AN ABSTRACT OF THE DISSERTATION OF

Liping Liu for the degree of Doctor of Philosophy in Computer Science presented on June 10, 2016.

Title: Machine Learning Methods for Computational Sustainability

Abstract approved: ______________________________

Thomas G. Dietterich

Maintaining the sustainability of the earth's ecosystems has attracted much attention as these ecosystems are facing more and more pressure from human activities. Machine learning can play an important role in promoting sustainability as a large amount of data is being collected from ecosystems. There are at least three important and representative issues in the study of sustainability: detecting the presence of species, modeling the distribution of species, and protecting endangered species. For these three issues, this thesis selects three typical problems as the main focus and studies these problems with different machine learning techniques. Specifically, this thesis investigates the problem of detecting bird species from bird song recordings, the problem of modeling migrating birds at the population level, and the problem of designing a conservation area for an endangered species. First, this thesis models the problem of bird song classification as a weakly-supervised learning problem and develops a probabilistic classification model for the learning problem. The thesis also analyzes the learnability of the superset label learning problem to determine conditions under which one can learn a good classifier from the training data. Second, the thesis models bird migration with a probabilistic graphical model at the population level using a Collective Graphical Model (CGM). The thesis proposes a Gaussian approximation to significantly improve the inference efficiency of the model. Theoretical results show that the proposed Gaussian approximation is correct and can be calculated efficiently. Third, the thesis studies a typical reserve design problem with a novel formulation of transductive classification. Then the thesis solves the formulation with two optimization algorithms. The learning techniques in this thesis are general and can also be applied to many other machine learning problems.

Machine Learning Methods for Computational Sustainability

by

Liping Liu

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 10, 2016
Commencement June 2016

ACKNOWLEDGEMENTS

I have benefited from so many people during the course of my Ph.D. study. Let me try to include everyone in this acknowledgment.

Foremost, I would like to express my special gratitude to my advisor, Thomas G. Dietterich. As an exemplary advisor, he provides me effective guidance as well as the freedom to explore. He is supportive and cares about my career success. By working with him, I didn't feel that the six-year Ph.D. study is long.

I would like to thank all collaborators. These great researchers have contributed in various ways to the thesis. From them, I have also learned all kinds of research skills, such as analytical and writing skills. Especially, I would like to thank Xiaoli Fern and Raviv Raich, who have done excellent work on processing the bird acoustic data, which is the basis of the first chapter. I would also like to thank all other professors who have spent their precious time on my problems.

I feel grateful to have great colleagues in our research group and so many good friends in the EECS department. I have enjoyed our discussions and paper reading. Some of them have proofread the drafts of my papers.

I wish to acknowledge the support of the US National Science Foundation under grants No. 1125228, No. 1125228, and No. 1331932.

Last, I thank my family, especially my wife Yuanli Pei, for their love and understanding. My thesis would not be possible without my family's support.

CONTRIBUTION OF AUTHORS

Thanks to my collaborators, Daniel Sheldon, Nan Li, and Zhi-Hua Zhou. Daniel Sheldon has contributed to Chapter 4. Nan Li and Zhi-Hua Zhou have contributed to Chapter 5.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

This work is dedicated to a sustainable earth.

Chapter 1: Introduction

1.1 Introduction

With increasing human population and the consumption of resources, the human society is exerting more and more pressure on the natural environment, and ecosystems are becoming more and more vulnerable. In 1987, the well-known report, "Our Common Future", from the United Nations World Commission on Environment and Development (WCED) proposed the concept of *sustainable development*. The goal of sustainable development is to balance societal needs and the development of natural systems. A major concern of sustainable development is the protection and management of ecosystems [15].

Computational sustainability [28] is an emerging interdisciplinary research field that aims to apply computational methods to the study of sustainable development. One important reason for using computational methods is the large amount of data collected from ecosystems. In recent years, we have made revolutionary progress in collecting data by various means [46] from the ecosystem. One example is the National Ecological Observatory Network (NEON) funded by NSF, which aims to collect data from 20 ecological domains in the US territory for about 30 years. Massive ecological data makes possible the study of ecosystems in a data-intensive way. We can use computational methods to extract knowledge of the ecosystems from the data and make data-intensive policies to manage the ecosystems. Fortunately, the development of machine learning has already provided many techniques that can be adapted to problems in the study of sustainability.

There are at least three important issues in computational sustainability: data collection and interpretation, model fitting, and policy making. When the focus is on ecosystem management, the thesis substantiates the three issues as detecting species, modeling species distribution and behaviors, and policy making for ecosystem management. These three issues are logically connected. We need first to collect and interpret data to detect species. Then we can fit models to species data to understand mechanisms of ecosystems. With the data and knowledge of the ecosystems, we can create data-intensive policies to protect the endangered species. The three issues are introduced in detail as follows.

Detecting species in ecosystems is not an easy task, since species tend to hide themselves. Researchers have placed different kinds of sensors, including cameras [52] and sound recorders [12], into the natural environment to collect data. There are also data from the citizen science projects [65, 37], where people act as "novel" sensors and report their observations of species. One prominent project is the eBird project [65], which receives millions of checklists each year in recent years. There are several problems to address to collect data with high quality and to explain the data as activities of species. The first problem is the noise and bias in observations. For example, bird watchers certainly can not give a precise number of birds in an area, and the observed counts of birds are often affected by their observation process. We need to use some techniques to filter out the noise and calibrate the bias. The second problem is to extract species data from raw sensor data. We need data processing models to identify species from the raw data. There are also other problems, for example, data sparsity due to the large area of some ecosystems and data heterogeneity due to different data sources. All these problems call for novel techniques of data analysis.

Machine learning techniques give ecologists new tools for studying and understanding ecosystems. Ecological data can be analyzed to understand species behaviors and model species distribution. With a large amount of data, the study can be conducted at a very large scale. We often need to consider the correlation between species distribution and environmental factors. For example, the SpatioTemporal Exploratory Model (STEM) [26] fits the distribution of bird species at the continental level to environmental data. The study of species behaviors needs to overcome the difficulty of data incompleteness. Often we do not have full information about the underlying dynamics of the ecosystem, so we need to best exploit the pieces of evidence from the data and complete the story with our best guess.

Computational methods also play important roles in policy-making for the management of ecosystems. The main concern of policy-making is how to allocate limited resources and maximize the positive outcome of protecting or managing ecosystems. For example, in the problem of designing corridors for wildlife, Conrad et al. [19] study how to effectively use limited funds to best satisfy species' diverse needs of corridors in a geographical area. Fang et al. [24] and Carthy et al. [17] optimize patrol strategies against poaching and illegal logging. Taleghan et al. [67] studies the general problem of MDP planning, which can be applied to the protection of endangered species and the control of invasive species. One important research direction is planning the conservation area for endangered species. Like the problems mentioned above, the resources for conservation are limited. Unlike those problems, the success of a plan depends heavily on the uncertain species distribution. Full information about species distribution is often unknown, since it is very costly to survey the presence/absence of the species at each site in the planning area. Previous work often estimates species distribution first and then optimizes the conservation area against the estimated distribution. The optimization problem can be solved either by integer programming [30] or in a greedy manner [39].

In many cases, these three issues are often connected in the same application. For example, we often need to consider the problem of data noise when modeling species distribution. When we design a conservation area, we need to consider the distribution of species.

1.2 Contribution

This thesis focuses on the three important issues described and studies a representative problem for each issue.

The thesis first studies the problem of detecting bird species from birdsong recordings. In this problem, long recordings of birdsong are collected by unattended microphones in the H. J. Andrews Experimental Forest. The learning task is to identify the bird species responsible for syllables in these recordings [13]. Long recordings are first cut into short recordings, each of which has a length about 10 seconds and contains syllables of one or more species. It is very difficult for experts to give precise species identification for each syllable in the recording. Instead, experts label each 10-second recording with the set of bird species heard in that recording. At the same time, each syllable is segmented out from the recording [50], and a feature vector is extracted to describe the syllable [14]. Though the label/species of each syllable is unknown, it is certainly in the label set of the recording. The label set of the recording is the *label superset* of the true label of each syllable in the recording. One important step for identifying species is to learn a multiclass classifier that maps feature vectors to bird species. Label supersets provide a weak form of supervision for this learning task. Previous work by Briggs et al. [13] has modeled this learning problem with the formulation of instance annotation. This thesis models the problem as a superset label learning problem [20] and proposes a probabilistic learning model for the problem. This thesis also analyzes the learnability of the superset label learning problem.

The second part of the thesis studies the problem of modeling the bird migration behavior of single species from eBird checklists. To simplify the problem, the map of the Eastern US is first gridded into map cells, and birds are assumed to follow a time-inhomogeneous Markov chain model and transit among these map cells. The observations in this problem consists of noisy counts of the number of birds in all map cells at all time steps obtained from the eBird data. To study bird migration behaviors, the goal is to recover the model parameters of the Markov chain model from these observed noisy counts. Previous work by Sheldon et al. [60] tries to solve this problem by maximizing the likelihood of the data with the Expectation-Maximization (EM) algorithm. However, the E-step inference problem, inferring the mean of the counts of transitions among map cells, is very hard to solve. This inference problem is a typical inference problem of the Collective Graphical Model (CGM) [60] and is intractable in general. Previous approximate inference methods are either slow or inaccurate. This thesis proposes to approximate the CGM distribution by Gaussian distribution and then infer transition counts from this Gaussian approximation.

For the third issue, this thesis investigates a typical reserve design problem, in which we need to purchase land to construct a conservation area for an endangered species. It is assumed that the planning area consists of land parcels of the same area. The task is to select k land parcels and maximize the number of selected land parcels that contain the species of interest. Potentially two subproblems need to be addressed: modeling species distribution within the planning area and making the purchase decision. In many cases, the first subproblem is a learning problem. Ecologists can provide a *training set* of land parcels, each of which is labeled with the presence/absence of the species. Each land parcel, either in our planning area or in the training set, can be described by a set of features, such as elevation, precipitation, and vegetation coverage. The training set provides supervision information for estimating species distribution in the planning area. Traditional methods solve the two subproblems in two separate steps: (step 1) estimating species distribution with a supervised learning model and (step 2) ranking land parcels to obtain the top k. This thesis shows that it is unnecessary to train a ranking model as an intermediate step in solving the problem and proposes a new formulation, transductive precision@k, to combine the two subproblems into one learning problem. This thesis also develops a learning method to optimize the selection of the k land parcels directly.

With applications in sustainability as the motivations, this thesis reports research on general machine learning techniques. On one hand, such techniques can solve similar problems in other applications. For example, superset label learning is a general learning formulation and can model many other applicational problems, such as face tagging [20]. So the proposed learning algorithm and the theoretical analysis in this thesis can be directly applied to such problems. On the other hand, the machine learning research in this thesis also has theoretical significance. For example, the analysis of Gaussian Collective Graphical Model provides an alternative way of understanding the relationship between the precision matrix and the graph structure of a probabilistic graphical model.

1.3 Organization

The thesis is composed by four manuscripts published in three conferences related to machine learning. The first two chapters are two manuscripts on superset label learning (SLL). The first manuscript proposes a new learning model, Logistic Stick-Breaking Conditional Mixture Model (LSB-CMM), for the superset label learning problem and applies the new model to the bird species identification problem. The second manuscript makes a theoretical analysis of the learnability of the SLL problem. The third manuscript models bird migration at the population level with the Collective Graphical Model (CGM). The fourth manuscript studies a reserve design problem using the transductive top k formulation.

Manuscript 1: A Conditional Multinomial Mixture Model for Superset Label Learning

Li-Ping Liu, Thomas G. Dietterich

Advances in Neural Information Processing Systems 25
Edited by F. Pereira and C. J. C. Burges and L. Bottou and K. Q. Weinberger
Published by Curran Associates, Inc., 2012
Pages 548–556.

Chapter 2: A Conditional Multinomial Mixture Model for Superset Label Learning

Abstract

In the superset label learning problem (SLL), each training instance provides a *set* of candidate labels of which one is the true label of the instance. As in ordinary regression, the candidate label set is a noisy version of the true label. In this work, we solve the problem by maximizing the likelihood of the candidate label sets of training instances. We propose a probabilistic model, the Logistic Stick-Breaking Conditional Multinomial Model (LSB-CMM), to do the job. The LSB-CMM is derived from the logistic stick-breaking process. It first maps data points to mixture components and then assigns to each mixture component a label drawn from a component-specific multinomial distribution. The mixture components can capture underlying structure in the data, which is very useful when the model is weakly supervised. This advantage comes at little cost, since the model introduces few additional parameters. Experimental tests on several real-world problems with superset labels show results that are competitive or superior to the state of the art. The discovered underlying structures also provide improved explanations of the classification predictions.

2.1 Introduction

In supervised classification, the goal is to learn a classifier from a collection of training instances, where each instance has a unique class label. However, in many settings, it is difficult to obtain such precisely-labeled data. Fortunately, it is often possible to obtain a *set* of labels for each instance, where the correct label is one of the elements of the set.

For example, captions on pictures (in newspapers, facebook, etc.) typically identify all of the people in the picture but do not necessarily indicate which face belongs to each person. Imprecisely-labeled training examples can be created by detecting each face in the image and defining a label set containing all of the names mentioned in the caption. A similar case arises in bird song classification [13]. In this task, a field recording of multiple birds singing is divided into 10-second segments, and experts identify the species of all of the birds singing in each segment without localizing each species to a specific part of the spectrogram. These examples show that superset-labeled data are typically much cheaper to acquire than standard single-labeled data. If effective learning algorithms can be devised for superset-labeled data, then they would have wide application.

The superset label learning problem has been studied under two main formulations. In the multi-instance multi-label (MIML) formulation [78], the training data consist of pairs (B_i, Y_i), where $B_i = \{\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n_i}\}$ is a set of instances and Y_i is a set of labels. The assumption is that for every instance $\mathbf{x}_{i,j} \in B_i$, its true label $y_{i,j} \in Y_i$. The work of Jie et al. [32] and Briggs et al. [13] learn classifiers from such set-labeled bags.

In the superset label formulation (which has sometimes been confusingly called the "partial label" problem) [29, 33, 31, 51, 21, 20], each instance $\mathbf{x}_n$ has a candidate label set Y_n that contains the unknown true label y_n. This formulation ignores any bag structure and views each instance independently. It is more general than the MIML formulation, since any MIML problem can be converted to a superset label problem (with loss of the bag information). Furthermore, the superset label formulation is natural in many applications that do not involve bags of instances. For example, in some applications, annotators may be unsure of the correct label, so permitting them to provide a superset of the correct label avoids the risk of mislabeling. In this paper, we employ the superset label formulation. Other relevant work includes Nguyen et al. [51] and Cour et al. [20] who extend SVMs to handle superset labeled data.

In the superset label problem, the label set Y_n can be viewed as a corruption of the true label. The standard approach to learning with corrupted labels is to assume a generic noise process and incorporate it into the likelihood function. In standard supervised learning, it is common to assume that the observed label is sampled from a Bernoulli random variable whose most likely outcome is equal to the true label. In ordinary least-squares regression, the assumption is that the observed value is drawn from a Gaussian distribution whose mean is equal to the true value and whose variance is a constant σ^2. In the superset label problem, we will assume that the observed label set Y_n is drawn from a set-valued distribution $p(Y_n|y_n)$ that depends only on the true label. When computing the likelihood, this will allow us to treat the true label as a latent variable that can be marginalized away.

When the label information is imprecise, the learning algorithm has to depend more on underlying structure in the data. Indeed, many semi-supervised learning methods [80] model cluster structure of the training data explicitly or implicitly. This suggests that the underlying structure of the data should also play an important role in the superset label problem.

In this paper, we propose the Logistic Stick-Breaking Conditional Multinomial Model (LSB-

CMM) for the superset label learning problem. The model has two components: the mapping component and the coding component. Given an input $\mathbf{x}_n$, the mapping component maps $\mathbf{x}_n$ to a region k. Then the coding component generates the label according to a multinomial distribution associated with k. The mapping component is implemented by the Logistic Stick Breaking Process (LSBP) [56] whose Bernoulli probabilities are from discriminative functions. The mapping and coding components are optimized simultaneously with the variational EM algorithm.

LSB-CMM addresses the superset label problem in several aspects. First, the mapping component models the cluster structure with a set of regions. The fact that instances in the same region often have the same label is important for inferring the true label from noisy candidate label sets. Second, the regions do not directly correspond to classes. Instead, the number of regions is automatically determined by data, and it can be much larger than the number of classes. Third, the results of the LSB-CMM model can be more easily interpreted than the approaches based on SVMs [20, 13]. The regions provide information about how data are organized in the classification problem.

2.2 The Logistic Stick Breaking Conditional Multinomial Model

The superset label learning problem seeks to train a classifier $f : \mathcal{R}^d \mapsto \{1, \cdots, L\}$ on a given dataset $(\mathbf{x}, Y) = \{(\mathbf{x}_n, Y_n)\}_{n=1}^N$, where each instance $\mathbf{x}_n \in \mathcal{R}^d$ has a *candidate label set* $Y_n \subset \{1, \cdots, L\}$. The true labels $y = \{y_n\}_{n=1}^N$ are not directly observed. The only information is that the true label y_n of instance $\mathbf{x}_n$ is in the candidate set Y_n. The extra labels $\{l | l \neq y_n, l \in Y_n\}$ causing ambiguity will be called the *distractor labels*. For any test instance $(\mathbf{x}_t, y_t)$ drawn from the same distribution as $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, the trained classifier f should be able to map $\mathbf{x}_t$ to y_t with high probability. When $|Y_n| = 1$ for all n, the problem is a supervised classification problem. We require $|Y_n| < L$ for all n; that is, every candidate label set must provide at least some information about the true label of the instance.

2.2.1 The Model

As stated in the introduction, the candidate label set is a noisy version of the true label. To train a classifier, we first need a likelihood function $p(Y_n|\mathbf{x}_n)$. The key to our approach is to write the function as $p(Y_n|\mathbf{x}_n) = \sum_{y_n=1}^{L} p(Y_n|y_n)p(y_n|\mathbf{x}_n)$, where each term is the product of the underlying true classifier, $p(y_n|\mathbf{x}_n)$, and the noise model $p(Y_n|y_n)$. We then make the following assumption about the noise distribution:

Assumption: *All labels in the candidate label set Y_n have the same probability of generating Y_n, but no label outside of Y_n can generate Y_n*

$$p(Y_n|y_n = l) = \begin{cases} \lambda(Y_n) & \text{if } l \in Y_n \\ 0 & \text{if } l \notin Y_n \end{cases} . \tag{2.1}$$

This assumption enforces three constraints. First, the set of labels Y_n is conditionally independent of the input $\mathbf{x}_n$ given y_n. Second, labels that do not appear in Y_n have probability 0 of generating Y_n. Third, all of the labels in Y_n have equal probability of generating Y_n (symmetry). Note that these constraints do *not* imply that the training data are correctly labeled. That is, suppose that the most likely label for a particular input $\mathbf{x}_n$ is $y_n = l$. Because $p(y_n|\mathbf{x}_n)$ is a multinomial distribution, a different label $y_n = l'$ might be assigned to $\mathbf{x}_n$ by the labeling process. Then this label is further corrupted by adding distractor labels to produce Y_n. Hence, it could be that $l \notin Y_n$. In short, in this model, we have the usual "multinomial noise" in the labels which is then further compounded by "superset noise". The third constraint can be criticized for being simplistic; we believe it can be replaced with a learned noise model in future work.

Given (2.1), we can marginalize away y_n in the following optimization problem maximizing the likelihood of observed candidate labels.

$$\begin{aligned} f^* &= \arg\max_f \sum_{n=1}^{N} \log \sum_{y_n=1}^{L} p(y_n|\mathbf{x}_n; f) p(Y_n|y_n) \\ &= \arg\max_f \sum_{n=1}^{N} \log \sum_{y_n \in Y_n} p(y_n|\mathbf{x}_n; f) + \sum_{n=1}^{N} \log(\lambda(Y_n)). \end{aligned} \tag{2.2}$$

Under the conditional independence and symmetry assumptions, the last term does not depend on f and so can be ignored in the optimization. This result is consistent with the formulation in [33].

We propose the Logistic Stick-Breaking Conditional Multinomial Model to instantiate f (see Figure 2.1). In LSB-CMM, we introduce a set of K regions (mixture components) $\{1, \ldots, K\}$. LSB-CMM has two components. The mapping component maps each instance $\mathbf{x}_n$ to a region $z_n, z_n \in \{1, \ldots, K\}$. Then the coding component draws a label y_n from the multinomial distribution indexed by z_n with parameter θ_{z_n}. We denote the region indices of the training instances

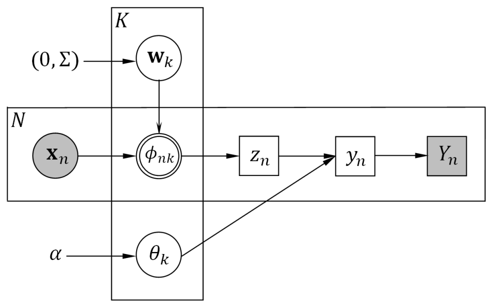

Figure 2.1: The LSB-CMM. Square nodes are discrete, circle nodes are continuous, and double-circle nodes are deterministic.

by $z = (z_n)_{n=1}^N$.

In the mapping component, we employ the Logistic Stick Breaking Process (LSBP) [56] to model the instance-region relationship. LSBP is a modification of the Dirichlet Process (DP) [68]. In LSBP, the sequence of Bernoulli probabilities are the outputs of a sequence of logistic functions instead of being random draws from a Beta distribution as in the Dirichlet process. The input to the k-th logistic function is the dot product of $\mathbf{x}_n$ and a learned weight vector $\mathbf{w}_k \in R^{d+1}$. (The added dimension corresponds to a zeroth feature fixed to be 1 to provide an intercept term.) To regularize these logistic functions, we posit that each $\mathbf{w}_k$ is drawn from a Gaussian distribution Normal$(0, \Sigma)$, where $\Sigma = \text{diag}(\infty, \sigma^2, \cdots, \sigma^2)$. This prior distribution regularizes all terms in $\mathbf{w}_k$ except the intercept. For each $\mathbf{x}_n$, a sequence of probabilities $\{v_{nk}\}_{k=1}^K$ is generated from logistic functions, where $v_{nk} = \text{expit}(\mathbf{w}_k^T \mathbf{x}_n)$ and $\text{expit}(u) = 1/(1 + \exp(-u))$ is the logistic function. We truncate k at K by setting $\mathbf{w}_K = (+\infty, 0, \cdots, 0)$ and thus $v_{nK} = 1$. Let $\mathbf{w}$ denote the collection of all K $\mathbf{w}_k$. Given the probabilities $v_{n1}, \ldots, v_{nK}$ computed from $\mathbf{x}_n$, we choose the region z_n according to a stick-breaking procedure:

$$p(z_n = k) = \phi_{nk} = v_{nk} \prod_{i=1}^{k-1} (1 - v_{ni}). \tag{2.3}$$

Here we stipulate that the product is 1 when $k = 1$. Let $\phi_n = (\phi_{n1}, \cdots, \phi_{nK})$ constitute the parameter of a multinomial distribution. Then z_n is drawn from this distribution.

In the coding component of LSB-CMM, we first draw K L-dimensional multinomial probabilities $\theta = \{\theta_k\}_{k=1}^K$ from the prior Dirichlet distribution with parameter α. Then, for each instance $\mathbf{x}_n$ with mixture z_n, its label y_n is drawn from the multinomial distribution with θ_{z_n}. In the traditional multi-class problem, y_n is observed. However, in the SLL problem y_n is not observed and Y_n is generated from y_n.

The whole generative process of the model is summarized below:

$$\begin{aligned}
\mathbf{w}_k &\sim \text{Normal}(0, \Sigma), 1 \leq k \leq K-1, \;\; \mathbf{w}_K = (+\infty, 0, \cdots, 0) && (2.4)\\
z_n &\sim \text{Mult}(\phi_n), \quad \phi_{nk} = \text{expit}(\mathbf{w}_k^T \mathbf{x}_n) \prod_{i=1}^{k-1} (1 - \text{expit}(\mathbf{w}_i^T \mathbf{x}_n)) && (2.5)\\
\theta_k &\sim \text{Dirichlet}(\alpha) && (2.6)\\
y_n &\sim \text{Mult}(\theta_{z_n}) && (2.7)\\
Y_n &\sim \text{Dist1}(y_n) \quad \text{(Dist1 is some distribution satisfying (2.1))} && (2.8)
\end{aligned}$$

As shown in (2.2), the model needs to maximize the likelihood that each y_n is in Y_n. After incorporating the priors, we can write the penalized maximum likelihood objective as

$$\max LL = \sum_{n=1}^{N} \log \left(\sum_{y_n \in Y_n} p(y_n | \mathbf{x}_n, \mathbf{w}, \alpha) \right) + \log(p(\mathbf{w}|0, \Sigma)). \tag{2.9}$$

This cannot be solved directly, so we apply variational EM [9].

2.2.2 Variational EM

The hidden variables in the model are y, z, and θ. For these hidden variables, we introduce the variational distribution $q(y, z, \theta | \hat{\phi}, \hat{\alpha})$, where $\hat{\phi} = \{\hat{\phi}_n\}_{n=1}^N$ and $\hat{\alpha} = \{\hat{\alpha}_k\}_{k=1}^K$ are the parameters. Then we factorize q as

$$q(z, y, \theta | \hat{\phi}, \hat{\alpha}) = \prod_{n=1}^{N} q(z_n, y_n | \hat{\phi}_n) \prod_{k=1}^{K} q(\theta_k | \hat{\alpha}_k), \tag{2.10}$$

where $\hat{\phi}_n$ is a $K \times L$ matrix and $q(z_n, y_n | \hat{\phi}_n)$ is a multinomial distribution in which $p(z_n = k, y_n = l) = \hat{\phi}_{nkl}$. This distribution is constrained by the candidate label set: if a label $l \notin Y_n$, then $\hat{\phi}_{nkl} = 0$ for any value of k. The distribution $q(\theta_k | \hat{\alpha}_k)$ is a Dirichlet distribution with parameter $\hat{\alpha}_k$.

After we set the distribution $q(z, y, \theta)$, our variational EM follows standard methods. The detailed derivation can be found in Appendix A. Here we only show the final updating step with some analysis.

In the E step, the parameters of variational distribution are updated as (2.11) and (2.12).

$$\hat{\phi}_{nkl} \propto \begin{cases} \phi_{nk} \, \exp\left(E_{q(\theta_k|\hat{\alpha}_k)}\left[\log(\theta_{kl})\right]\right), & \text{if } l \in Y_n \\ 0, & \text{if } l \notin Y_n \end{cases}, \tag{2.11}$$

$$\hat{\alpha}_k = \alpha + \sum_{n=1}^{N} \hat{\phi}_{nkl} \,. \tag{2.12}$$

The update of $\hat{\phi}_n$ in (2.11) indicates the key difference between the LSB-CMM model and traditional clustering models. The formation of regions is directed by both instance similarities and class labels. If the instance $\mathbf{x}_n$ wants to join region k (i.e., $\sum_l \hat{\phi}_{nkl}$ is large), then it must be similar to $\mathbf{w}_k$ as well as to instances in that region in order to make ϕ_{nk} large. Simultaneously, its candidate labels must fit the "label flavor" of region k, where the "label flavor" means region k prefers labels having large values in $\hat{\alpha}_k$. The update of $\hat{\alpha}$ in (2.12) can be interpreted as having each instance $\mathbf{x}_n$ vote for the label l for region k with weight $\hat{\phi}_{nkl}$.

In the M step, we need to solve the mximization problem in (2.13) for each $\mathbf{w}_k, 1 \leq k \leq K-1$. Note that $\mathbf{w}_K$ is fixed. Each $\mathbf{w}_k$ can be optimized separately. The optimization problem is similar to the problem of logistic regression and is also a concave maximization problem, which can be solved by any gradient-based method, such as BFGS.

$$\max_{\mathbf{w}_k} \; -\frac{1}{2}\mathbf{w}_k^T \Sigma^{-1} \mathbf{w}_k + \sum_{n=1}^{N} \left[\hat{\phi}_{nk} \log(\text{expit}(\mathbf{w}_k^T \mathbf{x}_n)) + \hat{\psi}_{nk} \log(1 - \text{expit}(\mathbf{w}_k^T \mathbf{x}_n))\right], \tag{2.13}$$

where $\hat{\phi}_{nk} = \sum_{l=1}^{L} \hat{\phi}_{nkl}$ and $\hat{\psi}_{nk} = \sum_{j=k+1}^{K} \hat{\phi}_{nj}$. Intuitively, the variable $\hat{\phi}_{nk}$ is the probability that instance $\mathbf{x}_n$ belongs to region k, and $\hat{\psi}_{nk}$ is the probability that $\mathbf{x}_n$ belongs to region $\{k+1, \cdots, K\}$. Therefore, the optimal $\mathbf{w}_k$ discriminates instances in region k against instances in regions $\geq k$.

2.2.3 Prediction

For a test instance $\mathbf{x}_t$, we predict the label with maximum posterior probability. The test instance can be mapped to a region with $\mathbf{w}$, but the coding matrix θ is marginalized out in the EM. We use the variational distribution $p(\theta_k|\hat{\alpha}_k)$ as the prior of each θ_k and integrate out all θ_k-s. Given a test point $\mathbf{x}_t$, the prediction is the label l that maximizes the probability $p(y_t = l|\mathbf{x}_t, \mathbf{w}, \hat{\alpha})$ calculated as (2.14). The detailed derivation is also in Appendix A.

$$p(y_t = l|\mathbf{x}_t, \mathbf{w}, \hat{\alpha}) = \sum_{k=1}^{K} \phi_{tk} \frac{\hat{\alpha}_{kl}}{\sum_l \hat{\alpha}_{kl}} , \tag{2.14}$$

where $\phi_{tk} = \Big(\text{expit}(\mathbf{w}_k^T \mathbf{x}_t) \prod_{i=1}^{k-1}(1 - \text{expit}(\mathbf{w}_i^T \mathbf{x}_t))\Big)$. The test instance goes to region k with probability ϕ_{tk}, and its label is decided by the votes ($\hat{\alpha}_k$) in that region.

2.2.4 Complexity Analysis and Practical Issues

In the E step, for each region k, the algorithm iterates over all candidate labels of all instances, so the complexity is $O(NKL)$. In the M step, the algorithm solves $K-1$ separate optimization problems. Suppose each optimization problem takes $O(VNd)$ time, where V is the number of BFGS iterations. Then the complexity is $O(KVNd)$. Since V is usually larger than L, the overall complexity of one EM iteration is $O(KVNd)$. Suppose the EM steps converge within m iterations, where m is usually less than 50, then the overall complexity is $O(mKVNd)$. The space complexity is $O(NK)$, since we only store the matrix $\sum_{l=1}^{L} \hat{\phi}_{nkl}$ and the matrix $\hat{\alpha}$.

In prediction, the mapping phase requires $O(Kd)$ time to multiply $\mathbf{w}$ and the test instance. After the stick breaking process, which takes $O(K)$ calculations, the coding phase requires $O(KL)$ calculation. Thus the overall time complexity is $O(K \max\{d, L\})$. Hence, the prediction time is comparable to that of logistic regression.

There are several practical issues that affect the performance of the model. **Initialization:** From the model design, we can expect that instances in the same region have the same label. Therefore, it is reasonable to initialize $\hat{\alpha}$ to have each region prefer only one label, that is, each $\hat{\alpha}_k$ has one element with large value and all others with small values. We initialize ϕ to $\phi_{nk} = \frac{1}{K}$, so that all regions have equal probability to be chosen at the start. Initialization of these two variables is enough to begin the EM iterations. We find that such initialization works well for our model and generally is better than random initialization. **Calculation of $E_{q(\theta_k|\hat{\alpha}_k)}[\log(\theta_{kl})]$ in (2.11):** Although it has a closed-form solution, we encountered numerical issues, so we calculate it via Monte Carlo sampling. This does not change complexity analysis above, since the training is dominated by M step. **Priors:** We found that using a non-informative prior for Dirichlet(α) worked best. From (2.12) and (2.14), we can see that when θ is marginalized, the distribution is non-informative when α is set to small values. We use $\alpha = 0.05$ in our experiments.

2.3 Experiments

In this section, we describe the results of several experiments we conducted to study the behavior of our proposed model. First, we experiment with a toy problem to show that our algorithm can solve problems with linearly-inseparable classes. Second, we perform controlled experiments on three synthetic datasets to study the robustness of LSB-CMM with respect to the degree of ambiguity of the label sets. Third, we experiment with three real-world datasets.

LSB-CMM Model: The LSB-CMM model has three parameters K, σ^2, α. We find that the model is insensitive to K if it is sufficiently large. We set $K = 10$ for the toy problems and $K = 5L$ for other problems. α is set to 0.05 for all experiments. When the data is standardized, the regularization parameter $\sigma^2 = 1$ generally gives good results, so σ^2 is set to 1 in all superset label tasks.

Baselines: We compared the LSB-CMM model with three state-of-the-art methods. **Supervised SVM:** the SVM is always trained with the true labels. Its performance can be viewed as an upper bound on the performance of any SSL algorithm. LIBSVM [18] with RBF kernel was run to construct a multi-class classifier in *one-vs-one* mode. One third of the training data was used to tune the C parameter and the RBF kernel parameter γ. **CLPL:** CLPL [20] is a linear model that encourages large average scores of candidate labels. The model is insensitive to the C parameter, so we set the C value to 1000 (the default value in their code). **SIM:** SIM [13] minimizes the ranking loss of instances in a bag. In controlled experiments and in one of the real-world problems, we could not make the comparison to LSB-CMM because of the lack of bag information. The λ parameter is set to 10^{-8} based on authors' recommendation.

2.3.1 A Toy Problems

In this experiment, we generate a linearly-inseparable SLL problem. The data has two dimensions and six clusters drawn from six normal distributions with means at the corners of a hexagon. We assign a label to each cluster so that the problem is linearly-inseparable (see (2.2)). In the first task, we give the model the true labels. In the second task, we add a distractor label for two thirds of all instances (gray data points in the figure). The distractor label is randomly chosen from the two labels other than the true label. The decision boundaries found by LSB-CMM

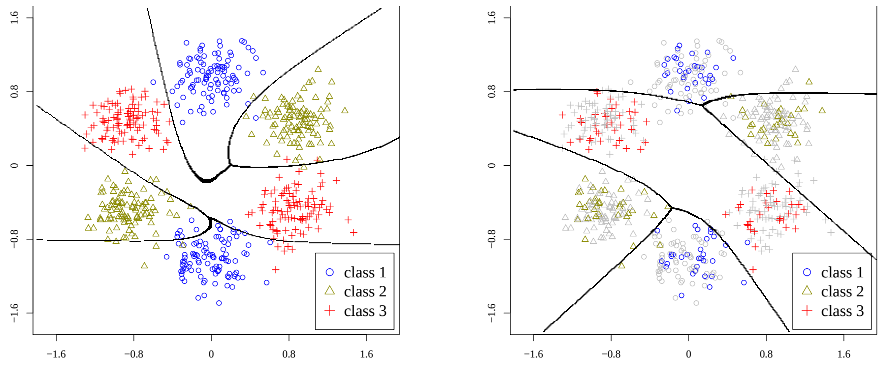

Figure 2.2: Decision boundaries of LSB-CMM on a linearly-inseparable problem. Left: all data points have true labels. Right: labels of gray data points are corrupted.

in both tasks are shown in (2.2)). We can see that LSB-CMM can successfully give nonlinear decision boundaries for this problem. After injecting distractor labels, LSB-CMM still recovers the boundaries between classes. There is minor change of the boundary at the edge of the cluster, while the main part of each cluster is classified correctly.

2.3.2 Controlled Experiments

We conducted controlled experiments on three UCI [42] datasets: $\{segment$ (2310 instances, 7 classes), $pendigits$ (10992 instances, 10 classes), and $usps$ (9298 instances, 10 classes)$\}$. Ten-fold cross validation is performed on all three datasets. For each training instance, we add distractor labels with controlled probability. As in [20], we use p, q, and ε to control the ambiguity level of candidate label sets. The roles and values of these three variables are as follows: p is the probability that an instance has distractor labels ($p = 1$ for all controlled experiments); $q \in \{1, 2, 3, 4\}$ is the number of distractor labels; and $\varepsilon \in \{0.3, 0.7, 0.9, 0.95\}$ is the maximum probability that a distractor label co-occurs with the true label [20], also called the ambiguity degree.

We have two settings for these three variables. In the first setting, we hold $q = 1$ and vary ε, that is, for each label l, we choose a specific label $l' \neq l$ as the (unique) distractor label with probability ε or choose any other label with probability $1 - \varepsilon$. In the extreme case when $\varepsilon = 1$, l' and l always co-occur, and they cannot be distinguished by any classifier. In the second setting, we vary q and pick distractor labels randomly for each candidate label set.

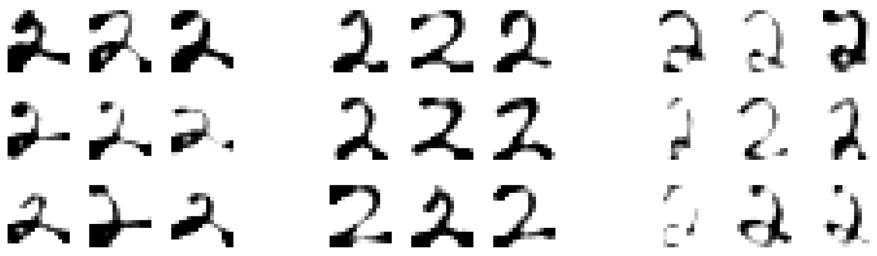

Figure 2.3: Three regions learned by the model on $usps$

The results are shown in Figure (2.4). Our LSB-CMM model significantly outperforms the CLPL approach. As the number of distractor labels increases, performance of both methods goes down, but not too much. When the true label is combined with different distractor labels, the disambiguation is easy. The co-occurring distractor labels provide much less disambiguation. This explains why large ambiguity degree hurts the performance of both methods. The small dataset ($segment$) suffers even more from large ambiguity degree, because there are fewer data points that can “break” the strong correlation between the true label and the distractors.

To explore why the LSB-CMM model has good performance, we investigated the regions learned by the model. Recall that ϕ_{nk} is the probability that $\mathbf{x}_n$ is sent to region k. In each region k, the representative instances have large values of ϕ_{nk}. We examined all ϕ_{nk} from the model trained on the $usps$ dataset with 3 random distractor labels. For each region k, we selected the 9 most representative instances. Figure (2.3) shows representative instances for three regions. These are all from class “2” but are written in different styles. This shows that the LSB-CMM model can discover the sub-classes in the data. In some applications, the whole class is not easy to discriminate from other classes, but sometimes each sub-class can be easily identified. In such cases, LSB-CMM will be very useful and can improve performance.

Explanation of the results via regions can also give better understanding of the learned classifier. In order to analyze the performance of the classifier learned from data with either superset labels or fully observed labels, one traditional method is to compute the confusion matrix. While the confusion matrix can only tell the relationships between classes, the mixture analysis can indicate precisely which subclass of a class are confused with which subclasses of other classes. The regions can also help the user identify and define new classes as refinements of existing ones.

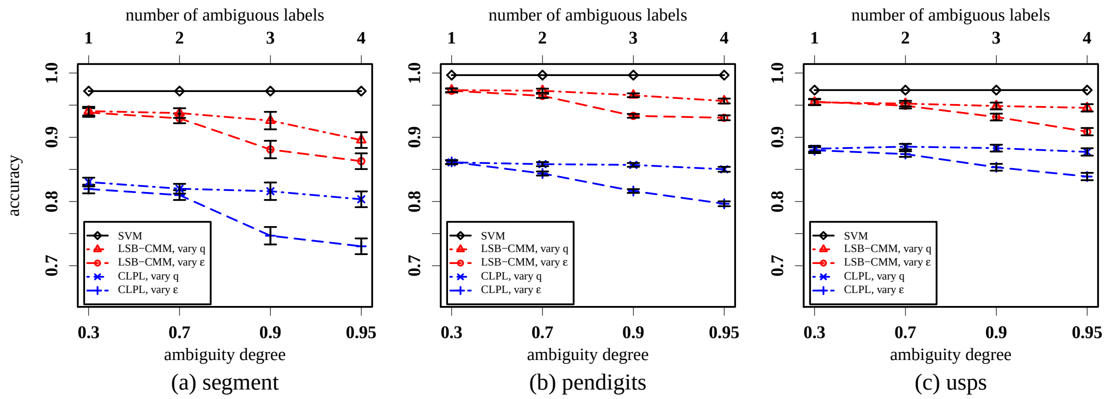

Figure 2.4: Classification performance on synthetic data (red: LSB-CMM; blue: CLPL). The dot-dash line is for different q values (number of distractor labels) as shown on the top x-axis. The dashed line is for different ε (ambiguity degree) values as shown on the bottom x-axis.

2.3.3 Real-World Problems

We apply our model on three real-world problems. **1) BirdSong dataset** [13]: This contains 548 10-second bird song recordings. Each recording contains 1-40 syllables. In total there are 4998 syllables. Each syllable is described by 38 features. The labels of each recording are the bird species that were singing during that 10-second period, and these species become candidate labels set of each syllable in the recording. **2) MSRCv2 dataset**: This dataset contains 591 images with 23 classes. The ground truth segmentations (regions with labels) are given. The labels of all segmentations in an image are treated as candidate labels for each segmentation. Each segmentation is described by 48-dimensional gradient and color histograms. **3) Lost dataset** [20]: This dataset contains 1122 faces, and each face has the true label and a set of candidate labels. Each face is described by 108 PCA components. Since the bag information (i.e., which faces are in the same scene) is missing, SIM is not compared to our model on this dataset. We run 10-fold cross validation on these three datasets. The BirdSong and MSRCv2 datasets are split by recordings/images, and the Lost dataset is split by faces.

The classification accuracies are shown in Table (2.1). Accuracies of the three superset label learning algorithms are compared using the paired t-test at the 95% confidence level. Values statistically indistinguishable from the best performance are shown in bold. Our LSB-CMM model out-performs the other two methods on the BirdSong database, and its performance is comparable to SIM on the MSRCv2 dataset and to CLPL on the Lost dataset. It should be noted

Table 2.1: Classification Accuracies for Superset Label Problems

	LSB-CMM	SIM	CLPL	SVM
BirdSong	**0.715**(**0.042**)	0.589(0.035)	0.637(0.034)	0.790(0.027)
MSRCv2	**0.459**(**0.032**)	**0.454**(**0.043**)	0.411(0.044)	0.673(0.043)
Lost	**0.703**(**0.058**)	-	**0.710**(**0.045**)	0.817(0.038)

that the input features are very coarse, which means that the cluster structure of the data is not well maintained. The relatively low performance of the SVM confirms this. If the instances were more precisely described by finer features, one would expect our model to perform better in those cases as well.

2.4 Conclusions

This paper introduced the Logistic Stick-Breaking Conditional Multinomial Model to address the superset label learning problem. The mixture representation allows LSB-CMM to discover cluster structure that has predictive power for the superset labels in the training data. Hence, if two labels co-occur, LSB-CMM is not forced to choose one of them to assign to the training example but instead can create a region that maps to both of them. Nonetheless, each region does predict from a multinomial, so the model still ultimately seeks to predict a single label. Our experiments show that the performance of the model is either better than or comparable to state-of-the-art methods.

Manuscript 2: Learnability of the Superset Label Learning Problem

Li-Ping Liu, Thomas G. Dietterich

Proceedings of the 31st International Conference on Machine Learning (ICML-14)
Edited by T. Jebara and E. P. Xing
Published by JMLR Workshop and Conference Proceedings, 2014
Pages 1629-1637

Chapter 3: Learnability of the Superset Label Learning Problem

Abstract

In the Superset Label Learning (SLL) problem, weak supervision is provided in the form of a *superset* of labels that contains the true label. If the classifier predicts a label outside of the superset, it commits a *superset error*. Most existing SLL algorithms learn a multiclass classifier by minimizing the superset error. However, only limited theoretical analysis has been dedicated to this approach. In this paper, we analyze Empirical Risk Minimizing learners that use the superset error as the empirical risk measure. SLL data can arise either in the form of independent instances or as multiple-instance bags. For both scenarios, we give the conditions for ERM learnability and sample complexity for the realizable case.

3.1 Introduction

In multiclass supervised learning, the task is to learn a classifier that maps an object to one of several candidate classes. When each training example is labeled with one label, many successful multiclass learning methods can solve this problem [3, 48]. In some applications, however, we cannot obtain training examples of this kind. Instead, for each instance we are given a *set* of possible labels. The set is guaranteed to contain the true label, and it also possibly contains one or more *distractor labels*.

Despite these distractor labels, we still wish learn a multiclass classifier that has low error when measured according to the traditional 0/1 misclassification loss. This learning problem has been given several names, including the "multiple label problem", the "partial label problem" and the "superset label learning problem" [33, 51, 20, 43]. In this paper, we adopt the last of these.

Several learning algorithms for the superset label learning problem have shown good experimental results. Most of these algorithms seek an hypothesis that explicitly minimizes superset errors on the training instances (possibly including a regularization penalty). An exception is the ECOC-based algorithm proposed by Zhang et al. [77]. Though it does not minimize superset errors explicitly, this algorithm generally predicts a class label of an training instance from its superset and thus are also minimizing superset errors. In this paper, we define the superset error as the empirical risk in the SLL problem, and we analyze the performance of learners that minimize the empirical risk (ERM learners). We only investigate the realizable case where the true multiclass classifier is in the hypothesis space.

The key to SLL learnability is that any hypothesis with non-zero classification error must have a significant chance of making superset errors. This in turn depends on the size and distribution of the label supersets. Small supersets, for example, are more informative than large ones. Precisely speaking, a sufficient condition for learnability is that the classification error can be bounded by the superset error.

The SLL problem arises in two settings. In the first setting, which we call SSL-I, instances are independent of each other, and the superset is selected independently for each instance. The second setting, which we call SLL-B, arises from the multi-instance multi-label learning (MIML) problem [78, 79], where the training data are given in the form of MIML bags.

For SLL-I, Cour et al. [20] proposed the concept of *ambiguity degree*. This bounds the probability that a specific distractor label appears in the superset of a given instance. When the ambiguity degree is less than 1, they give a relationship between the classification error and the superset error. With the same condition, we show that the sample complexity of SLL-I with ambiguity degree zero matches the complexity of multiclass classification.

For SLL-B, the training data is given in the form of independent MIML bags. Each MIML bag consists of a set of instances and a set of labels (the "bag label"). Each instance has exactly one (unknown) label, and the bag label is exactly the union of the labels of all of the instances. (See [79] for discussion of other ways in which MIML bags can be generated.) The learner observes the bag label but not the labels of the individual instances.

It is interesting to note that several previous papers test their algorithms on synthetic data corresponding to the SSL-I scenario, but then apply them to a real application corresponding to the SSL-B setting.

To show learnability, we convert the SLL-B problem to a binary classification problem with the general condition that the classification error can be bounded by the superset error times a multiplicative constant. We then provide a concrete condition for learnability: for any pair of class labels, they must not always co-occur on a bag label. That is, there must be non-zero probability of observing a bag that contains instances of only one of the two labels. Given enough training data, we can verify with high confidence whether this condition holds.

The success of weakly supervised learning depends on the degree of correlation between the supervision information and the classification error. We show that superset learning exhibits a strong correlation between these two because a superset error is always caused by a classification error. Our study of the SLL problem can be seen as a first step toward the analysis of more general weakly-supervised learning problems.

3.2 Related Work

Different superset label learning algorithms have been proposed by previous work [33, 51, 20, 43, 77]. All these algorithms employ some loss to minimize superset errors on the training set. Cour et al. [20] conducted some theoretical analysis of the problem. They proposed the concept of "ambiguity degree" and established a relationship between superset error and classification errors. They also gave a generalization bound for their algorithm. In their analysis, they assume instances are independent of each other.

Sample complexity analysis of multiclass learning provides the basis of our analysis of SLL problem with independent instances. The Natarajan dimension [49] is an important instrument for characterizing the capacity of multiclass hypothesis spaces. Ben-David et al. [8] and Daniely et al. [22] give sample complexity bounds in terms of this dimension.

The MIML framework was proposed by Zhou et al. [78], and the instance annotation problem is raised by Briggs et al. [13]. Though the algorithm for instance annotation explicitly uses bag information, it is still covered by our analysis of the SLL-B problem. There is some theoretical analysis of multi-instance learning [10, 45, 58], but we only know of one paper [74] on the learnability of the MIML problem. In that work, no assumption is made for the distribution of instances within a bag, but the labels on a bag must satisfy some correlation conditions. In our setting, the distribution of bag labels can be arbitrary, but we assume that the instances in the bag are independent of each other given their labels.

3.3 Superset Label Learning Problem with Independent Instances (SLL-I)

Let $\mathcal{X}$ be the instance space and $\mathcal{Y} = \{1, 2, \ldots, L\}$ be the finite label space. The superset space $\mathcal{S}$ is the powerset of $\mathcal{Y}$ without the empty set: $\mathcal{S} = 2^{\mathcal{Y}} - \{\emptyset\}$. A "complete" instance $(x, y, s) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{S}$ is composed of its features x, its true label y, and the label superset s. We decompose $\mathcal{D}$ into the standard multiclass distribution $\mathcal{D}^{xy}$ defined on $\mathcal{X} \times \mathcal{Y}$ and the label set conditional distribution $\mathcal{D}^{s}(x, y)$ defined over $\mathcal{S}$ given (x, y). We assume that $Pr_{s \sim \mathcal{D}^{s}(x,y)}(y \in s) = 1$, that is, the true label is always in the label superset. Other labels in the superset will be called *distractor labels*. Let $\mu(\cdot)$ denote the probability measure of a set; its subscript indicates the distribution. Denote a sample of instances by $\mathbf{z} = \big\{(x_i, y_i, s_i)\big\}_{i=1}^n$, where each instance is sampled from $\mathcal{D}$ independently. The size of the sample is always n. Although the true label is included in the training set in our notation, it is not visible to the learner. Let $\mathbb{I}(\cdot)$ denote the indicator function, which has the value 1 when its argument is true and 0 otherwise.

The hypothesis space is denoted by $\mathcal{H}$, and each $h \in \mathcal{H} : \mathcal{X} \to \mathcal{Y}$ is a multiclass classifier. The *expected classification error* of h is defined as

$$Err_{\mathcal{D}}(h) = E_{(x,y,s)\sim\mathcal{D}}\mathbb{I}(h(x) \neq y). \tag{3.1}$$

We use H_ϵ to denote the set of hypotheses with error at least ϵ, $H_\epsilon = \{h \in \mathcal{H} : Err_{\mathcal{D}}(h) \geq \epsilon\}$. The superset error is defined as the event that the predicted label is not in the superset: $h(x) \notin s$. The *expected superset error* and the *average superset error* on set $\mathbf{z}$ are defined as

$$\begin{aligned} Err^s_{\mathcal{D}}(h) &= E_{(x,y,s)\sim\mathcal{D}}\mathbb{I}(h(x) \notin s) && (3.2)\\ Err^s_{\mathbf{z}}(h) &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(h(x_i) \notin s_i) && (3.3)\end{aligned}$$

It is easy to see that the expected superset error is always no greater than the expected classification error.

For conciseness we often omit the word “expected” or “average” when referring these errors defined above. The meaning should be clear from how the error is calculated.

An Empirical Risk Minimizing (ERM) learner $\mathcal{A}$ for $\mathcal{H}$ is a function, $\mathcal{A} : \cup_{n=0}^{\infty}(\mathcal{X} \times \mathcal{S})^n \mapsto \mathcal{H}$. We define the *empirical risk* as the average superset error on the training set. The ERM learner for hypothesis space $\mathcal{H}$ always returns an hypothesis $h \in \mathcal{H}$ with minimum superset error for training set $\mathbf{z}$.

$$\mathcal{A}(\mathbf{z}) = \arg\min_{h\in\mathcal{H}} Err^s_{\mathbf{z}}(h)$$

Since the learning algorithm $\mathcal{A}$ can only observe the superset label, this definition of ERM is different from that of multiclass classification. In the realizable case, there exists $h_0 \in \mathcal{H}$ such that $Err_{\mathcal{D}}(h_0) = 0$.

3.3.1 Small ambiguity degree condition

On a training set with label supersets, an hypothesis will not be rejected by an ERM learner as long as its predictions are contained in the superset labels of the training instances. If a distractor label always co-occurs with one of the true labels under distribution $\mathcal{D}^s$, there will be no information to discriminate the true label from the distractor. On the contrary, if for any instance all labels except the true label have non-zero probability of being missing from the superset, then the learner always has some probability of rejecting an hypothesis if it predicts this instance incorrectly. The *ambiguity degree*, proposed by Cour et al. [20], is defined as

$$\gamma = \sup_{\substack{(x,y)\in\mathcal{X}\times\mathcal{Y},\ \ell\in\mathcal{Y}\ : \\ p(x,y)>0,\ \ell\neq y}} Pr_{s\sim\mathcal{D}^s(x,y)}(\ell \in s). \tag{3.4}$$

This is the maximum probability that some particular distractor label ℓ co-occurs with the true label y. If $\gamma = 0$, then with probability one there are no distractors. If $\gamma = 1$, then there exists at least one pair y and ℓ that always co-occur. If a problem exhibits ambiguity degree γ, then a classification error made on any instance will be detected (i.e., the prediction lies outside of the label superset) with probability at least $1-\gamma$:

$$Pr(h(x) \notin s | h(x) \neq y, x, y) \geq 1 - \gamma.$$

If an SSL-I problem exhibits $\gamma < 1$, then we say that it satisfies the *small ambiguity degree condition*. We prove that this is sufficient for ERM learnability of the SSL-I problem.

Theorem 3.3.1 *Suppose an SLL-I problem has ambiguity degree $\gamma, 0 \leq \gamma < 1$. Let $\theta = \log\frac{2}{1+\gamma}$, and suppose the Natarajan dimension of the hypothesis space $\mathcal{H}$ is $d_{\mathcal{H}}$. Define*

$$n_0(\mathcal{H}, \epsilon, \delta) = \frac{4}{\theta\epsilon}\left(d_{\mathcal{H}}\left(\log(4d_{\mathcal{H}}) + 2\log L + \log\frac{1}{\theta\epsilon}\right) + \log\frac{1}{\delta} + 1\right), \tag{3.5}$$

then when $n > n_0(\mathcal{H}, \epsilon, \delta)$, $Err_{\mathcal{D}}(\mathcal{A}(\mathbf{z})) < \epsilon$ with probability $1-\delta$.

We follow the method of proving learnability of binary classification [5] to prove this theorem. Let $R_{n,\epsilon}$ be the set of all n-samples for which there exists an ϵ-bad hypothesis h with zero empirical risk:

$$R_{n,\epsilon} = \{\mathbf{z} \in (\mathcal{X} \times \mathcal{Y} \times \mathcal{S})^n : \exists h \in H_\epsilon, Err^s_{\mathbf{z}}(h) = 0\}. \tag{3.6}$$

Essentially we need to show that $Pr(R_{n,\epsilon}) \leq \delta$.

The proof is composed of the following two lemmas.

Lemma 3.3.2 *We introduce a test set $\mathbf{z}'$ of size n with each instance drawn independently from distribution $\mathcal{D}$, and define the set $S_{n,\epsilon}$ to be the event that there exists a hypothesis in H_ϵ that makes no superset errors on training set $\mathbf{z}$ but makes at least $\frac{\epsilon}{2}$ classification errors on test set $\mathbf{z}'$.*

$$S_{n,\epsilon} = \Big\{(\mathbf{z}, \mathbf{z}') \in (\mathcal{X} \times \mathcal{Y} \times \mathcal{S})^{2n} : \exists h \in H_\epsilon, Err^s_{\mathbf{z}}(h) = 0, Err_{\mathbf{z}'}(h) \geq \frac{\epsilon}{2}\Big\}. \tag{3.7}$$

Then $Pr\big((\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon}\big) \geq \frac{1}{2} Pr(\mathbf{z} \in R_{n,\epsilon})$ when $n > \frac{8}{\epsilon}$.

Proof 3.3.3 *This lemma is used in many learnability proofs. Here we only give a proof sketch. $S_{n,\epsilon}$ is a subevent of $R_{n,\epsilon}$. We can apply the chain rule of probability to write $Pr\big((\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon}\big) = Pr\big((\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon} | \mathbf{z} \in R_{n,\epsilon}\big) Pr\big(\mathbf{z} \in R_{n,\epsilon}\big)$. Let $H(\mathbf{z}) = \{h \in \mathcal{H} : Err^s_{\mathbf{z}}(h) = 0\}$ be the set of hypotheses with zero empirical risk on the training sample. Then we have*

$$\begin{aligned} Pr\left((\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon} \middle| \mathbf{z} \in R_{n,\epsilon}\right) &= Pr\left(\left\{\exists h \in H_\epsilon \cap H(\mathbf{z}), Err_{\mathbf{z}'}(h) \geq \frac{\epsilon}{2}\right\} \middle| \mathbf{z} \in R_{n,\epsilon}\right) \\ &\geq Pr\left(\{h \in H_\epsilon \cap H(\mathbf{z}), Err_{\mathbf{z}'}(h) \geq \frac{\epsilon}{2}\} \middle| \mathbf{z} \in R_{n,\epsilon}\right) \end{aligned}$$

In the last line, h is a particular hypothesis in the intersection. Since h has error at least ϵ, we can bound the probability via the Chernoff bound. When $n > \frac{8}{\epsilon}$, we obtain $Pr\big((\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon} \mid \mathbf{z} \in R_{n,\epsilon}\big) > \frac{1}{2}$ which completes the proof. □

With Lemma 3.3.2, we can bound the probability of $R_{n,\epsilon}$ by bounding the probability of $S_{n,\epsilon}$. We will do this using the technique of swapping training/test instance pairs, which is used in various proofs of learnability.

In the following proof, the sets $\mathbf{z}$ and $\mathbf{z}'$ are expanded to $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ and $(\mathbf{x}', \mathbf{y}', \mathbf{s}')$ respectively when necessary. Let the training and test instances form n training/test pairs by arbitrary pairing. The two instances in each pair are respectively from the training set and the test set, and these two instances are both indexed by the pair index, which is indicated by a subscript. Define a group G of swaps with size $|G| = 2^n$. A swap $\sigma \in G$ has an index set $J_\sigma \subseteq \{1, \ldots, n\}$, and it swaps the training and test instances in the pairs indexed by J_σ. We write σ as a superscript to indicate the result of applying σ to the training and test sets, that is, $\sigma(\mathbf{z}, \mathbf{z}') = (\mathbf{z}^\sigma, \mathbf{z}'^\sigma)$.

Lemma 3.3.4 *If the hypothesis space $\mathcal{H}$ has Natarajan dimension $d_\mathcal{H}$ and $\gamma < 1$, then*

$$Pr(S_{n,\epsilon}) \leq (2n)^{d_\mathcal{H}} L^{2d_\mathcal{H}} \exp\left(-\frac{n\theta\epsilon}{2}\right)$$

Proof 3.3.5 *Since the swap does not change the measure of $(\mathbf{z}, \mathbf{z}')$,*

$$\begin{aligned} 2^n Pr\left((\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon}\right) &= \sum_{\sigma \in G} E\left[Pr\left((\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon} \middle| \mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}'\right)\right] \\ &= \sum_{\sigma \in G} E\left[Pr\left(\sigma(\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon} \middle| \mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}'\right)\right] \\ &= E\left[\sum_{\sigma \in G} Pr\left(\sigma(\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon} \middle| \mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}'\right)\right] \end{aligned} \tag{3.8}$$

The expectations are taken with respect to $(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}')$. The probability in the expectation comes from the randomness of $(\mathbf{s}, \mathbf{s}')$ given $(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}')$.

Let $\mathcal{H}|(\mathbf{x}, \mathbf{x}')$ be the set of hypothesis making different classifications for $(\mathbf{x}, \mathbf{x}')$. Define set $S^h_{n,\epsilon}$ for each hypothesis $h \in \mathcal{H}$ as

$$S^h_{n,\epsilon} = \left\{(\mathbf{z}, \mathbf{z}') : Err^s_\mathbf{z}(h) = 0, Err_{\mathbf{z}'}(h) \geq \frac{\epsilon}{2}\right\}$$

By the union bound, we have

$$\sum_{\sigma \in G} Pr\left(\sigma(\mathbf{z}, \mathbf{z}') \in S_{n,\epsilon} \middle| \mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}'\right) \leq \sum_{h \in \mathcal{H}|(\mathbf{x}, \mathbf{x}')} \sum_{\sigma \in G} Pr\left(\sigma(\mathbf{z}, \mathbf{z}') \in S^h_{n,\epsilon} \middle| \mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}'\right) \tag{3.9}$$

By the work of Natarajan [49], $\left|\mathcal{H}|(\mathbf{z}, \mathbf{z}')\right| \leq (2n)^{d_\mathcal{H}} L^{2d_\mathcal{H}}$. The only work left is to bound $\sum_{\sigma \in G} Pr\big(\sigma(\mathbf{z}, \mathbf{z}') \in S^h_{n,\epsilon} | \mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}'\big)$, and this part is our contribution.

Here is our strategy. We first fix $(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}')$ and σ and bound $Pr\big(\sigma(\mathbf{z}, \mathbf{z}') \in S^h_{n,\epsilon} \mid \mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}'\big)$ with the ambiguity degree assumption. Then we find an upper bound of the summation over σ.

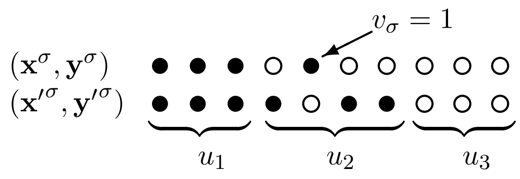
$v_\sigma = 1$
$(\mathbf{x}^\sigma, \mathbf{y}^\sigma)$
$(\mathbf{x}'^\sigma, \mathbf{y}'^\sigma)$
u_1
u_2
u_3

Figure 3.1: A situation of $(\mathbf{x}^\sigma, \mathbf{y}^\sigma, \mathbf{x}'^\sigma, \mathbf{y}'^\sigma)$. Black dots represent instances misclassified by h and circles represent instances correctly classified by h.

Start by expanding the condition in $S^h_{n,\epsilon}$,

$$\begin{aligned} Pr\big(\sigma(\mathbf{z}, \mathbf{z}') \in S^h_{n,\epsilon} \mid \mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}'\big) &= \mathbb{I}\big(Err_{\mathbf{z}'^\sigma}(h) \geq \frac{\epsilon}{2}\big) \cdot Pr\big(h(\mathbf{x}^\sigma_i) \in \mathbf{s}^\sigma_i, 1 \leq i \leq n | \mathbf{x}^\sigma, \mathbf{y}^\sigma\big) \\ &= \mathbb{I}\big(Err_{\mathbf{z}'^\sigma}(h) \geq \frac{\epsilon}{2}\big) \prod_{i=1}^{n} Pr\big(h(\mathbf{x}^\sigma_i) \in \mathbf{s}^\sigma_i | \mathbf{x}^\sigma_i, \mathbf{y}^\sigma_i\big). \end{aligned}$$

For a pair of training/test sets $(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}')$, let u_1, u_2 and u_3 represent the number of pairs for which h classifies both incorrectly, one incorrectly, and both correctly. Let v_σ, $0 \leq v_\sigma \leq u_2$, be the number of wrongly-predicted instances swapped into the training set $(\mathbf{x}^\sigma, \mathbf{y}^\sigma)$. One such situation is shown in Figure 3.1. There are $u_1 + u_2 - v_\sigma$ wrongly-predicted instances in the test set. The error condition $Err_{\mathbf{z}'^\sigma}(h) \geq \frac{\epsilon}{2}$ is equivalent to $u_1 + u_2 - v_\sigma \geq \frac{\epsilon}{2}n$, which always indicates $u_1 + u_2 \geq \frac{\epsilon}{2}n$. So we have $\mathbb{I}\big(Err_{\mathbf{z}'^\sigma}(h) \geq \frac{\epsilon}{2}\big) \leq \mathbb{I}\big(u_1 + u_2 \geq \frac{\epsilon}{2}n\big)$.

There are $u_1 + v_\sigma$ wrongly-predicted instances in the training set. Since the true label is in the superset with probability one, while the wrong label appears in the superset with probability no greater than γ by (3.4), we have $\prod_{i=1}^{n} Pr\big(h(\mathbf{x}^\sigma_i) \in \mathbf{s}^\sigma_i | \mathbf{x}^\sigma_i\big) \leq \gamma^{u_1+v_\sigma}$.

Now for a single swap σ, we have the bound

$$Pr\big(\sigma(\mathbf{z}, \mathbf{z}') \in S^h_{n,\epsilon} \mid \mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}'\big) \leq \mathbb{I}\big(u_1 + u_2 \geq \frac{\epsilon}{2}n\big)\gamma^{u_1+v_\sigma} \tag{3.10}$$

Let us sum up (3.10) over σ. Any swap σ can freely switch instances in $u_1 + u_3$ without changing the bound in (3.10) and choose from the u_2 pairs v_σ to switch. For each value $0 \leq j \leq$

u_2, there are $2^{u_1+u_3}\binom{u_2}{j}$ swaps that have $v_\sigma = j$. Therefore,

$$\begin{aligned}\sum_{\sigma\in G}\mathbb{I}\big(u_1+u_2\geq\frac{\epsilon}{2}n\big)\gamma^{u_1+v_\sigma} &\leq \mathbb{I}\big(u_1+u_2\geq\frac{\epsilon}{2}n\big)\ 2^{u_1+u_3}\sum_{j=0}^{u_2}\binom{u_2}{j}\gamma^{u_1+j}\\ &= \mathbb{I}\big(u_1+u_2\geq\frac{\epsilon}{2}n\big)\ 2^{n-u_2}\gamma^{u_1}(1+\gamma)^{u_2}\\ &= \mathbb{I}\big(u_1+u_2\geq\frac{\epsilon}{2}n\big)\ 2^{n}\gamma^{u_1}\left(\frac{1+\gamma}{2}\right)^{u_2}\end{aligned}$$

When $(\mathbf{x},\mathbf{y},\mathbf{x}',\mathbf{y}')$ and h make $u_1 = 0$ and $u_2 = \frac{\epsilon}{2}n$, the right side reaches its maximum $2^n\left(\frac{1+\gamma}{2}\right)^{n\epsilon/2}$, which is $2^n e^{-n\theta\epsilon/2}$ with the definition of θ in Theorem 3.3.1. Applying this to (3.9) and (3.8), completes the proof. □

Proof 3.3.6 (Proof of Theorem 3.3.1.) *By combining the results of the two lemmas, we have $P(R_{n,\epsilon}) \leq 2^{(d_\mathcal{H}+1)}n^{d_\mathcal{H}}L^{2d_\mathcal{H}}\exp(-\frac{n\theta\epsilon}{2})$. Bound this with δ on a log scale to obtain*

$$(d_\mathcal{H}+1)\log 2 + d_\mathcal{H}\log n + 2d_\mathcal{H}\log L - \frac{\theta\epsilon n}{2} \leq \log\delta$$

By bounding $\log n$ with $(\log(\frac{4d_\mathcal{H}}{\theta\epsilon})-1)+\frac{\theta\epsilon}{4d_\mathcal{H}}n$, we get a linear form for n. Then we solve for n to obtain the result. □

Remark Theorem 3.3.1 includes the multiclass problem as a special case. When $\gamma = 0$ and $\theta = \log 2$, we get the same sample complexity as the multiclass classification problem (See Daniely et al. [22], Theorem 6.).

The distribution of label supersets actually interacts with the hypothesis space in the analysis of learnability. With the small ambiguity degree condition, we only require that $\mathcal{H}$ has finite Natarajan dimension to ensure the ERM learnability. However, if the condition does not hold, we may need a more restricted hypothesis space to get a learnable problem. Suppose a label ℓ_2 is always a distractor of another label ℓ_1. Whenever the true label is ℓ_1, the label superset always contains ℓ_2; otherwise the superset contains only the true label. Such a distribution of superset labels certainly does not satisfy the small ambiguity degree condition. Meanwhile, if no hypothesis in $\mathcal{H}$ classifies ℓ_1 as ℓ_2, then the problem is still learnable by an ERM learner. This example suggests that the small ambiguity degree condition is not a necessary condition for ERM learnability.

3.3.2 A general condition for learnability of SLL-I

We can obtain a more general condition for ERM learnability by constructing a binary classification task from the superset label learning task. Two key issues need special attention in the construction: how to relate the classification errors of the two tasks, and how to specify the hypothesis space of the binary classification task and obtain its VC-dimension.

We first construct a binary classification problem from the SLL-I problem. Given an instance (x, s), the binary classifier needs to predict whether the label of x is outside of s, that is, to predict the value of $\mathbb{I}(y_x \notin s)$. If (x, s) is from the distribution $\mathcal{D}$, then "0" is always the correct prediction. Let $\mathcal{F}_{\mathcal{H}}$ be the space of these binary classifiers. $\mathcal{F}_{\mathcal{H}}$ is induced from $\mathcal{H}$ as follows:

$$\mathcal{F}_{\mathcal{H}} = \{f_h : \ f_h(x, s) = \mathbb{I}(h(x) \notin s), h \in \mathcal{H}\}.$$

Though the inducing relation is a surjection from $\mathcal{H}$ to $\mathcal{F}_{\mathcal{H}}$, we assume the subscript h in the notation f_h can be any $h \in \mathcal{H}$ inducing f_h. The binary classification error of f_h is the superset error of h in the SLL-I problem.

Denote the ERM learner of the binary classification problem by $\mathcal{A}^s$. The learner $\mathcal{A}^s$ actually calls $\mathcal{A}$, which returns an hypothesis $h^\star$ and then $\mathcal{A}^s$ returns the induced hypothesis $f_{h^\star}$. In the realizable case, both $h^\star$ and $f_{h^\star}$ have zero training error on their respective classification tasks.

A sufficient condition for learnability of an SLL-I problem is that any hypothesis with non-zero expected classification error will cause superset errors with large probability. Let η be defined by

$$\eta = \inf_{h \in \mathcal{H}: Err_{\mathcal{D}}(h) > 0} \frac{Err^s_{\mathcal{D}}(h)}{Err_{\mathcal{D}}(h)}. \tag{3.11}$$

Define the *tied error condition* as $\eta > 0$. Then we can bound the multiclass classification error by bounding the superset error when this condition holds.

We need to find the VC-dimension of $\mathcal{F}_{\mathcal{H}}$ first. It can be bounded by the Natarajan dimension of $\mathcal{H}$.

Lemma 3.3.7 *Denote the VC-dimension of $\mathcal{F}_{\mathcal{H}}$ as $d_{\mathcal{F}}$ and the Natarajan dimension of $\mathcal{H}$ is $d_{\mathcal{H}}$, then*

$$d_{\mathcal{F}} \quad < \quad 4\, d_{\mathcal{H}}(\log d_{\mathcal{H}} + 2 \log L)$$

Proof 3.3.8 *There is a set of $d_{\mathcal{F}}$ instances that can be shattered by $\mathcal{F}_{\mathcal{H}}$—that is, there are functions in $\mathcal{F}_{\mathcal{H}}$ that implement each of the $2^{d_{\mathcal{F}}}$ different ways of classifying these $d_{\mathcal{F}}$ instances. Any two different binary classifications must be the result of two different multiclass predictions for these $d_{\mathcal{F}}$ instances. According to Natarajan's original result [49] on Natarajan dimension, there are at most $d_{\mathcal{F}}^{d_{\mathcal{H}}} L^{2d_{\mathcal{H}}}$ different multiclass classifications on these instances. Therefore,*

$$2^{d_{\mathcal{F}}} \leq d_{\mathcal{F}}^{d_{\mathcal{H}}} L^{2d_{\mathcal{H}}}.$$

Taking the logarithm of both sides gives us

$$d_{\mathcal{F}} \log 2 \quad \leq \quad d_{\mathcal{H}} \log d_{\mathcal{F}} + 2d_{\mathcal{H}} \log L.$$

By bounding $\log d_{\mathcal{F}}$ above by $\log d_{\mathcal{H}} + \frac{d_{\mathcal{F}}}{ed_{\mathcal{H}}}$, we get

$$d_{\mathcal{F}} \log 2 \quad \leq \quad d_{\mathcal{H}} \log d_{\mathcal{H}} + \frac{d_{\mathcal{F}}}{e} + 2d_{\mathcal{H}} \log L.$$

By observing that $(\log 2 - e^{-1})^{-1} < 4$, we reach the result. $\square$

Now we can bound the multiclass classification error by bounding the superset error.

Theorem 3.3.9 *Assume $\eta > 0$ and $\mathcal{H}$ has a finite Natarajan dimension $d_{\mathcal{H}}$, then $\mathcal{A}$ returns an hypothesis with error less than ϵ with probability at least $1-\delta$ when the training set has size $n > n_1(\mathcal{H}, \delta, \epsilon)$, which is defined as*

$$n_1(\mathcal{H}, \delta, \epsilon) = \frac{4}{\eta\epsilon}\left(4d_{\mathcal{H}}(\log d_{\mathcal{H}} + 2\log L)\log\left(\frac{12}{\eta\epsilon}\right) + \log\left(\frac{2}{\delta}\right)\right). \tag{3.12}$$

Proof 3.3.10 *Learner $\mathcal{A}$ returns hypothesis $h^\star \in \mathcal{H}$ with $Err^s_{\mathcal{D}}(h^\star) = 0$. The corresponding binary hypothesis $f_{h^\star} \in \mathcal{F}_{\mathcal{H}}$ makes no superset error on the training data.*

By Lemma (3.3.7), $n_1(\mathcal{H}, \delta, \epsilon) \geq n_1^s(\mathcal{F}_{\mathcal{H}}, \delta, \eta\epsilon)$, where

$$n_1^s(\mathcal{H}, \delta, \eta\epsilon) = \frac{4}{\eta\epsilon}\left(d_{\mathcal{F}} \log\left(\frac{12}{\eta\epsilon}\right) + \log\left(\frac{2}{\delta}\right)\right).$$

When $n > n_1(\mathcal{H}, \delta, \epsilon) \geq n_1^s(\mathcal{F}_{\mathcal{H}}, \delta, \eta\epsilon)$, $Err^s_{\mathcal{D}}(f_{h^\star}) < \eta\epsilon$ with probability at least $1-\delta$ by Theorem 8.4.1 in Anthony et al. [5]. Therefore, we have $Err_{\mathcal{D}}(h^\star) < \epsilon$ with probability at least $1-\delta$. $\square$

The necessary condition for the learnability of the SLL problem is that no hypothesis have non-zero multiclass classification error but zero superset error. Otherwise, no training data can reject such an incorrect hypothesis.

Theorem 3.3.11 *If there exists an hypothesis $h \in \mathcal{H}$ such that $Err_{\mathcal{D}}(h) > 0$ and $Err^{s}_{\mathcal{D}}(h) = 0$, then the SLL-I problem is not learnable by an ERM learner.*

The gap between the general sufficient condition and the necessary condition is small.

Let's go back and check the small ambiguity degree condition against the tied error condition. The condition $\gamma < 1$ indicates $Pr(h(x) \notin s|x, y) \geq (1-\gamma)Pr(h(x) \neq y|x, y)$. By taking the expectation of both sides, we obtain the tied error condition $\eta \geq 1 - \gamma > 0$. This also shows that the tied error condition is more general. Though the tied error condition is less practical, it can be used as a guideline to find more sufficient conditions.

3.4 Superset Label Learning Problem with Bagged Training Data (SLL-B)

The second type of superset label learning problem arises in multi-instance multilabel learning (MIML) [78]. The data are given in the form of i.i.d. bags. Each bag contains multiple instances, which are generally not independent of each other. The bag label set consists of the labels of these instances. In this form of superset label learning problem, the bag label set provides the label superset for each instance in the bag. An ERM learning algorithm for SSL-I can naturally be applied here regardless of the dependency between instances. In this section, we will show learnability of this SSI-B problem.

We assume that each bag contains r instances, where r is fixed as a constant. The space of labeled instances is still $\mathcal{X} \times \mathcal{Y}$. The space of bag label sets is also $\mathcal{S}$. The space of bags is $\mathcal{B} = (\mathcal{X} \times \mathcal{Y})^r \times \mathcal{S}$. Denote a bag of instances as $B = (X, Y, S)$, where $(X, Y) \in (\mathcal{X} \times \mathcal{Y})^r$ and $S \in \mathcal{S}$. Note that although (X, Y) is written as a vector of instances for notational convenience, the order of these instances is not essential to any conclusion in this work. We assume that each bag $B = (X, Y, S)$ is sampled from the distribution $\mathcal{D}^B$. The label set S consists of labels in Y in the MIML setting, but the following analysis still applies when S contains extra labels. The learner can only observe (X, S) during training, whereas the learned hypothesis is tested on (X, Y) during testing. The boldface $\mathbf{B}$ always denotes a set with m independent bags drawn from the distribution $\mathcal{D}^B$.

The hypothesis space is still denoted by $\mathcal{H}$. The expected classification error of hypothesis $h \in \mathcal{H}$ on bagged data is defined as

$$Err_{\mathcal{D}^B}(h) \quad = \quad E_{B \sim \mathcal{D}^B}\Big[\frac{1}{r}\sum_{i=1}^{r}\mathbb{I}(h(X_i) \neq Y_i)\Big] \tag{3.13}$$

The expected superset error and the average superset error on the set $\mathbf{B}$ are defined as

$$Err^s_{\mathcal{D}^B}(h) \quad = \quad E_{B \sim \mathcal{D}^B}\Big[\frac{1}{r}\sum_{i=1}^{r}\mathbb{I}(h(X_i) \notin S_i)\Big] \tag{3.14}$$

$$Err^s_{\mathbf{B}}(h) \quad = \quad \frac{1}{mr}\sum_{B \in \mathbf{B}}\sum_{i=1}^{r}\mathbb{I}(h(X_i) \notin S_i). \tag{3.15}$$

The ERM learner $\mathcal{A}$ for hypothesis space $\mathcal{H}$ returns the hypothesis with minimum average superset error.

3.4.1 The general condition for learnability of SLL-B

Using a technique similar to that employed in the last section, we convert the SLL-B problem into a binary classification problem over bags. By bounding the error on the binary classification task, we can bound the multiclass classification error of the hypothesis returned by $\mathcal{A}$. Let $\mathcal{G}_\mathcal{H}$ be the hypothesis space for binary classification of bags. $\mathcal{G}_\mathcal{H}$ is induced from $\mathcal{H}$ as follows:

$$\mathcal{G}_\mathcal{H} = \{g_h : g_h(X, S) = \max_{1 \le i \le r} \mathbb{I}(h(X_i) \notin S_i), h \in \mathcal{H}\}.$$

Every hypothesis $g_h \in \mathcal{G}_\mathcal{H}$ is a binary classifier for bags that predicts whether any instance in the bag has its label outside of the bag label set. Since 0 is the correct classification for every bag from the distribution $\mathcal{D}^B$, we define the expected and empirical *bag error* of g_h as

$$Err^B_{\mathcal{D}^B}(g_h) \quad = \quad E_{B \sim \mathcal{D}^B} \max_{1 \le i \le r} \mathbb{I}(h(X_i) \notin S_i) \tag{3.16}$$

$$Err^B_{\mathbf{B}}(g_h) \quad = \quad \frac{1}{m}\sum_{B \in \mathbf{B}} \max_{1 \le i \le r} \mathbb{I}(h(X_i) \notin S_i). \tag{3.17}$$

It is easy to check the following relation between $Err^{B}_{\mathcal{D}^B}(g_h)$ and $Err^{s}_{\mathcal{D}^B}(h)$.

$$Err^{s}_{\mathcal{D}^B}(h) \leq Err^{B}_{\mathcal{D}^B}(g_h) \leq r Err^{s}_{\mathcal{D}^B}(h). \tag{3.18}$$

Denote by $\mathcal{A}^B$ the ERM learner for this binary classification problem. In the realizable case, if the hypothesis $g_{h^\star}$ is returned by $\mathcal{A}^B$, then $g_{h^\star}$ has no binary classification error on the training bags, and $h^\star$ makes no superset error on any instance in training data.

To bound the error for the binary classification problem, we need to bound the VC-dimension of $\mathcal{G}_{\mathcal{H}}$.

Lemma 3.4.1 *Denote the VC-dimension of $\mathcal{G}_{\mathcal{H}}$ by $d_{\mathcal{G}}$ and the Natarajan dimension of $\mathcal{H}$ by $d_{\mathcal{H}}$, then*

$$d_{\mathcal{G}} \quad < \quad 4\, d_{\mathcal{H}}(\log d_{\mathcal{H}} + \log r + 2\log L). \tag{3.19}$$

The proof of this lemma is almost the same as Lemma 3.3.7. The only difference is that different classifications of $d_{\mathcal{G}}$ bags are caused by different classifications of $rd_{\mathcal{G}}$ instances.

The tied error condition for the SLL-B problem is $\lambda > 0$,

$$\lambda = \inf_{h \in \mathcal{H}: Err_{\mathcal{D}}(h) > 0} \frac{Err^{s}_{\mathcal{D}^B}(h)}{Err_{\mathcal{D}^B}(h)}. \tag{3.20}$$

With the tied error condition, we can give the sample complexity of learning a multiclass classifier with MIML bags.

Theorem 3.4.2 *Suppose the tied error condition holds, $\lambda > 0$, and assume $\mathcal{H}$ has finite Natarajan dimension $d_{\mathcal{H}}$, then $\mathcal{A}(\mathbf{B})$ returns an hypothesis with error less than ϵ with probability at least $1-\delta$ when the training set $\mathbf{B}$ has size m, $m > m_0(\mathcal{H}, \delta, \epsilon)$,*

$$m_0(\mathcal{H}, \delta, \epsilon) = \frac{4}{\lambda\epsilon}\left(4 d_{\mathcal{H}} \log(d_{\mathcal{H}} r L^2) \log\left(\frac{12}{\lambda\epsilon}\right) + \log\left(\frac{2}{\delta}\right)\right) \tag{3.21}$$

Proof 3.4.3 *With the tied error condition and the relation in (3.18), we have $Err_{\mathcal{D}^B}(h) \leq \frac{1}{\lambda} Err^{B}_{\mathcal{D}^B}(g_h)$. The hypothesis $h^\star$ returned by $\mathcal{A}$ has zero superset error on the training bags, thus $g_{h^\star}$ has zero bag error. When $m > m_0$, we have $Err^{B}_{\mathcal{D}^B}(g_h) < \lambda\epsilon$ with probability at least $1-\delta$ by Theorem 8.4.1 in Anthony et al. [5]. Hence, we have $Err_{\mathcal{D}^B}(h) < \epsilon$ with probability at least $1-\delta$.* $\square$

3.4.2 The no co-occurring label condition

We now provide a more concrete sufficient condition for learnability with a principle similar to the ambiguity degree. It generally states that any two labels do not always co-occur in bag label sets.

First we make an assumption about the data distribution $\mathcal{D}^B$.

Assumption 3.4.4 (Conditional independence assumption)

$$Pr(X|Y) = \prod_{i=1}^{r} Pr(X_i|Y_i)$$

This assumption states that the covariates of an instance are determined by its label only. With this assumption, a bag is sampled in the following way. Instance labels Y of the bag are first drawn from the marginal distribution $\mathcal{D}^Y$ of $\mathcal{D}^B$, then for each $Y_i, 1 \leq i \leq r$, X_i is independently sampled from the conditional distribution $\mathcal{D}^x(Y_i)$ derived from $\mathcal{D}^B$.

Remark We can compare our assumption of bag distribution with assumptions in previous work. Blum et al. [10] assume that all instances in a bag are independently sampled from the instance distribution. As stated by Sabato et al. [58], this assumption is too strict for many applications. In their work as well as Wang et al. [74], the instances in a bag are assumed to be dependent, and they point out that a further assumption is needed to get a low error classifier. In our assumption above, we also assume dependency among instances in the same bag. The dependency only comes from the instance labels. This assumption is roughly consistent with human descriptions of the world. For example, in the description "a tall person stands by a red vehicle", the two labels "person" and "vehicle" capture most of the correlation, while the detailed descriptions of the person and the vehicle are typically much less correlated.

The distribution D^B of bags can be decomposed into $\mathcal{D}^Y$ and $\mathcal{D}^x(\ell), \ell \in \mathcal{Y}$. Here $\mathcal{D}^Y$ is a distribution over $\mathcal{Y}^r$. For each class $\ell \in \mathcal{Y}$, $\mathcal{D}^x(\ell)$ is a distribution over the instance space $\mathcal{X}$. The distribution of X is collectively denoted as $\mathcal{D}^X(Y)$.

With Assumption 3.4.4, we propose the *no co-occurring label condition* in the following theorem.

Theorem 3.4.5 *Define*

$$\alpha = \inf_{(\ell,\ell')\in I} E_{(X,Y,S)\sim\mathcal{D}^B}[\mathbb{I}(\ell \in S)\mathbb{I}(\ell' \notin S)]$$

where I is the index set $\{(\ell,\ell') : \ell,\ell' \in \mathcal{Y}, \ell \neq \ell'\}$. Suppose Assumption 3.4.4 holds and $\alpha > 0$, then

$$\inf_{h\in\mathcal{H}:\ Err_{\mathcal{D}^B}(h)>0} \frac{Err^s_{\mathcal{D}^B}(h)}{Err_{\mathcal{D}^B}(h)} \geq \frac{\alpha}{r}.$$

Proof 3.4.6 *Denote the row-normalized confusion matrix of each hypothesis $h \in \mathcal{H}$ as $U_h \in [0,1]^{L\times L}$.*

$$U_h(\ell,\ell') = Pr_{x\sim\mathcal{D}^x(\ell)}(h(x)=\ell'),\ \ 1 \leq \ell,\ell' \leq L$$

The entry (ℓ,ℓ') of U_h means a random instance from class ℓ is classified as class ℓ' by h with probability $U_h(\ell,\ell')$. Each row of U_h sums to 1. Denote $k(Y,\ell)$ as the number of occurrences of label ℓ in Y.

Then the error of h can be expressed as

$$\begin{aligned} Err_{\mathcal{D}^B}(h) &= \frac{1}{r}E_Y\Big[\sum_{i=1}^{r}\sum_{\ell'\neq Y_i} U_h(Y_i,\ell') \mid Y\Big] = \frac{1}{r}E_Y\Big[\sum_{(\ell,\ell')\in I} k(Y,\ell)U_h(\ell,\ell') \mid Y\Big] \\ &\leq \frac{1}{r}\sum_{(\ell,\ell')\in I} rU_h(\ell,\ell') = \sum_{(\ell,\ell')\in I} U_h(\ell,\ell'). \end{aligned}$$

The expected superset error of h can be computed as

$$\begin{aligned} Err^s_{\mathcal{D}^B}(h) &= \frac{1}{r}E_Y\Big[\sum_{i=1}^{r}\sum_{\ell'\neq Y_i} \mathbb{I}(\ell' \notin S_i)U_h(Y_i,\ell') \mid Y\Big] \\ &= \frac{1}{r}E_Y\Big[\sum_{(\ell,\ell')\in I} k(Y,\ell)\mathbb{I}(\ell' \notin S_i)U_h(\ell,\ell') \mid Y\Big] \\ &\geq \frac{1}{r}\sum_{(\ell,\ell')\in I} \alpha U_h(\ell,\ell'). \end{aligned}$$

These two inequalities hold for any $h \in \mathcal{H}$, so the theorem is proved. $\square$

With the no co-occurring label condition that $\alpha > 0$, the remaining learnability requirement is that the hypothesis space $\mathcal{H}$ have finite Natarajan dimension. A merit of the condition of Theorem 3.4.5 is that it can be checked on the training data with high confidence. Suppose we have a training data $\mathbf{B}$. Then the empirical estimate of α is

$$\alpha_{\mathbf{B}} = \frac{1}{m} \min_{(\ell,\ell') \in I} \sum_{(X,Y,S) \in \mathbf{B}} \mathbb{I}(\ell \in S)\mathbb{I}(\ell' \notin S),$$

If $\alpha_{\mathbf{B}} > 0$, then by the Chernoff bound, we can obtain a lower bound on $\alpha > 0$ with high confidence. Conversely, if $\alpha_{\mathbf{B}} = 0$ for a large training set, then it is quite possible that α is very small or even zero.

3.5 Conclusion and Future Work

In this paper, we analyzed the learnability of an ERM learner on two superset label learning problems: SLL-I (for independent instances) and SLL-B (for bagged instances). Both problems can be learned by the same learner regardless the (in)dependency among the instances.

For both problems, the key to ERM learnability is that the expected classification error of any hypothesis in the space can be bounded by the superset error. If the tied error condition holds, then we can construct a binary classification problem from the SLL problem and bound the expected error in the binary classification problem. By constructing a relationship between the VC-dimension and the Natarajan dimension of the original problem, the sample complexities can be given in terms of the Natarajan dimension.

For the SLL-I problem, the condition of small ambiguity degree guarantees learnability for problems with hypothesis spaces having finite Natarajan dimension. This condition leads to lower sample complexity bounds than those obtained from the general tied error condition. The sample complexity analysis with this condition generalizes the analysis of multiclass classification. For the SLL-B problem, we propose a reasonable assumption for the distribution of bags. With this assumption, we identify a practical condition stating that no two labels always co-occur in bag label sets.

There is more to explore in theoretical analysis of the superset label learning problem. Analysis is needed for the agnostic case. Another important issue is to allow noise in the training data by removing the assumption that the true label is always in the label superset.

Manuscript 3: Gaussian Approximation of Collective Graphical Models

Li-Ping Liu, Daniel Sheldon, Thomas G. Dietterich

Proceedings of the 31st International Conference on Machine Learning (ICML-14)
Edited by T. Jebara and E. P. Xing
Published by JMLR Workshop and Conference Proceedings, 2014
Pages 1602-1610

Chapter 4: Gaussian Approximation of Collective Graphical Models

Abstract

The Collective Graphical Model (CGM) models a population of independent and identically distributed individuals when only collective statistics (i.e., counts of individuals) are observed. Exact inference in CGMs is intractable, and previous work has explored Markov Chain Monte Carlo (MCMC) and MAP approximations for learning and inference. This paper studies Gaussian approximations to the CGM. As the population grows large, we show that the CGM distribution converges to a multivariate Gaussian distribution (GCGM) that maintains the conditional independence properties of the original CGM. If the observations are exact marginals of the CGM or marginals that are corrupted by Gaussian noise, inference in the GCGM approximation can be computed efficiently in closed form. If the observations follow a different noise model (e.g., Poisson), then expectation propagation provides efficient and accurate approximate inference. The accuracy and speed of GCGM inference is compared to the MCMC and MAP methods on a simulated bird migration problem. The GCGM matches or exceeds the accuracy of the MAP method while being significantly faster.

4.1 Introduction

Consider a setting in which we wish to model the behavior of a population of independent and identically distributed (i.i.d.) individuals but where we can only observe collective count data. For example, we might wish to model the relationship between education, sex, housing, and income from census data. For privacy reasons, the Census Bureau only releases count data such as the number of people having a given level of education or the number of men living in a particular region. Another example concerns modeling the behavior of animals from counts of (anonymous) individuals observed at various locations and times. This arises in modeling the migration of fish and birds.

The CGM is constructed by first defining the *individual model*—a graphical model describing a single individual. Let $\mathcal{C}$ and $\mathcal{S}$ be the clique set and the separator set of a junction tree constructed from the individual model. Then, we define N copies of this individual model to create a population of N i.i.d. individuals. This permits us to define count variables $\mathbf{n}_A$, where

$\mathbf{n}_A(i_A)$ is the number of individuals for which clique $A \in \mathcal{C} \cup \mathcal{S}$ is in configuration i_A. The counts $\mathbf{n} = (\mathbf{n}_A : A \in \mathcal{C} \cup \mathcal{S})$ are the sufficient statistics of the individual model. After marginalizing away the individuals, the CGM provides a model for the joint distribution of $\mathbf{n}$.

In typical applications of CGMs, we make noisy observations $\mathbf{y}$ that depends on some of the $\mathbf{n}$ variables, and we seek to answer queries about the distribution of some or all of the $\mathbf{n}$ conditioned on these observations. Let $\mathbf{y} = (\mathbf{y}_D : D \in \mathcal{D})$, where $\mathcal{D}$ is a set of cliques from the individual graphical model and $\mathbf{y}_D$ contains counts of settings of clique D. We require each $D \subseteq A$ for some clique $A \in \mathcal{C} \cup \mathcal{S}$ the individual model. In addition to the usual role in graphical models, the inference of the distribution of $\mathbf{n}$ also serves to estimate the parameters of the individual model (e.g. E step in EM learning), because $\mathbf{n}$ are sufficient statistics of the individual model. Inference for CGMs is much more difficult than for the individual model. Unlike the individual model, many conditional distributions in the CGM do not have a closed form. The space of possible configurations of the CGM is very large, because each count variable $\mathbf{n}_i$ can take values in $\{0, \ldots, N\}$.

The original CGM paper, Sheldon et al. [60] introduced a Gibbs sampling algorithm for sampling from $P(\mathbf{n}|\mathbf{y})$. Subsequent experiments showed that this exhibits slow mixing times, which motivated Sheldon et al. [61] to introduce an efficient algorithm for computing a MAP approximation based on minimizing a tractable convex approximation of the CGM distribution. Although the MAP approximation still scales exponentially in the domain size L of the individual-model variables, it was fast enough to permit fitting CGMs via EM on modest-sized instances ($L = 49$). However, given that we wish to apply this to problems where $L = 1000$, we need a method that is even more efficient.

This paper introduces a Gaussian approximation to the CGM. Because the count variables $\mathbf{n}_C$ have a multinomial distribution, it is reasonable to apply the Gaussian approximation. However, this approach raises three questions. First, is the Gaussian approximation asymptotically correct? Second, can it maintain the sparse dependency structure of the CGM distribution, which is critical to efficient inference? Third, how well does it work with natural (non-Gaussian) observation distributions for counts, such as the Poisson distribution? This paper answers these questions by proving an asymptotically correct Gaussian approximation for CGMs. It shows that this approximation, when done correctly, is able to preserve the dependency structure of the CGM. And it demonstrates that by applying expectation propagation (EP), non-Gaussian observation distributions can be handled. The result is a CGM inference procedure that gives good accuracy and achieves significant speedups over previous methods.

Beyond CGMs, our main result highlights a remarkable property of discrete graphical models: the asymptotic distribution of the vector of sufficient statistics is a Gaussian graphical model with the same conditional independence properties as the original model.

4.2 Problem Statement and Notation

Consider a graphical model defined on the graph $G = (V, E)$ with n nodes and clique set $\mathcal{C}$. Denote the random variables by $X_1, \ldots, X_n$. Assume for simplicity all variables take values in the same domain $\mathcal{X}$ of size L. Let $\mathbf{x} \in \mathcal{X}^n$ be a particular configuration of the variables, and let $\mathbf{x}_C$ be the subvector of variables belonging to C. For each clique $C \in \mathcal{C}$, let $\phi_C(\mathbf{x}_C)$ be a non-negative potential function. Then the probability model is:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \phi_C(\mathbf{x}_C) = \exp\Big(\sum_{C \in \mathcal{C}} \sum_{i_C \in \mathcal{X}^{|C|}} \theta_C(i_C) \cdot \mathbb{I}(\mathbf{x}_C = i_C) - Q(\boldsymbol{\theta}) \Big). \tag{4.1}$$

The second line shows the model in exponential-family form [72], where $\mathbb{I}(\pi)$ is an indicator variable for the event or expression π, and $\theta_C(i_C) = \log \phi_C(i_C)$ is an entry of the vector of natural parameters. The function $Q(\boldsymbol{\theta}) = \log Z$ is the log-partition function. Given a fixed set of parameters $\boldsymbol{\theta}$ and any subset $A \subseteq V$, the *marginal distribution* $\boldsymbol{\mu}_A$ is the vector with entries $\mu_A(i_A) = \Pr(X_A = i_A)$ for all possible $i_A \in \mathcal{X}^{|A|}$. In particular, we will be interested in the clique marginals $\boldsymbol{\mu}_C$ and the node marginals $\boldsymbol{\mu}_i := \boldsymbol{\mu}_{\{i\}}$.

Junction Trees. Our development relies on the existence of a *junction tree* [38] on the cliques of $\mathcal{C}$ to write the relevant CGM and GCGM distributions in closed form. Henceforth, *we assume that such a junction tree exists*. In practice, this means that one may need to add fill-in edges to the original model to obtain the *triangulated* graph G, of which $\mathcal{C}$ is the set of maximal cliques. This is a clear limitation for graphs with high tree-width. Our methods apply directly to trees and are most practical for low tree-width graphs. Since we use few properties of the junction tree directly, we review only the essential details here and review the reader to Lauritzen [38] for further details. Let C and C' be two cliques that are adjacent in $\mathcal{T}$; their intersection $S = C \cap C'$ is called a *separator*. Let $\mathcal{S}$ be the set of all separators of $\mathcal{T}$, and let $\nu(S)$ be the number of times S appears as a separator, i.e., the number of different edges (C, C') in $\mathcal{T}$ for which $S = C \cap C'$.

The CGM Distribution. Fix a sample size N and let $\mathbf{x}^1, \dots, \mathbf{x}^N$ be N i.i.d. random vectors distributed according to the graphical model G. For any set $A \subseteq V$ and particular setting $i_A \in \mathcal{X}^{|A|}$, define the count

$$\mathbf{n}_A(i_A) = \sum_{m=1}^{N} \mathbb{I}(\mathbf{x}_A^m = i_A). \tag{4.2}$$

Let $\mathbf{n}_A = (\mathbf{n}_A(i_A) : i_A \in \mathcal{X}^{|A|})$ be the complete vector of counts for all possible settings of the variables in A. In particular, let $\mathbf{n}_u := \mathbf{n}_{\{u\}}$ be the vector of node counts. Also, let $\mathbf{n} = (\mathbf{n}_A : A \in \mathcal{C} \cup \mathcal{S})$ be the combined vector of all clique and separator counts—these are sufficient statistics of the sample of size N from the graphical model. The distribution over this vector is the CGM distribution.

Proposition 4.2.1 *Let* $\mathbf{n}$ *be the vector of (clique and separator) sufficient statistics of a sample of size N from the discrete graphical model* (4.1)*. The probability mass function of* $\mathbf{n}$ *is given by* $p(\mathbf{n}; \boldsymbol{\theta}) = h(\mathbf{n}) f(\mathbf{n}; \boldsymbol{\theta})$ *where*

$$f(\mathbf{n};\boldsymbol{\theta}) = \exp\big(\sum_{C\in\mathcal{C}, i_C \in \mathcal{X}^{|C|}} \theta_C(i_C) \cdot \mathbf{n}_C(i_C) - NQ(\boldsymbol{\theta})\big) \tag{4.3}$$

$$h(\mathbf{n}) = N! \cdot \frac{\prod_{S\in\mathcal{S}} \prod_{i_S\in\mathcal{X}^{|S|}} \Big(\mathbf{n}_S(i_S)!\Big)^{\nu(S)}}{\prod_{C\in\mathcal{C}} \prod_{i_C\in\mathcal{X}^{|C|}} \mathbf{n}_C(i_C)!} \prod_{S\sim C\in\mathcal{T}, i_S\in\mathcal{X}^{|S|}} \mathbb{I}\Big(\mathbf{n}_S(i_S) = \sum_{i_{C\setminus S}} \mathbf{n}_C(i_S, i_{C\setminus S})\Big) \cdot \prod_{C\in\mathcal{C}} \mathbb{I}\big(\sum_{i_C\in\mathcal{X}^{|C|}} \mathbf{n}_C(i_C) = N\big). \tag{4.4}$$

Denote this distribution by $\mathrm{CGM}(N, \boldsymbol{\theta})$.

Here, the notation $S \sim C \in \mathcal{T}$ means that S is adjacent to C in $\mathcal{T}$. This proposition was first proved in nearly this form by Sundberg [66] (see also Lauritzen [38]). Proposition 4.2.1 differs from those presentations by writing $f(\mathbf{n}; \boldsymbol{\theta})$ in terms of the original parameters $\boldsymbol{\theta}$ instead of the clique and separator marginals $\{\boldsymbol{\mu}_C, \boldsymbol{\mu}_S\}$, and by including hard constraints in the base measure $h(\mathbf{n})$. The hard constraints enforce consistency of the sufficient statistics of all cliques on their adjacent separators, and were treated implicitly prior to Sheldon et al. [60]. A proof of the equivalence between our expression for $f(\mathbf{n}; \boldsymbol{\theta})$ and the expressions from prior work is given in the supplementary material. David et al. [23] refer to the same distribution as the *hyper-multinomial* distribution due to the fact that it follows conditional independence properties analogous to those in the original graphical model.

Proposition 4.2.2 *Let $A, B \in \mathcal{S} \cup \mathcal{C}$ be two sets that are separated by the separator S in $\mathcal{T}$. Then $\mathbf{n}_A \perp\!\!\!\perp \mathbf{n}_B \mid \mathbf{n}_S$.*

Proof 4.2.3 *The probability model $p(\mathbf{n}; \boldsymbol{\theta})$ factors over the clique and separator count vectors $\mathbf{n}_C$ and $\mathbf{n}_S$. The only factors where two different count vectors appear together are the consistency constraints where $\mathbf{n}_S$ and $\mathbf{n}_C$ appear together if S is adjacent to C in $\mathcal{T}$. Thus, the CGM is a graphical model with the same structure as $\mathcal{T}$, from which the claim follows.* $\square$

4.3 Approximating CGM by the Normal Distribution

In this section, we will develop a Gaussian approximation, GCGM, of the CGM and show that it is the asymptotically correct distribution as M goes to infinity. We then show that the GCGM has the same conditional independence structure as the CGM, and we explicitly derive the conditional distributions. These allow us to use Gaussian message passing in the GCGM as a practical approximate inference method for CGMs.

We will follow the most natural approach of approximating the CGM distribution by a multivariate Gaussian with the same mean and covariance matrix. The moments of the CGM distribution follow directly from those of the indicator variables of the individual model: Fix an outcome $\mathbf{x} = (x_1, \ldots, x_n)$ from the individual model and for any set $A \subseteq V$ let $\mathbf{I}_A = \big(\mathbb{I}(\mathbf{x}_A = i_A) : i_A \in \mathcal{X}^{|A|}\big)$ be the vector of all indicator variables for that set. The mean and covariance of any such vectors are given by

$$\mathbb{E}[\mathbf{I}_A] = \boldsymbol{\mu}_A \tag{4.5}$$

$$\operatorname{cov}(\mathbf{I}_A, \mathbf{I}_B) = \langle \boldsymbol{\mu}_{A,B} \rangle - \boldsymbol{\mu}_A \boldsymbol{\mu}_B^T. \tag{4.6}$$

Here, the notation $\langle \boldsymbol{\mu}_{A,B} \rangle$ refers to the matrix whose (i_A, i_B) entry is the marginal probability $\Pr(X_A = i_A, X_B = i_B)$. Note that Eq. (4.6) follows immediately from the definition of covariance for indicator variables, which is easily seen in the scalar form: $\operatorname{cov}(\mathbb{I}(X_A = i_A), \mathbb{I}(X_B = i_B)) = \Pr(X_A = i_A, X_B = i_B) - \Pr(X_A = i_A)\Pr(X_B = i_B)$. Eq. (4.6) also covers the case when $A \cap B$ is nonempty. In particular if $A = B = \{u\}$, then we recover $\operatorname{cov}(\mathbf{I}_u, \mathbf{I}_u) = \operatorname{diag}(\boldsymbol{\mu}_u) - \boldsymbol{\mu}_u \boldsymbol{\mu}_u^T$, which is the covariance matrix for the marginal multinomial distribution of $\mathbf{I}_u$.

From the preceding arguments, it becomes clear that the covariance matrix for the full vector of indicator variables has a simple block structure. Define $\mathbf{I} = (\mathbf{I}_A : A \in \mathcal{C} \cup \mathcal{S})$ to be the vector concatention of all the clique and separator indicator variables, and let $\boldsymbol{\mu} = (\boldsymbol{\mu}_A : A \in \mathcal{C} \cup \mathcal{S}) = \mathbb{E}[\mathbf{I}]$ be the corresponding vector concatenation of marginals. Then it follows from (4.6) that the covariance matrix is

$$\Sigma := \operatorname{cov}(\mathbf{I}, \mathbf{I}) = \hat{\Sigma} - \boldsymbol{\mu}\boldsymbol{\mu}^T, \tag{4.7}$$

where $\hat{\Sigma}$ is the matrix whose (A, B) block is the marginal distribution $\langle \boldsymbol{\mu}_{A,B} \rangle$. In the CGM model, the count vector $\mathbf{n}$ can be written as $\mathbf{n} = \sum_{m=1}^{N} \mathbf{I}^m$, where $\mathbf{I}^1, \ldots, \mathbf{I}^N$ are i.i.d. copies of $\mathbf{I}$. As a result, the moments of the CGM are obtained by scaling the moments of $\mathbf{I}$ by N. We thus arrive at the natural moment-matching Gaussian approximation of the CGM.

Definition The Gaussian CGM, denoted $\mathrm{GCGM}(N, \boldsymbol{\theta})$ is the multivariate normal distribution $\mathcal{N}(N\boldsymbol{\mu}, N\Sigma)$, where $\boldsymbol{\mu}$ is the vector of all clique and separator marginals of the graphical model with parameters $\boldsymbol{\theta}$, and Σ is defined in Equation (4.7).

In the following theorem, we show the GCGM is asymptotically correct and it is a Gaussian graphical model, which will lead to efficient inference algorithms.

Theorem 4.3.1 *Let $\mathbf{n}^N \sim \mathrm{CGM}(N, \boldsymbol{\theta})$ for $N = 1, 2, \ldots$. Then following are true:*

(i) The GCGM is asymptotically correct. That is, as $N \to \infty$ we have

$$\frac{1}{\sqrt{N}}(\mathbf{n}^N - N\boldsymbol{\mu}) \xrightarrow{D} \mathcal{N}(\mathbf{0}, \Sigma). \tag{4.8}$$

(ii) The GCGM is a Gaussian graphical model with the same conditional independence structure as the CGM. Let $\mathbf{z} \sim \mathrm{GCGM}(N, \boldsymbol{\theta})$ and let $A, B \in \mathcal{C} \cup \mathcal{S}$ be two sets that are separated by separator S in $\mathcal{T}$. Then $\mathbf{z}_A \perp\!\!\!\perp \mathbf{z}_B \mid \mathbf{z}_S$.

Proof 4.3.2 *Part (i) is a direct application of the multivariate central limit theorem to the random vector $\mathbf{n}^N$, which, as noted above, is a sum of i.i.d. random vectors $\mathbf{I}^1, \ldots, \mathbf{I}^N$ with mean $\boldsymbol{\mu}$ and covariance Σ [25].*

Part (ii) is a consequence of the fact that these conditional independence properties hold for each $\mathbf{n}^N$ (Proposition 4.2.2), so they also hold in the limit as $N \to \infty$. While this is intuitively clear, it seems to require further justification, which is provided in the supplementary material.
$\square$

4.3.1 Conditional Distributions

The goal is to use inference in the GCGM as a tractable approximate alternative inference method for CGMs. However, it is very difficult to compute the covariance matrix Σ over all cliques. In particular, note that the (C, C') block requires the joint marginal $\langle \boldsymbol{\mu}_{C,C'} \rangle$, and if C and C' are not adjacent in $\mathcal{T}$ this is hard to compute. Fortunately, we can sidestep the problem completely by leveraging the graph structure from Part (ii) of Theorem 4.3.1 to write the distribution as a product of conditional distributions whose parameters are easy to compute (this effectively means working with the inverse covariance matrix instead of Σ). We then perform inference by Gaussian message passing on the resulting model.

A challenge is that Σ is not full rank, so the GCGM distribution as written is degenerate and does not have a density. This can be seen by noting that any vector $\mathbf{n} \sim \mathrm{CGM}(N; \boldsymbol{\theta})$ with nonzero probability satisfies the affine consistency constraints from Eq. (4.4)—for example, each vector $\mathbf{n}_C$ and $\mathbf{n}_S$ sums to the population size N—and that these affine constraints also hold with probability one in the limiting distribution. To fix this, we instead use a linear transformation $\mathbb{T}$ to map $\mathbf{z}$ to a reduced vector $\tilde{\mathbf{z}} = \mathbb{T}\,\mathbf{z}$ such that the reduced covariance matrix $\tilde{\Sigma} = \mathbb{T}\,\Sigma\,\mathbb{T}^T$ is invertible. The work by Loh et al. [44] proposed a minimal representation of the graphical model in (4.1), and the corresponding random variable has a full rank covariance matrix. We will find a transformation $\mathbb{T}$ to project our indicator variable $\mathbf{I}$ into that form. Then $\mathbb{T}\,\mathbf{I}$ (as well as $\mathbb{T}\,\mathbf{n}$ and $\mathbb{T}\,\mathbf{z}$) will have a full rank covariance matrix.

Denote by $\mathcal{C}^+$ the maximal and non-maximal cliques in the triangulated graph. Note that each $D \in \mathcal{C}^+$ must be a subset of some $A \in \mathcal{C} \cup \mathcal{S}$ and each subset of A is also a clique in $\mathcal{C}^+$. For every $D \in \mathcal{C}^+$, let $\mathcal{X}_0^D = (\mathcal{X} \backslash \{L\})^{|D|}$ denote the space of possible configurations of D after excluding the largest value, L, from the domain of each variable in D. The corresponding random variable $\mathbb{I}$ in the minimal representation is defined as Loh et al. [44]:

$$\tilde{\mathbf{I}} = (\mathbb{I}(\mathbf{x}_D = i_D) : i_D \in \mathcal{X}_0^D, D \in \mathcal{C}^+) \ . \tag{4.9}$$

$\tilde{\mathbf{I}}_D$ can be calculated linearly from $\mathbf{I}_A$ when $D \subseteq A$ via the matrix $\mathbb{T}_{D,A}$ whose (i_D, i_A) entry is defined as

$$\mathbb{T}_{D,A}(i_D, i_A) \quad = \quad \mathbb{I}(i_D \sim_D i_A), \tag{4.10}$$

where $\sim_D$ means that i_D and i_A agree on the setting of the variables in D. It follows that

$\tilde{\mathbf{I}}_D = \mathbb{T}_{D,A}\ \mathbf{I}_A$. The whole transformation $\mathbb{T}$ can be built in blocks as follows: For every $D \in \mathcal{C}^+$, choose $A \in \mathcal{C} \cup \mathcal{S}$ and construct the $\mathbb{T}_{D,A}$ block via (4.10). Set all other blocks to zero. Due to the redundancy of $\mathbf{I}$, there might be many ways of choosing A for D and any one will work as long as $D \subseteq A$.

Proposition 4.3.3 *Define $\mathbb{T}$ as above, and define $\tilde{\mathbf{z}} = \mathbb{T}\ \mathbf{z}$, $\tilde{\mathbf{z}}_{A^+} = (\tilde{\mathbf{z}}_D : D \subseteq A), A \in \mathcal{C} \cup \mathcal{S}$. Then*

(i) If $A, B \in \mathcal{C} \cup S$ are separated by S in $\mathcal{T}$, it holds that $\tilde{\mathbf{z}}_{A^+} \perp\!\!\!\perp \tilde{\mathbf{z}}_{B^+} \mid \tilde{\mathbf{z}}_{S^+}$.

(ii) The covariance matrix of $\tilde{\mathbf{z}}$ has full rank.

Proof 4.3.4 *In the appendix, we show that for any $A \in \mathcal{C} \cup \mathcal{S}$, $\mathbf{I}_A$ can be linearly recovered from $\tilde{\mathbf{I}}_{A^+} = (\tilde{\mathbf{I}}_D : D \subseteq A)$. So there is a linear bijection between $\mathbf{I}_A$ and $\tilde{\mathbf{I}}_{A^+}$ (The mapping from $\mathbf{I}_A$ to $\tilde{\mathbf{I}}_{A^+}$ has been shown in the definition of $\mathbb{T}$). The same linear bijection relation also exists between $\mathbf{n}_A$ and $\tilde{\mathbf{n}}_{A^+} = \sum_{m=1}^{N} \tilde{\mathbf{I}}_{A^+}^m$ and between $\mathbf{z}_A$ and $\tilde{\mathbf{z}}_{A^+}$.*

Proof of (i): Since $\mathbf{z}_A \perp\!\!\!\perp \mathbf{z}_B \mid \mathbf{z}_S$, it follows that $\tilde{\mathbf{z}}_{A^+} \perp\!\!\!\perp \tilde{\mathbf{z}}_{B^+} \mid \mathbf{z}_S$ because $\tilde{\mathbf{z}}_{A^+}$ and $\tilde{\mathbf{z}}_{B^+}$ are deterministic functions of $\mathbf{z}_A$ and $\mathbf{z}_B$ respectively. Since $\mathbf{z}_S$ is a deterministic function of $\tilde{\mathbf{z}}_{S^+}$, the same property holds when we condition on $\tilde{\mathbf{z}}_{S^+}$ instead of $\mathbf{z}_S$.

Proof of (ii): The bijection between $\mathbf{I}$ and $\tilde{\mathbf{I}}$ indicates that the model representation of Loh et al. [44] defines the same model as (4.1). *By Loh et al. [44], $\tilde{\mathbf{I}}$ has full rank covariance matrix and so do $\tilde{\mathbf{n}}$ and $\tilde{\mathbf{z}}$.* □

With this result, the GCGM can be decomposed into conditional distributions, and each distribution is a non-degenerate Gaussian distribution.

Now let us consider the observations $\mathbf{y} = \{\mathbf{y}_D, D \in \mathcal{D}\}$, where $\mathcal{D}$ is the set of cliques for which we have observations. We require each $D \in \mathcal{D}$ be subset of some clique $C \in \mathcal{C}$. When choosing a distribution $p(\mathbf{y}_D|\mathbf{z}_C)$, a modeler has substantial flexibility. For example, $p(\mathbf{y}_D|\mathbf{z}_C)$ can be noiseless, $\mathbf{y}_D(i_D) = \sum_{i_{C \setminus D}} \mathbf{z}_C(i_D, i_{C\setminus D})$, which permits closed-form inference. Or $p(\mathbf{y}_D|\mathbf{z}_C)$ can consist of independent noisy observations: $p(\mathbf{y}_D|\mathbf{z}_C) = \prod_{i_D} p(\mathbf{y}_D(i_D)| \sum_{i_{C\setminus D}} \mathbf{z}_C(i_D, i_{C\setminus D}))$. With a little work, $p(\mathbf{y}_D|\mathbf{z}_C)$ can be represented by $p(\mathbf{y}_D|\tilde{\mathbf{z}}_{C^+})$.

4.3.2 Explicit Factored Density for Trees

We describe how to decompose GCGM for the special case when the original graphical model G is a tree. We assume that only counts of single nodes are observed. In this case, we can marginalize out edge (clique) counts $\mathbf{z}_{\{u,v\}}$ and retain only node (separator) counts $\mathbf{z}_u$. Because the GCGM has a normal distribution, marginalization is easy. The conditional distribution is then defined only on node counts. With the definition of $\tilde{\mathbf{z}}$ in Proposition (4.3.3) and the property of conditional independence, we can write

$$p(\tilde{\mathbf{z}}_1, \ldots, \tilde{\mathbf{z}}_n) = p(\tilde{\mathbf{z}}_r) \prod_{(u,v)\in E} p(\tilde{\mathbf{z}}_v \mid \tilde{\mathbf{z}}_u). \tag{4.11}$$

Here $r \in V$ is an arbitrarily-chosen root node, and E is the set of *directed* edges of G oriented away from r. The marginalization of the edges greatly reduces the size of the inference problem, and a similar technique is also applicable to general GCGMs.

Now specify the parameters of the Gaussian conditional densities $p(\tilde{\mathbf{z}}_v \mid \tilde{\mathbf{z}}_u)$ in Eq. (4.11). Assume the blocks $\mathbb{T}_{u,u}$ and $\mathbb{T}_{v,v}$ are defined as (4.10). Let $\tilde{\boldsymbol{\mu}}_u = \mathbb{T}_{u,u}\ \boldsymbol{\mu}_u$ be the marginal vector of node u without its last entry, and let $\langle \tilde{\boldsymbol{\mu}}_{u,v} \rangle = \mathbb{T}_{u,u} \langle \boldsymbol{\mu}_{u,v} \rangle \mathbb{T}_{v,v}^T$ be the marginal matrix over edge (u,v), minus the final row and column. Then the mean and covariance martix of the joint distribution are

$$\boldsymbol{\eta} := N \begin{bmatrix} \tilde{\boldsymbol{\mu}}_u \\ \tilde{\boldsymbol{\mu}}_v \end{bmatrix}, \quad N^2 \begin{bmatrix} \operatorname{diag}(\tilde{\boldsymbol{\mu}}_u) & \langle \tilde{\boldsymbol{\mu}}_{u,v} \rangle \\ \langle \tilde{\boldsymbol{\mu}}_{v,u} \rangle & \operatorname{diag}(\tilde{\boldsymbol{\mu}}_v) \end{bmatrix} - \boldsymbol{\eta}\boldsymbol{\eta}^T. \tag{4.12}$$

The conditional density $p(\tilde{\mathbf{z}}_v \mid \tilde{\mathbf{z}}_u)$ is obtained by standard Gaussian conditioning formulas.

If we need to infer $\mathbf{z}_{\{u,v\}}$ from some distribution $q(\tilde{\mathbf{z}}_u, \tilde{\mathbf{z}}_v)$, we first calculate the distribution $p(\tilde{\mathbf{z}}_{\{u,v\}} | \tilde{\mathbf{z}}_u, \tilde{\mathbf{z}}_v)$. This time we assume blocks $\mathbb{T}_{\{u,v\}^+,\{u,v\}} = (\mathbb{T}_{u,\{u,v\}} : D \in \{u,v\})$ are defined as (4.10). We can find the mean and variance of $p(\tilde{\mathbf{z}}_u, \tilde{\mathbf{z}}_v, \tilde{\mathbf{z}}_{\{u,v\}})$ by applying linear transformation $\mathbb{T}_{\{u,v\}^+,\{u,v\}}$ on the mean and variance of $\mathbf{z}_{\{u,v\}}$. Standard Gaussian conditioning formulas then give the conditional distribution $p(\tilde{\mathbf{z}}_{\{u,v\}} \mid \tilde{\mathbf{z}}_u, \tilde{\mathbf{z}}_v)$. Then we can recover the distribution of $\mathbf{z}_{\{u,v\}}$ from distribution $p(\tilde{\mathbf{z}}_{\{u,v\}} | \tilde{\mathbf{z}}_u, \tilde{\mathbf{z}}_v) q(\tilde{\mathbf{z}}_u, \tilde{\mathbf{z}}_v)$.

Remark: Our reasoning gives a completely different way of deriving some of the results of Loh et al. [44] concerning the sparsity pattern of the inverse covariance matrix of the sufficient statistics of a discrete graphical model. The conditional independence in Proposition 4.2.2 for the factored GCGM density translates directly to the sparsity pattern in the precision matrix

$\Gamma = \tilde{\Sigma}^{-1}$. Unlike the reasoning of Loh et al. [44], we derive the sparsity directly from the conditional independence properties of the asymptotic distribution (which are inherited from the CGM distribution) and the fact that the CGM and GCGM share the same covariance matrix.

4.4 Inference with Noisy Observations

We now consider the problem of inference in the GCGM when the observations are noisy. Throughout the remainder of the paper, we assume that the individual model—and, hence, the CGM—is a tree. In this case, the cliques correspond to edges and the separators correspond to nodes. We will also assume that only the nodes are observed. For notational simplicity, we will assume that every node is observed (with noise). (It is easy to marginalize out unobserved nodes if any.) From now on, we use uv instead of $\{u, v\}$ to represent edge clique. Finally, we assume that the entries have been dropped from the vector $\mathbf{z}$ as described in the previous section so that it has the factored density described in Eq. 4.11.

Denote the observation variable for node u by $\mathbf{y}_u$, and assume that it has a Poisson distribution. In the (exact) CGM, this would be written as $\mathbf{y}_u \sim \mathrm{Poisson}(\mathbf{n}_u)$. However, in our GCGM, this instead has the form

$$\mathbf{y}_u \sim \mathrm{Poisson}(\lambda \mathbf{z}_u), \tag{4.13}$$

where $\mathbf{z}_u$ is the corresponding continuous variable and λ determines the amount of noise in the distribution. Denote the vector of all observations by $\mathbf{y}$. Note that the missing entry of $\mathbf{z}_u$ must be reconstructed from the remaining entries when computing the likelihood.

With Poisson observations, there is no longer a closed-form solution to message passing in the GCGM. We address this by applying Expectation Propagation (EP) with the Laplace approximation. This method has been previously applied to nonlinear dynamical systems by Ypma et al. [75].

4.4.1 Inferring Node Counts

In the GCGM with observations, the potential on each edge $(u, v) \in E$ is defined as

$$\psi(\mathbf{z}_u, \mathbf{z}_v) = \begin{cases} p(\mathbf{z}_v, \mathbf{z}_u) p(\mathbf{y}_v|\mathbf{z}_v) p(\mathbf{y}_u|\mathbf{z}_u) & \text{if } u \text{ is root} \\ p(\mathbf{z}_v|\mathbf{z}_u) p(\mathbf{y}_v|\mathbf{z}_v) & \text{otherwise.} \end{cases} \tag{4.14}$$

We omit the subscripts on ψ for notational simplicity. The joint distribution of $(\mathbf{z}_v, \mathbf{z}_u)$ has mean and covariance shown in (4.12).

With EP, the model approximates potential on edge $(u, v) \in E$ with normal distribution in context $q_{\setminus uv}(\mathbf{z}_u)$ and $q_{\setminus uv}(\mathbf{z}_v)$. The context for edge (u, v) is defined as

$$q_{\setminus uv}(\mathbf{z}_u) = \prod_{(u,v') \in E, v' \neq v} q_{uv'}(\mathbf{z}_u) \tag{4.15}$$

$$q_{\setminus uv}(\mathbf{z}_v) = \prod_{(u',v) \in E, u' \neq u} q_{u'v}(\mathbf{z}_v), \tag{4.16}$$

where each $q_{uv'}(\mathbf{z}_u)$ and $q_{u'v}(\mathbf{z}_v)$ have the form of normal densities.

Let $\xi(\mathbf{z}_u, \mathbf{z}_v) = q_{\setminus uv}(\mathbf{z}_u) q_{\setminus uv}(\mathbf{z}_v) \psi(\mathbf{z}_u, \mathbf{z}_v)$. The EP update of $q_{uv}(\mathbf{z}_u)$ and $q_{uv}(\mathbf{z}_v)$ is computed as

$$q_{uv}(\mathbf{z}_u) = \frac{\mathrm{proj}_{\mathbf{z}_u}[\xi(\mathbf{z}_u, \mathbf{z}_v)]}{q_{\setminus uv}(\mathbf{z}_u)} \tag{4.17}$$

$$q_{uv}(\mathbf{z}_v) = \frac{\mathrm{proj}_{\mathbf{z}_v}[\xi(\mathbf{z}_u, \mathbf{z}_v)]}{q_{\setminus uv}(\mathbf{z}_v)}. \tag{4.18}$$

The projection operator, proj, is computed in two steps. First, we find a joint approximating normal distribution via the Laplace approximation and then we project this onto each of the random variables $\mathbf{z}_u$ and $\mathbf{z}_v$. In the Laplace approximation step, we need to find the mode of $\log \xi(\mathbf{z}_u, \mathbf{z}_v)$ and calculate its Hessian at the mode to obtain the mean and variance of the approximating normal distribution:

$$\mu_{uv}^{\xi} = \arg\max_{(\mathbf{z}_u, \mathbf{z}_v)} \log \xi(\mathbf{z}_u, \mathbf{z}_v) \tag{4.19}$$

$$\Sigma_{uv}^{\xi} = \left(\nabla^2_{(\mathbf{z}_u, \mathbf{z}_v) = \mu_{uv}^{\xi}} \log \xi(\mathbf{z}_u, \mathbf{z}_v) \right)^{-1}. \tag{4.20}$$

The optimization problem in (4.19) is solved by optimizing first over $\mathbf{z}_u$ then over $\mathbf{z}_v$. The optimal value of $\mathbf{z}_u$ can be computed in closed form in terms of $\mathbf{z}_v$, since only normal densities are involved. Then the optimal value of $\mathbf{z}_v$ is found via gradient methods (e.g., BFGS). The function $\log \xi(\mathbf{z}_u, \mathbf{z}_v)$ is concave, so we can always find the global optimum. Note that this decomposition approach only depends on the tree structure of the model and hence will work for any observation distribution.

At the mode, we find the mean and variance of the normal distribution approximating $p(\mathbf{z}_u, \mathbf{z}_v|\mathbf{y})$ via (4.19) and (4.20). With this distribution, the edge counts can be inferred with the method of Section 4.3.2. In the projection step in (4.17) and (4.18), this distribution is projected to one of $\mathbf{z}_u$ or $\mathbf{z}_v$ by marginalizing out the other.

4.4.2 Complexity analysis

What is the computational complexity of inference with the GCGM? When inferring node counts, we must solve the optimization problem and compute a fixed number of matrix inverses. Each matrix inverse takes time L^3 (by Gauss-Jordan elimination). In the Laplace approximation step, each gradient calculation takes $O(L^2)$ time. Suppose m iterations are needed. In the outer loop, suppose we must perform r passes of EP message passing and each iteration sweeps through the whole tree. Then the overall time is $O(r|E|\max(mL^2, L^3))$. The maximization problem in the Laplace approximation is smooth and concave, so it is relatively easy. In our experiments, EP usually converges within 10 iterations.

In the task of inferring edge counts, we only consider the complexity of calculating the mean, as this is all that is needed in our applications. This part is solved in closed form, with the most time-consuming operation being the matrix inversion. By exploiting the simple structure of the covariance matrix of $\mathbf{z}_{uv}$, we can obtain an inference method with time complexity of $O(L^3)$.

4.5 Experimental Evaluation

In this section, we evaluate the performance of our method and compare it to the MAP approximation of Sheldon et al. [61]. The evaluation data are generated from the bird migration model introduced in Sheldon et al. [61]. This model simulates the migration of a population of M birds on an $L = \ell \times \ell$ map. The entire population is initially located in the bottom left corner of the map. Each bird then makes independent migration decisions for $T = 20$ time steps. The transition probability from cell i to cell j at each time step is determined by a logistic regression equation that employs four features. These features encode the distance from cell i to cell j, the degree to which cell j falls near the path from cell i to the destination cell in the upper right corner, the degree to which cell j lies in the direction toward which the wind is blowing, and a factor that encourages the bird to stay in cell i. Let $\mathbf{w}$ denote the parameter vector for this logistic regression formula. In this simulation, the individual model for each bird is a T-step Markov chain $X = (X_1, \ldots, X_{20})$ where the domain of each X_t consists of the L cells in the map. The CGM variables $\mathbf{n} = (\mathbf{n}_1, \mathbf{n}_{1,2}, \mathbf{n}_2, \ldots, \mathbf{n}_T)$ are vectors of length L containing counts of the number of birds in each cell at time t and the number of birds moving from cell i to cell j from time t to time $t+1$. We will refer to these as the "node counts" (N) and the "edge counts" (E). At each time step t, the data generation model generates an observation vector $\mathbf{y}_t$ of length L which contains noisy counts of birds at all map cells at time t, $\mathbf{n}_t$. The observed counts are generated by a Poisson distribution with unit intensity.

We consider two inference tasks. In the first task, the parameters of the model are given, and the task is to infer the expected value of the posterior distribution over $\mathbf{n}_t$ for each time step t given the observations $\mathbf{y}_1, \ldots, \mathbf{y}_T$ (aka "smoothing"). We measure the accuracy of the node counts and edge counts separately.

An important experimental issue is that we cannot compute the true MAP estimates for the node and edge counts. Of course we have the values generated during the simulation, but because of the noise introduced into the observations, these are not necessarily the expected values of the posterior. Instead, we estimate the expected values by running the MCMC method [60] for a burn-in period of 1 million Gibbs iterations and then collecting samples from 10 million Gibbs iterations and averaging the results. We evaluate the accuracy of the approximate methods as the relative error $||\mathbf{n}_{app} - \mathbf{n}_{mcmc}||_1 / ||\mathbf{n}_{mcmc}||_1$, where $\mathbf{n}_{app}$ is the approximate estimate and $\mathbf{n}_{mcmc}$ is the value obtained from the Gibbs sampler. In each experiment, we report the mean and standard deviation of the relative error computed from 10 runs. Each run generates a new set of values for the node counts, edge counts, and observation counts and requires a separate MCMC baseline run.

We compare our method to the approximate MAP method introduced by Sheldon et al. [61]. By treating counts as continuous and approximating the log factorial function, their MAP method finds the approximate mode of the posterior distribution by solving a convex optimization problem. Their work shows that the MAP method is much more efficient than the Gibbs sampler and produces inference results and parameter estimates very similar to those obtained from long MCMC runs.

The second inference task is to estimate the parameters $\mathbf{w}$ of the transition model from the observations. This is performed via Expectation Maximization, where our GCGM method is applied to compute the E step. We compute the relative error with respect to the true model parameters.

Table 4.1 compares the inference accuracy of the approximate MAP and GCGM methods. In this table, we fixed $L = 36$, set the logistic regression coefficient vector $\mathbf{w} = (1, 2, 2, 2)$, and varied the population size $N \in \{36, 360, 1080, 3600\}$. At the smallest population size, the MAP approximation is slightly better, although the result is not statistically significant. This makes sense, since the Gaussian approximation is weakest when the population size is small. At all larger population sizes, the GCGM gives much more accurate results. Note that the MAP approximation exhibits much higher variance as well.

Table 4.1: Relative error in estimates of node counts ("N") and edge counts ("E") for different population sizes N.

$N =$	36	360	1080	3600
MAP(N)	.173±.020	.066±.015	.064±.012	.069±.013
MAP(E)	.350±.030	.164±.030	.166±.027	.178±.025
GCGM(N)	.184±.018	.039±.007	.017±.003	.009±.002
GCGM(E)	.401±.026	.076±.008	.034±.003	.017±.002

Our second inference experiment is to vary the magnitude of the logistic regression coefficients. With large coefficients, the transition probabilities become more extreme (closer to 0 and 1), and the Gaussian approximation should not work as well. We fixed $N = 1080$ and $L = 36$ and evaluated three different parameter vectors: $\mathbf{w}_{0.5} = (0.5, 1, 1, 1)$, $\mathbf{w}_1 = (1, 2, 2, 2)$ and $\mathbf{w}_2 = (2, 4, 4, 4)$. Table 4.2 shows that for $\mathbf{w}_{0.5}$ and $\mathbf{w}_1$, the GCGM is much more accurate, but for $\mathbf{w}_2$, the MAP approximation gives a slightly better result, although it is not statistically significant based on 10 trials.

Table 4.2: Relative error in estimates of node counts ("N") and edge counts ("E") for different settings of the logistic regression parameter vector $\mathbf{w}$

	$\mathbf{w}_{0.5}$	$\mathbf{w}_1$	$\mathbf{w}_2$
MAP(N)	.107±.014	.064±.012	.018±.004
MAP(E)	.293±.038	.166±.027	.031±.004
GCGM(N)	.013±.002	.017±.003	.024±.004
GCGM(E)	.032±.004	.034±.003	.037±.005

Our third inference experiment explores the effect of varying the size of the map. This increases the size of the domain for each of the random variables and also increases the number of values that must be estimated (as well as the amount of evidence that is observed). We vary $L \in \{16, 25, 36, 49\}$. We scale the population size accordingly, by setting $N = 30L$. We use the

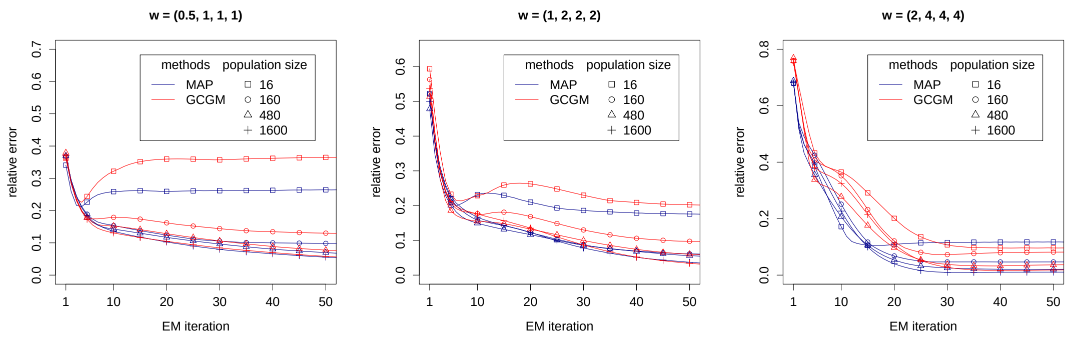

Figure 4.1: EM convergence curve different feature coefficient and population sizes

coefficient vector $\mathbf{w}_1$. The results in Table 4.3 show that for the smallest map, both methods give similar results. But as the number of cells grows, the relative error of the MAP approximation grows rapidly as does the variance of the result. In comparison, the relative error of the GCGM method barely changes.

Table 4.3: Relative inference error with different map size

$L =$	16	25	36	49
MAP(N)	.011±.005	.025±.007	.064±.012	.113±.015
MAP(E)	.013±.004	.056±.012	.166±.027	.297±.035
GCGM(N)	.017±.003	.017±.003	.017±.003	.020±.003
GCGM(E)	.024±.002	.027±.003	.034±.003	.048±.005

We now turn to measuring the relative accuracy of the methods during learning. In this experiment, we set $L = 16$ and vary the population size for $N \in \{16, 160, 480, 1600\}$. After each EM iteration, we compute the relative error as $||\mathbf{w}_{learn} - \mathbf{w}_{true}||_1/||\mathbf{w}_{true}||_1$, where $\mathbf{w}_{learn}$ is the parameter vector estimated by the learning methods and $\mathbf{w}_{true}$ is the parameter vector that was used to generate the data. Figure 4.1 shows the training curves for the three parameter vectors $\mathbf{w}_{0.5}, \mathbf{w}_1$, and $\mathbf{w}_2$. The results are consistent with our previous experiments. For small population sizes ($N = 16$ and $N = 160$), the GCGM does not do as well as the MAP approximation. In some cases, it overfits the data. For $N = 16$, the MAP approximation also exhibits overfitting. For $\mathbf{w}_2$, which creates extreme transition probabilities, we also observe that the MAP approximation learns faster, although the GCGM eventually matches its performance with enough EM iterations.

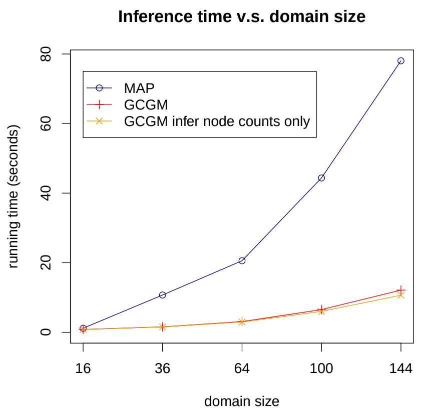

Figure 4.2: A comparison of inference run time with different numbers of cells L

Our final experiment measures the CPU time required to perform inference. In this experiment, we varied $L \in \{16, 36, 64, 100, 144\}$ and set $N = 100L$. We used parameter vector $\mathbf{w}_1$. We measured the CPU time consumed to infer the node counts and the edge counts. The MAP method infers the node and edge counts jointly, whereas the GCGM first infers the node counts and then computes the edge counts from them. We report the time required for computing just the node counts and also the total time required to compute the node and edge counts. Figure 4.2 shows that the running time of the MAP approximation is much larger than the running time of the GCGM approximation. For all values of L except 16, the average running time of GCGM is more than 6 times faster than for the MAP approximation. The plot also reveals that the computation time of GCGM is dominated by estimating the node counts. A detailed analysis of the implementation indicates that the Laplace optimization step is the most time-consuming.

In summary, the GCGM method achieves relative error that matches or is smaller than that achieved by the MAP approximation. This is true both when measured in terms of estimating the values of the latent node and edge counts and when estimating the parameters of the underlying graphical model. The GCGM method does this while running more than a factor of 6 faster. The GCGM approximation is particularly good when the population size is large and when the transition probabilities are not near 0 or 1. Conversely, when the population size is small or the probabilities are extreme, the MAP approximation gives better answers although the differences were not statistically significant based on only 10 trials. A surprising finding is that the MAP approximation has much larger variance in its answers than the GCGM method.

4.6 Concluding Remarks

This paper has introduced the Gaussian approximation (GCGM) to the Collective Graphical Model (CGM). We have shown that for the case where the observations only depend on the separators, the GCGM is the limiting distribution of the CGM as the population size $N \to \infty$. We showed that the GCGM covariance matrix maintains the conditional independence structure of the CGM, and we presented a method for efficiently inverting this covariance matrix. By applying expectation propagation, we developed an efficient algorithm for message passing in the GCGM with non-Gaussian observations. Experiments on a bird migration simulation showed that the GCGM method is at least as accurate as the MAP approximation of Sheldon et al. [61], that it exhibits much lower variance, and that it is 6 times faster to compute.

Manuscript 4: Transductive Optimization of Top k Precision

Li-Ping Liu, Thomas G. Dietterich, Nan Li, Zhi-Hua Zhou

Accepted by 25th International Joint Conference on Artificial Intelligence

Chapter 5: Transductive Optimization of Top k Precision

Abstract

Consider a binary classification problem in which the learner is given a labeled training set, an unlabeled test set, and is restricted to choosing exactly k test points to output as positive predictions. Problems of this kind—*transductive precision@k*—arise in many applications. Previous methods solve these problems in two separate steps, learning the model and selecting k test instances by thresholding their scores. In this way, model training is not aware of the constraint of choosing k test instances as positive in the test phase. This paper shows the importance of incorporating the knowledge of k into the learning process and introduces a new approach, Transductive Top K (TTK), that seeks to minimize the hinge loss over all training instances under the constraint that exactly k test instances are predicted as positive. The paper presents two optimization methods for this challenging problem. Experiments and analysis confirm the benefit of incoporating k in the learning process. In our experimental evaluations, the performance of TTK matches or exceeds existing state-of-the-art methods on 7 benchmark datasets for binary classification and 3 reserve design problem instances.

5.1 Introduction

In the *Transductive Precision@k* problem, the training set and the unlabeled test set are given, and the task is to predict exactly k test instances as positives. The precision of these selected instances—the fraction of correct positive predictions—is the only measure of importance. Our work is motivated by the problem of designing conservation reserves for an endangered species. Suppose a geographical region is divided into equal-sized cells of land. The species is present in positive cells and absent in negative cells. To protect the species, we seek to purchase some cells (“a conservation reserve”), and we want as many of those as possible to be positive cells. Suppose we have conducted a field survey of publicly-owned land to collect a training set of cells. With a fixed budget sufficient to purchase k cells, we want to decide which k privately-owned (and un-surveyed) cells to buy. In this paper, we assume that all cells have the same price. This is an instance of the Transductive Precision@k problem. Other instances arise in information retrieval and digital advertising.

The standard approach to this problem is to first train a classifier or ranker on the training data and then threshold the predicted test scores to obtain the k top-ranked test instances. Any model that outputs continuous scores (e.g., an SVM) can be employed in this two-step process. Better results can often be obtained by bipartite ranking algorithms [16, 57, 70, 1, 55, 40, 36], which seek to minimize a ranking loss. Recent work focuses even more tightly on the top-ranked instances. The MPG algorithm [73] formulates the ranking problem as an adversarial game and can optimize several ranking measures. The Accuracy At The Top (AATP) algorithm [11] seeks to optimize the ranking quality for a specified top quantile of the training data. Maximizing accuracy on the top quantile is intractable, so AATP optimizes a relaxation of the original objective. However, none of the algorithms above explicitly considers the constraint of choosing k test instances in model training.

Unlike the ranking problems discussed so far, our problem is transductive, because we have the unlabeled test examples available. There is a substantial body of research on transductive classification [34, 62, 54, 41]. Most transductive classification algorithms are inspired by either the large margin principle or the clustering principle. The goal of these algorithms is to develop classifiers that will perform well on the entire test set. Some transductive classifiers [34, 41] have a parameter to specify the desired ratio of positive predictions. However, such parameter is mainly used for training a stable classifier, and the ratio not strongly enforced on the test set.

In this paper, we claim that the knowledge of k helps to learn a better model in terms of the measure of top-k precision and that the trained model should be constrained to output a selection of k test instances as positive. We call this constraint the *k-constraint*. The benefit of incorporating k into the learning process can be understood in three ways. First, the k-constraint greatly reduces the hypothesis space and thus reduces the structural risk of the trained model. Second, the algorithm can take advantage of the knowledge of k to jointly optimize the scoring model and the score threshold with respect to the k-constraint. As a comparison, a two-step method trains a ranker that is optimal by some standard, but the ranker together with the threshold may not be optimal for the selection task. Third, the selection of k test points is directly obtained through model training, instead of learning a general classifier or ranker as an intermediate step. Vapnik's principle [71] dictates that we should not solve a more difficult problem on the way to solving the problem of interest.

In this paper, we jointly train the model and determine the threshold to obtain exactly k test instances as positive. We seek a decision boundary that predicts exactly k positives and has high precision on the training data. The paper proceeds as follows. We start by identifying a deterministic relation between the precision@k measure and the accuracy of any classifier that satisfies the k-constraint. This suggests that the learning objective should maximize classifier accuracy subject to the k-constraint. We adopt the space of linear decision boundaries and introduce an algorithm we call Transductive optimization of Top k precision (TTK). In the TTK optimization problem, the objective is to minimize the hinge loss on the training set subject to the k-constraint. This optimization problem is very challenging since it is highly non-convex. We first formulate this problem into an equivalent Mixed Integer Programming (MIP) problem and solve it with an off-the-shelf MIP solver. This method works for small problems. To solve larger problems, we also design a *feasible direction* algorithm, which we find experimentally to converge very rapidly. Finally, our theoretical analysis of the transductive precision@k problem shows that one should train different scoring functions for different values of k. As a byproduct of the work, We also find a problem in the optimization algorithm used in AATP, which solves a problem similar to ours.

In the experiment section, we first present a small synthetic dataset to show how the TTK algorithm improves the SVM decision boundary. Then, we show that our feasible direction method can find solutions nearly optimal as global optimal solutions. In the third part, we compare the TTK algorithm with five other algorithms on ten datasets. The results show that the TTK algorithm matches or exceeds the performance of these state-of-the-art algorithms on almost all of these datasets.

5.2 The TTK model

Let the distribution of the data be $\mathcal{D}$ with support in $\mathcal{X} \times \mathcal{Y}$. In this work, we assume $\mathcal{X} = \mathcal{R}^d$ and only consider the binary classification problem with $\mathcal{Y} = \{-1, 1\}$. By sampling from $\mathcal{D}$ independently, a training set $(\mathbf{x}, \mathbf{y}) = (x_i, y_i)_{i=1}^n$ and a test set $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = (\hat{x}_j, \hat{y}_j)_{j=1}^m$ are obtained, but the labeling $\hat{\mathbf{y}}$ of the test set is unknown. The problem is to train a classifier and maximize the precision at k on the test set. The hypothesis space is $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$ (functions mapping from $\mathcal{X}$ to $\mathcal{Y}$). The hypothesis $h \in \mathcal{H}$ is evaluated by the measure precision@k.

When we seek the best classifier from $\mathcal{H}$ for selecting k instances from the test set $\hat{\mathbf{x}}$, we only consider classifiers satisfying the k-constraint, that is, these classifiers must be in the hypothesis space $\mathcal{H}_k(\hat{\mathbf{x}}) = \{h \in \mathcal{H} | \sum_{j=1}^m \mathcal{I}[h(\hat{x}_j) = 1] = k\}$, where $\mathcal{I}[\cdot]$ is 1 if its argument is true and 0 otherwise. All classifiers not predicting k positives on the test set are excluded from $\mathcal{H}_k$. Note that any two-step method essentially reaches a classifier in $\mathcal{H}_k(\hat{\mathbf{x}})$ by setting a threshold in the second step to select k test instances. With these methods, the model optimized at the first step may be optimal in the original task, however, the classifier obtained by thresholding is often not optimal within $\mathcal{H}_k$.

To maximize the precision of $h \in \mathcal{H}_k(\hat{\mathbf{x}})$ on the test set, we essentially need to maximize the classification accuracy of h. This can be seen by the following relation. Let m_- be the number of negative test instances, and let m_{tp}, m_{fp} and m_{tn} denote the number of true positives, false positives, and true negatives (respectively) on the test set as determined by h. Then the precision@k of h can be expressed as

$$\rho(h) = \frac{1}{k} m_{\text{tp}} = \frac{1}{k}(m_{\text{tn}} + k - m_-) = \frac{1}{2k}(m_{\text{tp}} + m_{\text{tn}} + k - m_-). \tag{5.1}$$

Since the number of negative test instances m_- is unknown but fixed, there is a deterministic relationship between the accuracy $(m_{\text{tp}} + m_{\text{tn}})/m$ and the precision@k on the test set. Hence, increasing classification accuracy directly increases the precision. This motivates us to maximize the accuracy of the classifier on the test set while respecting the k-constraint.

In this section, we develop a learning algorithm for linear classifiers and thus $\mathcal{H} = \{h : \mathcal{X} \mapsto \mathcal{Y}, h(x; w, b) = \text{sign}(w^\top x + b)\}$. Our learning objective is to minimize the (regularized) hinge loss on the training set, which is a convex upper bound of the zero-one loss. Together with the k-constraint, the optimization problem is

$$\begin{aligned} & \min_{w,b} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}[1 - y_i\,(w^\top x_i + b)]_+, \qquad (5.2)\\ s.t. \qquad & \sum_{j=1}^{m} \mathcal{I}[w^\top \hat{x}_j + b > 0] = k\ , \end{aligned}$$

where $[\cdot]_+ = \max(\cdot, 0)$ calculates the hinge loss on each instance. Due to the piece-wise constant function in the constraint, the problem is very hard to solve.

We relax the equality constraint to an inequality constraint and get the following optimization problem.

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}[1 - y_i\,(w^\top x_i + b)]_+, \\ s.t. \qquad &\sum_{j=1}^{m}\mathcal{I}[w^\top \hat{x}_j + b > 0] \le k\ . \end{aligned} \tag{5.3}$$

This relaxation generally does not change the solution to the optimization problem. If we neglect the constraint, then the solution that minimizes the objective will be an SVM. In our applications, there are typically significantly more than k positive test points, so the SVM will usually predict more than k positives. In that case, the inequality constraint will be active, and the relaxed optimization problem will give the same solution as the original problem [1].

Even with the relaxed constraint, the problem is still hard, because the feasible region is non-convex. We first express the problem as a Mixed Integer Program (MIP). Let G be a large constant and η be a binary vector of length m. Then we can write the optimization problem as

$$\begin{aligned} &\min_{w,b,\eta} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}[1 - y_i\,(w^\top x_i + b)]_+, \\ s.t. \qquad &w^\top \hat{x}_j + b \le \eta_j G,\ \ j = 1, \dots, m \\ &\sum_{j=1}^{m}\eta_j = k \\ &\eta_j \in \{0, 1\},\ \ j = 1, \dots, m \end{aligned} \tag{5.4}$$

Theorem 5.2.1 *Optimization problems* (5.3) *and* (5.4) *are equivalent.*

Proof 5.2.2 *Since the two problems have the same objective that involves (w, b) only, we just need to show that the two constraint sets of the two problems are equivalent in terms of (w, b). Suppose (w^1, b^1) is a solution of* (5.3)*, then at most k instances have positive scores. Let η^1 be a binary vector with length m. Set $\eta^1_j = 1$ if instance j has positive score, and set other entries*

[1] In the extreme case that many data points are nearly identical, the original problem may not have a solution while the relaxed problem always has one.

of η^1 *to get* $\sum_j \eta_j^1 = k$. *Then* (w^1, b^1, η^1) *is a solution of* (5.4). *If* (w^2, b^2, η^2) *is a solution of* (5.4)*, then at most* k *test instances get positive scores, and then* (w^2, b^2) *is a solution of* (5.3). □

For this MIP, a globally optimal solution can be found for small problem instances via the branch-and-bound method with an off-the-shelf package. We used Gurobi[53].

For large problem instances, finding the global optimum of the MIP is impractical. We propose to employ a *feasible direction* algorithm [7], which is an iterative algorithm designed for constrained optimization. In each iteration, it first finds a descending direction in the feasible direction cone and then calculates a step size to make a descending step that leads to an improved feasible solution. The feasible direction algorithm fits this problem well. Because the constraint is a polyhedron, a step in any direction within the feasible direction cone will generate a feasible solution provided that the step length is sufficiently small. Since our objective is convex but the constraint is highly non-convex, we want to avoid making descending steps along the constraint boundary in order to avoid local minima.

In each iteration, we first need to find a descending direction. The subgradient $(\nabla w, \nabla b)$ of the objective with respect to (w, b) is calculated as follows. Let $\xi_i = 1 - y_i\,(w^\top x_i + b)$ be the hinge loss on instance i. Then

$$\nabla w = w - C \sum_{i:\xi_i > 0} y_i x_i\,, \quad \nabla b = -C \sum_{i:\xi_i > 0} y_i. \tag{5.5}$$

We need to project the negative subgradient $(-\nabla w, -\nabla b)$ to a feasible direction to get a feasible descending direction. Let L, E, and R be the sets of test instances predicted to be positive, predicted to be exactly on the decision boundary, and predicted to be negative:

$$\begin{aligned} L &= \{j:\ w^\top \hat{x}_j + b > 0\}, \\ E &= \{j:\ w^\top \hat{x}_j + b = 0\}, \\ R &= \{j:\ w^\top \hat{x}_j + b < 0\}. \end{aligned}$$

With the constraint in (5.3), there can be at most k instances in L.

When (w, b) changes in a descending direction, no instance can move directly from L to R or from R to L without going though set E due to the continuity of its score. A feasible direction only allows (w, b) to moves no more than $k - |L|$ from E to L. Therefore, the feasible direction cone is

$$\mathcal{F} = \left\{ (d_w, d_b) : \sum_{j \in E} \mathcal{I}[\hat{x}_j^\top d_w + d_b > 0] + |L| \leq k \right\}. \tag{5.6}$$

To avoid that the descending direction moves exessive instances from E to L, we project the direction into the null space of a set $B \subseteq E$ of test instances, then the instances in B will stay in E. We also need to guarantee that no more than $k - |L|$ instances in $E \setminus B$ moves from E to L. Now the problem is how to find the set B.

We first sort the instances in E in descending order according to the value of $-\hat{x}_j^\top \nabla w - \nabla b$. Let $j' : 1 \leq j' \leq |E|$ re-index the instances in this order. To construct the set B, we start with $B = \emptyset$ and the initial direction $(d_w, d_b) = -(\nabla w, \nabla b)$. The starting index is $j_0 = 1$ if $|L| = k$, and $j_0 = 2$ if $|L| < k$. Then with index j' starting from j_0 and increasing, we consecutively put instance j' into B and project (d_w, d_b) into the null space of $\{(\hat{x}_{j^\circ}, 1) : j^\circ \in B\}$. We stop at $j' = j_1$ when the direction (d_w, d_b) decreases the scores of all the remaining instances in E. The final projected direction is denoted by $(d_w^\star, d_b^\star)$. The direction $(d_w^\star, d_b^\star)$ has non-positive inner product with all instances with indices from $j' = j_0$ to $j' = |E|$, so these instances will not move into the set L when (w, b) moves in that direction. Only when $|L| < k$, is the first instance allowed to move from E to L. It is easy to check that the final projected direction $(d_w^\star, d_b^\star)$ is in the feasible cone $\mathcal{F}$ and that it is a descending direction. This subgradient projection algorithm is summarized in Algorithm 1.

In this design, we have the following considerations. When $|L| < k$, the instance in E that has the largest inner product with the negative subgradient is allowed to enter set L. We allow at most one instance to move from E to L to reduce the chance that (w, b) hits the boundary. In the projecting iterations, instances with large inner products are selected first to reduce the number of projections.

Once a descending direction is chosen, we perform a line search to determine the step size. We first find the maximum step size α that guarantees the feasibility of the descending step. That is, no points in R will cross the decision boundary and enter L with the step length α.

$$\alpha = \min_{j \in R \,:\, \hat{x}_j^\top d_w^\star + d_b > 0} \frac{-(\hat{x}_j^\top w + b)}{\hat{x}_j^\top d_w^\star + d_b}. \tag{5.7}$$

Then we do a line search in $[0, 0.5\alpha]$ to find the best step length $\alpha^\star$. Note that the objective

Algorithm 1 Find a descending feasible direction

```
**Input:** subgradient $(\nabla w, \nabla b)$, instance set $\{\hat{x}_j : j \in E\}$, size $|L|$, $k$
**Output:** descending feasible direction $(d_w^\star, d_b^\star)$
Sort instances in $E$ in descending order according to $-\hat{x}_j^\top \nabla w - \nabla b$
Initialize $(d_w, d_b) = -(\nabla w, \nabla b)$, $B = \emptyset$
$j_0 = \min(k - |L|, 1) + 1$
for $j' = j_0$ to $|E|$ do
  if $\exists j'' : \ j' \leq j'' \leq |E|, \ \hat{x}_{j''}^\top d_w + d_b > 0$ then
    $B = B \cup \{j'\}$
    project $(d_w, d_b)$ into the null space of $(\hat{x}_B, \mathbf{1})$
  else
    break
  end if
end for
$d_w^\star = d_w$, $d_b^\star = d_b$
```

function is a convex piece-wise quadratic function, so we only need to check these elbow points plus a minimum between two elbow points to find the best step length. We omit the details. The shrinkage 0.5 of α reduces the chance of (w, b) hitting the boundary.

We initialize w by training a standard linear SVM (although any linear model can be used) and then initialize b to satisfy the k-positive constraint. This gives us a pair (w, b) that is a feasible solution to (5.3). Then (w, b) is updated in each iteration according to $(w, b) := (w, b) + \alpha^\star(d_w^\star, d_b^\star)$ until convergence.

We set the maximum number of iterations, T, to 500; the algorithm typically requires only 200-300 iterations to converge. In each iteration, the two most expensive calculations are computing the subgradient and projecting the negative subgradient. The first calculation requires $O(nd)$ operations, and the second one takes at most $O(ud^2)$ operations, where u is the largest size of E. The overall running time is the time of training an initial model plus $O(T(nd + ud^2))$.

Though the problem is highly non-convex, the proposed projected subgradient method is very effective in practice, which is indicated by the comparison between solutions obtained by this method and optimal solutions obtained by Gurobi in the experiment section.

The AATP algorithm [11] faces an optimization problem similar to (5.3) and uses a different relaxation to find an appoximate solution. Here we show that their relaxation is very loose. The AATP objective is equivalent to ours, and the difference is that the constraint is posed on the training set. Their constraint is that the top q quantile of training instances must receive positive scores and all others, negative scores. The AATP authors assume that the optimal decision boundary must go though a *single* training instance, so their relaxation of the optimization problem is constrained to require *one* instance to be on the decision boundary. However, their assumption is incorrect, since the optimal solution would put *multiple* instances on the boundary. So their relaxation is very loose, and their solutions classify much more than quantile q of the instances as positive. Our analysis is verified by the experiment results, which will be shown in the experiment section.

5.3 Analysis

Before presenting experiments, we first argue that different values of k require us, in general, to train different models. We work with the population distribution $\mathcal{D}$ instead of with samples, and we assume linear models. Suppose the distributions of positive instances and negative instances have probability measures μ_+ and μ_- defined on $\mathcal{R}^d$. The total distribution is a mixture of the two distributions, and it has measure $\mu = \lambda\mu_+ + (1-\lambda)\mu_-$ with $\lambda \in (0,1)$. The classifier (w,b) defines a positive region $R_{w,b} = \{x \in \mathcal{R}^d, w^\top x + b > 0\}$. Assume $\mu_+(R_{w,b})$ and $\mu_-(R_{w,b})$ are both differentiable with respect to (w,b). If we consider classifiers that classify fraction q of the instances as positive, then $\mu(R_{w,b}) = q$. The precision of the classifier will be $\lambda\mu_+(R_{w,b}) \;/\; q$. The optimal classifier is therefore

$$\begin{aligned} (w^\star,\ b^\star) \quad &= \quad \arg\max_{(w,b)} \quad \lambda\mu_+(R_{w,b}) \\ &s.t. \quad \lambda\mu_+(R_{w,b}) + (1-\lambda)\mu_-(R_{w,b}) = q. \end{aligned} \tag{5.8}$$

If we change q, we might hope that we do not need to modify $w^\star$ but instead can just change $b^\star$. However, this is unlikely to work.

Theorem 5.3.1 *If $(w^\star, b_1)$ and $(w^\star, b_2)$ are two optimal solutions for* (5.8) *with two different quantile values q_1 and q_2, then $\exists s_1, t_1, s_2, t_2, \in \mathbf{R}$,*

$$s_1 \frac{\partial\mu_+(R_{w^\star,b_1})}{\partial(w^\star, b_1)} = t_1 \frac{\partial\mu_-(R_{w^\star,b_1})}{\partial(w^\star, b_1)}, \tag{5.9}$$

$$s_2 \frac{\partial\mu_+(R_{w^\star,b_2})}{\partial(w^\star, b_2)} = t_2 \frac{\partial\mu_-(R_{w^\star,b_2})}{\partial(w^\star, b_2)}. \tag{5.10}$$

The proof follows directly from the KKT conditions. Note that (5.9) and (5.10) are two vector

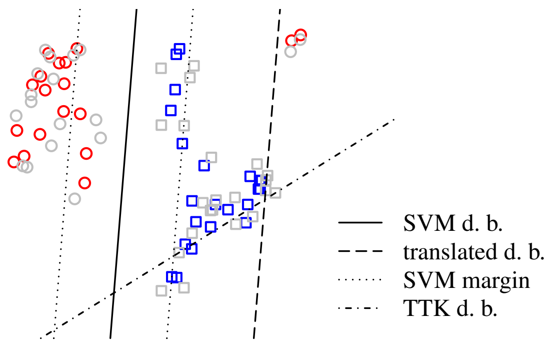

Figure 5.1: TTK improves the SVM decision boundary ("d. b."). Square/circle: positive/negative instance, colored/gray: training/testing instance. $k = 4$.

equations. When b_1 is changed into be b_2, the vectors of partial derivatives, $\partial\mu_+(R_{w^\star,b_1})/\partial(w^\star,b_1)$ and $\partial\mu_-(R_{w^\star,b_1})/\partial(w^\star,b_1)$ must change their directions in the same way to maintain optimality. This will only be possible for very special choices of μ_+ and μ_-. This suggests that $(w^\star,b^\star)$ should be optimized jointly to achieve each target quantile value q.

5.4 Experimental Tests

5.4.1 An illustrative synthetic dataset

We begin with a simple synthetic example to provide some intuition for how the TTK algorithm improves the SVM decision boundary, see Figure 5.1. The dataset consists of 40 training and 40 test instances. The training and testing sets each contain 22 positive and 18 negative instances. Our goal is to select $k = 4$ positive test instances. The bold line is the decision boundary of the SVM. It is an optimal linear classifier both for overall accuracy and for precision@k for $k = 24$. However, when we threshold the SVM score to select 4 test instances, this translates the decision boundary to the dashed line, which gives very poor precision of 0.5. This dashed line is the starting point of the TTK algorithm. After making feasible direction descent steps, TTK finds the solution shown by the dot-dash-dot line. The k test instances selected by this boundary are all positive. Notice that if $k = 24$, then the SVM decision boundary gives the optimal solution. This provides additional intuition for why the TTK algorithm should be rerun whenever we change the desired value of k.

Table 5.1: Training and test loss attained by different methods. The symbol "$+b_{adj}$" indicates that the bias term is adjusted to satisfy the k-constraint

dataset	value	SVM $+b_{adj}$	AATP $+b_{adj}$	TTK$_{\text{MIP}}$	TTK$_{\text{FD}}$
diabetes	train obj.	311 ± 25	265 ± 23	224 ± 7	226 ± 7
	test loss	323 ± 20	273 ± 24	235 ± 6	235 ± 5
ionosphere	train obj.	325 ± 46	474 ± 88	127 ± 4	136 ± 4
	test loss	338 ± 44	488 ± 85	146 ± 5	150 ± 7
sonar	train obj.	167 ± 52	166 ± 41	20 ± 8	30 ± 10
	test loss	216 ± 22	213 ± 30	103 ± 19	105 ± 24

Table 5.2: AATP and TTK solution statistics: Number of instances on the decision boundary ("# at d.b.") and fraction of instances predicted as positive ("fraction +")

dataset, dimension, ratio of positives	AATP		TTK$_{\text{MIP}}$	
	# at d.b.	fraction +	# at d.b.	fraction +
diabetes, $d = 8$, $n_+/n = 0.35$	1	0.12	5	0.05
ionosphere, $d = 33$, $n_+/n = 0.64$	1	0.53	21	0.05
sonar, $d = 60$, $n_+/n = 0.47$	1	0.46	40	0.05

5.4.2 Effectiveness of Optimization

One way to compare different algorithms is to see how well they optimize the training and test surrogate loss functions. We trained standard SVM, AATP, and TTK on three UCI [42] datasets: *diabetes*, *ionosphere* and *sonar*. The proposed TTK objective is solved by the MIP solver and the feasible direction method (denoted by TTK$_{\text{MIP}}$ and TTK$_{\text{FD}}$). We set k to select 5% of the test instances. For the SVM and AATP methods, we fit them to the training data and then obtain a top-k prediction by adjusting the intercept term b. We compare the regularized hinge loss on the training set and the hinge loss on the test set of each model after adjusting b, since the model with b adjusted is the true classifier used in the task. The hyper-parameter C is set to 1 for all methods. The results in Table 5.1 show that TTK with either solver obtains much lower losses than the competing methods. The small difference between the third (MIP) and the fourth (feasible direction) columns indicates that the feasible direction method finds near-optimal solutions.

The results also show that the AATP method does not minimize the objective very well. Due to its loose relaxation of the objective, the original AATP solution often predicts many more positives than the target quantile of 5% of the test instances. This requires to change the intercept

Table 5.3: Mean Precision (± 1 standard deviation) of classifiers when 5% of testing instances are predicted as positives.

dataset	SVM	TSVM	SVMperf	TopPush	AATP	TTK$_{\text{MIP}}$	TTK$_{\text{FD}}$
diabetes	**.86±.08**	**.86±.09**	.69±.20	**.80±.10**	.68±.28	**.85±.10**	**.86±.08**
ionosphere	.76±.13	.80±.17	.82±.22	.71±.16	**1.00±.00**	**.97±.05**	.84±.15
sonar	**.96±.08**	**.98±.06**	.85±.16	.88±.13	.90±.11	**.96±.08**	**1.00±.00**
german-numer	.70±.08	**.72±.08**	**.56±.17**	.63±.12	NA.	NA.	**.71±.06**
splice	**1.00±.00**	**1.00±.00**	**1.00±.01**	**1.00±.00**	NA.	NA.	**1.00±.00**
spambase	.97±.02	.97±.02	**.98±.01**	.96±.02	NA.	NA.	**.98±.01**
svmguide3	.86±.07	.85±.07	**.91±.04**	.83±.07	NA.	NA.	.87±.06
NY16	.64±.08	.64±.09	**.65±.12**	.62±.10	.62±.08	**.68±.07**	**.70±.09**
NY18	**.44±.11**	**.45±.10**	.36±.07	**.43±.13**	**.46±.12**	**.46±.08**	**.47±.12**
NY88	**.40±.08**	.33±.12	**.37±.15**	.34±.08	.31±.09	**.40±.09**	**.42±.07**
TTK$_{\text{MIP}}$ w/t/l	1/5/0	2/4/0	2/4/0	1/5/0	2/4/0		
TTK$_{\text{FD}}$ w/t/l	3/7/0	4/6/0	3/6/1	7/3/0	4/1/1		

term b to satisfy the k-constraint.

To further understand and compare the behavior of AATP and TTK, we performed a non-transductive experiment (by making the training and test sets identical). We measured the number of training instances that fall on the decision boundary and the fraction of training instances classified as positive (see Table 5.2). The optimal solution given by the MIP solver always puts multiple instances on the decision boundary, whereas the AATP method always puts a single instance on the boundary. The MIP always exactly achieves the desired k, whereas AATP always classifies many more than k instances as positive. This shows that the AATP assumption that the decision boundary should pass through exactly one training instance is wrong.

5.4.3 Precision evaluation on real-world datasets

In this subsection, we evaluate our TTK method on ten datasets. Seven datasets, {*diabetes, ionosphere, sonar, spambase, splice*} from UCI repository and {*german-numer, svmguide3*} from the LIBSVM web site, are widely studied binary classification datasets. The other three datasets, NY16, NY18 and NY88, are three species distribution datasets extracted from a large eBird dataset [65]; each of them has 634 instances and 38 features. The eBird dataset contains a large number of checklists of bird counts reported from birders around the world. Each checklist is associated with the position of the observation and a set of 38 features describing the habitat. We chose a subset of the data consisting of checklists of three species from New York state in June of 2012. To correct for spatial sampling bias, we formed spatial cells by imposing a grid over New York and combining all checklists reported within each grid cell. This gives 634 cells (instances). Each instance is labeled with whether a species was present or absent in the corresponding cell.

We compare the proposed TTK algorithm with 5 other algorithms. The SVM algorithm [59] is the baseline. The Transductive SVM (TSVM) [34] compared here uses the UniverSVM [63] implementation, which optimizes its objective with the convex-concave procedure. SVMperf [35] can optimize multiple ranking measures and is parameterized here to optimize precision@k. Two algorithms, Accuracy At The Top (AATP) [11] and TopPush [40], are specially designed for top precision optimization. Each algorithm is run 10 times on 10 random splits of each dataset. Each of these algorithms requires setting the regularization parameter C. This was done by performing five 2-fold internal cross-validation runs within each training set and selecting the value of C from the set $\{0.01, 0.1, 1, 10, 100\}$ that maximized precision on the top 5% of the (cross-validation) test points. With the chosen value of C, the algorithm was then run on the full training set (and unlabeled test set) and the precision on the top 5% was measured. The achieved precision values were then averaged across the 10 independent runs.

Table 5.3 shows the performance of the algorithms. For datasets with more than 1000 instances, the AATP and TTK_{MIP} algorithms do not finish within a practical amount of time, so results are not reported for these algorithms on those datasets. This is indicated in the table by "NA". The results for each pair of algorithms are compared by a paired-differences t-test at the $p < 0.05$ significance level. If one algorithm is not significantly worse than any of the other algorithms, then it is regarded as one the best and its performance is shown in bold face. Wins, ties and losses of of TTK_{MIP} and TTK_{FD} with respect to all other algorithms are reported in the last two rows of Table 5.3.

On each of the six small datasets, the performance of TTK_{MIP} matches or exceeds that of the other algorithms. The TTK_{FD} method does almost as well—it is among the best algorithms on 8 of the 10 datasets. It loses once to SVMperf (on svmguide3) and once to AATP (on ionosphere). None of the other methods performs as well. By comparing TTK_{FD} with SVM, we see that the performance is improved on almost all datasets, so the TTK_{FD} method can be viewed as a safe treatment of the SVM solution. As expected, the transductive SVM does not gain much advantage from the availability of the testing instances, because it seeks to optimize accuracy rather than precision@k. The TopPush algorithm is good at optimizing the precision of the very top instance. But when more positive instances are needed, the TopPush algorithm does not perform as well as TTK.

5.5 Summary

This paper introduced and studied the transductive precision@k problem, which is to train a model on a labeled training set and an unlabeled test set and then select a fixed number k of positive instances from the testing set. Most existing methods first train a scoring function and then adjust a threshold to select the top k test instances. We show that by learning the scoring function and the threshold together, we are able to achieve better results.

We presented the TTK method. The TTK objective is the same as the SVM objective, but TTK imposes the constraint that the learned model must select exactly k positive instances from the testing set. This constraint guarantees that the final classifier is optimized for its target task. The optimization problem is very challenging. We formulated it as a mixed integer program and solved it exactly via an MIP solver. We also designed a feasible direction algorithm for large problems. We compared both TTK algorithms to several state-of-the-art methods on ten datasets. The results indicate that the performance of the TTK methods matches or exceeds all of the other algorithms on most of these datasets.

Our analysis and experimental results show that the TTK objective is a step in the right direction. We believe that the performance can be further improved if we can minimize a tighter (possibly non-convex) bound on the zero-one loss.

Chapter 6: Conclusion

6.1 Conclusion

This thesis has studied various machine learning techniques to solve three representative problems in computational sustainability.

The first and second manuscripts have studied the superset label learning problem for detecting bird species from birdsong recordings. The first manuscript has proposed the new learning model, LSB-CMM, for the superset learning problem. Experiment results show that the proposed model performs well in the task of detecting bird species. The learnability analysis in the second manuscript can be applied directly to the problem of detecting bird species. We can check the bird song classification problem against the Conditional Independence Assumption (CIA) 3.4.4 and the No Co-occurring Label Condition (NCLC) 3.4.5. The CIA assumes that bird syllables are independent of each other given their species. This assumption is debatable: bird calls of different species tend to be independent of each other while bird calls of the same species may be correlated. The NCLC is satisfied in general, because any two species do not always co-occur in all possible short recordings. We can say that the problem is nearly learnable, and then the number of samples needed to train a good classifier would not be much more than the sample complexity given by 3.21. This analysis also points to a research direction of exploiting possible dependence among bird syllables to improve the classification performance.

The third manuscript has studied the inference problem of the Collective Graphical Model, which is used to model bird migration at the population level. This work approximates the CGM distribution of bird counts by a multivariate Gaussian distribution and then approximates CGM inference by inference with the Gaussian distribution. This work has also shown that the Gaussian approximation is the correct approximate distribution and that the approximate inference is computationally efficient. The proposed method of approximate inference solves the key inference problem in CGM modeling of bird migration. In this modeling problem, a Markov chain is assumed to be the migration model of individual birds, and the goal is to recover the parameters of the Markov chain by maximizing the likelihood of observed counts. With the counts of transitions among states introduced as hidden variables, the EM algorithm is employed to solve the MLE problem. While the M step is easy to solve, the E step requires inferring the mean of the transition counts. The proposed method approximately and efficiently solves this inference problem, though it needs slight adaption to handle Poisson noise in the observations. Synthetic experiments show that the EM algorithm can successfully recover the parameters of the individual migration model, which indirectly indicates that the mean of the transition counts inferred by the proposed method is accurate. The proposed method has also been applied to real bird migration data.

The fourth manuscript has proposed the formulation of transductive top k precision to model a reserve design problem. In this formulation, the input consists of a training set, a test set, and a number k, and the goal is to select k test instances and maximize the precision of the selection. Unlike traditional two-step methods, which first predict scores of the test instances with a trained model and then select the top k instances with a threshold, this new formulation aims to directly optimize the selection of k instances. This work then proposes the learning method, Transductive Top K (TTK), to optimize the selection of k test instances. By minimizing the hinge loss over the training set, TTK trains a classifier over a hypothesis space in which every classifier predicts exactly k instances as positive. The k positives predicted by the trained classifier form the final selection. In a typical reserve design problem, the training set consists of land parcels labeled with the presence/absence of the species, the test set consists of land parcels in the planning area, and the constraint is that at most k land parcels in the planning area can be purchased. All land parcels are described by feature vectors in the same way. The formulation of transductive top k precision fits this reserve design problem well, and the proposed learning method naturally applies. Experiment results show that the proposed learning method matches or improves the selection precision of traditional two-step learning methods on several tasks.

The learning techniques studied for the three problems can be generalized to other settings. Besides the superset label learning problem studied in this thesis, there are other forms of weak supervision in ecological data and the corresponding weakly supervised learning problems. The learning model and the learnability analysis in this thesis can be generalized to such problems. With the same principle of distribution approximation in this thesis, Gaussian distribution can be used to approximate the distribution of species counts in other modeling problems. The formulation of direct optimization of the selection of land parcels can be extended to problems with more complex selection constraints and multiple species. Therefore, the thesis will have contribution beyond the solutions of these three specific problems if the conducted research provides heuristics for solving similar problems in computational sustainability.

6.2 Future Work

Expert knowledge of ecologists has played important roles in the study of the three problems in this thesis. As end users of learning systems, ecologists generally provide their expert knowledge in three ways: providing data, giving advice for model definition, and verifying the outputs of learned models. In the bird song classification problem, the supervision information is the bird species labels given by experts. In the bird migration modeling, bird observations are from experts in the abstract meaning, and the model structure is defined with experts' insight. In the reserve design problem, the training labels are labeled by experts.

Ecologists need to interactively incorporate their expert knowledge into the learning system and improve its performance. With the traditional learning systems, the interactions between end users and learning systems are mediated by machine learning practitioners [4]. The high communication cost between end users and practitioners prevents efficient updates of learning models. Interactive machine learning [4] studies how to reduce the dependency on practitioners in the interactive learning process. Usually, end users are not familiar with machine learning techniques, so the objective is how to construct an interactive learning system that can actively learn from the expert knowledge as well as the data.

An efficient representation of expert knowledge is critical for the success of interactive learning systems applied in computational sustainability. Supervision information represented by data is a very inefficient representation of expert knowledge, because it is very costly for end users, especially busy scientists, to manually check and label data. Yu [76] and Stumpf et al. [64] propose to incorporate *rich feedback*, which is more expressive than labels, to the learning system. Stumpf et al. [64] study three forms, rule-based, keyword-based, and similarity-based, of rich feedback. There are other forms of rich feedback, such as qualitative monotonicities [2] and polyhedral knowledge sets [27]. With some representations, the end user still needs to intensively check the data to provide the knowledge.

Inspired by Morik et al. [47], this section proposes to represent expert knowledge with a knowledge base, and then use an Interactive Learning System Equipped with the Knowledge Base (ILSEKB) to learn from the expert knowledge and the data. Through the interactive learning process, the system needs to elicit knowledge from experts and put it into the knowledge base with appropriate representation. Then the ILSEKB needs to learn from both the data and the knowledge. From the perspective of learning from data, the system needs to use statistical random variables to model the distribution of the data and use the expert knowledge to restrict the relations among random variables. From the perspective of knowledge inference, the system needs to extract knowledge from the data and then do inference with the knowledge from both sources. After learning, the learning system needs to visualize its outputs in testing.

There are three advantages of ILSEKB compared to other interactive learning systems in applications of computational sustainability. First, it is relatively easy for experts to express their knowledge by rules and put it in the knowledge base. For example, ecologists can give a rule like "there are no Eastern Wood Peewee in December in Canada". Second, ILSEKB can efficiently elicit knowledge from experts through interactions. Through interactions with the learning system, the experts progressively provide knowledge to ILSEKB after they identify problems in its outputs. Ideally, the experts can efficiently express their knowledge, and ILSEKB can learn the knowledge effectively. Third, the knowledge base partially opens the black box of the learning process, therefore, the end user of ILSEKB can have better understanding and control of the learning system. Besides learning from the knowledge base, the learning system can also put the knowledge extracted from the data into the knowledge base, so the user of the learning system can scrutinize the system from both its outputs and extracted knowledge.

There are three issues to be considered to build an ILSEKB. The first issue is how to correspond the entities in the knowledge base to data items in the data. The learning system needs to find reference relations between entities in the knowledge base and data items to learn. One naive solution is to construct a vocabulary for all data items and restrict experts to use the fixed vocabulary to express their knowledge. The second issue is how to reconcile expert knowledge with the data. There might be conflicts or inconsistencies between the expert knowledge and the data. It is a good practice that the learning system gives feedback about such conflicts and inconsistencies. For example, the system can tell the expert that a rule is inconsistent with 5% of the data. Also, the learning system needs to resolve the conflict by either reducing the confidence of the knowledge or making some corrective treatment of the data. The third issue is how to construct the learning system from the data and the expert knowledge. The learning system invented by Morik et al. [47] extracts knowledge from the data and then infers actions in the knowledge space. The learning system can also use expert knowledge to edit the data, for example, re-weighting data points, and restricting the hypothesis space [76].

In the longer term, the ILSEKB can be a powerful learning framework if its knowledge base can be filled with other means besides experts' input. For example, different learning systems for different learning tasks can share knowledge via the knowledge base, which is one form of transfer learning [69]. The knowledge can also be accumulated in the knowledge base over many tasks and over a very long term, which is the research topic of lifelong machine learning. In the very ideal case, the knowledge base can also be updated with knowledge extracted from natural language, for example, scientific articles. Such an ILSEKB will be a big step forward in artificial intelligence.

Bibliography

[1] Shivani Agarwal. The Infinite Push: a New Support Vector Ranking Algorithm that Directly Optimizes Accuracy at the Absolute Top of the List. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 839–850, 2011.

[2] Eric E. Altendorf, Angelo C. Restificar, and Thomas G. Dietterich. Learning from Sparse Data by Exploiting Monotonicity Constraints. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 18–26, 2005.

[3] Mohamed Aly. Survey on Multiclass Classification Methods. Technical Report, 2005.

[4] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine*, 35(4):105–120, 2014.

[5] M. H. G. Anthony and N. Biggs. *Computational Learning Theory*. Cambridge University Press, 1997.

[6] Millennium Ecosystem Assessment. *Ecosystems and Human Well-being: Synthesis*. Island Press, Washington, DC, 2005.

[7] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley, 3rd edition, 2006.

[8] Shai Ben-David, Nicolò Cesa-Bianchi, David Haussler, and Philip M. Long. Characterizations of Learnability for Classes of $\{0, \ldots, n\}$-valued Functions. *Journal of Computer and System Sciences*, 50(1):74–86, 1995.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[10] Avrim Blum and Adam Kalai. A Note on Learning from Multiple-Instance Examples. *Machine Learning*, 30(1):23–29, 1998.

[11] Stephen Boyd, Corinna Cortes, Mehryar Mohri, and Ana Radovanovic. Accuracy at the Top. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 953–961. Curran Associates, Inc., 2012.

[12] T. Scott Brandes. Automated Sound Recording and Analysis Techniques for Bird Surveys and Conservation. *Bird Conservation International*, 18(S1), 2008.

[13] Forrest Briggs, Xiaoli Z. Fern, and Raviv Raich. Rank-loss Support Instance Machines for MIML Instance Annotation. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 534–542, 2012.

[14] Forrest Briggs, Raviv Raich, and Xiaoli Z. Fern. Audio Classification of Bird Species: a Statistical Manifold Approach. In *Proceedings of the Ninth International Conference on Data Mining*, pages 51–60, 2009.

[15] Peter F. Brussard, J. Michael Reed, and C. Richard Tracy. Ecosystem Management: What is it Really? *Landscape and Urban Planning*, 40(1-3):9–20, 1998.

[16] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 89 – 96, 2005.

[17] Sara Mc Carthy, Milind Tambe, Christopher Kiekintveld, Meredith L. Gore, and Alex Killion. Preventing Illegal Logging: Simultaneous Optimization of Resource Teams and Tactics for Security. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3880–3886, 2016.

[18] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

[19] Jon Conrad, Carla P. Gomes, Willem-Jan van Hoeve, Ashish Sabharwal, and Jordan F. Suter. Wildlife Corridors as a Connected Subgraph Problem. *Journal of Environmental Economics and Management*, 63:1–18, 2012.

[20] Timothee Cour, Ben Sapp, and Ben Taskar. Learning from Partial Labels. *Journal of Machine Learning Research*, 12(May):1501–1536, 2011.

[21] Timothee Cour, Benjamin Sapp, Chris Jordan, and Ben Taskar. Learning From Ambiguously Labeled Images. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 919–926, 2009.

[22] Amit Daniely, Sivan Sabato, Shai Ben-David, and Shai Shalev-Shwartz. Multiclass Learnability and the ERM Principle. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 207–232, 2011.

[23] A. P. Dawid and S. L. Lauritzen. Hyper Markov Laws in the Statistical Analysis of Decomposable Graphical Models. *The Annals of Statistics*, 21(3):1272–1317, 1993.

[24] Fei Fang, Thanh H. Nguyen, Rob Pickles, Wai Y. Lam, Gopalasamy R. Clements, Bo An, Amandeep Singh, and Milind Tambe. Deploying PAWS to Combat Poaching: Game-theoretic Patrolling in Areas with Complex Terrains (Demonstration). In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 4355–4356, 2016.

[25] William Feller. *An Introduction to Probability Theory and Its Applications, Vol. 2*. Wiley, 1968.

[26] Daniel Fink, Wesley M. Hochachka, Benjamin Zuckerberg, David W. Winkler, Ben Shaby, M. Arthur Munson, Giles Hooker, Mirek Riedewald, Daniel Sheldon, and Steve Kelling. Spatiotemporal Exploratory Models for Broad-Scale Survey Data. *Ecological Applications*, 20(8):2131–2147, 2010.

[27] Glenn M. Fung, Olvi L. Mangasarian, and Jude W. Shavlik. Knowledge-Based Support Vector Machine Classifiers. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 537–544. MIT Press, 2003.

[28] Carla Gomes. Computational Sustainability: Computational Methods for a Sustainable Environment, Economy, and Society. *National Academy of Engineering*, 39(4):5–13, 2009.

[29] Yves Grandvalet. Logistic Regression for Partial Labels. In *Proceedings of the Ninth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1935–1941, 2002.

[30] Robert G. Haight and Stephanie A. Snyder. Integer Programming Methods for Reserve Selection and Design. In A. Moilanen, K. A. Wilson, and H. P. Possingham, editors, *Spatial Conservation Prioritization - Quantitative Methods and Computational Tools*. Oxford University Press, Oxford, 2009.

[31] Eyke Hüllermeier and Jürgen Beringer. Learning from Ambiguously Labeled Examples. In *Proceedings of the Sixth International Symposium on Intelligent Data Analysis*, pages 168–179, 2005.

[32] Luo Jie and Francesco Orabona. Learning from Candidate Labeling Sets. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1504–1512. Curran Associates, Inc., 2010.

[33] Rong Jin and Zoubin Ghahramani. Learning with Multiple Labels. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 921–928. MIT Press, 2003.

[34] Thorsten Joachims. Transductive Inference for Text Classification Using Support Vector Machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209, 1999.

[35] Thorsten Joachims. A Support Vector Method for Multivariate Performance Measures. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 377 – 384, 2005.

[36] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Surrogate Functions for Maximizing Precision at the Top. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 189–198, 2015.

[37] Maxim Larrivée, Kathleen L. Prudic, Kent McFarland, and Jeremy T. Kerr. eButterfly: a Citizen-Based Butterfly Database in the Biological Sciences. `http://www.e-butterfly.org`, 2014.

[38] Steffen L. Lauritzen. *Graphical models*. Oxford University Press, 1996.

[39] Joona Lehtomäki and Atte Moilanen. Methods and Workflow for Spatial Conservation Prioritization Using Zonation. *Environmental Modelling & Software*, 47(Sep):128–137, 2013.

[40] Nan Li, Rong Jin, and Zhi-Hua Zhou. Top Rank Optimization in Linear Time. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1502–1510. Curran Associates, Inc., 2014.

[41] Yu-Feng Li, Ivor W. Tsang, James T. Kwok, and Zhi-Hua Zhou. Convex and Scalable Weakly Labeled SVMs. *Journal of Machine Learning Research*, 14(Jul):2151–2188, 2013.

[42] Moshe Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.

[43] Li-Ping Liu and Thomas G. Dietterich. A Conditional Multinomial Mixture Model for Superset Label Learning. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 548–556. Curran Associates, Inc., 2012.

[44] Po-Ling Loh and Martin J. Wainwright. Structure Estimation for Discrete Graphical Models: Generalized Covariance Matrices and their Inverses. *The Annals of Statistics*, 41(6):3022–3049, 2013.

[45] Philip M. Long and Lei Tan. PAC Learning Axis-Aligned Rectangles with Respect to Product Distributions from Multiple-Instance Examples. *Machine Learning*, 30(1):7–21, 1998.

[46] William K. Michener and James W. Brunt. *Ecological Data: Design, Management and Processing*. Wiley-Blackwell, 2000.

[47] Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining Statistical Learning with a Knowledge-Based Approach - a Case Study in Intensive Care Monitoring. In *Proceedings of the 16th International Conference on Machine Learning*, pages 268–277, 1999.

[48] Indraneel Mukherjee and Robert E. Schapire. A Theory of Multiclass Boosting. *Journal of Machine Learning Research*, 14(Feb):437–497, 2013.

[49] B. K. Natarajan. On Learning Sets and Functions. *Machine Learning*, 4(1):67–97, 1989.

[50] Lawrence Neal, Forrest Briggs, Raviv Raich, and Xiaoli Z.Fern. Time-Frequency Segmentation of Bird Song in Noisy Acoustic Environments. In *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2012–2015, 2011.

[51] Nam Nguyen and Rich Caruana. Classification with Partial Labels. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 551–559, 2008.

[52] Allan F. O'Connell, James D. Nichols, and K. Ullas Karanth. *Camera Traps in Animal Ecology: Methods and Analyses*. Springer, 2010.

[53] Gurobi Optimization. Gurobi Optimizer Reference Manual. `http://www.gurobi.com`, 2015.

[54] Dmitry Pechyony. *Theory and Practice of Transductive Learning*. PhD thesis, Department of Computer Science, Technion-Israel Institute of Technology, 2008.

[55] Alain Rakotomamonjy. Sparse Support Vector Infinite Push. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1335–1342, 2012.

[56] Lu Ren, Lan Du, Lawrence Carin, and David Dunson. Logistic Stick-Breaking Process. *Journal of Machine Learning Research*, 12(Jan):203–239, 2011.

[57] Cynthia Rudin. The P-Norm Push: A Simple Convex Ranking Algorithm That Concentrates at the Top of the List. *Journal of Machine Learning Research*, 10(Oct):2233–2271, 2009.

[58] Sivan Sabato and Naftali Tishby. Homogeneous Multi-Instance Learning with Arbitrary Dependence. In *Proceeding of the 22nd Annual Conference on Learning Theory*, pages 93–104, 2009.

[59] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[60] Daniel Sheldon and Thomas G. Dietterich. Collective Graphical Models. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1161–1169. Curran Associates, Inc., 2011.

[61] Daniel Sheldon, Tao Sun, Akshat Kumar, and Thomas G. Dietterich. Approximate Inference in Collective Graphical Models. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1004–1012, 2013.

[62] Vikas Sindhwani and S. Sathiya Keerthi. Large Scale Semi-Supervised Linear SVMs. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 477–484, 2006.

[63] F. Sinz and M. Roffilli. UniverSVM. `http://mloss.org/software/view/19/`, 2012.

[64] Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. Interacting Meaningfully with Machine Learning Systems: Three Experiments. *International Journal of Human-Computer Studies*, 67:639–662, 2009.

[65] Brian L. Sullivan, Christopher L. Wood, Marshall J. Iliff, Rick E. Bonney, and Daniel Finkand Steve Kelling. eBird: a Citizen-based Bird Observation Network in the Biological Sciences. *Biological Conservation*, 142:2282–2292, 2009.

[66] Rolf Sundberg. Some Results about Decomposable (or Markov-type) Models for Multidimensional Contingency Tables: Distribution of Marginals and Partitioning of Tests. *Scandinavian Journal of Statistics*, 2(2):71–79, 1975.

[67] Majid Alkaee Taleghan, Thomas G. Dietterich, Mark Crowley, Kim Hall, and H. Jo Albers. PAC Optimal MDP Planning for Ecosystem Management. *Journal of Machine Learning Research*, 16(Dec):3877–3903, 2015.

[68] Yee Whye Teh. Dirichlet processes. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning*, pages 280–287. Springer, 2010.

[69] Lisa Torrey and Jude Shavlik. Transfer Learning. In E. Soria, J. Martin, R. Magdalena, M. Martinez, and A. Serrano, editors, *Handbook of Research on Machine Learning Applications*. IGI Global, 2009.

[70] Nicolas Usunier, David Buffoni, and Patrick Gallinari. Ranking with Ordered Weighted Pairwise Classification. In *Proceedings of the 26th International Conference on Machine Learning*, pages 1057–1064, 2009.

[71] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[72] Martin J. Wainwright and Michael I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[73] Hong Wang, Wei Xing, Kaiser Asif, and Brian Ziebart. Adversarial Prediction Games for Multivariate Losses. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2728–2736. Curran Associates, Inc., 2015.

[74] Wei Wang and Zhi-Hua Zhou. Learnability of Multi-instance Multi-label Learning. *Chinese Science Bulletin*, 57(19):2488–2491, 2012.

[75] Alexander Ypma and Tom Heskes. Novel Approximations for Inference in Nonlinear Dynamical Systems Using Expectation Propagation. *Neurocomputing*, 69(1-3):85–99, 2005.

[76] Ting Yu. *Incorporating Prior Domain Knowledge into Inductive Machine Learning & Its Implementation in Contemporary Capital Markets*. PhD thesis, University of Technology Sydney, 2007.

[77] Min-Ling Zhang. Disambiguation-free Partial Label Learning. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 37–45, 2014.

[78] Zhi-Hua Zhou and Min-Ling Zhang. Multi-Instance Multi-Label Learning with Application to Scene Classification. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1609–1616. MIT Press, 2007.

[79] Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. Multi-instance Multi-label Learning. *Artificial Intelligence*, 176(1):2291–2320, 2012.

[80] Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool, 2009.

APPENDICES

Appendix A: Appendix for manuscript 1

A.1 Variational EM

The hidden variables in the model are y, z, and θ. For these hidden variables, we introduce the variational distribution $q(y, z, \theta|\hat{\phi}, \hat{\alpha})$, where $\hat{\phi} = \{\hat{\phi}_n\}_{n=1}^N$ and $\hat{\alpha} = \{\hat{\alpha}_k\}_{k=1}^K$ are the parameters. Then we factorize q as

$$q(z, y, \theta|\hat{\phi}, \hat{\alpha}) = \prod_{n=1}^{N} q(z_n, y_n|\hat{\phi}_n) \prod_{k=1}^{K} q(\theta_k|\hat{\alpha}_k), \tag{A.1}$$

where $\hat{\phi}_n$ is a $K \times L$ matrix and $q(z_n, y_n|\hat{\phi}_n)$ is a multinomial distribution in which $p(z_n = k, y_n = l) = \hat{\phi}_{nkl}$. This distribution is constrained by the candidate label set: if a label $l \notin Y_n$, then $\hat{\phi}_{nkl} = 0$ for any value of k. The distribution $q(\theta_k|\hat{\alpha}_k)$ is a Dirichlet distribution with parameter $\hat{\alpha}_k$.

With Jensen's inequality, the lower bound of the log likelihood is

$$\begin{aligned} LL &\geq E[\log p(z, y, \theta|\mathbf{x}, \mathbf{w}, \alpha)] - E[\log q(z, y, \theta|\hat{\phi}, \hat{\alpha})] + \log(p(\mathbf{w}|0, \Sigma)) \\ &= \sum_{n=1}^{N} E[\log p(z_n|\mathbf{x}_n, \mathbf{w})] + \sum_{k=1}^{K} E[\log p(\theta_k|\alpha)] + \sum_{n=1}^{N} E[\log p(y_n|z_n, \theta)] \\ &\quad - \sum_{n=1}^{N} E[\log q(y_n, z_n|\hat{\phi}_n)] - \sum_{k=1}^{K} E[\log q(\theta_k|\hat{\alpha}_k)] + \log(p(\mathbf{w}|0, \Sigma)), \end{aligned} \tag{A.2}$$

where $E[\cdot]$ is the expectation under the variational distribution $q(z, y, \theta|\hat{\phi}, \hat{\alpha})$.

Expand the expectation in the first, second and third term.

$$
\begin{aligned}
E[\log p(z_n|\mathbf{x}_n, \mathbf{w})] &= \sum_{k=1}^{K}\sum_{l=1}^{L} \hat{\phi}_{nkl} \log(\phi_{nk}), && \text{(A.3)} \\
E[\log p(y_n|z_n, \theta)] &= \sum_{k=1}^{K}\sum_{l=1}^{L} \hat{\phi}_{nkl} \int_{\theta_k} Dir(\theta_k; \hat{\alpha}_k) \log \theta_{kl} d\theta_k, && \text{(A.4)} \\
E[\log p(\theta_k|\alpha)] &\propto \int_{\theta_k} Dir(\theta_k; \hat{\alpha}_k) \sum_{l=1}^{L} (\alpha - 1) \log \theta_{kl} d\theta_k, && \text{(A.5)}
\end{aligned}
$$

where $Dir(\theta_k; \hat{\alpha}_k)$ is the density at θ_k of the Dirichlet distribution with $\hat{\alpha}_k$.

In the E step, this lower bound is maximized with respect to $\hat{\phi}$ and $\hat{\alpha}$. Each $\hat{\phi}_n$ can be optimized separately. Adding all terms involving $\hat{\phi}_n$ (i.e. the first, third and the fourth terms), we obtain

$$
\sum_{k=1}^{K}\sum_{l=1}^{L} \hat{\phi}_{nkl} \log\left(\phi_{nk} \exp(E_{q(\theta_k|\hat{\alpha}_k)}[\log(\theta_{kl})])\right) - \hat{\phi}_{nkl} \log(\hat{\phi}_{nkl}), \tag{A.6}
$$

Maximizing the term (A.6) is equivalent to minimizing the KL divergence between $\hat{\phi}_n$ and the term in the first logarithm function. With the constraint imposed by the candidate label set, the updating formula for $\hat{\phi}_n$ is (A.7). The update of $\hat{\alpha}_k$ for each k follows the standard procedure for variational inference in the exponential family and is shown in (A.8).

$$
\begin{aligned}
\hat{\phi}_{nkl} &\propto \begin{cases} \phi_{nk} \, \exp\left(E_{q(\theta_k|\hat{\alpha}_k)}\left[\log(\theta_{kl})\right]\right), & \text{if } l \in Y_n \\ 0, & \text{if } l \notin Y_n \end{cases} && \text{(A.7)} \\
\hat{\alpha}_k &= \alpha + \sum_{n=1}^{N} \hat{\phi}_{nkl} && \text{(A.8)}
\end{aligned}
$$

We calculate the expectation of $\log(\theta_{kl})$ via Monte Carlo sampling.

In the M step, the lower bound is maximized with respect to $\mathbf{w}$. Only the first and the last terms in the lower bound are related to $\mathbf{w}$, and each $\mathbf{w}_k, 1 \leq k \leq K-1$, can be maximized separately. After some derivation, we obtain the optimization problem in Eq. (A.9), which is similar to the problem of logistic regression. It is a concave maximization problem, so any gradient based method, such as BFGS, can find the global optimum.

$$\max_{\mathbf{w}_k} \quad -\frac{1}{2}\mathbf{w}_k^T \Sigma^{-1} \mathbf{w}_k + \sum_{n=1}^{N} \Big[\hat{\phi}_{nk} \log(\text{expit}(\mathbf{w}_k^T \mathbf{x}_n)) + \hat{\psi}_{nk} \log(1 - \text{expit}(\mathbf{w}_k^T \mathbf{x}_n))\Big], \quad \text{(A.9)}$$

where $\hat{\phi}_{nk} = \sum_{l=1}^{L} \hat{\phi}_{nkl}$ and $\hat{\psi}_{nk} = \sum_{j=k+1}^{K} \hat{\phi}_{nj}$.

A.2 Prediction

For a test instance $\mathbf{x}_t$, we predict the label with maximum posterior probability. The test instance can be mapped to a topic, but there is no coding matrix θ from the EM solution. We use the variational distribution $p(\theta_k|\hat{\alpha}_k)$ as the prior of each θ_k and integrate out all θ_ks. Given a test sample $\mathbf{x}_t$, the prediction l that maximizes the probability $p(y_t = l|\mathbf{x}_t, \mathbf{w}, \hat{\alpha})$ can be calculated as

$$\begin{aligned} p(y_t = l|\mathbf{x}_t, \mathbf{w}, \hat{\alpha}) &= \sum_{k=1}^{K} \int_{\theta_k} p(y_t = l, z_t = k, \theta_k|\mathbf{x}_t, \mathbf{w}, \hat{\alpha}) d\theta_k \\ &= \sum_{k=1}^{K} p(z_t = k|\mathbf{x}_t, \mathbf{w}) \int_{\theta_k} p(\theta_k|\hat{\alpha}_k) p(y_t = l|\theta_k) d\theta_k \\ &= \sum_{k=1}^{K} \phi_{tk} \frac{\hat{\alpha}_{kl}}{\sum_l \hat{\alpha}_{kl}} , \end{aligned} \quad \text{(A.10)}$$

where $\phi_{tk} = \Big(\text{expit}(\mathbf{w}_k^T \mathbf{x}_t) \prod_{i=1}^{k-1} (1 - \text{expit}(\mathbf{w}_i^T \mathbf{x}_t))\Big)$.

Appendix B: Appendix for manuscript 3

B.1 Proof of Proposition 4.2.1

The usual way of writing the CGM distribution is to replace $f(\mathbf{n};\boldsymbol{\theta})$ in Eq. (4.3) by

$$f'(\mathbf{n};\boldsymbol{\theta}) = \frac{\prod_{C\in\mathcal{C},i_C\in\mathcal{X}^{|C|}} \mu_C(i_C)^{\mathbf{n}_C(i_C)}}{\prod_{S\in\mathcal{S},i_S\in\mathcal{X}^{|S|}} \left(\mu_S(i_S)^{\mathbf{n}_S(i_S)}\right)^{\nu(S)}} \tag{B.1}$$

We will show that $f(\mathbf{n};\boldsymbol{\theta}) = f'(\mathbf{n};\boldsymbol{\theta})$ for any $\mathbf{n}$ such that $h(\mathbf{n}) > 0$ by showing that both descibe the probability of an ordered sample with sufficient statistics $\mathbf{n}$. Indeed, suppose there exists some ordered sample $\mathbf{X} = (\mathbf{x}^1,\ldots,\mathbf{x}^N)$ with sufficient statistics $\mathbf{n}$. Then it is clear from inspection of Eq. (4.3) and Eq. (B.1) that $f(\mathbf{n};\boldsymbol{\theta}) = \prod_{m=1}^{N} p(\mathbf{x}^m;\boldsymbol{\theta}) = f'(\mathbf{n};\boldsymbol{\theta})$ by the junction tree reparameterization of $p(\mathbf{x};\boldsymbol{\theta})$ [72]. It only remains to show that such an $\mathbf{X}$ exists whenever $h(\mathbf{n}) > 0$. This is exactly what was shown by Sheldon et al. [60]: for junction trees, the hard constraints of Eq. (4.4), which enforce local consistency on the integer count variables, are equivalent to the global consistency property that there exists some ordered sample $\mathbf{X}$ with sufficient statistics equal to $\mathbf{n}$. (Since these are integer count variables, the proof is quite different from the similar theorem that local consistency implies global consistency for marginal distributions.) We briefly note two interesting corollaries to this argument. First, by the same reasoning, *any* reparameterization of $p(\mathbf{x};\boldsymbol{\theta})$ that factors in the same way can be used to replace $f(\mathbf{n};\boldsymbol{\theta})$ in the CGM distribution. Second, we can see that the base measure $h(\mathbf{n})$ is exactly the *number of different ordered samples* with sufficient statistics equal to $\mathbf{n}$.

B.2 Proof of Theorem 4.3.1: Additional Details

Suppose $\{\mathbf{n}^N\}$ is a sequence of random vectors that converge in distribution to $\mathbf{n}$, and that $\mathbf{n}_A^N$, $\mathbf{n}_B^N$, and $\mathbf{n}_S^N$ are subvectors that satisfy

$$\mathbf{n}_A^N \perp\!\!\!\perp \mathbf{n}_B^N \mid \mathbf{n}_S^N \tag{B.2}$$

for all N. Let α, β, and γ be measurable sets in the appropriate spaces and define

$$\begin{aligned} z = \Pr(\mathbf{n}_A \in \alpha, \mathbf{n}_B \in \beta \mid \mathbf{n}_S \in \gamma) - \\ \Pr(\mathbf{n}_A \in \alpha \mid \mathbf{n}_S \in \gamma) \Pr(\mathbf{n}_B \in \beta \mid \mathbf{n}_S \in \gamma) \end{aligned} \tag{B.3}$$

Also let z^N be the same expression but with all instances of $\mathbf{n}$ replaced by $\mathbf{n}^N$ and note that $z^N = 0$ for all N by the assumed conditional independence property of Eq. (B.2). Because the sequence $\{\mathbf{n}^N\}$ converges in distribution to $\mathbf{n}$, we have convergence of each term in z^N to the corresponding term in z, which means that

$$z = \lim_{N\to\infty} z^N = \lim_{N\to\infty} 0 = 0,$$

so the conditional independence property of Eq. (B.2) also holds in the limit.

B.3 Proof of Theorem 4.3.3

We need to show $\mathbf{I}_A$ can be recovered from $\tilde{\mathbf{I}}_{A^+}$ with a linear function.

Suppose the last indicator variable in $\mathbf{I}_A$ is i_A^0, which corresponds to the setting that all nodes in A take value L. Let $\mathbf{I}'_A$ be a set of indicators which contains all entries in $\mathbf{I}_A$ but the last one i_A^0. Then $\mathbf{I}_A$ can be recovered from $\mathbf{I}'_A$ by the constraint $\sum_{i_A} \mathbf{I}_A(i_A) = 1$.

Now we only need to show that $\mathbf{I}'_A$ can be recovered from $\mathbf{I}_{A^+}$ linearly. We claim that there exists an invertible matrix $\mathbb{H}$ such that $\mathbb{H}\, \mathbf{I}'_A = \tilde{\mathbf{I}}_{A^+}$.

Showing the existence of $\mathbb{H}$. Let $\tilde{\mathbf{I}}_{A^+}(i_D)$ be the i_D entry of $\tilde{\mathbf{I}}_{A^+}$, which is for configuration i_D of clique $D, D \subseteq A$.

$$\tilde{\mathbf{I}}_{A^+}(i_D) \quad = \quad \sum_{i_{A\setminus D}} \mathbf{I}'_A(i_D, i_{A\setminus D}) \tag{B.4}$$

Since no nodes in D take value L by definition of $\tilde{\mathbf{I}}_D$, $(i_D, i_{A\setminus D})$ *cannot* be the missing entry i_A^0 of $\mathbf{I}'_A$, and the equation is always valid.

Showing that $\mathbb{H}$ is square. For each D, there are $(L-1)^{|D|}$ entries, and A has $\binom{|A|}{|D|}$ sub-cliques with size $|D|$. So $\tilde{\mathbf{I}}_{A^+}$ have overall $L^{|A|} - 1$ entries, which is the same as $\mathbf{I}'_A$. So $\mathbb{H}$ is a square matrix.

We view $\mathbf{I}'_A$ and $\tilde{\mathbf{I}}_{A^+}$ as matrices and each row is a indicator function of graph configurations.

Since no trivial linear combination of $\tilde{\mathbf{I}}_{A+}$ is a constant by the conclusion in Loh et al. [44], $\tilde{\mathbf{I}}_{A+}$ has linearly independent columns. Therefore, $\mathbb{H}$ must have full rank and $\mathbf{I}'_A$ must have linearly independent columns.