國立臺灣大學電機資訊學院資訊工程學研究所

碩士論文

Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

master thesis

二階正規化多標籤線性分類器比較

Comparison of L2-Regularized Multi-Class Linear Classifiers

黃天亮

Tian-Liang Huang

指導教授：林智仁 博士

Advisor: Chih-Jen Lin, Ph.D.

中華民國 99 年 7 月

July, 2010

國立臺灣大學碩士學位論文
口試委員會審定書

二階正規化多標籤線性分類器比較
Comparison of L2-Regularized Multi-Class Linear Classifiers

本論文係黃天亮君（學號 R97922002）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 99 年 7 月 21 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

林智仁

（指導教授）

林軒田

系 主 任 呂育道

中文摘要

分類問題的應用可見於許多方面，譬如在文件分類和多媒體資訊檢索等。支撐向量機被用於解決二元分類的問題，而也能進一步延伸至處理多標籤分類問題，其做法爲以二元分類問題爲子問題來解決整個多標籤分類問題。還有其他方法如最大熵方法將所有類別的資料集合一併來考慮，不同於前述方法先以類別來分出子問題。而最小化目的函數的方法中，較爲常用的是以座標下降法來逐步對各個子問題求解。另一解法爲信賴區域牛頓法，其方法能得到平方收斂速度。而本篇論文將比較數種多標籤分類器的效能並加以討論。

關鍵詞: 線性分類模型, 線性支持向量機, 多標籤分類, 最大熵方法, 座標下降法。

ABSTRACT

The classification problem appears in many applications such as document classification and web page search. Support vector machine(SVM) is one of the most popular tools used in classification task. One of the component in SVM is the kernel trick. We use kernels to map data into a higher dimentional space. And this technique is applied in non-linear SVMs. For large-scale sparce data, we use the linear kernel to deal with it. We call such SVM as the linear SVM. There are many kinds of SVMs in which different loss functions are applied. We call these SVMs as L1-SVM and L2-SVM in which L1-loss and L2-loss functions are used respectively. We can also apply SVMs to deal with multi-class classification with one-against-one or one-against-all approaches. In this thesis several models such as logistic regression, L1-SVM, L2-SVM, Crammer and Singer, and maximum entropy will be compared in the multi-class classification task.

KEYWORDS: Linear classification, Linear support vector machines, Multi-class classification, Maximum entropy, Coordinate descent.

TABLE OF CONTENTS

LIST OF TABLES

Table

CHAPTER I

Introduction

Support vector machine (SVM) (Boser et al., 1992; Cortes and Vapnik, 1995) is a popular data classification tool. SVM uses a hyperplane to distinguish between positive and negative instances, and keeps the margin as wide as possible. Sometimes it is not easy to seperate data points directly. The kernel trick is introduced to map the instances into another higher dimentional space to make the classification task easier. The SVM applying kernel trick is called non-linear SVM, otherwise it is called linear SVM. For many applications such as document classification in which data is usually really large, linear SVM is suitable for data not mapped where the performance is similar between linear and non-linear SVM.

Several works make much efforts to solve linear SVM in many ways. Lin et al. (2008) proposed a trust region Newton method (TRON) to solve large-scale l2-loss linear SVM. Chang et al. (2008) also solved l2-linear SVM, with the coordinate descent method and showed that it is more efficient and stable than Pegasos (Shalev-Shwartz et al., 2007) and TRON. Hsieh et al. (2008) again applied the coordinate descent method to solve linear SVMs with not only l2-loss but also l1-loss. It was shown to be faster than Pegasos, TRON and SVM$^{\text{perf}}$ (Joachims, 2006), and reach an ϵ-accurate solution in $O(\log(1/\epsilon))$ iterations.

The original SVM solves the binary classification problem. In other words, there

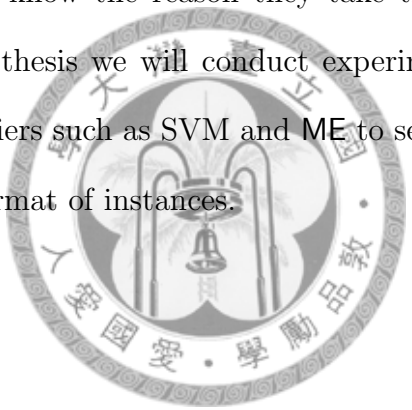
are only two classes of instances to be classified. If in one problem there are more than two classes, we call this a multi-class classification problem. There are already several works (e.g., Mayoraz and Alpaydin, 1999; Weston and Watkins, 1998) on solving multi-class problems. Hsu and Lin (2002) compares multi-class non-linear SVMs with RBF kernels of different approaches and other multi-classifiers. One group of approaches is based on several binary SVMs. In (Bottou et al., 1994), a one-against-all method is applied to deal with multi-class tasks. Rifkin and Klautau (2004) also studies the one-against-all approach. With the one-against-all method, for every class a classifier is trained. The instance will be classified to the class whose classifier gives the highest score. The other approach is called one-against-one (Knerr et al., 1990). In this approach the classifier for every pair of classes is trained. The decision of which class the instance belongs to depends on the majority votes of some associated classifiers.

Besides the approaches combining multiple binary classifiers to deal with multi-class tasks, the other group of approaches called "all-together" methods can be capable of multi-class classification. Crammer and Singer (2000) proposed a method to do with multi-class task and we usually solve the dual form. Because in the dual form there are still many variables, the coordinate descent method can be applied to solve the problem. Keerthi et al. (2008) extended the coordinate descent method to a sequential dual method to solve the dual form. Fan et al. (2008) provided an implementation in the LIBLINEAR package.

Another "all-together" model called Maximum Entropy (ME) (Berger et al., 1996) is widely used in NLP applications. ME is based on the joint features which are extracted from the instance-label pair. Huang et al. (2009) proposed an iterative scaling and coordinate descent method to solve the primal form of ME. The same method is also applied to solve logistic regression (LR) problem because LR can be considered as a

special case of ME. For the dual form of ME and LR, Yu et al. (2010) solved it with the coordinate descent method as well. In the experiments of both papers, the NLP data is of multi-class with 0/1 feature values, whereas the data for LR is binary. There are no experiments conducted with these methods on multi-class data of real-valued features.

It is quite interesting that people of NLP groups get used to apply ME in their researches. In ME data appear in the form of feature vectors encoded with all the instance-label pairs instead of simply considering the instance-label pairs appearing in the data. And NLP researchers manipulate their data with "feature reduction" approach. Since the performance of classifiers may be different in terms of instance format. We would like to know the reason they take this approach or it is just a customary choice. In this thesis we will conduct experiments on several multi-class data sets with linear classifiers such as SVM and ME to see the performance, and also study effects of different format of instances.



CHAPTER II

Models

2.1 Support Vector Machine

Support Vector Machine (SVM) (Boser et al., 1992; Cortes and Vapnik, 1995) is a popular tool for classification. Given a set of instance-label pairs $(\boldsymbol{x}_i, y_i)$, where $i = 1, \ldots, l$, $\boldsymbol{x}_i \in R^n$, $y_i \in \{+1, -1\}$, SVM solves the following unconstrained optimization problem,

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i).$$

$\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$ is the loss function and $C \in R$ is the penalty parameter. $\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$ is the regularization term and it is added to avoid overfitting. One can tune the performance, and strike a balance between loss function and regularization term by setting different values of C. There are two common forms of loss function. One is L1-SVM which solves the following problem,

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\max(1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i, 0). \tag{2.1}$$

The other is L2-SVM, which solves the following problem with squared loss function,

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\max(1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i, 0)^2. \tag{2.2}$$

Regularized logistic regression (LR) related to SVM uses logistic loss function and

solves the following problem,

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\log(1+e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i}). \tag{2.3}$$

The statements above follows Chang et al. (2008).

In some applications we want to add the bias term. For convenience we can extend the instance by including the bias term b:

$$\boldsymbol{x}_i^T \leftarrow [\boldsymbol{x}_i^T,\ 1] \text{ and } \boldsymbol{w}^T \leftarrow [\boldsymbol{w}^T,\ b].$$

In the multi-class classification problem, instance-label pairs also given in the format as that in binary problem, but $y_i \in \{1,\ldots,k\}$, where k is the number of labels. The i-th SVM solves the following problem,

$$\min_{\boldsymbol{w}_i} \quad \frac{1}{2}\boldsymbol{w}_i^T\boldsymbol{w}_i + C\sum_{i=1}^{l}\max(1-y_j\boldsymbol{w}_i^T\boldsymbol{x}_j, 0). \tag{2.4}$$

The decision function for predicting labels on $\boldsymbol{x}_j$ is

$$\arg\max_{i=1,\ldots,k} \boldsymbol{w}_i^T\boldsymbol{x}_j.$$

The approach used in (2.4) is called one-against-all (Bottou et al., 1994). For each class we train one model, in total we obtain k models. In i-th SVM we consider instances with label y_i as positive ones and others as negative instances.

The other approach is called one-against-one (Knerr et al., 1990). We train models for every pair of classes. For class i and j, we solve the following problem,

$$\min_{\boldsymbol{w}_{ij}} \quad \frac{1}{2}\boldsymbol{w}_{ij}^T\boldsymbol{w}_{ij} + C\sum_{k=1}^{l}\max(1-y_k\boldsymbol{w}_{ij}^T\boldsymbol{x}_k, 0). \tag{2.5}$$

We train models for every pair of classes, therefore $k(k-1)/2$ models could be obtained. When predicting labels, for each model we calculate $\boldsymbol{w}_{ij}^T\boldsymbol{x}_i$. If $\text{sign}(\boldsymbol{w}_{ij}^T\boldsymbol{x}_i)$ says $\boldsymbol{x}_i$ is in the class i, the model votes for class i, otherwise it votes for class j. $\boldsymbol{x}_i$ will be classified in the class with majority votes.

2.2 Crammer and Singer

Given a set of instance-label pairs $(\boldsymbol{x}_i, y_i)$, where $i = 1, \ldots, l$, $\boldsymbol{x} \in R^n$, $y_i \in \{1, \ldots, k\}$. The multi-class model proposed by Crammer and Singer (2000) is obtained by solving the following optimization problem,

$$\begin{aligned} \min_{\boldsymbol{w}_m, \xi_i} \quad & \frac{1}{2}\sum_{m=1}^{k} \boldsymbol{w}_m^T \boldsymbol{w}_m + C\sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & \boldsymbol{w}_{y_i}^T \boldsymbol{x}_i - \boldsymbol{w}_m^T \boldsymbol{x}_i \geq e_i^m - \xi_i, \ i = 1, \ldots, l, \end{aligned} \tag{2.6}$$

where

$$e_i^m = \begin{cases} 0 & \text{if } y_i = m, \\ 1 & \text{if } y_i \neq m. \end{cases}$$

The decision function is

$$\arg\max_{m=1,\ldots,k} \boldsymbol{w}_m^T \boldsymbol{x}.$$

The dual of (2.6) is:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{m=1}^{k} \|\boldsymbol{w}_m\|^2 + \sum_{i=1}^{l}\sum_{m=1}^{k} e_i^m \alpha_i^m \\ \text{subject to} \quad & \sum_{m=1}^{k} \alpha_i^m = 0, \forall i = 1, \ldots, l \\ & \alpha_i^m \leq C_{y_i}^m, \forall i = 1, \ldots, l, m = 1, \ldots, k, \end{aligned} \tag{2.7}$$

where

$$\boldsymbol{w}_m = \sum_{i=1}^{l} \alpha_i^m \boldsymbol{x}_i, \forall m, \quad \bar{\boldsymbol{\alpha}}_i = [\alpha_i^1, \ldots, \alpha_i^k]^T, \quad \boldsymbol{\alpha} = [\alpha_1^1, \ldots, \alpha_1^k, \ldots, \alpha_l^1, \ldots, \alpha_l^k]^T \tag{2.8}$$

and

$$C_{y_i}^m = \begin{cases} 0 & \text{if } y_i \neq m, \\ C & \text{if } y_i = m. \end{cases} \tag{2.9}$$

2.3 Maximum Entropy (ME)

Maximum entropy (ME) is widely applied in applications such as natural language processing (NLP) and document classification. It is suitable for problems with probability interpretations. In the application of NLP, one can predict the labels with the maximal probability for a given word sequence (Berger et al., 1996). This task is different from traditional classification problems in which labels are assigned to a single instance.

ME models the conditional probability as:

$$P_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i) \equiv \frac{S_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i)}{T_{\boldsymbol{w}}(\boldsymbol{x}_i)},$$
$$S_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i) \equiv e^{\boldsymbol{w}^T \boldsymbol{f}(\boldsymbol{x}_i, y_i)}, \quad T_{\boldsymbol{w}}(\boldsymbol{x}_i) \equiv \sum_{i=1} S_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i),$$

where $i = 1, \ldots, l$, $\boldsymbol{x} \in R^n$, $y_i \in \{1, \ldots, k\}$. $\boldsymbol{x}_i$ indicates a context, y_i is the label of the context. $\boldsymbol{f}(\boldsymbol{x}_i, y_i)$ is a real-valued vector extracted from context $\boldsymbol{x}_i$ and label y_i. It is assumed that $\boldsymbol{f}(\boldsymbol{x}_i, y_i)$ has finite features. $T_{\boldsymbol{w}}$ can be considered as a normalization term to make $\sum_i P_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i) = 1$.

One can obtain the empirical probability $\tilde{P}(\boldsymbol{x}_i, y_i)$ from training instances. ME minimizes the following negative likelihood,

$$\min_{\boldsymbol{w}} \quad -\sum_{\boldsymbol{x}_i, y_i} \tilde{P}(\boldsymbol{x}_i, y_i) P_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i),$$

or equivalently,

$$\min_{\boldsymbol{w}} \quad \sum_{\boldsymbol{x}_i} \tilde{P}(\boldsymbol{x}_i) \log P_{\boldsymbol{w}}(\boldsymbol{x}_i) - \boldsymbol{w}^T \tilde{P}(\boldsymbol{f}),$$

where

$$\tilde{P}(\boldsymbol{x}_i, y_i) = N_{\boldsymbol{x}_i, y_i}/N. \tag{2.10}$$

$N_{\boldsymbol{x}_i, y_i}$ is the number of occurrences of $(\boldsymbol{x}_i, y_i)$ in the training data and N is the total number of training samples. $\tilde{P}(\boldsymbol{x}_i) = \sum_i P(\boldsymbol{x}_i, y_i)$ is the marginal probability

of $\boldsymbol{x}_i$. $\tilde{P}(\boldsymbol{f}) = \sum_{\boldsymbol{x}_i, y_i} \boldsymbol{f}(\boldsymbol{x}_i, y_i)$ is the expected feature vector. In reading data of LIBSVM format, $\tilde{P}(\boldsymbol{x}_i, y_i) = 1/l$ because every instance has an unique label and there are totally l instances, and $\tilde{P}(\boldsymbol{x}_i) = 1/l$ because $\tilde{P}(\boldsymbol{x}_i, y_j) = 0\ \forall j \neq i$. One can add a regularization term to avoid overfitting and solve the following optimization problem,

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) \equiv \sum_{\boldsymbol{x}_i} \tilde{P}(\boldsymbol{x}_i) \log P_{\boldsymbol{w}}(\boldsymbol{x}_i) - \boldsymbol{w}^T \tilde{P}(\boldsymbol{f}) + \frac{1}{2\sigma^2} \boldsymbol{w}^T \boldsymbol{w}, \tag{2.11}$$

where σ^2 is the penalty parameter. The dual form of (2.11) is:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2\sigma^2} \boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{w}(\boldsymbol{\alpha}) + \sum_i \sum_{j:\alpha_{iy_j}>0} \alpha_{iy_j} \log \alpha_{iy_j} \\ \text{subject to} \quad & \sum_y \alpha_{iy} = \tilde{P}(\boldsymbol{x}_i) \text{ and } \alpha_{iy_j} \geq 0\ \forall i, j, \end{aligned} \tag{2.12}$$

where

$$\boldsymbol{w}(\boldsymbol{\alpha}) \equiv \sigma^2 \left(\tilde{P}(\boldsymbol{f}) - \sum_{i,j} \alpha_{iy_j} \boldsymbol{f}(\boldsymbol{x}_i, y_j) \right).$$

And the derivation of (2.12) is in Yu et al. (2010). The vector $\boldsymbol{\alpha} \in R^{lk}$ is composed of l blocks,

$$\boldsymbol{\alpha} = [\bar{\boldsymbol{\alpha}}_1, \ldots, \bar{\boldsymbol{\alpha}}_l]^T \text{ and } \bar{\boldsymbol{\alpha}}_i = [\alpha_{i1}, \ldots, \alpha_{ik}]^T,$$

where k is the number of labels, and $\boldsymbol{\alpha}_i$ corresponds to $\boldsymbol{x}_i$ in the data set. If $\boldsymbol{\alpha}^*$ and $\boldsymbol{w}^*$ are respectively the optimal solution for dual and primal problems, then $\boldsymbol{w}^*(\boldsymbol{\alpha}^*) = \boldsymbol{w}^*$.

CHAPTER III

Methods

In this chapter several optimization problems will be solved with different methods.

3.1 Trust Region Newton Method (TRON)

The trust region Newton method (TRON) (Lin et al., 2008) solves the optimization problem with the following sub-problem,

$$\begin{aligned} \min_{\boldsymbol{s}} \quad & q_k(\boldsymbol{s}) \\ \text{subject to} \quad & \|\boldsymbol{s}\| \leq \Delta_k, \end{aligned} \tag{3.1}$$

where $q_k(\boldsymbol{s})$ is the following quadratic model,

$$q_k(\boldsymbol{s}) \equiv \nabla f(\boldsymbol{w}^k)^T \boldsymbol{s} + \frac{1}{2} \boldsymbol{s}^T \nabla^2 f(\boldsymbol{w}^k)^T \boldsymbol{s},$$

where $\nabla f(\boldsymbol{w}^k)$ and $\nabla^2 f(\boldsymbol{w}^k)$ are the graident and the Hessian at $\boldsymbol{w}^k$ respectively, Δ_k is the size of the trust region. Generally TRON requires that the objective function is twice differentiable in order to calculate the Hessian-vector product.

Algorithm 1 Trust region Newton method.

- Given $\boldsymbol{w}^0$.
- For $k = 0, 1, \ldots$
  - Find an approximate solution $\boldsymbol{s}^k$ of the trust region sub-problem (3.1).
  - Update $\boldsymbol{w}^k$ and Δ_k according to (3.2).

At each iteration k, $\boldsymbol{w}^k$ and Δ_k is updated with the following rules,

$$\begin{aligned}
\boldsymbol{w}^{k+1} &= \begin{cases} \boldsymbol{w}^k + \boldsymbol{s}^k & \text{if } \rho_k > \eta_0, \\ \boldsymbol{w}^k & \text{if } \rho_k \le \eta_0, \end{cases} \\
\Delta_{k+1} &\in \begin{cases} [\sigma_1 \min\{\|\boldsymbol{s}^k\|, \Delta_k\}, \sigma_2 \Delta_k] & \text{if } \rho_k < \eta_1, \\ [\sigma_1 \Delta_k, \sigma_3 \Delta_k] & \text{if } \rho_k \in (\eta_1, \eta_2), \\ [\Delta_k, \sigma_3 \Delta_k] & \text{if } \rho_k > \eta_2, \end{cases} \\
\rho_k &= \frac{f(\boldsymbol{w}^k + \boldsymbol{s}^k) - f(\boldsymbol{w}^k)}{q_k(\boldsymbol{s}^k)},
\end{aligned} \tag{3.2}$$

where ρ_k is the ratio of the actual function value reduction to the approximated reduction. One can pre-specify parameters $\eta_0 > 1, 1 > \eta_2 > \eta_1 > 0$, and $\sigma_3 > 1 > \sigma_2 > \sigma_1 > 0$. In Lin et al. (2008) it is suggested that

$$\eta_0 = 10^{-4}, \eta_1 = 0.25, \eta_2 = 0.75,$$

$$\sigma_1 = 0.25, \sigma_2 = 0.5, \sigma_3 = 4.$$

The precedure is in Algorithm 1.

3.1.1 Logistic Regression (LR)

TRON requires gradient and Hessian for computation. The gradient of (2.3) is

$$\boldsymbol{w} + C \sum_{i=1}^{l} \frac{y_i \boldsymbol{x}_i}{e^{y_i \boldsymbol{w}^T \boldsymbol{x}_i} - 1}.$$

And the Hessian is

$$\mathcal{I} + 2CX^T DX,$$

where $\mathcal{I}$ is the identity matrix and D is a diagonal matrix with

$$D_{ii} = \sigma(y_i \boldsymbol{w}^T \boldsymbol{x}_i)/(1 - \sigma(y_i \boldsymbol{w}^T \boldsymbol{x}_i)), \sigma(y_i \boldsymbol{w}^T \boldsymbol{x}_i) = \frac{1}{1 + e^{y_i \boldsymbol{w}^T \boldsymbol{x}_i}}.$$

3.1.2 L2-Loss Support Vector Machine (L2-SVM)

The gradient of (2.2) is

$$\boldsymbol{w} + 2CX_{I,:}^T(X_{I,:}^T \boldsymbol{w} - \boldsymbol{y}_I),$$

where $I \equiv \{i \mid 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i > 0\}$ is the set of indices, $\boldsymbol{y} = [y_1, \ldots, y_l]^T$ and $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l]^T$.

Because (2.2) is differentiable but not twice differentiable, we use the generalized Hessian of (2.2) as the following,

$$\mathcal{I} + 2CX^T DX = \mathcal{I} + 2CX_{I,:}^T D_{I,I} X_{I,:}, \tag{3.3}$$

where $\mathcal{I}$ is the identity matrix. D is a diagonal matrix defined as the following,

$$D_{ii} = \begin{cases} 1 \text{ if } 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i > 0, \\ 0 \text{ if } 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i \leq 0. \end{cases}$$

The Hessian-vector product of (3.3) and $\boldsymbol{s}$ is

$$\boldsymbol{s} + 2CX_{I,:}^T(D_{I,I}(X_{I,:}\boldsymbol{s})).$$

3.2 Coordinate Descent

Coordinate descent method is applied to solve the following constrained optimization problem,

$$\begin{aligned} \min_{\boldsymbol{\alpha} \in R^l} \quad & f(\boldsymbol{\alpha}) \\ \text{subject to} \quad & A\boldsymbol{\alpha} = \boldsymbol{b} \quad \text{and } \mathbf{0} \leq \boldsymbol{\alpha} \leq C\boldsymbol{e}, \end{aligned}$$

where $A \in R^{m \times l}$, $\boldsymbol{b} \in R^l$ and $\boldsymbol{e}$ is a vector of ones.

It is too expensive to update all variables in one iteration. Coordinate descent method solves one variable or a block of elements of $\boldsymbol{\alpha}$ at a time to form a sub-problem as the following,

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & f(\boldsymbol{\alpha} + \boldsymbol{z}) \\ \text{subject to} \quad & z_i = 0, \quad \forall z_i \notin B \\ & A\boldsymbol{z} = \boldsymbol{0} \quad \text{and } 0 \leq \alpha_i + z_i \leq C, \forall z_i \in B, \end{aligned} \tag{3.4}$$

where $A\boldsymbol{z} = \boldsymbol{0}$ is the linear constraint, B contains variables to be updated.

3.2.1 Support Vector Machine Dual with L1-loss and L2-loss

Hsieh et al. (2008) proposed a coordinate descent method to solve the dual form of (2.2) and (2.1),

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \tfrac{1}{2}\boldsymbol{\alpha}^T \bar{Q}\boldsymbol{\alpha} + \boldsymbol{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq U, \forall i, \end{aligned} \tag{3.5}$$

where $\bar{Q} = Q + D$, D is a diagonal matrix and $Q_{ij} = y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j$. For L1-SVM, $U = C$ and $D_{ii} = 0$,$\forall i$ and for L2-SVM, $U = \infty$ and $D_{ii} = 1/(2C)$,$\forall i$.

We apply the coordinate descent method to solve the sub-problem,

$$\begin{aligned} \min_{z} \quad & f(\boldsymbol{\alpha}^{k,i} + z\boldsymbol{e}) \\ \text{subject to} \quad & 0 \leq \alpha_i^k + z \leq U, \end{aligned} \tag{3.6}$$

where $\boldsymbol{\alpha}^{k,i} = [\alpha_1^{k+1}, \ldots, \alpha_{i-1}^{k+1}, \alpha_i^k, \ldots, \alpha_l^k]^T$, $\forall i = 2, \ldots, l$ and $f(\boldsymbol{\alpha}^{k,i} + z\boldsymbol{e})$ is a single variable function of z:

$$f(\boldsymbol{\alpha}^{k,i} + z\boldsymbol{e}) = \frac{1}{2}\bar{Q}_{ii}d^2 + \nabla_i f(\boldsymbol{\alpha}^{k,i})d + \text{constant}, \tag{3.7}$$

where $\nabla_i f$ is the i-th component of the gradient ∇f.

If $Q_{ii} > 0$ one can easily obtain the solution:

$$\alpha_i^{k,i+1} = \min\left(\max\left(\alpha_i^{k,i} - \frac{\nabla_i f(\boldsymbol{\alpha}^{k,i})}{\bar{Q}_{ii}}, 0\right), U\right). \tag{3.8}$$

3.2.2 Crammer and Singer

Since (2.7) in which there are kl variables is too large, Keerthi et al. (2008) extends the coordinate descent method the solve the sub-problem by splitting $\boldsymbol{\alpha}$ into blocks. Every time we choose a block $\bar{\boldsymbol{\alpha}}_i$ to solve the following sub-problem,

$$\begin{aligned} \min_{\bar{\boldsymbol{\alpha}}_i} \quad & \sum_{m=1}^{k} \frac{1}{2} A(\alpha_i^m)^2 + B_m \alpha_i^m \\ \text{subject to} \quad & \sum_{m=1}^{k} \alpha_i^m = 0, \\ & \alpha_i^m \le C_{y_i}^m, m = \{1, \ldots, k\}, \end{aligned}$$

where

$$A = \boldsymbol{x}_i^T \boldsymbol{x}_i \text{ and } B_m = \boldsymbol{w}_m^T \boldsymbol{x}_i + e_i^m - A\alpha_i^m.$$

Because some variables will become bounded, they can be shrunken during training. We can eliminate the number of variabls need to be solved. These active variables could be considered as a sub-vector $\bar{\boldsymbol{\alpha}}_i^{U_i}$, where $U_i \subset \{1, \ldots, k\}$ is an active set. Then we solve the following sub-problem with other variables not in U_i fixed,

$$\begin{aligned} \min_{\bar{\boldsymbol{\alpha}}_i^{U_i}} \quad & \sum_{m \in U_i} \frac{1}{2} A(\alpha_i^m)^2 + B_m \alpha_i^m \\ \text{subject to} \quad & \sum_{m \in U_i} \alpha_i^m = -\sum_{m \notin U_i} \alpha_i^m, \\ & \alpha_i^m \le C_{y_i}^m, m \in U_i. \end{aligned} \tag{3.9}$$

There are two possibilities that we don't solve the sub-problem of $\bar{\boldsymbol{\alpha}}_i$. If $|U_i| < 2$, then the whole vector $\bar{\boldsymbol{\alpha}}_i$ is fixed by the constraints in (3.9), thus we can shrink the

Algorithm 2 The coordinate descent method for (2.7)

- Given $\boldsymbol{\alpha}$ and the corresponding $\boldsymbol{w}_m$
- While $\boldsymbol{\alpha}$ is not optimal
  - Randomly permute $\{1, \ldots, l\}$ to $\{\pi(1), \ldots, \pi(l)\}$
  - For $i = \pi(1), \ldots, \pi(l)$
    If $\bar{\boldsymbol{\alpha}}_i$ is active and $\boldsymbol{x}_i^T \boldsymbol{x}_i \neq 0$
    · Solve a $|U_i|$-variable sub-problem (3.9)
    · Maintain $\boldsymbol{w}_m$ for all m by (3.10)

whole vector $\bar{\boldsymbol{\alpha}}_i$. If $A = 0$ then $\boldsymbol{x}_i = \mathbf{0}$, with (2.8) we can know that any value of $\bar{\alpha}_i^m$ won't affect the vector $\boldsymbol{w}_m$, thus we can finish solving the sub-problem.

After solving the sub-problem, if $\hat{\alpha}_i^m$ is the new value and α_i^m is the old value then we can update $\boldsymbol{w}_m$ by

$$\boldsymbol{w}_m \leftarrow \boldsymbol{w}_m + (\alpha_i^m - \hat{\alpha}_i^m) y_i \boldsymbol{x}_i. \tag{3.10}$$

And we can save all the elements that $\hat{\alpha}_i^m \neq \alpha_i^m$ to save computation time. The procedure in in Algorithm 2.

3.2.3 Maximum Entropy (ME)

Yu et al. (2010) proposed a coordinate descent method to solve (2.12). Every time we pick a block $\bar{\boldsymbol{\alpha}}_i$ to form the sub-problem,

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & h(\boldsymbol{z}) \\ \text{subject to} \quad & \sum_j z_{y_j} = 0 \text{ and } z_{y_j} \geq -\alpha_{iy_j} \ \forall j, \end{aligned} \tag{3.11}$$

where

$$\begin{aligned} h(\boldsymbol{z}) \equiv \ & D^{\mathsf{ME}}(\bar{\boldsymbol{\alpha}}_1, \ldots, \bar{\boldsymbol{\alpha}}_i + \boldsymbol{z}, \ldots, \bar{\boldsymbol{\alpha}}_l) \\ = \ & \sum_j (\alpha_{iy_j} + z_{y_j}) \log(\alpha_{iy_j} + z_{y_j}) + \frac{1}{2\sigma^2} \left(\boldsymbol{w}(\boldsymbol{\alpha}) - \sigma^2 \sum_j z_{y_j} \boldsymbol{f}(\boldsymbol{x}_i, y_j) \right)^2 + \text{constant} \\ = \ & \sum_j (\alpha_{iy_j} + z_{y_j}) \log(\alpha_{iy_j} + z_{y_j}) - \sum_j z_{y_j} \boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{f}(\boldsymbol{x}_i, y_j) + \frac{\sigma^2}{2} \boldsymbol{z}^T K^i \boldsymbol{z} + \text{constant}, \end{aligned}$$

where $K^i \in R^{|Y| \times |Y|}$ is a matrix with $K^i_{yy'} = f(\boldsymbol{x}_i, y)^T f(\boldsymbol{x}_i, y'), \forall y, y' \in Y, Y = \{1, \ldots, k\}$.

Following Memisevic (2006), the sub-problem becomes a two-level coordinate descent method. Each time we pick two variables α_{iy_1} and α_{iy_2}, we form the following one-variable sub-problem,

$$\begin{aligned}
\min_{d} \quad & h(\boldsymbol{z} + d(\boldsymbol{e}_{y_1} - \boldsymbol{e}_{y_2})) \\
& = (\alpha_{iy_1} + z_{y_1} + d)\log(\alpha_{iy_1} + z_{y_1} + d) + (\alpha_{iy_2} + z_{y_2} - d)\log(\alpha_{iy_2} + z_{y_2} - d) \\
& + \left(\sigma^2\left((K^i\boldsymbol{z})_{y_1} - (K^i\boldsymbol{z})_{y_2}\right) - \boldsymbol{w}(\boldsymbol{\alpha})^T\left(\boldsymbol{f}(\boldsymbol{x}_i, y_1) - \boldsymbol{f}(\boldsymbol{x}_i, y_2)\right)\right) d \\
& + \frac{\sigma^2}{2}(K^i_{y_1y_1} + K^i_{y_2y_2} - 2K^i_{y_1y_2})d^2 + \text{constant} \\
\text{subject to} \quad & -(\alpha_{iy_1} + z_{y_1}) \le d \le \alpha_{iy_2} + z_{y_2}
\end{aligned} \tag{3.12}$$

In Yu et al. (2010), a coordinate descent method also proposed to solve LR dual problem. One can obtain the one-variable sub-problem,

$$\begin{aligned}
\min_{z} \quad & g(z) \equiv (c_1 + z)\log(c_1 + z) + (c_2 - z)\log(c_2 - z) + \frac{a}{2}z^2 + bz \\
\text{subject to} \quad & -c_1 \le z \le c_2,
\end{aligned} \tag{3.13}$$

where $c_1, c_2, a, b \in R$.

By assigning

$$\begin{aligned}
& a \leftarrow \sigma^2(K^i_{y_1y_1} + K^i_{y_2y_2} - 2K^i_{y_1y_2}) \\
& b \leftarrow \sigma^2\left((K^i\boldsymbol{z})_{y_1} - (K^i\boldsymbol{z})_{y_2}\right) - \boldsymbol{w}(\boldsymbol{\alpha})^T\left(\boldsymbol{f}(\boldsymbol{x}_i, y_1) - \boldsymbol{f}(\boldsymbol{x}_i, y_2)\right) \\
& c_1 \leftarrow \alpha_{iy_1} + z_{y_1} \text{ and } c_2 \leftarrow \alpha_{iy_2} + z_{y_2},
\end{aligned} \tag{3.14}$$

(3.12) is in the same form as (3.13), then we can solve it with the modified Newton method in Yu et al. (2010).

In Algorithm 3, $\boldsymbol{\alpha}$ can be initialized as:

$$\alpha_{iy_j} = \begin{cases} \tilde{P}(\boldsymbol{x}_i, y_j) & \text{if } |E_i| = 0, \\ \begin{cases} (1-\epsilon)\tilde{P}(\boldsymbol{x}_i, y_j) & \forall y_j \notin E_i \\ \frac{\epsilon}{|E_i|}\tilde{P}(\boldsymbol{x}_i) & \forall y_j \in E_i \end{cases} & \text{if } |E_i| \neq 0, \end{cases} \tag{3.15}$$

Algorithm 3 Coordinate Descent Method for the dual of ME (2.12)

- Set initial $\boldsymbol{\alpha}$ by (3.15).
- $\boldsymbol{w}(\boldsymbol{\alpha}) \leftarrow \sigma^2 \left(\sum_i \sum_j \left(\tilde{P}(x_i, y_j) - \alpha_{iy_j} \right) \boldsymbol{f}(x_i, y_j) \right)$.
- While $\boldsymbol{\alpha}$ is not optimal
  For $i = 1, \ldots, l$
  - Solve the sub-problem (3.11) and get the optimal $\hat{\boldsymbol{z}}^*$.
  - Update $\boldsymbol{\alpha}$ and $\boldsymbol{w}(\boldsymbol{\alpha})$ by (3.16).

where $E_i \equiv \{y_j \mid \tilde{P}(\boldsymbol{x}_i, y_j) = 0\}, \tilde{P}(\boldsymbol{x}_i)$ and $\tilde{P}(\boldsymbol{x}_i, y_j)$ is defined in (2.10). And the update rule for $\boldsymbol{w}(\boldsymbol{\alpha})$ and $\boldsymbol{\alpha}$ is

$$\begin{aligned} &\bar{\boldsymbol{\alpha}}_i \leftarrow \hat{\boldsymbol{z}}^*, \\ &\boldsymbol{w}(\boldsymbol{\alpha}) \leftarrow \boldsymbol{w}(\boldsymbol{\alpha}) - \sigma^2 \sum_j \left(\hat{z}^*_{y_j} - \alpha_{iy_j} \right) \boldsymbol{f}(\boldsymbol{x}_i, y_j). \end{aligned} \tag{3.16}$$

CHAPTER IV

Features of Different Schemes

Researchers of NLP groups tend to apply ME to conduct experiments on their data. It is quite interesting that why they applied ME instead of other classification models. One feature of ME is that instead of instances with knowledge of its associated label is required, feature vectors encoded with every instance-label pair are used in the optimization procedure. In NLP data, property (4.2) usually holds (Jurafsky and Martin, 2008). Therefore in the beginning we construct the feature vector with the following rules,

$$\boldsymbol{f}(\boldsymbol{x}_i, y_i) = [\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_t, \ldots, \boldsymbol{f}_k]^T, \tag{4.1}$$

$$\boldsymbol{f}_t = \begin{cases} \boldsymbol{x}_i^T & \text{if } y_i = t, \\ [0, \ldots, 0] & \text{otherwise.} \end{cases}$$

This setting satisfies the property,

$$\boldsymbol{f}(\boldsymbol{x}_i, y_i)^T \boldsymbol{f}(\boldsymbol{x}_i, y_j) = 0 \ \forall i \neq j, \tag{4.2}$$

which can be applied in Algorithm 3.

With (4.1), there are kn features in one vector, where k and n are the number of labels and features respectively. Since in SVM we consider instances as vector of dimention n, (4.1) could be treated as the 'expansion' of instances in ME dual. At first

glance the lengthened instances may lead to longer computation time. In fact, we could consider (4.1) as putting the original instance into a new longer instance of length kn, and this approach will lead to a vector with leading and trailing zeros. Because the instances are stored in a sparce format, the duplicating zeros add not much workloads for reading each feature vectors. But the number of instances becomes k times will lead to more cost in calculation than with SVM.

For all the feature vectors of the same associated label there are some positions where the value is always zero, these features can be reduced to reach the smaller size of feature vectors and the reduced vectors also possess the property (4.2). Data sets with high sparcity enjoy the advantage of this reducing technique. Reduced vectors can be written in the form of (4.3). Table 4.1 contains the statistics of size of original and reduced feature vectors of several data sets. news20, rcv1 and sector are benefit from the feature reducing approach.

$$\boldsymbol{f}'(\boldsymbol{x}_i, y_i) = [\boldsymbol{f}'_1, \boldsymbol{f}'_2, \ldots, \boldsymbol{f}'_t, \ldots, \boldsymbol{f}'_k]^T, \tag{4.3}$$

$$\boldsymbol{f}'_t = \boldsymbol{f}_{tI_t}$$

where

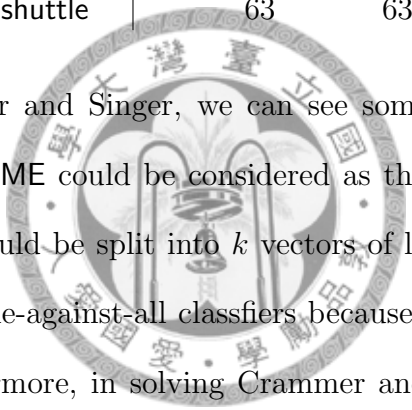
$$I_t = \{j \mid y_i = t,\ \exists x_{mj} \neq 0\ \forall m = 1, \ldots, l\}, \tag{4.4}$$

is the index set of reduced instances of label t and $\boldsymbol{f}_t$ is in (4.1).

ME dual-2 uses reduced feature vectors for computation. Before the computation a look-up table is constructed to memorize the positions of non-zero values in the feature vectors of different labels. The action of looking up the table can be the bottleneck. This phenomenon doesn't exist in ME dual because we just read all the features in the optimization procedure without accessing the table to check whether the features should be involved in the calculation or not.

Table 4.1: The number of features in the feature vectors used in Algorithm 3. 'original' indicates the length of the original feature vector while the length of the reduced feature vector is listed in the column 'reduced.'

Data set	original	reduced
news20	1,241,220	210,981
covtype	378	239
mnist	7,800	5,777
rcv1	2,503,508	478,263
sector	5,795,685	299,454
vehicle	300	300
iris	12	12
wine	39	39
glass	54	54
vowel	110	110
segment	133	126
dna	540	535
satimage	216	216
letter	416	416
shuttle	63	63



Comparing to Crammer and Singer, we can see some common properties. The solution of primal form of ME could be considered as the concatenaton of classifiers of each class, that is, $\boldsymbol{w}$ could be split into k vectors of length n. Thus we can treat these two models as the one-against-all classfiers because their decision functions are in the same form. Furthurmore, in solving Crammer and Singer, we can also apply the feature reduction technique to reduce the size of the primal solution, and the same trick could be adopted in the multi-class SVM with one-against-all approach. However, in the thesis we only compare the different scheme of features on ME dual to see the effect.

CHAPTER V

Experiment

In this chapter, we will run experiments with different methods on various data sets. The original LIBLINEAR package can deal with multi-class classification by applying one-against-all approach. The package is available at

`http://www.csie.ntu.edu.tw/~cjlin/liblinear/`.

And the one-against-one version is implemented by modifying the code of LIBLINEAR. The implementation of ME dual is similar to the Java code version in Yu et al. (2010).

All the experiments are conducted on a 64-bit machine with Intel Xeon 2.5GHz CPU and 16GB main memory.

5.1 Data Sets

Data sets are roughly seperated into two categories. Table 5.1 lists the statistics of large-scale data sets. Besides other data sets, mnist is scaled. Table 5.2 lists the UCI/Statlog data sets. In the UCI/Statlog data sets dna, satimage, letter and shuttle provide training and testing sets, while others don't provide training and testing sets so that these data sets will be divided into an 80/20 split as the training set and the testing set.

Table 5.1: The summary of data statistics. The second column shows the number of classes/labels in the data set. n is the number of features. The last two columns are the number of training and testing data respectively.

Data set	# classes	n	# training	# testing
news20	20	62,061	15,935	3,993
mnist	10	780	11,982	1,984
covtype	7	54	464,810	116,202
rcv1	53	47,236	518,571	15,564
sector	105	55,197	6,412	3,207
vehicle	3	100	78,823	19,705

Table 5.2: The summary of UCI/Statlog data statistics. The second column shows the number of classes/labels in the data set. n is the number of features. The last column is the number of data instances.

Data set	# classes	n	# instances
iris	3	4	150
wine	3	13	178
glass	6	9	214
vowel	11	10	528
segment	7	19	2,310
dna	3	180	2,000
satimage	6	36	4,435
letter	26	16	15,000
shuttle	7	9	43,500

All the data sets are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`

5.2 Setting

We compare the following six implementations.

1. LR: Logistic regression solved with trust region Newton method (Lin et al., 2008).

   We use the implemention of LIBLINEAR 2.91 with option `-s 0`.

2. L2-SVM: L2-loss support vector machine solved with coordinate descent method.

   The SVM uses L2-regularization term. The option in LIBLINEAR is `-s 2`.

3. L1-SVM: L1-loss support vector machine solved with coordinate descent method.

It also uses L2-regularization term. The option in LIBLINEAR is `-s 3`.

4. CS: Crammer and Singer method with coordinate descent method. The option in LIBLINEAR is `-s 4`.

5. ME dual: ME dual implementation using original feature vectors.

6. ME dual-2: Similar implementation as ME dual using reduced feature vectors.

For binary solvers such as LR, L2-SVM and L1-SVM, we implement the one-against-one approach which is denoted as "1-1" while one-against-all is denoted as "1-all."

We search for best parameters with cross-validation. C is found in the range of $[2^{-5}, 2^{-4}, \ldots, 2^{3}]$. Too large C leads to much more training time and cannot get much improvement in testing accuracy. And default value of ϵ is choosed as 0.1. For the stopping condition, ME dual uses the relative function difference,

$$\left| \frac{f_k(\boldsymbol{w}) - f_{k-1}(\boldsymbol{w}_{k-1})}{f_{k-1}(\boldsymbol{w}_{k-1})} \right|. \tag{5.1}$$

The procedure stops when (5.1) is less than 10^{-8}.

5.3 Comparison

The result for cross-validation is in Table 5.3. For LR, L2-SVM and L1-SVM, data sets only news20, sector and iris have higher accuraries with the one-against-all approach. LR and L1-SVM perform similarly for vehicle and wine. While the difference of accuracies for wine and dna is little by using L2-SVM with both approaches. One can also see that Crammer and Singer outperforms other models on sector, wine and shuttle, but the accuracy of every classifiers on wine is almost the same.

Table 5.4 gives the result for testing accuracy. LR, L2-SVM and L1-SVM perform better in one-against-all approach on news20, sector and wine. LR also has higher accuracy with one-against-all approach on iris. For Crammer and Singer, it outperforms

other classifiers on sector and shuttle. The difference of accuracies among Crammer and Singer, ME dual and ME dual-2 is less than one percent. Crammer and Singer outperforms the other two ME implementations on sector, iris, vowel, letter and shuttle, but worse on glass and satimage. For segment and dna, the three implementations perform similarly. We also observe that L1-SVM and L2-SVM outperform other classifiers most of the time.

Best C values of each classifier on every data sets are listed in Table 5.5. Data sets news20, wine and dna favor small C values while for vehicle, segment, letter and shuttle large C values are preferred.

For almost all the classifiers take more time in training with one-against-all approach. LR spends more time in training on sector with one-against-one approach. The same situation happens to L1-SVM on covtype. One can see that Crammer and Singer spends least time on mnist, rcv1 and sector. Among three "all-together" implementations, Crammer and Singer takes least time than other two on all data sets. The result of training time is in Table 5.6. For testing time, for all classifiers they spend more time with one-against-one models. The result is in Table 5.7.

Table 5.3: Cross-validation accuracy for each classifiers on data sets. For each data set, the highest accuracy is bold-faced.

	LR		L2-SVM		L1-SVM dual		CS	ME dual	ME dual-2
data	1-all	1-1	1-all	1-1	1-all	1-1			
news20	**83.44**	81.63	83.23	80.73	82.96	80.23	82.24	82.22	81.69
covtype	71.50	72.49	71.26	72.52	71.15	**72.67**	72.43	72.44	72.43
mnist	91.24	93.72	90.95	93.99	91.28	**94.10**	92.37	92.06	92.04
rcv1	92.93	93.27	92.93	93.42	92.81	**93.43**	93.08	93.04	93.02
sector	92.20	90.52	93.22	91.52	93.75	91.56	**94.01**	92.00	92.19
vehicle	80.29	80.40	80.04	80.34	80.18	**80.59**	80.28	80.32	80.32
iris	**88.33**	85.83	**88.33**	86.67	**88.33**	86.67	86.67	85.83	85.83
wine	97.20	97.20	**97.90**	97.20	**97.90**	**97.90**	**97.90**	96.50	95.80
glass	56.98	59.30	58.14	**59.88**	54.65	58.72	54.07	58.72	58.72
vowel	48.23	68.32	49.41	**76.12**	46.81	72.34	58.16	58.63	58.63
segment	91.99	93.99	91.99	**95.13**	92.26	94.91	94.21	93.99	93.99
dna	94.14	93.93	93.79	93.57	93.64	**94.21**	93.71	93.71	93.79
satimage	83.92	87.66	83.18	87.76	82.18	**87.89**	86.44	86.53	86.53
letter	68.01	81.13	66.63	**82.60**	63.34	82.53	75.94	74.98	74.98
shuttle	92.77	96.08	91.96	96.05	93.65	97.10	**97.15**	95.99	95.99

Table 5.4: Testing accuracy for each classifiers on data sets. For each data set, the highest accuracy is bold-faced.

	LR		L2-SVM		L1-SVM dual		CS	ME dual	ME dual-2
data	1-all	1-1	1-all	1-1	1-all	1-1			
news20	84.47	82.79	**84.72**	82.07	84.24	82.07	83.62	83.67	83.35
covtype	71.67	72.62	71.34	72.65	71.19	**72.79**	72.52	72.59	72.59
mnist	91.83	94.41	91.67	**94.47**	92.01	94.45	92.93	92.64	92.64
rcv1	92.10	92.41	92.02	92.47	91.98	**92.55**	92.30	92.33	92.25
sector	92.70	91.61	93.92	92.55	94.08	92.55	**94.36**	92.64	92.67
vehicle	80.51	80.62	80.17	80.35	80.38	**80.70**	80.44	80.44	80.44
iris	**93.33**	90.00	90.00	**93.33**	83.33	**93.33**	90.00	86.67	86.67
wine	**97.14**	94.29	**97.14**	94.29	**97.14**	94.29	94.29	94.29	94.29
glass	66.67	**73.81**	64.29	**73.81**	61.90	66.67	64.29	69.05	69.05
vowel	44.76	74.29	45.71	**78.10**	37.14	77.14	56.19	51.43	51.43
segment	90.04	91.77	89.83	**93.29**	90.69	93.07	91.99	91.13	91.13
dna	94.35	93.68	**94.44**	94.01	93.42	93.42	94.18	94.18	94.01
satimage	81.75	85.55	81.05	85.85	79.90	**86.15**	83.95	84.45	84.45
letter	68.00	81.60	66.34	82.92	62.76	**83.38**	76.78	75.18	75.18
shuttle	92.96	96.23	92.24	96.19	93.83	97.30	**97.39**	96.09	96.09

Table 5.5: The best C for each classifiers on data sets.

	LR		L2-SVM		L1-SVM dual		CS	ME dual	ME dual-2
data	1-all	1-1	1-all	1-1	1-all	1-1			
news20	2^{-1}	2^{-2}	2^{-5}	2^{-5}	2^{-5}	2^{-5}	2^{-5}	2^{-3}	2^{-4}
covtype	2^{3}	2^{3}	2^{-1}	2^{1}	2^{-1}	2^{1}	2^{1}	2^{3}	2^{3}
mnist	2^{0}	2^{0}	2^{-3}	2^{-5}	2^{0}	2^{-5}	2^{-5}	2^{-2}	2^{-2}
rcv1	2^{3}	2^{3}	2^{-1}	2^{-1}	2	2^{0}	2^{-1}	2^{2}	2^{2}
sector	2^{3}	2^{3}	2^{2}	2^{3}	2^{1}	2^{2}	2^{0}	2^{3}	2^{3}
vehicle	2^{3}	2^{3}	2^{2}	2^{0}	2^{3}	2^{3}	2^{3}	2^{3}	2^{3}
iris	2^{-1}	2^{3}	2^{0}	2^{0}	2^{2}	2^{2}	2^{0}	2^{-2}	2^{-2}
wine	2^{0}	2^{-2}	2^{-4}	2^{-5}	2^{-3}	2^{-5}	2^{-3}	2^{-2}	2^{-1}
glass	2^{3}	2^{3}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{1}	2^{1}
vowel	2^{2}	2^{3}	2^{-2}	2^{3}	2^{0}	2^{3}	2^{3}	2^{3}	2^{3}
segment	2^{3}	2^{3}	2^{3}	2^{3}	2^{3}	2^{3}	2^{3}	2^{3}	2^{3}
dna	2^{-2}	2^{-3}	2^{-5}	2^{-5}	2^{-5}	2^{-5}	2^{-5}	2^{-4}	2^{-4}
satimage	2^{3}	2^{2}	2^{0}	2^{-2}	2^{3}	2^{-1}	2^{2}	2^{3}	2^{3}
letter	2^{3}	2^{3}	2^{3}	2^{3}	2^{2}	2^{3}	2^{3}	2^{3}	2^{3}
shuttle	2^{3}	2^{2}	2^{3}	2^{3}	2^{3}	2^{3}	2^{3}	2^{1}	2^{1}

Table 5.6: Training time for each classifiers on data sets.

	LR		L2-SVM		L1-SVM dual		CS	ME dual	ME dual-2
data	1-all	1-1	1-all	1-1	1-all	1-1			
news20	74.15	31.65	29.97	25.09	2.34	0.84	18.73	273.16	268.69
covtype	70.37	44.06	32.18	17.45	58.07	87.78	192.30	1151.14	1288.89
mnist	75.67	42.87	27.89	17.98	60.65	4.44	3.36	76.84	114.07
rcv1	1701.79	906.45	527.28	462.41	140.58	109.51	40.16	1396.86	∞
sector	39.39	73.51	24.81	129.20	5.45	4.18	2.97	58.11	103.96
vehicle	19.44	13.75	14.19	9.68	23.61	13.13	24.28	31.33	41.06
glass	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01
vowel	0.00	0.02	0.00	0.01	0.00	0.01	0.03	0.13	0.13
segment	0.06	0.03	0.05	0.00	0.11	0.05	0.16	1.41	1.61
dna	0.03	0.02	0.02	0.01	0.01	0.00	0.01	0.04	0.05
satimage	0.15	0.10	0.06	0.04	0.34	0.02	0.20	1.47	1.8
letter	1.24	0.93	0.51	0.44	0.86	0.59	1.43	14.84	16.98
shuttle	0.91	0.52	0.31	0.17	0.26	0.24	0.18	1.89	2.06

Table 5.7: Testing time for each classifiers on data sets.

	LR		L2-SVM		L1-SVM dual		CS	ME dual	ME dual-2
data	1-all	1-1	1-all	1-1	1-all	1-1			
news20	0.01	0.14	0.01	0.21	0.03	0.14	0.01	0.02	0.02
covtype	0.04	0.09	0.05	0.07	0.03	0.06	0.06	0.11	0.05
mnist	0.06	0.07	0.02	0.08	0.03	0.11	0.01	0.08	0.00
rcv1	0.11	3.17	0.08	3.19	0.09	3.21	0.13	0.08	0.13
sector	0.17	7.43	0.15	7.40	0.16	7.46	0.12	0.25	0.09
vehicle	0.02	0.02	0.03	0.03	0.00	0.03	0.04	0.03	0.03
glass	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01
vowel	0.00	0.02	0.00	0.01	0.00	0.01	0.03	0.13	0.13
segment	0.06	0.03	0.05	0.00	0.11	0.05	0.16	1.41	1.61
dna	0.03	0.02	0.02	0.01	0.01	0.00	0.01	0.04	0.05
satimage	0.15	0.10	0.06	0.04	0.34	0.02	0.20	1.47	1.8
letter	1.24	0.93	0.51	0.44	0.86	0.59	1.43	14.84	16.98
shuttle	0.91	0.52	0.31	0.17	0.26	0.24	0.18	1.89	2.06

CHAPTER VI

Discussion and Conclusion

From the experiment, one can observe that one-against-one approach can reach higher accuracy for most of the time. If there are k classes in the data set, for one-against-all approach there are k sub-models whereas there are $k(k-1)/2$ sub-models with one-against-one approach. And the size of data to be trained for each sub-problem with one-against-one approach is less than that with one-against-all approach.

The difference of number of sub-models between both approaches will become large when there are large number of classes. Therefore the amount of time we spend and the amount of memory we take will be a crucial concern when we train data of many classes. One-against-one approach leads to higher accuracy because more sub-models for each class are obtained. But it also results in longer testing time because there are more models to be tested for each instance. In total one-against-one approach takes less time in training, but when the number of classes is really large like sector, the situation will become different. However, one should make a trade-off among different factors.

For "all-together" methods, the data sets in this work are not so favorable of these approaches. Overall ME dual and ME dual-2 take a large number of logarithm computations which take more time than regular arithmetic calculations. ME dual-2 uses reduced feature vectors and outputs the solution with rich sparcity, which leads to

less testing time. But it takes much time in looking up the table of indices of reduced feature vectors and results in more training time than ME dual. Although ME is widely used in the NLP applications, in the experiment it takes little advantage. Nevertheless, the feature reduction approach can be furthur studied that we can apply the technique to other models with one-against-all approach.

BIBLIOGRAPHY

A. L. Berger, V. J. Della Pietra, and S. A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwriting digit recognition. In *International Conference on Pattern Recognition*, pages 77–87. IEEE Computer Society Press, 1994.

K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. Coordinate descent method for large-scale L2-loss linear SVM. *Journal of Machine Learning Research*, 9:1369–1398, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cdl2.pdf`.

C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learing Theory*, pages 35–46, 2000.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf`.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

F.-L. Huang, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin. Iterative scaling and coordinate descent methods for maximum entropy. In *Proceedings of the 47th Annual Meeting of the Association of Computational Linguistics (ACL)*, 2009. Short paper.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition, 2008.

S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A sequential dual method for large scale multi-class linear SVMs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/sdm_kdd.pdf`.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing: Algorithms, Architectures and Applications*. Springer-Verlag, 1990.

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf`.

E. Mayoraz and E. Alpaydin. Support vector machines for multi-class classification. In *IWANN (2)*, pages 833–842, 1999. URL `http://citeseer.nj.nec.com/mayoraz98support.html`.

R. Memisevic. Dual optimization of conditional probability models. Technical report, Department of Computer Science, University of Toronto, 2006.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004. ISSN 1533-7928.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In *Proceedings of the Twenty Fourth International Conference on Machine Learning (ICML)*, 2007.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, 1998.

H.-F. Yu, F.-L. Huang, and C.-J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. Technical report, Department of Computer Science, National Taiwan University, March 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/maxent_dual.pdf`.