Preface

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

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

Espoo, September 2, 2014,

Hongyu Su

Contents

List of Publications

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

Author’s Contribution

List of Symbols

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

$(x_i, \mathbf{y}_i)$	Example and multilabel pair.
α	Dual variable.
$\bar{\mu}$	Marginal dual variable averaged over ensembles.
$\bar{\psi}_e(x, \mathbf{y}_e)$	Mean compatibility score of edge e with label $\mathbf{y}_e$ over ensemble.
$\bar{\psi}_j(x, \mathbf{y}_j)$	Mean compatibility score of node j being label y_j over ensemble.
$\Delta_I^R(x, \mathbf{y})$	Reconstruction error from individual base learners.
$\Delta_{MAM}^R(x, \mathbf{y})$	Reconstruction error from MAM.
$\ell(\cdot, \cdot)$	Loss function.
$\mathcal{G}$	A set of random output graphs.
$[\cdot]$	Indicator function.
μ	Marginal dual variable.
$\phi(x)$	Input feature map.
$\psi(x, \mathbf{y})$	Compatibility score of $(x, \mathbf{y})$.
$\psi^*(x, \mathbf{y})$	True compatibility score.

$\psi^{MAM}(x, \mathbf{y})$	Compatibility score from MAM ensemble.
$\Psi_E(x)$	Collection of all local edge potentials/compatibility scores.
$\psi_e(x, \mathbf{y}_e)$	Local edge potential/compatibility score of edge e with label $\mathbf{y}_e$.
$\Psi_E^{(t)}(x)$	Collection of all local edge potentials/compatibility scores from the t'th base learner.
$\psi_e^{(t)}(x, \mathbf{u}_e)$	Local compatibility score of edge e with labeled $\mathbf{u}_e$ from the t'th base learner.
$\psi_j(x, \mathbf{y}_j)$	Local node potential/compatibility score of node j with label $\mathbf{y}_j$.
$\Psi_j(x, y_j)$	Random variable of compatibility score on node j, $\Psi_j(x, y_j) = \{\psi_j^{(1)}(x, y_j), \cdots, \psi_j^{(T)}(x, y_j)\}$.
$\psi_j^*(x, y_j)$	True compatibility score of node j.
$\psi_j^{(t)}(x, \mathbf{y}_j)$	Local node potential/compatibility score of node j with label $\mathbf{y}_j$, from the t'th learner.
$\tilde{\Psi}(x)$	Collection of all node max-marginals.
$\tilde{\Psi}^{(t)}(x)$	Collection of all node max-marginals from the t'th base learner.
$\tilde{\psi}_j(x, u_j)$	Maximal marginal of node j with label u_j.
$\tilde{E}$	Edge set of the consensus graph.
$\tilde{G}$	Consensus graph.
$\Upsilon(\mathbf{y})$	Output feature map.
$\varphi(x, \mathbf{y})$	Joint feature map of example and label pair $(x, \mathbf{y})$.
$\mathbf{u}_e$	Possible label of edge e.
$\mathbf{w}$	Weight vector.
$\mathbf{y}$	Multilabel vector.
$\mathbf{y}_e$	Label of edge e.
$\mathbf{y}_{ie}$	Label of edge e for the i'th example.
$\mathcal{X}$	Domain of training data.

ξ	Slack parameter.
$\mathcal{Y}$	Multilabel space.
$\mathcal{Y}_j$	Microlabel space.
$A(\cdot)$	Aggregation function.
E	Edge set of G, revealing potential dependencies.
$E^{(t)}$	Edge set of the t'th random output graph.
F	Mapping function from $\mathcal{X}$ to $\mathcal{Y}$.
$F(\cdot)$	Ensemble model in general form.
$F^{(t)}(\cdot)$	The t'th base learner, graph labeling model.
G	Output graph structure.
$G^{(t)}$	The t'th random output graph.
i	Index of training data.
j	Index of a microlabel in a multilabel.
k	Length of a multilabel.
m	Size of training data.
T	Size of ensemble.
t	Index of individual base learner $t \in \{1, \cdots, T\}$
u_j	Possible label of node j.
V	Vertex set of G, corresponding to multilabel.
y_j	Microlabel, the j'th position in multilabel.

1. Introduction

1.1 Motivation and Background

1.2 Contributions and outline of the thesis

The thesis is structured as follows.

complex outputs: multiple interdependent output variables and structured output space

2. Background

2.1 Single Label Classification

2.1.1 Preliminary and Notation

We consider supervised learning problem which assumes an arbitrary input space $\mathcal{X}$, an output space $\mathcal{Y}$, and the training samples coming in pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. As we focus in this chapter standard supervised learning problem also known as binary classification, we explicitly assume the output space $\mathcal{Y} = \{-1, +1\}$. We point out that other kind of supervised learning problem can be formulated by altering the definition of the output space. For example, by setting $\mathcal{Y} = \{1, \cdots, K\}$ we have multiclass classification problem, and by setting $\mathcal{Y} = \mathbb{R}$ we have regression problem. Additionally, we assume a feature map $\phi : \mathcal{X} \to \mathcal{F}$, which embeds the input in some high dimensional feature space $\mathcal{F} = \mathbb{R}^D$. In particular, $\phi(\mathbf{x})$ is a real value vector of D dimension. The goal is to learn from a *hypothesis class* $\mathcal{F}$ a mapping function $f \in \mathcal{F} : \mathcal{X} \to \mathcal{Y}$ that maps an input $\mathbf{x} \in \mathcal{X}$ to an output $y \in \mathcal{Y}$.

The hypothesis class we consider is a set of *linear classifiers* that are parameterized by weight vector $\mathbf{w}$ and bias term b that takes the form

$$f(x; \mathbf{w}, b) = \langle \mathbf{w}, \mathbf{x} \rangle + b, \tag{2.1}$$

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

2.1.2 Perceptron

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

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

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

Assume data point (x_i, y_i) is misclassified, the signed distance from x_i to decision boundary can be computed

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

Denote by $\mathcal{X}^m$ the set of misclassified examples, the goal of perceptron is to minimize the distance over all misclassified examples defined as

Definition 1. *Perceptron Objective Function*

$$\min_{\mathbf{w},b} D(\mathbf{w},b) = \min_{\mathbf{w},b} \left(- \sum_{x_i \in \mathcal{X}^m} y_i [\langle \mathbf{w}, x_i \rangle + b] \right).$$

To achieve (Definition 1), we assume $\mathcal{X}^m$ is fixed and compute the partial gradient

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

In practice, the perceptron algorithm uses stochastic gradient descent that processes iteratively from training data one example at a time. At each step, the algorithm makes sure the current $\mathbf{w}$ and v will correctly separate the current training example. Once a misclassified example is visited, it adjust the parameter according to the update rule

$$\mathbf{w} \leftarrow \mathbf{w} + \rho y_i x_i,$$
$$b \leftarrow b + \rho y_i,$$

where ρ is the perceptron learning rate.

Theory 1. *(Block, 1962; Novikoff, 1962) Given a sequence of training examples in pairs $D = \{(x_1, y_1), \cdots, (x_m, y_m)\}$. Assume $||x_i|| \leq R$ for all $i \in \{1, \cdots, m\}$. Suppose for some $\gamma > 0$ there exists a unit length vector $||\hat{\mathbf{w}}|| = 1$ such that $y_i(\langle \hat{\mathbf{w}}, x_i \rangle) \geq \gamma$ for i (training data D is linearly separable). Then the number of mistakes the perceptron algorithm makes on the sequence of training data D is at most R^2/γ^2.*

Proof. Suppose the k'th mistake is made on the i'th training example, and current weight vector is $\mathbf{w}^k$. In addition, we set $\mathbf{w}^0 = \mathbf{0}$. As the Perceptron makes a mistake on (x_i, y_i), we immediately have

$$y_i \langle \mathbf{w}_k, x_i \rangle \leq 0.$$

According to update rule, we have

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \rho y_i x_i.$$

Then, the following holds

$$\langle \mathbf{w}^{k+1}, \hat{\mathbf{w}} \rangle = \langle \mathbf{w}^k, \hat{\mathbf{w}} \rangle + \rho y_i \langle x_i, \hat{\mathbf{w}} \rangle \geq \langle \mathbf{w}^k, \hat{\mathbf{w}} \rangle + \rho \gamma.$$

Straight forward induction give us

$$\langle \mathbf{w}^{k+1}, \hat{\mathbf{w}} \rangle \geq k\rho\gamma. \tag{2.2}$$

We also have

$$\left|\left|\mathbf{w}^{k+1}\right|\right|^2 = \left|\left|\mathbf{w}^k + \rho y_i x_i\right|\right|^2 = \left|\left|\mathbf{w}^k\right|\right|^2 + \rho^2 \left|\left|x_i\right|\right|^2 + 2\rho y_i \langle x_i, \mathbf{w}^k \rangle \leq \left|\left|\mathbf{w}^k\right|\right|^2 + \rho^2 R^2$$

Straight forward induction give us

$$\left|\left|\mathbf{w}^{k+1}\right|\right|^2 \leq k\rho^2 R^2. \tag{2.3}$$

Together with (2.2) and (2.3) we have

$$k \leq R^2/\gamma^2.$$

□

Theory 1 shows that if the data are linearly separable, the perceptron algorithm will make a finite number of mistakes. In particular, the Perceptron algorithm will iterate through the training set and will converge to a vector that well separates all training examples. Moreover, the number of mistakes will be bounded by the minimum gap between the positive and negative examples.

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

Voted Perceptron developed in (Freund and Schapire, 1999) is a straight forward modification of the standard Perceptron As described in (Algorithm 1), the Voted perceptron algorithm keeps track and updates during training all weight vectors. (line 9) and their weights (line 10). The weight can be seen as the 'survival time' of the weight vector during Perceptron training. The final decision rule is computed as the weighted average of all encountered weight vectors (line 15).

2.1.3 Logistic Regression

Logistic regression is a important aspect in statistics and is closely related to the Perceptron and the Support Vector Machine (Section **??**). It is a classification model rather than regression (Bishop, 2007). It models the

Algorithm 1 Voted Perceptron Learning Algorithm

Input: Training sample $\{(x_i, \mathbf{y}_i)\}_{i=1}^m$, iteration limit T

Output: Weight parameter $\mathbf{w}$

$\mathbf{w}^1 = \mathbf{0}, c^1 = 0$
$k = 1$
for $t = 1 \cdots T$ **do**
  for $i = 1 \cdots m$ **do**
    Compute $\hat{y} = \mathbf{sign}(\langle \mathbf{w}, x_i \rangle)$
    if $\hat{y} = y_i$ **then**
      $c^k = c^k + 1$
    else
      $\mathbf{w}^{k+1} = \mathbf{w}^k + \rho y_i x_i$
      $c^{k+1} = 1$
      $k = k + 1$
    end if
  end for
end for
return $\mathbf{w} = \sum_{l=1}^k c^l \mathbf{w}^l$

conditional probability $p(y = +1|\mathbf{x})$ for a binary output variable $y \in \mathcal{Y}$. To model the probability, we do not restrict to any particular form, as any unknown parameters can be estimated by *maximum likelihood estimation* (MLE). However, we are most interested in the simple linear model as described in (2.1).

To use linear model in Logistic Regression, the first choice is to let $p(y = +1|\mathbf{x})$ be a linear function of $\mathbf{x}$, where the problem is the linear function is unbounded but the probability $p(\mathbf{x}) \in [0, 1]$. Another choice is to let $\log p(y = +1|\mathbf{x})$ be a linear function of $\mathbf{x}$, where the problem is the log-likelihood ranges from zero to infinite. The choice in Logistic Regression is to use logistic transformation of the original probability function

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

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

$$p(y = +1|\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + e^{-\langle \mathbf{w}, x \rangle - b}}. \tag{2.4}$$

We can compute

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

Putting together (2.4) and (2.5), we can define Logistic Regression by

Definition 2. *Logistic Regression*

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

Naturally, we expect the prediction $y = +1$ when $p(y = +1|x; \mathbf{w}, b) \geq 0.5$ and $y = -1$, when $p(y = +1|x; \mathbf{w}, b) < 0.5$. Therefore, the decision rule is similar to perceptron where we predict $y = +1$ when $\langle \mathbf{w}, x \rangle + b \geq 0$, and $y = -1$ otherwise. Besides decision boundary, Logistic Regression can output the class probability from (Definition 2) as the the 'distance' of the data point to the decision boundary. It is the probabilistic output that makes Logistic Regression no more than a classifier, as it outputs detailed predictions (class probability).

As the model is capable of outputting class probability not just class category, to learn a Logistic Regression model we attempt to learn parameters $\mathbf{w}$ and b by maximizing the probability (likelihood) of the training data. In particular, the probability of training data $\mathbf{x}$ with class label y is $p(y|\mathbf{x})$. The likelihood of parameters given data can be computed from

$$L(\mathbf{w}, b; D) = \prod_{i=1}^{m} p(y_i|\mathbf{x}_i). \tag{2.6}$$

To apply MLE, it is easier if, instead of maximizing the likelihood, we maximize the log-likelihood, which turn the product (2.6) into sum

$$\log L(\mathbf{w}, b|D) = \sum_{i=1}^{m} \log p(y_i|\mathbf{x}_i) = -\sum_{i=1}^{m} \log(1 + e^{-y_i(\langle \mathbf{x}, \mathbf{w} \rangle + b)}). \tag{2.7}$$

Though MLE training for Logistic Regression model meet our need of fitting training data, it does not guarantee to generalize well on test data. To achieve better generalization power, various smoothing techniques have been proposed (Chen and Rosenfeld, 1999, 2000; Goodman, 2003), among which adding Gaussian prior on weight parameter $\mathbf{w}$ is one of the standard treatment. In practice, a zero-mean spherical Gaussian with variance σ^2 is assumed on $\mathbf{w}$. Thus, the maximum likelihood problem (2.6) is transformed into *Maximum A-Posteriori* (MAP) problem of the following form

$$p(\mathbf{w}, b|D; \sigma^2) = p(\mathbf{w}|\sigma^2) \prod_{i=1}^{m} p(y_i|\mathbf{x}_i) = e^{-\frac{||\mathbf{w}||^2}{\sigma^2}} \prod_{i=1}^{m} \frac{1}{1 + e^{-y_i(\langle \mathbf{x}, \mathbf{w} \rangle + b)}}. \tag{2.8}$$

Instead of maximizing the posteriori (2.8), it is easier to maximize the log-posteriori

$$\log p(\mathbf{w}, b|D; \sigma^2) = -\frac{||\mathbf{w}||^2}{\sigma^2} - \sum_{i=1}^{m} \log\left(1 + e^{-y_i(\langle \mathbf{x}, \mathbf{w} \rangle + b)}\right). \tag{2.9}$$

Numbers of optimization techniques have been proposed to solve the minimization problem (2.9) in Logistic Regression, many of which are covered in the survey (Minka, 2003). For example, *Itrative Scaling* methods were proposed and continuously developed in (Darroch and Ratcliff, 1972; Della Pietra et al., 1997; Berger, 1999; Goodman, 2002; Jin et al., 2003). A quasi-Newton method was discussed in (Minka, 2003). Komarek and Moore (2005); Lin et al. (2008) have proposed truncated Newton method. And coordinate descent method was proposed in (Huang et al., 2009).

There are also efforts being made to solve the logistic regression from its dual, of which the first algorithm was proposed in (Jaakkola and Haussler, 1999). The key is to introduce a set of tighter upper bound on (2.9) that are parameterized by α. The bound should have a simple form such that the maximizing (2.9) over w can be solved analytically with α. The solution to the original problem is transferred as finding the tighter upper bound for w, which is to minimize with respect to α. The dual form is given by

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i y_i x_i x_j y_j \alpha_j + \sum_{i=1}^{m}\left[\alpha_i \log \alpha_i + (\sigma^2 - \alpha_i)\log(\sigma^2 - \alpha_i)\right] \\ \textbf{s.t.} \quad & 0 \leq \alpha \leq \sigma^2, \forall i = \{1, \cdots, m\}. \end{aligned}$$

Later on, the iterative optimization method (Keerthi et al., 2005) and the dual coordinate descent method (Yu et al., 2011) are also developed and build on the dual form.

2.1.4 Support Vector Machine (SVM)

Support Vector Machine (SVM) transfers learning problem into optimization problem. The framework was firstly introduced in (Boser et al., 1992). Theory and algorithm are detailed in e.g. (Shawe-Taylor and Cristianini, 2004; Bishop, 2007). We begin our discussion by considering a very simple case where we assume the data are linear separable that is there is a hyperplane in the feature space that separates data into two classes. In addition we assume the *separating hyperplane* have the form

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

so that $y_i = -1$ if $f(x_i) < 0$, and $y_i = +1$ if $f(x_i) > 0$. Later, we could decide the label of the test example x_{ts} via $y_{ts} = \textbf{sign}(f(x_{ts}))$, given w that achieves correct separation on the training data.

However, there can be infinite number of separating hyperplane that solves the separation problem on the training data. We wish to find one

that generalize well also on test data. A good strategy is to look for a hyperplane that keeps the maximum distance from both the training data and the test data, which is often known as *max-margin principal*. To see this, imagining putting a separating hyperplane close to one class of examples which will achieve better classification on test examples from the other class. Base on the max-margin principal, we further define two *margin hyperplanes* parallel to the separating hyperplane as

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = \pm a$$

The distance between two margin hyperplane and the separating hyperplane, γ, is called *margin*. And now the goal is to find the separating hyperplane such that it is equal distance to both margin hyperplanes while margin is maximized. Basic geometry gives us

$$\gamma = \frac{a[\langle \mathbf{w}, x \rangle + b]}{||\mathbf{w}||}. \tag{2.10}$$

The max-margin solution (Bishop, 2007) is given by

$$(\hat{\mathbf{w}}, \hat{b}) = \underset{\mathbf{w},b}{\mathbf{argmax}} \left\{ \frac{1}{||\mathbf{w}||} \underset{i}{\mathbf{min}} [y_i(\langle \mathbf{w}, x_i \rangle + b)] \right\}.$$

Notice that if one scales $\mathbf{w}$ and b by a constance factor κ the margin γ (2.10) still invariant given $\mathbf{x}$. Therefore, to avoid ambiguity we set margin hyperplane as

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = \pm 1. \tag{2.11}$$

According to (2.11), the constraints for all examples to be well separated follows

$$\begin{aligned} \langle \mathbf{w}, x_i \rangle + b \geq +1, \quad \forall y_i = +1, \\ \langle \mathbf{w}, x_i \rangle + b \leq -1, \quad \forall y_i = -1. \end{aligned}$$

Together, we have

$$y_i[\langle \mathbf{w}, x_i \rangle + b] \geq 1, \forall i \in \{1, \cdots, m\}. \tag{2.12}$$

It then follows that we can formulate the primal problem of SVM as

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

$$\begin{aligned} \underset{\mathbf{w}}{\boldsymbol{min}} \quad & \frac{1}{2} ||\mathbf{w}||^2 \\ \boldsymbol{s.t.} \quad & y_i[\langle \mathbf{w}, x_i \rangle + b] \geq 1, \forall i \in \{1, \cdots, m\}, \end{aligned}$$

where the objective is to find the weight vector of minimum norm that corresponds to maximize the margin between two class of examples, and the constraints states that the training data should be well separated. [1] [2]

We do not use (Definition 3) in practice for two reasons. First, many real world data are not separable, so that solution to (Definition 3) does not always exist. Second, the data usually come with noise and error and we do not want resulting classifier over-fit the training data. Therefore, we relax the constraints by introducing for each training example x_i a *margin slack* parameter ξ_i and rewrite the constraints (2.12) as

$$y_i[\langle \mathbf{w}, x_i \rangle + b] \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in \{1, \cdots, m\},$$

where ξ_i will allow the violation of the contraints. In particular, with $\xi_i = 0$ the data point x_i is correctly classified, and lies either on the margin or on the correct side. With $0 < \xi_i \leq 1$ the data point is correctly classified and lies between margin and separating hyperplanes. With $\xi_i > 1$ the data point is misclassified. The new goal is to maximize the margin while penalize the data points that lie on the wrong side of the margin hyperplane as

Definition 4. *Soft-Margin* SVM *Optimization Problem in primal*

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

[3]

2.2 Structured Output Prediction

2.2.1 Preliminary and Notation

2.2.2 Maximum Entropy Markov Network (MEMM)

2.2.3 Perceptron for Structured Output

The perceptron algorithm, dated back to (Rosenblatt, 1958), is one of the oldest learning algorithm in machine learning. As it is extremely easy to

[1] define the margin as maximize the minimum distance
[2] introduce support vectors
[3] dual, kernel, optimization

implement and usually achieves good performance, the perceptron algorithm is still actively studied in the field. Structured perceptron, as suggested by its name, can be seen as the generalization of the perceptron algorithm to structured output space. It was firstly proposed in (Collins, 2002; Collins and Duffy, 2002) with the formalism incredibly similar to multiclass perceptron. The model assumes a score function $\langle \mathbf{w}, \mathbf{\Phi}(x, \mathbf{y}) \rangle$ as the inner product between a joint feature map $\mathbf{\Phi}(x, \mathbf{y})$ and some feature weight parameter $\mathbf{w}$. After weight parameter $\mathbf{w}$ is obtained, one need to solve the *argmax problem* to get the best output for a given input

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\mathbf{argmax}} \quad \langle \mathbf{w}, \mathbf{\Phi}(x_i, \mathbf{y}) \rangle. \tag{2.13}$$

The weight parameter $\mathbf{w}$ is learned via standard perceptron iterative update by solving argmax operation (2.13) in each iteration. In particular, the algorithm loops through training examples and updates $\mathbf{w}$ whenever the predicted label $\hat{\mathbf{y}}$ is different from true label $\mathbf{y}$, according to

$$\mathbf{w} \leftarrow \mathbf{w} + \left(\mathbf{\Phi}(x_i, \mathbf{y}_i) - \mathbf{\Phi}(x_i, \hat{\mathbf{y}}_i) \right).$$

The update usually leads to over-fitting and a simple refinement, called "average parameter" similar to (**?**), is proposed as shown in (Algorithm 2).

Algorithm 2 Structured Perceptron with Parameter Averaging

Input: Training sample $\{(x_i, \mathbf{y}_i)\}_{i=1}^{m}$

Output: Weight parameter $\mathbf{w}$

$\mathbf{w}^{t,i} = [0, \cdots, 0]$, $\forall t \in \{1, \cdots, T\}$, $\forall i \in \{1, \cdots, m\}$
$c = 1$
for $t = 1 \cdots T$ **do**
  for $i = 1 \cdots m$ **do**
    $\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\mathbf{argmax}} \quad \langle \mathbf{w}^{t-1,i-1}, \mathbf{\Phi}(x_i, \mathbf{y}) \rangle$
    if $\hat{\mathbf{y}} \neq \mathbf{y}$ **then**
      $\mathbf{w}^{t,i} = \mathbf{w}^{t-1,i-1} + \mathbf{\Phi}(x_i, \mathbf{y}_i) - \mathbf{\Phi}(x_i, \hat{\mathbf{y}}_i)$
    end if
    $c = c + 1$
  end for
end for
return $\mathbf{w} = \frac{1}{Tm} \sum_{t=1}^{T} \sum_{i=1}^{m} \mathbf{w}^{t,i}$

In fact, structural perceptron makes two strong assumptions that limit its power and applicability. First, the framework assumes 0/1 loss over

output variables, that is $\ell(\mathbf{y}, \hat{\mathbf{y}}) = \mathbf{1}_{[\mathbf{y} \neq \hat{\mathbf{y}}]}$, such that the nearly correct and the completely incorrect output labels will lead to the same update of the weight parameter during the training. Secondly, it tacitly assumes that the argmax problem can be solved efficiently, which is generally not true on many structured output space. In the original work (Collins, 2002), the algorithm relied on Viterbi decoding, which works on sequence tagging problem.

2.2.4 Conditional Random Field (CRF)

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

Definition 5. *Conditional Random Field (CRF)*
Denote by $Y = \{\mathbf{y}_1, \cdots, \mathbf{y}_m\}$ a set of output random variables, and by $X = \{x_1, \cdots, x_m\}$ a set of input random variables to condition on. Let $G = (E, V)$ be a graph such that $Y = (Y_v)_{v \in V}$. A Conditional Random Field (CRF) defines the conditional probability distribution

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

where $Z_{x,\mathbf{w}}$ is the partition function dependent on $\mathbf{x}$ that sums over all possible output variables

$$Z_{x,\mathbf{w}} = \sum_{\mathbf{y}' \in \mathcal{Y}} \exp \langle \mathbf{w}, \mathbf{\Phi}(x, \mathbf{y}') \rangle. \tag{2.14}$$

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

The parameter $\mathbf{w}$ is solved by introducing parameter prior and maximizing the log of the resulting MAP problem as described in (Taskar et al., 2002)

$$L(\mathbf{w}) = \sum_{i=1}^{m} \left[\langle \mathbf{w}, \mathbf{\Phi}(x_i, \mathbf{y}_i) \rangle - \log \sum_{\mathbf{y}' \in \mathcal{Y}} \exp \langle \mathbf{w}, \mathbf{\Phi}(x_i, \mathbf{y}') \rangle \right] - \frac{1}{\sigma^2} ||\mathbf{w}||^2 + C. \tag{2.15}$$

The parameter learning problem, as (2.15), is solved in (Lafferty et al., 2001) via improved iterative scaling algorithm (IIS) from (**?**). In order to make CRF work in practice, one have to make sure that the log normalization function (2.14) and the argmax inference can be solve efficiently.

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

The Max-Margin Markov Network (M^3N) unifies the frameworks of kernel based discriminative learning and probabilistic probabilistic graphical model (Taskar et al., 2004). The model defines a log-linear Markov network over a set of output variables, thus, it is capable of modelling the correlation between these output variables. Recall that Support Vector Machine (SVM) searches for a feature weight vector of smaller L_2 norm, which achieves a margin of at least one on training examples for the purpose of good generalization power. Following the same formalism, M^3N also defines a margin-based quadratic programming optimization problem for the weight vector of the model. In particular, it extends SVM to structured output space via a loss function ℓ that requires the score difference between true output label $\mathbf{y}$ and any incorrect label $\hat{\mathbf{y}}$ be at least $\ell(\mathbf{y}, \hat{\mathbf{y}})$. The primal optimization problem of M^3N is given as

Definition 6. M^3N *optimization problem in primal form*

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

where ξ_i denotes the slack allotted to each example to make sure solution can always be found, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss function between pseudo-label and the correct label, and C is the slack parameter that controls the amount of regularization in the model. The intuition behind the optimization is to maximizing the margin between the correct output label $\mathbf{y}$ and any incorrect label $\hat{\mathbf{y}}$. In particular, the margin is scaled by the loss function $\ell(\mathbf{y}, \hat{\mathbf{y}})$ such that the incorrect label with bigger loss will require larger margin.

It is immediately clear that the primal optimization problem of M^3N (Definition 6) is difficult to solve as there are exponential numbers of constraints, one being instantiated for each pair of training example and pseudo label $(x, \mathbf{y})$. The corresponding dual form is also difficult as there are exponential number of dual variables (Taskar et al., 2004, p. 4). Fortunately, by exploring the Markov network structure defined on the output labels, the original problem can be formulated into factorized dual quadratic programming, as long as the loss function ℓ and joint feature map $\mathbf{\Phi}$ are decomposable over the Markov network.

As the number of parameters (weight vector) is quadratic in the number of training examples and edges in the network, it still cannot fit into standard QP solver. In the original work (Taskar et al., 2004), the authors developed a coordinate descent method analogous to the sequential minimal optimization (SMO) used for SVM (Platt, 1998). Subsequently, there are many efficient optimization algorithms being proposed, including exponential gradient optimization methods developed in (Bartlett et al., 2005), extra-gradient methods described in (Taskar et al., 2006), sub-gradient methods proposed in (Ratliff et al., 2007), and conditional gradient methods in (Rousu et al., 2006, 2007).

In order to apply the optimization methods in practice, one have to solve the *loss augmented* inference problem defined as

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}/\mathbf{y}_i}{\mathbf{argmax}} \quad \langle \mathbf{w}, \mathbf{\Phi}(x_i, \mathbf{y}) \rangle + \ell(\mathbf{y}_i, \mathbf{y}). \tag{2.16}$$

To efficiently compute (2.16), the loss function need to be decomposable over the Markov network. Nevertheless, M^{3}N is more flexible compared to CRF due to the following two reasons. First, M^{3}N model does not require partition function to be calculated. Thus, it is more feasible in terms of computation and can be applied to problems that is NP-hard for CRF. Secondly, one can define more complex loss function on M^{3}N other than just 0/1 loss for CRF, as long as the function is decomposable.

2.2.6 Support Vector Machine for Interdependent and Structured Outputs (SVM$^{\text{STRUCT}}$)

The formulation of Support Vector Machine for interdependent and structured output space (SVM$^{\text{STRUCT}}$) described in (Tsochantaridis et al., 2004, 2005) is amazingly similar to M^{3}N. Compared to M^{3}N framework which scales the margin by the loss function, SVM$^{\text{STRUCT}}$ formulation scales the margin errors (*slack variables*) by the loss function. The primal optimization problem of SVM$^{\text{STRUCT}}$ is given as

Definition 7. SVM$^{\text{STRUCT}}$ *optimization problem in primal form*

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

where ξ_i denotes the slack allotted to each example, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss function between pseudo-label and the correct label, and C is the slack

parameter that controls the amount of regularization in the model. The primal form can be interpreted as maximizing the the margin between the correct training example and the pseudo-example. Allowing margin error is to make sure solution can always be found.

It seems intuitive to scale the margin error by the loss function, the advantage of which is also claimed the authors in (Tsochantaridis et al., 2004, p.3). They suggest that under M^{3}N framework the learning system will work hard on the examples that incur big loss thought they may not even close to be confusable compared to true target value $\mathbf{y}_i$.

In addition to the slight difference in loss scaling, the optimization technique employed by SVM$^{\text{STRUCT}}$ differ significantly compared to M^{3}N. Since the loss function in SVM$^{\text{STRUCT}}$ is not decomposable, the model cannot remove exponential number of constraints during optimization. On the other hand, Tsochantaridis et al. (2004) developed an iterative optimization approach that creates a nested sequence of successively tighter relaxation of the original problem via a cutting-plane methods (Bishop, 2007; Joachims et al., 2009). In particular, constraints are added as necessary. It is also shown that the iterative optimization algorithm will converge to some optimal solution of ϵ precision in polynomial number of steps.

The major drawback of SVM$^{\text{STRUCT}}$ is the inference is usually intractable. To find the most violating constraint, the model will compute the loss-augmented inference problem as in (Tsochantaridis et al., 2005)

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}/\mathbf{y}_i}{\mathbf{argmax}} \quad [1 - \langle \mathbf{w}, \mathbf{\Phi}(x_i, \mathbf{y}) \rangle] \, \ell(\mathbf{y}_i, \mathbf{y}), \tag{2.17}$$

where the loss function appears as a multiplicative term. As the model does not assume any property of decomposition on the loss function over output space, the loss-augmented inference problem (2.17) of SVM$^{\text{STRUCT}}$ is in practice intractable. The intractability of the inference problem can be seen as an exchange of the generality of the loss function which allows more complicated loss to be defined.

2.3 Ensemble Methods

3. Methods

4. Predicting Influence Network

4.1 Problem Definition

4.2 Prior Work

4.3 Methods

5. Theory

6. Applications

7. Future Work

Bibliography

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Jin, R., Yan, R., Zhang, J., and Hauptmann, A. G. (2003). A faster iterative scaling algorithm for conditional exponential model. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 282–289.

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

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.

Komarek, P. and Moore, A. (2005). Making logistic regression a core data mining tool: A practical investigation of accuracy, speed, and simplicity.

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

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.

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

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

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research.

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

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

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

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. (2006). Kernel-Based Learning of Hierarchical Multilabel Classification Models. *The Journal of Machine Learning Research*, 7:1601–1626.

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

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

Taskar, B., Abbeel, P., and Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, pages 485–492, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

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

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

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

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.