台 灣 大 學

資 訊 工 程 學 研 究 所

碩 士 論 文

支向機與屬性選擇
Combining SVMs with Various Feature Selection Strategies

研 究 生 ： 陳奕瑋
指 導 教 授 ： 林智仁教授

中 華 民 國 九 十 四 年

Combining SVMs with Various Feature Selection Strategies

by

Yi-Wei Chen

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Master of Science
(Computer Science and Information Engineering)
in National Taiwan University
2005

ABSTRACT

Feature selection is an important issue in many research areas. There are some reasons for selecting important features such as reducing the learning time, improving the accuracy, etc. This thesis investigates the performance of combining support vector machines (SVM) and various feature selection strategies. The first part of the thesis mainly describes the existing feature selection methods and our experience on using those methods to attend a competition. The second part studies more feature selection strategies using the SVM.

摘要

在很多領域裡，屬性選擇 (feature selection) 是一件很重要的事。做屬性選擇有很多好處，例如增快執行速度、提高測試的準確度等等。本論文探討利用支向機 (Support Vector Machine) 在不同的屬性選擇策略下分類的效果。論文的前半部分主要在討論目前已有的屬性選擇方法，以及利用這些方法來參與比賽所得的經驗。後半部份則對更多的方法作深入的研究。

TABLE OF CONTENTS

ABSTRACT . . . . . **ii**

LIST OF FIGURES . . . . . **vi**

LIST OF TABLES . . . . . **vii**

CHAPTER

I. Introduction 1

II. Basic Concepts of SVM 4

2.1 Linear Separating Hyperplane with Maximal Margin 4
2.2 Mapping Data to Higher Dimensional Spaces 6
2.3 The Dual Problem 9
2.4 Kernel and Decision Functions 10
2.5 Multi-class SVM 13
2.5.1 One-against-all Multi-class SVM 13
2.5.2 One-against-one Multi-class SVM 14
2.6 Parameter Selection 15

III. Existing Feature Selection Methods 17

3.1 Feature Ranking 17
3.1.1 Statistical Score 17
3.1.2 Random Shuffle on Features 20
3.1.3 Separating Hyperplane in SVM 21
3.2 Feature Selection 22
3.2.1 Forward/Backward Selection 22
3.2.2 Feature Ranking and Feature Number Estimation . 23
3.3 Feature Scaling 25
3.3.1 Radius-Margin Bound SVM 25
3.3.2 Bayesian SVM 27

IV. Experience on NIPS Competition 29

4.1 Introduction of the Competition 29
4.2 Performance Measures 30
4.2.1 Balanced Error Rate (BER) 30
4.2.2 Area Under Curve (AUC) 31
4.2.3 Fraction of Features 31
4.2.4 Fraction of Probes 31
4.3 Data Sets Information 32
4.3.1 Source of Data Sets 32
4.4 Strategies in Competition 33
4.4.1 No Selection: Direct Use of SVM 33
4.4.2 F-score for Feature Selection: F-score + SVM . . . 33
4.4.3 F-score and Random Forest for Feature Selection: F-score + RF + SVM 35
4.4.4 Random Forest and RM-bound SVM for Feature Selection 36
4.5 Experimental Results 36
4.6 Competition Results 38
4.7 Discussion and Conclusions from the Competition 38

V. Other Feature Ranking Methods by SVM 42

5.1 Normal Vector of the Decision Boundary in Nonlinear SVM . 42
5.2 Change of Decision Value in Nonlinear SVM 43
5.2.1 Instances from Underlying Distribution 44
5.2.2 Instances from Decision Boundary 48
5.3 Random Shuffle on Features using Probability SVM 49
5.3.1 SVM with Probability Output 49
5.3.2 Random Shuffle on Validation Data Features 50

VI. Experiments 52

6.1 Experiment Procedures 52
6.1.1 Feature Selection by Ranking 52
6.1.2 Feature Scaling using RM-bound SVM 54
6.2 Data Sets 54
6.3 Experimental Results 56
6.4 Analysis 57

VII. Discussion and Conclusions 65

BIBLIOGRAPHY **67**

LIST OF FIGURES

Figure

2.1 Separating hyperplane 6

2.2 An example which is not linearly separable 6

2.3 Support vectors (marked as +) are important data from training data 13

3.1 An example of one-dimensional data that cannot be separated by only one boundary point. 18

3.2 A two-dimensional data that is linearly separable. While the separating plane is nearly vertical, the feature of x-axis is more important. 22

4.1 Curves of F-scores against features; features with F-scores below the horizontal line are dropped 41

5.1 A two-dimensional illustration about "pushing" an instance $\mathbf{x}$ to the decision boundary 45

6.1 Comparisons of CV accuracy and testing accuracy against log number of features between feature ranking methods. 60

6.2 Comparisons of CV accuracy and testing accuracy against the log number of features between feature ranking methods. 61

6.3 Comparisons of CV accuracy and testing accuracy against log number of features between feature ranking methods. 62

6.4 Comparisons of CV accuracy and testing accuracy against log number of features between feature ranking methods. 63

6.5 Comparisons of CV accuracy and testing accuracy against log number of features between feature ranking methods. 64

LIST OF TABLES

Table

4.1 Statistics of competition data sets 32

4.2 Comparison of different methods during the development period: BERs of validation sets (in percentage); bold-faced entries correspond to approaches used to generate our final submission 37

4.3 CV BER on the training set (in percentage) 38

4.4 F-score threshold and the number of features selected in the approach F+SVM 38

4.5 NIPS 2003 challenge results of the development stage 39

4.6 NIPS 2003 challenge results of the final stage 40

6.1 Statistics of all data sets. The column "accuracy" is the testing accuracy of original problems. $(\log_2 C, \log_2 \gamma)$ is the optimal parameter obtained by parameter selection. 55

6.2 The testing accuracy obtained by each feature selection/scaling method. The numbers of features selected are also reported in parentheses for those five feature selection methods. Some results are not available due to the limitation of the programs. 58

CHAPTER I

Introduction

Support vector machines (SVM) [3, 12, 36] have been an effective technique for data classification. Not only it has a solid theoretical foundation, practical comparisons have also shown that SVM is competitive with existing methods such as neural networks and decision trees.

SVM uses a separating hyperplane to maximize the distance between two classes of data. For problems that can not be linearly separated in the original feature space, SVMs employ two techniques. First, SVM with a soft margin hyperplane and a penalty function of training errors is introduced. Second, using the kernel technique, the original feature space is non-linearly transformed into a higher dimensional kernel space. In this kernel space it is more possible to find a linear separating hyperplane. More details about basic concepts of SVM are described in Chapter II.

Feature selection is an important issue in many research areas, such as bioinformatics, chemistry [34, 15], and text categorization [27, 18]. There are some reasons for selecting important features. First, reducing the number of features decreases the learning time and storage requirements. Second, removing irrelevant features and keeping informative ones usually improve the accuracy and performance. Third, the subset of selected features may help to discover more knowledge about the data.

According to the relationship between the classifier and the selection strategy, a feature selection method can be categorized into one of the two types: "filter" methods and "wrapper" methods [19]. Filter methods are defined as a preprocessing step that removes irrelevant or unimportant features before classifiers begin to work. On the other hand, "wrapper" methods have close relations to the classifiers and will rely on them to select features.

There are some early studies on feature selection using SVMs. For example, [16] considers the normal vector of the decision boundary as feature weights. [39, 7] minimize generalization bounds with respect to feature weights. In addition, [9] treats the decision values as random variables and maximize the posteriori using a Bayesian framework. There are some other variants of SVMs which are designed to do feature selection [41, 30]. However, in general there is no common or the best way to conduct feature selections with SVMs.

Since more and more people work on feature selection, a competition which was especially designed for this topic was held in the 16th annual conference on Neural Information Processing Systems (NIPS). We took part in this contest and combined different selection strategies with SVMs. The paper [8] is a preliminary study on how to use SVMs to conduct feature selection. Now in this thesis, comparisons between more methods with SVMs are investigated, and more extensive experiments are conducted.

This thesis is organized as follows. Chapter II introduces basic concepts of SVM. Some existing general feature selection methods are discussed in Chapter III. In Chapter IV, we describe the result of attending the NIPS 2003 feature selection competition. Chapter V discusses more strategies with SVMs. Experiments and comparisons are listed in Chapter VI. Finally, discussion and conclusions are in

Chapter VII.

CHAPTER II

Basic Concepts of SVM

We introduce the basic concepts of the SVM in this chapter. Part of this chapter is modified from [24].

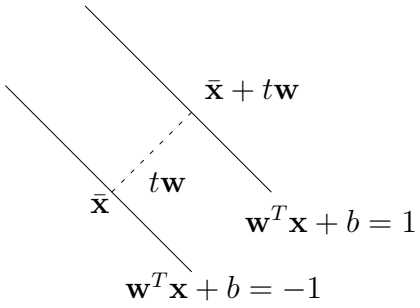
2.1 Linear Separating Hyperplane with Maximal Margin

The original idea of SVM classification is to use a linear separating hyperplane to create a classifier. Given training vectors $\mathbf{x}_i, i = 1, \ldots, l$ of length n, and a vector $\mathbf{y}$ defined as follows

$$y_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ in class 1,} \\ -1 & \text{if } \mathbf{x}_i \text{ in class 2,} \end{cases}$$

the support vector technique tries to find the separating hyperplane with the largest margin between two classes, measured along a line perpendicular to the hyperplane. For example, in Figure 2.1, two classes could be fully separated by a dotted line $\mathbf{w}^T\mathbf{x} + b = 0$. We would like to decide the line with the largest margin. In other words, intuitively we think that the distance between two classes of training data should be as large as possible. That means we find a line with parameters $\mathbf{w}$ and b such that the distance between $\mathbf{w}^T\mathbf{x} + b = \pm 1$ is maximized.

The distance between $\mathbf{w}^T\mathbf{x} + b = 1$ and -1 can be calculated by the following way. Consider a point $\bar{\mathbf{x}}$ on $\mathbf{w}^T\mathbf{x} + b = -1$:



As $\mathbf{w}$ is the "normal vector" of the line $\mathbf{w}^T\mathbf{x} + b = -1$, $\mathbf{w}$ and the line are perpendicular to each other. Starting from $\bar{\mathbf{x}}$ and moving along the direction $\mathbf{w}$, we assume $\bar{\mathbf{x}} + t\mathbf{w}$ touches the line $\mathbf{w}^T\mathbf{x} + b = 1$. Thus,

$$\mathbf{w}^T(\bar{\mathbf{x}} + t\mathbf{w}) + b = 1 \text{ and } \mathbf{w}^T\bar{\mathbf{x}} + b = -1.$$

Then, $t\mathbf{w}^T\mathbf{w} = 2$, so the distance (i.e., the length of $t\mathbf{w}$) is $\|t\mathbf{w}\| = 2\|\mathbf{w}\|/(\mathbf{w}^T\mathbf{w}) = 2/\|\mathbf{w}\|$. Note that $\|\mathbf{w}\| = \sqrt{w_1^2 + \cdots + w_n^2}$. As maximizing $2/\|\mathbf{w}\|$ is equivalent to minimizing $\mathbf{w}^T\mathbf{w}/2$, we have the following problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \\ & i = 1, \ldots, l. \end{aligned} \tag{2.1.1}$$

The constraint $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$ means

$$\begin{aligned} (\mathbf{w}^T\mathbf{x}_i) + b \geq 1 \quad & \text{if } y_i = 1, \\ (\mathbf{w}^T\mathbf{x}_i) + b \leq -1 \quad & \text{if } y_i = -1. \end{aligned}$$

That is, data in class 1 must be on the right-hand side of $\mathbf{w}^T\mathbf{x} + b = 0$ while data in the other class must be on the left-hand side. Note that the reason of maximizing the distance between $\mathbf{w}^T\mathbf{x} + b = \pm 1$ is related to Vapnik's Structural Risk Minimization [36].

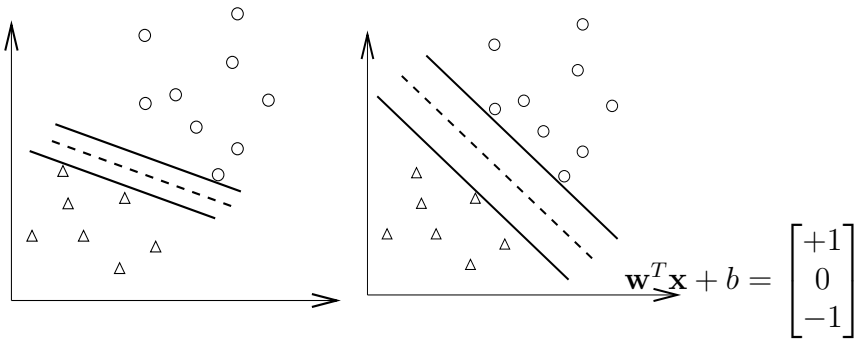


Figure 2.1: Separating hyperplane

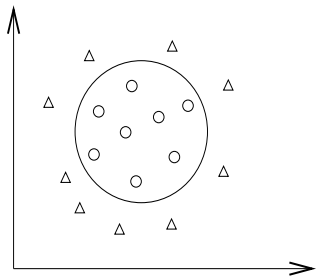


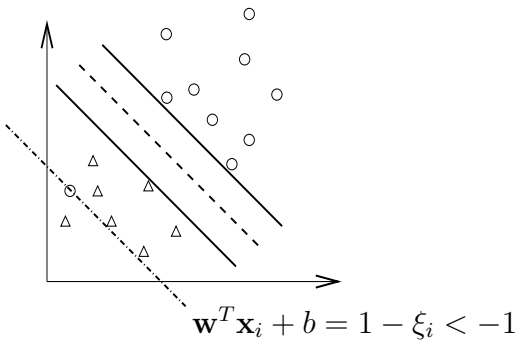
Figure 2.2: An example which is not linearly separable

2.2 Mapping Data to Higher Dimensional Spaces

Practically problems may not be linearly separable and an example is in Figure 2.2. Thus, there is no $(\mathbf{w}, b)$ which satisfies constraints of (2.1.1). In this situation, we say (2.1.1) is "infeasible." In [12] the authors introduced slack variables $\xi_i, i = 1, \ldots, l$ in the constraints:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, l. \end{aligned} \tag{2.2.1}$$

That is, constraints (2.2.1) allow that training data may not be on the correct side of the separating hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$. This situation happens when $\xi_i > 1$ and an example is in the following figure



We have $\xi \geq 0$ as if $\xi < 0$, then $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \geq 1$ and the training data is already on the correct side. The new problem is always feasible since for any $(\mathbf{w}, b)$,

$$\xi_i \equiv \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x} + b)), i = 1, \ldots, l,$$

lead to that $(\mathbf{w}, b, \boldsymbol{\xi})$ is a feasible solution.

Using this setting, we may worry that for linearly separable data, some $\xi_i > 1$ and hence corresponding data are wrongly classified. For the case that most data except some noisy ones are separable by a linear function, we would like $\mathbf{w}^T\mathbf{x} + b = 0$ correctly classifies the majority of points. Thus, in the objective function we add a penalty term $C \sum_{i=1}^{l} \xi_i$, where $C > 0$ is the penalty parameter. To have the objective value as small as possible, most ξ_i should be zero, so the constraint goes back to its original form. Theoretically we can prove that if data are linear separable and C is larger than a certain number, problem (2.2.1) goes back to (2.1.1) and all ξ_i are zero [25].

Unfortunately, such a setting is not enough for practical use. If data are distributed in a highly nonlinear way, employing only a linear function causes many training instances to be on the wrong side of the hyperplane. So underfitting occurs and the decision function does not perform well.

To fit the training data better, we may think of using a nonlinear curve like that in Figure 2.2. The problem is that it is very difficult to model nonlinear curves. All

we are familiar with are eliptic, hyperbolic, or parabolic curves, which are far from enough in practice. Instead of using more sophisticated curves, another approach is to map data into a higher dimensional space. For example, suppose the height and weight of some people are available and medical experts have identified that some of them are over-weighted or underweighted. We may consider two other attributes

$$\text{height-weight, weight}/(\text{height}^2).$$

Such features may provide more information for separating underweighted/overweighted people. Each new data instance is now in a four-dimensional space, so if the two new features are good, it should be easier to have a seperating hyperplane so that most ξ_i are zero.

Thus SVM non-linearly transforms the original input space into a higher dimensional feature space. More precisely, the training data $\mathbf{x}$ is mapped into a (possibly infinite) vector in a higher dimensional space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots].$$

In this higher dimensional space, it is more possible that data can be linearly separated. An example by mapping $\mathbf{x}$ from R^3 to R^{10} is as follows:

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3).$$

An extreme example is to map a data instance $x \in R^1$ to an infinite dimensional space:

$$\phi(x) = \left[1, \frac{x}{1!}, \frac{x^2}{2!}, \frac{x^3}{3!}, \ldots\right]^T.$$

We then try to find a linear separating plane in a higher dimensional space so

(2.2.1) becomes

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \\ & \xi_i \geq 0,\ i=1,\ldots,l. \end{aligned} \tag{2.2.2}$$

2.3 The Dual Problem

The remaining problem is how to effectively solve (2.2.2). Especially after data are mapped into a higher dimensional space, the number of variables $(\mathbf{w}, b)$ becomes very large or even infinite. We handle this difficulty by solving the dual problem of (2.2.2):

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) - \sum_{i=1}^{l}\alpha_i \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \qquad i=1,\ldots,l, \\ & \sum_{i=1}^{l} y_i\alpha_i = 0. \end{aligned} \tag{2.3.1}$$

This new problem of course is related to the original problem (2.2.2), and we hope that it can be more easily solved. Sometimes we write (2.3.1) in a matrix form for convenience:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \qquad i=1,\ldots,l, \\ & \mathbf{y}^T\boldsymbol{\alpha} = 0. \end{aligned} \tag{2.3.2}$$

In (2.3.2), $\mathbf{e}$ is the vector of all ones, C is the upper bound, Q is an l by l positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, and $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is the kernel, which will be addressed in Chapter 2.4.

If (2.3.2) is called the "dual" problem of (2.2.2), we refer to (2.2.2) as the "primal" problem. Suppose $(\bar{\mathbf{w}}, \bar{b}, \bar{\xi})$ and $\bar{\boldsymbol{\alpha}}$ are optimal solutions of the primal and dual problems, respectively, the following two properties hold:

$$\bar{w} = \sum_{i=1}^{l} \bar{\alpha}_i y_i \phi(\mathbf{x}_i), \tag{2.3.3}$$

$$\frac{1}{2}\bar{\mathbf{w}}^T\bar{\mathbf{w}} + C\sum_{i=1}^{l}\bar{\xi}_i = \mathbf{e}^T\bar{\boldsymbol{\alpha}} - \frac{1}{2}\bar{\boldsymbol{\alpha}}^T Q\bar{\boldsymbol{\alpha}}. \tag{2.3.4}$$

In other words, if the dual problem is solved with a solution $\bar{\boldsymbol{\alpha}}$, the optimal primal solution $\bar{\mathbf{w}}$ is easily obtained from (2.3.3). If an optimal $\bar{b}$ can also be easily found, the decision function is hence determined.

Thus, the crucial point is whether the dual is easier to be solved than the primal. The number of variables in the dual, which is the size of the training set: l, is a fixed number. In contrast, the number of variables in the primal problem varies depending on how data are mapped to a higher dimensional space. Therefore, moving from the primal to the dual means that we solve a finite-dimensional optimization problem instead of a possibly infinite-dimensional problem.

The remaining issue of using the dual problem is about the inner product $Q_{ij} = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. If $\phi(\mathbf{x})$ is an infinite-long vector, there is no way to fully write it down and then calculate the inner product. Thus, even though the dual possesses the advantage of having a finite number of variables, we even could not write the problem down before solving it. This is resolved by using special mapping functions ϕ so that $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is in a closed form. Details are in the next section.

2.4 Kernel and Decision Functions

Consider a special $\phi(\mathbf{x})$ mentioned earlier (assume $\mathbf{x} \in R^3$):

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3).$$

In this case it is easy to see that $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = (1+\mathbf{x}_i^T\mathbf{x}_j)^2$, which is easier to be calculated then doing a direct inner product. To be more precise, a direct calculation of $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ takes 10 multiplications and 9 additions, but using $(1+\mathbf{x}_i^T\mathbf{x}_j)^2$, only four multiplications and three additions are needed. Therefore, if a special $\phi(\mathbf{x})$ is considered, even though it is a long vector, $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ may still be easily available.

We call such inner products the "kernel function." Some popular kernels are, for example,

1. $e^{-\gamma||\mathbf{x}_i-\mathbf{x}_j||^2}$ (Gaussian kernel or Radial bassis function (RBF) kernel),

2. $(\mathbf{x}_i^T\mathbf{x}_j/\gamma+\delta)^d$ (polynomial kernel),

where γ, d, and δ are kernel parameters. The following calculation shows that the Gaussian (RBF) kernel indeed is an inner product of two vectors in an infinite dimensional space. Assume $x \in R^1$ and $\gamma > 0$.

$$
\begin{aligned}
& e^{-\gamma||\mathbf{x}_i-\mathbf{x}_j||^2} = e^{-\gamma(x_i-x_j)^2} \\
= \;& e^{-\gamma x_i^2+2\gamma x_i x_j-\gamma x_j^2} \\
= \;& e^{-\gamma x_i^2-\gamma x_j^2}\big(1+\frac{2\gamma x_i x_j}{1!}+\frac{(2\gamma x_i x_j)^2}{2!}+\frac{(2\gamma x_i x_j)^3}{3!}+\cdots\big) \\
= \;& e^{-\gamma x_i^2-\gamma x_j^2}\big(1\cdot 1+\sqrt{\frac{2\gamma}{1!}}x_i\cdot\sqrt{\frac{2\gamma}{1!}}x_j+\sqrt{\frac{(2\gamma)^2}{2!}}x_i^2\cdot\sqrt{\frac{(2\gamma)^2}{2!}}x_j^2 \\
& \qquad +\sqrt{\frac{(2\gamma)^3}{3!}}x_i^3\cdot\sqrt{\frac{(2\gamma)^3}{3!}}x_j^3+\cdots\big) \\
= \;& \phi(x_i)^T\phi(x_j),
\end{aligned}
$$

where

$$
\phi(x) = e^{-\gamma x^2}\left[1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \sqrt{\frac{(2\gamma)^3}{3!}}x^3, \cdots\right]^T.
$$

Note that $\gamma > 0$ is used for the existance of terms such as $\sqrt{\frac{2\gamma}{1!}}$, $\sqrt{\frac{(2\gamma)^3}{3!}}$, etc.

After (2.3.2) is solved with a solution $\boldsymbol{\alpha}$, the vector for which $\alpha_i > 0$ are called *support vectors*. Then, a decision function is written as

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\phi(\mathbf{x}) + b) = \text{sign}\left(\sum_{i=1}^{l} y_i\alpha_i\phi(\mathbf{x}_i)^T\phi(\mathbf{x}) + b\right). \tag{2.4.1}$$

In other words, for a test vector $\mathbf{x}$, if $\sum_{i=1}^{l} y_i\alpha_i\phi(\mathbf{x}_i)^T\phi(\mathbf{x}) + b > 0$, we classify it to be in the class 1. Otherwise, we think it is in the second class. We can see that only support vectors will affect results in the prediction stage. In general, the number of support vectors is not large. Therefore we can say SVM is used to find important data (support vectors) from training data.

We use Figure 2.3 as an illustration. Two classes of training data are not linearly separable. Using the RBF kernel, we obtain a hyperplane $\mathbf{w}^T\phi(\mathbf{x}) + b = 0$. In the original space, it is indeed a nonlinear curve

$$\sum_{i=1}^{l} y_i\alpha_i\phi(\mathbf{x}_i)^T\phi(\mathbf{x}) + b = 0. \tag{2.4.2}$$

In the figure, all points in red color are support vectors and they are selected from both classes of training data. Clearly support vectors are close to the nonlinear curve (2.4.2) are more important points.

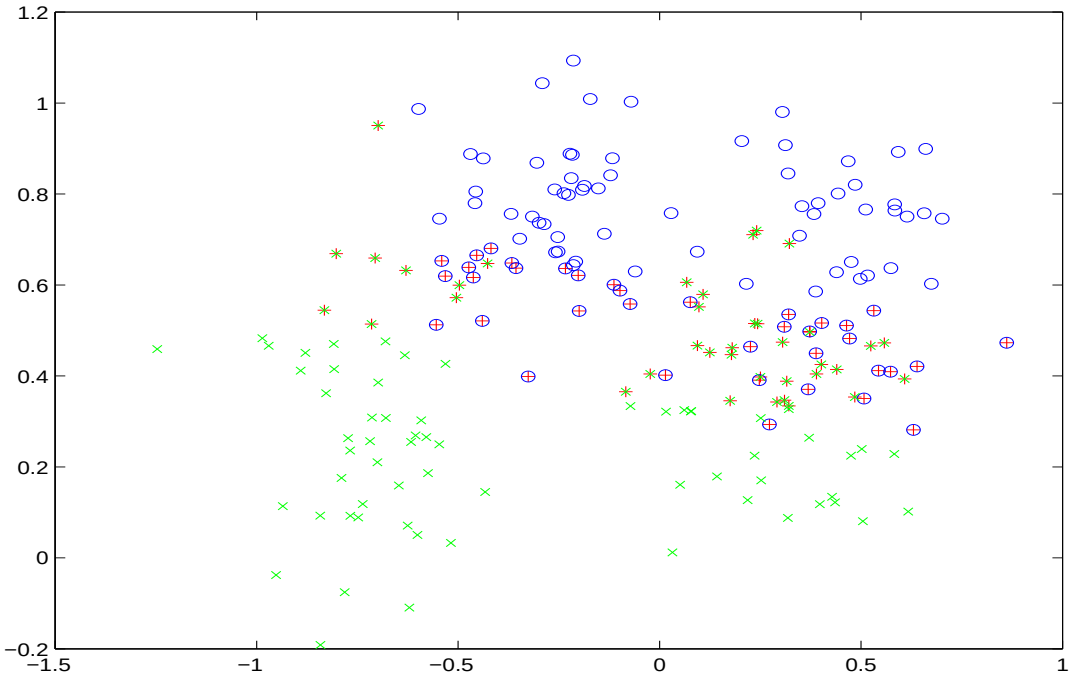


Figure 2.3: Support vectors (marked as +) are important data from training data

2.5 Multi-class SVM

The discussion so far assumes that data are in only two classes. Many practical applications involve more classes of data. For example, hand-written digit recognition considers data in 10 classes: digits 0 to 9. There are many ways to extend SVM for such cases. Here, we discuss two simple methods.

2.5.1 One-against-all Multi-class SVM

This commonly mis-named method should be called "one-against-the-rest." It constructs binary SVM models so that each one is trained with one class as positive and the rest as negative. We illustrate this method by a simple situation of four classes. The four two-class SVMs are

$y_i = 1$	$y_i = -1$	Decision function
class 1	classes 2,3,4	$f^1(\mathbf{x}) = (\mathbf{w}^1)^T\mathbf{x} + b^1$
class 2	classes 1,3,4	$f^2(\mathbf{x}) = (\mathbf{w}^2)^T\mathbf{x} + b^2$
class 3	classes 1,2,4	$f^3(\mathbf{x}) = (\mathbf{w}^3)^T\mathbf{x} + b^3$
class 4	classes 1,2,3	$f^4(\mathbf{x}) = (\mathbf{w}^4)^T\mathbf{x} + b^4$

For any test data $\mathbf{x}$, if it is in the ith class, we would expect that

$$f^i(\mathbf{x}) \geq 1 \text{ and } f^j(\mathbf{x}) \leq -1, \text{ if } j \neq i.$$

This "expectation" directly follows from our setting of training the four two-class problems and from the assumption that data are correctly separated. Therefore, $f^i(\mathbf{x})$ has the largest values among $f^1(\mathbf{x}), \ldots, f^4(\mathbf{x})$ and hence the decision rule is

$$\text{Predicted class } = \arg\max_{i=1,\ldots,4} f^i(\mathbf{x}).$$

2.5.2 One-against-one Multi-class SVM

This method also constructs several two-class SVMs but each one is by training data from only two different classes. Thus, this method is sometimes called a "pair-wise" approach. For the same example of four classes, six two-class problems are constructed:

$y_i = 1$	$y_i = -1$	Decision function
class 1	class 2	$f^{12}(\mathbf{x}) = (\mathbf{w}^{12})^T\mathbf{x} + b^{12}$
class 1	class 3	$f^{13}(\mathbf{x}) = (\mathbf{w}^{13})^T\mathbf{x} + b^{13}$
class 1	class 4	$f^{14}(\mathbf{x}) = (\mathbf{w}^{14})^T\mathbf{x} + b^{14}$
class 2	class 3	$f^{23}(\mathbf{x}) = (\mathbf{w}^{23})^T\mathbf{x} + b^{23}$
class 2	class 4	$f^{24}(\mathbf{x}) = (\mathbf{w}^{24})^T\mathbf{x} + b^{24}$
class 3	class 4	$f^{34}(\mathbf{x}) = (\mathbf{w}^{34})^T\mathbf{x} + b^{34}$

For any test data $\mathbf{x}$, we put it into the six functions. If the problem of classes i and j indicates the data $\mathbf{x}$ should be in i, the class i gets one vote. For example, assume

Classes		winner
1	2	1
1	3	1
1	4	1
2	3	2
2	4	4
3	4	3

Then, we have

class	1	2	3	4
# votes	3	1	1	1

Thus, $\mathbf{x}$ is predicted to be in the first class.

For a data set with m different classes, this method constructs $m(m-1)/2$ two-class SVMs. We may worry that sometimes more than one class obtains the highest number of votes. Practically this situation does not happen so often and there are some further strategies to handle it.

2.6 Parameter Selection

The performance of the SVM is sensitive to its kernel and parameters [17], so they must be chosen carefully when constructing a predicting model.

There are some commonly used kernels:

1. linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.

2. polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = \left(\gamma \mathbf{x}_i^T \mathbf{x}_j + r\right)^d, \gamma > 0$.

3. radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \gamma > 0$.

[17] recommends using the RBF kernel in the SVM. First, it nonlinearly maps instances into a higher dimensional space, and it can handle data which is not linearly separable. Second, [20] shows that the linear kernel is a special case of the RBF kernel. In addition, the sigmoid kernel behaves like the RBF kernel for certain parameters [26]. Third, the RBF kernel suffers less from numerical difficulties.

With the RBF kernel, there are two parameters to be determined in an SVM model now: the penalty parameter C in (2.2.1), and γ in the RBF kernel. To get good generalization ability, a cross validation (CV) process is conducted to decide parameters. The procedure to construct a model for predicting is therefore like the following:

Algorithm 1 SVM_Parameter_Selection

1. *Consider a list of* (C, γ), $C, \gamma > 0$. *For each hyperparameter pair* (C, γ) *in the search space, conduct five-fold cross validation on the training set.*

2. *Choose the parameter* (C, γ) *that leads to the lowest cross validation error rate.*

3. *Use the best parameter to create a model as the predictor.*

CHAPTER III

Existing Feature Selection Methods

In this chapter we describe several common strategies for feature selection. Some of them give only a ranking of features; some are able to directly select proper features for classification. There are methods that can only be applied to SVMs, but most general feature selection techniques are independent of the classifier used.

3.1 Feature Ranking

Among feature selection techniques, some could give an importance order of all the features. In the following we describe some methods of this type.

3.1.1 Statistical Score

Several statistical criteria such as correlation coefficient, Fisher's criterion [13, 28] could give feature rankings. They are independent of the choice of the classifier and are more like preprocessing steps. The advantage of such criteria is that they are simple and effective. However, effects of classifiers behind them are often ignored, and this is the main disadvantage of this type of approaches [21]. For example, consider a one dimensional two-class data set in Figure 3.1. Fisher's criterion or correlation coefficient may think that this feature is not discriminative. If the classifier used is

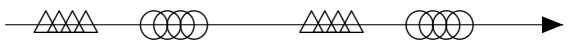


Figure 3.1: An example of one-dimensional data that cannot be separated by only one boundary point.

a linear one, this is the case indeed. However, a decision tree will perform well on this data set.

Here we concentrate on using Fisher's criterion in feature ranking. The Fisher's criterion of a single feature is defined as the following. Given a data set X with two classes, denote instances in class 1 as X^1, and those in class 2 as X^2. Assume $\bar{x}_j^k$ is the average of the jth feature in X^k. The Fisher score (F-score for short) of the jth feature is:

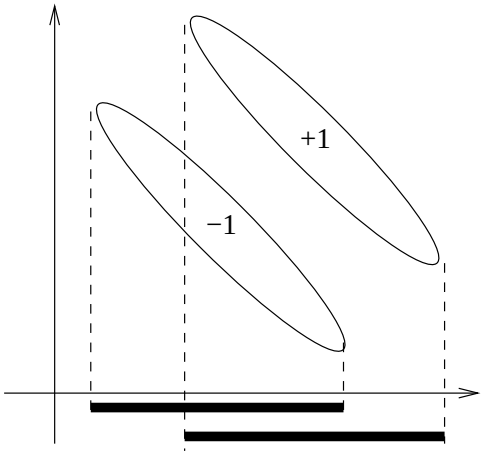
$$F(j) \equiv \frac{\left(\bar{x}_j^1 - \bar{x}_j^2\right)^2}{(s_j^1)^2 + (s_j^2)^2}, \tag{3.1.1}$$

where

$$(s_j^k)^2 = \sum_{\mathbf{x} \in X^k} \left(x_j - \bar{x}_j^k\right)^2.$$

The numerator indicates the discrimination between two classes, and the denominator indicates the scatter within each class. The larger the F-score is, the more likely this feature is more discriminative. Therefore, this score can be a criterion for feature selection.

A disadvantage of using F-score to obtain feature ranking is that it does not reveal mutual information among features. Consider one simple example in the following figure:



Both features of this data have low F-scores as in (3.1.1) the denominator (the sum of variances of the positive and negative sets) is much larger than the numerator. Despite this disadvantage, F-score is simple and generally quite effective.

For data with more than two classes, F-score can be extended as the following. Given a data set X with m classes, denote the set of instances in class k as X^k, and $|X^k| = l_k$, $k = 1, \ldots, m$. Assume $\bar{x}_j^k$ and $\bar{x}_j$ are the average of the jth feature in X^k and X, respectively. The Fisher score of the jth feature of this data set is defined as:

$$\hat{F}(j) \equiv \frac{S_B(j)}{S_W(j)}, \tag{3.1.2}$$

where

$$S_B(j) = \sum_{k=1}^{m} l_k \left(\bar{x}_j^k - \bar{x}_j\right)^2, \tag{3.1.3}$$

$$S_W(j) = \sum_{k=1}^{m} \sum_{\mathbf{x} \in X^k} \left(x_j - \bar{x}_j^k\right)^2. \tag{3.1.4}$$

Note that (3.1.1) is equivalent to (3.1.2) when $k = 2$: Since

$$\begin{aligned} S_W(j) &= \sum_{k=1}^{2} \sum_{\mathbf{x} \in X^k} \left(x_j - \bar{x}_j^k\right)^2 \\ &= (s_j^1)^2 + (s_j^2)^2, \end{aligned}$$

and

$$\begin{aligned} S_B(j) &= \sum_{k=1}^{2} l_k \left(\bar{x}_j^k - \bar{x}_j\right)^2 \\ &= l_1 \left(\bar{x}_j^1 - \frac{l_1\bar{x}_j^1 + l_2\bar{x}_j^2}{l_1+l_2}\right)^2 + l_2 \left(\bar{x}_j^2 - \frac{l_1\bar{x}_j^1 + l_2\bar{x}_j^2}{l_1+l_2}\right)^2 \\ &= l_1 \left(\frac{l_2}{l_1+l_2}\right)^2 \left(\bar{x}_j^1 - \bar{x}_j^2\right)^2 + l_2 \left(\frac{l_1}{l_1+l_2}\right)^2 \left(\bar{x}_j^2 - \bar{x}_j^1\right)^2 \\ &= \frac{l_1 l_2}{l_1+l_2} \left(\bar{x}_j^1 - \bar{x}_j^2\right)^2, \end{aligned}$$

for a two-class data set

$$\hat{F}(j) = \frac{l_1 l_2}{l_1+l_2} F(j), \tag{3.1.5}$$

and they both give the same feature ranking.

3.1.2 Random Shuffle on Features

Feature ranking obtained from statistical scores do not take classifiers' characteristics into account. Here we introduce a method which uses the ability of the underlying classifiers. It can be generally applied to all kinds of classifiers as well.

To introduce this idea, take random forest as an example. Random forest (RF) is a classification method, but it also provides feature importance [4]. A forest contains many decision trees. Each decision tree is constructed by instances ramdomly sampled with replacement. Therefore about one-third of training instances are left out. These are called out-of-bag (oob) data. This oob data can be used to estimate classification error.

The basic idea about giving feature importance is simple. It is achieved by using oob data. For every decision tree in the forest, put down its oob data and count the number of correct predictions. Then randomly permute the values of the jth feature among oob instances, and put these instances down the tree again. Substract the number of correct predictions in permuted oob data from that in unpermuted oob

data. The average of this number over all the trees in the forest is the raw importance score for the jth feature.

From the description above, it is easy to see that this idea can be extended to any other learning machines generally. First, determine the sort of hold-out technique to estimate a model's performance, such as oob data in random forest, or cross-validation for general machines. Second, build a model by the learning algorithm and estimate its performance by the above technique. Finally, random shuffle the specific feature in hold-out data, and then estimate the performance again using the same model in the previous step. The difference of two estimations can be seen as an influence of that feature. If a feature plays an important role in classifying, shuffling this feature will make it irrelevant to the labels and the performance will decrease.

Note that random shuffling or even removing features in the whole data is also a possible way to know the influence of features. However, most learning algorithms spend very large amount of time in training or even in hyper-parameter selection, while the predicting is much faster. Therefore it is not practical to shuffle features of the whole data if the size of data set is not small.

3.1.3 Separating Hyperplane in SVM

For a linear SVM, the normal direction of the separating hyperplane may give the importance of features [15, 16]. A simple illustration by a two-dimensional binary-class problem is in Figure 3.2. It is obvious that the feature of the horizontal axis has good separating ability, while the feature of the vertival axis is more like a noise. Furthermore, the projected length of $\mathbf{w}$, the normal vector of the separating hyperplane, on the horizontal axis is longer than that on the vertical axis. Therefore, one may conjecture that if the absolute value of a component in $\mathbf{w}$ is larger, its

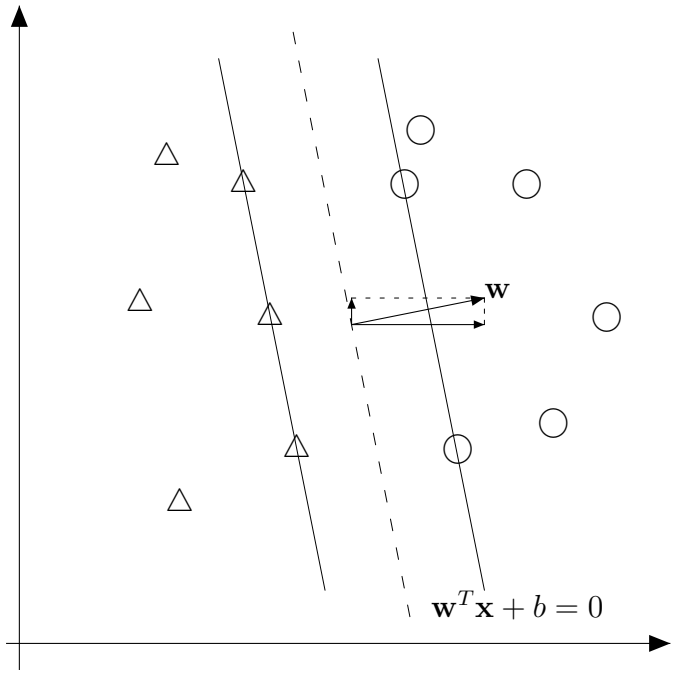


Figure 3.2: A two-dimensional data that is linearly separable. While the separating plane is nearly vertical, the feature of x-axis is more important.

corresponding feature is more important.

However, for non-linear cases the decision boundary is not a hyperplane anymore. That is, the influence of a feature on the decision boundary varies and is dependent on where it is in the feature space. Some solutions to the nonlinear case will be described Chapter V.

3.2 Feature Selection

In this section, we describe several techniques which select important features for learning. In some situations, the rank of features must be available first.

3.2.1 Forward/Backward Selection

When the number of features is small, a greedy strategy can be considered. Suppose there are n features in a data set. First we consider feature subsets that contain

only one feature. For $i = 1, \ldots, n$, we select the ith feature only and conduct the cross validation on the selected data. Therefore, the CV accuracy for n different feature subsets are obtained. The feature with the highest CV accuracy is thought as the most important one. Now, if feature subsets with two features are considered, instead of checking all $n(n-1)/2$ possible subsets, we only look at the subsets with the most important feature which is selected previously. Then, only $n-1$ feature subsets are of interest. For each subset, we check the CV accuracy of the selected data again and choose the best subset. This selection continues until the performance is not increased, or all the features are selected. Such a procedure is called forward (or incremental) selection, since one more feature is included every time. On the contrary, the procedure that starts from all the features and greedily remove one feature at a time is called backward elimination [14].

However, this procedure is time consuming if the size of features is large. A faster procedure can be applied if the rank of features is given. Instead of checking all the remaining features during each step of forward selection, the top-ranked feature is chosen. Therefore, the execution time of the whole procedure is proportional to the number of features.

Note that greedily selecting the best feature in each step does not guarantee to find the globally optimal set of features.

3.2.2 Feature Ranking and Feature Number Estimation

If a feature ranking is available, one does not need to conduct a forward or backward selection, whose time complexity, as indicated earlier, is linear to the number of features. We can decide the number of features by recursively removing half of the remaining features (or doubling the number of features). The feature set with the

highest CV accuracy is then selected. The number of CV steps is only the logarithm of the number of features. An example of using this procedure is in [34].

We summarize this procedure in the following.

Algorithm 2 Obtain_Feature_Size

1. *Partition the whole training data for k-fold cross-validation.*
2. *For each CV training set:*
   (a) *Obtain the feature ranking of this CV training set.*
   (b) *Let the working features be the whole features.*
   (c) *Under the working features, construct a model by the CV training set to predict the CV testing set.*
   (d) *Update working features by removing half features which are less important. Go to step 2(c) or stop if the number of working features is small.*
3. *Obtain the CV accuracy for different size of working features. Assign p to be the size with the maximal CV accuracy.*
4. *Calculate the feature ranking on the whole training data. Select the most important p features. These features are then used for the final model construction and prediction.*

Though computationally more efficient, this procedure checks fewer feature subsets. For example, it considers only at most ten sets in a data set with 1000 features.

Note that the rank of features for each CV training set is obtained in the beginning at step 2(a) and is not updated throughout iterations. For the same data under

different feature subsets, some feature ranking methods may obtain different rankings. Therefore, one may update the ranking after step 2(d), where half features are removed. An example of using this implementation is Recursive Feature Elimination (RFE) in [15]. However, [34] points out that such an aggressive update of feature rankings may more easily lead to overfitting.

3.3 Feature Scaling

Instead of selecting a subset of features, one may give weights on features. Hence important features play a more vital role in training and prediction. Such techniques are often associated with particular classification methods. Here we describe two techniques using SVM.

3.3.1 Radius-Margin Bound SVM

[7] considers the RBF kernel with feature-wise scaling factors:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\sum_{j=1}^{n} \gamma_j (x_j - x'_j)^2\right) . \tag{3.3.1}$$

By minimizing an estimation of generalization errors which is a function of $\gamma_1, \ldots, \gamma_n$, the feature importance is obtained.

Leave-one-out (loo) error is a common estimation of generalization errors but is not a smooth function of $\gamma_1, \ldots, \gamma_n$. Hence it is hard to directly minimize the loo error. For a hard-margin SVM, its loo error is bounded by a smoother function [36, 37]:

$$\text{loo} \le 4\|\mathbf{w}\|^2 R^2, \tag{3.3.2}$$

where $\|\mathbf{w}\|$ is the inverse of the margin of this SVM, and R is the radius of the smallest sphere containing all instances in the kernel space. Therefore we refer to this upper bound as the "radius margin (RM) bound." Then instead of minimizing

the loo error, we can minimize the radius margin bound to obtain certain $\gamma_1, \ldots, \gamma_n$. [36] states that R^2 is the objective value of the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & 1 - \boldsymbol{\beta}^T K \boldsymbol{\beta} \\ \text{subject to} \quad & 0 \le \beta_i, i = 1, \ldots, l, \\ & \mathbf{e}^T \boldsymbol{\beta} = 1, \end{aligned} \tag{3.3.3}$$

where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

However, hard-margin SVMs are not suitable for practical use as in general data are not linearly separable. Among soft-margin SVMs, an L2-SVM can be transformed to an equivalent hard-margin form. The formula of L2-SVMs is similar to (2.2.1) except the penalty term:

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \xi_i^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge 1 - \xi_i, \\ & \xi_i \ge 0, \ i = 1, \ldots, l. \end{aligned} \tag{3.3.4}$$

(3.3.4) can be reduced to a hard-margin form by using

$$\tilde{\mathbf{w}} \equiv \begin{bmatrix} \mathbf{w} \\ \sqrt{C}\boldsymbol{\xi} \end{bmatrix} \text{ and the } i\text{th training data as } \begin{bmatrix} \phi(\mathbf{x}_i) \\ y_i \mathbf{e}_i / \sqrt{C} \end{bmatrix}, \tag{3.3.5}$$

where $\mathbf{e}_i$ is a zero vector of length l except the ith component is one. The kernel function becomes

$$\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) \equiv K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C}, \tag{3.3.6}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Therefore the RM bound for L2-SVMs is

$$4\|\tilde{\mathbf{w}}\|^2 \tilde{R}^2. \tag{3.3.7}$$

$\|\tilde{\mathbf{w}}\|$ and $\tilde{R}$ are obtained by optimization problems. According to [37, 10], these two functions are differentiable to C and $\gamma_1, \ldots, \gamma_n$. Thus, gradient-based methods

can be applied to minimize this bound. Using these parameters, an SVM model can be built for future prediction. We call this machine an RM-bound SVM.

3.3.2 Bayesian SVM

[9] proposes a Bayesian technique for support vector classification. For a binary classfication problem with instances $\mathbf{x}_i$, $i = 1, \ldots, l$ and labels $y_i \in \{-1, 1\}$, they assume that the decision function f is the realization of random variables indexed by the training vectors $\mathbf{x}_i$ in a stationary zero-mean Gaussian process. In this case, the covariance between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ can be defined as:

$$\operatorname{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) = \kappa_0 \exp\left(-\frac{1}{2}\kappa\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) + \kappa_b. \tag{3.3.8}$$

Thus, the prior probability of random variables $\{f(\mathbf{x}_i)\}$ can be written as:

$$\mathcal{P}(\mathbf{f}|\theta) = \frac{1}{Z_{\mathbf{f}}} \exp\left(-\frac{1}{2}\mathbf{f}^T\Sigma^{-1}\mathbf{f}\right), \tag{3.3.9}$$

where $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_l))^T$, $Z_{\mathbf{f}} = (2\pi)^{n/2}|\Sigma|^{1/2}$, Σ is the covariance matrix defined in (3.3.8), and θ is the hyperparameter κ_0, κ, and κ_b. Furthermore, they define the "trigonometric likelihood" function:

$$\mathcal{P}\left(y_i|f(\mathbf{x}_i)\right) = \begin{cases} 0 & y_i f(\mathbf{x}_i) \in (-\infty, -1], \\ \cos^2\left(\frac{\pi}{4}(1 - y_i f(\mathbf{x}_i))\right) & y_i f(\mathbf{x}_i) \in (-1, +1), \\ 1 & y_i f(\mathbf{x}_i) \in [+1, +\infty), \end{cases} \tag{3.3.10}$$

where y_i is the label of the instance $\mathbf{x}_i$. Therefore, by the Bayesian rule,

$$\mathcal{P}(\mathbf{f}|\mathcal{D}, \theta) = \frac{1}{Z_S} \exp\left(-S(\mathbf{f})\right), \tag{3.3.11}$$

where

$$S(\mathbf{f}) = \frac{1}{2}\mathbf{f}^T\Sigma^{-1}\mathbf{f} + \sum_{i=1}^{l} \delta\left(y_i f(\mathbf{x}_i)\right). \tag{3.3.12}$$

Here δ is the "trigonometric loss function" derived from the trigonometric likelihood function (3.3.10):

$$\begin{aligned}
&\delta\left(y_i f(\mathbf{x}_i)\right) = -\log\left(\mathcal{P}\left(y_i | f(\mathbf{x}_i)\right)\right) \\
&= \begin{cases} +\infty & y_i f(\mathbf{x}_i) \in (-\infty, -1], \\ 2\log\sec\left(\frac{\pi}{4}(1 - y_i f(\mathbf{x}_i))\right) & y_i f(\mathbf{x}_i) \in (-1, +1), \\ 0 & y_i f(\mathbf{x}_i) \in [+1, +\infty). \end{cases}
\end{aligned} \tag{3.3.13}$$

Therefore, the posteriori is maximized and the following optimization problem has to be solved:

$$\min_{\mathbf{f}} S(\mathbf{f}) = \frac{1}{2}\mathbf{f}^T \Sigma^{-1} \mathbf{f} + \sum_{i=1}^{l} \delta\left(y_i f(\mathbf{x}_i)\right). \tag{3.3.14}$$

For feature selection, they also propose a feature-wise scaling kernel simliar to (3.3.1):

$$\mathrm{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \kappa_0 \exp\left(-\frac{1}{2}\sum_{j=1}^{n} \kappa_j (x_j - x'_j)^2\right) + \kappa_b. \tag{3.3.15}$$

Thus, to treat κ_j as variables in (3.3.14), the scaling factor of each feature can also be calculated.

For RM-bound SVM and Bayesian SVM, both of them try to maximize estimations of generalization ability. We will see in the next chapter that they have similar performance in some experiments.

CHAPTER IV

Experience on NIPS Competition

This chapter discusses our experience of attending a feature selection competition. It shows how to apply some feature selection techniques described in the previous chapter on competition data sets.

4.1 Introduction of the Competition

NIPS 2003 Feature Selection Challenge* was a competition held in the 16th annual conference on Neural Information Processing Systems (NIPS). There were five two-class data sets designed for the competition. and each data set was splitted into a training, a validation, and a testing set. The aim was to have the best performance on testing sets.

The competition was composed of two stages: only labels of training sets were available in the "development stage." Competitors submitted their predicted labels of the validation and the testing sets. An on-line judge reported the performance of the validation sets, such as the balanced error rate. However, the performance on the testing sets were not available, so competitors cannot know which strategy was better for the testing sets directly from submissions. The development stage ended

*http://clopinet.com/isabelle/Projects/NIPS2003/

on December 1^{st}, and then the "final stage" began. Labels of validation sets were given additionally in the final stage, so competitors could use this information to predict labels of the testing sets. The performance of the testing sets were still kept in secret. Only five submissions were allowed in the final stage and the deadline was on December 8^{th}.

In the contest, we used combinations of SVMs with various feature selection strategies. Some of them were "filters": general feature selection methods independent of SVM. That is, these methods selected important features first and then an SVM was applied for classification. On the other hand, some were wrapper-type methods: modifications of SVM which chose important features as well as conducted training/testing. Overall we ranked third as a group and were the winner of one data set.

4.2 Performance Measures

The following introduces performance measures which are considered in this challenge.

4.2.1 Balanced Error Rate (BER)

In NIPS 2003 Feature Selection Challenge, the main judging criterion was the balanced error rate (BER).

$$\text{BER} \equiv \frac{1}{2}\left(\frac{\#\text{ positive instances predicted wrong}}{\#\text{ positive instances}} + \frac{\#\text{ negative instances predicted wrong}}{\#\text{ negative instances}}\right). \tag{4.2.1}$$

For example, assume that a testing data set contains 90 positive and 10 negative instances. If all instances are predicted as positive, then the BER is 50% since the first term of (4.2.1) is 0/90 but the second is 10/10. In short, if there are fewer

negative examples, the errors on negative examples will count more.

4.2.2 Area Under Curve (AUC)

Besides predicted labels, competitors could to submit confidence to on the prediction. With confidence of each prediction, an ROC curve can be drawn. The area under curve (AUC) is defined as the area under this ROC curve.

4.2.3 Fraction of Features

The fraction of features is the ratio of the number of features used by the classifier to the total number of features in the data set.

4.2.4 Fraction of Probes

A certain number of features meaningless by design were introduced in generating the competition data. These features are called random probes. The ratio of random probes selected is also a criterion to judge the performance; the smaller it is, the more capable the selection approach is on filtering out irrelevant features.

The relative strength of classifiers was judged only on the BER. For methods having performance differences which were not statistically significant, the method using the smallest number of features would win. Fraction of probes were used to assess the relative strength of methods that were not significantly different both in the error rate and the number of features. In that case, the method with the smallest number of random probes in the feature set won.

Although there were several performance measures, throughout the competition we focused on how to achieve the smallest BER.

4.3 Data Sets Information

The organizers prepare competition data by transforming some publicly available data sets. Table 4.1 contains some statistics of competition data sets. However, the idendity of the original data was not revealed until the submission deadline, so competitors cannot use domain knowledge about the data. Moreover, irrelevant features were added as "random probes."

Table 4.1: Statistics of competition data sets

Data set	ARCENE	DEXTER	DOROTHEA	GISETTE	MADELON
Feature Numbers	10000	20000	100000	5000	500
Training Size	100	300	800	6000	2000
Validation Size	100	300	350	1000	600
Testing Size	700	2000	800	6500	1800

4.3.1 Source of Data Sets

The source of five competition data sets are described in the following.

ARCENE came from the cancer data set of National Cancer Institute (NCI) and Eastern Virginia Medical School (EVMS) together with 3,000 random probes. The labels indicated whether patterns were cancer or not.

DEXTER was a subset of Reuters text categorization benchmark. The labels indicated whether articles were about "corporate acquisitions." 9,947 features represented word frequencies while 10,053 random probes were added.

DOROTHEA was the Thrombin data set, which was from KDD (Knowledge Discovery in Data Mining) Cup 2001. The last 50,000 features were randomly permuted to be random probes.

GISETTE was transformed from MNIST [22], a handwritten digits recognition problem. The task was to separate digits "4" and "9." Features were created by a

random selection of subsets of products of pairs of pixel values plus 2,500 random probes.

MADELON was a artificial data with only five useful features. Clusters of different classes were placed on the summits of a five dimensional hypercube. Besides those five useful features, there were also five redundant features, ten repeated features, and 480 random probes.

4.4 Strategies in Competition

In this section, we discuss feature selection strategies tried during the competition. Each method is named as "A + B," where A is a filter to select features and B is a classifier or a wrapper. If a method is "A + B + C," it means that there are two filters A and B.

4.4.1 No Selection: Direct Use of SVM

The first strategy was to directly use SVMs without feature selection. Thus, the procedure in Section 2.6 was considered.

4.4.2 F-score for Feature Selection: F-score + SVM

Introduced in Section 3.1.1, F-score is a simple technique which measures the discrimination of a feature. In the competition, a variant of F-score different from (3.1.2) was used:

$$\tilde{F}(j) \equiv \frac{\tilde{S}_B(j)}{\tilde{S}_W(j)}, \tag{4.4.1}$$

where

$$\tilde{S}_B(j) = \sum_{k=1}^{2} \left(\bar{x}_j^k - \bar{x}_j\right)^2, \tag{4.4.2}$$

$$\tilde{S}_W(j) = \sum_{k=1}^{2} \frac{1}{l_k - 1} \sum_{\mathbf{x} \in X^k} \left(x_j - \bar{x}_j^k\right)^2 \tag{4.4.3}$$

with the same notation definition in (3.1.2)

We selected features with high F-scores and then applied SVM for training/prediction. The procedure is summarized below:

Algorithm 3 F-SCORE + SVM

1. *Calculate F-score of every feature.*
2. *Pick some possible thresholds by human eye to cut low and high F-scores.*
3. *For each threshold, do the following*
   (a) *Drop features with F-score below this threshold.*
   (b) *Randomly split the training data into X_{train} and X_{valid}.*
   (c) *Let X_{train} be the new training data. Use the SVM procedure in Chapter II to obtain a predictor; use the predictor to predict X_{valid}.*
   (d) *Repeat steps (a)-(c) five times, and then calculate the average validation error.*
4. *Choose the threshold with the lowest average validation error.*
5. *Drop features with F-score below the selected threshold. Then apply the SVM procedure in Chapter II.*

In the above procedure, possible thresholds were identified by "human eye." For data sets in this competition, there was a quite clear gap between high and lower F-scores (see Figure 4.1, which will be described in Section 4.5).

4.4.3 F-score and Random Forest for Feature Selection: F-score + RF + SVM

Random Forest was involved in our third method. We followed a similar procedure in [35] but used SVMs as classifiers instead.

In practice, the RF code we used could not handle too many features. Thus, before using RF to select features, we obtained a subset of features using F-score selection first. This approach is thus called "F-score + RF + SVM" and is summarized below:

Algorithm 4 F-score + RF + SVM

1. *F-score*

   (a) *Consider the subset of features obtained in Section 4.4.2.*

2. *RF*

   (a) *Initialize the RF working data set to include all training instances with the subset of features selected from Step 1. Use RF to obtain the rank of features.*

   (b) *Use RF as a predictor and conduct five-fold CV on the working set.*

   (c) *Update the working set by removing half features which are less important and go to Step 2b.*
   Stop if the number of features is small.

   (d) *Among various feature subsets chosen above, select the one with the lowest CV error.*

3. *SVM*

(a) Apply the SVM procedure in Chapter II on the training data with the selected features.

4.4.4 Random Forest and RM-bound SVM for Feature Selection

Our final strategy was to apply RM-bound SVMs described in Section 3.3.1 on competition data sets. When the number of features is large, minimizing the RM-bound is time consuming. Thus, we applied this technique only on the problem MADELON, which contained 500 features. To further reduce the computational burden, we used RF to pre-select important features. Thus, this method is referred to as "RF + RM-SVM."

4.5 Experimental Results

In the experiment, we used LIBSVM [6] for SVM classification. For feature selection methods, we used the randomForest [23] package[†] in the software R for RF and modified the implementation in [10] for the RM-bound SVM[‡]. Data sets were scaled before doing experiments. With training, validation, and testing data together, each feature is linearly scaled to $[0, 1]$. Except scaling, there was no other data preprocessing.

In the development stage, only labels of training sets were known. An on-line judge returned BER of what competitors predicted about validation sets, but labels of validation sets and even information of testing sets were kept unknown.

We mainly focused on three feature selection strategies discussed in Sections 4.4.1-4.4.3: SVM, F-score + SVM, and F-score + RF + SVM. For RF + RM-SVM, due to the large number of features, we only applied it on MADELON. The RF

[†] `http://www.r-project.org/`

[‡] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/`

procedure in Section 4.4.3 selected 16 features and then an RM-SVM scaled them. In all experiments we focused on getting the smallest BER.

For the strategy F-score + RF + SVM, after the initial selection by F-score, we found that RF retained all features. That is, by comparing cross-validation BER using different subsets of features, the one with all features was the best. Hence, F+RF+SVM was in fact the same as F+SVM for all the five data sets. Since our validation accuracy of DOROTHEA was not good when compared to some participants, we considered a heuristic by submitting results via the top 100, 200, and 300 features from RF. The BERs of the validation set were 0.1431, 0.1251, and 0.1498, respectively. Therefore, we considered "F-score + RF top 200 + SVM" for DOROTHEA.

Table 4.2 presents the BER on validation data sets by different feature selection strategies. It shows that no method is the best on all data sets.

Table 4.2: Comparison of different methods during the development period: BERs of validation sets (in percentage); bold-faced entries correspond to approaches used to generate our final submission

Data set	ARCENE	DEXTER	DOROTHEA	GISETTE	MADELON
SVM	**13.31**	11.67	33.98	2.10	40.17
F+SVM	21.43	**8.00**	21.38	**1.80**	13.00
F+RF+SVM	21.43	8.00	**12.51**	1.80	13.00
RF+RM-SVM[§]	–	–	–	–	**7.50**

In Table 4.3 the CV BER on the training set is listed. Results of the first three problems are quite different from those in Table 4.2. Due to the small training sets or other reasons, CV did not accurately indicate the future performance.

In Table 4.4, the first row indicates the threshold of F-score. The second row is the number of selected features which is compared to the total number of features in

[§]Our implementation of RF+RM-SVM is applicable to only MADELON, which has a smaller number of features.

Table 4.3: CV BER on the training set (in percentage)

Data set	ARCENE	DEXTER	DOROTHEA	GISETTE	MADELON
SVM	11.04	8.33	39.38	2.08	39.85
F+SVM	9.25	4.00	14.21	1.37	11.60

Table 4.4: F-score threshold and the number of features selected in the approach F+SVM

Data set	ARCENE	DEXTER	DOROTHEA	GISETTE	MADELON
F-score threshold	0.1	0.015	0.05	0.01	0.005
#features selected	661	209	445	913	13
#total features	10000	20000	100000	5000	500

the third row. Figure 4.1 presents the curve of F-scores against features.

4.6 Competition Results

For each data set, we submitted the final result using the method that led to the best validation accuracy in Table 4.2. A comparison of competition results (ours and winning entries) is in Tables 4.5 and 4.6. The column "Score" means the overall performance among all the methods. The rest of the columns are performance measures mentioned in Section 4.2.

For the development stage submissions, we ranked 1^{st} on GISETTE, 3^{rd} on MADELON, and 5^{th} on ARCENE. Overall we ranked 3^{rd} as a group and our best entry was the 6^{th}, using the criterion of the organizers. For the final stage submissions, we ranked 2^{nd} as a group and our best entry was the 4^{th}.

4.7 Discussion and Conclusions from the Competition

Usually SVMs suffer from a large number of features, but we found that a direct use of SVM works well on GISETTE and ARCENE. After the competition, we realized that GISETTE came from an OCR problem MNIST [22] with 784 features of gray-level

Table 4.5: NIPS 2003 challenge results of the development stage

Our best challenge entry					
Data set	Score	BER	AUC	Feat	Probe
OVERALL	52.00	9.31	90.69	24.9	12.0
ARCENE	74.55	15.27	84.73	100.0	30.0
DEXTER	0.00	6.50	93.50	1.0	10.5
DOROTHEA	-3.64	16.82	83.18	0.5	2.7
GISETTE	98.18	1.37	98.63	18.3	0.0
MADELON	90.91	6.61	93.39	4.8	16.7
The winning challenge entry					
Data set	Score	BER	AUC	Feat	Probe
OVERALL	88.00	6.84	97.22	80.3	47.8
ARCENE	98.18	13.30	93.48	100.0	30.0
DEXTER	96.36	3.90	99.01	1.5	12.9
DOROTHEA	98.18	8.54	95.92	100.0	50.0
GISETTE	98.18	1.37	98.63	18.3	0.0
MADELON	100.00	7.17	96.95	1.6	0.0

values. Thus, all features are of the same type and tend to be equally important. Earlier experience indicated that for such problems, SVM could handle a rather large set of features. As the 5,000 features of GISETTE were a combination of the original 784 features, SVM may still work well under the same explanation.

For the data set MADELON, the winner used Bayesian SVM described in Section 3.3.2. It is similar to RM-SVM by minimizing a function of feature-wise scaling factors. The main difference is that RM-SVM minimizes an loo bound, but Bayesian SVM derives a Bayesian evidence function. For this problem Tables 4.5 and 4.6 indicated that the two approaches achieved very similar BER. This result would indicate a strong relation between the two methods. Though they are derived from different aspects, it is worth investigating the possible connection.

In conclusion, we have tried several feature selection strategies in this competition. Most of them were independent of the classifier used. Strategies and algorithms were finely tuned respectively for each data set. However, it seems that there was no

Table 4.6: NIPS 2003 challenge results of the final stage

Our best challenge entry					
Data set	Score	BER	AUC	Feat	Probe
OVERALL	49.14	7.91	91.45	24.9	9.9
ARCENE	68.57	10.73	90.63	100.0	30.0
DEXTER	22.86	5.35	96.86	1.2	2.9
DOROTHEA	8.57	15.61	77.56	0.2	0.0
GISETTE	97.14	1.35	98.71	18.3	0.0
MADELON	71.43	7.11	92.89	3.2	0.0
The winning challenge entry					
Data set	Score	BER	AUC	Feat	Probe
OVERALL	88.00	6.84	97.22	80.3	47.8
ARCENE	94.29	11.86	95.47	10.7	1.0
DEXTER	100.00	3.30	96.70	18.6	42.1
DOROTHEA	97.14	8.61	95.92	100.0	50.0
GISETTE	97.14	1.35	98.71	18.3	0.0
MADELON	94.29	7.11	96.95	1.6	0.0

general method best to all data. This work was a preliminary study that for an SVM package which feature selection strategies should be included. Thus, more systematic comparisons with general strategies are investigated in the following chapters.

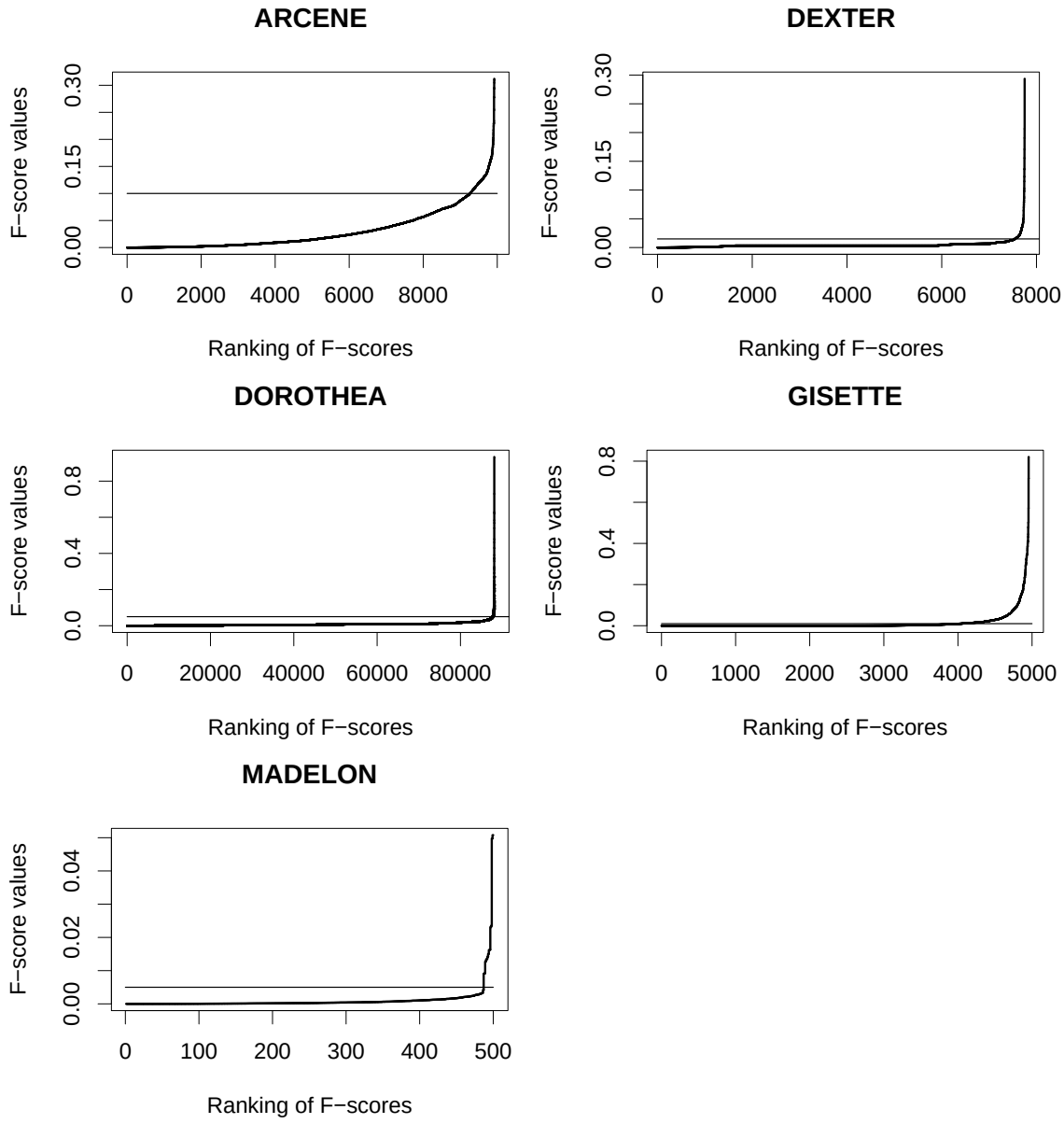


Figure 4.1: Curves of F-scores against features; features with F-scores below the horizontal line are dropped

CHAPTER V

Other Feature Ranking Methods by SVM

Among feature ranking/selection methods introduced in Chapter III, some are independent of any classifier, but some are specific to SVM. In this chapter, we first introduce the work of [16]. In Section 3.1.3, weights obtained by linear SVMs are used as feature ranking. [16] proposes a method that uses non-linear SVMs. Then, we extend two of the methods in Chapter III by using SVM:

1. Similar to [16], we also propose extensions which use the weights of nonlinear SVMs as the feature ranking.

2. Section 3.1.2 explains comparing performance between the original data and the new set by shuffling a feature vector gives the feature importance. There is no SVM implementation yet on using this approach, so we explore the use of this strategy here.

5.1 Normal Vector of the Decision Boundary in Nonlinear SVM

In Section 3.1.3, we mentioned that weights of non-linear SVMs can not be directly used as the feature ranking. [16] proposes that the gradient of the decision function on support vectors should be considered in the influence of features. First,

the gradient of the decision boundary means the normal direction of it, which is consistent with the linear case in Section 3.1.3. Second, since support vectors are meaningful instances and they "support" the decision boundary, it is reasonable to consider the gradient on these instances.

More precisely, if there are n features in the data, the importance of the jth feature is defined as

$$d_j \equiv \frac{1}{|SV|} \sum_{\mathbf{x}_i \in SV} \frac{d_{j,\mathbf{x}_i}}{\sum_{k=1}^{n} d_{k,\mathbf{x}_i}}, \tag{5.1.1}$$

where

$$d_{j,\mathbf{x}} = (\nabla f(\mathbf{x})_j)^2$$

is the square of the projected length of $\nabla f(\mathbf{x})$ on the jth feature, SV is the set of support vectors, and $f(\mathbf{x})$ is the decision function. The denominator $\sum_{k=1}^{n} d_{k,\mathbf{x}_i}$ ensures that each support vector's contribution sums to one. Moreover, in order to eliminate small contribution from features, they use the square of $\nabla f(\mathbf{x})_j$ instead of $|\nabla f(\mathbf{x})_j|$.

5.2 Change of Decision Value in Nonlinear SVM

Like the previous section, in this section it is assumed that the feature ranking score is the summation of "contributions" from some instances. Different approaches define their own contribution function.

However, different from the previous section which simply use the normal vector of the decision boundary, here we propose other methods where the change of decision values are observed.

5.2.1 Instances from Underlying Distribution

Assume instances of the data set come from a probability distribution P. Suppose a feature score with respect to an instance $\mathbf{s}(\mathbf{x})$ can be defined. It is reasonable to set the feature importance of the whole data set as the expectation of $\mathbf{s}(\mathbf{x})$:

$$E\{\mathbf{s}(\mathbf{x})\} = \int \mathbf{s}(\mathbf{x})P(\mathbf{x})d\mathbf{x}, \tag{5.2.1}$$

where $\mathbf{s}(\mathbf{x})$ is the feature importance contributed by $\mathbf{x}$.

How to properly define $\mathbf{s}(\mathbf{x})$ is an important issue. This can be done by observing the change of the decision value. Depending on how significant the changes of $\mathbf{x}$'s features may modify the decision value, we can have a rough estimation on the importance of features. For the decision value function f and a novel instance $\mathbf{x}$, let $g(\mathbf{x}, \mathbf{\Delta})$ be an approximation of $f(\mathbf{x} + \mathbf{\Delta})$ using the first order Taylor's expansion; that is, the decision value when the instance is shifted:

$$g(\mathbf{x}, \mathbf{\Delta}) \equiv f(\mathbf{x}) + \mathbf{\Delta}^T \nabla f(\mathbf{x}) \approx f(\mathbf{x} + \mathbf{\Delta}). \tag{5.2.2}$$

If $f(\mathbf{x}) < 0$, when $g(\mathbf{x}, \mathbf{\Delta}) \geq 0$,

$$\begin{aligned} & \mathbf{\Delta}^T \nabla f(\mathbf{x}) \geq -f(\mathbf{x}) \\ \Rightarrow \quad & \|\mathbf{\Delta}\| \|\nabla f(\mathbf{x})\| \geq -f(\mathbf{x}) \\ \Rightarrow \quad & \|\mathbf{\Delta}\| \geq \tfrac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|}. \end{aligned}$$

If $f(\mathbf{x}) \geq 0$, when $g(\mathbf{x}, \mathbf{\Delta}) \leq 0$, we also have $\|\mathbf{\Delta}\| \geq \frac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|}$. Furthermore, their equalities hold when This equality holds when $\mathbf{\Delta} = \alpha \nabla f(\mathbf{x})$, α is a scalar. The formulas above suggests that, if $\mathbf{x}$ is interfered by some noise, among all possible directions of the noise, it is the most likely to see the sign difference of the decision value when the direction of the noise is parallel to $\nabla f(\mathbf{x})$. Figure 5.1 contains a simple

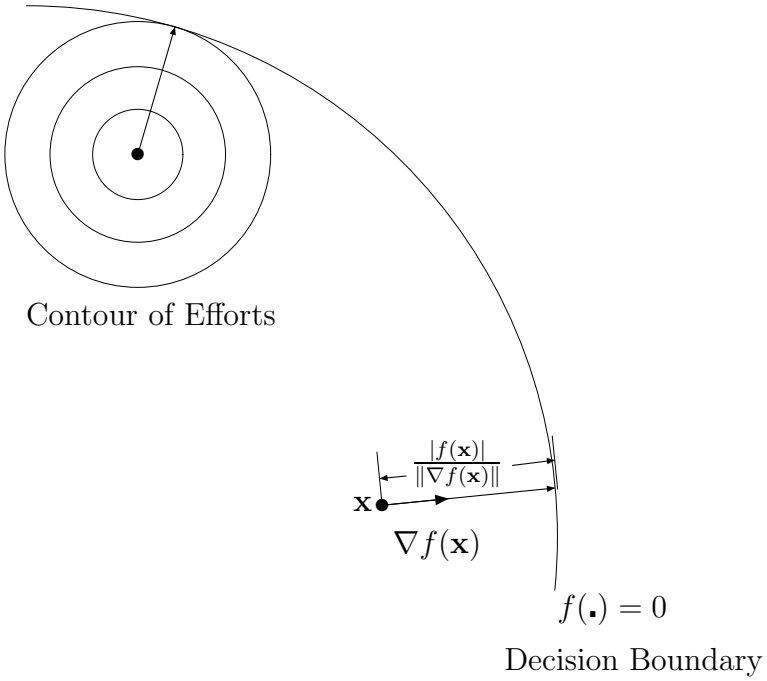


Figure 5.1: A two-dimensional illustration about "pushing" an instance $\mathbf{x}$ to the decision boundary

two-dimensional illustration. Therefore it shows that $s(\mathbf{x})_j$ should be proportional to $\nabla f(\mathbf{x})_j$, and the shorter $\boldsymbol{\Delta}$ is, the more $\mathbf{x}$ contributes to $\mathbf{s}(\mathbf{x})$. Thus, $\mathbf{s}(\mathbf{x})$ in (5.2.1) can be defined as:

$$\mathbf{s}(\mathbf{x}) \equiv D\left(\frac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|}\right)\frac{A(\nabla f(\mathbf{x}))}{\|\nabla f(\mathbf{x})\|}, \tag{5.2.3}$$

where $D(d)$ is a decreasing function of d, indicating that the farther the distance between $\mathbf{x}$ and the boundary is, the more effort is needed to push $\mathbf{x}$ to the boundary, and therefore the less it contributes to the feature importance. Here $A(\mathbf{v})$ is component-wise absolute function for vector $\mathbf{v}$.

Another problem is, for most cases the underlying distribution is not available. This is accomplished by using cross validation technique to do estimation. See Algorithm 5 for more details.

Specifically, we consider the RBF kernel to train a model. For a binary classifi-

cation problem, from (2.4.1), $\nabla f(\mathbf{x})$ is:

$$\begin{aligned}\nabla f(\mathbf{x}) &= \frac{\partial}{\partial \mathbf{x}} \sum_{\mathbf{x}_i \in SV} y_i \alpha_i \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\right) \\ &= \sum_{\mathbf{x}_i \in SV} -2\gamma y_i \alpha_i \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\right) (\mathbf{x} - \mathbf{x}_i). \end{aligned} \tag{5.2.4}$$

Under this setting, $D(d)$ is designed as,

$$D(d) \equiv \exp\left(-\gamma d\right), \tag{5.2.5}$$

where γ is the parameter in the RBF kernel. The reason why γ involves in $D(d)$ is due to the difference between feature space and kernel space: For any two instances $\mathbf{x}$ and $\bar{\mathbf{x}}$ in the feature space, for a large γ, the distance in the kernel space will dramatically increase when $\|\mathbf{x} - \bar{\mathbf{x}}\|$ increases. On the contrary, with a small γ, the distance in the kernel space increases more slowly. Since instances are pushed to the decision boundary in the feature space, in fact more efforts are needed if γ is larger. Therefore, $\exp(-\gamma d)$ is used as the effort function $D(d)$.

To summarize, for a two-class data X, Algorithm 5 gives the feature ranking from data distribution.

Algorithm 5 CHANGE_FROM_DISTRIBUTION_BINARY(X)

1. *Do parameter selection on X and find the best parameter (C, γ).*

2. *Split X into five subsets X_i, $i = 1, \ldots, 5$.*

3. *For each i, $i = 1, \ldots, 5$, using X/X_i as training data and (C, γ) as the parameter to train a model M_i. Assume $f(\mathbf{x})$ is the decision function of M_i. Let*

$$\mathbf{v}^{(i)} = \sum_{\mathbf{x} \in X_i} \mathbf{s}(\mathbf{x}) = \sum_{\mathbf{x} \in X_i} D\left(\frac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|}\right) \frac{A(\nabla f(\mathbf{x}))}{\|\nabla f(\mathbf{x})\|},$$

where $D(d) = \exp(-\gamma d)$, *and* $A(\mathbf{v})$ *is component-wise absolute function for vector* $\mathbf{v}$.

4. *Let* $\mathbf{v} = \sum_{i=1}^{5} \mathbf{v}^{(i)}$. *The importance of the* j*th feature is then* v_j.

For multi-class problems, we consider the one-against-one strategy described in Section 2.5.1. Since it involves several binary SVMs and predicts the label with the largest number of votes, the gradient of the decision function is hard to calculate or even does not exist. Therefore, the score of an instance $\mathbf{x}$ in every binary SVM is calculated, and $\mathbf{s}(\mathbf{x})$ is then defined as the summation of all the scores. Except this difference, the algorithm that gives multl-class data feature ranking is the same as Algorithm 5. It is summarized below.

Algorithm 6 CHANGE_FROM_DISTRIBUTION_MULTICLASS(X)

1. *Do parameter selection on* X *and find the best parameter* (C, γ).

2. *Split* X *into five subsets* X_i, $i = 1, \ldots, 5$.

3. *For each* i, $i = 1, \ldots, 5$:

   (a) *Use* X/X_i *as training data and* (C, γ) *as the parameter to train a model* M_i.

   (b) *Assume* M_i *contains* $k(k-1)/2$ *two-class sub-models, and their decision functions are* $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{k(k-1)/2}(\mathbf{x})$, *respectively. Here* k *is the number of classes. Let*

$$B(\mathbf{x}) = \sum_{j=1}^{k(k-1)/2} D\left(\frac{|f_j(\mathbf{x})|}{\|\nabla f_j(\mathbf{x})\|}\right) \frac{A(\nabla f_j(\mathbf{x}))}{\|\nabla f_j(\mathbf{x})\|},$$

   which is the contribution function of one single instance $\mathbf{x}$. *Here* $D(d) = \exp(-\gamma d)$, *and* $A(\mathbf{v})$ *is component-wise absolute function for vector* $\mathbf{v}$.

(c) Let $\mathbf{v}^{(i)} = \sum_{\mathbf{x} \in X_i} B(\mathbf{x})$.

4. *Let* $\mathbf{v} = \sum_{i=1}^{5} \mathbf{v}^{(i)}$. *The importance of the* j*th feature is then* v_j.

5.2.2 Instances from Decision Boundary

In Section 5.2.1, contributions from the whole underlying distribution is considered. However, if an instance is far from the decision boundary, it is less important. This fact motivates us an idea to consider support vectors only, which are more important instances.

Similar to the procedures in the previous section, another kind of feature importance criterion obtained from support vectors, is calculated as follows:

Algorithm 7 CHANGE_FROM_SUPPORT_VECTORS_MULTICLASS(X)

1. *Do parameter selection on* X *to obtain a model* M *with parameter* (C, γ).

2. *Assume* M *contains* $k(k-1)/2$ *binary-class submodels, and their decision functions are* $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{k(k-1)/2}(\mathbf{x})$, *respectively.* k *is the number of classes. Let*

$$B(\mathbf{x}) = \sum_{j=1}^{k(k-1)/2} D\left(\frac{|f_j(\mathbf{x})|}{\|\nabla f_j(\mathbf{x})\|}\right) \frac{A(\nabla f_j(\mathbf{x}))}{\|\nabla f_j(\mathbf{x})\|},$$

which is the contribution function of one single instance $\mathbf{x}$. *Here* $D(d) = \exp(-\gamma d)$, *and* $A(\mathbf{v})$ *is component-wise absolute function for vector* $\mathbf{v}$.

3. *Let* $\mathbf{v} = \sum_{\mathbf{x} \in SV} B(\mathbf{x})$, *where* SV *is the set of support vectors in* M. *The importance of the* j*th feature is then* v_j.

Note that the formula is almost the same as the work of [16] described in Section 5.1 except the weights of contributions. He makes the extension from the aspect of the normal vector of the decision boundary. We interpret this as observing the

change of decision values of instances. However, from this point of view, we can further introduce different contribution functions which are connected to the kernel.

5.3 Random Shuffle on Features using Probability SVM

Section 3.1.2 describes a way to give a feature ranking which is general to all kinds of classifiers. When we apply this idea on the SVM, the first idea is to count the "number" of instances which are correctly predicted, and see its difference after shuffling on each feature. However, we found that the differences are often the same for some of the features. That is, the ranking of features is not obviously revealed.

Instead of count the number of correct instances, we calculate the "probability" of predicting instances correctly. Therefore, we use an SVM with probability outputs as the classifier,

5.3.1 SVM with Probability Output

Standard SVMs do not provide probability outputs. For binary classification problems with labels $\{1, -1\}$, [31] suggests estimating the probability $p(y = 1|\mathbf{x})$ by a sigmoid function of the decision value $f(\mathbf{x})$:

$$p(y = 1|\mathbf{x}) = \frac{1}{1 + \exp\left(Af(\mathbf{x}) + B\right)} \tag{5.3.1}$$

with parameters A and B. For training data $\mathbf{x}_i \in R^n, i = 1, \ldots, l, y_i \in \{1, -1\}$, where N_+ instances are in class 1 and N_- instances are in class -1, the best values of (A, B) are estimated by solving the maximum likelihood problem:

$$\min_{A,B} F(A, B), \tag{5.3.2}$$

where

$$
\begin{aligned}
F(A,B) &= -\sum_{i=1}^{l}\left(t_i \log(p_i) + (1-t_i)\log(1-p_i)\right), \qquad (5.3.3)\\
p_i &= \frac{1}{1+\exp\left(Af(\mathbf{x}_i)+B\right)}, \text{ and}\\
t_i &= \begin{cases} 1-\frac{1}{N_++2} & \text{if } y_i = 1\\ \frac{1}{N_-+2} & \text{if } y_i = -1 \end{cases}, i = 1,\ldots,l.
\end{aligned}
$$

Here to avoid overfitting, t_i is assigned to be the values above instead of zero or one.

For multi-class problems, [40] uses pairwise coupling techique that combines all comparisons for each pair of classes. For data sets with k classes, suppose the estimated pairwise class probabilities r_{ij} of $P(y=i|y=i \text{ or } j, \mathbf{x})$ are available (by [31] for example), they propose an optimization problem to obtain multi-class probability estimates $\mathbf{p}$:

$$
\begin{aligned}
\min_{\mathbf{p}} \quad & \sum_{i=1}^{k}\sum_{j:j\neq i}\left(r_{ji}p_i - r_{ij}p_j\right)^2 \qquad (5.3.4)\\
\text{subject to} \quad & \sum_{i=1}^{k} p_i = 1, p_i \geq 0, \forall i.
\end{aligned}
$$

5.3.2 Random Shuffle on Validation Data Features

Using the idea mentioned in Section 3.1.2, we propose the procedure which randomly shuffle on features by using SVMs with probability outputs to obtain the importance of the jth feature in a data set X:

Algorithm 8 SHUFFLING_WITH_PROBABILITYSVM_MULTICLASS(X, j)

1. *Do parameter selection on X and find the best parameter (C, γ).*

2. *Split X into five subsets X_i, $i = 1,\ldots,5$.*

3. *For each* i, $i = 1, \ldots, 5$:

   (a) *Use* X/X_i *as training data and* (C, γ) *as the parameter to train a model* M_i.

   (b) *Randomly shuffle the* j*th feature of* X_i *to obtain a new validation set* $\tilde{X}_i$.

   (c) *For any* q, $q = 1, \ldots, |X_i|$, *let* $\mathbf{x}$ *be the* q*th instance of* X_i, *and* $\tilde{\mathbf{x}}$ *be the* q*th instance of* $\tilde{X}_i$. *Assume the real label of* $\mathbf{x}$ *(also the real label of* $\tilde{\mathbf{x}}$*) is* y. *Use* M_i *to predict* $\mathbf{x}$ *and* $\tilde{\mathbf{x}}$. *Let* $p^{(q)}$ *(*$\tilde{p}^{(q)}$*) be the probability that* $\mathbf{x}$ *(*$\tilde{\mathbf{x}}$*) is predicted as label* y, *respectively.*

   (d) *Let* $\mathbf{v}^{(i)} = \sum_{q=1}^{|X_i|} p^{(q)} - \tilde{p}^{(q)}$. *Here* $p^{(q)} - \tilde{p}^{(q)}$ *means the difference of the performance (i.e. the probability to correctly predict the* q*th instance) after shuffling.*

4. *Let* $\mathbf{v} = \sum_{i=1}^{5} \mathbf{v}^{(i)}$. *The importance of the* j*th feature is then* v_j.

Note that this importance may sometimes be negative, if the probability that instances are predicted as the true class is higher after shuffling.

CHAPTER VI

Experiments

In this chapter we describe the numerical experiments using feature selection methods mentioned in Chapters III and V.

6.1 Experiment Procedures

We tried several feature selection methods. Most of them combine feature ranking methods together with feature size estimation described in Section 3.2.2 to select important features. Another one is to use RM-bound to do feature scaling, which is mentioned in Section 3.3.1.

6.1.1 Feature Selection by Ranking

In this chapter we focus on five feature ranking strategies. The first one is multi-class F-score (3.1.2) in Section 3.1.1. The second and the third are two methods that observe the change of the decision values from data distribution and support vectors, which are Algorithm 6 and Algorithm 7 described in Section 5.2.1 and 5.2.2. Another method is the criterion (5.1.1) mentioned in Section 5.1. The last one is Algorithm 8, random shuffling on features by probability SVMs in Section 5.3.2.

After the feature ranking of a data set is obtained, a procedure similar to Algorithm 8 in Section 3.2.2 is applied to select important features. The only difference is

that features with non-positive ranking scores are removed. The detailed procedure is as follows:

Algorithm 9 FEATURE_SELECTION_BY_RANKING

1. *Calculate the feature ranking on the whole training data. Let* $\mathcal{F}$ *be the set of features which have positive feature importance. This is the set from which important features will be chosen.*

2. *Let*

$$s_1 = |\mathcal{F}|, s_i = \lfloor \frac{s_{i-1}}{2} \rfloor, i = 2, \ldots, 8,$$
$$\mathcal{S} = \{s_i | s_i \geq 2, i = 1, \ldots, 8\},$$

   which is the set of feature numbers to try during the selection. (Note that this means about at least 1% *of features are selected.)*

3. *Partition the whole training data for* 3*-fold cross-validation.*

4. *For each CV training set:*

   (a) *Obtain the feature ranking of this CV training set.*

   (b) *For every size* s *in* $\mathcal{S}$*, Let the working features be the most* s *important features of this CV training set. Under the working features, construct a model by the CV training set to predict the CV testing set.*

5. *Obtain the CV accuracy for each size in* $\mathcal{S}$*. Assign* p *to be the size with the maximal CV accuracy.*

6. *Calculate the feature ranking on the whole training data. Select the most important* p *features. These features are then used for the final model construction and prediction.*

Combining five feature ranking strategies with the procedure above, we have five feature selection method now.

6.1.2 Feature Scaling using RM-bound SVM

Besides methods that combine the feature ranking and SVMs, we also investigate feature scaling done by RM-bound SVMs introduced in Chapter 3.3.1. For an RM-bound SVM, its optimal parameters C and $\gamma_1, \ldots, \gamma_n$ are obtained by minimizing the loo-bound 3.3.7. The gradient-based methods mentioned in [10] is used to iteratively minimize the bound. The initial point is set to be $\left(\bar{C}, \bar{\gamma}, \ldots, \bar{\gamma}\right)$, where $\left(\bar{C}, \bar{\gamma}\right)$ is the optimal parameter of the standard SVM using all the features. Furthormore, during the iterative steps in minimization, when any one of $\gamma_1, \ldots, \gamma_n$ is smaller than $0.5e^{-10}$, that is, the scaling factor of one feature is very small, we propose two methods to handle this situation:

1. Stop the minimization. Use the current parameter to create the predicting model. Or,

2. remove features whose scaling factors smaller that $0.5e^{-10}$, and then use an RM-bound SVM to train the new data again until no features are removed.

Note that the second method is more aggressive than the first one. It removes unimportant features recursively.

Therefore, in addition to the five feature ranking methods, we have two feature scaling methods using RM-bound SVMs.

6.2 Data Sets

Table 6.1 lists all the data sets and their statistics for experiments.

Table 6.1: Statistics of all data sets. The column "accuracy" is the testing accuracy of original problems. $(\log_2 C, \log_2 \gamma)$ is the optimal parameter obtained by parameter selection.

Name	# train	# test	# feature	# class	accuracy	$(\log_2 C, \log_2 \gamma)$
arcene	100	100	10000	2	86.00%	(7,-13)
arrhythmia	280	140	278	2	80.71%	(3,-11)
covtype2	5200	61012	54	2	80.09%	(5,1)
dexter	300	300	20000	2	90.67%	(5,-9)
diabetes	468	300	8	2	76.33%	(7,-5)
digits	1000	1000	649	2	99.90%	(5,-13)
dorothea	800	350	100000	2	92.57%	(3,-13)
ijcnn	5000	45495	22	2	97.78%	(5,1)
image	1300	1010	18	2	97.72%	(9,-1)
madelon	2000	600	500	2	60.00%	(-5,-3)
ringnorm	400	7000	20	2	98.51%	(-1,1)
splice	1000	2175	60	2	89.38%	(5,-7)
tree	700	11692	18	2	88.49%	(13,-5)
twonorm	400	7000	20	2	97.11%	(1,1)
waveform	400	4600	21	2	89.28%	(-1,-1)
dna	2000	1186	180	3	94.27%	(3,-5)
protein	3000	6621	357	3	65.46%	(5,-3)
satimage	3104	2000	36	6	90.30%	(3,1)
shuttle	30450	14500	9	7	99.92%	(11,3)
usps	7291	2007	256	10	95.27%	(3,-5)

Problems arcene, dexter, dorothea, and madelon are from the NIPS feature selection challenge described in Chapter IV. Since the labels of competition testing data are not available, we use their training data as our training data and their validation data as our testing data here. Data arrhythmia, diabetes, and digits are available in the UCI machine learning repository [2]. Problem covtype2 is also from the UCI repository but is transformed to two classes [11]. Problems image, ringnorm, splice, twonorm, and waveform are originally from the UCI repository and are processed in [33]. Problem tree was originally used in [1]. The set ijcnn is from the first problem of IJCNN 2001 challenge [32] and we use the winner's transformation of raw data [5]. Data sets dna, shuttle and satimage are available at Statlog collection [29]. Problem protein is from [38]. Problem usps is a digit recognition problem from the United States Postal Service [36]. Due to the large training size of some data sets, for problem covtype2, ijcnn and protein, we take only their subsets as the training data. All data sets are available in `http://www.csie.ntu.edu.tw/~r92010/thesis/datasets`.

Before doing feature selection experiments, first we use the standard SVM to train and predict each data set with all features. Table 6.1 lists the optimal parameters by parameter selection and their corresponding testing accuracy.

6.3 Experimental Results

Table 6.2 lists the testing accuracy obtained by the feature selection/scaling methods mentioned in the previous section. The column "$\mathbf{w}$ Heiler" means the criterion (5.1.1). The column "RM-bound" uses the RM-bound SVM once, while the column "RM-bound agg." represents the strategy which uses the RM-bound SVM and removes unimportant features recursively. Since the numbers of some data sets are large, some experiments cannot be done due to the limitation of time or the memory.

To judge and compare feature selection methods which are combined with feature ranking, the ranking itself as well as the strategy to determine the number of features is important. Therefore, we compare the performance for all feature ranking methods under different sizes of selected features. Figure 6.1 to Figure 6.5 show the curves of the accuracy against the number of features. For every data set, the figure on the left, "estimates," shows the CV accuracy of Algorithm 9. The figure on the right, "testing," shows the testing accuracy of the testing data.

6.4 Analysis

The first observation from Table 6.2 is that for the RM-bound SVM, aggressively removing features usually performs no better than applying the RM-bound SVM once. This may suggest that the aggressive RM-bound SVM sometimes removes informative features. Furthermore, if we compare the two feature scaling methods with the other five feature selection methods, we can see that in most cases the feature scaling methods provide only the similar accuracy, although they indeed have good improvement on madelon, splice, and dna.

The second observation is that, among the five feature selection methods, in general there is no method that is the best for all data sets. For some problems they all select the whole set of features, so it is hard to judge them only from the testing accuracy of the selected feature subsets. If we consider the data set with a large number of features, such as arcene, arrhythmia, dexter, dorothea and madelon, we may conclude that the method using F-score criterion as the feature ranking is slightly better than the other four methods.

From Figure 6.1 to Figure 6.5, we can see that in general the trend of the CV accuracy against the number of features is roughly the same as the trend of the testing

Table 6.2: The testing accuracy obtained by each feature selection/scaling method. The numbers of features selected are also reported in parentheses for those five feature selection methods. Some results are not available due to the limitation of the programs.

Name	F-score	Change dist.	Change SV	**w** Heiler	Shuffle prob.	RM-bound	RM-bound agg.
arcene	**84.00%** (2480)	81.00% (1237)	83.00% (4960)	80.00% (1235)	83.00% (4610)	–	–
arrhythmia	**83.57%** (32)	77.14% (56)	76.43% (31)	77.14% (56)	77.86% (69)	79.29%	79.29%
covtype2	**80.09%** (54)	**80.09%** (54)	**80.09%** (54)	**80.09%** (53)	80.09% (52)	78.47%	77.90%
dexter	**91.67%** (242)	89.33% (3829)	88.33% (60)	88.67% (3828)	85.33% (62)	–	–
diabetes	**76.67%** (8)	**76.67%** (8)	**76.67%** (8)	**76.67%** (8)	76.00% (6)	**76.67%**	**76.67%**
digits	**100.00%** (81)	**100.00%** (40)	99.80% (5)	**100.00%** (40)	**100.00%** (6)	**100.00%**	**100.00%**
dorothea	**93.14%** (1376)	92.57% (669)	92.86% (688)	92.57% (669)	– –	–	–
ijcnn	**97.78%** (22)	**97.78%** (22)	**97.78%** (22)	**97.78%** (22)	**97.78%** (22)	95.64%	90.01%
image	**97.72%** (18)	**97.72%** (18)	**97.72%** (18)	97.23% (9)	**97.72%** (18)	97.43%	82.28%
madelon	87.50% (15)	87.83% (15)	87.83% (15)	87.83% (15)	51.83% (9)	92.00%	**92.33%**
ringnorm	**98.51%** (20)	**98.51%** (20)	**98.51%** (20)	**98.51%** (20)	**98.51%** (20)	97.51%	97.51%
splice	93.10% (7)	93.93% (7)	93.93% (7)	93.93% (7)	93.06% (6)	**94.94%**	92.41%
tree	**88.49%** (18)	88.09% (9)	**88.49%** (18)	88.09% (9)	88.45% (17)	87.40%	87.40%
twonorm	97.11% (20)	97.11% (20)	97.11% (20)	97.11% (20)	**97.23%** (19)	96.43%	96.43%
waveform	89.28% (21)	89.28% (21)	89.28% (21)	89.28% (21)	88.30% (8)	**89.52%**	**89.52%**
dna	94.69% (22)	95.19% (45)	94.27% (180)	94.27% (180)	94.27% (130)	**95.28%**	58.52%
protein	65.68% (178)	65.46% (356)	65.46% (356)	65.46% (356)	**65.70%** (136)	60.90%	59.89%
satimage	90.30% (36)	90.30% (36)	90.30% (36)	90.30% (36)	90.30% (36)	**90.50%**	**90.50%**
shuttle	**99.93%** (4)	**99.93%** (4)	**99.93%** (4)	**99.93%** (4)	99.92% (9)	99.86%	99.86%
usps	**95.27%** (256)	**95.27%** (256)	**95.27%** (256)	**95.27%** (256)	**95.27%** (256)	94.82%	94.82%

accuracy. This observation tells us that Algorithm 9 gives reasonable estimates of the selected feature size: if the CV accuracy of some feature size is higher, then so is the testing accuracy.

As mentioned earlier, for many data all methods selected the whole set of features, and we cannot tell whether the ranking is good or bad. Therefore, we try to observe the curve from those figures to see which feature ranking is better. However, the results are still dependent on data sets. For example, for usps and covtype2 the random shuffling provides good ranking while F-score does not. However, F-score performs well on arcene, dorothea, and protein. For data with a small number of features, we may conclude that F-score sometimes provides a slightly worse ranking compared to the other four strategies.

Since the method of [16] and the methods that observe the change of the decision values are quite similar, we also compare the performance on these three strategy. We found that for most data such as dna, they have almost the same performace and there is no significant difference.

There is also an interesting observation for data madelon when using the random shuffling on features. Its result is merely a random guess. We check the obtained values of the feature importance and find that they are extremely small. Thus, there is almost no performance difference after shuffling. We conjecture that because the number of features is large (500 features), and the testing accuracy using all features is small (60%), there are too many outliers.

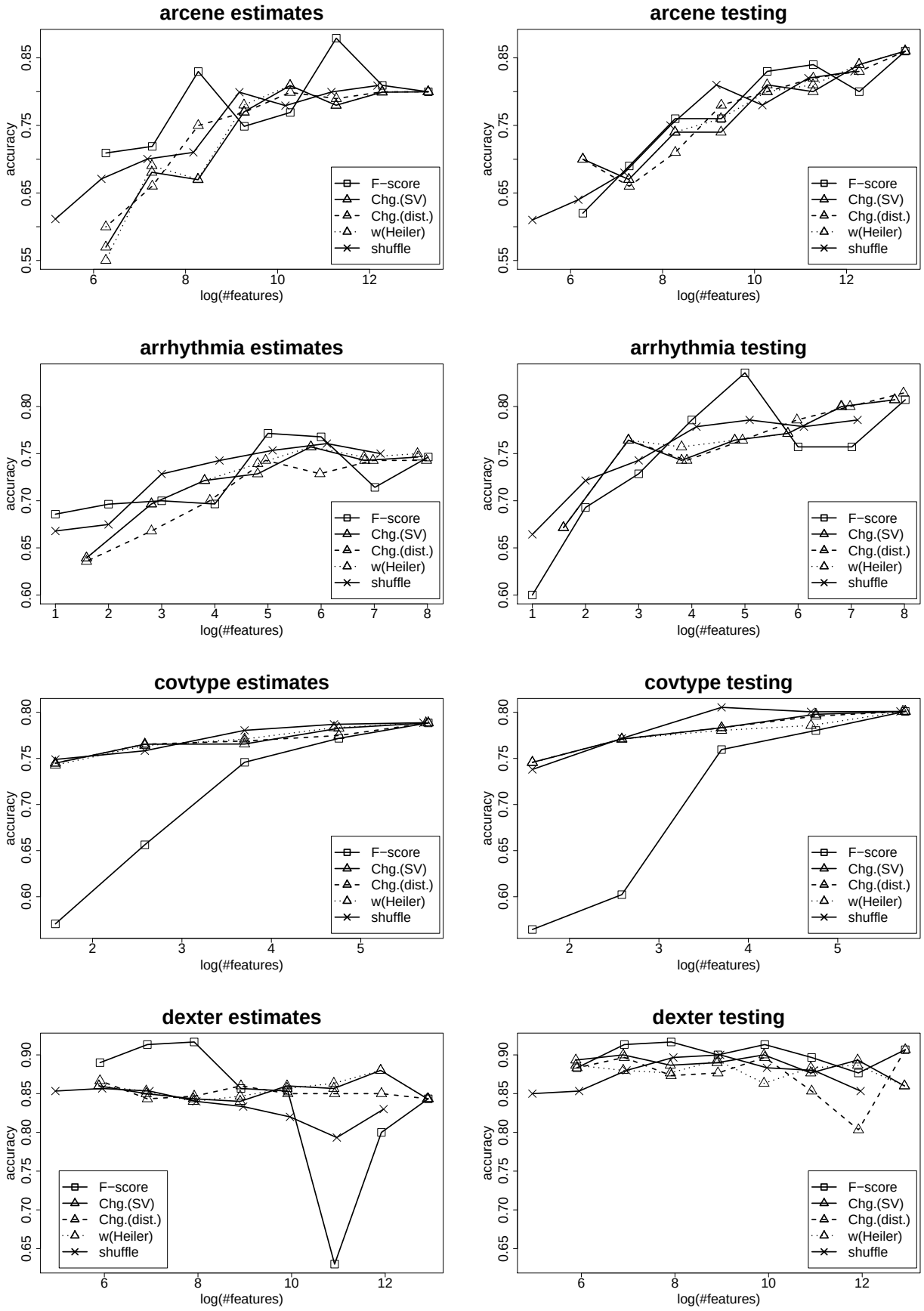


Figure 6.1: Comparisons of CV accuracy and testing accuracy against log number of features between feature ranking methods.

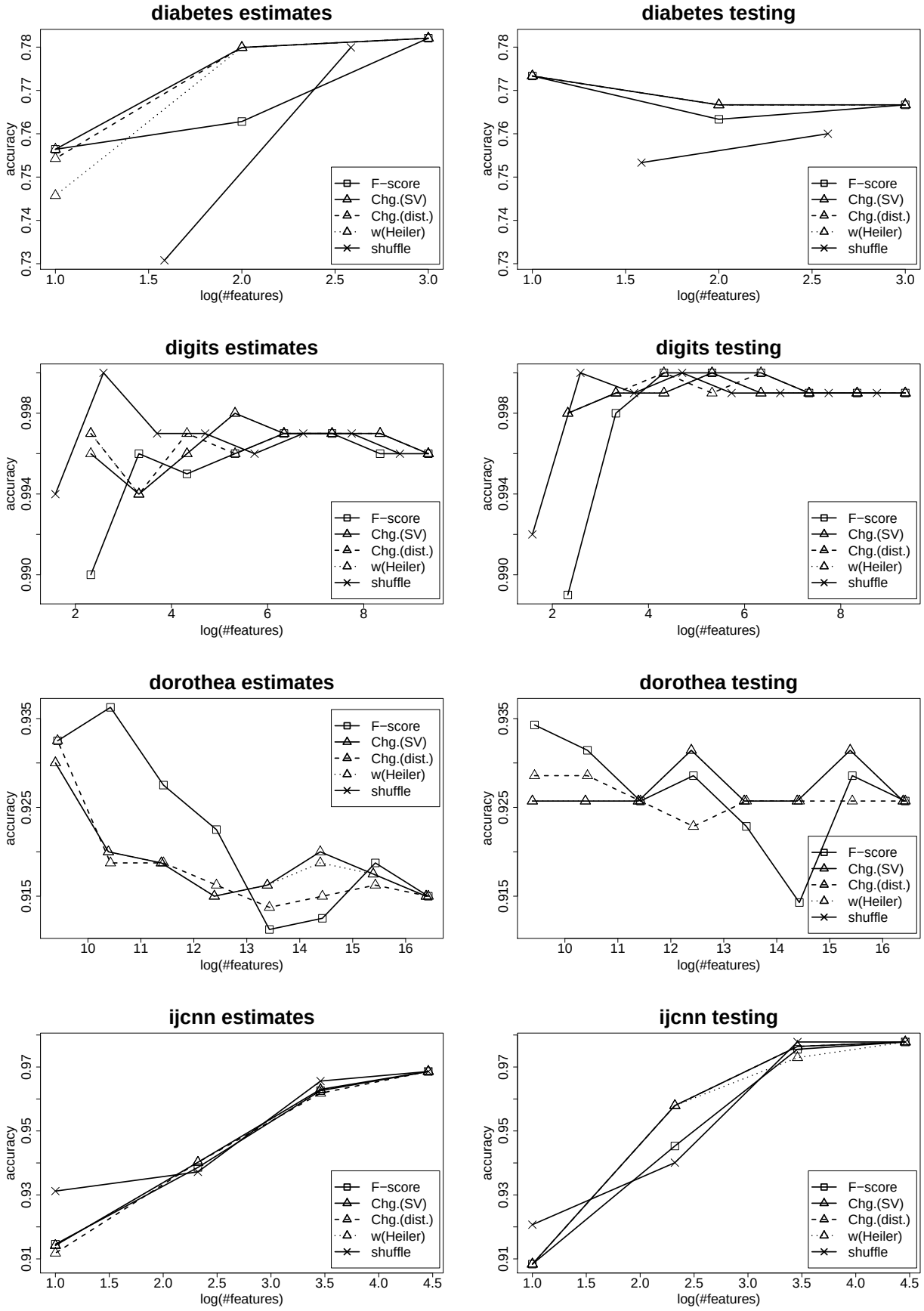


Figure 6.2: Comparisons of CV accuracy and testing accuracy against the log number of features between feature ranking methods.

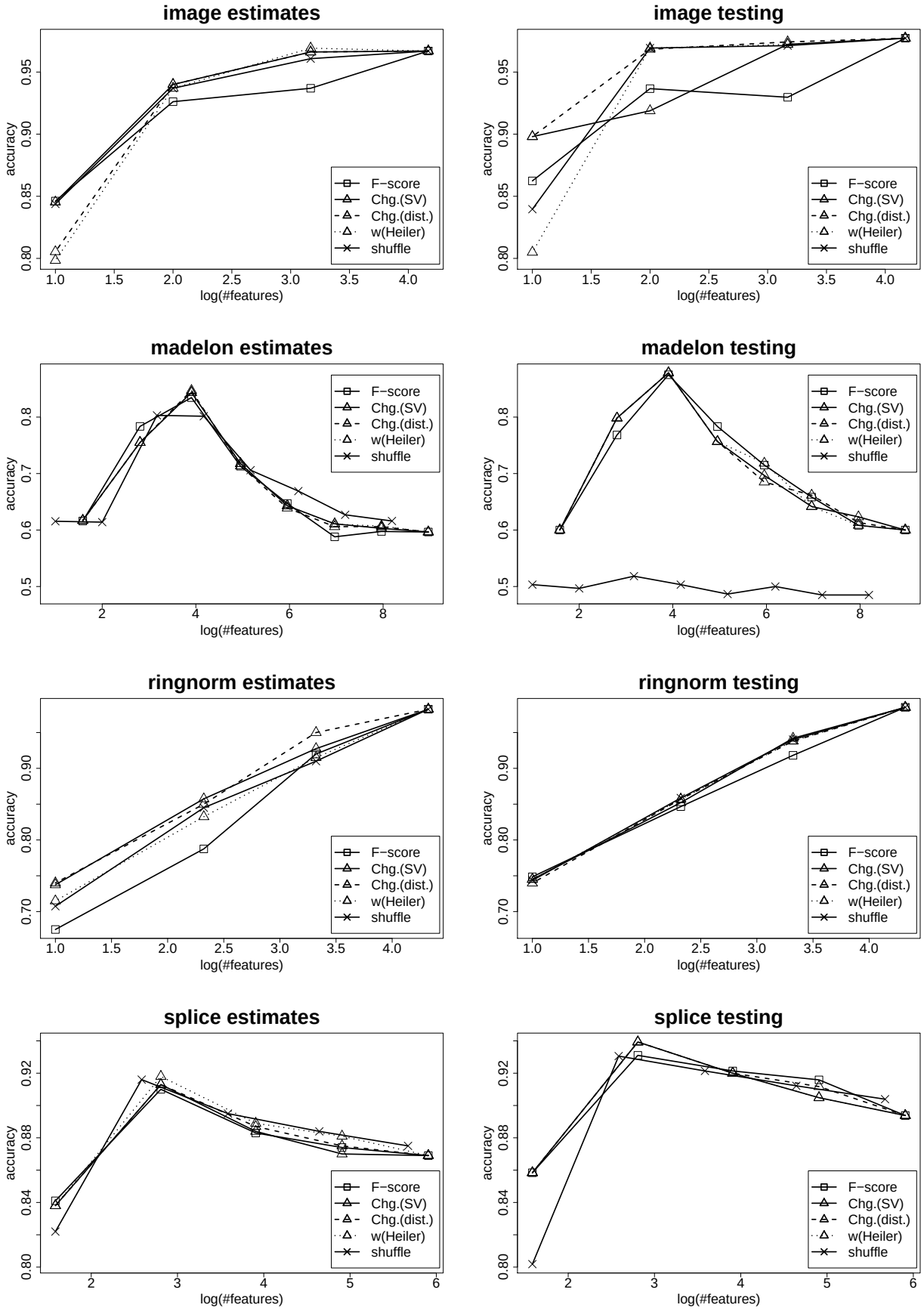


Figure 6.3: Comparisons of CV accuracy and testing accuracy against log number of features between feature ranking methods.

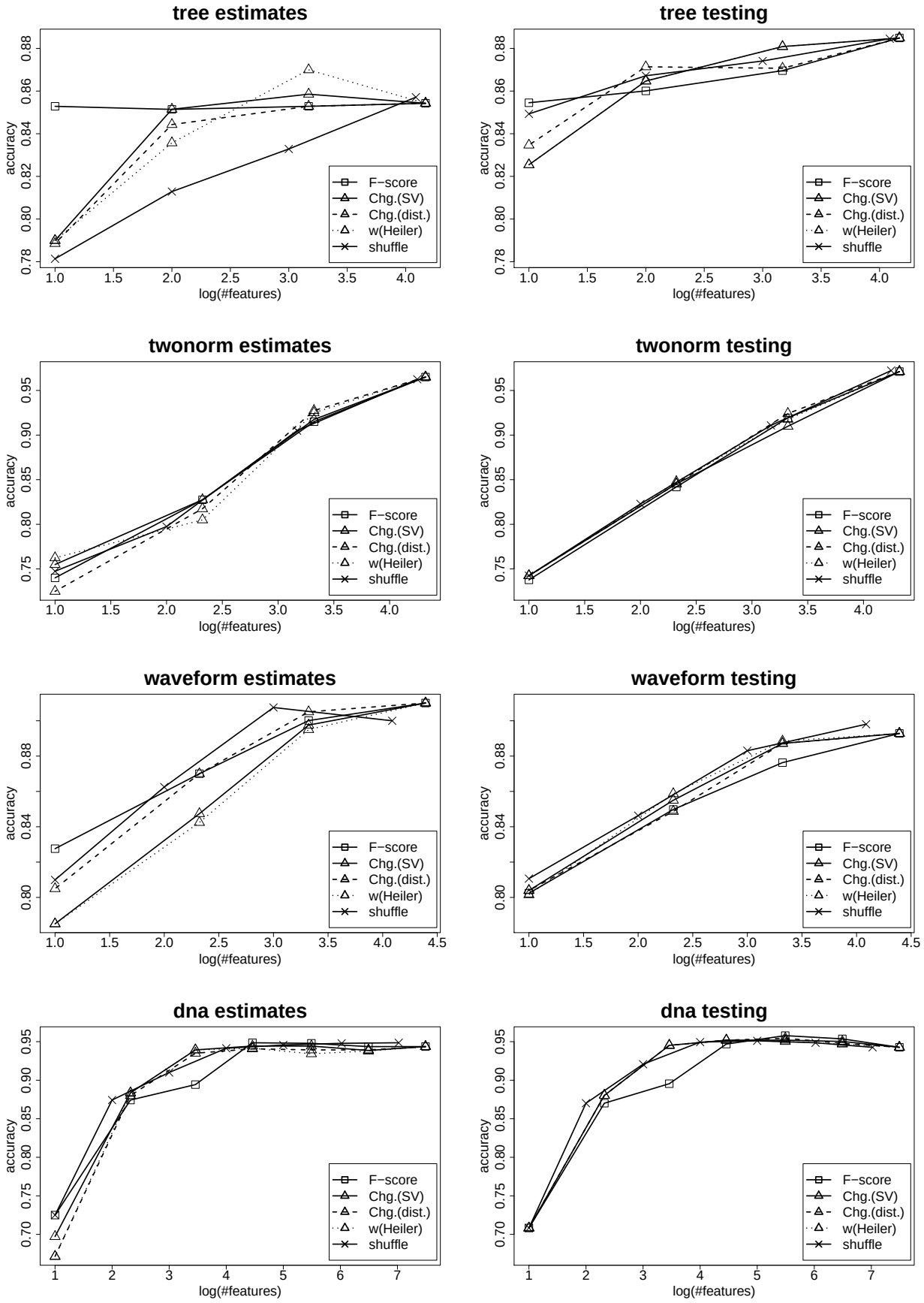


Figure 6.4: Comparisons of CV accuracy and testing accuracy against log number of features between feature ranking methods.

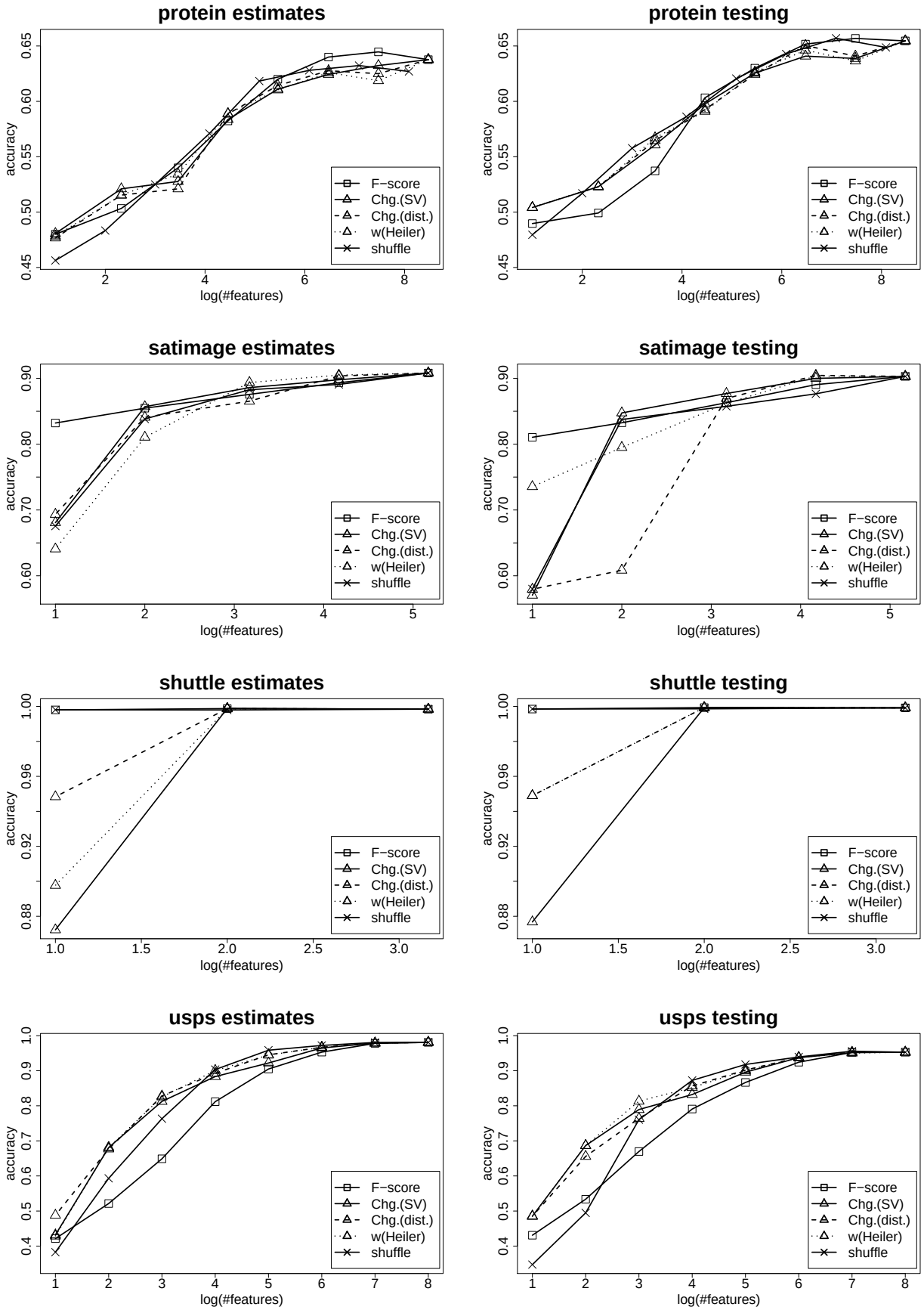


Figure 6.5: Comparisons of CV accuracy and testing accuracy against log number of features between feature ranking methods.

CHAPTER VII

Discussion and Conclusions

We have investigated several feature selection and feature scaling methods. The feature selection methods discussed in this thesis can be categorized into three kinds: the statistical score, the gradient of the decision values, and random shuffling on features. The first one is not related to the SVM while the second and the third rely on its performance.

From experiments we see that there is no method that is best for all data sets. Sometimes there is no significant difference on the performance. In addition, properties of data sets may affect the performance. For example, for madelon random shuffling on features performs very bad, but other methods they all successfully select good features. Therefore it is hard to find the best feature selection method.

Beside the prediction performance, training time is also an issue. F-score is very simple and efficient to calculate. On the other hand, approaches of using the change of the decision values (Section 5.2.2), and the normal vector of the decision boundary (Section 5.1) have to train a good SVM model with parameter selection. For strategies by observing the change of the distribution and the random shuffling on features they even spend more time. From the aspect of the training time, these two methods are not practical when the number of feature is large.

For the data sets used in this thesis we find that F-score performs well when the number of features is large, and for small data the two methods using the gradient of the decision values on support vectors are slightly better. Also for large data the F-score method is faster. In conclusion, the F-score criterion is good enough for feature selection. If the data is not too large and the training time is reasonable, methods that use the gradient of the decision values on support vectors can also be considered.

BIBLIOGRAPHY

[1] R. R. Bailey, E. J. Pettit, R. T. Borochoff, M. T. Manry, and X. Jiang. Automatic recognition of usgs land use/cover categories using statistical and neural networks classifiers. In *SPIE OE/Aerospace and Remote Sensing*, Bellingham, WA, 1993. SPIE.

[2] C. L. Blake and C. J. Merz. UCI repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, Irvine, CA, 1998. Available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[3] B. 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.

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

[5] C.-C. Chang and C.-J. Lin. IJCNN 2001 challenge: Generalization ability and text decoding. In *Proceedings of IJCNN*. IEEE, 2001.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[7] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46:131–159, 2002.

[8] Y.-W. Chen and C.-J. Lin. Combining svms with various feature selection strategies. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature extraction, foundations and Applications*. Springer, 2004.

[9] W. Chu, S. Keerthi, and C. Ong. Bayesian trigonometric support vector classifier. *Neural Computation*, 15(9):2227–2254, 2003.

[10] K.-M. Chung, W.-C. Kao, C.-L. Sun, L.-L. Wang, and C.-J. Lin. Radius margin bounds for support vector machines with the RBF kernel. *Neural Computation*, 15:2643–2681, 2003.

[11] R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(05):1105–1114, 2002.

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

[13] R. A. Fisher. The use of multiple measurements in taxonomic problem. *Annals of Eugenics*, 7:179–188, 1936.

[14] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[15] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[16] M. Heiler, D. Cremers, and C. Schnörr. Efficient feature subset selection for support vector machines. Technical Report 21, University of Mannheim, Germany, Department of Mathematics and Computer Science, Computer Vision, Graphics, and Pattern Recognition Group, D-68131 Mannheim, Germany, 2001.

[17] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, 2003.

[18] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of International Conference on Machine Learning*, 1999.

[19] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994. Journal version in AIJ, available at http://citeseer.nj.nec.com/13663.html.

[20] S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

[21] R. Kohovi and G. H. John. Wrappers for feature subset selection. *Artificial Intellgence*, 97(1-2):273–324, 1997.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. MNIST database available at `http://yann.lecun.com/exdb/mnist/`.

[23] A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2/3:18–22, December 2002.

[24] C.-J. Lin. *A Guide to Support Vector Machines.*

[25] C.-J. Lin. Formulations of support vector machines: a note from an optimization point of view. *Neural Computation*, 13(2):307–317, 2001.

[26] H.-T. Lin and C.-J. Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2003.

[27] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In J. W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 359–367, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[28] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, New York, 1992.

[29] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Prentice Hall, Englewood Cliffs, N.J., 1994. Data available at `http://www.ncc.up.pt/liacc/ML/statlog/datasets.html`.

[30] S. Perkins, K. Lacker, and J. Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.

[31] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, Cambridge, MA, 2000. MIT Press.

[32] D. Prokhorov. IJCNN 2001 neural network competition. Slide presentation in IJCNN'01, Ford Research Laboratory, 2001. `http://www.geocities.com/ijcnn/nnc_ijcnn01.pdf` .

[33] G. Rätsch. Benchmark data sets, 1999. Available at `http://ida.first.gmd.de/~raetsch/data/benchmarks.htm`.

[34] V. Svetnik, A. Liaw, C. Tong, and T. Wang. Application of Breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. In F. Roli, J. Kittler, and T. Windeatt, editors, *Proceedings of the 5th International Workshopon Multiple Classifier Systems*, Lecture Notes in Computer Science vol. 3077., pages 334–343. Springer, 2004.

[35] V. Svetnik, A. Liaw, C. Tong, and T. Wang. Application of breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. In *Multiple Classifier Systems*, pages 334–343, 2004.

[36] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

[37] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, 2000.

[38] J.-Y. Wang. Application of support vector machines in bioinformatics. Master's thesis, Department of Computer Science and Information Engineering, National Taiwan University, 2002.

[39] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *Advances in Neural Information Processing Systems*, volume 12, pages 668–674, 2000.

[40] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.

[41] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. Technical report, 2003.