Preface

Here is the preface Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Espoo, October 27, 2014,

Hongyu Su

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*, Volume, issue, pages, and other detailed information, Month Year.

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 *Proceedings of the* 28^{th} *Advances in Neural Information Processing Systems (NIPS 2014)*, to appear, December 2014.

Author's Contribution

Publication I: "Structured Prediction of Network Response"

Publication I presents a novel formalism of the network response prediction problem, and proposes a structured output prediction algorithm for the problem. The algorithm captures the contextual information and achieve a superior performance compared to several the state-of-the-art models.

The initial modeling idea was developed jointly by the authors. The problem was jointly formulated. The optimization algorithm was mainly contributed by the author and Prof. Rousu. Prof. Gionis had major contribution to the idea of SDP inference. The author implemented the algorithm, designed and conducted the experiment, and analyzed the results. All author participated in designing and writing the article.

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

The novelty of Publication II is to develop a structured output prediction model and apply it on drug-bioactivity prediction problem. The advanced model can captures the interdependency between cancer cell line targets. Thus the model enjoys better prediction performance compared to previous single-task prediction models.

The original modeling idea was jointly developed. The author contribute to the designing of the experiments and executing the algorithm. All author participated in writing the articles.

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

Publication III presents an ensemble approach for structured output prediction problem with application in molecular activity prediction. The model extends Publication II by tackling the situation where the structure of the output variables is not known, which largely widens the applicability of the structured output prediction models.

The idea of the ensemble learning strategy was jointly developed. In addition, the author contributed mainly to the algorithm design and implementation. The author performed the experiments and analyzed the results. The article was structured by authors and was written jointly.

Publication IV: "Multilabel Classification through Random Graph Ensembles"

Publication IV generalizes Publication III in several ways. By extending Publication III, the article introduces two other elegant aggregation frameworks for multi-task structured output prediction. A theory is also developed in this article which explains the performance improvement of the proposed model. It also extends the field of application from only molecular activity prediction problem to a set of heterogeneous multi-task prediction problems.

The original idea of generating aggregation strategy for structured output prediction was developed jointly by authors. The author designed and implemented the algorithms that bring the idea into practice. The theory part of the article is mainly contributed by the author. The article was structured by authors and was written jointly.

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 the joint learning and inference framework and developing rigours learning theory to backup the algorithm.

This article is the main outcome of author's research visit to University College, London. The idea was initialzed jointly by the authors prior to the visit. The theory is mostly contributed by Prof. Marchand, where

the author also joint the discussion and the development. The author contribute mainly to the designing and the implementation of the learning and optimization framework. The author conducted the experiments and analyze the results. All authors participated in writing the article.

List of Symbols

The author has made effort to declare and clarify the following mathematical symbols and notations before entering the main text of this thesis. This is to make sure symbols and notations are correct and consistent throughout the thesis. On the other hand, sometimes they might look different from how they were defined and used in the original research articles.

α_i	Dual variable corresponding to the i'th constraint.
δ	Width parameter of Gaussian function.
$\ell(\mathbf{y}_i, \mathbf{y}_j)$	Loss function defined on label $\mathbf{y}_i$ and label $\mathbf{y}_j$.
η	Learning rate.
$\mathcal{F}$	Input feature space.
γ	Margin as geometric distance of the data point to hyperplane.
γ_G	Loss scaling function based on graph G.
$\mathcal{G}$	A random sample of random output graphs.
$\mathcal{H}$	Hypothesis class.
$[\![\cdot]\!]$	Indicator function.
$\langle \cdot, \cdot \rangle$	Inner product.
μ	Marginalized dual variables.
$\boldsymbol{\phi}(x, \mathbf{y})$	Joint feature map of example and label pair $(x, \mathbf{y})$.
$\boldsymbol{\phi}_{T_t}(\mathbf{x}, \mathbf{y})$	Output feature map for the t'th random spanning tree.
$\psi(x, \mathbf{y})$	Compatibility score of $(x, \mathbf{y})$.

$\psi_e(x, \mathbf{y}_e)$	Local edge potential/compatibility score of edge e with label $\mathbf{y}_e$.
$\psi_e^{(t)}(x, \mathbf{u}_e)$	Local compatibility score of edge e with labeled $\mathbf{u}_e$ from the t'th base classifier.
$\boldsymbol{\psi}_E(x)$	Collection of local edge potentials/compatibility scores from all base classifiers.
ρ	The norm of the feature weight vectors.
$\mathbb{R}$	The set of real number.
$\mathcal{S}$	The set of training examples.
$\tilde{\boldsymbol{\psi}}(x)$	Collection of all node max-marginals from all base classifiers.
$\tilde{\boldsymbol{\psi}}^{(t)}(x)$	Collection of all node max-marginals from the t'th base learner.
$\tilde{\psi}_j(x, u_j)$	Maximal marginal of node j with label u_j.
$\tilde{E}$	Edge set of the consensus graph.
$\tilde{G}$	Consensus graph by pooling edges from all spanning trees.
$^\intercal$	Matrix transpose operation.
$\boldsymbol{\Upsilon}(\mathbf{y})$	Output feature map of multilabel $\mathbf{y}$ on output graph.
$\boldsymbol{\varphi}(\mathbf{x})$	Input feature map on input example $\mathbf{x}$.
$\mathbf{e}$	Matrix of one.
$\mathbf{I}$	Identity matrix.
$\mathbf{u}_e$	Possible edge label.
$\mathbf{w}$	Feature weight vector.
$\mathbf{w}_{T_t}$	Feature weight parameters on the t'th random spanning tree T_t.
$\mathbf{x}_i$	The i'th example from input space.
$\mathbf{y}[i]$	The i'th microlabel in multilabel vector $\mathbf{y}$.
$\mathbf{y}_e$	The label of the edge e.
$\mathbf{y}_i$	The i'th multilabel from multi-task output space.
$\boldsymbol{\mathcal{Y}}$	Multi-task output space.

$\boldsymbol{\mathcal{Y}}_e$	Edge label space.
$\mathcal{X}$	Input space.
ξ_i	Margin error of the i'th example.
$\mathcal{Y}$	Single-task output space.
A	Adjacency matrix of graph G
b	Bias term.
C	Margin slack parameter.
D	Geometric distance.
d	The dimension of the input feature.
E	Edge set of the graph G
$E_\times$	Edge set of product graph $G_\times$.
f	One mapping function from hypothesis class.
G	Output graph.
$G_\times$	Product graph.
K	Kernel function.
k	The total number of microlabel in the multilabel vector.
m	The number of training examples.
T	The total number of base classifer.
t	Index for the t'th base classifier in ensemble.
u_i	Possible node label.
V	Vertex set of the graph G
$V_\times$	Vertex set of product graph $G_\times$.
y_i	The i'th label from single-task output space.

1. Introduction

1.1 Scope of the Thesis

In this thesis, we study the *supervised learning* in structured output space, also known as *structured output prediction*.

Machine learning, an active research field in information and computer science, has drawn enormous attentions during the last few decades, not only because it develops intelligent systems that generalize well from previously observed examples; but also because it is rooted from rich statistical learning theories that prove the feasibility and guarantee the performance. Machine learning has in practice provided lots of inventions, that people may use many times a day without noticing, from web searching engine to self-driving car, from optical image recognizer to spam filter, from automatic stock exchange to online recommender system.

Supervised learning, one of the most important fields in machine learning, is usually instantiated as learning a function that is capable of predicting the best value for the output variable y given an observed input variable x. The function is learned by utilizing a set of observed input/output pairs known as training examples. In the classical supervised learning setting, the value that the output variable y can take is often no more than one discrete or continuous number. This is often referred as *single-task classification* in machine learning community. Many well-designed learning algorithms have been delivered for the problem and achieved remarkable success in many applications.

However, many real world problems require multiple interdependent output variables defined in structured output space to be predicted at the same time, known as *multi-task structured output prediction*. Otherwise, the problem can be summarized as learning a function for mapping from

arbitrary input to arbitrary output. In this case, single-task classification approach renders inefficiency and incapability of modeling the interdependency. In addition, the interdependency can be given either explicitly as the output graph that connecting multiple output variables (e.g. sequence parsing, network influence prediction, hierarchical document classification), or implicitly with the only assumption that the output variable are correlated (e.g. molecular classification, computer vision). Nevertheless, it is crucial to capture the common properties shared between similar output variables in order to achieve good performance and to generalize well.

Most existing methods for structured output prediction do not tackle the situation when the output graph structure is not observed. Especially, the learning models built on Markov network over multiple output variables will break down without the observed output structure. On the other hand, methods that do not explicitly assume the output structure usually have less power of modeling the interdependency and often struggle in optimization due to the exponential sized searching space.

In this thesis, we investigate the potentials of developing novel structured output prediction models to improve the prediction performance in real world multi-task classification problems (e.g. molecular activity prediction, network influence prediction). In particular, we present the novel structured output prediction models that are capable of learning without predefined structure. We focus on the efficiency and the scalability of the proposed structured output prediction models, as the inference problem is often difficult to solve. Additionally, our new models are well motivated. The theoretical study is able to explain the behaviour and to guarantee the performance of the models. In particular, our research questions can be casted as followings:

Q I: Is the structured output prediction model more suitable for predicting multiple interdependent variables in structured output space?

Q II: How to tackle the problem of structured output learning without observed output structure?

Q III: What is the motivation of the proposed structured output learning framework? Can we explain the improvement and guarantee the empirical risk of the proposed models?

Q IV: Can we efficiently optimize the proposed structured output prediction models?

1.2 Contribution and Outline of the Thesis

The main contribution of the thesis is to present a learning framework for structured output prediction with both empirical and theoretical guarantee. Under the framework, several learning algorithms are proposed. Unlike previous structured output learning algorithms, the proposed models do not assume any observed structured on the output variables, and thus largely widen the applicability of the structured output learning. Meanwhile, the thesis presents several attractive theoretical properties which support the new learning framework and guarantee the generalization error. In addition to the main contribution, the thesis also proposes a new learning algorithm that is capable of predicting a directed acyclic graph and demonstrating the performance on the influence network prediction problem. Besides, the thesis also focuses on elegant optimization strategy with which the proposed models often avoid the exponential parameter space and the $\mathcal{NP}$-hard inference problem.

The thesis is structured as follows. In Chapter 1, the scope of the research is briefly explained with several research questions being proposed which will be answered in this thesis. Chapter **??** provides the introduction to the problem settings, learning algorithms, and statistical models that are necessary to understand the rest part of the thesis. The chapter starts by introducing in Section **??** two fundamental concepts, empirical risk minimization and regularization, in statistical machine learning. Section **??** presents several single task classification algorithms, including perceptron (Section 2.2.2), logistic regression (Section 2.2.3), and SVM (Section 2.2.4). Section **??** will extend the previous section by introducing the learning algorithms designed for multiple interdependent variables in structured output space. Ensemble models for both single-task and multi-task classification will be briefly covered in Section **??**. Section **??** describes the performance measures and the statistical models that are used for model evaluation and comparison.

Chapter **??** introduces the novel statistical models developed by the author. In particular, Section 4.3 explains the structured output learning model for molecular activity prediction. Section 5.1 and Section 5.2 discuss three ensemble models (MVE, AMM, MAM) that tackle the problem in structured output prediction when the output structure is unknown. Section 5.3 presents RTA model that is a major step forward of MAM by performing joint learning and inference over a random sample of span-

ning trees. Section 4.4 introduces network influence prediction problem and presents SPIN that is capable of predicting a directed acyclic graph. Finally, Chapter 8 concludes the thesis by answering the research questions and pointing out future directions.

In this thesis, the ideas and the formalisms of the proposed statistical models for structured output prediction are explained. However, the technical details from the original publications are usually not repeated. Furthermore, the notations and the presentations of some proposed models are slightly improved to be better incorporated into a unified framework. The background information and the related work of some proposed models are sometimes elaborated in the thesis to help better understand their contributions to the community. Empirical evaluations of the proposed models are in nowhere restated, rather, they can be found from the original research articles.

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* and *regularization*. Together they give rise to the learning algorithms that will be presented in the following part of the thesis.

2.1.1 Empirical Risk Minimization

Without loss of generality, we assume two random variable $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ and $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$ are jointly distributed according to some fixed but unknown probability distribution $P(\mathbf{x}, \mathbf{y})$ over domain $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$. In addition, we are provided with examples in pair $(\mathbf{x}, \mathbf{y}) \in \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$ generated by sampling according to the distribution $P(\mathbf{x}, \mathbf{y})$. A *hypothesis class* $\mathcal{H}$ is a set of functions that the learning algorithm is allowed to search against. The basic goal for *statistical learning* is to provide from a hypothesis class $\mathcal{H}$ an *estimator* $f \in \mathcal{H} : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ that predicts the value of $\mathbf{y}$ given an arbitrary input $\mathbf{x}$.

To measure the goodness of the estimator, we define a risk function (loss function) $\mathcal{L}(\mathbf{y}, f(\mathbf{x}))$ between the true value $\mathbf{y}$ and the predicted value $f(\mathbf{x})$. The *true risk* of the estimator f over all examples from domain $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$ can be computed from

$$\mathcal{R}(f) = \int_{(\mathbf{x},\mathbf{y}) \in \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}} \mathcal{L}(\mathbf{y}, f(\mathbf{x})) P(\mathbf{x}, \mathbf{y}) \, d_{\mathbf{x}} d_{\mathbf{y}}. \tag{2.1}$$

As a results, we should search for an estimator f that minimizes the true risk defined by (2.1). However, the distribution $P(\mathbf{x}, \mathbf{y})$ that generates the examples is unknown by which it is impossible to compute (2.1) directly. Instead, we are given a random sample of m examples, denoted by $S = \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_m, \mathbf{y}_m)\}$, called *training data*. *Empirical risk* is defined as

the average error made by estimator f on the training data S of finite size

$$\mathcal{R}_{emp}(f) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\mathbf{y}_i, f(\mathbf{x}_i)). \tag{2.2}$$

This suggests the learning algorithm should search for an estimator f to minimize the empirical risk defined in (2.2). The strategy is known in machine learning community as *empirical risk minimization* (Vapnik, 1992).

2.1.2 Regularized Learning

The empirical risk minimization strategy is ill-posed which in general can provide infinity number of estimators with same empirical risk on the same training data. Besides, it often leads to *overfitting* when the dimensionality of the feature space is high and the number of training data is limited. In other words, underlying true distribution $P(\mathbf{x}, \mathbf{y})$ is difficult to to estimate from a finite sample of training examples. Thus, the estimator will generalize poorly on unseen test examples. The *regularization theory* provides a framework to alleviate these two difficulties. In particular, it suggests to minimize

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

where $\Omega(f)$ is the regularization function that controls the complexity of the estimator f by penalizing the norm of the feature weight vector.

In terms of the linear function class, there are varying way to defined the regularization term $\Omega(f)$ among which L_1-norm and L_2-norm regularization are most popular. The L_2-norm regularization, defined by

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

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), Support Vector Machines (Cortes and Vapnik, 1995). The L_1-norm regularization, defined by

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

provides sparse parameter estimation which means that we obtain a high dimensional weight vector with many zero entries. This is a attractive

property as the feature selection is incorporated into the learning and the resulting model often enjoys a high interpretability. The L_1-norm regularization has been applied in, for example, LASSO (Tibshirani, 1994). Besides, other regularization techniques have been widely studied, to name a few, $L_{1,2}$-norm regularization (Argyriou et al., 2007) and 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*. The author will explain three prominent algorithms in this area: Perceptron, Logistic Regression and Support Vector Machines. The optimization techniques and the latest advances of these algorithms will also be briefly discussed. The goal is to cover the background that is necessary to understand the algorithms presented in the later part of the dissertation.

2.2.1 Preliminary and Notations

We consider supervised learning problem which assumes an arbitrary input space $\boldsymbol{\mathcal{X}}$, an output space $\mathcal{Y}$, and the training samples coming in pairs $(\mathbf{x}_i, y_i) \in \boldsymbol{\mathcal{X}} \times \mathcal{Y}$. The definition of the output space will decide the type of the learning problem. For example, we obtain multiclass classification problem by setting $\mathcal{Y} = \{1, \cdots, K\}$ and regression problem by setting $\mathcal{Y} = \mathbb{R}$ where $\mathbb{R}$ is the set of real numbers. In this chapter we focus on standard supervised learning problem also known as *binary classification*, we explicitly assume the output space $\mathcal{Y} = \{-1, +1\}$ (instead of $\mathcal{Y} = \{0, +1\}$). Additionally, we assume a feature map $\boldsymbol{\varphi} : \boldsymbol{\mathcal{X}} \to \mathcal{F}$, which embeds the input into some high dimensional feature space $\mathcal{F} = \mathbb{R}^d$. In particular, $\boldsymbol{\varphi}(\mathbf{x})$ is a real value vector of d dimension. Given a training set of m training examples $\mathcal{S} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m)\}$, the goal is to learn from a *hypothesis class* $\mathcal{H}$ a mapping function $f \in \mathcal{H} : \boldsymbol{\mathcal{X}} \to \mathcal{Y}$ that maps an input $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ to an output $y \in \mathcal{Y}$.

We consider the hypothesis class to be a set of *linear classifiers* that are parameterized by the weight vector $\mathbf{w}$ and the bias term b defined as

$$f(\mathbf{x}; \mathbf{w}, b) = \langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle + b, \tag{2.3}$$

where by $\langle \cdot, \cdot \rangle$ we denote the inner product of two vectors

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

In addition, for any $1 \leq \rho \in \mathbb{R}$ we defined the L_ρ-norm of vector $\mathbf{w}$ as

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

We use $||\mathbf{w}||$ to denote the L_2-norm of vector $\mathbf{w}$ in the following part of the thesis for the convenience of the presentation.

2.2.2 Perceptron

The Rosenblatt's Perceptron algorithm (Rosenblatt, 1958, 1962) is one type of linear classifiers that is parameterized by a weight vector $\mathbf{w}$ and a bias term b. The decision boundary, given by

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

will separate the data into two classes. The goal of the Perceptron learning is to obtain $\mathbf{w}$ and b by minimizing the distance of misclassified training examples to the decision boundary.

Assume data point $(\mathbf{x}_i, y_i)$ is misclassified, the signed geometric distance from $\mathbf{x}_i$ to decision boundary can be computed according to

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

Denote by $\boldsymbol{\mathcal{X}}^m$ the set of misclassified examples, the objective function of Perceptron is defined as

Definition 1. *Perceptron Objective Function.*

$$\underset{\mathbf{w},b}{\boldsymbol{min}} \quad D(\mathbf{w}, b) = \underset{\mathbf{w},b}{\boldsymbol{min}} \left(- \sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}^m} y_i [\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}_i) \rangle + b] \right).$$

To optimize (Definition 1), we first assume $\boldsymbol{\mathcal{X}}^m$ is fixed and compute the partial gradient with respect to parameter $\mathbf{w}$ and b

$$\frac{\partial D(\mathbf{w}, b)}{\partial \mathbf{w}} = - \sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}^m} y_i \mathbf{x}_i, \quad \frac{\partial D(\mathbf{w}, b)}{\partial b} = - \sum_{\mathbf{x}_i \in \boldsymbol{\mathcal{X}}^m} y_i.$$

In practice, the Perceptron algorithm uses stochastic gradient descent that iteratively processes one example from training data at a time. In each step, the algorithm makes sure the current $\mathbf{w}$ and b will correctly

separate the current training example. Once a misclassified example is visited, it adjusts the parameters according to the update rule

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i, \quad b \leftarrow b + \eta y_i,$$

where η is the perceptron learning rate.

Theory 1 (Perceptron Convergence Theorem (Block, 1962; Novikoff, 1962))**.** *Given a sequence of training examples in pairs* $\mathcal{S} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m)\}$. *For simplicity, we consider a perceptron without bias term* b. *Assume* $||\mathbf{x}_i|| \leq R$ *for all* $i \in \{1, \cdots, m\}$. *Suppose for some* $\gamma > 0$ *there exists a unit length vector* $||\hat{\mathbf{w}}|| = 1$ *such that* $y_i(\langle \hat{\mathbf{w}}, \boldsymbol{\varphi}(\mathbf{x}_i)\rangle) \geq \gamma$ *holds for all* i *(training data* $\mathcal{S}$ *is linearly separable). Then the number of mistakes the Perceptron algorithm makes on the sequence of training data* $\mathcal{S}$ *is at most* R^2/γ^2.

Theory 1 shows that if the data are linearly separable, the perceptron algorithm will make a finite number of mistakes. In particular, the perceptron algorithm will iterate through the training set and will converge to a vector that well separates all training examples. Moreover, the number of mistakes will be bounded by the minimum gap between the positive and negative examples. That is the algorithm converges quickly when the gap γ is big. We have to keep in mind that the convergence theorem does not say anything about the quality of the solution, though it is built based on the assumption that there is a "gap" between positive and negative examples.

The standard Perceptron algorithm supposes that the data are linearly separable, iterates over the training set, and eventually converges. However, the setting is less interesting for most applications where the data might not be linearly separable or it takes too long for the algorithm to converge. Therefore, we need to find out the best decision rule given a set of updates.

Voted Perceptron developed in (Freund and Schapire, 1999) is a straight forward modification of the standard Perceptron. Voted Perceptron algorithm keeps track all weight vectors that appears during training phase and their weights. The weight can be seen as the "survival time" of the weight vector during the Perceptron training. The prediction for a new test example $\mathbf{x}_{ts}$ is computed by

$$y_{ts} = \mathbf{sign}\left(\sum_{l=1}^{k} c^l (\mathbf{sign}\langle \mathbf{w}^l, \boldsymbol{\varphi}(\mathbf{x}_{ts})\rangle + b^l)\right) \tag{2.4}$$

using the weight vectors $\mathbf{w}^l$ and the corresponding weights c^l obtained during learning. It is proved that Voted Perceptron (2.4) guarantees to achieve better generalization error than original Perceptron (Definition 1). However, Voted Perceptron does not work in practice as it needs to store all weight vectors encountered during training.

Averaged Perceptron is a simple alternative, where in stead of storing all weight vectors the algorithm only keeps the weighted sum $\sum_{l=1}^{k} c^l \mathbf{w}^l$. The prediction for a new test example $\mathbf{x}_{ts}$ is computed by

$$y_{ts} == \mathbf{sign}\left(\langle \sum_{l=1}^{k} c^l \mathbf{w}^l, \boldsymbol{\varphi}(\mathbf{x}_{ts})\rangle + \sum_{l=1}^{k} c^l b^l\right),$$

which is more efficient to compute and is extremely similar to (2.4) except for the inside **sign** operation.

2.2.3 Logistic Regression (LR)

Logistic Regression is another important method in statistical machine learning and is closely related to Perceptron (Section 2.2.2) and Support Vector Machines (Section 2.2.4). It is in fact a classification model rather than regression (Bishop, 2007). It 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 the simple linear model as presented in (2.3).

To use linear model in Logistic Regression, we apply logistic transformation of the original probability function

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

which by solving for $P(y = +1|\mathbf{x})$ results in

$$P(y = +1|\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + e^{-\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x})\rangle - b}}. \tag{2.5}$$

We can also compute

$$P(y = -1|\mathbf{x}; \mathbf{w}, b) = 1 - P(y = +1|\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + e^{\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x})\rangle + b}}. \tag{2.6}$$

By putting (2.5) and (2.6) together, we define Logistic Regression as

Definition 2. *Logistic Regression.*

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

Naturally, we expect the prediction $y = +1$ when $P(y = +1|\mathbf{x}; \mathbf{w}, b) \geq 0.5$ and $y = -1$, when $P(y = +1|\mathbf{x}; \mathbf{w}, b) < 0.5$. The decision rule is similar to Perceptron where we predict $y = +1$ when $\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}) \rangle + b \geq 0$, and $y = -1$ otherwise. Besides decision boundary, Logistic Regression can output the class probability from (Definition 2) 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, as it outputs detailed predictions (class probability).

As the model can output class probability, to obtain a Logistic Regression model we attempt to learn parameters $\mathbf{w}$ and b by maximizing the probability (likelihood) of the training data. In particular, the probability of training data $\mathbf{x}$ with class label y is $P(y|\mathbf{x})$. The likelihood of parameters given data can be computed from

$$L(\mathbf{w}, b; D) = \prod_{i=1}^{m} P(y_i|\mathbf{x}_i). \tag{2.7}$$

To apply MLE, it is easier if, instead of maximizing the likelihood, we maximize the log-likelihood, which turns the product (2.7) into sum

$$\log L(\mathbf{w}, b|D) = \sum_{i=1}^{m} \log P(y_i|\mathbf{x}_i) = -\sum_{i=1}^{m} \log(1 + e^{-y_i(\langle \boldsymbol{\varphi}(\mathbf{x}_i), \mathbf{w} \rangle + b)}). \tag{2.8}$$

Though MLE training for Logistic Regression meets our need of fitting the training data, it does not guarantee to generalize well on the test data. To achieve better generalization power, one can apply regularization technique as discussed in Section **??**. Many regularization methods for Logistic Regression have been proposed (Chen and Rosenfeld, 1999, 2000; Goodman, 2003), among which adding Gaussian prior on weight parameter $\mathbf{w}$ is one of the standard option. In practice, we assume $\mathbf{w}$ is generated according to a zero-mean spherical Gaussian with variance σ^2. Thus, the MLE problem (2.7) is transformed into *Maximum A-Posteriori* (MAP) problem of the following form

$$P(\mathbf{w}, b|D; \sigma^2) = P(\mathbf{w}|\sigma^2) \prod_{i=1}^{m} P(y_i|\boldsymbol{\varphi}(\mathbf{x}_i)) = e^{-\frac{||\mathbf{w}||^2}{\sigma^2}} \prod_{i=1}^{m} \frac{1}{1 + e^{-y_i(\langle \boldsymbol{\varphi}(\mathbf{x}_i), \mathbf{w} \rangle + b)}}. \tag{2.9}$$

Instead of maximizing the posteriori (2.9), it is easier to maximize the log-posteriori

$$\log P(\mathbf{w}, b|D; \sigma^2) = -\frac{||\mathbf{w}||^2}{\sigma^2} - \sum_{i=1}^{m} \log\left(1 + e^{-y_i(\langle \boldsymbol{\varphi}(\mathbf{x}_i), \mathbf{w} \rangle + b)}\right). \tag{2.10}$$

Numbers of optimization techniques have been proposed to solve the maximization problem (2.10) in Logistic Regression, many of which are covered in the survey (Minka, 2003). For example, Itrative Scaling methods were proposed and continuously developed in (Darroch and Ratcliff, 1972; Della Pietra et al., 1997; Berger, 1999; Goodman, 2002; Jin et al., 2003). A quasi-Newton method was discussed in (Minka, 2003). Komarek and Moore (2005); Lin et al. (2008) have proposed truncated Newton method. And coordinate descent method was proposed in (Huang et al., 2009).

There are also efforts being made to solve the logistic regression from its dual, of which the first algorithm was proposed in (Jaakkola and Haussler, 1999). Later on, other dual optimization algorithms have been developed, for example iterative optimization method (Keerthi et al., 2005) and dual coordinate descent method (Yu et al., 2011).

2.2.4 Support Vector Machines (SVM)

Support Vector Machines (SVM) is probably the most widely used classification algorithm in various real world applications. It is also among the best plug-and-play machine learning algorithms that usually achieves good classification performance. In this section we will begin the story of SVM by first introducing the *Maximum-Margin Principle* of separating training data. Then we will discuss the primal-dual optimization strategy and the kernel methods that allow SVM to deal with high dimensionality. In the end we will briefly cover the optimization strategy developed for SVM.

The algorithm framework of SVM was firstly introduced in (Boser et al., 1992; Cortes and Vapnik, 1995). Theory and algorithm are detailed in book chapters e.g. (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Bishop, 2007). We begin our discussion by considering a very simple case where we assume the data are linearly separable, that is there is a hyperplane in the feature space which separates the training data into two classes. In addition we assume the *separating hyperplane* has a simple form

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

such that we predict $y_i = -1$ if $f(\mathbf{x}_i) < 0$, and $y_i = +1$ if $f(\mathbf{x}_i) > 0$. Given w that achieves correct separation on the training data, we can decide the label of an arbitrary test example x_{ts} by the decision rule $y_{ts} =$

$\mathbf{sign}(f(\mathbf{x}_{ts}))$.

However, there can be infinite number of separating hyperplanes that solves the separation problem on the training data which is analogue to empirical risk minimization strategy presented in Section **??**. We wish to find the one that also generalizes well on the test data. A good strategy is to look for a hyperplane that keeps the maximum distance from examples of two classes. The strategy is often unknown as *Maximum-Margin Principle*. To see this, imagining putting a separating hyperplane close to one class of examples which will achieve better classification on the test examples from the other class.

Based on Maximum-Margin Principle, we further denote the *margin* for the i'th example by γ_i which is 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 we scale $\mathbf{w}$ and b with any constant (e.g. $\mathbf{w} \leftarrow \kappa\mathbf{w}, b \leftarrow \kappa b, \kappa \in \mathbb{R}$,) the margin γ_i stays unchanged. We still achieve the same classification performance and generalization power. As parameters are invariance to scaling, we set $||\mathbf{w}|| = 1$. Given a set of training example S, we also define the margin with respect to S as the minimum margin of individual training example

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

Our goal 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. Naturally, this corresponds to finding a "big-gap" between two classes. The max-margin solution is given as (Bishop, 2007)

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

The objective is to maximize the minimum margin over all training examples such that all examples are correctly separated with margin at least γ. However, 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 $\mathbf{w}$ with $\frac{\mathbf{w}}{\gamma}$, we obtain

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

which is equivalent to

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

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

where the objective is to find the weight vector of minimum norm that corresponds to maximize the margin between two class of examples. The constraints states that the training data should be well separated.

We do not use Definition 3 in practice for two reasons. First, many real world data are not linearly separable, where the solution to Definition 3 does not always exist. Secondly, the data usually come with noises and errors and 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

$$y_i(\langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in \{1, \cdots, m\}. \tag{2.11}$$

The margin slack parameter ξ_i will allow example with margin less than 1 (the violation of the constraints). 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 margin and separating hyperplane. With $\xi_i > 1$ the data point is misclassified locating on the other side of the hyperplane. With margin slack parameter, the new goal is to maximize the margin while penalizing the data points that either lies on the wrong side of the hyperplane or has margin less than one. This can be defined by

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

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

Optimization problem is usually transformed into dual form by introducing for each constraint a *Lagrangian multiplier* (dual variable) α. We obtain the following dual optimization problem

Definition 5. *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 KKT condition only examples with both $\xi_i = 0$ and satisfying equality constraints in (2.11) will be "active" and have dual variable $\alpha_i > 0$. These training examples lie on the margin and have $\gamma_i = 1$. They are called *support vectors* in SVM optimization. In fact the number of support vectors is usually smaller than the size of the training set. The property is extremely useful, as the weight vector can be expressed as the linear combination of the training examples

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

As most of the dual variables are zero, the evaluation can be done efficiently by maintaining a small set of non-zero αs.

In addition, we notice that to work out (Definition 5) it is not necessary to work on the explicit representation of $\mathbf{x}$, as only the value of the inner product $\langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{x}_j) \rangle$ is needed. As the algorithm can be represented entirely in terms of the inner product, we can also replace the inner product with $\langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{x}_j) \rangle$. In particular, we define the *kernel*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{x}_j) \rangle, \tag{2.12}$$

where $\boldsymbol{\varphi}(\cdot)$ is some high-dimensional feature map. Whenever we need $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$, we instead use $K(\mathbf{x}_i, \mathbf{x}_j)$. Besides, the property of kernel function (Schölkopf and Smola, 2002) allows us to compute (2.12) without explicitly accessing the high dimensional feature map. Thus the algorithm can tackle high dimensionality by learning from $\boldsymbol{\varphi}(\cdot)$ at a low computational cost.

In addition, according to *Mercer's Theorem* (Shawe-Taylor and Cristianini, 2004), every positive semidefinite symmetric function is a kernel. Kernel functions that heavily used in practice include the *linear kernel*

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

the *polynomial kernel*

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

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

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

where σ is the Gaussian width parameter.

The optimization techniques for solving SVM have been intensively studied. A "chunking" method was developed in (Vapnik, 1982) which takes the advantage that the SVM solution only depends on the nonzero Lagrangian multipliers. The method breaks down the original optimization problem into a set of smaller problems, and in each iteration solves a small problem which contains the nonzero Lagrangian multipliers from last step and a fixed number of worst examples that violate the constraints. Following the similar idea, Osuna et al. (1997) proposed a decomposition strategy to solve SVM on very large dataset. The algorithm decomposes the dual variables into an active part (working set) and an inactive part. In each iteration, the algorithm updates the variables in working set while keeps the rest unchanged by solving the smaller optimization problem arising from the working set. It adds and deletes the same amount of examples in the working set to maintain a constant memory. It gains advantage of linear memory requirement with respect to the number of support vectors and the size of the working set by compromising on the convergence time. (Joachims, 1998) has improved this decomposition strategy by applying a shrinking technique which will identify the dual variables that is less likely to change their values. Thus, the algorithm effectively maintains a good working set that makes much progress towards the object. The implementation of the algorithm is well-known as SVMLIGHT. The decomposition strategy was further developed by solving in each iteration a smallest as possible optimization that only contains two Lagrangian multipliers, which is known as *Sequential Minimum Optimization* (SMO) (Platt, 1998, 1999). SMO is the basis of another powerful software package LIBSVM developed by Chang and Lin (2011). Pérez-cruz et al. (2004) proposed a double-chunking method to further speed up the decomposition type of training. Similarly yet another approach "digesting" (Decoste and Schölkopf, 2002) optimizes subsets close to completion before adding new data, which saves huge amount of memory.

Another line of research to speed up SVM training is to divide the original problem into subproblems of smaller size, optimize SVM on each subproblem, and combine the results from each subproblem. In (Tresp, 2000; Schwaighofer and Tresp, 2001), the authors proposed a Bayesian scheme to combine the models from subproblems. Collobert et al. (2002) proposed a partition scheme. Graf et al. (2005) developed the SVMCASCADE that builds a partition tree of training examples, trains SVM on each partition, and returns only support vector from each partition. Quite recently, Hsieh

et al. (2014) proposed a divide and conquer solver that solves the original SVM problem with theoretical guarantee.

There are many other active lines of research on SVM optimization that aim to make kernel support vector machine scalable on very large scale datasets in terms of both computational time and memory space, to name but a few, 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), greedy method for basis selection (Keerthi et al., 2006), online SVM solver (Bordes et al., 2005), approximating kernel SVM objective function (Zhang et al., 2012; Le et al., 2013) which avoids high computational and memory cost but struggles in prediction performance, and approximating kernel matrix with 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 one of the prominent classification techniques widely used in machine learning community. In general, the methods train multiple base classifiers and combine them in order to achiever better classification performance. Most importantly, there is no requirement for base classifiers to be accuracy as long as they perform better than random guessing. There are several different types of ensemble algorithms, to name a few, bagging (Breiman, 1996a), boosting (Freund and Schapire, 1997; Schapire and Singer, 1999), stacking (Smyth and Wolpert, 1999), Bayesian averaging (Freund et al., 2004). Most of the algorithms have improved the classification performance when compared to their base classifier counterpart, some of them also enjoyed theoretical guarantee (Schapire et al., 1997; Koltchinskii and Panchenko, 2000; Cortes et al., 2014a,b). For our purposes, the section will start with and focus on bagging and boosting ensemble since both methods are best known approaches and are most related to this thesis. In the later part of the section, we will extend the ensemble methods to structured output space by reviewing several recently established algorithms.

2.3.1 Preliminary and Notations

In addition to the notations introduced in Section **??** and **??**, we assume there is a hypothesis class $\mathcal{F}$ where we generate weak/base hypothesis $f^t(x) \in \mathcal{F}$. By t we intend to index the t'th weak hypothesis. We denote by $H(x)$ the ensemble framework that is able to combine 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 as parameter x and output $y \in \mathcal{Y}$ for single label classification (y $\in \boldsymbol{\mathcal{Y}}$ in multilabel classification respectively).

2.3.2 Boosting

Boosting corresponds to a learning framework or a family of algorithms that take in a weak classifier and tuning it into a strong one. We begin our discussion from the notion of *concept class*. A *concept* is a boolean function over 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 that achieves high accuracy with high probability for all concepts in the class. On the other hand, a concept class is *weakly learnable* if the learning algorithm achieves arbitrary high accuracy where the only requirement is that the learning algorithm finds a function that performs better than coin flip. The concept of learnability were proposed in (Kearns and Valiant, 1989) together with the question whether the strong learnability and the weak learnability are equivalent which is known as *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 two classes of learnability are equal which lays the foundation of the boosting algorithm that aim to convert a weak learning algorithm into a strong one that achieves high performance.

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 of the weak learners. The exact algorithm is shown in (Algorithm 1). The fundamental idea of ADABOOST is to maintain a distribution D over all training examples, and update the distribution in each iteration such that difficult-to-classify examples will get more prob-

ability mass for the next iteration (line 8). In particular, the algorithm computes in each iteration 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^k for the current weak learner (line 5), and computes the adaptive parameter α^t for the current weak learner (line 6). The ensemble is the weighted combination of all weak learners.

$$H(\mathbf{x}) = \mathbf{sign}(\sum_{t=1}^{T} \alpha^t f^t(\mathbf{x})).$$

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: $\quad y_i = f^t(\mathbf{x}_i), \forall i \in \{1, \cdots, m\}$
5: $\quad \epsilon^t = \sum_{i=1}^m D^t(i)[\![y_i \neq \hat{y}_i]\!]$
6: $\quad \alpha^t = \frac{1}{2} \ln\left(\frac{1-\epsilon^t}{\epsilon^t}\right)$
7: $\quad Z = \sum_{i=1}^m D^{t-1}(i) \exp(-\alpha^t y_i \hat{y}_i)$
8: $\quad D^t(i) = \frac{1}{Z} D^{t-1}(i) \exp(-\alpha^t y_i \hat{y}_i), \forall i \in \{1, \cdots, m\}$
9: **end for**
10: **return** $H(\mathbf{x}) = \mathbf{sign}(\sum_{t=1}^T \alpha^t f^t(\mathbf{x}))$

For each weak learner $f^t(\mathbf{x})$, the strategy of updating adaptive parameter 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 current distribution D^t over training example and adaptive parameter for $f^t(\mathbf{x})$

$$\begin{aligned}
&\ell_{exp}(\alpha^t, f^t(\mathbf{x}), D^t) \\
&\quad = \sum_{i=1}^{m} D^t(i) \exp(-\hat{f}(\mathbf{x}_i)\alpha^t f^t(\mathbf{x}_i)) \\
&\quad = \exp(-\alpha^t) \sum_{i=1}^{m} D^t(i)[\![\hat{f}(\mathbf{x}_i) = f(\mathbf{x}_i)]\!] + \exp(\alpha^t) \sum_{i=1}^{m} D^t(i)[\![\hat{f}(\mathbf{x}_i) \neq f(\mathbf{x}_i)]\!] \\
&\quad = \exp(-\alpha^t)(1-\epsilon^t) + \exp(\alpha^t)\epsilon^t.
\end{aligned}$$

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

$$\frac{\partial \ell_{\exp}(\alpha^t, f^t(\mathbf{x}), D^t)}{\partial \alpha^t} = -\exp(-\alpha^t)(1-\epsilon^t) + \exp(\alpha^t)\epsilon^t = 0$$

Solve it for α^t, we get

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

Thus the rule for updating α^t will make sure the exponential loss of $\alpha^t f^t(\mathbf{x})$ will be minimized.

It is worth noticing that the ADABOOST algorithm describe in (algorithm 1) requires the learning algorithm $\mathcal{W}$ capable of learning from some specific distribution. The distribution will reflect the difficulty of the training examples during the learning. It is usually achieved by *reweighing* which initials a uniform distribution over training examples and updates the distribution after each iteration. For learning algorithms that cannot handle training data with specific distribution, we can apply *resampling* which samples training examples in each iteration according to some desired distribution.

Most recently, a new boosting algorithm DEEPBOOSTING was developed in (Cortes et al., 2014b). DEEPBOOSTING allows the learning algorithm to use complex hypothesis class contain very deep decision trees, by which it extends ADABOOST that only uses simple hypothesis class of decision tree with depth one. In addition, Cortes et al. (2014b) also developed rich learning bounds for DEEPBOOSTING optimization problem in terms of the Rademacher complexity theory. The new theory advances the previous performance guarantee of ADABOOST which is build in terms of margins of the training examples (Schapire et al., 1997; Koltchinskii and Panchenko, 2000).

2.3.3 Bootstrap Aggregating

Bootstrap Aggregating (BAGGING), initially developed by Breiman (1996a), is an ensemble method that exploits the independency between the weak learners. The algorithm is based on the fact that the error can be dramatically reduced by combining independent base learners. Denote by f_i the i'th weak learner. The ensemble prediction $H(\mathbf{x})$ is the averaged prediction over all weak learners

$$H(\mathbf{x}) = \mathbf{sign} \left(\sum_{i=1}^{T} f_i(\mathbf{x}) \right). \tag{2.13}$$

Denote by $\hat{f}$ the ground true function. Suppose base learner has probability ϵ of making independent mistake

$$P(f_i(\mathbf{x}) \neq \hat{f}(\mathbf{x})) = \epsilon.$$

As the bagging ensemble (2.13) makes a mistake when at least half of the weak learners make mistakes, we can compute the probability of bagging ensemble making mistake by

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

It is clear that the probability decreases exponentially with respect to the number of the weak learners. In extreme case, the probability will approach zero when T approaches infinity. Though it is impractical as the base learners are hardly independent since they are generated from the same training data, we still try to exploit the independency by adding randomness in the algorithm framework.

BAGGING uses bootstrap sampling (Efron and Tibshirani, 1994) to generate subsets of training examples. Given a training set of m training examples, a subset of same size is generated by sampling with replacement m time from original training set. The sampling procedure is then repeated T time in order to generate T subsets for constructing base learners. Note that the sampled subsets will be similar as they are sampled from the same training set. However, they will not be too similar 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 the training example is not sampled at all is $(1-\frac{1}{m})^m$. When m is large, the probability will approach 37%. That is around 37% of the training examples will not be represented in any given subset. The property of the bootstrap sampling 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).

Random Forest developed in (Breiman, 2001) is an important extension of BAGGING. The major different from BAGGING is that randomized feature selection is incorporated in random forest. In practice, the random forest algorithm selects in each step a subset of features and constructs conventional random tree classifier within the selected subset of features. The feature split process of decision tree is deterministic given a set of features. Liu et al. (2008) proposed VR-TREE which further extend random forest algorithm by introducing randomness in both feature selection and split processes.

3. Flat 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 the multilabel classification has progressed rapidly in the past two decades with many learning algorithms being developed and applied in real world classification tasks. In general, there are two lines of research for multilabel classification namely flat multilabel classification and structured output prediction. In flat multilabel classification, the multiple labels are essentially treated as a "flat" vector of labels. The structured output prediction, on the other hand, models the multiple labels via a graph structure that connects labels. This chapter will devote to flat multilabel classification in which we will present several classical algorithm techniques. The structured output prediction will be covered in the later part of the thesis.

Consider the infeasibility of listing all algorithms for flat multilabel classification, it is easier if we are able to categorize the algorithms into classes. In this chapter, we adopt the simple categorization presented in (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010) which gives us two major algorithm categories namely problem transformation and algorithm adaptation. For problem transformation category, we will present the methods that aim to transfer the flat multilabel classification problem into other well studied problem settings (e.g. single-label classification problem, ranking problem). For algorithm adaptation category, we will describe the methods that directly modify the popular learning techniques (e.g. LR and ensemble method) to solve the multilabel classification problem. Nevertheless, the present algorithms will explore the inherited difficulties of flat multilabel classification. That is the exponential multilabel space and the implicit label correlation. Besides, the author re-

alizes that it is impossible to cover every lines of research in the field of flat multilabel classification. Interested readers are pointed out to the recent research review articles, for example, (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010; Zhang and Zhou, 2014)

3.1 Preliminary and Notations

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 data 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 the 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 setting by letting $k = 1$. A vector $\mathbf{y} = (y_1, \cdots, y_k) \in \boldsymbol{\mathcal{Y}}$ is called *multilabel* and its element y_j is called *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 consisting 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 the training input and $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$ is an arbitrary output, is called *pseudo-example*. It is worthy of noticing that 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 is to learn a mapping function $f \in \mathcal{H} : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ that can compute the multilabel given an arbitrary input such that the predifined loss ℓ for the future unseen examples will be minimized. The loss function is usually tailored for the learning algorithm.

3.2 Problem Transformation

The problem transformation approach aims to transform the flat multilabel classification problem in to other well established problem settings. The most typical way is known as *binary relevant* proposed by Tsoumakas and Katakis (2007); Tsoumakas et al. (2010). BR amounts to transform the multilabel classification problem into a set of single-label classification problems and to independently learn a single-label classifier for each microlabel. We will focus on BR in this section. Another possibilities include, for example, to transform the multiple labels into label power set (Tsoumakas and Vlahavas, 2007) or to transform the problem into label ranking problem (Elisseeff and Weston, 2001; Brinker and Hüllermeier,

2007; Fürnkranz et al., 2008; Chiang et al., 2012). However, learning through label ranking will not be discussed into details as it slightly diverges from the main scope of the thesis.

3.2.1 Multilabel K-Nearest Neighbors (ML-KNN)

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

Mathematically let $C_{\mathbf{x}}(j)$ denote the number of the neighbors of $\mathbf{x}$ with 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 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 training data in terms of corresponding relative frequencies.

The central problem with ML-KNN is that ML-KNN is ignorant of the label correlations. In order to exploit the label dependency, other approaches have been proposed (Cheng and Hüllermeier, 2009; Younes et al., 2011). Besides, there are many algorithm alternatives that also align to the direction of KNN typed learning in flat multilabel classification, for example, Brinker and Hüllermeier (2007); Chiang et al. (2012) proposed methods that incorporates KNN with ranking model.

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 $h = \{h_1, \cdots, h_k\}$. The j'th classifier h_j is responsible for learning and predicting the j'th microlabel. In particular, for each microlabel $\mathbf{y}[j]$, CC first constructs a new training data S_j then learns a classifier h_j via arbitrary single-label classifier. S_j is built by taking the j'th microlabel as output variable and combing the feature space of $\mathbf{x}$ and all $j-1$ prior microlabels as the input feature which is defined by

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

Thus, CC takes the label correlation into account by incorporating label information as concatenated features in the input feature space. The idea has previously been applied in (Godbole and Sarawagi, 2004). However, the addition label information only takes a small part of the input feature space especially when the origin space is already high. In this case, one can only expect to gain additional performance from concatenated label information with high correlation to the output microlabel.

Besides, CC has been explained in terms of condition probability theory by Probabilistic Classifier Chains (PCC) presented in (Read et al., 2009; Dembczynski et al., 2010). Since there is no standard way for classification order of microlabels in CC, Ensemble Classifier Chains (ECC) has been developed in (Read et al., 2011) which further improves CC by generating and combining multiple chains of classifiers.

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) as ML-KNN. It extends ML-KNN by exploring the label correlation within the neighbors of an instant for posterior inference. It is also similar to CC but with major difference. The central idea of IBLR is to take the labels of the neighboring examples as the only features in order to evaluate the label of current example. Similar idea has 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 sets. Then the algorithm builds a logistic regression (LR) model (2.2.3) for each microlabel based on the label information collected from all training examples in $N_k(\mathbf{x})$. Mathematically IBLR defines a posterior probability for 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 is the index that iterates over all microlabels. $w_i^{(j)}(\mathbf{x})$ defined by

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

collects the i'th microlabel from each neighbor $\mathbf{x}' \in N_k(\mathbf{x})$ and weights the microlabel according to the similarity between $\mathbf{x}$ and $\mathbf{x}'$ encoded in $K(\mathbf{x}', \mathbf{x})$. $\alpha_i^{(j)}$ is the regression coefficient of the i'th microlabel information from the neighbors of $\mathbf{x}$ to the j'th microlabel of example $\mathbf{x}$. $w_0^{(j)}$ is a bias term.

3.3 Algorithm Adaptation

The central idea of the algorithm adaptation approach is to directly modify the popular single-label learning algorithms and apply them to the multilabel classification problems. The approach includes but is not limited to the ensemble typed learning, the LR typed learning, and regularized learning that are described in the following part of the section. There also exists methods that are adapted from, for example, perceptron (Crammer et al., 2003) and neural network (Zhang and Zhou, 2006). They are not detailed mostly due to the divergence from the main topic of the thesis.

3.3.1 Ensemble Methods for Flat Multilabel Classification

As it is straightforward to combine multiple scalar output variables, the ensemble methods have been initially developed for single-label classification (Breiman, 1996a; Freund and Schapire, 1997) or regression (Breiman,

1996a) tasks. However, it is not immediately apparent how to combine vector valued outputs in flat multilabel space.

ADABOOST.MH is the multilabel variance of original ADABOOST algorithm proposed by Schapire and Singer (1999), which is further developed by Esuli et al. (2008). The core idea of ADABOOST.MH is to adopt hamming loss instead of 0/1 loss. In particular, the algorithm reduces from a multilabel classification problem to a set of single label classification problems 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 detailed in (Algorithm 2). In particular, it maintains a distribution over all examples and labels. In each iteration, the algorithm takes in the distribution and all training examples, generates a weak learner $f^k(\mathbf{x})$ (line 3), computes hamming loss (line 5), computes the adaptive parameter α^t (line 6), and update the distribution (line 8). The returned prediction $H(x)$ is the combination of base learners $f^t(x)$ weighted by 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: $\quad \epsilon^t = \sum_{l=1}^{k} \sum_{i=1}^{m} D^t(i,l) [\![\mathbf{y}_i[l] \neq \hat{\mathbf{y}}_i[l]]\!]$
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** $H(\mathbf{x}) = \mathbf{sign}(\sum_{t=1}^{T} \alpha^t f^t(\mathbf{x}))$

Other ensemble flat multilabel classification methods based on boosting or bagging are also developed in (Wang et al., 2007; Yan et al., 2007; Kocev et al., 2013). Most recently, three other learning algorithms with performance guarantee are proposed in (Cortes et al., 2014a). Besides, there are quite a few ensemble approaches that are adapted from single label ensemble methods and are dedicated to solve real world classification problems. For example, (Collins and Koo, 2005; Zeman and Žabokrtský, 2005; Sagae and Lavie, 2006; Zhang et al., 2009) are applied in natural lan-

guage processing, and (Fiscus, 1997; Mohri et al., 2008; Petrov, 2010) are applied in text and speech recognition.

3.3.2 Correlated Logistic Regression (CORRLOG)

Correlated Logistic Regression (CORRLOG) developed by Bian et al. (2012) is a model based approach for flat multilabel classification. CORRLOG is a major step forward of IBLR by constructing a logistic regression classifier over all microlabels and by modeling pairwise microlabel correlation with a function defined on microlabel pairs.

In fact, CORRLOG is derived from Independent Logistic Regression (ILRS) model. Given an arbitrary training example and label $(\mathbf{x}, \mathbf{y})$, we can construct a set of independent logistic regreesion 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 all microlabels. The bias term as in Definition 2 is omitted which is equvilent to augment x with a constant term (Bian et al., 2012). Otherwise, (3.1) can be derived into Definition 2 by replacing w with $\frac{\mathbf{w}}{2}$. ILRS has the problem of ignoring the label correlation and overfitting the training data when the number of microlabel k is high. To reveal label correlation, CORRLOG augments the posteriori probability (3.1) by a function $Q(\mathbf{y})$ defined on the set of microlabels as

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

Put together (3.1) and (3.2), the formalism of 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 correlation by augmenting joint prediction with a quadratic term $Q(\mathbf{y})$ built from microlabel pairs.

3.3.3 Multitask Feature Learning (MTL)

Multitask Feature Learning (MTL) developed in (Argyriou et al., 2007) is another algorithm designed for flat multilabel classification. MTL is quite different from other learning algorithms discussed in the previous sections. In particular, the MTL approach is based on the assumption that

different microlabels are related such that they share a common underlying feature representations. Similar assumption is also applied in (Caruana, 1997; Baxter, 2000; Ben-David and Schuller, 2003).

Denote by $f_t(\mathbf{x})$ the task specific function for the t'th task. $f_t(\mathbf{x})$ can be expressed as

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

where $\mathbf{a}_t \in \mathbb{R}^d$ is the weight vector parameters for the t'th task. $h(\boldsymbol{\phi}(\mathbf{x}))$ is a linear feature mapping composed by

$$h(\boldsymbol{\phi}(\mathbf{x}))[i] = \langle \mathbf{u}_i, \boldsymbol{\phi}(\mathbf{x}) \rangle,$$

where $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^d$ is in original feature space and $\mathbf{u}_i \in \mathbb{R}^d$ is a linear feature map. We further denote by A the matrix composed by $\mathbf{a}_t$, and denote by U the matrix composed by $\mathbf{u}_i$. MTL assumes that tasks share a small set of features where A is supposed to be sparse with many entries equal to zero. The optimization problem of MTL is defined as

Definition 6. MTL *Optimization Problem in Primal*

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

where the objective is to minimize the empirical error $\ell(\mathbf{y}_{i,t}, \langle \mathbf{a}_t, \langle U, \boldsymbol{\phi}(\mathbf{x}_i) \rangle \rangle)$ and to keep a small proportion of the non-zero element in $\mathbf{a}_t$. C is the parameter to balance these two aspects. As (Definition 6) is non-convex and the second term is non-smooth, it is transformed into an equivalent form and solved by an alternative optimization approach.

An extension of MTL algorithm was proposed in (Argyriou et al., 2008a) which introduced a nonlinear generalization using kernel methods. In addition, similar but not identical algorithms (Argyriou et al., 2008b; jacob et al., 2009) are also proposed which assume that the tasks form clusters such that task specific weight vectors should be similar within the clusters. Recently, Romera-Paredes et al. (2012) proposed a method which exploits the information between unrelated microlabels, based on a similar assumption that 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 output variables as a "flat" vector, structure output prediction assumes multiple output variables are correlated and locate in the structured output space. It builds a graph structure (e.g. chain, tree) to connect multiple output variables by which it gains the advantage of exploring the label correlation compared to flat multilabel classification. In this chapter, we will start by introducing several classical structure output learning methods developed during the last decade. We will present our research of applying the structured output learning on molecular activity prediction problem. We will also present our new algorithm SPIN that is capable of predicting a directed acyclic graph on network influence prediction problem.

4.1 Preliminary and Notations

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

tify the node label y_i and the edge label $\mathbf{y}_{i,e}$ on the graph. In addition, we denote the possible label of node i by u_i and the possible label of edge $\mathbf{e}$ by $\mathbf{u}_e$ where u_i and $\mathbf{u}_e$ do not constraint to 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 include Structured Perceptron, Conditional Random Field, Structured SVM, and Max-Margin Markov Network.

4.2.1 Structured Perceptron

The Perceptrion developed by Rosenblatt (1958) is one of the oldest learning algorithm in machine learning. Structured Perceptron, as suggested by its name, can be seen as the generalization of Perceptron algorithm to structured output space. It was firstly proposed in (Collins, 2002; Collins and Duffy, 2002) with the formalism incredibly 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 some feature weight parameter $\mathbf{w}$. After weight parameter $\mathbf{w}$ is obtained, one need to solve the *argmax* problem in order to find the best output for a given input $\mathbf{x}$

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

This is solved by Viterbi decoding in (Collins, 2002).

The weight parameter $\mathbf{w}$ is learned via standard Perceptron iterative update by solving argmax problem (4.1) in each iteration. In particular, the algorithm loops through training examples and updates $\mathbf{w}$ whenever the predicted label $\hat{\mathbf{y}}_i$ is different from true label $\mathbf{y}_i$ 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 the idea of which is also used in “Average Perceptron” (Freund and Schapire, 1999).

The central problem with Structured Perceptron learning is its loss function. In fact, Structured Perceptron tacitly applies $0/1$ loss over output variables defined as

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = [\![\mathbf{y} \neq \hat{\mathbf{y}}]\!].$$

As a result, it is impossible to distinguish the nearly correct and the completely incorrect output multilabels. Both will lead to the same update to the feature weight parameter during the training in (4.2) .

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 model $P(\mathbf{y}|\mathbf{x})$ from input variable $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ and output variables $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$. It optimizes a log-loss which is analogue to $0/1$ loss over 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 $G = (E, V)$ be an output graph such that $\mathbf{y} = (y_v)_{v \in V}$. A *Conditional Random Field* (CRF) defines the 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 the joint feature map and $Z_{\mathbf{x},\mathbf{w}}$ is the partition function dependent on $\mathbf{x}$. $Z_{\mathbf{x},\mathbf{w}}$ 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 $\mathbf{x}$, the random variables y_v obey the Markov property with respect to output graph G.

Apply similar regularization technique as used on LR in Section 2.2.3, 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 solved in (Lafferty et al., 2001) by Improved Iterative Scaling algorithm (IIS) (Della Pietra et al., 1997). In order to make CRF work in practice, one has to make sure that the partition function (4.3) can be computed efficiently.

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

Taskar et al. (2004) designed Max-Margin Markov Network (M^3N) that combines the framework of kernel based discriminative learning and probabilistic graphical model. M^3N is extended from SVM (Section 2.2.4) to

structured output space. It also improves CRF (Section 4.2.2) in terms that the evaluation of the partition function (4.3) can be avoided by introducing the odd-ratio typed learning that is not dissimilar to LR (Section 2.2.3).

M^{3}N defined a log-linear Markov network over multiple output labels which explores the correlation in the label space. The compatibility score define 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 the multilabel $\mathbf{y}$ to the input $\mathbf{x}$ according to the Markov network. Feature weight parameter $\mathbf{w}$ in (4.5) ensures the example with correct multilabel will reach higher score than with incorrect multilabel. To learn the feature weight parameter $\mathbf{w}$, M^{3}N defines the margin as the difference in 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 (Section 2.2.4), M^{3}N requires the margin to be at least $\ell(\mathbf{y}_i, \mathbf{y})$. The primal optimization problem of M^{3}N can be formulated as

Definition 7. 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 solution can always be found, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss function between the correct multilabel $\mathbf{y}_i$ and the 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 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 primal optimization problem of M^{3}N in Definition 7 is difficult to solve as there are exponential numbers of constraints, one for each pseudo-example $(\mathbf{x}_i, \mathbf{y})$. The corresponding dual form is also difficult due to exponential number of dual variables (Taskar et al., 2004). By exploring the Markov network structure, the original optimization problem (Definition 7) can be formulated into factorized dual quadratic programming, as long as the loss function ℓ and 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 edges in the network, it still cannot fit into standard QP solver. (Taskar et al., 2004) developed a coordinate descent method analogous to the Sequential Minimal Optimization (SMO) used for SVM training (Platt, 1998, 1999). Subsequently, other efficient optimization algorithms have been proposed, which include exponential gradient optimization methods from Bartlett et al. (2005), extra-gradient methods from Taskar et al. (2006), sub-gradient methods from Ratliff et al. (2007), and conditional gradient methods from 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 partition function and allowing complex loss function.

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

Support Vector Machine 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 8. 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 pseudo-label and the correct label, and C is the slack parameter that controls the amount of regularization in the model. The interpretation of Definition 8 is also similar as of Definition 7. Besides, Tsochantaridis et al. (2004) suggest that M^{3}N will work hard on the pseudo-example $(\mathbf{x}_i, \mathbf{y})$ that incur 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 relaxation of the original problem via a cutting-plane methods (Bishop, 2007; Joachims et al., 2009). Constraints are added as necessary and the iterative optimization approach will converge to some 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, the model need to compute the loss-augmented inference problem (Tsochantaridis et al., 2005) defined as

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \boldsymbol{\mathcal{Y}}/\mathbf{y}_i}{\mathbf{argmax}} \left[1 - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle\right] \ell(\mathbf{y}_i, \mathbf{y}). \tag{4.7}$$

The loss function appears as a multiplicative term making (4.7) not decomposable. This give a intractable inference problem in general. In exchange of intractability, SSVM can work with complex loss functions that do not assume any property of decomposition. The generality of the loss function can be seen as an advantage compared to CRF and M^{3}N.

4.3 Structured Output Prediction for Molecular Activity Classification

The molecular activity classification problem has long been tackled under the single task classification setting. On the other hand, multiple interdependent molecular activities are often screened simultaneously. This makes single task classification approaches suffer from two formidable issues. The first issue is scalability which in order to predict multiple activities for one molecule a set of single task classifiers need to be built separately, which soon becomes infeasible when large amount of activities need to be predicted. The second issue lies in the fact the single task model discards the dependency information among multiple output targets. The dependency structure can be further utilized to improve the performance. Therefore, in Publication II we investigate the potential of using structured output learning framework to tackle the molecular activity classification problem.

4.3.1 Background and Introduction

Molecular classification, the goal of which is to predict the anticancer potentials of the drug-like molecules, is one crucial step in drug discovery and has draw enormous attention from machine learning community. 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).

Various machine learning methods have been developed for the 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 SVM based methods for reliable predictions (Trotter et al., 2001; Byvatov et al., 2003; Xue et al., 2004). Albeit with the large quantity of the computational models, they only focus on predicting single target variable (e.g. inhibition potential in one cell line). On the other hand, relative larger number of cell line targets are often screened at the same time. For example in the recent *NCI-60* Human Tumor Cell Line Screen project (Shoemaker, 2006), thousands of molecular structures are tested agains hundreds of cell line targets.

4.3.2 Methods

To efficiently and accurately predict molecular activities in multiple cell lines at the same time, we proposed a structured output learning algorithm in Publication II, which is to our knowledge the first multi-task learning approach for molecular classification problem. The algorithm can be seen as an instantiation of the structured output learning method MMCRF (Rousu et al., 2006). In particular, the model defines a compatibility score through inner product for a molecular structure $\mathbf{x}$ and the activities in multiple cell lines $\mathbf{y}$

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

where $\mathbf{w}$ are the feature weight parameters that ensures the molecular structure with correct activities will be scored higher than with incorrect activities. $\mathbf{w}$ are learned by maximizing the minimum loss-scaled margin between correct examples $(\mathbf{x}_i, \mathbf{y}_i)$ and incorrect examples $(\mathbf{x}_i, \mathbf{y})$ over the

whole training set

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

where $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss function defined as

$$\ell(\mathbf{y}_i, \mathbf{y}) = \sum_{j=1}^{k} [\![\mathbf{y}_i[j] \neq \mathbf{y}[j]]\!].$$

The loss scaled margin optimization will push high-loss pseudo-examples further away from the correct example than the low-loss pseudo-examples. The model is optimized by conditional gradient optimization (Bertsekas, 1995) in marginalized dual space (Taskar et al., 2004), which not only enjoys a polynomial-size parameter space in marginalized dual representation but also enables the kernel function that is capable of dealing with non-linearity of the complex molecular structures.

Kernel is the classical way of defining the similarity between complex objects. In Publication II, We used graph kernel to measure the similarity of pair of molecular structures. The common way to represent the structure of a molecule is to use an undirected labeled graph $G = (V, E)$, where vertices $V = \{v_1, \cdots, v_n\}$ corresponds to atoms and edge $E = \{e_1, \cdots, e_m\}$ corresponds to the covalent bonds. The adjacency matrix A of graph G is defined sush that its (i, j)'th entry $A_{i,j}$ equals to one if there is an edge between atom i and atom j.

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

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

and edges are computed from

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

The walk kernel can be defined on the adjacency matrix of $G_\times$

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

where $0 < \lambda < 1$. Using exponential series or geometric series the walk kernel can be compute in cubic time (Gärtner, 2003)

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

where we denote by $\mathbf{I}$ the identity matrix and by $\mathbf{e}$ the matrix of ones.

Weighted Decomposition Kernel in (Menchetti et al., 2005; Ceroni et al., 2007b) is an extension of substructure kernel (Komarek and Moore, 1999) by weighting identical part in two graph with contextual information. The kernel will evaluate the matching subgraphs (*contextor*) in the neighborhood of an atom (*selector*). We also 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 structured output learning algorithm described above, we assume the underlying Markov network structure is fully observed. In other words, we need to build a network connecting cell lines used as the output graph, with nodes correspond to cell lines and edges correspond to potential statistical dependencies. Auxiliary datasets are available for this purpose (Shoemaker, 2006) which implicitly describes the relationship between cell lines. In practice, we first computed the covariance matrix for cell lines based on auxiliary datasets, then extracted the network structure by applying the following two methods. In maximum spanning tree approach, we take the minimum number of edges that make a connected network whilst maximizing the edge weights. In correlation thresholding approach, we take all edges that exceed a fixed threshold, which typically generates a non-tree graph.

4.4 SPIN for Network Response Prediction

In Publication I, a structured output learning model for network response prediction problem is developed. The proposed model, SPIN, is *context-sensitive* and is different from the previous the state-of-the-art methods which only model the influence in terms of the network connectivity. The inference problem of SPIN is $\mathcal{NP}$-hard in general which was solved by a semidefinite programming algorithm SDP with approximation guarantee and by a fast GREEDY heuristics.

4.4.1 Background and Introduction

With the extensive availability of the large scale networks, there are tremendous 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 have been widely applied in many areas, to name but a few, spreadness of pathogens or infectious diseases (Hethcote, 2000; Anderson and May, 2002), diffusion of medical and technology innovation (Strang and Soule, 1998; Rogers, 2003), opinion and news formation (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), viral market (Domingos and Richardson, 2001; Kempe et al., 2003; Liben-Nowell and Kleinberg, 2003).

In the field of studying network influence, one primary interest is to discover the latent structures that reveal 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, or in epidemiological study where people are affected by pathogens through various ways. In this case, the task is to infer network structure in terms of edges connecting nodes given a set of action cascade. Many algorithms are designed to solve the problem in this setting from NETINF (Gomez Rodriguez et al., 2010), NETRATE (Rodriguez et al., 2011), KERNEL CASCADE (Du et al., 2012) to the most recent work about two stage model for inferring influence (Du et al., 2014), inference algorithm using cascade without timestamp (Amin et al., 2014), and general framework of inferring diffusion structure (Daneshmand et al., 2014). On the other hand, we argue the problem is unnecessarily hard to solve as in many cases the structure of the network is given (e.g. friendship network). There are also related researches aim 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 the network G, the influence from node u to u' not only depends on their connections but also depend on the action under consideration. For example, u' follows u' in twitter network, u' will

retweet the message from u if the message is related to *science* but not related to *politics*. Therefore, we propose the following definition of the network influence problem

Definition 9. *Network Influence Problem.*
Given a complex network and an action performed on the network, predict the subnetwork that responses to the action. In particular, which nodes will perform the action and which directed edges relay the action from one node to its neighbors.

4.4.2 Methods

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

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

It is easy to understand that the action a with correct response network $G_{\mathbf{a}}$ will achieve high score than with any incorrect response network $G'_{\mathbf{a}}$. As introduced in previous section, the joint feature map $\boldsymbol{\phi}(\mathbf{a}, G_{\mathbf{a}})$ is composed by the tensor product between action feature $\boldsymbol{\varphi}(\mathbf{a})$ and the features of the response network $\boldsymbol{\Upsilon}(G_{\mathbf{a}})$. In particular, $\boldsymbol{\varphi}(\mathbf{a})$ can be a bag-of-words features of the action (e.g. blog post) and $\boldsymbol{\Upsilon}(G_{\mathbf{a}})$ is a vector of edges and labels of the response network $G_{\mathbf{a}}$.

The feature weight vector w is learned through maximum-margin structured output learning by solving the following optimization problem

Definition 10. *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 by $\mathcal{H}(G)$ we denote the set of directed acyclic graphs of G.

To solve the above optimization problem, we have to compute both in training and in prediction the highest-scoring subgraph for an action. In particular, during the training the goal is to find the worst margin violating subgraph which corresponds to solving the loss-augmented maximization problem

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

During the prediction, the goal is to find the subgraph with maximum compatibility given the action

$$H^*(\mathbf{a}_i) = \underset{H \in \mathcal{H}(G)}{\mathbf{argmax}}\, F(\mathbf{a}_i, H; \mathbf{w}). \tag{4.8}$$

As the two problems are different only in terms of the definition of the score, we explain our inference algorithm based on (4.8) by writing the problem explicitly in terms of weight vectors and the feature maps

$$\begin{aligned} H^*(\mathbf{a}_i) &= \underset{H \in \mathcal{H}(G)}{\mathbf{argmax}}\, \langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{a}) \otimes \boldsymbol{\Upsilon}(H) \rangle \\ &= \underset{H \in \mathcal{H}(G)}{\mathbf{argmax}} \sum_{e \in E^H} s_{\mathbf{y}_e}(e, \mathbf{a}), \end{aligned} \tag{4.9}$$

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

Lemma 1. *Finding the graph that maximizes (4.9) is an $\mathcal{NP}$-hard problem.*

In supplementary material of Publication I, we have proved the $\mathcal{NP}$-hardness by forming a reduction from the MAX-CUT problem (Garey and Johnson, 1990). In addition, we proposed two algorithms to solve (4.9). The first is called SDP inference which introduces for each node $u \in V$ a binary variable $x_u \in \{-1, +1\}$ and transfer the inference problem into an integer quadratic program (IQP) problem. The IQP is then tackle by similar technique introduced 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 value by incomplete Cholesky decomposition. The benefit from SDP inference algorithm is the approximation guarantee. In particular, the proposed SDP inference algorithm is a 0.796 approximation of the original IQP.

As SDP inference is not scalable on large networks, we developed a GREEDY heuristic base on observation stated as the following lemma:

Lemma 2. *The inference problem (4.9) can be stated 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 supplementary material of Publication I. As a results, the GREEDY algorithm starts with an empty vertex set and adds one

vertex at a time such that the increment of the score is maximized over all inactivated vertices. The iteration ends when the objective cannot be improved. It is worthy noticing that we are not able to give any approximation guarantee for the solutions produced by the GREEDY algorithm. The property of submodularity, which is often used to analyze the greedy algorithm, only holds for the special case of our inference problem.

The decomposable loss function $\ell_G(G_{\mathbf{a}}, G'_{\mathbf{a}})$ will penalize the mistakes uniformly over the network. However, we wish the learning algorithm focus on the vicinity of the *foci* based on the observation that one certain action will only affect a small part of the network. This suggests we should penalize the mistake according to the distance to the *foci*. Therefore, we define the *symmetric-difference loss*, also known as *hamming loss*, applied on the nodes and edges of the network. In addition, we define the scaling function γ_G according to the distance to *foci*. The resulting loss function is

$$\ell_G(G_{\mathbf{a}}, G_{\mathbf{b}}) = \sum_{u \in V} \ell_v(G_{\mathbf{a}}, G_{\mathbf{b}}) \gamma_G(u_k; u) + \sum_{(u,u') \in V} \ell_e(G_{\mathbf{a}}, G_{\mathbf{b}}) \gamma_G(u_k; u).$$

We proposed two scaling function in Publication I. One is *exponential scaling* where mistakes are penalized by λ and λ is weighted exponentially according to the shortest path distance to *foci*. The other is *diffusion scaling* where mistakes are penalized by the values computed from diffusion kernel (Kondor and Lafferty, 2002). Diffusion scaling has the effect of shrinking the distance to the nodes that are connect to *foci* by many paths.

5. Structured Output Learning with Unknown Structure

5.1 Graph Labeling Ensemble (MVE)

Multi-task molecular activities classification with structured output learning that relies on the representation of the cell-lines structures allows us to model the dependencies betweem different output variables. When apply structured output learning, it is explicitly assumed the structure of the output network is fully observed. In Publication II, we supposed the feasibility of extracting the structures assuming it is given implicitly in the auxiliary datasets. For many real world structured output learning problems, we cannot take the structure as granted as it is often difficult to get the underlying dependency structure. Therefore, we explored in Publication III the possibility of constructing an simple ensemble of multiple structured output learning models built on random output graphs and applied the technique on molecular activity classification task.

5.1.1 Methods

We use MMCRF as base classifier trained on a set of random output graphs. Particularly, for each base model, a random graph G_t is generated to couple the output vector variables $\mathbf{y}$ which are the activities of molecular structure $\mathbf{x}$ over all cell lines. The base model MMCRF is then learned based on training data $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$ and the output graph G_t. After ensemble has been generated, the predictions are extracted from each base classifier and are collected for a post-processing step: we compute a majority vote over the graph labeling from the sign on the means of the base classifier's prediction

Definition 11. *Majority Voting Ensemble (*MVE*).*

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

where by T we denote the size of the ensemble, and by $F^{(t)}(\mathbf{x}) = \{F_j^{(t)}(x)\}_{j=1}^{k}$ we denote the predicted multilabel in the t'th base classifier. In words, the ensemble prediction on each microlabel is the most frequently appearing prediction among the base classifiers. It is also worthy of noticing that the MVE framework can be extended with any structured output learning models as long as they incorporate the output structure into learning process and make predictions based on the structure.

We designed two approaches to generate random output graphs. In random spanning tree approach, we first generate a random correlation matrix and extract the spanning tree out from the matrix. The approach will eventual output 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. The approach will output disconnected pairs.

5.2 Random Graph Ensemble (AMM, MAM)

Through extensive experiments in Publication III, we have validated the feasibility of constructing a majority voting ensemble (MVE) of structured output learner built on a set of random output graphs. The proposed method also improved the performance on molecular activity classification problem. In Publication IV, we aim to develop elegant aggregation techniques (AMM and MAM respectively) that will perform inference before or after forming ensemble. We tested the proposed techniques on a set of heterogeneous multilabel datasets from various domains to verify the wide applicability. In addition, we have brought forward a theory that explains the improvement of MAM framework. The theory is based on the reconstruction error of the compatibility score.

5.2.1 Background and Introduction

Besides the assumption made for MVE which the base classifier will devise the prediction according to the structure of the output graph, we also assume that the base classifier for AMM and MAM is modelled with a Markov network structure $G = (E, V)$. That is the base classifier defines a compatibility score $\psi(\mathbf{x}, \mathbf{y})$ for pairs $(\mathbf{x}, \mathbf{y}) \in \boldsymbol{\mathcal{X}} \times \mathcal{Y}$ according to the output

graph G, indicating how well the input and the output goes together

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

where by $\psi_e(\mathbf{x}, \mathbf{y}_e)$ we denote edge compatibility score, or *edge potential*, between input $\mathbf{x}$ and edge label $\mathbf{y}_e$ on edge e. $\mathbf{w}$ is the feature weight parameters that naturally ensures input $\mathbf{x}$ with correct output $\mathbf{y}$ achieves higher compatibility score than with incorrect output.

In addition, we assume we have access to the edge potentials between input and edge labels for each base classifiers

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

With edge compatibility scores, we can infer the max-marginal of label u_i on node y_i (Wainwright et al., 2005).

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

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

5.2.2 Methods

Denote by $\mathcal{G} = \{G^{(1)}, \cdots, G^{(T)}\}$ a set of random output graphs, and by $\{\tilde{\boldsymbol{\psi}}^{(1)}, \cdots, \tilde{\boldsymbol{\psi}}^{(T)}\}$ the max-marginal vectors from base classifiers that is built on each output graph. The AMM prediction on each node is obtained by averaging the max-marginals of base classifiers and choose the maximizing microlabel for the node

Definition 12. *Average of Max-Marginal Aggregation (*AMM*).*

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

and the predicted multilabel is composted by the predicted microlabels

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

AMM performs the inference to find the max-marginals before forming ensemble. On the other hand, the maximum of average marginals aggregation (MAM) will first collect the local edge potentials $\boldsymbol{\psi}_E^{(t)}$ from each base classifier, average them and finally performs inference on the global consensus graph $\hat{G} = (\hat{E}, V)$ with averaged edge potentials

Definition 13. *Maximum of Average marginals Aggregation (*MAM*).*

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

Publication IV, Lemma 1 allows us to simplify the computation of MAM in terms of dual variables and kernels. As a results, the MAM ensemble can be equivalently stated 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 by $\bar{\mu}(i,e,\mathbf{u}_e) = \frac{1}{T}\sum_{t=1}^{T}\mu^{(t)}(i,e,\mathbf{u}_e)$ we denote the marginalized dual variable averaged over ensemble, and

$$H_e(x_i,\mathbf{u}_e;x,\mathbf{y}_e) = K_{\boldsymbol{\phi}}(x,x_i)\cdot\left(K_{\Upsilon,e}(\mathbf{y}_{ie},\mathbf{y}_e) - K_{\Upsilon,e}(\mathbf{u}_e,\mathbf{y}_e)\right).$$

Besides the algorithm frameworks for random graph ensemble, we also developed theoretical analysis to explain the improvement of MAM ensemble. The analysis extends the theory on single task ensemble developed by Brown and Kuncheva (2010). In general, Publication IV, Theory 1 states that the reconstructive error from MAM ensemble is guaranteed to be less than or equal to the average reconstruction error from the base classifiers. The improvement can be decomposed into two terms, namely *diversity* and *coherence*. The former measures the variability of individual classifiers learned from different perspectives which shares the same argument as the analysis of single task ensemble model (Brown and Kuncheva, 2010), the later measures the correlation of microlabel predictions which higher correlation gives better improvements.

5.3 Random Spanning Tree Approximation (RTA)

Random Spanning Tree Approximation (RTA) model developed in Publication V is a major step forward of MAM. The goal of RTA is to bring in joint learning and inference framework as well as to provide extensive theory to guarantee the performance and the tractability of the inference. Especially, RTA explores the potential of tackling the intractable graph inference problem with a set of random spanning trees. Thus, it lays the foundation of performing inference on unknown structure to achieve accurate optimization with attainable computational efforts.

5.3.1 Background and Introduction

Limiting the applicability of structured output learning methods is the fact that the output structure is assume to be observed. However, it is often prohibitive to obtain the correct output structure, sometimes it is arguably a even harder problem that the structured output learning itself (Chickering et al., 1994). In stead, one can opt to the *complete graph*, in which there is an edge connecting each pair of vertices, as the output graph structure and rely on the learning algorithm to discover the proper 'parameters' defined for the edges (e.g. edge potentials). To understand this, think the edges that are not from the correct output graph will have zero 'parameters' in the complete graph.

Structured output learning with complete graph is by no mean a easier problem as the inference is $\mathcal{NP}$-hard in nature, which is often instantiated as finding the *maximum a posteriori* (MAP) configuration on a graph-structured probability distribution. In terms of the intractability issue of graph inference problem, many techniques have been proposed but with important differences. Jordan and Wainwright (2004) developed semi-definite programming convex relaxation for inference on graph with cycles. Wainwright et al. (2005) proposed MAP inference with tree-based and *linear programming* (LP) relaxation. Efficient inference on special case of graph has also been extensively studied in (Globerson and Jaakkola, 2007).

Publication V is motivated by well-established max-margin assumption presented in Section 2.2.4 and investigate whether the problem of inference over a complete graph in structured output learning can be avoided in a analogous way. Start from a sampling results, Publication V, Lemma 3 shows that with high probability a big fraction of margin from the complete graph can be obtained by a sample of spanning trees of small size. Besides, Publication V, Theorem 5 shows the good generalization can also be guaranteed when learning with in stead of complete graph a sample of spanning trees.

Thus, in addition to Publication IV, Theory 1, we further provide the theoretical justification of MAM that constructs a set of base classifiers trained on random spanning trees. Besides, Publication V, Theorem 5 suggests we should optimize the joint margin from all spanning trees, instead of optimizing them separately as MAM, which leads to the RTA model described in the following section.

5.3.2 Methods

Denote by $\mathcal{T} = \{T_1, \cdots, T_n\}$ a sample of n random spanning trees, and by $\{\mathbf{w}_{T_t} | T_t \in \mathcal{T}\}$ the feature weights to be learner on each tree. The goal of optimization is to maximize the joint margin from all spanning trees between correct training examples and other examples.

Definition 14. Primal L_2-norm Random Tree Approximation (RTA).

$$\begin{aligned} \underset{\mathbf{w}_{T_t}, \xi_i}{\boldsymbol{min}} \quad & \frac{1}{2}\sum_{t=1}^{n} ||\mathbf{w}_{T_t}||_2^2 + C\sum_{i=1}^{m}\xi_i \\ \boldsymbol{s.t.} \quad & \sum_{t=1}^{n}\langle \mathbf{w}_{T_t}, \boldsymbol{\phi}_{T_t}(\mathbf{x}_i, \mathbf{y}_i)\rangle - \underset{\mathbf{y}\neq\mathbf{y}_i}{\boldsymbol{max}} \sum_{t=1}^{n}\langle \mathbf{w}_{T_t}, \boldsymbol{\phi}_{T_t}(\mathbf{x}_i, \mathbf{y})\rangle \geq 1 - \xi_i, \\ & \xi_i \geq 0\,, \forall\, i \in \{1, \ldots, m\}, \end{aligned}$$

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.

The key for the optimization is to solve the '**argmax**' problem efficiently. This is a $\mathcal{NP}$-hard problem in practice, as the size of the multilabel space is exponential in terms of the number of microlabels. For the optimization problem, we proposed in Publication V a K-best inference algorithm working in $\Theta(Knk)$ time per data point, where k is the number of microlabels in the multilabel vector and K is the number of best multilabels we compute from each random spanning tree.

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

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

can be solved in $\Theta(k)$ time by *dynamic programming* also known as *max-product* or *min-sum*. However, there is no guarantee the the maximizer of (5.1) is also the global maximizer of (Definition 14) over all spanning trees. Therefore, we compute for each random spanning tree K-best multilabels, which in total costs $\Theta(Knk)$ time for all spanning trees. Publication V, Lemma 7 provides a method to retrieve the best multilabel from K-best list in linear time. Now the question is to make sure the global violation multilabel is within the list. Publication V, Lemma 8 guarantees that with high probability the global violation multilabel is in the list and K does not need to be large.

In addition, we derived the marginalized dual representation of (Definition 14) which not only enjoys a polynomial sized parameter space but also enable kernel function to deal with complex input space.

6. Evaluation and Comparison Methods

In this chapter, the author will briefly introduce the performance measures and statistical test methods that are involved in the research. These methods enable us to quantitatively evaluate the performance of the proposed models and to measure the significancy comparing to other well established methods.

6.1 Performance Measures

It is often difficult and expensive to obtain labeled data. In order to maximize the outcomes of the valuable labeled data, the common approach adopted by machine learning community is known as K-*fold cross-validation*. In particular, the origin set of labeled examples are partitioned into K disjoint subsets of same size denoted by $\{S_1, \cdots, S_K\}$. The cross-validation framework will repeat training and test procedures K times. In the i'th iteration, the method uses $K-1$ subsets of labeled examples $\cup_{j\in\{1,\cdots,K\}/i} S_j$ to build a training set in order to construct a classifier. The examples in the i'th subset is taken as test data of which the labels are assumed to be unknown to the classifier. The resulting classifier is used to predict the label of each example from the test set S_i. The performance is evaluated by comparing to the true labels from the test set S_i. In extreme case where we have K equal to the number of original labeled examples, there will be only one example in each test set. This is typically known as *leave-one-out cross-validation* (LOO).

Most machine learning algorithms explicitly assume that the training data and the test data are drawn from the same distribution. We should make sure that the statistical property of the original labeled data is well preserved when partitioning the original labeled examples into subsets. In single-label classification problem we can easily preserve the label dis-

tribution if we randomly sample labeled examples without replacement from the origin dataset. However, it is not clear how to preserve label distribution under multilabel classification setting. To this end, we use *stratification* (*stratified sampling*) method where we first group the labeled examples into equivalent classes based on the number of positive labels they have. Each class is then randomly split into K local fold, after that local folds are merged to create K global fold. The stratification scheme ensures that also smaller classes have representations in all folds.

After partitioning the original data into training set and test set, we adopt several well-known performance measures in order to quantitively evaluate the performance of difference classifiers. We report *multilabel accuracy* (*0/1 accuracy*) which counts the percentage of multilabel predictions that have all of the microlabel being correct. The counterpart is often known as *multilabel loss* (*0/1 loss*) defined as

$$\ell_{0/1} = \frac{1}{m} \sum_{i=1}^{m} [\![\mathbf{y}_i \neq \hat{\mathbf{y}}_i]\!],$$

where by $\mathbf{y}_i$ we denote the predicted multilabel of the i'th example $\mathbf{x}_i$, and by $\hat{\mathbf{y}}_i$ we denote ground true multilabel. We also report *microlabel accuracy* as the proportion of microlabel being correct, of which the counterpart is often known as *hamming loss* defined as

$$\ell_h = \frac{1}{mk} \sum_{i=1}^{m} \sum_{j=1}^{k} [\![\mathbf{y}_i[j] \neq \hat{\mathbf{y}}_i[j]]\!].$$

In addition, we report F_1 *score* as the harmonic mean of *microlabel precision* and *microlabel recall* defined as

$$F_1 = 2 \cdot \frac{Pre \times Rec}{Pre + Rec}.$$

Microlabel precision and microlabel recall are defined as

$$Pre = \frac{TP}{TP + FP}, \quad Rec = \frac{TP}{TP + FN},$$

where we use TP, FP, and FN to denote *true positive*, *false positive*, and *false negative* predictions respectively.

6.2 Statistical Tests

Once we have computed the performance measures of multiple machine learning algorithms, we want to compare the performance in order to decide the best model under consideration. To serve as a starting point,

we assume a very basic setting where we already have the performance of multiple machine learning algorithms on one selected dataset. It is straight forward to compare the performance of the algorithms in terms of the performance metrics and choose the best model based on the value of the metric without resorting to any statistical methods. We only assume that the metrics accurately reveal the performance of the algorithms. However, when a new learning algorithm is proposed, it is usually evaluated over multiple dataset to rule out the probability that it performs well on one particular dataset by chance. Statistical evaluation of experimental results have been considered as an essential part of comparison steps in such complicate scenarios. We will base our discuss on (Demšar, 2006; García and Herrera, 2008) in the following part of the section.

We first consider the case where we want to compare two learning algorithms over multiple datasets. In order to decide the best performing method, one way is to average the performance over all test datasets. However, it is debatable that the error rate from different domains (test datasets) are commensurable. Beside, the averaging is susceptible to the outliers. Another approach is to use paired t-test to check if the average difference of two learning algorithms over datasets is significantly different from zero. Paired t-test is problematic due to commensurability. Besides, the result will not be significant unless the sample size is larger enough. Demšar (2006) proposed Wilcoxon signed-ranks test which is a non-parametric alternative to the paired t-test to compared two algorithms over multiple datasets. Meanwhile, sign-test is also proposed in (Demšar, 2006). As the setting is not very interesting for our purpose, we will not expand our discussion on the details.

Secondly, we will consider a more common situation where we want to compare multiple learning algorithms over many datasets. In practice, we want to answer the question, for example, whether algorithm A is significantly better than algorithm C, and algorithm C outperformances algorithm B in a significantly manner. In statistics, this amounts to multiple hypothesis test. The common way is to use repeat-measures ANOVA (Fisher, 1959) with post-hoc Tukey test (Tukey, 1949). The test procedure is based on two assumptions: the first is known as *normality* meaning samples are drawn from normal distributions; the second is known as *sphericity* meaning samples should have equal variant. It is not difficult to see that the ANOVA is ill-posed as none the assumptions are met within the context of comparing the performance of learning algorithms

over datasets.

To alleviate the problems, Demšar (2006) suggest to use Friedman test (Friedman, 1937, 1940) with post-hoc Nemenyi test (Nemenyi, 1963). The goal of the test procedure is to examine whether a pair of algorithms are significantly different from each other under multiple hypothesis test setting. To perform Friedman test, we first compute the average rank R_j of the j'th algorithms. The Friedman test then compares the average ranks and under null-hypothesis states that the average ranks are equal. The test statistics can be computed according to the procedure discussed in (Demšar, 2006). If null-hypothesis is rejected, we can proceed with post-hoc Nemenyi test to compare algorithms with each other. In particular, we can compute the *critical different* (CD) and state that the performance of two classifiers are significantly different if the corresponding average rank differs by at least CD. As the post-hoc Nemenyi test is usually conservative and has little statistical power, we sometimes use Bonferroni-Dunn test (Dunn, 1961) to compare the learning algorithms against the selected control which is usually the newly proposed algorithm. The test results are often visualized as CD *diagram*.

7. Implementations

Developed methods are implemented as software packages and can be located from public domains.

1. RTA for joint learning and inference on a random sample of spanning trees for multilabel classification.
2. SPIN for network influence prediction.
3. MVE, AMM, MAM for learning an ensemble on a set of structured output learners built from random spanning trees.
4. Structured output prediction model for molecular activities classification problem.

8. Conclusion

8.1 Discussion

In this thesis, we study supervised learning in structured output space of which the task is to predict the best values for multiple interdependent variables given an arbitrary input variable. In particular, we focus on the problem of structured output learning when the output structure is unknown. We propose a learning framework that aims to optimize on a random sample of spanning trees as output structure. The novel learning framework is well motivated in terms of margin achieved by training examples, and the performance in terms of generalization error is also guaranteed. In addition to the new learning framework, we also develop a new structured output prediction method that is able to predict a directed acyclic graph (DAG) and apply the methods on network influence prediction problem. We also tackle the optimization problem of the proposed models for which the exponential sized parameter space and the $\mathcal{NP}$-hard inference problem can usually be avoided.

We have posed several research questions at the beginning of the thesis which serve as the guide through the presentation of the work. Section **??** and Section **??** introduce the backgrounds of both single-task and multi-task learnings. The sections also present several well-established algorithms that are relevant to the thesis. Section **??** discuss a few ensemble approaches that are close related to the learning framework developed in the thesis. After the introduction, Chapter **??** devotes to bringing in novel statistical models for structured output prediction that are developed by the author. We will revisit the research questions with the insights and the results we gain from the novel statistical models that are briefly presented in the thesis as well as detailed in the publications. In particular,

we give the answer to the following questions.

Q I: Is the structured output prediction models more suitable for predicting multiple interdependent variables in structured output space?

Multi-task structured output prediction problem presented in Section **??** requires multiple output variables to be predicted at the same time. The structured output prediction models are more suitable for the problem as it describes the interdependencies between the output variables and is more efficient compared to the single-task models presented in Section **??**. Besides, we have demonstrated the performance of the structured output prediction models in molecular activity prediction problem presented in Publication II and in network influence prediction problem presented in Publication I.

Q II: How to tackle the problem of structured output learning without observed output structure?

As it is often difficult to get the output structure that describes the interdependencies of the output variables, we have developed several algorithms which eventually lead to a general learning framework that learns from a random sample of spanning trees. In particular, we have developed the following learning algorithms. MVE which collects and postprocesses the predictions from each spanning tree. AMM and MAM both performs parameter learning based on individual random spanning tree. The only difference is that the inference of AMM is also computed on individual tree, on the other hand, the inference of MAM is based on the consensus graph by pooling edges from all spanning trees. Furthermore, RTA brings in the joint learning and inference such that all random spanning trees are optimized toward the same global objective.

Q III: What is the motivation of the proposed structured output learning framework? Can we explain the improvement and guarantee the empirical risk of the proposed models?

The proposed learning framework is designed for structured output prediction problem with unknown output structure. Learning with any observed structure can be seen as a special case of learning with a complete graph. The idea of learning from a random sample of spanning trees as output graph has been explained in Publication V as an approximation to learning with a complete graph in terms of

margin achieved on a set of training examples. Thus, the proposed new learning framework is well motivated. The improvement of the performance can be explained in terms of reconstruction error of the compatibility scores as in Publication I. Publication V also provide the generalization error for the proposed model.

Q IV: Can we efficiently optimize the proposed structured output prediction models?

The optimization problem in structured output prediction is usually difficult to optimize for two reasons, exponential sized parameter space due to the number of possible solutions and the $\mathcal{NP}$-hard inference problem instantiated as MAP configuration on the general graph. To avoid the exponential sized parameter space, we use marginalize dual decomposition technique which reduces the parameter space to polynomial size in terms of the number of edges in the output graph. We used different strategies to overcome the $\mathcal{NP}$-hard inference problem. We applied loopy belief propagation for the inference problem defined on the general graph structure with approximate solution. We used dynamic programming for the inference problem defined on a tree structure where the exact solution is guaranteed. We developed K-best inference algorithm in Publication V for the inference problem defined on a set of random spanning trees. We have also developed SDP inference algorithm for the inference problem defined on directed acyclic graph (DAG).

8.2 Future Work

What has been shown in the thesis is a new learning framework with algorithm instantiations for structured output prediction problems where the structure of the output is not known but is expected to play an important role. An immediate extension is to explore its enormous potentials in real world practical data analysis applications with the significant benefit that we do not need to know the output structure as *prior*. In addition, since the inference algorithm designed for optimizing over a random sample of spanning tree is tractable both in theoretic analysis and empirical results, we can apply the model on the structure prediction problems for which the output structure is rather complex.

With respect to algorithm development and theoretical study, we plan to

continue with RTA framework that performs joint learning and inference over a random sample of spanning trees. In particular, we will explore the possibility of applying L_1 norm combination of the feature weight parameter $\mathbf{w}_{T_t}$ of individual random spanning tree, instead of using convex L_2 norm combination. Learning with L_1 norm regularization of parameter combination can be expressed into an equivalent form of learning weighted L_2 norm regularization (Rakotomamonjy et al., 2008), where the weight of $\{\mathbf{w}_{T_t}\}_{t=1}^{T}$ is the extra parameter during the optimization. We also gain benefit by seeing the weight as the fitness of each spanning tree to current training data. Similar to L_2 norm, the studies on generalization error and the condition for the tractable inference are required. Besides the alternative optimization for the L_1 norm combination in (Rakotomamonjy et al., 2008), we plan to develop more elegant optimization strategy for the problem.

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 *Advances in Neural Information Processing Systems 19*. 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 - Part I*, ECML PKDD '08, 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 *Advances in Neural Information Processing Systems 17 (NIPS 2005)*, pages 113–120. MIT Press.

Baxter, J. (2000). A model of inductive bias learning. *J. Artif. Int. Res.*, 12(1):149–198.

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 of *Lecture Notes in Computer Science*, pages 567–580. Springer Berlin Heidelberg.

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. *J. Chem. Inf. Model.*, 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.

Block, H. D. (1962). The perceptron: A model for brain functioning. i. *Rev. Mod. Phys.*, 34:123–135.

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

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *J. Mach. Learn. Res.*, 6:1579–1619.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual workshop on Computational learning theory*, pages 144–152. ACM.

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.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Brinker, K. and Hüllermeier, E. (2007). Case-based multilabel ranking. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, 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. *J. Chem. Inf. Comput. Sci.*, 43:1882–1889.

Caruana, R. (1997). Multitask learning. *Mach. Learn.*, 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.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27.

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. *Speech and Audio Processing, IEEE Transactions on*, 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*, 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.

Collobert, R., Bengio, S., and Bengio, Y. (2002). A parallel mixture of svms for very large scale problems. *Neural Computation*, 14(5):1105–1114.

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.

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. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):380–393.

Dembczynski, K., Cheng, W., and Hüllermeier, E. (2010). Bayes optimal multilabel classification via probabilistic classifier chains. In Fürnkranz, J. and Joachims, T., editors, *ICML*, pages 279–286. Omnipress.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

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. *J. Mach. Learn. Res.*, 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., Smola, A. J., and Yuan, M. (2012). Learning networks of heterogeneous influence. In *Advances in Neural Information Processing Systems (NIPS)*.

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64.

Efron, B. and Tibshirani, R. (1994). An introduction to the bootstrap (crc, boca raton, fl).

Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In *In 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.

Fine, S. and Scheinberg, K. (2002). Efficient svm training using low-rank kernel representations. *J. Mach. Learn. Res.*, 2:243–264.

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

Fisher, S. R. A. (1959). Statistical methods and scientific inference.

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.

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):pp. 675–701.

Friedman, M. (1940). A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

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

García, S. and Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *J. Mach. Learn. Res.*, 9:2677–2694.

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 of *Lecture Notes in Computer Science*, 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 '02, pages 9–16.

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

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.

Graf, H. P., Cosatto, E., Bottou, L., Durdanovic, I., and Vapnik, V. (2005). Parallel support vector machines: The cascade svm. In *In Advances in Neural Information Processing Systems*, pages 521–528. MIT Press.

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 '04, pages 491–501, New York, NY, USA. ACM.

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.

Hsieh, C.-J., Si, S., and Dhillon, I. (2014). A divide-and-conquer solver for kernel support vector machines. In *Proceedings of the 31th International Conference on Machine Learning (ICML 2014)*, pages 566–574. JMLR.org.

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 *In Proceedings of the 7th Workshop on Artificial Intelligent and Statistics*. Morgan Kaufmann.

jacob, L., Bach, F., and Vert, J.-P. (2009). Clustered multi-task learning: a convex formulation. In *Advances in Neural Information Processing Systems 16 (NIPS 2009)*, pages 25–32. MIT Press.

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-03)*, pages 282–289.

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. *J. Mach. Learn. Res.*, 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.

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.

Kondor, R. I. and Lafferty, J. D. (2002). Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 315–322, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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*, pages 282–289. Morgan Kaufmann Publishers Inc.

Le, Q., Sarlos, T., and Smola, A. (2013). Fastfood - approximating kernel expansions in loglinear time. In *30th International Conference on Machine Learning (ICML)*.

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)*.

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, F. T., Ting, K. M., Yu, Y., and Zhou, Z.-H. (2008). Spectrum of variable-random trees. *J. Artif. Int. Res.*, 32(1):355–384.

Menchetti, S., Costa, F., and Frasconi, P. (2005). Weighted decomposition kernels. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 585–592, New York, NY, USA. 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.

Nemenyi, P. B. (1963). Distribution-free multiple comparisons. *PhD thesis, Princeton University*.

Novikoff, A. B. (1962). On convergence proofs on perceptrons. *Symposium on the Mathematical Theory of Automata*, 12:615–622.

Osuna, E., Freund, R., and Girosi, F. (1997). An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, 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. Journal of Machine Learning Research W&CP.

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 of *Lecture Notes in Computer Science*, pages 254–269. Springer Berlin Heidelberg.

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

Rodriguez, M. G., Balduzzi, D., and Schölkopf, B. (2011). Uncovering the temporal dynamics of diffusion networks. In Getoor, L. and Scheffer, T., editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 561–568, New York, NY, USA. 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 Lawrence, N. D. and Girolami, M., editors, *AISTATS*, volume 22 of *JMLR Proceedings*, 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.

Rosenblatt, F. (1962). Principles of neurodynamics.

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*, NAACL-Short '06, 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 of *Lecture Notes in Computer Science*, 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. *Mach. Learn.*, 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 Fourteenth International Conference on Machine Learning*, ICML '97, pages 322–330, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Schwaighofer, A. and Tresp, V. (2001). The bayesian committee support vector machine. In Dorffner, G., Bischof, H., and Hornik, K., editors, *Artificial Neural Networks — ICANN 2001*, volume 2130 of *Lecture Notes in Computer Science*, pages 411–417. Springer Berlin Heidelberg.

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 Jebara, T. and Xing, E. P., editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 701–709. JMLR Workshop and Conference Proceedings.

Smola, A. J. and Schökopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 911–918, San Francisco, CA, USA. 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'02, pages 485–492, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Taskar, B., Lacoste-Julian, S., and Jordan, M. I. (2006). Structured prediction via the extragradient method. In *Advances in Neural Information Processing Systems 18 (NIPS 2006)*, 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.

Tresp, V. (2000). A bayesian committee machine. *NEURAL COMPUTATION*, 12:2000.

Trotter, M., Buxton, M., and Holden, S. (2001). Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Comp. and Chem.*, 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, New York, NY, USA. 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 Kok, J., Koronacki, J., Mantaras, R., Matwin, S., Mladenič, D., and Skowron, A., editors, *Machine Learning: ECML 2007*, volume 4701 of *Lecture Notes in Computer Science*, pages 406–417. Springer Berlin Heidelberg.

Tukey, J. W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, 5(2):99–114.

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.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010). Graph kernels. *J. Mach. Learn. Res.*, 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.

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., 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. *J. Chem. Inf. Comput. Sci.*, 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 Lawrence, N. D. and Girolami, M. A., editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, 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 '08, pages 1232–1239, New York, NY, USA. ACM.

Zhang, M. and Zhou, Z. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *Granular Computing, 2005 IEEE International Conference on*, 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. *Knowledge and Data Engineering, IEEE Transactions on*, 18(10):1338–1351.

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

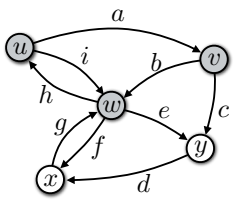
2. Preliminaries

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

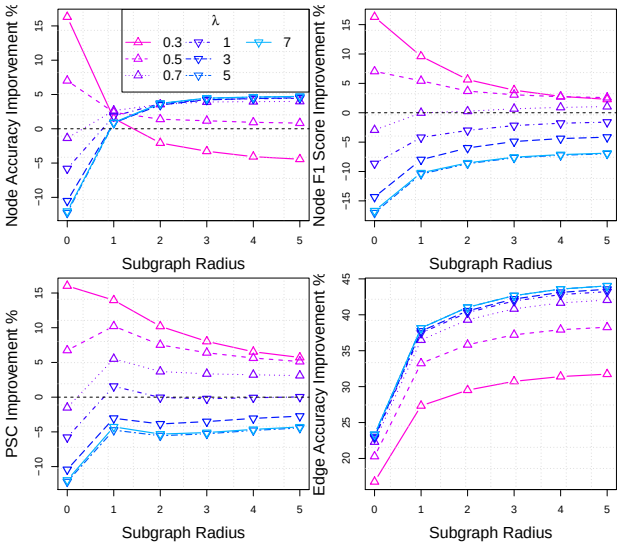


Figure 2. The improvement of prediction performance for different scaling factor λ with respect to SVM.

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`

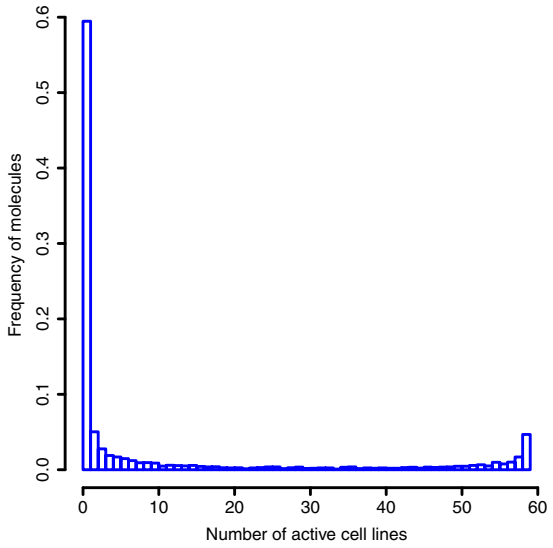


Fig. 1. Skewness of the multilabel distribution

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:

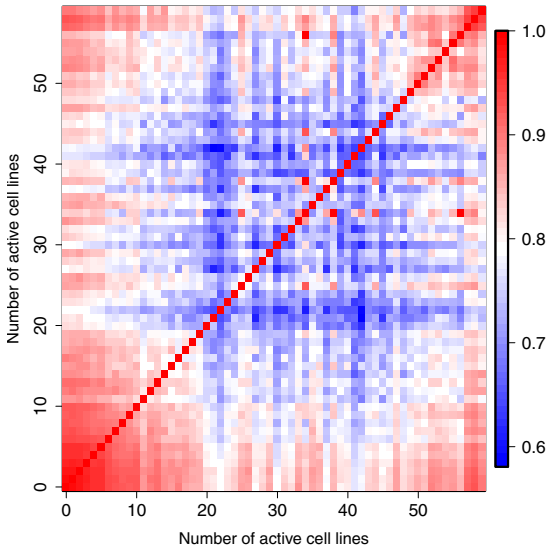


Fig. 2. Heatmap of the kernel space for the molecules sorted by the multilabel distribution

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`

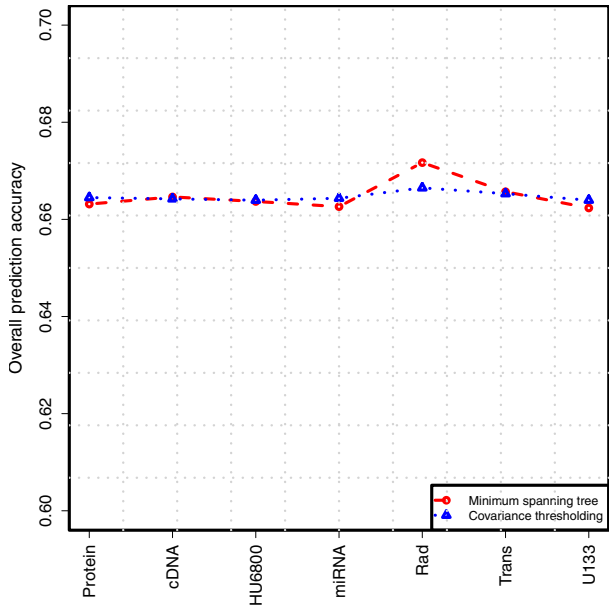


Fig. 3. Effects of Markov network construction methods and type of auxiliary data (from left to right: reverse-phase lysate arrays, cDNA arrays, Affymetric HU6800 arrays, miRNA arrays, RNA radiation arrays, transporter arrays, and Affymetrix U133 arrays)

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

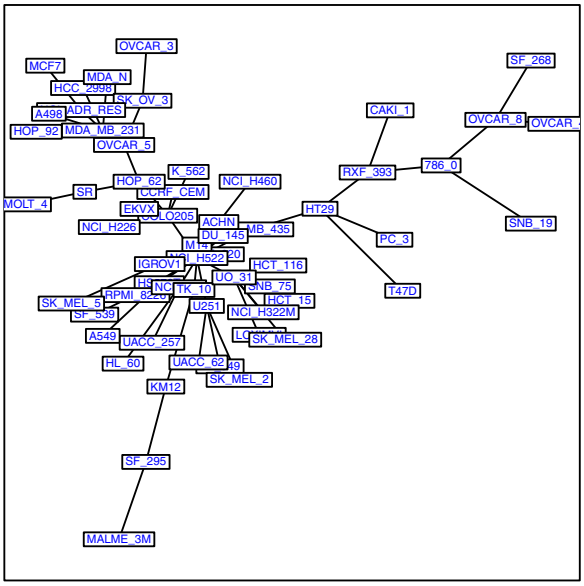


Fig. 4. Markov network constructed from maximum weighted spanning tree method on RNA radiation array data. The labels correspond to different cancer cell lines.

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.

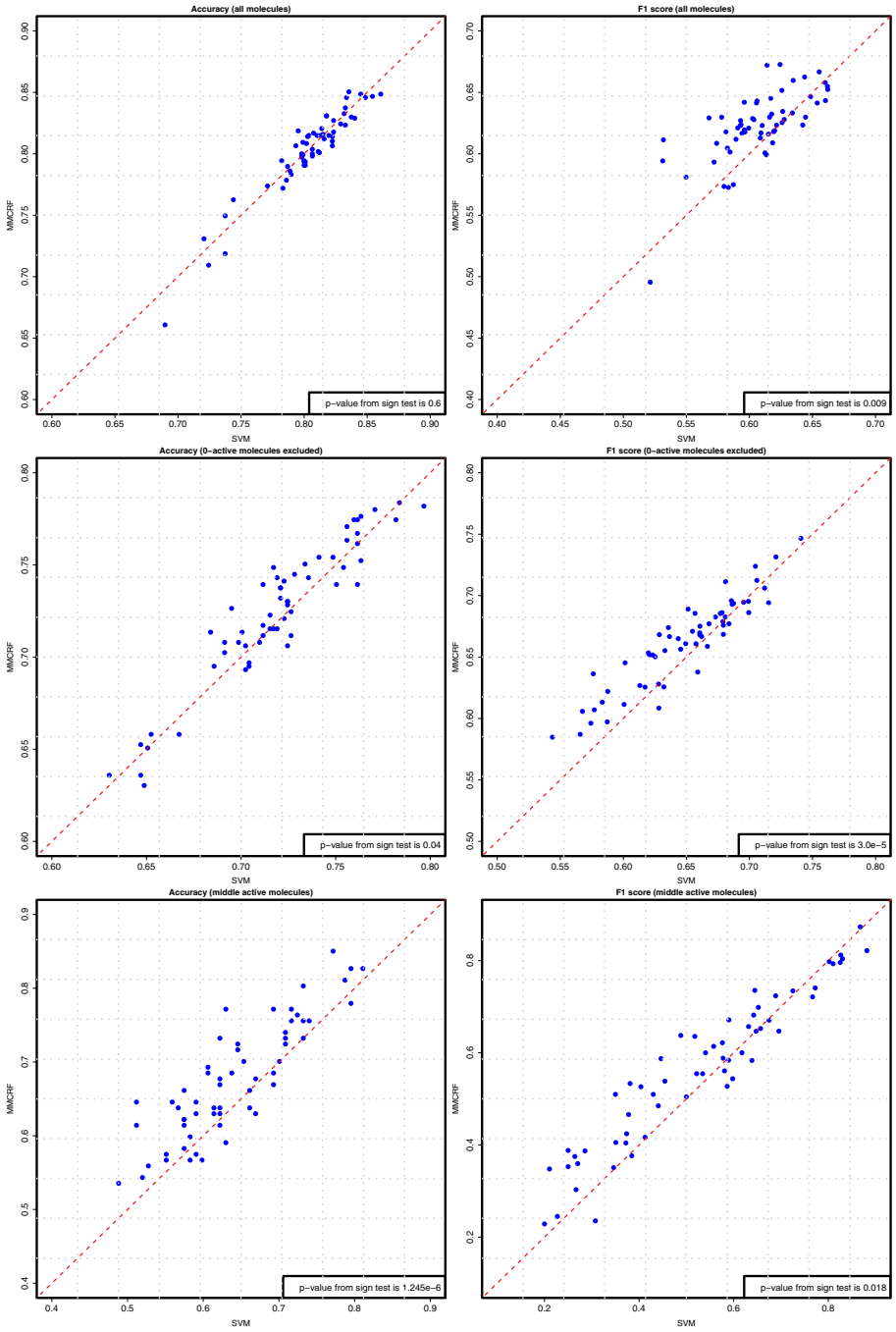


Fig. 5. Accuracy (left) and F1 score (right) of MMCRF vs. SVM on Full data (top), No-Zero-Active (middle) and Middle-Active molecules (bottom)

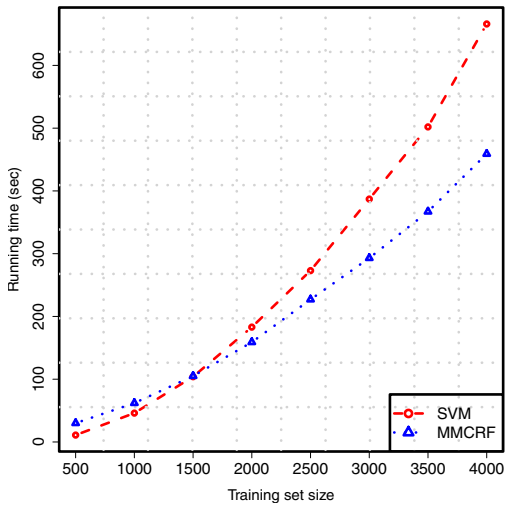


Fig. 6. Training time for SVM and MMCRF classifiers on training sets of different sizes

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].

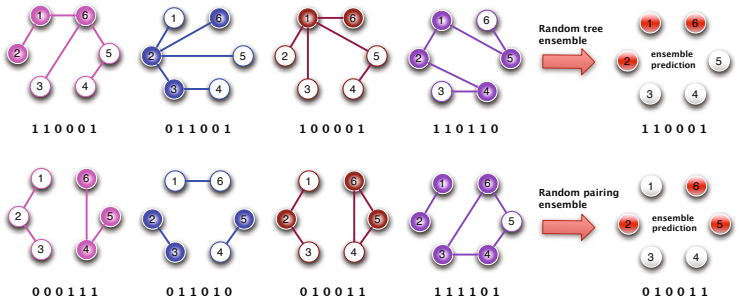


Fig. 1. Ensemble prediction from a set of random spanning trees (top) and a set of random pairings of tasks (bottom). The varying graph structures provide the required diversity among the ensemble components. Majority vote decides the final predicted label for each task.

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.

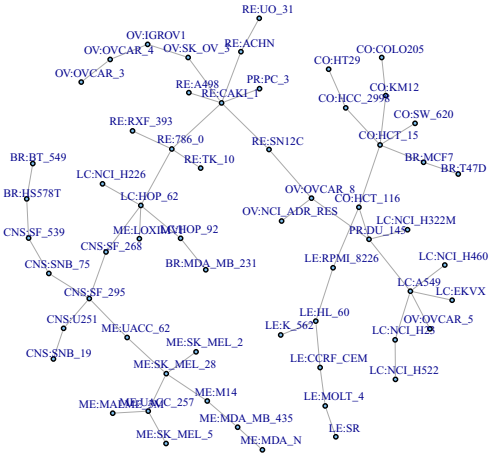


Fig. 2. Example of a cell line graph

- 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`

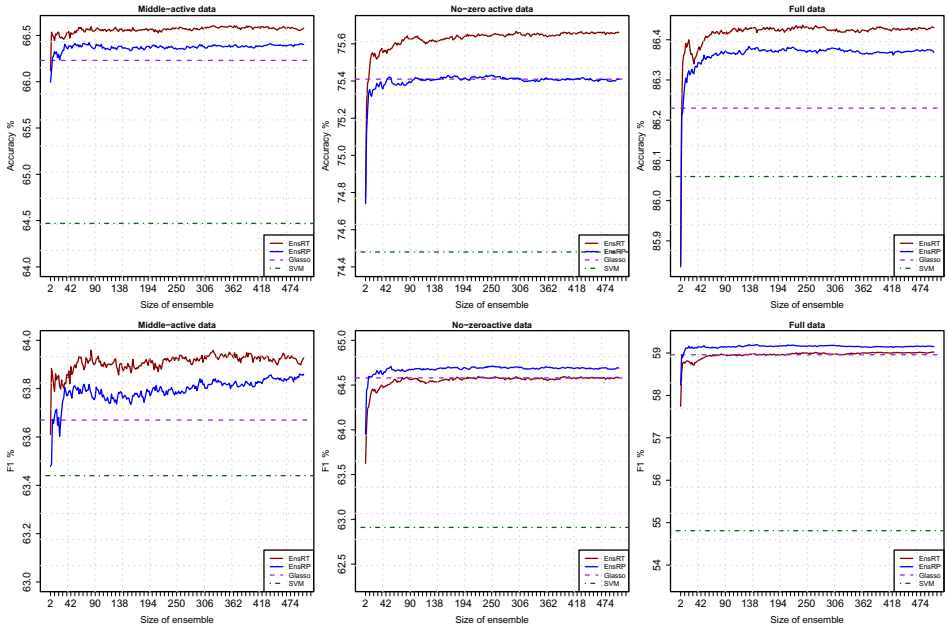


Fig. 3. Accuracy against number of individual classifiers in ensemble methods from different version of datasets. The red line corresponds to random tree ensemble, and blue line is random pairing ensemble. The performance of single models (SVM and MMCRF-Glasso) are depicted by the horizontal lines.

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*, Volume, issue, pages, and other detailed information, Month Year.

Springer

Dear Author,

Here are the proofs of your article.

- You can submit your corrections **online**, via **e-mail** or by **fax**.
- For **online** submission please insert your corrections in the online correction form. Always indicate the line number to which the correction refers.
- You can also insert your corrections in the proof PDF and **email** the annotated PDF.
- For fax submission, please ensure that your corrections are clearly legible. Use a fine black pen and write the correction in the margin, not too close to the edge of the page.
- Remember to note the **journal title**, **article number**, and **your name** when sending your response via e-mail or fax.
- **Check** the metadata sheet to make sure that the header information, especially author names and the corresponding affiliations are correctly shown.
- **Check** the questions that may have arisen during copy editing and insert your answers/ corrections.
- **Check** that the text is complete and that all figures, tables and their legends are included. Also check the accuracy of special characters, equations, and electronic supplementary material if applicable. If necessary refer to the *Edited manuscript*.
- The publication of inaccurate data such as dosages and units can have serious consequences. Please take particular care that all such details are correct.
- Please **do not** make changes that involve only matters of style. We have generally introduced forms that follow the journal's style.
  Substantial changes in content, e.g., new results, corrected values, title and authorship are not allowed without the approval of the responsible editor. In such a case, please contact the Editorial Office and return his/her consent together with the proof.
- If we do not receive your corrections **within 48 hours**, we will send you a reminder.
- Your article will be published **Online First** approximately one week after receipt of your corrected proofs. This is the **official first publication** citable with the DOI. **Further changes are, therefore, not possible.**
- The **printed version** will follow in a forthcoming issue.

Please note

After online publication, subscribers (personal/institutional) to this journal will have access to the complete article via the DOI using the URL: http://dx.doi.org/[DOI].
If you would like to know when your article has been published online, take advantage of our free alert service. For registration and further information go to: http://www.link.springer.com.

Due to the electronic nature of the procedure, the manuscript and the original figures will only be returned to you on special request. When you return your corrections, please inform us if you would like to have these documents returned.

Metadata of the article that will be visualized in OnlineFirst

Please note: Images will appear in color online but will be printed in black and white.

ArticleTitle	Multilabel classification through random graph ensembles	
Article Sub-Title		
Article CopyRight	The Author(s) (This will be the copyright line in the final PDF)	
Journal Name	Machine Learning	
Corresponding Author	Family Name	**Su**
	Particle	
	Given Name	**Hongyu**
	Suffix	
	Division	Department of Information and Computer Science, Helsinki Institute for Information Technology HIIT
	Organization	Aalto University
	Address	Konemiehentie 2, 02150 , Espoo, Finland
	Email	hongyu.su@aalto.fi
Author	Family Name	**Rousu**
	Particle	
	Given Name	**Juho**
	Suffix	
	Division	Department of Information and Computer Science, Helsinki Institute for Information Technology HIIT
	Organization	Aalto University
	Address	Konemiehentie 2, 02150 , Espoo, Finland
	Email	juho.rousu@aalto.fi
Schedule	Received	16 January 2014
	Revised	
	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 (separated by '-')	Multilabel classification - Structured output - Ensemble methods - Kernel methods - Graphical models
Footnote Information	Editors: Cheng Soon Ong, Tu Bao Ho, Wray Buntine, Bob Williamson, and Masashi Sugiyama.

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
Department of Information and Computer Science, Helsinki Institute for Information Technology HIIT, Aalto University, Konemiehentie 2, 02150 Espoo, Finland
e-mail: hongyu.su@aalto.fi

J. Rousu
e-mail: juho.rousu@aalto.fi

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

are given in form of a graph structure such as a sequence, a hierarchy or a network, structured output prediction (Taskar et al. 2003; 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

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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. 2003; 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

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

$$L(w, \xi, \alpha, \rho) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i,\mathbf{y}}\alpha(i,\mathbf{y})\left(\langle w, \Delta\varphi(x_i,\mathbf{y})\rangle + \ell(\mathbf{y}_i,\mathbf{y})\right) - \sum_i \xi_i\rho_i,$$

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 $\sum_y \alpha(i,y) \le C$ for all i, while the dual variables ρ_i are canceled out. Maximization with respect to α's gives the dual optimization problem as

$$\begin{aligned} \max_{\alpha\ge 0}\ & \alpha^T\ell - \frac{1}{2}\alpha^T K\alpha \\ \text{s.t.}\ & \sum_{\mathbf{y}}\alpha(i,\mathbf{y}) \le C, \forall i \in \{1,\ldots,m\}, \end{aligned} \tag{4}$$

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

$$K_\Upsilon(\mathbf{y},\mathbf{y}') = \langle\Upsilon(\mathbf{y}'),\Upsilon(\mathbf{y})\rangle = \sum_e K_{\Upsilon,e}(\mathbf{y}_e,\mathbf{y}'_e),$$

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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

$$K(x_i, \mathbf{y}; x_j, \mathbf{y}') = K_\phi(x,x') K_{\Delta\Upsilon}(\mathbf{y}_i, \mathbf{y}; \mathbf{y}_j, \mathbf{y}') = \\ = \sum_e K_\phi(x,x') K_{\Delta\Upsilon,e}(\mathbf{y}_e, \mathbf{y}'_e) = \sum_e K_e(x, \mathbf{y}_e; x', \mathbf{y}'_e),$$

where we have denoted

$$K_{\Delta\Upsilon,e}(\mathbf{y}_{ie}, \mathbf{y}_e; \mathbf{y}_{je}, \mathbf{y}'_e) = \\ \left(K_{\Upsilon,e}(\mathbf{y}_{ie}, \mathbf{y}_{je}) - K_{\Upsilon,e}(\mathbf{y}_{ie}, \mathbf{y}'_e) - K_{\Upsilon,e}(\mathbf{y}_e, \mathbf{y}_{je}) + K_{\Upsilon,e}(\mathbf{y}_e, \mathbf{y}'_e)\right).$$

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. 2003; Rousu et al. 2007) as

$$\max_{\mu\in\mathcal{M}} \boldsymbol{\mu}^T \ell - \frac{1}{2} \boldsymbol{\mu}^T K_E \boldsymbol{\mu}, \tag{8}$$

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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*

$$H_e(x_i, \mathbf{u}_e; x, \mathbf{y}_e) = K_\phi(x, x_i) \cdot \left(K_{\Upsilon,e}(\mathbf{y}_{ie}, \mathbf{y}_e) - K_{\Upsilon,e}(\mathbf{u}_e, \mathbf{y}_e) \right).$$

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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

$$\begin{aligned}
\bar{\mathbf{y}}(x) &= \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \sum_e \psi_e(x, \mathbf{y}_e) + \ell_e(\mathbf{y}_e, \mathbf{y}_{ie}) \\
&= \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \sum_{e,i,\mathbf{u}_e} \mu(i, e, \mathbf{u}_e) H_e(i, \mathbf{u}_e; x, \mathbf{y}_e) + \ell_e(\mathbf{y}_e, \mathbf{y}_{ie}). \qquad (11)
\end{aligned}$$

Similarly, the inference problem (10) solved in prediction phase can be solved in the factorized dual by

$$\begin{aligned}
\hat{\mathbf{y}}(x) &= \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \sum_e \psi_e(x, \mathbf{y}_e) = \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \sum_e \langle \mathbf{w}_e, \varphi_e(x, \mathbf{y}_e)\rangle \\
&= \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \sum_{e,i,\mathbf{u}_e} \mu(i, e, \mathbf{u}_e) H_e(i, \mathbf{u}_e; x, \mathbf{y}_e). \qquad (12)
\end{aligned}$$

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,

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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

$$F_j^{\mathrm{MVE}}(x) = \operatorname*{\mathbf{argmax}}_{y_j \in \mathcal{Y}_j} \left(\frac{1}{T} \sum_{i=1}^{T} \mathbf{1}_{\{F_j^{(t)}(x) = y_j\}} \right),$$

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

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

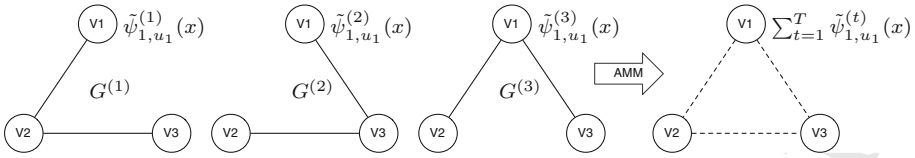


Fig. 1 An example of AMM scheme, where three base models are learned on the output graph $G^{(1)}, G^{(2)}, G^{(3)}$. Given an example x, each base model computes for node v_1 local max-marginals $\tilde{\psi}^{(t)}_{1,u_1}$ for all $u_1 \in \{+,-\}$. The AMM collects local max-marginals $\sum_{t=1}^{T}\tilde{\psi}^{(t)}_{1,u_1}(x)$, 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 $F_1(x) = -$

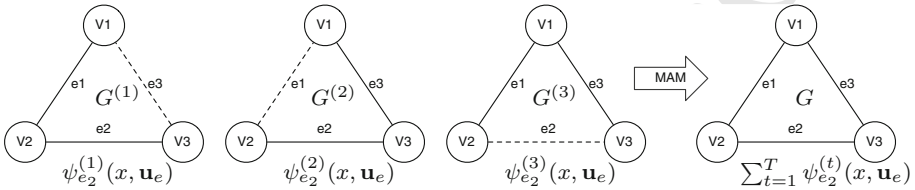


Fig. 2 An example of MAM scheme, where three base models are learned on the output graph $G^{(1)}, G^{(2)}, G^{(3)}$. Given an example x, each base model computes for edge e_2 local edge potentials $\psi^{(t)}_{e_2}(x,\mathbf{u}_e)$ for all $\mathbf{u}_e = \{--,-+,+-,++\}$. For graph $G^{(3)}$ where $e_2 \notin E^{(3)}$, 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

$$F^{\text{MAM}}(x) = \mathop{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \sum_{e\in E}\frac{1}{T}\sum_{t=1}^{T}\psi^{(t)}_{e}(x,\mathbf{y}_e) = \mathop{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \frac{1}{T}\sum_{t=1}^{T}\sum_{e\in E}\langle \mathbf{w}^{(t)}_{e}, \varphi_e(x,\mathbf{y}_e)\rangle. \tag{311}$$

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

$$\begin{aligned} F^{\text{MAM}}(x) &= \mathop{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \frac{1}{T}\sum_{t=1}^{T}\sum_{i,e,\mathbf{u}_e}\mu^{(t)}(i,e,\mathbf{u}_e)\cdot H_e(i,\mathbf{u}_e;x,\mathbf{y}_e) \\ &= \mathop{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \sum_{i,e,\mathbf{u}_e}\bar{\mu}(i,e,\mathbf{u}_e)H_e(i,\mathbf{u}_e;x,\mathbf{y}_e), \end{aligned} \tag{317, 318}$$

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.

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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

$$\Delta^R_I(x, \mathbf{y}) - \Delta^R_{MAM}(x, \mathbf{y}) = \boldsymbol{Var}\left(\sum_{j \in V} \Psi_j(x, y_j)\right) \geq 0.$$

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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

$$\begin{aligned}\Delta_I^R(x,\mathbf{y})-\Delta_{\mathrm{MAM}}^R(x,\mathbf{y}) &= \frac{1}{T}\sum_t\left(\psi^*(x,\mathbf{y})-\psi^{(t)}(x,\mathbf{y})\right)^2-\left(\psi^*(x,\mathbf{y})-\psi^{\mathrm{MAM}}(x,\mathbf{y})\right)^2\\ &= \frac{1}{T}\sum_t\left(\sum_{j\in V}\psi_j^*(x,y_j)-\sum_{j\in V}\psi_j^{(t)}(x,y_j)\right)^2\\ &\quad-\left(\sum_{j\in V}\psi_j^*(x,y_j)-\sum_{j\in V}\frac{1}{T}\sum_t\psi_j^{(t)}(x,y_j)\right)^2\\ &= \frac{1}{T}\sum_t\left(\sum_{j\in V}\psi_j^{(t)}(x,y_j)\right)^2-\left(\frac{1}{T}\sum_t\sum_{j\in V}\psi_j^{(t)}(x,y_j)\right)^2\\ &= \boldsymbol{Var}(\sum_{j\in V}\Psi_j(x,y_j))\\ &\geq 0.\end{aligned}$$

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.

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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.

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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.

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

- **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

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

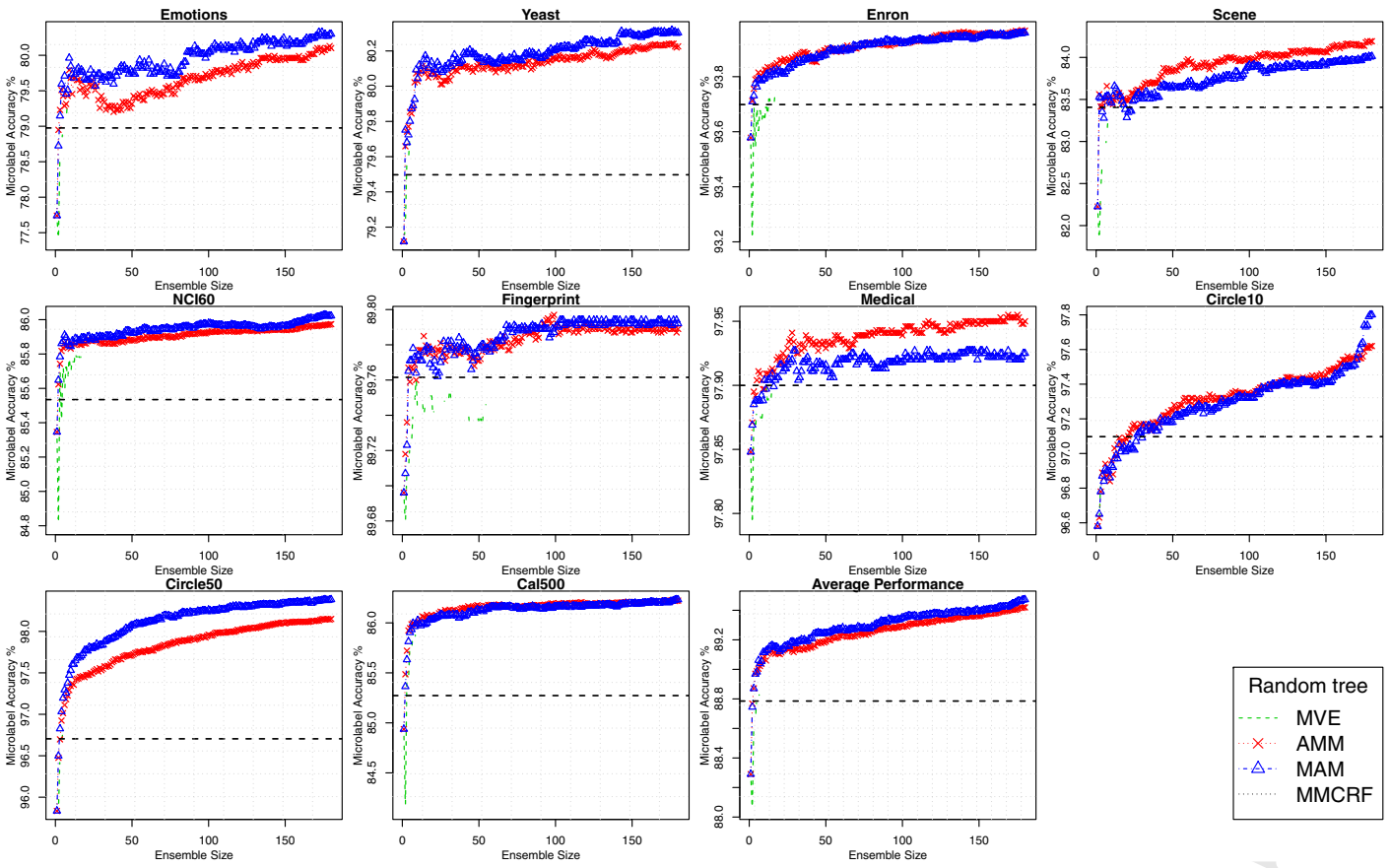


Fig. 3 Ensemble learning curve (microlabel accuracy) plotted as the size of ensemble. Average performance over datasets is shown as the last plot

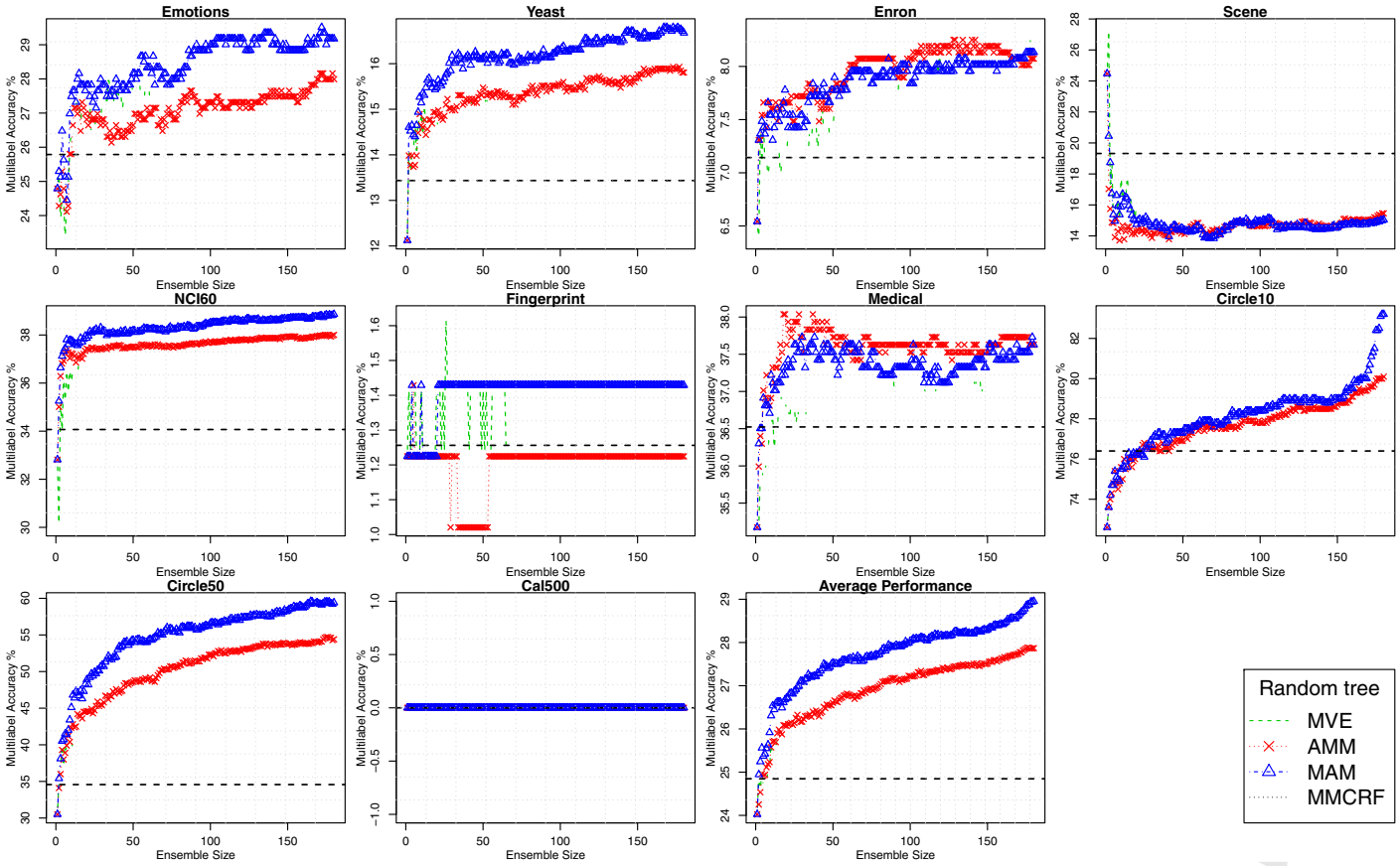


Fig. 4 Ensemble learning curve (multilabel accuracy) plotted as the size of ensemble. Average performance over datasets is shown as the last plot

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.

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

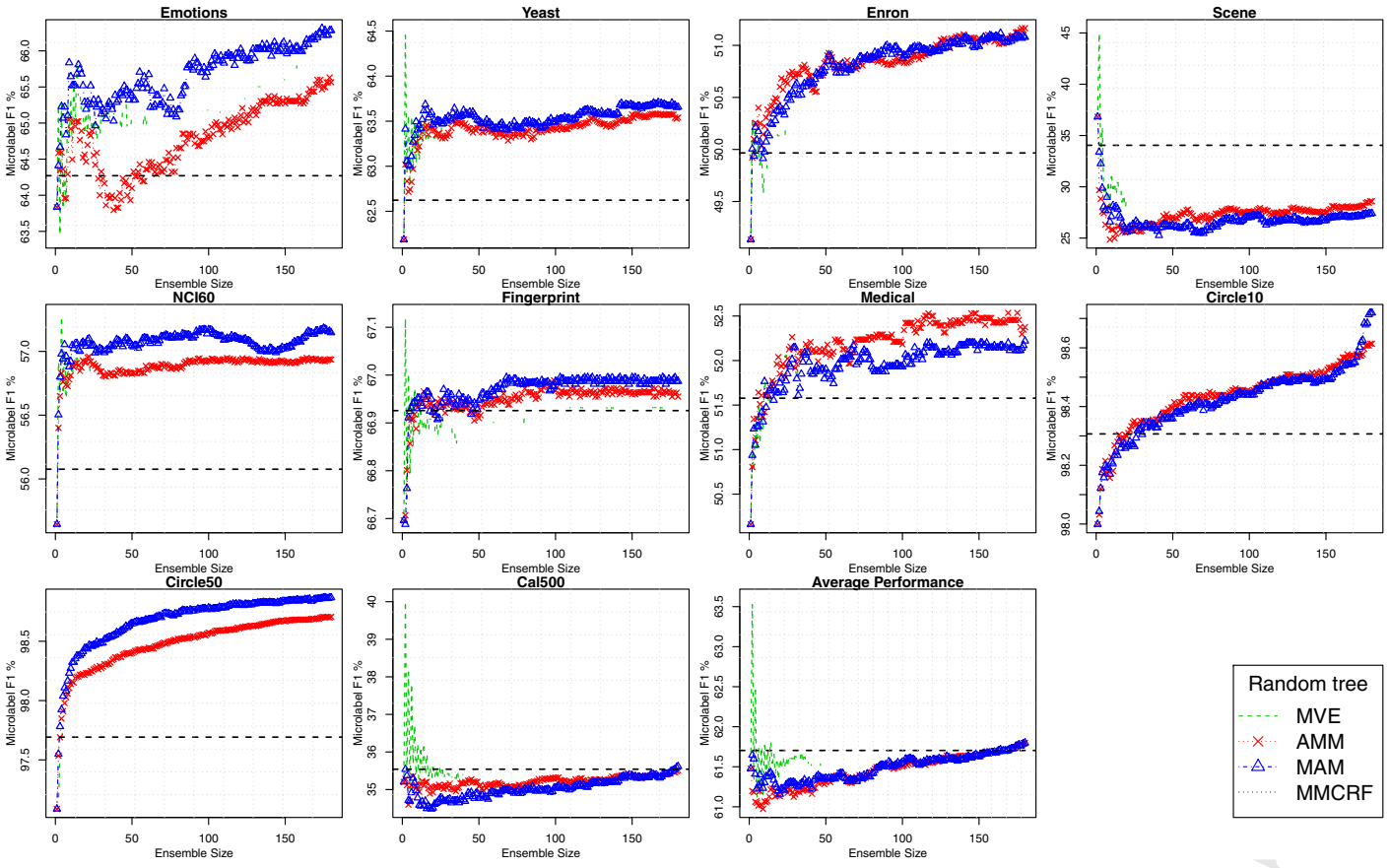


Fig. 5 Ensemble learning curve (microlabel F_1 score) plotted as the size of ensemble. Average performance over datasets is shown as the last plot

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

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

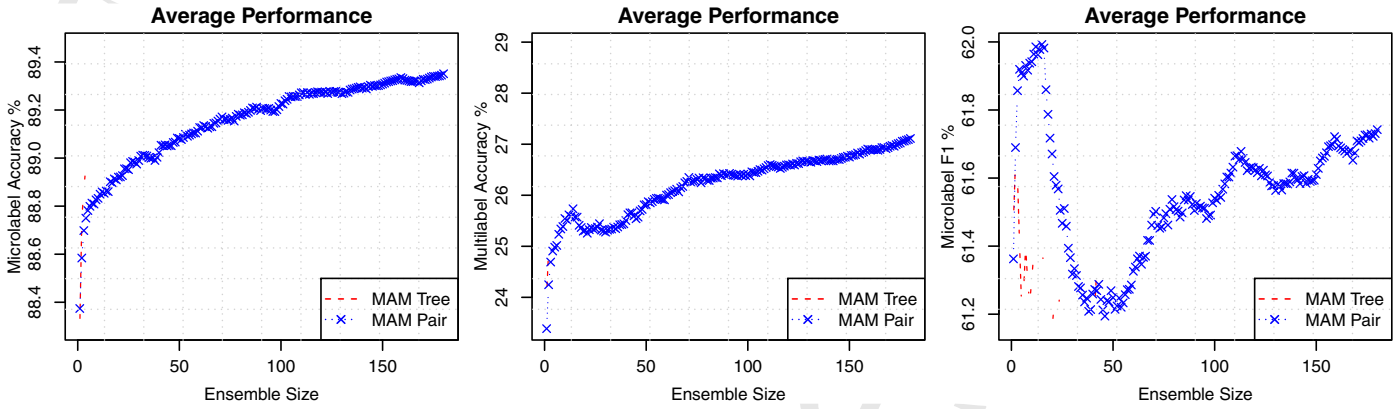


Fig. 6 Performance of MAM ensemble with random tree and random pair as output graph. Performance is averaged over 10 datasets and plotted as the size of ensemble

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

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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 '†' (at p-value = 0.1) or '††' (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.

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

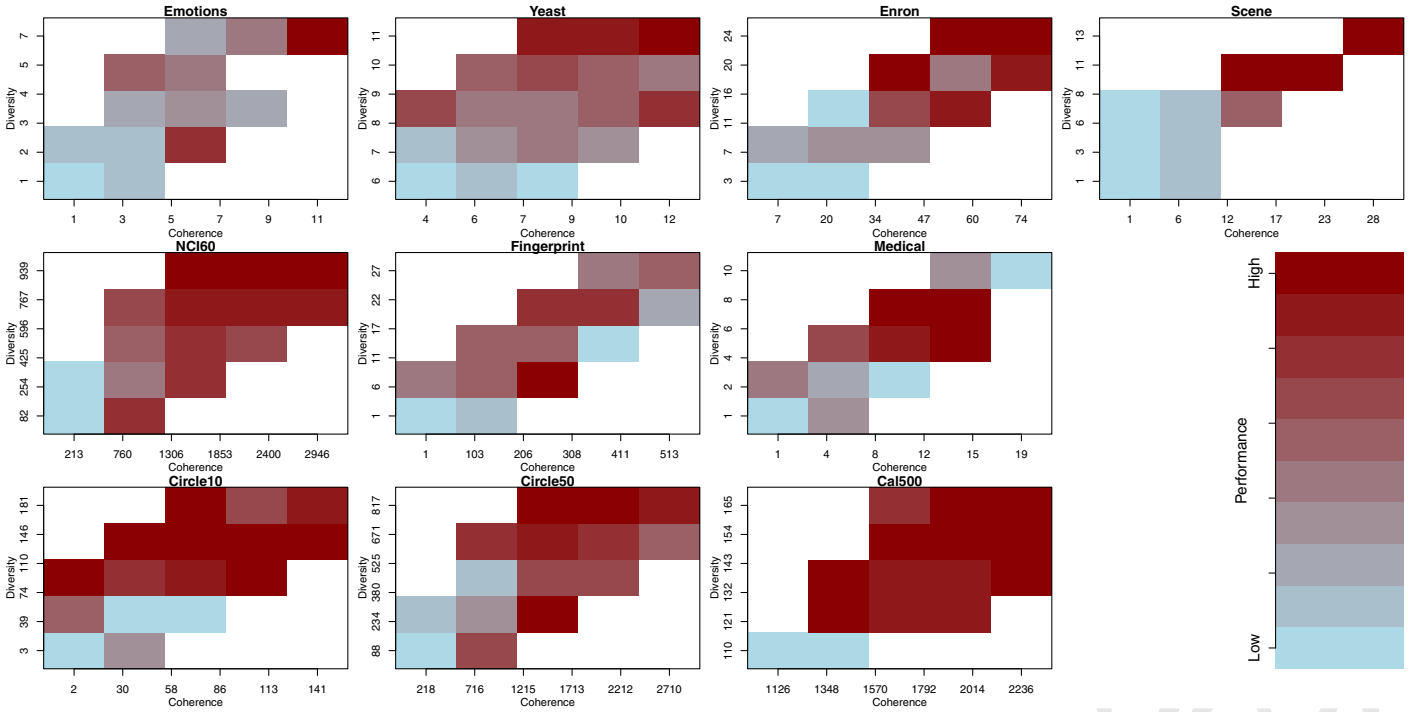


Fig. 7 Performance of MAM ensemble plotted in diversity and coherence space. Color of the blocks depicts average performance in term of microlabel accuracy comput
from the data points in the block. White area means there is no examples with corresponding diversity and coherence. Colors are normalized for datasets so that worst and be
performances are shown as light blue and red

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.

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

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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.

Nemenyi, P. B. (1963). Distribution-free multiple comparisons. PhD thesis, Princeton University.

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. *Predicting Structured Data*, 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. (2011). Multi-task drug bioactivity classification with graph labeling ensembles. *Pattern Recognition in Bioinformatics*, 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, C., & Koller, D. (2003). Max-margin markov networks. In: *Neural Information Processing Systems*, 25–32.

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.

Journal: Article No.: 5465 TYPESET DISK LE CP Disp.:2014/7/23 Pages: 27 Layout: Small

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

Journal: Article No.: **5465** TYPESET DISK LE CP Disp.:**2014/7/23** Pages: **27** Layout: **Small**

Journal: 10994
Article: 5465

Springer
the language of science

Author Query Form

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author

During the process of typesetting your article, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the 'Author's response' area provided below

Query	Details required	Author's response
1.	Reference Nemenyi (1963) is given in list but not cited in text. Please cite in text or delete from list.	
2.	Please supply the volume number for Refs. Rousu et al. (2006) and Su and Rousu (2011) if possible.	
3.	Please provide complete details for Ref. Taskar et al. (2003).	

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 *Proceedings of the* 28^{th} *Advances in Neural Information Processing Systems (NIPS 2014)*, to appear, December 2014.

Multilabel Structured Output Learning with Random Spanning Trees of Max-Margin Markov Networks

Mario Marchand
IFT-GLO, Université Laval
Québec (QC), Canada
mario.marchand@ift.ulaval.ca

Hongyu Su
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
Aalto University
Finland
juho.rousu@aalto.fi

John Shawe-Taylor
Dept of Computer Science, UCL
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 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

*Most of the work of this paper was carried out while E. Morvant was affiliated with IST Austria, Klosterneurburg.

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 used 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

$$\sum_{T \in S(G)} \sum_{(i,j) \in T} f\left((i,j)\right) = \ell^{\ell-2} \frac{2}{\ell} \sum_{(i,j) \in G} f((i,j)) \ .$$

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 $\mathcal{A}^{\gamma}(\Gamma_{\mathcal{T}}(\mathbf{w}, x, \mathbf{y})) \leq \mathcal{A}^{\gamma}\left(\Gamma(\mathbf{w}, x, \mathbf{y}) - 2\epsilon\right)$ 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 $\boldsymbol{\phi}_{T_i}$ 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

$$\hat{\mathbf{y}}_{T_i}(x) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} F_{\mathbf{w}_{T_i}}(x, \mathbf{y}) \stackrel{\text{def}}{=} \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{w}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x, \mathbf{y})\rangle, \tag{5}$$

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**

$$\mathop{\boldsymbol{max}}_{\boldsymbol{\mu}\in\mathcal{M}^m} \quad \frac{1}{|E_{\mathcal{T}}|}\sum_{e,k,\mathbf{u}_e} \mu(k,e,\mathbf{u}_e) - \frac{1}{2}\sum_{\substack{e,k,\mathbf{u}_e,\\ k',\mathbf{u}'_e}} \mu(k,e,\mathbf{u}_e)K^e_{\mathcal{T}}(x_k,\mathbf{u}_e;x'_k,\mathbf{u}'_e)\mu(k',e,\mathbf{u}'_e)\,,$$

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* $e=(v,v') \in E_{\mathcal{T}}$ *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*

$K^e_{\mathcal{T}}(x_k,\mathbf{u}_e;x_{k'},\mathbf{u}'_e) \stackrel{\text{def}}{=} \dfrac{N_{\mathcal{T}}(e)}{|E_{\mathcal{T}}|^2}\, K(x_k,x_{k'})\,\langle \boldsymbol{\psi}_e(y_{kv},y_{kv'}) - \boldsymbol{\psi}_e(u_v,u_{v'}), \boldsymbol{\psi}_e(y_{k'v},y_{k'v'}) - \boldsymbol{\psi}_e(u'_v,u'_{v'})\rangle$

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,* $N_{\mathcal{T}}(e)$ *denotes the number of times* e *appears among the trees of the ensemble.*

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*.

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

In this section we evaluate the performance of our random tree approximation (RTA) algorithm and compare it with the state-of-the-art methods through extensive experiments. We use ten multilabel datasets from different domains including chemical, biological, and text classification from [5].

For comparison, we select the following learning models. The Support Vector Machine (SVM) [10, 11] is used as a single target classifier, predicting each microlabel separately. Multitask Feature Learning (MTL) [12] is a multilabel classifier that assumes that the label specific functions are related such that they share a small subset of features. Max-Margin Conditional Random Fields (MMCRF) [9] is a multilabel classifier which uses the structure of the output graph that connects multiple labels. MMCRF uses the loopy belief propagation algorithm for approximate inference on the general graph. Maximum Average Marginal Aggregation (MAM) [5] 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.

Prediction performance. Following the setup in [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 $\{100, 50, 10, 1, 0.5, 0.1, 0.01\}$. Performances are computed by 5-fold cross-validation and are given as microlabel and error rates.

Table 1 shows the performance of different methods in terms of microlabel loss and 0/1 loss, where the best performing methods on each dataset is highlighted in '**boldface**' and the second best is in '*italics*' (see Table 2 in the supplementary material for standard deviation results). We observe that 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,

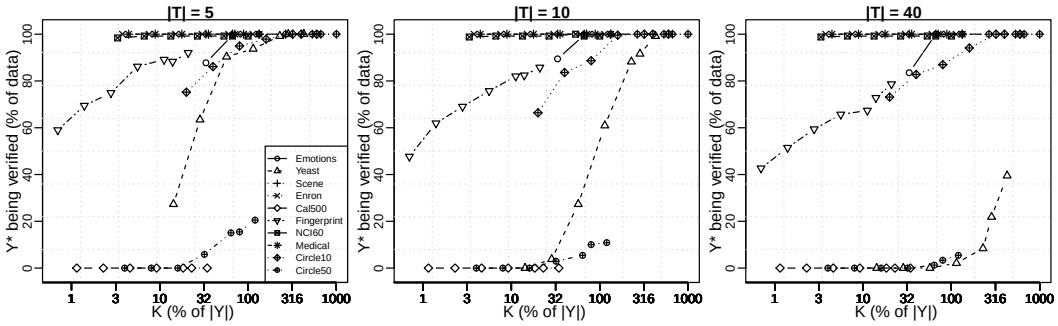


Figure 1: Percentage of examples with provably optimal $\mathbf{y}^*$ being in the K-best lists plotted as a function of K, scaled with respect to the number of microlabels in the dataset.

learning using approximate inference on a general graph seems less 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. In *Asian Conference on Machine Learning*, volume 29 of *JMLR Proceedings*, pages 404–418. JMLR.org, 2013.

[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.

SUPPLEMENTARY MATERIAL
of Multilabel Structured Output Learning with Random Spanning Trees of Max-Margin Markov Networks

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*

$$\mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} a_T^2 \;=\; 1 \quad ; \quad \mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} a_T \;\leq\; 1\,.$$

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

$$\mathop{\mathbf{E}}_{T\sim\mathcal{U}(G)} a_T \;\leq\; 1\,.$$

□

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*

$$\Gamma_{\mathcal{T}}(\mathbf{w},x,\mathbf{y}) \;\geq\; \Gamma(\mathbf{w},x,\mathbf{y}) - 2\epsilon \quad \forall(x,\mathbf{y})\in\mathcal{X}\times\mathcal{Y}\,,$$

whenever for all $(x,\mathbf{y})\in\mathcal{X}\times\mathcal{Y}$, *we have*

$$|F_{\mathcal{T}}(\mathbf{w},x,\mathbf{y}) - F(\mathbf{w},x,\mathbf{y})| \leq \epsilon\,.$$

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.