Customizing kernels in Support Vector Machines

by
Zhanyang Zhang

A thesis presented to the
University of Waterloo
in fulfilment of the requirement for the degree of
Master of Mathematics
in
Statistics
Waterloo, Ontario, Canada, 2007

Contents

Acknowledgements

I wish to sincerely thank my supervisor Professor Paul Marriott for his guidance in defining the thesis and encouraging independent thought during this research. I also want to thank Professor Mu Zhu for the great advice he provided me. Lastly, I want to thank my parents and my girl friend Ting Lu as well, who have always been so supportive to me during my study at Waterloo.

Abstract

Support Vector Machines have been used to do classification and regression analysis. One important part of SVMs are the kernels. Although there are several widely used kernel functions, a carefully designed kernel will help to improve the accuracy of SVMs. We present two methods in terms of customizing kernels: one is combining existed kernels as new kernels, the other one is to do feature selection.

We did theoretical analysis in the interpretation of feature spaces of combined kernels. Further an experiment on a chemical data set showed improvements of a linear-Gaussian combined kernel over single kernels. Though the improvements are not universal, we present a new idea of creating kernels in SVMs.

Chapter 1

A Review of Support Vector Machines

Support Vector Machines (SVMs) have become a popular tool in recent years in the literature of statistical learning, classification and pattern recognition. Many scientists and researchers have been working on both the theory and application of SVMs. They has been successfully applied in areas such as hand-written character recognition, text categorization, computer vision, and bioinformatics. And the algorithm itself has been improved significantly compared to the original idea due to Vapnik and etc.

The earliest SVM was a hyperplane classifier which separates the training data in an n-dimensional input space according to a certain separated criterion. Different criterions will lead to some variation in the algorithms of SVMs. For instance, hard margin criterion leads to maximal margin SVMs; soft margin ends up wtih soft margin SVMs. A special case is when the criterion is chosen to be some particular loss function, then the analysis can be extended to regression. Detailed definition of different SVMs will be discussed during the following chapters.

Later kernel functions are introduced to reproduce feature spaces in which an optimal hyperplane classifier is searched for just as the original idea of SVMs. It turns out that the algorithm of kernel Support Vector Machines is formally similar to the original one. According to Mercer's theorem [11], a function has to satisfy several properties to be a kernel function. Every kernel function implicitly

determines a feature space with probably a higher dimension. Frequently it is difficult to find a reasonable linear classifier in the original input space, but we can expect the existence of a linear classifier in a certain feature space determined by some chosen kernel. The power of kernels lies in the fact that they allow the flexibility of constructing various of non-linear relationships among data points of the original input space to accommodate particular data structures.

The implementation of SVMs is a quadratic programming problem, for which plenty of algorithms are available, such as Sequential Minimal Optimization (SMO). The solution of the parameters in support vector machines turns out to be sparse, indicating only part of the training data are taking effect in determining the optimal classifier. Those data points are called *support vectors*.

1.1 Historical development of Support Vector Machines

The concept of Support Vector Machines was first introduced by Vladimir Vapnik et al [3]. at the computational learning theory conference held in 1992. In his paper, he proposed both the theoretical features and application of *maximal margin classifier*. He also introduced the concept of *support vectors* for a hyperplane classifier. However the main features of Support Vector Machines had been around since the 1960s. Duda and Hart [14] discussed large margin hyperplane in their book published in 1973. The use of kernels was discussed by Wahba [8], Aronszajn [2], Poggio [18] and others. In a 1964 paper, Aizermann et al. [1] talked about the geometric interpretation of kernels as inner products in a feature space. Mangasarian [13] in his 1965 paper discussed the optimization techniques which was similar to the method used in Vapnik's 1992 paper. Smith [7], Bennett and Mangasarian [12] further proposed the slack variables technique used in linear and nonlinear separation problems in 1960s. Those important previous findings led to Vapnik's revolutionary paper of 1992. In 1995 Cotes and Vapnik [5] advanced the maximal margin classifier presented in 1992 to *soft margin classifier* in order to deal with more noisy data. The analysis was extended to support vector regression in the same year at Vapnik's book [19].

During the late 1990s, SVMs had aroused many people' interest. People from different disciplines such as statistics, optimization, functional analysis, and com-

puter science worked together to contribute to the development of SVMs. Weston and Watkins[10] in 1999 extended the analysis to multi-class casees. Smola, Scholkopf and Mueller[17] investigated support vector machines from a new perspective by defining a general cost function. In 1998, Scholkopf, Smola, et al[15]. proposed a new algorithm called *v-Support Vector Machines* which is similar to *soft margin SVMs*. Critianini and Taylor's book [6] published in 1999 is a first comprehensive introduction of SVMs.

In the last several years, SVMs have been successfully applied in various of fields such as gene expression and digit recognition. Gene expression data comes from DNA microarrays. A DNA microarray (also commonly known as gene or genome chip, DNA chip, or gene array) is a collection of microscopic DNA spots attached to a solid surface, such as glass, plastic or silicon chip forming an array.

A certain pattern of the DNA sequence may indicate a particular biological expression or disease, for instance,the appearance of a certain cancer. Categorization methods are needed here to distinguish different types of genes according to their expression. Different from traditional data sets, gene expression data sets usually have very large dimension; the data itself is usually very unbalanced (many more negative instances than positive instances. And traditional classification methods might not perform well on such data. However, SVMs are believed to be capable to do the job. Brown et al. [4] gives a good use of SVMs applied to gene expression data for classifying unseen genes.

Another successful real world application of SVMs is the hand-written digit recognition problem. The original problem is motivated by the need of US Postal Service to automate sorting mail using the hand-written Zip codes. The data information is encoded in such a way: each digit is located at a 16×16 matrix and the information is recorded in every single grid (the dimension of the data is then 256). The number of observations in the training set is 7291, with every record a 257 (256 predictor plus one categorical response) dimensional vector representing a digit between 0 and 9. The number of observations in the test set is 2007. Several multi-class SVMs have been tried on this data set. The training set is reported to be totally separable with a maximal margin SVM with kernel function chosen to be polynomial having degree 3. The results of experiments have been reported in a series of papers by Burges, Cotes, Scholkopf, Vapnik etc. and summarized in

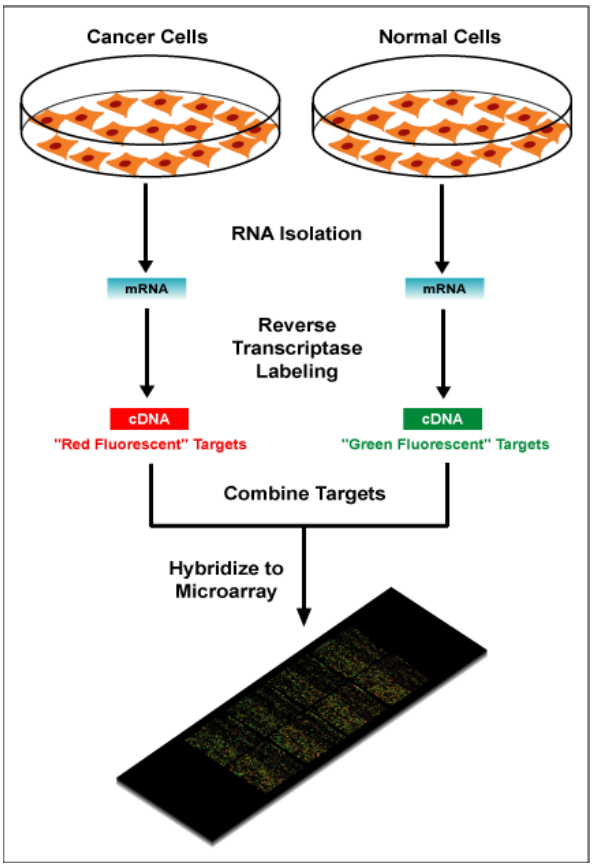

Figure 1.1: *An example of getting DNA microarray data. The first step is isolate mRNA information from two types of cells. This step is usually called RNA isolation. The next step is called Reverse Transcriptase Labeling, meaning to reverse the information back to cDNA and label it with two different fluorochrome according to the different types. The final step is to attach the DNA information to a solid surface such as a silicon chip. These microarrays can be used to identify disease genes by comparing gene expression in diseased and normal cells. Source: http://en.wikipedia.org/wiki/Dnamicroarray.*

[20].

1.2 Maximal Margin Support Vector Machines

Maximal Margin SVMs are the earliest SVMs. The name comes from using a maximal margin as the optimization criterion in model training. Before going further to show how the problem is presented and solved, it is necessary to look at some mathematical theories first.

1.2.1 Mathematical setups

In a classification problem, a set of collected data $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$, is called training set (A data set on which the learning algorithm is trained). $\mathcal{X} = \{x_1, x_2, ...x_l\}$ is usually called the p-dimensional predictors (covariances, explanatory variables), and $\mathcal{Y} = \{y_1, y_2, ...y_l\}$ is referred as the labels (response variables). For instance, in the USPS data discussed above, the 16×16 matrix entries will be 256-dimensional predictors. The categorical variable valued from 0 to 9 indicating the digit value is the response. Typically, $\mathcal{Y}$ takes values in unordered sets in classification problems. For instance, $\mathcal{Y} = \{-1, +1\}$ gives a binary classification.

The solving of a classification problem is to search for an optimal classifier (Optimal means minimizing or maximizing some criterion) $f \in \mathcal{F}$, mapping from $\mathcal{X}$ to $\mathcal{Y}$, where $\mathcal{F}$ is a certain function class.

$$f : \mathcal{X} \longmapsto \mathcal{Y}$$

One common function class is the linear function class. However, such a class may not be large enough for all kinds of classification problems. One way of introducing non-linear classifiers is to map the input space to a feature space (also called embedding), then search for a linear relation in the embedding space. Typically, a function Φ maps the original p-dimensional predictors $\mathcal{X}$ into a m-dimensional feature space $\mathcal{X}^*$.

$$\Phi : \mathcal{X} \longmapsto \mathcal{X}^*$$

$$\mathcal{X} \subseteq \mathbb{R}^p, \mathcal{X}^* \subseteq \mathbb{R}^m$$

f is chosen to be a linear function in the feature space $\mathcal{X}^*$:

$$f(x) = w^T \Phi(x) + b$$

$$\text{where } w = (w_1, w_2, ..., w_m) \in \mathbb{R}^m \text{ and } b \in \mathbb{R}^1.$$

This function is chosen such that this hyperplane will optimally separate data points in the feature space. Again, being optimal depends on what optimization criterion is chosen.

In Support Vector Machines, the chosen optimization criterion involve inner products of embedded vector $\langle \Phi(x_i), \Phi(x_j) \rangle$, and the estimate of the weight vector $w = (w_1, w_2, ..., w_m)$ is a linear combination of embedded vector $\Phi(x_i), i = 1, 2, ...l$

$$\hat{w} = \sum_{i=1}^{l} c_i \Phi(x_i)$$

Where $c_i \in R, i = 1, 2, ..l$ are the coefficients. Under such a circumstance, the optimal classifier is :

$$\widehat{f(x)} = \sum_{i=1}^{l} c_i \langle \Phi(x_i), \Phi(x) \rangle + \hat{b}$$

So it's important to introduce an inner product function taking values on $\mathcal{X}^* \times \mathcal{X}^*$. A kernel function $K(\cdot, \cdot)$ in SVMs is defined to be an inner product in the embedding feature space:

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

It should be noted that in SVMs, the form of the chosen embedding function Φ need not to be known. Once a kernel function is specified, the embedding function is implicitly determined. It is the kernel function that characterizes a certain SVM when the optimization criterion is given. As a similarity measure in the embedded space, kernel functions is the essence of SVMs. However, a function has to satisfy several conditions to be a kernel function.

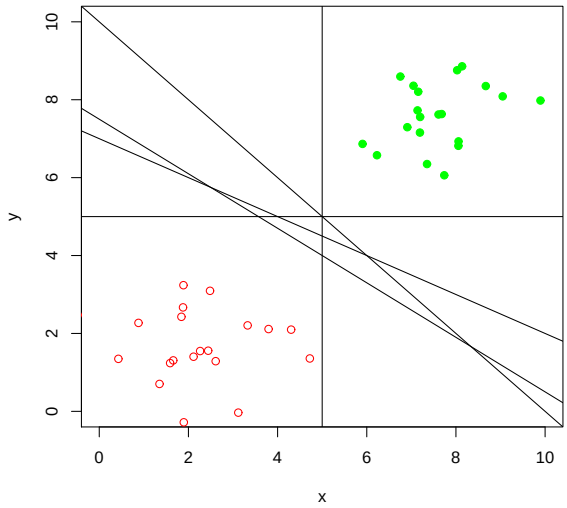

Figure 1.2: *Possible 5 linear classifiers which separate two different classes completely in a linear separable data set. The data set is simulated from two groups of Gaussian. There are infinitely many linear functions that could make total separation on this training set. Thus, it's important to choose an "optimal" one according to some optimization criterion. For instance, if we choose the margin of a classifier as the criterion and aim to maximize it, we will end up with a SVM classifier.*

1.2.2 The Maximal Margin Problem

Consider the simplest case, $\mathcal{Y} = \{-1, +1\}$, indicating a binary response. The reason why Y is set to be $\{-1, +1\}$ instead of $\{0, 1\}$ is to better represent the optimization problem. Suppose the training set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$ is *linear separable*, which means there exists a hyperplane (w, b) and a constant $c > 0$, such that

$$y_i(w^t x_i + b) > c$$

for all $i = 1, 2, ..., l$. No complicated feature space is induced and the projection function Φ is chosen to be the identity since the data set itself can be separated by a hyperplane. The *margin* M of a hyperplane classifier $f(x) = w^t x + b$ is defined to be the minimal one among all the distances of data points to the hyperplane.

$$M = \min_i d_i$$

where d_i is the distance of the ith observation to the hyperplane, $i = 1, 2, ...l$

$$d_i = \frac{y_i(w^t x_i + b)}{||w||}$$

For a point (x_i, y_i), its distance to a hyperplane $w^t x + b = 0$ can be computed in the following way:
Let x^* denote the corresponding foot of perpendicular. We know that

$$d_i = ||x_i - x^*||$$

In the other hand, we know the direction $x_i - x^*$ is perpendicular to the hyperplane $w^t x + b = 0$. So $x_i - x^*$ must have the same direction with w, and we can assume:

$$x_i - x^* = kw$$

where $k \in R^1$ is a constant. Because x^* is on the hyperplane, we have

$$\begin{aligned} w^t x^* + b &= 0 \\ w^t(x_i - kw) + b &= 0 \\ w^t x_i + b - kw^t w &= 0 \end{aligned}$$

Thus we have

$$k = \frac{w^t x_i + b}{w^t w} = \frac{w^t x_i + b}{||w||^2}$$

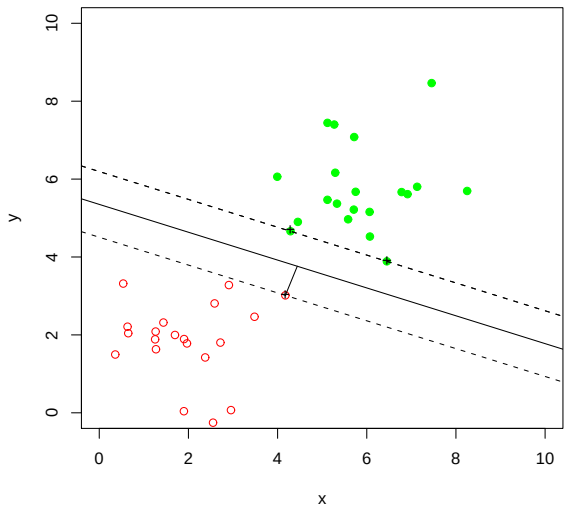

Figure 1.3: *Maximal Margin Classifier for a linear separable data set. Points with "+" are support vectors. The data is simulated as two groups of Gaussian with different centers. The total number of training set is 40 with only 3 support vectors. The margin of this classifier is the distance from any of the three support vectors to the hyperplane (Those 3 distances are equal.)*

Since $x_i - x^* = kw$, we have:

$$d_i = ||x_i - x^*|| = |k|||w|| = \frac{|w^t x_i + b|}{||w||^2}||w|| = \frac{|w^t x_i + b|}{||w||}$$

Note the classifier is assumed to be capable of separating the two classes completely. This means there exists a constant $c > 0$, such that $y_i(w^t x_i + b) > c$ for all $i = 1, 2, ..., l$. So for those data labeled with +1, $(w^t x_i + b) > 0$, and for those labeled with -1, $(w^t x_i + b) < 0$. Thus we can claim that $|w^t x_i + b| = y_i(w^t x_i + b)$. Further, we have

$$d_i = \frac{|(w^t x_i + b)|}{||w||} = \frac{y_i(w^t x_i + b)}{||w||}$$

For a linearly separable training set, there is infinite number of separating hyperplane (See figure 1.2), among which the Maximal Margin Classifier gives the largest margin. This optimization problem can be presented as follows:

$$\max_{(w,b)} M$$

subject to

$$y_i(w^t x_i + b) \geq 1 \qquad (i = 1, 2, ...l)$$

Since the actual value of the positive constant c does not effect the problem, it can be set to 1. According to the constraints we see that $d_i \geq \frac{1}{\|w\|}$, i=1,2,...l.
The margin M is the minimum of d_i, thus $M \geq \frac{1}{\|w\|}$. If (w^*, b^*) is the hyperplane which has the largest margin M^*, we must have points on the boundary satisfying $y_i(w^{*t}x + b^*) = 1$ (See figure 1.3). Under such a case, $M^* = 1/\|w^*\|$. So if we want to maximize M under constraints, we can simply minimize $\|w\|$ under the same contraints, or equivalently minimize $\frac{1}{2}\|w\|^2$ for convenience. So another presentation of the maximal margin problem is given as:

$$\min_{(w,b)} \frac{1}{2}\|w\|^2 \tag{1.1}$$

subject to

$$-(y_i(w^t x_i + b) - 1) \leq 0 \qquad (i = 1, 2, ...l)$$

The constraints on (w, b) are linear inequalities, which define a convex region, usually called feasible region. The objective function is quadratic and convex, so we have a quadratic convex optimization problem, for which plenty of algorithms are available. However, when it comes to a real algorithm, problems with equality constraints are much easier to handle with. It is usual for such a problem to transform it into its dual presentation. Another very important advantage of solving the dual problem rather than primal is that the objective function in the dual problem has a nice inner product form. This form will also make it easier to further introduce kernel functions in the objective. In order to see how the dual problem arises, let us first look at some classical optimization theory.

1.2.3 Optimization theorem

In the following, we are going to quote and prove several important optimization theorem which Support Vector Machines stand on.

Fermat's Theory (Optimization with no constraints) Suppose $f : \Theta \to R^1$, where $\Theta \subset R^n$ is open and $f \in C^1$, a necessary condition for θ^* to be a minimum of $f(\theta)$ is that

$$\nabla f(\theta^*) = 0$$

If f happens to be a convex function, then the condition is also sufficient.

Proof (Necessary condition):
Suppose $\nabla f(\theta^*) > 0$, according to the definition of

$$\|\nabla f(\theta^*)\|^2 = lim_{h\to 0^+} \frac{f(\theta^*) - f(\theta^* - h\nabla f(\theta^*))}{h}$$

Thus we have $f(\theta^*) - f(\theta^* - h\nabla f(\theta^*)) > 0$ which contradicts with the minimum of θ^*. The same argument will follow when $\nabla f(\theta^*) < 0$.
End of Proof.

Proof (Sufficient condition):
Prove with contradiction. Suppose there exist another $\theta \in \Theta$, such that $f(\theta^*) > f(\theta)$, let $v = \theta - \theta^*$thus:

$$0 = \nabla f(\theta^*) \cdot v = lim_{h\to 0^+} \frac{f(\theta^* + hv) - f(\theta^*)}{h}.$$

Because $f(\theta^* + hv) = f((1-h)\theta^* + h\theta)$ and f is a convex function, we have

$$f(\theta^* + hv) = f((1-h)\theta^* + h\theta) \leq (1-h)f(\theta^*) + hf(\theta)$$

substitute the relationship, there will be a contradiction:

$$0 \leq lim_{h\to 0^+} \frac{h(f(\theta) - f(\theta^*))}{h} < 0.$$

End of Proof.

Now look at the following generalized optimization problem:
minimize

$$f(\theta)$$

subject to

$$\begin{cases} g_i(\theta) \leq 0 (i = 1, 2, ...l) \\ \theta \in \Theta \end{cases}$$

Usually this problem is called the primal problem. For a problem with constraints like the above one, we can consider the so called Lagrangian function:

$$L(\theta, \alpha) = f(\theta) + \sum_{i=1}^{l} \alpha_i g_i(\theta)$$

Those $\alpha_i \geq 0$ are the Lagrangian multipliers. Now we can define the corresponding dual problem:

$$\max_{\alpha} \inf_{\theta \in \Theta} L(\theta, \alpha)$$

Subject to

$$\alpha \geq 0$$

Further we can define:

$$\tilde{L}(\alpha) = \inf_{\theta \in \Theta} L(\theta, \alpha)$$

Then the dual problem is to maximize $\tilde{L}(\alpha)$ for all non-negative α_i. Next, we will show the two optimization problems (one minimizing $f(\theta)$, the other maximizing $\tilde{L}(\alpha)$) have the following relationship:

Theorem: Weak duality If θ is a feasible solution of the original primal problem and α is a feasible solution of the Lagrangian dual problem, then $\tilde{L}(\alpha) \leq f(\theta)$.

Proof:

$$\begin{aligned} \tilde{L}(\alpha) &= \inf_{\theta \in \Theta} L(\theta, \alpha) \\ &\leq L(\theta, \alpha) \\ &= f(\theta) + \sum_{i=1}^{l} \alpha_i g_i(\theta) \\ &\leq f(\theta) \end{aligned}$$

Since

$$\alpha_i \geq 0, g_i(\theta) \leq 0$$

End of Proof.

If it turns out that if there exists some θ^*, α^*, such that $f(\theta^*) = \tilde{L}(\alpha^*)$, provided that $\alpha^* \geq 0, g_i(\theta^*) \leq 0$, then θ^*, α^* solve the original primal problem and the Lagrangian dual problem. A necessary condition for such a case is that $\alpha^* g_i(\theta^*) = 0$ for every $i = 1, 2, 3, ...l$. The reason is if there exist some j such that $\alpha^* g_j(\theta^*) < 0$, then $\tilde{L}(\alpha^*, \beta^*)$ would be strictly less than $f(\theta^*)$. This condition is often called Karush-Kuhn-Tucker complementarity condition. In order to ensure that solving the primal problem is equal to solving the dual problem, we need the following

Kuhn-Tucker Theorem:

The Kuhn-Tucker Theorem Suppose that $\Theta \subset R^n$ is open, $f, g_1, g_2 ... g_l : \Theta \to R^1$ are C^1, f is a convex function, and the feasible region

$$D = \{\theta \in \Theta : g_i(\theta) \leq 0, i = 1, 2, 3, ...l\}$$

is also convex. Necessary and sufficient conditions for a normal point $\theta^* \in D$ to be an optimum are the existence of $\alpha_i \in R^1, \alpha_i \geq 0, i = 1, 2, 3...., $ such that

$$\nabla f(\theta^*) + \sum_{i=1}^{l} \alpha_i \nabla g_i(\theta^*) = 0$$

and

$$\alpha_i g_i(\theta^*) = 0, i = 1, 2, 3....l$$

Proof (Sufficient conditions)
We will prove it by contradiction. Suppose there exists a θ such that $f(\theta) < f(\theta^*)$. Let $v = \theta - \theta^*$, according to the definition of derivatives,

$$\begin{aligned} \nabla f(\theta^*) \cdot v &= \lim_{h \to 0^+} \frac{f(\theta^* + hv) - f(\theta^*)}{h} \\ &= \lim_{h \to 0^+} \frac{f(\theta^* + h(\theta - \theta^*)) - f(\theta^*)}{h} \end{aligned}$$

Because f is convex,

$$\begin{aligned} \nabla f(\theta^*) \cdot v &\leq \lim_{h \to 0^+} \frac{(1-h)f(\theta^*) + hf(\theta) - f(\theta^*)}{h} \\ &= \lim_{h \to 0^+} \frac{h(f(\theta) - f(\theta^*))}{h} \\ &= f(\theta) - f(\theta^*) \\ &< 0 \end{aligned}$$

On the other hand, for each i with $g_i(\theta^*) = 0$, we have

$$\begin{aligned} \nabla g_i(\theta^*) \cdot v &= \lim_{h \to 0^+} \frac{g_i(\theta^* + hv) - g_i(\theta^*)}{h} \\ &= \lim_{h \to 0^+} \frac{g_i(\theta^* + hv)}{h} \\ &\leq 0 \end{aligned}$$

Note $g_i(\theta^* + hv) = g_i(h\theta + (1-h)\theta^*)$, and $h\theta + (1-h)\theta^* \in D$ since D is convex. So we have $g_i(\theta^* + hv) \leq 0$, hence $\nabla g_i(\theta^*) \cdot v \leq 0$ for each i with $g_i(\theta^*) = 0$.
However, this will lead to a contradiction what we have known:

$$\nabla f(\theta^*) \cdot v = -\sum_{i=1}^{l} \alpha_i \nabla g_i(\theta^*) \cdot v \geq 0$$

Since $\alpha_i g_i(\theta^*) = 0$ for each i, the each term in the RHS would be other zero or non-negative.
Thus we have proved that θ^* must be the solution of the optimization problem.

The necessary condition (A sketch proof):
Suppose $\theta^* \in D$ is the optimum. Consider the following cone:

$$C = \{\sum_{i=1}^{l} \lambda_i \nabla g(\theta^*) | \lambda_i > 0\}$$

We aim to show that $-\nabla f(\theta^*)$ is in the polar of C, which means there exists $\lambda_i^* \geq 0, i = 1, 2, ..l$ such that

$$-\nabla f(\theta^*) = \sum_{i=1}^{l} \lambda_i \nabla g(\theta^*)$$

According to **Generalized Farkas Theorem** [16], if this is not true, there is some vector v in the polar of C, such that

$$-\nabla f(\theta^*) \cdot v > 0$$

Again the definition of derivatives indicate that this is a contrary of the assumption that θ^* is a minimizer.
End of proof: Kuhn-Tucker Theorem

Now guaranteed by the Kuhn-Tucker Theorem, we can state the following theorem which will ensure the safety of moving from primal to dual:

Theorem: Strong Duality Suppose that $\Theta \subset R^n$ is open, $f, g_1, g_2 ... g_l : \Theta \to R^1$ are C^1, f is a convex function, and the feasible region

$$D = \{\theta \in \Theta : g_i(\theta) \leq 0, i = 1, 2, 3, ...l\}$$

is also convex, then the duality gap is zero (the primal and dual have the same optimal value).

Proof: Suppose θ^* is the optimal solution of the primal problem. According to the necessary condition of Kuhn-Tucker Theorem, there exists $\alpha_i^* \geq 0, i = 1, 2, 3...l$, such that

$$\nabla f(\theta^*) + \sum_{i=1}^{l} \alpha_i^* \nabla g_i(\theta^*) = 0$$

According to the sufficient condition of Fermat's theorem, when $\alpha = \alpha^*$, θ^* minimise the Lagrange Primal $L(\theta, \alpha^*) = f(\theta) + \sum_{i=1}^{l} \alpha_i^* g_i(\theta)$.

Thus we have the following equation:

$$\tilde{L}(\alpha^*) = L(\theta^*, \alpha^*) = f(\theta^*) + \sum_{i=1}^{l} \alpha_i^* g_i(\theta^*)$$

By the KKT complementarity condition, we have $\alpha_i g_i(\theta^*) = 0, i = 1, 2, 3....l$ and $\tilde{L}(\alpha^*) = f(\theta^*)$.
According to the Weak Duality Theorem,

$$\sup_{\alpha \geq 0} \tilde{L}(\alpha) \leq f(\theta^*)$$

So α^* must be the optimal solution of the dual problem, and the values of the two optimization problem are equal:

$$L(\tilde{\alpha^*}) = f(\theta^*)$$

End of proof.

1.2.4 The solving of Maximal Margin SVMs

Now we know under some regularity conditions, the primal optimization problem(2) of maximizing the margin can be transformed into its dual presentation. Based the dual presentation, we can further introduce the *Lagrange Primal*:

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i [y_i(w^t x_i + b) - 1] \tag{1.2}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. The dual can be easily obtained by setting to zero the derivatives of the Lagrangian primal with respect to (w, b), and substituting the relations back to the Lagrangian. This is guaranteed by the convexity of the problem (thus ensured by the sufficient condition of Fermat's Theorem)

Take derivatives with respect to w, b and set them to zero:

$$\begin{aligned} \frac{\partial L_p}{\partial w} &= 0: \qquad w - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \\ \frac{\partial L_p}{\partial b} &= 0: \qquad \sum_{i=1}^{l} \alpha_i y_i = 0 \end{aligned}$$

Substituting them back to the Lagrange Primal, we have

$$\begin{aligned} L_p &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i [y_i(w^t x_i + b) - 1] \\ &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i w^t x_i - \sum_{i=1}^{l} \alpha_i y_i b + \sum_{i=1}^{l} \alpha_i \\ &= \frac{1}{2}\|x\|^2 - w^t(\sum_{i=1}^{l} \alpha_i y_i x_i) - (\sum_{i=1}^{l} \alpha_i y_i) b + \sum_{i=1}^{l} \alpha_i \end{aligned}$$

Substitute $w = \sum_{i=1}^{l} \alpha_i y_i x_i, \sum_{i=1}^{l} \alpha_i y_i = 0$, we have the following:

$$\begin{aligned} L_p &= \frac{1}{2}\|w\|^2 - w^t w - 0 + \sum_{i=1}^{l} \alpha_i \\ &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\|w\|^2 \\ &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \end{aligned}$$

Finally we optain the *Lagrange Dual* problem:

$$\max_{\alpha} L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{1.3}$$

subject to

$$\begin{cases} \sum_{i=1}^{l} \alpha_i y_i = 0 \\ \alpha_i \geq 0 (i = 1, 2, ...l) \end{cases}$$

The reason that $\sum_{i=1}^{l} \alpha_i y_i = 0$ appears in the constraints is that this relation have not been used in obtaining the objective function of the dual. Instead of optimizing with (w, b), L_D search for optimal α_is corresponding to every observation.

Let $\hat{\alpha}_i$ denote the optimal solution of the dual problem. By the Karush-Kuhn-Tacker complementarity condition:

$$\hat{\alpha}_i [y_i (w^t x_i + b) - 1] = 0, \qquad i = 1, 2, ...l$$

So for only those points on the boundary, namely $y_i(w^t x_i + b) = 1$, the corresponding α_i is non-zero. These data points, which completely determine the hyperplane classifier (w, b) , are called *support vectors*.
The optimal weight vector $\hat{w}$ is a linear combination of observations as shown before:

$$\hat{w} = \sum_{i=1}^{l} \hat{\alpha}_i y_i x_i = \sum_{i \in SV} \hat{\alpha}_i y_i x_i \tag{1.4}$$

where SV refers to the set of support vectors.
The intercept $\hat{b}$ is determined by $y^*(w^t x^* + b) = 1$, where (x^*, y^*) is an arbitrary support vector. Finally, the optimal hyperplane classifier is given as:

$$\widehat{f(x)} = \hat{w}^t x + \hat{b} = \sum_{i \in SV} \hat{\alpha}_i y_i \langle x_i, x \rangle + \hat{b}. \tag{1.5}$$

1.3 Soft Margin SVMs

Mostly the training sets are not linear separable, meaning that the above optimization problem is an ill-posed one. This motivates the introducing of *slack variables* ξ_i to presentation (2)(still considering binary classification with no feature space induced):

$$\min_{(w,b,\xi_i)} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} \xi_i \tag{1.6}$$

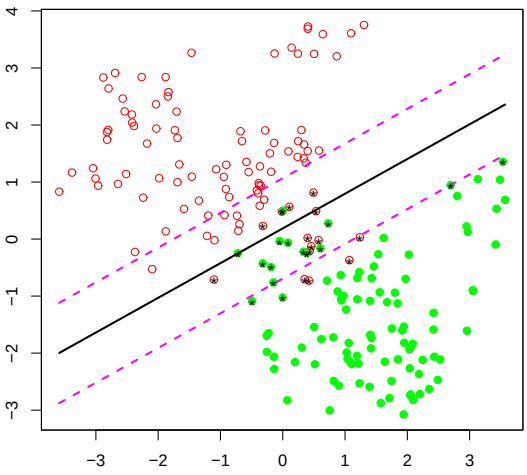

Figure 1.4: *An example of soft margin SVM. Since the data set is not linear separable, maximum margin SVMs can not be used here. The solid line is the fitted classification boundary:* $w^t x + b = 0$*; the dashed lines are contours of :* $w^t x + b = \pm 1$*. The marked points are support vectors. Notice the number of support vectors become significantly larger than maximum margin SVM.*

subject to

$$\begin{cases} y_i(w^t x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \\ i = 1, 2, ...l \end{cases}$$

$C > 0$ is often referred as the penalized constant or *cost*. It has to be determined by cross-validation (this is usually called parameter tuning). The *hard-Margin Classifier* only minimize the norm of w such that the hyperplane successfully separates all l samples in the training set. The slack variables ξ_i are introduced to allow for some points to be misclassified since no hyperplane can separate them all. Thus the cost parameter C is a trade-off between model complexity and the accuracy on a training set. Figure 2.4 shows a simulated example of a soft margin SVM.

The equivalent Lagrange Dual is given by:

$$\max_{\alpha} L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{1.7}$$

subject to

$$\begin{cases} \sum_{i=1}^{l} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \qquad i = 1, 2, ...l \end{cases}$$

The hyperplane classifier is still determined by $\alpha_i s$ by the same way:

$$\hat{w} = \sum_{i=1}^{l} \hat{\alpha}_i y_i x_i = \sum_{i \in SV} \hat{\alpha}_i y_i x_i$$

where SV corresponds to the subset with α_i non-zero. $\hat{b}$ is chosen by $y_i(w^t x_i + b) = 1$ with $0 < \alpha_i < C$. The optimal classifier is the same as (6):

$$\hat{f}(x) = \hat{w}^t x + \hat{b} = \sum_{i \in SV} \hat{\alpha}_i y_i \langle x_i, x \rangle + \hat{b}$$

1.4 SVMs with kernels

In most cases trying to find an optimal linear SVM in the original input space $\mathcal{X}$ is difficult. However, as discussed before, it is likely that a linear classifier will perform well in a feature space (the original input space mapped by a feature mapping function Φ). Figure 1.5 and figure 1.6 show an example where no

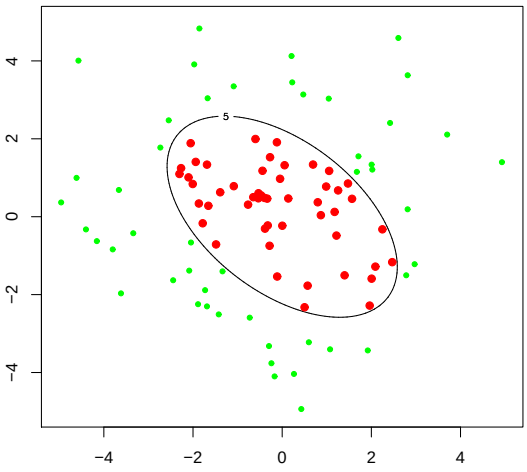

Figure 1.5: *An example which linear SVMs are not adequate. The solid ellipse line is the real boundary:* $x^2 + xy + y^2 = 5$. *Data is simulated according to this boundary: For any given point (x,y), if* $x^2 + xy + y^2 > 5$, *then it is marked as class 0 (the smaller points); otherwise it is marked as class 1 (the larger points). With such a distribution, any linear classifier will not do well in separating those two classes.*

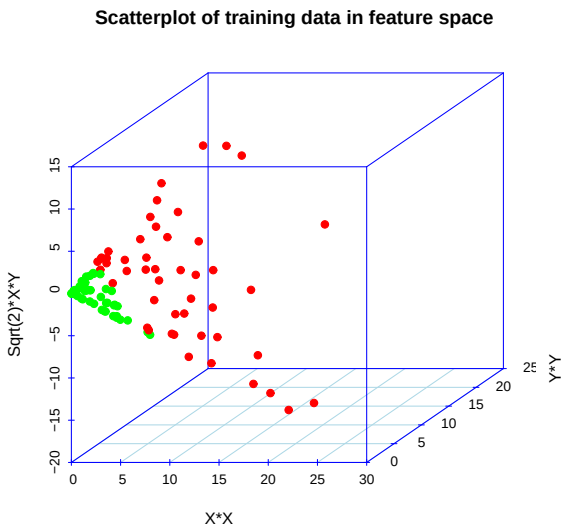

Figure 1.6: *Mapping the original training set to a 3-dimensional feature space where linear SVMs can perform well. As we can see from this figure, the separating plane can be searched in this 3-dimensional space. The mapping function takes the form of :* $\Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$.

linear SVMs can do well in the original space, but linear SVMs can work in the 3-dimensional feature space.

SVMs with induced feature spaces are often referred to as kernel SVMs. By introducing the feature space, we are actually searching for nonlinear relationship in the original input space. Figure 1.7 shows this non-linear relationship in an example using soft Margin SVMs with a Gaussian kernel. This simulation study is inspired by a visualization method introduced in the book by Hastie et. al. (2001) [9].

The nice thing about SVMs is that we can simply move from linear SVMs to kernel SVMs while keeping the general presentation formulas. Indeed, when mapping function is introduced, instead working on the points $(x_1, x_2, ...x_l)$, we are working on $(\Phi(x_1), \Phi(x_2), ..., \Phi(x_i))$. As a contrast to the presentation of linear SVMs in (8), the corresponding Lagrange Dual when introducing the feature space is then:

$$\max_{\alpha} L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle \tag{1.8}$$

subject to

$$\begin{cases} \sum_{i=1}^{l} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C, \qquad i = 1, 2, ...l \end{cases}$$

The w vector in feature space: $f(x) = w^t \Phi(x) + b$ is given by:

$$\hat{w} = \sum_{i=1}^{l} \hat{\alpha}_i y_i \Phi(x_i) = \sum_{i \in SV} \hat{\alpha}_i y_i \Phi(x_i)$$

where $\hat{b}$ is determined by $y_j(\hat{w}^t \Phi(x_i) + \hat{b} = 1$, or $y_j(\sum_{i \in SV} \hat{\alpha}_i y_i \langle \Phi(x_i), \Phi(x_j) \rangle + \hat{b}) = 1$ for some j satisfying with $0 < \alpha_j < C$. The optimal classifier $\widehat{f(x)}$ is:

$$\widehat{f(x)} = \hat{w}^t x + \hat{b} = \sum_{i \in SV} \hat{\alpha}_i y_i \langle \Phi(x_i), \Phi(x) \rangle + \hat{b} \tag{1.9}$$

Note expressions (9) and (10) only involve inner products of training points, so without explicitly giving the mapping Φ one can still train a SVM as long as the kernel function

$$K(x, z) = \langle \Phi(x), \Phi(z) \rangle$$

is specified. However, for a function $K(x, z)$ to be a kernel function, it first has to satisfy some properties. Obviously, K must be symmetric, because inner product

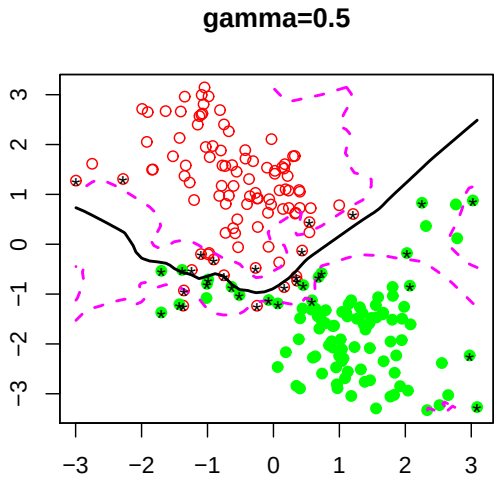

Figure 1.7: *An example of soft margin SVM with kernels. The Gaussian kernel:* $K(x,z) = \exp -\sigma\|x-z\|^2$ *is used. The solid line is the fitted classification boundary:* $w^t\phi(x)+b=0$*; the dashed lines are contours of :* $w^t\phi(x)+b=\pm 1$*. Parameters in the Gaussian kernel are set to be* $\sigma = 0.5, cost = 5$*.*

is symmetric: $K(x,z) = K(z,x)$.
For a given $K(x,z)$, the question is whether we can find out the associated feature mapping ϕ_j. Though ϕ_j is not needed with SVMs, specifying the form of ϕ_j will help us understand what kind of feature space we are in when searching for a linear classifier in that space.
Several widely used kernels are:

Linear Kernel	$K(x,z) = \langle x,z\rangle$,
Polynomial Kernel	$K(x,z) = (\langle x,z\rangle + c)^d$,
Gaussian Kernel	$K(x,z) = \exp(-\sigma\|x-z\|^2)$,
Sigmoid Kernel	$K(x,z) = \tanh(\gamma\langle x,z\rangle - \theta)$.

Except for the linear kernel, it is not so obvious why those other three would kernel functions. This question is answered if the associated feature mapping function ϕ is specified. For example, a polynomial kernel with degree 2 and offset 1 in a 2-dimensional space

$$K(x,z) = [x_1z_1 + x_2z_2) + 1]^2$$

can be actually expressed in the following form:

$$\begin{aligned} K(x,z) &= x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2 + 2x_1z_1 + 2x_2z_2 + 1 \\ &= (x_1^2,\ \sqrt{2}x_1x_2, x_2^2,\ \sqrt{2}x_1,\ \sqrt{2}x_2, 1)\cdot(z_1^2,\ \sqrt{2}z_1z_2, z_2^2,\ \sqrt{2}z_1,\ \sqrt{2}z_2, 1) \\ &= \phi(x)\cdot\phi(z) \end{aligned}$$

where the mapping function is given by

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

In other words, the polynomial kernel has implicitly defined a feature mapping function from R^2 to R^6.

In order to ensure that any given function $K(x,y)$ is a kernel function, namely, K can be expressed as a product of feature mappings, we need to introduce Mercer's theorem [11]:

Theorem (Mercer) *Let X ba a compact subset of R^n. Let $L_2(X)$ denote the L-2 function space on X. Suppose K is a continuous symmetric function such that the integral operator $T_K : L_2(X) \to L_2(X)$,*

$$(T_Kf)(\cdot) = \int_X K(\cdot,x)f(x)dx$$

is positive, which means

$$\int_{X\times X} K(x,z)f(x)f(z)dxdz \geq 0$$

for all $f \in L_2(X)$. *Then we can expand* $K(x,z)$ *in a uniformly convergent series (on* $X \times X$ *) in terms of eigen-functions* $\phi_j \in L_2(X)$, *normalized in such a way that* $\| \phi_j \|_{L_2} = 1$, *and positive associated eigenvalues* $\lambda_j > 0$,

$$K(x,z) = \sum_{i=1}^{\infty} \lambda_j \phi_j(x)\phi_j(z).$$

According to Mercer's theorem, a kernel function K has to satisfy: for any finite subset of X, the corresponding kernel matrix determined by $[K_{ij}]$ where $K_{ij} = K(x[i], x[j])$ is symmetric and positive semi-definite. For Mercer kernels, there always exist implicitly determined mapping functions so that the kernel can be presented as inner products of mapping functions. But the representation is not unique. For instance, we can choose the mapping function to be

$$\xi_j(x) = \sqrt{\lambda_j}\phi_j(x)$$

With ϕ_j, λ_j be the corresponding eigenvalue and eigenfunction of $K(x,z)$. This will ensure that SVMs are searched in some feature space. However, experiments show that non-Mercer Kernels also give good results sometimes. For example, the sigmoid kernel is not a Mercer one but is proved useful.
The choice of a kernel and its fine-tuning are essential in fitting SVMs. The class of kernel functions which can be chosen from is large. And there are many ways to produce new kernels with some widely used kernels. Especially,

Linear combinations of kernels is a kernel	$K(x,z) = \alpha K_1(x,z) + \beta K_2(x,z)$
Products of kernels is a kernel	$K(x,z) = K_1(z,z)^{\alpha} K_2(x,z)^{\beta}$

where $\alpha \geq 0, \beta \geq 0$.

Before we develop our strategies to customize kernels for a particular data structure, let's first investigate how those four widely used kernels look like in a simulated example.

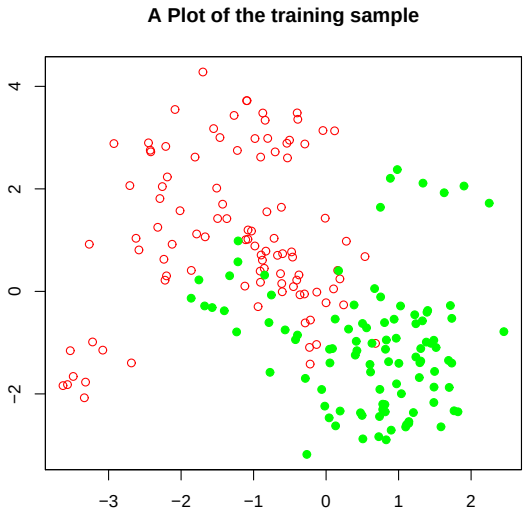

Figure 1.8: *A scatter plot of the training sample. Data is simulated as mixtures of Gaussian. One group centers at* $(-1, 1)$*, the other centers at* $(1, -1)$

1.4.1 Kernel visualization via a simple simulated experiment

In order to the see the behaviors of several widely used kernels in SVMs, we try to visualize them in a two dimensional space. The data is simulated in such the following way:
Step1: Generate two group of Bivariate Gaussian observations with each size 10, one center is at $(-1, 1)$, and the other center is at $(1, -1)$. Group1=$\{\mu_i^+, i = 1, ...10\}$, and Group2= $\{\mu_i^-, i = 1, ...10\}$. $\mu_i^+ \sim N((-1, 1), I), \mu_i^- \sim N((1, -1), I)$.

Step2: Each time randomly pick a center $\mu_j^+$from Group1, generate a Bivariate Gaussian observation $X_i^+ \sim N(\mu_j^+, \frac{1}{5}I)$. Keep doing this 100 times. In the same way, generate 100 Bivariate Gaussian from centers in Group2.

Step3: Generate a label vector Y with size $200 : Y[1 : 100] = 1; Y[100 : 200] = -1$. See figure 1.8 for the scatter plot.

Combine the simulated data $(X_i^+, Y_i), i = 1, ...100$ and $(X_i^-, Y_i), i = 100, ...200$ as training sample, fit 4 different SVMs corresponding to 4 common-used kernels:

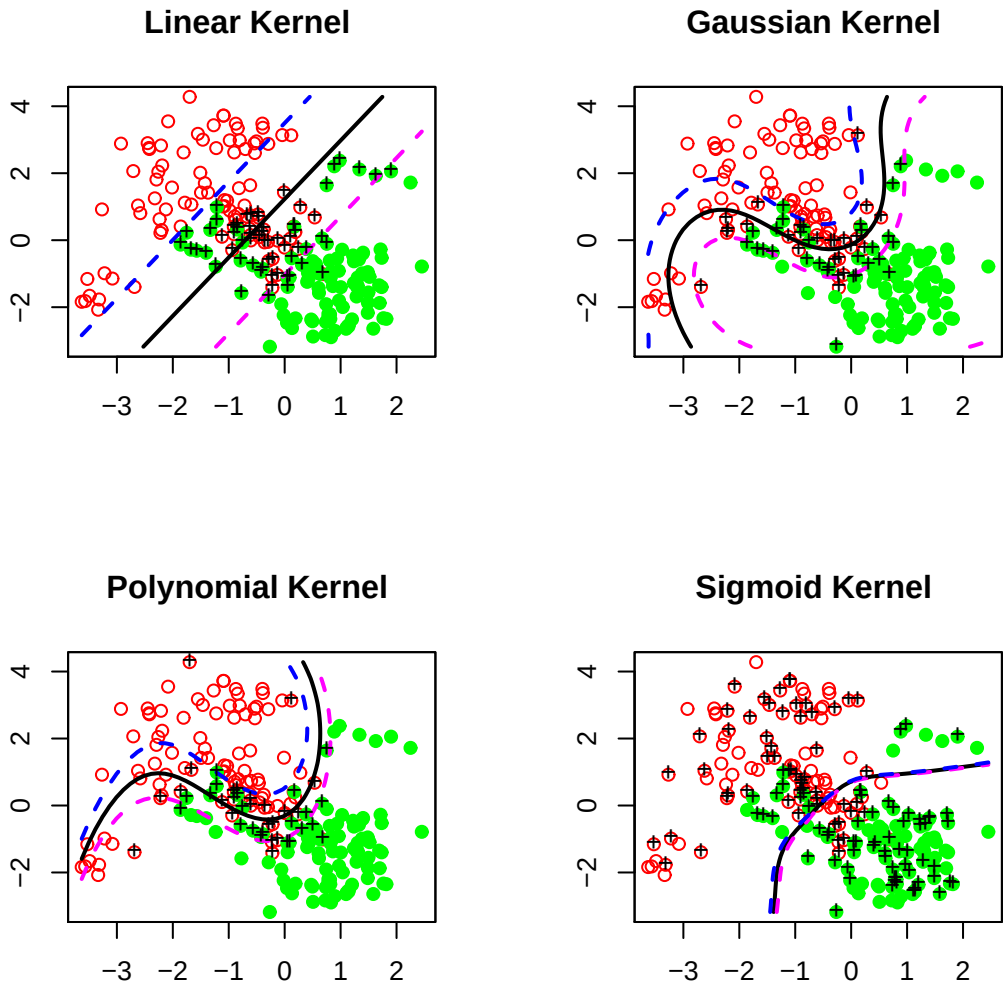

Figure 1.9: *Four different kernels in classification. The four corresponding kernels used are* $K_{linear}(x,z) = \langle x,z\rangle$, $K_{polynomial}(x,z) = (\langle x,z\rangle + 1)^3$, $K_{gaussian}(x,z) = \exp(\frac{-\|x-z\|^2}{2})$, *and* $K_{sigmoid}(x,z) = \tanh(3\langle x,z\rangle + 1)$. *The solid line is the classification boundary, a contour for* $\hat{f} = 0$. *The dashed lines are contours for* $\hat{f} = -1$, $\hat{f} = 1$. *Points with " + " are support vectors. The cost parameter C is set to be the default value 1 for all four kernels.*

Linear, Gaussian, Polynomial, and Sigmoid. Parameters are specified as follows:

$$\begin{aligned} K_{linear}(x,z) &= \langle x,z\rangle, \\ K_{poly}(x,z) &= (\langle x,z\rangle+1)^3, \\ K_{gaussian}(x,z) &= \exp(\frac{-\|x-z\|^2}{2},) \\ K_{sigmoid}(x,z) &= \tanh(3\langle x,z\rangle+1). \end{aligned}$$

Generate a test sample $T = \{t_i\}$, then make predictions $\hat{f}(t_i)$ based on the fitted model. Hence we can plot the classification boundary by making contour plots of $\hat{f}(t_i) = 0, 1, -1$. See figure 1.9.

Judging from these four plots, we can see that Gaussian kernel does better than linear kernel in the middle of the plane, while worse than linear kernel at both corners. Unlike linear kernel, the Gaussian kernel is a local one. For instance, a test point x is classified as in group1, if $\hat{f}(x) > 0$. Note

$$\widehat{f(x)} = w^t\phi(x) + b = \sum_{i\in SV} \alpha_i y_i \exp -\sigma\|x_i - x\|^2$$

If the point x is far away from most support vectors, $\hat{f}(x)$ will not be significantly different from 0, which makes the classification of x not reliable. Consider this example: there are few training data and hence support vectors at both corners. This explains why Gaussian kernel does not work well at corners.

Except for the sigmoid kernel, most support vectors lie in between the dashed line for those other three, which should be the case. The numbers of support vector machines for linear, Gaussian and polynomial do not have a large difference. while the sigmoid kernel identify too many number of support vectors. The ratio of the number of support vectors to the total number of training data is believed to relate to the upper bound of generalized classification error, thus should be controlled.

1.5 Support Vector Regression

Support Vector Machines can be applied to regression by introducing a proper loss function. Suppose we have a training set $\{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$, where

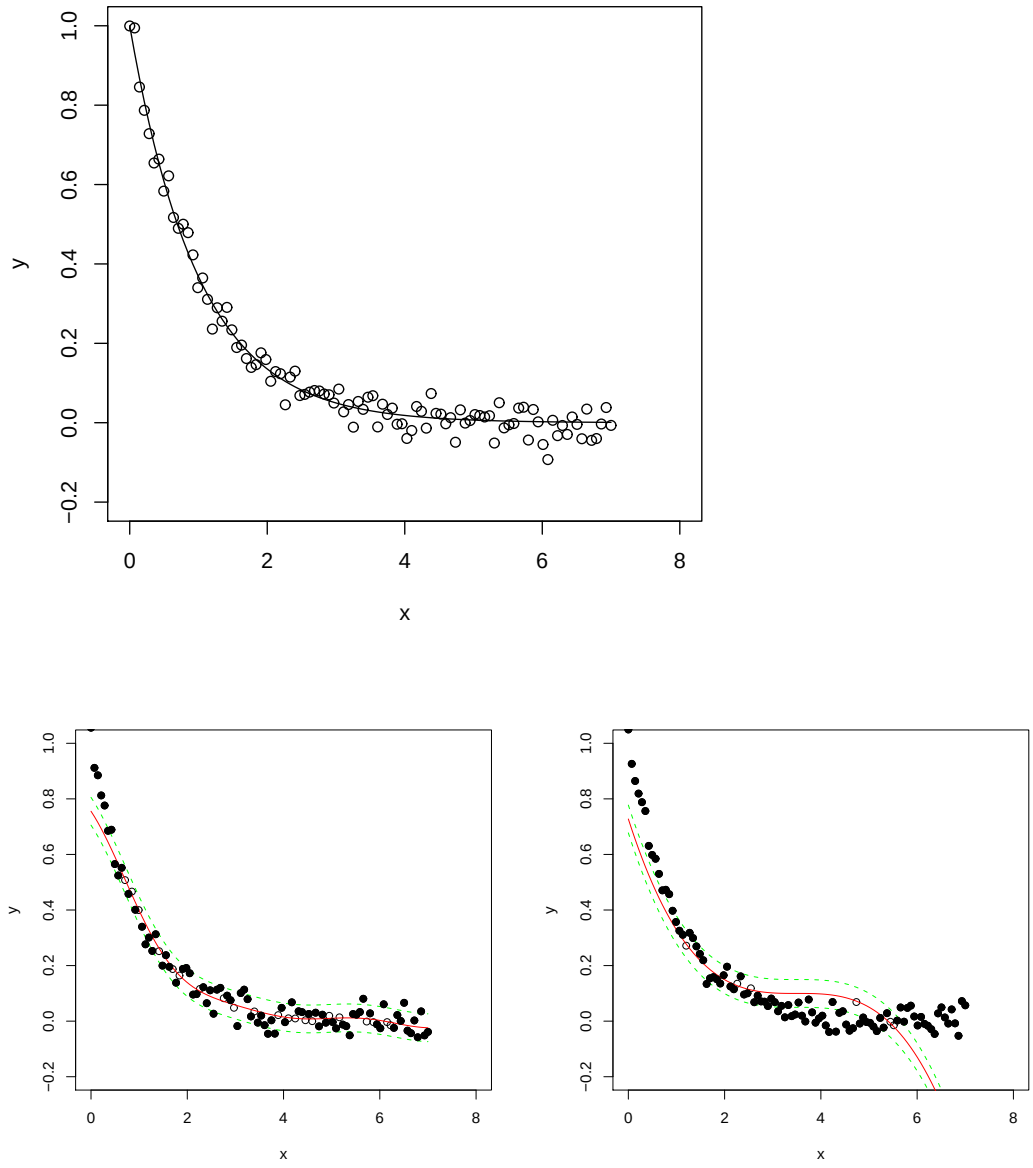

Figure 1.10: *A simulated experiment of SVM regression. The data is generated in such away:* $y_i = \exp(-x_i) + \epsilon_i$, *where* $\epsilon_i \sim N(0,1)$. *The upper plot shows the 2-dimensional training sample, where the solid line is:* $y_i = \exp(-x_i)$. *The left of the lower plots shows the fitted kernel support vector regression model with Gaussian kernel. The model parameters are set to be* $\sigma = 0.1, C = 5$. *The right of the lower plots shows the fitted kernel support vector regression model with polynomial kernel. The model parameters are set to be: degree=3, offset=1.*

$y_i, i = 1, 2, ...l$ are treated as continuous variables. Consider the following linear regressor:

$$f(x) =< w, x > +b$$

We aim to find an "optimal" $\hat{f}$. Under least square regression, the loss function is gives as:

$$L(x, y, f) = \sum_{i=1}^{l} (f(x_i) - y)^2$$

Here in support vector regression, we define the loss function to be the form of:

$$L^{\epsilon}(x, y, f) = max(0, |f(x) - y| - \epsilon)$$

The introducing of $\epsilon \geq 0$ will ensure the existence of a global minimum and a sparse solution. L^{ϵ} is called linear ϵ-insensitive loss. Sometimes, the quadratic ϵ-insensitive loss $L_2^{\epsilon} = [L^{\epsilon}]^2$ is also used.
Based on the linear ϵ-insensitive loss, optimal regressor is given by minimizing the functional:

$$\frac{1}{2}||w||^2 + C \sum_{i=1}^{l} L^{\epsilon}(x_i, y_i, f)$$

The cost parameter C is introduced to measure the trade-off between complexity and loss. In order to make the optimization problem to have a nice form, two slack variables ξ^+, ξ^- are introduced, one for exceeding the target value by more than ϵ, and the other for being more than below the target value. Thus, an equivalent representation is:

$$\min_{(w,b,\xi^+,\xi^-)} \qquad \frac{1}{2}||w||^2 + C \sum_{i=1}^{l} (\xi^+ + \xi^-)$$

subject to

$$\begin{cases} < w, x_i > +b - y_i - \epsilon \leq \xi^+, i = 1, 2...l \\ y_i - (< w, x_i > +b) - \epsilon \leq \xi^-, i = 1, 2...l \\ \xi^+, \xi^- \geq 0, i = 1, 2...l \end{cases}$$

This problem can be transformed into the following one by introducing the Lagrange multipliers α:

$$\max_{\alpha_i} \sum_{i=1}^{l} y_i \alpha_i - \epsilon \sum_{i=1}^{l} |\alpha_i| - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j < x_i, x_j >$$

subject to

$$\begin{cases} \sum_{i=1}^{l} \alpha_i = 0 \\ -C \leq \alpha_i \leq C, i = 1, 2...l \end{cases}$$

According to Karush-Kuhn-Tucker conditions, only part of these α_i^* are non-zeros, which are called support vectors. The optimal regressor can be represented as $\hat{f}(x) = \sum_{i \in SV} \alpha_i^* < x_i, x > +b^*$, where b^* is chosen so that $f(x_i) - y_i = -\epsilon$ for any i with $0 < \alpha_i^* < C$.
When kernels $K(x_i, x_j)$ are introduced, the optimization problem become:

$$\max_{\alpha_i} \sum_{i=1}^{l} y_i \alpha_i - \epsilon \sum_{i=1}^{l} |\alpha_i| - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j K(x_i, x_j)$$

subject to

$$\begin{cases} \sum_{i=1}^{l} \alpha_i = 0 \\ -C \leq \alpha_i \leq C, i = 1, 2...l \end{cases}$$

The corresponding non-linear regressor is given as $\hat{f}(x) = \sum_{i \in SV} \alpha_i^* K(x_i, x) + b^*$.
Figure 1.10 give examples of fitting SVMs using Gaussian and polynomial kernel functions.

1.6 Implementation of SVMs in R

There are many packages to implement SVMs. In the popular statistical tool *R*, one widely used package to run SVMs is "e1071". In this package, there is a function called "svm" which is capable to fit different SVMs with different kernels. The following codes are just a simple example of running SVMs in *R*:

```
> library(e1071)
> data(spam) #load the spam data
> dim(spam)
[1] 4601   58
> table(spam$type)
nonspam    spam
 2788    1813

> N=dim(spam)[1]
```

```
> N
[1] 4601

> s=sample(1:N, 3000)
> train=spam[s,] #make the training sample
> test=spam[-s,] #make the test sample
>
> # fit svm with linear kernel
> model.linear=svm(type~., data=train, kernel="linear", cost=1)
> pred.linear=predict(model.linear,test)
> table(pred.linear, test$type)

pred.linear nonspam spam
nonspam     919   77
spam         41  564
>
> #fit svm with gaussian kernel
> model.gaussian=svm(type~., data=train, gamma=1/58, cost=1,
kernel="radial")
> pred.gaussian=predict(model.gaussian,test)
> table(pred.gaussian, test$type)

pred.gaussian nonspam spam
nonspam     922   71
spam         38  570
>
> #fit svm with polynomial kernel
> model.poly=svm(type~., data=train, degree=3,coef0=1,
kernel="polynomial",cost=1)
> pred.poly=predict(model.poly,test)
> table(pred.poly, test$type)

pred.poly nonspam spam
nonspam     939  361
spam         21  280
>
```

The data in the example is known in machine learning literature as the ”spam” data, which is about spam email automatical detection. It can be downloaded from the web site: http://www.ics.uci.edu/ mlearn/databases/spambase/ .

In this data set, there is a label ”type” indicating whether a record is a spam or a non-spam email. The predictors are mostly the frequencies of certain words appearing in the email. Those selected words are thought to have some relation with the outcome of whether an email is a spam or not. As shown in the codes, the dimension of spam data is 4601×58: 4601 instances : 1813 spam and 2788 non-spam ; 58 attributes : 57 predictors and 1 response.

We fit 3 models with linear kernel, Gaussian kernel, and polynomial kernel respectively in the codes. It should be noted that several parameters need to be specified when fitting SVMs. In all three models, the ”cost” parameter denotes the constant of the regularization term in the Lagrange formulation - C in the following expression:

$$\min_{(w,b,\xi_i)} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i$$

The ”gamma” parameter denote σ in the Gaussian kernel:

$$K(x,z) = \exp(-\sigma\|x-z\|^2)$$

The ”degree” and ”coef0” parameters denote d and c in the polynomial kernel:

$$K(x,z) = (\langle x,z\rangle + c)^d$$

Those parameter will effect the performance of SVMs so that need to be numerically tuned. The following we give an example using the Gaussian kernel to illustrate one way to tune parameters in SVMs.

In the SVM model with Gaussian kernel, there are two parameters that need to be tuned: ”gamma” and ”cost”. At first, we need to generate two vectors as candidates for ”gamma” and ”cost”. Then for each pair of candidates, we do a 5-folder cross validation and record the average errors as criterions to evaluate the performance of models. For a classification table like the following table 1.1

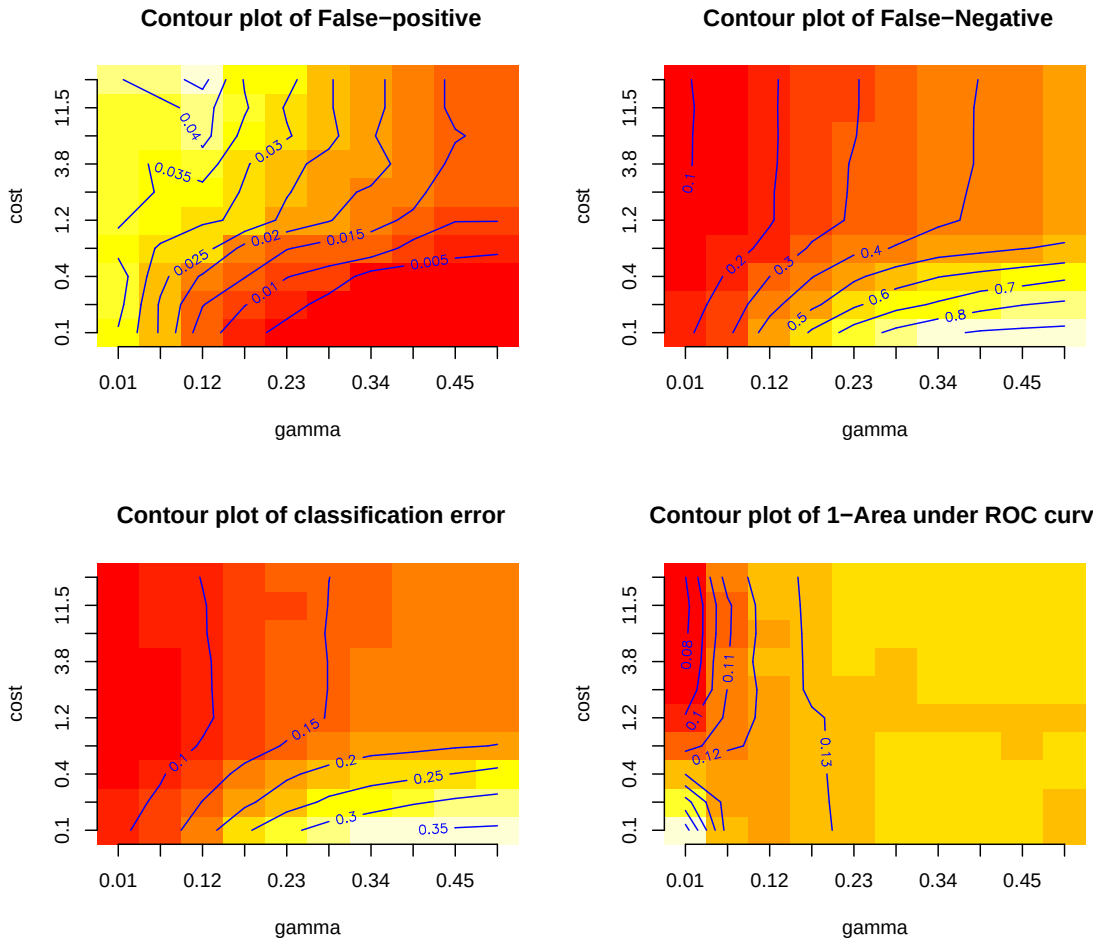

Figure 1.11: *Tuning of parameters in Gaussian kernel on the spam data. These are four contour plots of those four criterions. Those two parameters that need to be tuned are gamma and cost in a SVM model with the Gaussian kernel. Finally, we select gamma=0.01, cost=5 as the overall best parameters in the Gaussian kernel on the spam data.*

	0 (observed)	1 (observed)
0 (predicted)	a	b
1 (predicted)	c	d

Table 1.1: A classification table with predicted and observed frequencies

We define three types of errors: CE=Classification error rate, FP=False positive error rate, FN=False negative error rate, where

$$CE = \frac{b+c}{a+b+c+d}$$

$$FP = \frac{c}{c+a}$$

$$FN = \frac{b}{b+d}$$

The other criterion that we might look at is the area under ROC(Receiver Operating Characteristic) curve. A good discussion of ROC curve can be found at wikipedia.org: http://en.wikipedia.org/wiki/Roccurve. The area under ROC curve is a value between 0 and 1. The closer is this value to 1, the closer is the model to perfect classification.

In order to pick up the best "gamma" and "cost", we make four contour plots of those four criterions: classification error rate, false positive error rate, false negative error rate and 1− area under ROC curve. The reason of using "1− area" instead of "area" itself is to be consistent with the other 3 criterions (So that the lower the better). Figure 1.11 shows this plot. Judging from the plot, we choose gamma=0.01 and cost=5 as the tuned parameters in the Gaussian kernel, since all four criterions seem to be lower at the point $(0.01, 5)$.

Next to make a comparison between SVM with default Gaussian kernel and SVM with tuned Gaussian kernel,we randomly select 100 instances as the test set and leave the others as the training set. To further illustrate the performance of SVMs, a logistic regression model is also fitted. Figure 1.12 shows the ROC curves of those three models. Table 1.2 shows the four criterions of the three models on this testing set. Judging from those results, the tuned SVM (gamma=0.01, cost=5) does give the most accurate prediction.

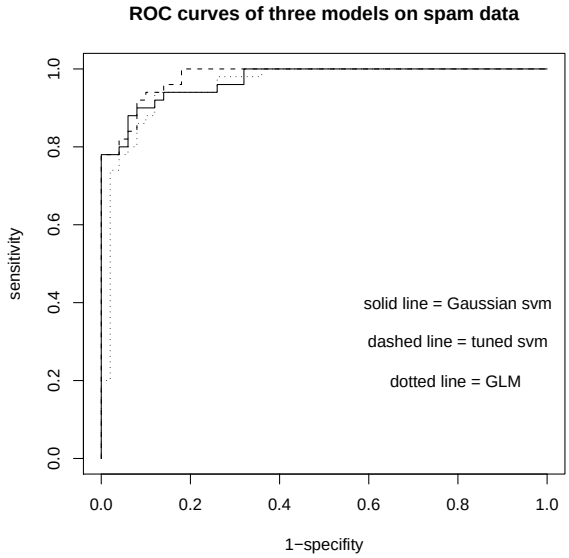

Figure 1.12: *ROC curves of three models on testing spam data. There are 100 instances in the testing set. The solid line denote the ROC curve of SVM with default Gaussian kernel. The dashed line denote the ROC curve of SVM with tuned Gaussian kernel, and the associated parameters are set to be: gamma=0.01, cos=5. The dotted line denote the ROC curve of logistic regression.*

model	FP	FN	CE	Area
SVM with default Gaussian kernel	0.06	0.2	0.13	0.9696
Logistic regression model	0.04	0.26	0.15	0.9548
Tuned SVM with Gaussian kernel	0.02	0.18	0.10	0.9888

Table 1.2: A summary of errors using 3 models on the spam data

It should be noted that "e1071" is not the only package that can handle with SVMs. In fact, one limitation of "e1071" is that it can only deal with several fixed kernels like linear, Gaussian, sigmoid, and polynomial. This limitation is avoided in another R package "kernlab". In this package, there is a function "ipop" which can solve quadratic programming problems with customized kernels. In other words, this function is able to solve SVMs with kernels defined by users. However, one drawback of this function is that it is slower than the "svm" function in "e1071". Both functions agree well in making predictions with SVMs.

1.7 Strength and weakness of SVMS

Support vector machines as a machine learning algorithm have their advantage compared to traditional pattern recognition methods. It is believed that well designed SVMs usually give more precise predication in classification problems.

Compared to other machine learning algorithms, SVMs also have several advantages. In most cases, when it refers to a criterion such as the classification error, SVMs are more accurate than other learning methods such as KNN, Neural networks. Under the convexity condition, SVMs won't get stuck at some local minima as some learning algorithms may have. Since we can cooperate all kinds of kernel functions into the dual problem in SVMs, the range of applications that can potentially solved by SVMs is large. In another word, theoretically we can always specify the most appropriate kernel function to accompany some data structures.

However, there are weaknesses in support vector machines. Although the plenty choice of kernel function makes the model very flexible, it is usually quite difficult to find the most suitable kernel. The other problem is that if there are only limited number of training examples too rich a hypothesis will lead to over fitting

and hence poor generalization. The third problem lies in the fact that when the number of observations becomes too large, the large number of parameters in SVMs (the Lagrangian multipliers) are chosen by tuning heuristics, making the system difficult and unreliable to use. Usually the training of SVMs is relatively slower than other machine learning algorithms since it has a large number of parameters to optimize. Another notable disadvantage of SVMs which is common in all machine learning algorithms is frequently a lack of understanding the learning models, thus make it difficult to do statistical inference. For instance, non convincing interpretation of the support vectors in SVMs has been given.

Chapter 2

Methods of customizing kernels in fitting SVMs

2.1 Linear combination of kernel functions

We know kernel functions are essential in fitting SVMs, however, no convincing theorem has been given regarding what kind a kernel would be proper for a particular data structure. What we do know is the choice of a kernel corresponds to the choosing of a similarity measure and the choosing of feature space, as a kernel implicitly determines the feature space for learning.

We have already seen that strength and drawbacks both exist in those 4 widely used kernel function in the simulated example (chapter 1). Since no kernel seems to be universally better than the others, we might further think of linear combination of kernels. The motivation lies in the hope that a combined one can capture the merits of the existed kernels while throw away the drawbacks.

We have already showed in chapter 1 that a linear combination of kernels with non-negative coefficients is also a kernel. On the same training sample as illustrated in figure 1.8, we use the following two types of kernels to fit SVMs:

$$\begin{aligned} K_{l-g}(x,z) &= \beta K_{linear}(x,z) + (1-\beta)K_{Gaussian}(x,z) \\ K_{l-p}(x,z) &= \beta K_{linear}(x,z) + (1-\beta)K_{Poly}(x,z) \end{aligned}$$

Several β are tried to see how the mixture kernels behave. The results are illus-

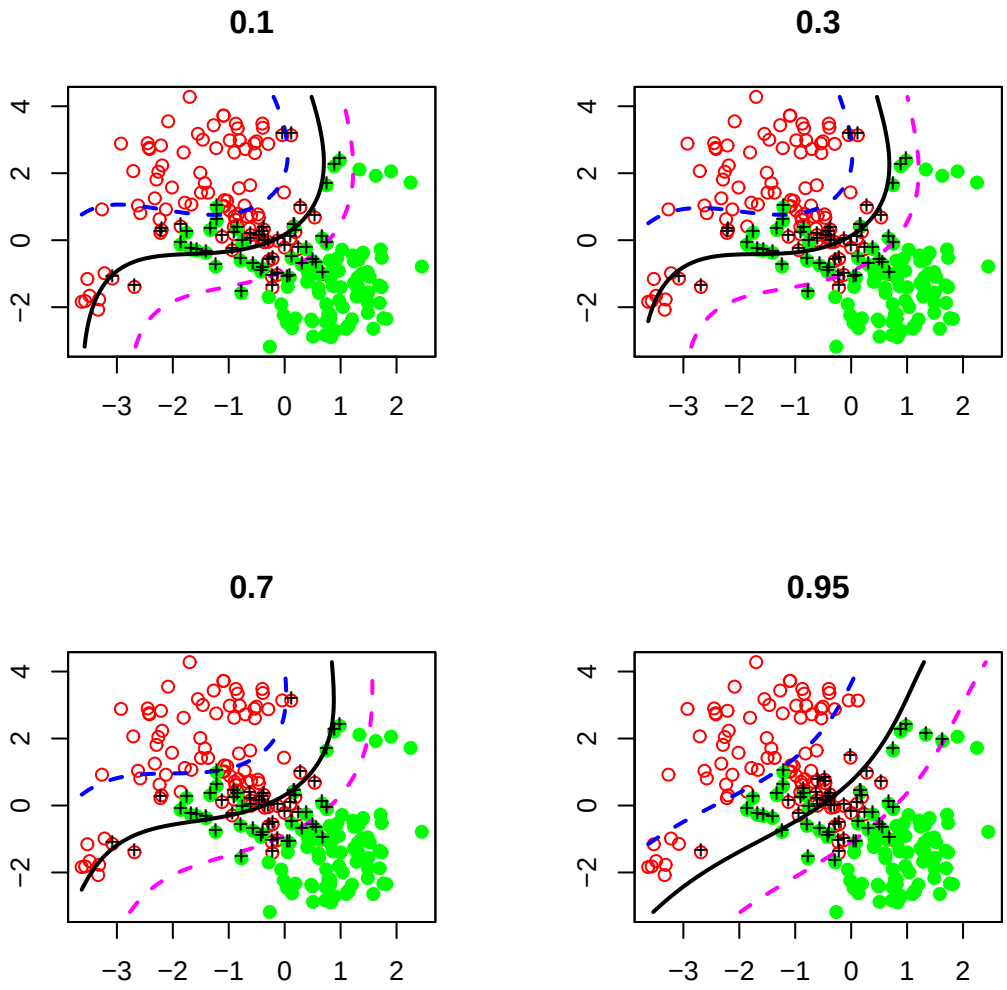

Figure 2.1: *An example of mixtures of linear and Gaussian kernels. The training sample is the same as shown in figure 8 (two groups of Gaussian). The new kernels used are linear combination of a linear kernel and Gaussian kernel:* $K = \beta K_{linear} + (1-\beta)K_{Gaussian}$, $\beta = 0.1, 0.3, 0.7, 0.95$ *respectively. We can see as* β *becomes larger, the classification boundary behaves closely to a linear classifier.*

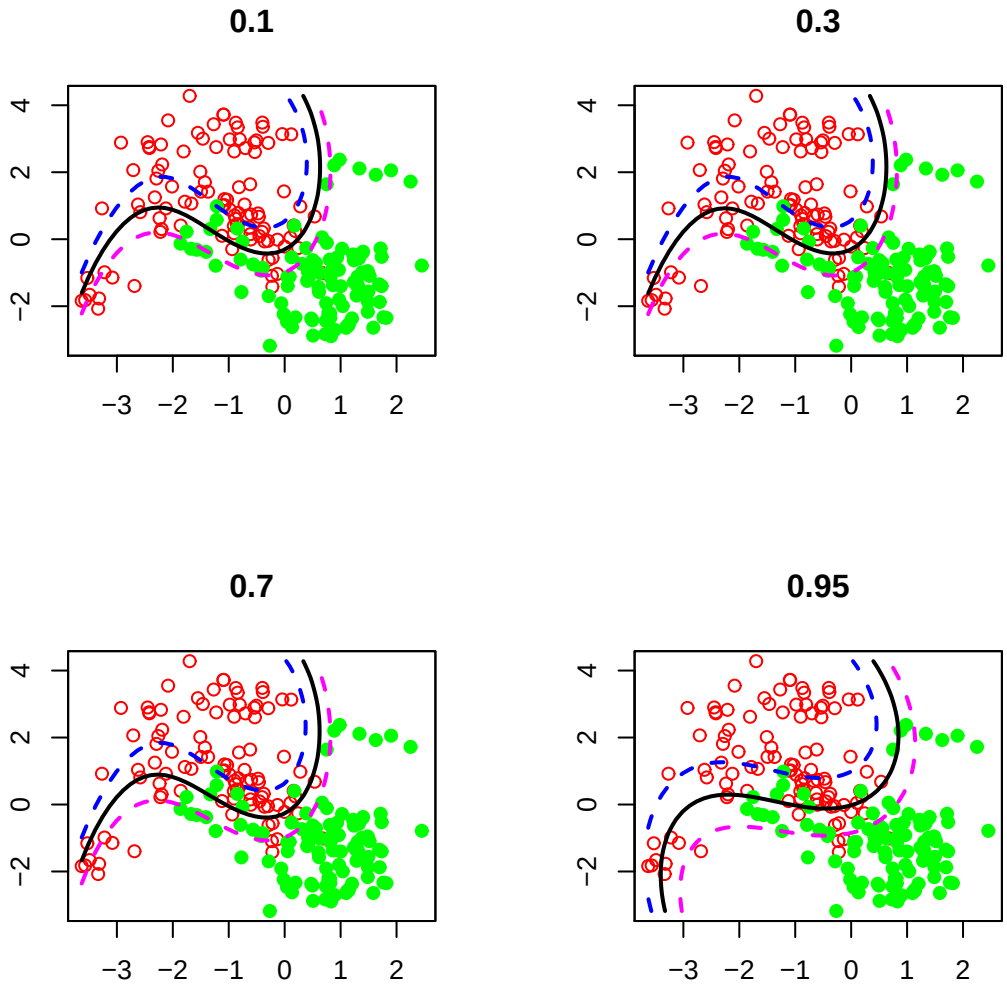

Figure 2.2: *An example of mixtures of linear and Polynomial kernels. The training sample is the same as shown in figure 8 (two groups of Gaussian). The new kernels used are linear combination of a linear kernel and Gaussian kernel:* $K = \beta K_{linear} + (1-\beta)K_{polynomial}$, $\beta = 0.1, 0.3, 0.7, 0.95$ *respectively.*

trated in figure 2.1 and 2.2. From the plots we can see that β becomes larger, the classification boundary behaves closely to a linear classifier. When $\beta = 0.9$, the classifier seems to perform better both at corners and middle of the plane, catching both the merits of linear kernels and Gaussian kernels.

To further investigate why convex combination might be useful, let us look at the new feature mapping function under combined kernel. Suppose

$$K(x, z) = \beta K_1(x, z) + (1 - \beta)K_2(x, z), \beta > 0$$

According to Mercer's theorem, let K_1, K_2 have such representations:

$$\begin{aligned} K_1(x, z) &= \sum_{i=1}^{\infty} \phi_i(x)\phi_i(z) \\ K_2(x, z) &= \sum_{i=1}^{\infty} \psi_i(x)\psi_i(z) \end{aligned}$$

Since we know as the convex combination of K_1, K_2, $K(x, z)$ also has a similar representation:

$$\begin{aligned} K(x, z) &= \sum_{i=1}^{\infty} \xi_i(x)\xi_i(z) \\ &= \sum_{i=1}^{\infty} [\beta\phi_i(x)\phi_i(z) + (1 - \beta)\psi_i(x)\psi_i(z)] \end{aligned}$$

Let

$$\xi_i(x)\xi_i(z) = \beta\phi_i(x)\phi_i(z) + (1 - \beta)\psi_i(x)\psi_i(z)$$

Further, let $x = z$, we have at least one form of representation $\xi_i(x)$ in terms of ϕ and ψ:

$$\xi_i(x) = \sqrt{\beta\phi_i(x)^2 + (1 - \beta)\psi_i(x)^2} \tag{2.1}$$

In order to visualize those features implicitly determined by various of kernel functions, we did a simulation:
First of all, generate a one dimensional array with length 50 and range: $[-4, 4]$,

denoted as $x[i], i = 1, 2, ...50$; Next, generate different kernel matrices with different kernel functions. For a kernel function K, the corresponding kernel matrix is:

$$[\mathbf{K}_{ij}] = K(x[i], x[j])$$

Note the kernel matrices are semi-positive definite. Thus we can use R to do matrix decomposition to obtain the corresponding eigenvectors:

$$\mathbf{K}=\mathbf{A}^k \Lambda \mathbf{A}$$

Where $A[, j] = v_j, j = 1, ..50$ is the jth eigenvector and Λ is the diagonal matrix with eigenvalues $\lambda_i, i = 1, ...50$. Further, $\mathbf{K}[i, j]$ can be expressed in the following form:

$$\mathbf{K}[i, j] = \sum_{k=1}^{50} \lambda_k v_{ik} v_{jk}$$

So we can define the feature mapping to be: $\Phi : R^1 \rightarrow R^{50}$:

$$\Phi(x[i]) = (\sqrt{\lambda_1} v_{i1}, \sqrt{\lambda_2} v_{i2}, ... \sqrt{\lambda_{50}} v_{i50})^t$$

And the kth element of $\Phi(x[i])$ is:

$$\phi_k(x[i]) = \sqrt{\lambda_k} v_{ik}$$

Because we have the following relation:

$$\begin{aligned} \mathbf{K}[i, j] &= K(x[i], x[j]) \\ &= \langle \Phi(x[i]), \Phi(x[j]) \rangle \\ &= \sum_{k=1}^{50} \phi_k(x[i]) \phi_k(x[j]) \end{aligned}$$

At last, we can make plots of $\phi_k(x[i])$ vs $x[i], i = 1, 2, ...50$. We only gives the first 6 element of the feature mapping Φ. Figure 2.3, 2.4, 2.5 and 2.6 give this kind of plot.

Figure 2.3 shows the first 6th feature mapping of a gaussian kernel. We have stated that the dimension of the feature space corresponding to a Gaussian kernel is in fact infinite. What's more, every element in the feature space defined by the Gaussian kernel will have norm 1: Suppose $\Phi(x) = u$, then:

$$||u|| = \langle \Phi(x), \Phi(x) \rangle = K(x, x) = 1$$

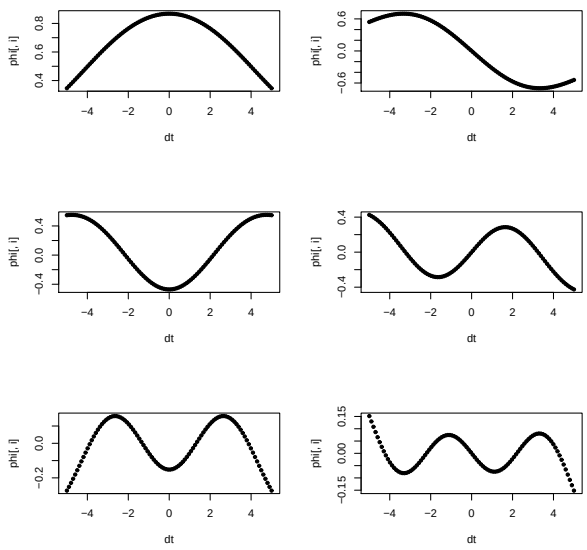

Figure 2.3: *A simulated experiment: The first 6 feature plots of Gaussian kernel:* $K(x, z) = \exp(-0.5|x - z|^2)$

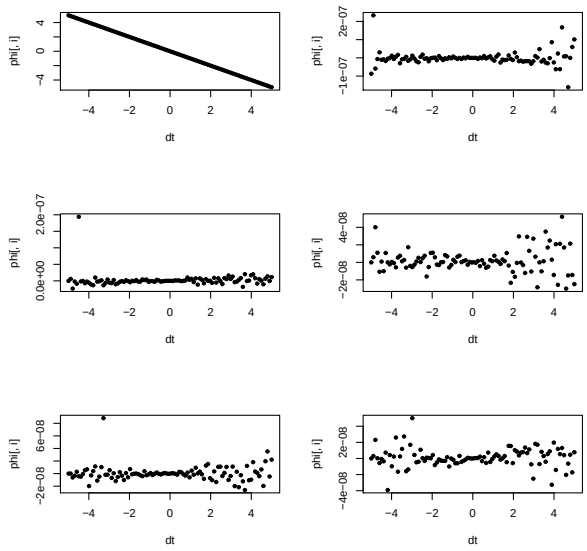

Figure 2.4: *A simulated experiment: The first 6 feature plot of Linear kernel:* $K(x, z) = \langle x, z \rangle$

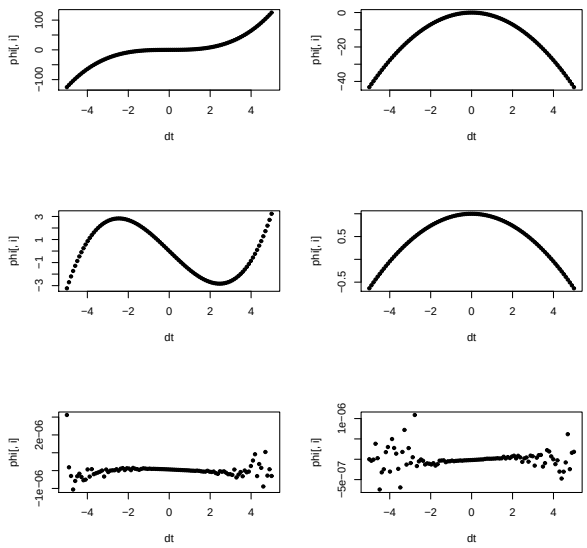

Figure 2.5: *A simulated experiment: The first 6 feature plot of Polynomial kernel:* $K(x, z) = (\langle x, z \rangle + 1)^3$

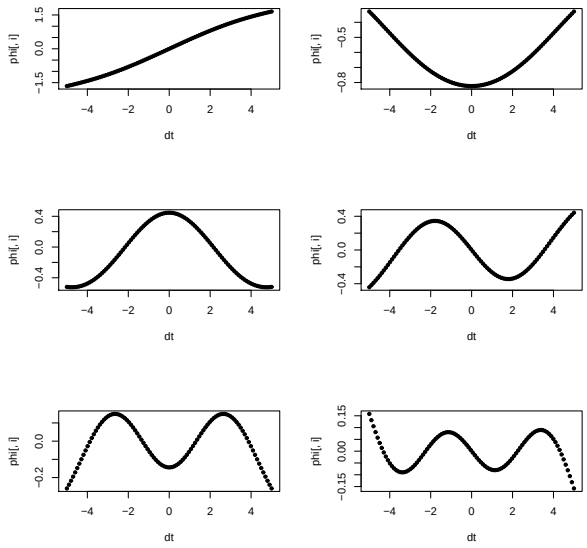

Figure 2.6: *A simulated experiment: The first 6 feature plot of combined kernel:* $K_{l-g}(x, z) = 0.8K_{linear}(x, z) + 0.2K_{Gaussian}(x, z)$

Further it can be shown that the feature mappings of a Gaussian kernel involves $\sin(nx), \cos(nx)$. As we can see from the plot that those functions behaves like sin or cos functions.
Figure 2.4 shows the situation of a linear kernel. From the plot we can see that it actually only has one feature $\phi_1(x) = -x$ (the sign does not matter here). Figure 2.5 shows 4 features for a polynomial kernel with degree 3. The possible features are combinations of the basis: $\{1, x, x^2, x^3\}$.

Figure 2.6 shows the first 6th feature plot of a kernel function which is a combination of a linear one and a Gaussian one:

$$K_{l-g}(x, z) = 0.8K_{linear}(x, z) + 0.2K_{Gaussian}(x, z)$$

If we denote the possible feature mappings for the Gaussian one as: $\phi_1, \phi_2, ...$, and denote the mappings for the new kernel as $\xi_1, \xi_2, ...$, by (10) we have:

$$\begin{aligned} \xi_1(x) &= \sqrt{0.8x^2 + 0.2\phi_1(x)} \\ \xi_i(x) &= \sqrt{0.2}\phi_i(x) \qquad i = 2, 3, 4, ... \end{aligned}$$

Because all the other features in the linear case are 0s. As we can see from figure 2.6, the first has both linearity and non-linearity and the others look similar to the Gaussian ones.
This also gives an interpretation of combining two kernel together: by combining a Gaussian kernel to a linear kernel, we are essentially adding more Gaussian features to the implicitly determined feature space. Some times the dominant feature might be linear, but more features might be needed to give more accurate classification.

2.2 Feature selection in designing kernels

Kernel matrices are essential in fitting SVMs. So it is important to customize kernels in SVMs. When it refers to the design of kernel matrices, we have two options. One is to try to decide what kind of kernel functions to use, for example, linear, Gaussian, or combined, etc. We have discussed this a lot. The other option is to decide which predictors to include in the model.

Just as we do feature selection in fitting linear models or fitting generalized linear models, sometimes it is quite helpful to do feature selection in fitting SVMs. Because in some cases, due to the extremely large dimension of data sets, other kernel functions rather than linear function might be too time and resource consuming when fitting SVMs. Thus in order to improve model accuracy, it might be helpful to go the second option. And experiments have shown that feature selection does improve the performance of SVMs. Particularly in microarray data analysis using SVMs, feature selection is practised a lot. Zhang, et al. [21] gave a good discussion of feature selection and proposed a recursive feature selection method in fitting SVMs.

According to their method, a score criterion s_j for the jth feature is defined to evaluate each feature's contribution to the classification model. Remember the model assumption in linear SVMs is:

$$f(x) = w^t x + b$$

If $f(x) > 0$ then a point x will be classified in class X^+; otherwise it will be classified in class X^-. Suppose n_1 is the number of samples of class X^+ and n_2 is the number of samples of class X^-. Let m^+ denote the mean of class X^+ and m^- denote the mean of class X^-. It is reasonable to define such a total score of separation to represent how well the data has been separated:

$$\begin{aligned} S &= \frac{1}{n_1}\sum_{x^+\in X^+} f(x^+) - \frac{1}{n_2}\sum_{x^-\in X^-} f(x^-) \\ &= \frac{1}{n_1}(w^t \sum_{x^+\in X^+} x^+ + n_1 b) - \frac{1}{n_2}(w^t \sum_{x^-\in X^-} x^- + n_2 b) \\ &= w^t[\frac{1}{n_1}\sum_{x^+\in X^+} x^+ - \frac{1}{n_2}\sum_{x^-\in X^-} x^-] \\ &= w^t[m^+ - m^-] \\ &= \sum_{j=1}^{d} w_j[m_j^+ - m_j^-] \end{aligned}$$

Where $w = (w_1, w_2, ..., w_d)$, m_j^+ denote the jth element of m^+ and m_j^- denote the jth element of m^-.

Further for the jth predictor, we can define its contribution to the separation as:

$$s_j = w_j^t \ [m_j^+ - m_j^-]$$

If the total number of predictors are d, for a fitted model with all d predictors included, we can rank their score s_j:

$$s_{(1)} \geq s_{(2)} \geq s_{(3)} \ldots \geq s_{(d)}$$

Based on this ranking, feature selection can be done repeatedly eliminating the predictors that ranks the last until an optimization goal is achieved. Detailed algorithm of this recursive feature selection method can be found in the paper by Zhang and et al. [21]

Chapter 3

Data Analysis

The following we are trying to apply the idea of combined kernel in SVMs to real data analysis. We first give a detailed comparison on a chemical data between SVMs with ordinary kernel functions and SVMs with combined kernel functions. Next we display more experiments on several widely used data sets in the machine learning literature.

3.1 A comparison between linear kernel, Gaussian kernel and combined kernel on a chemical data

The data we are going to use is about drug discovery. The total number of observations is 1280. There are two types of predictors in the data set: 480 binary predictors and 6 continuous predictors. The outcome variable is binary with two labels ″0″ and ″1″. Table 3.1 shows the frequency table of the response variable ″y″:

class	0	1	total
frequency	1223	57	1280
percent%	95.5	4.5	100

Table 3.1: A frequency table on the chemical data

Judging from this table, we can see that the two classes are quite unbalanced. One noticeable thing is the fact that this is a large n and small p data set: very

high dimension ($p = 486$) and relatively small n (1280). The other noticeable thing is that the predictors are mixtures of binary variables and continuous variables. Due to the particular structure of the predictors, a well designed kernel might give good predictions.

3.1.1 SVMs with Gaussian Kernel on the chemical data

First, SVMs with the Gaussian kernel are tested. The R package used is "e1071" with the particular function "svm". There are two parameters that need to be numerically tuned in SVMs with Gaussian kernels. One parameter is σ in the kernel function:

$$K(x, z) = \exp(-\sigma \|x - z\|^2)$$

Note sometimes γ is used instead of σ. The other parameter C is the constant of the regularization term in the Lagrange formulation.

$$\min_{(\mathbf{W}, b, \xi_i)} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{l} \xi_i$$

In order to find out the best σ and C, two vectors "gamma" and "cost" are created to represent those candidates. 10 Elements in "gamma" correspond to candidates for σ and 10 elements in "cost" correspond to candidates for C. "gamma" and "cost" are chosen as

$$gamma = [0.0001, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.015, 0.02, 0.04, 0.08]$$

$$cost = [0.01, 0.1, 0.5, 1, 1.5, 2.5, 3.5, 5.5, 7.5, 10.5]$$

Those candidates are chosen around the default values. And the default values of gamma and cost in the R function "svm" are set to : gamma=1/[the data dimension]=1/487=0.00205, cost=1. Next we need experiments to determine the most appropriate parameters for our models.

For every pair of elements in "gamma" and "cost", we need to specify some training sets and testing sets to fit SVM models and make predictions. Because the number of "1"s is quite small (57), we use the following sample strategy:

Of all the 1223 ″0″s, we randomly choose half of them as the first part of the training set, leaving the other half ″0″s as the first part of the test set. Of all the 57 ″1″s, we randomly choose half of them as the second part of the training set, leaving the other half ″1″s as second part of the test set. The whole training and test sets can be obtained after combining the first part and second part together. So the number of observations in the training set is:

$$[1223/2] + [57/2] = 612 + 28 = 640$$

And the number of observations in the test set is:

$$(1233 - 612) + (57 - 28) = 640$$

For every pair of ″gamma″ and ″cost″, we generate 5 pairs of training sets and testing sets (every pair is a 640 training + a 640 testing), then fit 5 models and make 5 predictions on the testing sets. Three types of errors are reported according to the classification table (see table 3.2) of the predicted outcomes and the observed outcomes.

	0 (obs)	1 (obs)
0 (pred)	a	b
1 (pred)	c	d

Table 3.2: A typical classification table

The three types of errors are defined to be: CE=Classification error rate, FP=False positive error rate, FN=False negative error rate, where

$$CE = \frac{b + c}{a + b + c + d}$$

$$FP = \frac{c}{c + a}$$

$$FN = \frac{b}{b + d}$$

The average errors from the 5 models are recorded to three 10×10 matrices representing those three types of errors. Contour plots are made to help us to select the best γ (or σ) and ″cost″. See figure 3.1.

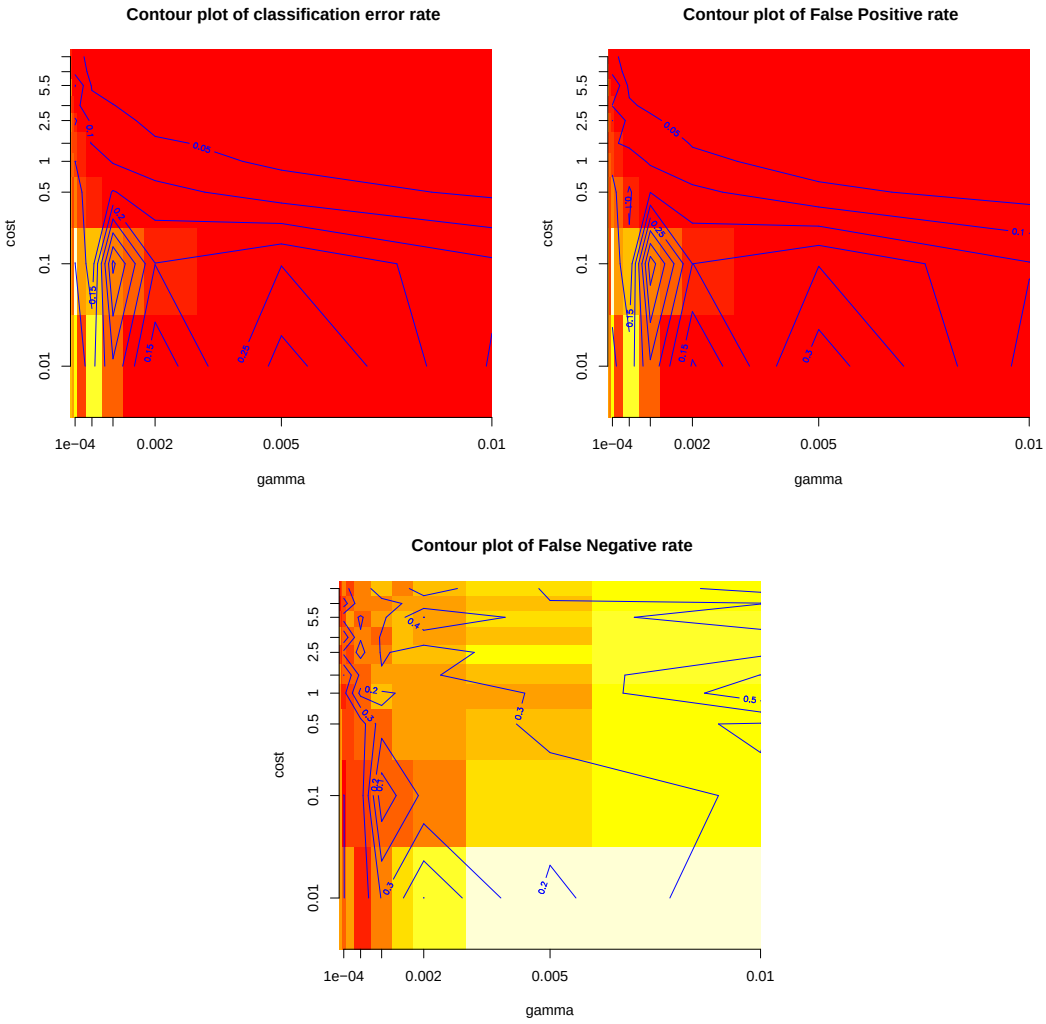

Figure 3.1: *Contour plots of three types of errors for the Gaussian kernel.*

Judging from the contour plots, it seems that when σ and "cost" are around the default values (0.00205, 1), the overall performance of SVM on the three types of errors is better than the other candidates. Finally we set

$$\sigma^* = 0.002, cost^* = 1$$

Furthermore, we generate 8 pairs of training sets and testing sets using the same method described before, on which the Gaussian SVMs with $\sigma^* = 0.002, cost^* = 1$ is tested. Three types of errors using the tuned Gaussian kernels are shown in table 3.3.

Further more, those 8 pairs of data sets will also be used in evaluating linear SVMs and SVMs with combined kernels for comparison reasons. It should be noted that here we are actually using the whole data set to do both parameter tuning and model evaluation. This will tend to bias the model performance upwards. Ideally, we should cut the whole data set into two parts: one part for parameter tuning and the other one for model evaluation. One reason we did not conduct this is that the number of "1"s in the whole data set is rather small. If we split the data into two pieces, the number of "1"s in each piece will become even smaller and the models would become unstable. The other reason is that our aim is to compare the performance among SVMs with different kernels. Since the same strategy is applied across all three SVMs with different kernels, we would expect uniform biases with all three models. Thus the comparison will be still meaningful.

data — error	CE	FP	FN
set 1	0.0734	0.0638	0.2759
set 2	0.0719	0.0573	0.3793
set 3	0.0781	0.0687	0.2759
set 4	0.0719	0.0622	0.2759
set 5	0.0734	0.0606	0.3448
set 6	0.0953	0.0835	0.3448
set 7	0.0656	0.0507	0.3793
set 8	0.0563	0.0442	0.3103
average	0.0732	0.0613	0.3232
standard error	0.0111	0.0118	0.0449

Table 3.3: Errors of tuned SVMs with Gaussian kernel on the chemical data

3.1.2 SVMs with Linear Kernel on the chemical data

Next, linear kernel SVMs are tested on the data set. The linear kernel function

$$K(x, z) = \langle x, z \rangle$$

does not involve parameters that need to be numerically tuned. However, like the Gaussian kernel, there is still a "cost" parameter C that needed to be tuned. This C is is the constant of the regularization term in the Lagrange formulation. Again, we need to choose some values as the candidates of the "cost" parameter C. We set the candidate "cost" vector to be the same as the previous one in tuning of the Gaussian SVMs:

$$cost = [0.01, 0.1, 0.5, 1, 1.5, 2.5, 3.5, 5.5, 7.5, 10.5]$$

We use the same strategy as above in determining the optimal "cost" parameter: For every single value of "cost" parameter $C_j, j = 1, 2, ..10$, we generate 10 pairs of training set and testing set $(A_i, B_i), i = 1, 2, ...10$. Let A_i denote the training set, and let B_i denote the testing set. Note the same sample generating method as the Gaussian one is used. For every pair of (A_i, B_i), together with C_j, we can fit a model, make a predication and produce three types of errors. Next we record the average errors as 3 vectors $CE_j, FP_j, FN_j, j = 1, ...10$:

$$CE_j = \frac{1}{10} \sum_{i=1}^{10} CE(A_i, B_i, C_j)$$

$$FP_j = \frac{1}{10} \sum_{i=1}^{10} FP(A_i, B_i, C_j)$$

$$FN_j = \frac{1}{10} \sum_{i=1}^{10} FN(A_i, B_i, C_j)$$

Making plots of CE, FP and FN versus the candidate "cost" vectors will help us find out that overall best "cost" parameter C.
Judging from the figure 3.2, we can see that when C equals to the default value 1, the three types of errors are controlled in a relatively low level. Thus we set the optimal cost parameter in the linear kernel to be 1:

$$C^* = 1$$

To further make a comparison, we use this tuned C^* and fit 8 linear kernel SVMs on the same 8 sets. The three types of error rates are listed in table 3.4.

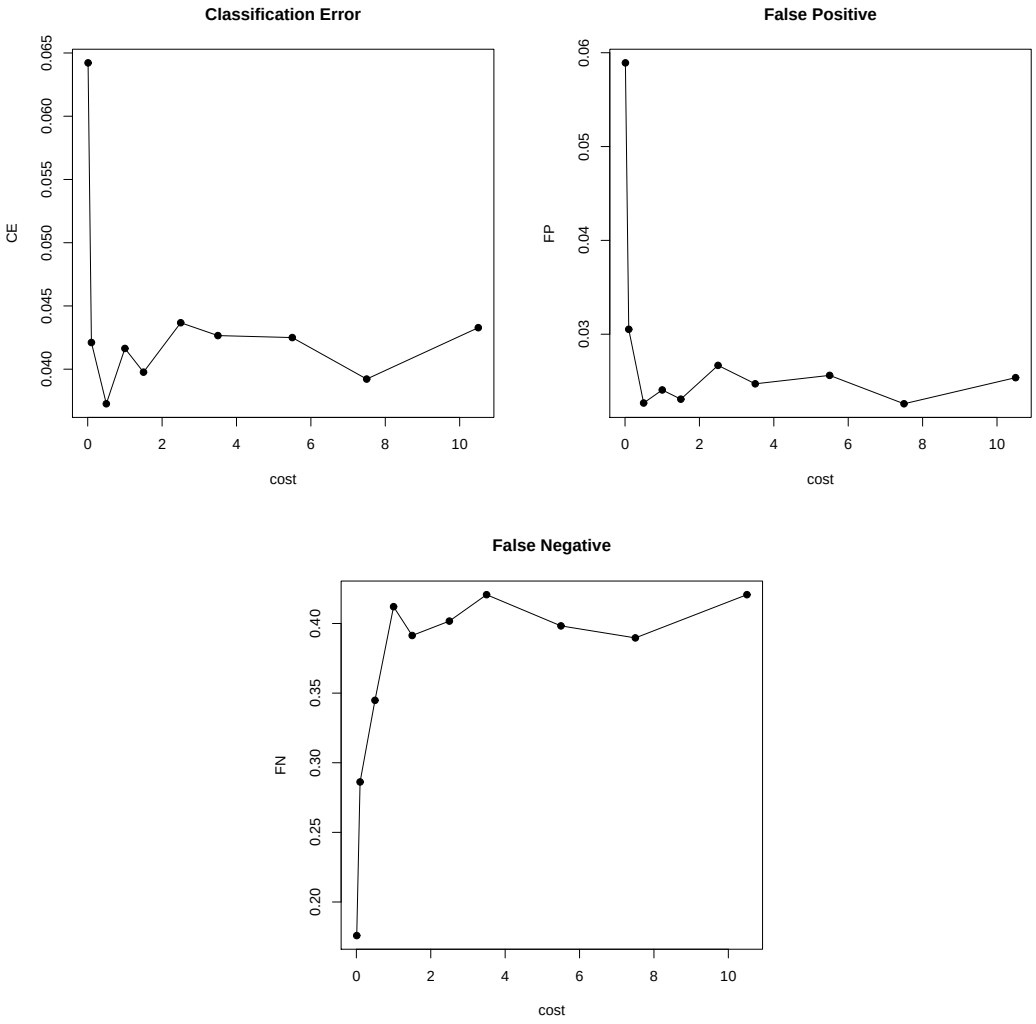

Figure 3.2: *Errors vs cost for SVMs with linear kernel*

data — error	CE	FP	FN
set 1	0.0422	0.0262	0.3793
set 2	0.0391	0.0229	0.3793
set 3	0.0406	0.0295	0.2759
set 4	0.0422	0.0196	0.5172
set 5	0.0359	0.0180	0.4138
set 6	0.0375	0.0196	0.4138
set 7	0.0297	0.0033	0.5862
set 8	0.0391	0.0278	0.2759
average	0.0382	0.0208	0.4051
standard error	0.0041	0.0082	0.1071

Table 3.4: Errors of tuned SVMs with Linear kernel on the chemical data

3.1.3 SVMs with combination kernels on the chemical data

In the following, we are going to use a new kernel which is a convex combination of a linear kernel and a Gaussian kernel. Note there are two different types of predictors : 480 binary predictors and 6 continuous predictors. For the binary part, we impose a linear kernel, and for the continuous part, we impose a Gaussian kernel. The reason why we set kernels like this is that experiments showed that Gaussian kernel works better on continuous predictors while linear kernel works better on discrete predictors. Our new kernel is a convex combination of those two parts (not just a combination of kernel functions, but a combination of two different information source):

$$K(x,z) = \beta K_l(x[1:480], z[1:480]) + (1-\beta)K_G(x[481:486], z[481:486])$$

$$0 \leq \beta \leq 1$$

where K_l denote the linear kernel and K_G denote the Gaussian kernel:

$$K_l(u,v) = \langle u, v \rangle$$

$$K_G(u,v) = \exp[-\sigma \|u-v\|^2]$$

For such a new kernel, there are 4 parameters that need to be numerically be tuned: the cost parameter in the linear part C_l; the cost parameter in the Gaussian part C_G ; the "gamma" parameter σ_G and the association parameter β. Tuning all

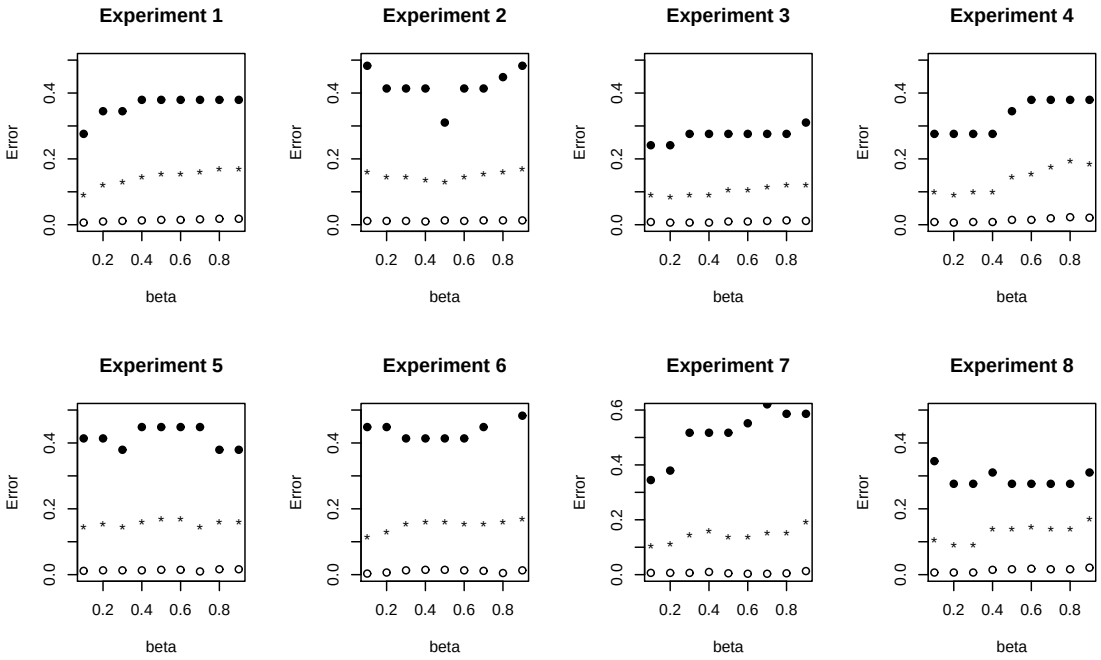

Figure 3.3: *Averaged three types of error vs β from 8 experiments: Circles denote false positive; large solid points denote false negative; small solid points denote five times of classification error rate.*

of those 4 parameters together turns out to be too time and resource consuming. Thus, we set three of those 4 parameters to be the default value:

$$C_l = C_G = 1$$

$$\sigma_G = \frac{1}{dimension} = \frac{1}{6} = 0.167$$

Then we are going to choose an optimal β from the following 9 values:

$$\beta = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$$

Although we might not achieve a best prediction by only tuning β, later we are going to show that a sub-optimal model with combined kernels are better than single kernels on this chemical data.

We did 8 experiments : For each experiment, generating 10 pairs of samples (training+testing); fix every single value of β, fitting 10 models, and record the average errors. After that, making plots of beta versus errors.

Judging from figure 3.3, when β is in between 0.1 and 0.3, those three types of errors are relatively low. Thus we set the optimal β to be 0.2:

$$\beta^* = 0.2$$

Again, using this tuned β^*, we fit 8 models on the same 8 sample sets as the Gaussian one and the linear one. The errors are reported in table 3.5:

data — error	CE	FP	FN
set 1	0.0250	0.0098	0.3448
set 2	0.0297	0.0115	0.4138
set 3	0.0172	0.0065	0.2414
set 4	0.0187	0.0065	0.2759
set 5	0.0312	0.0131	0.4138
set 6	0.0266	0.0065	0.4483
set 7	0.0234	0.0065	0.3793
set 8	0.0187	0.0065	0.2759
average	0.0238	0.0083	0.3491
standard error	0.0053	0.0027	0.0770

Table 3.5: Errors of tuned SVMs with combined kernel on the chemical data

To further comparison the performance of those three models, we plot three box plots according to three different types of errors. See figure 3.4. A summary of the average errors can be found in the table 3.6.

error	CE	FP	FN
Gaussian	0.0732	0.0613	0.3232
Linear	0.0382	0.0208	0.4051
Combined	0.0238	0.0083	0.3491

Table 3.6: Average Errors of SVMs with three different kernels on the chemical data

Form this table, we can see that, compared to the linear one, the combined one improves 37% on classification error, 60% on false positive rate, 14% on false negative rate; while compared to the Gaussian one, the combined one improves 67% and 86% on classification error rate and false positive respectively, though it is 8% worse on false negative. So we can conclude that the model with linear-Gaussian combined kernel does give a good prediction on this chemical data.

It is also helpful to look at the number of support vectors of the three different models on the 8 sample sets. A summary can be found in the table 3.7:

# of SVs	Gaussian	Linear	Combined
set 1	284	68	72
set 2	226	85	67
set 3	275	77	62
set 4	264	73	68
set 5	218	65	66
set 6	240	87	82
set 7	240	68	58
set 8	251	75	78
average	249.75	74.75	70.375
percent	39.02%	11.68%	11.00 %

Table 3.7: The number of support vectors of the 3 models on the chemical data

The average percent = the average number of support vector / 640 (the total number of samples in the training set). Usually the lower this percent is, the better the model will be. It should be noted that models with Gaussian kernels often have larger number of support vectors. But the number of support vectors in the combined case does not increase though Gaussian kernel are imposed on part of predictors, which indicates the combined kernel is appropriate for this chemical data.

3.2 More experiments using ordinary kernels and customized kernels

The following we have done 3 more experiments on 3 different real data sets, using both ordinary kernel functions and linear-Gaussian combined functions. On every data set, we have done 7-folder cross validation to make comparisons.

Data description
SPAM E-mail Database:
Source: http://www.ics.uci.edu/ mlearn/databases/spambase/
Number of Instances: 4601 (1813 Spam = 39.4%)
Number of Attributes: 58 (57 continuous, 1 nominal class label)
Model fitting: Fit 5 SVM models using kernels as Gaussian, Linear, Polynomial, Sigmoid, and Linear-Gaussian combined kernel with $\beta = 0.2$. Parameters in the kernels are set to be default. Using 7-folder cross validation.

Pima Indians Diabetes Database:
Source: http://www.ics.uci.edu/ mlearn/databases/pima-indians-diabetes/
Number of Instances: 768(type 1 =268 34.9%)
Number of Attributes: 9(8 continuous, 1 nominal class label)
Model fitting: Fit 5 SVM models using kernels as Gaussian, Linear, Polynomial, Sigmoid, and Linear-Gaussian combined kernel with $\beta = 0.8$. Parameters in the kernels are set to be default. Using 7-folder cross validation.

Wisconsin Breast Cancer Database:
Source:http://www.ics.uci.edu/ mlearn/databases/breast-cancer-wisconsin/

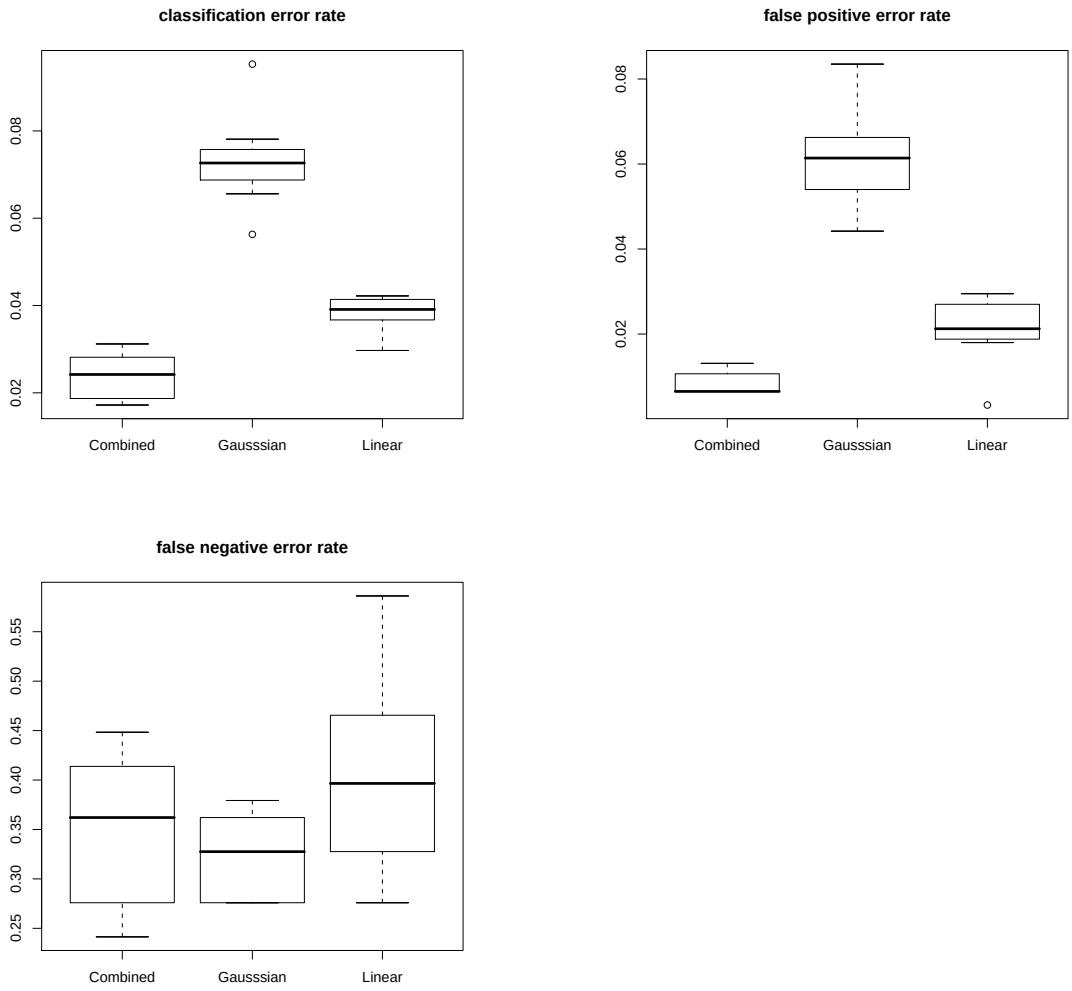

Figure 3.4: *Box plots of the three types of errors on the 8 sample sets of the chemical data. Three kernels are compared: linear kernel, the Gaussian kernel and the combined kernel.*

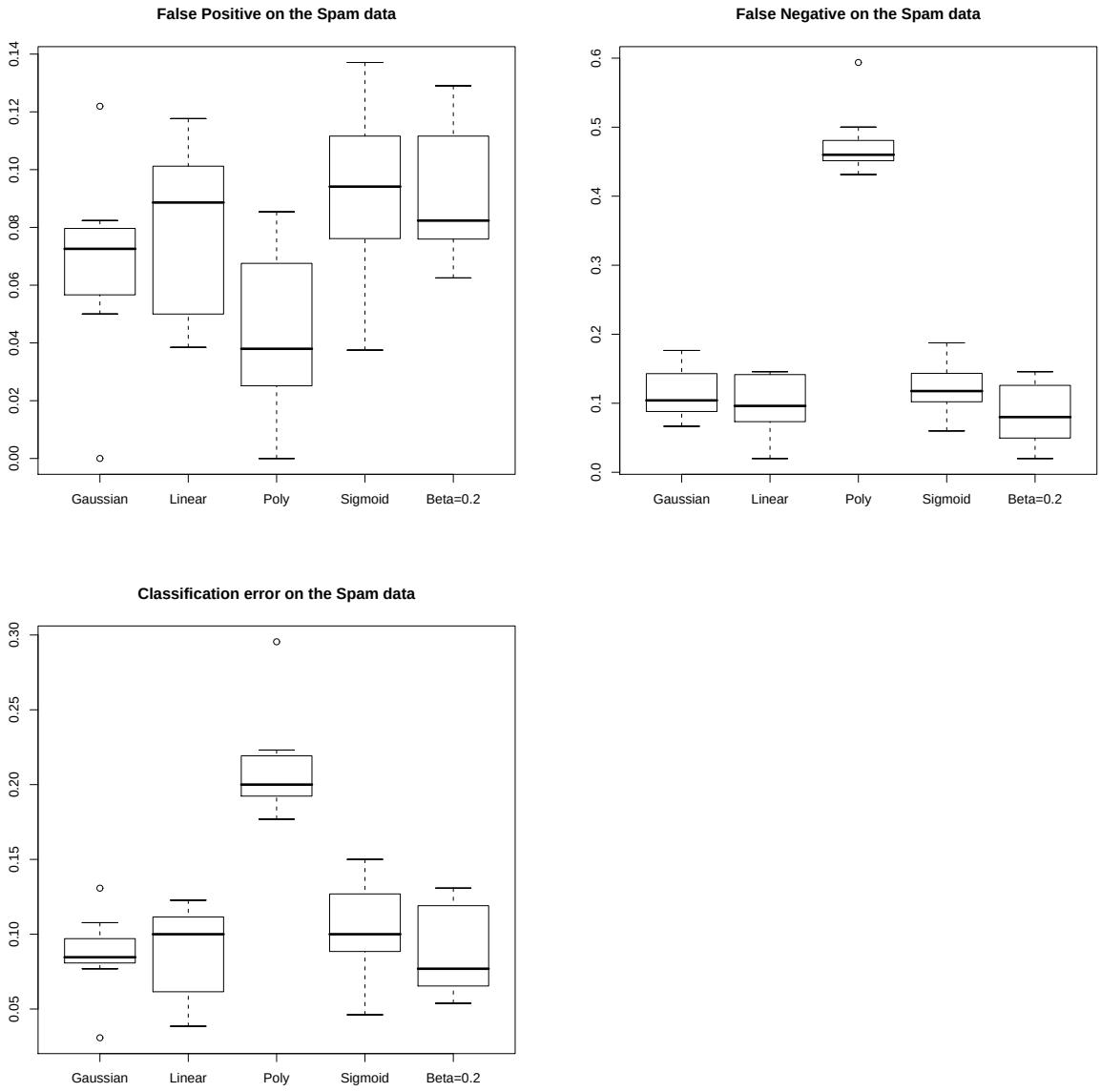

Figure 3.5: *7-folder cross validated error rate on the Spam data*

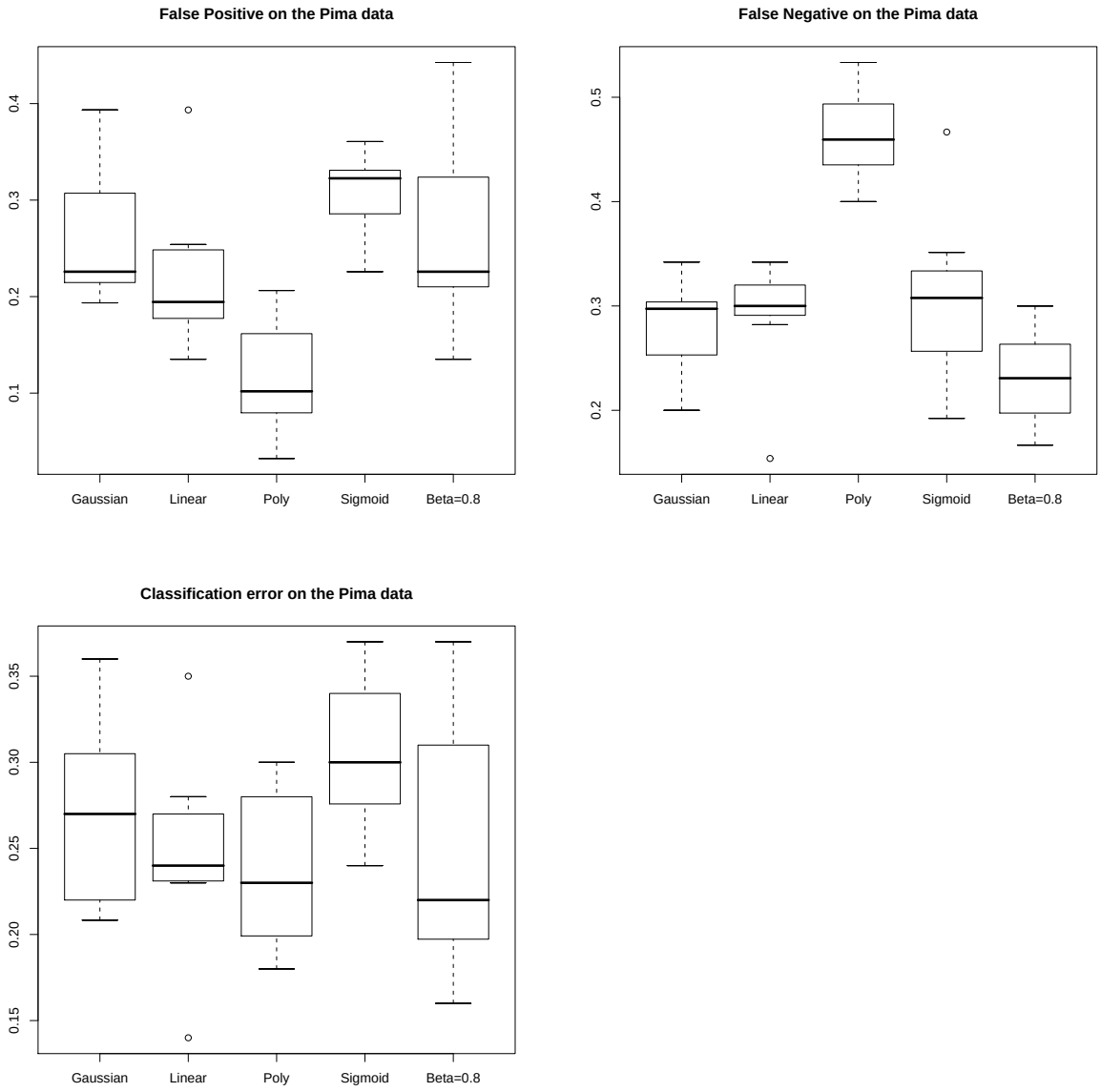

Figure 3.6: *7-folder cross validated error rate on the Pima data*

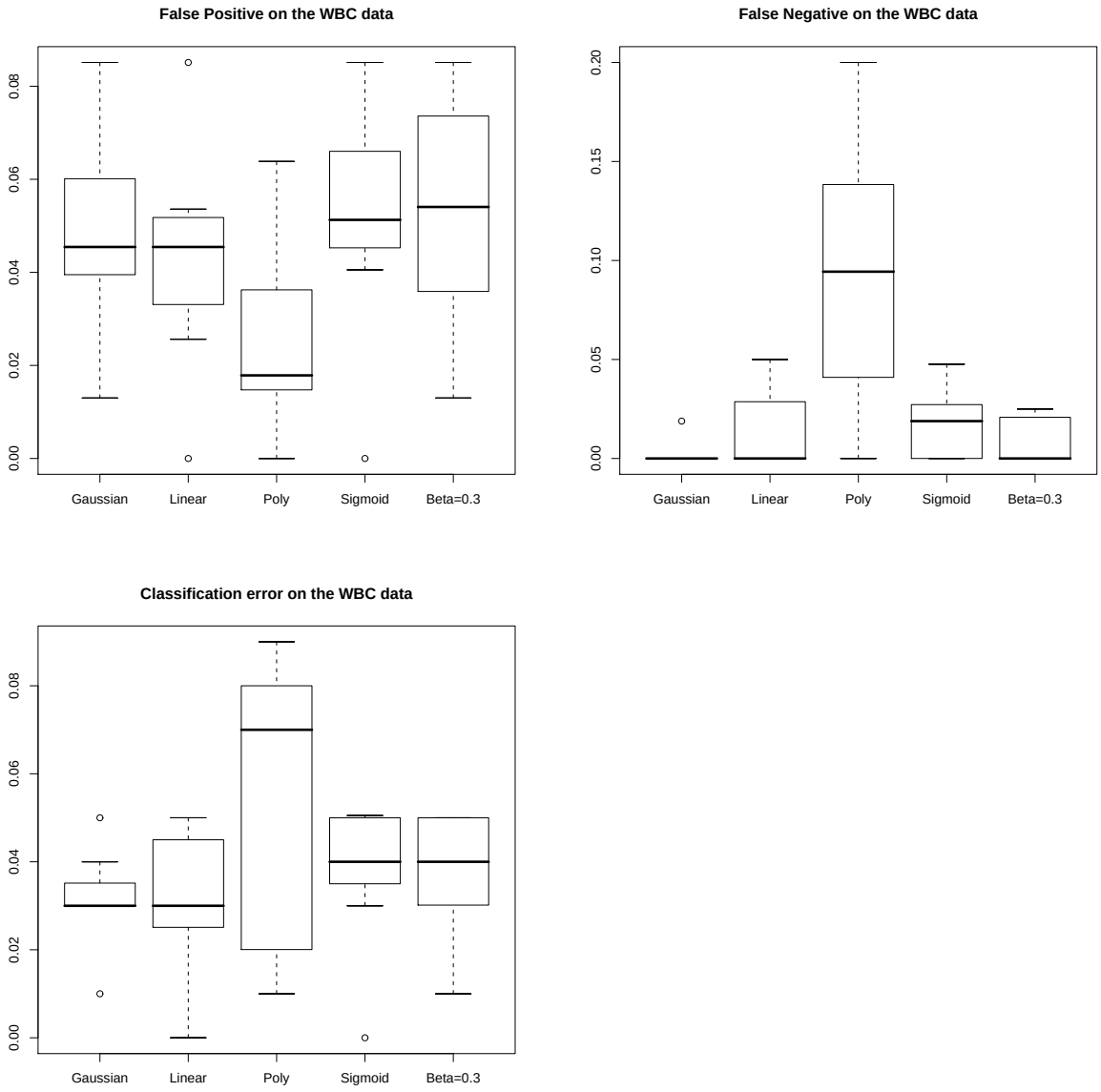

Figure 3.7: *7-folder cross validated error rate on the WBC data*

Number of Instances: 699 Benign:458(65.5%) Malignant: 241(34.5%)
Number of Attributes: 11(10 continuous, 1 nominal class label)
Model fitting: Fit 5 SVM models using kernels as Gaussian, Linear, Polynomial, Sigmoid, and Linear-Gaussian combined kernel with $\beta = 0.3$. Parameters in the kernels are set to be default. Using 7-folder cross validation.

Cross validated error Summary						
Dataset	Error	Gaussian	Linear	Poly	Sigmoid	Combined
WBC	FP	0.049	0.043	0.026	0.051	0.053
	FN	0.003	0.015	0.093	0.017	0.010
	CE	0.031	0.031	0.053	0.037	0.037
Pima	FP	0.265	0.225	0.118	0.306	0.267
	FN	0.279	0.288	0.464	0.307	0.231
	CE	0.270	0.247	0.238	0.306	0.252
Spam	FP	0.067	0.078	0.044	0.092	0.093
	FN	0.116	0.099	0.478	0.122	0.085
	CE	0.086	0.087	0.214	0.104	0.090

Table 3.8: Error Summary using different kernels on 3 data sets

Table 3.8 is a summary of cross validated errors. Figure 3.5, 3.6 and 3.7 shows the box plots of three types of errors according to different types of models. Judging from the table and plots, we can see that although models with linear-Gaussian combined kernel does not perform better than all the other ordinary kernels on all 3 data sets, it does provide another option when people need to think about what kind of kernels to use in their SVMs.

Chapter 4

Conclusion

The first objective of this thesis is to present a new method of designing kernels in SVMs, explain how it can be done and try to interpret it. The second objective is to compare the new technique to the existed techniques.

A review of SVMs was given in Chapter 1. From the simplest case–a linear maximal margin SVM, we went further to soft margin SVMs with kernels and SVM regression. We also showed how to implement SVMs in R using the two packages "e1071" and "kernlab".

In chapter 2, we discussed two solutions in customizing kernels when fitting SVMs. One is to use convex combination of two existed kernels as new kernels in SVMs. The other one is to exclude some predictors in the kernel functions when computing kernel matrices. We call the second one as feature selection in SVMs. For the method of combined kernels, we did some theoretical analysis based on Mercer's theorem [11]. We gave one possible form of feature mappings of a combined kernel in terms of the feature mappings corresponding to the original kernels. We also did an simulation study to try to interpret the feature mappings of a combined kernel. For the second method, we described and explained the recursive feature selection algorithm due to Zhang et al. [21]

We did a data analysis on a chemical data in chapter 3 to implement the method of combined kernels. The predictors in the data have two types: 480 binary predictors and 6 continuous predictors. The combined kernel we used was a convex

combination of a linear kernel on the binary predictors and a Gaussian kernel on the continuous predictors. We did cross validation to determine the association parameter of two kernels. A comparison between this combined kernel and single kernels like linear or Gaussian was also made. Judging from the results, compared to a linear kernel, the linear-Gaussian combined kernel improves 37% on classification error rate, 60% on false positive rate, and 14% on false negative rate; while compared to the Gaussian one, the combined one improves 67% and 86% on classification error rate and false positive respectively, though it is 8% worse on false negative rate.
Further we did three more data analysis on other three real data sets. Again, only linear-Gaussian combined kernel was tried. We also made comparisons between the combined ones and the single ones. Though no significant improvements on error rates had been found by using the combined kernels, we did present a new possible way to create kernels in SVMs.

For future work, the feature selection method discussed in chapter 2 can also be implemented and compared with those SVMs where no feature selections have been done. And we can practise both methods in one analysis: doing feature selection first, then trying combined kernels.
In the data analysis part, we only tried linear-Gaussian combined kernel. It is meaningful to see how other combined kernels behave, like linear-polynomial, Gaussian-polynomial, and etc. Further more, a convex combination of more than 2 kernels might also be worth investigating. Finally, the methods in customizing kernels can also be applied to SVM regression.

Bibliography

[1] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the Acercan Mathematical Society*, 68:337–404, 1950.

[3] B.E.Boser, I.M.Guyon, and V.N.Vapnik. A training algotighm for optimal margin classifiers. *Proceedings of the 5th Annual ACM workshop on Computational Learning*, pages 144–152, 1992.

[4] M. Brown and W.Grundy. Knowledge-based analysis of microarray gene expression data using support vector machines. Technical report, 1999.

[5] C.Cortes and V. Vapnik. Support vector networks. *Machine learning*, 20:273–297, 1995.

[6] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and Other Kernel-based learning methods*. Cambridge University Press, 1999.

[7] F.W.Smith. Pattern classifier design by linear programming. *IEEE Transactions on Computers*, 17:367–372, 1968.

[8] G.Wahba. Spline models for observational data. *CBMS-NSF Regional conference series in Applied Mathematics*, 59, 1990.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elemetns of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[10] J.Weston and C. Herbrich. Support vector machines for multi-class pattern recognition. *Proceedings of the 6th European Symposium on Artificial Neural Networks (ESANN)*, 1999.

[11] Jorgens Konrad. *Linear integral operators*. Pitman, 1982.

[12] K.P.Bennett and O.L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[13] O.L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operational Research*, 13:444–452, 1965.

[14] R.O.Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[15] B. Scholkopf, A. Smola, R. Williamson, and P. Barlett. New support vector algorithms. Technical report, 1998.

[16] K. Shimizu, E. Aiyoshi, and R. Katayama. Generalized farkas' theorem and optimization of infinitely constrained problems. *Journal of Optimization Theory and Applications*, 40(3):451–462, 1983.

[17] A. Smola, B. Scholkopf, and K. R. Muller. General cost functions for support vector regression. *Proc. of the Ninth Australian Conf. on Neural networks*, pages 79–83, 1998.

[18] T.Poggio. and F.Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), 1990.

[19] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

[20] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[21] X. Zhang and et al. Recursive svm feature selection and sample classification for mass-spectrometry and microarray data. *BMC Bioinformatics*, 7:197–210, 2006.