Linear Classification

10/36-702

Han Liu and Larry Wasserman

In these notes we discuss parametric classification, in particular, linear classification, from several different points of view. We begin with a review of classification.

1 Review of Classification

The problem of predicting a discrete random variable Y from another random variable X is called *classification*, also sometimes called *discrimination*, *pattern classification* or *pattern recognition*. We observe iid data $(X_1, Y_1), \ldots, (X_n, Y_n) \sim P$ where $X_i \in \mathbb{R}^d$ and $Y_i \in \{0, 1, \ldots, K-1\}$. Often, the covariates X are also called *features*. The goal is to predict Y given a new X; here are some examples:

1. The Iris Flower study. The data are 50 samples from each of three species of Iris flowers, *Iris setosa*, *Iris virginica* and *Iris versicolor*; see Figure 1. The length and width of the sepal and petal are measured for each specimen, and the task is to predict the species of a new Iris flower based on these features.

2. The Coronary Risk-Factor Study (CORIS). The data consist of attributes of 462 males between the ages of 15 and 64 from three rural areas in South Africa. The outcome Y is the presence ($Y = 1$) or absence ($Y = 0$) of coronary heart disease and there are 9 covariates: systolic blood pressure, cumulative tobacco (kg), ldl (low density lipoprotein cholesterol), adiposity, famhist (family history of heart disease), typea (type-A behavior), obesity, alcohol (current alcohol consumption), and age. The goal is to predict Y from all these covariates.

3. Handwriting Digit Recognition. Here each Y is one of the ten digits from 0 to 9. There are 256 covariates $X_1, \ldots, X_{256}$ corresponding to the intensity values of the pixels in a 16×16 image; see Figure 2.

4. Political Blog Classification. A collection of 403 political blogs were collected during two months before the 2004 presidential election. The goal is to predict whether a blog is *liberal* ($Y = 0$) or *conservative* ($Y = 1$) given the content of the blog.

A *classification rule*, or *classifier*, is a function $h : \mathcal{X} \to \{0, \ldots, K-1\}$ where $\mathcal{X}$ is the domain of X. When we observe a new X, we predict Y to be $h(X)$. Intuitively, the classification rule h partitions the input space $\mathcal{X}$ into K disjoint *decision regions* whose boundaries are called *decision boundaries*. In these notes, we consider *linear classifiers* whose decision boundaries

Figure 1: Three different species of the Iris data. *Iris setosa* (Left), *Iris versicolor* (Middle), and *Iris virginica* (Right).

are linear functions of the covariate X. For $K = 2$, we have a *binary classification* problem. For $K > 2$, we have a *multiclass classification* problem. To simplify the discussion, we mainly discuss binary classification, and briefly explain how methods can extend to the multiclass case.

A binary classifier h is a function from $\mathcal{X}$ to $\{0, 1\}$. It is linear if there exists a function $H(x) = \beta_0 + \beta^T x$ such that $h(x) = I(H(x) > 0)$. $H(x)$ is also called a *linear discriminant function*. The decision boundary is therefore defined as the set $\{x \in \mathbb{R}^d \ : \ H(x) = 0\}$, which corresponds to a $(d-1)$-dimensional hyperplane within the d-dimensional input space $\mathcal{X}$.

The *classification risk*, or *error rate*, of h is defined as

$$R(h) = \mathbb{P}\big(Y \neq h(X)\big) \tag{1}$$

and the *empirical classification error* or *training error* is

$$\widehat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} I\big(h(X_i) \neq Y_i\big). \tag{2}$$

Here is some notation that we will use.

X	covariate (feature)
$\mathcal{X}$	domain of X, usually $\mathcal{X} \subset \mathbb{R}^d$
Y	response (pattern)
h	binary classifier, $h : \mathcal{X} \to \{0, 1\}$
H	linear discriminant function, $H(x) = \beta_0 + \beta^T x$ and $h(x) = I\big(H(x) > 0\big)$
m	regression function, $m(x) = \mathbb{E}(Y\|X = x) = \mathbb{P}(Y = 1\|X = x)$
P_X	marginal distribution of X
p_j	$p_j(x) = p(x\|Y = j)$, the conditional density[1] of X given that $Y = j$
π_1	$\pi_1 = \mathbb{P}(Y = 1)$
P	joint distribution of (X, Y)

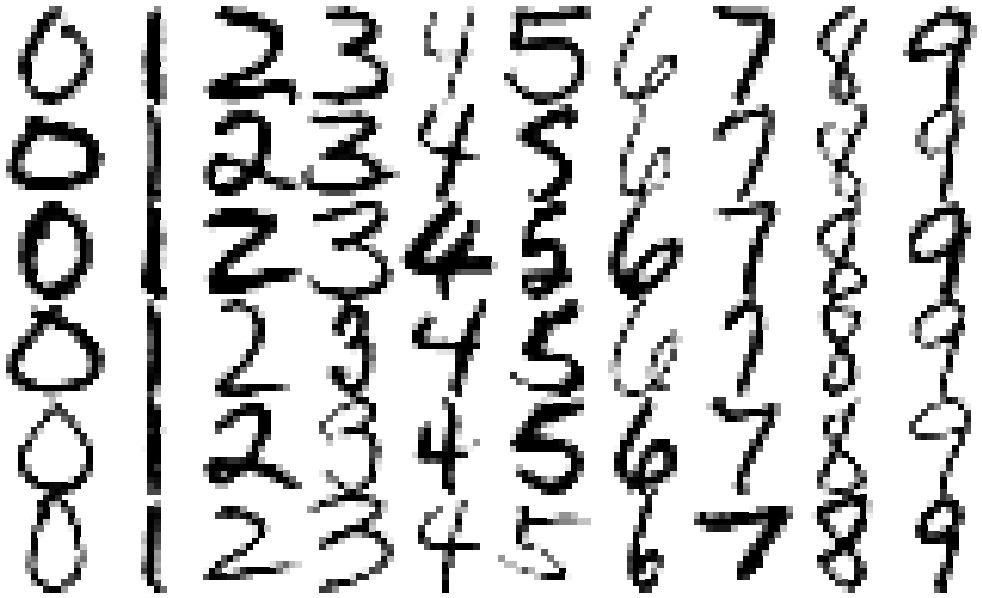


Figure 2: Examples from the zipcode data.

Now we review some key results.

Theorem 1 *The rule h that minimizes $R(h)$ is*

$$h^*(x) = \begin{cases} 1 & \text{if } m(x) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $m(x) = \mathbb{E}(Y|X = x) = \mathbb{P}(Y = 1|X = x)$ denotes the regression function.

The rule h^* is called the *Bayes rule*. The risk $R^* = R(h^*)$ of the Bayes rule is called the *Bayes risk*. The set $\{x \in \mathcal{X} : \ m(x) = 1/2\}$ is called the *Bayes decision boundary*.

Proof. We will show that $R(h) - R(h^*) \geq 0$. Note that

$$R(h) = \mathbb{P}\big(\{Y \neq h(X)\}\big) = \int \mathbb{P}\big(Y \neq h(X)|X = x\big) dP_X(x).$$

It suffices to show that

$$\mathbb{P}\big(Y \neq h(X)|X = x\big) - \mathbb{P}\big(Y \neq h^*(X)|X = x\big) \geq 0 \quad \text{for all } x \in \mathcal{X}. \tag{4}$$

Now,

$$\begin{aligned}
\mathbb{P}\big(Y \neq h(X)|X=x\big) &= 1 - \mathbb{P}\big(Y = h(X)|X=x\big) && (5)\\
&= 1 - \Big(\mathbb{P}\big(Y=1, h(X)=1|X=x\big) + \mathbb{P}\big(Y=0, h(X)=0|X=x\big)\Big) && (6)\\
&= 1 - \Big(h(x)\mathbb{P}(Y=1|X=x) + \big(1-h(x)\big)\mathbb{P}(Y=0|X=x)\Big) && (7)\\
&= 1 - \Big(h(x)m(x) + \big(1-h(x)\big)\big(1-m(x)\big)\Big). && (8)
\end{aligned}$$

Hence,

$$\begin{aligned}
&\mathbb{P}\big(Y \neq h(X)|X=x\big) - \mathbb{P}\big(Y \neq h^*(X)|X=x\big)\\
&= \Big(h^*(x)m(x) + \big(1-h^*(x)\big)\big(1-m(x)\big)\Big) - \Big(h(x)m(x) + \big(1-h(x)\big)\big(1-m(x)\big)\Big)\\
&= \big(2m(x)-1\big)\big(h^*(x)-h(x)\big) = 2\left(m(x) - \frac{1}{2}\right)\big(h^*(x)-h(x)\big). && (9)
\end{aligned}$$

When $m(x) \geq 1/2$ and $h^*(x) = 1$, (9) is non-negative. When $m(x) < 1/2$ and $h^*(x) = 0$, (9) is again non-negative. This proves (4). □

We can rewrite h^* in a different way. From Bayes' theorem

$$\begin{aligned}
m(x) &= \mathbb{P}(Y=1|X=x) = \frac{p(x|Y=1)\mathbb{P}(Y=1)}{p(x|Y=1)\mathbb{P}(Y=1) + p(x|Y=0)\mathbb{P}(Y=0)}\\
&= \frac{\pi_1 p_1(x)}{\pi_1 p_1(x) + (1-\pi_1)p_0(x)}. && (10)
\end{aligned}$$

where $\pi_1 = \mathbb{P}(Y=1)$. From the above equality, we have that

$$m(x) > \frac{1}{2} \quad \text{is equivalent to} \quad \frac{p_1(x)}{p_0(x)} > \frac{1-\pi_1}{\pi_1}. \tag{11}$$

Thus the Bayes rule can be rewritten as

$$h^*(x) = \begin{cases} 1 & \text{if } \frac{p_1(x)}{p_0(x)} > \frac{1-\pi_1}{\pi_1} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

If $\mathcal{H}$ is a set of classifiers then the classifier $h_o \in \mathcal{H}$ that minimizes $R(h)$ is the *oracle classifier*. Formally,

$$R(h_o) = \inf_{h \in \mathcal{H}} R(h)$$

and $R_o = R(h_o)$ is called the *oracle risk* of $\mathcal{H}$. In general, if h is any classifier and R^* is the Bayes risk then,

$$R(h) - R^* = \underbrace{R(h) - R(h_o)}_{\text{distance from oracle}} + \underbrace{R(h_o) - R^*}_{\text{distance of oracle from Bayes error}}. \tag{13}$$

The first term is analogous to the variance, and the second is analogous to the squared bias in linear regression.

For a binary classifier problem, given a covariate X we only need to predict its class label $Y = 0$ or $Y = 1$. This is in contrast to a regression problem where we need to predict a real-valued response $Y \in \mathbb{R}$. Intuitively, classification is a much easier task than regression. To rigorously formalize this, let $m^*(x) = \mathbb{E}(Y|X = x)$ be the true regression function and let $h^*(x)$ be the corresponding Bayes rule. Let $\widehat{m}(x)$ be an estimate of $m^*(x)$ and define the *plug-in classification rule*:

$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \widehat{m}(x) > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

We have the following theorem.

Theorem 2 *The risk of the plug-in classifier rule in* (14) *satisfies*

$$R(\widehat{h}) - R^* \leq 2\sqrt{\int (\widehat{m}(x) - m^*(x))^2 dP_X(x)}.$$

Proof. In the proof of Theorem 1 we showed that

$$\begin{aligned} \mathbb{P}\big(Y \neq \widehat{h}(X)|X = x\big) - \mathbb{P}\big(Y \neq h^*(X)|X = x\big) &= \big(2\widehat{m}(x) - 1\big)\big(h^*(x) - \widehat{h}(x)\big) \\ &= \big|2\widehat{m}(x) - 1\big| I\big(h^*(x) \neq \widehat{h}(x)\big) = 2\big|\widehat{m}(x) - 1/2\big| I\big(h^*(x) \neq \widehat{h}(x)\big). \end{aligned}$$

Now, when $h^*(x) \neq \widehat{h}(x)$, there are two possible cases: (i) $\widehat{h}(x) = 1$ and $h^*(x) = 0$; (ii) $\widehat{h}(x) = 0$ and $h^*(x) = 1$. In both cases, we have that $\big|\widehat{m}(x) - m^*(x)\big| \geq \big|\widehat{m}(x) - 1/2\big|$. Therefore,

$$\begin{aligned} \mathbb{P}\big(\widehat{h}(X) \neq Y\big) - \mathbb{P}\big(h^*(X) \neq Y\big) &= 2\int \big|\widehat{m}(x) - 1/2\big| I\big(h^*(x) \neq \widehat{h}(x)\big) dP_X(x) \\ &\leq 2\int \big|\widehat{m}(x) - m^*(x)\big| I\big(h^*(x) \neq \widehat{h}(x)\big) dP_X(x) \\ &\leq 2\int \big|\widehat{m}(x) - m^*(x)\big| dP_X(x) \qquad (15) \\ &\leq 2\sqrt{\int \big(\widehat{m}(x) - m^*(x)\big)^2 dP_X(x)}. \qquad (16) \end{aligned}$$

The last inequality follows from the fact that $\mathbb{E}|Z| \leq \sqrt{\mathbb{E}Z^2}$ for any Z. □

This theorem implies that if the regression estimate $\widehat{m}(x)$ is close to $m^*(x)$ then the plug-in classification risk will be close to the Bayes risk. The converse is *not* necessarily true. It is possible for $\widehat{m}$ to be far from $m^*(x)$ and still lead to a good classifier. As long as $\widehat{m}(x)$ and $m^*(x)$ are on the same side of $1/2$ they yield the same classifier.

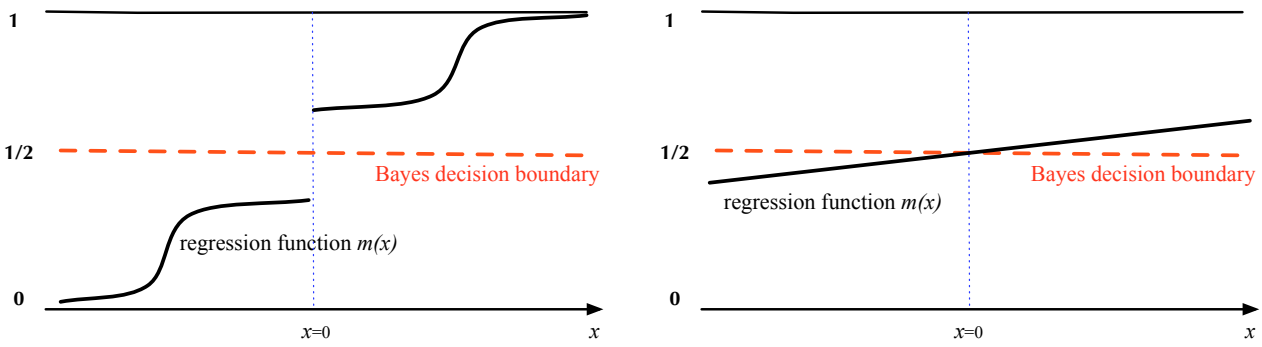


Figure 3: The Bayes rule is $h^*(x) = I(x > 0)$ in both plots, which show the regression function $m(x) = \mathbb{E}(Y|x)$ for two problems. The left plot shows an easy problem; there is little ambiguity around the decision boundary. The right plot shows a hard problem; it is hard to know from the data if you are to the left or right of the decision boundary.

Example 3 *Figure 3 shows two one-dimensional regression functions. In both cases, the Bayes rule is* $h^*(x) = I(x > 0)$ *and the decision boundary is* $\mathcal{D} = \{x = 0\}$. *The left plot illustrates an easy problem; there is little ambiguity around the decision boundary. Even a poor estimate of* $m(x)$ *will recover the correct decision boundary. The right plot illustrates a hard problem; it is hard to know from the data if you are to the left or right of the decision boundary.*

2 Empirical Risk Minimization

The conceptually simplest approach is empirical risk minimization (ERM) where we minimize the training error over all linear classifiers. Let $H_\beta(x) = \beta^T x$ (where $x(1) = 1$) and $h_\beta(x) = I(H_\beta(x) > 0)$. Thus we define $\widehat{\beta}$ to be the value of β that minimizes

$$\widehat{R}_n(\beta) = \frac{1}{n}\sum_{i=1}^{n} I(Y_i \neq h_\beta(X_i)).$$

The problem with this approach is that it is difficult to minimize $\widehat{R}_n(\beta)$. The theory for the ERM is straightforward. Recall that the set of linear classifiers has VC dimension $d + 1$. Therefore, using the VC theorem that we covered in 36-705, we have the following result. Let $\widehat{h}$ be the linear classifier that minimizes training error. Let h_* be the best linear classifier. Then

$$P(R(\widehat{h}) - R(h_*) > \epsilon) \leq 8n^{d+1}e^{-n\epsilon^2/128}.$$

The result can be improved if there are not too many data points near the decision boundary. We'll state a result due to Koltchinski and Panchenko (2002) that involves *the margin*. (See also Kakade, Sridharan and Tewari 2009). Let us take $Y_i \in \{-1, +1\}$ so we can write $h(x) = \text{sign}(\beta^T x)$. Suppose that $|X(j)| \leq A < \infty$ for each j. We also restrict ourselved to the set of linear classifiers $h(x) = \text{sign}(\beta^T x)$ with $|\beta(j)| \leq A$. Define the *margin-sensitive loss*

$$\phi_\gamma(u) = \begin{cases} 1 & \text{if } u \leq 0 \\ 1 - \frac{u}{\gamma} & \text{if } 0 < u \leq \gamma \\ 0 & \text{if } u > \gamma. \end{cases}$$

Then, for any such classifier h, with probability at least $1 - \delta$,

$$P(Y \neq h(X)) \leq \frac{1}{n}\sum_{i=1}^{n} \phi_\gamma(Y_i h(X_i)) + \frac{4A^{3/2}d}{\gamma n} + \left(\frac{8}{\gamma} + 1\right)\sqrt{\frac{\log(4/\delta)}{2n}}.$$

This means that, if there are few observations near the boundary, then, by taking γ large, we can make the loss small. However, the restriction to bounded covariates and bounded classifiers is non-trivial.

3 Gaussian Discriminant Analysis

Suppose that $p_0(x) = p(x|Y = 0)$ and $p_1(x) = p(x|Y = 1)$ are both multivariate Gaussians:

$$p_k(x) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)\right\}, \quad k = 0, 1.$$

where Σ_1 and Σ_2 are both $d \times d$ covariance matrices. Thus, $X|Y = 0 \sim N(\mu_0, \Sigma_0)$ and $X|Y = 1 \sim N(\mu_1, \Sigma_1)$.

Given a square matrix A, we define $|A|$ to be the determinant of A. For a binary classification problem with Gaussian distributions, we have the following theorem.

Theorem 4 *If $X|Y = 0 \sim N(\mu_0, \Sigma_0)$ and $X|Y = 1 \sim N(\mu_1, \Sigma_1)$, then the Bayes rule is*

$$h^*(x) = \begin{cases} 1 & \text{if } r_1^2 < r_0^2 + 2\log\left(\frac{\pi_1}{1-\pi_1}\right) + \log\left(\frac{|\Sigma_0|}{|\Sigma_1|}\right) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $r_i^2 = (x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i)$ for $i = 1, 2$ is the Mahalanobis distance.

Proof. By definition, the Bayes rule is $h^*(x) = I\big(\pi_1 p_1(x) > (1 - \pi_1)p_0(x)\big)$. Plug-in the specific forms of p_0 and p_1 and take the logarithms we get $h^*(x) = 1$ if and only if

$$\begin{aligned}(x - \mu_1)^T \Sigma_1^{-1}(x - \mu_1) - 2\log \pi_1 + \log\big(|\Sigma_1|\big) \\ < (x - \mu_0)^T \Sigma_0^{-1}(x - \mu_0) - 2\log(1 - \pi_1) + \log\big(|\Sigma_0|\big).\end{aligned} \tag{18}$$

The theorem immediately follows from some simple algebra. □

Let $\pi_0 = 1 - \pi_1$. An equivalent way of expressing the Bayes rule is

$$h^*(x) = \text{argmax}_{k \in \{0,1\}} \delta_k(x) \tag{19}$$

where

$$\delta_k(x) = -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) + \log \pi_k \tag{20}$$

is called the *Gaussian discriminant function*. The decision boundary of the above classifier can be characterized by the set $\{x \in \mathcal{X} : \delta_1(x) = \delta_0(x)\}$, which is quadratic so this procedure is called *quadratic discriminant analysis* (QDA).

In practice, we use sample quantities of $\pi_0, \pi_1, \mu_1, \mu_2, \Sigma_0, \Sigma_1$ in place of their population values, namely

$$\widehat{\pi}_0 = \frac{1}{n}\sum_{i=1}^{n}(1 - Y_i), \quad \widehat{\pi}_1 = \frac{1}{n}\sum_{i=1}^{n} Y_i, \tag{21}$$

$$\widehat{\mu}_0 = \frac{1}{n_0}\sum_{i:\ Y_i=0} X_i, \quad \widehat{\mu}_1 = \frac{1}{n_1}\sum_{i:\ Y_i=1} X_i, \tag{22}$$

$$\widehat{\Sigma}_0 = \frac{1}{n_0 - 1}\sum_{i:\ Y_i=0} (X_i - \widehat{\mu}_0)(X_i - \widehat{\mu}_0)^T, \tag{23}$$

$$\widehat{\Sigma}_1 = \frac{1}{n_1 - 1}\sum_{i:\ Y_i=1} (X_i - \widehat{\mu}_1)(X_i - \widehat{\mu}_1)^T, \tag{24}$$

where $n_0 = \sum_i (1 - Y_i)$ and $n_1 = \sum_i Y_i$. (Note: we could also estimate Σ_0 and Σ_1 using their maximum likelihood estimates, which replace $n_0 - 1$ and $n_1 - 1$ with n_0 and n_1.)

A simplification occurs if we assume that $\Sigma_0 = \Sigma_1 = \Sigma$. In this case, the Bayes rule is

$$h^*(x) = \text{argmax}_k \delta_k(x) \tag{25}$$

where now

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \log \pi_k. \tag{26}$$

Hence, the classifier is linear. The parameters are estimated as before, except that we use a pooled estimate of the Σ:

$$\widehat{\Sigma} = \frac{(n_0 - 1)\widehat{\Sigma}_0 + (n_1 - 1)\widehat{\Sigma}_1}{n_0 + n_1 - 2}. \tag{27}$$

The classification rule is

$$h^*(x) = \begin{cases} 1 & \text{if } \delta_1(x) > \delta_0(x) \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

The decision boundary $\{x \in \mathcal{X} : \ \delta_0(x) = \delta_1(x)\}$ is linear so this method is called *linear discrimination analysis* (LDA).

When the dimension d is large, fully specifying the QDA decision boundary requires $d + d(d-1)$ parameters, and fully specifying the LDA decision boundary requires $d + d(d-1)/2$ parameters. Such a large number of free parameters might induce a large variance. To further regularize the model, two popular methods are *diagonal quadratic discriminant analysis* (DQDA) and *diagonal linear discriminant analysis* (DLDA). The only difference between DQDA and DLDA with QDA and LDA is that after calculating $\widehat{\Sigma}_1$ and $\widehat{\Sigma}_0$ as in (24), we set all the off-diagonal elements to be zero. This is also called the independence rule.

We now generalize to the case where Y takes on more than two values. That is, $Y \in \{0, \ldots, K-1\}$ for $K > 2$. First, we characterize the Bayes classifier under this multiclass setting.

Theorem 5 *Let $R(h) = \mathbb{P}\left(h(X) \neq Y\right)$ be the classification error of a classification rule $h(x)$. The Bayes rule $h^*(X)$ minimizing $R(h)$ can be written as*

$$h^*(x) = \text{argmax}_k \mathbb{P}\left(Y = k | X = x\right) \tag{29}$$

Proof. We have

$$\begin{aligned} R(h) &= 1 - \mathbb{P}\left(h(X) = Y\right) && (30)\\ &= 1 - \sum_{k=0}^{K-1} \mathbb{P}\left(h(X) = k, Y = k\right) && (31)\\ &= 1 - \sum_{k=0}^{K-1} \mathbb{E}\Big[I\big(h(X) = k\big)\mathbb{P}\left(Y = k|X\right)\Big] && (32) \end{aligned}$$

It's clear that $h^*(X) = \text{argmax}_k \mathbb{P}\left(Y = k|X\right)$ achieves the minimized classification error $1 - \mathbb{E}\big[\max_k \mathbb{P}\left(Y = k|X\right)\big]$. □

Let $\pi_k = \mathbb{P}(Y = k)$. The next theorem extends QDA and LDA to the multiclass setting.

Theorem 6 *Suppose that $Y \in \{0, \ldots, K-1\}$ with $K \geq 2$. If $p_k(x) = p(x|Y = k)$ is Gaussian : $X|Y = k \sim N\big(\mu_k, \Sigma_k\big)$, the Bayes rule for the multiclass QDA can be written as*

$$h^*(x) = \text{argmax}_k \delta_k(x)$$

where

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^T\Sigma_k^{-1}(x - \mu_k) + \log \pi_k. \tag{33}$$

If all Gaussians have an equal variance Σ, then

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log \pi_k. \tag{34}$$

Let $n_k = \sum_i I(y_i = k)$ for $k = 0, \ldots, K-1$. The estimated sample quantities of π_k, μ_k, Σ_k, and Σ are:

$$\widehat{\pi}_k = \frac{1}{n}\sum_{i=1}^{n} I(y_i = k), \quad \widehat{\mu}_k = \frac{1}{n_k}\sum_{i:\ Y_i=k} X_i, \tag{35}$$

$$\widehat{\Sigma}_k = \frac{1}{n_k - 1}\sum_{i:\ Y_i=k} (X_i - \widehat{\mu}_k)(X_i - \widehat{\mu}_k)^T, \tag{36}$$

$$\widehat{\Sigma} = \frac{\sum_{k=0}^{K-1}(n_k - 1)\widehat{\Sigma}_k}{n - K}. \tag{37}$$

Example 7 *Let us return to the Iris data example. Recall that there are 150 observations made on three classes of the iris flower: Iris setosa, Iris versicolor, and Iris virginica. There are four features: sepal length, sepal width, petal length, and petal width. In Figure 4 we visualize the datasets. Within each class, we plot the densities for each feature. It's easy to see that the distributions of petal length and petal width are quite different across different classes, which suggests that they are very informative features.*

Figures 5 and 6 provide multiple figure arrays illustrating the classification of observations based on LDA and QDA for every combination of two features. The classification boundaries and error are obtained by simply restricting the data to these a given pair of features before fitting the model. We see that the decision boundaries for LDA are linear, while the decision boundaries for QDA are highly nonlinear. The training errors for LDA and QDA on this data are both 0.02*. From these figures, we see that it is very easy to discriminate the observations of class Iris setosa from those of the other two classes.*

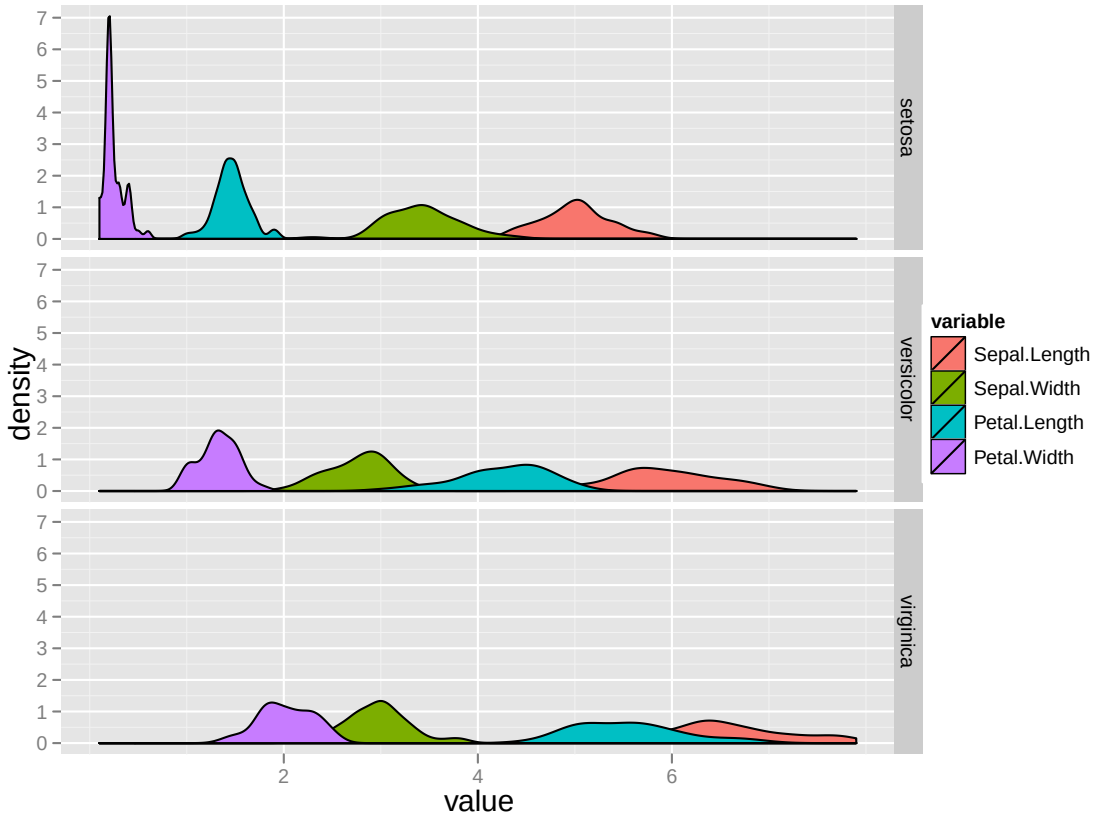


Figure 4: The Iris data: The estimated densities for different features are plotted within each class. It's easy to see that the distributions of petal length and petal width are quite different across different classes, which suggests that they are very informative features.

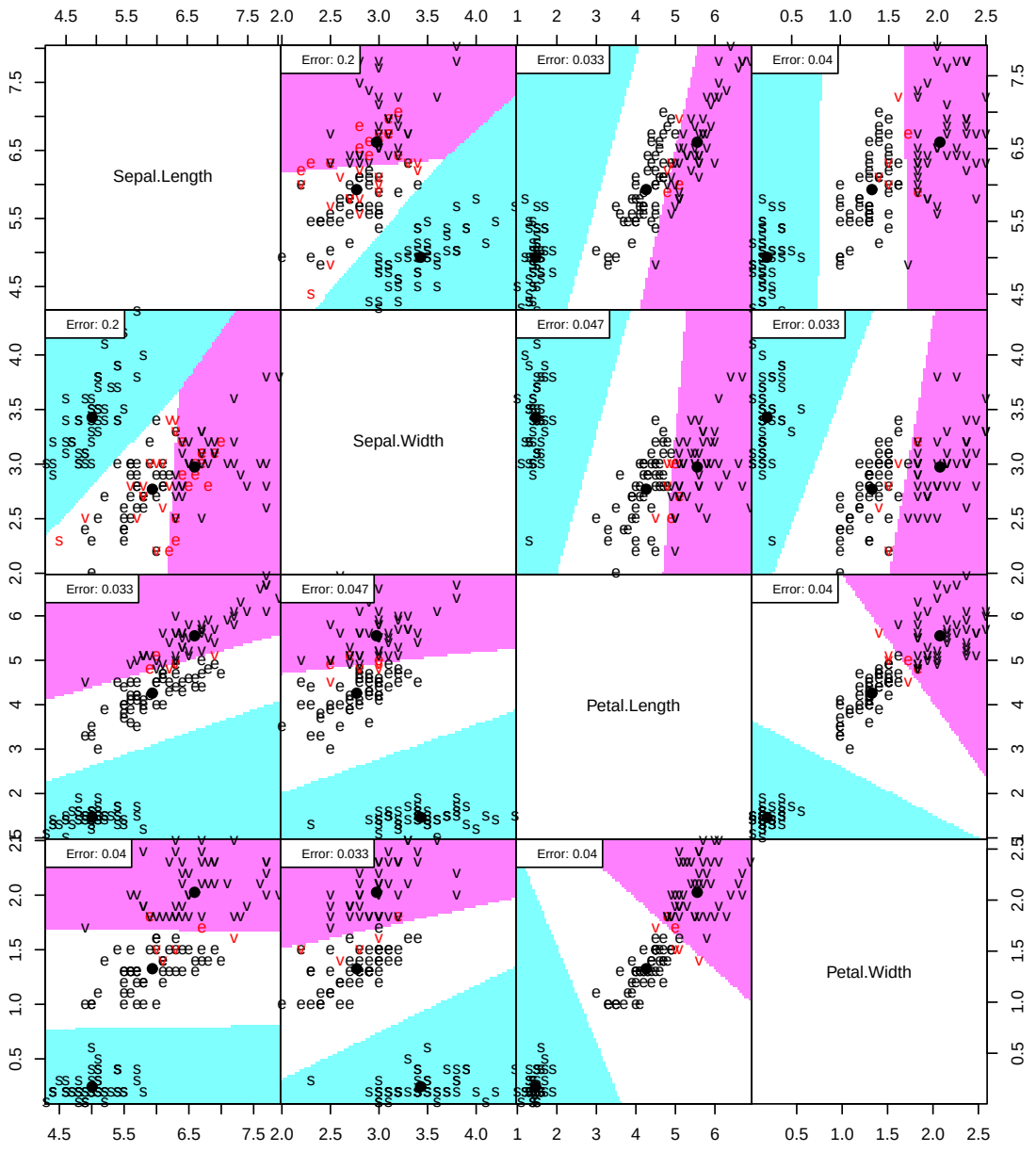


Figure 5: Classifying the Iris data using LDA. The multiple figure array illustrates the classification of observations based on LDA for every combination of two features. The classification boundaries and error are obtained by simply restricting the data to a given pair of features before fitting the model. In these plots, "s" represents the class label *Iris setosa*, "e" represents the class label *Iris versicolor*, and "v" represents the class label *Iris virginica*. The red letters illustrate the misclassified observations.

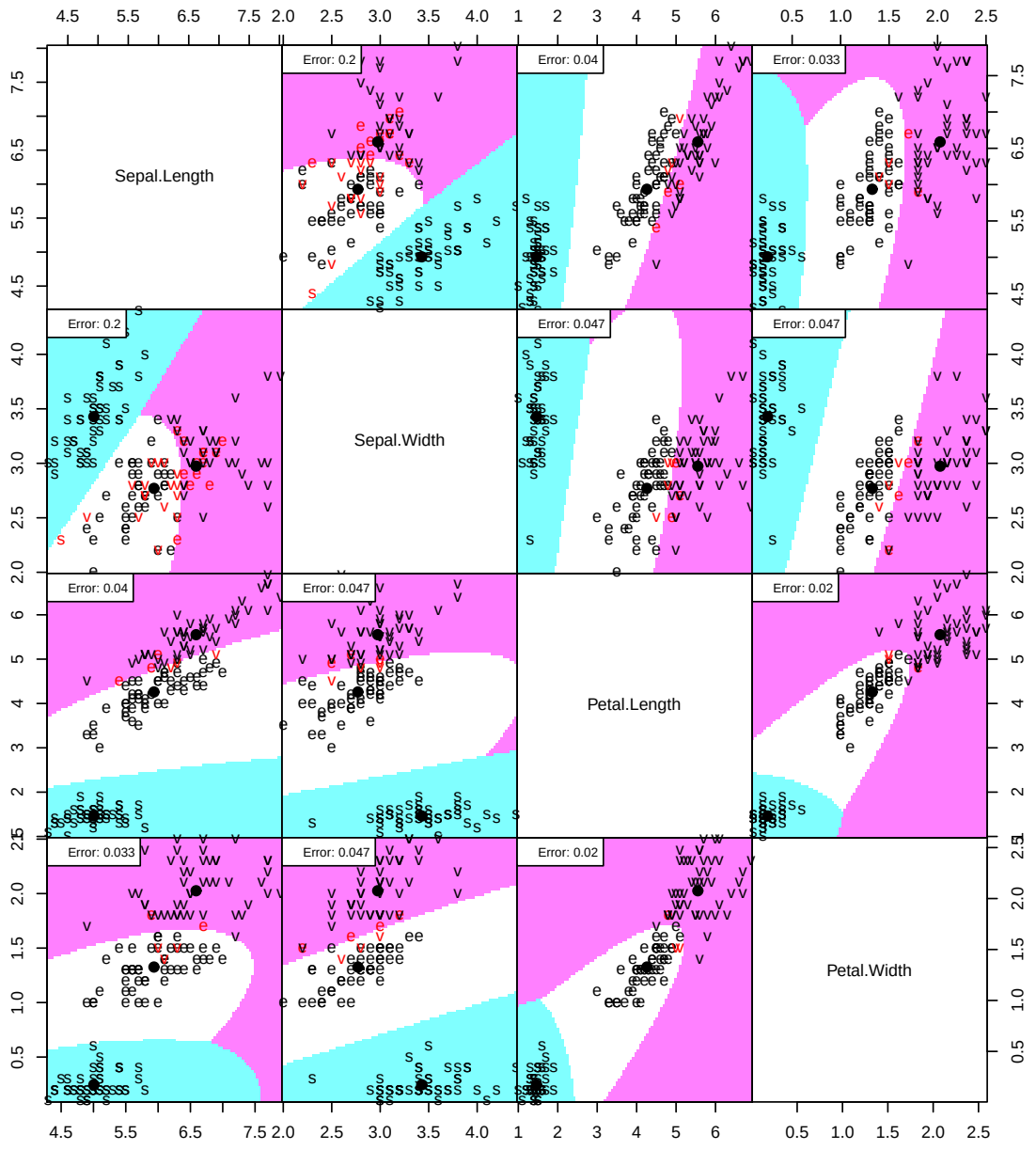


Figure 6: Classifying the Iris data using QDA. The multiple figure array illustrates the classification of observations based on QDA for every combination of two features. The classification boundaries are displayed and the classification error by simply casting the data onto these two features are calculated. In these plots, "s" represents the class label *Iris setosa*, "e" represents the class label *Iris versicolor*, and "v" represents the class label *Iris virginica*. The red letters illustrate the misclassified observations.

4 Fisher Linear Discriminant Analysis

There is another version of linear discriminant analysis due to Fisher (1936). The idea is to first reduce the covariates to one dimension by projecting the data onto a line. Algebraically, this means replacing the covariate $X = (X_1, \ldots, X_d)^T$ with a linear combination $U = w^T X = \sum_{j=1}^d w_j X_j$. The goal is to choose the vector $w = (w_1, \ldots, w_d)^T$ that "best separates the data into two groups." Then we perform classification with the one-dimensional covariate U instead of X.

What do we mean by "best separates the data into two groups"? Formally, we would like the two groups to have means that as far apart as possible relative to their spread. Let μ_j denote the mean of X for $Y = j$, $j = 0, 1$. And let Σ be the covariance matrix of X. Then, for $j = 0, 1$, $\mathbb{E}(U|Y = j) = \mathbb{E}(w^T X|Y = j) = w^T \mu_j$ and $\text{Var}(U) = w^T \Sigma w$. Define the separation by

$$\begin{aligned} J(w) &= \frac{\big(\mathbb{E}(U|Y=0) - \mathbb{E}(U|Y=1)\big)^2}{w^T \Sigma w} \\ &= \frac{(w^T \mu_0 - w^T \mu_1)^2}{w^T \Sigma w} \\ &= \frac{w^T (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T w}{w^T \Sigma w}. \end{aligned}$$

J is sometimes called the Rayleigh coefficient. Our goal is to find w that maximizes $J(w)$. Since $J(w)$ involves unknown population quantities $\Sigma_0, \Sigma_1, \mu_0, \mu_1$, we estimate J as follows. Let $n_j = \sum_{i=1}^n I(Y_i = j)$ be the number of observations in class j, let $\widehat{\mu}_j$ be the sample mean vector of the X's for class j, and let Σ_j be the sample covariance matrix for all observations in class j. Define

$$\widehat{J}(w) = \frac{w^T S_B w}{w^T S_W w} \tag{38}$$

where

$$\begin{aligned} S_B &= (\widehat{\mu}_0 - \widehat{\mu}_1)(\widehat{\mu}_0 - \widehat{\mu}_1)^T, \\ S_W &= \frac{(n_0 - 1)S_0 + (n_1 - 1)S_1}{(n_0 - 1) + (n_1 - 1)}. \end{aligned}$$

Theorem 8 *The vector*

$$\widehat{w} = S_W^{-1}(\widehat{\mu}_0 - \widehat{\mu}_1) \tag{39}$$

is a maximizer of $\widehat{J}(w)$.

Proof. Maximizing $\widehat{J}(w)$ is equivalent to maximizing $w^T S_B w$ subject to the constraint that $w^T S_W w = 1$. This is a generalized eigenvalue problem. By the definition of eigenvector and

eigenvalue, the maximizer $\widehat{w}$ should be the eigenvector of $S_W^{-1}S_B$ corresponding to the largest eigenvalue. The key observation is that $S_B = (\widehat{\mu}_0 - \widehat{\mu}_1)(\widehat{\mu}_0 - \widehat{\mu}_1)^T$, which implies that for any vector w, $S_B w$ must be in the direction of $\widehat{\mu}_0 - \widehat{\mu}_1$. The desired result immediately follows. □

We call

$$f(x) = \widehat{w}^T x = (\widehat{\mu}_0 - \widehat{\mu}_1)^T S_W^{-1} x \tag{40}$$

the *Fisher linear discriminant function*. Given a cutting threshold $c_m \in \mathbb{R}$, Fisher's classification rule is

$$h(x) = \begin{cases} 0 & \text{if } \widehat{w}^T x \geq c_m \\ 1 & \text{if } \widehat{w}^T x < c_m. \end{cases} \tag{41}$$

Fisher's rule is the same as the Gaussian LDA rule in (26) when

$$c_m = \frac{1}{2}(\widehat{\mu}_0 - \widehat{\mu}_1)^T S_W^{-1}(\widehat{\mu}_0 + \widehat{\mu}_1) - \log\left(\frac{\widehat{\pi}_0}{\widehat{\pi}_1}\right). \tag{42}$$

5 Logistic Regression

One approach to binary classification is to estimate the regression function $m(x) = \mathbb{E}(Y|X = x) = \mathbb{P}(Y = 1|X = x)$ and, once we have an estimate $\widehat{m}(x)$, use the classification rule

$$\widehat{h}(x) = \begin{cases} 1 & \text{if } \widehat{m}(x) > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{43}$$

For binary classification problems, one possible choice is the linear regression model

$$Y = m(X) + \epsilon = \beta_0 + \sum_{j=1}^{d} \beta_j X_j + \epsilon. \tag{44}$$

The linear regression model does not explicitly constrain Y to take on binary values. A more natural alternative is to use *logistic regression*, which is the most common binary classification method.

Before we describe the logistic regression model, let's recall some basic facts about binary random variables. If Y takes values 0 and 1, we say that Y has a Bernoulli distribution with parameter $\pi_1 = \mathbb{P}(Y = 1)$. The probability mass function for Y is $p(y; \pi_1) = \pi_1^y (1 - \pi_1)^{1-y}$ for $y = 0, 1$. The likelihood function for π_1 based on iid data $Y_1, \ldots, Y_n$ is

$$\mathcal{L}(\pi_1) = \prod_{i=1}^{n} p(Y_i; \pi_1) = \prod_{i=1}^{n} \pi_1^{Y_i}(1 - \pi_1)^{1-Y_i}. \tag{45}$$

In the logistic regression model, we assume that

$$m(x) = \mathbb{P}(Y = 1|X = x) = \frac{\exp\big(\beta_0 + x^T\beta\big)}{1 + \exp\big(\beta_0 + x^T\beta\big)} \equiv \pi_1(x, \beta_0, \beta). \tag{46}$$

In other words, given $X = x$, Y is Bernoulli with mean $\pi_1(x, \beta_0, \beta)$. We can write the model as

$$\text{logit}\big(\mathbb{P}(Y = 1|X = x)\big) = \beta_0 + x^T\beta \tag{47}$$

where $\text{logit}(a) = \log(a/(1-a))$. The name "logistic regression" comes from the fact that $\exp(x)/(1+\exp(x))$ is called the logistic function.

Lemma 9 *Both linear regression and logistic regression models have linear decision boundaries.*

Proof. The linear decision boundary for linear regression is straightforward. The same result for logistic regression follows from the monotonicity of the logistic function. □

The parameters β_0 and $\beta = (\beta_1, \ldots, \beta_d)^T$ can be estimated by maximum conditional likelihood. The conditional likelihood function for β is

$$\mathcal{L}(\beta_0, \beta) = \prod_{i=1}^{n} \pi(x_i, \beta_0, \beta)^{Y_i}(1 - \pi(x_i, \beta_0, \beta))^{1-Y_i}.$$

Thus the conditional log-likelihood is

$$\begin{aligned} \ell(\beta_0, \beta) &= \sum_{i=1}^{n}\Big\{Y_i \log \pi(x_i, \beta_0, \beta) - (1 - y_i)\log(1 - \pi(x_i, \beta_0, \beta)\Big\} &(48)\\ &= \sum_{i=1}^{n}\Big\{Y_i(\beta_0 + x_i^T\beta) - \log\big(1 + \exp(\beta_0 + x_i^T\beta)\big)\Big\}. &(49)\end{aligned}$$

The maximum conditional likelihood estimators $\widehat{\beta}_0$ and $\widehat{\beta}$ cannot be found in closed form. However, the loglikelihood function is concave and can be efficiently solve by the Newton's method in an iterative manner as follows.

Note that the logistic regression classifier is essentially replacing the 0-1 loss with a smooth loss function. In other words, it uses a *surrogate loss function.*

For notational simplicity, we redefine (local to this section) the d-dimensional covariate x_i and parameter vector β as the following $(d+1)$-dimensional vectors:

$$x_i \leftarrow (1, x_i^T)^T \text{ and } \beta \leftarrow (\beta_0, \beta^T)^T. \tag{50}$$

Thus, we write $\pi_1(x, \beta_0, \beta)$ as $\pi_1(x, \beta)$ and $\ell(\beta_0, \beta)$ as $\ell(\beta)$.

To maximize $\ell(\beta)$, the $(k+1)$th Newton step in the algorithm replaces the kth iterate $\widehat{\beta}^{(k)}$ by

$$\widehat{\beta}^{(k+1)} \leftarrow \widehat{\beta}^{(k)} - \left(\frac{\partial^2 \ell(\widehat{\beta}^{(k)})}{\partial\beta\partial\beta^T}\right)^{-1} \frac{\partial \ell(\widehat{\beta}^{(k)})}{\partial\beta}. \tag{51}$$

The gradient $\partial s \frac{\partial \ell(\widehat{\beta}^{(k)})}{\partial\beta}$ and Hessian $\partial s \frac{\partial^2 \ell(\widehat{\beta}^{(k)})}{\partial\beta\partial\beta^T}$ are both evaluated at $\widehat{\beta}^{(k)}$ and can be written as

$$\frac{\partial \ell(\widehat{\beta}^{(k)})}{\partial\beta} = \sum_{i=1}^{n} (\pi(x_i, \widehat{\beta}^{(k)}) - Y_i) X_i \text{ and } \frac{\partial^2 \ell(\widehat{\beta}^{(k)})}{\partial\beta\partial\beta^T} = -\mathbb{X}^T \mathbb{W} \mathbb{X} \tag{52}$$

where $\mathbb{W} = \text{diag}(w_{11}^{(k)}, w_{22}^{(k)}, \ldots, w_{dd}^{(k)})$ is a diagonal matrix with

$$w_{ii}^{(k)} = \pi(x_i, \widehat{\beta}^{(k)})\big(1 - \pi(x_i, \widehat{\beta}^{(k)})\big). \tag{53}$$

Let $\pi_1^{(k)} = \big(\pi_1(x_1, \widehat{\beta}^{(k)}), \ldots, \pi_1(x_n, \widehat{\beta}^{(k)})\big)^T$, (51) can be written as

$$\begin{aligned}
\widehat{\beta}^{(k+1)} &= \widehat{\beta}^{(k)} + (\mathbb{X}^T \mathbb{W} \mathbb{X})^{-1} \mathbb{X}^T (y - \pi_1^{(k)}) && (54)\\
&= (\mathbb{X}^T \mathbb{W} \mathbb{X})^{-1} \mathbb{X}^T \mathbb{W} \big(\mathbb{X} \widehat{\beta}^{(k)} + \mathbb{W}^{-1} (y - \pi_1^{(k)})\big) && (55)\\
&= (\mathbb{X}^T \mathbb{W} \mathbb{X})^{-1} \mathbb{X}^T \mathbb{W} z^{(k)} && (56)
\end{aligned}$$

where $z^{(k)} \equiv (z_1^{(k)}, \ldots, z_n^{(k)})^T = \mathbb{X}^T \widehat{\beta}^{(k)} + \mathbb{W}^{-1}(y - \pi_1^{(k)})$ with

$$z_i^{(k)} = \log\left(\frac{\pi_1(x_i, \widehat{\beta}^{(k)})}{1 - \pi_1(x_i, \widehat{\beta}^{(k)})} + \frac{y_i - \pi_1(x_i, \widehat{\beta}^{(k)})}{\pi_1(x_i, \widehat{\beta}^{(k)})(1 - \pi_1(x_i, \widehat{\beta}^{(k)}))}\right). \tag{57}$$

Given the current estimate $\widehat{\beta}^{(k)}$, the above Newton iteration forms a quadratic approximation to the negative log-likelihood using Taylor expansion at $\widehat{\beta}^{(k)}$:

$$-\ell(\beta) = \underbrace{\frac{1}{2}(z - \mathbb{X}\beta)^T \mathbb{W} (z - \mathbb{X}\beta)}_{\ell_Q(\beta)} + \text{constant}. \tag{58}$$

The update equation (56) corresponds to solving a quadratic optimization

$$\widehat{\beta}^{(k+1)} = \underset{\beta}{\text{argmin}}\, \ell_Q(\beta). \tag{59}$$

We then get an iterative algorithm called *iteratively reweighted least squares*. See Figure 7.

Iteratively Reweighted Least Squares Algorithm

Choose starting values $\widehat{\beta}^{(0)} = (\widehat{\beta}_0^{(0)}, \widehat{\beta}_1^{(0)}, \ldots, \widehat{\beta}_d^{(0)})^T$ and compute $\pi_1(x_i, \widehat{\beta}^{(0)})$ using Equation (46), for $i = 1, \ldots, n$ with β_j replaced by its initial value $\widehat{\beta}_j^{(0)}$.

For $k = 1, 2, \ldots$, iterate the following steps until convergence.

1. Calculate $z_i^{(k)}$ according to (57) for $i = 1, \ldots, n$.
2. Calculate $\widehat{\beta}^{(k+1)}$ according to (56). This corresponds to doing a weighted linear regression of z on $\mathbb{X}$.
3. Update the $\pi(x_i, \widehat{\beta})$'s using (46) with the current estimate of $\widehat{\beta}^{(k+1)}$.

Figure 7: Finding the Logistic Regression MLE.

We can get the estimated standard errors of the final solution $\widehat{\beta}$. For the kth iteration, recall that the Fisher information matrix $I(\widehat{\beta}^{(k)})$ takes the form

$$I(\widehat{\beta}^{(k)}) = -\mathbb{E}\left(\frac{\partial^2 \ell(\widehat{\beta}^{(k)})}{\partial \beta \partial \beta^T}\right) \approx \mathbb{X}^T \mathbb{W} \mathbb{X}, \tag{60}$$

we estimate the standard error of $\widehat{\beta}_j$ as the jth diagonal element of $I(\widehat{\beta})^{-1}$.

Example 10 *We apply the logistic regression on the Coronary Risk-Factor Study (CORIS) data and yields the following estimates and Wald statistics W_j for the coefficients:*

Covariate	$\widehat{\beta}_j$	se	W_j	p-value
Intercept	*-6.145*	*1.300*	*-4.738*	*0.000*
sbp	*0.007*	*0.006*	*1.138*	*0.255*
tobacco	*0.079*	*0.027*	*2.991*	*0.003*
ldl	*0.174*	*0.059*	*2.925*	*0.003*
adiposity	*0.019*	*0.029*	*0.637*	*0.524*
famhist	*0.925*	*0.227*	*4.078*	*0.000*
typea	*0.040*	*0.012*	*3.233*	*0.001*
obesity	*-0.063*	*0.044*	*-1.427*	*0.153*
alcohol	*0.000*	*0.004*	*0.027*	*0.979*
age	*0.045*	*0.012*	*3.754*	*0.000*

6 Logistic Regression Versus LDA

There is a close connection between logistic regression and Gaussian LDA. Let (X, Y) be a pair of random variables where Y is binary and let $p_0(x) = p(x|Y = 0)$, $p_1(x) = p(x|Y = 1)$, $\pi_1 = \mathbb{P}(Y = 1)$. By Bayes' theorem,

$$\mathbb{P}(Y = 1|X = x) = \frac{p(x|Y = 1)\pi_1}{p(x|Y = 1)\pi_1 + p(x|Y = 0)(1 - \pi_1)} \tag{61}$$

If we assume that each group is Gaussian with the same covariance matrix Σ, i.e., $X|Y = 0 \sim N(\mu_0, \Sigma)$ and $X|Y = 1 \sim N(\mu_1, \Sigma)$, we have

$$\begin{aligned} \log\left(\frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = 0|X = x)}\right) &= \log\left(\frac{\pi}{1 - \pi}\right) - \frac{1}{2}(\mu_0 + \mu_1)^T\Sigma^{-1}(\mu_1 - \mu_0) && (62)\\ &\quad + x^T\Sigma^{-1}(\mu_1 - \mu_0) && (63)\\ &\equiv \alpha_0 + \alpha^T x. && (64) \end{aligned}$$

On the other hand, the logistic regression model is, by assumption,

$$\log\left(\frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = 0|X = x)}\right) = \beta_0 + \beta^T x.$$

These are the same model since they both lead to classification rules that are linear in x. The difference is in how we estimate the parameters.

This is an example of a generative versus a discriminative model. In Gaussian LDA we estimate the whole joint distribution by maximizing the full likelihood

$$\prod_{i=1}^{n} p(X_i, Y_i) = \underbrace{\prod_{i=1}^{n} p(X_i|Y_i)}_{\text{Gaussian}} \underbrace{\prod_{i=1}^{n} p(Y_i)}_{\text{Bernoulli}}. \tag{65}$$

In logistic regression we maximize the conditional likelihood $\prod_{i=1}^{n} p(Y_i|X_i)$ but ignore the second term $p(X_i)$:

$$\prod_{i=1}^{n} p(X_i, Y_i) = \underbrace{\prod_{i=1}^{n} p(Y_i|X_i)}_{\text{logistic}} \underbrace{\prod_{i=1}^{n} p(X_i)}_{\text{ignored}}. \tag{66}$$

Since classification only requires the knowledge of $p(y|x)$, we don't really need to estimate the whole joint distribution. Logistic regression leaves the marginal distribution $p(x)$ unspecified so it relies on less parametric assumption than LDA. This is an advantage of the logistic regression approach over LDA. However, if the true class conditional distributions are Gaussian, the logistic regression will be asymptotically less efficient than LDA, i.e. to achieve a certain level of classification error, the logistic regression requires more samples.

7 Regularized Logistic Regression

As with linear regression, when the dimension d of the covariate is large, we cannot simply fit a logistic model to all the variables without experiencing numerical and statistical problems. Akin to the lasso, we will use *regularized logistic regression*, which includes *sparse logistic regression* and *ridge logistic regression*.

Let $\ell(\beta_0, \beta)$ be the log-likelihood defined in (49). The *sparse logistic regression* estimator is an ℓ_1-regularized conditional log-likelihood estimator

$$\widehat{\beta}_0, \widehat{\beta} = \underset{\beta_0,\beta}{\operatorname{argmin}} \Big\{ -\ell(\beta_0, \beta) + \lambda \|\beta\|_1 \Big\}. \tag{67}$$

Similarly, the *ridge logistic regression* estimator is an ℓ_2-regularized conditional log-likelihood estimator

$$\widehat{\beta}_0, \widehat{\beta} = \underset{\beta_0,\beta}{\operatorname{argmin}} \Big\{ -\ell(\beta_0, \beta) + \lambda \|\beta\|_2^2 \Big\}. \tag{68}$$

The algorithm for logistic ridge regression only requires a simple modification of the iteratively reweighed least squares algorithm and is left as an exercise.

For sparse logistic regression, an easy way to calculate $\widehat{\beta}_0$ and $\widehat{\beta}$ is to apply a ℓ_1-regularized Newton procedure. Similar to the Newton method for unregularized logistic regression, for the kth iteration, we first form a quadratic approximation to the negative log-likelihood $\ell(\beta_0, \beta)$ based on the current estimates $\widehat{\beta}^{(k)}$.

$$-\ell(\beta_0, \beta) = \underbrace{\frac{1}{2} \sum_{i=1}^{n} w_{ii} \big(z_i^{(k)} - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij} \big)^2}_{\ell_Q(\beta_0, \beta)} + \text{constant}. \tag{69}$$

where w_{ii} and $z_i^{(k)}$ are defined in (53) and (57). Since we have a ℓ_1-regularization term, the updating formula for the estimate in the $(k+1)$th step then becomes

$$\widehat{\beta}^{(k+1)}, \widehat{\beta}^{(k+1)} = \underset{\beta_0,\beta}{\operatorname{argmin}} \Big\{ \frac{1}{2} \sum_{i=1}^{n} w_{ii} \big(z_i^{(k)} - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij} \big)^2 + \lambda \|\beta\|_1 \Big\}. \tag{70}$$

This is a weighted lasso problem and can be solved using coordinate descent. See Figure 8.

Even though the above iterative procedure does not guarantee theoretical convergence, it works very well in practice.

Sparse Logistic Regression Using Coordinate Descent

Choose starting values $\widehat{\beta}^{(0)} = (\widehat{\beta}_0^{(0)}, \widehat{\beta}_1^{(0)}, \ldots, \widehat{\beta}_d^{(0)})^T$

(Outer loop) For $k = 1, 2, \ldots$, iterate the following steps until convergence.

1. For $i = 1, \ldots, n$, calculate $\pi_1(x_i, \widehat{\beta}^{(k)})$, $z_i^{(k)}$, $w_{ii}^{(k)}$ according to (46), (57), and (53).

2. $\alpha_0 = \partial s \frac{\sum_{i=1}^n w_{ii}^{(k)} z_i^{(k)}}{\sum_{i=1}^n w_{ii}^{(k)}}$ and $\alpha_\ell = \widehat{\beta}_\ell^{(k)}$ for $\ell = 1, \ldots d$.

3. (Inner loop) iterate the following steps until convergence

   For $j \in \{1, \ldots, d\}$

   (a) For $i = 1, \ldots, n$, calculate $r_{ij} = z_i^{(k)} - \alpha_0 - \sum_{\ell \neq j} \alpha_\ell x_{i\ell}$.

   (b) Calculate $u_j^{(k)} = \sum_{i=1}^n w_{ii}^{(k)} r_{ij} x_{ij}$ and $v_j^{(k)} = \sum_{i=1}^n w_{ii}^{(k)} x_{ij}^2$.

   (c) $\alpha_j = \text{sign}\big(u_j^{(k)}\big) \left[\frac{|u_j^{(k)}| - \lambda}{v_j^{(k)}} \right]_+$.

4. $\widehat{\beta}_0^{(k+1)} = \alpha_0$ and $\widehat{\beta}_\ell^{(k+1)} = \alpha_\ell$ for $\ell = 1, \ldots d$.

5. Update the $\pi(x_i, \widehat{\beta})$'s using (46) with the current estimate of $\widehat{\beta}^{(k+1)}$.

Figure 8: Sparse Logistic Regression

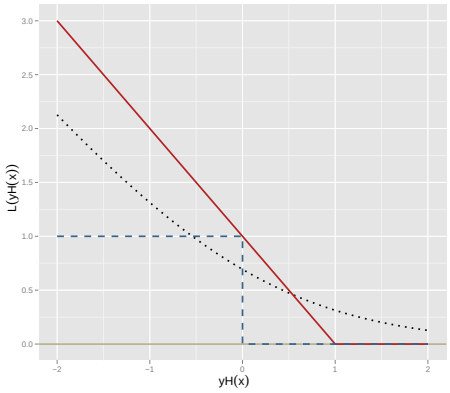


Figure 9: The 0-1 classification loss (blue dashed line), hinge loss (red solid line) and logistic loss (black dotted line).

8 Support Vector Machines

The *support vector machine* (*SVM*) classifier is a linear classifier that replaces the 0-1 loss with a surrogate loss function. (Logistic regression uses a different surrogate.) In this section, the outcomes are coded as -1 and $+1$. The 0-1 loss is $L(x, y, \beta) = I(y \neq h_\beta(x)) = I(yH_\beta(x) < 0)$ with the *hinge loss* $L_{\text{hinge}}\big(y_i, H(x_i)\big) \equiv \big[1 - Y_i H(X_i)\big]_+$ instead of the logistic loss. This is the smallest convex function that lies above the 0-1 loss. (When we discuss nonparameteric classifiers, we will consider more general support vector machines.)

The support vector machine classifier is $\widehat{h}(x) = I\big(\widehat{H}(x) > 0\big)$ where the hyperplane $\widehat{H}(x) = \widehat{\beta}_0 + \widehat{\beta}^T x$ is obtained by minimizing

$$\sum_{i=1}^{n} \big[1 - Y_i H(X_i)\big]_+ + \frac{\lambda}{2}\|\beta\|_2^2 \tag{71}$$

where $\lambda > 0$ and the factor $1/2$ is only for notational convenience.

Figure 9 compares the hinge loss, 0-1 loss, and logistic loss. The advantage of the hinge loss is that it is convex, and it has a corner which leads to efficient computation and the minimizer of $\mathbb{E}\big(1 - YH(X)\big)_+$ is the Bayes rule. A disadvantage of the hinge loss is that one can't recover the regression function $m(x) = \mathbb{E}(Y|X = x)$.

The SVM classifier is often developed from a geometric perspective. Suppose first that the data are *linearly separable*, that is, there exists a hyperplane that perfectly separates the two classes. How can we find a separating hyperplane? LDA is not guaranteed to find it. A separating hyperplane will minimize

$$-\sum_{i \in \mathcal{M}} Y_i H(X_i).$$

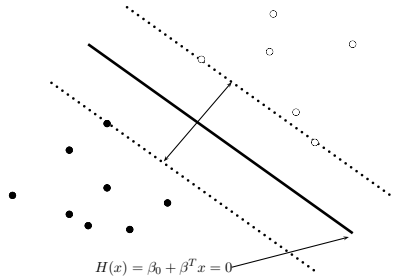


Figure 10: The hyperplane $H(x)$ has the largest margin of all hyperplanes that separate the two classes.

where $\mathcal{M}$ is the index set of all misclassified data points. Rosenblatt's perceptron algorithm takes starting values and iteratively updates the coefficients as:

$$\begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} \longleftarrow \begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} + \rho \begin{pmatrix} Y_i X_i \\ Y_i \end{pmatrix}$$

where $\rho > 0$ is the learning rate. If the data are linearly separable, the perceptron algorithm is guaranteed to converge to a separating hyperplane. However, there could be many separating hyperplanes. Different starting values may lead to different separating hyperplanes. The question is, which separating hyperplane is the best?

Intuitively, it seems reasonable to choose the hyperplane "furthest" from the data in the sense that it separates the +1's and -1's and maximizes the distance to the closest point. This hyperplane is called the *maximum margin hyperplane*. The margin is the distance from the hyperplane to the nearest data point Points on the boundary of the margin are called *support vectors*. See Figure 10. The goal, then, is to find a separating hyperplane which maximizes the margin. After some simple algebra, we can show that (71) exactly achieves this goal. In fact, (71) also works for data that are not linearly separable.

The unconstrained optimization problem (71) can be equivalently formulated in constrained form:

$$\min_{\beta_0, \beta} \quad \left\{ \frac{1}{2} \|\beta\|_2^2 + \frac{1}{\lambda} \sum_{i=1}^{n} \xi_i \right\} \tag{72}$$

$$\text{subject to} \quad \forall i, \xi_i \geq 0 \text{ and } \xi_i \geq 1 - y_i H(x_i). \tag{73}$$

Given two vectors a and b, let $\langle a, b \rangle = a^T b = \sum_j a_j b_j$ denote the inner product of a and b. The following lemma provides the dual of the optimization problem in (72).

Lemma 11 *The dual of the SVM optimization problem in* (72) *takes the form*

$$\widehat{\alpha} = \underset{\alpha\in\mathbb{R}^n}{\operatorname{argmax}}\Big\{\sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{k=1}^n \alpha_i\alpha_k Y_i y_k \langle X_i, x_k\rangle\Big\} \tag{74}$$

$$\text{subject to } 0 \le \alpha_1, \ldots, \alpha_n \le \frac{1}{\lambda} \text{ and } \sum_{i=1}^n \alpha_i y_i = 0, \tag{75}$$

with the primal-dual relationship $\widehat{\beta} = \sum_{i=1}^n \widehat{\alpha} y_i x_i$. *We also have*

$$\widehat{\alpha}_i\Big(1 - \xi_i - y_i\big(\widehat{\beta}_0 + \widehat{\beta}x_i\big)\Big) = 0, \quad i = 1, \ldots, n. \tag{76}$$

Proof. Let $\alpha_i, \gamma_i \ge 0$ be the Lagrange multipliers. The Lagrangian function can be written as

$$L(\xi, \beta, \beta_0, \alpha, \gamma) = \frac{1}{2}\|\beta\|_2^2 + \frac{1}{\lambda}\sum_{i=1}^n \xi_i + \sum_{i=1}^n \alpha_i\big(1 - \xi_i - y_i H(x_i)\big) - \sum_{i=1}^n \gamma_i\xi_i. \tag{77}$$

The Karush-Kuhn-Tucker conditions are

$$\forall i,\ \alpha_i \ge 0, \gamma_i \ge 0,\ \xi_i \ge 0 \text{ and } \xi_i \ge 1 - y_i H(x_i), \tag{78}$$

$$\beta = \sum_{i=1}^n \alpha_i y_i x_i, \quad \sum_{i=1}^n \alpha_i y_i = 0 \ \text{ and } \ \alpha_i + \gamma_i = 1/\lambda, \tag{79}$$

$$\forall i,\ \alpha_i\big(1 - \xi_i - y_i H(x_i)\big) = 0 \ \text{ and } \ \gamma_i\xi_i = 0. \tag{80}$$

The dual formulation in (74) follows by plugging (78) and (79) into (77). The primal-dual complementary slackness condition (76) is obtained from the first equation in (80). □

The dual problem (74) is easier to solve than the primal problem (71). The data points (X_i, y_i) for which $\widehat{\alpha}_i > 0$ are called *support vectors*. By (76) and (72), for all the data points (x_i, y_i) satisfying $y_i\big(\widehat{\beta}_0 + \widehat{\beta}^T x_i\big) > 1$, there must be $\widehat{\alpha}_i = 0$. The solution for the dual problem is sparse. From the first equality in (79), we see that the final estimate $\widehat{\beta}$ is a linear combination only of these support vectors. Among these support vectors, if $\alpha_i < 1/\lambda$, we call (x_i, y_i) a *margin point*. For a margin point (x_i, y_i), the last equality in (79) implies that $\gamma_i > 0$, then the second equality in (80) implies $\xi_i = 0$. Moreover, using the first equality in (80) , we get

$$\widehat{\beta}_0 = -Y_i X_i^T \widehat{\beta}. \tag{81}$$

Therefore, once $\widehat{\beta}$ is given, we could calculate $\widehat{\beta}_0$ using any margin point (X_i, y_i).

Example 12 *We consider classifying two types of irises, versicolor and viriginica. There are 50 observations in each class. The covariates are "Sepal.Length" "Sepal.Width" "Petal.Length" and "Petal.Width". After fitting a SVM we get a 3/100 misclassification rate. The SVM uses 33 support vectors.*

9 Case Study I: Supernova Classification

A *supernova* is an exploding star. Type Ia supernovae are a special class of supernovae that are very useful in astrophysics research. These supernovae have a characteristic *light curve*, which is a plot of the luminosity of the supernova versus time. The maximum brightness of all type Ia supernovae is approximately the same. In other words, the true (or absolute) brightness of a type Ia supernova is known. On the other hand, the apparent (or observed) brightness of a supernova can be measured directly. Since we know both the absolute and apparent brightness of a type Ia supernova, we can compute its distance. Because of this, type Ia supernovae are sometimes called standard candles. Two supernovae, one type Ia and one non-type Ia, are illustrated in Figure 11. Astronomers also measure the *redshift* of the supernova, which is essentially the speed at which the supernova is moving away from us. The relationship between distance and redshift provides important information for astrophysicists in studying the large scale structure of the universe.

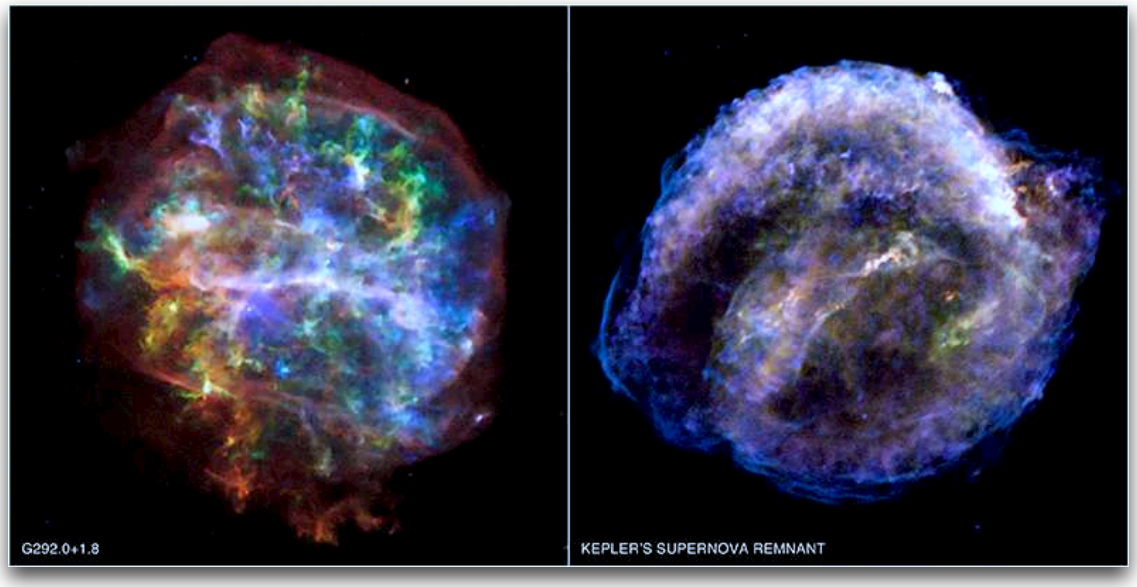


Figure 11: Two supernova remnants from the NASA's Chandra X-ray Observatory study. The image in the right panel, the so-called Kepler supernova remnant, is "Type Ia". Such supernovae have a very symmetric, circular remnant. This type of supernova is thought to be caused by a thermonuclear explosion of a white dwarf, and is often used by astronomers as a "standard candle" for measuring cosmic distances. The image in the left panel is a different type of supernova that comes from "core collapse." Such supernovae are distinctly more asymmetric. (Credit: NASA/CXC/UCSC/L. Lopez et al.)

A challenge in astrophysics is to classify supernovae to be type Ia versus other types. [**?**] released a mixture of real and realistically simulated supernovae and challenged the scientific community to find effective ways to classify the type Ia supernovae. The dataset consists of

about 20,000 simulated supernovae. For each supernova, there are a few noisy measurements of the flux (brightness) in four different filters. These four filters correspond to different wavelengths. Specifically, the filters correspond to the g-band (green), r-band (red), i-band (infrared) and z-band (blue). See Figure 12.

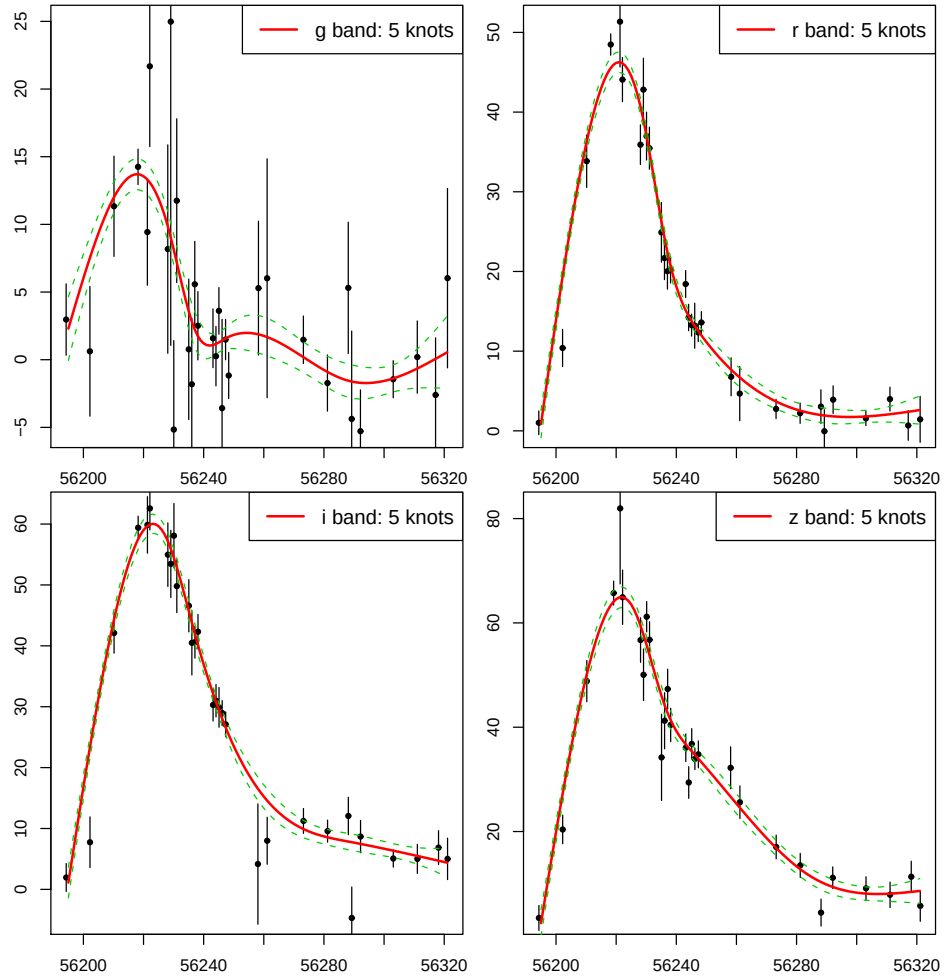
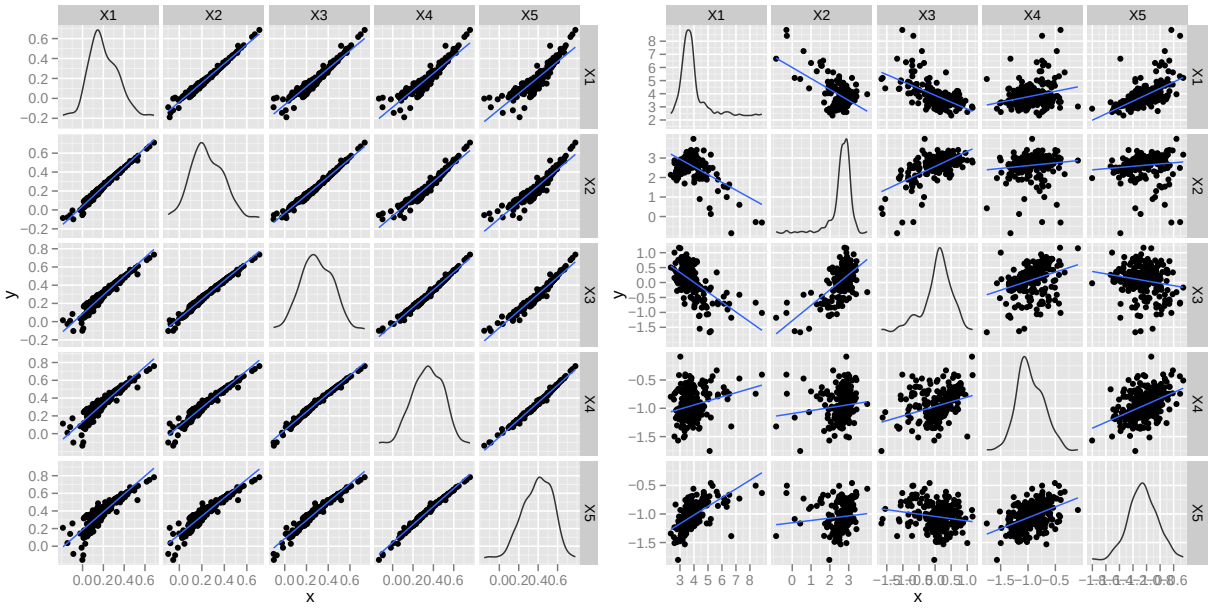


Figure 12: Four filters (g, r, i, z-bands) corresponding to a type Ia supernova *DES-SN000051*. For each band, a weighted regression spline fit (solid red) with the corresponding standard error curves (dashed green) is provided. The black points with bars represent the flux values and their estimated standard errors.

To estimate a linear classifier we need to preprocess the data to extract features. One difficulty is that each supernova is only measured at a few irregular time points, and these time points are not aligned. To handle this problem we use nonparametric regression to get a smooth curve. (We used the estimated measurement errors of each flux as weights and fitted a weighted least squares regression spline to smooth each supernova.) All four filters

of each supernova are then aligned according to the peak of the r-band. We also rescale so that all the curves have the same maximum.

The goal of this study is to build linear classifiers to predict whether a supernova is type Ia or not. For simplicity, we only use the information in the r-band. First, we align the fitted regression spline curves of all supernovae by calibrating their maximum peaks and set the corresponding time point to be day 0. There are altogether $19,679$ supernovae in the dataset with $1,367$ being labeled. To get a higher signal-to-noise ratio, we throw away all supernovae with less than 10 r-band flux measurements. We finally get a trimmed dataset with 255 supernovae, 206 of which are type Ia and 49 of which are non-type Ia.



time domain frequency domain

Figure 13: The matrix of scatterplots of the first five features of the supernova data. On the diagonal cells are the estimated univariate densities of each feature. The off-diagonal cells visualize the pairwise scatter plots of the two corresponding variables with a least squares fit. We see the time-domain features are highly correlated, while the frequency-domain features are almost uncorrelated.

We use two types of features: the *time-domain* features and *frequency-domain* features. For the time-domain features, the features are the interpolated regression spline values according to an equally spaced time grid. In this study, the grid has length 100, ranging from day -20 to day 80. Since all the fitted regression curves have similar global shapes, the time-domain features are expected to be highly correlated. This conjecture is confirmed by the matrix of scatterplots of the first five features in 13. To make the features less correlated, we also extract the frequency-domain features, which are simply the discrete cosine transformations of the corresponding time-domain features. More specifically, given the time domain features $X_1, \ldots, X_d$ ($d = 100$), Their corresponding frequency domain features $\widetilde{X}_1, \ldots, \widetilde{X}_d$ can be

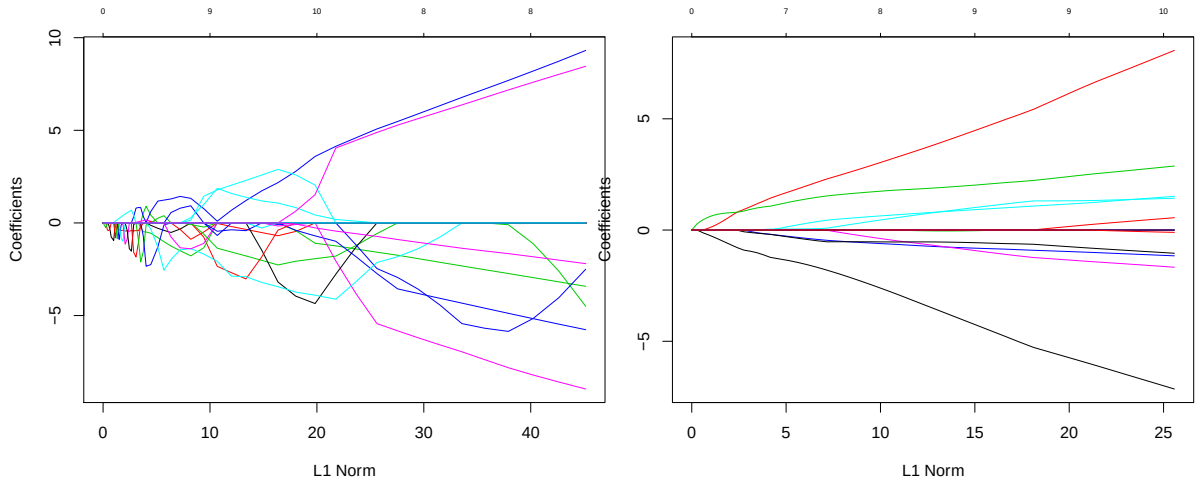
written as

$$\widetilde{X}_j = \frac{2}{d}\sum_{k=1}^{d} X_k \cos\Big[\frac{\pi}{d}\big(k-\frac{1}{2}\big)(j-1)\Big] \text{ for } j = 1, \ldots, d. \tag{82}$$

The right panel of Figure 13 illustrates the scatter matrix of the first 5 frequency-domain features. In contrast to the time-domain features, the frequency-domain features have low correlation.

We apply sparse logistic regression (LR), support vector machines (SVM), diagonal linear discriminant analysis (DLDA), and diagonal quadratic discriminant analysis (DQDA) on this dataset. For each method, we conduct 100 runs, within each run, 40% of the data are randomly selected as training and the remaining 60% are used for testing.

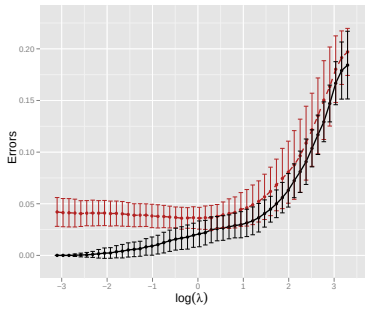
Figure 14 illustrates the regularization paths of sparse logistic regression using the ime-domain and frequency-domain features. A regularization path provides the coefficient value of each feature over all regularization parameters. Since the time-domain features are highly correlated, the corresponding regularization path is quite irregular. In contrast, the paths for the frequency-domain features behave stably.



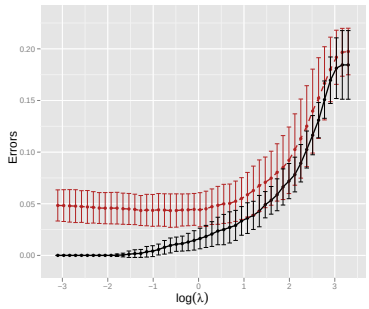
time domain frequency domain

Figure 14: The regularization paths of sparse logistic regression using the features of time-domain and frequency-domain. The vertical axis corresponds to the values of the coefficients, plotted as a function of their ℓ_1-norm. The path using time-domain features are highly irregular, while the path using frequency-domain features are more stable.

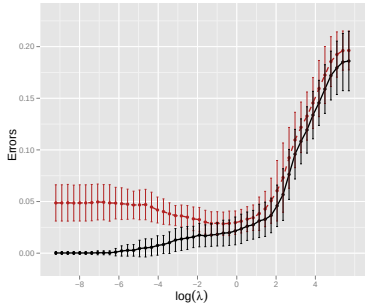
Figure 15 compares the classification performance of all these methods. The results show that classification in the frequency domain is not helpful. The regularization paths of the SVM are the same in both the time and frequency domains. This is expected since the discrete cosine transformation is an orthonormal transformation, which corresponds to rotatating the data in the feature space while preserving their Euclidean distances and inner products. It is easy



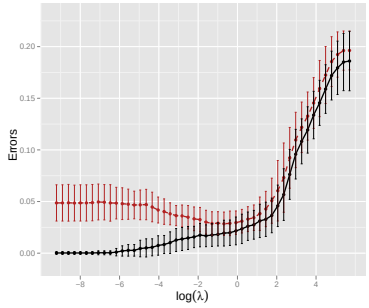
Sparse LR (time domain)



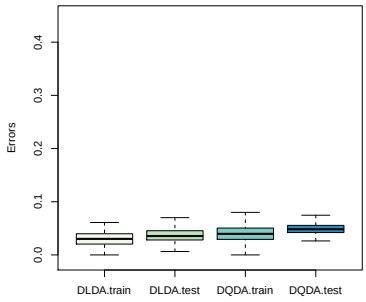
Sparse LR (frequency domain)



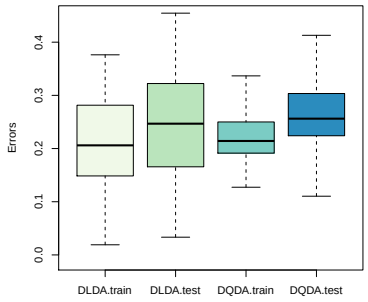
SVM (time domain)



SVM (frequency domain)



Discriminant Analysis (time domain)



Discriminant Analysis (frequency domain)

Figure 15: Comparison of different methods on the supernova dataset using both the time-domain (left column) and frequency-domain features (right Column). Top four figures: mean error curves (black: training error, red: test error) and their corresponding standard error bars for sparse logistic regression (LR) and support vector machines (SVM). Bottom two figures: boxplots of the training and test errors of diagonal linear discriminant analysis (DLDA) and diagonal quadratic discriminant analysis (DQDA). For the time-domain features, the SVM achieves the smallest test error among all methods.

to see that the SVM is rotation invariant. Sparse logistic regression is not rotation invariant due to the ℓ_1-norm regularization term. The performance of the sparse logistic regression in the frequency domain is worse than that in the time domain. The DLDA and DQDA are also not rotation invariant; their performances decreases significantly in the frequency domain compared to those in the time domain. In both time and frequency domains, the SVM outperforms all the other methods. Then follows sparse logistic regression, which is better than DLDA and DQDA.

10 Case Study II: Political Blog Classification

In this example, we classify political blogs according to whether their political leanings are *liberal* or *conservative*. Snapshots of two political blogs are shown in Figure 16.

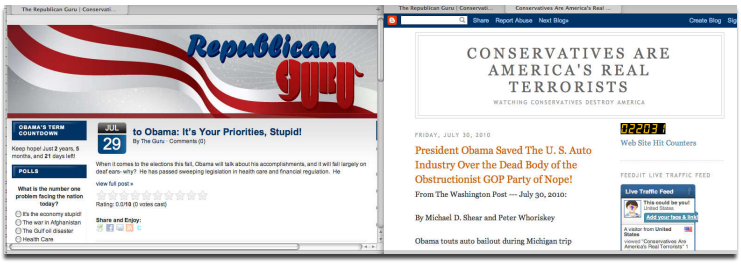
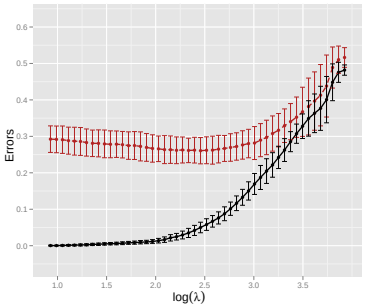


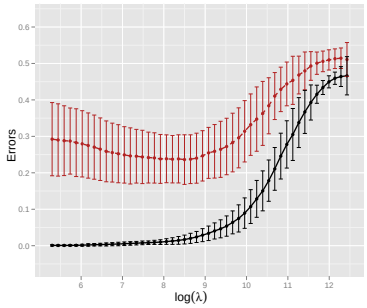
Figure 16: Examples of two political blogs with different orientations, one conservative and the other liberal.

The data consist of 403 political blogs in a two-month window before the 2004 presidential election. Among these blogs, 205 are *liberal* and 198 are *conservative*. We use *bag-of-words* features, i.e., each unique word from these 403 blogs serves as a feature. For each blog, the value of a feature is the number of occurrences of the word normalized by the total number of words in the blog. After converting all words to lower case, we remove stop words and only retain words with at least 10 occurrences across all the 403 blogs. This results in 23,955 features, each of which corresponds to an English word. Such features are only a crude representation of the text represented as an unordered collection of words, disregarding all grammatical structure. We also extracted features that use hyperlink information. In particular, we selected 292 out of the 403 blogs that are heavily linked to, and for each blog $i = 1, \ldots, 403$, its linkage information is represented as a 292-dimensional binary vector $(x_{i1}, \ldots, x_{i292})^T$ where $x_{ij} = 1$ if the ith blog has a link to the jth feature blog. The total number of covariates is then $23{,}955 + 292 = 24{,}247$. Even though the link features only constitute a small proportion, they are important for predictive accuracy.

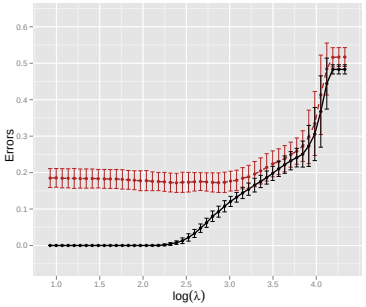
We run the full regularization paths of sparse logistic regression and support vector machines,



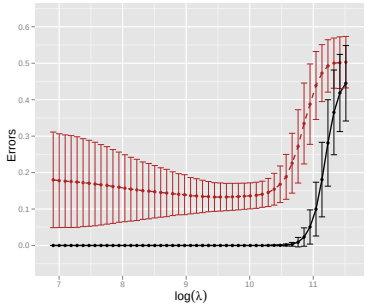
Sparse LR



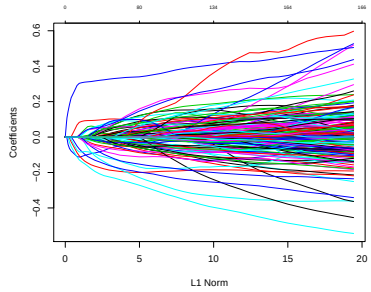
SVM



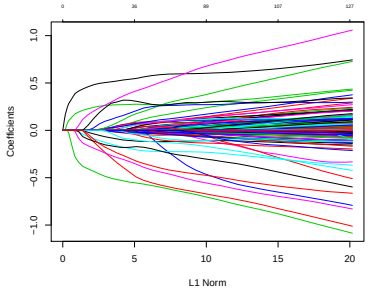
Sparse LR (with linkage information)



SVM (with linkage information)



Sparse LR path (no linkage information)



Sparse LR path (with linkage information)

Figure 17: Comparison of the sparse logistic regression (LR) and support vector machine (SVM) on the political blog data. (Top four figures): The mean error curves (Black: training error, Red: test error) and their corresponding standard error bars of the sparse LR and SVM, with and without the linkage information. (Bottom two figures): Two typical regularization paths of the sparse logistic LR with and without the linkage information. On this dataset, the diagonal linear discriminant analysis (DLDA) achieves a test error 0.303 (sd = 0.07) without the linkage information and a test error 0.159 (sd = 0.02) with the linkage information.

100 times each. For each run, the data are randomly partitioned into training (60%) and testing (40%) sets. Figure 17 shows the mean error curves with their standard errors. From Figure 17, we see that linkage information is crucial. Without the linkage features, the smallest mean test error of the support vector machine along the regularization path is 0.247, while that of the sparse logistic regression is 0.270. With the link features, the smallest test error for the support vector machine becomes 0.132. Although the support vector machine has a better mean error curve, it has much larger standard error. Two typical regularization paths for sparse logistic regression with and without using the link features are provided at the bottom of Figure 17. By examining these paths, we see that when the link features are used, 11 of the first 20 selected features are link features. In this case, although thev class conditional distribution is obviously not Gaussian, we still apply the diagonal linear discriminant analysis (DLDA) on this dataset for a comparative study. Without the linkage features, the DLDA has a mean test error 0.303 (sd = 0.07). With the linkage features, DLDA has a mean test error 0.159 (sd = 0.02).