# Contents

# 1. Introduction

## 1.1 Scope of the Thesis

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

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

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

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

*Graph labeling* is another type of structured output prediction in which

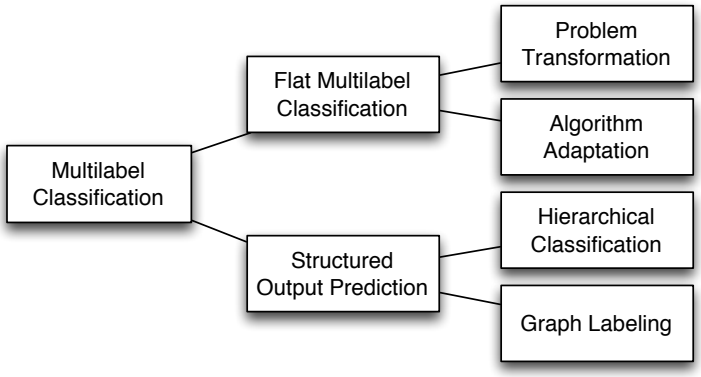


**Figure 1.1.** The taxonomy of the multilabel classification approaches.

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

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

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

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

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

## 1.2 Contributions and Organization

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

The remaining part of this thesis is structured as follows. Chapter 2 gives the background information to the learning problem in terms of

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

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

# 2. Regularized Learning for Classification

## 2.1 Regularized Risk Minimization

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

### 2.1.1 Empirical Risk Minimization

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

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

(Cortes and Vapnik, 1995)

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

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

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

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

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

*Exponential Loss* in ADABOOST (Schapire and Singer, 1999)

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

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

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

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

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

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

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

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

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

### 2.1.2 Regularized Learning

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

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

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

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

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

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

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

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

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

## 2.2 Single-Label Classification

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

### 2.2.1 Preliminaries

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

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

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

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

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

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

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

### 2.2.2 Logistic Regression (LR)

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

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

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

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

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

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

We can also compute

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 2.2.3 Support Vector Machines (SVM)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Definition 5.** *Kernel Function. A kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a function that for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ satisfies*

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

*where $\boldsymbol{\varphi} : \mathcal{X} \to \boldsymbol{\mathcal{F}}$ is a feature map that encodes from an input space $\mathcal{X}$ to a feature space $\boldsymbol{\mathcal{F}}$.*

**Definition 6.** *Positive Semi-definite Kernel. Given a non-empty set* $\mathcal{S} = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\} \subseteq \mathcal{X}$ *of* $m$ *objects, a kernel* $k : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ *is called a positive semi-definite kernel if it is symmetric and positive semi-definite. That is*

$$k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i) \quad \textit{and} \quad \sum_{i,j=1}^{m} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

*holds for any* $c_1, \cdots, c_m \in \mathbb{R}$.

Definition 6 implies that any kernel defined on a finite set of objects is a positive semi-definite kernel. In particular, the symmetry can be verified by

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

and the positive semi-definiteness can be verified by

$$\sum_{i,j=1}^{m} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{i,j=1}^{m} c_i c_j \langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{x}_j) \rangle = || \sum_{i=1}^{m} c_i \boldsymbol{\varphi}(\mathbf{x}_i) ||^2 \geq 0.$$

We refer to positive semi-definite kernels simply as *kernels* in the remaining part of the thesis.

**Definition 7.** *Reproduced Kernel Hilbert Space. For any kernel function* $k$ *defined on a space* $\mathcal{X}$*, there always exists a Reproduced Kernel Hilbert Space (RKHS)* $\boldsymbol{\mathcal{F}}$ *and a mapping function* $\boldsymbol{\varphi} : \mathcal{X} \to \boldsymbol{\mathcal{F}}$ *such that* $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{y}_i) \rangle$ *holds for any* $\mathbf{x} \in \mathcal{X}$.

Definition 7 shows that any kernel function can be represented as an inner product in some feature space $\boldsymbol{\mathcal{F}}$. Kernel enables us to work in a high dimensional feature space $\boldsymbol{\mathcal{F}}$ without ever computing the exact coordinate or evaluating the inner product explicitly in that space. Instead, kernel can be computed based on the inner product in the original input space $\mathcal{X}$. The simplest kernel is *linear kernel*

$$k_{\text{LINEAR}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle,$$

which assumes a linear feature map $\boldsymbol{\varphi}_{\text{LINEAR}} : \mathcal{X} \to \mathcal{X}$. Kernels that are heavily used in practice include *polynomial kernel*

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

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

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

where $\sigma$ is the Gaussian width parameter.

**Definition 8.** *Kernel Matrix. Given any choice of $m$ objects $\mathcal{S} = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\} \subseteq \mathcal{X}$ and a kernel function $k$ on $S$, the $m \times m$ matrix $K = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$ is called a Kernel matrix (Gram matrix) of kernel $k$ with respect to $\mathcal{S}$.*

The kernel matrix is usually normalized by

$$K(i,j) = \frac{K(i,j)}{\sqrt{K(i,i)K(j,j)}}$$

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

**Definition 9.** *Positive Semi-definite Matrix. An $m \times m$ symmetric matrix $K$ is a positive semi-definite matrix if it satisfies*

$$\sum_{i,j=1}^{m} c_i c_j K(i,j) \geq 0$$

*for any $c_1, \cdots, c_m \in \mathbb{R}$.*

It has been shown that Kernel matrices are positive semi-definite (Shawe-Taylor and Cristianini, 2004). Positive semi-definiteness distinguishes between general similarity measures and kernels. The property is essential in kernel based methods from two perspectives. First, it ensures kernel based methods will converge to a relevant solution in convex optimization (Boyd and Vandenberghe, 2004). Secondly, it is a key assumption in *Reproducing Kernel Hilbert Space* theory described as follows.

The algorithms for solving the optimization problem of SVM have been intensively studied, for example, the "chunking" method (Vapnik, 1982; Pérez-Cruz et al., 2004), the decomposition method (Osuna et al., 1997; Joachims, 1998), Sequential Minimum Optimization (SMO) (Platt, 1998, 1999), and the "digesting" method (Decoste and Schölkopf, 2002). There are some recent studies that aim to scale SVM learning on the large scale datasets, for example, representing the training data with a small set of landmark points (Pavlov et al., 2000; Boley and Cao, 2004; Yu et al., 2005; Zhang et al., 2008), the greedy method for basis selections (Keerthi et al., 2006), the online SVM solver (Bordes et al., 2005), approximating the objective function of SVM (Zhang et al., 2012; Le et al., 2013), and approximating the kernel matrix with a low-rank matrix (Smola and Schökopf, 2000; Fine and Scheinberg, 2002; Drineas and Mahoney, 2005; Si et al., 2014).

## 2.3 Ensemble Methods

Ensemble methods are general classification techniques in machine learning. The methods train several base classifiers and combine them in order to achieve the more accurate predictions. There are several variants of ensemble methods, to name but a few, bagging (Breiman, 1996a), boosting (Freund and Schapire, 1997; Schapire and Singer, 1999), stacking (Smyth and Wolpert, 1999), and Bayesian averaging (Freund et al., 2004). Ensemble methods have improved the classification performance when compared to their base learner counterpart, some of them are also supported with the theoretical analysis which guarantees the performance (Schapire et al., 1997; Koltchinskii and Panchenko, 2000; Cortes et al., 2014a,b). This section will be devoted to bagging and boosting as both methods are extensively studied and are quite relevant to this thesis.

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

### 2.3.1 Preliminaries

In addition to the notations introduced in Section 2.2.1, we assume there is a hypothesis class $\mathcal{H}'$ where we generate weak/base hypotheses $f^t(x) \in \mathcal{H}'$. We use $t$ to index the $t$'th weak hypothesis. Let $H(x)$ denote the ensemble framework which combines multiple weak hypotheses and generate a stronger one. In many cases, no other information about $f^t(x)$ is available to $H(x)$ except that each weak hypothesis will take in a parameter x and generate an output $y \in \mathcal{Y}$.

### 2.3.2 Boosting

*Boosting* corresponds to a learning framework or a family of algorithms that takes in a weak classifier and tunes it into a strong one. We begin our discussion from the *concept class*. A *concept* is a boolean function over a domain $\boldsymbol{\mathcal{X}}$, and a *concept class* is a class of concepts. A concept class is *strongly learnable* if there exists a polynomial learning algorithm which achieves a high accuracy with a high probability for all concepts in

the class. On the other hand, a concept class is *weakly learnable* if the learning algorithm achieves an arbitrary high accuracy where the only requirement is that the learning algorithm finds a function which performs better than the coin flipping. The concept of learnability was proposed by Kearns and Valiant (1989) together with the question whether the strong learnability and the weak learnability are equivalent which is known as the *hypothesis boosting problem*. Finding a weak learner which performs better than random guessing is easy in practice, but finding a strong learner is usually difficult. Schapire (1990) has proved that the two classes of learnability are equivalent which lays the foundation of the boosting algorithm that tunes a weak learning algorithm into a strong one.

Adaptive Boosting (ADABOOST) 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 to minimize the exponential loss (2.3) of each weak learner. The algorithm is shown in Algorithm 1. The central idea of ADABOOST is to maintain a distribution $D$ over all training examples, and update the distribution in each iteration such that the difficult-to-classify examples will get more probability mass for the next iteration (line 7). Particularly, in each iteration, the algorithm computes a weak learner $f^t(\mathbf{x})$ based on all training examples and the current distribution $D^t$ (line 3), calculates the weighted training error $\epsilon^t$ (line 4), and computes the adaptive parameter $\alpha^t$ (line 5). The ensemble prediction is the weighted combination of all weak learners (line 9).

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

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

To minimize $\mathcal{L}_{exp}(y, \alpha^t f^t(\mathbf{x}); D^t)$, we take the partial derivative with re-

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

spect to $\alpha^t$ and set it to zero

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

Solve it for $\alpha^t$, we get

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

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

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

### 2.3.3 Bootstrap Aggregating

*Bootstrap Aggregating* (BAGGING) (Breiman, 1996a) is an ensemble method that exploits the independency between weak learners. The algorithm is based on the fact that errors can be dramatically reduced by combining

independent base learners. Let $f_t$ denote the $t$'th weak learner. The ensemble prediction $H(\mathbf{x})$ is the averaged prediction over all weak learners

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

We assume a base learner has a probability $\epsilon$ of making an independent mistake

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

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

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

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

*Bootstrap Sampling* (Efron and Tibshirani, 1994) is applied in BAGGING to generate subsets of training examples. Given a training set of $m$ training examples, a subset of the same size is generated by sampling with replacement $m$ times from the original training set. The sampling procedure is repeated $T$ times to generate $T$ subsets for constructing base learners. Sampled subsets will be similar since they are sampled from the same training set. However, they will not be too similar in that each subset will only cover around $63\%$ of the original training data under the condition that $m$ is large. To see this, consider the probability that the $i$'th training examples is not sampled once is $(1-\frac{1}{m})$, and the probability that it is not sampled at all is $(1-\frac{1}{m})^m$. When $m$ is large, this probability will approach $37\%$. That is, around $37\%$ of the training examples will not appear in any sampled training set. The property of Bootstrap Sampling also allows us to efficiently estimate the generalization error of the base learner known as *out-of-bag estimation* (Breiman, 1996b; Tibshirani, 1996; Wolpert and Macready, 1999).

# 3. Multilabel Classification

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

It is prohibitive to present all the algorithms developed for flat multilabel classification, it is easier if we can categorize the algorithms into groups. In this chapter, we adopt the categorization scheme (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010) which gives us two major groups of algorithms, namely problem transformation and algorithm adaptation. Problem transformation aims to transfer the flat multilabel classification problem into other well-studied problems, for example, single-label classification, label ranking, label power set. The algorithm adaptation directly modifies the established learning techniques to solve the multilabel classification problem. Nevertheless, the presented algorithms aim to tackle the central problems of flat multilabel classification, namely to explore the exponential sized multilabel space and to model the corre-

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

## 3.1 Preliminaries

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

## 3.2 Problem Transformation

Problem transformation aims to transform the flat multilabel classification problem into other well studied problems. The most typical way of the transformation is *binary relevance* (BR) (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010), which transforms a multilabel classification problem into a set of single-label classification problems and to independently learns a single-label classifier for each subproblem. There exists many other types of transformations, for example, into the label power set problem (Tsoumakas and Vlahavas, 2007), and into the label ranking problem (Elisseeff and Weston, 2002; Brinker and Hüllermeier, 2007; Fürnkranz et al., 2008; Chiang et al., 2012). However, learning by

label ranking will not be explained in detail as it slightly diverges from the main scope of this thesis. We will focus on BR in this section.

### 3.2.1 Multilabel $K$-Nearest Neighbors (ML-KNN)

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

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

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

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

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

### 3.2.2 Classifier Chains (CC)

Classification Chains (CC) (Read et al., 2009, 2011) is another problem transformation approach for flat multilabel classification. CC involves $k$ binary transformations and forms a chain of $k$ binary classifiers $h =$

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

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

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

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

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

### 3.2.3 Instant Based Logistic Regression (IBLR)

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

In particular, for each unseen example $\mathbf{x}$, IBLR first identifies the set of $K$-nearest neighbors $N_k(\mathbf{x})$ from the training data. The algorithm builds a Logistic Regression model (Section 2.2.2) for each microlabel based on the label information collected from the examples in $N_k(\mathbf{x})$. Mathematically, IBLR defines a posterior probability of the $j$'th microlabel of $\mathbf{x}$ being

labeled as $u_j$ by

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

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

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

where $i$ iterates over all microlabels. $w_i^{(j)}(\mathbf{x})$ 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 microlabels according to the similarity between $\mathbf{x}$ and $\mathbf{x}'$ encoded in $K(\mathbf{x}', \mathbf{x})$. $\alpha_i^{(j)}$ is the regression coefficient.

## 3.3 Algorithm Adaptation

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

### 3.3.1 Ensemble Methods for Flat Multilabel Classification

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

ADABOOST.MH is a multilabel variance of the ADABOOST algorithm (Schapire and Singer, 1999; Esuli et al., 2008). The core idea of ADABOOST.MH is to apply the hamming loss instead of the 0/1 loss. In particular, the algorithm reduces a multilabel classification problem into a single-label classification problem by replacing each training example $(\mathbf{x}_i, \mathbf{y}_i)$ with $k$

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

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

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

### 3.3.2 Correlated Logistic Regression (CORRLOG)

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

In fact, CORRLOG is derived from Independent Logistic Regression (ILRS).

Given a pair of an arbitrary training example and a label $(\mathbf{x}, \mathbf{y})$, we can construct a set of ILRS classifiers, one for each microlabel. The posterior probability can be computed by

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

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

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

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

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

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

### 3.3.3 Multitask Feature Learning (MTL)

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

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

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

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

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

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

**Definition 10.** MTL *Optimization Problem in Primal*

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

where $C$ is a positive parameter that controls the balance of the regularization term and the risk minimization term. The optimization problem is an instantiation of the regularized risk minimization (2.7) with the hamming loss and the $L_{2,1}$-norm regularization. As Definition 10 is non-convex and the second term is non-smooth, the optimization is transformed into an equivalent form which is solved by an alternative optimization approach (Argyriou et al., 2007).

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

# 4. Structured Output Prediction

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

## 4.1 Preliminaries

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

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

## 4.2 Related Methods

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

### 4.2.1 Structured Perceptron

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

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

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

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

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

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

The central problem with Structured Perceptron is the loss function. In fact, Structured Perceptron tacitly applies $0/1$ loss (2.2) on multilabels,

with which it is impossible to distinguish a nearly correct multilabel and a completely incorrect one. Both will lead to the same update to the feature weight parameter (4.2) during learning.

### 4.2.2 Conditional Random Field (CRF)

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

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

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

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

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

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

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

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

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

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

Taskar et al. (2004) proposed Max-Margin Markov Network (M$^3$N) that

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

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

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

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

**Definition 11.** 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 $\xi_i$ is the slack allotted to each example to make sure the solution can always be found, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss function between a correct multilabel $\mathbf{y}_i$ and an incorrect multilabel $\mathbf{y}$, $C$ is the slack parameter that controls the amount of regularization in the model.

For each example $\mathbf{x}_i$, the optimization calls for maximizing the margin between the correct label $\mathbf{y}_i$ and any incorrect labels $\mathbf{y}$. The margin is scaled by the loss function $\ell(\mathbf{y}_i, \mathbf{y})$ such that the completely incorrect multilabel will incur bigger loss than the nearly correct multilabel. The loss scaled margin optimization will push the high-loss pseudo-examples further away from the correct example than the low-loss pseudo-examples. Definition 11 is an instantiation of the regularized risk minimization (2.7) with the hamming loss and the $L_2$-norm regularization (2.8).

The primal optimization problem of M$^3$N in Definition 11 is difficult to solve as there are exponential number of constraints, one for each pseudo-example $(\mathbf{x}_i, \mathbf{y})$. The corresponding dual form is also difficult due to the

exponential number of dual variables (Taskar et al., 2004). By exploring the Markov network structure, the original optimization problem (Definition 11) can be formulated into a factorized dual quadratic programming, as long as the loss function $\ell$ and the joint feature map $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ are decomposable over the Markov network.

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

To use $\text{M}^3\text{N}$ 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, $\text{M}^3\text{N}$ improves CRF by avoiding the evaluation of the partition function and allowing complex loss functions to be defined.

### 4.2.4 Max-Margin Conditional Random Fields (MMCRF)

Max-Margin conditional random field (MMCRF) (Rousu et al., 2007) is a structured output learning method, that extends $\text{M}^3\text{N}$ by defining the joint feature map as the tensor product between an input feature map and an output feature map, and by developing an efficient optimization strategy. MMCRF is applied in su10 in which the task is to reliably predict the multiple interdependent molecular activities.

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

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

where the joint feature map $\boldsymbol{\phi}_e(\mathbf{x}, \mathbf{y}_e) = \boldsymbol{\varphi}(\mathbf{x}) \otimes \boldsymbol{\Upsilon}_e(\mathbf{y}_e)$ is defined as the tensor product between an input feature map and an output feature map which is the label of an edge $e \in E$ in an output Markov network $G$ with

respect to a multilabel $\mathbf{y}$. To obtain $\mathbf{w}$, one needs to solve the primal optimization problem that is not dissimilar to Definition 11. After the feature weight parameter $\mathbf{w}$ is obtained, the prediction of an input example can be computed by solving the following argmax problem

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

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

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

Support Vector Machines for interdependent and structured output space (SSVM) is developed by 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 12.** 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 $\xi_i$ is the slack allotted to each example, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss function between a correct multilabel and an incorrect multilabel, and $C$ is the slack parameter that controls the amount of regularization in the model. The interpretation of Definition 12 is also similar to that of Definition 11. Besides, Tsochantaridis et al. (2004) suggests that M$^3$N will work hard on the pseudo-examples $(\mathbf{x}_i, \mathbf{y})$ which incur a big loss though they may not even close to be confusable to the true multilabel $\mathbf{y}_i$.

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

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

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

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

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

## 4.3 SPIN for Network Response Prediction

su14a presents a novel definition of the network response prediction problem and develops a structured output learning model for the problem. Unlike the previous methods which model the influence in terms of the network connectivity, the proposed model (SPIN) is *context-sensitive*. That is, the influence dynamics also depend on the properties of the action performed on the underlying network. The inference problem of SPIN is $\mathcal{NP}$-hard in general. We develop a semidefinite programming algorithm (SDP) with an approximation guarantee as well as a fast GREEDY heuristics.

### 4.3.1 Background

With the extensive availability of the large scale networks, there is an increasing amount of interest in studying the phenomena of the network influence, in particular, the structure, the function, and the influence dynamics. The outcome of the network influence research has been widely applied to many areas, for example, the spreading of pathogens or infectious diseases (Hethcote, 2000; Anderson and May, 2002), the diffusion of medical and technology innovations (Strang and Soule, 1998; Rogers,

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

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

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

**Definition 13.** *Network Response Problem. Given a complex network and an action performed on the network, predict an optimal subnetwork that best responds to the action. In particular, which nodes perform the action and which directed edges relay the action from one node to its neighbors.*

### 4.3.2 Methods

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

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

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

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

**Definition 14.** *Primal* SPIN *Optimization Problem.*

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

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

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

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

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

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

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

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

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

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

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

**Lemma 1.** *The inference problem (4.10) can be expressed equivalently with a set of activated vertices $V_p^H$ and the marginal gain function $F_m(v_i)$ defined on each vertex $v_i \in V_p^H$ as*

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

The proof is given in the supplementary material of su14a. As a result, the GREEDY algorithm starts with an empty vertex set and adds one vertex in each iteration such that the increment of the score is maximized over all possible choices of inactivated vertices. The procedure ends when the objective cannot be improved. It is worth pointing out that we are not able to give any approximation guarantee for the solutions produced by the GREEDY algorithm. The property of sub-modularity, which is often used to analyze the greedy algorithm, only holds for the special case of our inference problem.

# 5. Structured Output Prediction with Unknown Output Graphs

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

## 5.1 Structured Output Prediction for Molecular Classification

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

ber of molecular activities at the same time. The second challenge is that single-label classification ignores the correlation between multiple output variables. On the other hand, multiple molecular activities are often correlated which can be utilized to improve the classification performance. In su10, we explore the potential of structured output learning in the molecular activity classification problem. To apply structured output learning, we extract output graphs from several auxiliary datasets which encode the correlation between multiple activities of molecules.

### 5.1.1 Background

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

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

### 5.1.2 Methods

To efficiently and accurately predict molecular activities in multiple cell lines at the same time, we applied a structured output learning approach in su10, which is to our knowledge the first multilabel classification approach for the molecular classification problem. The algorithm is an in-

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

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

where $\mathbf{w}$ is the feature weight parameter to ensure that a molecule with the correct activity will be scored higher than with any incorrect activities. $\mathbf{w}$ is obtained by maximizing the minimum loss-scaled margin between the correct examples $(\mathbf{x}_i, \mathbf{y}_i)$ and the incorrect pseudo-examples $(\mathbf{x}_i, \mathbf{y})$ over all training examples, which amounts to solving the optimization problem that is not dissimilar to Definition 11.

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

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

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

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

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

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

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

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

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

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

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

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

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

### 5.2 Graph Labeling Ensemble (MVE)

The structured output learning approaches, relying on the representation of multiple output variables through an output graph, allow us to exploit the correlation between labels. To apply structured output learning, it is assumed that the structure of the output graph is known *apriori*. For the molecular classification problem in su10 where the output graph is not observed, we can extract the structure of the output graph by examining a collection of auxiliary datasets which explicitly encode the correlation of labels. For most real world multilabel classification problems, however, we cannot take for granted the availability of the output graph or the auxiliary data that reveals the label correlations. Therefore, in su11, we explore the potential of using majority vote to combine the predictions

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

### 5.2.1 Methods

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

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

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

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

## 5.3 Random Graph Ensemble (AMM, MAM)

Section 5.2 has shown that the prediction performance in the molecular classification problem can be improved by applying majority vote to combine the predictions from multiple structured output learners built on a

collection of random output graphs. This section is based on su14b in which we present two aggregation techniques to combine multiple structured output learners. The proposed model, namely AMM and MAM, also perform inference before or after combining the base learners. The performance of the proposed models is evaluated on a set of heterogeneous multilabel datasets from a variety of domains. In addition, we study the performance of MAM in terms of the reconstruction error of the compatibility score.

### 5.3.1 Background

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

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

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

In addition, we assume that we have access to the set of edge potentials of the $t$'th base classifier

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

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

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

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

### 5.3.2 Methods

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

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

The predicted multilabel by AMM is composed by the predicted microlabels

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

AMM performs inference to find the set of max-marginals before combining base classifiers. On the other hand, the Maximum-of-Average-Marginals (MAM) aggregation first collects the local edge potentials $\boldsymbol{\psi}_E^{(t)}$ from each base learner, averages them and performs a final inference with the averaged edge potentials on a global consensus graph $\hat{G} = (\hat{E}, V)$ where $\hat{E} = \bigcup_{t=1}^{T} E^{(t)}$ is the union of distinct edges of the set of random output graphs. Mathematically, MAM is defined as

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

In addition, [Lemma 1]su14b simplifies the computation of MAM in terms of dual variables and kernels.

Besides the proposed algorithms, we also present a theoretical analysis to explain the improvement of MAM. The analysis extends the theory of single-label ensemble (Brown and Kuncheva, 2010). In particular, [Theorem 1]su14b states that the reconstructive error of MAM is guaranteed to be less than or equal to the average reconstruction error of base classifiers. The improvement can be decomposed into two terms, namely *diversity* and *coherence*. The former measures the variability of the individual classifiers learned from different perspectives which shares the same argument as the analysis of single-label ensemble (Brown and Kuncheva, 2010). The latter measures the correlation of microlabel predictions in which the correlation has a positive effect on the performance.

## 5.4 Random Spanning Tree Approximation (RTA)

su14c presents Random Spanning Tree Approximation (RTA) for structured output learning in which the output graph is not observed but believed to play an important role during learning. RTA is a major step forward of MAM by bringing in a joint learning and inference framework such that the base learners built from a collection of random spanning trees are optimized simultaneously towards the same global objective. su14c also presents the theoretical studies which not only explain the intuition behind the learning model but also guarantee the performance by the generalization error analysis. Meanwhile, RTA lays the foundation of tackling the intractability of the graph inference on unknown graph structures in which the fast optimization and accurate predictions can be achieved with attainable computational efforts.

### 5.4.1 Background

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

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

su14c is motivated by the well-established Maximum-Margin Principle

as described in Section 2.2.3. The work investigates whether the problem of inference over a complete graph in structured output learning can be avoided by exploring the properties of the Maximum-Margin Principle. Starting from a sampling results, [Lemma 3]su14c shows that with a high probability a big fraction of the margin achieved by a complete graph can be obtained by a random sample of spanning trees of a small size. Besides, [Theorem 5]su14c shows that the good generalization error can also be guaranteed when learning with, instead of a complete graph, a random sample of spanning trees.

Thus, in addition to [Theory 1]su14b, we further provide the theoretical justification of combining a set of base learners trained on a collection of random output graphs. Besides, [Theorem 5]su14c suggests we should, instead of optimizing the margin separately on each spanning tree similar to that in MAM, optimize the joint margin from all spanning trees. The strategy leads to the learning model presented in the following section.

### 5.4.2 Methods

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

**Definition 15. Primal $L_2$-norm Random Tree Approximation (RTA).**

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

where $\boldsymbol{\phi}_{T_t}(\mathbf{x}, \mathbf{y})$ is the feature map that is local on each tree $T_t$, $\xi_i$ is the margin slack allocated for each $\mathbf{x}_i$, and $C$ is the slack parameter that controls the amount of regularization. Definition 15 is an instantiation of the regularized learning (Section 2.7) in terms of the $L_2$-norm regularization (2.8) and the $0/1$ loss (2.2).

The key for the optimization is to solve the argmax problem efficiently. This is an $\mathcal{NP}$-hard problem in practice, as the size of the multilabel space is exponential in the number of microlabels. In su14c, we have developed a $K$-best inference algorithm working in $\Theta(Knk)$ time per data

point, where $k$ is the number of microlabels and $K$ is the number of best multilabels we compute from each random spanning tree.

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

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

can be solved in $\Theta(k)$ time by *dynamic programming* also known as *max-product* or *min-sum*. However, there is no guarantee that the maximizer of (5.1) is also the global maximizer of Definition 15 over the set of random spanning trees. Therefore, we compute the top $K$-best multilabels for each random spanning tree. In total the computation costs $\Theta(Knk)$ time for all spanning trees. [Lemma 7]su14c provides a method to retrieve the best multilabel from the $K$-best multilabel list in linear time. We still need to make sure that the global maximizer is within the $K$-best multilabel list. [Lemma 8]su14c guarantees that with a high probability the global maximizing multilabel is in the list and $K$ does not need to be large.

In addition, we derived the marginalized dual representation of the primal optimization problem in Definition 15, which not only works with a polynomial sized parameter space but also enables kernels to tackle the complex input space.

# 6. Implementations

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

1. RTA, developed in su14c, is a structured output learning algorithm for multilabel classification with an unknown output graph. RTA performs joint learning and inference on a random sample of spanning trees.

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

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

2. SPIN, developed in su14a, is a structured output learning algorithm for multilabel classification with an observed output graph. SPIN can predict an optimal direct acyclic graph (DAG) that best responds to an input. The algorithm has been applied to the network response prediction problem within the context of social network analysis.

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

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

3. MVE developed in su11 as well as AMM and MAM developed in su14b are the structured output learning algorithms that are not constrained by the availability of the output graph. The algorithms combine a set of structured output learners built on a collection of random output graphs.

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

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

# 7. Conclusion

## 7.1 Discussion

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

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

Meanwhile, the current structured output learning approaches rely on an output graph connecting multiple output variables to exploit the correlation between labels. Thus, the applicability of structured output learn-

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

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

## 7.2 Future Work

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

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

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

# Bibliography


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge Univeristy Press, Cambridge, UK.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[conference]Hongyu Su, Aristides Gionis, Juho RousuStructured Prediction of Network ResponseProceedings of the $31^{th}$ International Conference

on Machine Learning (ICML 2014)Beijing, China, 2014. JMLR W&CP volume 32:442-450June2014Copyright 2014 by the authorssu14a

su14a presents a novel definition of the network response prediction problem, and proposes a new structured output learning algorithm to solve the problem. The proposed algorithm, SPIN, captures the contextual information and improves the state-of-the-art models in terms of the prediction performance.

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

publications/su14a.pdf publications/su14a$_supplementary.pdf$

[conference]Hongyu Su, Markus Heinonen, Juho RousuMultilabel Classification of Drug-like Molecules via Max-margin Conditional Random FieldsProceedings of the $5^{th}$ International Conference on Pattern Recognition in Bioinformatics (PRIB 2010)Nijmegen, The Netherlands, 2010. Springer LNBI volume 6282:265-273September2010Copyright 2014 by the authorssu10

su10 presents a structured output learning algorithm for the multilabel molecular classification problem. The new algorithm incorporates the correlation between multiple output variables into the learning process. It outperforms the previous single-label classification approaches in terms of the prediction performance.

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

publications/su10.pdf

[conference]Hongyu Su, Juho RousuMulti-task Drug Bioactivity Classification with Graph Labeling EnsemblesProceedings of the $6^{th}$ International Conference on Pattern Recognition in Bioinformatics (PRIB 2011)Delft, The Netherlands, 2011. Springer LNBI volume 7035:157-167November2011Copyright 2014 by the authorssu11

su11 extends su10 by applying majority vote to combine the predictions from a set of structured output learners built on a collection of random output graphs.

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

publications/su11.pdf

Hongyu Su, Juho RousuMultilabel Classification through Random Graph EnsemblesMachine LearningDOI:10.1007/s10994-014-5465-9, 26 PagesSeptember2014Copyright 2014 by the authorssu14b

su14b extends su11 by introducing two aggregation techniques which perform inference before or after combining multiple structure output learners. su14b also presents a theoretical study that explains the performance of the proposed algorithms. The developed algorithms are evaluated on a collection of heterogeneous multilabel prediction problems.

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

publications/su14b.pdf

[conference]Mario Marchand, Hongyu Su, Emilie Morvant, Juho Rousu, John Shawe-TaylorMultilabel Structured Output Learning with Random Spanning Trees of Max-Margin Markov NetworksAdvances in Neural Information Processing Systems 27 (NIPS 2014)873-881December2014Copyright 2014 by the authorssu14c

su14c is a major step forward of su14b by introducing a joint learning and inference framework, and developing rigorous theories to backup the algorithm.

The idea was jointly initialized by the authors. The author worked jointly on developing the theories and the proofs. The author made a major contribution to the design of the learning framework and the optimization algorithm. The author implemented the learning system and the optimization algorithm. The author designed and performed the experiments, and analyzed the results. The author worked jointly on writing the research article. publications/su14c.pdf publications/su14c$_supplementary.pdf$