Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Hongyu Su, Aristides Gionis, Juho Rousu. Structured Prediction of Network Response. In *Proceedings of the* 31^{th} *International Conference on Machine Learning (ICML 2014)*, Beijing, China, 2014. JMLR W&CP volume 32:442-450, June 2014.

II Hongyu Su, Markus Heinonen, Juho Rousu. Multilabel Classification of Drug-like Molecules via Max-margin Conditional Random Fields. In *Proceedings of the* 5^{th} *International Conference on Pattern Recognition in Bioinformatics (PRIB 2010)*, Nijmegen, The Netherlands, 2010. Springer LNBI volume 6282:265-273, September 2010.

III Hongyu Su, Juho Rousu. Multi-task Drug Bioactivity Classification with Graph Labeling Ensembles. In *Proceedings of the* 6^{th} *International Conference on Pattern Recognition in Bioinformatics (PRIB 2011)*, Delft, The Netherlands, 2011. Springer LNBI volume 7035:157-167, November 2011.

IV Hongyu Su, Juho Rousu. Multilabel Classification through Random Graph Ensembles. *Machine Learning*, DOI:10.1007/s10994-014-5465-9, Published Online 26 Pages, September 2014.

V Mario Marchand, Hongyu Su, Emilie Morvant, Juho Rousu, John Shawe-Taylor. Multilabel Structured Output Learning with Random Spanning Trees of Max-Margin Markov Networks. In *Advances in Neural Information Processing Systems 26 (NIPS 2014)*, Accepted 9 Pages, December 2014.

Author's Contribution

Publication I: "Structured Prediction of Network Response"

Publication I presents a novel definition of the network response prediction problem, and proposes a new structured output learning algorithm to solve the problem. The proposed algorithm, SPIN, captures the contextual information and improves the state-of-the-art models in terms of the prediction performance.

The definition of the problem and the original modeling idea were jointly developed. The author made a major contribution to the design of the learning system and the optimization algorithm. The author implemented the learning system and the optimization algorithm. The author designed and performed the experiments and analyzed the results. The author worked jointly on writing the research article.

Publication II: "Multilabel Classification of Drug-like Molecules via Max-margin Conditional Random Fields"

Publication II presents a structured output learning algorithm for the multilabel molecular classification problem. The new algorithm incorporates the correlation between multiple output variables into the learning process. It outperforms the previous single-label classification approaches in terms of the prediction performance.

The original modeling idea was jointly developed. The author implemented the learning system. The author designed and performed the experiments and analyzed the results. The author worked jointly on writing the research article.

Publication III: "Multi-task Drug Bioactivity Classification with Graph Labeling Ensembles"

Publication III extends Publication II by applying majority vote to combine the predictions from a set of structured output learners built on a collection of random output graphs.

The learning strategy was jointly developed. The author made a major contribution to the design of the learning system and the optimization algorithm. The author implemented the algorithms, designed and performed the experiments, and analyzed the results. The author worked jointly on writing the research article.

Publication IV: "Multilabel Classification through Random Graph Ensembles"

Publication IV extends Publication III by introducing two aggregation techniques which perform inference before or after combining multiple structure output learners. Publication IV also presents a theoretical study that explains the performance of the proposed algorithms. The developed algorithms are evaluated on a collection of heterogeneous multilabel prediction problems.

The idea was jointly developed by the authors. The author designed and implemented the learning frameworks and the optimization algorithms. The author conducted the theoretical study which explains the performance of the proposed learning algorithms. The author designed and performed the experiments and analyzed the results. The author worked jointly on writing the research article.

Publication V: "Multilabel Structured Output Learning with Random Spanning Trees of Max-Margin Markov Networks"

Publication V is a major step forward of Publication IV by introducing a joint learning and inference framework, and developing rigorous theories to backup the algorithm.

The idea was jointly initialized by the authors. The author worked jointly on developing the theories and the proofs. The author made a major contribution to the design of the learning framework and the optimization algorithm. The author implemented the learning system and the

optimization algorithm. The author designed and performed the experiments, and analyzed the results. The author worked jointly on writing the research article.

1. Introduction

1.1 Scope of the Thesis

Machine learning, defined by Arthur Samuel in 1959 as "a field of study that gives computers the ability to learn without being explicitly programmed", has gained in popularity and become an active research field in computer science during the last few decades. Machine learning not only produces intelligent systems that generalize well from previously observed examples, but is also firmly rooted in statistical learning theory that establishes the conditions guaranteeing good generalization (Vapnik, 1998, 1999). Machine learning appears in many real world applications, to name but a few, ranking web pages in internet search (Richardson et al., 2006), spam filtering in email (Goodman and tau Yih, 2006), recommender systems for online shopping (Bell and Koren, 2007), and image and speech recognitions (Bengio, 2009). With the increasing availability of large scale datasets, machine learning is expected to play an indispensable role in many research fields (Fan and Bifet, 2013).

Supervised learning, an important paradigm in machine learning, is usually defined as learning a function that is capable of predicting the best value for an output variable given an input variable. The function is learned by exploring a set of observed input/output pairs known as training examples. In the classical supervised learning setting, there is only one variable to be predicted. This is called *single-label classification* if the output variable is discrete, or *regression* if the output variable is continuous. Many single-label classification models have been designed and applied in practice, for example, the Perceptron (Rosenblatt, 1958), Logistic Regression (Chen and Rosenfeld, 1999), and Support Vector Machines (Cortes and Vapnik, 1995).

Multilabel classification is a natural extension to single-label classification by defining multiple interdependent output variables associated with each input. These type of problems are prevalent in everyday life. For example, a movie can be classified as "sci-fi", "thriller" and "crime"; a news article can be categorized as "science", "drug discovery" and "genomics"; a gene can be associated with multiple functions in genomics research; a surveillance photo can be tagged with "car", "building" and "road". When multiple output variables are treated as a "flat" vector, the problem is often called *flat multilabel classification*. Flat multilabel classification is one branch of multilabel classification that has seen interest from the machine learning community (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010). As multiple output variables can be "on" and "off" simultaneously, various flat multilabel classification algorithms have been developed that aim to explore the correlation between multiple output variables in order to make accurate predictions. In particular, Tsoumakas and Katakis (2007) summarized the established flat multilabel classification algorithms into two categories, namely problem transformation (Zhang and Zhou, 2005; Read et al., 2009; Cheng and Hüllermeier, 2009) and algorithm adaptation (Schapire and Singer, 1999; Bian et al., 2012).

There exists another line of research in multilabel classification known as *structured output prediction* where a complex structure (*output graph*) is defined on multiple output variables to model dependencies in a more comprehensive way. *Hierarchical classification* is one type of structured output prediction in which the prediction needs to be reconciled along a pre-established hierarchical structure (Silla and Freitas, 2011). Hierarchical classification is usually applied to the problem in which hierarchical structure represents different levels of granularity. The hierarchy can be either a rooted tree such as in the document classification problem (Hao et al., 2007; Li et al., 2007; Rousu et al., 2006), or a directed acyclic graph (DAG) with parent-children relationships such as in the gene function prediction problem (Barutcuoglu et al., 2006). There exists a large body of work on hierarchical classification from the early approaches which use the hierarchical structure heuristically for preprocessing or post-processing (Koller and Sahami, 1997; Dumais and Chen, 2000; Liu et al., 2005; DeCoro et al., 2007) to the recent approaches which encodes the structure into the learning process (Cai and Hofmann, 2004; Cesa-bianchi et al., 2005; Rousu et al., 2006; Gopal et al., 2012).

Graph labeling is another type of structured output prediction in which

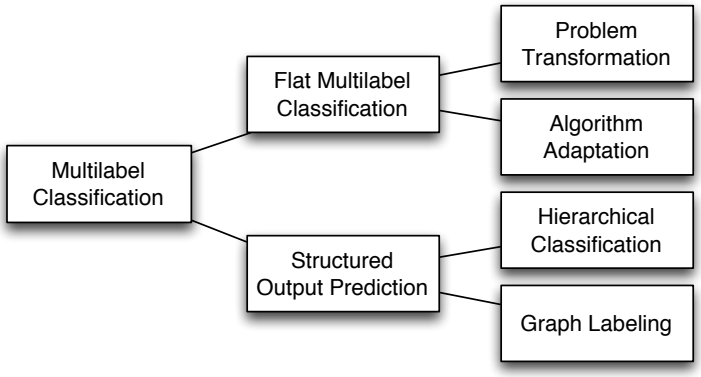


Figure 1.1. The taxonomy of the multilabel classification approaches.

the output graph often takes a more general form and does not require the concept of "level" compared to hierarchical classification. The approach can be applied to a wider range of problems, for example, speech tagging with sequence structure (Collins, 2002), and action recognition with Markov network structure (Wang and Mori, 2011). The graph labeling approach often directly incorporates the output graph into learning and exploits the correlation between labels to improve classification performance (Collins, 2002; Lafferty et al., 2001; Taskar et al., 2002, 2004; Tsochantaridis et al., 2004; Rousu et al., 2007). For graph labeling or structured output learning in general, one central problem is the output graph is assumed to be known *apriori*. However, this cannot be taken for granted as the proper dependency structure for the output variables is either hidden or difficult to retrieve in many applications (Chickering et al., 1994).

Figure 1.1 illustrates the taxonomy of multilabel classification. However, it is worth pointing out that there is no clear line drawn between different categories. In particular, some hierarchical classification models can also belong to the graph labeling category, for example, those developed in (Tsochantaridis et al., 2004; Rousu et al., 2006). As we focus on graph labeling in structured output prediction, we will explicitly use "structured output prediction" to refer to "graph labeling" throughout the thesis.

In this thesis, we extend the applicability of structured output learning by developing several new learning models and applying them to real world multilabel classification problems. In addition, we work on the problem of structured output learning when the output dependency structure is not observed. The models thus created are not restricted to the

availability of the output graph and can therefore be applied to a wider range of multilabel classification problems. We also investigate the efficiency and the scalability of the inference algorithms in the proposed structured output learning models. Finally, we study the theoretic aspect of the proposed models. The research questions can be summarized as follows.

- Should we tackle multilabel classification with structured output learning rather than flat multilabel classification?
- How to apply structured output learning to the multilabel classification problems when the output graph is not known *apriori*?
- Can we provide any theoretical studies to explain the behavior of the proposed learning models and to guarantee the performance?
- Can we efficiently solve the inference problems of the proposed structured output learning models?

1.2 Contributions and Organization

The contributions of the thesis are several novel statistical learning models that widen the applicability of structured output learning. The thesis starts by reviewing several lines of research in classification learning. The first contribution is to develop a new structured output learning model for the multilabel classification problem with an observed output graph. The proposed model can predict an optimal directed acyclic graph (DAG) from an observed underlying network which best responses to an input. The model has applied to network response prediction within the context of social network analysis. For the general multilabel classification problems in which the output graph is not known *apriori*, we develop several new models that combine a set of structured output learners built on a collection of random output graphs. In addition, we develop a joint learning and inference framework that is based on Max-Margin learning on a random sample of spanning trees. Thus, the proposed methods are not constrained by the availability of the output graphs. Moreover, we provide the theoretic studies which not only explain the intuition behind the formalisms but also guarantee the generalization error of the proposed models.

The remaining part of this thesis is structured as follows. Chapter 2 gives the background information to the learning problem in terms of classification, covering the basic concepts in classification learning including regularized risk minimization in Section 2.1, single-label classification in Section 2.2, and ensemble learning in Section 2.3. Chapter 3 introduces the multilabel classification problem which is the core problem under study in this thesis. The chapter also describes the flat multilabel classification approach which is a standard treatment for the multilabel classification problem. Chapter 4 and Chapter 5 present the main contributions of the thesis. In particular, Chapter 4 presents the structured output learning models developed for the multilabel classification problem with an observed output graph. The methods presented extend the flat multilabel classification approaches described in the previous chapter. Chapter 5 presents several models developed for structured output learning when the output graph is not observed. Chapter 6 describes the implementation details of the developed models. Chapter 7 concludes the thesis and details the future research directions.

This thesis presents the idea and the background of the proposed structured output learning models. The formalisms of the proposed models are also briefly explained. The notation and the presentation of some of the proposed models are slightly improved to incorporate the models into an unified framework. The technical details and the empirical evaluations of the proposed models are not repeated, rather, they can be found from the original research articles in the latter part of the thesis.

2. Regularized Learning for Classification

2.1 Regularized Risk Minimization

In this section, the author will introduce two fundamental concepts in statistical machine learning, known as *empirical risk minimization* (Vapnik, 1992) and *regularization* (Evgeniou et al., 1999) which create most learning algorithms presented in the following part of this thesis.

2.1.1 Empirical Risk Minimization

We assume that two random variables $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ and $y \in \mathcal{Y}$ are jointly distributed according to some fixed but unknown probability distribution $P(\mathbf{x}, y)$ over a domain $\boldsymbol{\mathcal{X}} \times \mathcal{Y}$, where $\boldsymbol{\mathcal{X}}$ is an input (instance) space and $\mathcal{Y}$ is an output (label) space. The definition of the output space $\mathcal{Y}$ will decide the type of the learning problem, for example, multiclass classification by setting $\mathcal{Y} = \{1, \cdots, K\}$, regression by setting $\mathcal{Y} = \mathbb{R}$ where $\mathbb{R}$ is a set of real numbers, binary classification by setting $\mathcal{Y} = \{-1, +1\}$, and multilabel binary classification by setting $\mathcal{Y} = \{-1, +1\}^k$. In addition, we are provided with paired examples $(\mathbf{x}, y) \in \boldsymbol{\mathcal{X}} \times \mathcal{Y}$ which are generated by sampling according to the distribution $P(\mathbf{x}, y)$. A *hypothesis class* $\mathcal{H}$ is a set of functions that a learning algorithm is allowed to search against. The goal of statistical learning is to provide an *estimator* $f \in \mathcal{H} : \boldsymbol{\mathcal{X}} \to \mathcal{Y}$ which predicts the best value of an output y given an input $\mathbf{x}$.

We use *loss function* $\mathcal{L}(y, f(\mathbf{x})) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ to measure the goodness of an estimator, which is a monotonic bounded function between a true value y and an estimated value $f(\mathbf{x})$. There are many ways to define the loss function including, for example, *Hinge Loss* in Support Vector Machines

(Cortes and Vapnik, 1995)

$$\mathcal{L}_{hinge}(y, f(\mathbf{x})) = \mathbf{max}(0, 1 - yf(\mathbf{x})),\ \mathcal{Y} = [-1, +1], \tag{2.1}$$

$0/1$ *Loss* in structured SVM (Tsochantaridis et al., 2004)

$$\mathcal{L}_{0/1}(y, f(\mathbf{x})) = \mathbf{1}_{\{y=f(\mathbf{x})\}},\ \mathcal{Y} = \{-1, +1\}^k, \tag{2.2}$$

Squared Loss in Ridge Regression (Hoerl and Kennard, 2000)

$$\mathcal{L}_{squared}(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2,\ \mathcal{Y} = \mathbb{R},$$

Exponential Loss in ADABOOST (Schapire and Singer, 1999)

$$\mathcal{L}_{exp}(y, f(\mathbf{x})) = \exp(-yf(\mathbf{x})),\ \mathcal{Y} = \mathbb{R}, \tag{2.3}$$

and *Logistic Loss* in Logistic Regression (Chen and Rosenfeld, 1999)

$$\mathcal{L}_{log}(y, f(\mathbf{x})) = \log(1 + \exp(-yf(\mathbf{x}))),\ \mathcal{Y} = [-1, +1]. \tag{2.4}$$

We will study the loss functions with the corresponding learning algorithms in detail in the following part of this thesis.

The *true risk* of an estimator f over all examples from a domain $\boldsymbol{\mathcal{X}} \times \mathcal{Y}$ is then defined as

$$\mathcal{R}(f) = \int_{(\mathbf{x},y)\in\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}} \mathcal{L}(y, f(\mathbf{x}))P(\mathbf{x}, y)\, d_{\mathbf{x}}d_y. \tag{2.5}$$

As a result, the learning algorithm should search for an estimator $f \in \mathcal{H}$ which minimizes the true risk (2.5). However, it is impossible to compute the true risk (2.5) directly, as the distribution $P(\mathbf{x}, y)$ is unknown. Instead we are given a random sample of m examples, denoted by $S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m)\}$, called *training data*. The *Empirical risk* of an estimator $f \in \mathcal{H}$ is defined as the average error made by the estimator on a training data S of a finite size

$$\mathcal{R}_{emp}(f) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(y_i, f(\mathbf{x}_i)). \tag{2.6}$$

This suggests that the learning algorithm should search for an estimator to minimize the empirical risk (2.6), which is called empirical risk minimization (Vapnik, 1992) in machine learning.

2.1.2 Regularized Learning

The empirical risk minimization strategy is ill-posed as it will provide an infinite number of estimators with the same empirical risk on the same

training data. Besides, it quite often leads to *overfitting*, in particular when the dimensionality of the feature space is high and the number of training examples is relatively small. That is, the underlying true distribution $P(\mathbf{x}, y)$ is difficult to estimate based on a finite sample of training examples. As a result, the estimator will generalize poorly on unseen test examples. Regularization theory (Evgeniou et al., 1999, 2002) provides a framework to tackle those two problems. In particular, it suggests to minimize

$$\mathcal{J}(f) = \mathcal{R}_{emp}(f) + \lambda\Omega(f), \tag{2.7}$$

where $\Omega(f)$ is a regularization function that controls the complexity of the estimator by penalizing the norm of the feature weight vector, λ is a positive parameter that controls the relative weight between the empirical risk term and the regularization term.

For the linear function class, there are several ways to define the regularization term including, for example, the L_1-norm and the L_2-norm regularizations. The L_2-norm regularization, defined by

$$\Omega_{L_2}(f) = ||\mathbf{w}||_2 = \left(\sum_{i=1}^{d} |\mathbf{w}[i]|^2\right)^{\frac{1}{2}}, \tag{2.8}$$

controls the complexity of the estimator f and provides a smooth solution. It has been applied in, for example, Ridge Regression (Hoerl and Kennard, 2000), Logistic Regression (Chen and Rosenfeld, 2000), and Support Vector Machines (Cortes and Vapnik, 1995). On the other hand, the L_1-norm regularization, defined by

$$\Omega_{L_1}(f) = ||\mathbf{w}||_1 = \sum_{i=1}^{d} |\mathbf{w}[i]|,$$

provides a sparse parameter estimation such that we obtain a high dimensional feature weight vector with many zero entries. This is an attractive property as *feature selection* is incorporated into the learning process. Therefore, the resulting model is usually easy to interpret. The L_1-norm regularization has been applied in, for example, LASSO (Tibshirani, 1994). Many other regularization techniques have been widely studied, for example, the $L_{1,2}$-norm regularization (Argyriou et al., 2007), and the elastic net regularization (Zou and Hastie, 2005).

2.2 Single-Label Classification

In this section, the author will introduce the basic classification problem known as single-label classification, and explain two prominent algorithms in this area, namely Logistic Regression and Support Vector Machines. Optimization techniques and the latest advances of these two algorithms will also be briefly discussed. The goal is to provide background information that is necessary to understand the algorithms presented in the latter part of this thesis.

2.2.1 Preliminaries

In this section, we focus on the standard supervised learning problem also known as *binary classification*, by explicitly assuming the output space $\mathcal{Y} = \{-1, +1\}$. Additionally, we assume a feature map $\boldsymbol{\varphi} : \boldsymbol{\mathcal{X}} \rightarrow \boldsymbol{\mathcal{F}}$, which embeds an input into some high dimensional feature space $\boldsymbol{\mathcal{F}} = \mathbb{R}^d$. In particular, $\boldsymbol{\varphi}(\mathbf{x})$ is a vector of real values in d dimensions. We consider the hypothesis class to be a set of *linear classifiers* that is parameterized by a weight vector $\mathbf{w}$ and a bias term b defined as

$$f(\mathbf{x}; \mathbf{w}, b) = \langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle + b, \tag{2.9}$$

where $\langle \cdot , \cdot \rangle$ denotes the inner product of two vectors

$$\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle = \sum_{i=1}^{d} \mathbf{w}[i] \, \boldsymbol{\varphi}(\mathbf{x})[i].$$

For any $1 \leq \rho \in \mathbb{R}$, the L_ρ-norm of a vector $\mathbf{w}$ is defined as

$$||\mathbf{w}||_\rho = \left(\sum_{i=1}^{d} |\mathbf{w}[i]|^\rho \right)^{\frac{1}{\rho}}.$$

For the convenience of presentation, we will explicitly use $||\mathbf{w}||$ to refer to the L_2-norm of $\mathbf{w}$ throughout the thesis.

2.2.2 Logistic Regression (LR)

Logistic Regression is a classification model rather than a regression model (Bishop, 2007). The formalism nicely transits from risk minimization (Section 2.1.2) to regularized risk minimization (Section 2.1.2). Logistic Regression has been extended to many other classification algorithms presented in the latter part of the thesis, for example, IBLR in Section 3.2.3, and CORRLOG in Section 3.3.2). The central idea of Logistic Regression,

the odd-ratio type learning in particular, is also the building block of M^{3}N in Section 4.2.3 and many other algorithms developed in this thesis.

Logistic Regression models the conditional probability $P(y = +1|\mathbf{x})$ for a binary output variable $y \in \mathcal{Y}$. To model the probability, we do not restrict to any particular form, as any unknown parameters can be estimated by *Maximum Likelihood Estimation* (MLE). However, we are most interested in a simple linear model as described in (2.9). To apply the linear model, we compute the logistic transformation of the original conditional probability by

$$\log \frac{P(y = +1|\mathbf{x})}{P(y = -1|\mathbf{x})} = \langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle + b.$$

Solving for $P(y = +1|\mathbf{x})$, we obain

$$P(y = +1|\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + e^{-\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle - b}}. \tag{2.10}$$

We can also compute

$$P(y = -1|\mathbf{x}; \mathbf{w}, b) = 1 - P(y = +1|\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + e^{\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle + b}}. \tag{2.11}$$

Putting (2.10) and (2.11) together, we define Logistic Regression as

Definition 1. *Logistic Regression (*LR*).*

$$P(y|\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + e^{-y(\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle - b)}}.$$

We predict $y = +1$ when $P(y = +1|\mathbf{x}; \mathbf{w}, b) \geq 0.5$, and $y = -1$ otherwise. The decision rule is such that we predict $y = +1$ when $\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle + b \geq 0$, and $y = -1$ otherwise. Besides the decision boundary, Logistic Regression can output the class probability of a data point as the the "distance" of the data point to the decision boundary. It is the probabilistic output that makes Logistic Regression no more than a classifier.

The parameter $\mathbf{w}$ and b can be obtained by maximizing the probability (likelihood) of the training data. The likelihood of parameters given data can be computed by

$$L(\mathbf{w}, b|D) = \prod_{i=1}^{m} P(y_i|\mathbf{x}_i). \tag{2.12}$$

To apply MLE, it is easier if, instead of maximizing the likelihood, we maximize the log-likelihood, which turns the product (2.12) into sum

$$\log L(\mathbf{w}, b|D) = \sum_{i=1}^{m} \log P(y_i|\mathbf{x}_i) = -\sum_{i=1}^{m} \log(1 + e^{-y_i(\langle \boldsymbol{\varphi}(\mathbf{x}_i), \mathbf{w} \rangle + b)}). \tag{2.13}$$

MLE can generate a LR model that fits the training data. However, there is no guarantee that the model also generalize well on the unseen test data. To achieve a better generalization power, we apply the regularization technique presented in Section 2.1.2. Many regularization methods for LR have been developed (Chen and Rosenfeld, 1999, 2000; Goodman, 2003) among which adding Gaussian prior on weight parameter w is a standard option. In practice, we assume w is generated according to a zero-mean spherical Gaussian with variance σ^2. Thus, the MLE problem (2.13) is transformed into the *Maximum A-Posteriori* (MAP) problem of the following form

$$P(\mathbf{w}, b|D; \sigma^2) = P(\mathbf{w}|\sigma^2) \prod_{i=1}^{m} P(y_i|\boldsymbol{\varphi}(\mathbf{x}_i)) = e^{-\frac{||\mathbf{w}||^2}{\sigma^2}} \prod_{i=1}^{m} \frac{1}{1 + e^{-y_i(\langle \boldsymbol{\varphi}(\mathbf{x}_i), \mathbf{w} \rangle + b)}}. \tag{2.14}$$

Instead of maximizing the posteriori (2.14), it is easier to maximize the log-posteriori

$$\log P(\mathbf{w}, b|D; \sigma^2) = -\frac{||\mathbf{w}||^2}{\sigma^2} - \sum_{i=1}^{m} \log\left(1 + e^{-y_i(\langle \boldsymbol{\varphi}(\mathbf{x}_i), \mathbf{w} \rangle + b)}\right). \tag{2.15}$$

In fact, (2.15) is an instantiation of the regularized risk minimization strategy described in (2.7) with the L_2-norm regularization (2.8) and the logistic loss (2.4).

Many optimization techniques have been proposed (Minka, 2003), for example, the itrative scaling method (Darroch and Ratcliff, 1972; Della Pietra et al., 1997; Berger, 1999; Goodman, 2002; Jin et al., 2003), the quasi-Newton method (Minka, 2003), the truncated Newton method (Komarek and Moore, 2005; Lin et al., 2008), and the coordinate descent method (Huang et al., 2009). There also exists a line of research that aims to optimize LR from the dual representation (Jaakkola and Haussler, 1999; Keerthi et al., 2005; Yu et al., 2011).

2.2.3 Support Vector Machines (SVM)

Support Vector Machines (SVM) is probably the most widely used single-label classification algorithm in machine learning. Its extensions for multilabel classification will be described in the latter part of the thesis (e.g., SSVM in Section 4.2.5). In this section, we first introduce *Maximum-Margin Principle* which is also the basis of many structured output learning models, for example, M^3N in Section 4.2.3, SPIN in Section 4.3, and RTA in Section 5.4. After that, we will discuss the formalism of SVM, the

primal-dual optimization strategy, and kernel methods which allow SVM to deal with the high dimensionality of the input feature space. In the end we will briefly present the optimization strategies developed for SVM.

The framework of SVM was originally introduced in (Cortes and Vapnik, 1995). The theory and the algorithm details of SVM are also presented in the book chapters (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Bishop, 2007). We begin our discussion by considering a very simple case where the training data is assumed to be linearly separable. There exists a *hyperplane* in the feature space which separates the training data into two classes. Additionally, we assume the separating hyperplane has a simple linear form (2.9) as

$$f(\mathbf{x}) = \langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle + b = 0.$$

As a result, we predict $y_i = +1$ if $f(\mathbf{x}_i) \geq 0$ and $y_i = -1$ otherwise. Given that a feature weight parameter $\mathbf{w}$ achieves a correct separation on the training data, we can decide the label of an unseen test example $\mathbf{x}_{ts}$ by the decision rule $y_{ts} = \mathbf{sign}(f(\mathbf{x}_{ts}))$.

There can be an infinite number of separating hyperplanes that solves the separation problem on the same training data, which is also suggested by the empirical risk minimization strategy presented in Section 2.1.1. We wish to find the hyperplane which also generalizes well on the test data. A good strategy is to look for a hyperplane that keeps the maximum distance from the examples of two classes, which is known as *Maximum-Margin Principle*. To see this, imagine putting a separating hyperplane close to one class of examples, which will achieve better classification performance for the test examples from the other class.

We further use γ_i to denote the *margin* of the i'th example defined as the geometric distance from the data point to the separating hyperplane

$$\gamma_i = \frac{y_i(\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}_i) \rangle + b)}{||\mathbf{w}||}.$$

We notice if $\mathbf{w}$ and b are scaled by any constant $\kappa \in \mathbb{R}$ (e.g., $\mathbf{w} \leftarrow \kappa\mathbf{w}, b \leftarrow \kappa b$) the margin γ_i stays unchanged. The same classification performance and generalization power can still be obtained. As the parameters are invariance to scaling, we set $||\mathbf{w}|| = 1$. Given a collection of training examples S, we define the margin with respect to S as the minimum margin achieved by an individual training example

$$\gamma = \min_{i \in \{1, \cdots, m\}} \gamma_i.$$

Based on the Maximum-Margin Principle, the goal of learning is to find the separating hyperplane such that it maximizes the margin with respect to all training examples while separating the training examples into two classes. This corresponds to finding a "big gap" between the examples of two classes in the feature space. The corresponding optimization problem is given in (Bishop, 2007) as

$$\begin{aligned} \max_{\mathbf{w},b,\gamma} \quad & \gamma \\ \textbf{s.t.} \quad & y_i(\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}_i)\rangle + b) \geq \gamma,\ ||\mathbf{w}|| = 1,\ \forall i \in \{1, \cdots, m\}. \end{aligned}$$

This is very difficult to optimize not only because the constraint $||\mathbf{w}|| = 1$ is non-convex, but also the optimization is not in any standard form. By replacing w with $\frac{\mathbf{w}}{\gamma}$, we obtain the following optimization problem

Definition 2. *Primal Hard-Margin* SVM *Optimization Problem.*

$$\begin{aligned} \boldsymbol{min}_{\mathbf{w}} \quad & \frac{1}{2}\,||\mathbf{w}||^2 \\ \boldsymbol{s.t.} \quad & y_i(\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}_i)\rangle + b) \geq 1, \forall i \in \{1, \cdots, m\}, \end{aligned}$$

where the goal is to find a weight vector of the minimum norm which corresponds to maximize the margin between the examples of two classes. The constraints state that the training examples should be correctly separated.

We do not use Definition 2 in practice for two reasons. First, many real world data is not linearly separable, where the solution to the optimization problem in Definition 2 does not always exist. Secondly, the data usually comes with noises and errors. We do not want the resulting classifier to over-fit the training data. Therefore, we relax the constraints by introducing a *margin slack* parameter ξ_i for each training example x_i and rewrite the constraints as

$$y_i(\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in \{1, \cdots, m\}. \tag{2.16}$$

ξ_i will allow data points to have a margin less than 1. In particular, with $\xi_i = 0$, the data point x_i is correctly classified, and lies either on the margin or on the correct side. With $0 < \xi_i \leq 1$, the data point is correctly classified, and lies between the margin and the separating hyperplane. With $\xi_i > 1$, the data point is misclassified locating on the other side of the separating hyperplane. Now the new goal is to maximize the margin while penalize the data points which either lie on the wrong side of the hyperplane or have a margin less than one. This can be defined by

Definition 3. *Primal Soft-Margin* SVM *Optimization Problem.*

$$\begin{aligned} \underset{\mathbf{w},\xi_i}{\boldsymbol{min}} \quad & \frac{1}{2}\,||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \\ \boldsymbol{s.t.} \quad & y_i(\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in \{1, \cdots, m\}. \end{aligned}$$

Definition 3 is an instantiation of the regularized risk minimization strategy (2.7) with the L_2-norm regularization (2.8) and the hinge loss (2.1). The optimization problem is usually transformed into a dual form by introducing for each constraint a *Lagrangian multiplier* (dual variable) α. We defined the dual optimization problem as

Definition 4. *Dual Soft-Margin* SVM *Optimization Problem.*

$$\begin{aligned} \underset{\alpha}{\boldsymbol{max}} \quad & \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{m}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{x}_j)\rangle \\ \boldsymbol{s.t.} \quad & \sum_{i=1}^{m}\alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C,\ \forall i \in \{1, \cdots, m\}. \end{aligned}$$

It is not difficult to verify that according to Karush-Kuhn Tucher (K.K.T) conditions only the examples with $\xi_i = 0$ and satisfying the equality constraints (2.16) will be "active", have a dual variable $\alpha_i > 0$ and lie on the margin with $\gamma_i = 1$. They are called *support vectors* during the optimization of SVM. In fact the number of support vectors is usually smaller than the size of the training data. As the weight vector can be expressed as a linear combination of training examples (Shawe-Taylor and Cristianini, 2004)

$$\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \boldsymbol{\varphi}(\mathbf{x}_i),$$

the evaluation can be done efficiently by maintaining a small set of non-zero dual variables.

To solve the optimization problem in Definition 4, we only need the result of the inner product $\langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{x}_j)\rangle$ rather than work explicitly in the feature space of $\boldsymbol{\varphi}(\mathbf{x})$. This suggests that training data can be radically represented through pairwise similarities. In particular, we defined a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that training data $\mathcal{S}$ is represented through a $m \times m$ matrix of pairwise similarities.

Definition 5. *Kernel Function. A function $K : \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{X}} \to \mathbb{R}$ is called a kernel if it is symmetric and positive semidefinite. That is,*

$$K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i) \quad \textit{and} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}} c_i c_j K(\mathbf{x}_i, \mathbf{x}_i) \geq 0$$

hold for any $c_i \in \mathbb{R}$ and a nonempty set $\mathcal{X}$.

Mercer's Theorem (Shawe-Taylor and Cristianini, 2004) has shown that every positive semidefinite symmetric function is a kernel.

Definition 6. *Reproduced Kernel Hilbert Space. For any kernel function K defined on a space $\mathcal{X}$, there always exists a Reproduced Kernel Hilbert Space (RKHS) $\boldsymbol{\mathcal{F}}$ and a mapping function $\boldsymbol{\varphi} : \mathcal{X} \to \boldsymbol{\mathcal{F}}$ such that $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{y}_i) \rangle$ holds for any $\mathbf{x} \in \mathcal{X}$.*

Definition 6 shows that any kernel function can be represented as an inner product in some feature space $\boldsymbol{\mathcal{F}}$. Kernel enables us to work in a high dimensional feature space $\boldsymbol{\mathcal{F}}$ without ever computing the exact coordinate or evaluating the inner product explicitly in that space. Instead, kernel can be computed based on the inner product in the original space $\mathcal{X}$.

Kernel functions that are heavily used in practice include *linear kernel*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle,$$

polynomial kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + b)^d,$$

where b is a bias term and d is the degree of polynomial, and *Gaussian kernel* (RBF)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right),$$

where σ is the Gaussian width parameter.

Definition 7. *Kernel Matrix. Given a kernel K and a nonempty set $\mathcal{S}$, an $m \times m$ matrix $\hat{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$ is called a Kernel matrix (Gram matrix) of kernel K with respect to $\mathcal{S}$.*

The kernel matrix is usually normalized by

$$\hat{K}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\hat{K}(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{\hat{K}(\mathbf{x}_i, \mathbf{x}_i)\hat{K}(\mathbf{x}_j, \mathbf{x}_j)}}$$

to make sure that all elements in the kernel matrix lie on a unit hypersphere.

The algorithms for solving the optimization problem of SVM have been intensively studied, for example, the "chunking" method (Vapnik, 1982; Pérez-Cruz et al., 2004), the decomposition method (Osuna et al., 1997; Joachims, 1998), Sequential Minimum Optimization (SMO) (Platt, 1998, 1999), and the "digesting" method (Decoste and Schölkopf, 2002). There

are some recent studies that aim to scale SVM learning on the large scale datasets, for example, representing the training data with a small set of landmark points (Pavlov et al., 2000; Boley and Cao, 2004; Yu et al., 2005; Zhang et al., 2008), the greedy method for basis selections (Keerthi et al., 2006), the online SVM solver (Bordes et al., 2005), approximating the objective function of SVM (Zhang et al., 2012; Le et al., 2013), and approximating the kernel matrix with a low-rank matrix (Smola and Schökopf, 2000; Fine and Scheinberg, 2002; Drineas and Mahoney, 2005; Si et al., 2014).

2.3 Ensemble Methods

Ensemble methods are general classification techniques in machine learning. The methods train several base classifiers and combine them in order to achieve the more accurate predictions. Most importantly, there is no requirement for base classifiers to be accurate as long as they perform better than random guessing. There are several variants of ensemble methods, to name but a few, bagging (Breiman, 1996a), boosting (Freund and Schapire, 1997; Schapire and Singer, 1999), stacking (Smyth and Wolpert, 1999), and Bayesian averaging (Freund et al., 2004). Ensemble methods have improved the classification performance when compared to their base learner counterpart, some of them are also supported with the theoretical analysis which guarantees the performance (Schapire et al., 1997; Koltchinskii and Panchenko, 2000; Cortes et al., 2014a,b). This section will be devoted to bagging and boosting as both methods are extensively studied and are quite relevant to this thesis.

Ensemble methods and their theories are primarily developed for single-label classification. The extensions for multilabel classification will be briefly presented in Section 3.3.1. Moreover, we will present several new learning algorithms in the latter part of the thesis, which are related to the ensemble methods presented in this section but with significant differences.

2.3.1 Preliminaries

In addition to the notations introduced in Section 2.2.1, we assume there is a hypothesis class $\mathcal{H}'$ where we generate weak/base hypotheses $f^t(x) \in \mathcal{H}'$. We use t to index the t'th weak hypothesis. Let $H(x)$ denote the ensem-

ble framework which combines multiple weak hypotheses and generate a stronger one. In many cases, no other information about $f^t(x)$ is available to $H(x)$ except that each weak hypothesis will take in a parameter x and generate an output $y \in \mathcal{Y}$.

2.3.2 Boosting

Boosting corresponds to a learning framework or a family of algorithms that takes in a weak classifier and tuning it into a strong one. We begin our discussion from the *concept class*. A *concept* is a boolean function over a domain $\boldsymbol{\mathcal{X}}$, and a *concept class* is a class of concepts. A concept class is *strongly learnable* if there exists a polynomial learning algorithm which achieves a high accuracy with a high probability for all concepts in the class. On the other hand, a concept class is *weakly learnable* if the learning algorithm achieves an arbitrary high accuracy where the only requirement is that the learning algorithm finds a function which performs better than the coin flipping. The concept of learnability was proposed in (Kearns and Valiant, 1989) together with the question whether the strong learnability and the weak learnability are equivalent which is known as the *hypothesis boosting problem*. Finding a weak learner which performs better than random guessing is easy in practice, but finding a strong learner is usually difficult. Schapire (1990) has proved that the two classes of learnability are equivalent which lays the foundation of the boosting algorithm that tunes a weak learning algorithm into a strong one.

Adaptive Boosting (ADABOOST) developed by Freund and Schapire (1997) is the very first practical boosting algorithm and is the most influential one. In addition, Schapire and Singer (1999) proposed a variant of the algorithm which updates the adaptive parameters in order to minimize the exponential loss (2.3) of each weak learner. The algorithm is shown in Algorithm 1. The central idea of ADABOOST is to maintain a distribution D over all training examples, and update the distribution in each iteration such that the difficult-to-classify examples will get more probability mass for the next iteration (line 7). Particularly, in each iteration, the algorithm computes a weak learner $f^t(\mathbf{x})$ base on all training examples and the current distribution D^t (line 3), calculates the weighted training error ϵ^t (line 4), and computes the adaptive parameter α^t (line 5). The ensemble prediction is the weighted combination of all weak learners (line 9).

For each weak learner $f^t(\mathbf{x})$, the strategy of updating the adaptive pa-

Algorithm 1 ADABOOST

Input: Training sample $\mathcal{S} = \{(x_i, \mathbf{y}_i)\}_{i=1}^m$, learning function $\mathcal{W}$, number of weak learners T

Output: Boosting ensemble $H(\mathbf{x})$

1: Initialize $D^0(i) = \frac{1}{m}, i \in \{1, \cdots, m\}$
2: **for** $t = 1 \cdots T$ **do**
3: $\quad f^t(\mathbf{x}) \leftarrow \mathcal{W}(\mathcal{S}, D^{i-1})$
4: $\quad \epsilon^t = \sum_{i=1}^m D^t(i)\mathbf{1}_{\{y_i \neq f^t(\mathbf{x}_i)\}}$
5: $\quad \alpha^t = \frac{1}{2}\ln\left(\frac{1-\epsilon^t}{\epsilon^t}\right)$
6: $\quad Z = \sum_{i=1}^m D^{t-1}(i)\exp(-\alpha^t y_i f^t(\mathbf{x}_i))$
7: $\quad D^t(i) = \frac{1}{Z} D^{t-1}(i)\exp(-\alpha^t y_i f^t(\mathbf{x}_i)), \forall i \in \{1, \cdots, m\}$
8: **end for**
9: **return** $H(\mathbf{x}) = \mathbf{sign}(\sum_{t=1}^T \alpha^t f^t(\mathbf{x}))$

rameter α^t is to ensure that the exponential loss of $\alpha^t f^t(\mathbf{x})$ is minimized. To see this, we first compute the exponential loss for $\alpha^t f^t(\mathbf{x})$ given the current distribution D^t and the adaptive parameter α^t

$$\begin{aligned}
&\mathcal{L}_{exp}(y, \alpha^t f^t(\mathbf{x}); D^t) \\
&\quad = \sum_{i=1}^m D^t(i)\exp(-y_i \alpha^t f^t(\mathbf{x}_i)) \\
&\quad = \exp(-\alpha^t)\sum_{i=1}^m D^t(i)\mathbf{1}_{\{y_i = f(\mathbf{x}_i)\}} + \exp(\alpha^t)\sum_{i=1}^m D^t(i)\mathbf{1}_{\{y_i \neq f(\mathbf{x}_i)\}} \\
&\quad = \exp(-\alpha^t)(1-\epsilon^t) + \exp(\alpha^t)\epsilon^t.
\end{aligned}$$

To minimize $\mathcal{L}_{exp}(y, \alpha^t f^t(\mathbf{x}); D^t)$, we take the partial derivative with respect to α^t and set it to zero

$$\frac{\partial \mathcal{L}_{exp}(y, \alpha^t f^t(\mathbf{x}); D^t)}{\partial \alpha^t} = -\exp(-\alpha^t)(1-\epsilon^t) + \exp(\alpha^t)\epsilon^t = 0.$$

Solve it for α^t, we get

$$\alpha^t = \frac{1}{2}\ln\left(\frac{1-\epsilon^t}{\epsilon^t}\right).$$

It is worth pointing out that ADABOOST described in (algorithm 1) requires the learning algorithm $\mathcal{W}$ work with some specific distribution defined on the training data. The distribution is usually generated by *reweighing* which initializes a uniform distribution over all training examples and updates the distribution in each iteration. For the learning algorithms that cannot work with distributions, *resampling* is often applied which generates a new training dataset in each iteration by sampling training examples according to some desired distribution.

DEEPBOOSTING, developed in (Cortes et al., 2014b), improves ADABOOST by allowing the base learning algorithm to use a complex hypothesis class. The theoretic analysis of DEEPBOOSTING also advances the previous performance guarantee of ADABOOST (Schapire et al., 1997; Koltchinskii and Panchenko, 2000).

2.3.3 Bootstrap Aggregating

Bootstrap Aggregating (BAGGING), proposed by Breiman (1996a), is an ensemble method that exploits the independency between weak learners. The algorithm is based on the fact that errors can be dramatically reduced by combining independent base learners. Let f_t denote the t'th weak learner. The ensemble prediction $H(\mathbf{x})$ is the averaged prediction over all weak learners

$$H(\mathbf{x}) = \mathbf{sign}\left(\sum_{t=1}^{T} f_t(\mathbf{x})\right). \tag{2.17}$$

We assume a base learner has a probability ϵ of making an independent mistake

$$P(f_t(\mathbf{x}) \neq y) = \epsilon.$$

As BAGGING (2.17) makes a mistake when at least half of the weak learners make mistakes, the probability of BAGGING making mistake can be computed by

$$P(H(\mathbf{x}) \neq y) = \sum_{t=0}^{T/2} \begin{pmatrix} \mathrm{T} \\ \mathrm{t} \end{pmatrix} (1-\epsilon)^t \epsilon^{T-t} \leq \exp\left(-\frac{1}{2}T(2\epsilon-1)^2\right).$$

The probability decreases exponentially in the number of weak learners, which will approach zero when the number of weak learners approaches infinity. However, it does not hold in practice as the base learners are generated from the same training data which can hardly be independent from each other. The goal of BAGGING is to best exploit the independency by adding randomness into the algorithm.

Bootstrap Sampling (Efron and Tibshirani, 1994) is applied in BAGGING to generate subsets of training examples. Given a training set of m training examples, a subset of the same size is generated by sampling with replacement m time from the original training set. The sampling procedure is repeated T times in order to generate T subsets for constructing base learners. Sampled subsets will be similar since they are sampled from the same training set. However, they will not be too similar in which

each subset will only cover around 63% of the origin training data under the condition that m is large. To see this, consider the probability that the i'th training examples is not sampled once is $(1 - \frac{1}{m})$, and the probability that it is not sampled at all is $(1 - \frac{1}{m})^m$. When m is large, this probability will approach 37%. That is, around 37% of the training examples will not appear in any sampled training set. The property of Bootstrap Sampling also allows us to efficiently estimate the generalization error of the base learner known as *out-of-bag estimation* (Breiman, 1996b; Tibshirani, 1996; Wolpert and Macready, 1999).

3. Multilabel Classification

Multilabel classification is a natural extension to single-label classification presented in Section 2.2. In multilabel classification, each input (instant) is simultaneously associated with multiple outputs (labels). The research of multilabel classification has progressed rapidly in the last two decades with many learning algorithms being developed and applied to the real world classification problems (Lafferty et al., 2001; Taskar et al., 2002, 2004; Tsochantaridis et al., 2004; Rousu et al., 2007). In general, there are two broad categories of research in multilabel classification, namely flat multilabel classification and structured output prediction. In flat multilabel classification, multiple interdependent labels are treated essentially as a "flat" vector of labels. Structured output prediction, on the other hand, models the correlation between multiple labels with an output graph that connects labels. This chapter will be devoted to flat multilabel classification in which several well-established algorithms will be presented. Structured output prediction will be covered in the latter part of the thesis.

It is prohibitive to explain all algorithms for flat multilabel classification, it is easier if we are able to categorize the algorithms into groups. In this chapter, we adopt the categorization scheme proposed in (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010) which gives us two major algorithm groups, namely problem transformation and algorithm adaptation. For the problem transformation group, we present the algorithms which aim to transfer the flat multilabel classification problem into other well studied problems, for example, single-label classification, and label ranking. For the algorithm adaptation group, we describe the algorithms which directly modify the established learning techniques (e.g., Logistic Regression, ensemble method) to solve the multilabel classification problem. Nevertheless, the presented algorithms aim to tackle the difficulties

of flat multilabel classification, namely the exponential sized multilabel space and the implicit correlation between multiple labels. It is impossible to cover every lines of research in the field of flat multilabel classification. Readers are pointed out to the recent research survey articles (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010; Zhang and Zhou, 2014).

3.1 Preliminaries

We borrow most notations from the single-label classification setting described in Section 2.2.1. In particular, we examine the following multilabel classification problem. We assume training examples are drawn from a domain $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$, where $\boldsymbol{\mathcal{X}}$ is a set of inputs (instants) and $\boldsymbol{\mathcal{Y}}$ is a set of outputs (multilabels). $\boldsymbol{\mathcal{Y}} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k$ is composed by a Cartesian product of k sets $\mathcal{Y}_i = \{-1, +1\}$. We retain a single-label classification problem by setting $k = 1$. A vector $\mathbf{y} = (y_1, \cdots, y_k) \in \boldsymbol{\mathcal{Y}}$ is called a *multilabel* and its element y_j is called a *microlabel*. We use $\mathbf{y}_i[j]$ to denote the j'th microlabel in the i'th multilabel. In addition, we are given a training set of m labeled examples $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m \in \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$. A pair $(\mathbf{x}_i, \mathbf{y})$, where $\mathbf{x}_i$ is a training input and $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$ is an arbitrary output, is called *pseudo-example*. It is worth pointing out that the pseudo-example $(\mathbf{x}_i, \mathbf{y})$ can be generated from a different distribution that generates training examples $(\mathbf{x}_i, \mathbf{y}_i)$. The goal of learning is to find a mapping function $f \in \mathcal{H} : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ which can compute the best multilabel for an input example such that the predefined loss ℓ for the unseen examples will be minimized.

3.2 Problem Transformation

The problem transformation approaches aim to transform the flat multilabel classification problem into other well studied problems. The most typical way is known as *binary relevance* (BR) (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010), which amounts to transforming a multilabel classification problem into a set of single-label classification problems and to independently learning a single-label classifier for each subproblem. There are many other type of transformations, for example, into the label power set problem (Tsoumakas and Vlahavas, 2007), and into the label ranking problem (Elisseeff and Weston, 2002; Brinker and Hüller-

meier, 2007; Fürnkranz et al., 2008; Chiang et al., 2012). However, learning by label ranking will not be explained in detail as it slightly diverges from the main scope of the thesis. We will focus on BR in this section.

3.2.1 Multilabel K-Nearest Neighbors (ML-KNN)

Multilabel K-Nearest Neightbors (ML-KNN) developed in (Zhang and Zhou, 2005, 2007) is perhaps the most famous binary relevance classifier for flat multilabel classification. ML-KNN is also an instance based learning approach (Aha et al., 1991) which is derived from the K-Nearest Neightbor (KNN) algorithm designed for single-label classification. ML-KNN transforms the flat multilabel classification problem into a set of single-label classification problems and processes each microlabel independently. For each unseen example $\mathbf{x}$, ML-KNN first identifies a set of K-nearest neighbors $N(\mathbf{x})$ from the training set. After that, the algorithm predicts the multilabel $\mathbf{y}$ by examining the set of labels collected from its K-nearest neightbors.

Mathematically, let $C_{\mathbf{x}}(j)$ denote the number of neighbors of $\mathbf{x}$ with the j'th label being "$+1$", let $H^b_{\mathbf{x}}(j)$ denote the event that the j'th label of $\mathbf{x}$ is $b \in \mathcal{Y}_j$, and let $E^l_{\mathbf{x}}(j)$ denote the event that $0 \leq l \leq K$ neighbors of x have the j'th label being "$+1$". ML-KNN processes each microlabel at a time and determines the value of the j'th microlabel by examining the following Maximize A-Posteriori (MAP) problem

$$\mathbf{y}[j]^* = \underset{b \in \mathcal{Y}_j}{\mathbf{argmax}}\, P(H^b_{\mathbf{x}}(j) | E^{C_{\mathbf{x}(j)}}_{\mathbf{x}}(j)) = \underset{b \in \mathcal{Y}_j}{\mathbf{argmax}}\, \frac{P(H^b_{\mathbf{x}}(j)) P(E^{C_{\mathbf{x}(j)}}_{\mathbf{x}}(j)) | H^b_{\mathbf{x}}(j))}{P(E^{C_{\mathbf{x}(j)}}_{\mathbf{x}}(j))}.$$

The prior probability distribution $H^b_{\mathbf{x}}(j)$ and the likelihood distribution $P(H^b_{\mathbf{x}}(j) | E^{C_{\mathbf{x}(j)}}_{\mathbf{x}}(j))$ can be estimated from the training data in terms of relative frequencies.

The central problem of ML-KNN is that the algorithm ignores the correlation between labels. Cheng and Hüllermeier (2009); Younes et al. (2011) proposed the variants of ML-KNN that aim to explore the label correlations. In addition, there exists many alternatives which also align to the direction of KNN typed learning for flat multilabel classification (Brinker and Hüllermeier, 2007; Chiang et al., 2012).

3.2.2 Classifier Chains (CC)

Classification Chains (CC) developed in (Read et al., 2009, 2011) is another problem transformation approach for flat multilabel classification.

CC involves k binary transformations and forms a chain of k binary classifiers $h = \{h_1, \cdots, h_k\}$, in which the j'th classifier h_j is built for predicting the j'th microlabel. For the j'th microlabel $\mathbf{y}[j]$, CC first constructs a new training data S_j by taking the j'th microlabel as the output variable and combing the original feature space with all $j-1$ prior microlabels as the new input features defined by

$$S_j = \{((\mathbf{x}_i[1], \cdots, \mathbf{x}_i[d], \mathbf{y}_i[1], \cdots, \mathbf{y}_i[j-1]), \mathbf{y}_i[j])\}_{i=1}^{m}.$$

A classifier h_j is built by applying any single-label classification algorithm on S_j.

Thus, CC takes the correlation between labels into consideration by incorporating the label information as the concatenated features in the new input feature space. The idea is not new which has been previously applied in (Godbole and Sarawagi, 2004). CC makes a strong assumption that there is a high correlation between the output microlabel and the concatenated microlabels. The central problem of CC is that the additional label information only takes a small part of the input feature space especially when the dimension of the original feature space is already high.

Probabilistic Classifier Chains (PCC) extends CC by analyzing the algorithm with the condition probability theory (Read et al., 2009; Dembczynski et al., 2010). In addition, Ensemble Classifier Chains (ECC) has been developed which improves CC by generating and combining multiple chains of classifiers (Read et al., 2011).

3.2.3 Instant Based Logistic Regression (IBLR)

Cheng and Hüllermeier (2009) developed Instance Based Logistic Regression (IBLR) with an extension to flat multilabel classification. IBLR is also an instant base learning approach (Aha et al., 1991) that is similar to ML-KNN. It extends ML-KNN by exploring the correlation between labels within the neighbors of an instant for posterior inference. The central idea of IBLR is to take the labels of the examples in the neighbor as the only features to predict the label of the current example. Similar ideas have been applied in collective classification (Ghamrawi and McCallum, 2005) and link based classification (Getoor, 2005; Getoor and Taskar, 2007).

In particular, for each unseen example $\mathbf{x}$, IBLR first identifies the set of K-nearest neighbors $N_k(\mathbf{x})$ from the training data. The algorithm builds a Logistic Regression model (Section 2.2.2) for each microlabel based on

the label information collected from the examples in $N_k(\mathbf{x})$. Mathematically, IBLR defines a posterior probability of the j'th microlabel of $\mathbf{x}$ being labeled as u_j by

$$\pi^{(j)} = P(\mathbf{y}(\mathbf{x})[j] = u_j | N_k(\mathbf{x})), u_j \in \mathcal{Y}_j.$$

It constructs a Logistic Regression classifier for $\pi^{(j)}$ which can be derived from the following

$$\log \frac{\pi^{(j)}}{1-\pi^{(j)}} = w_0^{(j)} + \sum_{i=1}^{k} \alpha_i^{(j)} \cdot w_i^{(j)}(\mathbf{x}),$$

where i iterates over all microlabels. $w_i^{(j)}(\mathbf{x})$ is defined by

$$w_i^{(j)}(\mathbf{x}) = \sum_{\mathbf{x}' \in N_k(\mathbf{x})} K(\mathbf{x}', \mathbf{x}) \cdot \mathbf{y}(\mathbf{x}')[i]$$

which collects the i'th microlabel from each neighbor $\mathbf{x}' \in N_k(\mathbf{x})$ and weights the microlabels according to the similarity between $\mathbf{x}$ and $\mathbf{x}'$ encoded in $K(\mathbf{x}', \mathbf{x})$. $\alpha_i^{(j)}$ is the regression coefficient.

3.3 Algorithm Adaptation

The idea of algorithm adaptation is to directly modify popular single-label classification algorithms to solve the multilabel classification problems. We will present the algorithms that are modified from ensemble methods and Logistic Regression. There also exists many other algorithms in the algorithm adaptation category, for example, the method based on label ranking (Crammer et al., 2003), and the method based on neural network (Zhang and Zhou, 2006). These methods are not explained in detail due to the divergence from the main scope of this thesis.

3.3.1 Ensemble Methods for Flat Multilabel Classification

Ensemble methods have been initially developed for single-label classification (Breiman, 1996a; Freund and Schapire, 1997) or regression (Breiman, 1996a), as it is straightforward to combine multiple scalar output variables. However, it is not immediately clear how to combine vector valued outputs in flat multilabel classification.

ADABOOST.MH is a multilabel variance of the ADABOOST algorithm (Schapire and Singer, 1999; Esuli et al., 2008). The core idea of ADABOOST.MH is to apply the hamming loss instead of the 0/1 loss. In particular, the

algorithm reduces a multilabel classification problem into a single-label classification problem by replacing each training example $(\mathbf{x}_i, \mathbf{y}_i)$ with k examples $\{(\mathbf{x}_i, \mathbf{y}_i[l])\}_{l=1}^{k}$. The algorithm is described in Algorithm 2. In particular, it maintains a distribution over all examples and labels. In each iteration, the algorithm takes in the distribution over all training examples, generates a weak learner $f^k(\mathbf{x})$ (line 3), computes the hamming loss (line 5), computes the adaptive parameter α^t (line 6), and update the distribution (line 8). The prediction $H(x)$ is a weighted combination of the base learners $f^t(x)$ weighted by the adapter parameters α^t.

Algorithm 2 ADABOOST.MH

Input: Training sample $\mathcal{S} = \{(x_i, \mathbf{y}_i)\}_{i=1}^{m}$, learning function $\mathcal{W}$, number of weak learners T

Output: Boosting ensemble $H(\mathbf{x})$

1: Initialize $D^0(i,l) = \frac{1}{mk}, i \in \{1, \cdots, m\}$
2: **for** $t = 1 \cdots T$ **do**
3: $\quad f^t(\mathbf{x}) \leftarrow \mathcal{W}(\mathcal{S}, D^{i-1})$
4: $\quad \mathbf{y}_i = f^t(\mathbf{x}_i), \forall i \in \{1, \cdots, m\}$
5: $\quad \epsilon^t = \sum_{l=1}^{k} \sum_{i=1}^{m} D^t(i,l) \mathbf{1}_{\{\mathbf{y}_i[l] \neq \hat{\mathbf{y}}_i[l]\}}$
6: $\quad \alpha^t = \frac{1}{2} \ln \left(\frac{1-\epsilon^t}{\epsilon^t} \right)$
7: $\quad Z = \sum_{i=1}^{m} \sum_{l=1}^{k} D^{t-1}(i,l) \exp(-\alpha^t \mathbf{y}_i[l] \hat{\mathbf{y}}_i[l])$
8: $\quad D^t(i,l) = \frac{1}{Z} D^{t-1}(i,l) \exp(-\alpha^t \mathbf{y}_i[l] \hat{\mathbf{y}}_i[l]), \forall i, \forall l$
9: **end for**
10: **return** $H(\mathbf{x}) = \mathbf{sign}(\sum_{t=1}^{T} \alpha^t f^t(\mathbf{x}))$

Besides ADABOOST.MH, some other ensemble methods for multilabel classification have also been developed that are based on boosting or bagging (Wang et al., 2007; Yan et al., 2007; Kocev et al., 2013). In addition, there is a large body of work which aim to apply ensemble methods to solve the real world multilabel classification problems, for example, natural language processing (Collins and Koo, 2005; Zeman and Žabokrtský, 2005; Sagae and Lavie, 2006; Zhang et al., 2009), and text and speech recognition (Fiscus, 1997; Mohri et al., 2008; Petrov, 2010).

3.3.2 Correlated Logistic Regression (CORRLOG)

Correlated Logistic Regression (CORRLOG) is a model based approach for flat multilabel classification (Bian et al., 2012). CORRLOG is a major step forward of IBLR by constructing a logistic regression classifier over all microlabels and by modeling the pairwise correlation of labels with a func-

tion defined on the microlabel pairs.

In fact, CORRLOG is derived from Independent Logistic Regression (ILRS). Given a pair of an arbitrary training example and a label $(\mathbf{x}, \mathbf{y})$, we can construct a set of ILRS classifiers, one for each microlabel. The posteriori probability can be computed by

$$P_{\text{ILRS}}(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{k} P_{\text{LR}}(\mathbf{y}[j]|\mathbf{x}) = \prod_{j=1}^{k} \frac{\exp(\mathbf{y}[j]\mathbf{w}^{\intercal}\mathbf{x})}{\exp(\mathbf{w}^{\intercal}\mathbf{x}) + \exp(-\mathbf{w}^{\intercal}\mathbf{x})}, \tag{3.1}$$

where j is the index that iterates over microlabels. The bias term as that appears in Definition 1 is omitted which is equivalent to augmenting x with a constant term (Bian et al., 2012). Otherwise, (3.1) can be derived into the same form of Definition 1 by replacing w with $\frac{\mathbf{w}}{2}$. ILRS has the problem of ignoring the correlation between labels and overfitting the training data when the number of microlabels is large. To alleviate the problems, CORRLOG augments the posteriori probability (3.1) by a function $Q(\mathbf{y})$ defined on the pairs of microlabels as

$$Q(\mathbf{y}) = \exp\left\{\sum_{k<j} \alpha_{k,j}\mathbf{y}[k]\mathbf{y}[j]\right\}. \tag{3.2}$$

Putting together (3.1) and (3.2), CORRLOG can be defined as

$$\begin{aligned} P_{\text{CORRLOG}}(\mathbf{y}|\mathbf{x}) &\propto P_{\text{ILRS}}(\mathbf{y}|\mathbf{x})Q(\mathbf{y}) \\ &= \exp\left\{\sum_{j=1}^{m} \mathbf{y}(x)[j]\mathbf{w}^{\intercal}\mathbf{x} + \sum_{k<j} \alpha_{kj}\mathbf{y}(\mathbf{x})[k]\mathbf{y}(\mathbf{x})[j]\right\}. \end{aligned}$$

Thus, CORRLOG examines the pairwise label correlations by augmenting the joint prediction with a quadratic term $Q(\mathbf{y})$ built from the pairs of microlabels.

3.3.3 Multitask Feature Learning (MTL)

Multitask Feature Learning (MTL) (Argyriou et al., 2007) is another algorithm designed for flat multilabel classification. MTL is quite different other flat multilabel classification algorithms discussed in the previous part. Specifically, MTL is based on the assumption that different microlabels are related such that they share a common underlying feature representation. Similar assumptions are also made in other models (Caruana, 1997; Baxter, 2000; Ben-David and Schuller, 2003).

Let $f_t(\mathbf{x})$ denote a label specific function for the t'th microlabel. $f_t(\mathbf{x})$ can

be expressed as

$$f_t(\mathbf{x}) = \langle \mathbf{a}_t, h(\mathbf{x}) \rangle = \sum_{i=1}^{d} \mathbf{a}_t[i] h(\mathbf{x})[i],$$

where $\mathbf{a}_t \in \mathbb{R}^d$ is the weight parameter for the t'th microlabel. $h(\mathbf{x})$ is a linear feature mapping function defined as

$$h(\mathbf{x}) = \langle U, \boldsymbol{\phi}(\mathbf{x}) \rangle,$$

where $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^d$ is in the original feature space and $U \in \mathbb{R}^{d \times d}$ is a square matrix. We further use A to denote the matrix composed by $\mathbf{a}_t$. MTL assumes that microlabels share a small set of features in which A is assumed to be sparse with many entries being zero. The optimization problem of MTL is defined as

Definition 8. MTL *Optimization Problem in Primal*

$$\underset{\substack{U \in \mathbb{R}^{d \times d} \\ A \in \mathbb{R}^{d \times T}}}{\boldsymbol{min}} \left\{ \sum_{t=1}^{T} \sum_{i=1}^{m} \ell(\mathbf{y}_{i,t}, \langle \mathbf{a}_t, \langle U, \boldsymbol{\phi}(\mathbf{x}_i) \rangle \rangle) + C \, ||A||_{2,1}^2 \right\},$$

where C is a positive parameter that controls the balance of the regularization term and the risk minimization term. The optimization problem is an instantiation of the regularized risk minimization (2.7) with the hamming loss and the $L_{2,1}$-norm regularization. As Definition 8 is non-convex and the second term is non-smooth, the optimization is transformed into an equivalent form which is solved by an alternative optimization approach (Argyriou et al., 2007).

Argyriou et al. (2008a) developed an extension of MTL that introduces a nonlinear generalization using kernel methods. In addition, Argyriou et al. (2008b); Jacob et al. (2009) have developed several similar but not identical algorithms based on the assumption that microlabels form clusters such that label specific weight vectors should be similar within the clusters. Recently, Romera-Paredes et al. (2012) proposed a method that exploits the information between unrelated microlabels based on a similar assumption that the microlabels of different groups tend not to share any features.

4. Structured Output Prediction

Structured output prediction is a natural extension to flat multilabel classification presented in Chapter 3. Unlike flat multilabel classification which takes multiple interdependent output variables essentially as a "flat" vector, structured output prediction assumes that multiple output variables are correlated and located in a structured output space. It assumes that there exists an output graph (e.g., a chain, a spanning tree) connecting multiple output variables by which the correlation between labels can be utilized during learning. In this chapter, we will start by introducing several structured output learning algorithms developed during the last decade. We will present our new algorithm SPIN that can predict an optimal directed acyclic graph (DAG) which best "responses" to an input, and examine the performance on the network response prediction problem within the context of social network analysis.

4.1 Preliminaries

Multilabel classification deals with multiple interdependent output variables, $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$. The problem is called structured output prediction when these variables are located in a structured output space. That is, the correlation between labels is described by an output graph connecting multiple labels. In particular, we define the output graph $G = (E, V)$ by a node set $V = \{1, \cdots, k\}$ which corresponds to the microlabels $\{y_1, \cdots, y_k\}$ and an edge set $E = V \times V$ which represents the correlation between pair of microlabels. For an edge $e = (j, j') \in E$, we use $\mathbf{y}_e$ to denote the label of the edge $\mathbf{e}$ with respect to a multilabel $\mathbf{y}$ by concatenating the head label y_j and the tail label $y_{j'}$, with an edge label domain $\mathbf{y}_e \in \boldsymbol{\mathcal{Y}}_e = \mathcal{Y}_j \times \mathcal{Y}_{j'}$. We use $\mathbf{y}_{i,e}$ to denote the edge label of an example $(\mathbf{x}_i, \mathbf{y}_i)$ on an edge $\mathbf{e}$. Thus, given a training example $(\mathbf{x}_i, \mathbf{y}_i)$ and an output graph G, we can

uniquely identify the node label y_i and the edge label $\mathbf{y}_{i,e}$ of the output graph. In addition, we denote the possible label of a node i by u_i and the possible label of an edge $\mathbf{e}$ by $\mathbf{u}_e$ where u_i and $\mathbf{u}_e$ are not constrained by any multilabel $\mathbf{y}$. Naturally, $u_i \in \mathcal{Y}_i$ and $\mathbf{u}_e \in \boldsymbol{\mathcal{Y}}_e$.

4.2 Related Methods

In this section, we will briefly present several related algorithms for structured output prediction including Structured Perceptron, Conditional Random Field, Max-Margin Conditional Random Fields, Structured SVM, and Max-Margin Markov Networks.

4.2.1 Structured Perceptron

The Perceptron developed by Rosenblatt (1958) is one of the oldest algorithms in machine learning. The structured Perceptron, as suggested by its name, is a generalization of the Perceptron algorithm to the structured output space. It was proposed in (Collins, 2002; Collins and Duffy, 2002) with the formalism quite similar to Multiclass Perceptron. The model assumes a score function $\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle$ as the inner product between a joint feature map $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ and a feature weight parameter $\mathbf{w}$. After the feature weight parameter $\mathbf{w}$ is obtained, one needs to solve the argmax problem to find the best output for a given input $\mathbf{x}$, which is defined as

$$\hat{\mathbf{y}} = \operatorname*{\mathbf{argmax}}_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle. \tag{4.1}$$

The argmax problem is solved in (Collins, 2002) by Viterbi decoding.

The weight parameter $\mathbf{w}$ is learned through the standard Perceptron iterative update by solving the argmax problem (4.1) in each iteration. In particular, the algorithm loops through all training examples and updates $\mathbf{w}$ whenever the predicted multilabel $\hat{\mathbf{y}}_i$ is different from the true multilabel $\mathbf{y}_i$. The update is defined by

$$\mathbf{w} \leftarrow \mathbf{w} + \left(\boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\phi}(\mathbf{x}_i, \hat{\mathbf{y}}_i) \right). \tag{4.2}$$

The update usually leads to over-fitting. A simple refinement is usually applied which is similar to Averaged Perceptron (Freund and Schapire, 1999).

The central problem with Structured Perceptron is the loss function. In fact, Structured Perceptron tacitly applies $0/1$ loss (2.2) on multilabels, with which it is impossible to distinguish a nearly correct multilabel and a

completely incorrect one. Both will lead to the same update to the feature weight parameter (4.2) during learning.

4.2.2 Conditional Random Field (CRF)

Condition Random Field (CRF) developed in (Lafferty et al., 2001; Taskar et al., 2002) is a discriminative framework that constructs a conditional probability $P(\mathbf{y}|\mathbf{x})$ for an input variable $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ and an output variables $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$. It optimizes the log-loss which is analogue to the $0/1$ loss in the structured output space.

Mathematically, let $Y = \{\mathbf{y}_1, \cdots, \mathbf{y}_m\}$ denote a set of output random variables and $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ denote a set of input random variables to condition on. Let $G = (E, V)$ denote an output graph such that $\mathbf{y} = (y_v)_{v \in V}$. CRF defines a conditional probability distribution

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x},\mathbf{w}}} \exp \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle,$$

where $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ is a joint feature map. $Z_{\mathbf{x},\mathbf{w}}$ is the partition function dependent on $\mathbf{x}$ that sums over all possible multilabels

$$Z_{\mathbf{x},\mathbf{w}} = \sum_{\mathbf{y}' \in \mathcal{Y}} \exp \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}') \rangle. \tag{4.3}$$

Thus, when conditioned on $\mathbf{x}$, random variables y_v obey the Markov property with respect to the output graph G.

Applying the similar regularization technique as applied to Logistic Regression in Section 2.2.2, the feature weight parameter $\mathbf{w}$ can be solved by introducing a Gaussian prior and maximizing the logarithm of the resulting *Maximize A-Posteriori* (MAP) problem (Taskar et al., 2002) defined as

$$L(\mathbf{w}) = \sum_{i=1}^{m} \left[\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) \rangle - \log \sum_{\mathbf{y}' \in \boldsymbol{\mathcal{Y}}} \exp \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}') \rangle \right] - \frac{1}{\sigma^2} ||\mathbf{w}||^2 . \tag{4.4}$$

The optimization problem derived from (4.4) is an instantiation of the regularized risk minimization (2.7) with the log-loss and the L_2-norm regularization (2.8). It is solved in (Lafferty et al., 2001) by an improved iterative scaling algorithm (IIS) (Della Pietra et al., 1997). In order to make CRF work in practice, one also need to make sure that the partition function (4.3) can be evaluated efficiently.

4.2.3 Max-Margin Markov Network (M^3N)

Taskar et al. (2004) proposed Max-Margin Markov Network (M^3N) that combines the framework of the kernel based discriminative learning and

the probabilistic graphical model. M^{3}N extends SVM (Section 2.2.3) to the structured output space. It also improves CRF (Section 4.2.2) by which the evaluation of the partition function (4.3) can be avoided by introducing the odd-ratio typed learning that is not dissimilar to Logistic Regression presented in Section 2.2.2.

M^{3}N defines a log-linear Markov network over multiple labels which exploits the correlation between labels. The compatibility score defined by

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle \tag{4.5}$$

can be seen as the affinity of a multilabel $\mathbf{y}$ to an input $\mathbf{x}$ according to an output graph. The feature weight parameter $\mathbf{w}$ in (4.5) ensures the example with the correct multilabel will obtain a higher score than with any incorrect multilabels. M^{3}N defines a margin as the difference of compatibility scores between the correct example $(\mathbf{x}_i, \mathbf{y}_i)$ and the psudo-example $(\mathbf{x}_i, \mathbf{y})$. Under the Maximum-Margin principle in Section 2.2.3, M^{3}N requires the margin to be at least $\ell(\mathbf{y}_i, \mathbf{y})$. To learn the feature weight parameter $\mathbf{w}$, we need to solve the following primal optimization problem

Definition 9. M^{3}N *Optimization Problem in Primal.*

$$\begin{aligned} \underset{\mathbf{w}, \xi_i}{\boldsymbol{min}} \quad & \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{m} \xi_i \\ \boldsymbol{s.t.} \quad & \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle \geq \ell(\mathbf{y}_i, \mathbf{y}) - \xi_i, \\ & \forall \xi_i \geq 0, \, \forall \mathbf{y} \in \boldsymbol{\mathcal{Y}} / \mathbf{y}_i, \, \forall i \in \{1, \cdots, m\}, \end{aligned}$$

where ξ_i is the slack allotted to each example to make sure the solution can always be found, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss function between a correct multilabel $\mathbf{y}_i$ and an incorrect multilabel $\mathbf{y}$, C is the slack parameter that controls the amount of regularization in the model.

For each example $\mathbf{x}_i$, the optimization calls for maximizing the margin between the correct label $\mathbf{y}_i$ and any incorrect labels $\mathbf{y}$. The margin is scaled by the loss function $\ell(\mathbf{y}_i, \mathbf{y})$ such that the completely incorrect multilabel will incur bigger loss than the nearly correct multilabel. The loss scaled margin optimization will push the high-loss pseudo-examples further away from the correct example than the low-loss pseudo-examples. Definition 9 is an instantiation of the regularized risk minimization (2.7) with the hamming loss and the L_2-norm regularization (2.8).

The primal optimization problem of M^{3}N in Definition 9 is difficult to solve as there are exponential number of constraints, one for each pseudo-example $(\mathbf{x}_i, \mathbf{y})$. The corresponding dual form is also difficult due to the

exponential number of dual variables (Taskar et al., 2004). By exploring the Markov network structure, the original optimization problem (Definition 9) can be formulated into a factorized dual quadratic programming, as long as the loss function ℓ and the joint feature map $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ are decomposable over the Markov network.

As the number of parameters is quadratic in the number of training examples and the edges of the Markov network, it still cannot fit into the standard Quadratic Programming (QP) solver. Taskar et al. (2004) developed a coordinate descent method analogous to Sequential Minimal Optimization (SMO) (Platt, 1998, 1999). Many other efficient optimization algorithms have been proposed, for example, exponential gradient optimization method in (Bartlett et al., 2005), extra-gradient method in (Taskar et al., 2006), sub-gradient method in Ratliff et al. (2007), and conditional gradient method in (Rousu et al., 2006, 2007).

In order to use M^{3}N in practice, one have to solve the *loss augmented* inference problem defined as

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \boldsymbol{\mathcal{Y}} / \mathbf{y}_i}{\mathbf{argmax}} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle + \ell(\mathbf{y}_i, \mathbf{y}). \tag{4.6}$$

To compute (4.6) efficiently, the loss function need to be decomposable over the Markov network. Nevertheless, M^{3}N improves CRF by avoiding the evaluation of the partition function and allowing complex loss functions to be defined.

4.2.4 Max-Margin Conditional Random Fields (MMCRF)

Max-Margin conditional random field (MMCRF) is a structured output learning method developed in (Rousu et al., 2007). It extends M^{3}N by defining the joint feature map as the tensor product between an input feature map and an output feature map, and by developing an efficient optimization strategy. MMCRF is applied in Publication II in which the task is to reliably predict the multiple interdependent molecular activities.

In particular, MMCRF uses *exponential family* to model the conditional probability of a multilabel $\mathbf{y}$ given an input example $\mathbf{x}$

$$P(\mathbf{y}|\mathbf{x}) \propto \exp(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle) = \prod_{e \in E} \exp(\langle \mathbf{w}_e, \boldsymbol{\phi}_e(\mathbf{x}, \mathbf{y}_e) \rangle),$$

where $\boldsymbol{\phi}_e(\mathbf{x}, \mathbf{y}_e) = \boldsymbol{\varphi}(\mathbf{x}) \otimes \boldsymbol{\Upsilon}_e(\mathbf{y}_e)$ is the tensor product between an input feature map and an output feature map which is the label of an edge $e \in E$ in an output Markov network G with respect to a multilabel $\mathbf{y}$. In order to obtain $\mathbf{w}$, one needs to solve the primal optimization problem that

is not dissimilar to Definition 9. After the feature weight parameter w is obtained, the prediction of an input example can be computed by solving the following argmax problem

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \boldsymbol{\mathcal{Y}}/\mathbf{y}_i}{\mathbf{argmax}} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle. \tag{4.7}$$

To solve the optimization problem, MMCRF uses the conditional gradient optimization method (Bertsekas, 1995) in the marginalized dual space (Taskar et al., 2004), which not only benefits from a polynomial-size parameter space but also enables kernels that can deal with the non-linearity of the complex molecular structures. The inference problem (4.7) is solved by loopy belief propagation (LBP) which is an instantiation of the message-passing algorithm developed in (Wainwright and Jordan, 2003).

4.2.5 Support Vector Machines for Interdependent and Structured Outputs (SSVM)

Support Vector Machines for interdependent and structured output space (SSVM) is developed in (Tsochantaridis et al., 2004, 2005). The formalism of SSVM is quite similar to M^{3}N described in Section 4.2.3. Compared to M^{3}N which scales the margin by the loss function, SSVM scales the margin errors (*slacks*) by the loss function. The primal optimization problem of SSVM can be defined as

Definition 10. SSVM *Optimization Problem in Primal.*

$$\begin{aligned} \underset{\mathbf{w}, \xi_i}{\boldsymbol{min}} \quad & \frac{1}{2} ||\mathbf{w}||^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i \\ \boldsymbol{s.t.} \quad & \langle \mathbf{w}, \boldsymbol{\phi}(x_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \boldsymbol{\phi}(x_i, \mathbf{y}) \rangle \geq 1 - \frac{\xi_i}{\ell(\mathbf{y}_i, \mathbf{y})}, \\ & \forall \xi_i \geq 0, \, \forall \mathbf{y} \in \boldsymbol{\mathcal{Y}}/\mathbf{y}_i, \, \forall i \in \{1, \cdots, m\}, \end{aligned}$$

where ξ_i is the slack allotted to each example, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss function between a correct multilabel and an incorrect multilabel, and C is the slack parameter that controls the amount of regularization in the model. The interpretation of Definition 10 is also similar to that of Definition 9. Besides, Tsochantaridis et al. (2004) suggests that M^{3}N will work hard on the pseudo-examples $(\mathbf{x}_i, \mathbf{y})$ which incur a big loss though they may not even close to be confusable to the true multilabel $\mathbf{y}_i$.

On the other hand, the optimization techniques employed by SSVM differ significantly compared to M^{3}N. SSVM will have to work with the exponential number of constraints as the optimization is not decomposable

over the Markov network. Tsochantaridis et al. (2004) developed an iterative optimization approach that creates a nested sequence of successively tighter relaxations of the original problem via a cutting-plane method (Bishop, 2007; Joachims et al., 2009). Constraints are added as necessary and the iterative optimization approach will converge to an optimal solution of ϵ precision within a polynomial number of iterations.

Besides the issue during the optimization, another problem with SSVM is the intractability of the inference problem. To find the most violating constraint, we need to compute the loss-augmented inference problem defined in (Tsochantaridis et al., 2005) as

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \boldsymbol{\mathcal{Y}}/\mathbf{y}_i}{\mathbf{argmax}} \left[1 - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle\right] \ell(\mathbf{y}_i, \mathbf{y}). \tag{4.8}$$

The loss function appears as a multiplicative term making (4.8) not decomposable over the Markov network. This gives an intractable inference problem in general. In exchange of the intractability, SSVM can work with complex loss functions which do not assume any properties of decomposition. The generality of the loss function can be seen as an advantage compared to CRF, M^3N, and MMCRF.

4.3 SPIN for Network Response Prediction

Publication I presents a novel definition of the network response prediction problem and develops a structured output learning model for the problem. Unlike the previous methods which model the influence in terms of the network connectivity, the proposed model (SPIN) is *context-sensitive*. That is, the influence dynamics also depend on the properties of the action performed on the underlying network. The inference problem of SPIN is $\mathcal{NP}$-hard in general. We develop a semidefinite programming algorithm (SDP) with an approximation guarantee as well as a fast GREEDY heuristics.

4.3.1 Background

With the extensive availability of the large scale networks, there is an increasing amount of interest in studying the phenomena of the network influence, in particular, the structure, the function, and the influence dynamics. The outcome of the network influence research has been widely applied to many areas, for example, the spreadness of pathogens or infectious diseases (Hethcote, 2000; Anderson and May, 2002), the diffusion

of medical and technology innovations (Strang and Soule, 1998; Rogers, 2003), the opinion and news formations (Adar et al., 2004; Gruhl et al., 2004; Adar and Adamic, 2005; Leskovec et al., 2007; Liben-Nowell and Kleinberg, 2008; Leskovec et al., 2009), and the viral market (Domingos and Richardson, 2001; Kempe et al., 2003; Liben-Nowell and Kleinberg, 2003).

In the field of studying the network influence, one primary interest is to discover the latent structure that reveals the dynamics of influences. In general, the problem can be defined into two different ways depending on the availability of the underlying network structure. On one hand, one would assume that the underlying structure is hidden or incomplete and the only observation is a cascade of actions. The instantiation of the setting can be, for example, online news agents sharing information but not physically connected, in the epidemiological study where people are affected by pathogens through various ways. The task is to infer the network structure in terms of edges connecting nodes given a collection of actions. Many algorithms are designed to solve the problem in this setting, for example, NETINF (Gomez Rodriguez et al., 2010), NETRATE (Rodriguez et al., 2011), KERNEL CASCADE (Du et al., 2012), the two stage model for inferring influence (Du et al., 2014), the inference algorithm using cascades without any timestamps (Amin et al., 2014), and the general framework of inferring the diffusion structure (Daneshmand et al., 2014). On the other hand, we argue the problem is unnecessarily hard as in many cases the structure of the network is observed (e.g., the friendship network, the citation network). There are also many related research that aims to discover the hidden variables in the network (Saito et al., 2008; Goyal et al., 2010).

However, none of them consider the property of the action. In particular, our network influence problem is motivated by the following observation: for a given action a performed on a network G, the influence from a node u to a node u' not only depends on their connections but also depends on the action under consideration. For example, u' is a follower of u in Twitter, u' will retweet the message from u if it is related to *science* but not related to *politics*. Therefore, we propose the following definition of the network response problem

Definition 11. *Network Response Problem. Given a complex network and an action performed on the network, predict an optimal subnetwork that best responses to the action. In particular, which nodes perform the action*

and which directed edges relay the action from one node to its neighbors.

4.3.2 Methods

We approach the problem by structured output learning, where we define a computability score as the inner product between an action a and a response network $G_{\mathbf{a}}$

$$F(\mathbf{a}, G_{\mathbf{a}}; \mathbf{w}) = \langle \mathbf{a}, \boldsymbol{\phi}(\mathbf{a}, G_{\mathbf{a}}) \rangle.$$

Intuitively, the action a with a correct response network $G_{\mathbf{a}}$ will achieve high score than with any incorrect response network $G'_{\mathbf{a}}$. The joint feature map $\boldsymbol{\phi}(\mathbf{a}, G_{\mathbf{a}})$ is composed by the tensor product between an input feature map of an action $\boldsymbol{\varphi}(\mathbf{a})$ and an output feature map of a response network $\boldsymbol{\Upsilon}(G_{\mathbf{a}})$. In particular, $\boldsymbol{\varphi}(\mathbf{a})$ can be a bag-of-words feature of an action (e.g., a posted message on Twitter) and $\boldsymbol{\Upsilon}(G_{\mathbf{a}})$ can be a vector of edges and labels of a response network $G_{\mathbf{a}}$.

The feature weight parameter $\mathbf{w}$ is learned through maximum-margin structured output learning by solving the following optimization problem

Definition 12. *Primal* SPIN *Optimization Problem.*

$$\begin{aligned} \underset{\mathbf{w}, \xi_i}{\boldsymbol{min}} \quad & \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{m} \xi_i \\ \boldsymbol{s.t.} \quad & F(\mathbf{a}_i, G_{\mathbf{a}_i}; \mathbf{w}) > \underset{G'_{\mathbf{a}_i} \in \mathcal{H}(G)/G_{\mathbf{a}_i}}{\boldsymbol{max}} (F(\mathbf{a}_i, G'_{\mathbf{a}_i}; \mathbf{w}) + \ell_G(G_{\mathbf{a}_i}, G'_{\mathbf{a}_i})) - \xi_i, \\ & \xi_i \geq 0, \forall i \in \{1, \cdots, m\}, \end{aligned}$$

where $\mathcal{H}(G)$ denotes a set of directed acyclic graphs of G. To solve the above optimization problem, we have to compute the highest-scoring subgraph given an action. In particular, the goal during training is to find the worst margin violating subgraph which corresponds to solving the following loss-augmented maximization problem

$$H^*(\mathbf{a}_i) = \underset{G'_{\mathbf{a}_i} \in \mathcal{H}(G)/G_{\mathbf{a}_i}}{\mathbf{argmax}} (F(\mathbf{a}_i, G'_{\mathbf{a}_i}; \mathbf{w}) + \ell_G(G_{\mathbf{a}_i}, G'_{\mathbf{a}_i})).$$

The goal during prediction is to find the subgraph with maximum compatibility given an action

$$H^*(\mathbf{a}_i) = \underset{H \in \mathcal{H}(G)}{\mathbf{argmax}} F(\mathbf{a}_i, H; \mathbf{w}). \tag{4.9}$$

As these two problems are different only in terms of the definition of the scores, we explain our inference algorithm based on (4.9) by writing the

problem explicitly in terms of the weight vectors and the feature maps

$$\begin{aligned} H^*(\mathbf{a}_i) &= \underset{H \in \mathcal{H}(G)}{\mathbf{argmax}} \langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{a}) \otimes \boldsymbol{\Upsilon}(H) \rangle \\ &= \underset{H \in \mathcal{H}(G)}{\mathbf{argmax}} \sum_{e \in E^H} s_{\mathbf{y}_e}(e, \mathbf{a}), \end{aligned} \tag{4.10}$$

where $s_{\mathbf{y}_e}(e, \mathbf{a}) = \sum_i \mathbf{w}_{i,e,\mathbf{y}_e} \boldsymbol{\varphi}(\mathbf{a})$ denotes the score of an edge e with an edge label $\mathbf{y}_e$.

We have proved the $\mathcal{NP}$-hardness of (4.10) by forming a reduction from the MAX-CUT problem (Garey and Johnson, 1990). In addition, we proposed two algorithms to solve (4.10). The first is called SDP inference which introduces for each node $u \in V$ a binary variable $x_u \in \{-1, +1\}$ and transforms the inference problem into an Integer Quadratic Programming (IQP) problem. The IQP is tackled by a similar technique proposed in (Goemans and Williamson, 1995) such that each variable x_u is relaxed to a vector $\mathbf{v}_u \in \mathbb{R}^n$ and the relaxed problem is solved by Semidefinite Programming (SDP). The resulting vector is rounded back into binary values by Incomplete Cholesky Decomposition. The benefit from SDP inference algorithm is an approximation guarantee. In particular, the proposed SDP inference algorithm is a 0.796 approximation of the original IQP.

As SDP inference is not scalable to the large scale networks, we develop a GREEDY heuristic based on the observation stated in the following lemma:

Lemma 1. *The inference problem (4.10) can be expressed equivalently as a function of activated vertices*

$$H^*(\mathbf{a}) = \underset{H \in \mathcal{H}(G)}{\boldsymbol{argmax}} \sum_{v_i \in V_p^H} F_m(v_i).$$

The proof is given in the supplementary material of Publication I. As a result, the GREEDY algorithm starts with an empty vertex set and adds one vertex in each iteration such that the increment of the score is maximized over all possible choices of inactivated vertices. The procedure ends when the objective cannot be improved. It is worth pointing out that we are not able to give any approximation guarantee for the solutions produced by the GREEDY algorithm. The property of sub-modularity, which is often used to analyze the greedy algorithm, only holds for the special case of our inference problem.

5. Structured Output Prediction with Unknown Output Graphs

Structured output learning relies on an output graph connecting multi-label interdependent output variables to exploit the the correlation between labels. The applicability of structured output learning is limited due to the fact that the output graph needs to be known *apriori*. In this chapter, we aim to develop several structured output learning algorithms that are not constrained by the availability of the output graph. As a result, structured output learning can be applied to a wide range of multilabel classification problems. In Section 5.1, we study the multilabel molecular classification problem with structured output learning in which the output graph is extracted from auxiliary datasets. In Section 5.2, we present MVE which uses majority vote to combines the predictions from a set of structured output learners built on a collection of random output graphs. In Section 5.3, we present two aggregation techniques, namely AMM and MAM, which perform inference on output graphs before or after combining multiple structured output learners. In Section 5.4, we present RTA which is a joint learning and inference model that performs Max-Margin learning on a random sample of spanning trees.

5.1 Structured Output Prediction for Molecular Classification

The molecular classification problem has been tackled by a variety of single-label classification approaches (Menchetti et al., 2005; Singh et al., 2012; Dutt, 2012). On the other hand, multiple interdependent molecular activities are often screened simultaneously in the field of drug research (Shoemaker, 2006), which presents two challenges for single-label classification. The first is the scalability in which a set of single-label classifiers needs to be built to predict multiple activities of molecules. This becomes infeasible when we need to examine a large number of molecular activi-

ties at the same time. The second challenge is that single-label classification ignores the correlation between multiple output variables. On the other hand, multiple molecular activities are often correlated which can be utilized to improve the classification performance. In Publication II, we explore the potential of structured output learning in the molecular activity classification problem. To apply structured output learning, we extract output graphs from several auxiliary datasets which encode the correlation between multiple activities of molecules.

5.1.1 Background

Molecular classification, the goal of which is to predict the anti-cancer potentials of drug-like molecules, is a crucial step in drug discovery and has gained in popularity from the machine learning community (Singh et al., 2012; Dutt, 2012). Viable molecular structures are scanned, searched, or designed for therapeutic efficacy. In particular, expensive preclinical *in vitro* and *in vivo* drug tests can be largely avoided and special efforts can be devoted to few promising candidate molecules, once accurate *in silico* models are available (Burbidge et al., 2001).

A variety of machine learning methods have been developed for this task, to name but a few, inductive logic programming (King et al., 1996), artificial neural network (Bernazzani et al., 2006), kernel methods for nonlinear molecular properties (Trotter et al., 2001; Ralaivola et al., 2005; Swamidass et al., 2005; Ceroni et al., 2007a), and the SVM based methods (Trotter et al., 2001; Byvatov et al., 2003; Xue et al., 2004). Albeit with a large quantity of developed methods, they only focus on predicting a single output variable (e.g., the inhibition potential of a molecule in a target cell line). On the other hand, a large number of interdependent molecular activities are often screened at the same time in the field of drug research. For example, in the recent *NCI-60* Human Tumor Cell Line Screen project (Shoemaker, 2006), thousands of molecular structures are tested agains hundreds of target cell lines.

5.1.2 Methods

To efficiently and accurately predict molecular activities in multiple cell lines at the same time, we applied a structured output learning approach in Publication II, which is to our knowledge the first multilabel classification approach for the molecular classification problem. The algorithm is

an instantiation of MMCRF (Rousu et al., 2007) presented in Section 4.2.4. In particular, the model defines a compatibility score through the inner product of a molecular structure $\mathbf{x}$ and the activities in multiple target cell lines $\mathbf{y}$

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle,$$

where $\mathbf{w}$ is the feature weight parameter to ensure that a molecule with correct activities will be scored higher than with incorrect activities. $\mathbf{w}$ is obtained by maximizing the minimum loss-scaled margin between the correct examples $(\mathbf{x}_i, \mathbf{y}_i)$ and the incorrect pseudo-examples $(\mathbf{x}_i, \mathbf{y})$ over all training examples, which amounts to solving the optimization problem that is not dissimilar to Definition 9.

As MMCRF kernelizes input, we use graph kernel to measure the similarity between a pair of molecular structures. The common way to represent the structure of a molecule is to use an undirected labeled graph $G = (V, E)$ with a set of vertices $V = \{v_1, \cdots, v_n\}$ that corresponds to atoms and a set of edges $E = \{e_1, \cdots, e_m\}$ that corresponds to covalent bonds. The adjacency matrix A of a graph G is defined such that the (i, j)'th entry $A_{i,j}$ equals to one if there is an edge connecting the i'th and the j'th atoms.

Walk Kernel (Kashima et al., 2003; Gärtner, 2003) computes the sum of all matching walks in a pair of graphs. The contribution of each matching walk is down scaled exponentially by the length of the walk. Let w_m denote a walk of length m such that there exists an edge for each pair of vertices (v_i, v_{i+1}) for all $i \in \{1, \cdots, m-1\}$. In addition, we use $G_\times(G_1, G_2)$ to denote the direct product graph of two graphs G_1 and G_2, in which the set of vertices in $G_\times$ is computed by

$$V_\times(G_1, G_2) = \{(v_1, v_2) \in V_1 \times V_2,\ label(v_1) = label(v_2)\},$$

and the set of edges in $G_\times$ are computed by

$$E_\times(G_1, G_2) = \{((v_1, v_2), (u_1, u_2)) \in V_\times \times V_\times,\ (v_1, u_i) \in E_1 \wedge (v_2, u_2) \in E_2\}.$$

Walk kernel can be equivalently expressed in terms of the adjacency matrix $A_\times$ of the product graph $G_\times$ as

$$K_{wk}(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} \left[\sum_{n=0}^{\infty} \lambda^n A_\times^n \right]_{i,j},$$

where $0 < \lambda \leq 1$ is a scaling parameter. Using exponential series or geometric series, walk kernel can be evaluated in cubic time (Gärtner,

2003) in the number of vertices $V_\times$ according to

$$K_{wk}(G_1, G_2) = \mathbf{e}^\intercal(\mathbf{I} - \lambda A_\times)^{-1}\mathbf{e},$$

where $\mathbf{I}$ denotes an identity matrix and $\mathbf{e}$ denotes a matrix of ones.

Weighted Decomposition Kernel developed in (Menchetti et al., 2005; Ceroni et al., 2007b) is an extension of *Substructure Kernel* (Komarek and Moore, 1999) that weights the identical atoms of two graphs by contextual information. The contextual information is defined as the matching subgraph in the neighbourhood of an atom. In addition, we used *Tanimoto Kernel* (Ralaivola et al., 2005) on a finite set of molecular fingerprints (Wang et al., 2009). See (Vishwanathan et al., 2010) for a more comprehensive survey on graph kernels.

To apply the structured output learning method described above, we need to have an output graph connecting labels given *apriori*. However, the output graph is not known in the molecular classification problem. There exists a variety of auxiliary datasets (Shoemaker, 2006) which implicitly encodes the correlation of labels (target cell lines). To extract the output graph, we first compute a covariance matrix of cell lines from the auxiliary data, then extract the structure of the output graph by the following two methods. In maximum spanning tree approach, we take the minimum number of edges that make a connected graph whilst maximize the sum of edge weights. In correlation thresholding approach, we take all edges that exceed a fixed threshold in terms of the pairwise correlation, which typically generates a non-tree graph.

5.2 Graph Labeling Ensemble (MVE)

The structured output learning approaches, relying on the representation of multiple output variables through an output graph, allow us to exploit the correlation between labels. To apply structured output learning, it is assumed that the structure of the output graph is known *apriori*. For the molecular classification problem in Publication II where the output graph is not observed, we can extract the structure of the output graph by examining a collection of auxiliary datasets which explicitly encode the correlation of labels. For most real world multilabel classification problems, however, we cannot take for granted the availability of the output graph or the auxiliary data that reveals the label correlations. Therefore, in Publication III, we explore the potential of using majority vote to com-

bine the predictions from a set of structured output learners built on a collection of random output graphs. We also examine the classification performance on the molecular classification problem.

5.2.1 Methods

We use MMCRF as the base classifier trained on a collection of random output graphs. In particular, a random graph G_t is generated for each base learner to couple the multiple labels which are the activities of the molecule x in all target cell lines. The base model MMCRF is learned with the training data $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$ and the output graph G_t. After all base learners have been generated, the predictions are extracted from the base learners and are collected for a post-processing step, in which we compute a majority vote over the graph labeling from the sign on the means of the base classifier's prediction

$$F_j^{\text{MVE}} = \underset{y_j \in \mathcal{Y}_j}{\textbf{argmax}} \left(\frac{1}{T} \sum_{i=1}^{T} \mathbf{1}_{\{F_j^{(t)}(x)=y_j\}} \right), \forall j \in \{1, \cdots, k\},$$

where T denotes the size of the ensemble which is also the number of random output graphs, and $F^{(t)}(\mathbf{x}) = \{F_j^{(t)}(x)\}_{j=1}^k$ denotes the predicted multilabel from the t'th base learner. That is, the ensemble prediction on each microlabel is the most frequently appearing prediction among the base classifiers. It is also worth pointing out that MVE is not restricted to the base learner MMCRF and can be extended with any other structured output learning models as long as the model incorporates the output structure into learning and makes predictions based on the structure of the output graph.

In addition, we design two approaches to generate the random output graphs. In random spanning tree approach, we first generate a random correlation matrix and extract a spanning tree out from the matrix. This approach outputs a tree structure connecting all vertices. In random pairing approach, we randomly draw two vertices at a time and couple the two with an edge. This approach outputs a set of disconnected pairs.

5.3 Random Graph Ensemble (AMM, MAM)

Section 5.2 has shown that the prediction performance of the molecular classification problem can be improved by applying majority vote to com-

bine the predictions from multiple structured output learners built on a collection of random output graphs. This section is based on Publication IV in which we present two aggregation techniques to combine multiple structured output learners. The proposed model, namely AMM and MAM, also perform inference before or after combining the base learners. The performance of the proposed models is evaluated on a set of heterogeneous multilabel datasets from a variety of domains. In addition, we study the performance of MAM in terms of the reconstruction error of the compatibility score.

5.3.1 Background

We still work with the assumption made for MVE in Section 5.2 in which we assume the structure of the output graph is incorporated during learning and the prediction is made according to the structure. In addition, we assume that the base learner for AMM and MAM is defined on a Markov network. That is, the base learner computes a compatibility score $\psi(\mathbf{x}, \mathbf{y})$ for $(\mathbf{x}, \mathbf{y}) \in \boldsymbol{\mathcal{X}} \times \mathcal{Y}$ based on the output graph G, indicating how well an input $\mathbf{x}$ gets along with an output $\mathbf{y}$. The compatibility score $\psi(\mathbf{x}, \mathbf{y})$ is defined as

$$\psi(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle = \sum_{e \in E} \langle \mathbf{w}, \boldsymbol{\phi}_e(\mathbf{x}, \mathbf{y}_e) \rangle = \sum_{e \in E} \psi_e(\mathbf{x}, \mathbf{y}_e),$$

where $\psi_e(\mathbf{x}, \mathbf{y}_e)$ denotes the edge compatibility score (*edge potential*) between an input $\mathbf{x}$ and the edge label $\mathbf{y}_e$ of an edge e. $\mathbf{w}$ is the feature weight parameter which ensures that an input $\mathbf{x}$ with the correct output $\mathbf{y}$ will achieve higher compatibility score than with any incorrect outputs.

In addition, we assume that we have access to the set of edge potentials of each base classifier

$$\boldsymbol{\psi}_E^{(t)} = (\psi_e^{(t)}(\mathbf{x}, \mathbf{u}_e))_{e \in E^{(t)}, \mathbf{u}_e \in \mathcal{Y}_e}.$$

With edge compatibility scores, we can infer the *max-marginal* (Wainwright et al., 2005) of the i'th node which is the score incurred by assigning a label $u_i \in \mathcal{Y}_i$ to the i'th node defined by

$$\tilde{\psi}_j(\mathbf{x}, u_j) = \max_{\mathbf{y} \in \mathcal{Y}, y_j = u_j} \sum_{e \in E} \psi_e(\mathbf{x}, \mathbf{y}_e).$$

In words, the max-marginal is the maximum score of a multilabel consistent with $y_i = u_i$. We use $\tilde{\boldsymbol{\psi}} = (\tilde{\psi}_j(\mathbf{x}, u_j))_{j \in V, u_j \in \mathcal{Y}_j}$ to denote the collection of max-marginals.

5.3.2 Methods

Let $\mathcal{G} = \{G^{(1)}, \cdots, G^{(T)}\}$ denote a set of random output graphs, and let $\{\tilde{\boldsymbol{\psi}}^{(1)}, \cdots, \tilde{\boldsymbol{\psi}}^{(T)}\}$ denote the max-marginal vectors from the base learners built on a collection of random output graphs. The prediction of the Average-of-Max-Marginal (AMM) aggregation on the j'th node is obtained by averaging the max-marginals from all base classifiers and choose the maximizing microlabel for the node

$$F_j^{\text{AMM}} = \underset{u_j \in \mathcal{Y}_j}{\mathbf{argmax}} \frac{1}{T} \sum_{t=1}^{T} \tilde{\psi}_{j,u_j}^{(t)}(\mathbf{x}).$$

The predicted multilabel by AMM is composed by the predicted microlabels

$$F^{\text{AMM}} = \left(F_j^{\text{AMM}}\right)_{j \in V}.$$

AMM performs inference to find the set of max-marginals before combining base classifiers. On the other hand, the Maximum-of-Average-Marginals (MAM) aggregation first collects the local edge potentials $\boldsymbol{\psi}_E^{(t)}$ from each base learner, averages them and performs a finial inference on a global consensus graph $\hat{G} = (\hat{E}, V)$ with the averaged edge potentials. MAM is defined as

$$F^{\text{MAM}}(x) = \underset{\mathbf{y} \in \mathcal{Y}}{\mathbf{argmax}} \sum_{e \in \hat{E}} \frac{1}{T} \sum_{t=1}^{T} \psi_e^{(t)}(x, \mathbf{y}_e) = \underset{\mathbf{y} \in \mathcal{Y}}{\mathbf{argmax}} \frac{1}{T} \sum_{t=1}^{T} \sum_{e \in \hat{E}} \langle \mathbf{w}_e^{(t)}, \boldsymbol{\phi}_e(x, \mathbf{y}_e) \rangle.$$

In addition, Publication IV, Lemma 1 simplifies the computation of MAM in terms of dual variables and kernels.

Besides the proposed algorithms, we also present a theoretical analysis to explain the improvement of MAM. The analysis extends the theory of single-label ensemble developed in (Brown and Kuncheva, 2010). In particular, Publication IV, Theory 1 states that the reconstructive error of MAM is guaranteed to be less than or equal to the average reconstruction error of base classifiers. The improvement can be decomposed into two terms, namely *diversity* and *coherence*. The former measures the variability of the individual classifiers learned from different perspectives which shares the same argument as the analysis of single-label ensemble (Brown and Kuncheva, 2010). The latter measures the correlation of microlabel predictions in which the correlation has a positive effect on the performance.

5.4 Random Spanning Tree Approximation (RTA)

Publication V presents Random Spanning Tree Approximation (RTA) for structured output learning in which the output graph is not observed but believed to play an important role during learning. RTA is a major step forward of MAM by bringing in a joint learning and inference framework such that the base learners built from a collection of random spanning trees are optimized simultaneously towards the same global objective. Publication V also presents the theoretical studies which not only explain the intuition behind the learning model but also guarantee the performance by the generalization error analysis. Meanwhile, RTA lays the foundation of tackling the intractability of the graph inference on unknown graph structures in which fast optimization and accurate predictions can be achieved with attainable computational efforts.

5.4.1 Background

The applicability of structured output learning is limited due to the fact that the output graph is assume to be known *apriori*. It is difficult to learn the correlation structure of labels from data (Chickering et al., 1994) if it is not harder than structured output learning. Instead we can resort to a *complete graph* by assuming that a complete set of pairwise correlations have enough expression power to describe the correlation between labels. With the complete graph as the output graph, we can construct a structured output learner and rely on the optimization algorithm to correctly reveal the hidden "parameters" defined on the edges of the compete graph (e.g., edge potentials).

Structured output learning on a complete graph is not an easy problem as the inference is $\mathcal{NP}$-hard in nature. The inference problem is often instantiated as finding a *Maximum A-Posteriori* (MAP) configuration on a graph structured probability distribution. In terms of the intractability issue of the graph inference problem, many techniques have been proposed but with important differences. Jordan and Wainwright (2004) developed a semi-definite programming convex relaxation for the inference on the graph with cycles. Wainwright et al. (2005) proposed a MAP inference with the tree-based and *linear programming* (LP) relaxation. Efficient inference algorithms on special graphs have also been studied in (Globerson and Jaakkola, 2007).

Publication V is motivated by the well-established Maximum-Margin

Principle as described in Section 2.2.3. The work investigates whether the problem of inference over a complete graph in structured output learning can be avoided by exploring the properties of the Maximum-Margin Principle. Start from a sampling results, Publication V, Lemma 3 shows that with a high probability a big fraction of the margin achieved by a complete graph can be obtained by a random sample of spanning trees of a small size. Besides, Publication V, Theorem 5 shows that the good generalization error can also be guaranteed when learning with, instead of a complete graph, a random sample of spanning trees.

Thus, in addition to Publication IV, Theory 1, we further provide the theoretical justification of combining a set of base learners trained on a random sample of spanning trees. Besides, Publication V, Theorem 5 suggests we should, instead of optimizing the margin separately on each spanning tree similar to that in MAM, optimize the joint margin from all spanning trees. The strategy leads to the learning model presented in the following section.

5.4.2 Methods

Let $\mathcal{T} = \{T_1, \cdots, T_n\}$ denote a sample of n random spanning trees, and $\{\mathbf{w}_{T_t} | T_t \in \mathcal{T}\}$ denote the feature weight parameters to be learner on each tree. For each example $(\mathbf{x}_i, \mathbf{y}_i)$, the goal of the optimization is to maximize the joint margin from all spanning trees between the correct training examples and the pseudo-examples $(\mathbf{x}_i, \mathbf{y})$ defined as

Definition 13. Primal L_2-norm Random Tree Approximation (RTA).

$$\begin{aligned} \underset{\mathbf{w}_{T_t}, \xi_i}{\boldsymbol{min}} \quad & \frac{1}{2} \sum_{t=1}^{n} ||\mathbf{w}_{T_t}||^2 + C \sum_{i=1}^{m} \xi_i \\ \boldsymbol{s.t.} \quad & \sum_{t=1}^{n} \langle \mathbf{w}_{T_t}, \boldsymbol{\phi}_{T_t}(\mathbf{x}_i, \mathbf{y}_i) \rangle - \underset{\mathbf{y} \neq \mathbf{y}_i}{\boldsymbol{max}} \sum_{t=1}^{n} \langle \mathbf{w}_{T_t}, \boldsymbol{\phi}_{T_t}(\mathbf{x}_i, \mathbf{y}) \rangle \geq 1 - \xi_i, \\ & \xi_i \geq 0 \,, \forall\, i \in \{1, \ldots, m\}, \end{aligned}$$

where $\boldsymbol{\phi}_{T_t}(\mathbf{x}, \mathbf{y})$ is the feature map that is local on each tree T_t, ξ_i is the margin slack allocated for each $\mathbf{x}_i$, and C is the slack parameter that controls the amount of regularization. Definition 13 is an instantiation of the regularized learning (Section 2.7) in terms of the L_2-norm regularization (2.8) and $0/1$ loss (2.2).

The key for the optimization is to solve the argmax problem efficiently. This is an $\mathcal{NP}$-hard problem in practice, as the size of the multilabel space is exponential in the number of microlabels. In Publication V, we have

developed a K-best inference algorithm working in $\Theta(Knk)$ time per data point, where k is the number of microlabels and K is the number of best multilabels we compute from each random spanning tree.

It is known that the exact solution for the inference problem on an individual tree T_t is tractable (Koller and Friedman, 2009) for which

$$\hat{\mathbf{y}}_{T_t}(x) = \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} F_{\mathbf{w}_{T_t}}(x,\mathbf{y}) \;=\; \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{w}_{T_t}, \boldsymbol{\phi}_{T_t}(x,\mathbf{y})\rangle, \tag{5.1}$$

can be solved in $\Theta(k)$ time by *dynamic programming* also known as *max-product* or *min-sum*. However, there is no guarantee that the maximizer of (5.1) is also the global maximizer of Definition 13 over the set of random spanning trees. Therefore, we compute the top K-best multilabels for each random spanning tree. In total the computation costs $\Theta(Knk)$ time for all spanning trees. Publication V, Lemma 7 provides a method to retrieve the best multilabel from the K-best multilabel list in linear time. We still need to make sure that the global maximizer is within the K-best multilabel list. Publication V, Lemma 8 guarantees that with a high probability the global maximizing multilabel is in the list and K does not need to be large.

In addition, we derived the marginalized dual representation of the primal optimization problem in Definition 13, which not only works with a polynomial sized parameter space but also enables kernels to tackle the complex input space.

6. Implementations

The main contributions of this thesis are several new structured output learning models for multilabel classification problems. Additionally, each developed model has been implemented into a software package. In this chapter, the author aims to briefly discuss the implementations and point out the locations from which the software packages can be found.

1. RTA, developed in Publication V, is a structured output learning algorithm for multilabel classification with an unknown output graph. RTA performs joint learning and inference on a random sample of spanning trees.

   (a) The learning system is implemented in MATLAB. The inference algorithm is implemented in C. The parallelization of the inference algorithm is implemented with OPENMP. Other parts of RTA are mostly implemented in MATLAB.

   (b) The package can be found from `http://git@github.com:hongyusu/random_spanning_tree_approximation.git`.

2. SPIN, developed in Publication I, is a structured output learning algorithm for multilabel classification with an observed output graph. SPIN can predict an optimal direct acyclic graph (DAG) that best responses to an input. The algorithm has been applied to the network response prediction problem within the context of social network analysis.

   (a) The learning system of SPIN is implemented in MATLAB. The SDP inference algorithm is implemented with CVX toolbox which is designed for convex programming. The data preprocessing with Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is implemented in PYTHON and MATLAB.

(b) The package can be found from `http://git@github.com:hongyusu/SPIN.git`.

3. MVE developed in Publication III as well as AMM and MAM developed in Publication IV are the structured output learning algorithms that are not constrained by the availability of the output graph. The algorithms combine a set of structured output learners built on a collection of random output graphs.

   (a) The learning systems are mostly implemented in MATLAB.

   (b) The packages can be found from `http://git@github.com:hongyusu/RandomOutputGraphEnsemble.git`.

7. Conclusion

7.1 Discussion

In this thesis, we have studied supervised learning for classification. In particular, we focused on multilabel classification where the task is to predict the best values for multiple interdependent output variables given an input example. As the multiple output variables can be "on" or "off" simultaneously, the central problem in multilabel classification is how to best exploit the correlation between labels in order to make accurate predictions. The problem has previously been tackled by the flat multilabel classification approaches which treat multiple output variables essentially as a "flat" vector. The approaches have difficulty of modeling the correlation between labels. Structured output learning arises as a natural extension to flat multilabel classification in which the correlation is modeled by an output graph connecting labels.

The first outcome of the thesis is a new structured output learning model for multilabel classification in which the output graph is known *apriori*. In particular, the proposed algorithm SPIN can predict a directed acyclic graph (DAG) from an observed underlying network which best "responds" to an input. The empirical evaluation on the network response prediction problem within the context of social network analysis shows that the proposed model outperforms several state-of-the-art flat multilabel classification approaches. The study demonstrates that accurate predictions can be achieved by structured output learning when the output graph is known and utilized during learning.

Meanwhile, we realize that the current structured output learning approaches rely on an output graph connecting multiple output variables in order to exploit the correlation between labels. Thus, the applicability

of structured output learning is limited due to the fact that the output graph needs to be known *apriori*. The second outcome of the thesis is that we have developed several new models for structured output learning which are no longer constrained by the availability of the output graph. Analog but with significant differences to the previously established ensemble methods, the proposed models aim to combine a set of structured output learners which are built on a collection of random output graphs. In particular, MVE uses majority vote to directly combine the predictions from the base learners, while AMM and MAM perform additional inference on the output graphs before or after combining the base learners. Meanwhile, we have developed RTA based on the theoretical study. The proposed model performs max-margin learning on a random sample of spanning trees. The joint learning and inference in RTA ensures that the base learners, which are built from a set of random spanning trees, are optimized simultaneously towards a same global objective. RTA has also laid the foundation of tackling the intractability of the graph inference on any unknown graph structures in which fast optimization and accurate predictions can be achieved with attainable computational efforts.

In addition to the practical learning algorithms, the thesis also contributes to the theoretical studies which not only explain the intuition behind the formalisms but also guarantee the generalization error of the proposed models.

7.2 Future Work

The work presented in the thesis can be extended along two main directions. First, the algorithms developed in this thesis can be applied to many other multilabel classification problems in which the output graph does not need to be observed but is believed to play an important role during learning. Secondly, the development of the learning algorithms and the theoretical studies are readily to be continued. As the first direction is straight-forward and application oriented, we will focus on the second part.

To serve as a starting point, the inference algorithm for SPIN can be further developed, which not only needs to be scalable to large scale social network datasets but should also have an approximation guarantee. Secondly, we plan to study RTA in multilabel classification problems with a general output graph. The setting is interesting as output variables

in many real world problems are usually organized by a graph structure which is more complex that a spanning tree or a chain but is less complex than a complete graph. The exact inference is often prohibitive for any polynomial time algorithms. In particular, we are interested in the change of the properties (e.g., the general error bound, the conditions for the exact inference) when we randomly sample spanning trees according to a general graph rather than a complete graph. Thirdly, we plan to investigate the possibility of learning the feature weight parameters defined on a random collection of spanning trees through a L_1-norm regularized combination. Learning with the L_1-norm regularized parameter combination has previously been studied in multiple kernel learning (Rakotomamonjy et al., 2008), which can be formulated into an equivalent form of learning a weighted L_2-norm combination. The additional weight can be seen as the affinity of an output graph to the training data, which can be used to select relevant output structures. We need to study the corresponding optimization algorithm as the current alternative optimization developed in (Rakotomamonjy et al., 2008) is not efficient. The theoretical studies of RTA should also be extended to the new algorithms.

Bibliography

Adar, E. and Adamic, L. A. (2005). Tracking information epidemics in blogspace. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '05, pages 207–214, Washington, DC, USA. IEEE Computer Society.

Adar, E., Zhang, L., Adamic, L., and Lukose, R. (2004). Implicit structure and the dynamics of blogspace. In *Workshop on the Weblogging Ecosystem*, volume 13.

Aha, D., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Amin, K., Heidari, H., and Kearns, M. (2014). Learning from contagion (without timestamps). In *Proceedings of the 31th International Conference on Machine Learning (ICML 2004)*, pages 823–830. JMLR.org.

Anderson, R. M. and May, R. M. (2002). *Infectious Diseases of Humans: Dynamics and Control*. Oxford Press.

Argyriou, A., Evgeniou, T., and Pontil, M. (2007). Multi-task feature learning. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 41–48. MIT Press.

Argyriou, A., Evgeniou, T., and Pontil, M. (2008a). Convex multi-task feature learning. *Machine Learning*, 73(3):243–272.

Argyriou, A., Maurer, A., and Pontil, M. (2008b). An algorithm for transfer learning in a heterogeneous environment. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2008)*, pages 71–85, Berlin, Heidelberg. Springer-Verlag.

Bartlett, P. L., Collins, M., Taskar, B., and McAllester, D. A. (2005). Exponentiated gradient algorithms for large-margin structured classification. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 113–120. MIT Press.

Barutcuoglu, Z., Schapire, R. E., and Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12(1):149–198.

Bell, R. M. and Koren, Y. (2007). Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79.

Ben-David, S. and Schuller, R. (2003). Exploiting task relatedness for multiple task learning. In Schölkopf, B. and Warmuth, M., editors, *Learning Theory and Kernel Machines*, volume 2777, pages 567–580. Springer Berlin Heidelberg.

Bengio, Y. (2009). Learning deep architectures for ai. *Foundation and Trends of Machine Learning*, 2(1):1–127.

Berger, A. (1999). The improved iterative scaling algorithm: A gentle introduction. *Machine Learning*.

Bernazzani, L., Duce, C., Micheli, A., Mollica, V., Sperduti, A., Starita, A., and Tine, M. (2006). Predicting physical-chemical properties of compounds from molecular structures by recursive neural networks. *Journal of Chemical Information and Modeling*, 46:2030–2042.

Bertsekas, D. P. (1995). *Nonlinear Programming*. Athena Scientific, Belmont, MA.

Bian, W., Xie, B., and Tao, D. (2012). Corrlog: Correlated logistic models for joint prediction of multiple labels. *Journal of Machine Learning Research - Proceedings Track*, pages 109–117.

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing 2011 edition.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Boley, D. and Cao, D. (2004). Training support vector machine using adaptive clustering. In *Proceedings of the 4th SIAM International Conference on Data Mining*.

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (1996b). Out-of-bag estimation. Technical report, Statistics Department, University of California, Berkeley.

Brinker, K. and Hüllermeier, E. (2007). Case-based multilabel ranking. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI 2007)*, pages 702–707, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Brown, G. and Kuncheva, L. I. (2010). "good" and "bad" diversity in majority vote ensembles. In *Multiple Classifier Systems*, pages 124–133. Springer.

Burbidge, R., Trotter, M., Buxton, B., and Holden, S. (2001). Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Computers and Chemistry*, 26(1):5 – 14.

Byvatov, E., Fechner, U., Sadowski, J., and Schneider, G. (2003). Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *Journal of Chemical Information and Computer Science*, 43:1882–1889.

Cai, L. and Hofmann, T. (2004). Hierarchical document categorization with support vector machines. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 78–87, New York, NY, USA. ACM.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

Ceroni, A., Costa, F., and Frasconi, P. (2007a). Classification of small molecules by two- and three-dimensional decomposition kernels. *Bioinformatics*, 23:2038–2045.

Ceroni, A., Costa, F., and Frasconi, P. (2007b). Classification of small molecules by two- and three-dimensional decomposition kernels. *Bioinformatics*, 23(16):2038–2045.

Cesa-bianchi, N., Gentile, C., Tironi, A., and Zaniboni, L. (2005). Incremental algorithms for hierarchical classification. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 233–240. MIT Press.

Chen, S. and Rosenfeld, R. (1999). *A Gaussian Prior for Smoothing Maximum Entropy Models*. PhD thesis, Computer Science Department, Carnegie Mellon University. Technical Report CMU-CS-99-108.

Chen, S. and Rosenfeld, R. (2000). A survey of smoothing techniques for me models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Cheng, W. and Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225.

Chiang, T.-H., Lo, H.-Y., and Lin, S.-D. (2012). A ranking-based knn approach for multi-label classification. In Hoi, S. C. H. and Buntine, W. L., editors, *ACML*, volume 25 of *JMLR Proceedings*, pages 81–96. JMLR.org.

Chickering, D. M., Geiger, D., and Heckerman, D. (1994). Learning bayesian networks is np-hard. Technical Report MSR-TR-94-17, Microsoft Research.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics.

Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002)*, pages 263–270, Stroudsburg, PA, USA. Association for Computational Linguistics.

Collins, M. and Koo, T. (2005). Discriminative reranking for natural language parsing. *Comput. Linguist.*, 31(1):25–70.

Cortes, C., Kuznetsov, V., and Mohri, M. (2014a). Ensemble methods for structured prediction. In *Proceedings of the 31th International Conference on Machine Learning (ICML 2004)*, pages 823–830. JMLR.org.

Cortes, C., Mohri, M., and Syed, U. (2014b). Deep boosting. In *Proceedings of the 31th International Conference on Machine Learning (ICML 2004)*, pages 823–830. JMLR.org.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Crammer, K., Singer, Y., K, J., Hofmann, T., Poggio, T., and Shawe-taylor, J. (2003). A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3:2003.

Daneshmand, H., Gomez-Rodriguez, M., Song, L., and Schoelkopf, B. (2014). Estimating diffusion network structures: Recovery conditions, sample complexity and soft-thresholding algorithm. In *Proceedings of the 31th International Conference on Machine Learning (ICML 2004)*, pages 823–830. JMLR.org.

Darroch, J. and Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(4):25–40.

DeCoro, C., Barutcuoglu, Z., and Fiebrink, R. (2007). Bayesian aggregation for hierarchical genre classification. In *International Symposium on Music Information Retrieval 2007*.

Decoste, D. and Schölkopf, B. (2002). Training invariant support vector machines. *Machine Learning*, 46(1-3):161–190.

Della Pietra, S., Della Pietra, V., and Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

Dembczynski, K., Cheng, W., and Hüllermeier, E. (2010). Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, pages 279–286. JMLR.org.

Domingos, P. and Richardson, M. (2001). Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 57–66, New York, NY, USA. ACM.

Drineas, P. and Mahoney, M. W. (2005). On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175.

Du, N., Liang, Y., Balcan, M.-F., and Song, L. (2014). Influence function learning in information diffusion networks. In *Proceedings of the 31th International Conference on Machine Learning (ICML 2004)*, pages 823–830. JMLR.org.

Du, N., Song, L., Yuan, M., and Smola, A. J. (2012). Learning networks of heterogeneous influence. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 2780–2788. Curran Associates, Inc.

Dumais, S. and Chen, H. (2000). Hierarchical classification of web content. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 256–263, New York, NY, USA. ACM.

Dutt, R.and Madan, A. K. (2012). Classification models for anticancer activity. *Machine Learning*, 12(23/24):2705.

Efron, B. and Tibshirani, R. (1994). An introduction to the bootstrap.

Elisseeff, A. and Weston, J. (2002). A kernel method for multi-labelled classification. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press.

Esuli, A., Fagni, T., and Sebastiani, F. (2008). Boosting multi-label hierarchical text categorization. *Information Retrieval*, 11(4):287–313.

Evgeniou, T., Poggio, T., Pontil, M., and Verri, A. (2002). Regularization and statistical learning theory for data analysis. *Comput. Stat. Data Anal.*, 38(4):421–432.

Evgeniou, T., Pontil, M., and Poggio, T. (1999). A unified framework for regularization networks and support vector machines. Technical report, Cambridge, MA, USA.

Fan, W. and Bifet, A. (2013). Mining big data: Current status, and forecast to the future. *SIGKDD Explor. Newsl.*, 14(2):1–5.

Fine, S. and Scheinberg, K. (2002). Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264.

Fiscus, J. (1997). A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *Proceedings of the 1997 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 347–354.

Freund, Y., Mansour, Y., and Schapire, R. E. (2004). Generalization bounds for averaged classifiers. *THE ANNALS OF STATISTICS*, 32:1698–1722.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139.

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., and Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gärtner, T. (2003). A survey of kernels for structured data. *SIGKDD Explor. Newsl.*, 5(1):49–58.

Getoor, L. (2005). Link-based classification. In *Advanced Methods for Knowledge Discovery from Complex Data*, Advanced Information and Knowledge Processing, pages 189–207. Springer London.

Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning*. the MIT Press.

Ghamrawi, N. and McCallum, A. (2005). Collective multi-label classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 195–200, New York, NY, USA. ACM.

Globerson, A. and Jaakkola, T. S. (2007). Approximate inference using planar graph decomposition. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 473–480. MIT Press.

Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In Dai, H., Srikant, R., and Zhang, C., editors, *Advances in Knowledge Discovery and Data Mining*, volume 3056, pages 22–30. Springer Berlin Heidelberg.

Goemans, M. and Williamson, D. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115 – 1145.

Gomez Rodriguez, M., Leskovec, J., and Krause, A. (2010). Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 1019–1028, New York, NY, USA. ACM.

Goodman, J. (2002). Sequential conditional generalized iterative scaling. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002)*, ACL '02, pages 9–16.

Goodman, J. (2003). Exponential priors for maximum entropy models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 305–312.

Goodman, J. and tau Yih, W. (2006). Online discriminative spam filter training. In *CEAS'06*, pages –1–1.

Gopal, S., Yang, Y., Bai, B., and Niculescu-mizil, A. (2012). Bayesian models for large-scale hierarchical classification. In Bartlett, P., Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 2420–2428.

Goyal, A., Bonchi, F., and Lakshmanan, L. V. (2010). Learning influence probabilities in social networks. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 241–250, New York, NY, USA. ACM.

Gruhl, D., Guha, R., Liben-Nowell, D., and Tomkins, A. (2004). Information diffusion through blogspace. In *Proceedings of the 13th International Conference on World Wide Web (WWW 2004)*, pages 491–501, New York, NY, USA. ACM.

Hao, P.-Y., Chiang, J.-H., and Tu, Y.-K. (2007). Hierarchically svm classification based on support vector clustering method and its application to document categorization. *Expert Syst. Appl.*, 33(3):627–635.

Hethcote, H. W. (2000). The mathematics of infectious diseases. *SIAM Review*, 42:599–653.

Hoerl, A. E. and Kennard, R. W. (2000). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86.

Huang, F.-L., Hsieh, C.-J., Chang, K.-W., and Lin, C.-J. (2009). Iterative scaling and coordinate descent methods for maximum entropy. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 285–288.

Jaakkola, T. S. and Haussler, D. (1999). Probabilistic kernel regression models. In *Proceedings of the 7th Workshop on Artificial Intelligent and Statistics (AISTATS 1999)*. Morgan Kaufmann.

Jacob, L., philippe Vert, J., and Bach, F. R. (2009). Clustered multi-task learning: A convex formulation. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 745–752. Curran Associates, Inc.

Jin, R., Yan, R., Zhang, J., and Hauptmann, A. G. (2003). A faster iterative scaling algorithm for conditional exponential model. In *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, pages 282–289. JMLR.org.

Joachims, T. (1998). Making large-scale svm learning practical. LS8-Report 24, Universität Dortmund, LS VIII-Report.

Joachims, T., Finley, T., and Yu, C.-N. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59.

Jordan, M. I. and Wainwright, M. J. (2004). Semidefinite relaxations for approximate inference on graphs with cycles. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 369–376. MIT Press.

Kashima, H., Tsuda, K., and Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. AAAI Press.

Kearns, M. and Valiant, L. (1989). Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95.

Keerthi, S., Duan, K., Shevade, S., and Poo, A. (2005). A fast dual algorithm for kernel logistic regression. *Machine Learning*, 61(1-3):151–165.

Keerthi, S. S., Chapelle, O., and DeCoste, D. (2006). Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493–1515.

Kempe, D., Kleinberg, J., and Tardos, E. (2003). Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA. ACM.

King, R. D., Muggleton, S. H., Srinivasan, A., and Sternberg, M. J. (1996). Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences*, 93(1):438–442.

Kocev, D., Vens, C., Struyf, J., and Deroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recogn.*, 46(3):817–833.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Koller, D. and Sahami, M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 170–178.

Koltchinskii, V. and Panchenko, D. (2000). Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30:2002.

Komarek, P. and Moore, A. (1999). Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz.

Komarek, P. and Moore, A. (2005). Making logistic regression a core data mining tool: A practical investigation of accuracy, speed, and simplicity. Technical Report CMU-RI-TR-05-27, Robotics Institute.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 8th International Conference on Machine Learning (ICML 2001)*, pages 282–289. Morgan Kaufmann Publishers Inc.

Le, Q., Sarlos, T., and Smola, A. (2013). Fastfood - approximating kernel expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pages 244–252. JMLR.org.

Leskovec, J., Backstrom, L., and Kleinberg, J. (2009). Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 497–506, New York, NY, USA. ACM.

Leskovec, J., McGlohon, M., Faloutsos, C., Glance, N., and Hurst, M. (2007). Cascading behavior in large blog graphs: Patterns and a model. In *Society of Applied and Industrial Mathematics: Data Mining (SDM07)*.

Li, T., Zhu, S., and Ogihara, M. (2007). Hierarchical document classification using automatically generated hierarchy. *J. Intell. Inf. Syst.*, 29(2):211–230.

Liben-Nowell, D. and Kleinberg, J. (2003). The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 556–559, New York, NY, USA. ACM.

Liben-Nowell, D. and Kleinberg, J. (2008). Tracing information flow on a global scale using Internet chain-letter data. *Proceedings of the National Academy of Sciences*, 105(12):4633–4638.

Lin, C.-J., Weng, R. C., and Keerthi, S. S. (2008). Trust region newton method for logistic regression. *Journal of Machine Learning Research*, 9:627–650.

Liu, T.-Y., Yang, Y., Wan, H., Zeng, H.-J., Chen, Z., and Ma, W.-Y. (2005). Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explor. Newsl.*, 7(1):36–43.

Menchetti, S., Costa, F., and Frasconi, P. (2005). Weighted decomposition kernels. In *Proceedings of the 22th International Conference on Machine Learning (ICML 2005)*, pages 585–592. ACM.

Minka, T. P. (2003). A comparison of numerical optimizers for logistic regression.

Mohri, M., Pereira, F., and Riley, M. (2008). Speech recognition with weighted finite-state transducers. In Benesty, J., Sondhi, M., and Huang, Y., editors, *Springer Handbook of Speech Processing*, pages 559–584. Springer Berlin Heidelberg.

Osuna, E., Freund, R., and Girosi, F. (1997). An improved training algorithm for support vector machines. In *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing [1997] VII*, pages 276–285.

Pavlov, D., Chudova, D., and Smyth, P. (2000). Towards scalable support vector machines using squashing. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 295–299, New York, NY, USA. ACM.

Petrov, S. (2010). Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 19–27, Stroudsburg, PA, USA. Association for Computational Linguistics.

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research.

Platt, J. C. (1999). Advances in kernel methods. chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pages 185–208. MIT Press, Cambridge, MA, USA.

Pérez-Cruz, O., Figueiras-Vidal, A. R., and Artés-Rodríguez, A. (2004). Double chunking for solving svms for very large datasets. In *Learning'04, Elche, Spain (2004)*.

Rakotomamonjy, A., , Bach, F., Canu, S., and Grandvalet, Y. (2008). implemkl. *Journal of Machine Learning Research 9*.

Ralaivola, L., Swamidass, S., Saigo, H., and Baldi, P. (2005). Graph kernels for chemical informatics. *Neural Networks*, 18:1093–1110.

Ratliff, N., Bagnell, J. A. D., and Zinkevich, M. (2007). (online) subgradient methods for structured prediction. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, volume 2. JMLR.org.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2009). Classifier chains for multi-label classification. In Buntine, W., Grobelnik, M., Mladenić, D., and Shawe-Taylor, J., editors, *Machine Learning and Knowledge Discovery in Databases*, volume 5782, pages 254–269. Springer Berlin Heidelberg.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359.

Richardson, M., Prakash, A., and Brill, E. (2006). Beyond pagerank: Machine learning for static ranking. In *Proceedings of the 15th International Conference on World Wide Web (WWW 2006)*, pages 707–715, New York, NY, USA. ACM.

Rodriguez, M. G., Balduzzi, D., and Schölkopf, B. (2011). Uncovering the temporal dynamics of diffusion networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, pages 561–568. ACM.

Rogers, E. M. (2003). *Diffusion of Innovations*. The Free Press, 5th ed. 2003 edition.

Romera-Paredes, B., Argyriou, A., Berthouze, N., and Pontil, M. (2012). Exploiting unrelated tasks in multi-task learning. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS 2012)*, volume 22, pages 951–959. JMLR.org.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *The Journal of Machine Learning Research*, 7:1601–1626.

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. (2007). Efficient algorithms for max-margin structured classification. *Predicting Structured Data*, pages 105–129.

Sagae, K. and Lavie, A. (2006). Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, Stroudsburg, PA, USA. Association for Computational Linguistics.

Saito, K., Nakano, R., and Kimura, M. (2008). Prediction of information diffusion probabilities for independent cascade model. In Lovrek, I., Howlett, R., and Jain, L., editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 5179, pages 67–75. Springer Berlin Heidelberg.

Schapire, R. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Schapire, R. E., Freund, Y., Barlett, P., and Lee, W. S. (1997). Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the 14th International Conference on Machine Learning (ICML 1997)*, pages 322–330. Morgan Kaufmann Publishers Inc.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels - Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Shoemaker, R. H. (2006). The NCI60 human tumour cell line anticancer drug screen. *Nat Rev Cancer*, 6(10):813–823.

Si, S., jui Hsieh, C., and Dhillon, I. (2014). Memory efficient kernel approximation. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, pages 701–709. JMLR.org.

Silla, Jr., C. N. and Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22(1-2):31–72.

Singh, N., Chaudhury, S., Liu, R., AbdulHameed, M. D. M., Tawa, G., and Wallqvist, A. (2012). Qsar classification model for antibacterial compounds and its use in virtual screening. *Journal of Chemical Information and Modeling*, 52(10):2559–2569.

Smola, A. J. and Schökopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 911–918. Morgan Kaufmann Publishers Inc.

Smyth, P. and Wolpert, D. (1999). Linearly combining density estimators via stacking. *Machine Learning*, 36(1-2):59–83.

Strang, D. and Soule, S. A. (1998). Diffusion in Organizations and Social Movements: From Hybrid Corn to Poison Pills. *Annual Review of Sociology*, 24(1):265–290.

Swamidass, S., Chen, J., Bruand, J., Phung, P., Ralaivola, L., and Baldi, P. (2005). Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21:359–368.

Taskar, B., Abbeel, P., and Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI 2002)*, pages 485–492, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Taskar, B., Guestrin, C., and Koller, D. (2004). Max-margin markov networks. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press.

Taskar, B., Lacoste-Julian, S., and Jordan, M. I. (2006). Structured prediction via the extragradient method. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems 18*, pages 1345–1352. MIT Press.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

Tibshirani, R. (1996). Bias, variance and prediction error for classification rules.

Trotter, M., Buxton, M., and Holden, S. (2001). Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Journal of Computational Chemistry.*, 26:1–20.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21th International Conference on Machine Learning (ICML 2004)*, pages 823–830. ACM.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2010). Mining multi-label data. In Maimon, O. and Rokach, L., editors, *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer US.

Tsoumakas, G. and Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the 2007 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2007)*, pages 406–417. Springer Berlin Heidelberg.

Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag.

Vapnik, V. (1992). Principles of risk minimization for learning theory. In Moody, J., Hanson, S., and Lippmann, R., editors, *Advances in Neural Information Processing Systems 4*, pages 831–838. Morgan-Kaufmann.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Network*, 10(5):988–999.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242.

Wainwright, M., Jaakkola, T., and Willsky, A. (2005). Map estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717.

Wainwright, M. J. and Jordan, M. I. (2003). Graphical models, exponential families, and variational inference. *Foundation and Trends in Machine Learning*, 1(1-2):1–305.

Wang, Q. I., Lin, D., and Schuurmans, D. (2007). Simple training of dependency parsers via structured boosting. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1756–1762, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Wang, Y. and Mori, G. (2011). Hidden part models for human action recognition: Probabilistic versus max margin. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1310–1323.

Wang, Y., Xiao, J., Suzek, T. O., Zhang, J., Wang, J., and Bryant, S. H. (2009). Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucleic Acids Research*, 37(suppl 2):W623–W633.

Wolpert, D. and Macready, W. (1999). An efficient method to estimate bagging's generalization error. *Machine Learning*, 35(1):41–55.

Xue, Y., Li, Z., Yap, C., Sun, L., Chen, X., and Chen, Y. (2004). Effect of molecular descriptor feature selection in support vector machine classification of pharmacokinetic and toxicological properties of chemical agents. *Journal of Chemical Information and Computer Science*, 44:1630–1638.

Yan, R., Tesic, J., and Smith, J. R. (2007). Model-shared subspace boosting for multi-label classification. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 834–843, New York, NY, USA. ACM.

Younes, Z., Abdallah, F., Denoeux, T., and Snoussi, H. (2011). A dependent multilabel classification method derived from the k-nearest neighbor rule. *EURASIP J. Adv. Sig. Proc.*, 2011.

Yu, H., Yang, J., Han, J., and Li, X. (2005). Making svms scalable to large data sets using hierarchical cluster indexing. *Data Min. Knowl. Discov.*, 11(3):295–321.

Yu, H.-F., Huang, F.-L., and Lin, C.-J. (2011). Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75.

Zeman, D. and Žabokrtský, Z. (2005). Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 171–178, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhang, H., Zhang, M., Tan, C. L., and Li, H. (2009). K-best combination of syntactic parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1552–1560, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhang, K., Lan, L., Wang, Z., and Moerchen, F. (2012). Scaling up kernel svm on limited resources: A low-rank linearization approach. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2012)*, volume 22, pages 1425–1434.

Zhang, K., Tsang, I. W., and Kwok, J. T. (2008). Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 1232–1239. ACM.

Zhang, M. and Zhou, Z. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *Proceedings of the 2005 IEEE International Conference on Granular Computing*, volume 2, pages 718–721 Vol. 2.

Zhang, M. and Zhou, Z. (2007). Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40:2007.

Zhang, M.-L. and Zhou, Z.-H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

Zhang, M.-L. and Zhou, Z.-H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.

Publication I

Hongyu Su, Aristides Gionis, Juho Rousu. Structured Prediction of Network Response. In *Proceedings of the* 31^{th} *International Conference on Machine Learning (ICML 2014)*, Beijing, China, 2014. JMLR W&CP volume 32:442-450, June 2014.

Structured Prediction of Network Response

Hongyu Su HONGYU.SU@AALTO.FI
Aristides Gionis ARISTIDES.GIONIS@AALTO.FI
Juho Rousu JUHO.ROUSU@AALTO.FI
Helsinki Institute for Information Technology (HIIT)
Department of Information and Computer Science, Aalto University, Finland

Abstract

We introduce the following *network response* problem: given a complex network and an action, predict the subnetwork that responds to action, that is, which nodes perform the action and which directed edges relay the action to the adjacent nodes.

We approach the problem through max-margin structured learning, in which a compatibility score is learned between the actions and their activated subnetworks. Thus, unlike the most popular influence network approaches, our method, called SPIN, is *context-sensitive*, namely, the presence, the direction and the dynamics of influences depend on the properties of the actions. The inference problems of finding the highest scoring as well as the worst margin violating networks, are proven to be **NP**-hard. To solve the problems, we present an approximate inference method through a semi-definite programming relaxation (SDP), as well as a more scalable greedy heuristic algorithm.

In our experiments, we demonstrate that taking advantage of the context given by the actions and the network structure leads SPIN to a markedly better predictive performance over competing methods.

1. Introduction

With the widespread use and extensive availability of large-scale networks, an increasing amount of research has been proposed to study the structure and function of networks. In particular, network analysis has been applied to study dynamic phenomena and complex interactions, such as information propagation, opinion formation, adoption of technological innovations, viral marketing, and disease spreading (De Choudhury et al., 2010; Kempe et al., 2003; Watts & Dodds, 2007).

Influence models typically consider *actions* performed by the network nodes. Examples of such actions include buying a product or (re)posting a news story in one's social network. Often, network nodes perform such actions as a result of influence from neighbouring nodes, and a number of different models have been proposed to quantify influence in a network, most notably the independent-cascade and the linear-threshold models (Kempe et al., 2003). On the other hand, performing an action may also come as a result to an external (out of the network) stimulus, a situation that has also been subject to modeling and analysis (Anagnostopoulos et al., 2008). A typical assumption made by existing models is that influence among nodes depends only on the nodes that perform the action and not on the action itself.

A central question in the study of network influence, is to infer the latent structure that governs the influence dynamics. This question can be formulated in different ways. In one case no underlying network is available (for example, news agencies that do not link each other) and one asks to infer the hidden network structure, e.g., to discover implicit edges between the network nodes (De Choudhury et al., 2010; Du et al., 2012; Eagle et al., 2009; Gomez-Rodriguez et al., 2010; 2011). However, this problem is an unnecessarily hard one to solve in many applications. On the other hand, in many applications the network is known (e.g., "follower" links in twitter), and the research question is to estimate the hidden variables of the influence model (Goyal et al., 2010; Saito et al., 2008).

The present paper is motivated by the following observation: the influence between two nodes in the network does not depend only on the nodes and their connections, but also depends on the action under consideration. For example, if u and v represent users in twitter, v may be influenced from u regarding topics related to *science* but not

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

regarding topics related to, say, *politics*. Thus, in our view, the influence model needs to be *context-sensitive*.

We thus consider the following *network response* problem: given an action, predict which nodes in the network will perform it and along which edges the action will spread. We approach the problem via structured output learning that models the activated *response network* as a directed graph. We learn a function for mappings between action descriptions and the response subnetwork. Given an action, the model is able to predict a directed subnetwork that is most favourable to performing the action.

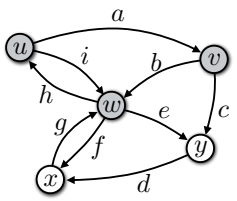
2. Preliminaries

We consider a *directed* network $G = (V, E)$ where the nodes $v \in V$ represent entities, and edges $e = (u, v) \in E$ represent relationships among entities. As discussed in the introduction, for each edge (u, v) we assume that node v can be influenced by node u. In real applications, some networks are directed (e.g., follower networks), while other networks are undirected (e.g., friendship networks). For simplicity of exposition, and without loss of generality we formulate our problem for directed networks; indeed an undirected edge can be modeled by considering pair of directed edges. In our experiments we also consider undirected networks.

In addition to other nodes, we allow the nodes to be influenced by external stimuli, modelled by a *root* node r, which is connected to all other nodes in the network, namely $(r, v) \in E$, for all $v \in V \setminus \{r\}$. Reversely, no node can influence r, so $(v, r) \notin E$, for all $v \in V \setminus \{r\}$.

The second ingredient of our model consists of the actions performed by the network nodes. We write A to denote the underlying *action space*, that is, the set of all possible actions, and we use $\mathbf{a}$ to indicate a particular action in A. We assume that actions in A are represented using a feature map $\phi : A \to \mathcal{F}_A$ to an associated inner product space $\mathcal{F}_A$. For example, $\mathcal{F}_A$ can be a vector space of dimension k, where each action $\mathbf{a}$ is represented by a k-dimensional vector $\phi(\mathbf{a})$. In the social-network application discussed in the introduction, where actions $\mathbf{a}$ correspond to news articles posted by users, $\phi(\mathbf{a})$ can be the bag-of-words representation of the news article $\mathbf{a}$.

We assume that the network gets exposed to an action $\mathbf{a} \in A$, and in response a subgraph $G_\mathbf{a} = (V_\mathbf{a}, E_\mathbf{a}) \subseteq G$, called the *response network* gets activated. The nodes $V_\mathbf{a} \subseteq V$ are the ones that get activated and $E_\mathbf{a} \subseteq E$ is the set of induced edges. We assume that the root r is always activated, i.e., $r \in V_\mathbf{a}$. Note that even though r is directly connected to each node $v \in V_\mathbf{a}$, in every response network $G_\mathbf{a}$, some nodes in $V_\mathbf{a}$ may exercise on v stronger influence than the influence that r exercises on v. The nodes that get directly activated by the root node r as a response to an action are called the *focal points* or *foci* of the response network.



$(u, \mathbf{a}, t = 1)$
$(v, \mathbf{a}, t = 2)$
$(w, \mathbf{a}, t = 5)$

$\mathbf{a} \to \phi(\mathbf{a})$

$$G_\mathbf{a} \to \psi(G_\mathbf{a}) = (a_{\mathtt{pp}}, a_{\mathtt{pn}}, a_{\mathtt{nn}}, b_{\mathtt{pp}}, b_{\mathtt{pn}}, b_{\mathtt{nn}}, c_{\mathtt{pp}}, c_{\mathtt{pn}}, c_{\mathtt{nn}}, d_{\mathtt{pp}}, d_{\mathtt{pn}}, d_{\mathtt{nn}}, \ldots)$$
$$= (\ 1,\ \ 0,\ \ 0,\ \ 1,\ \ 0,\ \ 0,\ \ 0,\ \ 1,\ \ 0,\ \ 0,\ \ 0,\ \ 1,\ \ \ldots)$$

$$\gamma(u) = (\gamma(u; a), \gamma(u; b), \gamma(u; c), \gamma(u; d), \ldots) = (1, \lambda, \lambda, \lambda^2, \ldots)$$

Figure 1. An action $\mathbf{a}$ perfromed by nodes u, v, w of a directed network at times $1, 2, 5$, respectively. Nodes x and y do not perform the action. The action $\mathbf{a}$ is represented by input feature map $\phi(\mathbf{a})$. The response network $G_\mathbf{a}$ is represented by output feature map $\psi(G_\mathbf{a})$ that encodes the propagation of the action $\mathbf{a}$ with respect to edge e (details in the text). Finally, γ is a scaling function (see Sec. 3.4). For instance, $\gamma(u)$ represents a vector of exponentially-decaying weights for node u with respect to all edges.

We assume a dataset $\{(\mathbf{a}_i, G_{\mathbf{a}_i})\}_{i=1}^m$ of m training examples, where each example $(\mathbf{a}_i, G_{\mathbf{a}_i})$ consists of an action $\mathbf{a}_i$ and the output $G_{\mathbf{a}_i}$ encoding the response network activated by $\mathbf{a}_i$. Our intention is to build a model that given a previously unobserved action $\mathbf{a}$, predicts the response network $G_\mathbf{a}$.

3. Model for network responses

3.1. Structured-prediction model

Our method is based on embedding the input and output into a joint feature space and learning in that space a linear compatibility score

$$F(\mathbf{a}, G_\mathbf{a}; \mathbf{w}) = \langle \mathbf{w}, \varphi(\mathbf{a}, G_\mathbf{a}) \rangle.$$

The score $F(\mathbf{a}, G_\mathbf{a}; \mathbf{w})$ is given by the inner product of parameters $\mathbf{w}$ and the joint feature $\varphi(\mathbf{a}, G_\mathbf{a})$. As the joint feature we will use the tensor product $\varphi(\mathbf{a}, G_\mathbf{a}) = \phi(\mathbf{a}) \otimes \psi(G_\mathbf{a})$ of the input feature map $\phi(\mathbf{a})$ of action $\mathbf{a}$, and the output feature map $\psi(G_\mathbf{a})$ that represents the response network $G_\mathbf{a}$ to the action $\mathbf{a}$. The tensor product $\varphi(\mathbf{a}, G_\mathbf{a})$ consists of all pairs of input and output features $\varphi_{ij}(\mathbf{a}, G_\mathbf{a}) = \phi_i(\mathbf{a})\psi_j(G_\mathbf{a})$.

The output features will encode the activated subgraph in the network. We use labels $\{\mathtt{p}, \mathtt{n}\}$ to indicate whether nodes perform an action (positive vs. negative). Similarly, we use edge labels $\{\mathtt{pp}, \mathtt{pn}, \mathtt{nn}\}$ to indicate the role of edges in the propagation of actions. In particular, for each edge $(u, v) = e$ of a response network $G_\mathbf{a}$ and each label $\ell \in \{\mathtt{pp}, \mathtt{pn}, \mathtt{nn}\}$ we define the feature $\psi_{e,\ell}(G_\mathbf{a})$ to

be 1 if and only if e is of type ℓ in $G_{\mathbf{a}}$ (and 0 otherwise). For example, $\psi_{(u,v),\mathtt{pp}}(G_{\mathbf{a}}) = 1$ indicates that both nodes u and v are activated in $G_{\mathbf{a}}$ and u precedes v in the partial order of activation.

An example of the model is shown in Figure 1. For the sake of brievity in the figure, we abuse notation and we use e_ℓ to denote $\psi_{e,\ell}(G_{\mathbf{a}})$. For instance, in this example we have $a_{\mathtt{pp}} = (u,v)_{\mathtt{pp}} = 1$ since both u and v are activated and u precedes v in the activation order, and thus it is possible that u has influenced v.

3.2. Maximum-margin structured learning

The feature weight parameters $\mathbf{w}$ of the compatibility score function F are learned by solving a regularized structured-output learning problem

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}\,||\mathbf{w}||_2^2 + C\sum_{i=1}^{m}\xi_i, \qquad (1)\\ \text{s.t.} \quad & F(\mathbf{a}_i, G_{\mathbf{a}_i}; \mathbf{w}) > \operatorname*{\mathbf{argmax}}_{G'_{\mathbf{a}_i}\in\mathcal{H}(G)} \big(F(\mathbf{a}_i, G'_{\mathbf{a}_i}; \mathbf{w}) \\ & +\ell_G(G'_{\mathbf{a}_i}, G_{\mathbf{a}_i})\big) - \xi_i, \xi_i \geq 0, \forall i = \{1,\cdots,m\}. \end{aligned}$$

The impact of the constraints on the above optimization problem is to push the compatibility score of input $\mathbf{a}_i$ with output $G_{\mathbf{a}_i}$ above the scores of all competing outputs $G'_{\mathbf{a}_i} \in \mathcal{H}(G)$ with a margin proportional to the loss $\ell_G(G'_{\mathbf{a}_i}, G_{\mathbf{a}_i})$ between the correct $G_{\mathbf{a}_i}$ and any competing subgraph $G'_{\mathbf{a}_i}$. $\mathcal{H}(G)$ is the set of directed acyclic subgraphs of G rooted at r. The slack variable ξ_i is used to relax the constraints so that a feasible solution can always be found. C is a slack parameter that controls the amount of regularization in the model. The objective minimizes an L_2-norm regularizer of the weight vector and the slack allocated to the training set. This is equivalent to maximizing the margin subject to allowing some data to be outliers. In practice, the optimization problem (Eq. 1) is tackled by marginal dual conditional gradient optimization (Rousu et al., 2007).

3.3. The inference problem

In the structured prediction model, both in training and in prediction, we need to solve the problem of finding the highest-scoring subgraph for an action. The two problems differ only in the definition of the score: in training we need to iteratively find the subgraph that violates its margins the most, whilst in prediction we need to find the subgraph with the maximum compatibility to a given action. We explain our inference algorithms for the latter problem and note that the first problem is a straightforward variant.

Given feature weights $\mathbf{w}$ and a network $G = (V, E)$, the prediction for a new input action $\mathbf{a}$ is the maximally-scoring response graph $H^* = (V^H, E^H)$

$$H^*(\mathbf{a}) = \operatorname*{\mathbf{argmax}}_{H\in\mathcal{H}(G)} F(\mathbf{a}, H; \mathbf{w}).$$

Writing this problem explicitly, in terms of the parameters and the feature maps gives

$$\begin{aligned} H^*(\mathbf{a}) &= \operatorname*{\mathbf{argmax}}_{H\in\mathcal{H}(G)} \langle \mathbf{w}, \phi(\mathbf{a}) \otimes \psi(H)\rangle \\ &= \operatorname*{\mathbf{argmax}}_{H\in\mathcal{H}(G)} \sum_{e\in E^H} s_{y_e}(e, \mathbf{a}), \qquad (2) \end{aligned}$$

where we have substituted $s_{y_e}(e,\mathbf{a}) = \sum_i w_{i,e,y_e}\phi_i(\mathbf{a})$. We will abbreviate $s_{y_e}(e,\mathbf{a})$ to $s_{y_e}(e)$, as the action $\mathbf{a}$ is fixed for an individual inference problem. The output response network H can be specified by a node label $y_v \in \{\mathtt{p},\mathtt{n}\}$, where $y_v = \mathtt{p}$ if and only if v is activated. We write H_y to emphasize the dependence of the output subgraph H from labelling y. The node labels y_v induce edge labels y_e. The score function $s(e)$ can be interpreted as a score function for the edges, given by the current input $\mathbf{a}$ and weight vector $\mathbf{w}$. The variable y_e indicates the possible labels of an edge e, and for each possible label the score function $s(e)$ assigns a different score. Depending on the values that y_e can take, the inference problem can be further diverged into two modes:

Activation mode. We assume $y_e \in \{\mathtt{pp},\mathtt{pn}\}$ where $y_e = \mathtt{pp}$ implies node v is activated by u via a directed edge $e = (u,v)$, and $y_e = \mathtt{pn}$ means that the activation cannot pass through e. In activation mode, the inference problem is transformed as finding the maximally scoring node label y_v and corresponding edge label y_e, consistent with an activated subgraph H_y given a set of edge scores $s_{y_e}(e)$.

Negative-feed mode. In addition to the setting in activation mode, we also explicitly model the inactive network by assume $y_e \in \{\mathtt{pp},\mathtt{pn},\mathtt{nn}\}$, where by $y_e = \mathtt{nn}$ we denote our belief that both u and v should be inactive given action $\mathbf{a}$. The inference problem is then to find the maximally scoring node labels and induced edge labels with regards to an activated subgraph together with the inactive counterpart given a set of edge score $s_{y_e}(e)$.

It is not difficult to show that the inference problem (Eq. 2) is **NP**-hard. The proof of the following lemma, which provides a reduction from the MAX-CUT problem, is given in the supplementary material.

Lemma 1 *Finding the graph that maximizes Eq. (2) is an* **NP***-hard problem.*

To solve the inference problem we propose two algorithms, described on the negative-feed mode. Similar techniques can be adapted to the activation-mode by setting edge score $s_{\mathtt{nn}}(e) = 0$. The first algorithm is based on a semidefinite programming (SDP) relaxation, similar to the one

used for MAX-CUT and satisfiability problems (Goemans & Williamson, 1995). The SDP algorithm offers a constant-factor approximation guarantee for the inference problem. However, it requires solving semidefinite programs. Efficient solvers do exist, but the method is not scalable to large datasets. Besides, it cannot handle the order of activations. In contrast, our second approach is a more efficient GREEDY algorithm that models activation order in a natural way, but it does not provide any quality guarantee.

The SDP inference. Recall that for each edge $(u, v) \in E$ we are given three scores: $s_{\mathtt{pp}}(u, v)$, $s_{\mathtt{pn}}(u, v)$, and $s_{\mathtt{nn}}(u, v)$. The inference problem is to assign a label p or n for each vertex $u \in V$. If a vertex u is assigned to label p we say that u is *activated*. If both vertices u and v of an edge $(u, v) \in E$ are activated, a gain $s_{\mathtt{pp}}(u, v)$ incurs. Respectively, the assignments pn and nn yield gains $s_{\mathtt{pn}}(u, v)$ and $s_{\mathtt{nn}}(u, v)$. The objective is to find the assignments that maximizes the total gain.

We formulate this optimization problem as a quadratic program. We introduce a variable $x_u \in \{-1, +1\}$, for each $u \in V$. We also introduce a special variable $x_0 \in \{-1, +1\}$, which is used to distinguish the activated vertices. In particular, if $x_u = x_0$ we consider that the vertex u is assigned to label p, and thus it is activated, while $x_u = -x_0$ implies that u is assigned to n and not activated. The network-response inference problem can now be written as (QP):

$$\begin{aligned}\textbf{max}\ \frac{1}{4} \sum_{(u,v)\in E} & [s_{\mathtt{pn}}(u, v)(1 + x_0x_u - x_0x_v - x_ux_v) \\ & + s_{\mathtt{nn}}(u, v)(1 - x_0x_u - x_0x_v + x_ux_v) \\ & + s_{\mathtt{pp}}(u, v)(1 + x_0x_u + x_0x_v + x_ux_v)], \\ \text{s.t.}\quad & x_0, x_u, x_v \in \{-1, +1\}, \text{ for all } u, v \in V.\end{aligned}$$

The intuition behind the formulation of Problem (QP) is that there is gain $s_{\mathtt{pn}}(u, v)$ if $x_0 = x_u = -x_v$, a gain $s_{\mathtt{nn}}(u, v)$ if $x_0 = -x_u = -x_v$, and a gain $s_{\mathtt{pp}}(u, v)$ if $x_0 = x_u = x_v$.

To solve the problem (QP), we use the similar technique introduced by Goemans & Williamson (1995), such that each variable x_u is *relaxed* to a *vector* $\mathbf{v}_u \in \mathbb{R}^n$. The relaxed quadratic program becomes (RQP):

$$\begin{aligned}\textbf{max}\ \frac{1}{4} \sum_{(u,v)\in E} & [s_{\mathtt{pn}}(u, v)(1 + \mathbf{v}_0\mathbf{v}_u - \mathbf{v}_0\mathbf{v}_v - \mathbf{v}_u\mathbf{v}_v) \\ & + s_{\mathtt{nn}}(u, v)(1 - \mathbf{v}_0\mathbf{v}_u - \mathbf{v}_0\mathbf{v}_v + \mathbf{v}_u\mathbf{v}_v) \\ & + s_{\mathtt{pp}}(u, v)(1 + \mathbf{v}_0\mathbf{v}_u + \mathbf{v}_0\mathbf{v}_v + \mathbf{v}_u\mathbf{v}_v)], \\ \text{s.t.}\quad & \mathbf{v}_i \in \mathbb{R}^n, \text{ for all } i = 0, \ldots, n.\end{aligned}$$

Consider an $(n+1) \times (n+1)$ matrix Y whose (u, v) entry is $y_{u,v} = \mathbf{v}_u \cdot \mathbf{v}_v$. If V is the matrix having $\mathbf{v}_u$'s as its columns, i.e., $V = [\mathbf{v}_0 \ldots \mathbf{v}_k]$, then $Y = V^TV$, implying that the matrix Y is semidefinite, a fact we denote by $Y \succeq 0$. Problem (RQP) now becomes (SDP):

$$\begin{aligned}\textbf{max}\ \frac{1}{4} \sum_{u,v=1}^{k} & [s_{\mathtt{pn}}(u, v)(1 + y_{0,u} - y_{0,v} - y_{u,v}) \\ & + s_{\mathtt{nn}}(u, v)(1 - y_{0,u} - y_{0,v} + y_{u,v}) \\ & + s_{\mathtt{pp}}(u, v)(1 + y_{0,u} + y_{0,v} + y_{u,v})], \\ \text{s.t.}\quad & Y \succeq 0.\end{aligned}$$

Problem (SDP) asks to find a semidefinite matrix, so that a linear function on the entries of the matrix is optimized. This problem can be solved by semidefinite programming within accuracy ϵ, in time that it is polynomial on k and $\frac{1}{\epsilon}$. After solving the semidefinite program one needs to *round* each vector $\mathbf{v}_u$ to the variable $x_u \in \{-1, +1\}$ in the following way:

1. Factorize Y with Cholesky decomposition to find $V = [\mathbf{v}_0, \mathbf{v}_1 \ldots \mathbf{v}_n]$.
2. Select a random vector $\mathbf{r}$.
3. For each $u = 0, 1, \ldots, n$, if $\mathbf{v}_u \cdot \mathbf{r} \geq 0$ set $x_u = 1$, otherwise set $x_u = -1$.

Let Z be the value of the solution obtained by the above algorithm. Let Z^* be the optimal value of Problem (QP) and Z_R the optimal value of Problem (SDP). Since Problem (SDP) is a relaxation of Problem (QP) it is $Z_R \geq Z^*$. Furthermore, it can be shown that for the *expected* value of Z it holds $E[Z] \geq (\alpha - \epsilon)Z_R$, with $\alpha > 0.796$ and where expectation is taken over the choice of $\mathbf{r}$. Thus the above algorithm is a 0.796 approximation algorithm for Problem (QP).

The GREEDY inference. The inference (Eq. 2) is defined on all edges of the network, which can be expressed equivalently as a function of activated vertices (see details in supplementary)

$$H^*(\mathbf{a}) = \underset{H \in \mathcal{H}(G)}{\textbf{argmax}} \sum_{v_i \in V_{\mathtt{p}}^H} F_m(v_i),$$

where $V_{\mathtt{p}}^H$ is a set of activated vertices. $F_m(v_i)$ is the marginal gain on each node that is comprised partially from changing edge label from pn to pp on incoming edges $\{(v_p, v_i) \mid v_p \in \text{parents}(v_i)\}$, and partially from changing edge label from nn to pn on outgoing edges $\{(v_i, v_c) \mid v_c \in \text{parents}(v_i)\}$ defined as

$$\begin{aligned}F_m(v_i) = & \sum_{v_p \in \text{parents}(v_i)} [s_{\mathtt{pp}}(v_p, v_i) - s_{\mathtt{pn}}(v_p, v_i)] \\ & + \sum_{v_c \in \text{children}(v_i)} [s_{\mathtt{pn}}(v_i, v_c) - s_{\mathtt{nn}}(v_i, v_c)].\end{aligned}$$

It is difficult to maximize the sum of marginal gains as the activated subnetwork is unknown. One can instead compute for each vertex the maximized marginal gain $\mathbf{max}_{v_i} F_m(v_i)$ in an iterative fashion as long as $F_m(v_i) \geq 0$, which leads to a greedy algorithm described as follows. The algorithm starts with an activated vertex set $V_{\mathtt{p}}^H = \{r\}$. In each iteration, it chooses a vertex $v_i \in V/V_{\mathtt{p}}^H$ and adds to $V_{\mathtt{p}}^H$ such that v_i is the current maximizer of $F_m(v)$. The procedure terminates if the maximized gain is smaller than 0. E^H can be obtained by adding edges $e = (v_i, v_j) \in E$, if $v_i, v_j \in V_{\mathtt{p}}^H$ and v_i was added to $V_{\mathtt{p}}^H$ prior to v_j. The time complexity for greedy inference algorithm is $O(|E| \log |V|)$. See supplementary material for details of the algorithm.

We note that we have not been able to show an approximation guarantee for the quality of solutions produced by the GREEDY algorithm. A property that it is typically used to analyse greedy methods is *submodularity*. However, for this particular problem submodularity does not hold (it only holds in the special case of MAX-CUT, i.e., when $s_{\mathtt{pp}}(e) = s_{\mathtt{nn}}(e) = 0$ and $s_{\mathtt{pn}}(e) = 1$).

3.4. Loss functions

Instead of penalizing prediction mistakes uniformly on the network G, we wish to focus in the vicinity of the response network. To achieve this effect we scale the loss accrued on the nodes and edges by their distance to the children of the root of the response network.

As the loss function in (1) we use *symmetric-difference loss* (or Hamming loss), applied to the nodes and the edges of the subgraphs separately, and scaled by function $\gamma_G(v_k)$ according distance to the focal point v_k.

$$\ell_G(G_{\mathbf{a}}, G_{\mathbf{b}}) = \sum_{v \in V} \ell_v^{\Delta}(G_{\mathbf{a}}, G_{\mathbf{b}}) \gamma_G(v_k; v) + \sum_{(v,v') \in E} \ell_{v,v'}^{\Delta}(G_{\mathbf{a}}, G_{\mathbf{b}}) \gamma_G(v_k; v),$$

where $\ell_v^{\Delta}(G_{\mathbf{a}}, G_{\mathbf{b}}) = [v \in V_a \Delta V_b]$, $\ell_e^{\Delta}(G_{\mathbf{a}}, G_{\mathbf{b}}) = [e \in E_a \Delta E_b]$, $S \Delta S'$ denotes the symmetric difference of two sets S and S'. We consider the following strategies to construct the scaling function $\gamma_G(v_k)$:

Exponential scaling. Mistakes are penalized by λ and λ is weighted exponentially according to the shortest path distance to the focal point v_k. Given focal point v_k, edge (v_i, v_j), and distance matrix D between the nodes, the scaling function is defined as

$$\gamma_G(v_k; v_i, v_j) = \begin{cases} 1 & \text{if } i = 0 \\ \lambda^{D(k,i)} & \text{if } i \neq 0 \text{ and } D(k,i) \leq R \\ \lambda^{(R+1)} & \text{if } D(k,i) > R \end{cases}$$

where $\lambda > 0$ is the scaling factor and $R > 1$ is a radius parameter. Edges outside the radius have equal scalings.

Diffusion scaling. The diffusion kernel defines a distance-based function between nodes v_i and v_j (Kondor & Lafferty, 2002). The kernel value $K(i, j)$ corresponds to the probability of a random walk from node v_i to node v_j. Given the adjacency matrix L of the network G, the diffusion kernel is computed as

$$K = \lim_{s \to \infty} \left(I + \frac{\beta L}{s} \right)^s = \exp(\beta L),$$

where I is the identity matrix and β is the a parameter that controls how much the random walks deviate from the focal point. Given focal node v_k, edge (v_i, v_j), and diffusion kernel K the scaling function is defined as

$$\gamma_G(v_k; v_i, v_j) = \begin{cases} 1 & \text{if } i = 0, \\ K(v_k, v_i) & \text{otherwise.} \end{cases}$$

The scaling function keeps the loss value on the edges connecting the focal point, and scale other edges by the weights computed from diffusion kernel. Diffusion scaling has the effect of shrinking the distance to nodes that connects to the focal point by many paths.

4. Experimental evaluation

In this section, we evaluate the performance of SPIN and compare it with the state-of-the-art methods through extensive experiments. We use two real-world datasets, DBLP and Memetracker, described below. Statistics of the datasets are given in Table 1.

DBLP[1] dataset is a collection of bibliographic information on major computer science journals and proceedings. We extract a subset of original data by using "inproceedings" articles from year 2000. First, we construct an undirected DBLP network G by connecting pairs of authors who have coauthored more that p papers ($p = 5, 10, 15$). After that, we generate a set of experimental networks of different size by performing snowball sampling (Goodman, 1961). For each experimental network, we extract all the documents for which at least one of their authors is a node in the network. We apply LDA algorithm (Blei et al., 2002) on the titles of extracted documents to generated topics. Topics are associated with publications, timestamped by publication dates, and described by bag-of-word features computed from LDA. In this way, a topic can be seen as an action and we will study the influence among authors.

Memetracker[2] dataset is a set of phrases propagated over prominent online news sites in March 2009. We construct

[1] http://www.informatik.uni-trier.de/~ley/db/

[2] http://Memetracker.org

directed networks G for Memetracker dataset by connecting two websites via a directed edge if there are at least five phrases copying from one website to the other. A posted phrase corresponds to an action, which again is timestamped and represented with bag-of-word features.

4.1. Experimental setup and metrics

SPIN can be applied to predict action-specific network response (contenxt-aware) when action representation $\phi(a)$ is given as input. It is also capable of predicting edge influence scores in context-free mode when $\phi(a)$ is treated as unknown. For comparison purposes, we evaluate SPIN against the following the state-of-the-art methods:

- Support Vector Machine (SVM) is used as a single target classifier used to predict the response network via decomposing it as a bag of nodes and edges, and predicting each element in the bag.
- Max-Margin Conditional Random Field (MMCRF) (Rousu et al., 2007; Su et al., 2010) is a multi-label classifier that utilizes the structure of output graph G. The model predicts the node labels of the network.
- Expectation-Maximization for the independent cascade model (ICM-EM) (Saito et al., 2008) is a context-free model that infers the influence probability of the network given a directed network and a set of action cascades. Here we use the implementation from Mathioudakis et al. (2011) of this algorithm, which is publicly available[3].
- Netrate (Gomez-Rodriguez et al., 2011) models the network influence as temporal processes occurs at difference rate. It infers the directed edges of the global network and estimates the transmission rate of each edge.

To quantitatively evaluate the performance of the tested methods in predicting node and edge labels, we adopt two popular metrics: *accuracy* and F_1 *score*, defined as

$$F_1 = \frac{2 \cdot P \cdot R}{P + R},$$

where P is precision and R is recall. We also define *Predicted Subgraph Coverage* (PSC) as

$$\text{PSC} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{v \in V_i} |G_v|,$$

where V_i is the set of focal points given action a_i, n is the number of nodes in the network, and m is the number of actions. PSC expresses the relative size of a correctly predicted subgraph G_v in terms of node predictions that cover the focal points v.

[3] https://dl.dropboxusercontent.com/u/21620176/public_html/spine/index.html

Dataset	Training Example	Feature Space	Network	
			$\|V\|$	$\|E\|$
DBLP S100	440	1190	100	204
DBLP M100	478	1127	100	151
DBLP M500	2119	3619	500	699
DBLP M700	2800	4369	699	952
DBLP M1k	3720	5281	1000	1368
DBLP M2k	6030	7183	2000	2687
DBLP L100	509	1274	100	152
DBLP L500	1869	3424	499	701
DBLP L700	2620	4300	699	960
DBLP L1k	3560	5405	1000	1368
DBLP L2k	3618	5454	1023	1402
memeS	4632	181	82	325
memeM	4804	179	182	521
memeL	4809	179	333	597

Table 1. Statistics of DBLP and Memetracker datasets.

Data	Accuracy			F_1 Score			Time (10^2s)		
	SDP	Neg	Act	SDP	Neg	Act	SDP	Neg	Act
S100	**79.9**	*77.6*	72.9	**57.2**	*56.2*	55.5	16.0	*1.5*	**0.2**
M100	**75.8**	*73.6*	68.5	51.6	*53.1*	**54.5**	15.2	*1.4*	**0.2**
L100	**75.1**	*72.0*	67.4	53.5	*56.9*	**57.2**	13.7	*1.6*	**0.3**
Geom.	**76.9**	*74.3*	69.6	52.0	*55.4*	**55.7**	15.0	*1.5*	**0.3**

Table 2. Comparison of different inference algorithms. *Geom.* is geometric mean of rows.

Our metrics are computed both in *global context* where we pool all the nodes and edges from the background network, as well as in *local context* where we only collect the nodes and edges within certain radius R of the focal points. The experimental results are from a five-fold cross validation.

4.2. Experimental results

We examine whether our context-sensitive structure predictor can boost the performance of predicting network responses. We compare SPIN with other methods in both context-sensitive and context-free problems. We show that SPIN can perform significantly better in terms of predicting action-specific network responses.

Comparison of inference algorithms. Table 2 shows the geometric mean of node accuracy, F_1 and running time over parameter space on three DBLP datasets, where "Neg" and "Act" represent the GREEDY inference defined on the negative-feed and the activation modes. SDP is also formulated on the negative-feed mode. In general, the inference algorithm based on negative-feed mode outperforms activation mode in terms of accuracy. The difference in F_1 is smaller in comparison. SDP based inference surpasses GREEDY inference in accuracy, however, by a small margin. In addition, GREEDY inference is almost 10 times faster even on small datasets, where running time is total time used for cross validation. For the following experiments, we opted for GREEDY inference in negative-feed mode as the inference engine of SPIN.

Dataset	Node Accuracy			Node F_1 Score			Edge Acc		*PSC*			Time (10^3s)		
	SVM	MMCRF	SPIN	SVM	MMCRF	SPIN	SVM	SPIN	SVM	MMCRF	SPIN	SVM	MMCRF	SPIN
memeS	**73.4**	68.0	*72.2*	39.0	*39.8*	**47.1**	**62.7**	45.6	23.4	*25.3*	**33.6**	6.6	**2.9**	*4.1*
memeM	**82.1**	79.0	*81.5*	29.1	*30.1*	**38.0**	61.1	**68.8**	18.6	*18.8*	**28.3**	13.7	**3.2**	*7.3*
memeL	**89.9**	88.3	*89.8*	26.7	*27.1*	**35.0**	45.5	**80.0**	17.7	*18.9*	**27.6**	19.9	**5.9**	*11.8*
M100	71.2	*73.6*	**76.7**	49.3	*50.8*	**54.3**	33.3	**61.7**	33.3	**35.6**	*34.6*	**0.1**	0.2	**0.1**
M500	89.0	*91.4*	**92.0**	**18.8**	13.5	*14.6*	28.2	**92.6**	*29.3*	26.4	**29.5**	9.0	*3.8*	**3.2**
M700	91.9	**94.1**	*92.1*	*13.8*	7.3	**14.2**	26.3	**93.0**	*29.4*	23.9	**34.4**	18.5	*8.3*	**4.4**
M1k	94.1	**95.8**	*94.2*	**10.9**	3.5	*9.3*	26.6	**94.7**	*33.7*	16.6	**35.2**	42.2	*14.7*	**10.4**
M2k	*96.8*	**97.6**	96.7	**6.2**	1.4	*3.4*	25.3	**97.6**	**34.6**	9.6	*14.7*	165.0	*88.4*	**54.1**
L100	69.4	*72.2*	**75.7**	51.1	*53.1*	**57.4**	31.6	**62.3**	30.9	*31.7*	**33.4**	**0.1**	*0.2*	0.3
L500	85.9	**89.1**	*86.8*	*21.7*	15.1	**24.7**	27.9	**87.9**	*14.2*	11.2	**19.7**	6.5	*3.2*	**2.1**
L700	89.7	**92.4**	*89.7*	*16.2*	9.4	**17.3**	26.5	**90.4**	*9.5*	6.7	**12.5**	16.0	*7.8*	**5.3**
L1k	*92.4*	**94.4**	91.5	*12.4*	6.4	**13.9**	26.4	**92.3**	*6.1*	4.4	**8.4**	40.3	*13.7*	**10.4**
L2k	*92.5*	**94.5**	91.9	*12.3*	5.4	**12.7**	26.5	**93.2**	*6.0*	2.9	**7.2**	41.9	*21.9*	**13.1**
Geom.	85.5	*86.4*	**86.6**	*19.8*	12.6	**20.3**	32.6	**79.7**	*18.9*	14.2	**21.7**	9.4	*4.6*	**4.3**

Table 3. Comparison of prediction performance on global context. The best in bold-face, the second best in italic.

Context-aware prediction. We apply SPIN with exponential scaling to predict context-sensitive network responses. Comparison of prediction performance against SVM and MMCRF is listed in Table 3. We show that SPIN can dramatically boost the performance of all measures except node accuracy: MMCRF wins in node accuracy, but SPIN is the second best and the difference is small. In terms of time consumption for training, SPIN is around three times faster than SVM and two times faster than MMCRF on the largest M2k dataset.

Context-free network influence prediction. Here we compare SPIN to methods developed for influence network prediction, namely Netrate and ICM-EM, on Memetracker data. To make the comparison fair to the competition, we convert the network to undirected network and replace action features by a constant value. For SPIN, we further represent each undirected edge by two directed edges. The measure of success is *Precision*@K, where we ask for top-K percent edge predictions from each model and compute the precision. Table 4 shows *Precision*@K as function of K, where the performance of SPIN surpasses ICM-EM and Netrate in all spectrum of K with a noticeable margin. ICM-EM has the least accurate predictions of the three, but achieves by far the the best running time. SPIN and Netrate solve more complex convex optimization problems, leading more accurate predictions at the cost of more CPU time needed for training, SPIN being the more efficient on the largest dataset, memeL.

The good performance of SPIN compared to Netrate is mostly explained by the fact that Netrate solves a much harder problem in which the underlying undirected network is assumed to be unknown, while SPIN is able to leverage the known network structure. In the experiment reported, the edge predictions from Netrate are filtered against the underlying complex network, in order to excessively penalize influence predictions along non-linked nodes.

Effect of loss scaling. Figure 2 depicts the effect of parameter λ of the exponential loss scaling to prediction performance on subgraphs of different radius. SVM (dashed line) is used as the baseline. When $0 < \lambda < 1$, the node prediction accuracy (top, left) and F_1 (top, right) decrease by the increasing subgraph radius, while $\lambda \geq 1$ leads to the opposite behavior allowing larger subgraph to be learned. Predicted subgraph coverage decreases by incresing λ. Edge prediction accuracy (bottom, right) increases monotonically in λ implying that predicting the longer influence paths is a hard problem for SVM. In Table 5 we examine the performance of diffusion scaling. The numbers reported are geometric means over the different Memetracker and DBLP datasets. We observe a decreased performance when increasing the parameter β, which corresponds to smoothing the distance matrix. This indicates

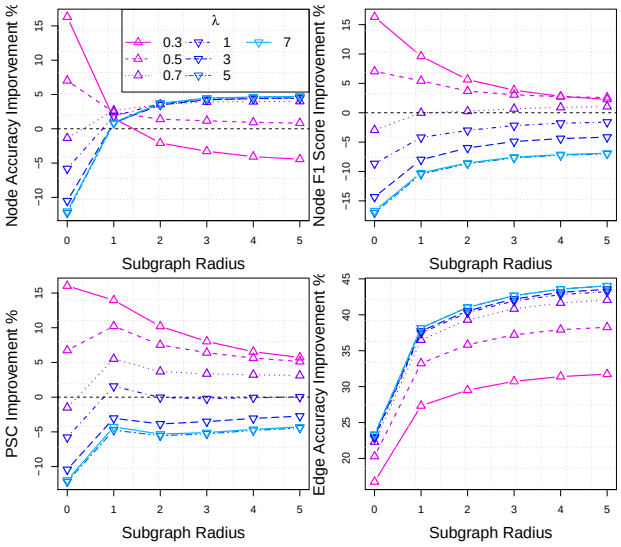


Figure 2. The improvement of prediction performance for different scaling factor λ with respect to SVM.

Dataset	Model	T (10^3s)	Precision @ K									
			10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
memeS	SPIN	*5.50*	**82.9**	**81.0**	**76.0**	**74.0**	**74.0**	**70.0**	**69.8**	**67.9**	**66.7**	**64.7**
	ICM-EM	**0.01**	60.3	63.5	65.1	62.0	62.0	61.5	62.2	60.4	60.7	*61.9*
	NETRATE	5.83	*76.2*	*73.8*	*70.4*	*68.7*	*68.7*	*66.8*	*64.9*	*63.4*	*62.9*	*61.9*
memeM	SPIN	*5.52*	**82.7**	**72.1**	**70.5**	**69.2**	**69.2**	**67.9**	**66.2**	**65.6**	**64.3**	**64.2**
	ICM-EM	**0.02**	56.3	55.3	56.8	57.4	57.4	56.3	57.5	57.8	58.3	*58.5*
	NETRATE	13.93	*61.2*	*64.6*	*62.9*	*62.5*	*62.5*	*62.4*	*61.2*	*60.1*	*58.7*	*58.5*
memeL	SPIN	*4.75*	**82.2**	**73.6**	**69.1**	**66.7**	**66.7**	**65.9**	**66.1**	**65.9**	**63.9**	**63.6**
	ICM-EM	**0.01**	52.1	55.7	54.2	56.5	56.5	56.7	57.4	58.0	*57.6*	*57.0*
	NETRATE	12.63	*56.5*	*57.8*	*60.0*	*59.3*	*59.3*	*59.4*	*58.9*	*58.4*	57.5	*57.0*

Table 4. Model performance in context-free influence network prediction.

Loss Scaling	Node Acc		Node F_1		Edge Acc		*PSC*		Time (10^3s)	
	Meme	DBLP	Meme	DBLP	Meme	DBLP	Meme	DBLP	Meme	DBLP
Dif $\beta = 0.1$	80.8	86.5	40.0	28.6	63.0	80.5	30.2	30.3	68.3	2.7
Dif $\beta = 0.5$	66.4	86.5	42.5	28.5	40.9	80.5	33.0	30.2	50.9	4.0
Dif $\beta = 0.8$	63.5	86.5	40.9	28.5	39.3	80.5	31.2	30.2	32.6	3.2
Exp $\lambda = 0.5$	80.9	83.9	39.7	28.7	63.1	77.7	29.7	24.3	71.0	10.8

Table 5. Comparison of diffusion scaling with exponential scaling.

that emphasizing connections between long-distance nodes makes prediction more difficult, a finding consistent with the results on exponential scaling. Setting $\beta = 0.1$ leads to comparable performance over exponential scaling with $\lambda = 0.5$, with slight improvement on the DBLP datasets.

5. Discussion

We have presented a novel approach, based on structured output learning, to the problem of modelling influence in networks. In contrast to previous state-of-the-art approaches, such as Netrate and ICM-EM, our proposal, named SPIN, is a context-sensitive model. SPIN does not try to force global influence parameters, but instead it incorporates the action space into the learning process and makes predictions tailored to the action under consideration. Our method can provide a useful tool in market research or other application scenarios when actions arise from a high-dimensional space, and one wants to make predictions for actions not seen before. Another benefit of our approach, compared to other state-of-the-art methods, is that our method does not make explicit assumptions regarding the underlying propagation model. Additionally, action responses are explicitly formulated as directed acyclic subgraphs, and the model is capable of predicting the complete subgraph structure. We proved that the inference problem of SPIN is $\mathbf{NP}$-hard, and we provided an approximation algorithm based on semidefinite programming (SDP). In addition, we developed a greedy heuristic algorithm for the inference problem that scales linearly in the size of the network, with time consumption in the same ballpark as Netrate. With extensive experiments we show that SPIN can dramatically boost the performance of action-based network-response prediction. SPIN can also be applied in context-free prediction where it captures the edge influence weight of the network.

References

Anagnostopoulos, Aris, Kumar, Ravi, and Mahdian, Mohammad. Influence and correlation in social networks. *KDD*, 2008.

Blei, D., Ng, A., and Jordan, M. Latent dirichlet allocation. In Dietterich, T., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

De Choudhury, Munmun, Mason, Winter A, Hofman, Jake M, and Watts, Duncan J. Inferring relevant social networks from interpersonal communication. *WWW*, pp. 301–310, 2010.

Du, Nan, Song, Le, Smola, Alex, and Yuan, Ming. Learning Networks of Heterogeneous Influence. *NIPS*, 2012.

Eagle, Nathan, Pentland, Alex Sandy, and Lazer, David. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36):15274–15278, 2009.

Goemans, Michel and Williamson, David. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *JACM*, 42 (6), 1995.

Gomez-Rodriguez, Manuel, Leskovec, Jure, and Krause, Andreas. Inferring Networks of Diffusion and Influence. *KDD*, 2010.

Gomez-Rodriguez, Manuel, Balduzzi, David, and Schölkopf, Bernhard. Uncovering the Temporal Dynamics of Diffusion Networks. *ICML*, 2011.

Goodman, Leo A. Snowball sampling. *The annals of mathematical statistics*, 32(1):148–170, 1961.

Goyal, Amit, Bonchi, Francesco, and Lakshmanan, Laks VS. Learning influence probabilities in social networks. *WSDM*, 2010.

Kempe, David, Kleinberg, Jon, and Tardos, Éva. Maximizing the spread of influence through a social network. In *KDD*, 2003.

Kondor, I.R. and Lafferty, J. D. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the ICML*, 2002.

Mathioudakis, Michael, Bonchi, Francesco, Castillo, Carlos, Gionis, Aristides, and Ukkonen, Antti. Sparsification of influence networks. *KDD*, 2011.

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. Efficient algorithms for max-margin structured classification. *Predicting Structured Data*, pp. 105–129, 2007.

Saito, Kazumi, Nakano, Ryohei, and Kimura, Masahiro. Prediction of information diffusion probabilities for independent cascade model. In *Knowledge-Based Intelligent Information and Engineering Systems (KES)*, 2008.

Su, Hongyu, Heinonen, Markus, and Rousu, Juho. Structured output prediction of anti-cancer drug activity. In *Proceedings of the 5th IAPR international conference on Pattern recognition in bioinformatics*, PRIB'10, 2010.

Watts, Duncan J and Dodds, Peter Sheridan. Influentials, networks, and public opinion formation. *Journal of consumer research*, 34(4):441–458, 2007.

Publication II

Hongyu Su, Markus Heinonen, Juho Rousu. Multilabel Classification of Drug-like Molecules via Max-margin Conditional Random Fields. In *Proceedings of the* 5^{th} *International Conference on Pattern Recognition in Bioinformatics (PRIB 2010)*, Nijmegen, The Netherlands, 2010. Springer LNBI volume 6282:265-273, September 2010.

Structured Output Prediction of Anti-cancer Drug Activity

Hongyu Su, Markus Heinonen, and Juho Rousu

Department of Computer Science
P.O. Box 68, 00014 University of Helsinki, Finland
{hongyu.su,markus.heinonen,juho.rousu}@cs.helsinki.fi
http://www.cs.helsinki.fi/group/sysfys

Abstract. We present a structured output prediction approach for classifying potential anti-cancer drugs. Our QSAR model takes as input a description of a molecule and predicts the activity against a set of cancer cell lines in one shot. Statistical dependencies between the cell lines are encoded by a Markov network that has cell lines as nodes and edges represent similarity according to an auxiliary dataset. Molecules are represented via kernels based on molecular graphs. Margin-based learning is applied to separate correct multilabels from incorrect ones. The performance of the multilabel classification method is shown in our experiments with NCI-Cancer data containing the cancer inhibition potential of drug-like molecules against 59 cancer cell lines. In the experiments, our method outperforms the state-of-the-art SVM method.

1 Introduction

Machine learning has become increasingly important in drug discovery where viable molecular structures are searched or designed for therapeutic efficacy. In particular, Quantitative Structure-Activity Relationship (QSAR) models, relating the molecular structures to bioactivity (therapeutical effect, side-effects, toxicity, etc.) are routinely built using state-of-the-art machine learning methods. In particular, the costly pre-clinical *in vitro* and *in vivo* testing of drug candidates can be focused to the most promising molecules, if accurate *in silico* models are available [16].

Molecular classification—the task of predicing the presence or absense of the bioactivity of interest—has been tackled with a variety of methods, including inductive logic programming [9] and artificial neural networks [1]. During the last decade kernel methods [11,16,4] have emerged as an computationally effective way to handle the non-linear properties of chemicals. In numerous studies, SVM-based methods have obtained promising results [3,16,20]. However, classification methods focusing on a single target variable are probably not optimally suited to drug screening applications where large number of target cell lines are to be handled.

In this paper we propose, to our knowledge, the first multilabel learning approach for molecular classification. Our method belongs to the structured output

T.M.H. Dijkstra et al. (Eds.): PRIB 2010, LNBI 6282, pp. 38–49, 2010.

prediction family [15,17,12,13], where graphical models and kernels have been successfully married in recent years. In our approach, the drug targets (cancer cell lines) are organized in a Markov network, drug molecules are represented by kernels and discriminative max-margin training is used to learn the parameters. Alternatively, our method can be interpreted as a form of multitask learning [5] where the Markov network couples the tasks (cell lines) and joint features are learned for pairs of similar tasks.

2 Methods

2.1 Structured Output Learning with MMCRF

The model used is this paper is an instantiation of the structured output prediction framework MMCRF [13] for associative Markov networks and can also be seen as a sibling method to HM^3[12], which is designed for hierarchies. We give a brief outline here, the interested reader may check the details from the above references.

The MMCRF learning algorithm takes as input a matrix $K = (k(x_i, x_j))_{i,j=1}^m$ of kernel values $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ between the training patterns, where $\phi(x)$ denotes a feature description of an input pattern (in our case a potential drug molecule), and a label matrix $Y = (\mathbf{y}_i)_{i=1}^m$ containing the multilabels $\mathbf{y}_i = (y_1, \ldots, y_k)$ of the training patterns. The components $y_j \in \{-1, +1\}$ of the multilabel are called microlabels and in our case correspond to different cancer cell lines. In addition, the algorithm assumes an associative network $G = (V, E)$ to be given, where node $j \in V$ corresponds to the j'th component of the multilabel and the edges $e = (j, j') \in E$ correspond to a microlabel dependency structure.

The model learned by MMCRF takes the form of a conditional random field with exponential edge-potentials,

$$P(\mathbf{y}|x) \propto \prod_{e \in E} \exp\left(\mathbf{w}_e^T \varphi_e(x, \mathbf{y}_e)\right) = \exp\left(\mathbf{w}^T \varphi(x, \mathbf{y})\right),$$

where $\mathbf{y}_e = (y_j, y_{j'})$ denotes the pair of microlabels of the edge $e = (j, j')$. A joint feature map $\varphi_e(x, \mathbf{y}) = \phi(x) \otimes \psi_e(\mathbf{y}_e)$ for an edge is composed via tensor product of input $\phi(x)$ and output feature map $\psi(\mathbf{y})$, thus including all pairs of input and output features. The output feature map is composed of indicator functions $\psi_e^u(\mathbf{y}) = [\![\mathbf{y}_e = u]\!]$ where u ranges over the four possible labelings of an edge given binary node labels. The corresponding weights are denoted by $\mathbf{w}_e$. The benefit of the tensor product representation is that context (edge-labeling) sensitive weights can be learned for input features and no prior alignment of input and output features needs to be assumed.

The parameters are learned by maximizing the minimum loss-scaled margin between the correct training examples $(x_i, \mathbf{y}_i)$ and incorrect pseudo-examples

$(x_i, \mathbf{y}), \mathbf{y} \neq \mathbf{y}_i$, while controlling the norm of the weight vector. The primal soft-margin optimization problem takes the form

$$\begin{aligned} \underset{\mathbf{w},\xi\geq 0}{\text{minimize}}\ & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t. } & \mathbf{w}^T\varphi(x_i,\mathbf{y}_i) - \mathbf{w}^T\varphi(x_i,\mathbf{y}) \geq \ell(\mathbf{y}_i,\mathbf{y}) - \xi_i, \\ & \text{for all } i \text{ and } \mathbf{y}, \end{aligned} \tag{1}$$

where ξ_i denote the slacks allotted to each example. The effect of loss-scaling is to push high-loss pseudo-examples further away from the correct example than the low-loss pseudo-examples, which, intuitively, decreases the risk of incurring high-loss. We use *Hamming loss*

$$\ell_\Delta(\mathbf{y},\mathbf{u}) = \sum_j [\![y_j \neq u_j]\!]$$

that is gradually increasing in the number of incorrect microlabels so that we can make a difference between 'nearly correct' and 'clearly incorrect' multilabel predictions.

The MMCRF algorithm [13] optimizes the model (1) in the so called marginal dual form, that has several benefits: the use of kernels to represent high-dimensional inputs, and polynomial-size of the optimization problem with respect to the size of the output structure. Efficient optimization is achieved via the conditional gradient algorithm [2] with feasible ascent directions found by loopy belief propagation over the Markov network G.

2.2 Kernels for Drug-Like Molecules

A major challenge for any statistical learning model is to define a measure of similarity. In chemical community, widely researched quantitative structure-activity relationship (QSAR) theory asserts that compounds having similar physicochemical and geometric properties should have related bioactivity [7]. Various descriptors have been used to represent molecules with fixed-length feature vectors, such as atom counts, topological and shape indices, quantum-chemical and geometric properties [19]. Kernels computed from the structured representation of molecules extend the scope of the traditional approaches by allowing complex derived features to be used (walks, subgraphs, properties) while avoiding excessive computational cost [11].

In this paper, we experiment with a set of graph kernels designed for classification of drug-like molecules, including walk kernel [6], weighted decomposition kernel [10] and Tanimoto kernel [11]. All of them rely on representing the molecule as a labeled graph with atoms as nodes and bonds between the atoms as the edges.

Walk kernel. [8,6] computes the sum of matching walks (a sequence of labeled nodes so that there exists an edge for each pair of adjacent nodes) in a pair

of graphs. The contribution of each matching walk is downscaled exponentially according to its length. We consider finite-length walk kernel where only walks of length p are counted. The finite walk kernel can be efficiently computed using dynamic programming.

Weighted decomposition kernel. [4] is an extension of the substructure kernel by weighting identical parts in a pair of graphs based on contextual information [4]. The kernel looks at matching subgraphs (*contextor*) in the neighborhood of *selector* atoms.

Tanimoto kernel. [11] is a kernel computed from two molecule fingerprints by checking the fraction of features that occur in both fingerprints of all features. *Hash fingerprints* enumerates all linear fragments of a given length, while *substructure keys* correspond to molecular substructures in a predefined set designed by domain experts. Based on good performance in preliminary studies, in this paper we concentrate on hash fingerprints.

2.3 Markov Network Generation for Cancer Cell Lines

In order to use MMCRF to classify drug molecules we need to build a Markov network for the cell lines used as the output, with nodes corresponding to cell lines and edges to potential statistical dependencies. To build the network we used auxiliary data (e.g. mRNA and protein expression, mutational status, chromosomal aberrations, DNA copy number variations, etc) available on the cancer cell lines from NCI database[1]. The basic approach is to construct from this data a correlation matrix between the pairs of cell lines and extract the Markov network from the matrix by favoring high-valued pairs. The following methods of network extraction were considered:

- Maximum weight spanning tree. Take the minimum number of edges that make a connected network whilst maximizing the edge weights.
- Correlation thresholding. Take all edges that exceed fixed threshold. This approach typically generates a general non-tree graph.

3 Experiments

3.1 NCI-Cancer Dataset

In this paper we use the NCI-Cancer dataset obtained through PubChem Bioassay[2] [18] data repository. The dataset initiated by National Cancer Institute and National Institutes of Health (NCI/NIH) contains bioactivity information of large number of molecules against several human cancer cell lines in 9 different tissue types, including leukemia, melanoma and cancers of the lung, colon, brain, ovary, breast, prostate, and kidney. For each molecule tested against a certain cell line, the dataset provide a bioactivity outcome that we use as the classes (active, inactive).

[1] `http://discover.nci.nih.gov/cellminer/home.do`
[2] `http://pubchem.ncbi.nlm.nih.gov`

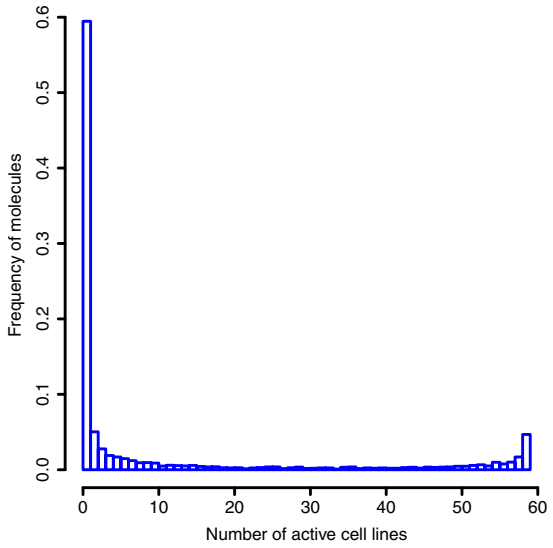


Fig. 1. Skewness of the multilabel distribution

3.2 Data Preprocessing

Currently, there are 43884 molecules in the PubChem Bioassay database together with anti-cancer activities in 73 cell lines. 59 cell lines have screen experimental results for most molecules and 4554 molecules have no missing data in these cell lines, therefore these cell lines and molecules are selected and employed in our experiments.

However, molecular activity data are highly biased over the cell lines. Figure 1 shows the molecular activity distribution over all 59 cell lines. Most of the molecules are inactive in all cell lines, while a relatively large proportion of molecules are active against almost all cell lines, which can be taken as toxics. These molecules are less likely to be potential drug candidates than the ones in the middle part of the histogram.

Figure 2 shows a heatmap of normalized Tanimoto kernel, where molecules have been sorted by the number of cell lines they are active in. The heatmap shows that the molecules in the two extremes of the multilabel distribution form groups of high similarity whereas the molecules in the middle are much more dissimilar both to each other and to the extreme groups. The result seems to indicate that the majority of molecules in the dataset are either very specific or very general in the targets they are active against. Other kernels mentioned in section 2.2 produce a similar heatmap indicating that the phenomenon is not kernel-specific.

Because of the above-mentioned skewness, we prepared different versions of the dataset:

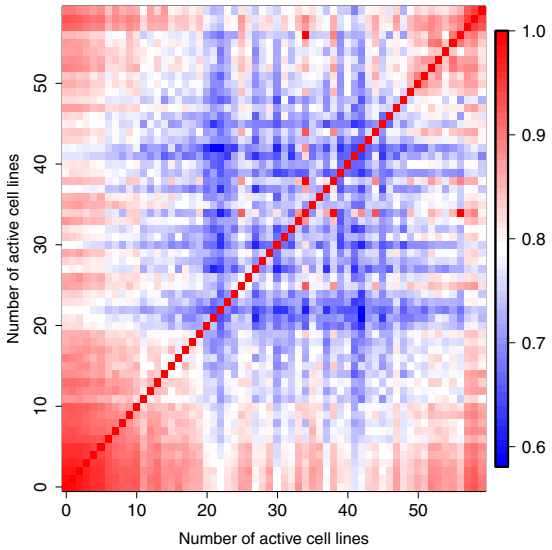


Fig. 2. Heatmap of the kernel space for the molecules sorted by the multilabel distribution

Full. This dataset contains all 4554 molecules in the NCI-Cancer dataset that have their activity class (active vs. incative) recorded against all 59 cancer cell lines.

No-Zero-Active. From this dataset, we removed all molecules that are not active towards any of the cell lines (corresponding to the leftmost peak in Figure 1). The remaining 2305 molecules are all active against at least one cell line.

Middle-Active. Here, we followed the preprocessing suggested in [14], and selected molecules that are active in more than 10 cell lines and inactive in more than 10 cell lines. As a result, 544 molecules remained and were employed in our experiments.

3.3 Experiment Setup

We conducted experiments to compare the effect of various kernels, as well as the performances of support vector machine (SVM) and MMCRF. We used the SVM implementation of the LibSVM software package written in C++[3]. We tested SVM with different margin C parameters, relative hard margin ($C = 100$) emerging as the value used in subsequent experiments. The same value was used for MMCRF classifier as well.

Because of the skewness of the multilabel distribution (c.f. 1) we used the following *stratified 5-fold cross-validation* scheme in all experiments reported:

[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

we group the molecules in equivalence classes based on the number of cell lines they are active against. Then each group is randomly split among the five folds. This ensures that also the smaller groups have representation in all folds.

3.4 Kernel Setup

For the three kernel methods, walk kernel (WK) was constructed using parameters $\lambda = 0.1$ and $p = 6$ as recommended in [6]. The Weighted decomposition kernel (WDK) used context radius 3 as in [4], and a single attribute (atom type) was sufficient to give the best performance. We also used hash fragments as molecular fingerprints generated by OpenBabel[4] (using default value $n = 6$ for linear structure length), which is a chemical toolbox available in public domain. All kernels were normalized.

4 Results

4.1 Effect of Markov Network Generation Methods

We report overall prediction accuracies on the Middle-Active dataset from various Markov networks shown in Figure 3. X-axis corresponds to different microarray experiments. The accuracies from different Markov networks differ slightly. The best accuracy was achieved by using maximum weighted spanning tree approach on RNA radiation arrays dataset, shown in Figure 4, which describes profiles of radiation response in cell lines. This meets our expectations since cancer cells mostly mutated from normal cells and normal cells with radiation treatments can possibly explain the mutations.

4.2 Effect of molecule kernels

In Table 1, we report overall accuracies and microlabel F1 scores using SVM with different kernels on the Middle-Active dataset. The results were from a five-fold cross validation procedure. Here, the three kernel methods achieve almost the same accuracies in SVM classifier, while Tanimoto kernel is slightly better than others in microlabel F1 score. Thus we deemed Tanimoto kernel to be the best kernel in this experiment and chose it for the subsequent experiments.

4.3 Effect of Dataset Versions

Figure 5 gives overall accuracy and microlabel F1 score of MMCRF versus SVM for each cell line on the three versions of the data. Points above the diagonal line correspond to improvements in accuracies or F1 scores by MMCRF classifier. MMCRF improves the F1 score over SVM on each version of the data in statistically significant manner, as judged by the two-tailed sign test. Accuracy is improved in two versions, No-Zero-Actives and the Middle-Active molecules,

[4] `http://openbabel.org`

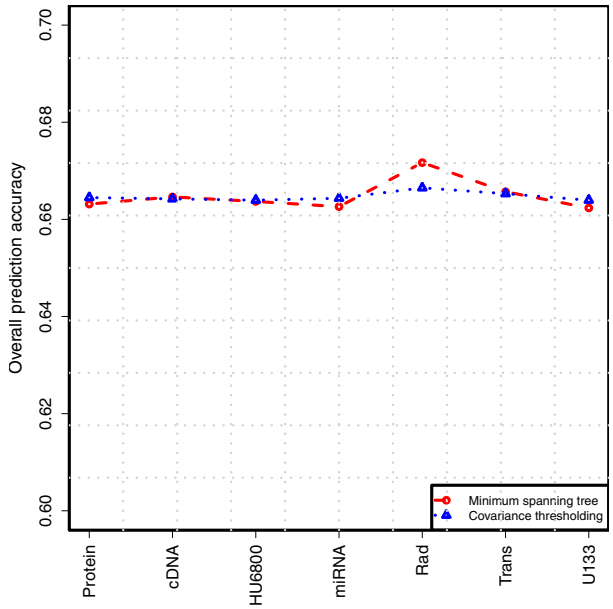


Fig. 3. Effects of Markov network construction methods and type of auxiliary data (from left to right: reverse-phase lysate arrays, cDNA arrays, Affymetric HU6800 arrays, miRNA arrays, RNA radiation arrays, transporter arrays, and Affymetrix U133 arrays)

Table 1. Accuracies and microlabel F1 scores of MMCRF and SVM with different kernels

Classifier	Kernel	Accuracy	F1 score
SVM	WK	64.6%	49.0%
	WDK	63.9%	51.6%
	Tanimoto	64.1%	52.7%
MMCRF	Tanimoto	67.6%	56.2%

again in statistically significant manner. Among the Middle-Active dataset, the difference in accuracy (bottom, left of Figure 5) is sometimes drastic, around 10 percentage units in favor of MMCRF for a significant fraction of the cell lines.

4.4 Agreement of MMCRF and SVM Predictions

For a closer look at the predictions of MMCRF and SVM, Table 2 depicts the agreement of the two models among positive and negative classes. Both models were trained on the Full dataset. Overall, the two models agree on the label most of the time (close to 90% of positive predictions and close to 95% of the negative predictions). MMCRF is markedly more accurate than SVM on the

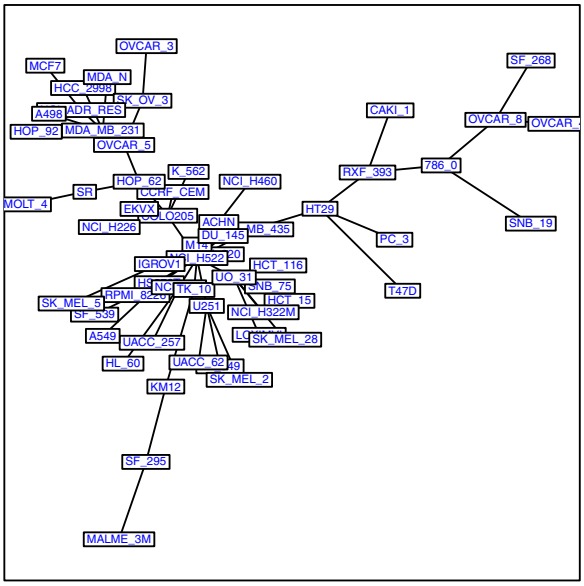


Fig. 4. Markov network constructed from maximum weighted spanning tree method on RNA radiation array data. The labels correspond to different cancer cell lines.

Table 2. Agreement of MMCRF and SVM on the positive (left) and negative (right) classes

	Positive class		**Negative class**	
	SVM Correct	SVM Incorrect	SVM Correct	SVM Incorrect
MMCRF Correct	$48.6 \pm 4.1\%$	$7.1 \pm 2.6\%$	$88.0 \pm 4.9\%$	$2.2 \pm 1.2\%$
MMCRF Incorrect	$3.4 \pm 1.3\%$	$40.9 \pm 3.4\%$	$3.8 \pm 1.7\%$	$6.1 \pm 3.0\%$

positive class while SVM is slightly more accurate among the negative class. Qualitatively similar results are obtained when the zero-active molecules are removed from the data (data not shown).

4.5 Computation Time

Besides predictive accuracy, training time of classifiers is important when a large number of drug targets need to be processed. The potential benefit of multilabel classification is the fact that only single model needs to be trained instead of a bag of binary classifiers.

We compared the running time needed to construct MMCRF classifier (implemented in native MATLAB) against libSVM classifier (C++). We conducted the experiment on a 2.0GHz computer with 8GB memory. Figure 6 shows that MMCRF scales better when training set increases.

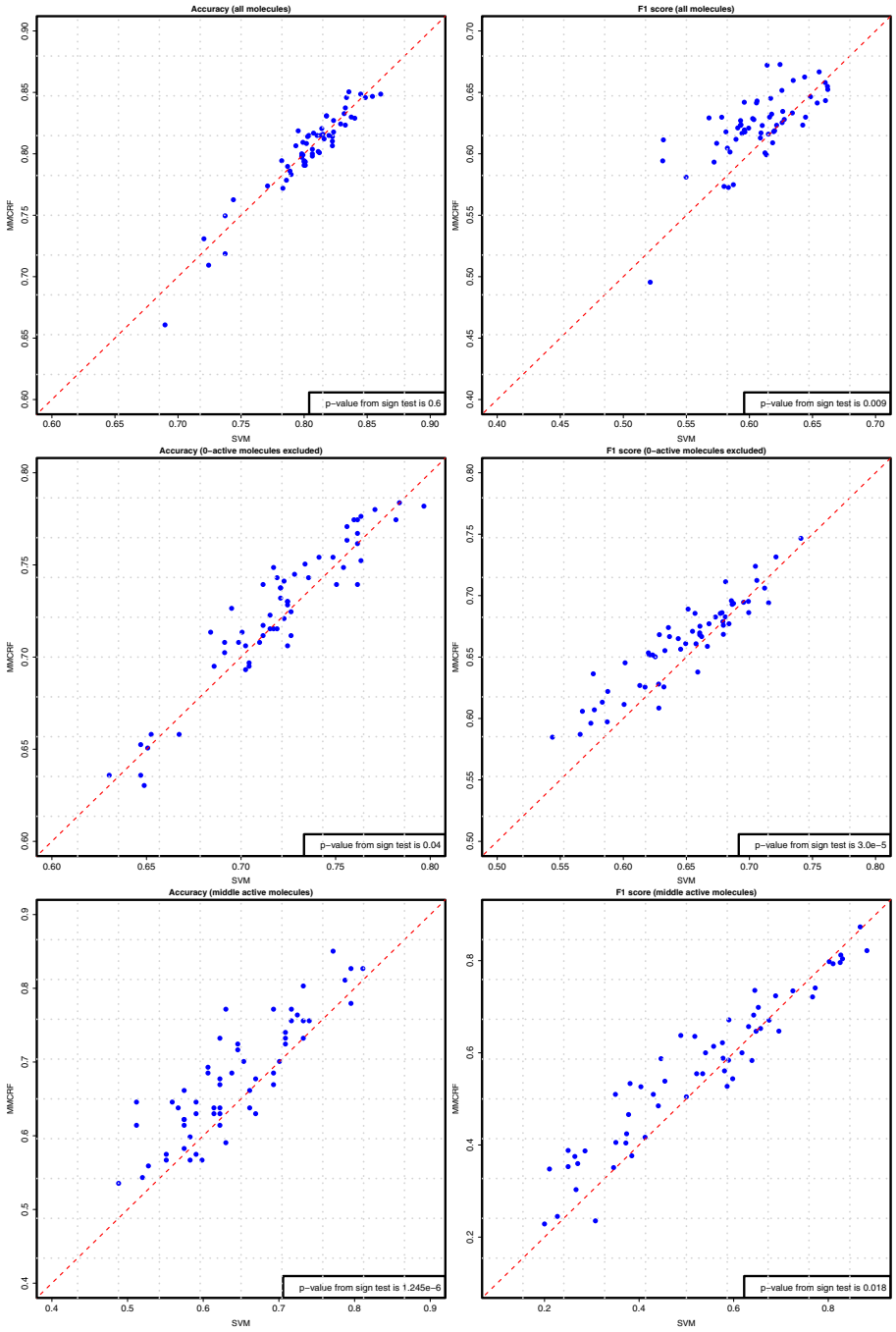


Fig. 5. Accuracy (left) and F1 score (right) of MMCRF vs. SVM on Full data (top), No-Zero-Active (middle) and Middle-Active molecules (bottom)

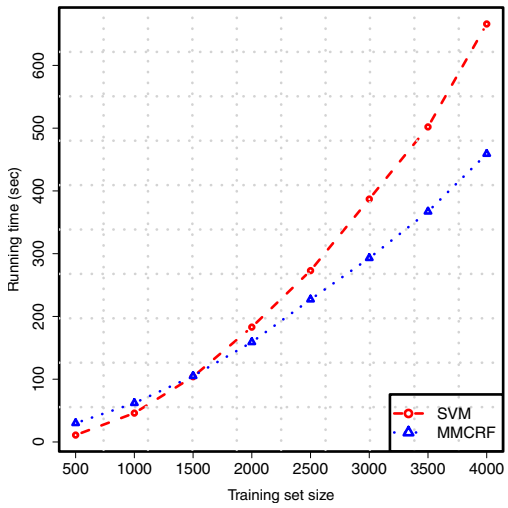


Fig. 6. Training time for SVM and MMCRF classifiers on training sets of different sizes

5 Conclusions

We presented a multilabel classification approach to drug activity classification using the Max-Margin Conditional Random Field algorithm. In the experiments against a large set of cancer lines the method significantly outperformed SVM in training time and accuracy. In particular, drastic improvements could be seen in the setup where molecules with extreme activity (active against no or a very small fraction, or a very large fraction of the cell lines) were excluded from the data. The remaining middle ground of selectively active molecules is in our view more important from drug screening applications point of view, than the two extremes.

The MMCRF software and preprocessed versions of the data are available from `http://cs.helsinki.fi/group/sysfys/software`.

Acknowledgements

This work was financially supported by Academy of Finland grant 118653 (ALGODAN) and in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, ICT-2007-216886. This publication only reflects the authors' views.

References

1. Bernazzani, L., Duce, C., Micheli, A., Mollica, V., Sperduti, A., Starita, A., Tine, M.: Predicting physical-chemical properties of compounds from molecular structures by recursive neural networks. J. Chem. Inf. Model. 46, 2030–2042 (2006)
2. Bertsekas, D.: Nonlinear Programming. Athena Scientific (1999)

3. Byvatov, E., Fechner, U., Sadowski, J., Schneider, G.: Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. J. Chem. Inf. Comput. Sci. 43, 1882–1889 (2003)
4. Ceroni, A., Costa, F., Frasconi, P.: Classification of small molecules by two- and three-dimensional decomposition kernels. Bioinformatics 23, 2038–2045 (2007)
5. Evgeniou, T., Pontil, M.: Regularized multi–task learning. In: KDD'04, pp. 109–117. ACM Press, New York (2004)
6. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explor. Newsl. 5(1), 49–58 (2003)
7. Karelson, M.: Molecular Descriptors in QSAR/QSPR. Wiley-Interscience, Hoboken (2000)
8. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: Proceedings of the 20th International Conference on Machine Learning (ICML), Washington, DC, United States (2003)
9. King, R., Muggleton, S., Srinivasan, A., Sternberg, M.: Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. PNAS 93, 438–442 (1996)
10. Menchetti, S., Costa, F., Frasconi, P.: Weighted decomposition kernels. In: International Conference on Machine Learning, pp. 585–592. ACM Press, New York (2005)
11. Ralaivola, L., Swamidass, S., Saigo, H., Baldi, P.: Graph kernels for chemical informatics. Neural Networks 18, 1093–1110 (2005)
12. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Kernel-Based Learning of Hierarchical Multilabel Classification Models. JMLR 7, 1601–1626 (2006)
13. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Efficient algorithms for max-margin structured classification. Predicting Structured Data, 105–129 (2007)
14. Shivakumar, P., Krauthammer, M.: Structural similarity assessment for drug sensitivity prediction in cancer. Bioinformatics 10, S17 (2009)
15. Taskar, B., Guestrin, C., Koller, D.: Max-margin markov networks. In: Neural Information Processing Systems 2003 (2003)
16. Trotter, M., Buxton, M., Holden, S.: Drug design by machine learning: support vector machines for pharmaceutical data analysis. Comp. and Chem. 26, 1–20 (2001)
17. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: ICML'04, pp. 823–830 (2004)
18. Wang, Y., Bolton, E., Dracheva, S., Karapetyan, K., Shoemaker, B., Suzek, T., Wang, J., Xiao, J., Zhang, J., Bryant, S.: An overview of the pubchem bioassay resource. Nucleic Acids Research 38, D255–D266 (2009)
19. Xue, Y., Li, Z., Yap, C., et al.: Effect of molecular descriptor feature selection in support vector machine classification of pharmacokinetic and toxicological properties of chemical agents. J. Chem. Inf. Comput. Sci. 44, 1630–1638 (2004)
20. Zernov, V., Balakin, K., Ivaschenko, A., Savchuk, N., Pletnev, I.: Drug discovery using support vector machines. The case studies of drug-likeness, agrochemical-likeness, and enzyme inhibition predictions. J. Chem. Inf. Comput. Sci. 43, 2048–2056 (2003)

Publication III

Hongyu Su, Juho Rousu. Multi-task Drug Bioactivity Classification with Graph Labeling Ensembles. In *Proceedings of the* 6^{th} *International Conference on Pattern Recognition in Bioinformatics (PRIB 2011)*, Delft, The Netherlands, 2011. Springer LNBI volume 7035:157-167, November 2011.

Multi-task Drug Bioactivity Classification with Graph Labeling Ensembles

Hongyu Su and Juho Rousu

Department of Computer Science,
P.O. Box 68, 00014 University of Helsinki, Finland
{hongyu.su,juho.rousu}@cs.helsinki.fi

Abstract. We present a new method for drug bioactivity classification based on learning an ensemble of multi-task classifiers. As the base classifiers of the ensemble we use Max-Margin Conditional Random Field (MMCRF) models, which have previously obtained the state-of-the-art accuracy in this problem. MMCRF relies on a graph structure coupling the set of tasks together, and thus turns the multi-task learning problem into a graph labeling problem. In our ensemble method the graphs of the base classifiers are random, constructed by random pairing or random spanning tree extraction over the set of tasks.

We compare the ensemble approaches on datasets containing the cancer inhibition potential of drug-like molecules against 60 cancer cell lines. In our experiments we find that ensembles based on random graphs surpass the accuracy of single SVM as well as a single MMCRF model relying on a graph built from auxiliary data.

Keywords: drug bioactivity prediction; multi-task learning; ensemble methods; kernel methods.

1 Introduction

Molecular classification, the task of predicting the presence or absence of the bioactivity of interest, has been benefited from variety of methods in statistics and machine learning [7]. In particular, kernel methods [9,16,2,7] have emerged as an effective way to handle the non-linear properties of chemicals. However, classification methods focusing on a single target variable are probably not optimally suited to drug screening applications where large number of target cell lines are to be handled.

In [15] a multi-task (or multilabel) learning approach was proposed to classify molecules according to their activity against a set of cancer cell lines. It was shown that the multilabel learning setup improves predictive performance over a set of support vector machine based single target classifiers. The multilabel classifier applied, Max-Margin Conditional Random Field (MMCRF) [11] relies on a graph structure coupling the outputs together. In [15] the graph was extracted

M. Loog et al. (Eds.): PRIB 2011, LNBI 7036, pp. 157–167, 2011.

from auxiliary data, concerning sets of experiments conducted on the cancer cell lines, by simple techniques such as correlation thresholding and maximum weight spanning tree extraction.

In this paper, we develop ensemble learning methods for the multi-task learning setup. In our method, MMCRF models are used as the ensemble components. Unlike other ensemble learners for multi-task setups, our method does not require bootstrapping of the training data or changing instance weights to induce diversity among the ensemble components. In our case, the diversity is provided by the randomization of the output graphs, which combined with discriminative training of the base MMCRF classifiers, realizes the benefits typically seen from ensemble approaches. The random graph approach is compared against single classifiers and ensembles on graphs built from auxiliary data with different graph extraction methods, including inverse covariance learning [5] that is theoretically superior to correlation thresholding for extracting statistical dependencies.

Ensembles of multi-task or multilabel classifiers have been proposed in a few papers prior to ours, but with important differences both in the methods and the applications. In general, the previous approaches can be divided into two groups based on the source of diversity among the base classifiers of the ensemble. The first group of methods, boosting type, relies on changing the weights of the training instances so that difficult-to-classify instances gradually receive more and more weight. The Boostexter method [12] by the inventors of boosting has a multilabel learning variant. Later, Esuli et al. [4] developed a hierarchical multilabel variant of AdaBoost. Neither method explicitly considers label dependencies but the multilabel is considered essentially a flat vector. The second group of methods, bagging, is based on bootstrap sampling the training set several times and building the base classifiers from the bootstrap samples. Averaging over the ensemble gives the final predictions. Schietgat et al [13] concentrate a bagging in multilabel gene function prediction. They build ensembles of predictive clustering trees (PCT) by bagging, that is, bootstrap sampling of the data several times to arrive at a set of different models. In their approach, there is also no structure defined for the tasks, but the multilabel is essentially treated as a flat vector. Finally, Yan et al. [18] select different random subsets of input features and examples to induce the base classifiers.

The remainder of the article is structured as follows. In section 2 we present the base classifier MMCRF and the multi-task ensemble learning approach. In section 3 we validate the methods empirically, in particular we show that the ensemble approach exceeds the accuracy of MMCRF, which to our knowledge currently has the state-of-the-art predictive performance. In section 4 we aim to provide intuition of the hows and whys of the behaviour of the new method by relating the new ensemble approach to other multi-task and multilabel ensemble approaches. In section 5 finish with concluding remarks.

2 Ensemble Learning with Max-Margin Conditional Random Field Models on Random Graphs

Ensemble learners [3,8], such as boosting [12] and bagging [1] are based on the notion that a set of *weak leaners*, those that have accuracy higher than coin tossing, may produce a strong learner with high accuracy when appropriately combined. It has been found that the diversity among the base models is the key property. The diversity may arise from re-weighting of examples, bootstrap resampling of examples, from the different inductive biases of the base learners, or in multiclass setup, or by generating a set of derived binary classification tasks (one-vs-the rest, one vs. one, and error-correcting output codes [3]).

In this section we present our ensemble learning approach where the diversity among the base learners comes from a different source, namely randomized graph structures that are used to couple the tasks. We use a majority voting approach over the predictions of the base classifiers, namely labelings of the randomized graphs. Two basic types of graphs are used, random spanning trees and random pairings of targets (Figure 1). As the base learner, we use the MMCRF algorithm [11].

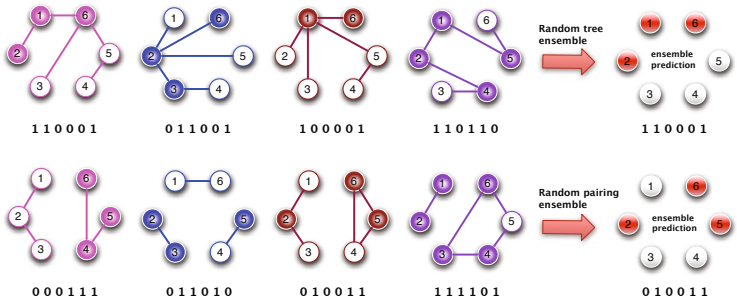


Fig. 1. Ensemble prediction from a set of random spanning trees (top) and a set of random pairings of tasks (bottom). The varying graph structures provide the required diversity among the ensemble components. Majority vote decides the final predicted label for each task.

The method for generating the ensemble is depicted in Algorithm 1. The algorithm receives a training sample of molecules x_i, computes the input kernel K and embarks on the ensemble learning phase. For each base model, a random graph G_t of the type specified by the user is drawn to couple the outputs $\mathbf{y}_i$ which are the inhibition potentials of molecule x_i against 60 cancer cell lines. The input kernel, label data and the graph are given as input to the MMCRF (see Section 2.1) that learns the graph labeling. After the ensemble has been generated, the ensemble prediction is extracted in post-processing: we extract the majority vote over the graph labelings from the sign of the mean of the base classifier predictions.

Algorithm 1. Ensemble learning algorithm with random graph multi-task classifiers

Input: Training sample $S = \{(x_i, \mathbf{y}_i)\}_{i=1}^m$, ensemble size T, type of the graph generated $graphType$, n the number of nodes in the graph, type of input kernel applied $kernelType$

Output: Multi-task classification ensemble $\left(f^{(1)}, \ldots, f^{(T)}\right)$

1: $K = buildKernel(\{x_i\}_{i=1}^m, kernelType)$
2: $t = 0$
3: **while** $t < T$ **do**
4: $\quad t = t + 1$
5: $\quad G_t = randomGraph(n, graphType)$
6: $\quad f^{(t)} = learnMMCRF(K, (\mathbf{y}_i)_{i=1}^m, G_t)$
7: **end while**
8: **return** $f = \left(f^{(1)}, \ldots, f^{(T)}\right)$

2.1 Learning Graph Labeling with MMCRF

The MMCRF method used as the base learner in the multi-task ensembles is an instantiation of the structured output prediction framework MMCRF [11] for associative Markov networks and can also be seen as a sibling method to HM^3[10], which is designed for hierarchies. We give a brief outline here, the interested reader may check the details from the above references.

The MMCRF learning algorithm takes as input a matrix $K = (k(x_i, x_j))_{i,j=1}^m$ of kernel values $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ between the training patterns, where $\phi(x)$ denotes a feature description of an input pattern (in our case a potential drug molecule), and a label matrix $Y = (\mathbf{y}_i)_{i=1}^m$ containing the multilabels $\mathbf{y}_i = (y_1, \ldots, y_k)$ of the training patterns. The components $y_j \in \{-1, +1\}$ of the multilabel are called microlabels, which in multi-task learning setup, correspond to labels of different tasks. In addition, the algorithm assumes an associative network $G = (V, E)$ to be given, where node $j \in V$ corresponds to the j'th component of the multilabel and the edges $e = (j, j') \in E$ correspond to a microlabel dependency structure.

The model learned by MMCRF takes the form of a conditional random field with exponential edge-potentials,

$$P(\mathbf{y}|x) \propto \prod_{e \in E} \exp\left(\mathbf{w}_e^T \varphi_e(x, \mathbf{y}_e)\right) = \exp\left(\mathbf{w}^T \varphi(x, \mathbf{y})\right),$$

where $\mathbf{y}_e = (y_j, y_{j'})$ denotes the pair of microlabels of the edge $e = (j, j')$. A joint feature map $\varphi(x, \mathbf{y}) = \phi(x) \otimes \psi(\mathbf{y})$ is composed via tensor product of input $\phi(x)$ and output feature map $\psi(\mathbf{y})$, thus including all pairs of input and output features. The output feature map is composed of indicator functions $\psi_e^u(\mathbf{y}) = [\![\mathbf{y}_e = u]\!]$ where u ranges over the four possible labelings of an edge given binary node labels. The corresponding weights are denoted by $\mathbf{w} = (\mathbf{w}_e)_e$. The benefit of the tensor product representation is that context (edge-labeling)

sensitive weights can be learned for input features and no prior alignment of input and output features needs to be assumed.

The parameters are learned by maximizing the minimum loss-scaled margin between the correct training examples $(x_i, \mathbf{y}_i)$ and incorrect pseudo-examples $(x_i, \mathbf{y}), \mathbf{y} \neq \mathbf{y}_i$, while controlling the norm of the weight vector. The dual soft-margin optimization problem takes the form

$$\begin{aligned} \min_{\alpha \geq 0} \ & \sum_{i,\mathbf{y}} \alpha(i,\mathbf{y}) \ell(\mathbf{y}_i, \mathbf{y}) - \frac{1}{2} \sum_{i,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha(i,y) K(x_i, \mathbf{y}; x_j, \mathbf{y}') \alpha(i,\mathbf{y}') \\ \text{s.t.} \ & \sum_{\mathbf{y}} \alpha(i,\mathbf{y}) \leq C, \forall i, \end{aligned} \tag{1}$$

where $K(x_i, \mathbf{y}; x_j, \mathbf{y}') = \Delta\varphi(i,\mathbf{y})^T \Delta\varphi(j,\mathbf{y}') = K_X(x_i, x_j) \odot K_{\Delta Y}(\mathbf{y}, \mathbf{y}')$ is the joint kernel composed of the input $K_X(x_i, x_j)$ and output $K_{\Delta Y}(\mathbf{y}_i, \mathbf{y}')$ kernels. The underlying joint feature map is expressed by

$$\Delta\varphi(i,\mathbf{y}) = (\varphi(x_i, \mathbf{y}_i) - \varphi(x, \mathbf{y})) = \phi(x_i) \otimes (\psi(\mathbf{y}_i) - \psi(\mathbf{y})) \,,$$

that is, joint feature difference vectors between the true $(\mathbf{y}_i)$ and a competing output $(\mathbf{y})$.

As the input kernel we use the hash fingerprint Tanimoto kernel [9] that was previously shown [15] to be a well performing kernel in this task. Hash fingerprints enumerate all linear fragments of length n in a molecule. A hash function assigns each fragment a hash value that determines its position in descriptor space. Given two fingerprint vectors x and z, Tanimoto kernel is the way to measure their similarity defined as

$$K_X(x,z) = \frac{|I(x) \cap I(z)|}{|I(x) \cup I(z)|},$$

where $I(x)$ denotes the set of indices of 1-bits in x .

As the loss function we use *Hamming loss*

$$\ell_\Delta(\mathbf{y}, \mathbf{u}) = \sum_j [\![y_j \neq u_j]\!]$$

that is gradually increasing in the number of incorrect microlabels so that we can make a difference between 'nearly correct' and 'clearly incorrect' multilabel predictions.

2.2 Graph Generation for Cancer Cell Lines

In the anti-cancer bioactivity prediction problem, a single task entails classification of drug molecules according to whether they are active or inactive against one of the 60 cancer cell lines. The nodes of the graph G to be labeled thus correspond to cancer cell lines. The edges of the graph depict coupling of the tasks, denoting a potential statistical dependency that is to be utilized in predicting the graph labels (Figure 2).

To generate random graphs G_t we use two approaches.

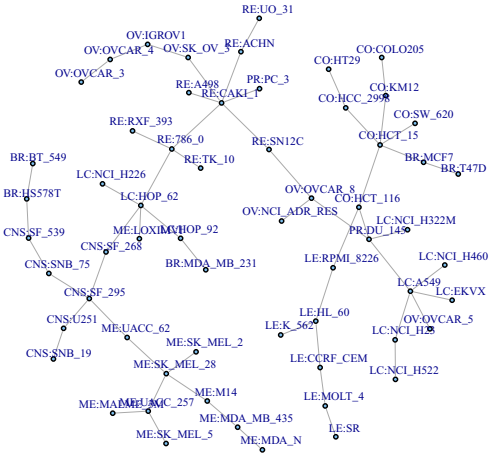


Fig. 2. Example of a cell line graph

- In the random spanning tree approach, we first generate a random correlation matrix and extract the spanning tree out of the matrix with the above described approach.
- In the random pairing approach, one takes each vertex in turn, randomly draws another vertex and couples the two with an edge.

We note that the random graph approach lets us build ensembles whose size is not limited in practice.

We compare the random graphs against the approach used by [15], namely, graphs built from Radiation RNA Array data, available for the cancer cell lines from NCI database[1]. To extract a graph out of the correlation matrix we use the graphical lasso [5] which estimates a sparse graph model by using L_1 (lasso) regularization on inverse covariance matrix, and is theoretically a better method than the simple thresholding of the covariance matrix, applied in [15]. Graphical lasso assumes multivariate Gaussian distribution over cell lines with mean μ and covariance matrix Σ. The inverse covariance matrix Σ^{-1} is a good indicator for conditional independencies [6], where variable i and j are conditional independent given other variables if the $ij$the entry of Σ^{-1} is zero. It imposes L_1 penalty during the estimation of Σ^{-1} to increase the sparsity of the resulted graph. The objective is to maximize the penalized log-likelihood

$$\log\det\Sigma^{-1} - \mathrm{tr}(S\Sigma^{-1}) - \rho||\Sigma^{-1}||_1,$$

where tr is the trace of the matrix, S is empirical covariance matrix, and $||\Sigma^{-1}||_1$ is the L_1 norm of Σ^{-1}. Particularly in our application, we post processed the estimated sparse graph to be a tree-liked one.

[1] `http://discover.nci.nih.gov/cellminer/home.do`

3 Experiments

3.1 Data and Preprocessing

In this paper we use the NCI-Cancer dataset obtained through PubChem Bioassay[2] [17] data repository. The dataset, initiated by National Cancer Institute and National Institutes of Health (NCI/NIH), contains bioactivity information of large number of molecules against several human cancer cell lines in nine different tissue types including leukemia, melanoma and cancers of the lung, colon, brain, ovary, breast, prostate, and kidney. For each molecule tested against a certain cell line, the dataset provide a bioactivity outcome that we use as the classes (active, inactive).

Currently, there are 43197 molecules in the PubChem Bioassay database together with their activities information in 73 cancer cell lines. 60 cell lines have screen experimental results for most molecules and 4547 molecules have no missing data in these cell lines. Therefore these cell lines and molecules are selected and employed in our experiments. However, molecular activity data are highly biased over the 60 cell lines: Around 60% of molecules are inactive in all cell lines, while still a relatively large proportion of molecules are active against all cell lines. These molecules are less likely to be potential drug candidates than the ones in the middle part of the histogram.

To tackle the skewness problem, Su et al. [15] prepared three different versions of the datasets, which approach is also followed here:

Full Data. This dataset contains all 4547 molecules in the NCI-Cancer dataset that have their activity class (active vs. inactive) recorded against all 60 cancer cell lines.

No-Zero-Active. From full data, we removed all molecules that are not active towards any of the cell lines. The remaining 2303 molecules are all active against at least one cell line.

Middle-Active. Here, we followed the preprocessing suggested in [14], and selected molecules that are active in more than 10 cell lines and inactive in more than 10 cell lines. As a result, 545 molecules remained and were employed in our experiments.

3.2 Compared Methods

Three kinds of multi-task classifier ensembles are compared:

- SVM: Support vector machines (SVM) are used as the single-task non-ensemble baseline classifier.
- MMCRF-Glasso: An MMCRF model where the underlying graph connecting the tasks is built by graphical lasso from auxiliary data.
- MMCRF-EnsRT: An ensemble of 1-500 MMCRF models, where the graph connecting the tasks is built by a random spanning tree.

[2] `http://pubchem.ncbi.nlm.nih.gov`

– MMCRF-EnsRP: An ensemble of 1-500 MMCRF models, where the graph connecting the tasks is built by random pairing of the tasks.

In the tests by [15], a relatively hard margin ($C = 100$) emerged as the most favorable for SVM, while MMCRF proved to be quite insensitive as regarding margin softness. Here we used the same value for all compared classifiers.

3.3 Experiment Setup and Performance Measures

Because of the skewness of the multilabel distribution we used the following *stratified 5-fold cross-validation* scheme in all experiments reported such that we group the molecules in equivalence classes based on the number of cell lines they are active against. Then each group is randomly split among the five folds. The procedure ensures that also the smaller groups have representation in all folds. Besides overall classification accuracy, we also report microlabel F_1 score, the harmonic mean of precision and recall

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}.$$

In particular, we pool together individual microlabel predictions over all examples and all cell lines, and count accuracy and F_1 from the pool.

We generated hash fragments features from OpenBabel[3] which is a chemical toolbox available in public domain. We used default value for enumerating all linear structures up to length seven. Then Tanimoto kernel was built based on hash fingerprints features and normalized.

3.4 Results

Figure 3 illustrates the performance of the compared methods on the three versions of the datasets. All models based on MMCRF are clearly more accurate than SVM. Among single models and small ensembles, MMCRF-Glasso is the most competitive method, showing that the auxiliary data contains information that can be successfully used to improve predictive performance.

Both the random pairing and random tree based ensembles steadily improve accuracy and F_1 score as the number of base models increases. SVM falls behind the random graph ensembles even on small ensemble sizes ($T < 5$). With larger ensemble sizes, both types of ensembles end up superior to MMCRF-Glasso in terms of classification accuracy. In terms of F_1 score, the best method depends on the dataset: on the Middle-Active dataset, the random tree ensemble outperform random pairing one, and MMCRF-Glasso is slightly behind. On No-zero-Active and Full data, random pairing ensemble ends up the best method. This result might reflect the sizes of the datasets: the Middle-Active dataset is significantly smaller than the other two, and perhaps the random pairing ensemble requires more data for best results.

[3] `http://openbabel.org`

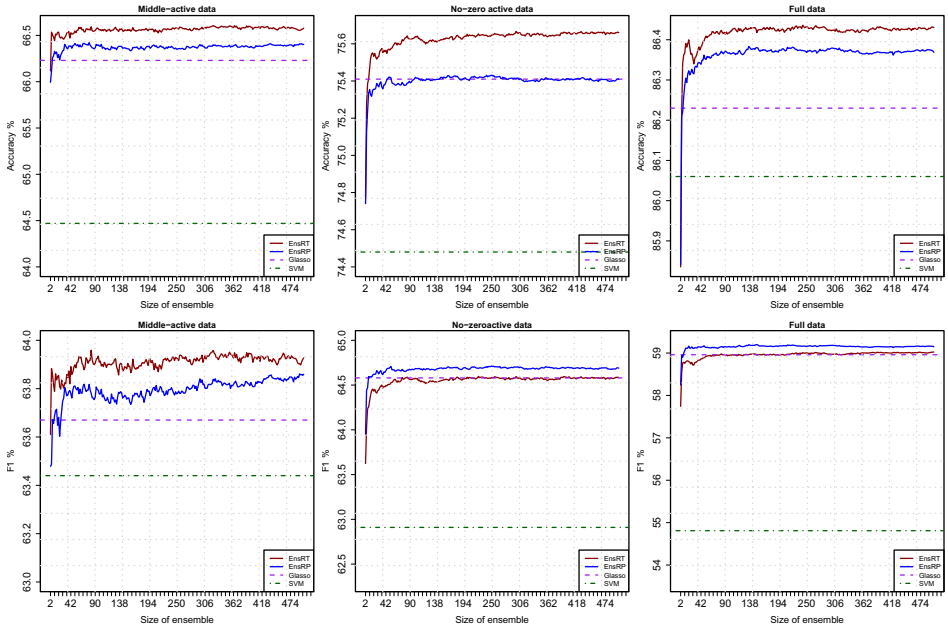


Fig. 3. Accuracy against number of individual classifiers in ensemble methods from different version of datasets. The red line corresponds to random tree ensemble, and blue line is random pairing ensemble. The performance of single models (SVM and MMCRF-Glasso) are depicted by the horizontal lines.

Table 1 shows the prediction performance from SVM, Glasso, EnsRP and EnsRT from three versions of the dataset. We performed two-tailed sign test to identify whether the differences in accuracy and F_1 score in individual cell lines are statistically significant. P-values for the difference over the worst classifier and the ones towards the best classifier are shown as asterisks and crosses. The result shows that, in terms of accuracy and F_1 the multi-task methods outperform SVM in all versions of datasets in a statistically significant manner. EnsRT outperforms Glasso in terms of accuracy in statistically very significant manner.

4 Discussion

The results of this paper show that ensemble methods can be effectively combined with a graph-based multi-task learner such as MMCRF. From machine learning point of view, perhaps the most surprising result obtained here is that in an ensemble, the base graph labeling models can be successfully learnt on random graphs, as opposed to using some auxiliary data or prior knowledge to extract graphs that aim to reflect statistical dependencies.

The present ensemble method differs from previous approaches in that the diversity among the base classifiers arises from the different random output

Table 1. Overall accuracy and microlabel F_1 scores. P-values for the differences over the worst classifier in each version of the dataset are marked with asterisks. P-values for the differences towards the best classifier are marked with crosses. Single, double and triple symbols correspond to p-value below 0.05, 0.01 and 0.001.

Dataset	Accuracy				$\mathbf{F_1}$			
	SVM	Glasso	EnsRP	EnsRT	SVM	Glasso	EnsRP	EnsRT
Middle-Active	$64.5\%_{\dagger\,\dagger\,\dagger}$	$66.2\%^{***}_{\dagger\dagger\dagger}$	$66.5\%^{***}_{\dagger\dagger}$	$\mathbf{66.6}\%^{***}$	$63.4\%_{\dagger}$	63.7%	$63.9\%^{*}$	$\mathbf{63.9}\%^{*}$
No-Zero-Active	$74.5\%_{\dagger\,\dagger\,\dagger}$	$75.4\%^{***}_{\dagger\,\dagger\,\dagger}$	$75.4\%^{***}_{\dagger\,\dagger\,\dagger}$	$\mathbf{75.7}\%^{***}$	$62.9\%_{\dagger\,\dagger\,\dagger}$	$64.6\%^{***}$	$\mathbf{64.7}\%^{***}$	$64.6\%^{***}$
Full	$86.1\%_{\dagger\,\dagger\,\dagger}$	$86.2\%^{***}_{\dagger\,\dagger\,\dagger}$	$86.3\%^{***}$	$\mathbf{86.4}\%^{***}$	$54.8\%_{\dagger\,\dagger\,\dagger}$	$59.0\%^{***}$	$\mathbf{59.2}\%^{***}$	$59.0\%^{***}_{\dagger\,\dagger\,\dagger}$

structures, we do not reweight training examples as in boosting and we do not resample the data like in bagging methods. At the same time, each weak learner is trained to discriminate between different multilabels as well as possible.

Another way to understand the phenomenon is to see the edges of the task network as 'experts', and the collection of edges adjacent to a node as a 'expert committee' voting on the node label, each from a different context. The random pairing of tasks then induces a random set of experts. Random tree of tasks, in addition, makes the experts to negotiate on all node labels in order to keep the tree labeled consistently. Our experiments suggest that enforcing this consistency also may be beneficial.

5 Conclusions

We presented an ensemble approach for multi-task classification of drug bioactivity. The base classifiers of the ensemble, learned by Max-Margin Conditional Random Field algorithm (MMCRF), predict a labeling of a graph coupling the tasks together. The predictive performance of two types of ensembles, one based on random pairing of tasks, another based on a random spanning tree of tasks, surpasses that of SVM as well as single MMCRF model where the underlying graph has been built from auxiliary data using graphical lasso.

Acknowledgements. The work was financially supported by Helsinki Doctoral Programme in Computer Science (Hecse), Academy of Finland grant 118653 (ALGODAN), and in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, ICT-2007-216886. This publication only reflects the authors' views.

References

1. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
2. Ceroni, A., Costa, F., Frasconi, P.: Classification of small molecules by two- and three-dimensional decomposition kernels. Bioinformatics 23, 2038–2045 (2007)
3. Dietterich, T.: Ensemble methods in machine learning. Multiple classifier systems, 1–15 (2000)

4. Esuli, A., Fagni, T., Sebastiani, F.: Boosting multi-label hierarchical text categorization. Information Retrieval 11(4), 287–313 (2008)
5. Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical lasso. Biostatistics 9(3), 432–441 (2008)
6. Meinshausen, N., Bühlmann, P., Zürich, E.: High dimensional graphs and variable selection with the lasso. Annals of Statistics 34, 1436–1462 (2006)
7. Obrezanova, O., Segall, M.D.: Gaussian processes for classification: Qsar modeling of admet and target activity. Journal of Chemical Information and Modeling 50(6), 1053–1061 (2010)
8. Opitz, D., Maclin, R.: Popular ensemble methods: an empirical study. Journal of Artificial Intelligence Research 11, 169–198 (1999)
9. Ralaivola, L., Swamidass, S., Saigo, H., Baldi, P.: Graph kernels for chemical informatics. Neural Networks 18, 1093–1110 (2005)
10. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Kernel-Based Learning of Hierarchical Multilabel Classification Models. The Journal of Machine Learning Research 7, 1601–1626 (2006)
11. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Efficient algorithms for max-margin structured classification. Predicting Structured Data, 105–129 (2007)
12. Schapire, R.E., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine Learning 39(2/3), 135–168 (2000)
13. Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., Džeroski, S.: Predicting gene function using hierarchical multi-label decision tree ensembles. BMC bioinformatics 11(1), 2 (2010)
14. Shivakumar, P., Krauthammer, M.: Structural similarity assessment for drug sensitivity prediction in cancer. Bioinformatics 10, S17 (2009)
15. Su, H., Heinonen, M., Rousu, J.: Structured Output Prediction of Anti-cancer Drug Activity. In: Dijkstra, T.M.H., Tsivtsivadze, E., Marchiori, E., Heskes, T. (eds.) PRIB 2010. LNCS, vol. 6282, pp. 38–49. Springer, Heidelberg (2010)
16. Trotter, M., Buxton, M., Holden, S.: Drug design by machine learning: support vector machines for pharmaceutical data analysis. Comp. and Chem. 26, 1–20 (2001)
17. Wang, Y., Bolton, E., Dracheva, S., Karapetyan, K., Shoemaker, B., Suzek, T., Wang, J., Xiao, J., Zhang, J., Bryant, S.: An overview of the pubchem bioassay resource. Nucleic Acids Research 38, D255–D266 (2009)
18. Yan, R., Tesic, J., Smith, J.: Model-shared subspace boosting for multi-label classification. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 834–843. ACM (2007)

Publication IV

Hongyu Su, Juho Rousu. Multilabel Classification through Random Graph Ensembles. *Machine Learning*, DOI:10.1007/s10994-014-5465-9, Published Online 26 Pages, September 2014.

Mach Learn
DOI 10.1007/s10994-014-5465-9

Multilabel classification through random graph ensembles

Hongyu Su · Juho Rousu

Received: 16 January 2014 / Accepted: 14 July 2014

Abstract We present new methods for multilabel classification, relying on ensemble learning on a collection of random output graphs imposed on the multilabel, and a kernel-based structured output learner as the base classifier. For ensemble learning, differences among the output graphs provide the required base classifier diversity and lead to improved performance in the increasing size of the ensemble. We study different methods of forming the ensemble prediction, including majority voting and two methods that perform inferences over the graph structures before or after combining the base models into the ensemble. We put forward a theoretical explanation of the behaviour of multilabel ensembles in terms of the diversity and coherence of microlabel predictions, generalizing previous work on single target ensembles. We compare our methods on a set of heterogeneous multilabel benchmark problems against the state-of-the-art machine learning approaches, including multilabel AdaBoost, convex multitask feature learning, as well as single target learning approaches represented by Bagging and SVM. In our experiments, the random graph ensembles are very competitive and robust, ranking first or second on most of the datasets. Overall, our results show that our proposed random graph ensembles are viable alternatives to flat multilabel and multitask learners.

Keywords Multilabel classification · Structured output · Ensemble methods · Kernel methods · Graphical models

1 Introduction

Multilabel and multitask classification rely on representations and learning methods that allow us to leverage the dependencies between the different labels. When such dependencies

Editors: Cheng Soon Ong, Tu Bao Ho, Wray Buntine, Bob Williamson, and Masashi Sugiyama.

H. Su ( ) · J. Rousu
Helsinki Institute for Information Technology HIIT, Department of Information and Computer Science, Aalto University, Konemiehentie 2, 02150 Espoo, Finland
e-mail: hongyu.su@aalto.fi

J. Rousu
e-mail: juho.rousu@aalto.fi

Published online: 15 August 2014

are given in form of a graph structure such as a sequence, a hierarchy or a network, structured output prediction (Taskar et al. 2004; Tsochantaridis et al. 2004; Rousu et al. 2006) becomes a viable option, and has achieved a remarkable success. For multilabel classification, limiting the applicability of the structured output prediction methods is the very fact they require the predefined output structure to be at hand, or alternatively auxiliary data where the structure can be learned from. When these are not available, flat multilabel learners or collections of single target classifiers are thus often resorted to.

In this paper, we study a different approach, namely using ensembles of graph labeling classifiers, trained on randomly generated output graph structures. The methods are based on the idea that variation in the graph structures shifts the inductive bias of the base learners and causes diversity in the predicted multilabels. Each base learner, on the other hand, is trained to predict as good as possible multilabels, which makes them satisfy the weak learning assumption, necessary for successful ensemble learning.

Ensembles of multitask or multilabel classifiers have been proposed, but with important differences. The first group of methods, boosting type, rely on changing the weights of the training instances so that difficult to classify instances gradually receive more and more weights. The AdaBoost boosting framework has spawned multilabel variants (Schapire and Singer 2000; Esuli et al. 2008). In these methods the multilabel is considered essentially as a flat vector. The second group of methods, Bagging, are based on bootstrap sampling the training set several times and building the base classifiers from the bootstrap samples. Thirdly, randomization has been used as the means of achieving diversity by Yan et al. (2007) who select different random subsets of input features and examples to induce the base classifiers, and by Su and Rousu (2011) who use majority voting over random graphs in drug bioactivity prediction context. Here we extend the last approach to two other types of ensembles and a wider set of applications, with gain in prediction performances. A preliminary version of this article appeared as Su and Rousu (2013).

The remainder of the article is structured as follows. In Sect. 2 we present the structured output model used as the graph labeling base classifier. In Sect. 3 we present three aggregation strategies based on random graph labeling. In Sect. 4 we present empirical evaluation of the methods. In Sect. 5 we present concluding remarks.

2 Multilabel classification through graph labeling

We start by detailing the graph labeling classification methods that are subsequently used as the base classifier. We examine the following multilabel classification setting. We assume data from a domain $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is a set and $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k$ is the set of multilabels, represented by a Cartesian product of the sets $\mathcal{Y}_j = \{1, \ldots, l_j\}$, $j = 1, \ldots, k$. In particular, setting $k = 1, l_1 = 2$ ($\mathcal{Y} = \{1, 2\}$) corresponds to binary classification problem. A vector $\mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y}$ is called the *multilabel* and the components y_j are called the *microlabels*. We assume that a training set $\{(x_i, \mathbf{y}_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}$ has been given. A pair $(x_i, \mathbf{y})$ where x_i is a training pattern and $\mathbf{y} \in \mathcal{Y}$ is arbitrary, is called a *pseudo-example*, to denote the fact that the pair may or may not be generated by the distribution generating the training examples. The goal is to learn a model $F : \mathcal{X} \mapsto \mathcal{Y}$ so that the expected loss over predictions on future instances is minimized, where the loss function is chosen suitably for multilabel learning problems. By $\mathbf{1}_{\{\cdot\}}$ we denote the indicator function $\mathbf{1}_{\{A\}} = 1$, if A is true, $\mathbf{1}_{\{A\}} = 0$ otherwise.

Here, we consider solving multilabel classification with graph labeling classifiers that, in addition to the training set, assumes a graph $G = (V, E)$ with nodes $V = \{1, \ldots, k\}$ corresponding to microlabels and edges $E \subseteq V \times V$ denoting potential dependencies between the

microlabels. For any edge $e = (j, j') \in E$, we denote by $\mathbf{y}_e = (y_j, y_{j'})$ the edge label of e in multilabel $\mathbf{y}$, induced by concatenating the microlabels corresponding to end points of e, with corresponding domain of edge labels $\mathcal{Y}_e = \mathcal{Y}_j \times \mathcal{Y}_{j'}$. By $\mathbf{y}_{ie}$ we denote the label of the edge e in the i'th training example. Hence, for a fixed multilabel $\mathbf{y}$, we can compute corresponding node label y_j of node $j \in V$ and edge label $\mathbf{y}_e$ of edge $e \in E$. We also use separate notation for node and edge labels that are free, that is, not bound to any particular multilabel. We denote by u_j a possible label of node j, and $\mathbf{u}_e$ a possible labels of edge e. Naturally, $u_j \in \mathcal{Y}_j$ and $\mathbf{u}_e \in \mathcal{Y}_e$. See supplementary material for a comprehensive list of notations.

2.1 Graph labeling classifier

As the graph labeling classifier in this work we use max-margin structured output prediction, with the aim to learn a compatibility score for pairs $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, indicating how well an input goes together with an output. Naturally, such a score for coupling an input x with the correct multilabel $\mathbf{y}$ should be higher than the score of the same input with an incorrect multilabel $\mathbf{y}'$. The compatibility score between an input x and a multilabel $\mathbf{y}$ takes the form

$$\psi(x, \mathbf{y}) = \langle w, \varphi(x, \mathbf{y}) \rangle = \sum_{e \in E} \langle w_e, \varphi_e(x, \mathbf{y}_e) \rangle = \sum_{e \in E} \psi_e(x, \mathbf{y}_e), \tag{1}$$

where by $\langle \cdot, \cdot \rangle$ we denote the inner product and parameter w contains the feature weights to be learned. $\psi_e(x, \mathbf{y}_e)$ is a shorthand for the compatibility score, or the potential, between the input x and an edge label $\mathbf{y}_e$, defined as $\psi_e(x, \mathbf{y}_e) = \langle w_e, \varphi_e(x, \mathbf{y}_e) \rangle$, where w_e is the element of w that associates to edge e.

The joint feature map

$$\varphi(x, \mathbf{y}) = \phi(x) \otimes \Upsilon(\mathbf{y}) = \phi(x) \otimes (\Upsilon_e(\mathbf{y}_e))_{e \in E} = (\varphi_e(x, \mathbf{y}_e))_{e \in E}$$

is given by a tensor product of an input feature $\phi(x)$ and the feature space embedding of the multilabel $\Upsilon(\mathbf{y}) = (\Upsilon_e(\mathbf{y}_e))_{e \in E}$, consisting of edge label indicators $\Upsilon_e(\mathbf{y}_e) = \left(\mathbf{1}_{\{\mathbf{y}_e = \mathbf{u}_e\}}\right)_{\mathbf{u}_e \in \mathcal{Y}_e}$. The benefit of the tensor product representation is that context (edge label) sensitive weights can be learned for input features and no prior alignment of input and output features needs to be assumed.

The parameters w of the model are learned through max-margin optimization, where the primal optimization problem takes the form (e.g. Taskar et al. 2004; Tsochantaridis et al. 2004; Rousu et al. 2006)

$$\begin{aligned} &\min_{w, \xi} \frac{1}{2} ||w||^2 + C \sum_{i=1}^{m} \xi_i \\ &\text{s.t. } \langle w, \varphi(x_i, \mathbf{y}_i) \rangle \geq \max_{\mathbf{y} \in \mathcal{Y}} \left(\langle w, \varphi(x_i, \mathbf{y}) \rangle + \ell(\mathbf{y}_i, \mathbf{y}) \right) - \xi_i, \\ &\quad \xi_i \geq 0, \ \forall i \in \{1, \ldots, m\}, \end{aligned} \tag{2}$$

where ξ_i denotes the slack allotted to each example, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss function between pseudo-label and correct label, and C is the slack parameter that controls the amount of regularization in the model. The primal form can be interpreted as maximizing the minimum margin between the correct training example and incorrect pseudo-examples, scaled by the loss function. The intuition behind loss-scaled margin is that example with nearly correct multilabel would require smaller margin than with multilabel that is quite different from the correct one. Denoting $\Delta\varphi(x_i, \mathbf{y}) = \varphi(x_i, \mathbf{y}_i) - \varphi(x_i, \mathbf{y})$, the Lagrangian of the problem is given by

$$L(w, \xi, \alpha, \rho) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i,\mathbf{y}}\alpha(i,\mathbf{y})\left(\langle w, \Delta\varphi(x_i,\mathbf{y})\rangle + \ell(\mathbf{y}_i,\mathbf{y})\right) - \sum_i \xi_i\rho_i,$$

where by setting derivatives to zero with respect to w we obtain

$$w = \sum_{i,\mathbf{y}}\alpha(i,\mathbf{y})\Delta\varphi(x_i,\mathbf{y}), \tag{3}$$

and the zero derivatives for ξ give the box constraint $\sum_y \alpha(i, y) \leq C$ for all i, while the dual variables ρ_i are canceled out. Maximization with respect to α's gives the dual optimization problem as

$$\begin{aligned} \max_{\alpha\geq 0}\ & \alpha^T\ell - \frac{1}{2}\alpha^T K\alpha \\ \text{s.t.}\ & \sum_{\mathbf{y}}\alpha(i,\mathbf{y}) \leq C, \forall i \in \{1,\ldots,m\}, \end{aligned} \tag{4}$$

where $\alpha = (\alpha(i,\mathbf{y}))_{i,\mathbf{y}}$ denotes the vector of dual variables and $\ell = (\ell(\mathbf{y}_i,\mathbf{y}))_{i,\mathbf{y}}$ is the vector of losses for each pseudo-example $(x_i, \mathbf{y})$. The joint kernel

$$\begin{aligned} K(x_i,\mathbf{y};x_j,\mathbf{y}') &= \langle\varphi(x_i,\mathbf{y}_i) - \varphi(x_i,\mathbf{y}), \varphi(x_j,\mathbf{y}_j) - \varphi(x_j,\mathbf{y}')\rangle \\ &= \langle\phi(x_i),\phi(x_j)\rangle_\phi \cdot \langle(\Upsilon(\mathbf{y}_i) - \Upsilon(\mathbf{y}), \Upsilon(\mathbf{y}_j) - \Upsilon(\mathbf{y}')\rangle_\Upsilon \\ &= K_\phi(x_i,x_j)\cdot\left(K_\Upsilon(\mathbf{y}_i,\mathbf{y}_j) - K_\Upsilon(\mathbf{y}_i,\mathbf{y}') - K_\Upsilon(\mathbf{y},\mathbf{y}_j) + K_\Upsilon(\mathbf{y},\mathbf{y}')\right) \\ &= K_\phi(x_i,x_j)\cdot K_{\Delta\Upsilon}(\mathbf{y}_i,\mathbf{y};\mathbf{y}_j,\mathbf{y}') \end{aligned}$$

is composed by product of input kernel $K_\phi(x_i, x_j) = \langle x_i, x_j\rangle_\phi$ and output kernel

$$K_{\Delta\Upsilon}(\mathbf{y}_i,\mathbf{y};\mathbf{y}_j,\mathbf{y}') = \left(K_\Upsilon(\mathbf{y}_i,\mathbf{y}_j) - K_\Upsilon(\mathbf{y}_i,\mathbf{y}') - K_\Upsilon(\mathbf{y},\mathbf{y}_j) + K_\Upsilon(\mathbf{y},\mathbf{y}')\right),$$

where $K_\Upsilon(\mathbf{y},\mathbf{y}') = \langle\Upsilon(\mathbf{y}'),\Upsilon(\mathbf{y})\rangle$.

2.2 Factorized dual form

The dual optimization problem (4) is a challenging one to solve due to the exponential-sized dual variable space, thus efficient algorithms are required. A tractable form is obtained via factorizing the problem according to the graph structure. Following Rousu et al. (2007), we transform (4) into the factorized dual form, where the edge-marginals of dual variables are used in place of the original dual variables

$$\mu(i,e,\mathbf{u}_e) = \sum_{\mathbf{y}\in\mathcal{Y}}\mathbf{1}_{\{\Upsilon_e(\mathbf{y})=\mathbf{u}_e\}}\alpha(i,\mathbf{y}), \tag{5}$$

where $e = (j, j') \in E$ is an edge in the output graph and $\mathbf{u}_e \in \mathcal{Y}_j \times \mathcal{Y}_{j'}$ is a possible label for the edge (j, j').

The output kernel decomposes by the edges of the graph as

$$K_\Upsilon(\mathbf{y},\mathbf{y}') = \langle\Upsilon(\mathbf{y}'),\Upsilon(\mathbf{y})\rangle = \sum_e K_{\Upsilon,e}(\mathbf{y}_e,\mathbf{y}'_e),$$

where $K_{\Upsilon,e}(u,u') = \langle \Upsilon_e(u), \Upsilon_e(u') \rangle_\Upsilon$. Given the joint features defined by the tensor product, the joint kernel also decomposes as

$$\begin{aligned} K(x_i, \mathbf{y}; x_j, \mathbf{y}') &= K_\phi(x,x') K_{\Delta\Upsilon}(\mathbf{y}_i, \mathbf{y}; \mathbf{y}_j, \mathbf{y}') = \\ &= \sum_e K_\phi(x,x') K_{\Delta\Upsilon,e}(\mathbf{y}_e, \mathbf{y}'_e) = \sum_e K_e(x, \mathbf{y}_e; x', \mathbf{y}'_e), \end{aligned}$$

where we have denoted

$$\begin{aligned} &K_{\Delta\Upsilon,e}(\mathbf{y}_{ie}, \mathbf{y}_e; \mathbf{y}_{je}, \mathbf{y}'_e) = \\ &\qquad \left(K_{\Upsilon,e}(\mathbf{y}_{ie}, \mathbf{y}_{je}) - K_{\Upsilon,e}(\mathbf{y}_{ie}, \mathbf{y}'_e) - K_{\Upsilon,e}(\mathbf{y}_e, \mathbf{y}_{je}) + K_{\Upsilon,e}(\mathbf{y}_e, \mathbf{y}'_e) \right). \end{aligned}$$

Using the above, the quadratic part of the objective factorizes as follows

$$\begin{aligned} \alpha^T K \alpha &= \sum_e \sum_{i,j} \sum_{\mathbf{y},\mathbf{y}'} \alpha(i,\mathbf{y}) K_e(x_i, \mathbf{y}_e; x_j, \mathbf{y}'_e) \alpha(j,\mathbf{y}') \\ &= \sum_e \sum_{i,j} \sum_{\mathbf{u},\mathbf{u}'} K_e(x_i, \mathbf{u}; x_j, \mathbf{u}') \sum_{\mathbf{y}:\mathbf{y}_e=\mathbf{u}} \sum_{\mathbf{y}':\mathbf{y}'_e=\mathbf{u}'} \alpha(i,\mathbf{y}) \alpha(j,\mathbf{y}') \\ &= \sum_e \sum_{i,j} \sum_{\mathbf{u},\mathbf{u}'} \mu(i,e,\mathbf{u}) K_e(x_i, \mathbf{u}; x_j, \mathbf{u}') \mu(j,e,\mathbf{u}') \\ &= \mu^T K_E \mu, \end{aligned} \tag{6}$$

where $K_E = \operatorname{diag}(K_e, e \in E)$ is a block diagonal matrix with edge-specific kernel blocks K_e and $\mu = (\mu(i,e,u))_{i,e,u}$ is the vector of marginal dual variables. We assume a loss function that can be expressed in a decomposed form as

$$\ell(\mathbf{y}, \mathbf{y}') = \sum_e \ell_e(\mathbf{y}_e, \mathbf{y}'_e),$$

a property that is satisfied by the Hamming loss family, counting incorrectly predicted nodes (i.e. microlabel loss) or edges, and are thus suitable for our purpose. With a decomposable loss function, the linear part of the objective satisfies

$$\begin{aligned} \sum_{i=1}^m \sum_{\mathbf{y}\in\mathcal{Y}} \alpha(i,\mathbf{y}) \ell(\mathbf{y}_i, \mathbf{y}) &= \sum_{i=1}^m \sum_{\mathbf{y}} \alpha(i,\mathbf{y}) \sum_e \ell_e(\mathbf{y}_{ie}, \mathbf{y}_e) = \\ &= \sum_{i=1}^m \sum_{e\in E} \sum_{u\in\mathcal{Y}_e} \sum_{\mathbf{y}:\mathbf{y}_e=u} \alpha(i,\mathbf{y}) \ell_e(\mathbf{y}_{ie}, u) \\ &= \sum_{i=1}^m \sum_{e\in E} \sum_{u\in\mathcal{Y}_e} \mu(e,u) \ell_e(\mathbf{y}_{ie}, u) = \sum_{i=1}^m \mu_i^T \ell_i = \mu^T \ell_E, \end{aligned} \tag{7}$$

where $\ell_E = (\ell_i)_{i=1}^m = (\ell_e(i,u))_{i=1,e\in E,u\in\mathcal{Y}_e}^m$ is the vector of losses. Given the above, we can state the dual problem (4) in equivalent form (c.f. Taskar et al. 2004; Rousu et al. 2007) as

$$\max_{\mu\in\mathcal{M}} \boldsymbol{\mu}^T \ell - \frac{1}{2} \boldsymbol{\mu}^T K_E \boldsymbol{\mu}, \tag{8}$$

where the constraint set is the marginal polytope (c.f. Rousu et al. 2007; Wainwright et al. 2005)

$$\mathcal{M} = \{\mu | \exists \alpha \in \mathcal{A} \text{ s.t. } \mu(i, e, \mathbf{u}_e) = \sum_{\mathbf{y} \in \mathcal{Y}} \mathbf{1}_{\{\mathbf{y}_{ie} = \mathbf{u}_e\}} \alpha(i, \mathbf{y}), \forall i, \mathbf{u}_e, e\}$$

of the dual variables, the set of all combinations of marginal dual variables (5) of training examples that correspond to some α in the original dual feasible set $\mathcal{A} = \{\alpha | \alpha(i, \mathbf{y}) \geq 0, \sum_{\mathbf{y}} \alpha(i, \mathbf{y}) \leq C, \forall i\}$ in (4). Note that the above definition of $\mathcal{M}$ automatically takes care of the consistency constraints between marginal dual variables (c.f. Rousu et al. 2007).

The factorized dual problem (8) is a quadratic program with a number of variables *linear* in both the size of the output network and the number of training examples. There is an exponential reduction in the number of dual variables from the original dual (4), however, with the penalty of more complex feasible polytope $\mathcal{M}$. For solving (8) we use MMCRF (Rousu et al. 2007) that relies on a conditional gradient method. Update directions are found in linear time via probabilistic inference (explained in the next section), making use of the the exact correspondence of maximum margin violating multilabel in the primal (2) and steepest feasible gradient of the dual objective (4).

2.3 Inference

In both training and prediction, efficient inference is required over the multilabel spaces. In training any of the models (2, 4, 8), one needs to iteratively solve the *loss-augmented inference* problem

$$\begin{aligned} \bar{\mathbf{y}}(x_i) &= \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}} \left(\langle w, \varphi(x_i, \mathbf{y}) \rangle + \ell(\mathbf{y}_i, \mathbf{y}) \right) \\ &= \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_e \langle w_e, \varphi_e(x_i, \mathbf{y}_e) \rangle + \ell_e(\mathbf{y}_e, \mathbf{y}_{ie}) \end{aligned} \tag{9}$$

that finds for each example the multilabel that violates its margins the most (i.e. the worst margin violator) given the current w. Depending on the optimization algorithm, the worst-margin violator may be used to grow a constraint set (column-generation methods), or to define an update direction (structured perceptron, conditional gradient).

In the prediction phase, the inference problem to be solved is simply to find the highest scoring multilabel for each example:

$$\hat{\mathbf{y}}(x) = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \varphi(x, \mathbf{y}) \rangle = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_e \langle w_e, \varphi_e(x, \mathbf{y}) \rangle \tag{10}$$

Both of the above inference problems can be solved in the factorized dual, thus allowing us to take advantage of kernels for complex and high-dimensional inputs, as well as the linear-size dual variable space.

Next, we put forward a lemma that shows explicitly how the compatibility score $\psi_e(x, \mathbf{y}_e) = \langle w_e, \varphi_e(x, \mathbf{y}_e) \rangle$ of labeling an edge e as $\mathbf{y}_e$ given input x can be expressed in terms of kernels and marginal dual variables. We note that the property is already used in marginal dual based structured output methods such as MMCRF, however, below we make the property explicit, to facilitate the description of the ensemble learning methods.

Lemma 1 *Let w be the solution to (2), $\varphi(x, \mathbf{y})$ be the joint feature map, and let $G = (V, E)$ be the graph defining the output graph structure, and let us denote*

$$H_e(x_i, \mathbf{u}_e; x, \mathbf{y}_e) = K_\phi(x, x_i) \cdot \left(K_{\Upsilon,e}(\mathbf{y}_{ie}, \mathbf{y}_e) - K_{\Upsilon,e}(\mathbf{u}_e, \mathbf{y}_e) \right).$$

Then, we have

$$\psi_e(x, \mathbf{y}_e) = \langle w_e, \varphi_e(x, \mathbf{y}_e)\rangle = \sum_{i,\mathbf{u}_e} \mu(i, e, \mathbf{u}_e) \cdot H_e(x_i, \mathbf{u}_e; x, \mathbf{y}_e),$$

where μ is the marginal dual variable learned by solving optimization problem (8).

Proof Using (3) and (5), and elementary tensor algebra, the compatibility score of a example $(x, \mathbf{y}')$ is given by

$$\begin{aligned}
\langle w, \varphi(x, \mathbf{y}')\rangle &= \sum_i \sum_{\mathbf{y}} \alpha(i, \mathbf{y})\langle \Delta\varphi(x_i, \mathbf{y}), \varphi(x, \mathbf{y}')\rangle \\
&= \sum_i \sum_e \sum_{\mathbf{u}_e} \sum_{\mathbf{y}:\mathbf{y}_e=\mathbf{u}_e} \alpha(i, \mathbf{y})\langle \Delta\varphi_e(x_i, \mathbf{u}_e), \varphi_e(x, \mathbf{y}'_e)\rangle \\
&= \sum_e \sum_i \sum_{\mathbf{u}_e} \mu(i, e, \mathbf{u}_e)\langle \phi(x_i) \otimes (\Upsilon_e(\mathbf{y}_{ie}) - \Upsilon_e(\mathbf{u}_e)), \phi(x) \otimes \Upsilon_e(\mathbf{y}'_e)\rangle \\
&= \sum_e \sum_i \sum_{\mathbf{u}_e} \mu(i, e, \mathbf{u}_e) K_\phi(x_i, x)\langle \Upsilon_e(\mathbf{y}_{ie}) - \Upsilon_e(\mathbf{u}_e), \Upsilon_e(\mathbf{y}'_e)\rangle \\
&= \sum_e \sum_i \sum_{\mathbf{u}_e} \mu(i, e, \mathbf{u}_e) K_\phi(x_i, x) \cdot \left(K_{\Upsilon,e}(\mathbf{y}_{ie}, \mathbf{y}'_e) - K_{\Upsilon,e}(\mathbf{u}_e, \mathbf{y}'_e)\right) \\
&= \sum_e \sum_{i,\mathbf{u}_e} \mu(i, e, \mathbf{u}_e) \cdot H_e(x_i, \mathbf{u}_e; x, \mathbf{y}'_e).
\end{aligned}$$

The loss-augmented inference problem can thus be equivalently expressed in the factorized dual by

$$\begin{aligned}
\bar{\mathbf{y}}(x) &= \mathbf{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_e \psi_e(x, \mathbf{y}_e) + \ell_e(\mathbf{y}_e, \mathbf{y}_{ie}) \\
&= \mathbf{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_{e,i,\mathbf{u}_e} \mu(i, e, \mathbf{u}_e) H_e(i, \mathbf{u}_e; x, \mathbf{y}_e) + \ell_e(\mathbf{y}_e, \mathbf{y}_{ie}).
\end{aligned} \tag{11}$$

Similarly, the inference problem (10) solved in prediction phase can be solved in the factorized dual by

$$\begin{aligned}
\hat{\mathbf{y}}(x) &= \mathbf{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_e \psi_e(x, \mathbf{y}_e) = \mathbf{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_e \langle \mathbf{w}_e, \varphi_e(x, \mathbf{y}_e)\rangle \\
&= \mathbf{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_{e,i,\mathbf{u}_e} \mu(i, e, \mathbf{u}_e) H_e(i, \mathbf{u}_e; x, \mathbf{y}_e).
\end{aligned} \tag{12}$$

To solve (11) or (12) any commonly used inference technique for graphical models can be applied. In this paper we use MMCRF that relies on the message-passing method, also referred as loopy belief propagation (LBP). We use early stopping in inference of LBP, so that the number of iterations is limited by the diameter of the output graph G.

3 Learning graph labeling ensembles

In this section we consider generating ensembles of multilabel classifiers, where each base model is a graph labeling classifier. Algorithm 1 depicts the general form of the learning approach. We assume a function to output a random graph $G^{(t)}$ for each stage of the ensemble,

a base learner $F^{(t)}(\cdot)$ to learn the graph labeling model, and an aggregation function $A(\cdot)$ to compose the ensemble model. The prediction of the model is then obtained by aggregating the base model predictions

$$F(x) = A(F^{(1)}(x), \ldots, F^{(T)}(x)).$$

Given a set of base models trained on different graph structures we expect the predicted labels of the ensemble have diversity which is known to be necessary for ensemble learning. At the same time, since the graph labeling classifiers aim to learn accurate multilabels, we expect the individual base classifiers to be reasonably accurate, irrespective of the slight changes in the underlying graphs. Indeed, in this work we use randomly generated graphs to emphasize this point. We consider the following three aggregation methods:

– In *majority-voting-ensemble*, each base learner gives a prediction of the multilabel. The ensemble prediction is obtained by taking the most frequent value for each microlabel. Majority voting aggregation is admissible for any multilabel classifier.

Second, we consider two aggregation strategies that assume the base classifier has a conditional random field structure:

– In *average-of-maximum-marginals aggregation*, each base learner infers local maximum marginal scores for each microlabel. The ensemble prediction is taken as the value with highest average local score.
– In *maximum-of-average-marginals aggregation*, the local edge potentials of each base model are first averaged over the ensemble and maximum global marginal scores are inferred from the averages.

In the following, we detail the above aggregation strategies.

3.1 Majority voting ensemble (MVE)

The first ensemble model we consider is the majority voting ensemble (MVE), which was introduced in drug bioactivity prediction context by Su and Rousu (2011). In MVE, the ensemble prediction on each microlabel is the most frequently appearing prediction among the base classifiers

$$F_j^{\text{MVE}}(x) = \mathbf{argmax}_{y_j \in \mathcal{Y}_j} \left(\frac{1}{T} \sum_{i=1}^{T} \mathbf{1}_{\{F_j^{(t)}(x) = y_j\}} \right),$$

where $F^{(t)}(x) = (F_j^{(t)}(x))_{j=1}^{k}$ is the predicted multilabel in the t'th base classifier. When using (8) as the base classifier, predictions $F^{(t)}(x)$ are obtained via solving the inference problem (12). We note, however, in principle, any multilabel learner will fit into the MVE

Input: Training sample $S = \{(x_i, \mathbf{y}_i)\}_{i=1}^{m}$, ensemble size T, graph generating oracle function $outputGraph : t \in \{1, \ldots, T\} \mapsto \mathcal{G}_k$, aggregation function $A(\cdot) : \mathcal{F} \times \cdots \times \mathcal{F} \mapsto \mathcal{Y}$
Output: Multilabel classification ensemble $F(\cdot) : \mathcal{X} \mapsto \mathcal{Y}$
1: **for** $t \in \{1, \ldots, T\}$ **do**
2: $G^{(t)} = \text{outputGraph}(t)$
3: $F^t(\cdot) = \text{learnGraphLabelingClassifier}((x_i)_{i=1}^{m}, (\mathbf{y}_i)_{i=1}^{m}, G^{(t)})$
4: **end for**
5: $F(\cdot) = A(F^{(1)}(\cdot), \ldots, F^{(T)}(\cdot))$

Algorithm 1: Graph Labeling Ensemble Learning

framework as long as it adapts to a collection of output graphs $\mathcal{G} = \{G^{(1)}, \cdots, G^{(T)}\}$ and generates multilabel predictions accordingly from each graph.

3.2 Average-of-max-marginal aggregation (AMM)

Next, we consider an ensemble model where we perform inference over the graph to extract information on the learned compatibility scores in each base model. Thus, we assume that we have access to the compatibility scores between the inputs and edge labels

$$\Psi_E^{(t)}(x) = (\psi_e^{(t)}(x, \mathbf{u}_e))_{e \in E^{(t)}, \mathbf{u}_e \in \mathcal{Y}_e}.$$

In the AMM model, our goal is to infer for each microlabel u_j of each node j its *max-marginal* (Wainwright et al. 2005), that is, the maximum score of a multilabel that is consistent with $y_j = u_j$

$$\tilde{\psi}_j(x, u_j) = \max_{\{\mathbf{y} \in \mathcal{Y} : y_j = u_j\}} \sum_e \psi_e(x, \mathbf{y}_e). \tag{13}$$

One readily sees (13) as a variant of the inference problem (12), with similar solution techniques. The maximization operation fixes the label of the node $y_j = u_j$ and queries the optimal configuration for the remaining part of output graph. In message-passing algorithms, only slight modification is needed to make sure that only the messages consistent with the microlabel restriction are considered. To obtain the vector $\tilde{\Psi}(x) = (\tilde{\psi}_j(x, u_j))_{j,u_j}$ the same inference is repeated for each target-microlabel pair (j, u_j), hence it has quadratic time complexity in the number of edges in the output graph.

Given the max-marginals of the base models, the AMM ensemble is constructed as follows. Let $\mathcal{G} = \{G^{(1)}, \cdots, G^{(T)}\}$ be a set of output graphs, and let $\{\tilde{\Psi}^{(1)}(x), \cdots, \tilde{\Psi}^{(T)}(x)\}$ be the max-marginal vectors of the base classifiers trained on the output graphs. The ensemble prediction for each target is obtained by averaging the max-marginals of the base models and choosing the maximizing microlabel for the node:

$$F_j^{\text{AMM}}(x) = \operatorname*{argmax}_{u_j \in \mathcal{Y}_j} \frac{1}{|T|} \sum_{t=1}^{T} \tilde{\psi}_{j,u_j}^{(t)}(x),$$

and the predicted multilabel is composed from the predicted microlabels

$$F^{\text{AMM}}(x) = \left(F_j^{\text{AMM}}(x)\right)_{j \in V}.$$

An illustration of AMM ensemble scheme is shown in Fig. 1. Edge information on individual base learner are not preserved during AMM ensemble, which is shown as dash line in Fig. 1. In principle, AMM ensemble can give different predictions compared to MVE, since the most frequent label may not be the ensemble prediction if it has lower average max-marginal score.

3.3 Maximum-of-average-marginals aggregation (MAM)

The next model, the maximum-of-average-marginals (MAM) ensemble, first collects the local compatibility scores $\Psi_E^{(t)}(x)$ from individual base learners, averages them and finally performs inference on the global consensus graph with averaged edge potentials. The model is defined as

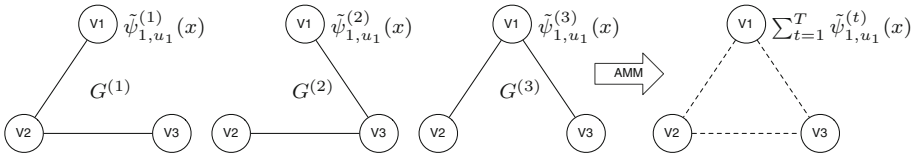


Fig. 1 An example of AMM scheme, where three base models are learned on the output graph $G^{(1)}, G^{(2)}, G^{(3)}$. Given an example x, each base model computes for node v_1 local max-marginals $\tilde{\psi}^{(t)}_{1,u_1}$ for all $u_1 \in \{+,-\}$. The AMM collects local max-marginals $\sum_{t=1}^{T}\tilde{\psi}^{(t)}_{1,u_1}(x)$, and outputs $F_1(x) = +$ if $\sum_{t=1}^{T}\tilde{\psi}^{(t)}_{1,+}(x) \geq \sum_{t=1}^{T}\tilde{\psi}^{(t)}_{1,-}(x)$, otherwise outputs $F_1(x) = -$

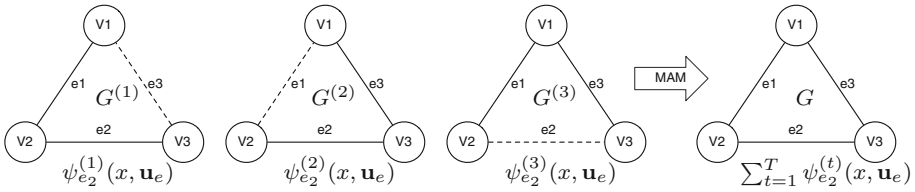


Fig. 2 An example of MAM scheme, where three base models are learned on the output graph $G^{(1)}, G^{(2)}, G^{(3)}$. Given an example x, each base model computes for edge e_2 local edge potentials $\psi^{(t)}_{e_2}(x,\mathbf{u}_e)$ for all $\mathbf{u}_e = \{--,-+,+-,++\}$. For graph $G^{(3)}$ where $e_2 \notin E^{(3)}$, we first impute corresponding marginal dual variable of e_2 on $G^{(3)}$according to local consistency constraints. Similar computations are required for edge e_1 and e_3. The finial prediction is through inference over averaged edge potentials on consensus graph G

$$F^{\text{MAM}}(x) = \underset{\mathbf{y}\in\mathcal{Y}}{\mathbf{argmax}} \sum_{e\in E}\frac{1}{T}\sum_{t=1}^{T}\psi^{(t)}_e(x,\mathbf{y}_e) = \underset{\mathbf{y}\in\mathcal{Y}}{\mathbf{argmax}}\ \frac{1}{T}\sum_{t=1}^{T}\sum_{e\in E}\langle \mathbf{w}^{(t)}_e, \varphi_e(x,\mathbf{y}_e)\rangle.$$

With the factorized dual representation, this ensemble scheme can be implemented simply and efficiently in terms of marginal dual variables and the associated kernels, which saves us from explicitly computing the local compatibility scores from each base learner. Using Lemma (1), the above can be equivalently expressed as

$$\begin{aligned} F^{\text{MAM}}(x) &= \underset{\mathbf{y}\in\mathcal{Y}}{\mathbf{argmax}}\ \frac{1}{T}\sum_{t=1}^{T}\sum_{i,e,\mathbf{u}_e}\mu^{(t)}(i,e,\mathbf{u}_e)\cdot H_e(i,\mathbf{u}_e;x,\mathbf{y}_e) \\ &= \underset{\mathbf{y}\in\mathcal{Y}}{\mathbf{argmax}}\sum_{i,e,\mathbf{u}_e}\bar{\mu}(i,e,\mathbf{u}_e)H_e(i,\mathbf{u}_e;x,\mathbf{y}_e), \end{aligned}$$

where we denote by $\bar{\mu}(i,e,\mathbf{u}_e) = \frac{1}{T}\sum_{t=1}^{T}\mu^{(t)}(i,e,\mathbf{u}_e)$ the marginal dual variable averaged over the ensemble.

We note that $\mu^{(t)}$ is originally defined on edge set $E^{(t)}$, $\mu^{(t)}$ from different random output graph are not mutually consistent. In practice, we first construct a consensus graph $G = (E,V)$ by pooling edge sets $E^{(t)}$, then complete $\mu^{(t)}$ on E where missing components are imputed via exploring local consistency conditions and solving constrained least square problem. Thus, the ensemble prediction can be computed in marginal dual form without explicit access to input features, and the only input needed from the different base models are the values of the marginal dual variables. An example that illustrates the MAM ensemble scheme is shown in Fig. 2.

3.4 The MAM ensemble analysis

Here, we present theoretical analysis of the improvement of the MAM ensemble over the mean of the base classifiers. The analysis follows the spirit of the single-label ensemble analysis by Brown and Kuncheva (2010), generalizing it to the multilabel MAM ensemble.

Assume there is a collection of T individual base learners, indexed by $t \in \{1, \cdots, T\}$, that output compatibility scores $\psi_e^{(t)}(x, \mathbf{u}_e)$ for all $t \in \{1, \ldots, T\}$, $e \in E^{(t)}$, and $\mathbf{u}_e \in \mathcal{Y}_e$. For the purposes of this analysis, we express the compatibility scores in terms of the nodes (microlabels) instead of the edges and their labels. We denote by

$$\psi_j(x, y_j) = \sum_{\substack{e=(j,j'),\\ e \in N(j)}} \mathbf{1}_{\{y_j = u_j\}} \frac{1}{2} \psi_e(x, \mathbf{u}_e)$$

the sum of compatibility scores of the set of edges $N(j)$ incident to node j with consistent label $\mathbf{y}_e = (y_j, y_{j'})$, $y_j = u_j$. Then, the compatibility score for the input and the multilabel in (1) can be alternatively expressed as

$$\psi(x, \mathbf{y}) = \sum_{e \in E} \psi_e(x, \mathbf{y}_e) = \sum_{j \in V} \psi_j(x, y_j).$$

The compatibility score from MAM ensemble can be similarly represented in terms of the nodes by

$$\psi^{\text{MAM}}(x, \mathbf{y}) = \frac{1}{T} \sum_t \psi^{(t)}(x, \mathbf{y}) = \sum_{e \in E} \bar{\psi}_e(x, \mathbf{y}_e) = \sum_{j \in V} \bar{\psi}_j(x, y_j),$$

where we have denoted $\bar{\psi}_j(x, y_j) = \frac{1}{T} \sum_t \psi_j^{(t)}(x, y_j)$ and $\bar{\psi}_e(x, \mathbf{y}_e) = \frac{1}{T} \sum_t \psi_e^{(t)}(x, \mathbf{y}_e)$.

Assume now the ground truth, the optimal compatibility score of an example and multilabel pair $(x, \mathbf{y})$, is given by $\psi^*(x, \mathbf{y}) = \sum_{j \in V} \psi_j^*(x, y_j)$. We study the reconstruction error of the compatibility score distribution, given by the squared distance of the estimated score distributions from the ensemble and the ground truth. The reconstruction error of the MAM ensemble can be expressed as

$$\Delta^R_{\text{MAM}}(x, \mathbf{y}) = \left(\psi^*(x, \mathbf{y}) - \psi^{\text{MAM}}(x, \mathbf{y})\right)^2,$$

and the average reconstruction error of the base learners can be expressed as

$$\Delta^R_I(x, \mathbf{y}) = \frac{1}{T} \sum_t \left(\psi^*(x, \mathbf{y}) - \psi^{(t)}(x, \mathbf{y})\right)^2.$$

We denote by $\Psi_j(x, y_j)$ a random variable of the compatibility scores obtained by the base learners and $\{\psi_j^{(1)}(x, y_j), \cdots, \psi_j^{(T)}(x, y_j)\}$ as a sample from its distribution. We have the following result:

Theorem 1 *The reconstruction error of compatibility score distribution given by MAM ensemble $\Delta^R_{MAM}(x, \mathbf{y})$ is guaranteed to be no greater than the average reconstruction error given by individual base learners $\Delta^R_I(x, \mathbf{y})$.*

In addition, the gap can be estimated as

$$\Delta^R_I(x, \mathbf{y}) - \Delta^R_{MAM}(x, \mathbf{y}) = \boldsymbol{Var}\left(\sum_{j \in V} \Psi_j(x, y_j)\right) \geq 0.$$

The variance can be further expanded as

$$\boldsymbol{Var}\left(\sum_{j\in V}\Psi_j(x,y_j)\right)=\underbrace{\sum_{j\in V}\boldsymbol{Var}(\Psi_j(x,y_j))}_{diversity}+\underbrace{\sum_{\substack{p,q\in V,\\ p\neq q}}\boldsymbol{Cov}(\Psi_p(x,y_p),\Psi_q(x,y_q))}_{coherence}.$$

Proof By expanding and simplifying the squares we get

$$\begin{aligned}
\Delta_I^R(x,\mathbf{y})-\Delta_{\text{MAM}}^R(x,\mathbf{y}) &= \frac{1}{T}\sum_t\left(\psi^*(x,\mathbf{y})-\psi^{(t)}(x,\mathbf{y})\right)^2-\left(\psi^*(x,\mathbf{y})-\psi^{\text{MAM}}(x,\mathbf{y})\right)^2\\
&= \frac{1}{T}\sum_t\left(\sum_{j\in V}\psi_j^*(x,y_j)-\sum_{j\in V}\psi_j^{(t)}(x,y_j)\right)^2\\
&\quad-\left(\sum_{j\in V}\psi_j^*(x,y_j)-\sum_{j\in V}\frac{1}{T}\sum_t\psi_j^{(t)}(x,y_j)\right)^2\\
&= \frac{1}{T}\sum_t\left(\sum_{j\in V}\psi_j^{(t)}(x,y_j)\right)^2-\left(\frac{1}{T}\sum_t\sum_{j\in V}\psi_j^{(t)}(x,y_j)\right)^2\\
&= \boldsymbol{Var}(\sum_{j\in V}\Psi_j(x,y_j))\\
&\geq 0.
\end{aligned}$$

The expression of variance can be further expanded as

$$\begin{aligned}
\boldsymbol{Var}\left(\sum_{j\in V}\Psi_j(x,y_j)\right) &= \sum_{p,q\in V}\boldsymbol{Cov}(\Psi_p(x,y_p),\Psi_q(x,y_q))\\
&= \sum_{j\in V}\boldsymbol{Var}(\Psi_j(x,y_j))+\sum_{\substack{p,q\in V,\\ p\neq q}}\boldsymbol{Cov}(\Psi_p(x,y_p),\Psi_q(x,y_q)).
\end{aligned}$$

The Theorem 1 states that the reconstruction error from MAM ensemble is guaranteed to be less than or equal to the average reconstruction error from the individuals. In particular, the improvement can be further addressed by two terms, namely *diversity* and *coherence*. The classifier diversity measures the variance of predictions from base learners independently on each single labels. It has been previously studied in single-label classifier ensemble context by Krogh and Vedelsby (1995). The diversity term prefers the variabilities of individuals that are learned from different perspectives. It is a well known factor to improve the ensemble performance. The coherence term, that is specific to the multilabel classifiers, indicates that the more the microlabel predictions vary together, the greater advantage multilabel ensemble gets over the base learners. This supports our intuitive understanding that microlabel correlations are keys to successful multilabel learning.

Table 1 Statistics of multilabel datasets used in our experiments. For *NCI60* and *Fingerprint* dataset where there is no explicit feature representation, the rows of kernel matrix is assumed as feature vector

Dataset	**Statistics**				
	Instances	Labels	Features	Cardinality	Density
Emotions	593	6	72	1.87	0.31
Yeast	2417	14	103	4.24	0.30
Scene	2407	6	294	1.07	0.18
Enron	1702	53	1001	3.36	0.06
Cal500	502	174	68	26.04	0.15
Fingerprint	490	286	490	49.10	0.17
NCI60	4547	60	4547	11.05	0.18
Medical	978	45	1449	1.14	0.03
Circle10	1000	10	3	8.54	0.85
Circle50	1000	50	3	35.63	0.71

4 Experiments

4.1 Datasets

We experiment on a collection of ten multilabel datasets from different domains, including chemical, biological, and text classification problems. The *NCI60* dataset contains 4547 drug candidates with their cancer inhibition potentials in 60 cell line targets. The *Fingerprint* dataset links 490 molecular mass spectra together to 286 molecular substructures used as prediction targets. Four text classification datasets[1] are also used in our experiment. In addition, two artificial *Circle* dataset are generated according to Bian et al. (2012) with different amount of labels. An overview of the datasets is shown in Table 1, where *cardinality* is defined as the average number of positive microlabels for each example

$$cardinality = \frac{1}{m}\sum_{i=1}^{m} |\{j|\mathbf{y}_{ij} = 1\}|,$$

and *density* is the average number of labels for each example divided by the size of label space defined as

$$density = \frac{cardinality}{k}.$$

4.2 Kernels

We use kernel methods to describe the similarity between complex data objects in some experiment datasets. We calculate linear kernel on datasets where instants are described by feature vectors. For text classification datasets, we first compute weighted features with term frequency inverse document frequency model (TF-IDF) (c.f. Rajaraman and Ullman 2011). TF-IDF weights reflect how important a word is to a document in a collection of corpus defined as the ratio between the word frequency in a document and the word frequency in the a collection of corpus. We compute linear kernel of the weighted features.

[1] Available at http://mulan.sourceforge.net/datasets.html.

As the input kernel of the *Fingerprint* dataset where we have for each instant a mass spectrometry (MS) data, we calculated quadratic kernel over the 'bag' of mass/charge peak intensities. As the input kernel of the *cancer* dataset where each object is described as a molecular graph, we used the hash fingerprint Tanimoto kernel (Ralaivola et al. 2005) that enumerates all linear fragments up to length n in a molecule x. A hash function assigns each fragment a hash value that determines its position in descriptor space $\phi(x)$. Given two binary bit vectors $\phi(x)$ and $\phi(y)$ as descriptors, Tanimoto kernel is defined as

$$K(x, y) = \frac{|I(\phi(x)) \cap I(\phi(y))|}{|I(\phi(x)) \cup I(\phi(y))|},$$

where $I(\phi(x))$ denotes the set of indices of 1-bits in $\phi(x)$.

In practice, some learning algorithms required kernelized input while others need feature representation of input data. Due to the intractability of using explicit features for complex data and in order to achieve a fair comparison, we take precomputed kernel matrix as rows of feature vectors for the learning algorithms that required input of feature vectors.

4.3 Obtaining random output graphs

The structure of the output graph is significant both in term of efficiency of learning and inference, and the predictive performance. We consider the following two approaches to generate random output graphs.

- In the random pair approach, one takes each vertex in turn, randomly draw another vertex and couples the two with an edge.
- In the random spanning tree approach, one first draws a random $k \times k$ weight matrix W with non-negative edge weights and then extracts a maximum weight spanning tree out of the matrix, using w_{ij} as the weight for edge connecting labels i and j.

The random pair approach generally produces a set of disconnected graphs, which may not let the base learner to fully benefit from complex multilabel dependencies. On the other hand, the learning of the base classifier is potentially made more efficient due to the graph simplicity. The random spanning tree approach connects all targets so that complex multilabel dependencies can be learned. Also, the tree structure facilitates efficient inference.

4.4 Compared classification methods

For comparison, we choose the following established classification methods form different perspectives towards multilabel classification, accounting for single-label and multilabel, as well as ensemble and standalone methods:

- **MMCRF** (Rousu et al. 2007) is used both as a standalone multilabel classifier and the base classifier in the ensembles. Individual MMCRF models are trained with two kinds of output graphs, random tree and random pair graph.
- **SVM** is a discriminative learning method that has become very popular over recent years, described in several textbooks (Cristianini and Shawe-Taylor 2000; Schölkopf and Smola 2001). For multilabel classification task, we split the multilabel task into a collection of single-label classification problems. Then we apply SVM on each single problem and compute the predictions. The drawback of SVM on multilabel classification task is the computation becomes infeasible as the number of the labels increases. Beside, this approach assumes independency between labels, it does not get any benefit from dependencies defined by complex structures of the label space. SVM serves as the single-label non-ensemble baseline learner.

- **MTL** is a multi-task feature learning methods developed in Argyriou et al. (2008), which is used as multilabel baseline learner. The underlying assumption of MTL is that the task specific functions are related such that they share a small subset of features.
- **Adaboost** is an ensemble method that has been extensively studied both empirically and theoretically since it was developed in Freund and Schapire (1997). The idea behind the model is that a distribution is assigned over data points. In each iteration, a weak hypothesis is calculated based on current distribution, and the distribution is updated according to the performance of the weak hypothesis. As a results, the difficult examples will receive more weight (probability mass) after the update, and will be emphasized by the base learner in the next round.
  In addition, Adaboost for multilabel classification using Hamming loss (AdaboostMH), is designed for incorporating multilabel learner into Adaboost framework (Schapire and Singer 1998). The only difference is the distribution is assigned to each example and microlabel pair and updated accordingly. In our study, we use real-valued decision tree with at most 100 leaves as base learner of AdaboostMH, and generate an ensemble with 180 weak hypothesises.
- **Bagging** (bootstrapping aggregation) was introduced in Breiman (1996) as an ensemble method of combining multiple weak leaners. It creates individual weak hypothesises for its ensemble by calling base learner repeatedly on the random subsets of the training set. The training set for the weak learner in each round is generated by randomly sampling with replacement. As a result, many original training examples may be repeated many times while others may be left out. In our experiment, we randomly select 40% of the data as input to SVM to compute a weak hypothesis, and repeat the process until we collect an ensemble of 180 weak hypotheses.

4.5 Parameter selection and evaluation measures

We first sample 10 % data uniform at random from each experimental dataset for the purpose of parameter selection. SVM, MMCRF and MAM ensemble each have a margin softness parameter C, which potentially needs to be tuned. We tested the value of parameter C from a set $\{0.01, 0.1, 0.5, 1, 5, 10\}$ based on tuning data, then keep the best ones for the following validation step. We also perform extensive selection on γ parameters in MTL model in the same range as margin softness parameters.

We observe that most of the multilabel datasets are highly biased with regards to multilabel density. Therefore, we use the following *stratified* 5*-fold cross validation* scheme in the experiments reported, such that we group examples in equivalence classes based on the number of positive labels they have. Each equivalence class is then randomly split into five local folds, after that the local folds are merged to create five global folds. The proposed procedure ensures that also the smaller classes have representations in all folds.

To quantitatively evaluate the performance of different classifiers, we adopt several performance measures. We report *multilabel accuracy* which counts the proportion of multilabel predictions that have all of the microlabels being correct, *microlabel accuracy* as the proportion of microlabel being correct, and *microlabel* F_1 *score* that is the harmonic mean of microlabel precision and recall $F_1 = 2 \cdot \frac{Pre \times Rec}{Pre + Rce}$.

4.6 Comparison of different ensemble approaches

We evaluate our proposed ensemble approaches by learning ensemble with 180 base learners. The learning curves as the size of ensemble on varying datasets are shown in Figs. 3, 4, and

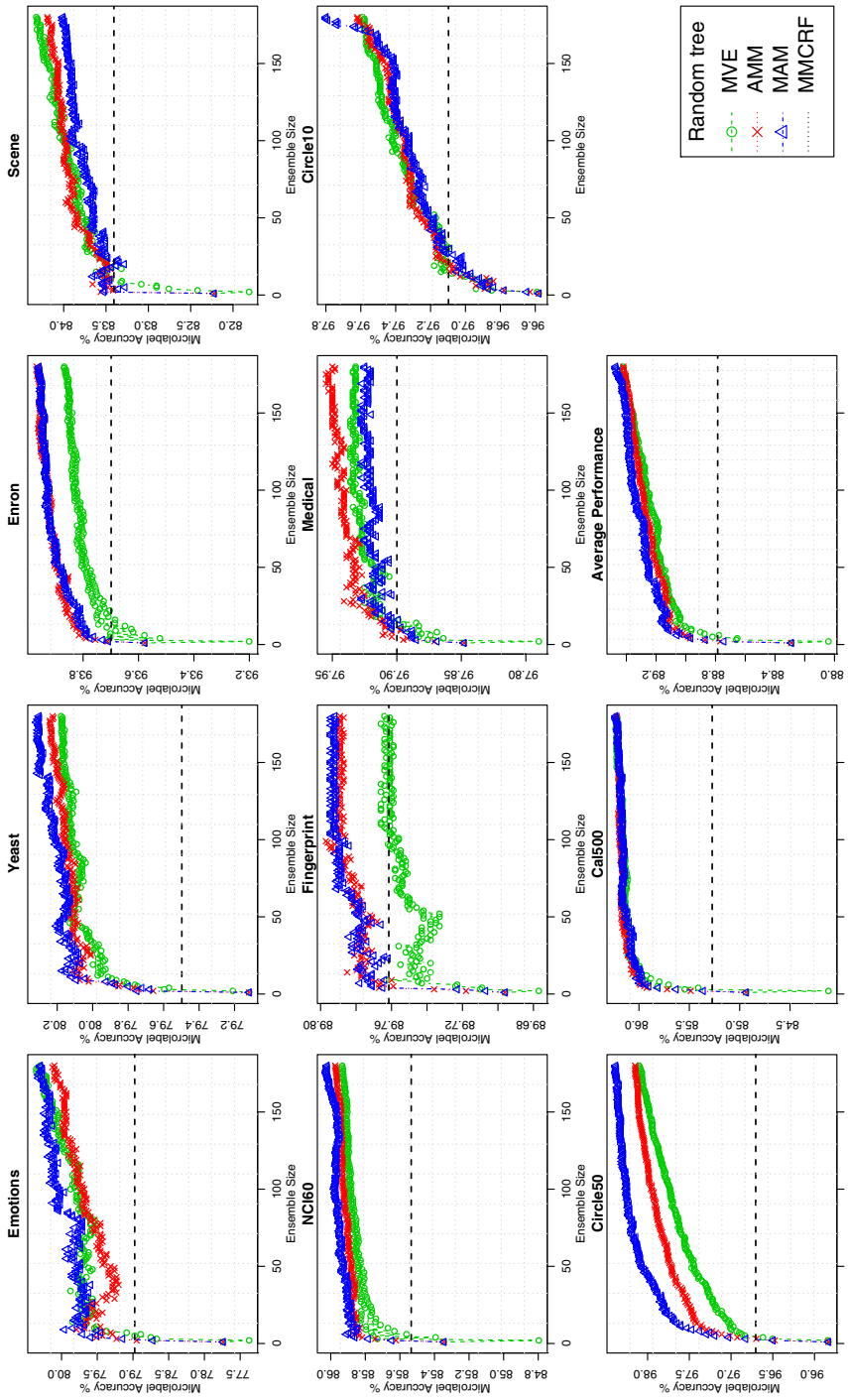


Fig. 3 Ensemble learning curve (microlabel accuracy) plotted as the size of ensemble. Average performance over datasets is shown as the last plot

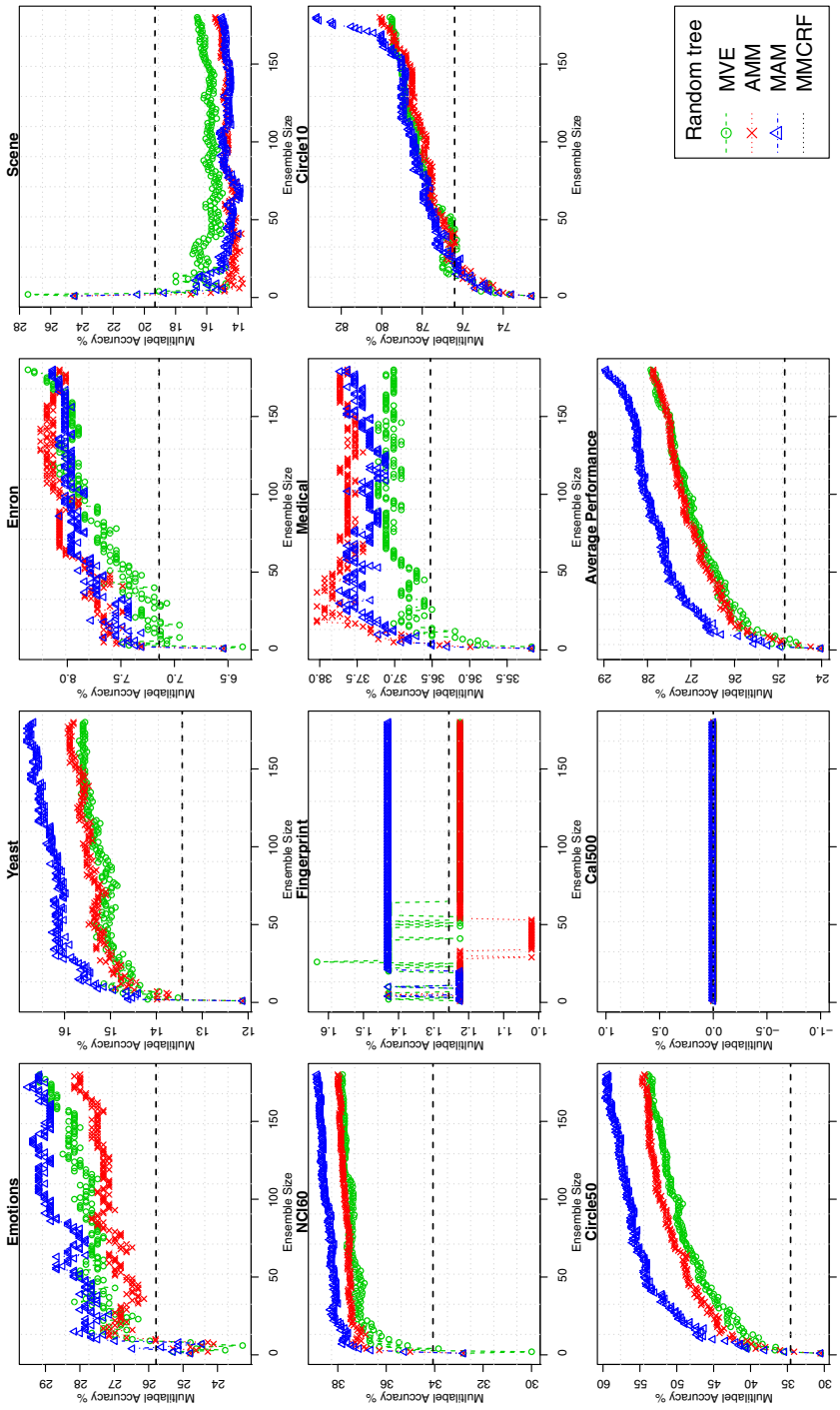


Fig. 4 Ensemble learning curve (multilabel accuracy) plotted as the size of ensemble. Average performance over datasets is shown as the last plot

5 for microlabel accuracy, multilabel accuracy, and microlabel F1 score, respectively. The base learners are trained with random tree as output graph structure.

There is a clear trend of improving microlabel accuracy for proposed ensemble approaches as more individual base models are combined. On most datasets and algorithms the ensemble accuracy increases fast and levels off rather quickly, the most obvious exception being the Circle10 dataset where improvement can be still seen beyond ensemble size 180. In addition, all three proposed ensemble learners (MVE, AMM, MAM) outperform their base learner MMCRF (horizontal dash lines) with consistent and noticeable margins, which is best seen from the learning curves of the average performance.

Similar patterns of learning curves are also observed in microlabel F_1 (Fig. 4) and multilabel accuracy (Fig. 5), with a few exceptions. The Fingerprint and Cal500 datasets prove to be difficult to learn in that very few multilabels are perfectly predicted, this is not surprising as these datasets have a large number of microlabels. The datasets also have the largest proportion of positive microlabels, which is reflected in the low F1 score. *Scene* dataset is the only exception where increasing the number of base learners seems to hurt the ensemble performance in microlabel F_1 and multilabel accuracy. In fact *Scene* is practically a single-label multiclass dataset, having very few examples with more than one positive microlabel. This contradicts the implicit assumption of graph based learners that there are rich dependency structures between different labels that could be revealed by the different random graphs. Among the extreme label sparsity, the ensemble learners appear to predict more negative labels for each example which leads to decreased performances in F_1 and multilabel accuracy space. We also observe large fluctuations in the initial part of MVE learning curves of *Fingerprint* and *Cal500* datasets in F_1 score space, implying MVE is not as stable as AMM and MAM approaches.

In particular, the performance of MAM ensemble surpasses MVE and AMM in eight out of ten datasets, the exceptions being *Scene* and *Medical*, making it the best among all proposed ensemble approaches. Consequently, we choose MAM for the further studies described in the following sections.

4.7 Effect of the structure of output graph

To find out which is the more beneficial output graph structure, we carry out empirical studies on MAM ensemble with random tree and random pair graph as output graph structure. Table 2 illustrates the performance of two output structures in terms of microlabel accuracy, multilabel accuracy and microlabel F_1 score. The results show that random tree and random pair graph are competitive output graph structures in terms of microlabel accuracy and F_1 score, with random tree achieves slightly better results. In addition, we observe noticeable difference in multilabel accuracy, where random tree behaves better than random pair graph. One way to understand this is to realize that random tree is able to connect all output labels so that learning and inference can work over the the whole label space. On the other hand, random pair approach divides the label space into isolated pairs where there is no cross-talk between pairs.

We continue by studying learning curves of average performance of MAM ensemble on two different output structures. Fig. 6 illustrates that MAM ensemble with random tree as output structure consistently outperforms random pair in accuracy space. The performance differences in F_1 space are not clear where we see the random pair approach fluctuating around random tree curve. Base on the experiments, we deem random tree the better of the two output graph structures.

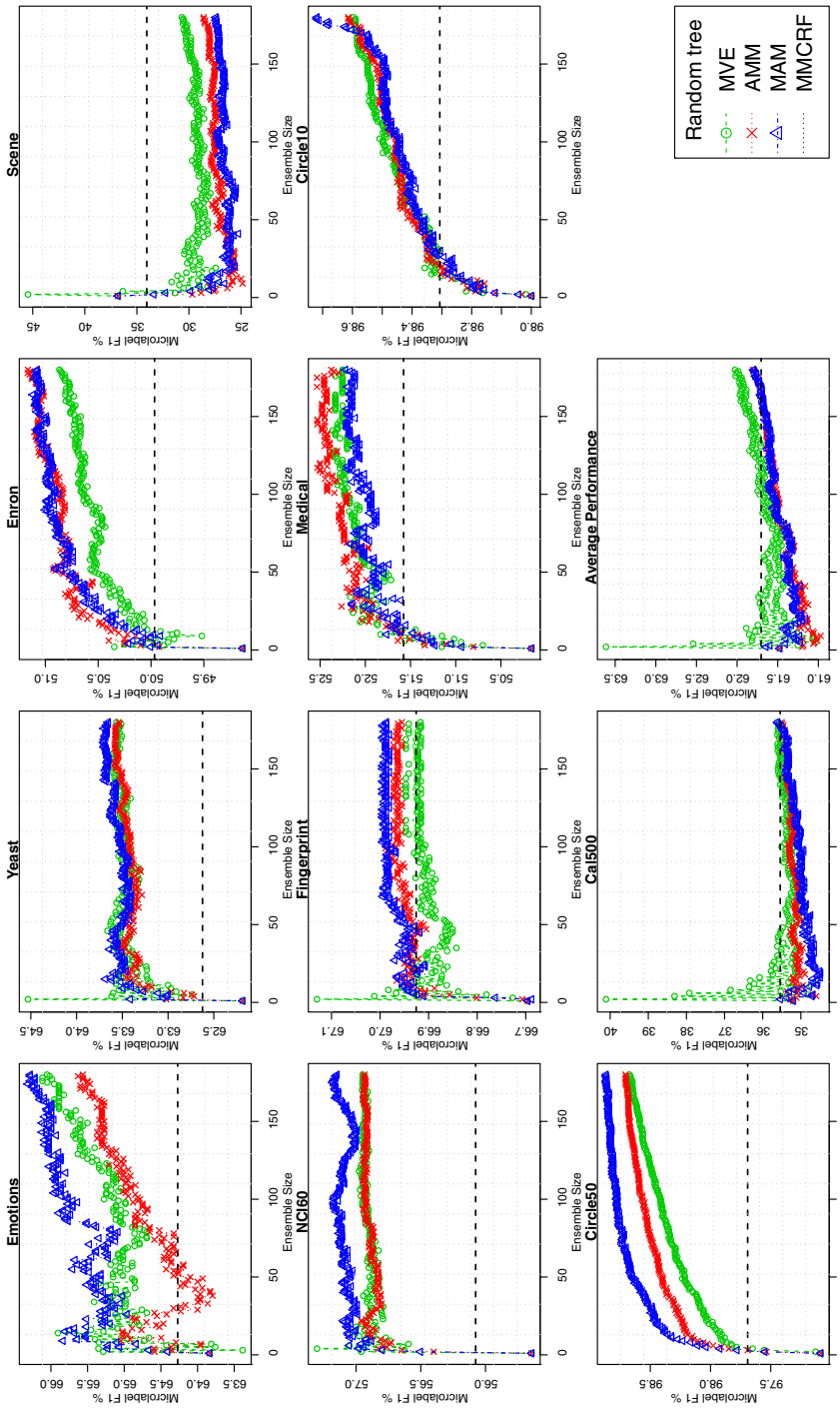


Fig. 5 Ensemble learning curve (microlabel F_1 score) plotted as the size of ensemble. Average performance over datasets is shown as the last plot

Table 2 Prediction performance of MAM ensemble with random tree and random pair graph in terms of microlabel accuracy, multilabel accuracy, and microlabel F_1 score

Dataset	Microlabel Acc %		Multilabel Acc %		Microlabel F_1 %	
	Pair	Tree	Pair	Tree	Pair	Tree
Emotions	**80.4** ± 2.4	80.3 ± 1.4	27.8 ± 3.4	**29.2** ± 4.2	65.7 ± 4.3	**66.3** ± 2.3
Yeast	80.2 ± 0.7	**80.3** ± 0.5	15.9 ± 1.1	**16.7** ± 0.4	63.5 ± 1.4	**63.7** ± 1.1
Scene	84.0 ± 0.5	**84.0** ± 0.1	**16.4** ± 1.9	15.0 ± 0.9	**28.9** ± 2.5	27.4 ± 2.4
Enron	**94.1** ± 0.1	94.0 ± 0.2	7.7 ± 1.0	**8.1** ± 2.3	51.1 ± 1.9	**51.1** ± 1.3
Cal500	**86.2** ± 0.1	86.2 ± 0.2	0.0 ± 0.0	0.0 ± 0.0	35.2 ± 0.8	**35.6** ± 0.4
Fingerprint	89.8 ± 0.5	**89.8** ± 0.3	1.2 ± 0.6	**1.4** ± 0.6	66.9 ± 2.5	**67.0** ± 1.9
NCI60	85.9 ± 0.8	**86.0** ± 0.9	37.9 ± 1.2	**38.9** ± 1.2	57.1 ± 3.8	**57.1** ± 3.2
Medical	97.9 ± 0.2	**97.9** ± 0.1	37.6 ± 4.3	**37.6** ± 2.5	52.2 ± 4.6	**52.2** ± 3.2
Circle10	97.5 ± 0.4	**97.8** ± 0.4	79.0 ± 2.0	**83.2** ± 3.5	98.5 ± 0.2	**98.7** ± 0.3
Circle50	97.6 ± 0.3	**98.4** ± 0.3	47.6 ± 5.9	**59.4** ± 5.6	98.3 ± 0.2	**98.9** ± 0.2

4.8 Multilabel prediction performance

In the following experiments we examine whether our proposed ensemble model (MAM) can boost the prediction performance in multilabel classification problems. Therefore, we compare our model with other advanced methods including both single-label and multilabel classifiers, both standalone and ensemble frameworks. Table 3 shows the performance of difference methods in terms of microlabel accuracy, multilabel accuracy and microlabel F_1 score, where the best performance in each dataset is emphasised in boldface and the second best is shown in italics.

We observe from Table 3 that MAM receives in general higher evaluation scores than the competitors. In particular, it achieves nine times as top two performing methods in microlabel accuracy, eight times in multilabel accuracy, and eight times in microlabel F_1 score. The only datasets where MAM is consistently outside the top two is the *Scene* dataset. As discussed above, the dataset is practically a single-label multiclass dataset. On this dataset the single target classifiers SVM and Bagging outperform all compared multilabel classifiers.

In these experiments, MMCRF also performs robustly, being in top two on half of the datasets with respect to microlabel and multilabel accuracy. However, it quite consistently trails to MAM in all three evaluation scores, the *Scene* dataset again being the exception. We also notice that the standalone single target classifier SVM is competitive against most multilabel methods, performs better than Bagging, AdaBoost and MTL with respect to microlabel and microlabel accuracy.

4.9 Statistical evaluation

To statistically evaluate the performance of different methods over multiple datasets, we first apply paired t-test on the values shown in Table 3. In particular, we compute a test statistic (with a p-value) for each ordered pair of methods to assess whether the average performance of the first is better than the second in a statistically significant manner. The result, shown in Table 4, indicates that, in terms of microlabel accuracy, MAM significantly outperforms MMCRF, AdaBoost and Bagging and almost significantly outperforms MTL, while the performance is not significantly different from SVM. In multilabel accuracy, MAM

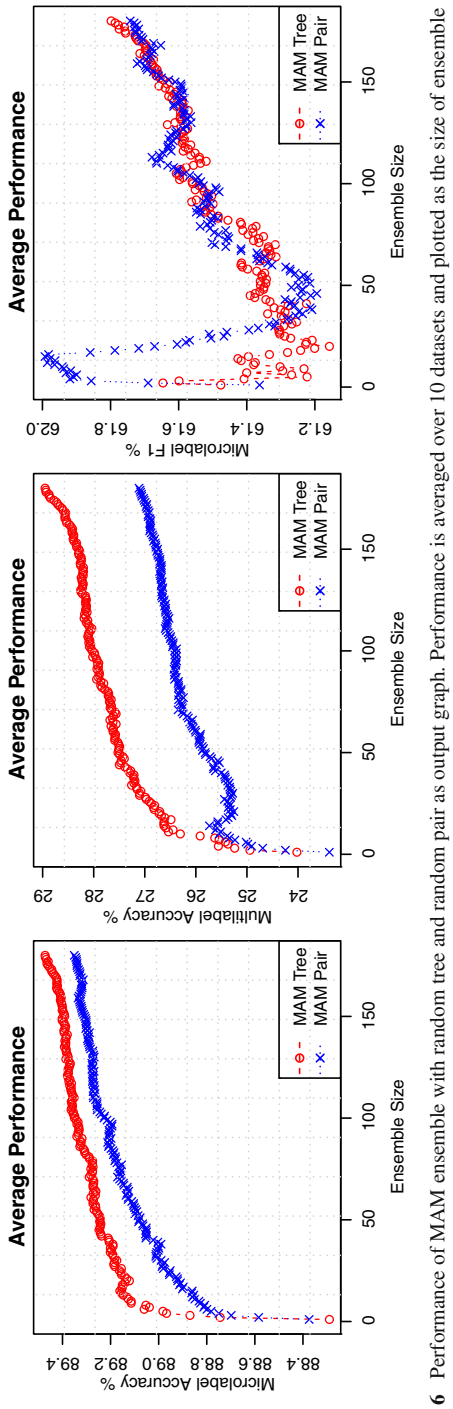


Fig. 6 Performance of MAM ensemble with random tree and random pair as output graph. Performance is averaged over 10 datasets and plotted as the size of ensemble

Table 3 Prediction performance of methods in terms of microlabel accuracy (top), microlabel F_1 score (middle), and multilabel accuracy (bottom). '–' represents no positive predictions. 'Avg. Rank' is the average rank of the performance over datasets

Dataset	SVM	Bagging	AdaBoost	MTL	MMCRF	MAM
Microlabel accuracy %						
Emotions	77.3±1.9	74.1±1.8	76.8±1.6	79.8±1.8	*79.0*±0.9	**80.3**±1.4
Yeast	*80.0±0.7*	78.4±0.9	74.8±0.7	79.3±0.5	79.5±0.6	**80.3**±0.5
Scene	**90.2**±0.3	87.8±0.5	84.3±0.9	*88.4±0.5*	83.4±0.3	84.0±0.1
Enron	93.6±0.2	*93.7±0.1*	86.2±0.3	93.5±0.2	93.7±0.2	**94.0**±0.2
Cal500	**86.3**±0.3	86.0±0.2	74.9±0.7	*86.2±0.3*	85.3±0.3	*86.2±0.2*
Fingerprint	89.7±0.3	85.0±0.4	84.1±0.7	82.7±0.6	**89.8**±0.6	**89.8**±0.3
NCI60	84.7±0.7	79.5±0.4	79.3±0.8	84.0±0.6	*85.5±1.3*	**86.0**±0.9
Medical	97.4±0.0	97.4±0.1	91.4±0.3	97.4±0.1	**97.9**±0.1	**97.9**±0.1
Circle10	94.8±0.9	92.9±0.7	**98.0**±0.3	93.7±0.7	97.1±0.3	*97.8±0.4*
Circle50	94.1±0.5	91.7±0.5	96.6±0.3	93.8±0.5	*96.7±0.3*	**98.4**±0.3
Avg. Rank	3.0	4.5[a]	4.8[a]	4.0[a]	3.0	**1.8**
Microlabel F_1 score %						
Emotions	57.1±4.4	61.5±3.1	*66.2±2.9*	64.6±3.0	64.3±1.2	**66.3**±2.3
Yeast	62.6±1.1	**65.5**±1.4	63.5±1.2	60.2±1.2	62.6±1.2	*63.7±1.1*
Scene	*68.3±1.4*	**69.9**±1.4	64.8±2.1	61.5±2.1	34.0±2.7	27.4±2.4
Enron	29.4±1.5	38.8±1.0	42.3±1.1	–	*50.0±1.0*	**51.1**±1.3
Cal500	31.4±0.6	*40.1±0.8*	**44.3**±1.5	28.6±1.3	35.5±0.4	35.6±0.4
Fingerprint	66.3±0.7	64.4±0.5	62.8±1.2	0.4±0.3	*66.9±0.8*	**67.0**±1.9
NCI60	45.9±3.6	53.9±1.2	32.9±2.7	32.9±3.4	*56.1±3.7*	**57.1**±3.2
Medical	–	–	33.7±1.2	–	*51.6±2.7*	**52.2**±3.2
Circle10	97.0±0.6	96.0±0.4	**98.8**±0.2	96.4±0.4	*98.3±0.2*	*98.7±0.3*
Circle50	96.0±0.3	94.5±0.3	97.6±0.2	95.7±0.3	*97.7±0.3*	**98.9**±0.2
Avg. Rank	4.2[a]	3.8[b]	3.0	5.2[a]	3.0	**1.9**
Multilabel accuracy %						
Emotions	21.2±3.4	20.9±2.6	23.8±2.3	25.5±3.5	*25.8±3.1*	**29.2**±4.2
Yeast	*14.0±2.8*	13.1±1.9	7.5±1.3	11.3±1.0	13.4±1.5	**16.7**±0.4
Scene	**52.8**±1.4	*46.5±1.9*	34.7±2.2	44.8±3.6	19.3±1.2	15.0±0.9
Enron	0.4±0.3	0.1±0.2	0.0±0.0	0.4±0.4	*7.1±2.8*	**8.1**±2.3
Cal500	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Fingerprint	1.0±0.7	0.0±0.0	0.0±0.0	0.0±0.0	*1.2±0.5*	**1.4**±0.6
NCI60	*43.1±1.3*	21.1±0.9	2.5±0.6	**47.0**±2.0	34.1±1.4	38.9±1.2
Medical	8.2±2.1	8.2±2.7	5.1±2.0	8.2±2.3	*36.5±3.3*	**37.6**±2.5
Circle10	69.1±3.8	64.8±3.3	**86.0**±2.7	66.8±3.4	76.4±2.1	*83.2±3.5*
Circle50	29.7±2.0	21.7±3.9	28.9±3.4	27.7±3.3	*34.6±4.5*	**59.4**±5.5
Avg. Rank	3.1	4.7[a]	4.5[a]	3.9[b]	2.9	**2.0**

The average rank is marked with [a] (resp. [b]) if the algorithm performs significantly different at p-value = 0.05 (resp. at p-value = 0.1) from the top performing one according to two-tailed Bonferroni–Dunn test

Table 4 Paired t-test to assess whether the method from group A outperforms the one from group B in a significant manner. By '†, ∗, ‡' we denote the performance in microlabel accuracy, microlabel F_1 score, and multilabel accuracy, respectively

Group A	Group B					
	SVM	Bagging	AdaBoost	MTL	MMCRF	MAM
SVM	–	††, ‡	††	∗	–	–
Bagging	∗∗	–	–	∗∗	–	–
Adaboost	–	–	–	∗∗	–	–
MTL	–	†	††	–	–	–
MMCRF	–	††	††	∗∗	–	–
MAM	–	††, ‡	††, ‡	†, ∗∗	††, ‡	–

By double marks (e.g. '††') we denote p-value = 0.05, and by single mark (e.g. '†') we denote p-value = 0.1. By '–' we denote not significant given above p-values

outperforms Bagging, Adaboost and MMCRF in almost significant manner. SVM, MTL and MMCRF perform similarly to each other, and are better than Bagging and Adaboost with respect to microlabel accuracy. In addition, in microlabel F_1 score, we notice that all methods are competitive against MTL, and SVM performs better than Bagging.

As suggested by Demšar (2006), several critical assumptions might be violated when performing paired t-test to compare classifiers over multiple datasets. Similarly, other commonly used statistical tests might also be ill-posed in this scope (e.g. sign test, Wilcoxon signed-ranks test, ANOVA with post-hoc Tukey test). We therefore follow the test procedure proposed in Demšar (2006). First, we compute the rank of each model based on the performance on different datasets, where the best performing algorithm getting the rank of one. In case of ties, averaged ranks are then assigned. Then we use Friedman test (Friedman 1937) which compares the average rank of each algorithm, and under null hypothesis, states that all algorithms are equivalent with equal average ranks. P-values calculated from Friedman test for microlabel accuracy, microlabel F_1 score, and multilabel accuracy are 0.001, 0.002 and 0.005, respectively. As a result, we reject the null-hypothesis and proceed with post-hoc two-tailed Bonferroni-Dunn test (Dunn 1961), where all other methods are compared against the top performing control (MAM). We compute the *critical difference* $CD = 2.2$ at p-value = 0.05, and $CD = 1.9$ at p-value = 0.1 (see details in supplementary material). The performance of an algorithm is significantly different from the control if the corresponding average ranks differ by at least CD. The corresponding rank is marked with 'b' (at p-value = 0.1) or 'a' (at p-value = 0.05) in Table 3. We observe from the results that in microlabel accuracy and multilabel accuracy, the performance differences of SVM and MMCRF to MAM fail to be statistically significant. On the other hand, Bagging, Adaboost and MTL perform significantly worse than MAM in terms of microlabel accuracy and multilabel accuracy. In addition, with respect to microlabel F_1 score, the performances of MMCRF and Adaboost are not significantly different from MAM, while SVM, Bagging and MTL perform worse than MAM in a significant manner.

Overall, the results indicate that ensemble by MAM is a robust and competitive alternative for multilabel classification.

4.10 Effect of diversity and coherence

To explain the performance of MAM as well as to empirically validate the diversity and coherence arguments stated in Theorem 1, we conduct the following experiment.

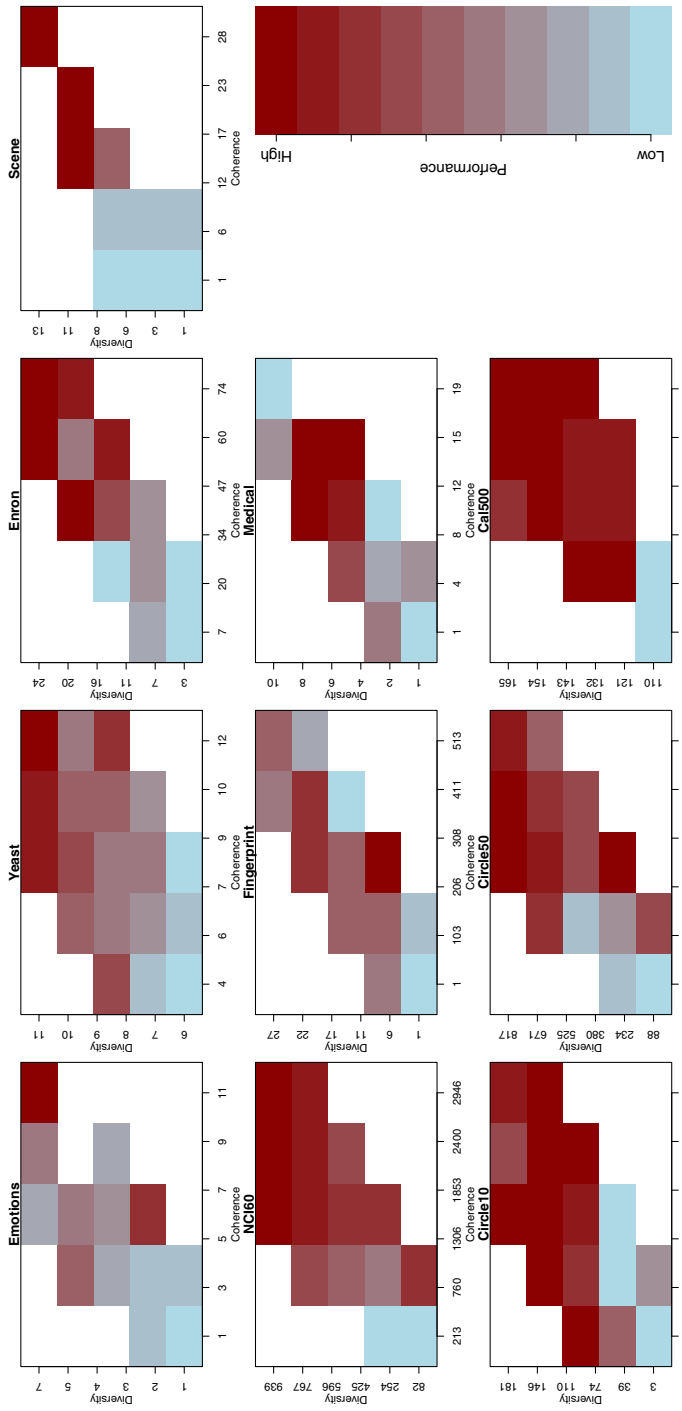


Fig. 7 Performance of MAM ensemble plotted in diversity and coherence space. Color of the blocks depicts average performance in term of microlabel accuracy computed from the data points in the block. White area means there is no examples with corresponding diversity and coherence. Colors are normalized for datasets so that worst and best performances are shown as light blue and red

We train a MAM ensemble model for each dataset consist of 30 base learners with a random spanning tree as output graph structure. For each example-label pair $(x_i, \mathbf{y}_i)$ and the corresponding set of microlabels $\mathbf{y}_i = \{y_{i,1}, \cdots, y_{i,l}\}$, we then calculate from each base learner t a set of node compatibility scores $\{\psi^t(x_i, y_{i,1}), \cdots, \psi^t(x_i, y_{i,l})\}$. Next, the node compatibility scores from different base learners are pooled together to get $\Psi_j(x_i, y_{i,j}) = \{\psi^1(x_i, y_{i,j}), \cdots, \psi^{30}(x_i, y_{i,j})\}$ for all $j \in \{1, \cdots, l\}$. Diversity and coherence of pair $(x_i, \mathbf{y}_i)$ can be calculated from $\{\Psi_j(x_i, y_{i,j})\}_{j=1}^{l}$ according to

$$Diversity = \sum_{j \in \{1 \cdots l\}} \boldsymbol{Var}(\Psi_j(x_i, y_{ij})),$$

$$Coherence = \sum_{\substack{p,q \in \{1 \cdots l\}, \\ p \neq q}} \boldsymbol{Cov}(\Psi_p(x_i, y_{ip}), \Psi_q(x_i, y_{iq})),$$

which locates pair $(x_i, \mathbf{y}_i)$ in the diversity-coherence space. We also compute the microlabel accuracy from the microlabels in $\mathbf{y}_i$ based on the prediction from MAM ensemble. The accuracy of different diversity-coherence region in the space is computed as the average microlabel accuracy of examples in that region. The results are shown in Fig. 7.

We observe from the results a pattern of increasing prediction performance from lower left corner to upper right corner. In particular, microlabel accuracy are lower for examples with both low diversity and coherence computed based on current set of base learners, shown as the light blue blocks in lower left corner. On the other hand, we achieve higher prediction accuracy on examples with high diversity and coherence, which are shown as red blocks in the upper right corner. In addition, fixing one factor while increasing the other usually leads to improved performance.

The observations demonstrates both diversity and coherence have positive effects on the performance of MAM ensemble. They reflect different aspects of the ensemble. To improve the quality of the prediction, one should aim to increase either the diversity of the base learner on a single microlabel or the coherence among microlabel pairs.

5 Conclusions

In this paper we have put forward new methods for multilabel classification, relying on ensemble learning on random output graphs. In our experiments, models thus created have favourable predictive performances on a heterogeneous collection of multilabel datasets, compared to several established methods. The theoretical analysis of the MAM ensemble highlights the covariance of the compatibility scores between the inputs and microlabels learned by the base learners as the quantity explaining the advantage of the ensemble prediction over the base learners.

We note in passing that it is straightforward to generalize the theoretical analysis to any multilabel classifiers that give scores to microlabels; there is no dependency on random graph classifiers in the analysis.

The empirical evaluation supports the theoretical analysis, explaining the performance of the proposed ensemble models. Our results indicate that structured output prediction methods can be successfully applied to problems where no prior known output structure exists, and thus widen the applicability of the structured output prediction.

Acknowledgments The work was financially supported by Helsinki Doctoral Programme in Computer Science (Hecse), Academy of Finland grant 118653 (ALGODAN), IST Programme of the European Community under the PASCAL2 Network of Excellence, ICT-2007-216886. This publication only reflects the authors' views.

References

Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, *73*(3), 243–272.

Bian, W., Xie, B., & Tao, D. (2012). Corrlog: Correlated logistic models for joint prediction of multiple labels. In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, vol. 22, pp. 109–117.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Brown, G., & Kuncheva, LI. (2010). Good and bad diversity in majority vote ensembles. In: *Multiple classifier systems* (pp. 124–133). Berlin: Springer.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods* (1st ed.). Cambridge: Cambridge University Press.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30.

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, *56*(293), 52–64.

Esuli, A., Fagni, T., & Sebastiani, F. (2008). Boosting multi-label hierarchical text categorization. *Information Retrieval*, *11*(4), 287–313.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*(200), 675–701.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. In: *Advances in neural information processing systems* (pp. 231–238). Cambridge, MA: MIT Press.

Rajaraman, A., & Ullman, J. (2011). *Mining of massive datasets*. Cambridge: Cambridge University Press.

Ralaivola, L., Swamidass, S., Saigo, H., & Baldi, P. (2005). Graph kernels for chemical informatics. *Neural Networks*, *18*, 1093–1110.

Rousu, J., Saunders, C., Szedmak, S., & Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *The Journal of Machine Learning Research*, *7*, 1601–1626.

Rousu, J., Saunders, C., Szedmak, S., & Shawe-Taylor, J. (2007). Efficient algorithms for max-margin structured classification. In *Predicting structured data*, pp. 105–129.

Schapire, R., & Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. In: *Proceedings of the Annual Conference on Computational Learning Theory* (pp. 80–91). New York: ACM Press.

Schapire, R. E., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, *39*(2/3), 135–168.

Schölkopf, B., & Smola, A. (2001). *Learning with Kernels*. Cambridge, MA: MIT Press.

Su, H., & Rousu, J. (2001). Multi-task drug bioactivity classification with graph labeling ensembles. In *Proceedings of the 6th International Conference on Pattern Recognition in Bioinformatics (PRIB2011), Lecture Note in Computer Science (LNCS)*, (Vol. 7035, pp.157–167).

Su, H., & Rousu, J. (2013). Multilabel classification through random graph ensembles. In: *Proceedings, 5th Asian conference on machine learning (ACML2013), Journal of Machine Learning Research W&CP*, vol. 29, pp. 404–418.

Taskar, B., Guestrin, G., & Koller, D. (2004). Max-Margin Markov networks. In S. Thrun, L. K. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems*, (Vol. 16, pp. 25–32). MIT Press.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In: *Proceedings of the twenty-first international conference on machine learning* ICML'04, pp. 823–830.

Wainwright, M., Jaakkola, T., & Willsky, A. (2005). MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, *51*(11), 3697–3717.

Yan, R., Tesic, J., & Smith, J. (2007). Model-shared subspace boosting for multi-label classification. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 834–843). ACM

Publication V

Mario Marchand, Hongyu Su, Emilie Morvant, Juho Rousu, John Shawe-Taylor. Multilabel Structured Output Learning with Random Spanning Trees of Max-Margin Markov Networks. In *Advances in Neural Information Processing Systems 26 (NIPS 2014)*, Accepted 9 Pages, December 2014.

Multilabel Structured Output Learning with Random Spanning Trees of Max-Margin Markov Networks

Mario Marchand
Département d'informatique et génie logiciel
Université Laval
Québec (QC), Canada
mario.marchand@ift.ulaval.ca

Hongyu Su
Helsinki Institute for Information Technology
Dept of Information and Computer Science
Aalto University, Finland
hongyu.su@aalto.fi

Emilie Morvant*
LaHC, UMR CNRS 5516
Univ. of St-Etienne, France
emilie.morvant@univ-st-etienne.fr

Juho Rousu
Helsinki Institute for Information Technology
Dept of Information and Computer Science
Aalto University, Finland
juho.rousu@aalto.fi

John Shawe-Taylor
Department of Computer Science
University College London
London, UK
j.shawe-taylor@ucl.ac.uk

Abstract

We show that the usual score function for conditional Markov networks can be written as the expectation over the scores of their spanning trees. We also show that a small random sample of these output trees can attain a significant fraction of the margin obtained by the complete graph and we provide conditions under which we can perform tractable inference. The experimental results confirm that practical learning is scalable to realistic datasets using this approach.

1 Introduction

Finding an hyperplane that minimizes the number of misclassifications is $\mathcal{NP}$-hard. But the support vector machine (SVM) substitutes the hinge for the discrete loss and, modulo a margin assumption, can nonetheless efficiently find a hyperplane with a guarantee of good generalization. This paper investigates whether the problem of inference over a complete graph in structured output prediction can be avoided in an analogous way based on a margin assumption.

We first show that the score function for the complete output graph can be expressed as the expectation over the scores of random spanning trees. A sampling result then shows that a small random sample of these output trees can attain a significant fraction of the margin obtained by the complete graph. Together with a generalization bound for the sample of trees, this shows that we can obtain good generalization using the average scores of a sample of trees in place of the complete graph. We have thus reduced the intractable inference problem to a convex optimization not dissimilar to a SVM. The key inference problem to enable learning with this ensemble now becomes finding the maximum violator for the (finite sample) average tree score. We then provide the conditions under which the inference problem is tractable. Experimental results confirm this prediction and show that

*Most of this work was carried out while E. Morvant was affiliated with IST Austria, Klosterneurburg.

practical learning is scalable to realistic datasets using this approach with the resulting classification accuracy enhanced over more naive ways of training the individual tree score functions.

The paper aims at exploring the potential ramifications of the random spanning tree observation both theoretically and practically. As such, we think that we have laid the foundations for a fruitful approach to tackle the intractability of inference in a number of scenarios. Other attractive features are that we do not require knowledge of the output graph's structure, that the optimization is convex, and that the accuracy of the optimization can be traded against computation. Our approach is firmly rooted in the maximum margin Markov network analysis [1]. Other ways to address the intractability of loopy graph inference have included using approximate MAP inference with tree-based and LP relaxations [2], semi-definite programming convex relaxations [3], special cases of graph classes for which inference is efficient [4], use of random tree score functions in heuristic combinations [5]. Our work is not based on any of these approaches, despite superficial resemblances to, *e.g.,* the trees in tree-based relaxations and the use of random trees in [5]. We believe it represents a distinct approach to a fundamental problem of learning and, as such, is worthy of further investigation.

2 Definitions and Assumptions

We consider supervised learning problems where the input space $\mathcal{X}$ is arbitrary and the output space $\mathcal{Y}$ consists of the set of all ℓ-dimensional multilabel vectors $(y_1, \ldots, y_\ell) \stackrel{\text{def}}{=} \mathbf{y}$ where each $y_i \in \{1, \ldots, r_i\}$ for some finite positive integer r_i. Each example $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ is mapped to a joint feature vector $\boldsymbol{\phi}(x, \mathbf{y})$. Given a weight vector $\mathbf{w}$ in the space of joint feature vectors, the predicted output $\mathbf{y}_{\mathbf{w}}(x)$ at input $x \in \mathcal{X}$, is given by the output $\mathbf{y}$ maximizing the *score* $F(\mathbf{w}, x, \mathbf{y})$, *i.e.*,

$$\mathbf{y}_{\mathbf{w}}(x) \stackrel{\text{def}}{=} \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{w}, x, \mathbf{y}) \quad ; \quad \text{where} \quad F(\mathbf{w}, x, \mathbf{y}) \stackrel{\text{def}}{=} \langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y}) \rangle \,, \tag{1}$$

and where $\langle \cdot, \cdot \rangle$ denotes the inner product in the joint feature space. Hence, $\mathbf{y}_{\mathbf{w}}(x)$ is obtained by solving the so-called *inference* problem, which is known to be $\mathcal{NP}$-hard for many output feature maps [6, 7]. Consequently, we aim at using an output feature map for which the inference problem can be solved by a polynomial time algorithm such as dynamic programming. The *margin* $\Gamma(\mathbf{w}, x, \mathbf{y})$ achieved by predictor $\mathbf{w}$ at example $(x, \mathbf{y})$ is defined as,

$$\Gamma(\mathbf{w}, x, \mathbf{y}) \stackrel{\text{def}}{=} \min_{\mathbf{y}' \neq \mathbf{y}} \left[F(\mathbf{w}, x, \mathbf{y}) - F(\mathbf{w}, x, \mathbf{y}') \right] .$$

We consider the case where the feature map $\boldsymbol{\phi}$ is a potential function for a Markov network defined by a complete graph G with ℓ nodes and $\ell(\ell-1)/2$ undirected edges. Each node i of G represents an output variable y_i and there exists an edge (i, j) of G for each pair (y_i, y_j) of output variables. For any example $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, its joint feature vector is given by

$$\boldsymbol{\phi}(x, \mathbf{y}) = \big(\boldsymbol{\phi}_{i,j}(x, y_i, y_j)\big)_{(i,j) \in G} = \big(\boldsymbol{\varphi}(x) \otimes \boldsymbol{\psi}_{i,j}(y_i, y_j)\big)_{(i,j) \in G} \,,$$

where $\otimes$ is the Kronecker product. Hence, any predictor $\mathbf{w}$ can be written as $\mathbf{w} = (\mathbf{w}_{i,j})_{(i,j) \in G}$ where $\mathbf{w}_{i,j}$ is $\mathbf{w}$'s weight on $\boldsymbol{\phi}_{i,j}(x, y_i, y_j)$. Therefore, for any $\mathbf{w}$ and any $(x, \mathbf{y})$, we have

$$F(\mathbf{w}, x, \mathbf{y}) \; = \langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y}) \rangle = \sum_{(i,j) \in G} \langle \mathbf{w}_{i,j}, \boldsymbol{\phi}_{i,j}(x, y_i, y_j) \rangle \; = \sum_{(i,j) \in G} F_{i,j}(\mathbf{w}_{i,j}, x, y_i, y_j) \,,$$

where we denote by $F_{i,j}(\mathbf{w}_{i,j}, x, y_i, y_j) = \langle \mathbf{w}_{i,j}, \boldsymbol{\phi}_{i,j}(x, y_i, y_j)$ the score of labeling the edge (i, j) by (y_i, y_j) given input x.

For any vector $\mathbf{a}$, let $\|\mathbf{a}\|$ denote its L_2 norm. Throughout the paper, we make the assumption that we have a normalized joint feature space such that $\|\boldsymbol{\phi}(x, \mathbf{y})\| = 1$ for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ and $\|\boldsymbol{\phi}_{i,j}(x, y_i, y_j)\|$ is the same for all $(i, j) \in G$. Since the complete graph G has $\binom{\ell}{2}$ edges, it follows that $\|\boldsymbol{\phi}_{i,j}(x, y_i, y_j)\|^2 = \binom{\ell}{2}^{-1}$ for all $(i, j) \in G$.

We also have a training set $S \stackrel{\text{def}}{=} \{(x_1, \mathbf{y}_1), \ldots, (x_m, \mathbf{y}_m)\}$ where each example is generated independently according to some unknown distribution D. Mathematically, we do not assume the existence of a predictor $\mathbf{w}$ achieving some positive margin $\Gamma(\mathbf{w}, x, \mathbf{y})$ on each $(x, \mathbf{y}) \in S$. Indeed,

for some S, there might not exist any $\mathbf{w}$ where $\Gamma(\mathbf{w}, x, \mathbf{y}) > 0$ for all $(x, \mathbf{y}) \in S$. However, the generalization guarantee will be best when $\mathbf{w}$ achieves a large margin on most training points.

Given any $\gamma > 0$, and any $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, the *hinge loss* (at scale γ) incurred on $(x, \mathbf{y})$ by a unit L_2 norm predictor $\mathbf{w}$ that achieves a (possibly negative) margin $\Gamma(\mathbf{w}, x, \mathbf{y})$ is given by $\mathcal{L}^\gamma(\Gamma(\mathbf{w}, x, \mathbf{y}))$, where the so-called *hinge loss function* $\mathcal{L}^\gamma$ is defined as $\mathcal{L}^\gamma(s) \stackrel{\text{def}}{=} \max(0, 1 - s/\gamma) \ \forall s \in \mathbb{R}$. We will also make use of the *ramp loss function* $\mathcal{A}^\gamma$ defined by $\mathcal{A}^\gamma(s) \stackrel{\text{def}}{=} \min(1, \mathcal{L}^\gamma(s)) \ \forall s \in \mathbb{R}$.

The proofs of all the rigorous results of this paper are provided in the supplementary material.

3 Superposition of Random Spanning Trees

Given a complete graph G of ℓ nodes (representing the Markov network), let $S(G)$ denote the set of all $\ell^{\ell-2}$ spanning trees of G. Recall that each spanning tree of G has $\ell - 1$ edges. Hence, for any edge $(i, j) \in G$, the number of trees in $S(G)$ covering that edge (i, j) is given by $\ell^{\ell-2}(\ell-1)/\binom{\ell}{2} = (2/\ell)\ell^{\ell-2}$. Therefore, for any function f of the edges of G we have

$$\sum_{T \in S(G)} \sum_{(i,j) \in T} f((i,j)) = \ell^{\ell-2} \frac{2}{\ell} \sum_{(i,j) \in G} f((i,j)) \ .$$

Given any spanning tree T of G and given any predictor $\mathbf{w}$, let $\mathbf{w}_T$ denote the projection of $\mathbf{w}$ on the edges of T. Namely, $(\mathbf{w}_T)_{i,j} = \mathbf{w}_{i,j}$ if $(i, j) \in T$, and $(\mathbf{w}_T)_{i,j} = 0$ otherwise. Let us also denote by $\boldsymbol{\phi}_T(x, \mathbf{y})$, the projection of $\boldsymbol{\phi}(x, \mathbf{y})$ on the edges of T. Namely, $(\boldsymbol{\phi}_T(x, \mathbf{y}))_{i,j} = \boldsymbol{\phi}_{i,j}(x, y_i, y_j)$ if $(i, j) \in T$, and $(\boldsymbol{\phi}_T(x, \mathbf{y}))_{i,j} = 0$ otherwise. Recall that $\|\boldsymbol{\phi}_{i,j}(x, y_i, y_j)\|^2 = \binom{\ell}{2}^{-1} \ \forall (i, j) \in G$. Thus, for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ and for all $T \in S(G)$, we have

$$\|\boldsymbol{\phi}_T(x, \mathbf{y})\|^2 = \sum_{(i,j) \in T} \|\boldsymbol{\phi}_{i,j}(x, y_i, y_j)\|^2 = \frac{\ell - 1}{\binom{\ell}{2}} = \frac{2}{\ell} \, .$$

We now establish how $F(\mathbf{w}, x, \mathbf{y})$ can be written as an expectation over all the spanning trees of G.

Lemma 1. *Let* $\hat{\mathbf{w}}_T \stackrel{def}{=} \mathbf{w}_T / \|\mathbf{w}_T\|$, $\hat{\boldsymbol{\phi}}_T \stackrel{def}{=} \boldsymbol{\phi}_T / \|\boldsymbol{\phi}_T\|$. *Let* $\mathcal{U}(G)$ *denote the uniform distribution on* $S(G)$. *Then, we have*

$$F(\mathbf{w}, x, \mathbf{y}) \ = \ \mathop{\mathbf{E}}_{T \sim \mathcal{U}(G)} a_T \langle \hat{\mathbf{w}}_T, \hat{\boldsymbol{\phi}}_T(x, \mathbf{y}) \rangle, \quad \textit{where } a_T \stackrel{def}{=} \sqrt{\frac{\ell}{2}} \, \|\mathbf{w}_T\| \, .$$

Moreover, for any $\mathbf{w}$ *such that* $\|\mathbf{w}\| = 1$, *we have:* $\mathop{\mathbf{E}}_{T \sim \mathcal{U}(G)} a_T^2 \ = \ 1$, *and* $\mathop{\mathbf{E}}_{T \sim \mathcal{U}(G)} a_T \ \leq \ 1 \, .$

Let $\mathcal{T} \stackrel{\text{def}}{=} \{T_1, \ldots, T_n\}$ be a sample of n spanning trees of G where each T_i is sampled independently according to $\mathcal{U}(G)$. Given any unit L_2 norm predictor $\mathbf{w}$ on the complete graph G, our task is to investigate how the margins $\Gamma(\mathbf{w}, x, \mathbf{y})$, for each $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, will be modified if we approximate the (true) expectation over all spanning trees by an average over the sample $\mathcal{T}$.

For this task, we consider any $(x, \mathbf{y})$ and any $\mathbf{w}$ of unit L_2 norm. Let $F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y})$ denote the estimation of $F(\mathbf{w}, x, \mathbf{y})$ on the tree sample $\mathcal{T}$,

$$F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) \ \stackrel{\text{def}}{=} \ \frac{1}{n} \sum_{i=1}^{n} a_{T_i} \langle \hat{\mathbf{w}}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x, \mathbf{y}) \rangle \, ,$$

and let $\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y})$ denote the estimation of $\Gamma(\mathbf{w}, x, \mathbf{y})$ on the tree sample $\mathcal{T}$,

$$\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) \stackrel{\text{def}}{=} \min_{\mathbf{y}' \neq \mathbf{y}} \left[F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) - F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}') \right] \, .$$

The following lemma states how $\Gamma_{\mathcal{T}}$ relates to Γ.

Lemma 2. *Consider any unit* L_2 *norm predictor* $\mathbf{w}$ *on the complete graph* G *that achieves a margin of* $\Gamma(\mathbf{w}, x, \mathbf{y})$ *for each* $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, *then we have*

$$\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) \ \geq \ \Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon \quad \forall (x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y} \, ,$$

whenever we have $|F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) - F(\mathbf{w}, x, \mathbf{y})| \leq \epsilon$ *for all* $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$.

Lemma 2 has important consequences whenever $|F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) - F(\mathbf{w}, x, \mathbf{y})| \leq \epsilon$ for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. Indeed, if $\mathbf{w}$ achieves a hard margin $\Gamma(\mathbf{w}, x, \mathbf{y}) \geq \gamma > 0$ for all $(x, \mathbf{y}) \in S$, then we have that $\mathbf{w}$ also achieves a hard margin of $\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) \geq \gamma - 2\epsilon$ on each $(x, \mathbf{y}) \in S$ when using the tree sample $\mathcal{T}$ instead of the full graph G. More generally, if $\mathbf{w}$ achieves a ramp loss of $\mathcal{A}^{\gamma}(\Gamma(\mathbf{w}, x, \mathbf{y}))$ for each $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, then $\mathbf{w}$ achieves a ramp loss of $\mathcal{A}^{\gamma}(\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y})) \leq \mathcal{A}^{\gamma}(\Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon)$ for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ when using the tree sample $\mathcal{T}$ instead of the full graph G. This last property follows directly from the fact that $\mathcal{A}^{\gamma}(s)$ is a non-increasing function of s.

The next lemma tells us that, apart from a slow $\ln^2(\sqrt{n})$ dependence, a sample of $n \in \Theta(\ell^2/\epsilon^2)$ spanning trees is sufficient to assure that the condition of Lemma 2 holds with high probability for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. Such a fast convergence rate was made possible by using PAC-Bayesian methods which, in our case, prevented us of using the union bound over all possible $\mathbf{y} \in \mathcal{Y}$.

Lemma 3. *Consider any $\epsilon > 0$ and any unit L_2 norm predictor $\mathbf{w}$ for the complete graph G acting on a normalized joint feature space. For any $\delta \in (0, 1)$, let*

$$n \;\geq\; \frac{\ell^2}{\epsilon^2}\left(\frac{1}{16} + \frac{1}{2}\ln\frac{8\sqrt{n}}{\delta}\right)^2 . \tag{2}$$

Then with probability of at least $1 - \delta/2$ over all samples $\mathcal{T}$ generated according to $\mathcal{U}(G)^n$, we have, simultaneously for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, that $|F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) - F(\mathbf{w}, x, \mathbf{y})| \;\leq \epsilon$.

Given a sample $\mathcal{T}$ of n spanning trees of G, we now consider an arbitrary set $\mathcal{W} \stackrel{\text{def}}{=} \{\hat{\mathbf{w}}_{T_1}, \ldots, \hat{\mathbf{w}}_{T_n}\}$ of unit L_2 norm weight vectors where each $\hat{\mathbf{w}}_{T_i}$ operates on a unit L_2 norm feature vector $\hat{\boldsymbol{\phi}}_{T_i}(x, \mathbf{y})$. For any $\mathcal{T}$ and any such set $\mathcal{W}$, we consider an arbitrary unit L_2 norm conical combination of each weight in $\mathcal{W}$ realized by a n-dimensional weight vector $\mathbf{q} \stackrel{\text{def}}{=} (q_1, \ldots, q_n)$, where $\sum_{i=1}^n q_i^2 = 1$ and each $q_i \geq 0$. Given any $(x, \mathbf{y})$ and any $\mathcal{T}$, we define the score $F_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y})$ achieved on $(x, \mathbf{y})$ by the conical combination $(\mathcal{W}, \mathbf{q})$ on $\mathcal{T}$ as

$$F_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}) \stackrel{\text{def}}{=} \frac{1}{\sqrt{n}} \sum_{i=1}^{n} q_i \langle \hat{\mathbf{w}}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x, \mathbf{y}) \rangle \,, \tag{3}$$

where the $\sqrt{n}$ denominator ensures that we always have $F_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}) \leq 1$ in view of the fact that $\sum_{i=1}^n q_i$ can be as large as $\sqrt{n}$. Note also that $F_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y})$ is the score of the feature vector obtained by the concatenation of all the weight vectors in $\mathcal{W}$ (and weighted by $\mathbf{q}$) acting on a feature vector obtained by concatenating each $\hat{\boldsymbol{\phi}}_{T_i}$ multiplied by $1/\sqrt{n}$. Hence, given $\mathcal{T}$, we define the margin $\Gamma_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y})$ achieved on $(x, \mathbf{y})$ by the conical combination $(\mathcal{W}, \mathbf{q})$ on $\mathcal{T}$ as

$$\Gamma_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}) \;\stackrel{\text{def}}{=}\; \min_{\mathbf{y}' \neq \mathbf{y}} \left[F_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}) - F_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}')\right] . \tag{4}$$

For any unit L_2 norm predictor $\mathbf{w}$ that achieves a margin of $\Gamma(\mathbf{w}, x, \mathbf{y})$ for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, we now show that there exists, with high probability, a unit L_2 norm conical combination $(\mathcal{W}, \mathbf{q})$ on $\mathcal{T}$ achieving margins that are not much smaller than $\Gamma(\mathbf{w}, x, \mathbf{y})$.

Theorem 4. *Consider any unit L_2 norm predictor $\mathbf{w}$ for the complete graph G, acting on a normalized joint feature space, achieving a margin of $\Gamma(\mathbf{w}, x, \mathbf{y})$ for each $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. Then for any $\epsilon > 0$, and any n satisfying Lemma 3, for any $\delta \in (0, 1]$, with probability of at least $1 - \delta$ over all samples $\mathcal{T}$ generated according to $\mathcal{U}(G)^n$, there exists a unit L_2 norm conical combination $(\mathcal{W}, \mathbf{q})$ on $\mathcal{T}$ such that, simultaneously for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, we have*

$$\Gamma_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}) \;\geq\; \frac{1}{\sqrt{1+\epsilon}} \left[\Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon\right] .$$

From Theorem 4, and since $\mathcal{A}^{\gamma}(s)$ is a non-increasing function of s, it follows that, with probability at least $1 - \delta$ over the random draws of $\mathcal{T} \sim \mathcal{U}(G)^n$, there exists $(\mathcal{W}, \mathbf{q})$ on $\mathcal{T}$ such that, simultaneously for all $\forall (x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, for any n satisfying Lemma 3 we have

$$\mathcal{A}^{\gamma}(\Gamma_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y})) \;\leq\; \mathcal{A}^{\gamma}\left(\left[\Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon\right](1+\epsilon)^{-1/2}\right) .$$

Hence, instead of searching for a predictor $\mathbf{w}$ for the complete graph G that achieves a small expected ramp loss $\mathbf{E}_{(x,\mathbf{y})\sim D}\mathcal{A}^{\gamma}(\Gamma(\mathbf{w}, x, \mathbf{y}))$, Theorem 4 tells us that we can settle the search for a

unit L_2 norm conical combination $(\mathcal{W}, \mathbf{q})$ on a sample $\mathcal{T}$ of randomly-generated spanning trees of G that achieves small $\mathbf{E}_{(x,\mathbf{y})\sim D}\mathcal{A}^\gamma(\Gamma_\mathcal{T}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}))$. But recall that $\Gamma_\mathcal{T}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}))$ is the margin of a weight vector obtained by the concatenation of all the weight vectors in $\mathcal{W}$ (weighted by $\mathbf{q}$) on a feature vector obtained by the concatenation of the n feature vectors $(1/\sqrt{n})\hat{\boldsymbol{\phi}}_{T_i}$. It thus follows that any standard risk bound for the SVM applies directly to $\mathbf{E}_{(x,\mathbf{y})\sim D}\mathcal{A}^\gamma(\Gamma_\mathcal{T}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}))$. Hence, by adapting the SVM risk bound of [8], we have the following result.

Theorem 5. *Consider any sample $\mathcal{T}$ of n spanning trees of the complete graph G. For any $\gamma > 0$ and any $0 < \delta \leq 1$, with probability of at least $1 - \delta$ over the random draws of $S \sim D^m$, simultaneously for all unit L_2 norm conical combinations $(\mathcal{W}, \mathbf{q})$ on $\mathcal{T}$, we have*

$$\mathop{\mathbf{E}}_{(x,\mathbf{y})\sim D} \mathcal{A}^\gamma(\Gamma_\mathcal{T}(\mathcal{W}, \mathbf{q}, x, \mathbf{y})) \;\leq\; \frac{1}{m}\sum_{i=1}^{m} \mathcal{A}^\gamma(\Gamma_\mathcal{T}(\mathcal{W}, \mathbf{q}, x_i, \mathbf{y}_i)) + \frac{2}{\gamma\sqrt{m}} + 3\sqrt{\frac{\ln(2/\delta)}{2m}}\,.$$

Hence, according to this theorem, the conical combination $(\mathcal{W}, \mathbf{q})$ having the best generalization guarantee is the one which minimizes the sum of the first two terms on the right hand side of the inequality. Note that the theorem is still valid if we replace, in the empirical risk term, the non-convex ramp loss $\mathcal{A}^\gamma$ by the convex hinge loss $\mathcal{L}^\gamma$. This provides the theoretical basis of the proposed optimization problem for learning $(\mathcal{W}, \mathbf{q})$ on the sample $\mathcal{T}$.

4 A L_2-Norm Random Spanning Tree Approximation Approach

If we introduce the usual slack variables $\xi_k \stackrel{\text{def}}{=} \gamma \cdot \mathcal{L}^\gamma(\Gamma_\mathcal{T}(\mathcal{W}, \mathbf{q}, x_k, \mathbf{y}_k)$, Theorem 5 suggests that we should minimize $\frac{1}{\gamma}\sum_{k=1}^m \xi_k$ for some fixed margin value $\gamma > 0$. Rather than performing this task for several values of γ, we show in the supplementary material that we can, equivalently, solve the following optimization problem for several values of $C > 0$.

Definition 6. Primal L_2-norm Random Tree Approximation.

$$\begin{aligned}
\underset{\mathbf{w}_{T_i},\xi_k}{\textbf{\textit{min}}} \quad & \frac{1}{2}\sum_{i=1}^{n} ||\mathbf{w}_{T_i}||_2^2 + C\sum_{k=1}^{m}\xi_k \\
\textbf{\textit{s.t.}} \quad & \sum_{i=1}^{n}\langle \mathbf{w}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x_k, \mathbf{y}_k)\rangle - \underset{\mathbf{y}\neq\mathbf{y}_k}{\textbf{\textit{max}}} \sum_{i=1}^{n}\langle \mathbf{w}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x_k, \mathbf{y})\rangle \geq 1 - \xi_k, \\
& \xi_k \geq 0\,, \forall\, k \in \{1, \ldots, m\},
\end{aligned}$$

where $\{\mathbf{w}_{T_i} | T_i \in \mathcal{T}\}$ are the feature weights to be learned on each tree, ξ_k is the margin slack allocated for each x_k, and C is the slack parameter that controls the amount of regularization.

This primal form has the interpretation of maximizing the joint margins from individual trees between (correct) training examples and all the other (incorrect) examples.

The key for the efficient optimization is solving the 'argmax' problem efficiently. In particular, we note that the space of all multilabels is exponential in size, thus forbidding exhaustive enumeration over it. In the following, we show how exact inference over a collection $\mathcal{T}$ of trees can be implemented in $\Theta(Kn\ell)$ time per data point, where K is the smallest number such that the average score of the K'th best multilabel for each tree of $\mathcal{T}$ is at most $F_\mathcal{T}(x, \mathbf{y}) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n \langle \mathbf{w}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x, \mathbf{y})\rangle$. Whenever K is polynomial in the number of labels, this gives us exact polynomial-time inference over the ensemble of trees.

4.1 Fast inference over a collection of trees

It is well known that the exact solution to the inference problem

$$\hat{\mathbf{y}}_{T_i}(x) = \underset{\mathbf{y}\in\mathcal{Y}}{\text{argmax}}\; F_{\mathbf{w}_{T_i}}(x, \mathbf{y}) \stackrel{\text{def}}{=} \underset{\mathbf{y}\in\mathcal{Y}}{\text{argmax}}\; \langle \mathbf{w}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x, \mathbf{y})\rangle, \tag{5}$$

on an individual tree T_i can be obtained in $\Theta(\ell)$ time by dynamic programming. However, there is no guarantee that the maximizer $\hat{\mathbf{y}}_{T_i}$ of Equation (5) is also a maximizer of $F_\mathcal{T}$. In practice, $\hat{\mathbf{y}}_{T_i}$

can differ for each spanning tree $T_i \in \mathcal{T}$. Hence, instead of using only the best scoring multilabel $\hat{\mathbf{y}}_{T_i}$ from each individual $T_i \in \mathcal{T}$, we consider the set of the K highest scoring multilabels $\mathcal{Y}_{T_i,K} = \{\hat{\mathbf{y}}_{T_i,1}, \cdots, \hat{\mathbf{y}}_{T_i,K}\}$ of $F_{\mathbf{w}_{T_i}}(x, \mathbf{y})$. In the supplementary material we describe a dynamic programming to find the K highest multilabels in $\Theta(K\ell)$ time. Running this algorithm for all of the trees gives us a candidate set of $\Theta(Kn)$ multilabels $\mathcal{Y}_{\mathcal{T},K} = \mathcal{Y}_{T_1,K} \cup \cdots \cup \mathcal{Y}_{T_n,K}$. We now state a key lemma that will enable us to verify if the candidate set contains the maximizer of $F_{\mathcal{T}}$.

Lemma 7. *Let* $\mathbf{y}^\star_K = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}_{\mathcal{T},K}} F_{\mathcal{T}}(x, \mathbf{y})$ *be the highest scoring multilabel in* $\mathcal{Y}_{\mathcal{T},K}$*. Suppose that*

$$F_{\mathcal{T}}(x, \mathbf{y}^\star_K) \geq \frac{1}{n} \sum_{i=1}^{n} F_{\mathbf{w}_{T_i}}(x, \mathbf{y}_{T_i,K}) \stackrel{\text{def}}{=} \theta_x(K).$$

It follows that $F_{\mathcal{T}}(x, \mathbf{y}^\star_K) = \max_{\mathbf{y} \in \mathcal{Y}} F_{\mathcal{T}}(x, \mathbf{y})$.

We can use any K satisfying the lemma as the length of K-best lists, and be assured that $\mathbf{y}^\star_K$ is a maximizer of $F_{\mathcal{T}}$.

We now examine the conditions under which the highest scoring multilabel is present in our candidate set $\mathcal{Y}_{\mathcal{T},K}$ with high probability. For any $x \in \mathcal{X}$ and any predictor $\mathbf{w}$, let $\hat{\mathbf{y}} \stackrel{\text{def}}{=} \mathbf{y}_{\mathbf{w}}(x) \stackrel{\text{def}}{=} \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{w}, x, \mathbf{y})$ be the highest scoring multilabel in $\mathcal{Y}$ for predictor $\mathbf{w}$ on the complete graph G.

For any $\mathbf{y} \in \mathcal{Y}$, let $K_T(\mathbf{y})$ be the rank of $\mathbf{y}$ in tree T and let $\rho_T(\mathbf{y}) \stackrel{\text{def}}{=} K_T(\mathbf{y})/|\mathcal{Y}|$ be the normalized rank of $\mathbf{y}$ in tree T. We then have $0 < \rho_T(\mathbf{y}) \leq 1$ and $\rho_T(\mathbf{y}') = \min_{\mathbf{y} \in \mathcal{Y}} \rho_T(\mathbf{y})$ whenever $\mathbf{y}'$ is a highest scoring multilabel in tree T. Since $\mathbf{w}$ and x are arbitrary and fixed, let us drop them momentarily from the notation and let $F(\mathbf{y}) \stackrel{\text{def}}{=} F(\mathbf{w}, x, \mathbf{y})$, and $F_T(\mathbf{y}) \stackrel{\text{def}}{=} F_{\mathbf{w}_T}(x, \mathbf{y})$. Let $\mathcal{U}(\mathcal{Y})$ denote the uniform distribution of multilabels on $\mathcal{Y}$. Then, let $\mu_T \stackrel{\text{def}}{=} \mathbf{E}_{\mathbf{y} \sim \mathcal{U}(\mathcal{Y})} F_T(\mathbf{y})$ and $\mu \stackrel{\text{def}}{=} \mathbf{E}_{T \sim \mathcal{U}(G)} \mu_T$.

Let $\mathcal{T} \sim \mathcal{U}(G)^n$ be a sample of n spanning trees of G. Since the scoring function F_T of each tree T of G is bounded in absolute value, it follows that F_T is a σ_T-sub-Gaussian random variable for some $\sigma_T > 0$. We now show that, with high probability, there exists a tree $T \in \mathcal{T}$ such that $\rho_T(\hat{\mathbf{y}})$ is decreasing exponentially rapidly with $(F(\hat{\mathbf{y}}) - \mu)/\sigma$, where $\sigma^2 \stackrel{\text{def}}{=} \mathbf{E}_{T \sim \mathcal{U}(G)} \sigma_T^2$.

Lemma 8. *Let the scoring function* F_T *of each spanning tree of* G *be a* σ_T*-sub-Gaussian random variable under the uniform distribution of labels;* i.e.*, for each* T *on* G*, there exists* $\sigma_T > 0$ *such that for any* $\lambda > 0$ *we have*

$$\mathop{\mathbf{E}}_{\mathbf{y} \sim \mathcal{U}(\mathcal{Y})} e^{\lambda (F_T(\mathbf{y}) - \mu_T)} \;\leq\; e^{\frac{\lambda^2}{2} \sigma_T^2}\,.$$

Let $\sigma^2 \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{T \sim \mathcal{U}(G)} \sigma_T^2$*, and let* $\alpha \stackrel{\text{def}}{=} \Pr_{T \sim \mathcal{U}(G)} \Big(\mu_T \leq \mu \;\wedge\; F_T(\hat{\mathbf{y}}) \geq F(\hat{\mathbf{y}}) \;\wedge\; \sigma_T^2 \leq \sigma^2 \Big)$*. Then,*

$$\Pr_{\mathcal{T} \sim \mathcal{U}(G)^n} \left(\exists T \in \mathcal{T} \colon\; \rho_T(\hat{\mathbf{y}}) \leq e^{-\frac{1}{2} \frac{(F(\hat{\mathbf{y}}) - \mu)^2}{\sigma^2}} \right) \geq 1 - (1 - \alpha)^n\,.$$

Thus, even for very small α, when n is large enough, there exists, with high probability, a tree $T \in \mathcal{T}$ such that $\hat{\mathbf{y}}$ has a small $\rho_T(\hat{\mathbf{y}})$ whenever $[F(\hat{\mathbf{y}}) - \mu]/\sigma$ is large for G. For example, when $|\mathcal{Y}| = 2^\ell$ (the multiple binary classification case), we have with probability of at least $1 - (1 - \alpha)^n$, that there exists $T \in \mathcal{T}$ such that $K_T(\hat{\mathbf{y}}) = 1$ whenever $F(\hat{\mathbf{y}}) - \mu \geq \sigma \sqrt{2\ell \ln 2}$.

4.2 Optimization

To optimize the L_2-norm RTA problem (Definition 6) we convert it to the marginalized dual form (see the supplementary material for the derivation), which gives us a polynomial-size problem (in the number of microlabels) and allows us to use kernels to tackle complex input spaces efficiently.

Definition 9. **L_2-norm RTA Marginalized Dual**

$$\max_{\boldsymbol{\mu} \in \mathcal{M}^m} \quad \frac{1}{|E_{\mathcal{T}}|} \sum_{e,k,\mathbf{u}_e} \mu(k, e, \mathbf{u}_e) - \frac{1}{2} \sum_{\substack{e,k,\mathbf{u}_e, \\ k',\mathbf{u}'_e}} \mu(k, e, \mathbf{u}_e) K^e_{\mathcal{T}}(x_k, \mathbf{u}_e; x'_k, \mathbf{u}'_e) \mu(k', e, \mathbf{u}'_e)\,,$$

where $E_{\mathcal{T}}$ *is the union of the sets of edges appearing in* $\mathcal{T}$*, and* $\boldsymbol{\mu} \in \mathcal{M}^m$ *are the marginal dual variables* $\boldsymbol{\mu} \stackrel{\text{def}}{=} (\mu(k, e, \mathbf{u}_e))_{k,e,\mathbf{u}_e}$*, with the triplet* $(k, e, \mathbf{u}_e)$ *corresponding to labeling the edge*

DATASET	MICROLABEL LOSS (%)					0/1 LOSS (%)				
	SVM	MTL	MMCRF	MAM	RTA	SVM	MTL	MMCRF	MAM	RTA
EMOTIONS	22.4	20.2	20.1	*19.5*	**18.8**	77.8	74.5	71.3	*69.6*	**66.3**
YEAST	*20.0*	20.7	21.7	20.1	**19.8**	*85.9*	88.7	93.0	86.0	**77.7**
SCENE	*9.8*	11.6	18.4	17.0	**8.8**	*47.2*	55.2	72.2	94.6	**30.2**
ENRON	6.4	6.5	6.2	**5.0**	*5.3*	99.6	99.6	92.7	*87.9*	**87.7**
CAL500	**13.7**	*13.8*	**13.7**	**13.7**	*13.8*	100.0	100.0	100.0	100.0	100.0
FINGERPRINT	**10.3**	17.3	*10.5*	*10.5*	10.7	99.0	100.0	*99.6*	*99.6*	**96.7**
NCI60	15.3	16.0	*14.6*	**14.3**	14.9	56.9	*53.0*	63.1	60.0	**52.9**
MEDICAL	*2.6*	*2.6*	**2.1**	**2.1**	**2.1**	91.8	91.8	63.8	*63.1*	**58.8**
CIRCLE10	4.7	6.3	2.6	*2.5*	**0.6**	28.9	33.2	20.3	*17.7*	**4.0**
CIRCLE50	5.7	6.2	**1.5**	*2.1*	3.8	69.8	72.3	**38.8**	*46.2*	*52.8*

Table 1: Prediction performance of each algorithm in terms of microlabel loss and 0/1 loss. The best performing algorithm is highlighted with **boldface**, the second best is in *italic*.

$e=(v,v')\in E_{\mathcal{T}}$ of the output graph by $\mathbf{u}_e=(u_v,u_{v'})\in\mathcal{Y}_v\times\mathcal{Y}_{v'}$ for the training example x_k. Also, $\mathcal{M}^m$ is the marginal dual feasible set and

$$K^e_{\mathcal{T}}(x_k,\mathbf{u}_e;x_{k'},\mathbf{u}'_e)\stackrel{def}{=}\frac{N_{\mathcal{T}}(e)}{|E_{\mathcal{T}}|^2}\,K(x_k,x_{k'})\left\langle\boldsymbol{\psi}_e(y_{kv},y_{kv'})-\boldsymbol{\psi}_e(u_v,u_{v'}),\boldsymbol{\psi}_e(y_{k'v},y_{k'v'})-\boldsymbol{\psi}_e(u'_v,u'_{v'})\right\rangle$$

is the joint kernel of input features and the differences of output features of true and competing multilabels ($\mathbf{y}_k,\mathbf{u}$), projected to the edge e. Finally, $\mathcal{N}_{\mathcal{T}}(e)$ denotes the number of times e appears among the trees of the ensemble.

The master algorithm described in the supplementary material iterates over each training example until convergence. The processing of each training example x_k proceeds by finding the worst violating multilabel of the ensemble defined as

$$\bar{\mathbf{y}}_k\stackrel{\text{def}}{=}\operatorname*{argmax}_{\mathbf{y}\neq\mathbf{y}_k}F_{\mathcal{T}}(x_k,\mathbf{y})\,, \tag{6}$$

using the K-best inference approach of the previous section, with the modification that the correct multilabel is excluded from the K-best lists. The worst violator $\bar{\mathbf{y}}_k$ is mapped to a vertex

$$\bar{\boldsymbol{\mu}}(x_k)=C\cdot([\bar{\mathbf{y}}_e=\mathbf{u}_e])_{e,\mathbf{u}_e}\in\mathcal{M}_k$$

corresponding to the steepest feasible ascent direction (*c.f*, [9]) in the marginal dual feasible set $\mathcal{M}_k$ of example x_k, thus giving us a subgradient of the objective of Definition 9. An exact line search is used to find the saddle point between the current solution and $\bar{\boldsymbol{\mu}}$.

5 Empirical Evaluation

We compare our method RTA to Support Vector Machine (SVM) [10, 11], Multitask Feature Learning (MTL) [12], Max-Margin Conditional Random Fields (MMCRF) [9] which uses the loopy belief propagation algorithm for approximate inference on the general graph, and Maximum Average Marginal Aggregation (MAM) [5] which is a multilabel ensemble model that trains a set of random tree based learners separately and performs the final approximate inference on a union graph of the edge potential functions of the trees. We use ten multilabel datasets from [5]. Following [5], MAM is constructed with 180 tree based learners, and for MMCRF a consensus graph is created by pooling edges from 40 trees. We train RTA with up to 40 spanning trees and with K up to 32. The linear kernel is used for methods that require kernelized input. Margin slack parameters are selected from $\{100, 50, 10, 1, 0.5, 0.1, 0.01\}$. We use 5-fold cross-validation to compute the results.

Prediction performance. Table 1 shows the performance in terms of microlabel loss and 0/1 loss. The best methods are highlighted in '**boldface**' and the second best in '*italics*' (see supplementary material for full results). RTA quite often improves over MAM in 0/1 accuracy, sometimes with noticeable margin except for *Enron* and *Circle50*. The performances in microlabel accuracy are quite similar while RTA is slightly above the competition. This demonstrates the advantage of RTA that gains by optimizing on a collection of trees simultaneously rather than optimizing on individual trees as MAM. In addition, learning using approximate inference on a general graph seems less

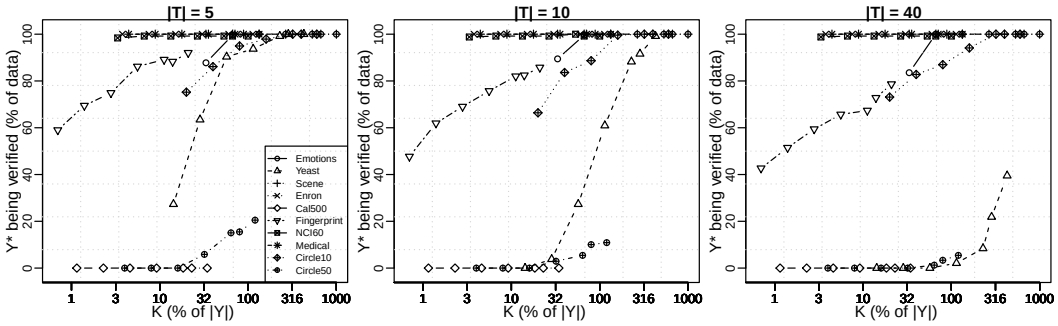


Figure 1: Percentage of examples with provably optimal $\mathbf{y}^*$ being in the K-best lists plotted as a function of K, scaled with respect to the number of microlabels in the dataset.

favorable as the tree-based methods, as MMCRF quite consistently trails to RTA and MAM in both microlabel and 0/1 error, except for *Circle50* where it outperforms other models. Finally, we notice that SVM, as a single label classifier, is very competitive against most multilabel methods for microlabel accuracy.

Exactness of inference on the collection of trees. We now study the empirical behavior of the inference (see Section 4) on the collection of trees, which, if taken as a single general graph, would call for solving an $\mathcal{NP}$-hard inference problem. We provide here empirical evidence that we can perform exact inference on most examples in most datasets in polynomial time.
We ran the K-best inference on eleven datasets where the RTA models were trained with different amounts of spanning trees $|\mathcal{T}| = \{5, 10, 40\}$ and values for $K = \{2, 4, 8, 16, 32, 40, 60\}$. For each parameter combination and for each example, we recorded whether the K-best inference was provably exact on the collection (*i.e.*, if Lemma 7 was satisfied). Figure 1 plots the percentage of examples where the inference was indeed provably exact. The values are shown as a function of K, expressed as the percentage of the number of microlabels in each dataset. Hence, 100% means $K = \ell$, which denotes low polynomial ($\Theta(n\ell^2)$) time inference in the exponential size multilabel space.

We observe, from Figure 1, on some datasets (*e.g.*, *Medical*, *NCI60*), that the inference task is very easy since exact inference can be computed for most of the examples even with K values that are below 50% of the number of microlabels. By setting $K = \ell$ (*i.e.*, 100%) we can perform exact inference for about 90% of the examples on nine datasets with five trees, and eight datasets with 40 trees. On two of the datasets (*Cal500*, *Circle50*), inference is not (in general) exact with low values of K. Allowing K to grow superlinearly on ℓ would possibly permit exact inference on these datasets. However, this is left for future studies.

Finally, we note that the difficulty of performing provably exact inference slightly increases when more spanning trees are used. We have observed that, in most cases, the optimal multilabel $\mathbf{y}^*$ is still on the K-best lists but the conditions of Lemma 7 are no longer satisfied, hence forbidding us to prove exactness of the inference. Thus, working to establish alternative proofs of exactness is a worthy future research direction.

6 Conclusion

The main theoretical result of the paper is the demonstration that if a large margin structured output predictor exists, then combining a small sample of random trees will, with high probability, generate a predictor with good generalization. The key attraction of this approach is the tractability of the inference problem for the ensemble of trees, both indicated by our theoretical analysis and supported by our empirical results. However, as a by-product, we have a significant added benefit: we do not need to know the output structure *a priori* as this is generated implicitly in the learned weights for the trees. This is used to significant advantage in our experiments that automatically leverage correlations between the multiple target outputs to give a substantive increase in accuracy. It also suggests that the approach has enormous potential for applications where the structure of the output is not known but is expected to play an important role.

References

[1] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press, 2004.

[2] Martin J. Wainwright, Tommy S. Jaakkola, and Alan S. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.

[3] Michael I. Jordan and Martin J Wainwright. Semidefinite relaxations for approximate inference on graphs with cycles. In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 369–376. MIT Press, 2004.

[4] Amir Globerson and Tommi S. Jaakkola. Approximate inference using planar graph decomposition. In B. Schölkopf, J.C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 473–480. MIT Press, 2007.

[5] Hongyu Su and Juho Rousu. Multilabel classification through random graph ensembles. *Machine Learning*, dx.doi.org/10.1007/s10994-014-5465-9, 2014.

[6] Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, New York, 1999.

[7] Thomas Gärtner and Shankar Vembu. On structured output training: hard cases and an efficient alternative. *Machine Learning*, 79:227–242, 2009.

[8] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[9] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Efficient algorithms for max-margin structured classification. *Predicting Structured Data*, pages 105–129, 2007.

[10] Kristin P. Bennett. Combining support vector and mathematical programming methods for classifications. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, pages 307–326. MIT Press, Cambridge, MA, 1999.

[11] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, U.K., 2000.

[12] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[13] Yevgeny Seldin, François Laviolette, Nicolò Cesa-Bianchi, John Shawe-Taylor, and Peter Auer. PAC-Bayesian inequalities for martingales. *IEEE Transactions on Information Theory*, 58:7086–7093, 2012.

[14] Andreas Maurer. A note on the PAC Bayesian theorem. *CoRR*, cs.LG/0411099, 2004.

[15] David McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51:5–21, 2003.

[16] Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626, December 2006.

Multilabel Structured Output Learning with Random Spanning Trees of Max-Margin Markov Networks (supplementary material) by: Mario Marchand, Hongyu Su, Emilie Morvant, Juho Rousou, and John Shawe-Taylor

Lemma 1

Let $\hat{\mathbf{w}}_T \stackrel{def}{=} \mathbf{w}_T/\|\mathbf{w}_T\|$, $\hat{\boldsymbol{\phi}}_T \stackrel{def}{=} \boldsymbol{\phi}_T/\|\boldsymbol{\phi}_T\|$. *Let* $\mathcal{U}(G)$ *denote the uniform distribution on* $S(G)$. *Then, we have*

$$F(\mathbf{w}, x, \mathbf{y}) \;=\; \mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} a_T \langle \hat{\mathbf{w}}_T, \hat{\boldsymbol{\phi}}_T(x,\mathbf{y})\rangle, \text{ where } a_T \stackrel{def}{=} \sqrt{\frac{\ell}{2}}\,\|\mathbf{w}_T\|\,.$$

Moreover, for any $\mathbf{w}$ *such that* $\|\mathbf{w}\| = 1$, *we have*

$$\mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} a_T^2 \;=\; 1 \quad ; \quad \mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} a_T \;\le\; 1\,.$$

Proof.

$$\begin{aligned}
F(\mathbf{w}, x, \mathbf{y}) &= \langle \mathbf{w}, \boldsymbol{\phi}(x,\mathbf{y})\rangle \\
&= \sum_{(i,j)\in G} \langle \mathbf{w}_{i,j}, \boldsymbol{\phi}_{i,j}(x, y_i, y_j)\rangle \;=\; \frac{1}{\ell^{\ell-2}}\frac{\ell}{2}\sum_{T\in S(G)}\sum_{(i,j)\in T}\langle \mathbf{w}_{i,j}, \boldsymbol{\phi}_{i,j}(x, y_i, y_j)\rangle \\
&= \frac{\ell}{2}\mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} \langle \mathbf{w}_T, \boldsymbol{\phi}_T(x,\mathbf{y})\rangle \;=\; \frac{\ell}{2}\mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} \|\mathbf{w}_T\|\|\boldsymbol{\phi}_T(x,\mathbf{y})\|\langle \hat{\mathbf{w}}_T, \hat{\boldsymbol{\phi}}_T(x,\mathbf{y})\rangle \\
&= \mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} \sqrt{\frac{\ell}{2}}\,\|\mathbf{w}_T\|\langle \hat{\mathbf{w}}_T, \hat{\boldsymbol{\phi}}_T(x,\mathbf{y})\rangle \;=\; \mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} a_T\langle \hat{\mathbf{w}}_T, \hat{\boldsymbol{\phi}}_T(x,\mathbf{y})\rangle\,,
\end{aligned}$$

where

$$a_T \stackrel{\text{def}}{=} \sqrt{\frac{\ell}{2}}\,\|\mathbf{w}_T\|\,.$$

Now, for any $\mathbf{w}$ such that $\|\mathbf{w}\| = 1$, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} a_T^2 &= \frac{\ell}{2}\mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} \|\mathbf{w}_T\|^2 \;=\; \frac{\ell}{2}\frac{1}{\ell^{\ell-2}}\sum_{T\in S(G)}\|\mathbf{w}_T\|^2 \;=\; \frac{\ell}{2}\frac{1}{\ell^{\ell-2}}\sum_{T\in S(G)}\sum_{(i,j)\in T}\|\mathbf{w}_{i,j}\|^2 \\
&= \sum_{(i,j)\in G}\|\mathbf{w}_{i,j}\|^2 \;=\; \|\mathbf{w}\|^2 \;=\; 1\,.
\end{aligned}$$

Since the variance of a_T must be positive, we have, for any $\mathbf{w}$ of unit L_2 norm, that

$$\mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} a_T \;\le\; 1\,.$$

□

Lemma 2

Consider any unit L_2 *norm predictor* $\mathbf{w}$ *on the complete graph* G *that achieves a margin of* $\Gamma(\mathbf{w}, x, \mathbf{y})$ *for each* $(x,\mathbf{y}) \in \mathcal{X}\times\mathcal{Y}$, *then we have*

$$\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) \;\ge\; \Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon \quad \forall (x,\mathbf{y}) \in \mathcal{X}\times\mathcal{Y}\,,$$

whenever for all $(x,\mathbf{y}) \in \mathcal{X}\times\mathcal{Y}$, *we have*

$$|F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) - F(\mathbf{w}, x, \mathbf{y})| \le \epsilon\,.$$

Proof. From the condition of the lemma, we have simultaneously for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ and $(x, \mathbf{y}') \in \mathcal{X} \times \mathcal{Y}$, that

$$F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) \;\geq\; F(\mathbf{w}, x, \mathbf{y}) - \epsilon \quad \text{AND} \quad F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}') \;\leq\; F(\mathbf{w}, x, \mathbf{y}') + \epsilon\,.$$

Therefore,

$$F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) - F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}') \;\geq\; F(\mathbf{w}, x, \mathbf{y}) - F(\mathbf{w}, x, \mathbf{y}') - 2\epsilon\,.$$

Hence, for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, we have

$$\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) \;\geq\; \Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon\,.$$

□

Lemma 3

Consider any $\epsilon > 0$ and any unit L_2 norm predictor $\mathbf{w}$ for the complete graph G acting on a normalized joint feature space. For any $\delta \in (0, 1)$, let

$$n \;\geq\; \frac{\ell^2}{\epsilon^2}\left(\frac{1}{16} + \frac{1}{2}\ln\frac{8\sqrt{n}}{\delta}\right)^2\,. \tag{2}$$

Then with probability of at least $1 - \delta/2$ over all samples $\mathcal{T}$ generated according to $\mathcal{U}(G)^n$, we have, simultaneously for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, that

$$|F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) - F(\mathbf{w}, x, \mathbf{y})| \;\leq \epsilon\,.$$

Proof. Consider an isotropic Gaussian distribution of joint feature vectors of variance σ^2, centred on $\boldsymbol{\phi}(x, \mathbf{y})$, with a density given by

$$Q_{\boldsymbol{\phi}}(\boldsymbol{\zeta}) \;\stackrel{\text{def}}{=}\; \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp -\frac{\|\boldsymbol{\zeta} - \boldsymbol{\phi}\|^2}{2\sigma^2}\,,$$

where N is the dimension of the feature vectors. When the feature space is infinite-dimensional, we can consider Q to be a Gaussian process. The end results will not depend on N.

Given the fixed $\mathbf{w}$ stated in the theorem, let us define the risk $R(Q_{\boldsymbol{\phi}}, \mathbf{w}_T)$ of $Q_{\boldsymbol{\phi}}$ on the tree T by $\mathop{\mathbf{E}}_{\boldsymbol{\zeta} \sim Q_{\boldsymbol{\phi}}} \langle \mathbf{w}_T, \boldsymbol{\zeta} \rangle$. By the linearity of $\langle \cdot, \cdot \rangle$, we have

$$R(Q_{\boldsymbol{\phi}}, \mathbf{w}_T) \;\stackrel{\text{def}}{=}\; \mathop{\mathbf{E}}_{\boldsymbol{\zeta} \sim Q_{\boldsymbol{\phi}}} \langle \mathbf{w}_T, \boldsymbol{\zeta} \rangle \;=\; \langle \mathbf{w}_T, \mathop{\mathbf{E}}_{\boldsymbol{\zeta} \sim Q_{\boldsymbol{\phi}}} \boldsymbol{\zeta} \rangle \;=\; \langle \mathbf{w}_T, \boldsymbol{\phi} \rangle\,,$$

which is independent of σ.

Gibbs' risk $R(Q_{\boldsymbol{\phi}})$ and its empirical estimate $R_{\mathcal{T}}(Q_{\boldsymbol{\phi}})$ are defined as

$$\begin{aligned} R(Q_{\boldsymbol{\phi}}) \;&\stackrel{\text{def}}{=}\; \mathop{\mathbf{E}}_{T \sim \mathcal{U}(G)} R(Q_{\boldsymbol{\phi}}, \mathbf{w}_T) \;=\; \mathop{\mathbf{E}}_{T \sim \mathcal{U}(G)} \langle \mathbf{w}_T, \boldsymbol{\phi} \rangle \\ R_{\mathcal{T}}(Q_{\boldsymbol{\phi}}) \;&\stackrel{\text{def}}{=}\; \frac{1}{n}\sum_{i=1}^{n} R(Q_{\boldsymbol{\phi}}, \mathbf{w}_{T_i}) \;=\; \frac{1}{n}\sum_{i=1}^{n} \langle \mathbf{w}_{T_i}, \boldsymbol{\phi} \rangle\,. \end{aligned}$$

Consequently, from the definitions of F and $F_{\mathcal{T}}$, we have

$$\begin{aligned} F(\mathbf{w}, x, \mathbf{y}) \;&=\; \frac{\ell}{2} R(Q_{\boldsymbol{\phi}(x,\mathbf{y})}) \\ F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) \;&=\; \frac{\ell}{2} R_{\mathcal{T}}(Q_{\boldsymbol{\phi}(x,\mathbf{y})})\,. \end{aligned}$$

Recall that $\boldsymbol{\phi}$ is a normalized feature map that applies to all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. Therefore, if we have with probability $\geq 1 - \delta/2$ that, simultaneously for all $\boldsymbol{\phi}$ of unit L_2 norm,

$$\frac{\ell}{2}|R_{\mathcal{T}}(Q_{\boldsymbol{\phi}}) - R(Q_{\boldsymbol{\phi}})| \leq \epsilon\,, \tag{7}$$

then, with the same probability, we will have simultaneously $\forall (x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, that

$$|F_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) - F(\mathbf{w}, x, \mathbf{y})| \leq \epsilon \,,$$

and, consequently, the lemma will be proved.

To prove that we satisfy Equation (7) with probability $\geq 1 - \delta/2$ simultaneously for all $\boldsymbol{\phi}$ of unit L_2 norm, let us adapt some elements of PAC-Bayes theory to our case. Note that we cannot use the usual PAC-Bayes bounds, such as those proposed by [13] because, in our case, the loss $\langle \mathbf{w}_T, \boldsymbol{\zeta} \rangle$ of each individual "predictor" $\boldsymbol{\zeta}$ is unbounded.

The distribution $Q_{\boldsymbol{\phi}}$ defined above constitutes the posterior distribution. For the prior P, let us use an isotropic Gaussian with variance σ^2 centered at the origin. Hence $P = Q_{\mathbf{0}}$. In that case we have

$$\mathrm{KL}(Q_{\boldsymbol{\phi}} \| P) = \frac{\|\boldsymbol{\phi}\|^2}{2\sigma^2} \;=\; \frac{1}{2\sigma^2} \,.$$

Given a tree sample $\mathcal{T}$ of n spanning trees, let

$$\Delta \mathbf{w} \;\stackrel{\text{def}}{=}\; \frac{1}{n} \sum_{k=1}^{n} \mathbf{w}_{T_k} - \mathop{\mathbf{E}}_{T \sim \mathcal{U}(G)} \mathbf{w}_T \,,$$

and consider the Gaussian quadrature

$$\begin{aligned}
\mathcal{I} \;&\stackrel{\text{def}}{=}\; \mathop{\mathbf{E}}_{\boldsymbol{\zeta} \sim P} e^{\sqrt{n} |\langle \Delta \mathbf{w}, \boldsymbol{\zeta} \rangle|} \\
&=\; e^{\frac{1}{2} n \sigma^2 \|\Delta \mathbf{w}\|^2} \left(1 + \mathrm{Erf} \left[\sqrt{\frac{n}{2}} \|\Delta \mathbf{w}\| \sigma \right] \right) \\
&\leq\; 2 e^{\frac{1}{2} n \sigma^2 \|\Delta \mathbf{w}\|^2} \,.
\end{aligned}$$

We can then use this result for $\mathcal{I}$ to upper bound the Laplace transform $\mathcal{L}$ in the following way.

$$\begin{aligned}
\mathcal{L} \;&\stackrel{\text{def}}{=}\; \mathop{\mathbf{E}}_{\mathcal{T} \sim \mathcal{U}(G)^n} \mathop{\mathbf{E}}_{\boldsymbol{\zeta} \sim P} e^{\sqrt{n} |\langle \Delta \mathbf{w}, \boldsymbol{\zeta} \rangle|} \\
&\leq\; 2 \mathop{\mathbf{E}}_{\mathcal{T} \sim \mathcal{U}(G)^n} e^{\frac{1}{2} n \sigma^2 \|\Delta \mathbf{w}\|^2} \\
&=\; 2 \mathop{\mathbf{E}}_{\mathcal{T} \sim \mathcal{U}(G)^n} e^{\frac{1}{2} n \sigma^2 \sum_{(i,j) \in G} \|(\Delta \mathbf{w})_{i,j}\|^2} \,.
\end{aligned}$$

Since

$$\mathop{\mathbf{E}}_{T \sim \mathcal{U}(G)} \mathbf{w}_T = \frac{2}{\ell} \mathbf{w} \,,$$

we can write

$$\|(\Delta \mathbf{w})_{i,j}\|^2 \;=\; \left\| \frac{1}{n} \sum_{k=1}^{n} (\mathbf{w}_{T_k})_{i,j} - \frac{2}{\ell} \mathbf{w}_{i,j} \right\|^2 \,.$$

Note that for each $(i, j) \in G$, any sample $\mathcal{T}$, and each $T_k \in \mathcal{T}$, we can write

$$(\mathbf{w}_{T_k})_{i,j} \;=\; \mathbf{w}_{i,j} Z_{i,j}^k \,.$$

where $Z_{i,j}^k = 1$ if $(i, j) \in T_k$ and $Z_{i,j}^k = 0$ if $(i, j) \notin T_k$. Hence, we have

$$\|(\Delta \mathbf{w})_{i,j}\|^2 \;=\; \|\mathbf{w}_{i,j}\|^2 \left(\frac{1}{n} \sum_{k=1}^{n} Z_{i,j}^k - \frac{2}{\ell} \right)^2 \,.$$

Hence, for $\sigma^2 \leq 4$ and $p \stackrel{\text{def}}{=} 2/\ell$, we have

$$\begin{aligned}
\mathcal{L} \;&\leq\; 2 \mathop{\mathbf{E}}_{\mathcal{T} \sim \mathcal{U}(G)^n} e^{\frac{1}{2} n \sigma^2 \sum_{(i,j) \in G} \|\mathbf{w}_{i,j}\|^2 \left(\frac{1}{n} \sum_{k=1}^{n} Z_{i,j}^k - \frac{2}{\ell} \right)^2} \\
&\leq\; 2 \mathop{\mathbf{E}}_{\mathcal{T} \sim \mathcal{U}(G)^n} e^{2n \sum_{(i,j) \in G} \|\mathbf{w}_{i,j}\|^2 \left(\frac{1}{n} \sum_{k=1}^{n} Z_{i,j}^k - p \right)^2} \\
&\leq\; 2 \sum_{(i,j) \in G} \|\mathbf{w}_{i,j}\|^2 \mathop{\mathbf{E}}_{\mathcal{T} \sim \mathcal{U}(G)^n} e^{2n \left(\frac{1}{n} \sum_{k=1}^{n} Z_{i,j}^k - p \right)^2} \,,
\end{aligned}$$

where the last inequality is obtained by using $\sum_{(i,j)\in G} \|\mathbf{w}_{i,j}\|^2 = 1$ and by using Jensen's inequality on the convexity of the exponential.

Now, for any $(q,p) \in [0,1]^2$, let

$$\mathrm{kl}(q\|p) \stackrel{\text{def}}{=} q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p} \,.$$

Then, by using $2(q-p)^2 \leq \mathrm{kl}(q\|p)$ (Pinsker's inequality), we have for $n \geq 8$,

$$\mathcal{L} \;\leq\; 2 \sum_{(i,j)\in G} \|\mathbf{w}_{i,j}\|^2 \mathop{\mathbf{E}}_{\mathcal{T}\sim\mathcal{U}(G)^n} e^{n\mathrm{kl}\left(\frac{1}{n}\sum_{k=1}^n Z_{i,j}^k \| p\right)} \;\leq\; 4\sqrt{n} \,,$$

where the last inequality follows from Maurer's lemma [14] applied, for any fixed $(i,j) \in G$, to the collection of n independent Bernoulli variables $Z_{i,j}^k$ of probability p.

The rest of the proof follows directly from standard PAC-Bayes theory [15, 13], which, for completeness, we briefly outline here.

Since

$$\mathop{\mathbf{E}}_{\boldsymbol{\zeta}\sim P} e^{\sqrt{n}|\langle \Delta\mathbf{w}, \boldsymbol{\zeta}\rangle|}$$

is a non negative random variable, Markov's inequality implies that with probability $> 1-\delta/2$ over the random draws of $\mathcal{T}$, we have

$$\ln \mathop{\mathbf{E}}_{\boldsymbol{\zeta}\sim P} e^{\sqrt{n}|\langle \Delta\mathbf{w}, \boldsymbol{\zeta}\rangle|} \;\leq \ln \frac{8\sqrt{n}}{\delta} \,.$$

By the change of measure inequality, we have with probability $> 1-\delta/2$ over the random draws of $\mathcal{T}$, simultaneously for all $\boldsymbol{\phi}$,

$$\sqrt{n} \mathop{\mathbf{E}}_{\boldsymbol{\zeta}\sim Q_{\boldsymbol{\phi}}} |\langle \Delta\mathbf{w}, \boldsymbol{\zeta}\rangle| \;\leq\; \mathrm{KL}\left(Q_{\boldsymbol{\phi}} \| P\right) + \ln \frac{8\sqrt{n}}{\delta} \,.$$

Hence, by using Jensen's inequality on the convex absolute value function, we have with probability $> 1-\delta/2$ over the random draws of $\mathcal{T}$, simultaneously for all $\boldsymbol{\phi}$,

$$|\langle \Delta\mathbf{w}, \boldsymbol{\phi}\rangle| \;\leq\; \frac{1}{\sqrt{n}} \left[\mathrm{KL}\left(Q_{\boldsymbol{\phi}} \| P\right) + \ln \frac{8\sqrt{n}}{\delta} \right] .$$

Note that we have $\mathrm{KL}(Q_{\boldsymbol{\phi}}\|P) = 1/8$ for $\sigma^2 = 4$ (which is the value we shall use). Also note that the left hand side of this equation equals to $|R_{\mathcal{T}}(Q_{\boldsymbol{\phi}}) - R(Q_{\boldsymbol{\phi}})|$. In that case, we satisfy Equation (7) with probability $1-\delta/2$ simultaneously for all $\boldsymbol{\phi}$ of unit L_2 norm whenever we satisfy

$$\frac{\ell}{2\sqrt{n}} \left[\frac{1}{8} + \ln \frac{8\sqrt{n}}{\delta} \right] \;\leq\; \epsilon \,,$$

which is the condition on n given by the theorem. □

Theorem 4

Consider any unit L_2 norm predictor $\mathbf{w}$ for the complete graph G, acting on a normalized joint feature space, achieving a margin of $\Gamma(\mathbf{w}, x, \mathbf{y})$ for each $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. Then for any $\epsilon > 0$, and any n satisfying Lemma 3, for any $\delta \in (0,1]$, with probability of at least $1-\delta$ over all samples $\mathcal{T}$ generated according to $\mathcal{U}(G)^n$, there exists a unit L_2 norm conical combination $(\mathcal{W}, \mathbf{q})$ on $\mathcal{T}$ such that, simultaneously $\forall (x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, we have

$$\Gamma_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}) \geq \frac{1}{\sqrt{1+\epsilon}} \left[\Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon \right] .$$

Proof. For any $\mathcal{T}$, consider a conical combination $(\mathcal{W}, \mathbf{q})$ where each $\hat{\mathbf{w}}_{T_i} \in \mathcal{W}$ is obtained by projecting $\mathbf{w}$ on T_i and normalizing to unit L_2 norm and where

$$q_i = \frac{a_{T_i}}{\sqrt{\sum_{i=1}^n a_{T_i}^2}} .$$

Then, from equations (3) and (4), and from the definition of $\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y})$, we find that for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, we have

$$\Gamma_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}) = \sqrt{\frac{n}{\sum_{i=1}^n a_{T_i}^2}} \Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) .$$

Now, by using Hoeffding's inequality, it is straightforward to show that for any $\delta \in (0, 1]$, we have

$$\Pr_{\mathcal{T} \sim \mathcal{U}(G)^n} \left(\frac{1}{n} \sum_{i=1}^n a_{T_i}^2 \leq 1 + \epsilon \right) \geq 1 - \delta/2 .$$

whenever $n \geq \frac{\ell^2}{8\epsilon} \ln\left(\frac{2}{\delta}\right)$. Since n satisfies the condition of Lemma 3, we see that it also satisfies this condition whenever $\epsilon \leq 1/2$. Hence, with probability of at least $1 - \delta/2$, we have

$$\sum_{i=1}^n a_{T_i}^2 \leq n(1 + \epsilon) .$$

Moreover Lemma 2 and Lemma 3 imply that, with probability of at least $1 - \delta/2$, we have simultaneously for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$,

$$\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y}) \geq \Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon .$$

Hence, from the union bound, with probability of at least $1 - \delta$, simultaneously $\forall (x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, we have

$$\Gamma_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x, \mathbf{y}) \geq \frac{1}{\sqrt{1 + \epsilon}} \left[\Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon \right] .$$

□

Derivation of the Primal L_2-norm Random Tree Approximation

If we introduce the usual slack variables $\xi_i \stackrel{\text{def}}{=} \gamma \cdot \mathcal{L}^{\gamma}(\Gamma_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x_i, \mathbf{y}_i))$, Theorem 5 suggests that we should minimize $\frac{1}{\gamma} \sum_{k=1}^m \xi_k$ for some fixed margin value $\gamma > 0$. Rather than performing this task for several values of γ, we can, equivalently, solve the following optimization problem for several values of $C > 0$.

$$\begin{aligned} \min_{\boldsymbol{\xi}, \gamma, \mathbf{q}, \mathcal{W}} \quad & \frac{1}{2\gamma^2} + \frac{C}{\gamma} \sum_{k=1}^m \xi_k \\ \mathbf{s.t.}: \quad & \Gamma_{\mathcal{T}}(\mathcal{W}, \mathbf{q}, x_k, \mathbf{y}_k) \geq \gamma - \xi_k, \ \xi_k \geq 0, \ \forall\, k \in \{1, \ldots, m\} , \\ & \sum_{i=1}^n q_i^2 = 1, \ q_i \geq 0, \ \|\mathbf{w}_{T_i}\|^2 = 1, \ \forall\, i \in \{1, \ldots, n\} . \end{aligned} \tag{8}$$

If we now use instead $\zeta_k \stackrel{\text{def}}{=} \xi_k / \gamma$, and $\mathbf{v}_{T_i} \stackrel{\text{def}}{=} q_i \mathbf{w}_{T_i} / \gamma$, we then have $\sum_{i=1}^n \|\mathbf{v}_{T_i}\|^2 = 1/\gamma^2$ (under the constraints of problem (8)). If $\mathcal{V} \stackrel{\text{def}}{=} \{\mathbf{v}_{T_1}, \ldots, \mathbf{v}_{T_n}\}$, optimization problem (8) is then equivalent to

$$\begin{aligned} \min_{\boldsymbol{\zeta}, \mathcal{V}} \quad & \frac{1}{2} \sum_{i=1}^n \|\mathbf{v}_{T_i}\|^2 + C \sum_{k=1}^m \zeta_k \\ \mathbf{s.t.}: \quad & \Gamma_{\mathcal{T}}(\mathcal{V}, \mathbf{1}, x_k, \mathbf{y}_k) \geq 1 - \zeta_k, \ \zeta_k \geq 0, \ \forall\, k \in \{1, \ldots, m\} . \end{aligned} \tag{9}$$

Note that, following our definitions, we now have

$$\Gamma_{\mathcal{T}}(\mathcal{V}, \mathbf{1}, x, \mathbf{y}) \quad = \quad \frac{1}{\sqrt{n}} \sum_{i=1}^n \langle \mathbf{v}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x, \mathbf{y}) \rangle - \max_{\mathbf{y}' \neq \mathbf{y}} \frac{1}{\sqrt{n}} \sum_{i=1}^n \langle \mathbf{v}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x, \mathbf{y}') \rangle .$$

We then obtain the optimization problem of Property 6 with the change of variables $\mathbf{w}_{T_i} \leftarrow \mathbf{v}_{T_i} / \sqrt{n}$, $\xi_k \leftarrow \zeta_k$, and $C \leftarrow C / \sqrt{n}$.

Lemma 7

Let $\mathbf{y}^\star_K = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}_{\mathcal{T},K}} F_{\mathcal{T}}(x,\mathbf{y})$ *be the highest scoring multilabel in* $\mathcal{Y}_{\mathcal{T},K}$*. Suppose that*

$$F_{\mathcal{T}}(x,\mathbf{y}^\star_K) \geq \frac{1}{n}\sum_{i=1}^{n} F_{\mathbf{w}_{T_i}}(x,\mathbf{y}_{T_i,K}) \stackrel{def}{=} \theta_x(K)$$

It follows that $F_{\mathcal{T}}(x,\mathbf{y}^\star_K) = \max_{\mathbf{y}\in\mathcal{Y}} F_{\mathcal{T}}(x,\mathbf{y})$.

Proof. Consider a multilabel $\mathbf{y}^\dagger \notin \mathcal{Y}_{\mathcal{T},K}$. It follows that for all T_i we have

$$F_{\mathbf{w}_{T_i}}(x,\mathbf{y}^\dagger) \leq F_{\mathbf{w}_{T_i}}(x,\mathbf{y}_{T_i,K}).$$

Hence,

$$F_{\mathcal{T}}(x,\mathbf{y}^\dagger) \;=\; \frac{1}{n}\sum_{i=1}^{n} F_{\mathbf{w}_{T_i}}(x,\mathbf{y}^\dagger) \;\leq\; \frac{1}{n}\sum_{i=1}^{n} F_{\mathbf{w}_{T_i}}(x,\mathbf{y}_{T_i,K}) \;\leq\; F_{\mathcal{T}}(x,\mathbf{y}^\star_K),$$

as required. □

Lemma 8

Let the scoring function F_T *of each spanning tree of* G *be a* σ_T*-sub-Gaussian random variable under the uniform distribution of labels;* i.e.*, for each* T *on* G*, there exists* $\sigma_T > 0$ *such that for any* $\lambda > 0$ *we have*

$$\mathop{\mathbf{E}}_{\mathbf{y}\sim\mathcal{U}(\mathcal{Y})} e^{\lambda(F_T(\mathbf{y})-\mu_T)} \;\leq\; e^{\frac{\lambda^2}{2}\sigma_T^2}\,.$$

Let $\sigma^2 \stackrel{def}{=} \mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} \sigma_T^2$*, and let*

$$\alpha \stackrel{def}{=} \Pr_{T\sim\mathcal{U}(G)}\left(\mu_T \leq \mu \;\wedge\; F_T(\hat{\mathbf{y}}) \geq F(\hat{\mathbf{y}}) \;\wedge\; \sigma_T^2 \leq \sigma^2\right).$$

Then

$$\Pr_{\mathcal{T}\sim\mathcal{U}(G)^n}\left(\exists T\in\mathcal{T}\colon\; \rho_T(\hat{\mathbf{y}}) \leq e^{-\frac{1}{2}\frac{(F(\hat{\mathbf{y}})-\mu)^2}{\sigma^2}}\right) \geq 1-(1-\alpha)^n\,.$$

Proof. From the definition of $\rho(\hat{\mathbf{y}})$ and for any $\lambda > 0$, we have

$$\begin{aligned}
\rho_T(\mathbf{y}^*) &= \Pr_{\mathbf{y}\sim\mathcal{U}(\mathcal{Y})}\left(F_T(\mathbf{y}) \geq F_T(\hat{\mathbf{y}})\right)\\
&= \Pr_{\mathbf{y}\sim\mathcal{U}(\mathcal{Y})}\left(F_T(\mathbf{y}) - \mu_T \geq F_T(\hat{\mathbf{y}}) - \mu_T\right)\\
&= \Pr_{\mathbf{y}\sim\mathcal{U}(\mathcal{Y})}\left(e^{\lambda(F_T(\mathbf{y})-\mu_T)} \geq e^{\lambda(F_T(\hat{\mathbf{y}})-\mu_T)}\right)\\
&\leq e^{-\lambda(F_T(\hat{\mathbf{y}})-\mu_T)} \mathop{\mathbf{E}}_{\mathbf{y}\sim\mathcal{U}(\mathcal{Y})} e^{\lambda(F_T(\mathbf{y})-\mu_T)} \qquad (10)\\
&\leq e^{-\lambda(F_T(\hat{\mathbf{y}})-\mu_T)} e^{\frac{\lambda^2}{2}\sigma_T^2}\,, \qquad (11)
\end{aligned}$$

where we have used Markov's inequality for line (10) and the fact that F_T is a σ_T-sub-Gaussian variable for line (11). Hence, from this equation and from the definition of α, we have that

$$\Pr_{T\sim\mathcal{U}(G)}\left(\rho_T(\hat{\mathbf{y}}) \leq e^{-\lambda(F_T(\hat{\mathbf{y}})-\mu_T)} e^{\frac{\lambda^2}{2}\sigma_T^2} \leq e^{-\lambda(F(\hat{\mathbf{y}})-\mu)} e^{\frac{\lambda^2}{2}\sigma^2}\right) \;\geq\; \alpha\,.$$

Hence,

$$\Pr_{\mathcal{T}\sim\mathcal{U}(G)^n}\left(\forall T\in\mathcal{T}\colon\; \rho_T(\hat{\mathbf{y}}) > e^{-\lambda(F(\hat{\mathbf{y}})-\mu)} e^{\frac{\lambda^2}{2}\sigma^2}\right) \;\leq\; (1-\alpha)^n\,,$$

which is equivalent to the statement of the lemma when we choose $\lambda = [F(\hat{\mathbf{y}}) - \mu]/\sigma^2$. □

The K-best Inference Algorithm

Algorithm 1 depicts the K-best inference algorithm for the ensemble of rooted spanning trees. The algorithm takes as input the collection of spanning trees $T_i \in \mathcal{T}$, the edge labeling scores

$$F_{E_{\mathcal{T}}} = \{F_{T_i,v,v'}(y_v, y_{v'})\}_{(v,v')\in E_i, y_v \in \mathcal{Y}_v, y_{v'} \in \mathcal{Y}_{v'}, T_i \in \mathcal{T}}$$

for fixed x_k and $\mathbf{w}$, the length of K-best list, and optionally (for training) also the true multilabel $\mathbf{y}_k$ for x_k.

As a rooted tree implicitly orients the edges, for convenience we denote the edges as directed $v \to pa(v)$, where $pa(v)$ denotes the parent (i.e. the adjacent node on the path towards the root) of v. By $ch(v)$ we denote the set of children of v. Moreover, we denote the subtree of T_i rooted at a node v as T_v and by $T_{v' \to v}$ the subtree consisting of $T_{v'}$ plus the edge $v' \to v$ and the node v.

The algorithm performs a dynamic programming over each tree in turn, extracting the K-best list of multilabels and their scores, and aggregates the results of the trees, retrieving the highest scoring multilabel of the ensemble, the worst violating multilabel and the threshold score of the K-best lists.

The dynamic programming is based on traversing the tree in post-order, so that children of the node are always processed before the parent. The algorithm maintains sorted K best lists of candidate labelings of the subtrees T_v and $T_{v' \to v}$, using the following data structures:

- Score matrix P_v, where element $P_v(y, r)$ records the score of the r'th best multilabel of the subtree T_v when node v is labeled as y.
- Pointer matrix C_v, where element $C_v(y, r)$ keeps track of the ranks of the child nodes $v' \in ch(v)$ in the message matrix $M_{v' \to v}$ that contributes to the score $P_v(y, r)$.
- Message matrix $M_{v \to pa(v)}$, where element $M_{v \to pa(v)}(y', r)$ records the score of r'th best multilabel of the subtree $T_{v \to pa(v)}$ when the label of $pa(v)$ is y'.
- Configuration matrix $C_{v \to pa(v)}$, where element $C_{v \to pa(v)}(y', r)$ traces the label and rank (y, r) of child v that achieves $M_{v \to pa(v)}(y', r)$.

The processing of a node v entails the following steps. First, the K-best lists of the children of the node stored in $M_{v' \to v}$ are merged in amortized $\Theta(K)$ time per child node. This is achieved by processing two child lists in tandem starting from the top of the lists and in each step picking the best pair of items to merge. This process results in the score matrix P_v and the pointer matrix C_v.

Second, the K-best lists of $T_{v \to pa(v)}$ corresponding to all possible labels y' of $pa(v)$ are formed. This is achieved by keeping the label of the head of the edge $v \to pa(v)$ fixed, and picking the best combination of labeling the tail of the edge and selecting a multilabel of T_v consistent with that label. This process results in the matrices $M_{v \to pa(v)}$ and $C_{v \to pa(v)}$. Also this step can be performed in $\Theta(K)$ time.

The iteration ends when the root v_{root} has updated its score $P_{v_{root}}$. Finally, the multilabels in form $\mathcal{Y}_{T_i,K}$ are traced using the pointers stored in C_v and $C_{v \to pa(v)}$. The time complexity for a single tree is $\Theta(K\ell)$, and repeating the process for n trees gives total time complexity of $\Theta(nK\ell)$.

Master algorithm for training the model

The master algorithm (Algorithm 2) iterates over each training example until convergence. The processing of each training example proceeds by identifying the K worst violators of each tree together with the threhold score $\theta_i = \theta_{x_i}$ (line 5), determining the worst ensemble violator from among them (line 6) and updating each tree by the worst ensemble violator (line 8). During the early stages of the algorithm, it is not essential to identify the worst violator. We therefore propose that initially $K = 2$, and the iterations continue until no violators are identified (line 7). We then increment K and continue until the condition (line 10-12) given by Lemma 7 is satisfied so that we are assured of having converged to the global optimum.

Algorithm 1 Algorithm to obtain top K multilabels on a collection of spanning trees.

$FindKBest(\mathcal{T}, F_{E_\mathcal{T}}, K, \mathbf{y}_i)$

Input: Collection of rooted spanning trees $T_i = (E_i, V_i)$,
edge labeling scores $F_{E_\mathcal{T}} = \{F_{T,v,v'}(y_v, y_{v'})\}$

Output: The best scoring multilabel $\mathbf{y}^*$, worst violator $\bar{\mathbf{y}}$, threshold θ_i

for $T_i \in \mathcal{T}$ **do**
  Initialize $P_v, C_v, M_{v\to pa(v)}, C_{v\to pa(v)}, \forall v \in V_i$
  I = nodes indices in post-order of the tree T_i
  for $j = 1 : |I|$ **do**
    $v = v_{I(j)}$
    % Collect and merge K-best lists of children
    if $ch(v) \neq \emptyset$ **then**
      $P_v(y) = P_v(y) + \underset{r_v, v' \in ch(v)}{\textbf{kmax}} \left(\sum_{v' \in ch(v)} \left(M_{v'\to v}(y, r_v) \right) \right)$
      $C_v(y) = P_v(y) + \underset{r_v, v' \in ch(v)}{\textbf{argkmax}} \left(\sum_{v' \in ch(v)} \left(M_{v'\to v}(y, r_v) \right) \right)$
    end if
    % Form the K-best list of $T_{v\to pa(v)}$
    $M_{v\to pa(v)}(y_{pa(v)}) = \underset{y,r}{\textbf{kmax}} \left(P_v(y, r) + F_{T,v\to pa(v)}(y_v, y_{pa(v)}) \right)$
    $C_{v\to pa(v)}(y_{pa(v)}) = \underset{u_v, r}{\textbf{argkmax}} \left(P_v(u_v, r) + F_{T,v\to pa(v)}(u_v, y_{pa(v)}) \right)$
  end for
  Trace back with C_v and $C_{v\to pa(v)}$ to get $\mathcal{Y}_{T_i,K}$.
end for

$$\mathcal{Y}_{\mathcal{T},K} = \bigcup_{T_i \in \mathcal{T}} \mathcal{Y}_{T_i,K}$$

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{Y}_{\mathcal{T},K}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{\substack{(v,v')= \\ e \in E_i}} F_{T_i,v,v'}(y_v, y_{v'})$$

$$\bar{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}_{\mathcal{T},K} \setminus \mathbf{y}_i}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{\substack{(v,v')= \\ e \in E_i}} F_{T_i,v,v'}(y_v, y_{v'})$$

$$\theta_i = \sum_{i=1}^{n} \sum_{\substack{(v,v')= \\ e \in E_i}} F_{T_i,v,v'}(y_{T_i,K,v}, y_{T_i,k,v'})$$

Algorithm 2 Master algorithm.

Input: Training sample $\{(x_k, \mathbf{y}_k)\}_{k=1}^m$, collection of spanning trees $\mathcal{T}$, minimum violation γ_0
Output: Scoring function $F_{\mathcal{T}}$

$K_k = 2, \forall k \in \{1, \cdots, m\}$; $\mathbf{w}_{T_i} = 0, \forall\, T_i \in \mathcal{T}$; $converged = false$
while $not(converged)$ **do**
  $converged = true$
  for $k = \{1, \ldots, m\}$ **do**
    $S_{\mathcal{T}} = \{S_{T_i,e}(k, \mathbf{u}_e) | S_{T_i,e}(k, \mathbf{u}_e) = \langle \mathbf{w}_{T_i,e}, \phi_{T_i,e}(x_k, \mathbf{u}_e) \rangle, \forall (e \in E_i, T_i \in \mathcal{T}, \mathbf{u}_e \in \mathcal{Y}_v \times \mathcal{Y}_{v'})\}$
    $[\mathbf{y}^*, \bar{\mathbf{y}}, \theta_i] = FindKBest(\mathcal{T}, S_{\mathcal{T}}, K_i, \mathbf{y}_i)$
    if $F_{\mathcal{T}}(x_i, \bar{\mathbf{y}}) - F_{\mathcal{T}}(x_i, \mathbf{y}_i) \geq \gamma_0$ **then**
      $\{\mathbf{w}_{T_i}\}_{T_i \in \mathcal{T}} = updateTrees(\{\mathbf{w}_{T_i}\}_{T_i \in \mathcal{T}}, x_i, \bar{\mathbf{y}})$
      $converged = false$
    else
      if $\theta_i > F_{\mathcal{T}}(x_i, \bar{\mathbf{y}})$ **then**
        $K_i = \min(2K_i, |\mathcal{Y}|)$
        $converged = false$
      end if
    end if
  end for
end while

Derivation of the Marginal Dual

Definition 6. Primal L_2-norm Random Tree Approximation

$$\begin{aligned} \min_{\mathbf{w}_{T_i}, \xi_k} \quad & \frac{1}{2} \sum_{i=1}^{n} ||\mathbf{w}_{T_i}||_2^2 + C \sum_{k=1}^{m} \xi_k \\ \text{s.t.} \quad & \sum_{i=1}^{n} \langle \mathbf{w}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x_k, \mathbf{y}_k) \rangle - \max_{\mathbf{y} \neq \mathbf{y}_k} \sum_{i=1}^{n} \langle \mathbf{w}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x_k, \mathbf{y}) \rangle \geq 1 - \xi_k \\ & \xi_k \geq 0, \forall\, k \in \{1, \ldots, m\}, \end{aligned}$$

where $\{\mathbf{w}_{T_i} | T_i \in \mathcal{T}\}$ are the feature weights to be learned on each tree, ξ_k is the margin slack allocated for each example x_k, and C is the slack parameter that controls the amount of regularization in the model. This primal form has the interpretation of maximizing the joint margins from individual trees between (correct) training examples and all the other (incorrect) examples.

The Lagrangian of the primal form (Definition 6) is

$$\begin{aligned} \mathcal{L}(\mathbf{w}_{T_i}, \xi, \boldsymbol{\alpha}, \boldsymbol{\beta}) \; = & \frac{1}{2} \sum_{i=1}^{n} ||\mathbf{w}_{T_i}||_2^2 + C \sum_{k=1}^{m} \xi_k - \sum_{k=1}^{m} \beta_k \xi_k \\ & - \sum_{k=1}^{m} \sum_{\mathbf{y} \neq \mathbf{y}_k} \alpha_{k,\mathbf{y}} \left(\sum_{i=1}^{n} \langle \mathbf{w}_{T_i}, \Delta \hat{\boldsymbol{\phi}}_{T_i}(x_k, \mathbf{y}_k) \rangle - 1 + \xi_k \right), \end{aligned}$$

where α_k and β_k are Lagrangian multipliers that correspond to the constraints of the primal form, and $\Delta \hat{\boldsymbol{\phi}}_{T_i}(x_k, \mathbf{y}_k) = \hat{\boldsymbol{\phi}}_{T_i}(x_k, \mathbf{y}_k) - \hat{\boldsymbol{\phi}}_{T_i}(x_k, \mathbf{y})$. Note that given a training example-label pair $(x_k, \mathbf{y}_k)$ there are exponential number of $\alpha_{k,\mathbf{y}}$ one for each constraint defined by incorrect example-label pair $(x_k, \mathbf{y})$.

Setting the gradient of Lagrangian with respect to primal variables to zero, we obtain the following equalities:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}_{T_i}} &= \mathbf{w}_{T_i} - \sum_{k=1}^{m} \sum_{\mathbf{y} \neq \mathbf{y}_k} \alpha_{k,\mathbf{y}} \Delta \hat{\boldsymbol{\phi}}_{T_i}(x_k, \mathbf{y}_k) = 0, \\ \frac{\partial \mathcal{L}}{\partial \xi_k} &= C - \sum_{\mathbf{y} \neq \mathbf{y}_k} \alpha_{k,\mathbf{y}} - \beta_k = 0, \end{aligned}$$

which give the following dual optimization problem.

Definition 10. Dual L_2-norm Random Tree Approximation

$$\begin{aligned} \underset{\boldsymbol{\alpha}\geq 0}{\textbf{\textit{max}}} \quad & \boldsymbol{\alpha}^\intercal \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\intercal \left(\sum_{i=1}^{n} K_{T_i}\right)\boldsymbol{\alpha} \\ \textbf{\textit{s.t.}} \quad & \sum_{\mathbf{y}\neq\mathbf{y}_k} \alpha_{k,\mathbf{y}} \leq C, \forall\, k \in \{1,\ldots,m\}, \end{aligned}$$

where $\boldsymbol{\alpha} = (\alpha_{k,\mathbf{y}})_{k,\mathbf{y}}$ is the vector of dual variables. The joint kernel

$$\begin{aligned} K_{T_i}(x_k,\mathbf{y};x_{k'},\mathbf{y}') &= \langle \hat{\boldsymbol{\phi}}_{T_i}(x_k,\mathbf{y}_k) - \hat{\boldsymbol{\phi}}_{T_i}(x_k,\mathbf{y}), \hat{\boldsymbol{\phi}}_{T_i}(x_{k'},\mathbf{y}_{k'}) - \hat{\boldsymbol{\phi}}_{T_i}(x_{k'},\mathbf{y}')\rangle \\ &= \langle \varphi(x_k),\varphi(x_{k'})\rangle_\varphi \cdot \langle \psi_{T_i}(\mathbf{y}_k) - \psi_{T_i}(\mathbf{y}), \psi_{T_i}(\mathbf{y}_{k'}) - \psi_{T_i}(\mathbf{y}')\rangle_\psi \\ &= K^\varphi(x_k,x_{k'}) \cdot \Big(K^\psi_{T_i}(\mathbf{y}_k,\mathbf{y}_{k'}) - K^\psi_{T_i}(\mathbf{y}_k,\mathbf{y}') - K^\psi_{T_i}(\mathbf{y},\mathbf{y}_{k'}) + K^\psi_{T_i}(\mathbf{y},\mathbf{y}')\Big) \\ &= K^\varphi(x_k,x_{k'}) \cdot K^{\Delta\psi}_{T_i}(\mathbf{y}_k,\mathbf{y};\mathbf{y}_{k'},\mathbf{y}') \end{aligned}$$

is composed by input kernel K^φ and output kernel $K^\psi_{T_i}$ defined by

$$\begin{aligned} K^\varphi(x_k,x_{k'}) &= \langle \varphi(x_k),\varphi(x_{k'})\rangle_\varphi \\ K^{\Delta\psi}_{T_i}(\mathbf{y}_k,\mathbf{y};\mathbf{y}_{k'},\mathbf{y}') &= K^\psi_{T_i}(\mathbf{y}_k,\mathbf{y}_{k'}) - K^\psi_{T_i}(\mathbf{y}_k,\mathbf{y}') - K^\psi_{T_i}(\mathbf{y}_{k'},\mathbf{y}) + K^\psi_{T_i}(\mathbf{y},\mathbf{y}'). \end{aligned}$$

To take advantage of the spanning tree structure in solving the problem, we further factorize the dual (Definition 10) according to the output structure [9, 16]. by defining a marginal dual variable μ as

$$\mu(k,e,\mathbf{u}_e) = \sum_{\mathbf{y}\neq\mathbf{y}_k} \mathbf{1}_{\{\psi(\mathbf{y})=\mathbf{u}_e\}}\boldsymbol{\alpha}_{k,\mathbf{y}},$$

where $e=(j,j')\in E$ is an edge in the output graph and $\mathbf{u}_e \in \mathcal{Y}\times\mathcal{Y}'$ is a possible label of edge e. As each marginal dual variable $\mu(k,e,\mathbf{u}_e)$ is the sum of a collection of dual variables $\alpha_{k,\mathbf{y}}$ that has consistent label $(u_j,u_{j'})=\mathbf{u}_e$, we have the following

$$\sum_{\mathbf{u}_e} \mu(k,e,\mathbf{u}_e) = \sum_{\mathbf{y}\neq\mathbf{y}_k} \boldsymbol{\alpha}_{k,\mathbf{y}} \tag{12}$$

for an arbitrary edge e, independently of the structure of the trees.

The linear part of the objective (Definition 10) can be stated in term of $\boldsymbol{\mu}$ for an arbitrary collection of trees as

$$\boldsymbol{\alpha}^\intercal\mathbf{1} = \sum_{k=1}^{m}\sum_{\mathbf{y}\neq\mathbf{y}_k}\alpha_{k,\mathbf{y}} = \frac{1}{|E_\mathcal{T}|}\sum_{k=1}^{m}\sum_{e\in E_\mathcal{T}}\sum_{\mathbf{u}_e}\mu(k,e,\mathbf{u}_e) = \frac{1}{|E_\mathcal{T}|}\sum_{e,k,\mathbf{u}_e}\mu(k,e,\mathbf{u}_e)\,,$$

where edge $e=(j,j')\in E_\mathcal{T}$ appearing in the collection of trees $\mathcal{T}$.

We observe that the label kernel of tree T_i, $K^\psi_{T_i}$, decomposes on the edges of the tree as

$$K^\psi_{T_i}(\mathbf{y},\mathbf{y}') = \langle \mathbf{y},\mathbf{y}'\rangle_\psi = \sum_{e\in E_i}\langle y_e,y'_e\rangle_\psi = \sum_{e\in E_i} K^{\psi,e}(y_e,y'_e).$$

Thus, the output kernel $K^{\Delta\psi}_{T_i}$ and the joint kernel K_{T_i} also decompose

$$\begin{aligned} K^{\Delta\psi}_{T_i}(\mathbf{y}_k,\mathbf{y};\mathbf{y}_{k'},\mathbf{y}') &= \Big(K^\psi_{T_i}(\mathbf{y}_k,\mathbf{y}_{k'}) - K^\psi_{T_i}(\mathbf{y}_k,\mathbf{y}') - K^\psi_{T_i}(\mathbf{y}_{k'},\mathbf{y}) + K^\psi_{T_i}(\mathbf{y},\mathbf{y}')\Big) \\ &= \sum_{e\in E_i}\Big(K^{\psi,e}_{T_i}(y_{ke},y_{k'e}) - K^{\psi,e}_{T_i}(y_{ke},y'_e) - K^{\psi,e}_{T_i}(y_e,y_{k'e}) + K^{\psi,e}_{T_i}(y_e,y'_e)\Big) \\ &= \sum_{e\in E_i} K^{\Delta\psi,e}_{T_i}(y_{ke},y_e;y_{k'e},y'_e), \\ K_{T_i}(x_k,\mathbf{y};x_{k'},\mathbf{y}') &= K^\psi(x_k,x_{k'})\cdot K^{\Delta\psi}_{T_i}(\mathbf{y}_k,\mathbf{y};\mathbf{y}_{k'},\mathbf{y}') \\ &= K^\psi(x_k,x_{k'})\cdot\sum_{e\in E_i} K^{\Delta\psi,e}(y_{ke},y_e;y_{k'e},y'_e) \\ &= \sum_{e\in E_i} K^e(x_k,y_e;x_{k'},y'_e). \end{aligned}$$

The sum of joint kernels of the trees can be expressed as

$$\begin{aligned}\sum_{i=1}^{n} K_{T_i}(x_k, \mathbf{y}; x_{k'}, \mathbf{y}') &= \sum_{i=1}^{n} \sum_{e \in E_i} K^e(x_k, y_e; x_{k'}, y'_e) \\ &= \sum_{e \in E_{\mathcal{T}}} \sum_{\substack{T_i \in \mathcal{T}: \\ e \in E_i}} K^e(x_k, y_e; x_{k'}, y'_e) \\ &= \sum_{e \in E_{\mathcal{T}}} N_{\mathcal{T}}(e) K^e(x_k, y_e; x_{k'}, y'_e)\end{aligned}$$

where $N_{\mathcal{T}}(e)$ denotes the number of occurrences of edge e in the collection of trees $\mathcal{T}$.

Taking advantage of the above decomposition and of the Equation (12) the quadratic part of the objective (Definition 10) can be stated in term of $\boldsymbol{\mu}$ as

$$\begin{aligned}& -\frac{1}{2}\boldsymbol{\alpha}^\intercal \left(\sum_{i=1}^{n} K_{T_i}\right)\boldsymbol{\alpha} \\ =& -\frac{1}{2}\boldsymbol{\alpha}^\intercal \left(\sum_{e \in E_{\mathcal{T}}} N_{\mathcal{T}}(e) K^e(x_k, \mathbf{y}; x_{k'}, \mathbf{y}')\right)\boldsymbol{\alpha} \\ =& -\frac{1}{2}\sum_{k,k'=1}^{m} \sum_{e \in E_{\mathcal{T}}} N_{\mathcal{T}}(e) \sum_{\substack{\mathbf{y} \neq \mathbf{y}_k \\ \mathbf{y}' \neq \mathbf{y}_{k'}}} \alpha(k, \mathbf{y}) K^e(x_k, y_e; x_{k'}, y'_e)\alpha(k', \mathbf{y}') \\ =& -\frac{1}{2}\sum_{k,k'=1}^{m} \sum_{e \in E_{\mathcal{T}}} N_{\mathcal{T}}(e) \sum_{\mathbf{u}_e, \mathbf{u}'_e} \sum_{\substack{\mathbf{y} \neq \mathbf{y}_k : y_e = \mathbf{u}_e \\ \mathbf{y}' \neq \mathbf{y}_{k'} : y'_e = \mathbf{u}'_e}} \alpha(k, \mathbf{y}) K^e(x_k, \mathbf{u}_e; x_{k'}, \mathbf{u}'_e)\alpha(k', \mathbf{y}') \\ =& -\frac{1}{2}\sum_{k,k'=1}^{m} \sum_{e \in E_{\mathcal{T}}} \frac{N_{\mathcal{T}}(e)}{|E_{\mathcal{T}}|^2} \sum_{\mathbf{u}_e, \mathbf{u}'_e} \mu(k, e, \mathbf{u}_e) K^e(x_k, \mathbf{u}_e; x_{k'}, \mathbf{u}'_e)\mu(k', e, \mathbf{u}'_e) \\ =& -\frac{1}{2}\sum_{\substack{e,k,\mathbf{u}_e, \\ k',\mathbf{u}'_e}} \mu(k, e, \mathbf{u}_e)\, K^e_{\mathcal{T}}(x_k, \mathbf{u}_e; x_{k'}, \mathbf{u}'_e)\, \mu(k', e, \mathbf{u}'_e),\end{aligned}$$

where $E_{\mathcal{T}}$ is the union of the sets of edges appearing in $\mathcal{T}$.

We then arrive at the following definition.

Definition 9. Marginalized Dual L_2-norm Random Tree Approximation

$$\max_{\boldsymbol{\mu} \in \mathcal{M}^m} \quad \frac{1}{|E_{\mathcal{T}}|}\sum_{e,k,\mathbf{u}_e} \mu(k, e, \mathbf{u}_e) - \frac{1}{2}\sum_{\substack{e,k,\mathbf{u}_e, \\ k',\mathbf{u}'_e}} \mu(k, e, \mathbf{u}_e) K^e_{\mathcal{T}}(x_k, \mathbf{u}_e; x'_k, \mathbf{u}'_e)\mu(k', e, \mathbf{u}'_e)\,,$$

where $\mathcal{M}^m$ is marginal dual feasible set defined as (*c.f.*, [9])

$$\mathcal{M}^m = \left\{ \boldsymbol{\mu} \,|\, \mu(k, e, \mathbf{u}_e) = \sum_{\mathbf{y} \neq \mathbf{y}_k} \mathbf{1}_{\{\mathbf{y}_{ke} = \mathbf{u}_e\}} \boldsymbol{\alpha}(k, \mathbf{y})\,,\ \forall (k, e, \mathbf{u}_e) \right\}.$$

The feasible set is composed of a Cartesian product of m identical polytopes $\mathcal{M}^m = \mathcal{M} \times \cdots \times \mathcal{M}$, one for each training example. Furthermore, each $\boldsymbol{\mu} \in \mathcal{M}$ corresponds to some dual variable $\boldsymbol{\alpha}$ in the original dual feasible set $\mathcal{A} = \{\boldsymbol{\alpha} | \alpha(k, \mathbf{y}) \geq 0, \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha(k, \mathbf{y}) \leq C, \forall k\}$.

Experimental Results

Table 2 provides the standard deviation results of the prediction performance results of Table 1 for each algorithm in terms of the microlabel and 0/1 error rates. Values are obtained by five fold cross-validation.

DATASET	MICROLABEL LOSS (%)					0/1 LOSS (%)				
	SVM	MTL	MMCRF	MAM	RTA	SVM	MTL	MMCRF	MAM	RTA
EMOTIONS	1.9	1.8	0.9	1.4	0.6	3.4	3.5	3.1	4.2	1.5
YEAST	0.7	0.5	0.6	0.5	0.6	2.8	1.0	1.5	0.4	1.2
SCENE	0.3	0.5	0.3	0.1	0.3	1.4	3.6	1.2	0.9	0.6
ENRON	0.2	0.2	0.2	0.2	0.2	0.3	0.4	2.8	2.3	0.9
CAL500	0.3	0.3	0.3	0.2	0.4	0.0	0.0	0.0	0.0	0.0
FINGERPRINT	0.3	0.6	0.6	0.3	0.6	0.7	0.0	0.5	0.6	1.3
NCI60	0.7	0.6	1.3	0.9	1.6	1.3	2.0	1.4	1.2	2.2
MEDICAL	0.0	0.1	0.1	0.1	0.2	2.1	2.3	3.3	2.5	3.6
CIRCLE10	0.9	0.7	0.3	0.4	0.3	3.8	3.4	2.1	3.5	1.7
CIRCLE50	0.5	0.5	0.3	0.3	0.6	2.0	3.3	4.5	5.5	2.2

Table 2: Standard deviation of prediction performance for each algorithm in terms of microlabel loss and 0/1 loss.