Statistical Classification

Minsoo Kim
Pomona College
Advisor: Jo Hardin

April 2, 2010

Contents

Chapter 1

Introduction

Consider the following problem: suppose you are a doctor whose patient has just been hospitalized with a heart attack. Will he or she have another heart attack within the next six months? Or this one: suppose you are a bank, and given a person who wants to take out a small business loan, will she default on the loan? Or how about this one: how can your email server tell which emails are spam and which ones are actual mail?

Intuitively, there is a sense that all of the above situations can be resolved by examining empirical data and figuring out what factors are important in each. For example, in the heart attack case, one might want to look through hospitalization records of patients who have had two heart attacks in a six month period and see if your patient resembles them in age, demographics, diet and exercise habits, and other clinical measurements. Or for loans, one might evaluate future borrowers on their credit score, income, and number of credit cards they have, and make a decision based on previous borrowers with similar profiles.

The above situations are examples of classification problems. ***Classification*** is a statistical method used to build predicative models to separate and classify new data points. Let us examine the spam filtering example in more detail. The populations we want to distinguish between are junk emails, those emails from advertisers or hackers that fill up your inbox and cause everyone a headache, and real emails. Next, suppose you have a collection of emails already, say N of them, some of them junk and some of them real, and from these emails we want to build ways to filter out spam. We refer to this collection of emails as the ***training set***, which is, in general, a set of data that is already classified that we use to build our classification model. We refer to the training data as $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$.

Now, from our training data, we need to determine which variables, called ***features***, we want to measure to assess whether future emails are spam or not. Features can be continuous or discrete: an example of a discrete feature is whether the sender of an email comes from a .edu address or not, and an example of a continuous feature is the percentage of an email that a certain word or character (like "free", "your", or "!") takes up. If m features are measured, each email contains a $m \times 1$ row vector $\mathbf{x}_i$, for $i \in \{1, \dots, N\}$ of data, and we refer to the space $\mathbb{R}^m$ as the ***feature space***. The training data is a $N \times m$ matrix, where entry x_{ij} represents the j^{th} feature of the i^{th} email. Finally, using our training data X, classification will make a ***decision function***, $D(\mathbf{x})$, a function that takes a new data point and produces a prediction of its population

The three examples above are just a small sample of the uses of classification. Given these preliminaries, this paper will examine various methods of classification, starting with linear discriminant analysis and Fisher's discriminant analysis, to Support Vector Machines (SVMs). Our work will conclude with an analysis of the linear classifiers and SVMs applied to an actual data set.

Chapter 2

Basic Discriminants

The first classification methods we examine are linear discriminant; particularly, Linear Discriminant Analysis and Fisher's Discriminant Analysis. They are similar in that they both produce linear decision functions that in fact are nearly identical, but the two methods have different assumptions and different approaches. In this chapter the two methods are compared and contrasted.

2.1 Linear Discriminant Analysis for Two Populations

Given a pair of known populations, π_1 and π_2, assume that π_1 has a probability density function (pdf) $f_1(\mathbf{x})$, and similarly, π_2 has a pdf of $f_2(\mathbf{x})$, where $f_1(\mathbf{x}) \neq f_2(\mathbf{x})$. Then intuitively, a decision function for the two populations would arise from looking at the ***probability ratio***: $D(\mathbf{x}) = \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}$. A new observation $\mathbf{x}$ is classified as π_1 if $D(\mathbf{x}) > 1$ and π_2 if $D(\mathbf{x}) < 1$. (For cases where $D(\mathbf{x}) = 1$, the vector $\mathbf{x}$ is unclassifiable.) Let Ω be the space of all possible observations, and denote the set of $\mathbf{x} \in \Omega$ where $\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} > 1$ as R_1, and similarly the set of $\mathbf{x} \in \Omega$ where $\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} < 1$ as R_2. (Denote the set $\left\{\mathbf{x} \mid \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = 1\right\}$ as being R_3.)

Such a decision function is simple but effective: by determining from which population $\mathbf{x}$ is more likely to have come, one can make quick predictions about its origin. However note that the probability ratio decision function is surely not fool proof: when the probabilities $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are close together (or not), there is always a chance that $\mathbf{x}$ could be from π_2 when $f_1(\mathbf{x}) > f_2(\mathbf{x})$. (Or visa versa.) The conditional probability, $P(2|1)$, of classifying an observation $\mathbf{x}$ as π_2 when in fact $\mathbf{x} \in \pi_1$ is

$$P(2|1) = P(\mathbf{x} \in R_2 \mid \pi_1) = \int_{R_2} f_1(\mathbf{x}) d\mathbf{x}. \tag{2.1}$$

Similarly, the conditional probability of classifying an observation x as π_1 when in fact $\mathbf{x} \in \pi_2$ is

$$P(1|2) = P(\mathbf{x} \in R_1 \mid \pi_2) = \int_{R_1} f_2(\mathbf{x}) d\mathbf{x}. \tag{2.2}$$

Often times, the costs of misclassification for the two populations are different; take for example testing for a fatal but easily curable disease. The cost of misclassifying a patient as healthy when he or she is sick in this case would be higher than misclassifying a healthy patient as sick. To account for such situations, we would like a classification method that would classify patients as healthy only when there is sufficient evidence to overcome a certain cost threshold. We define $c(1|2)$ as the cost of misclassifying a data point in π_1 when it is actually a member of π_2, and similarly, $c(2|1)$ as the cost of misclassification into π_2 when $\mathbf{x} \in \pi_1$.

Another factor that can affect accurate classification is the prior probability of belonging to one or the other of the populations. Again referring to the above example, if said disease has a low prevalence, even a test with a high sensitivity will classify many patients as sick when they are not when administered to enough people, making it difficult to determine whether a positive test actually indicates a sick patient. Some sort of weight ought to be given to the prior probability that a random observation $\mathbf{x}$ is from π_1 or π_2, and we denote such probabilities as p_1 and p_2 respectively.

Note that with our prior probabilities, we can find the overall probabilities of misclassification by substituting our priors into the earlier conditional probabilities:

$$P(\text{observation comes from } \pi_1 \text{ and is misclassified as } \pi_2) \tag{2.3}$$
$$= P(\mathbf{x} \in R_2 \mid \pi_1) P(\pi_1) \tag{2.4}$$
$$= P(2|1) \times p_1, \tag{2.5}$$

and

$$P(\text{observation comes from } \pi_2 \text{ and is misclassified as } \pi_1) \tag{2.6}$$
$$= P(\mathbf{x} \in R_1 \mid \pi_2) P(\pi_2) \tag{2.7}$$
$$= P(1|2) \times p_2. \tag{2.8}$$

Then, the ***expected cost of misclassification*** (ECM) is given by multiplying the cost of misclassification by the overall probability of misclassification for each population:

$$ECM = c(2|1)P(2|1)p_1 + c(1|2)P(1|2)p_1. \tag{2.9}$$

One criterion for a classification method is to minimize the ECM, which leads us to the following result:

Theorem 2.1.1. *(Johnson and Wichern, [4]) The regions R_1 and R_2 that minimize the ECM are defined by the values of $\boldsymbol{x}$ for which the following inequalities hold:*

$$R_1 : \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} > \frac{c(1|2)}{c(2|1)} \times \frac{p_2}{p_1}, \tag{2.10}$$
$$R_2 : \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} < \frac{c(1|2)}{c(2|1)} \times \frac{p_2}{p_1}. \tag{2.11}$$

Proof. Let us substitute the integral expressions for $P(2|1)$ and $P(1|2)$ given by (2.1) and (2.2) into the ECM:

$$ECM = c(2|1)p_1 \int_{R_2} f_1(\mathbf{x}) d\mathbf{x} + c(1|2)p_2 \int_{R_1} f_2(\mathbf{x}) d\mathbf{x}. \tag{2.12}$$

Note that $\Omega = R_1 + R_2 + R_3$, so

$$1 = \int_{\Omega} f_1(\mathbf{x})d\mathbf{x} = \int_{R_1} f_1(\mathbf{x})d\mathbf{x} + \int_{R_2} f_1(\mathbf{x})d\mathbf{x} + \int_{R_3} f_1(\mathbf{x})d\mathbf{x}. \tag{2.13}$$

Since we know that $f_1(\mathbf{x}) \neq f_2(\mathbf{x})$, R_3 must be a union of distinct points, meaning $\int_{R_3} f_1(\mathbf{x})d\mathbf{x} = 0$, leaving us with

$$1 = \int_{R_1} f_1(\mathbf{x})d\mathbf{x} + \int_{R_2} f_1(\mathbf{x})d\mathbf{x}. \tag{2.14}$$

Plugging (2.14) into (2.12), we see

$$ECM = c(2|1)p_1 \left[1 - \int_{R_1} f_1(\mathbf{x})d\mathbf{x}\right] + c(1|2)p_2 \int_{R_1} f_2(\mathbf{x})d\mathbf{x} \tag{2.15}$$

$$\Rightarrow ECM = \int_{R_1} \left[c(1|2)p_2 f_2(\mathbf{x}) - c(2|1)p_1 f_1(\mathbf{x})\right] d\mathbf{x} + c(2|1)p_1. \tag{2.16}$$

Recall that $p_1, p_2, c(1|2), c(2|1)$ are all positive since they are all probabilities and $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are both positive functions. Then by inspection, we can see that the ECM is minimized when R_1 is defined by those $\mathbf{x}$ such that $[c(1|2)p_2 f_2(\mathbf{x}) - c(2|1)p_1 f_1(\mathbf{x})] \leq 0$. However, if in (2.13) we had decided to use $f_2(\mathbf{x})$ instead of $f_1(\mathbf{x})$, (2.16) would have been

$$ECM = \int_{R_2} \left[c(2|1)p_1 f_1(\mathbf{x}) - c(1|2)p_2 f_2(\mathbf{x})\right] d\mathbf{x} + c(1|2)p_2,$$

leading to the definition of R_2 as the set $\{\mathbf{x} \mid [c(2|1)p_1 f_1(\mathbf{x}) - c(1|2)p_2 f_2(\mathbf{x})] \leq 0\}$. As those $\mathbf{x}$ that satisfy $c(1|2)p_2 f_2(\mathbf{x}) = c(2|1)p_1 f_1(\mathbf{x})$ can therefore be defined as both R_1 and R_2, we require that the inequalities in Theorem (2.1.1) defining the classification regions be strict and denote the case of equality as unclassifiable. (Note that unclassifiable points happen with probability zero.) □

The decision function given in Theorem (2.1.1) compares the probability ratio to the ***cost ratio*** and ***prior probability***. The use of ratios is important as often times, it is much easier estimate the cost ratio than each cost explicitly. For example, if we are considering the costs of an state university of educating an eventual dropout versus the costs of not educating a eventual graduate, the former can be estimated with the school taxes and tuition, but the latter is more difficult to gauge. However, it could still be accurate to predict that such a cost ratio might be 5:1 or so.

Let us examine the case where $f_i(\mathbf{x})$ are multivariate normal densities with known mean vectors $\boldsymbol{\mu}_i$ and covariance matrices Σ_i.

2.1.1 Classification of Normal Populations when $\Sigma_1 = \Sigma_2 = \Sigma$

Suppose that the density functions for $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ for population π_1 and π_2 are given by

$$f_i(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{m}{2}} |\Sigma|^{\frac{1}{2}}} exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)\right] \text{ for } \quad i = 1, 2, \tag{2.17}$$

for $\mathbf{x} \in \mathbb{R}^m$. Then by substituting (2.17) into Theorem (2.1.1), we get the following minimum ECM regions:

$$R_1 : exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)\right] > \frac{c(1|2)}{c(2|1)} \times \frac{p_2}{p_1}, \tag{2.18}$$

$$R_2 : exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)\right] < \frac{c(1|2)}{c(2|1)} \times \frac{p_2}{p_1}. \tag{2.19}$$

Given the regions R_1 and R_2, we can construct the following classification rule:

Theorem 2.1.2. *[4] Let the populations π_1 and π_2 be described by multivariate normal densities with known parameters $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and $\Sigma_1 = \Sigma_2 = \Sigma$. Then the allocation rule that minimizes the ECM is as follows:*
Allocate $\boldsymbol{x}$ to π_1 if

$$(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\Sigma^{-1}\boldsymbol{x} - \frac{1}{2}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2) > ln\left[\frac{c(1|2)}{c(2|1)} \times \frac{p_2}{p_1}\right].$$

Else, allocate $\boldsymbol{x}$ to π_2. And as before, equality implies that the data point is unclassifiable.

Proof. Note that

$$-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_2), \tag{2.20}$$

$$= \frac{1}{2}\left[(\mathbf{x}^T\Sigma^{-1} - \boldsymbol{\mu}_2^T\Sigma^{-1})(\mathbf{x}-\boldsymbol{\mu}_2) - (\mathbf{x}^T\Sigma^{-1} - \boldsymbol{\mu}_1^T\Sigma^{-1})(\mathbf{x}-\boldsymbol{\mu}_1)\right], \tag{2.21}$$

$$= \frac{1}{2}\Bigg[\mathbf{x}^T\Sigma^{-1}\mathbf{x} - \mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x} + \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2$$
$$- \mathbf{x}^T\Sigma^{-1}\mathbf{x} + \mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_1^T\Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1\Bigg]. \tag{2.22}$$

Here, we can cancel out the $\mathbf{x}^T\Sigma^{-1}\mathbf{x}$ term and moreover, since $(\mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_2)^T = \boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x}$ is a constant,

$$-\mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x} = -2\boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x}, \tag{2.23}$$

$$\mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_1^T\Sigma^{-1}\mathbf{x} = 2\boldsymbol{\mu}_1^T\Sigma^{-1}\mathbf{x}. \tag{2.24}$$

(Recall that Σ is a symmetric matrix.) Substituting equation (2.23) and (2.24) into equation (2.22), we see,

$$\frac{1}{2}\Bigg[\mathbf{x}^T\Sigma^{-1}\mathbf{x} - \mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x} + \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2$$

$$-\mathbf{x}^T\Sigma^{-1}\mathbf{x} + \mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_1^T\Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1\Bigg] =$$

$$\frac{1}{2}\Bigg[2\boldsymbol{\mu}_1^T\Sigma^{-1}\mathbf{x} - 2\boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x} + \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1\Bigg] = \tag{2.25}$$

$$(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2). \tag{2.26}$$

Consequently, after taking the natural log of (2.18), the desired allocation rule follows. □

However, theorem (2.1.2) supposes that the values of $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma$ are known, which is almost never the case. Thus, we need to use estimates from our data. Suppose then we have a set of training data, $\{\mathbf{x}_{ij}\}$, for the i^{th} ($i = 1$ or 2) population and the j^{th} data value (j from 1 to n_1 or n_2). Then the *sample mean vector* is calculated by averaging the data vectors from each class, i.e. $\bar{\mathbf{x}}_i = \frac{\sum_{j=1}^{n_i}\mathbf{x}_{ij}}{n_i}$. Moreover, we will calculate the covariance matrices for each population by $S_i = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}(\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)(\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)^T$. (Note that $\bar{\mathbf{x}}_i$ is a $(m \times 1)$ vector and S_i is a $(m \times m)$ matrix.) Since we assume that the two covariance matrices are equal, we will pool the two sample covariance matrices to derive an unbiased estimate of Σ:

$$\begin{aligned} S_{pooled} &= \left[\frac{n_1 - 1}{(n_1 - 1) + (n_2 - 1)}\right]S_1 + \left[\frac{n_2 - 1}{(n_1 - 1) + (n_2 - 1)}\right]S_2, \\ &= \frac{(n_1 - 1)S_1 + (n_2 - 1)S_2}{n_1 + n_2 - 2}. \end{aligned}$$

Substituting our estimates into our allocation rule yields:

Theorem 2.1.3. *[4] Allocate $\boldsymbol{x}$ to π_1 if*

$$(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_{pooled}^{-1}\boldsymbol{x} - \frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_{pooled}^{-1}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2) > ln\left(\frac{c(1|2)}{c(2|1)} \times \frac{p_2}{p_1}\right). \tag{2.27}$$

Note that once we use parameter estimates rather than the known quantities, then the derived allocation rule is an estimate of the optimal allocation rule. However, if sample sizes are large, then it is reasonable to expect the estimate to be close enough to the optimal rule.

Lastly, let us consider the case in which the cost ratios and prior probability ratios are equal

$\left(\text{i.e., } \frac{c(1|2)}{c(2|1)} \times \frac{p_2}{p_1} = 1\right)$, changing the equation in Theorem (2.1.3) to

$$(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T S_{pooled}^{-1} \mathbf{x} - \frac{1}{2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T S_{pooled}^{-1}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2) > ln[1] = \tag{2.28}$$

$$(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T S_{pooled}^{-1} \mathbf{x} - \frac{1}{2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T S_{pooled}^{-1}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2) > 0 \tag{2.29}$$

$$\Rightarrow (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T S_{pooled}^{-1} \mathbf{x} > \frac{1}{2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T S_{pooled}^{-1}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2) \tag{2.30}$$

If we denote $(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T S_{pooled}^{-1}$ as a row vector, say $\hat{\mathbf{l}}$, then we can write (2.30) as

$$\hat{\mathbf{l}}\mathbf{x} > \frac{1}{2}\hat{\mathbf{l}}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2). \tag{2.31}$$

In essence, we can think of the decision rule as reducing the number of dimensions of our data to one by projecting it on our vector $\hat{\mathbf{l}}$, and allocating a new point $\mathbf{x}$ by where it lies relative to the midpoint of the two classes, $\frac{1}{2}\hat{\mathbf{l}}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2)$.

2.2 Fisher's Discriminant Analysis

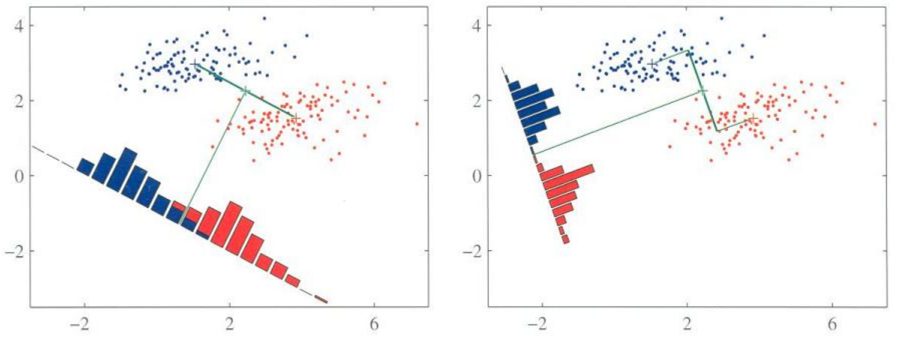

Figure 4.6 The left plot shows samples from two classes (depicted in red and blue) along with the histograms resulting from projection onto the line joining the class means. Note that there is considerable class overlap in the projected space. The right plot shows the corresponding projection based on the Fisher linear discriminant, showing the greatly improved class separation.

Figure 2.1: Fisher's Discriminant Analysis: Striking a balance between class separation and in-class variance, [2].

R.A. Fisher proposed another method for linear discriminant analysis that did not presuppose any known distribution of the training data. He does, however, begin with the idea of dimensionality reduction shown above. Again, suppose that we have our training data set $\{\mathbf{x}_{ij}\}$ as defined above, where populations π_1 and π_2 don't necessarily have the same distribution but have equal covariance matrices $\Sigma_1 = \Sigma_2 = \Sigma$. Then, if given a vector $\mathbf{w}$ with unit length, we can project our set in the direction of $\mathbf{w}$ by taking the dot product of the two vectors : $\{\mathbf{w} \cdot \mathbf{x}_{ij}\}$.

Onto what kind of vector $\mathbf{w}$ do we want to project our data? To make a good classifier, the two projected class means ought to be far apart, i.e. we want to make $\mathbf{w}^T(\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1)$ large. (To find a bounded solution, we specify that we want $||\mathbf{w}|| = 1$.) That way the differences between the two classes are maximally highlighted and evaluating the projection of a new point against the projected sample means can accurately classify the largest number of points. The goal of being able to classify the largest number of unknown points accurately based on the training data is called the *generalization ability* and motivates Support Vector Machines, the next chapter.

But separating the projected class means is only one concern, since if the classes have a large variance on the projected vector, then the class overlap can potentially make classification inaccurate. Refer to the figure on the opposite page. The picture on the left projects the data onto the vector connecting the two sample means, assuring that the distance between the two projected means is maximized. However, since the two projected classes are not partitioned on the projection vector, the problem of classifying new points is no easier. However, in the picture on the right, the chosen projection vector separates the projected sample means *and* partitions the two classes on the projection vector. Classification of a new point $\mathbf{x}_0$ then is based on comparing the projection of $\mathbf{x}_0$ to the midpoint between the two projected sample means. To tend toward a vector that partitions our classes, we want to minimize the within-class variance, i.e. we want to minimize

$$\mathbf{w}^T S_{pooled}\mathbf{w} = \frac{\mathbf{w}^T \left(\sum_{j=1}^{n_1}(\mathbf{x}_{1j} - \bar{\mathbf{x}}_1)(\mathbf{x}_{1j} - \bar{\mathbf{x}}_1)^T + \sum_{j=1}^{n_2}(\mathbf{x}_{2j} - \bar{\mathbf{x}}_2)(\mathbf{x}_{2j} - \bar{\mathbf{x}}_2)^T \right)\mathbf{w}}{n_1 + n_2 - 2}.$$

(Recall that $\mathbf{w}^T S_{pooled}\mathbf{w}$ is the weighted sum of the variances of the two projected populations.)

We can optimize the difference between projected means and the in-class variance in one equation so that we can find our desired projection vector:

$$\text{Maximize } J(\mathbf{w}) = \frac{(\mathbf{w}^T(\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1))^2}{\mathbf{w}^T S_{pooled}\mathbf{w}} \text{ such that } ||\mathbf{w}|| = 1. \tag{2.32}$$

We then prove the maximization lemma to show that Fisher's Discriminant Analysis gives $\mathbf{w}\ \alpha\ (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T S_{pooled}^{-1}$.

Lemma 2.2.1. *(**Maximization Lemma**,[4]): For B a positive definite matrix, $\boldsymbol{d}$ a given vector, and $\boldsymbol{x}$ a non zero arbitrary vector,*

$$\max \frac{(\boldsymbol{x}^T\boldsymbol{d})^2}{\boldsymbol{x}^T B\boldsymbol{x}} = \boldsymbol{d}^T B^{-1}\boldsymbol{d} \text{ such that } ||\boldsymbol{w}|| = 1, \tag{2.33}$$

with the maximum attained when $\boldsymbol{x} = cB^{-1}\boldsymbol{d}$ for some constant $c \neq 0$.

Proof. By using the extended Cauchy-Schwarz inequality, $(\mathbf{x}^T\mathbf{d})^2 \leq (\mathbf{x}^T B\mathbf{x})(\mathbf{d}^T B^{-1}\mathbf{d})$. Dividing both sides by $(\mathbf{x}^T B\mathbf{x})$ yields $\dfrac{(\mathbf{x}^T\mathbf{d})^2}{(\mathbf{x}^T B\mathbf{x})} \leq \mathbf{d}^T B^{-1}\mathbf{d}$. Thus the maximum is $\mathbf{d}^T B^{-1}\mathbf{d}$, as desired. Setting the equality and solving for $\mathbf{x}$ yields $\mathbf{x} = cB^{-1}\mathbf{d}$. □

Note that this is not a classification method in and of itself, because Fisher's procedure only find the optimal vector to project data onto. However, as mentioned above, classification should compare the projection of new points to the midpoint between the two projected sample means:

Theorem 2.2.2. *(Allocation Rule Based on FDA): Classify an unclassified point $\boldsymbol{x}_0$ to π_1 if*

$$y_0 = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_{pooled}^{-1} \boldsymbol{x}_0 > \hat{m} = \frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_{pooled}^{-1}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2), \tag{2.34}$$

to π_2 if

$$y_0 = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_{pooled}^{-1} \boldsymbol{x}_0 < \hat{m} = \frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_{pooled}^{-1}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2), \tag{2.35}$$

and unclassifiable if equality holds.

Recall that in the previous section, we defined a vector $\hat{\mathbf{l}} = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T S_{pooled}^{-1}$ and showed that given no cost or prior probability concerns, Linear Discriminant Analysis could be interpreted as projecting an unknown datum onto a vector and then comparing the projection of the unknown point to the midpoint between the two projected sample means. Thus, we can see that Fisher's method is identical to assuming that the two populations are distributed similarly and with equal covariance matrices, given no cost or prior probability concerns.

We should mention that the assumption of equal covariance matrices is important, because that justifies our choice of the midpoint as the classification critical value. Consider the trivial example shown on the next page. Here the two populations have clearly difference variances, and thus one population occupies a greater portion of the projection vector. Then the midpoint is a poor choice for a critical value, and an accurate decision function will have its critical value closer to μ_1.

2.3 Further Concerns

In this chapter, we have looked at two basic methods of linear discrimination, one that is parametric and another that is non-parametric. However, we only covered cases where the two within-class variances were equal ($\Sigma_1 = \Sigma_2 = \Sigma$). We have just seen that Fisher's Discriminant Analysis fails when the two class variances are different, but Linear Discriminant Analysis can account for this by directly plugging the two covariance matrices in separately to the pdfs, $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$. Another issue that was not covered in this section is the difference between the estimated allocation and the optimal allocation when using LDA. The interested reader can find plenty of literature outlining the difference, and a good place to start is section 11.4 in [4].

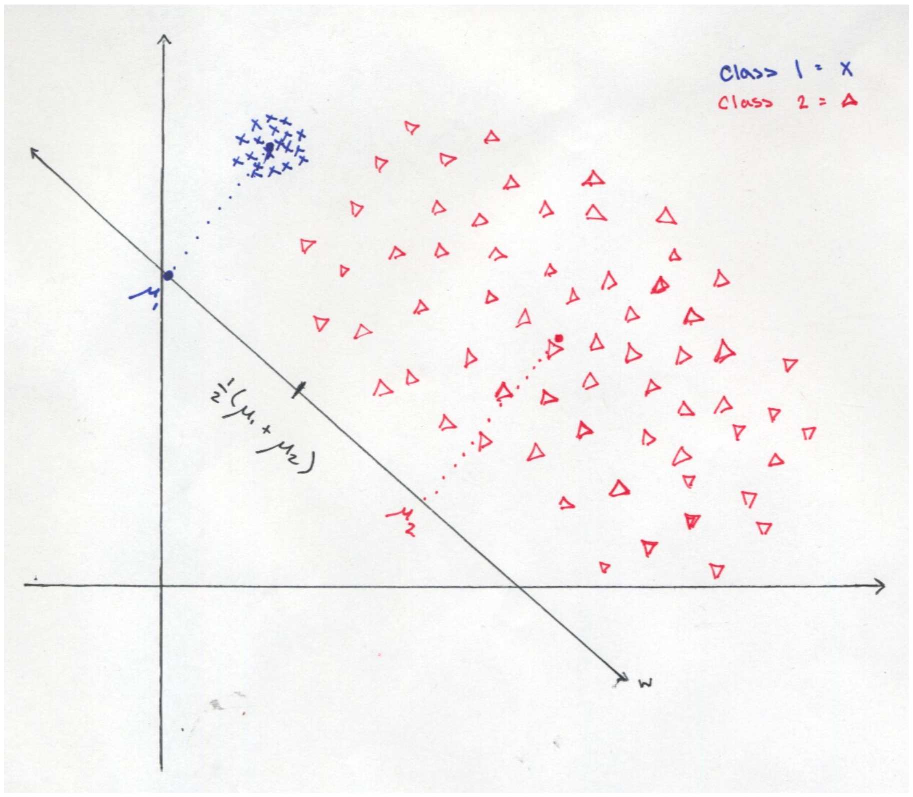

Figure 2.2: A trivial example of two unequal class variances.

Chapter 3

Support Vector Machines

In training any classifier, an obvious goal would be to correctly classify our training data. But if we create a classifier that is too well fit to our data, we run the risk of developing a model that performs poorly when asked to classify unknown data, a condition known as ***overfitting***. This is a particularly important concern when the available amount of training data is small or suspected to be a poor representation of the general population. The motivation behind Support Vector Machines is to maximize the *generalization ability*, meaning that it finds the classifier that most accurately can predict unknown data points based on the training data.

For now, we focus on case of trying to distinguish between two classes.

3.1 Lagrange Multipliers

Before discussing SVMs, it is useful to review Lagrange Multipliers, as they play a major role in the construction of SVMs. In short, *Lagrange Multipliers* are constants that help solve constrained maximization/minimization problems.

Suppose we want to find the maximum value for a given function $f(\mathbf{x})$ subject to a constraint $g(\mathbf{x}) = 0$. Geometrically, if $\mathbf{x} \in \mathbb{R}^m$, then the constraint $g(\mathbf{x}) = 0$ represents some $m-1$ dimensional surface in $\mathbb{R}^m$.

Lemma 3.1.1. *$\nabla g(\boldsymbol{x})$ is orthogonal to $g(\boldsymbol{x})$ for all $\boldsymbol{x}$ on the surface $g(\boldsymbol{x}) = 0$. [2]*

Proof. Consider the linear approximation of $g(\mathbf{x})$: $g(\mathbf{x}+\boldsymbol{\epsilon}) \cong g(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla g(\mathbf{x})$ for two points $\mathbf{x}, \mathbf{x}+\boldsymbol{\epsilon}$ that are on the surface $g(\mathbf{x}) = 0$. Since $g(\mathbf{x}) = 0 = g(\mathbf{x}+\boldsymbol{\epsilon})$, we know $\boldsymbol{\epsilon}^T \nabla g(\mathbf{x}) \cong 0$. Moreover, as $||\boldsymbol{\epsilon}||$ approaches 0, $\boldsymbol{\epsilon}$ becomes parallel to the surface $g(\mathbf{x}) = 0$ and so we conclude that $\nabla g(\mathbf{x})$ is orthogonal to the surface. $\square$

Moreover, if $f(\mathbf{x})$ is maximized on $g(\mathbf{x}) = 0$, then $\nabla f(\mathbf{x})$ is also orthogonal to $g(\mathbf{x}) = 0$. Recall

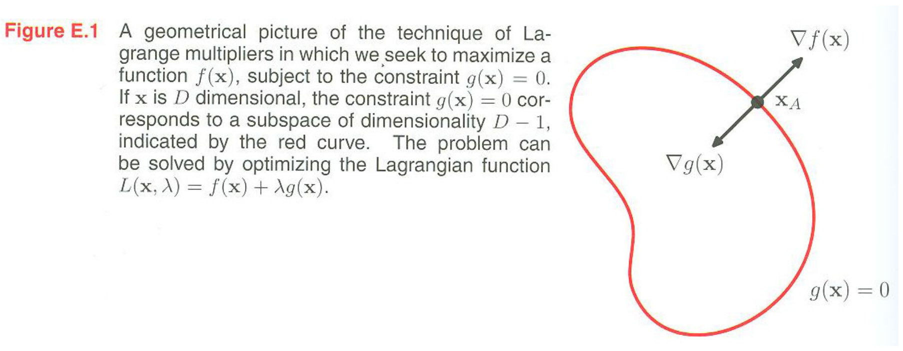
Figure E.1 A geometrical picture of the technique of Lagrange multipliers in which we seek to maximize a function $f(\mathbf{x})$, subject to the constraint $g(\mathbf{x}) = 0$. If $\mathbf{x}$ is D dimensional, the constraint $g(\mathbf{x}) = 0$ corresponds to a subspace of dimensionality $D - 1$, indicated by the red curve. The problem can be solved by optimizing the Lagrangian function $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$.

Figure 3.1: Picture from Bishop, [2].

that $\nabla f(\mathbf{x})$ is the direction in which $f(\mathbf{x})$ increases the fastest. If $\nabla f(\mathbf{x})$ is not orthogonal to $g(\mathbf{x}) = 0$, then it is possible to move along $g(\mathbf{x}) = 0$ in the direction of $\nabla f(\mathbf{x})$, increasing the value $f(\mathbf{x})$. Thus, there must exist some constant $\lambda \neq 0$ so that

$$\nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = 0. \tag{3.1}$$

Note that λ can have either sign.

We can create a new function, called the *Lagrange Function*, $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$, and by finding the stationary points of the Lagrange Function, i.e. $\nabla_{\mathbf{x},\lambda} L = 0$, we can solve for the point $\mathbf{x}_0$ that maximizes $f(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$. Note that $\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda) = 0$ gives us the constraints given by equations (3.1), and $\nabla_\lambda L(\mathbf{x}, \lambda) = 0$ gives us our original constraint, $g(\mathbf{x}) = 0$.

Lagrange Multipliers also solve optimization problems with constraints using inequalities. Again, suppose we wish to maximize $f(\mathbf{x})$ subject to the constraint $g(\mathbf{x}) \geq 0$. There are now two possible solutions. If the solution $\mathbf{x}$ satisfies $g(\mathbf{x}) > 0$, the constraint is called *inactive*, as the same solution $\mathbf{x}$ can be found by optimizing $f(\mathbf{x})$ without any constraint. This solution corresponds to the stationary point where $\nabla f(\mathbf{x}) = 0$, meaning that $\lambda = 0$ for an inactive constraint. However, if the optimal solution $\mathbf{x}$ satisfies $g(\mathbf{x}) = 0$, the constraint is called *active*, and is identical to the previous situation where $\lambda \neq 0$. However, now the sign of λ is significant, as $f(\mathbf{x})$ is maximized when $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ point in opposite directions. (If $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ did not point in different directions, it would be possible to find an new $\mathbf{x}'$ that both $f(\mathbf{x})$ and $g(\mathbf{x})$ are increased, violating the optimality of $\mathbf{x}$.) We can write this new fact as

$$\nabla f(\mathbf{x}) = -\lambda \nabla g(\mathbf{x}), \tag{3.2}$$

where $\lambda \geq 0$.

For an active or inactive constraint, we know that $\lambda \cdot g(\mathbf{x}) = 0$. So the $\mathbf{x}$ that maximizes $f(\mathbf{x})$ subject to the constraint $g(\mathbf{x}) \geq 0$ is the $\mathbf{x}$ that satisfies equations (3.1) and

$$g(\mathbf{x}) \geq 0, \tag{3.3}$$
$$\lambda \geq 0, \tag{3.4}$$
$$\lambda g(\mathbf{x}) = 0. \tag{3.5}$$

These are called the *Karush-Kuhn-Tucker*(KKT) conditions. [2]

Alternatively, minimizing $f(\mathbf{x})$ subject to $g(\mathbf{x}) \geq 0$ is equivalent to minimizing the Lagrangian function $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$ subject to the same KKT conditions.(Equations (3.1),(3.3),(3.4),(3.5))

Figure E.3 Illustration of the problem of maximizing $f(\mathbf{x})$ subject to the inequality constraint $g(\mathbf{x}) \geqslant 0$.

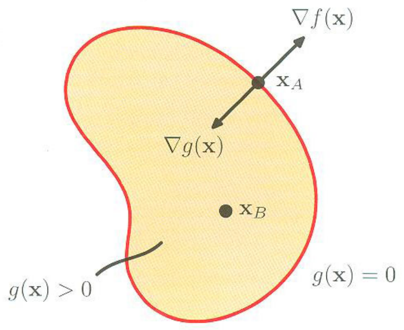

Figure 3.2: Picture from Bishop, [2].

Finally, multiple constraints can be added to an optimization problem by introducing more Lagrange multipliers: To maximize $f(\mathbf{x})$ subject to $g_i(\mathbf{x}) = 0$ constraints for $i \in \{1, \ldots, k\}$ and $h_j(\mathbf{x}) \geq 0$ for $j \in \{1, \ldots, l\}$, maximize the value of the Lagrangian function

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{l} \mu_j h_j(\mathbf{x}), \tag{3.6}$$

subject to the following conditions:

$$\begin{aligned} h_j(\mathbf{x}) &\geq 0, \\ \mu_j &\geq 0, \\ \mu_j \cdot h_j(\mathbf{x}) &= 0. \end{aligned}$$

3.2 Hard Margin Support Vector Machines

Let N m-dimensional training inputs $\mathbf{x}_i$ for $i \in \{1, 2, \ldots, N\}$ belong to either population 1 or population 2, and the associated labels y_i be 1 for population 1 and -1 for population 2. Moreover, assume that the data points are ***linearly separable***, meaning that there exists some m-dimensional hyperplane that separates the data points into the two groups. More specifically, we can find a decision function

$$D(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b,$$

where $\mathbf{w}$ is an m-dimensional vector and b is a constant bias term so that

$$D(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \begin{cases} > 0 & \text{for} \quad y_i = 1, \\ < 0 & \text{for} \quad y_i = -1. \end{cases} \tag{3.7}$$

(Note that $D(\mathbf{x})$ is of the form of a hyperplane.) Since we assumed linear separability, we can rewrite our decision function. Let $A = \{a_i\}$ denote the set of all constants such that $y_i(\mathbf{w}^T \mathbf{x}_i + b) = a_i$;

taking the minimum of the set A, we can divide through by the minimum value, redefine the values for $\mathbf{w}$ and b and rewrite (3.7) as

$$D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \begin{cases} > 1 & \text{for} \quad y_i = 1 \\ < -1 & \text{for} \quad y_i = -1, \end{cases}$$

or alternatively,

$$y_i(\mathbf{w}^T\mathbf{x} + b) > 1. \tag{3.8}$$

Note then that the hyperplane

$$D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = c, \quad -1 < c < 1 \tag{3.9}$$

separates the two populations. Furthermore, the hyperplane with $c = 0$ lies between the hyperplanes with $c = 1$ and $c = -1$.

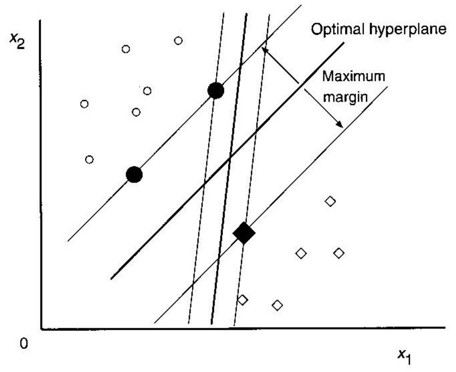

Figure 3.3: Picture from Abe, [1].

The distance between the nearest datum and the hyperplane $D(\mathbf{x})$ is called the ***margin***, and as we divided out by the minimum of our set A, we know that at least one datum is located on the hyperplane $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \pm 1$. Thus, the ***optimal separating hyperplane***, or the hyperplane that maximizes the margin, for $-1 < c < 1$ is $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$. Lastly, consider the region $\{\mathbf{x}| -1 < D(\mathbf{x}) < 1\}$. None of the training data is within the margin, because of our assumption of linear separability. This region is called the *generalizable region* because based on the training data, we can extend membership in either class 1 or 2 up to the decision hyperplane. Note that we picked our optimal separating hyperplane to be the one that falls in the middle of the margin, $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$, so that we don't favor classifying points into one class over the other.

Before continuing further, we should discuss some properties of the optimal separating hyperplane. Given a set of linearly separable data, it should be clear that infinitely many separating hyperplanes can be drawn in between the two populations. Moreover, it is just as clear that not all separating hyperplanes have the same generalization ability. In fact, if the data have no outliers and we assume the unknown data will obey the same probability law as that of the training data, then the generalization ability will be maximized if the optimal separating hyperplane is selected as the separating hyperplane.

Lemma 3.2.1. *(Abe, [1].) Let $\widehat{D}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0$ be the optimal separating hyperplane and we want to show that the Euclidean distance between $\widehat{D}(\boldsymbol{x})$ and a datum $\boldsymbol{x}_i$ is given by $\frac{|\widehat{D}(\boldsymbol{x}_i)|}{||\boldsymbol{w}||}$.*

Proof. Note that any two vectors $\mathbf{a}, \mathbf{b}$ are orthogonal if their dot product is zero. Consider then the hyperplane $D(\mathbf{x}) = 0$ with no bias term, namely $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = 0$. This hyperplane contains all vectors that product with the vector $\mathbf{w}$ to equal zero, and thus we can say that $\mathbf{w}$ is orthogonal to the hyperplane. Thus, the line that connects $\mathbf{x}_0$ and is orthogonal to the hyperplane is given by

$$\frac{a\mathbf{w}}{||\mathbf{w}||} + \mathbf{x}_0, \tag{3.10}$$

where $|a|$ is the Euclidean distance from $\mathbf{x}_0$ to the hyperplane. (Note that we can see this if we understand $\frac{\mathbf{w}}{||\mathbf{w}||}$ as the "slope" of the line, and since we divide by $||\mathbf{w}||$, we reduce the vector $\mathbf{w}$ down to a direction and we are taking $|a|$ steps toward the hyperplane in that direction.) This line (3.10) crosses $D(\mathbf{x})$ precisely at the point where

$$D\left(\frac{a\mathbf{w}}{||\mathbf{w}||} + \mathbf{x}_0\right) = 0. \tag{3.11}$$

Solving (3.11) for a, we see

$$\begin{aligned} D\left(\frac{a\mathbf{w}}{||\mathbf{w}||} + \mathbf{x}_0\right) = \mathbf{w}^T\left(\frac{a\mathbf{w}}{||\mathbf{w}||} + \mathbf{x}_0\right) &= \frac{a\mathbf{w}^2}{||\mathbf{w}||} + \mathbf{w}^T\mathbf{x}_0 \\ &= a||\mathbf{w}|| + D(\mathbf{x}_0) = 0 \\ \Rightarrow a &= \frac{-D(\mathbf{x}_0)}{||\mathbf{w}||}. \end{aligned}$$

Thus the case when $D(\mathbf{x})$ has no bias term is shown.

Now suppose that $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$. We claim that the vectors in $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ are simply the vectors in $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ translated by $-\frac{b\mathbf{w}}{||\mathbf{w}||^2}$, which is in the direction of $\mathbf{w}$, preserving orthogonality.

Claim: If $\mathbf{x}_0 \in \{\mathbf{x}|D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = 0\}$, then $\mathbf{x}_0 - \frac{b\mathbf{w}}{||\mathbf{w}||^2} \in \{\mathbf{x}|D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0\}$.

We need to show that $D(\mathbf{x}_0 - \frac{b\mathbf{w}}{||\mathbf{w}||^2}) = 0$.

$$\begin{aligned} D\left(\mathbf{x}_0 - \frac{b\mathbf{w}}{||\mathbf{w}||^2}\right) &= \mathbf{w}^T\left(\mathbf{x}_0 - \frac{b\mathbf{w}}{||\mathbf{w}||^2}\right) + b \\ &= \mathbf{w}^T\mathbf{x}_0 - \mathbf{w}^T\frac{b\mathbf{w}}{||\mathbf{w}||^2} + b \qquad = 0 + -\frac{b\mathbf{w}^2}{||\mathbf{w}||^2} + b = -b + b = 0. \end{aligned}$$

Denote $\mathbf{x}_0 - \frac{b\mathbf{w}}{||\mathbf{w}||^2}$ as some vector $\hat{\mathbf{x}}_0$. Thus, since $\mathbf{w}$ is orthogonal to the vectors in $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$, the line that connects $\hat{\mathbf{x}}_0$ and $D(\mathbf{x})$ is again given by

$$\frac{a\mathbf{w}}{||\mathbf{w}||} + \hat{\mathbf{x}}_0, \tag{3.12}$$

which crosses our new hyperplane $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$ when

$$D\left(\frac{a\mathbf{w}}{||\mathbf{w}||} + \hat{\mathbf{x}}_0\right) = 0. \tag{3.13}$$

Again solving (3.13) for a yields

$$D\left(\frac{a\mathbf{w}}{||\mathbf{w}||}+\hat{\mathbf{x}}_0\right)=\mathbf{w}^T\left(\frac{a\mathbf{w}}{||\mathbf{w}||}+\hat{\mathbf{x}}_0\right)+b=\frac{a\mathbf{w}^2}{||\mathbf{w}||}+\mathbf{w}^T\hat{\mathbf{x}}_i+b \tag{3.14}$$

$$=\frac{a}{||\mathbf{w}||}+D(\hat{\mathbf{x}})=0 \tag{3.15}$$

$$\Rightarrow a=\frac{-D(\hat{\mathbf{x}}_0)}{||\mathbf{w}||}, \tag{3.16}$$

the desired result. □

If the groups are linearly separable, all the training data $\{\mathbf{x}_i\}$ for $i \in \{1,\dots,N\}$ must satisfy

$$\frac{y_i\widehat{D}(\mathbf{x}_i)}{||\mathbf{w}||}\geq\delta, \tag{3.17}$$

where δ is the margin, the minimum distance from a training datum to the hyperplane $\widehat{D}(\mathbf{x})=0$. Note that if $(\mathbf{w}, b)$ is a solution for the coefficients of the optimal hyperplane, then $(c\mathbf{w}, cb)$ is also a solution, where c is a non zero constant. So we also impose the following constraint:

$$\delta||\mathbf{w}||=1. \tag{3.18}$$

The implication of (3.18) is that the larger the Euclidean norm of $\mathbf{w}$, the linear coefficient of our hyperplane, the smaller our margin δ will be. As we want to maximize the generalization region by maximizing the margin, we are searching for the vector $\mathbf{w}$ with the smallest norm such that (3.8) is satisfied. To find such a $\mathbf{w}$, we will minimize the function

$$Q(\mathbf{w})=\frac{1}{2}||\mathbf{w}||^2, \tag{3.19}$$

with respect to $\mathbf{w}$ and b subject to the constraints given by (3.8):

$$y_i(\mathbf{w}^T\mathbf{x}+b)>1.$$

The use of the square of the Euclidean norm in (3.19) is to make the optimization problem a quadratic programming problem, and since we assumed linear separability, there must exist $\mathbf{w}$ and b that satisfy (3.19) and (3.8). An advantage of quadratic programming is that even though many distinct solutions may exist, they all give the same value for (3.19), meaning that the different solutions give the same margin.

Note that the same optimal separating hyperplane will be reached if we only consider those data points on the margin, namely those $\mathbf{x}_i$ that satisfy

$$D(\mathbf{x}_i)=\mathbf{w}^T\mathbf{x}_i+b=\pm1. \tag{3.20}$$

These data points are called *support vectors*, and SVMs are called *sparse learning machines* since they are able to classify new points only using a subset of the original data. As we will see later, the definition of support vectors can be refined further, as some points that satisfy equation (3.20) can also be deleted without changing the optimal hyperplane.

Consider the optimization problem given by (3.19) and (3.8). The variables of the convex optimization problem are given by $\mathbf{w}$ and b, meaning that the number of variables is the dimension of the training data, equivalently the dimension of $\mathbf{w}$, plus 1 for b: $m+1$. When the number of features we extract from the training data is small, we can solve this optimization problem by quadratic programming. However, there are instances in which we want to project the input data into a higher dimensional space, sometimes infinite, so it is necessary to construct the a dual optimization problem of (3.19) and (3.8). The construction below will use Lagrange multipliers, and more importantly, limit the decision variables in the dual by the number of training data available.

We first combine (3.19) and (3.8) into a Lagrangian function:

$$Q(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i\{y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\}, \tag{3.21}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ are non negative Lagrange multipliers. The optimal solution is given by the saddle point of (3.21), where (3.21) is minimized with respect to $\mathbf{w}$ and b, but maximized with respect to $\boldsymbol{\alpha}$, meaning it satisfies the KKT conditions:

$$\frac{\partial Q(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = 0; \tag{3.22}$$

$$\frac{\partial Q(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0; \tag{3.23}$$

$$\alpha_i\{y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\} = 0, \tag{3.24}$$

$$\alpha_i \geq 0, \tag{3.25}$$

for all $i \in \{1, \ldots, N\}$. From (3.24), we see that for every training data point $\mathbf{x}_i$, either $\alpha_i = 0$, or both $\alpha_i \neq 0$ and $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$ must hold. We define a ***support vector*** to be those data points $\mathbf{x}_i$ such that $\alpha_i \neq 0$. Note that this definition excludes the cases in which $\alpha_i = 0$ and $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$.

Using (3.21), we can evaluate (3.22) and (3.23) to yield:

$$w = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i, \tag{3.26}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i. \tag{3.27}$$

Finally, we can plug (3.26) and (3.27) back into (3.21) to create our dual formulation

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j, \tag{3.28}$$

subject to $\alpha_i \geq 0 \ \forall \ i \in \{1, \ldots, N\}$ and (3.27). This dual formulation of a support vector machine is called a ***Hard Margin Support Vector Machine***[1]. (Hard-Margin follows from the assumption of linear separability, no points in the generalizable region, meaning that there exists a strict margin.)

Using our earlier definition of support vectors, we then create a decision function

$$D(x) = \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b, \tag{3.29}$$

where S denotes the set of support vectors, and from the KKT condition given by (3.23) applied to (3.21), we see

$$b = \frac{1}{n(S)} \sum_{i \in S} (y_i - \mathbf{w}^T \mathbf{x}_i), \tag{3.30}$$

where $n(S)$ is the number of support vectors. Then, given an unknown datum $\widehat{x}$, it is classified in

$$\begin{cases} \text{Population 1} & \text{if} \quad D(\widehat{\mathbf{x}}) > 0, \\ \text{Population 2} & \text{if} \quad D(\widehat{\mathbf{x}}) < 0. \end{cases} \tag{3.31}$$

Note that if $D(\widehat{x}) = 0$, $\widehat{x}$ is on the boundary and thus is unclassifiable. (However, this happens with probability 0.)

In essence, we can think of this decision function as a process of evaluating how similar a test point is to a given support vector. Recall that the dot product between two vectors, **a** and **b** is

$$\mathbf{a} \cdot \mathbf{b} = ||\mathbf{a}|| \cdot ||\mathbf{b}|| cos(\theta),$$

where θ is the angle in between the two vectors. So in a SVM, if the angle between a unknown point and a support vector is large and close to 90 degrees, meaning the two vectors differ significantly in their directions, then $cos(\theta)$ will be close to zero and that support vector will only contribute a small amount to the end decision. On the other hand, when θ is small, the test vector and support vector point in similar directions, and $cos(\theta)$ will be close to 1, letting that support vector influence the decision more.

However, there are instances when $cos(\theta)$ can mistake two very different points as similar. Take for example the adjacent figure, which shows two classes in a positively sloped orientation. Notice that the test point has a similar direction to points in both classes, but since the magnitude of vectors in the class further from the origin are greater, those support vectors will have a greater influence in the decision.

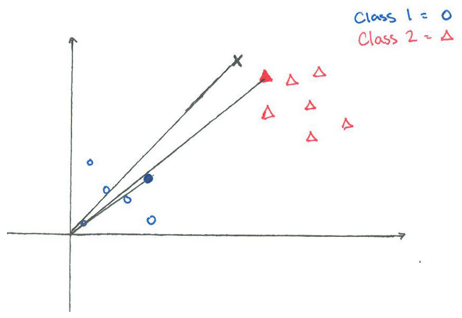

Figure 3.4: A case where $\cos(\theta)$ is not enough to evaluate similarity.

Finally, we turn our attention to the Lagrange multipliers, $\{\alpha_i\}$. These α's can be thought of as "weights" that we assign the different support vector, meaning that similarity to some support vectors is more important that to other support vectors. Thus, given an unknown test point $\widehat{\mathbf{x}}$, the Support Vector Machine decision function numerically "scores" $\widehat{\mathbf{x}}$ against the set of all support vectors, and depending on the sign of the sum of all of the scores, assigns $\widehat{\mathbf{x}}$ to either class 1 or 2.

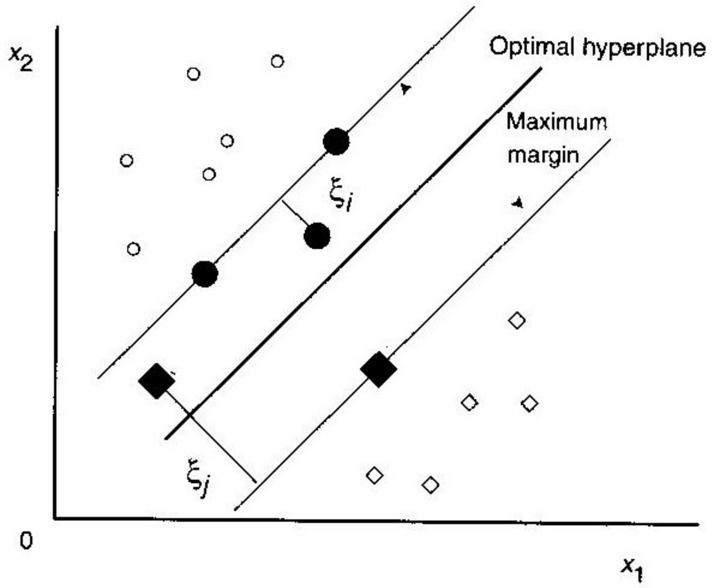

Figure 3.5: Soft Margin Support Vector Machines with *slack variables*, [1]

3.3 L1 Soft Margin Support Vector Machines

In the previous section, we assumed a largely infeasible claim, namely that our data set was linearly separable. In real life, such data sets are hard to come by and largely uninteresting. However, there are a a number of methods we can employ to deal with more realistic situations. One of them is simply to allow some data to cross the separating hyperplane, using slack variables to account for the effect on the margin. Instead of the constraints given by linear separability, equations (3.8), consider the following constraints;

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) > 1 - \xi_i. \tag{3.32}$$

Here, if $\xi_i = 0$, then the datum $\mathbf{x}_i$ is not inside the margin, but if $\xi_i > 0$, then the associated $\mathbf{x}_i$ is inside the margin. Moreover, $\xi_i \geq 1$ implies that the datum $\mathbf{x}_i$ is on the wrong side of the the separating hyperplane, and $\mathbf{x}_i$ would be *incorrectly* classified by our decision function.

As we want to limit the instances of training data misclassifications, our optimization problem now becomes

$$Q(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i^p, \tag{3.33}$$

subject to

$$y_i(w^T x_i + b) > 1 - \xi_i, \tag{3.34}$$
$$\xi_i \geq 0 \tag{3.35}$$

There is a trade-off between how many points we allow in the margin to how big the generalizable region can be, since we want to minimize $Q(\mathbf{w}, b, \boldsymbol{\xi})$. To manage this trade-off, there is an additional parameter that needs to be chosen, C, which acts as a cost or penalty to data appearing inside the margin or begin misclassified. Thus, if C is small, we allow more points to violate the margin and be misclassified, whereas if C is large, the solution will shrink the margin in order to optimize (3.33). Usually p is set to either 1 or 2; for our purposes, we will set $p = 1$, resulting in a ***L1*** Soft Margin SVM. [1]

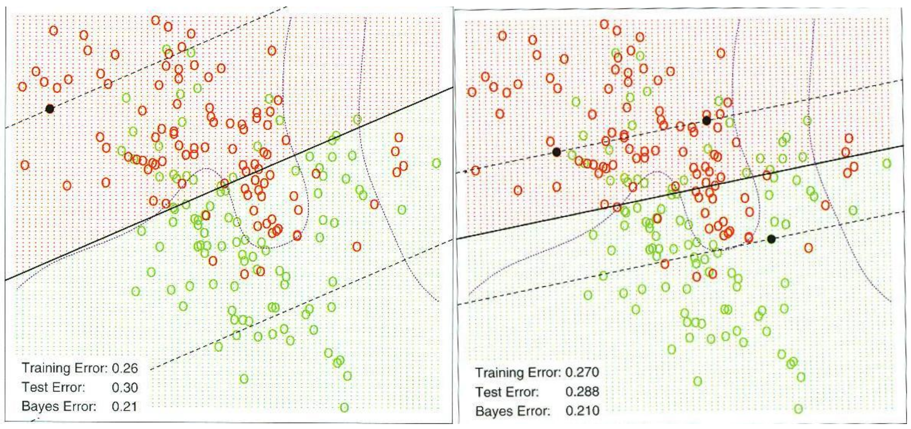

Figure 3.6: The picture on the right was generated using a $C = .01$ while the picture on the left was generated with $C = 10000$. The solid black line is the decision hyperplane and the dotted lines are the margins for two classes. On the left, 85% of the points are on the wrong side of their margin, while only 62% on the right, a result of the size of the margin. Picture from Hastie, [6].

As an aside, SVMs will find the optimum separating hyperplane that is equally far from both classes. However, when working with LDAs, one could use cost ratios and prior probability ratios to influence classification toward one class. Similarly, SVMs can be influenced to associate points to a certain class only with sufficient confidence by manipulating the cost parameter C above. Instead of having one C for all training data, each training data $\mathbf{x}_i$ can have its own cost parameter C_i, meaning $C\sum_{i=1}^{N}\xi_i^p \longrightarrow \sum_{i=1}^{N} C_i\xi_i^p$ in equation (3.33). (Although associating different costs to the different classes, C_1 and C_2, is usually sufficient.) By penalizing misclassifications of one class more than the other, the optimal separating hyperplane will allow more points of one class in the margin than the other. Referring back to the earlier example mentioned last chapter, to reduce cases of misclassifying sick patients as healthy, we can associate a cost parameter C_1 to healthy training data and C_2 to sick training data, where $C_1 \ll C_2$. Then, sick patients will have to present with very healthy features to be classified as healthy, while healthy patients with mildly sick features may be classified as sick.

Returning to the construction the dual problem to this optimization, we combine (3.34) and

(3.35) with (3.33) using Lagrange multipliers to create another Lagrangian function:

$$Q(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\{y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i\} - \sum_{i=1}^{N}\beta_i\xi_i. \tag{3.36}$$

We use two sets of Lagrange multipliers, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_N)$. The KKT conditions for (3.36) yield

$$\frac{\partial Q(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \mathbf{w}} = 0, \tag{3.37}$$

$$\frac{\partial Q(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial b} = 0, \tag{3.38}$$

$$\frac{\partial Q(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \boldsymbol{\xi}} = 0, \tag{3.39}$$

$$\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i) = 0, \tag{3.40}$$

$$\beta_i\xi_i = 0, \tag{3.41}$$

$$\alpha_i \geq 0, \quad \beta_i \geq 0, \quad \xi_i \geq 0. \tag{3.42}$$

In an identical fashion, we use our Lagrangian function (3.36) to evaluate the partial derivatives:

$$w = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i, \tag{3.43}$$

$$0 = \sum_{i=1}^{N}\alpha_i y_i, \tag{3.44}$$

$$C = \alpha_i + \beta_i \text{ for } i \in \{1, \ldots, N\}. \tag{3.45}$$

Using these, we can construct the dual optimization problem as

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \quad s.t. \ \sum_{i=1}^{n}\alpha_i y_i = 0 \ \ \alpha_i \geq 0 \tag{3.46}$$

$$and \ C \geq \alpha_i \geq 0. \tag{3.47}$$

From this, we can see that the L1 soft margin case tantamount to the hard margin case, except that α has an upper bound. Moreover, by using equations (3.40), (3.41), (3.42), and (3.45), three types of training data emerge [1]:

1. $\alpha_i = 0$. Then $\xi_i = 0$. Thus $\mathbf{x}_i$ is correctly classified and is not a support vector.

2. $0 < \alpha_i < C$. Then $y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i = 0$ and $\xi_i = 0$. (Note that (3.45) yields that $\beta_i = C - \alpha_i$.) Thus, $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$ and $\mathbf{x}_i$ is a support vector. Moreover, since $C > \alpha_i$, this is an *unbounded* support vector.

3. $C = \alpha_i$. Then $y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i = 0$ and $\xi_i \geq 0$. Thus $\mathbf{x}_i$ is again a support vector, but as α_i is bounded by C, this is a *bounded* support vector. Note that $\xi_i \geq 0$ implies that both the points that exist in the margin and get misclassified by our decision function are used as support vectors.

Going back to using intuition to understand SVMs, the upper limit on the α's means that no single support vector can influence the decision by more than C. So, even if a support vector is very clearly violating the decision boundary, the most it can contribute to the decision is capped by C.

The decision function for a soft margin SVM is identical to the hard margin case:

$$D(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i \mathbf{x}^T \mathbf{x} + b, \tag{3.48}$$

where S is the set of all support indices. However, calculating b is slightly different; we only use the unbounded support vectors, as we don't have any way to bound points that are horribly misclassified:

$$b = \frac{1}{n(U)} \sum_{i \in U} (y_i - \mathbf{w}^T \mathbf{x}_i), \tag{3.49}$$

where U is the set of unbounded support vectors. Again, an unknown datum $\widehat{\mathbf{x}}$ is classified to

$$\begin{cases} \text{Population 1} & \text{if} \quad D(\widehat{\mathbf{x}}) > 0, \\ \text{Population 2} & \text{if} \quad D(\widehat{\mathbf{x}}) < 0. \end{cases} \tag{3.50}$$

3.4 Kernel Methods

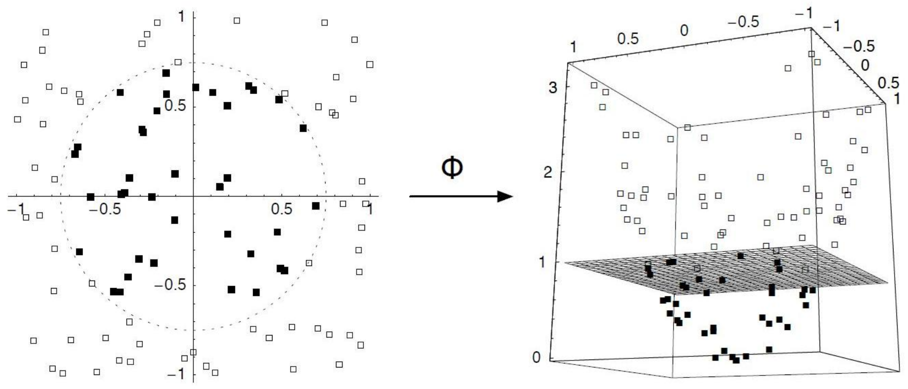

Figure 1: The transformation function $\Phi : (x, y) \to (x, y, x^2 + y^2)$ maps the non-linearly separable 2D data on the left to the linearly separable 3D data on the right

Figure 3.7: A visual representation of projecting data to find a separating hyperplane, [3].

In addition to soft margin techniques, one can also use *kernel methods* to project data into higher dimensions, so that a linearly separating hyperplane an be found. Consider the figure shown above. Before projection, the data show two classes, one that is arranged in a circle around the origin, while the other class is the space outside of the circle. Any straight line in this two dimensional graph will not be a good classifier between the two classes. But the projection $\Phi : (\mathbf{x}, \mathbf{y}) \longrightarrow (\mathbf{x}, \mathbf{y}, \mathbf{x}^2 + \mathbf{y}^2)$ turns the two dimensional graph into a three dimensional plot, and as the picture shows, the flat hyperplane that goes through the origin is a linear separator for the two classes. Thus, linearly inseparable data can be projected into a higher space such that a linear classifier can be found.

However, projecting data into higher dimensions can often be computationally infeasible, and finding suitable projections is difficult. Note that the decision function for a SVM computes and uses only the dot product of the training data values, motivating the use of kernels. Kernel Methods will apply the idea of projecting data into higher dimensions without actually projecting the data; they simply use a *kernel function* to compute the dot product of two points in the higher dimensional projected space.

A natural question to ask then is which functions constitute valid kernels, or more generally, if given some function $k(\mathbf{x}, \mathbf{x}')$, can we show that $k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})\Phi(\mathbf{x}')$ for some higher dimensional projection Φ? Trivially, we can calculate and expand a given kernel function explicitly and use algebra and inspection to find the projection by hand. However, a simpler method to verify kernels is to use *Mercer's Theorem*, which states that any function with a positive semi-definite Gram matrix has a representation as the convergent sum of product functions. Simply put, Mercer's Theorem allows us to classify all positive semi-definite Gram matrices as inner products on some high dimensional Hilbert space, and consequently, the associated functions can be used as kernels. Here, we provide a proof of a specific case of Mercer's Theorem, which is sufficient for use in SVMs. (For clarity, we denote the dot product between two vectors as $< \mathbf{a}, \mathbf{b} >$ rather than $\mathbf{a}^T\mathbf{b}$.)

Definition 3.4.1. *Given a symmetric function $H(\boldsymbol{x}, \boldsymbol{x}')$ and a set of vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, the* **Gram Matrix** *is defined as*

$$\begin{bmatrix} H(\boldsymbol{x}_1, \boldsymbol{x}_1) & \ldots & H(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ H(\boldsymbol{x}_2, \boldsymbol{x}_1) & \ldots & H(\boldsymbol{x}_2, \boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ H(\boldsymbol{x}_m, \boldsymbol{x}_1) & \ldots & H(\boldsymbol{x}_m, \boldsymbol{x}_m) \end{bmatrix} = H_N$$

Definition 3.4.2. *A $N \times N$ matrix H is called* **positive semi-definite** *if*

$$\boldsymbol{w}^T H \boldsymbol{w} \geq 0$$

for any column vector $\boldsymbol{w} \in \mathbb{R}^N$. Similarly, a function $H(\boldsymbol{x}, \boldsymbol{x}')$ is called positive semi-definite *if the Gram matrix it induces is positive semi-definite.*

Theorem 3.4.3. *(Abe, [1]) Let X be the input space and $H(\boldsymbol{x}, \boldsymbol{x}')$ be a positive semi-definite function for $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \in X$. Define H_0 as the linear space spanned by the functions $\{H_{\boldsymbol{x}_i} | \boldsymbol{x}_i \in S\}$ where*

$$H_{\boldsymbol{x}_i}(\boldsymbol{x}) = H(\boldsymbol{x}_i, \boldsymbol{x}).$$

Then there exists a Hilbert space $\mathcal{H}$, which is the completion of H_0, and a mapping from X to $\mathcal{H}$ such that

$$H(\boldsymbol{x}, \boldsymbol{x}') =< H_{\boldsymbol{x}}, H'_{\boldsymbol{x}} > .$$

Theorem (3.4.3) is claiming that with any function that produces a positive semi definite Gram matrix, a natural mapping ϕ to a Hilbert space $\mathcal{H}$ (that consists of functions) given by

$$\phi : X \longrightarrow \mathcal{H}, \tag{3.51}$$
$$\phi : x \longmapsto H_x(x), \tag{3.52}$$

equates the functions

$$H(x,x') : X \times X \longrightarrow \mathbb{R}, \tag{3.53}$$
$$< H_x, H_{x'} > : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{R}, \tag{3.54}$$

meaning that the functions are essentially the same, as long as the matching inputs from the proper spaces are used.

Proof. Note that H_0 is a linear space, meaning that any member of H_0 can be written as a linear combination of various $H_{\mathbf{x}_i}$. As such, define $f, g, h \in H_0$ as

$$f = \sum_{\mathbf{x}_i \in X} c_i H_{\mathbf{x}_i} \tag{3.55}$$
$$g = \sum_{\mathbf{x}'_j \in X} d_j H_{\mathbf{x}'_j} \tag{3.56}$$
$$h = \sum_{\mathbf{x}''_k \in X} e_k H_{\mathbf{x}''_k}. \tag{3.57}$$

We define the dot product between two members of H_0 as follows:

$$< f, g > = \sum_{\mathbf{x}'_j \in X} d_j f(\mathbf{x}'_j) \tag{3.58}$$
$$= \sum_{\mathbf{x}_i \in X} \sum_{\mathbf{x}'_j \in X} c_i d_j H(\mathbf{x}, \mathbf{x}') \tag{3.59}$$
$$= \sum_{\mathbf{x}_i \in X} c_i g(\mathbf{x}_i) \tag{3.60}$$

To show that this is a dot product, we need to show symmetry, linearity, and that $< f, f > \geq 0$ with equality if $f = 0$ (positive-definiteness).

- Symmetry follows naturally from the definition.
- Linearity: We want to show that $< f, (g+h) > = < f, g > + < f, h >$.

$$\begin{aligned} < f, (g+h) > &= \sum_{\mathbf{x}_i \in X} c_i (g+h)(\mathbf{x}_i) \\ &= \sum_{\mathbf{x}_i \in X} c_i g(\mathbf{x}_i) + \sum_{\mathbf{x}_i \in X} c_i h(\mathbf{x}_i) \qquad \text{(since } g, h \text{ are linear)} \\ &= < f, g > + < f, h > \end{aligned}$$

- Positive-Definiteness: We want to show that $< f, f > \geq 0$.
  Since we know that $H(\mathbf{x}, \mathbf{x}')$ is positive semi-definite, we know that

$$\mathbf{w}^T H_N \mathbf{w} \geq 0 \tag{3.61}$$

  where H_N is the Gram matrix for $H(\mathbf{x}, \mathbf{x}')$ and for any $\mathbf{w}$. If we let $\mathbf{w} = [c_1 \dots c_N]$, then (3.61) implies that

$$\sum_{\mathbf{x}_i, \mathbf{x}_j} c_i c_j H(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

  Finally, since f is a linear combination of $H_{\mathbf{x}_i}$, we can conclude $< f, f > \geq 0$.

Thus, (3.60) is the dot product, and we can use it to complete H_0 to a Hilbert space $\mathcal{H}$.
Lastly, (3.60) shows us that

$$< f, H_{\mathbf{x}} >= f(\mathbf{x}),$$

meaning that in particular

$$< H_{\mathbf{x}}, H'_{\mathbf{x}} >= H(\mathbf{x}, \mathbf{x'}).$$

$\square$

The above proof is very technical and requires some familiarity with functional spaces, but the results are indeed powerful. Now we can easily check to see if a function is a kernel, namely by checking to see that the function's Gram matrix has non-negative eigenvalues. (Recall that any arbitrary vector can be written as the linear combination of eigenvectors.) Moreover, as examples of the possible kernels one can construct, the follow list shows some of the ways to combine kernels to create new ones.

Theorem 3.4.4. *(Bishop, [2]) Given two valid kernels $k_1(\boldsymbol{x}, \boldsymbol{x}')$ and $k_2(\boldsymbol{x}, \boldsymbol{x}')$, the following are also valid kernels:*

$$k(\boldsymbol{x}, \boldsymbol{x}') = c k_1(\boldsymbol{x}, \boldsymbol{x}'), \tag{3.62}$$
$$k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}'), \tag{3.63}$$
$$k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') k_2(\boldsymbol{x}, \boldsymbol{x}'), \tag{3.64}$$
$$k(\boldsymbol{x}, \boldsymbol{x}') = f(\boldsymbol{x}) k_1(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}'), \tag{3.65}$$
$$k(\boldsymbol{x}, \boldsymbol{x}') = p(k_1(\boldsymbol{x}, \boldsymbol{x}')), \tag{3.66}$$
$$k(\boldsymbol{x}, \boldsymbol{x}') = exp(k_1(\boldsymbol{x}, \boldsymbol{x}')), \tag{3.67}$$
$$k(\boldsymbol{x}, \boldsymbol{x}') = k_3(\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}')), \tag{3.68}$$
$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T A \boldsymbol{x}', \tag{3.69}$$

where $c \in \mathbb{R}^+$, $f(\boldsymbol{x})$ is any function, $p(x)$ is a polynomial with non-negative coefficients, $\Phi(\boldsymbol{x})$ is a projection in a space where $k_3(\boldsymbol{x}, \boldsymbol{x}')$ is a valid kernel, and A is a symmetric positive semi-definite matrix.

Theorem (3.4.4) lets us immediately conclude that polynomials of the form $(\mathbf{x}^T\mathbf{x}')^k$ are kernels by line (3.64), since we already know that the normal dot product is a kernel. (Note that $(\mathbf{x}^T\mathbf{x}')^k$ is the homogeneous polynomial containing all terms consisting of $\mathbf{x}$ and $\mathbf{x}'$ of degree k.) More generally, also using line (3.63) shows us that polynomials of the form $(\mathbf{x}^T\mathbf{x}' + c)^k$, which contain all terms consisting of $\mathbf{x}$ and $\mathbf{x}'$ up to degree k, are also kernels, provided that $c > 0$.

Another commonly used kernel is of the form

$$k(\mathbf{x}, \mathbf{x}') = exp\Bigg(-\gamma||\mathbf{x} - \mathbf{x}'||^2 \Bigg) \tag{3.70}$$

and called the *Gaussian Kernel* or the *Radial Basis Function Kernel.* To prove that the Gaussian kernel is valid, we can expand the square,

$$||\mathbf{x} - \mathbf{x}'||^2 = \mathbf{x}^T\mathbf{x} + (\mathbf{x}')^T\mathbf{x}' - 2\mathbf{x}^T\mathbf{x}',$$

resulting in

$$k(\mathbf{x}, \mathbf{x}') = exp(-\gamma\mathbf{x}^T\mathbf{x})exp(-\gamma(\mathbf{x}')^T\mathbf{x}')exp(\gamma\mathbf{x}^T\mathbf{x}'),$$

which is a valid kernel by (3.65) and (3.66). Note that the projection space that corresponds to the Gaussian kernel has infinite dimensionality.

After having extensively covered the construction and derivation of kernels, how does one use them in a SVM? Fortunately, they easily fit into the optimization step of a SVM formulation. We rewrite the dual problem of a soft margin SVM, equation (3.33), since most practical applications will use both:

$$\max Q(\boldsymbol{\alpha}) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2}\sum_{i,j}^{M} \alpha_i\alpha_j y_i y_j H(\mathbf{x}_i, \mathbf{x}_j) \tag{3.71}$$

subject to the constraints

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \alpha_i \geq 0 \qquad C \geq \alpha_i \geq 0. \tag{3.72}$$

Then optimizing with respect to the KKT conditions yields the decision function

$$D(\mathbf{x}) = \sum_{i\in S} \alpha_i y_i H(\mathbf{x}_i, \mathbf{x}) + b, \tag{3.73}$$

where b is given by

$$b = \frac{1}{n(U)} \sum_{j\in U} \Bigg(y_j - \sum_{i=1}^{N} \alpha_i y_i H(\mathbf{x}_i, \mathbf{x}_j) \Bigg), \tag{3.74}$$

U being the set of unbounded support vectors. (This is slightly different from the original soft margin b.) As before,

$$\text{Classify } \mathbf{x} \begin{cases} \text{Population1} & \text{if} \quad D(\mathbf{x}) > 0, \\ \text{Population2} & \text{if} \quad D(\mathbf{x}) < 0. \end{cases} \tag{3.75}$$

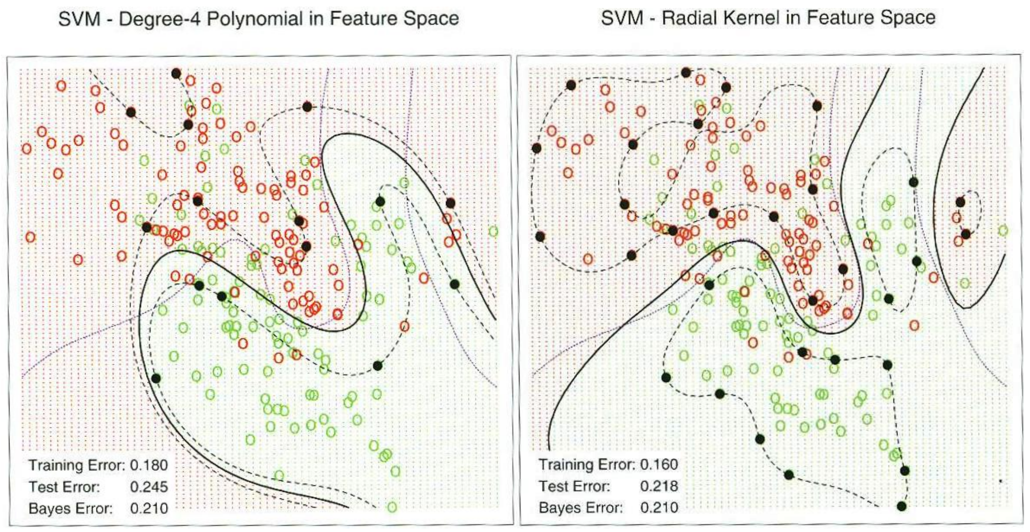

Figure 3.8: Use of a kernel can significantly improve classification for a SVM. Here, we see two examples of kernels: the picture on the left was generated with a degree four polynomial of the form $(\mathbf{x}^T\mathbf{x} + c)^4$, while the picture on the right uses the Gaussian kernel. The decision hyperplane is the solid black line, and the margins for the two classes are represented by dashed black lines. (The margin is denoted by *unbounded* support vectors, the solid black points.) Note that the decision hyperplane in the original feature space is no longer a straight line and instead curves and adapts to the data.

3.5 Further Concerns

Kernel selection is an important aspect of training an accurate SVM. Many times it is not necessary to even use a kernel; in text classification applications, the features extracted are words, and since dictionaries contain tens of thousands of words, dimensionality is already high enough to usually ensure accurate classification with just a linear soft margin SVM. However, other data sets clearly benefit from the use of kernels, and the most commonly used kernel is the Gaussian kernel.

Consider the Gaussian kernel:

$$k(\mathbf{x}, \mathbf{x}') = exp\Bigg(- \gamma ||\mathbf{x} - \mathbf{x}'||^2 \Bigg). \tag{3.76}$$

In short, it measures similarity by the "distance" between two points, usually with the Euclidean metric although any valid norm would work. Notice that given a certain point $\widehat{\mathbf{x}}$, an infinite amount of points give the same value of $k(\widehat{\mathbf{x}}, \mathbf{x})$, namely those that are equidistance to the point $\widehat{\mathbf{x}}$. This is the reason that this kernel is sometimes called a *radial basis function*. Lastly, notice that as a point gets further and further away from $\widehat{\mathbf{x}}$, the value of $k(\widehat{\mathbf{x}}, \mathbf{x})$ decreases exponentially, which is why it is also called the *Gaussian* kernel.

Even though the class of kernels to be used is known, the value of σ and the cost parameter C must still be determined or estimated. Many algorithms to find the values of these parameters exist, and many of them simply use a hill-climbing approach to find optimal combinations of σ and

C that maximize training data accuracy. However, this returns us to the question of *over-fitting*: with large values of both σ and C, one can ensure high training accuracy, but almost assuredly poor generalization ability and classification. Striking a balance between training and test accuracy is difficult, requires careful consideration of the data and classification model, and will differ from case to case.

Chapter 4

Applications in R

In this chapter, we'll use R to perform classification with Linear Discriminant Analysis and a Support Vector Machine on a data set. The data set we'll use is called "synth.te" ([5]) and is included in the latest release of R. It is a two dimensional data set with 1000 observations, half from one class 1 and the other half from class 2. The data is oriented around four central points, with the lower two groups coming from class 1 and the upper groups coming from class 2. Note that "synth.te" was originally meant to be paired with "synth.tr" with .tr being the training set and .te being the test set, but we will randomly select a training set from synth.te, testing our classifier on the points that remain. Lastly, there is some overlap in between the classes, meaning that we have a linearly inseparable data set.

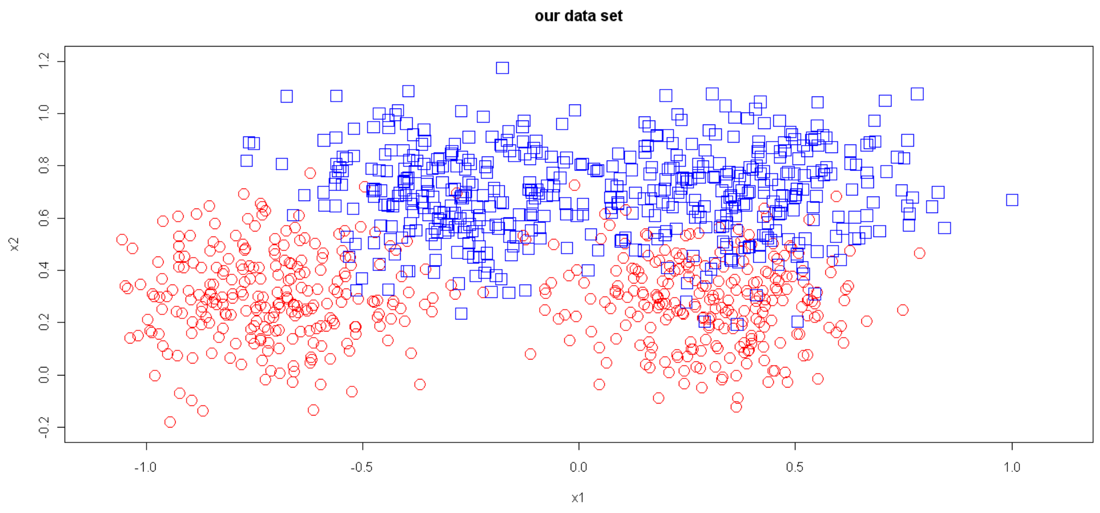

Figure 4.1: Our data set. Red circles indicate observations from class 1, and blue squares are observations from class 2.

4.1 LDA in R

We begin with Linear Discriminant Analysis, which can be found in the MASS library. To start, we select a random sample of 500 from our data to be our training set. Note that the lda function in MASS *will* assume a common covariance matrix of the two training populations, which is in our case,

$$\Sigma = \begin{vmatrix} 0.21714987 & 0.01868103 \\ 0.01868103 & 0.06993958 \end{vmatrix} \tag{4.1}$$

Moreover, we assume that the prior probabilities and costs of misclassification are equal for both classes.

	Class 1	**Class 2**	**Overall**
Training Points	242	258	500
Training $\bar{x}_i$	(-0.17517086,0.2874690)	(0.05162236,0.6924096)	NA
Correctly Classified Test Data	233	210	443
Incorrectly Classified Test Data	25	32	57
Classification Accuracy	90.31%	86.78%	88.6%

LDA classification yields good results, but as the training data looks bi-modal, the assumption of multivariate normality reduces test accuracy. One could run two different LDA classifiers on the two halfs of the data to correct this, or perhaps design an LDA classifier that looks at the ratio of bi-modal distributions.

Recall that given equal costs of misclassification and prior probabilities for the two classes, we can interpret LDA much like FDA, in that there is some other vector $\hat{\mathbf{l}}$ that is being projected onto. This vector for our case is

$$\hat{\mathbf{l}} = (0.5712041, 5.7619468) \tag{4.2}$$

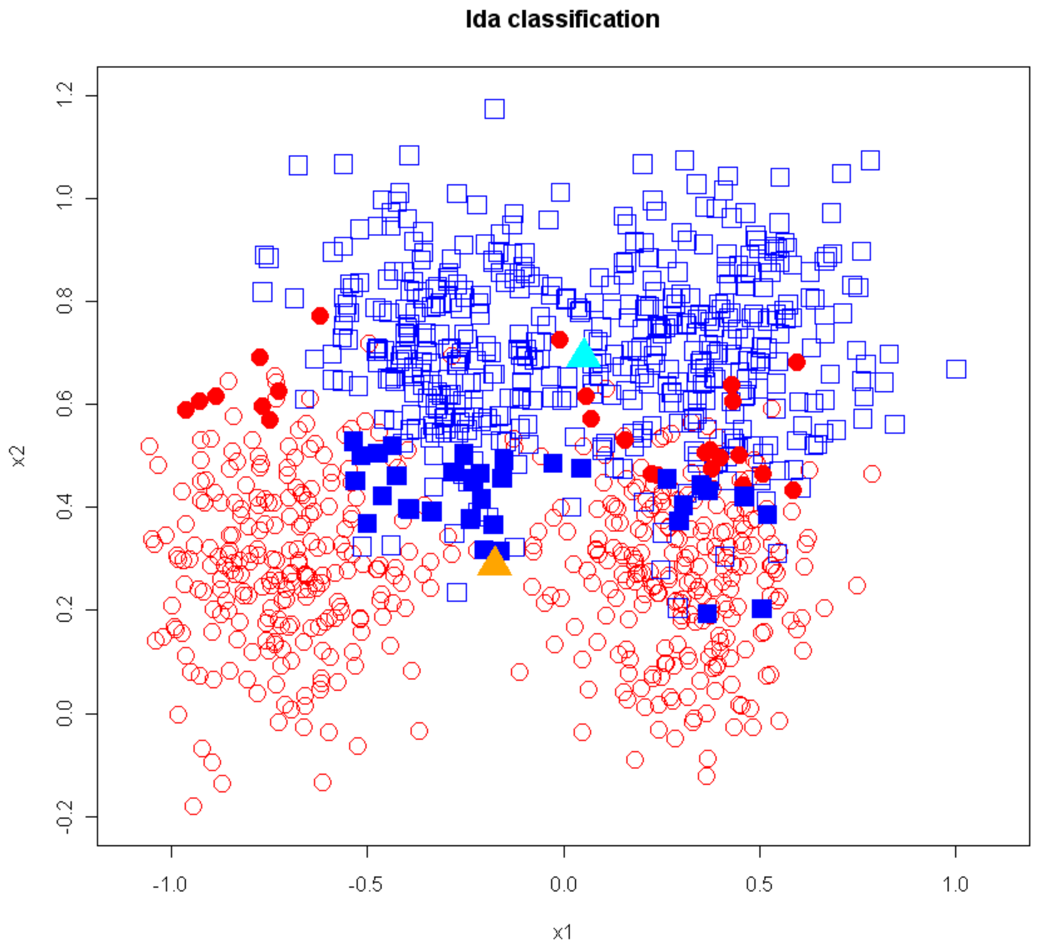

Figure 4.2: LDA classification on our data. Class 1's sample mean is given by the orange triangle and Class 2's sample mean is given by the cyan triangle. The filled in red data points are points from class 1 that were misclassified as class 2, and similarly, the filled in blue points are datum from class 2 that were misclassified as class 1.

4.2 Support Vector Machines in R

Next, the package “e1071” in the CRAN database contains a formulation of libsvm, a Support Vector Machine algorithm, for use in R. We will use the same test and training data from the LDA example, and classify using a linear kernel, and a cost coefficient of $C = 10$.

	Class 1	**Class 2**	**Overall**
Number of Support Vectors	70	70	140
Correctly Classified Test Data	235	210	445
Incorrectly Classified Test Data	23	32	55
Classification Accuracy	91.09%	86.78%	89%

As you can see, the linear SVM does little better than a straight LDA analysis.

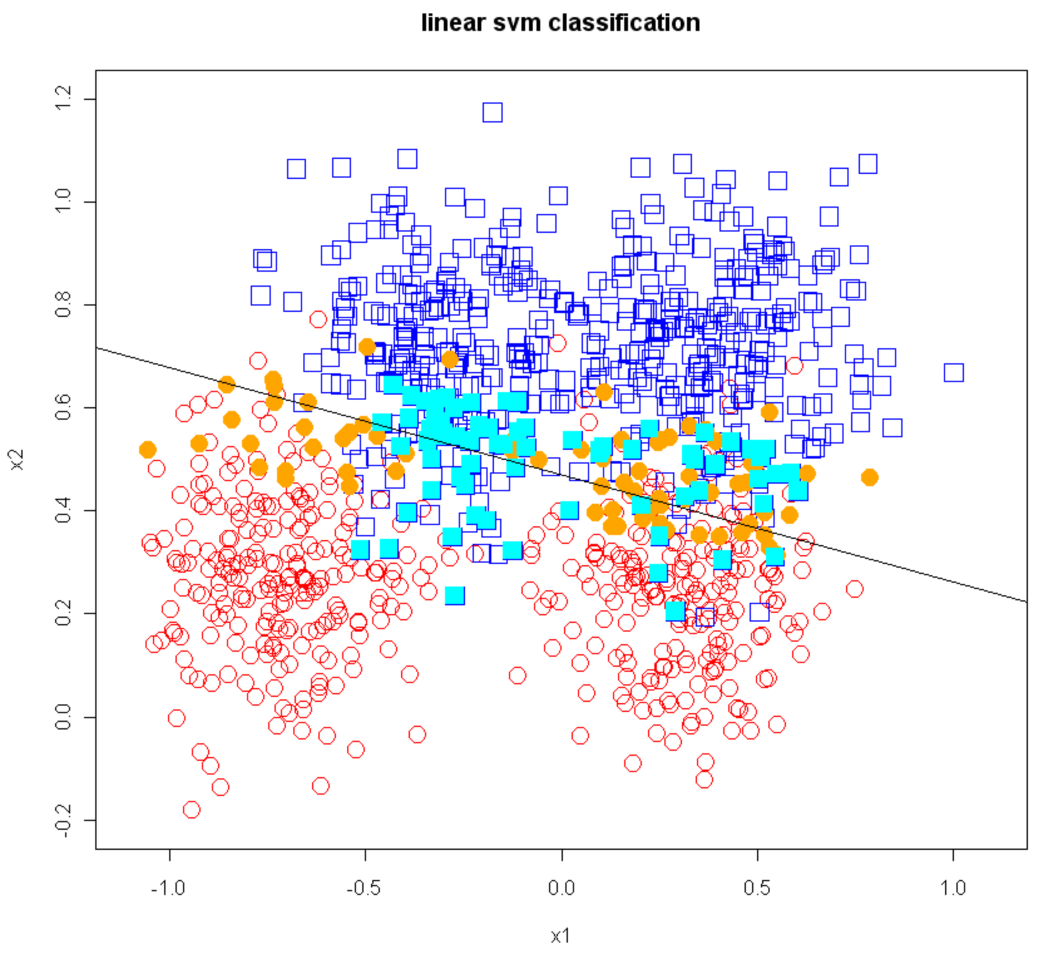

Figure 4.3: A linear SVM classifier. The orange points represent support vectors from class 1, and similarly the cyan points are support vectors for class two. The line splitting the data is a hyperplane representing the decision function, namely $D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, where $\mathbf{w} = (-0.5030882, -2.4113652)$. Consequently, if a new unclassified point falls below the hyperplane, we can associate it to class 1, and if it falls above the hyperplane, the datum can be classified as class 2.

R also lets us implement different kernels in our SVM.

	Polynomial - Cubic $(((\gamma\mathbf{x}^T\mathbf{x}')^3)$	**Gaussian** $(exp(-\gamma\|\|\mathbf{x}-\mathbf{x}'\|\|^2))$
Gamma	.5	.5
Number **of Support Vectors**	Class 1: 79 Class 2: 78 Overall:157	Class 1: 56 Class 2: 58 Overall: 114
Correctly **Classified Test Data**	Class 1: 220 Class 2: 228 Overall: 448	Class 1: 242 Class 2: 216 Overall: 458
Incorrectly **Classified Test Data**	Class 1: 38 Class 2: 14 Overall: 52	Class 1: 16 Class 2: 26 Overall: 42
Classification **Accuracy**	Class 1: 85.27% Class 2: 94.21% Overall: 89.6%	Class 1: 93.8% Class 2: 89.26% Overall: 91.6%
w	(136.9053, -173,2019)	(-1.022526, -13.773888)

Using a kernel doesn't vastly improve accuracy in either case; this could be due to the fact that the data is split almost linearly, meaning that using a polynomial can only help marginally, or hurt as was the case here. But notice that the radial kernel will return the highest accuracy rates.

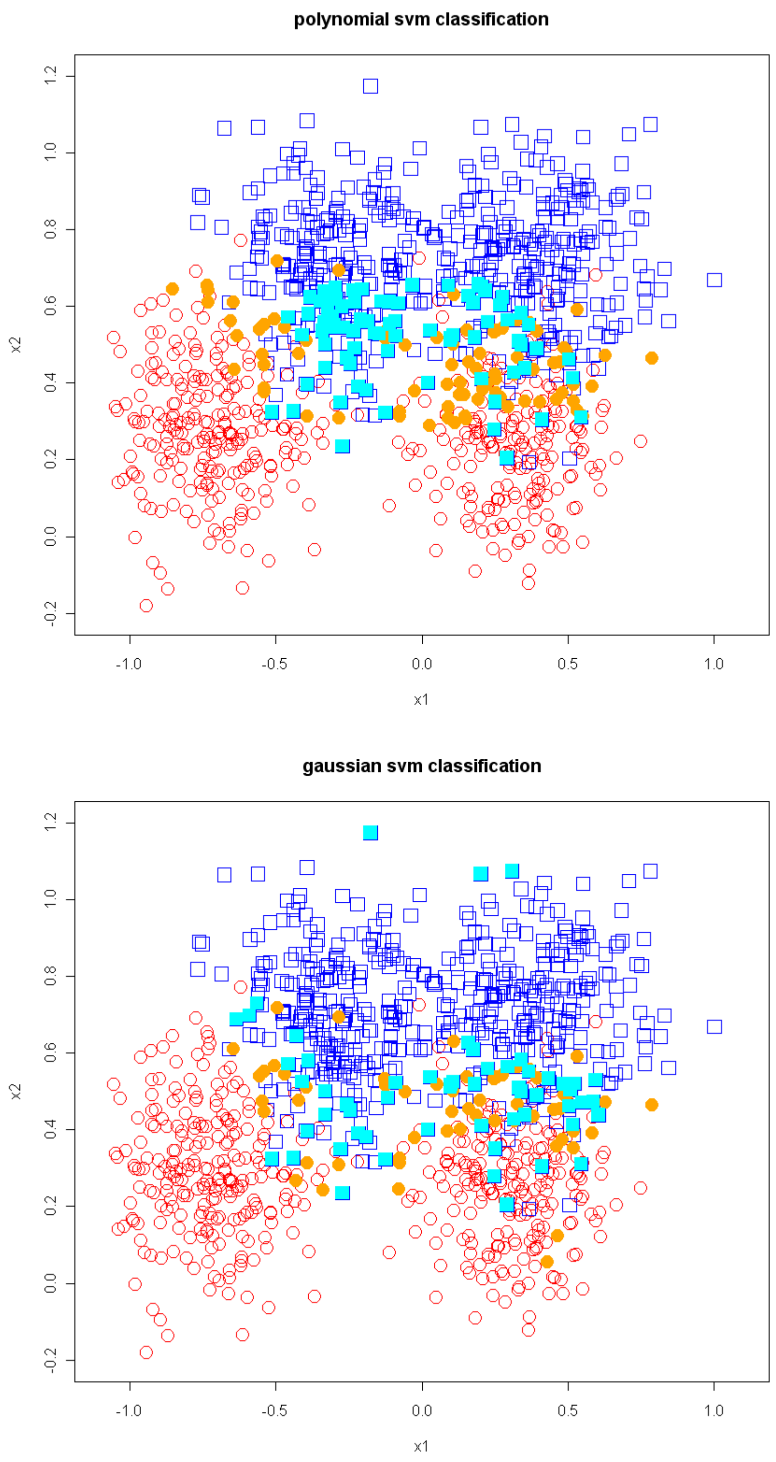

Figure 4.4: A comparison between a cubic kernel and a Gaussian one. Note that the pattern of support vectors differs between the two, and each differs from the linear kernel. Also, the decision hyperplane has been omitted from the pictures, since it would no longer be a straight line in the original feature space.

Lastly, we will consider the question of how more or less training data effect classification accuracy. The tables below show average accuracies for the four types of classifier shown before over 500 random samplings of training sets of size n:

	LDA	**Linear SVM**	**Polynomial SVM**	**Radial SVM**
$n = 975$	88.62%	88.72%	89.33%	92.1%
$n = 950$	88.05%	88.11%	88.65%	91.82%
$n = 900$	88.37%	88.45%	89.4%	91.84%
$n = 750$	88.48%	88.62%	89.08%	91.84%
$n = 600$	88.63%	88.78%	89.11%	91.78%
$n = 500$	88.62%	88.79%	88.91%	91.67%
$n = 250$	88.76%	88.83%	88.33%	91.32%
$n = 100$	88.82%	88.52%	87.23%	90.51%
$n = 50$	88.48%	87.84%	85.53%	89.35%
$n = 25$	87.79%	86.54%	83.17%	86.76%

Unsurprisingly, classification accuracy is decreased with decreased amounts of training data; with less points to train a model on, it makes sense that a model's predicative power lowers. More surprising is the variability between accuracy rates for linear models versus kernel models. Both the LDA and Linear SVM have less than 2.5% change between its highest and lowest accuracy rates, but the Polynomial and Kernel SVMs have around 6% change between its highest and lowest rates, dipping below the accuracy of both linear models. This may be a result of the complexity of the model versus the simplicity of the data.

It is also worth nothing the nature of the different trends of the different classifiers. Discounting the case where $n = 975$, every classifier reaches its highest accuracy rate when n is closer to 500 except for the radial classifier, whose accuracy seems to be an nearly monotonic function of the size of the training set.

Lastly, as a caveat to this simulation, remember that the size of the training set is inversely proportional to the size of the test set. More simulations on distinct test and training sets would lead to more definitive answers.

Chapter 5

Conclusion

To say that there remain many other issues when classifying data would be an understatement.

In this thesis, we have covered three types of classification techniques, one using probability ratios, another using dimensionality reduction, and finally one using the maximum margin. However, numerous other classification techniques exist, such as decision tree models or neural networks. Many classification tasks will warrant their own custom classifier, mixing and matching aspects of many classifiers, to best adapt to their specific challenges. That said, many people do consider Support Vector Machines a benchmark to evaluate new classifiers. Given their relatively simplicity, their ability to use kernels to model many different types of data, and most importantly, their sparse learning technique using support vectors which speeds up the computation time of new classifications, it's not hard to see why.

However, a drawback to SVMs is that their construction only assumes two groups, so extending this model to classify many different groups often decreases its effectiveness. Two ways to extend classification to multiple groups are commonly used. One of them is called the "one against all" formulation, meaning that you pick a group, and pool every data point that isn't in that group into a separate group and try to classify. If there is only one positive classification, then we can classify that point, but if more than one or no positive results occur, the point is unclassifiable. The other is called "pairwise" formulation, meaning that you attempt to classify a point using all possible pairs of groups and if there is a group that the point is classified into the most often, then that group is the final classification. Both of these methods can create large regions that are unclassifiable. Many current research questions involve trying to find optimal classifications for these "unclassifiable" regions.

Lastly, even though this thesis has covered much of the mathematics behind classification, it is still very much an art because classification seeks to name the unknown. Our best hope then is to use the tools at our disposal as carefully as we can. But even when selecting which model to follow or the values of certain parameters, many trade-offs exist between competing goals. Experience is essential to obtain good classification, and I hope this thesis has encouraged more exploration of the subject.

I would like to thank my advisor, Jo Hardin, for introducing me to classification and working with me throughout the year. Thesis would have been much more difficult without her motivation and guidance. Additionally, I would like to thank the math department for the last four years of morning classes and late night problem sets. Finally, as Kathy will probably be the only one to get this far anyways, thank you Kathy for running the department and keeping everyone sane in Millikan.

Bibliography

[1] Shigeo Abe. *Support Vector Machines for Pattern Classification.* Springer Verlag, London, England, 2005.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer Science+Business Media, LLC, 2006.

[3] Sean Luke Keith M. Sullivan. Evolving kernels for support vector machine classification. *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1702–1707, 1990.

[4] Dean W. Wichern Richard A. Johnson. *Applied Multivariate Statistical Analysis.* Prentice-Hall, Inc., 3rd edition, 1992.

[5] B.D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[6] Jerome Friedman Trevor Hastie, Robert Tibshirana. *The Elements of Statistical Learning: Data Mining, inference, and Prediction.* Springer Verlag, 2001.