Advanced Computational Econometrics: Machine Learning

Chapter 2: Linear Classification

Thierry Denœux

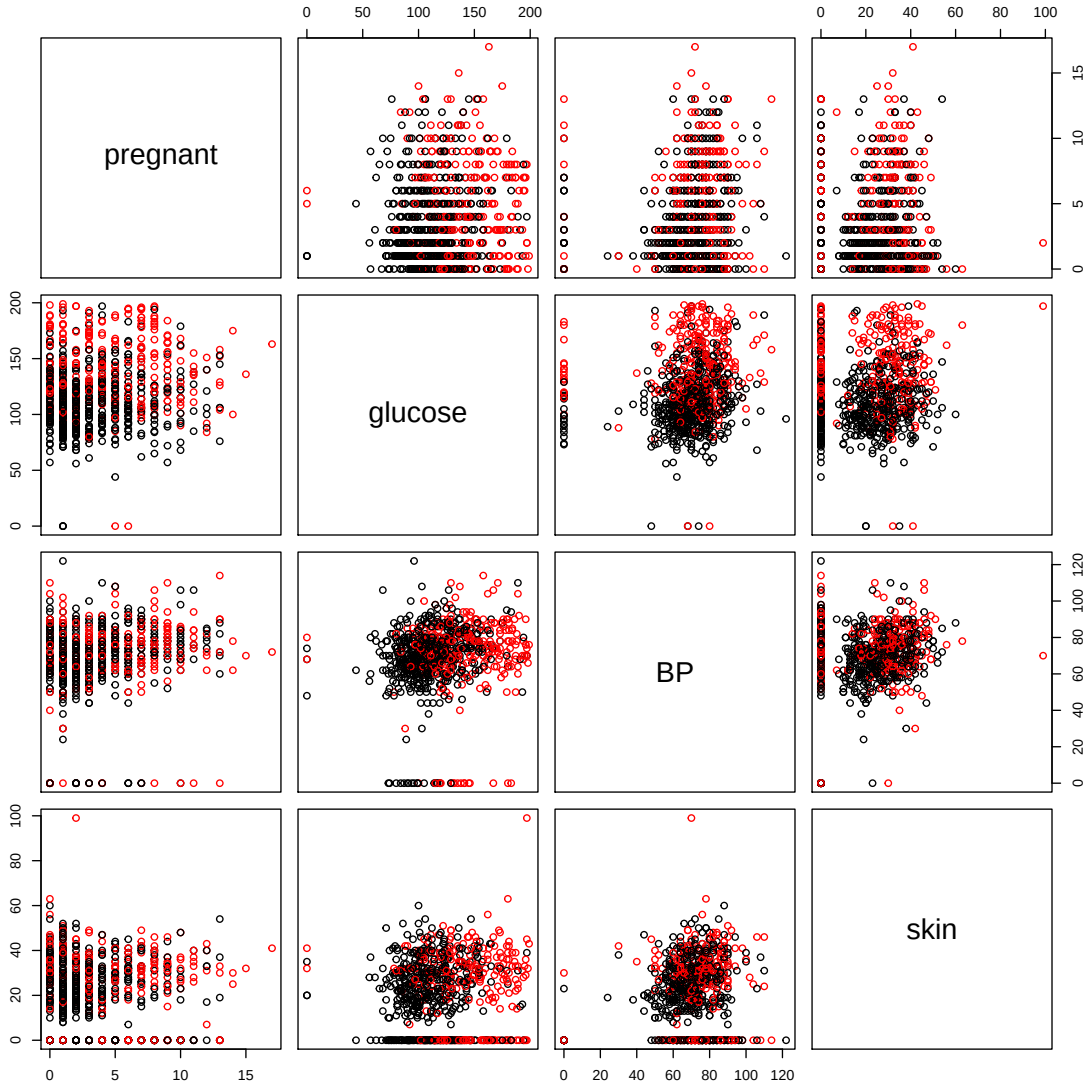
July 2019

Overview

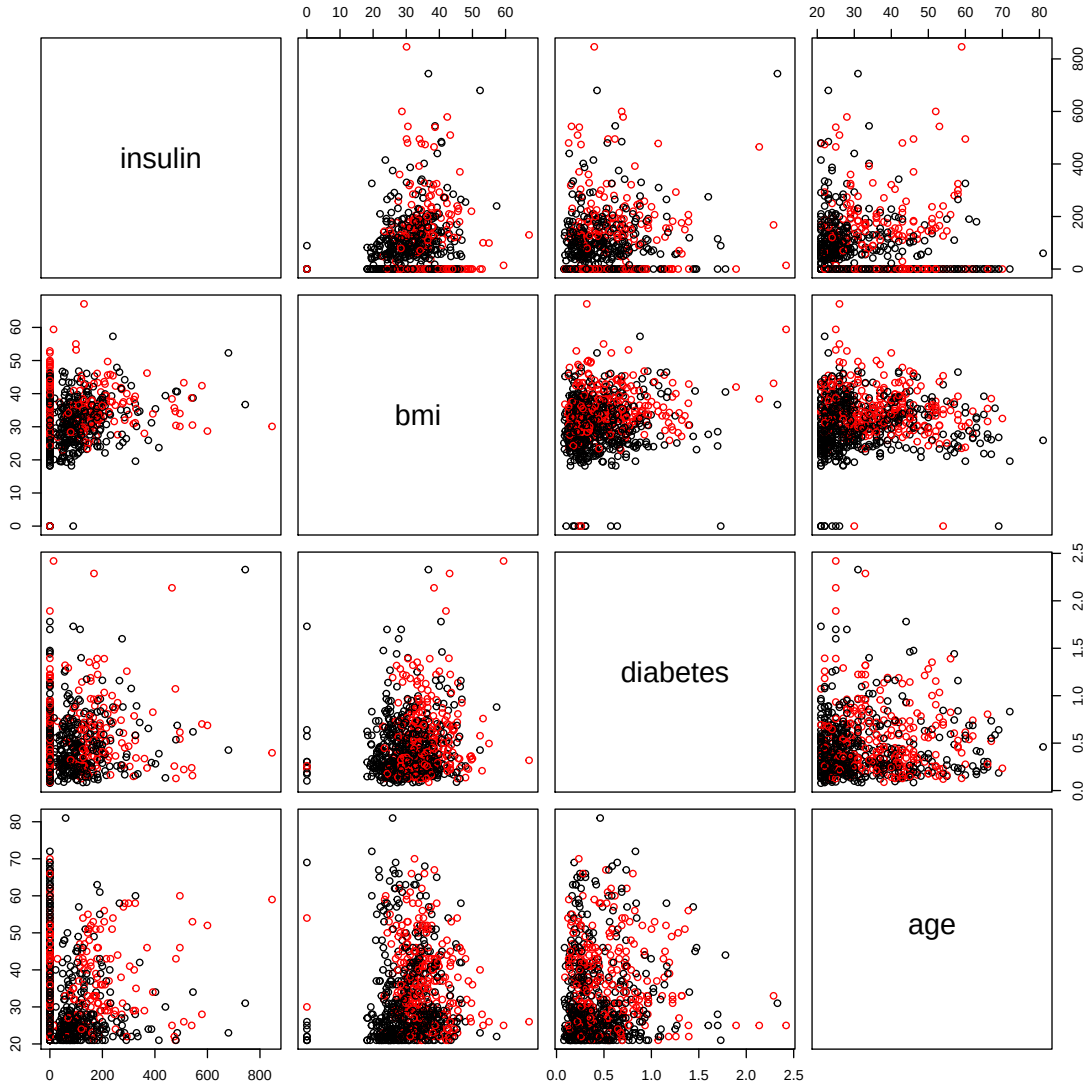
Introductory example

- Data about diabetes in the population of Pima Indians leaving near Phoenix, Arizona, USA.
- All 768 patients were females and at least 21 years old.
- Variables:
  1. Number of times pregnant
  2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
  3. Diastolic blood pressure (mm Hg)
  4. Triceps skin fold thickness (mm)
  5. 2-Hour serum insulin (mu U/ml)
  6. Body mass index (weight in kg/(height in m)^2)
  7. Diabetes pedigree function
  8. Age (years)
  9. Positive (1) or negative (0) of diabetes
- Problem: predict the onset of diabetes for the 8 predictors.

Introductory example (continued)



Introductory example (continued)

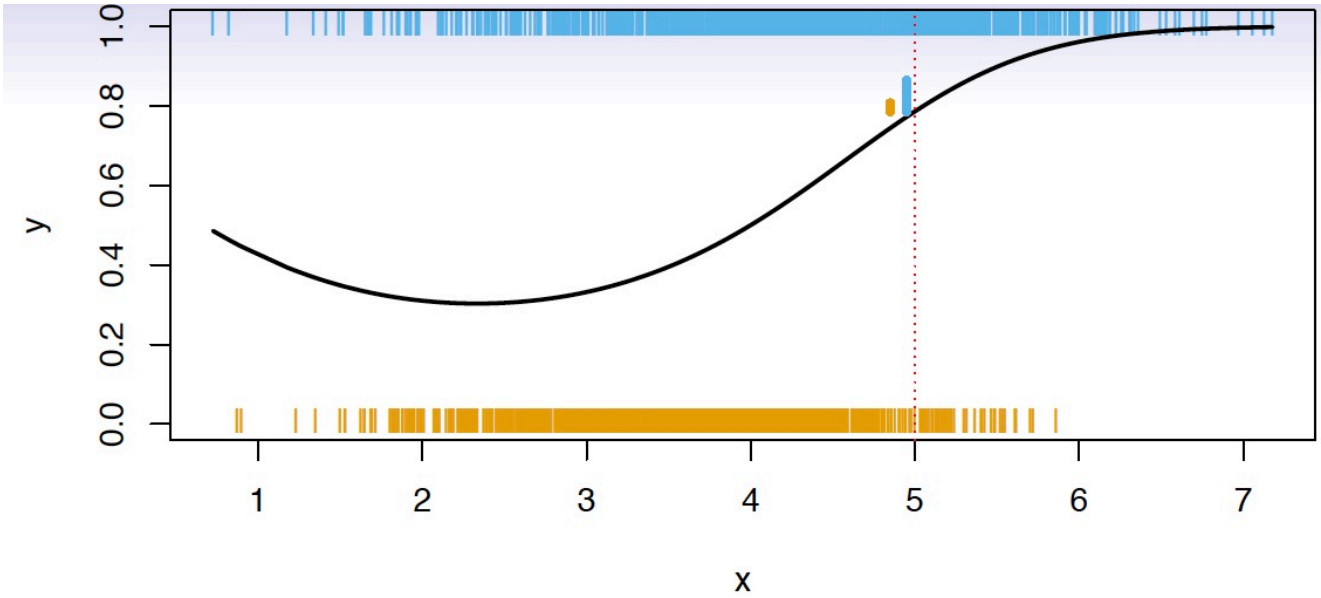


Classification problems

- In classifcation problems, the response variable Y is qualitative – e.g., diagnosis of diabetes is one in $\mathcal{C} = \{\text{positive}, \text{negative}\}$, email is one of $\mathcal{C} = \{\text{spam}, \text{email}\}$, digit class is one of $\mathcal{C} = \{0, 1, \ldots, 9\}$, etc.
- Our goals are to:
  - Build a classifier $C(X)$ that assigns a class label from $\mathcal{C}$ to a future unlabeled observation X.
  - Assess the uncertainty in each classification
  - Understand the roles of the different predictors among $X = (X_1, X_2, \ldots, X_p)$.
- Is there an ideal C(X)?

Overview

Conditional class probabilities



- Suppose the c elements in $\mathcal{C}$ are numbered $1, 2, \ldots, c$. Let

$$P_k(x) = \mathbb{P}(Y = k \mid X = x), \quad k = 1, 2, \ldots, c.$$

- These are the conditional class probabilities at x; e.g. see little barplot at $x = 5$.

The Bayes classifer

- The conditional error probability for classifier $C(X)$ is

$$\begin{aligned}\mathbb{P}(\text{error} \mid X = x) &= \mathbb{P}(C(X) \neq Y \mid X = x) \\ &= 1 - \mathbb{P}(C(X) = Y \mid X = x)\end{aligned}$$

- If $C(X) = k$, then

$$\mathbb{P}(\text{error} \mid X = x) = 1 - \mathbb{P}(Y = k \mid X = x) = 1 - P_k(x)$$

- To minimize $\mathbb{P}(\text{error} \mid X = x)$, we must choose k such that $P_k(x)$ is maximum.
- The corresponding classifier $C^*(x)$ is called the Bayes classifier. It has the lowest error probability.

Bayes error rate

- For $X = x$, the Bayes classifier predicts class k^* such that $P_{k^*}(x) = \max_k P_k(x)$, and the conditional error probability

$$1 - P_{k^*}(x) = 1 - \max_k P_k(x)$$

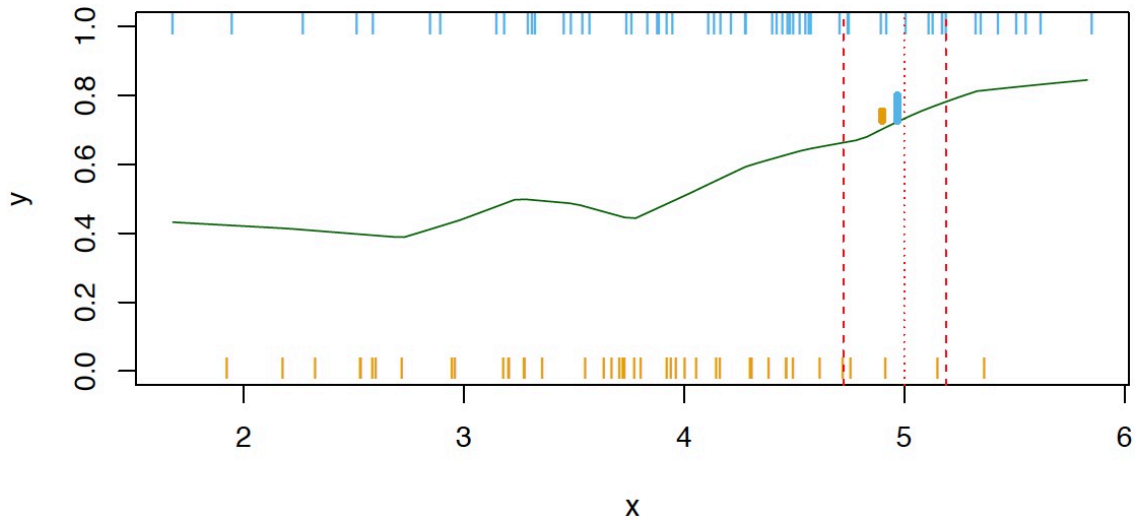
- The error probability of the Bayes classifier (averaged over all values of X) is

$$\text{Err}_B = \mathbb{E}\left[1 - \max_k P_k(X)\right] = \int \left[1 - \max_k P_k(x)\right] p_X(x) dx$$

- This probability is called the Bayes error rate. It is the lowest error probability that can be achieved by a classifier. It characterizes the difficulty of the classification task.

Overview

Voting K-nearest-neighbor rule



- Nearest-neighbor averaging can be used as in regression: we can define

$$\widehat{P}_k(x) = \text{Ave}\{I(Y = k) \mid X \in \mathcal{N}_K(x)\}$$

where $\mathcal{N}_K(x)$ is the set of the K nearest neighbors of x in the training set.

- The corresponding classifier is called the voting K-nearest neighbor (K-NN) classifier. (It also breaks down as dimension grows. However, the impact on $C(x)$ is less than that on $\widehat{P}_k(x)$, $k = 1, \ldots, c$).

Properties of the k-nearest-neighbor rule

- Let Err_1 be the asymptotic (as $N \to +\infty$) error probability of the 1-NN rule. It can be shown that it is at most equal to twice the Bayes error rate:

$$\text{Err}_B \leq \text{Err}_1 \leq 2\,\text{Err}_B.$$

- We can say that half of the information provided by the training set is contained in the nearest neighbor (asymptotically).

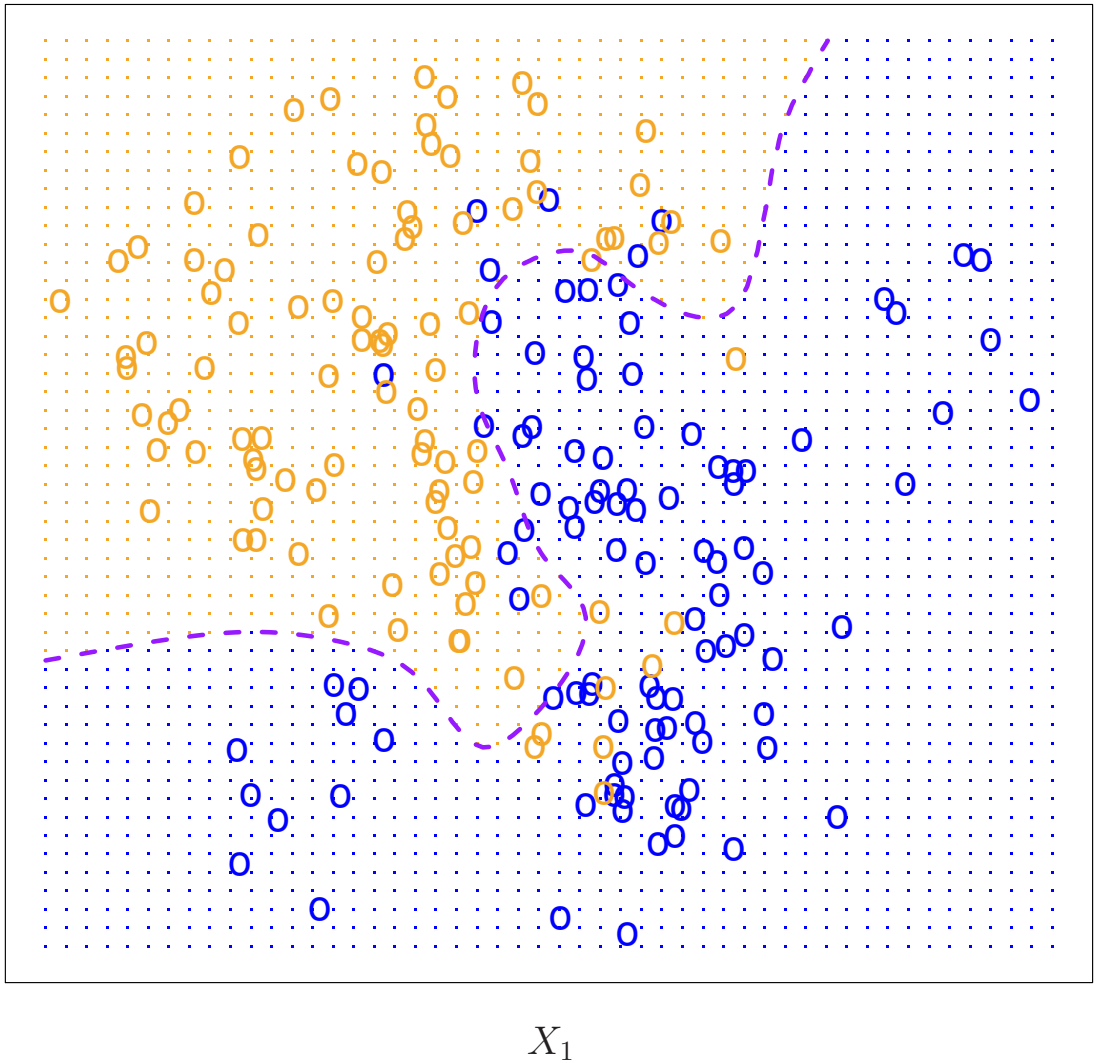
Error rate estimation

- Typically, we measure the performance of a classifier $C(x)$ using the test misclassification error rate:

$$\text{Err}_{\text{Te}} = \text{Ave}_{i \in \text{Te}}\, I[y_i \neq C(x_i)],$$

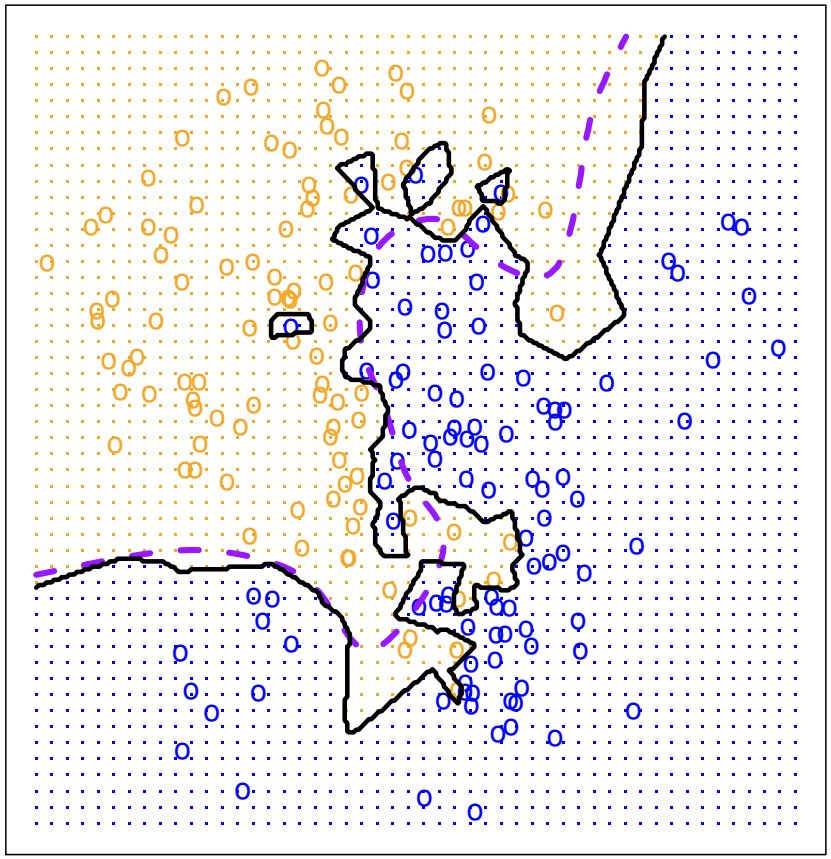
  where Te is a test dataset.
- The test error rate allows us to select the best model in a set of candidate models (more on this later).

Example: simulated data and Bayes decision boundary

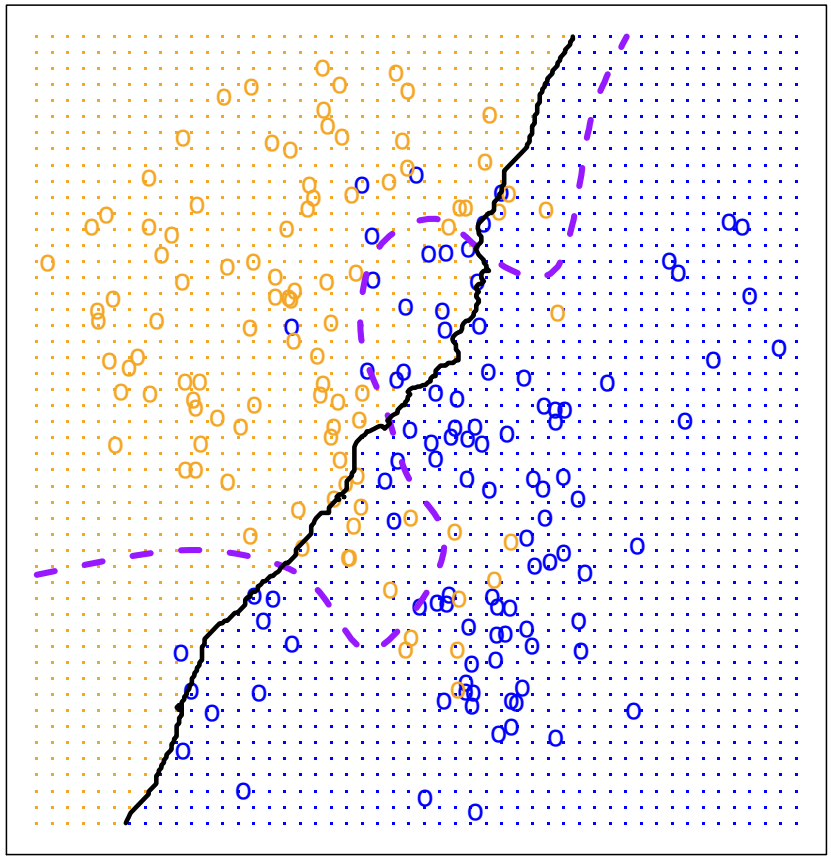


Decision boundaries for $K = 1$ and $K = 100$

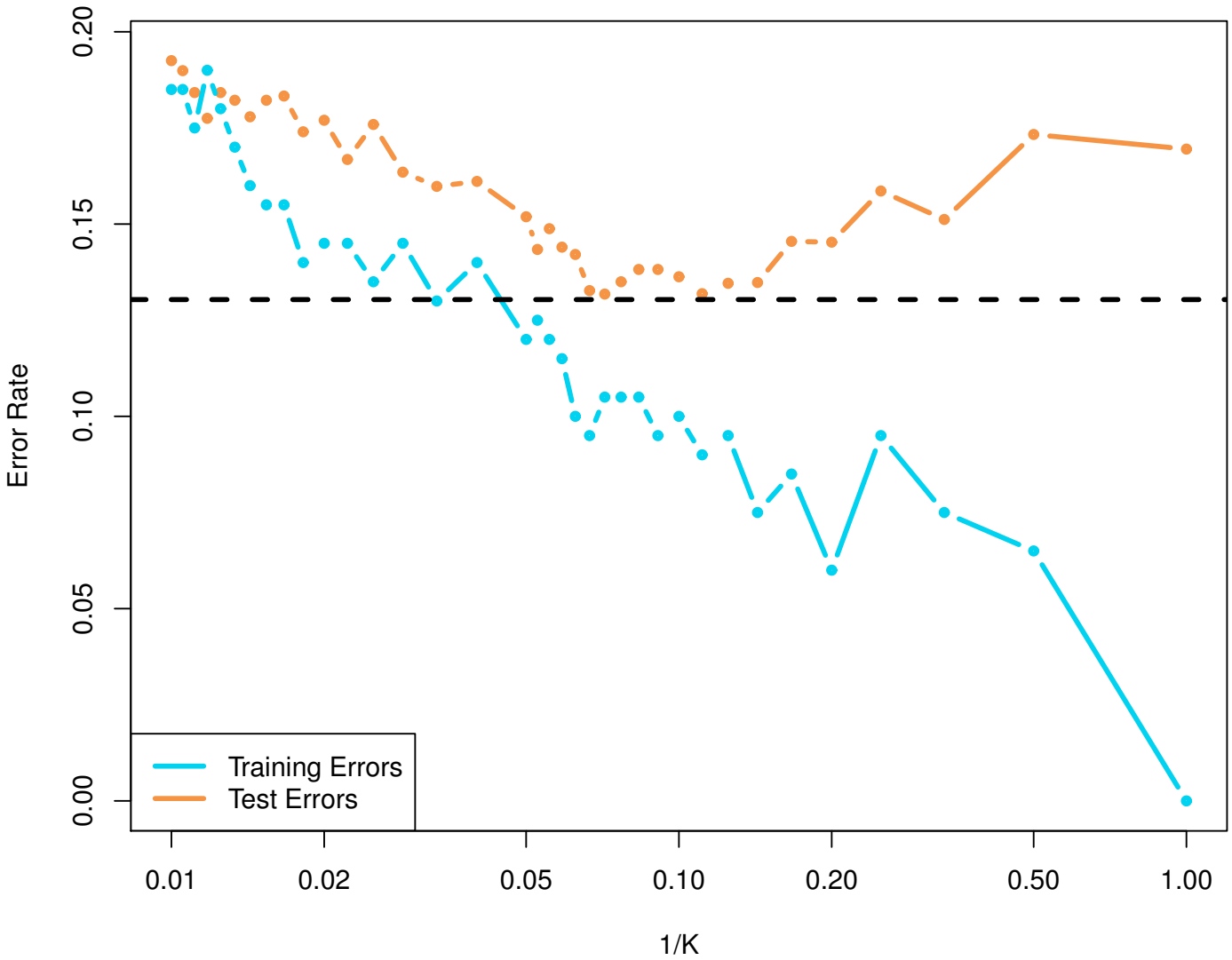
KNN: K=1



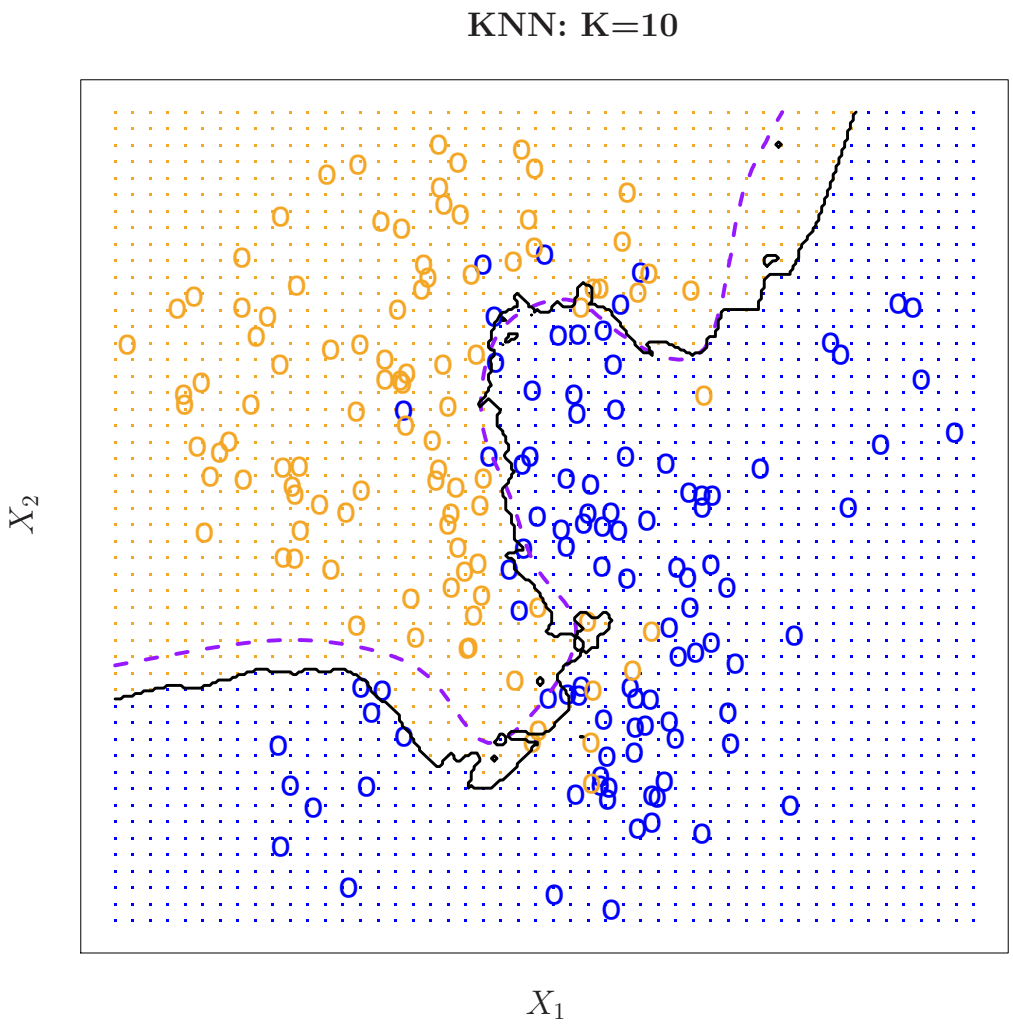
KNN: K=100



Training and test error rates vs. $1/K$



Decision boundary for the best value of K

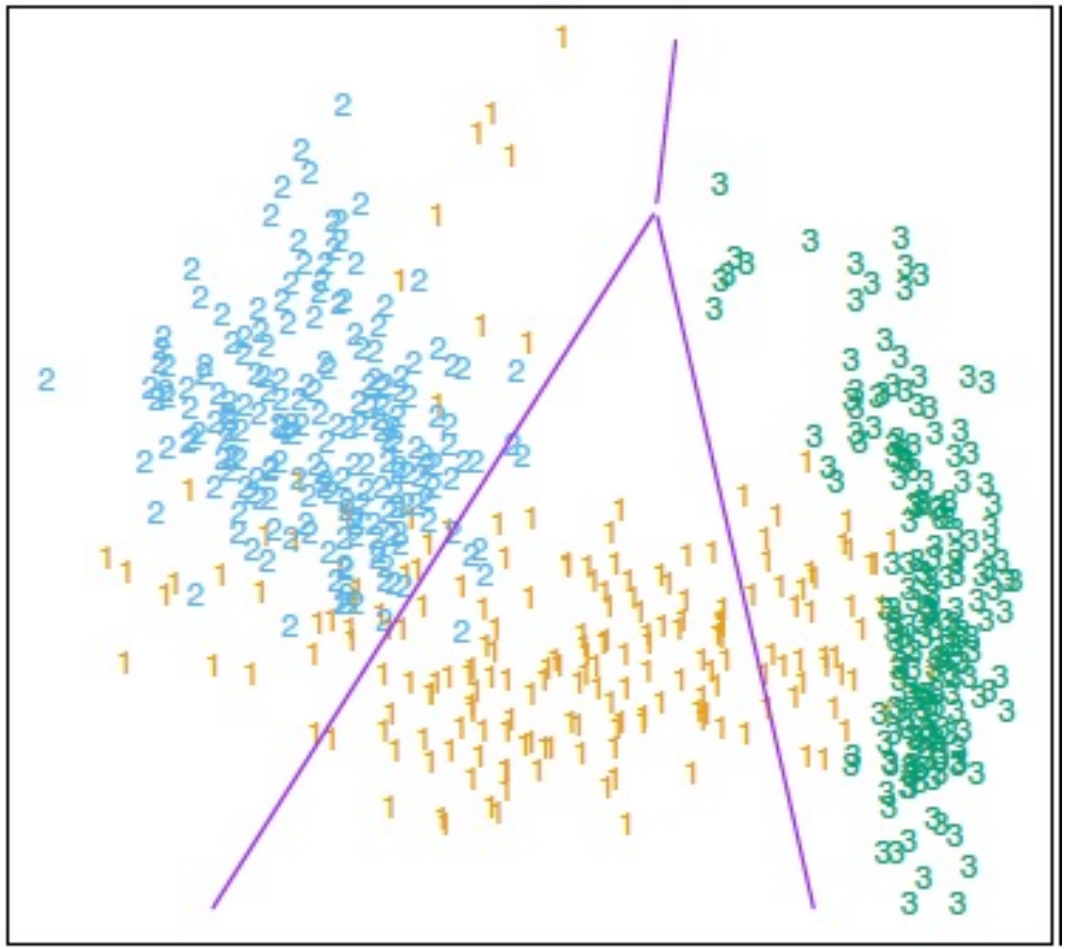


Overview

Linear classification

- In this chapter we focus on linear methods for classification.
- Since our classifier $C(x)$ takes values in a discrete set $\mathcal{C}$, we can always divide the input space into a collection of decision regions labeled according to the classification.
- As we have just seen, the boundaries of these regions can be rough or smooth, depending on the prediction function.
- For an important class of procedures, these decision boundaries are linear; this is what we will mean by linear methods for classification.

Example



Generative vs. discriminative models

- To approximate Bayes' rule, we need to estimate the conditional probabilities $P_k(x)$.
- We can distinguish two kinds of models for classification:

  Generative models describe the joint distribution of the inputs X and the label Y. Using Bayes' theorem, we then get the conditional class probabilities $P_k(x)$, and we pick the most likely class label k.

  Discriminative models represent the conditional probabilities $P_k(x)$ directly, or a direct map from inputs x to $\mathcal{C}$.
- In this chapter, we will focus on two classifiers (and some variants):
  1. A linear classifier based on a generative model: Linear Discriminant Analysis (LDA).
  2. A linear classifier based on a generative model: Logistic regression.

Overview

Overview

Bayes' theorem

- We have seen that the Bayes rule with minimum classification error selects the class with the highest conditional probability.
- Consequently, we need to know the conditional class probabilities $P_k(x)$, $k = 1, \ldots, c$, for optimal classification.
- Suppose $p_k(x)$ is the class-conditional density of X in class $Y = k$, and let $\pi_k = \mathbb{P}(Y = k)$ be the prior probability of class k, with $\sum_{k=1}^{K} \pi_k = 1$.
- A simple application of Bayes' theorem gives us

$$P_k(x) = \frac{p_k(x)\pi_k}{\sum_{\ell=1}^{c} p_\ell(x)\pi_\ell}, \quad k = 1, \ldots, c$$

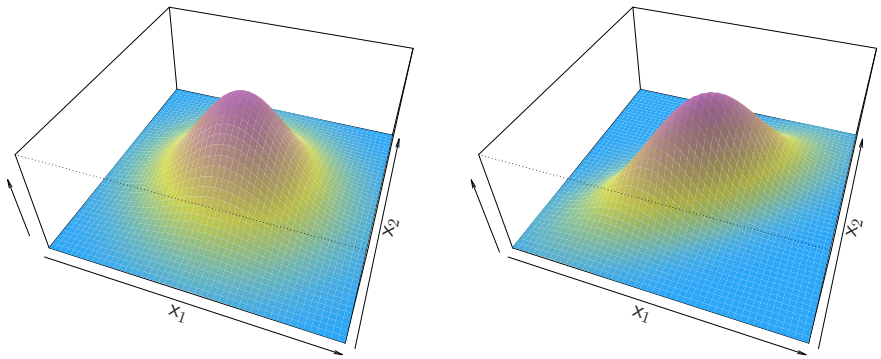
- Linear and quadratic discriminant analysis use multivariate normal densities for $p_k(x)$.

Model

- Suppose that we model each class density as a multivariate Gaussian

$$p_k(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}_k|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T\mathbf{\Sigma}_k^{-1}(x-\mu_k)\right\},$$

where $\mu_k = \mathbb{E}(X \mid Y = k)$ and $\mathbf{\Sigma}_k = \text{Var}(X \mid Y = k)$.



- Linear discriminant analysis (LDA) arises in the special case when we assume that the classes have a common covariance matrix

$$\mathbf{\Sigma}_k = \mathbf{\Sigma}, \quad \text{for all } k.$$

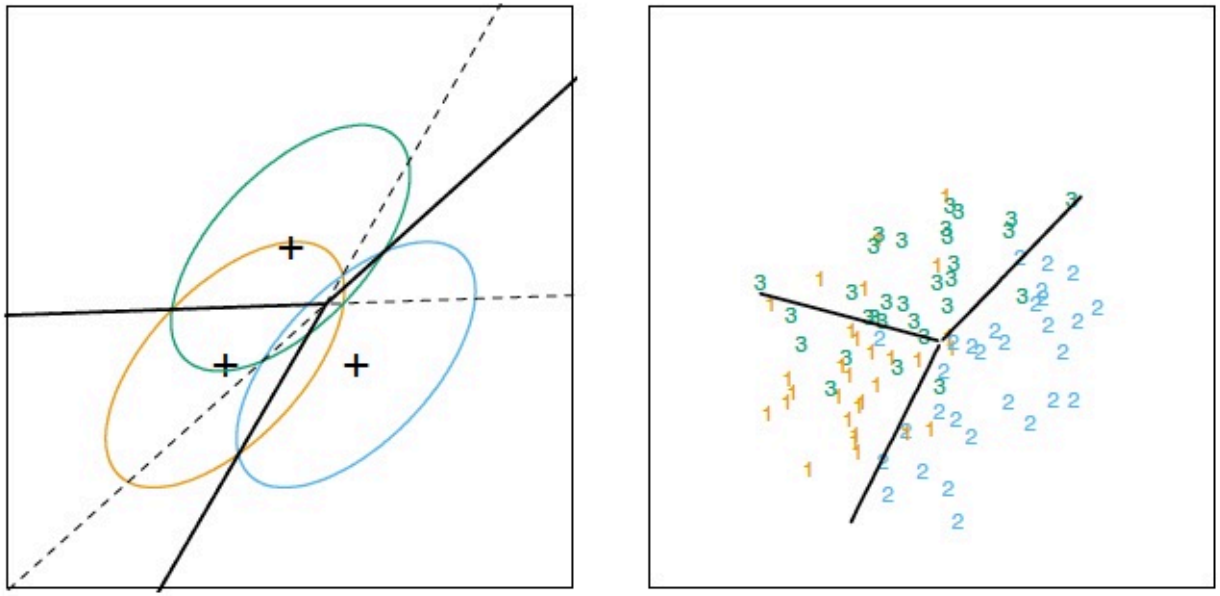
Linear boundaries

- In comparing two classes k and ℓ, it is sufficient to look at the log-ratio, and we see that

$$\begin{aligned}\log \frac{P_k(x)}{P_\ell(x)} &= \log \frac{p_k(x)}{p_\ell(x)} + \log \frac{\pi_k}{\pi_\ell} \\ &= \log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k + \mu_\ell)^T \mathbf{\Sigma}^{-1}(\mu_k - \mu_\ell) \\ &\quad + x^T \mathbf{\Sigma}^{-1}(\mu_k - \mu_\ell),\end{aligned}$$

  an equation linear in x.
- This linear log-odds function implies that the decision boundary between classes k and ℓ (the set of points such that both conditional probabilities are equal) is linear in x; in p dimensions it is a hyperplane.
- This is true for any pair of classes, so all the decision boundaries are linear.

Example



Left: contours of constant density enclosing 95% of the probability in each case. The Bayes decision boundaries between each pair of classes are shown (broken straight lines), and the Bayes decision boundaries separating all three classes are the thicker solid lines. Right: a sample of 30 vectors drawn from each distribution, and the fitted LDA decision boundaries.

Overview

Likelihood function

- The model parameters are π_k, μ_k $(k = 1, \ldots, c)$ and the common covariance matrix $\mathbf{\Sigma}$. Let $\boldsymbol{\theta}$ be the vector of all parameters.
- The likelihood function is

$$
\begin{aligned}
L(\boldsymbol{\theta}) &= \prod_{i=1}^{n} p(x_i, y_i) = \prod_{i=1}^{N} p(x_i \mid y_i) p(y_i) \\
&= \prod_{i=1}^{n} \prod_{k=1}^{c} p_k(x_i)^{y_{ik}} \pi_k^{y_{ik}} \\
&= \prod_{i=1}^{n} \prod_{k=1}^{c} \phi(x_i; \mu_k, \mathbf{\Sigma})^{y_{ik}} \pi_k^{y_{ik}}
\end{aligned}
$$

where $y_{ik} = I(y_i = k)$ and $\phi(x; \mu_k, \mathbf{\Sigma})$ is the normal density with mean μ_k and variance Σ.

Maximum likelihood estimates

- The maximum likelihood estimates are

$$\widehat{\pi}_k = \frac{n_k}{n}, \quad \widehat{\mu}_k = \frac{1}{n_k} \sum_{i=1}^{n} y_{ik} x_i, \quad \text{and} \quad \widehat{\mathbf{\Sigma}} = \frac{1}{n} \sum_{k=1}^{c} n_k \widehat{\mathbf{\Sigma}}_k$$

  where $\widehat{\mathbf{\Sigma}}_k$ is the empirical variance matrix in class k:

$$\widehat{\mathbf{\Sigma}}_k = \frac{1}{n_k} \sum_{i=1}^{n} y_{ik} (x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^T,$$

  with $y_{ik} = I(y_i = k)$ and $n_k = \sum_{i=1}^{n} y_{ik}$.

- It can be shown that $\widehat{\mathbf{\Sigma}}$ is biased. An unbiased estimator of $\mathbf{\Sigma}$ is

$$\mathbf{S} = \frac{n}{n-c} \widehat{\mathbf{\Sigma}}.$$

Example: Letter recognition dataset

- Source: P. W. Frey and D. J. Slate, *Machine Learning*, Vol 6 #2, March 91.
- *"The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli."*

LDA in R

```
letter <- read.table("letter-recognition.data",header=FALSE)
n<-nrow(letter)

library(MASS)

napp=15000
ntst=n-napp
train<-sample(1:n,napp)
letter.test<-letter[-train,]
letter.train<-letter[train,]

lda.letter<- lda(V1~.,data=letter.train)

pred.letters<-predict(lda.letter,newdata=letter.test)

perf <-table(letter.test$V1,pred.letters$class)
1-sum(diag(perf))/ntst
```

Confusion matrix and test error rate

Console ~/Documents/R/Scripts/teaching/sy19/

```
> table(letter.test$V1,pred.letters$class)

    A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y   Z
A 155   0   0   0   0   0   0   2   0   1   1   0   2   0   2   0   0   0   6   0   0   0   2   0   4   0
B   0 134   0   9   0   0   3   4   0   0   1   0   1   0   3   0   0  21  14   0   0   0   0   5   0   1
C   0   0 151   0   9   2  16   2   0   0   9   0   0   0   1   0   0   0   3   2   1   1   1   0   0   0
D   0  10   0 158   0   0   1   2   1   4   0   0   4   3   8   0   0   0   4   0   0   0   0  10   0   0
E   0  12  19   1  90   1  23   0   2   0   5   0   0   0   0   0   2   6   5   2   2   0   0  23   0   6
F   0  11   0   3   0 138   2   0   1   0   0   0   0   1   0  26   7   1   2   7   1   0   1   5   1   0
G   1  10  25   1   6   0  95   5   0   0  10   1   1   1   3   0  14   7  12   0   0   0   2   2   0   0
H   0   5   0   7   0   2   2  82   0   0  13   0   1  13  18   2   6   6   0   0   6   0   1  13   0   0
I   0   3   0   2   1   3   1   0 153   1   0   0   0   0   0   1   4   1   6   0   0   0   0   2   3   0
J   1   1   0   3   0  10   0   1  28 128   0   0   0   0   4   4   3   0  20   0   0   0   0   7   0   0
K   0   3   1   3   4   0   4   1   0   0 124   0   2   0   1   0   0  19   0   0   1   2   0  10   0   0
L   3   6   1   0   6   0  10   0   4   4   3 143   0   0   0   0   1   1   3   0   0   0   0   3   0   0
M   1   0   0   0   0   0   0   5   0   2   1   0 189   6   3   0   0   3   0   0   0   0   3   0   0   0
N   0   0   0   5   0   0   0   8   0   0   3   0   1 158   3   0   0   1   0   0   1   0   6   2   0   0
O   5   0   0  18   0   0   5  21   0   0   2   0   0   3 116   0   1   0   0   0   0   0   0   5   1   0   0
P   0   5   0   2   0  18   1   0   0   0   2   0   0   1   1 165   8   0   0   0   0   1   4   0   8   0
Q   2  11   0   1   1   0  14   2   0   0   1   2   1   0  19   0 123   1   6   0   0   1   3   5   1   0
R   0  16   0  11   0   0   1   7   0   0   7   0   1   1   0   0   0 136   0   0   0   0   0   5   0   0
S   8  15   0   2   1   5   8   1   2   1   0   3   0   0   1   0   1   8  72   0   0   0   0  15   3  29
T   0   1   0   1   2  16   4   2   1   0   2   0   0   0   1   1   0   1   4 142   0   2   0   1   1   6
U   0   0   1   0   0   0   0  14   0   0   3   0  11   7  11   0   0   0   0   0 158   0   3   0   0   0
V   0   1   0   0   0   0   2   4   0   0   2   0   1   0   3   1   0   1   0   2   1 156   5   0   2   0
W   0   0   0   0   0   0   0   9   0   0   1   0   4   4   0   0   0   0   0   0   0   1 175   0   0   0
X   0   6   0   2   0   0   2   0   0   0   5   0   0   0   1   0   9   3  18   0   5   0   0 143   3   1
Y   0   0   0   1   0  18   0   0   0   0   0   0   0   0   1   0  24   0   6   8   3  47   0   0 101   0
Z   0   0   0   0  14   2   2   0   0   6   0   0   0   0   0   0   4   1  21   0   0   0   0   2   0 115
> 1-sum(diag(perf))/ntst
[1] 0.3
>
```

Overview

Case $c = 2$: fixing the threshold

- In the case of $c = 2$ classes, LDA assigns x to class 2 if

$$x^T \widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_2 - \widehat{\mu}_1) > s,$$

  where the threshold s depends on the estimated prior probabilities $\widehat{\pi}_1$ and $\widehat{\pi}_2$.
- If the prior probabilities cannot be estimated, or if the model assumption are not verified, a different threshold may give better result.
- The Receiver Operating Characteristic (ROC) curve describes the performance of the classifier for any value of s.

Confusion matrix ($c = 2$)

- Assuming $c = 2$, call one class "positive" and the other one "negative".
- For a given threshold s, we get a confusion matrix such as

	predicted	
true	P	N
P	true positive (TP)	false negative (FN)
N	false positive (FP)	true negative (TN)

- The true positive rate (sensitivity) and false positive rate (1-specificity) are defined, respectively, as

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}$$

- If we decrease s, we increase both the TPR and the FPR.
- The ROC curve is a plot of the TPR as a function of the FPR, for different values of s.

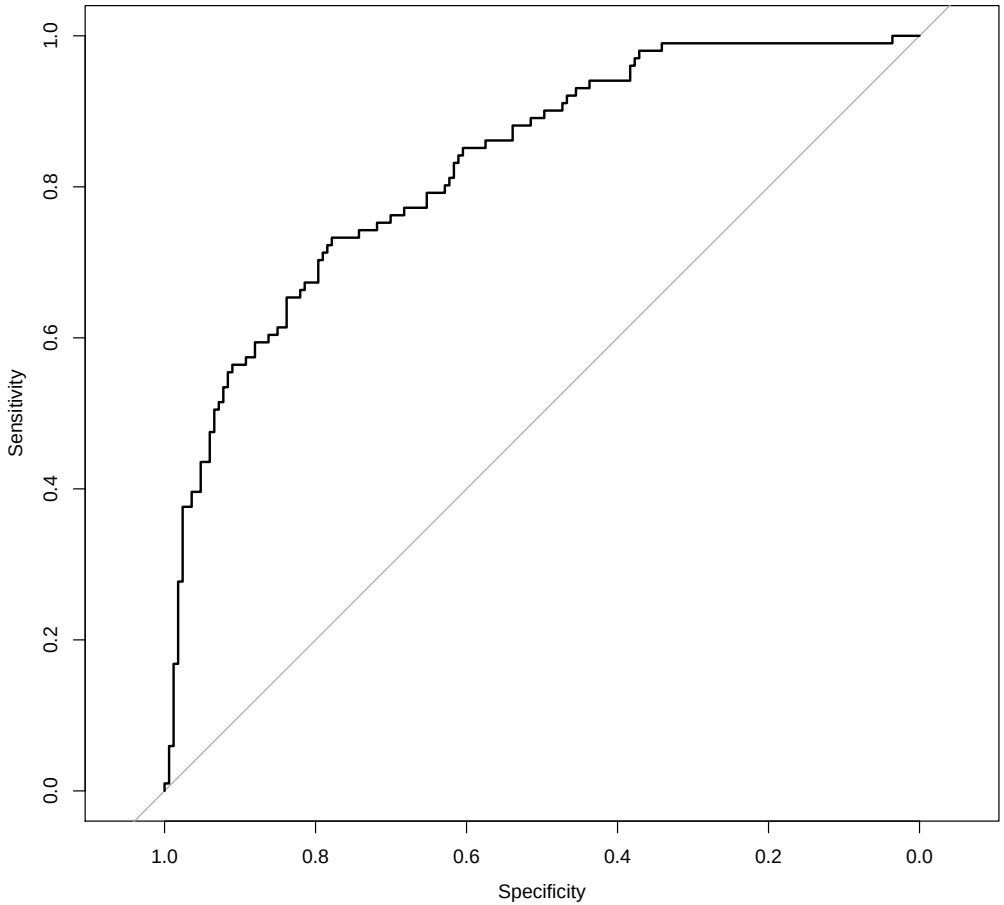
LDA in R

```
pima<-read.csv('pima-indians-diabetes.data',header=FALSE)
names(pima)<-c("pregnant","glucose","BP","skin","insulin","bmi","diabetes",
"age","class")
n<-nrow(pima)
napp=500
ntst=n-napp
train<-sample(1:n,napp)
pima.test<-pima[-train,]
pima.train<-pima[train,]
lda.pima<- lda(class~.,data=pima.train)
pred.pima<-predict(lda.pima,newdata=pima.test)
table(pima.test$class,pred.pima$class)
> perf

    0   1
0 152  15
1  45  56
```

Here, the TPR is 56/(45+56)=0.55, and the FPR is 15/(152+15)=0.089.
The error rate is $(15 + 45)/268 \approx 0.22$.

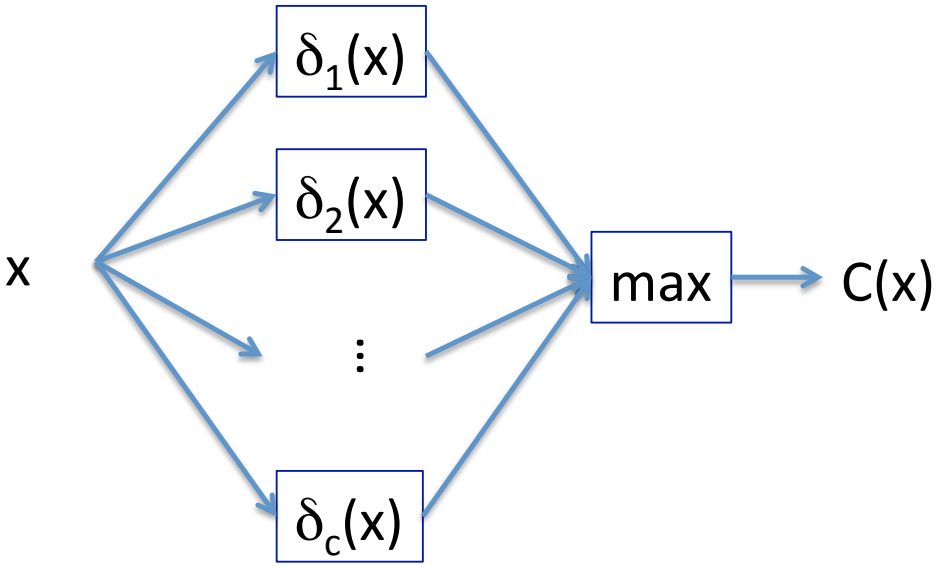
LDA in R

```
library(pROC)
roc_curve<-roc(pima.test$V9,as.vector(pred.pima$x))
plot(roc_curve)
```



Overview

Discriminant functions



Discriminant functions are functions $\delta_k(x)$ such that classifier C(x) can be written as

$$C(x) = \arg\max_k \delta_k(x).$$

- They can be obtained by computing the log of $P_k(x)$, and dropping the terms that are constant for all classes. We get

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

- The LDA classifier can thus be implemented using linear discriminant functions.

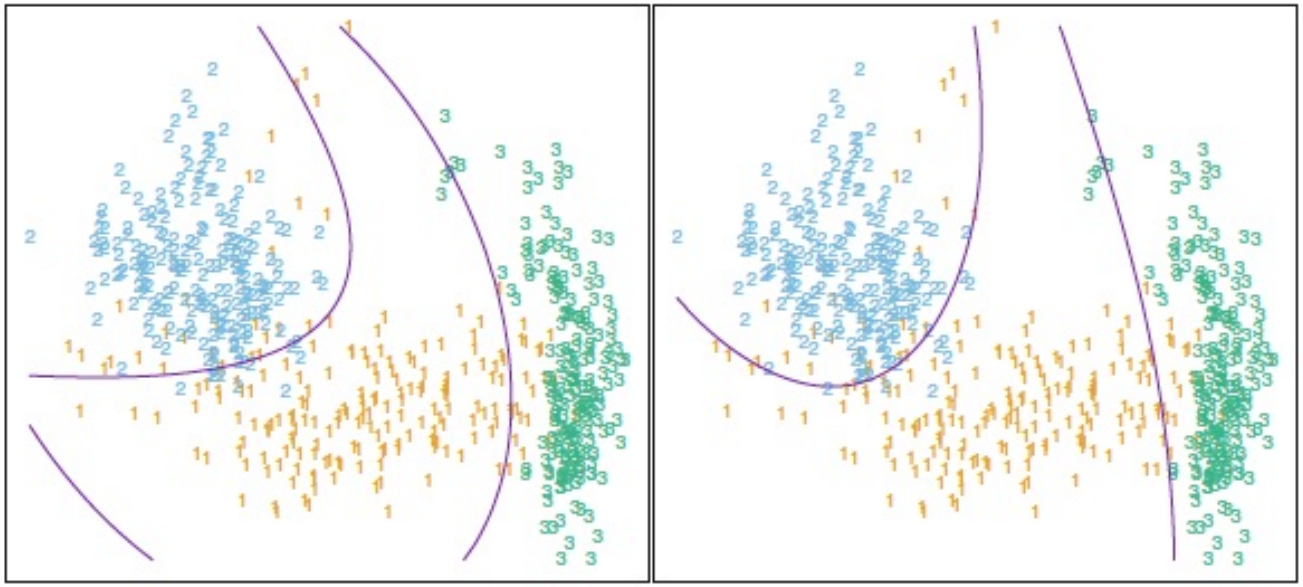
Quadratic Discriminant Analysis (QDA)

- If the $\mathbf{\Sigma}_k$ are not assumed to be equal, then the quadratic terms do not cancel out, and we get quadratic discriminant functions:

$$\begin{aligned}\delta_k(x) &= \log p_k(x) + \log \pi_k \\ &= -\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\mathbf{\Sigma}_k| + \log \pi_k\end{aligned}$$

- To apply this rule, we plug-in the ML estimates $\widehat{\mu}_k$, $\widehat{\pi}_k$ and $\widehat{\mathbf{\Sigma}}_k$.

Example



Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries obtained using LDA in the five-dimensional space X_1, X_2, X_1X_2, X_1^2, X_2^2. The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.

Application of QDA to the letter recognition data

```
qda.letter<- qda(V1~.,data=letter.train)
pred.letters<-predict(qda.letter,newdata=letter.test)

perf <-table(letter.test$V1,pred.letters$class)
1-sum(diag(perf))/ntst

0.1218
```

Naive Bayes classifiers

- Starting from the QDA model, we get a simpler model by assuming that the covariance matrices Σ_k are diagonal:

$$\mathbf{\Sigma}_k = \text{diag}(\sigma_{k1}^2, \ldots, \sigma_{kp}^2),$$

  where $\sigma_{kj}^2 = \text{Var}(X_j \mid Y = k)$.
- This assumption means that the predictors are conditionally independent given the class variable Y, i.e., for all $k \in \{1, \ldots, c\}$,

$$p_k(x_1, \ldots, x_p) = \prod_{j=1}^{p} p_{kj}(x_j)$$

- Remark: conditional independence does not imply independence. (Example: Height and vocabulary of kids are not independent; but they are conditionally independent given age).

Naive Bayes classifiers (continued)

- To estimate $\mathbf{\Sigma}_k$ under the conditional independence assumption, we simply set the off-diagonal in $\widehat{\mathbf{\Sigma}}_k$ to 0. The variance σ_{kj}^2 of X_j conditionally on $Y = k$ is estimated by

$$\widehat{\sigma}_{kj}^2 = \frac{1}{n_k} \sum_{i=1}^{n} y_{ik}(x_{ij} - \widehat{\mu}_{kj})^2.$$

- A further simplification is achieved by assuming that the covariance matrices are diagonal and equal:

$$\mathbf{\Sigma}_1 = \cdots = \mathbf{\Sigma}_c = \mathbf{\Sigma} = \text{diag}(\sigma_1^2, \ldots, \sigma_p^2).$$

- In spite of their simplicity, naive Bayes classifiers often (but not always) have very good performances.
- Can accommodate mixed feature vectors (qualitative and quantitative). If X_j is qualitative, replace $p_{kj}(x_j)$ with probability mass function (histogram) over discrete categories.

Naive Bayes classifier in R

```
library(naivebayes)
naive.letter<- naive_bayes(V1~.,data=letter.train)
pred.letters.naive<-predict(naive.letter,newdata=letter.test)

perf.naive <-table(letter.test$V1,pred.letters.naive)
1-sum(diag(perf.naive))/ntst

0.3474
```

Comparison of the different models

- QDA is the most general model. However, it does not always yield the best performances, because it has the biggest number of parameters.
- Although LDA also has a number of parameters proportion to p^2, it is usually much more stable than QDA. This method is recommended when n is small.
- Naive Bayes classifiers have a number of parameters proportional to p. They usually outperform other methods when p is very large.

Model	Number of parameters
QDA	$c\left(p + \frac{p(p+1)}{2}\right) + c - 1$
naive QDA	$2cp + c - 1$
LDA	$cp + \frac{p(p+1)}{2} + c - 1$
naive LDA	$cp + p + c - 1$

Example

- We consider $c = 2$ classes with $p = 3$ normally distributed input variables, with the following parameters

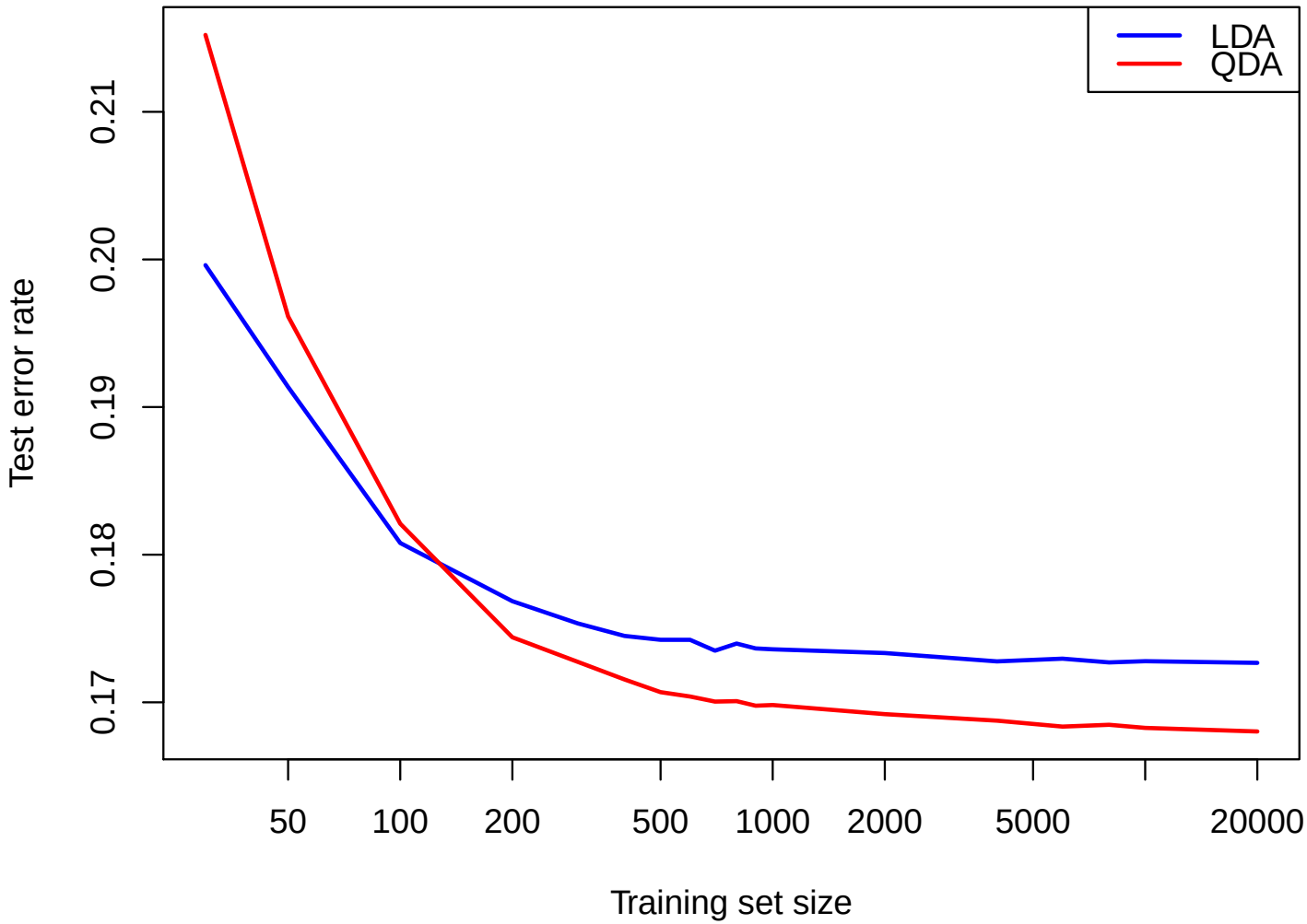
$$\pi_1 = \pi_2 = 0.5$$

$$\mu_1 = (0, 0, 0)^T, \quad \mu_2 = (1, 1, 1)^T$$

$$\Sigma_1 = I_3, \quad \Sigma_2 = 0.7 I_3.$$

- LDA and QDA classifiers were trained using training sets of different sizes between 30 and 20,000, and their error rate was estimated using a test set of size 20,000.
- For each training set size, the experiment was repeated 20 times. The next figure shows themean error rates over the 20 replications.

Result


LDA
QDA
Test error rate
0.17
0.18
0.19
0.20
0.21
50
100
200
500
1000
2000
5000
20000
Training set size

Overview

Overview

Binomial Logistic regression

- Consider a binary classification problem with $c = 2$ classes, $Y \in \{0, 1\}$. Let $P(x) = \mathbb{P}(Y = 1 \mid X = x)$ be the conditional probability of class $Y = 1$.
- We want to find a simple model for $P(x)$. An idea could be to use a linear model of the form

$$P(x) = \beta_0 + \beta^T x,$$

  with $\beta \in \mathbb{R}^p$ and $\beta_0 \in \mathbb{R}$, but this is not suitable because $\beta_0 + \beta^T x$ can take any value in $\mathbb{R}$, whereas $P(x) \in [0, 1]$.
- How to take into account the constraint $P(x) \in [0, 1]$?

Binomial Logistic regression

- A better model is to postulate that the log of the ratio between the probabilities of class 1 over class 0 is linear in x. This is the (Binomial) Logistic Regression (LR) model:

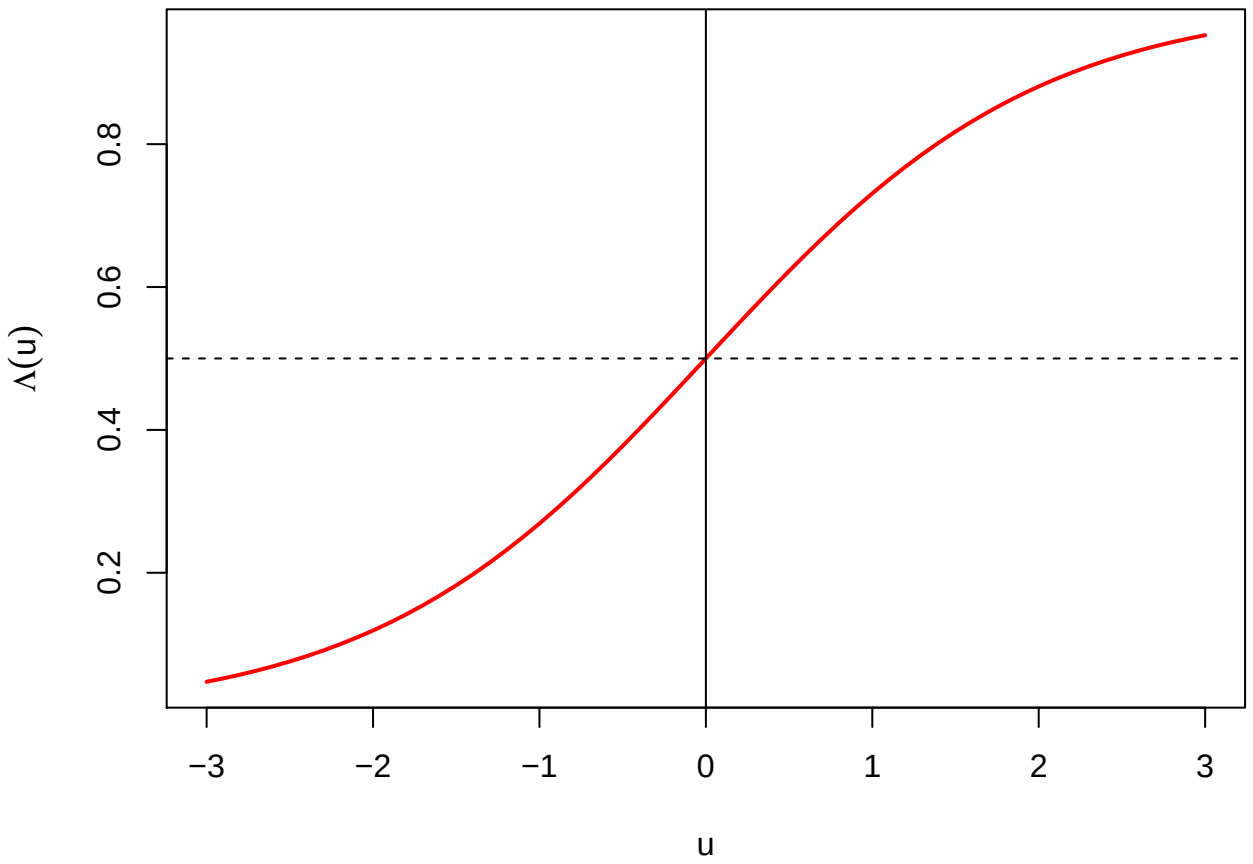
$$\log \frac{P(x)}{1 - P(x)} = \beta_0 + \beta^T x$$

- Equivalently, we can write

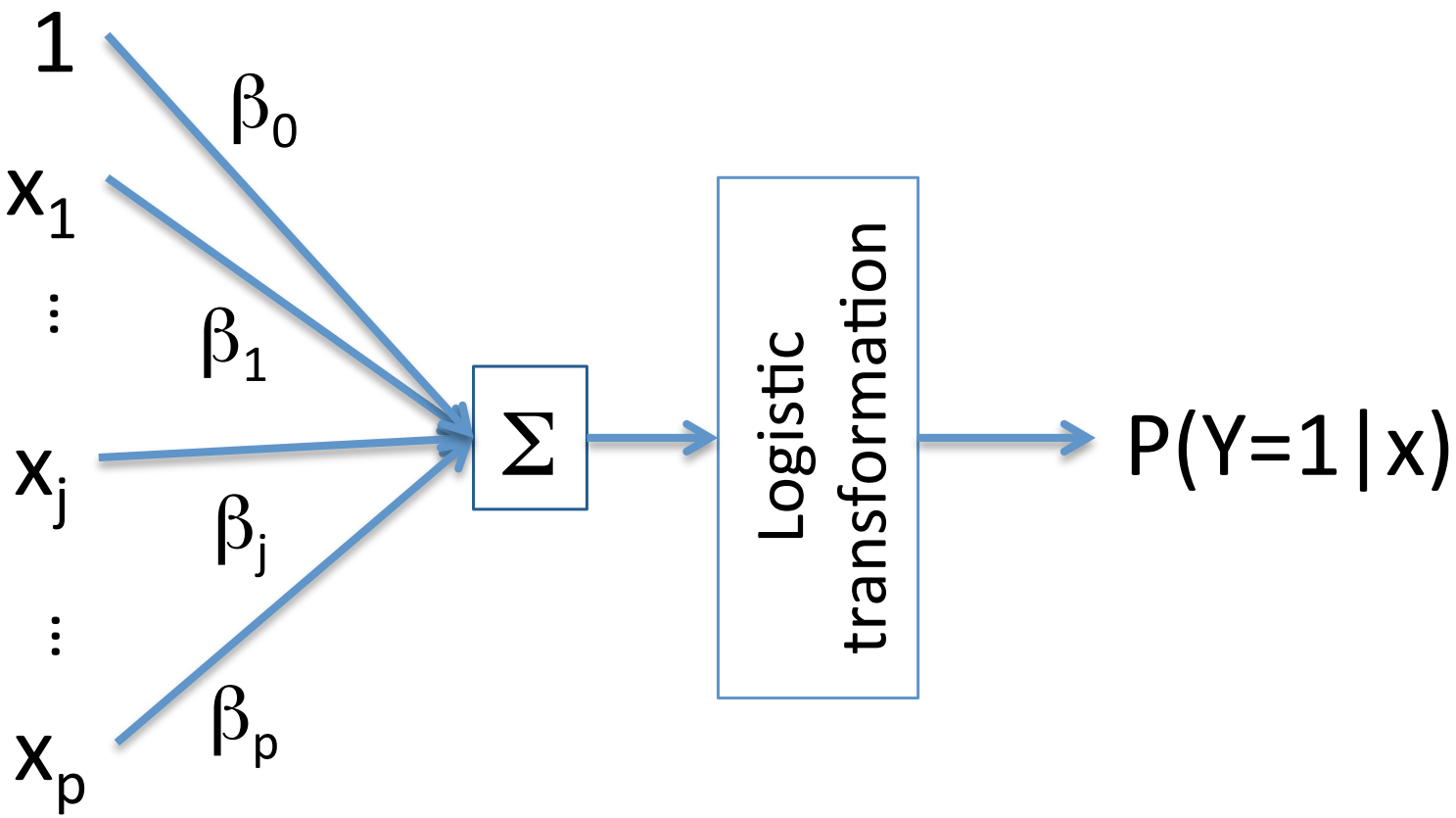
$$P(x) = \frac{1}{1 + \exp[-(\beta_0 + \beta^T x)]} = \Lambda(\beta_0 + \beta^T x),$$

where $\Lambda(u) = \frac{1}{1+\exp(-u)}$ is called the logistic function.

Plot of the logistic function



Graphical representation of binomial logistic regression



Overview

Conditional likelihood function

Logistic regression models are usually fit by maximizing the conditional likelihood, which is the likelihood function, assuming the x_i are fixed. It is defined as

$$L(\beta) = \prod_{i=1}^{n} \mathbb{P}(Y_i = y_i | X_i = x_i; \beta)$$
$$= \prod_{i=1}^{n} P(x_i; \beta)^{y_i} \left[1 - P(x_i; \beta)\right]^{1-y_i}$$

where $y_i \in \{0, 1\}$ and $P(x_i; \beta) = \mathbb{P}(Y = 1 | X = x_i; \beta)$.

Conditional log-likelihood function

The conditional log-likelihood is

$$\begin{aligned}\ell(\beta) &= \sum_{i=1}^{n} \{y_i \log P(x_i;\beta) + (1-y_i)\log(1-P(x_i;\beta))\} \\ &= \sum_{i=1}^{n} \left\{ y_i \beta^T x_i - \log(1+\exp(\beta^T x_i)) \right\},\end{aligned}$$

where we assume that the vector of inputs x_i includes the constant term 1 to accommodate the intercept.

Maximization

- To maximize the log-likelihood, we set its derivatives to zero.
- The gradient of the log-likelihood is

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{n} y_i x_i - \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)} x_i$$
$$= \sum_{i=1}^{N} x_i \left(y_i - P(x_i; \beta)\right) = \mathbf{X}^T(\mathbf{y} - \mathbf{P})$$

  where $\mathbf{y}$ denote the vector of y_i values, $\mathbf{X}$ the $n \times (p+1)$ matrix of x_i values, $\mathbf{P}$ the vector of fitted probabilities with ith element $P(x_i; \beta)$.
- The score equation $\frac{\partial \ell}{\partial \beta} = 0$ is non linear: we use the Newton-Raphson algorithm.

Hessian matrix and update equation

- From

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{n} x_i \left(y_i - P(x_i; \beta) \right),$$

the Hessian matrix is

$$\begin{aligned} \frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} &= -\sum_{i=1}^{n} x_i x_i^T P(x_i; \beta)(1 - P(x_i; \beta)) \\ &= -\mathbf{X}^T \mathbf{W} \mathbf{X}, \end{aligned}$$

where $\mathbf{W}$ an $n \times n$ diagonal matrix of weights with ith diagonal element $P(x_i; \beta)(1 - P(x_i; \beta))$.

- We get the following update equation:

$$\beta^{(t+1)} = \beta^{(t)} + (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{p})$$

Asymptotic distribution of $\widehat{\beta}$

- A central limit theorem then shows that the distribution of $\widehat{\beta}$ converges to

$$\mathcal{N}(\beta, (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1}).$$

  when $n \to +\infty$.
- This results makes it possible to compute confidence intervals and to test the significance of the coefficients β_j.

Binomial logistic regression in R

```
glm.fit<- glm(class~.,data=pima.train,family=binomial)
summary(glm.fit)
```

Console ~/Documents/R/Scripts/teaching/sy19/

```
> summary(glm.fit)

Call:
glm(formula = class ~ ., family = binomial, data = pima.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6283  -0.7258  -0.3775   0.7200   2.7248

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.169838   0.933412  -9.824  < 2e-16 ***
pregnant     0.092700   0.039180   2.366  0.01798 *
glucose      0.035910   0.004587   7.829 4.93e-15 ***
BP          -0.013326   0.006252  -2.132  0.03305 *
skin        -0.001035   0.008580  -0.121  0.90401
insulin     -0.001459   0.001146  -1.274  0.20274
bmi          0.103880   0.018931   5.487 4.08e-08 ***
diabetes     1.132297   0.368532   3.072  0.00212 **
age          0.024392   0.011426   2.135  0.03277 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 655.68  on 499  degrees of freedom
Residual deviance: 464.59  on 491  degrees of freedom
AIC: 482.59
```

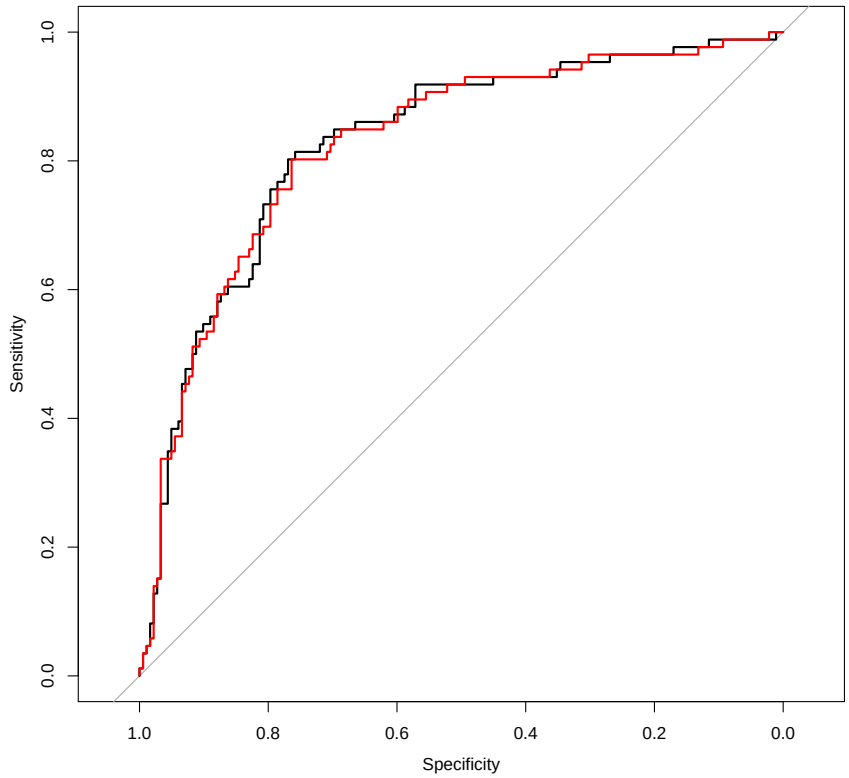
Prediction

```
pred.pima.glm<-predict(glm.fit,newdata=pima.test,type='response')

table(pima.test$class,pred.pima.glm>0.5)

     FALSE  TRUE
  0    160    22
  1     36    50
```

The error rate is $(22 + 36)/268 \approx 0.22$.

ROC curve: comparison with LDA

```
logit<-predict(glm.fit,newdata=pima.test,type='link')
roc_curve<-roc(pima.test$class,as.vector(pred.pima$x)) # LDA
plot(roc_curve)

roc_glm<-roc(pima.test$class,logit)
plot(roc_glm,add=TRUE,col='red')
```



Overview

Model

- Multinomial logistic regression extends binomial logistic regression to $c > 2$ by assuming the following model:

$$\log P_k(x) = \beta_k^T x + \beta_{k0} + \gamma, \quad k = 1, \ldots, c$$

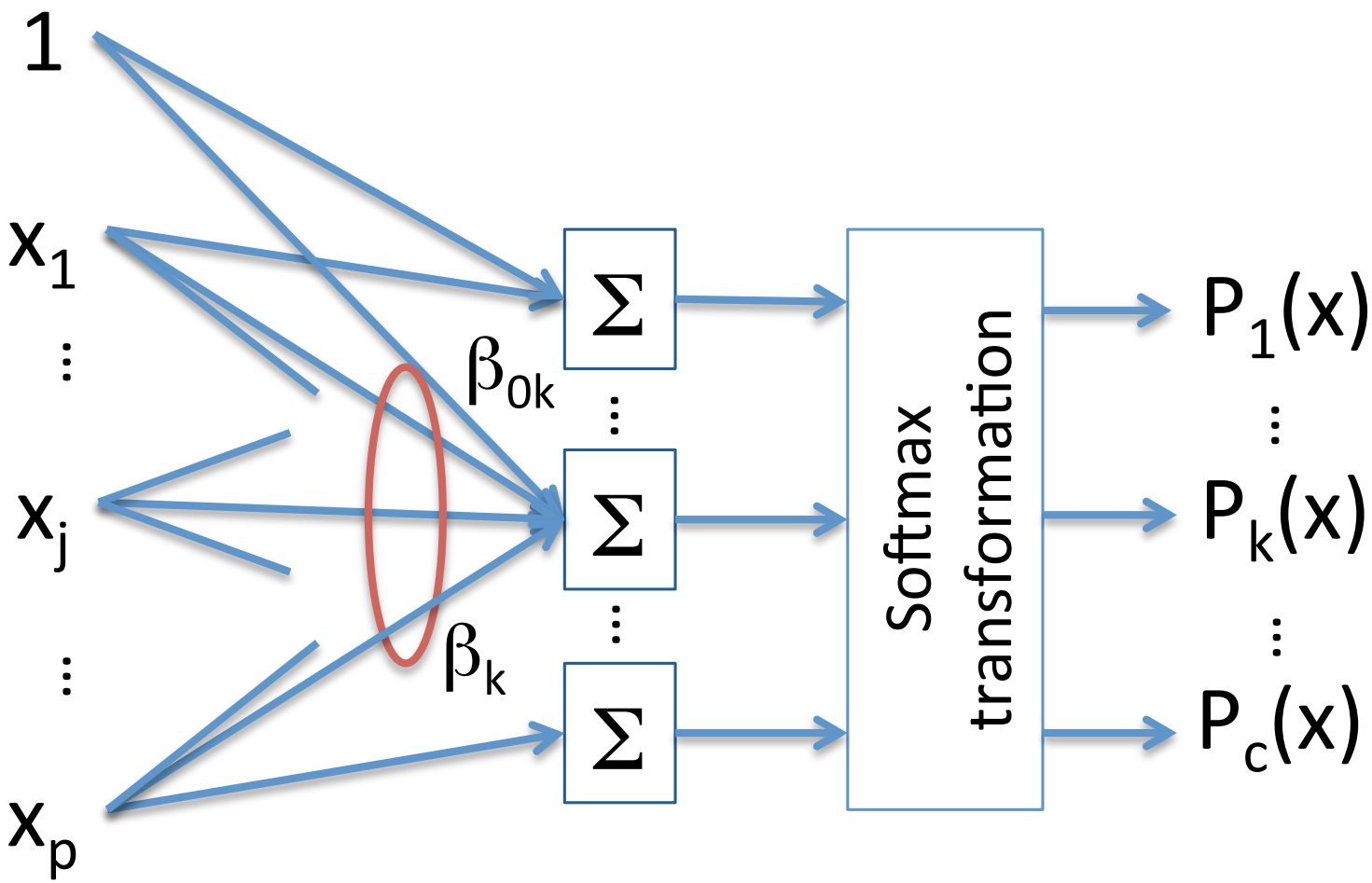
where $P_k(x) = \mathbb{P}(Y = k \mid X = x)$, $\beta_k \in \mathbb{R}^p$, $\beta_{k0} \in \mathbb{R}$ and $\gamma \in \mathbb{R}$ is a constant that does not depend on k.

- The conditional probability $P_k(x)$ can then be expressed as

$$P_k(x) = \frac{\exp(\beta_k^T x + \beta_{k0})}{\sum_{l=1}^{c} \exp(\beta_l^T x + \beta_{l0})}.$$

- This transformation from $\beta_k^T x + \beta_{k0} \in \mathbb{R}$ to $P_k(x) \in [0, 1]$ such that $\sum_{k=1}^{c} P_k(x) = 1$ is called the softmax transformation.

Graphical representation of multinomial logistic regression



Learning

- The conditional likelihood for the multinomial model is

$$L(\beta) = \prod_{i=1}^{n} \mathbb{P}(Y_i = y_i \mid X_i = x_i; \beta)$$
$$= \prod_{i=1}^{n} \prod_{k=1}^{c} [P_k(x_i; \beta)]^{y_{ik}}$$

- The conditional log-likelihood is

$$\ell(\beta) = \sum_{i=1}^{N} \sum_{k=1}^{c} y_{ik} \log P_k(x_i; \beta),$$

- It can be maximized by the Newton-Raphson algorithm as in the binary case.

Multinomial logistic regression in R

```
library(nnet)
fit<-multinom(V1~.,data=letter.train)
pred.letters<-predict(fit,newdata=letter.test)

perf <-table(letter.test$V1,pred.letters)
1-sum(diag(perf))/ntst

0.2726
```

Logistic regression vs. LDA

- For a two-class problem, we have seen that, for LDA

$$\log \frac{P(x)}{1 - P(x)} = \alpha_0 + \alpha^T x$$

- So it has the same form as logistic regression. The difference is in how the parameters are estimated:
  - Logistic regression uses the conditional likelihood based on the conditional probabilities $P_k(x)$ (discriminative model).
  - LDA uses the full likelihood based on the joint distribution of X and Y (generative model).
- Despite these differences, in practice the results are often very similar.
- LDA tends to be more stable when the classes are well-separated, but it is also less robust to outliers.