j} \widetilde{\widetilde{a}}_{j,\mathbf{y}}$ and $\widetilde{\widetilde{a}}_{\max} = \max_{\mathbf{y},j} \widetilde{\widetilde{a}}_{j,\mathbf{y}}$
5: Construct $\widetilde{\mathbf{A}}'' \in [0,1]^{s \times |\mathcal{Y}|}$ by shifting and scaling $\widetilde{\widetilde{\mathbf{A}}}$ as
   $$\widetilde{a}''_{j,\mathbf{y}} = \frac{\widetilde{\widetilde{a}}_{j,\mathbf{y}} - \widetilde{\widetilde{a}}_{\min}}{\widetilde{\widetilde{a}}_{\max} - \widetilde{\widetilde{a}}_{\min}} \in [0,1] \quad \forall j \in [s] \,.$$
6: Compute $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$, where $\psi$ is defined as
   $$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^s \Big( \widetilde{a}''_{j,\mathbf{y}} \phi_{\log}(1, u_j) + (1 - \widetilde{a}''_{j,\mathbf{y}}) \phi_{\log}(0, u_j) \Big)$$
7: Output:
   Multi-label classifier
   $$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \mathcal{Y}} \Big\{ \frac{1}{s} \|\widehat{\mathbf{y}}\|_1 + \sum_{j=1}^s [(\widetilde{\widetilde{a}}_{\max} - \widetilde{\widetilde{a}}_{\min}) \cdot \gamma^{-1}_{\log}(\widehat{f}_j(x)) + \widetilde{\widetilde{a}}_{\min}] \cdot \frac{1 - 2\widehat{y}_j}{s} \Big\}$$
```

alternative faster noise-corrected output coding algorithm – that we call NCOC-Ham-IFN – that decomposes the problem of estimating statistics $\mathbf{q}(x)$ into a different set of s binary CPE problems obtained using a different coding matrix $\widetilde{\mathbf{A}}''$ that does not require inverting $\mathbf{C}^\top$.

Recall from Example 6.1 that $\mathbf{L}^{\text{Ham}} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ where $\mathbf{A} \in [0,1]^{s \times |\mathcal{Y}|}$ with $a_{j,\mathbf{y}} = y_j$, $\mathbf{B} \in \mathbb{R}^{s \times |\mathcal{Y}|}$ with $b_{j,\widehat{\mathbf{y}}} = \frac{1 - 2\widehat{y}_j}{s}$, and $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ with $t_{\widehat{\mathbf{y}}} = \sum_{j=1}^s \frac{\widehat{y}_j}{s} = \frac{1}{s}\|\widehat{\mathbf{y}}\|_1$. Recall also that the s-dimensional vector statistic $\mathbf{q}(x)$ defined as

$$\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x)$$

is Bayes-sufficient for $\mathbf{L}^{\mathrm{Ham}}$. We will express the desired statistics $\mathbf{q}(x)$ in terms of $\widetilde{\boldsymbol{\eta}}(x)$ using a matrix $\widetilde{\widetilde{\mathbf{A}}}$ that does not require inverting $\mathbf{C}^\top$, and will then use a shifted and scaled version of this matrix to obtain $\widetilde{\mathbf{A}}''$.

We have that

$$q_j(x) = \mathbf{P}(Y_j = 1|x) \quad \forall j \in [s]\,.$$

Next, as in Section 6.5.1, define $\mathbf{q}'(x) = \mathbf{A}\widetilde{\boldsymbol{\eta}}(x)$ so that

$$q'_j(x) = \mathbf{P}(\widetilde{Y}_j = 1|x) \quad \forall j \in [s]\,.$$

Under the IFN model where $c_{0,1}^{(j)} + c_{1,0}^{(j)} < 1\ \forall j \in [s]$, $q_j(x)$ and $q'_j(x)$ are related by

$$q_j(x) = \frac{q'_j(x) - c_{0,1}^{(j)}}{1 - c_{0,1}^{(j)} - c_{1,0}^{(j)}} \quad \forall j \in [s]\,.$$

Let $\mathbf{c}_{0,1} = [c_{0,1}^{(1)}, ..., c_{0,1}^{(s)}]^\top \in [0,1)^s$. Define a diagonal matrix $\boldsymbol{\Lambda}$ of size $s \times s$ such that its i-th diagonal element is $\frac{1}{1-c_{0,1}^{(i)}-c_{1,0}^{(i)}}$. Then we can write

$$\mathbf{q}(x) = \boldsymbol{\Lambda}(\mathbf{q}'(x) - \mathbf{c}_{0,1})\,.$$

Now, define

$$\widetilde{\widetilde{\mathbf{A}}} = \boldsymbol{\Lambda}(\mathbf{A} - \mathbf{c}_{0,1}\mathbf{1}^\top)\,,$$

where $\mathbf{1}$ here is a $|\mathcal{Y}| \times 1$ vector. Then $\widetilde{\widetilde{\mathbf{A}}}$ also has size $s \times |\mathcal{Y}|$, and we have

$$\begin{aligned}
\widetilde{\widetilde{\mathbf{A}}}\widetilde{\boldsymbol{\eta}}(x) &= \boldsymbol{\Lambda}(\mathbf{A} - \mathbf{c}_{0,1}\mathbf{1}^\top)\widetilde{\boldsymbol{\eta}}(x) \\
&= \boldsymbol{\Lambda}(\mathbf{A}\widetilde{\boldsymbol{\eta}}(x) - \mathbf{c}_{0,1}\mathbf{1}^\top\widetilde{\boldsymbol{\eta}}(x)) \\
&= \boldsymbol{\Lambda}(\mathbf{q}'(x) - \mathbf{c}_{0,1}) \\
&= \mathbf{q}(x)\,.
\end{aligned}$$

Thus we have that statistics $\mathbf{q}(x)$ can be expressed in terms of $\widetilde{\boldsymbol{\eta}}(x)$ as $\mathbf{q}(x) = \widetilde{\widetilde{\mathbf{A}}}\widetilde{\boldsymbol{\eta}}(x)$, where $\widetilde{\widetilde{\mathbf{A}}}$ does not require inverting $\mathbf{C}^\top$.

Again, in order to set up suitably weighted binary CPE problems that can allow us to estimate these statistics from the noisy training sample, we will use a shifted and scaled matrix $\widetilde{\mathbf{A}}''$ to estimate related statistics $\mathbf{q}''(x)$, and then factor back in the scaling and shifting when making a final prediction. Towards this, define $\widetilde{\widetilde{a}}_{\min} = \min_{\mathbf{y},j} \widetilde{\widetilde{a}}_{j,\mathbf{y}}$ and $\widetilde{\widetilde{a}}_{\max} = \max_{\mathbf{y},j} \widetilde{\widetilde{a}}_{j,\mathbf{y}}$, and define the entries of $\widetilde{\mathbf{A}}'' \in [0,1]^{s\times|\mathcal{Y}|}$ as

$$\widetilde{a}''_{j,\mathbf{y}} = \frac{\widetilde{\widetilde{a}}_{j,\mathbf{y}} - \widetilde{\widetilde{a}}_{\min}}{\widetilde{\widetilde{a}}_{\max} - \widetilde{\widetilde{a}}_{\min}} \in [0,1] \quad \forall j \in [s]\,.$$

We note again that matrix $\widetilde{\mathbf{A}}''$ here is different from that used for the NCOC algorithm in Section 6.5.3; in particular, now we do not need to compute $(\mathbf{C}^\top)^{-1}$. Next, define $\mathbf{q}''(x) = \widetilde{\mathbf{A}}''\widetilde{\boldsymbol{\eta}}(x)$. Then, for each $j \in [s]$, we set up a weighted binary CPE problem with weights $(\widetilde{a}''_{j,\mathbf{y}}, (1-\widetilde{a}''_{j,\mathbf{y}}))$ to estimate $q''_j(x)$. Finally, given estimated statistics $\widehat{\mathbf{q}}''(x)$ estimated in this way from the noisy training sample, our NCOC-Ham-IFN algorithm outputs the multi-label classifier

$$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \Big\{ \frac{1}{s}\|\widehat{\mathbf{y}}\|_1 + \sum_{j=1}^{s} [(\widetilde{\widetilde{a}}_{\max} - \widetilde{\widetilde{a}}_{\min}) \cdot \widehat{q}''_j(x) + \widetilde{\widetilde{a}}_{\min}] \cdot \frac{1 - 2\widehat{y}_j}{s} \Big\}\,.$$

Estimating $\mathbf{q}''(x)$. Our implementation of NCOC-Ham-IFN uses weighted binary logistic loss minimizers for the weighted CPE learners. In particular, we first learn a vector of s real-valued functions $\widehat{\mathbf{f}} : \mathcal{X}\to\mathbb{R}^s$ by minimizing the s-dimensional convex surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^s \to \mathbb{R}_+$ defined as

$$\psi(\mathbf{y},\mathbf{u}) = \sum_{j=1}^{s} \Big(\widetilde{a}''_{j,\mathbf{y}} \phi_{\log}(1,u_j) + (1-\widetilde{a}''_{j,\mathbf{y}})\phi_{\log}(0,u_j) \Big)$$

over the noisy training sample $\widetilde{S}$. Specifically, $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f}\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^{m} \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$ for a suitable class of real-valued vector functions $\mathcal{F} \subseteq \{\mathbf{f} : \mathcal{X}\to\mathbb{R}^s\}$. The estimated statistics are then given by $\widehat{q}''_j(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$.

Detailed pseudocode is in Algorithm B.4.

B.2. Supplement to Section 6.6

B.2.1. Proof of Theorem 6.1

In this section, we prove a more general version of Theorem 6.1 that can handle the case when estimated noise rates $\widehat{c}_{0,1}^{(j)}$ and $\widehat{c}_{1,0}^{(j)}$ are used in place of the true noise rates $c_{0,1}^{(j)}$ and $c_{1,0}^{(j)}$. Therefore, setting $\widehat{c}_{0,1}^{(j)} = c_{0,1}^{(j)}$ and $\widehat{c}_{1,0}^{(j)} = c_{1,0}^{(j)}$ for $j \in [s]$ in the theorem below proves Theorem 6.1.

Theorem B.1 (More general version of Theorem 6.1). *Consider Hamming loss $\mathbf{L}^{\text{Ham}}$ (Eq. (6.1)) under IFN model. Assume $c_{0,1}^{(j)} + c_{1,0}^{(j)} < 1$ for all $j \in [s]$. Let D be any distribution on $\mathcal{X} \times \mathcal{Y}$ with corresponding noisy distribution $\widetilde{D}$. Let $\widehat{c}_{0,1}^{(j)} \geq 0$ and $\widehat{c}_{1,0}^{(j)} \geq 0$ be estimated noise rates such that $\widehat{c}_{0,1}^{(j)} + \widehat{c}_{1,0}^{(j)} < 1$ for $j \in [s]$. Suppose NCPLUG (Section 6.5.1) is run with noisy training sample $\widetilde{S}$ (in which examples are sampled i.i.d. from $\widetilde{D}$) and estimated noise rates $\widehat{c}_{0,1}^{(j)}$ and $\widehat{c}_{1,0}^{(j)}$ in place of $c_{0,1}^{(j)}$ and $c_{1,0}^{(j)}$, for all $j \in [s]$. Let $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ be as defined in Section 6.5.1. Then we have*

$$\text{regret}_D^{\mathbf{L}^{\text{Ham}}}[\widehat{\mathbf{h}}] \leq \frac{1}{\sqrt{s}} \max_i \frac{1}{1 - \widehat{c}_{0,1}^{(i)} - \widehat{c}_{1,0}^{(i)}} \sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]} + \\ \frac{2}{s} \sum_{j=1}^{s} \frac{|\widehat{c}_{0,1}^{(j)} + \widehat{c}_{1,0}^{(j)} - c_{0,1}^{(j)} - c_{1,0}^{(j)}| + |c_{0,1}^{(j)}(1 - \widehat{c}_{1,0}^{(j)}) - \widehat{c}_{0,1}^{(j)}(1 - c_{1,0}^{(j)})|}{(1 - c_{0,1}^{(j)} - c_{1,0}^{(j)})(1 - \widehat{c}_{0,1}^{(j)} - \widehat{c}_{1,0}^{(j)})}.$$

Moreover, if $\widehat{c}_{0,1}^{(j)} = c_{0,1}^{(j)}$ and $\widehat{c}_{1,0}^{(j)} = c_{1,0}^{(j)}$ for all $j \in [s]$, the above bound can be simplified to

$$\text{regret}_D^{\mathbf{L}^{\text{Ham}}}[\widehat{\mathbf{h}}] \leq \frac{1}{\sqrt{s}} \max_i \frac{1}{1 - c_{0,1}^{(i)} - c_{1,0}^{(i)}} \sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]}.$$

We will need the following lemma from Agarwal (2014) in the proof.

Lemma B.2 (Property of the binary logistic loss (Agarwal, 2014)). *Recall that the binary logistic loss $\phi_{\log} : \{0,1\} \times \mathbb{R} \to \mathbb{R}_+$ is defined as*

$$\phi_{\log}(y, u) = \ln(1 + e^{-(2y-1)u}),$$

and the invertible link function $\gamma_{\log} : [0,1] \to \mathbb{R}$ *together with its inverse* $\gamma_{\log}^{-1} : \mathbb{R} \to [0,1]$ *is given by*

$$\gamma_{\log}(p) = \ln(\frac{p}{1-p});$$
$$\gamma_{\log}^{-1}(u) = \frac{1}{1+\exp(-u)}.$$

Then for all $q \in [0,1]$ *and* $u \in \mathbb{R}$*:*

$$\mathbf{E}_{Y \sim \textit{Bernoulli}(q)}\Big[\phi_{\log}(Y,u) - \phi_{\log}(Y, \gamma_{\log}(q))\Big] \;\geq\; 2\Big(\gamma_{\log}^{-1}(u) - q\Big)^2,$$

where $Y \sim$ *Bernoulli*(q) *denotes a Bernoulli random variable that takes value* 1 *with probability* q *and value* 0 *with probability* $1-q$.

Proof of Theorem B.1.

Proof. Recall that the multi-label surrogate $\psi : \mathcal{Y} \times \mathbb{R}^s \to \mathbb{R}_+$ here is given by

$$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^{s} \phi_{\log}(y_j, u_j). \tag{B.1}$$

Also, given $\widehat{q}'_j(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$ estimated from $\widetilde{S}$, the multi-label classifier output by our NCPLUG algorithm is given by

$$\widehat{h}_j(x) = \mathbf{1}\Big(\frac{\widehat{q}'_j(x) - \widehat{c}^{(j)}_{0,1}}{1 - \widehat{c}^{(j)}_{0,1} - \widehat{c}^{(j)}_{1,0}} \geq \tfrac{1}{2}\Big) \quad \forall j \in [s]. \tag{B.2}$$

To simplify notations, in what follows, we let $\mathbf{L} = \mathbf{L}^{\text{Ham}}$, $\phi = \phi_{\log}$, $\gamma = \gamma_{\log}$, $\mathbf{f} = \widehat{\mathbf{f}}$, and $\mathbf{h} = \widehat{\mathbf{h}}$. We also let $q_j(x) = \mathbf{P}(Y_j = 1|x)$ and $\widetilde{q}_j(x) = \mathbf{P}(\widetilde{Y}_j = 1|x)$ for $j \in [s]$.

$$
\begin{aligned}
&\mathrm{regret}^{\mathbf{L}}_{D}[\mathbf{h}] \\
&= \frac{1}{s}\sum_{j=1}^{s}\mathbf{E}_x\Big[(1-2q_j(x))\cdot[\mathbf{1}(\frac{\gamma^{-1}(f_j(x))-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}\geq\frac{1}{2})-\mathbf{1}(q_j(x)\geq\frac{1}{2})]\Big] \\
&\leq \frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\Big[|\frac{\gamma^{-1}(f_j(x))-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-q_j(x)|\Big] \\
&= \frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\Big[|\frac{\gamma^{-1}(f_j(x))-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-\frac{\widetilde{q}_j(x)-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}+\frac{\widetilde{q}_j(x)-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-q_j(x)|\Big] \\
&\leq \frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\Big[|\frac{\gamma^{-1}(f_j(x))-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-\frac{\widetilde{q}_j(x)-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}|+|\frac{\widetilde{q}_j(x)-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-q_j(x)|\Big] \\
&= \frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\Big[|\frac{\gamma^{-1}(f_j(x))-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-\frac{\widetilde{q}_j(x)-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}|\Big]+\frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\Big[|\frac{\widetilde{q}_j(x)-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-q_j(x)|\Big]. \qquad \text{(B.3)}
\end{aligned}
$$

We bound the first sum as follows:

$$
\begin{aligned}
&\frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\Big[|\frac{\gamma^{-1}(f_j(x))-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-\frac{\widetilde{q}_j(x)-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}|\Big] \\
&= \frac{2}{s}\sum_{j=1}^{s}\frac{1}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}\mathbf{E}_x\Big[|\gamma^{-1}(f_j(x))-\widetilde{q}_j(x)|\Big] \\
&\leq \frac{2}{s}\max_i\frac{1}{1-\widehat{c}^{(i)}_{0,1}-\widehat{c}^{(i)}_{1,0}}\mathbf{E}_x\Big[\sum_{j=1}^{s}|\gamma^{-1}(f_j(x))-\widetilde{q}_j(x)|\Big] \\
&\leq \frac{2}{s}\max_i\frac{1}{1-\widehat{c}^{(i)}_{0,1}-\widehat{c}^{(i)}_{1,0}}\mathbf{E}_x\Big[\sqrt{s}\sqrt{\sum_{j=1}^{s}\Big(\gamma^{-1}(f_j(x))-\widetilde{q}_j(x)\Big)^2}\Big] \\
&= \frac{2}{\sqrt{s}}\max_i\frac{1}{1-\widehat{c}^{(i)}_{0,1}-\widehat{c}^{(i)}_{1,0}}\mathbf{E}_x\Big[\sqrt{\sum_{j=1}^{s}\Big(\gamma^{-1}(f_j(x))-\widetilde{q}_j(x)\Big)^2}\Big]. \qquad \text{(B.4)}
\end{aligned}
$$

By Lemma B.2, we have

$$\begin{aligned}
&\mathbf{E}_x\Big[\sum_{j=1}^{s}\Big(\gamma^{-1}(f_j(x)) - \widetilde{q}_j(x)\Big)^2\Big] \\
&\leq \frac{1}{2}\mathbf{E}_x\Big[\sum_{j=1}^{s}\mathbf{E}_{y\sim\text{Bernoulli}(\widetilde{q}_j(x))}\Big[\phi(y, f_j(x)) - \phi(y, \gamma(\widetilde{q}_j(x)))\Big]\Big] \\
&= \frac{1}{2}\mathbf{E}_x\Big[\mathbf{E}_{\mathbf{y}|x\sim\widetilde{\boldsymbol{\eta}}(x)}\Big[\psi(\mathbf{y}, \mathbf{f}(x)) - \inf_{\mathbf{u}\in\mathbb{R}^r}\psi(\mathbf{y}, \mathbf{u})\Big]\Big] \\
&= \frac{1}{2}\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]\,. \qquad \text{(B.5)}
\end{aligned}$$

Next, we bound the second sum in Eq. (B.3) as follows:

$$\begin{aligned}
&\frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\Big[\Big|\frac{\widetilde{q}_j(x) - \widehat{c}_{0,1}^{(j)}}{1 - \widehat{c}_{0,1}^{(j)} - \widehat{c}_{1,0}^{(j)}} - q_j(x)\Big|\Big] \\
&= \frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\Big[\Big|\frac{\widetilde{q}_j(x) - \widehat{c}_{0,1}^{(j)}}{1 - \widehat{c}_{0,1}^{(j)} - \widehat{c}_{1,0}^{(j)}} - \frac{\widetilde{q}_j(x) - c_{0,1}^{(j)}}{1 - c_{0,1}^{(j)} - c_{1,0}^{(j)}}\Big|\Big] \\
&= \frac{2}{s}\sum_{j=1}^{s}\frac{\mathbf{E}_x\Big[|\widetilde{q}_j(x)(\widehat{c}_{0,1}^{(j)} + \widehat{c}_{1,0}^{(j)} - c_{0,1}^{(j)} - c_{1,0}^{(j)}) + \widehat{c}_{0,1}^{(j)}(c_{1,0}^{(j)} - 1) + c_{0,1}^{(j)}(1 - \widehat{c}_{1,0}^{(j)})|\Big]}{(1 - c_{0,1}^{(j)} - c_{1,0}^{(j)})(1 - \widehat{c}_{0,1}^{(j)} - \widehat{c}_{1,0}^{(j)})} \\
&\leq \frac{2}{s}\sum_{j=1}^{s}\frac{|\widehat{c}_{0,1}^{(j)} + \widehat{c}_{1,0}^{(j)} - c_{0,1}^{(j)} - c_{1,0}^{(j)}| + |c_{0,1}^{(j)}(1 - \widehat{c}_{1,0}^{(j)}) - \widehat{c}_{0,1}^{(j)}(1 - c_{1,0}^{(j)})|}{(1 - c_{0,1}^{(j)} - c_{1,0}^{(j)})(1 - \widehat{c}_{0,1}^{(j)} - \widehat{c}_{1,0}^{(j)})}\,. \qquad \text{(B.6)}
\end{aligned}$$

Combining Eqs. (B.3), (B.4), (B.5) and (B.6) and applying Jensen's inequality (to the convex function $x \mapsto x^2$), we have

$$\begin{aligned}
\text{regret}_D^{\mathbf{L}}[\mathbf{h}] \leq \frac{1}{\sqrt{s}}\max_i \frac{1}{1 - \widehat{c}_{0,1}^{(i)} - \widehat{c}_{1,0}^{(i)}}\sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]} + \\
\frac{2}{s}\sum_{j=1}^{s}\frac{|\widehat{c}_{0,1}^{(j)} + \widehat{c}_{1,0}^{(j)} - c_{0,1}^{(j)} - c_{1,0}^{(j)}| + |c_{0,1}^{(j)}(1 - \widehat{c}_{1,0}^{(j)}) - \widehat{c}_{0,1}^{(j)}(1 - c_{1,0}^{(j)})|}{(1 - c_{0,1}^{(j)} - c_{1,0}^{(j)})(1 - \widehat{c}_{0,1}^{(j)} - \widehat{c}_{1,0}^{(j)})}\,.
\end{aligned}$$

□

B.2.2. Proof of Theorem 6.2

In this section, we prove a more general version of Theorem 6.2 that can handle the case when an estimated noise matrix $\widehat{\mathbf{C}}$ is used in place of the true noise matrix $\mathbf{C}$. Therefore, setting $\widehat{\mathbf{C}} = \mathbf{C}$ in the theorem below proves Theorem 6.2.

Theorem B.3 (**More general version of Theorem 6.2**). *Consider F-measure $\mathbf{L}^{F_1}$ (Eq. (6.2)) under the general CCN model. Assume noise matrix $\mathbf{C}$ is invertible. Let D be any distribution on $\mathcal{X} \times \mathcal{Y}$ with corresponding noisy distribution $\widetilde{D}$. Let $\widehat{\mathbf{C}}$ be an estimated (invertible) noise matrix for the noise matrix $\mathbf{C}$. Suppose NCEFP (Section 6.5.2) is run with noisy training sample $\widetilde{S}$ (in which examples are sampled i.i.d. from $\widetilde{D}$) and estimated noise matrix $\widehat{\mathbf{C}}$ in place of $\mathbf{C}$. Let $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ be as defined in Section 6.5.2, and let $\mathbf{A} \in [0,1]^{(s^2+1)\times|\mathcal{Y}|}$, $\mathbf{B} \in \mathbb{R}^{(s^2+1)\times|\mathcal{Y}|}$, and $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ be as defined in Example 6.2. Then we have*

$$\mathrm{regret}_D^{\mathbf{L}^{F_1}}[\widehat{\mathbf{h}}] \leq 2 \max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \left(\|\mathbf{A}\|_2 \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 + 2\|\mathbf{A}\|_1 \|(\widehat{\mathbf{C}}^\top)^{-1}\|_1 s \sqrt{2\mathrm{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]} \right).$$

Further, if $\widehat{\mathbf{C}} = \mathbf{C}$, the above bound can be simplified to

$$\mathrm{regret}_D^{\mathbf{L}^{F_1}}[\widehat{\mathbf{h}}] \leq 4s \max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \|\mathbf{A}\|_1 \|(\mathbf{C}^\top)^{-1}\|_1 \sqrt{2\mathrm{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]}\,.$$

We will need the following lemma from Zhang et al. (2021) in the proof.

Lemma B.4 (**Property of the multiclass logistic loss (Zhang et al., 2021)**). *Recall that the multiclass logistic loss $\phi_{\mathrm{mlog}} : [n] \times \mathbb{R}^{n-1} \to \mathbb{R}_+$ is defined as*

$$\phi_{\mathrm{mlog}}(y, \mathbf{u}) = \begin{cases} -\ln\left(\frac{\exp(u_y)}{1+\sum_{i=1}^{n-1}\exp(u_i)}\right) & \text{if } y \in [n-1] \\ \ln\left(1 + \sum_{i=1}^{n-1}\exp(u_i)\right) & \text{if } y = n \end{cases},$$

and the invertible link function $\gamma_{\mathrm{mlog}} : \Delta_n \to \mathbb{R}^{n-1}$ together with its inverse $\gamma_{\mathrm{mlog}}^{-1} : \mathbb{R}^{n-1} \to \Delta_n$ is *given by*

$$\boldsymbol{\gamma}_{\text{mlog}}(\mathbf{p}) = \begin{pmatrix} \ln(\frac{p_1}{p_n}) \\ \vdots \\ \ln(\frac{p_{n-1}}{p_n}) \end{pmatrix}; \quad \boldsymbol{\gamma}_{\text{mlog}}^{-1}(\mathbf{u}) = \begin{pmatrix} \frac{\exp(u_1)}{1+\sum_{i=1}^{n-1}\exp(u_i)} \\ \vdots \\ \frac{\exp(u_n)}{1+\sum_{i=1}^{n-1}\exp(u_i)} \\ \frac{1}{1+\sum_{i=1}^{n-1}\exp(u_i)} \end{pmatrix}.$$

Then for all $\mathbf{p} \in \Delta_n$ *and* $\mathbf{u} \in \mathbb{R}^{n-1}$,

$$\mathbf{E}_{Y\sim\mathbf{p}}\Big[\phi_{\text{mlog}}(Y,\mathbf{u}) - \phi_{\text{mlog}}(Y, \boldsymbol{\gamma}_{\text{mlog}}(\mathbf{p}))\Big] \geq \frac{1}{2}\big\|\boldsymbol{\gamma}_{\text{mlog}}^{-1}(\mathbf{u}) - \mathbf{p}\big\|_2^2.$$

Proof of Theorem B.3.

Proof. For clarity, we redefine here all quantities involving $\widehat{\mathbf{C}}$. In particular, define $\widehat{\mathbf{A}} = \mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}$, $\widehat{a}_{\min} = \min(\min_{\mathbf{y}} \widehat{a}_{0,\mathbf{y}}, \min_{\mathbf{y},jk} \widehat{a}_{jk,\mathbf{y}})$ and $\widehat{a}_{\max} = \max(\max_{\mathbf{y}} \widehat{a}_{0,\mathbf{y}}, \max_{\mathbf{y},jk} \widehat{a}_{jk,\mathbf{y}})$, and let

$$\widehat{a}'_{0,\mathbf{y}} = \frac{\widehat{a}_{0,\mathbf{y}} - \widehat{a}_{\min}}{\widehat{a}_{\max} - \widehat{a}_{\min}} \in [0,1],$$

and

$$\widehat{a}'_{jk,\mathbf{y}} = \frac{\widehat{a}_{jk,\mathbf{y}} - \widehat{a}_{\min}}{s \cdot (\widehat{a}_{\max} - \widehat{a}_{\min})} \in [0,1] \quad \forall j,k \in [s].$$

Here, the multi-label surrogate $\psi : \mathcal{Y} \times \mathbb{R}^{s^2+1} \to \mathbb{R}_+$ then becomes

$$\begin{aligned}\psi(\mathbf{y},\mathbf{u}) =\,& \widehat{a}'_{0,\mathbf{y}} \cdot \phi_{\log}(1,u_0) + (1-\widehat{a}'_{0,\mathbf{y}}) \cdot \phi_{\log}(0,u_0) + \\ & \sum_{j=1}^{s}\Big[\sum_{k=1}^{s}\widehat{a}'_{jk,\mathbf{y}}\phi_{\text{mlog}}(k,(u_{j1},...,u_{js})) + (1-\sum_{k=1}^{s}\widehat{a}'_{jk,\mathbf{y}})\phi_{\text{mlog}}(s+1,(u_{j1},...,u_{js}))\Big]. \quad \text{(B.7)}\end{aligned}$$

Also, given $\widehat{q}'_0(x) = \gamma_{\log}^{-1}(\widehat{f}_0(x))$ and $\widehat{q}'_{jk}(x) = \big(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(\widehat{f}_{j1}(x),...,\widehat{f}_{js}(x))\big)_k$ estimated from $\widetilde{S}$, the multi-label classifier output by our NCEFP algorithm is given by

$$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \Big\{ 1 + [(\widehat{a}_{\max} - \widehat{a}_{\min}) \cdot \widehat{q}_0'(x) + \widehat{a}_{\min}] \cdot b_{0,\widehat{\mathbf{y}}} \\ + \sum_{j=1}^{s}\sum_{k=1}^{s} [s \cdot (\widehat{a}_{\max} - \widehat{a}_{\min}) \cdot \widehat{q}_{jk}'(x) + \widehat{a}_{\min}] \cdot b_{jk,\widehat{\mathbf{y}}} \Big\}. \tag{B.8}$$

We use $\langle \cdot, \cdot \rangle$ to denote the standard inner product. Let $\widehat{\alpha} = (\widehat{a}_{\max} - \widehat{a}_{\min})$ and $\widehat{\beta} = \widehat{a}_{\min}$. To simplify notations, in what follows, we let $\mathbf{L} = \mathbf{L}^{F_1}$, $\mathbf{f} = \widehat{\mathbf{f}}$ and $\mathbf{h} = \widehat{\mathbf{h}}$.

We let

$$\begin{aligned} g(\mathbf{f}(x), \widehat{\mathbf{y}}) := {} & (\widehat{\alpha}\gamma_{\log}^{-1}(f_0(x)) + \widehat{\beta}) \cdot (b_{0,\mathbf{h}(x)} - b_{0,\widehat{\mathbf{y}}}) \\ & + \widehat{\alpha} s \sum_{j=1}^{s}\sum_{k=1}^{s} (\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k \cdot (b_{jk,\mathbf{h}(x)} - b_{jk,\widehat{\mathbf{y}}}) \\ & + \widehat{\beta} \sum_{j=1}^{s}\sum_{k=1}^{s} (b_{jk,\mathbf{h}(x)} - b_{jk,\widehat{\mathbf{y}}}) \\ & + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}}). \end{aligned} \tag{B.9}$$

$$
\begin{aligned}
&\mathrm{regret}_D^{\mathbf{L}}[\mathbf{h}] \\
&= \mathbf{E}_x\left[\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathbf{h}(x)}\rangle - \min_{\widehat{\mathbf{y}}\in\mathcal{Y}}\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\widehat{\mathbf{y}}}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathbf{h}(x)} - \boldsymbol{\ell}_{\widehat{\mathbf{y}}}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \boldsymbol{\eta}(x), \mathbf{A}^\top \mathbf{b}_{\mathbf{h}(x)} + t_{\mathbf{h}(x)}\mathbf{1} - \mathbf{A}^\top \mathbf{b}_{\widehat{\mathbf{y}}} - t_{\widehat{\mathbf{y}}}\mathbf{1}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}) + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\mathbf{1}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}})\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\right]\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \mathbf{A}\boldsymbol{\eta}(x), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\right]\right] \quad \text{(by property of adjoint)} \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \mathbf{A}\boldsymbol{\eta}(x), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}}) - g(\mathbf{f}(x),\widehat{\mathbf{y}}) + g(\mathbf{f}(x),\widehat{\mathbf{y}})\right]\right] \\
&\leq \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \mathbf{A}\boldsymbol{\eta}(x), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}}) - g(\mathbf{f}(x),\widehat{\mathbf{y}})\right]\right] \\
&\qquad \left(\text{Since by Eq. (B.8), } g(\mathbf{f}(x),\widehat{\mathbf{y}}) \leq 0 \text{ for all } \widehat{\mathbf{y}}\right) \\
&= \mathbf{E}_x\Bigg[\max_{\widehat{\mathbf{y}}}\Big[[(\mathbf{A}\boldsymbol{\eta}(x))_0 - \widehat{\alpha}\gamma_{\log}^{-1}(f_0(x)) - \widehat{\beta}]\cdot(b_{0,\mathbf{h}(x)} - b_{0,\widehat{\mathbf{y}}}) \\
&\qquad\qquad + \sum_{j=1}^{s}\sum_{k=1}^{s}[(\mathbf{A}\boldsymbol{\eta}(x))_{jk} - \widehat{\alpha}s(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k - \widehat{\beta}]\cdot(b_{jk,\mathbf{h}(x)} - b_{jk,\widehat{\mathbf{y}}})\Big]\Bigg] \\
&\leq \mathbf{E}_x\Bigg[\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \Big[[(\mathbf{A}\boldsymbol{\eta}(x))_0 - \widehat{\alpha}\gamma_{\log}^{-1}(f_0(x)) - \widehat{\beta}]^2 + \\
&\qquad\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}[(\mathbf{A}\boldsymbol{\eta}(x))_{jk} - \widehat{\alpha}s(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k - \widehat{\beta}]^2\Big]^{\frac{1}{2}}\Bigg] \\
&\qquad \text{(by the Cauchy-Schwarz inequality)} \\
&\leq 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\Bigg[\Big[[(\mathbf{A}\boldsymbol{\eta}(x))_0 - \widehat{\alpha}\gamma_{\log}^{-1}(f_0(x)) - \widehat{\beta}]^2 + \\
&\qquad\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}[(\mathbf{A}\boldsymbol{\eta}(x))_{jk} - \widehat{\alpha}s(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k - \widehat{\beta}]^2\Big]^{\frac{1}{2}}\Bigg]. \qquad \text{(B.10)}
\end{aligned}
$$

For all $x \in \mathcal{X}$, let $\boldsymbol{\zeta}(x) \in \mathbb{R}^{s^2+1}$ be such that

$$\zeta_0(x) = \widehat{\alpha}\gamma_{\log}^{-1}(f_0(x)), \quad \zeta_{jk}(x) = \widehat{\alpha}s(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k\,. \tag{B.11}$$

Then Eq. (B.10) can be written as

$$\begin{aligned}
\mathrm{regret}_D^{\mathbf{L}}[\mathbf{h}] &\leq 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\Big[\|\mathbf{A}\boldsymbol{\eta}(x) - \boldsymbol{\zeta}(x) - \widehat{\beta}\mathbf{1}\|_2\Big] \\
&= 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\Big[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) + \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\zeta}(x) - \widehat{\beta}\mathbf{1}\|_2\Big] \\
&\leq 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\Big[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2 + \|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\zeta}(x) - \widehat{\beta}\mathbf{1}\|_2\Big]\,.
\end{aligned} \tag{B.12}$$

Note that,

$$\begin{aligned}
&\mathbf{E}_x\Big[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2\Big] \\
&= \mathbf{E}_x\Big[\|\mathbf{A}(\mathbf{C}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x) - \mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x)\|_2\Big] \\
&= \mathbf{E}_x\Big[\|\mathbf{A}\big((\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\big)\widetilde{\boldsymbol{\eta}}(x)\|_2\Big] \\
&\leq \|\mathbf{A}\|_2 \cdot \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 \cdot \mathbf{E}_x\Big[\|\widetilde{\boldsymbol{\eta}}(x)\|_2\Big] \\
&\leq \|\mathbf{A}\|_2 \cdot \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2\,.
\end{aligned} \tag{B.13}$$

Moreover,

$$
\begin{aligned}
&\mathbf{E}_x\left[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)-\boldsymbol{\zeta}(x)-\widehat{\beta}\mathbf{1}\|_2^2\right]\\
&=\mathbf{E}_x\Bigg[[(\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x))_0-\widehat{\alpha}\gamma_{\log}^{-1}(f_0(x))-\widehat{\beta}]^2+\\
&\qquad\sum_{j=1}^{s}\sum_{k=1}^{s}[(\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x))_{jk}-\widehat{\alpha}s(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k-\widehat{\beta}]^2\Bigg]\\
&=\mathbf{E}_x\Bigg[[\sum_{\mathbf{y}}\widehat{a}_{0,\mathbf{y}}\cdot\widetilde{\eta}_{\mathbf{y}}(x)-\widehat{\alpha}\gamma_{\log}^{-1}(f_0(x))-\widehat{\beta}]^2+\\
&\qquad\sum_{j=1}^{s}\sum_{k=1}^{s}[\sum_{\mathbf{y}}\widehat{a}_{jk,\mathbf{y}}\cdot\widetilde{\eta}_{\mathbf{y}}(x)-\widehat{\alpha}s(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k-\widehat{\beta}]^2\Bigg]\\
&=\mathbf{E}_x\Bigg[[\sum_{\mathbf{y}}(\widehat{\alpha}\cdot\widehat{a}'_{0,\mathbf{y}}+\widehat{\beta})\cdot\widetilde{\eta}_{\mathbf{y}}(x)-\widehat{\alpha}\gamma_{\log}^{-1}(f_0(x))-\widehat{\beta}]^2+\\
&\qquad\sum_{j=1}^{s}\sum_{k=1}^{s}[\sum_{\mathbf{y}}(\widehat{\alpha}s\cdot\widehat{a}'_{jk,\mathbf{y}}+\widehat{\beta})\cdot\widetilde{\eta}_{\mathbf{y}}(x)-\widehat{\alpha}s(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k-\widehat{\beta}]^2\Bigg]\\
&=\mathbf{E}_x\Bigg[[\widehat{\alpha}(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0-\widehat{\alpha}\gamma_{\log}^{-1}(f_0(x))]^2+\\
&\qquad\sum_{j=1}^{s}\sum_{k=1}^{s}[\widehat{\alpha}s(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk}-\widehat{\alpha}s(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k]^2\Bigg]\\
&=\mathbf{E}_x\left[\widehat{\alpha}^2[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0-\gamma_{\log}^{-1}(f_0(x))]^2+\widehat{\alpha}^2s^2\sum_{j=1}^{s}\sum_{k=1}^{s}[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk}-(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k]^2\right].
\end{aligned}
\tag{B.14}
$$

Note that $(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\in[0,1]$. By Lemma B.2, we have

$$
\begin{aligned}
&\left[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0-\gamma_{\log}^{-1}(f_0(x))\right]^2\\
&\le\frac{1}{2}\mathbf{E}_{y\sim\text{Bernoulli}\left((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\right)}\left[\phi_{\log}\Big(y,f_0(x)\Big)-\phi_{\log}\Big(y,\gamma_{\log}\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\big)\Big)\right].
\end{aligned}
\tag{B.15}
$$

By Lemma B.4, we have

$$\begin{aligned}
&\sum_{k=1}^{s}\Big[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} - (\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k\Big]^2 \\
&\leq \Big[1 - \sum_{k=1}^{s}(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} - 1 + \sum_{k=1}^{s}(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k\Big]^2 \\
&\quad + \sum_{k=1}^{s}\Big[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} - (\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k\Big]^2 \\
&\leq 2\mathbf{E}_{y\sim\mathbf{p}_j(x)}\Big[\phi_{\mathrm{mlog}}\Big(y, (f_{j1}(x), ..., f_{js}(x))\Big) - \phi_{\mathrm{mlog}}\Big(y, \boldsymbol{\gamma}_{\mathrm{mlog}}\big(\mathbf{p}_j(x)\big)\Big)\Big], \qquad \text{(B.16)}
\end{aligned}$$

where $\mathbf{p}_j(x) \in \Delta_{s+1}$ is

$$\mathbf{p}_j(x) = \begin{bmatrix} (\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{j1} \\ \vdots \\ (\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{js} \\ 1 - \sum_{k=1}^{s}(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} \end{bmatrix}. \qquad \text{(B.17)}$$

Then, Eq. (B.14) becomes

$$
\begin{aligned}
&\mathbf{E}_x\left[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)-\boldsymbol{\zeta}(x)-\widehat{\beta}\mathbf{1}\|_2^2\right]\\
&=\mathbf{E}_x\left[\widehat{\alpha}^2[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0-\gamma_{\text{log}}^{-1}(f_0(x))]^2+\widehat{\alpha}^2s^2\sum_{j=1}^{s}\sum_{k=1}^{s}[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk}-(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),\ldots,f_{js}(x)))_k]^2\right]\\
&\le\mathbf{E}_x\left[\widehat{\alpha}^2\frac{1}{2}\mathbf{E}_{y\sim\text{Bernoulli}\left((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\right)}\left[\phi_{\text{log}}\Big(y,f_0(x)\Big)-\phi_{\text{log}}\Big(y,\gamma_{\text{log}}\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\big)\Big)\right]\right.\\
&\qquad\left.+2\widehat{\alpha}^2s^2\sum_{j=1}^{s}\mathbf{E}_{y\sim\mathbf{p}_j(x)}\left[\phi_{\text{mlog}}\Big(y,(f_{j1}(x),\ldots,f_{js}(x))\Big)-\phi_{\text{mlog}}\Big(y,\boldsymbol{\gamma}_{\text{mlog}}\big(\mathbf{p}_j(x)\big)\Big)\right]\right]\\
&\le2\widehat{\alpha}^2s^2\cdot\mathbf{E}_x\left[\mathbf{E}_{y\sim\text{Bernoulli}\left((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\right)}\left[\phi_{\text{log}}\Big(y,f_0(x)\Big)-\phi_{\text{log}}\Big(y,\gamma_{\text{log}}\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\big)\Big)\right]\right.\\
&\qquad\qquad\left.+\sum_{j=1}^{s}\mathbf{E}_{y\sim\mathbf{p}_j(x)}\left[\phi_{\text{mlog}}\Big(y,(f_{j1}(x),\ldots,f_{js}(x))\Big)-\phi_{\text{mlog}}\Big(y,\boldsymbol{\gamma}_{\text{mlog}}\big(\mathbf{p}_j(x)\big)\Big)\right]\right]\\
&=2\widehat{\alpha}^2s^2\cdot\mathbf{E}_x\left[\mathbf{E}_{\mathbf{y}|x\sim\widetilde{\boldsymbol{\eta}}(x)}\left[\psi(\mathbf{y},\mathbf{f}(x))-\inf_{\mathbf{u}\in\mathbb{R}^{s^2+1}}\psi(\mathbf{y},\mathbf{u})\right]\right]\\
&=2\widehat{\alpha}^2s^2\cdot\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]\,. \qquad\qquad (\text{B.18})
\end{aligned}
$$

Combining Eqs. (B.12), (B.13) and (B.18) and applying Jensen's inequality (to the convex function $x\mapsto x^2$), we have

$$
\text{regret}_D^{\mathbf{L}}[\mathbf{h}]\le2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2\cdot\left(\|\mathbf{A}\|_2\|(\mathbf{C}^\top)^{-1}-(\widehat{\mathbf{C}}^\top)^{-1}\|_2+\widehat{\alpha}s\sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]}\right).
$$

We can further bound $\widehat{\alpha}$ by

$$
\begin{aligned}
\widehat{\alpha}&=\widehat{a}_{\max}-\widehat{a}_{\min}\\
&\le2\|\widehat{\mathbf{A}}\|_1\\
&=2\|\mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}\|_1\\
&\le2\|\mathbf{A}\|_1\|(\widehat{\mathbf{C}}^\top)^{-1}\|_1\,.
\end{aligned}
$$

□

B.2.3. Proof of Theorem 6.3

In this section, we prove a more general version of Theorem 6.3 that can handle the case when an estimated noise matrix $\widehat{\mathbf{C}}$ is used in place of the true noise matrix $\mathbf{C}$. Therefore, setting $\widehat{\mathbf{C}} = \mathbf{C}$ in the theorem below proves Theorem 6.3.

Theorem B.5 (More general version of Theorem 6.3). *Consider a general low-rank loss matrix* $\mathbf{L}$ *written as* $\mathbf{L} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ *for some* $\mathbf{A} \in [0,1]^{r\times|\mathcal{Y}|}, \mathbf{B} \in \mathbb{R}^{r\times|\mathcal{Y}|}, \mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$*, under the general CCN model. Assume noise matrix* $\mathbf{C}$ *is invertible. Let* D *be any distribution on* $\mathcal{X} \times \mathcal{Y}$ *with corresponding noisy distribution* $\widetilde{D}$*. Let* $\widehat{\mathbf{C}}$ *be an estimated (invertible) noise matrix for the noise matrix* $\mathbf{C}$*. Suppose NCOC (Section 6.5.3) is run with noisy training sample* $\widetilde{S}$ *(in which examples are sampled i.i.d. from* $\widetilde{D}$*) and estimated noise matrix* $\widehat{\mathbf{C}}$ *in place of* $\mathbf{C}$*. Let* $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ *be as defined in Section 6.5.3. Then we have*

$$\text{regret}_D^{\mathbf{L}}[\widehat{\mathbf{h}}] \le 2\max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \left(\|\mathbf{A}\|_2 \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 + \|\mathbf{A}\|_1 \|(\widehat{\mathbf{C}}^\top)^{-1}\|_1 \sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]} \right).$$

Further, if $\widehat{\mathbf{C}} = \mathbf{C}$*, the above bound can be simplified to*

$$\text{regret}_D^{\mathbf{L}}[\widehat{\mathbf{h}}] \le 2\max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \|\mathbf{A}\|_1 \|(\mathbf{C}^\top)^{-1}\|_1 \sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]}\,.$$

Proof. For clarity, we redefine here all quantities involving $\widehat{\mathbf{C}}$. In particular, define $\widehat{\mathbf{A}} = \mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}$, $\widehat{a}_{\min} = \min_{\mathbf{y},j} \widehat{a}_{j,\mathbf{y}}$ and $\widehat{a}_{\max} = \max_{\mathbf{y},j} \widehat{a}_{j,\mathbf{y}}$, and let

$$\widehat{a}'_{j,\mathbf{y}} = \frac{\widehat{a}_{j,\mathbf{y}} - \widehat{a}_{\min}}{\widehat{a}_{\max} - \widehat{a}_{\min}} \in [0,1] \quad \forall j \in [r]\,.$$

Here, the multi-label surrogate $\psi : \mathcal{Y} \times \mathbb{R}^r \to \mathbb{R}_+$ then becomes

$$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^{r} \Big(\widehat{a}'_{j,\mathbf{y}} \phi(1, u_j) + (1 - \widehat{a}'_{j,\mathbf{y}}) \phi(0, u_j) \Big)\,. \tag{B.19}$$

Also, given $\widehat{q}'_j(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$ estimated from $\widetilde{S}$, the multi-label classifier output by our NCOC algorithm is given by

$$\widehat{\mathbf{h}}(x) = \underset{\widehat{\mathbf{y}} \in \mathcal{Y}}{\operatorname{argmin}} \left\{ t_{\widehat{\mathbf{y}}} + \sum_{j=1}^{r} [(\widehat{a}_{\max} - \widehat{a}_{\min}) \cdot \widehat{q}'_j(x) + \widehat{a}_{\min}] \cdot b_{j,\widehat{\mathbf{y}}} \right\}. \tag{B.20}$$

We use $\langle \cdot, \cdot \rangle$ to denote the standard inner product. Let $\widehat{\alpha} = (\widehat{a}_{\max} - \widehat{a}_{\min})$ and $\widehat{\beta} = \widehat{a}_{\min}$. To simplify notations, in what follows, we let $\phi = \phi_{\log}$, $\gamma = \gamma_{\log}$, $\mathbf{f} = \widehat{\mathbf{f}}$, and $\mathbf{h} = \widehat{\mathbf{h}}$. For $\mathbf{u} \in \mathbb{R}^r$, let $\boldsymbol{\gamma}^{-1}(\mathbf{u}) \in [0, 1]^r$ be such that the i-indexed entry of $\boldsymbol{\gamma}^{-1}(\mathbf{u})$ is simply $\gamma^{-1}(u_i)$.

$$
\begin{aligned}
&\mathrm{regret}_D^{\mathbf{L}}[\mathbf{h}] \\
&= \mathbf{E}_x\left[\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathbf{h}(x)}\rangle - \min_{\widehat{\mathbf{y}}\in\mathcal{Y}}\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\widehat{\mathbf{y}}}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathbf{h}(x)} - \boldsymbol{\ell}_{\widehat{\mathbf{y}}}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \boldsymbol{\eta}(x), \mathbf{A}^\top \mathbf{b}_{\mathbf{h}(x)} + t_{\mathbf{h}(x)}\mathbf{1} - \mathbf{A}^\top \mathbf{b}_{\widehat{\mathbf{y}}} - t_{\widehat{\mathbf{y}}}\mathbf{1}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}) + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\mathbf{1}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}})\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\right]\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \mathbf{A}\boldsymbol{\eta}(x), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\right]\right] \quad \text{(by property of adjoint)} \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \mathbf{A}\boldsymbol{\eta}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1}), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + \right.\right. \\
&\qquad \left.\left. \langle (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1}), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\right]\right] \\
&\le \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \mathbf{A}\boldsymbol{\eta}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1}), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle\right] \\
&\qquad \left(\text{Since by Eq. (B.20), } \langle (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1}), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}}) \le 0 \text{ for all } \widehat{\mathbf{y}}\right) \\
&\le \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2 \cdot \max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\|_2\right] \\
&\qquad \text{(by the Cauchy-Schwarz inequality)} \\
&\le 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2\right] \\
&= 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) + \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2\right] \\
&\le 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2 + \|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2\right] \\
&= 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2\right] + 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2\right].
\end{aligned}
\tag{B.21}
$$

Note that,

$$\begin{aligned}
&\mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2\right] \\
&= \mathbf{E}_x\left[\|\mathbf{A}(\mathbf{C}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x) - \mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x)\|_2\right] \\
&= \mathbf{E}_x\left[\|\mathbf{A}\big((\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\big)\widetilde{\boldsymbol{\eta}}(x)\|_2\right] \\
&\leq \|\mathbf{A}\|_2 \cdot \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 \cdot \mathbf{E}_x\left[\|\widetilde{\boldsymbol{\eta}}(x)\|_2\right] \\
&\leq \|\mathbf{A}\|_2 \cdot \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 \,. \qquad \text{(B.22)}
\end{aligned}$$

Moreover,

$$\begin{aligned}
&\mathbf{E}_x\left[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2^2\right] \\
&= \mathbf{E}_x\left[\|(\widehat{\mathbf{A}} - \widehat{\beta}\mathbf{1}\mathbf{1}^\top)\widetilde{\boldsymbol{\eta}}(x) - \widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x))\|_2^2\right] \\
&= \mathbf{E}_x\left[\widehat{\alpha}^2\|\frac{(\widehat{\mathbf{A}} - \widehat{\beta}\mathbf{1}\mathbf{1}^\top)}{\widehat{\alpha}}\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\gamma}^{-1}(\mathbf{f}(x))\|_2^2\right] \\
&= \mathbf{E}_x\left[\widehat{\alpha}^2\|\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\gamma}^{-1}(\mathbf{f}(x))\|_2^2\right] \quad \left(\text{because } \widehat{\mathbf{A}}' = \frac{\widehat{\mathbf{A}} - \widehat{a}_{\min}\mathbf{1}\mathbf{1}^\top}{\widehat{a}_{\max} - \widehat{a}_{\min}} = \frac{\widehat{\mathbf{A}} - \widehat{\beta}\mathbf{1}\mathbf{1}^\top}{\widehat{\alpha}}\right) \\
&= \mathbf{E}_x\left[\widehat{\alpha}^2\sum_{j=1}^{r}\left[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j - \gamma^{-1}(f_j(x))\right]^2\right]. \qquad \text{(B.23)}
\end{aligned}$$

Note that $(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j \in [0,1]$. By Lemma B.2, we have

$$\left[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j - \gamma^{-1}(f_j(x))\right]^2 \leq \frac{1}{2}\mathbf{E}_{y\sim\text{Bernoulli}\left((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j\right)}\left[\phi\Big(y, f_j(x)\Big) - \phi\Big(y, \gamma\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j\big)\Big)\right].$$

Then Eq. (B.23) becomes

$$
\begin{aligned}
&\mathbf{E}_x\left[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2^2\right] \\
&= \mathbf{E}_x\left[\widehat{\alpha}^2 \sum_{j=1}^{r} \left[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j - \gamma^{-1}(f_j(x))\right]^2\right] \\
&\le \mathbf{E}_x\left[\widehat{\alpha}^2 \sum_{j=1}^{r} \frac{1}{2}\mathbf{E}_{y\sim\text{Bernoulli}\left((\widetilde{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j\right)}\left[\phi\Big(y, f_j(x)\Big) - \phi\Big(y, \gamma\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j\big)\Big)\right]\right] \\
&= \frac{\widehat{\alpha}^2}{2}\mathbf{E}_x\left[\mathbf{E}_{\mathbf{y}|x\sim\widetilde{\boldsymbol{\eta}}(x)}\left[\psi(\mathbf{y}, \mathbf{f}(x)) - \inf_{\mathbf{u}\in\mathbb{R}^r} \psi(\mathbf{y}, \mathbf{u})\right]\right] \\
&= \frac{\widehat{\alpha}^2}{2}\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]\,. \qquad\qquad \text{(B.24)}
\end{aligned}
$$

Combining Eqs. (B.21), (B.22) and (B.24) and applying Jensen's inequality (to the convex function $x \mapsto x^2$), we have

$$
\text{regret}_D^{\mathbf{L}}[\mathbf{h}] \le 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \left(\|\mathbf{A}\|_2\|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 + \widehat{\alpha}\sqrt{\frac{1}{2}\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]}\right).
$$

We can further bound $\widehat{\alpha}$ by

$$
\begin{aligned}
\widehat{\alpha} &= \widehat{a}_{\max} - \widehat{a}_{\min} \\
&\le 2\|\widehat{\mathbf{A}}\|_1 \\
&= 2\|\mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}\|_1 \\
&\le 2\|\mathbf{A}\|_1\|(\widehat{\mathbf{C}}^\top)^{-1}\|_1\,.
\end{aligned}
$$

□

B.3. Supplement to Section 6.7

Under an STSN model where the set of s tags $\mathcal{T} = [s]$ is partitioned into K ($K \leq s$) groups of tags $G_1, ..., G_K$, the label space $\mathcal{Y} \subseteq \{0,1\}^s$ contains only $\mathbf{y} \in \{0,1\}^s$ with $\|\mathbf{y}_{G_k}\|_1 \leq 1$ for all groups G_k (since each group has at most 1 active tag). In this setting as well, the output mappings $\widehat{\mathbf{f}}(x) \mapsto \widehat{\mathbf{h}}(x)$ that appear at the end of the NCEFP and NCOC algorithms, that require solving a combinatorial optimization problem over $\widehat{\mathbf{y}} \in \mathcal{Y}$, can be computed efficiently. We give details below.

Reduced low-rank decomposition for $\mathbf{L}^{F_1}$ for sparse label space $\mathcal{Y}$, and reduced vector function $\widehat{\mathbf{f}}$. Let $\mathcal{Y}_K = \{\mathbf{y} \in \{0,1\}^s : \|\mathbf{y}\|_1 \leq K\}$. For $\mathcal{Y} \subseteq \mathcal{Y}_K$ (as is the case under the STSN model), in the decomposition of $\mathbf{L}^{F_1}$ in Example 6.2, several entries of matrix $\mathbf{A}$ become 0; in particular, $a_{jk,\mathbf{y}} = 0$ for $k > K$. Therefore, the factorization in this case simplifies to

$$\ell^{F_1}_{\mathbf{y},\widehat{\mathbf{y}}} = 1 - \mathbf{1}(\|\mathbf{y}\|_1 = 0) \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{j=1}^{s} \sum_{k=1}^{K} \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j \cdot \frac{2 \cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1} \,. \tag{B.25}$$

In other words, we have $\mathbf{L}^{F_1} = (\mathbf{A}^{\text{red}})^\top (\mathbf{B}^{\text{red}}) + \mathbf{1}\mathbf{t}^\top$ where $\mathbf{A}^{\text{red}} \in [0,1]^{(sK+1) \times |\mathcal{Y}|}$ with $a^{\text{red}}_{0,\mathbf{y}} = a_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)$ and $a^{\text{red}}_{jk,\mathbf{y}} = a_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j \; \forall j \in [s], \forall k \in [K]$, $\mathbf{B}^{\text{red}} \in \mathbb{R}^{(sK+1) \times |\mathcal{Y}|}$ with $b^{\text{red}}_{0,\widehat{\mathbf{y}}} = b_{0,\widehat{\mathbf{y}}} = -\mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0)$ and $b^{\text{red}}_{jk,\widehat{\mathbf{y}}} = b_{jk,\widehat{\mathbf{y}}} = -\frac{2 \cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1} \; \forall j \in [s], \forall k \in [K]$, and $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ with $t_{\widehat{\mathbf{y}}} = 1$. Accordingly, the NCEFP algorithm in this case reduces to solving one binary problem and s ($(K+1)$-class) multiclass problems. The NCOC algorithm requires solving $(sK+1)$ binary problems. Therefore, in both cases, one needs to learn only a $(sK+1)$-dimensional real-valued vector function $\widehat{\mathbf{f}} : \mathcal{X} \rightarrow \mathbb{R}^{sK+1}$; vector predictions $\widehat{\mathbf{f}}(x)$ then need to be mapped to the label space $\mathcal{Y}$ to obtain the final multi-label predictions $\widehat{\mathbf{h}}(x) \in \mathcal{Y}$. The output mappings for F_1-measure below therefore map vectors $\mathbf{u} \in \mathbb{R}^{sK+1}$ to multi-label predictions $\widehat{\mathbf{y}} \in \mathcal{Y}$.

NCEFP output mapping for F_1-measure under STSN. Algorithm B.5 specifies the output mapping $\widehat{\mathbf{f}}(x) \mapsto \widehat{\mathbf{h}}(x)$ in this case. The complexity for computing $\mathbf{T}$ is $O(sK^2)$. The complexity for the for loop is $O(sK^2)$. So the total complexity of the output mapping is $O(sK^2)$. (The procedure here is a modification of the procedure described in Dembczynski et al. (2011); Zhang et al. (2020) for the case when $\mathcal{Y} = \{0,1\}^s$. The complexity of the original output mapping in that case is $O(s^3)$;

therefore, our complexity of $O(sK^2)$ is an improvement under the STSN setting.)

Algorithm B.5 NCEFP output mapping for F_1-measure under STSN

```
 1: Input: Vector $\mathbf{u} = \big(u_0, (u_{jk})_{j=1,...,s,k=1,...,K}\big)^\top \in \mathbb{R}^{sK+1}$
 2: Define matrices $\mathbf{Q} \in [0,1]^{s\times K}$ and $\mathbf{V} \in \mathbb{R}^{K\times K}$ as follows:
        $Q_{j,k} = s(\widetilde{a}_{\max} - \widetilde{a}_{\min})(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(u_{j1}, ..., u_{js}))_k + \widetilde{a}_{\min}$
        $V_{k,l} = \frac{-2}{k+l}$
 3: Compute $\mathbf{T} = \mathbf{QV}$
 4: For $l = 1 \dots K$: // $K$ is the number of groups
 5:     For each group $G_k$, define $g_k^l = \text{argmin}_{j\in G_k}\{T_{j,l}\}$
 6:     Find the $l$ smallest numbers among $\{T_{g_1^l,l}, ..., T_{g_K^l,l}\}$; call them $T_{j_1^l,l}, \dots, T_{j_l^l,l}$
 7:     Define $\widehat{\mathbf{y}}^{l,*} \in \mathcal{Y} \cap \{\mathbf{y} \in \{0,1\}^s : \|\mathbf{y}\|_1 = l\}$ as follows:
            $\widehat{y}_j^{l,*} = \begin{cases} 1 & \text{if } j \in \{j_1^l, \dots, j_l^l\} \\ 0 & \text{otherwise.} \end{cases}$
 8:     Set $z_l^* = \sum_{j=1}^s \widehat{y}_j^{l,*} T_{j,l}$
 9: End for
10: Pick $\widehat{\mathbf{y}}^*$ as follows:
        $\widehat{\mathbf{y}}^* \in \underset{\widehat{\mathbf{y}}\in\{\mathbf{0},\widehat{\mathbf{y}}^{1,*},...,\widehat{\mathbf{y}}^{K,*}\}}{\text{argmin}} -\mathbf{1}(\widehat{\mathbf{y}} = \mathbf{0})\cdot((\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma_{\log}^{-1}(u_0) + \widetilde{a}_{\min}) + \mathbf{1}(\widehat{\mathbf{y}} \neq \mathbf{0}) \cdot z_{\|\widehat{\mathbf{y}}\|_1}^*$
11: Output: $\widehat{\mathbf{y}}^* \in \mathcal{Y}$
```

NCOC output mapping for Hamming loss under STSN. Algorithm B.6 specifies the output mapping $\widehat{\mathbf{f}}(x) \mapsto \widehat{\mathbf{h}}(x)$ in this case. The total complexity is $O(s)$.

NCOC output mapping for F_1-measure under STSN. Algorithm B.7 specifies the output mapping $\widehat{\mathbf{f}}(x) \mapsto \widehat{\mathbf{h}}(x)$ in this case (Algorithm B.7 differs from Algorithm B.5 in line 2.). The complexity for computing $\mathbf{T}$ is $O(sK^2)$. The complexity for the for loop is $O(sK^2)$. So the total complexity of the output mapping is $O(sK^2)$. (The procedure here is a modification of the procedure described in Dembczynski et al. (2011); Zhang et al. (2020) for the case when $\mathcal{Y} = \{0,1\}^s$. The complexity of the original output mapping in that case is $O(s^3)$; therefore, our complexity of $O(sK^2)$ is an improvement under the STSN setting.)

Algorithm B.6 NCOC output mapping for Hamming loss under STSN

```
1: Input: Vector $\mathbf{u} \in \mathbb{R}^s$
2: Define $\widehat{\mathbf{y}} \in \{0,1\}^s$ as follows:
   $$\widehat{y}_j = \begin{cases} 1 & \text{if } (\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma_{\log}^{-1}(u_j) + \widetilde{a}_{\min} > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$
3: For $k = 1 \dots K$: // $K$ is the number of groups
4:     if $\|\widehat{\mathbf{y}}_{G_k}\|_1 > 1$:
5:         Set $\widehat{y}_j = 0$ for all $j \in G_k$
6:         Define $j_k = \mathrm{argmax}_{j \in G_k} (\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma_{\log}^{-1}(u_j) + \widetilde{a}_{\min}$, and set $\widehat{y}_{j_k} = 1$
7:     End if
8: End for
9: Output: $\widehat{\mathbf{y}} \in \mathcal{Y}$
```

Algorithm B.7 NCOC output mapping for F_1-measure under STSN

```
1: Input: Vector $\mathbf{u} = \big(u_0, (u_{jk})_{j=1,\dots,s,k=1,\dots,K}\big)^\top \in \mathbb{R}^{sK+1}$
2: Define matrices $\mathbf{Q} \in [0,1]^{s \times K}$ and $\mathbf{V} \in \mathbb{R}^{K \times K}$ as follows:
   $$Q_{j,k} = (\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma_{\log}^{-1}(u_{jk}) + \widetilde{a}_{\min}$$
   $$V_{k,l} = \frac{-2}{k+l}$$
3: Compute $\mathbf{T} = \mathbf{QV}$
4: For $l = 1 \dots K$: // $K$ is the number of groups
5:     For each group $G_k$, define $g_k^l = \mathrm{argmin}_{j \in G_k}\{T_{j,l}\}$
6:     Find the $l$ smallest numbers among $\{T_{g_1^l,l}, ..., T_{g_K^l,l}\}$; call them $T_{j_1^l,l}, \dots, T_{j_l^l,l}$
7:     Define $\widehat{\mathbf{y}}^{l,*} \in \mathcal{Y} \cap \{\mathbf{y} \in \{0,1\}^s : \|\mathbf{y}\|_1 = l\}$ as follows:
       $$\widehat{y}_j^{l,*} = \begin{cases} 1 & \text{if } j \in \{j_1^l, \dots, j_l^l\} \\ 0 & \text{otherwise.} \end{cases}$$
8:     Set $z_l^* = \sum_{j=1}^s \widehat{y}_j^{l,*} T_{j,l}$
9: End for
10: Pick $\widehat{\mathbf{y}}^*$ as follows:
   $$\widehat{\mathbf{y}}^* \in \underset{\widehat{\mathbf{y}} \in \{\mathbf{0}, \widehat{\mathbf{y}}^{1,*}, \dots, \widehat{\mathbf{y}}^{K,*}\}}{\mathrm{argmin}} -\mathbf{1}(\widehat{\mathbf{y}} = \mathbf{0}) \cdot ((\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma_{\log}^{-1}(u_0) + \widetilde{a}_{\min}) + \mathbf{1}(\widehat{\mathbf{y}} \neq \mathbf{0}) \cdot z^*_{\|\widehat{\mathbf{y}}\|_1}$$
11: Output: $\widehat{\mathbf{y}}^* \in \mathcal{Y}$
```

B.4. Supplement to Section 6.8

B.4.1. Synthetic data: data generating process for F_1-measure under STSN

We generated a multi-label dataset with instances x in $\mathcal{X} = \mathbb{R}^{100}$ and $s = 10$ tags with $K = 5$ groups $\mathcal{G} = \{\{1,2,3\},\{4,5,6\},\{7,8\},\{9\},\{10\}\}$, so $|\mathcal{Y}| = 192$ and $\|\mathbf{y}\|_1 \leq K = 5$ for all $\mathbf{y} \in \mathcal{Y}$. By the reduced factorization of $\mathbf{L}^{F_1}$ in Eq. (B.25) in Section B.3, in this case, $\mathbf{A}^{\text{red}} \in [0,1]^{(sK+1)\times|\mathcal{Y}|} = [0,1]^{51\times 192}$ consists of:

$$a^{\text{red}}_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)\,; \qquad a^{\text{red}}_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k)\cdot y_j \quad \forall j \in [s],\, \forall k \in [K]\,.$$

We first fixed matrix $\mathbf{W} \in [0.1,1]^{51\times 100}$ with entries drawn uniformly; we checked that $\mathbf{W}$ has full row rank. We also fixed a vector $\boldsymbol{\alpha} \in [0.01, 0.02]^{192}$ with entries drawn uniformly. To generate a data point $(x, \mathbf{y})$, we then did the following: we first sampled $\boldsymbol{\eta}(x) \in \Delta_{192} \equiv \Delta_{|\mathcal{Y}|}$ from Dirichlet$(\boldsymbol{\alpha})$. We set $\mathbf{q}(x) = \mathbf{A}^{\text{red}}\boldsymbol{\eta}(x) \in [0,1]^{51}$. We then took $x = \mathbf{W}^{\dagger}\mathbf{q}(x)$, and drew $\mathbf{y} \sim \boldsymbol{\eta}(x)$, where $\mathbf{W}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{W}$.

B.4.2. Synthetic data: data generating process for Hamming loss under IFN

We generated a multi-label dataset with instances x in $\mathcal{X} = \mathbb{R}^{100}$ and $s = 8$. Here, $\mathcal{Y} = \{0,1\}^8$. By the factorization of $\mathbf{L}^{\text{Ham}}$ in Eq. (6.3), in this case, $\mathbf{A} \in [0,1]^{s\times 2^s} = [0,1]^{8\times 256}$. We first fixed matrix $\mathbf{W} \in [0.1,1]^{8\times 100}$ with entries drawn uniformly; we checked that $\mathbf{W}$ has full row rank. Since labels $\mathbf{y}$ in real data tend to be very sparse (have few active tags), we simulated this observation by considering a subset of $\mathcal{Y}$, denoted by $\mathcal{Y}_4$, that contains labels $\mathbf{y}$ with $\|\mathbf{y}\|_1 \leq 4$, so $\mathcal{Y}_4 = \{\mathbf{y} \in \mathcal{Y} : \|\mathbf{y}\|_1 \leq 4\}$. Then we fixed a vector $\boldsymbol{\alpha} \in [0.01, 0.02]^{|\mathcal{Y}_4|}$ with entries drawn uniformly. To generate a data point $(x, \mathbf{y})$, we first sampled $\bar{\boldsymbol{\eta}}(x) \in \Delta_{|\mathcal{Y}_4|}$ from Dirichlet$(\boldsymbol{\alpha})$. Then we set $\boldsymbol{\eta}(x) \in \Delta_{2^s} \equiv \Delta_{|\mathcal{Y}|}$ as follows: for $\mathbf{y} \in \mathcal{Y}$, if $\mathbf{y} \in \mathcal{Y}_4$, set the $\mathbf{y}$-indexed entry of $\boldsymbol{\eta}(x)$ to be the value in the $\mathbf{y}$-indexed entry of $\bar{\boldsymbol{\eta}}(x)$; if $\mathbf{y} \notin \mathcal{Y}_4$, then set the $\mathbf{y}$-indexed entry of $\boldsymbol{\eta}(x)$ to be a small value 10^{-3}. Afterwards, we re-normalized $\boldsymbol{\eta}(x)$. We set $\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x) \in [0,1]^8$. We then took $x = \mathbf{W}^{\dagger}\mathbf{q}(x)$, and drew $\mathbf{y} \sim \boldsymbol{\eta}(x)$, where $\mathbf{W}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{W}$.

B.4.3. Synthetic data: additional implementation details

For all synthetic data experiments (and for all algorithms in the experiments), we used the Adam optimizer (Kingma and Ba, 2015) provided by PyTorch (Paszke et al., 2019) with batch size 100 and no weight decay. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was halved at the end of every 5 epochs.

B.4.4. Real data: additional implementation details for experiments on Mediamill data

Regularization parameters were chosen by cross-validation from $\{0, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. We used the Adam optimizer provided by PyTorch with batch size 100. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was halved at the end of every 5 epochs.

B.4.5. Real data: additional implementation details for experiments on Multi-MNIST data

Hamming loss under IFN. We chose the following neural network architecture for all algorithms:[31]

```
model = torch.nn.Sequential(
    torch.nn.Conv2d(1, 32, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(32, 64, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(64, 128, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Flatten(),
    torch.nn.LazyLinear(128),
```

[31] We used the Adagrad optimizer (Duchi et al., 2011) provided by PyTorch with batch size 128 and weight decay 0.001. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was tuned automatically by the optimizer.

```
    torch.nn.ReLU(),
    torch.nn.LazyLinear(output_dim)
)
```

Hamming loss and F_1-measure under STSN. For Hamming loss under STSN, we chose the following neural network architecture for all algorithms:[32]

```
model = torch.nn.Sequential(
    torch.nn.Conv2d(1, 32, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(32, 64, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(64, 128, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Flatten(),
    torch.nn.LazyLinear(128),
    torch.nn.ReLU(),
    torch.nn.LazyLinear(output_dim)
)
```

For F_1-measure under STSN, we chose the following neural network architecture for all algorithms:[33]

```
model = torch.nn.Sequential(
```

[32]We used the Adagrad optimizer provided by PyTorch with batch size 128 and weight decay 0. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was tuned automatically by the optimizer.

[33]We used the Adagrad optimizer provided by PyTorch with batch size 128 and weight decay 0. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was tuned automatically by the optimizer.

```
    torch.nn.Conv2d(1, 32, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(32, 64, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(64, 128, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Flatten(),
    torch.nn.LazyLinear(512),
    torch.nn.ReLU(),
    torch.nn.LazyLinear(output_dim)
)
```

APPENDIX C

SUPPLEMENTAL MATERIAL FOR CHAPTER 7

Proposition C.1. *$\mathcal{C}_D$ is a convex set.*

Proof. Let $\mathbf{C}$ and $\mathbf{C}'$ be two confusion matrices in $\mathcal{C}_D$. There exist randomized classifiers $h, h' : \mathcal{X} \to \Delta_n$ such that $\mathbf{C} = \mathbf{C}^D[h]$ and $\mathbf{C}' = \mathbf{C}^D[h']$. For any $\gamma \in [0,1]$, $\gamma h + (1-\gamma)h'$ is a randomized classifier. So

$$\begin{aligned}
&\gamma C_{i,j} + (1-\gamma)C'_{i,j} \\
&= \gamma \mathbf{P}_{(X,Y)\sim D, Y'\sim h(X)}(Y=i, Y'=j) + (1-\gamma)\mathbf{P}_{(X,Y)\sim D, Y'\sim h'(X)}(Y=i, Y'=j) \\
&= \mathbf{P}_{(X,Y)\sim D}(X,Y)\Big[\gamma \mathbf{P}_{Y'\sim h(X)}(Y=i, Y'=j|X,Y) + (1-\gamma)\mathbf{P}_{Y'\sim h'(X)}(Y=i, Y'=j)\Big] \\
&= \mathbf{P}_{(X,Y)\sim D}(X,Y)\mathbf{P}_{Y'\sim \gamma h(X)+(1-\gamma)h'(X)}(Y=i, Y'=j|X,Y) \\
&= C^D_{i,j}[\gamma h + (1-\gamma)h'] \,.
\end{aligned}$$

It follows that $\gamma\mathbf{C} + (1-\gamma)\mathbf{C}' = \mathbf{C}^D[\gamma h + (1-\gamma)h'] \in \mathcal{C}_D$. $\square$

Learning from noisy labels for monotonic convex performance measures.

Theorem C.2 (Form of Bayes optimal classifier for monotonic convex ψ in the noisy label setting)**.** *Let D be a distribution such that $\boldsymbol{\eta}(X)$ is a continuous random vector. Assume that monotonic convex performance measure ψ is differentiable over $\mathcal{C}_D$. Let $h^* : \mathcal{X} \to \Delta_n$ be a Bayes optimal classifier for ψ-performance w.r.t. D.*[34] *Define $\mathbf{L}^* = (\mathbf{T}^\top)^{-1}\nabla\psi(\mathbf{C}^D[h^*])$. Then any Bayes optimal classifier for $\mathbf{L}^*$-performance w.r.t. $\widetilde{D}$ is also Bayes optimal for ψ-performance w.r.t. D.*

Lemma C.3 (Lemma 25 of Narasimhan et al. (2015))**.** *Let D be a distribution such that $\boldsymbol{\eta}(X)$ is a continuous random vector. Let $\mathbf{L} \in \mathbb{R}^{n\times n}$ be such that no two columns are identical. Denote by*

[34] The existence of such a classifier is guaranteed by Lemma 12 of Narasimhan et al. (2015).

$\overline{\mathcal{C}_D}$ the closure of $\mathcal{C}_D$. Then,

$$\operatorname*{argmin}_{\mathbf{C}\in\mathcal{C}_D}\langle \mathbf{L}, \mathbf{C}\rangle = \operatorname*{argmin}_{\mathbf{C}\in\overline{\mathcal{C}_D}}\langle \mathbf{L}, \mathbf{C}\rangle\,.$$

Moreover, the above set is a singleton.

Proof of Theorem C.2.

Proof. We first show

$$\operatorname*{argmin}_{\widetilde{\mathbf{C}}\in\mathcal{C}_{\widetilde{D}}}\langle \mathbf{L}^*, \widetilde{\mathbf{C}}\rangle = \operatorname*{argmin}_{\widetilde{\mathbf{C}}\in\overline{\mathcal{C}_{\widetilde{D}}}}\langle \mathbf{L}^*, \widetilde{\mathbf{C}}\rangle\,, \tag{C.1}$$

and the above set is a singleton.

Since D is a distribution such that $\boldsymbol{\eta}(X)$ is a continuous random vector, $\widetilde{\boldsymbol{\eta}}(X) = \mathbf{T}\boldsymbol{\eta}(X)$ is a continuous random vector as well. By the monotonic condition on ψ, $\nabla\psi(\mathbf{C}^D[h^*])$ has negative diagonal entries and non-negative off-diagonal entries. So no two columns of $\nabla\psi(\mathbf{C}^D[h^*])$ are identical. Because $\mathbf{L}^*$ is $(\mathbf{T}^\top)^{-1}\nabla\psi(\mathbf{C}^D[h^*])$ and $\mathbf{T}$ is invertible, it follows that no two columns of $\mathbf{L}^*$ are identical. Applying Lemma C.3 establishes the claim above.

Let $\widetilde{\mathbf{C}}^* = \operatorname{argmin}_{\widetilde{\mathbf{C}}\in\mathcal{C}_{\widetilde{D}}}\langle \mathbf{L}^*, \widetilde{\mathbf{C}}\rangle$. Since $\mathbf{L}^*$ is $(\mathbf{T}^\top)^{-1}\nabla\psi(\mathbf{C}^D[h^*])$, we have

$$\langle \mathbf{L}^*, \widetilde{\mathbf{C}}^*\rangle = \langle (\mathbf{T}^\top)^{-1}\nabla\psi(\mathbf{C}^D[h^*]), \widetilde{\mathbf{C}}^*\rangle\,. \tag{C.2}$$

Recall $h^* : \mathcal{X} \to \Delta_n$ is a Bayes optimal classifier for ψ-performance w.r.t. D. Lemma 12 of Narasimhan et al. (2015) shows $\mathbf{C}^D[h^*] = \operatorname{argmin}_{\mathbf{C}\in\overline{\mathcal{C}_D}}\psi(\mathbf{C})$. Since ψ is differentiable over $\mathcal{C}_D$ and $\mathcal{C}_D$ is convex, by first order optimality condition, for all $\mathbf{C}\in\mathcal{C}_D$,

$$\langle \nabla\psi(\mathbf{C}^D[h^*]), \mathbf{C}^D[h^*]\rangle \le \langle \nabla\psi(\mathbf{C}^D[h^*]), \mathbf{C}\rangle\,. \tag{C.3}$$

We are now ready show any Bayes optimal classifier for $\mathbf{L}^*$-performance w.r.t. $\widetilde{D}$ is also Bayes optimal for ψ-performance w.r.t. D.

For Bayes optimal classifier g^* for $\mathbf{L}^*$-performance w.r.t. $\widetilde{D}$, we have $\mathbf{C}^{\widetilde{D}}[g^*] = \widetilde{\mathbf{C}}^*$ due to the singleton set in Eq. (C.1). By Eqs. (C.2) and (C.3),

$$\begin{aligned}\langle \mathbf{L}^*, \mathbf{C}^{\widetilde{D}}[g^*]\rangle &= \langle (\mathbf{T}^\top)^{-1}\nabla\psi(\mathbf{C}^D[h^*]), \mathbf{C}^{\widetilde{D}}[g^*]\rangle \\ &= \langle \nabla\psi(\mathbf{C}^D[h^*]), \mathbf{T}^{-1}\mathbf{C}^{\widetilde{D}}[g^*]\rangle \\ &= \langle \nabla\psi(\mathbf{C}^D[h^*]), \mathbf{C}^D[g^*]\rangle \\ &\geq \langle \nabla\psi(\mathbf{C}^D[h^*]), \mathbf{C}^D[h^*]\rangle\,.\end{aligned}$$

On the other hand,

$$\begin{aligned}\langle \nabla\psi(\mathbf{C}^D[h^*]), \mathbf{C}^D[h^*]\rangle &= \langle (\mathbf{T}^\top)^{-1}\nabla\psi(\mathbf{C}^D[h^*]), \mathbf{T}\mathbf{C}^D[h^*]\rangle \\ &= \langle \mathbf{L}^*, \mathbf{T}\mathbf{C}^D[h^*]\rangle \\ &\geq \langle \mathbf{L}^*, \mathbf{C}^{\widetilde{D}}[g^*]\rangle\,.\end{aligned}$$

So, we have

$$\langle \nabla\psi(\mathbf{C}^D[h^*]), \mathbf{C}^D[g^*]\rangle = \langle \mathbf{L}^*, \mathbf{C}^{\widetilde{D}}[g^*]\rangle = \langle \nabla\psi(\mathbf{C}^D[h^*]), \mathbf{C}^D[h^*]\rangle\,.$$

By Lemma C.3 and Eq. (C.3), this means $\mathbf{C}^D[g^*] = \mathbf{C}^D[h^*]$. So, g^* is also Bayes optimal for ψ-performance measure w.r.t. D. $\square$

Learning from noisy labels for ratio-of-linear performance measures.

Theorem C.4 (Form of Bayes optimal classifier for ratio-of-linear ψ in the noisy label setting)**.** *Consider ratio-of-linear performance measure $\psi(\mathbf{C}) = \frac{\langle \mathbf{A},\mathbf{C}\rangle}{\langle \mathbf{B},\mathbf{C}\rangle}$ with $\langle \mathbf{B},\mathbf{C}\rangle > 0\ \forall \mathbf{C} \in \mathcal{C}_D$. Let $\Psi_D^{\psi,*} = \inf_{\mathbf{C}\in\mathcal{C}_D} \psi(\mathbf{C})$ be the Bayes optimal ψ-performance w.r.t. D. Let $\mathbf{L}^* = (\mathbf{T}^\top)^{-1}(\mathbf{A} - \Psi_D^{\psi,*}\mathbf{B})$. Then any Bayes optimal classifier for $\mathbf{L}^*$-performance w.r.t. $\widetilde{D}$ is also Bayes optimal for ψ-performance* *w.r.t.* D.

Proof. Let $\mathbf{L}' = \mathbf{T}^\top \mathbf{L}^* = (\mathbf{A} - \Psi_D^{\psi,*}\mathbf{B})$. Theorem 11 of Narasimhan et al. (2015) shows that any Bayes optimal classifier for $\mathbf{L}'$-performance w.r.t. D is also Bayes optimal for for ψ-performance w.r.t. D. By Proposition 7.1, any Bayes optimal classifier for $\mathbf{L}^*$-performance w.r.t. $\widetilde{D}$ is also Bayes optimal for $\mathbf{L}'$-performance w.r.t. D. The claim follows. $\square$

BIBLIOGRAPHY

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Shivani Agarwal. Surrogate regret bounds for bipartite ranking via strongly proper losses. *Journal of Machine Learning Research*, 15:1653–1674, 2014.

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *J. Mach. Learn. Res.*, 1:113–141, 2000.

Dana Angluin and Philip D. Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, 1987.

Javed A. Aslam and Scott E. Decatur. On the sample complexity of noise-tolerant learning. *Inf. Process. Lett.*, 57(4):189–195, 1996.

Junwen Bai, Shufeng Kong, and Carla P. Gomes. Disentangled variational autoencoder based multi-label classification with covariance-aware multivariate probit model. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4313–4321. ijcai.org, 2020.

Han Bao and Masashi Sugiyama. Calibrated surrogate maximization of linear-fractional utility in binary classification. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, volume 108 of *Proceedings of Machine Learning Research*, pages 2337–2347. PMLR, 2020.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482, 2002.

Peter L. Bartlett and Marten H. Wegkamp. Classification with a reject option using a hinge loss. *J. Mach. Learn. Res.*, 9:1823–1840, 2008.

Peter L. Bartlett, Michael Collins, Benjamin Taskar, and David A. McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004]*, pages 113–120, 2004.

Peter L. Bartlett, Michael Jordan, and Jon McAuliffe. Convexity, classification and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Avrim Blum and Tom M. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT 1998*, pages 92–100. ACM, 1998.

Tom Bylander. Learning linear threshold functions in the presence of classification noise. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory, COLT 1994*, pages 340–347. ACM, 1994.

Yuzhou Cao, Tianchi Cai, Lei Feng, Lihong Gu, Jinjie Gu, Bo An, Gang Niu, and Masashi Sugiyama. Generalizing consistent multi-class classification with rejection to be compatible with arbitrary losses. In *NeurIPS*, 2022.

Nicolò Cesa-Bianchi, Eli Dichterman, Paul Fischer, Eli Shamir, and Hans Ulrich Simon. Sample-efficient strategies for learning in the presence of noise. *J. ACM*, 46(5):684–719, 1999.

Nontawat Charoenphakdee, Zhenghang Cui, Yivan Zhang, and Masashi Sugiyama. Classification with rejection based on cost-sensitive classification. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 1507–1517. PMLR, 2021.

Jiacheng Cheng, Tongliang Liu, Kotagiri Ramamohanarao, and Dacheng Tao. Learning with bounded instance and label-dependent label noise. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 1789–1799. PMLR, 2020.

C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Trans. Inf. Theory*, 16(1): 41–46, 1970.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *J. Mach. Learn. Res.*, 9:1775–1822, 2008.

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Boosting with abstention. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pages 1660–1668, 2016a.

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *Algorithmic Learning Theory - 27th International Conference, ALT 2016*, volume 9925 of *Lecture Notes in Computer Science*, pages 67–82, 2016b.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 279–286, 2010a.

Krzysztof Dembczynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. Regret analysis for performance metrics in multi-label classification: The case of hamming and subset zero-one loss. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Proceedings, Part I*, volume 6321 of *Lecture Notes in Computer Science*, pages 280–295. Springer, 2010b.

Krzysztof Dembczynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. An exact algorithm for F-measure maximization. In *Advances in Neural Information Processing Systems 24*, pages 1404–1412, 2011.

Krzysztof Dembczynski, Arkadiusz Jachnik, Wojciech Kotlowski, Willem Waegeman, and Eyke Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1130–1138, 2013.

Krzysztof Dembczynski, Wojciech Kotlowski, Oluwasanmi Koyejo, and Nagarajan Natarajan. Consistency analysis for binary classification revisited. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 961–969. PMLR, 2017.

Thomas Dietterich and G Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

John Duchi, Lester Mackey, and Michael Jordan. On the consistency of ranking algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *J. Mach. Learn. Res.*, 11:1605–1641, 2010.

Jun-Peng Fang and Min-Ling Zhang. Partial multi-label learning via credible label elicitation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 3518–3525. AAAI Press, 2019.

Thomas Finley and Thorsten Joachims. Training structural svms when exact inference is intractable. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, volume 307 of *ACM International Conference Proceeding Series*, pages 304–311. ACM, 2008.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Networks Learn. Syst.*, 25(5):845–869, 2014.

Wei Gao and Zhi-Hua Zhou. On the consistency of multi-label learning. *Artificial Intelligence*, 199-200:22–44, 2013.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 4878–4887, 2017.

Aritra Ghosh, Naresh Manwani, and P. S. Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015.

Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017*, pages 1919–1925. AAAI Press, 2017.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.

Bo Han, Quanming Yao, Tongliang Liu, Gang Niu, Ivor W. Tsang, James T. Kwok, and Masashi Sugiyama. A survey of label-noise representation learning: Past, present and future. *CoRR*, abs/2011.04406, 2020.

Hangfeng He, Mingyuan Zhang, Qiang Ning, and Dan Roth. Foreseeing the benefits of incidental supervision. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 1782–1800. Association for Computational Linguistics, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE Computer Society, 2016.

Mengying Hu, Hu Han, Shiguang Shan, and Xilin Chen. Multi-label learning from noisy labels with non-linear feature transformation. In *Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, 2018*, volume 11365 of *Lecture Notes in Computer Science*, pages 404–419. Springer, 2018.

Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 377–384, 2005.

Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998.

Kenneth Kennedy, Brian Mac Namee, and Sarah Jane Delany. Learning without default: A study of one-class classification and the low-default portfolio problem. In *Artificial Intelligence and Cognitive Science - 20th Irish Conference, AICS 2009, Revised Selected Papers*, volume 6206 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 2009.

Jin-Dong Kim, Yue Wang, and Yasunori Yamamoto. The genia event extraction shared task, 2013 edition - overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15. Association for Computational Linguistics, 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, 2015.

Oluwasanmi Koyejo, Nagarajan Natarajan, Pradeep Ravikumar, and Inderjit S. Dhillon. Consistent binary classification with generalized performance metrics. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 2744–2752, 2014.

Oluwasanmi Koyejo, Nagarajan Natarajan, Pradeep Ravikumar, and Inderjit S. Dhillon. Consistent multilabel classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 3321–3329, 2015.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Himanshu Kumar, Naresh Manwani, and P. S. Sastry. Robust learning of multi-label classifiers under label noise. In *CoDS-COMAD 2020: 7th ACM IKDD CoDS and 25th COMAD*, pages 90–97. ACM, 2020.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289. Morgan Kaufmann, 2001.

Steve Lawrence, Ian Burns, Andrew D. Back, Ah Chung Tsoi, and C. Lee Giles. Neural network classification and prior class probabilities. In *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 295–309. Springer, 2012.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

Claude Lemaréchal. S. boyd, l. vandenberghe, convex optimization, cambridge university press, 2004 hardback, ISBN 0 521 83378 7. *Eur. J. Oper. Res.*, 170(1):326–327, 2006.

Jia Li, Kaiser Asif, Hong Wang, Brian D. Ziebart, and Tanya Y. Berger-Wolf. Adversarial sequence tagging. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, pages 1690–1696. IJCAI/AAAI Press, 2016.

Shikun Li, Xiaobo Xia, Hansong Zhang, Yibing Zhan, Shiming Ge, and Tongliang Liu. Estimating noise transition matrix with label correlations for noisy multi-label learning. In *Advances in Neural Information Processing Systems*, 2022.

Xuefeng Li, Tongliang Liu, Bo Han, Gang Niu, and Masashi Sugiyama. Provably end-to-end label-noise learning without anchor points. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 6403–6413. PMLR, 2021.

Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(3):447–461, 2016.

Yang Liu and Hongyi Guo. Peer loss functions: Learning from noisy labels without knowing noise rates. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119, pages 6226–6236. PMLR, 2020.

Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Mach. Learn.*, 78(3):287–304, 2010.

Philip M. Long and Rocco A. Servedio. Consistency versus realizable H-consistency for multiclass classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval.* Cambridge University Press, 2008.

Andreas Maurer. A vector-contraction inequality for rademacher complexities. In *Algorithmic Learning Theory - 27th International Conference, ALT 2016*, volume 9925 of *Lecture Notes in Computer Science*, pages 3–17, 2016.

Colin McDiarmid. *On the method of bounded differences*, page 148–188. London Mathematical Society Lecture Note Series. Cambridge University Press, 1989.

Aditya Krishna Menon, Harikrishna Narasimhan, Shivani Agarwal, and Sanjay Chawla. On the statistical consistency of algorithms for binary classification under class imbalance. In *Proceedings* *of the 30th International Conference on Machine Learning, ICML 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 603–611. JMLR.org, 2013.

Aditya Krishna Menon, Brendan van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, volume 37, pages 125–134. JMLR.org, 2015.

Aditya Krishna Menon, Brendan van Rooyen, and Nagarajan Natarajan. Learning from binary labels with instance-dependent noise. *Mach. Learn.*, 107(8-10):1561–1595, 2018.

Aditya Krishna Menon, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Multilabel reductions: what is my loss optimising? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 10599–10610, 2019.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning.* Adaptive computation and machine learning. MIT Press, 2012.

Harikrishna Narasimhan, Rohit Vaish, and Shivani Agarwal. On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 1493–1501, 2014.

Harikrishna Narasimhan, Harish G. Ramaswamy, Aadirupa Saha, and Shivani Agarwal. Consistent multiclass algorithms for complex performance measures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2398–2407. JMLR.org, 2015.

Harikrishna Narasimhan, Harish G. Ramaswamy, Shiv Kumar Tavker, Drona Khurana, Praneeth Netrapalli, and Shivani Agarwal. Consistent multiclass algorithms for complex metrics and constraints. *CoRR*, abs/2210.09695, 2022.

Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pages 1196–1204, 2013.

Nagarajan Natarajan, Oluwasanmi Koyejo, Pradeep Ravikumar, and Inderjit S. Dhillon. Optimal classification with multivariate losses. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org, 2016.

Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Cost-sensitive learning with noisy labels. *J. Mach. Learn. Res.*, 18:155:1–155:33, 2017.

Chenri Ni, Nontawat Charoenphakdee, Junya Honda, and Masashi Sugiyama. On the calibration of multiclass classification with rejection. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 2582–2592, 2019.

Shameem Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. Optimizing f-measures by cost-sensitive classification. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 2123–2131, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Giorgio Patrini, Frank Nielsen, Richard Nock, and Marcello Carioni. Loss factorization, weakly supervised learning and label noise robustness. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 708–717. JMLR.org, 2016.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 2233–2241. IEEE Computer Society, 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

James Petterson and Tibério S. Caetano. Reverse multi-label learning. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010*, pages 1912–1920. Curran Associates, Inc., 2010.

James Petterson and Tibério S. Caetano. Submodular multi-label learning. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011*, pages 1512–1520, 2011.

Ignazio Pillai, Giorgio Fumera, and Fabio Roli. Designing multi-label classifiers that maximize F measures: State of the art. *Pattern Recognition*, 61:394–404, 2017.

Amir Rahimi, Amirreza Shaban, Ching-An Cheng, Richard Hartley, and Byron Boots. Intra order-preserving functions for calibration of multi-class neural networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

Harish G. Ramaswamy. *Design and Analysis of Consistent Algorithms for Multiclass Learning Problems*. PhD thesis, Indian Institute of Science, 2015.

Harish G. Ramaswamy and Shivani Agarwal. Classification calibration dimension for general multiclass losses. In *Neural Information Processing Systems*, 2012.

Harish G. Ramaswamy and Shivani Agarwal. Convex calibration dimension for multiclass loss matrices. *Journal of Machine Learning Research*, 17:14:1–14:45, 2016.

Harish G. Ramaswamy, Shivani Agarwal, and Ambuj Tewari. Convex calibrated surrogates for low-rank loss matrices with applications to subset ranking losses. In *Advances in Neural Information Processing Systems*, 2013.

Harish G. Ramaswamy, Balaji Srinivasan Babu, Shivani Agarwal, and Robert C. Williamson. On the consistency of output code based learning algorithms for multiclass learning problems. In *Proceedings of the 27th Conference on Learning Theory (COLT)*, pages 885–902, 2014.

Harish G. Ramaswamy, Ambuj Tewari, and Shivani Agarwal. Convex calibrated surrogates for hierarchical classification. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1852–1860, 2015.

Harish G. Ramaswamy, Ambuj Tewari, and Shivani Agarwal. Consistent algorithms for multiclass classification with an abstain option. *Electronic Journal of Statistics*, 12(1):530 – 554, 2018.

Mark D. Reid and Robert C. Williamson. Composite binary losses. *Journal of Machine Learning Research*, 11:2387–2422, 2010.

Clayton Scott. Calibrated asymmetric surrogate losses. *Electronic Journal of Statistics*, 6:958–992, 2012.

Clayton Scott. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015*, volume 38. JMLR.org, 2015.

Clayton Scott, Gilles Blanchard, and Gregory Handy. Classification with asymmetric label noise: Consistency and maximal denoising. In *COLT 2013 - The 26th Annual Conference on Learning Theory*, volume 30, pages 489–511. JMLR.org, 2013.

Song-Qing Shen, Bin-Bin Yang, and Wei Gao. AUC optimization with a reject option. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, pages 5684–5691. AAAI Press, 2020.

Cees Snoek, Marcel Worring, Jan C. van Gemert, Jan-Mark Geusebroek, and Arnold W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia.

In *Proceedings of the 14th ACM International Conference on Multimedia, 2006*, pages 421–430. ACM, 2006.

Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Trans. Neural Networks Learn. Syst.*, 34(11): 8135–8153, 2023.

Ingo Steinwart. How to compare different loss functions and their risks. *Constructive Approximation*, 26:225–287, 2007.

Guillaume Stempfel and Liva Ralaivola. Learning svms from sloppily labeled data. In *Artificial Neural Networks - ICANN 2009, 19th International Conference, Proceedings, Part I*, volume 5768 of *Lecture Notes in Computer Science*, pages 884–893. Springer, 2009.

Lijuan Sun, Songhe Feng, Tao Wang, Congyan Lang, and Yi Jin. Partial multi-label learning by low-rank and sparse decomposition. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 5016–5023. AAAI Press, 2019.

Shao-Hua Sun. Multi-digit mnist for few-shot learning. *GitHub repository, https://github.com/shaohua0116/MultiDigitMNIST*, 2019.

Yanmin Sun, Mohamed S. Kamel, and Yang Wang. Boosting for learning multiple classes with imbalanced class distribution. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, pages 592–602. IEEE Computer Society, 2006.

Benjamin Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003]*, pages 25–32. MIT Press, 2003.

Ambuj Tewari and Peter L. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8:1007–1025, 2007.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.

Brendan van Rooyen and Robert C. Williamson. A theory of learning with corrupted labels. *J. Mach. Learn. Res.*, 18:228:1–228:50, 2017.

Brendan van Rooyen, Aditya Krishna Menon, and Robert C. Williamson. Learning with symmetric label noise: The importance of being unhinged. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 10–18, 2015.

Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge J. Belongie.

Learning from noisy large-scale datasets with minimal supervision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 6575–6583. IEEE Computer Society, 2017.

Haobo Wang, Weiwei Liu, Yang Zhao, Chen Zhang, Tianlei Hu, and Gang Chen. Discriminative and correlative partial multi-label learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3691–3697. ijcai.org, 2019.

Hong Wang, Ashkan Rezaei, and Brian D. Ziebart. Adversarial structured prediction for multivariate measures. *CoRR*, abs/1712.07374, 2017.

Ruxin Wang, Tongliang Liu, and Dacheng Tao. Multiclass learning with partially corrupted labels. *IEEE Trans. Neural Networks Learn. Syst.*, 29(6):2568–2580, 2018.

Shuo Wang and Xin Yao. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Trans. Syst. Man Cybern. Part B*, 42(4):1119–1130, 2012.

Jason Weston and Chris Watkins. Support vector machines for multi-class pattern recognition. In *7th European Symposium On Artificial Neural Networks*, 1999.

Robert C. Williamson, Elodie Vernet, and Mark D. Reid. Composite multiclass losses. *J. Mach. Learn. Res.*, 17:223:1–223:52, 2016.

Guoqiang Wu and Jun Zhu. Multi-label classification: do hamming loss and subset accuracy really conflict with each other? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

Guoqiang Wu, Yingjie Tian, and Chunhua Zhang. A unified framework implementing linear binary relevance for multi-label learning. *Neurocomputing*, 289:86–100, 2018.

Guoqiang Wu, Chongxuan Li, Kun Xu, and Jun Zhu. Rethinking and reweighting the univariate losses for multi-label ranking: Consistency and generalization. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pages 14332–14344, 2021.

Xi-Zhu Wu and Zhi-Hua Zhou. A unified view of multi-label performance measures. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3780–3788, 2017.

Marek Wydmuch, Kalina Jasinska, Mikhail Kuznetsov, Róbert Busa-Fekete, and Krzysztof Dembczynski. A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 6358–6368, 2018.

Xiaobo Xia, Tongliang Liu, Nannan Wang, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? In *Advances in Neural Informa-*

tion Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, pages 6835–6846, 2019.

Ming-Kun Xie and Sheng-Jun Huang. Partial multi-label learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018*, pages 4302–4309. AAAI Press, 2018.

Ming-Kun Xie and Sheng-Jun Huang. Partial multi-label learning with noisy label identification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 6454–6461. AAAI Press, 2020.

Ming-Kun Xie and Sheng-Jun Huang. CCMN: A general framework for learning with class-conditional multi-label noise. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(1):154–166, 2023.

Forest Yang and Sanmi Koyejo. On the consistency of top-k surrogate losses. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 10727–10735. PMLR, 2020.

Yu Yao, Tongliang Liu, Bo Han, Mingming Gong, Jiankang Deng, Gang Niu, and Masashi Sugiyama. Dual T: reducing estimation error for transition matrix in label-noise learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

Nan Ye, Kian Ming Adam Chai, Wee Sun Lee, and Hai Leong Chieu. Optimizing f-measure: A tale of two approaches. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*. icml.cc / Omnipress, 2012.

Ming Yuan and Marten H. Wegkamp. Classification methods with reject option based on convex risk minimization. *J. Mach. Learn. Res.*, 11:111–130, 2010.

Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.

Mingyuan Zhang and Shivani Agarwal. Bayes consistency vs. h-consistency: The interplay between surrogate loss functions and the scoring function class. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

Mingyuan Zhang and Shivani Agarwal. Multiclass learning from noisy labels for non-decomposable performance measures. In *International Conference on Artificial Intelligence and Statistics 2024, AISTATS 2024*, volume 238 of *Proceedings of Machine Learning Research*, pages 2170–2178. PMLR, 2024.

Mingyuan Zhang, Harish Guruprasad Ramaswamy, and Shivani Agarwal. Convex calibrated surrogates for the multi-label f-measure. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 11246–11255. PMLR, 2020.

Mingyuan Zhang, Jane Lee, and Shivani Agarwal. Learning from noisy labels with no change to the training process. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12468–12478. PMLR, 2021.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32(1):56–134, 2004a.

Tong Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251, 2004b.

Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 8792–8802, 2018.

Wenting Zhao and Carla P. Gomes. Evaluating multi-label classifiers with noisy labels. *CoRR*, abs/2102.08427, 2021.