# ML01 – Introduction to Machine Learning

## Linear and Quadratic Classification

Thierry Denœux

tdenoeux@utc.fr
https://www.hds.utc.fr/~tdenoeux

Université de technologie de Compiègne

Spring 2021

# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Classification

- In classification problems, the response variable $Y$ is nominal – e.g.,
  - Email is one of $\mathcal{C} = \{\text{spam}, \text{email}\}$
  - Facial expression is one of $\mathcal{C} = \{\text{sadness}, \text{joy}, \text{disgust}, \ldots\}$
  - Object is one of $\mathcal{C} = \{\text{pedestrian}, \text{car}, \text{bike}, \ldots\}$, etc.
- Suppose the elements in $\mathcal{C}$ are numbered $1, 2, \ldots, c$. Variable $Y$ partitions the underlying population in $c$ classes. It is called the class variable.
- Our goals are to:
  - Build a classifier $C : \mathbb{R}^p \to \mathcal{C}$ that predicts the class a future observation $X$.
  - Assess the uncertainty in each classification
  - Understand the roles of the different predictors among $X = (X_1, X_2, \ldots, X_p)$.

# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Formalization

- We have a feature (predictor) vector $X$, and a discrete response variable $Y$, both random.
- To represent the joint distribution of $(X, Y)$, we can give:
  1. The marginal distribution of $Y$. We use the notation

$$\pi_k = \mathbb{P}(Y = k),$$

and we call $\pi_k$ the prior probability of class $k$. We have

$$\sum_{k=1}^{c} \pi_k = 1$$

  2. The conditional probability density function (pdf) of $X$ given $Y = k$, for $k = 1, \ldots, c$. We use the notation

$$p_k(x) = p(x \mid Y = k)$$

# Formalization (continued)

We can then compute

- The marginal (mixture) pdf of $X$ as

$$p(x) = \sum_{k=1}^{c} p_k(x)\pi_k$$

- The conditional distribution of $Y$ given $X = x$ using Bayes' theorem. Let

$$P_k(x) = \mathbb{P}(Y = k \mid X = x)$$

denote the posterior (conditional) class probabilities. We have

$$\boxed{P_k(x) = \frac{p_k(x)\pi_k}{p(x)}, \quad k = 1, \ldots, c}$$

# Example

- Consider a classification problem with $c = 3$ classes and $p = 1$ feature.
- Assume that

$$\pi_1 = 0.3, \quad \pi_2 = 0.5, \quad \pi_3 = 0.2$$

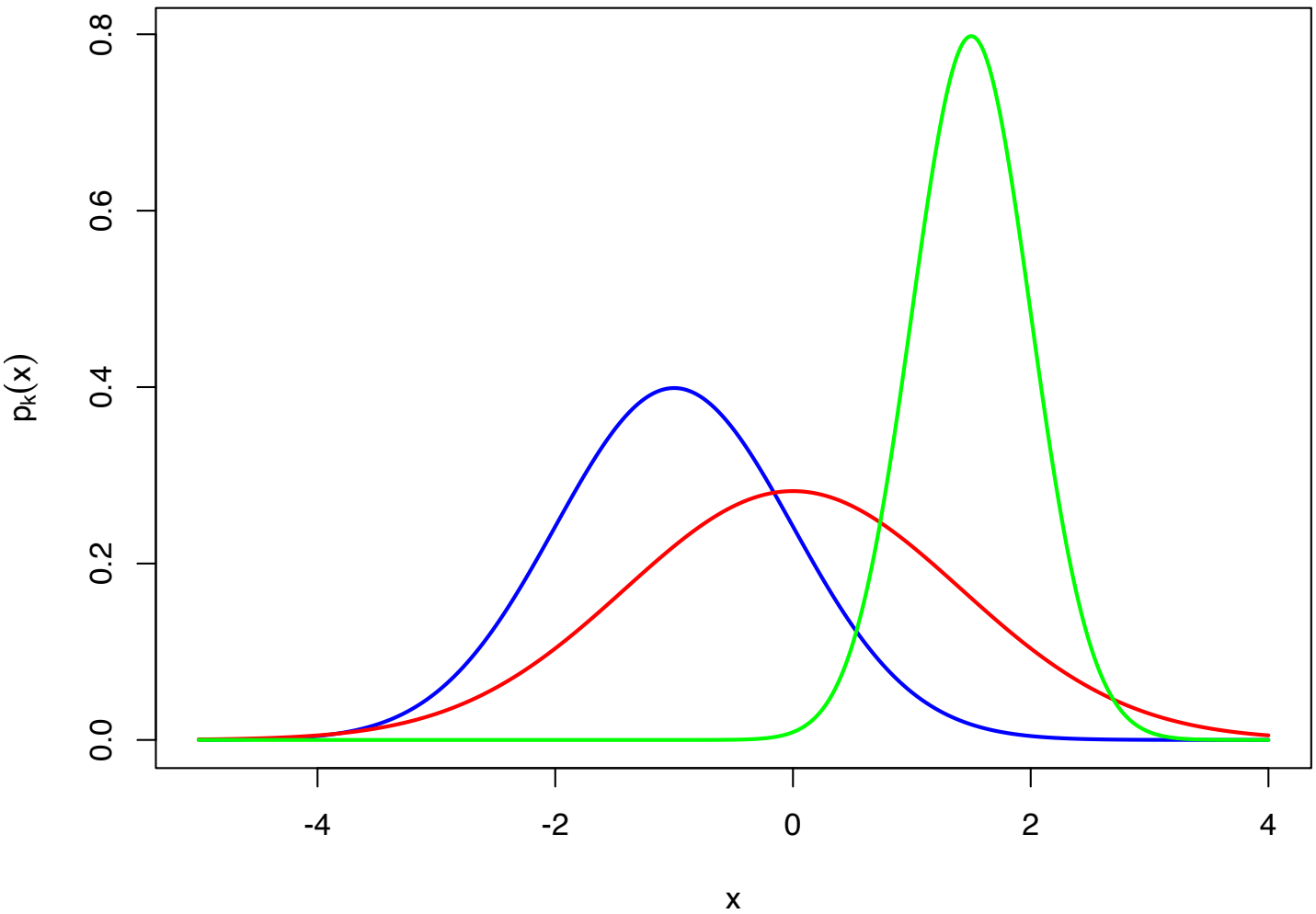
$$p_k(x) = \phi(x; \mu_k, \sigma_k)$$

where $\phi$ is the normal pdf, with

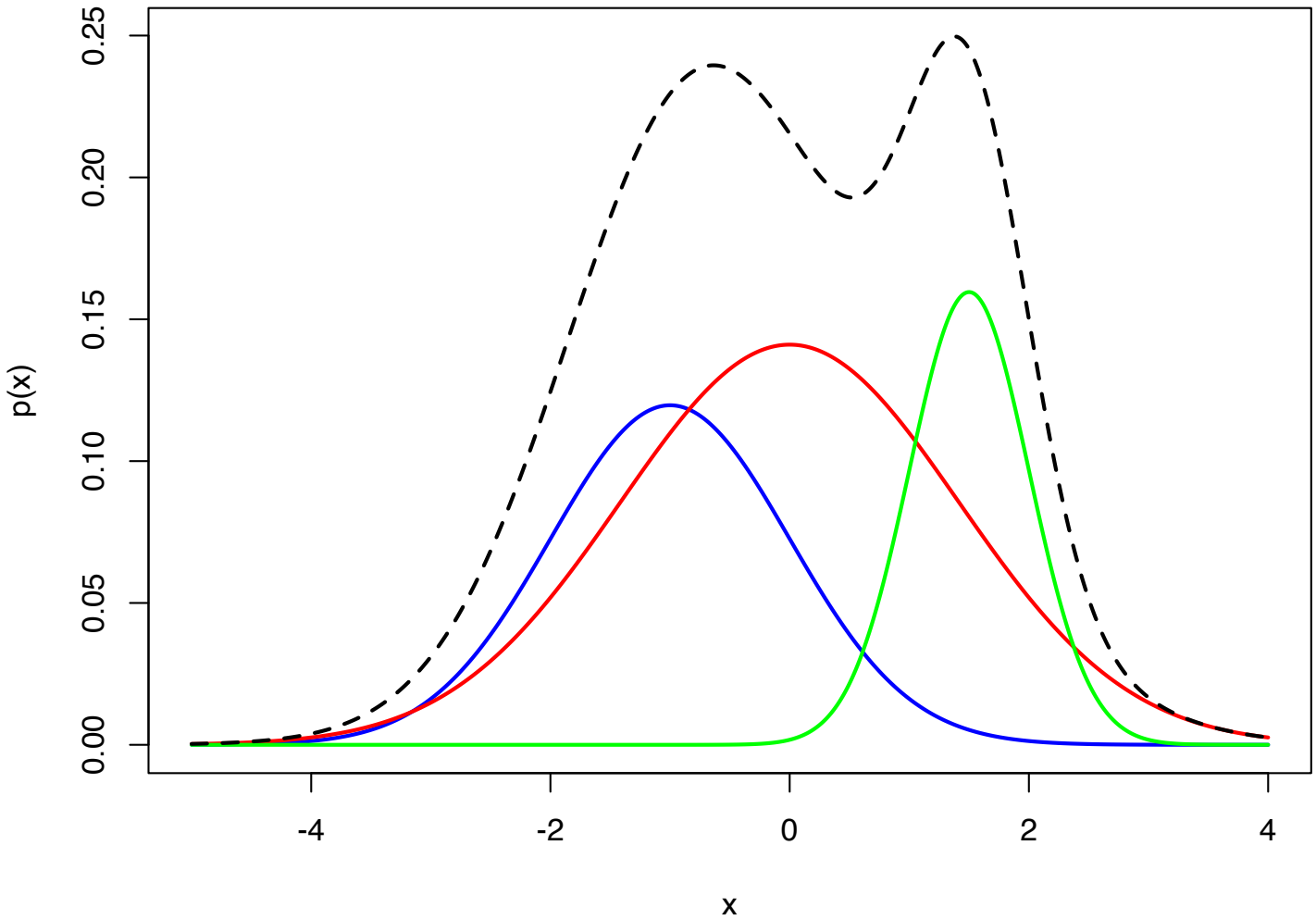
$$\mu_1 = -1, \quad \mu_2 = 0, \quad \mu_3 = 1.5$$

$$\sigma_1 = 1, \quad \sigma_2 = \sqrt{2}, \quad \sigma_3 = 0.5$$

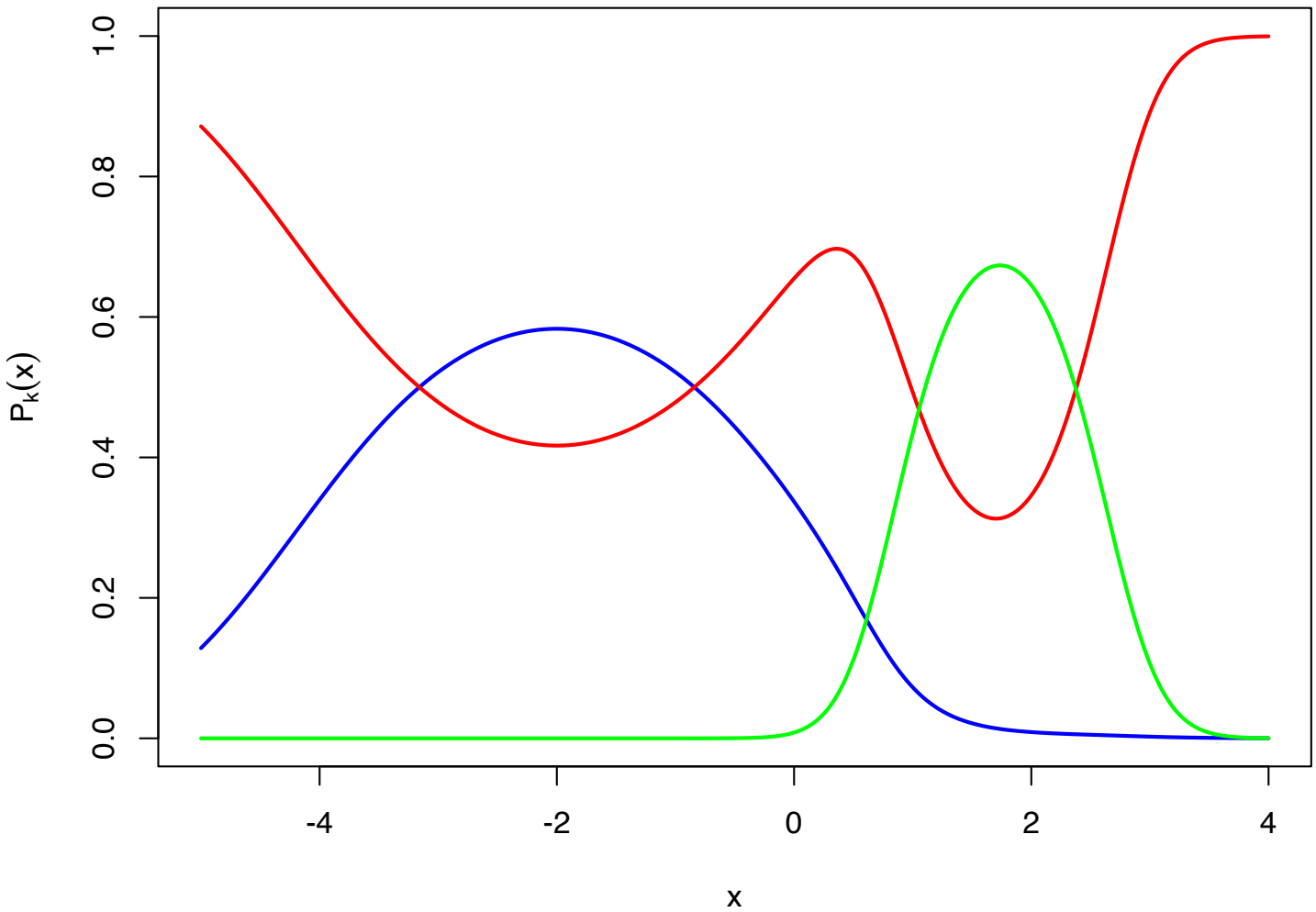
# Example: conditional densities $p_k(x)$



# Example: marginal density $p(x)$



# Example: posterior probabilities $P_k(x)$



# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# The Bayes classifier

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

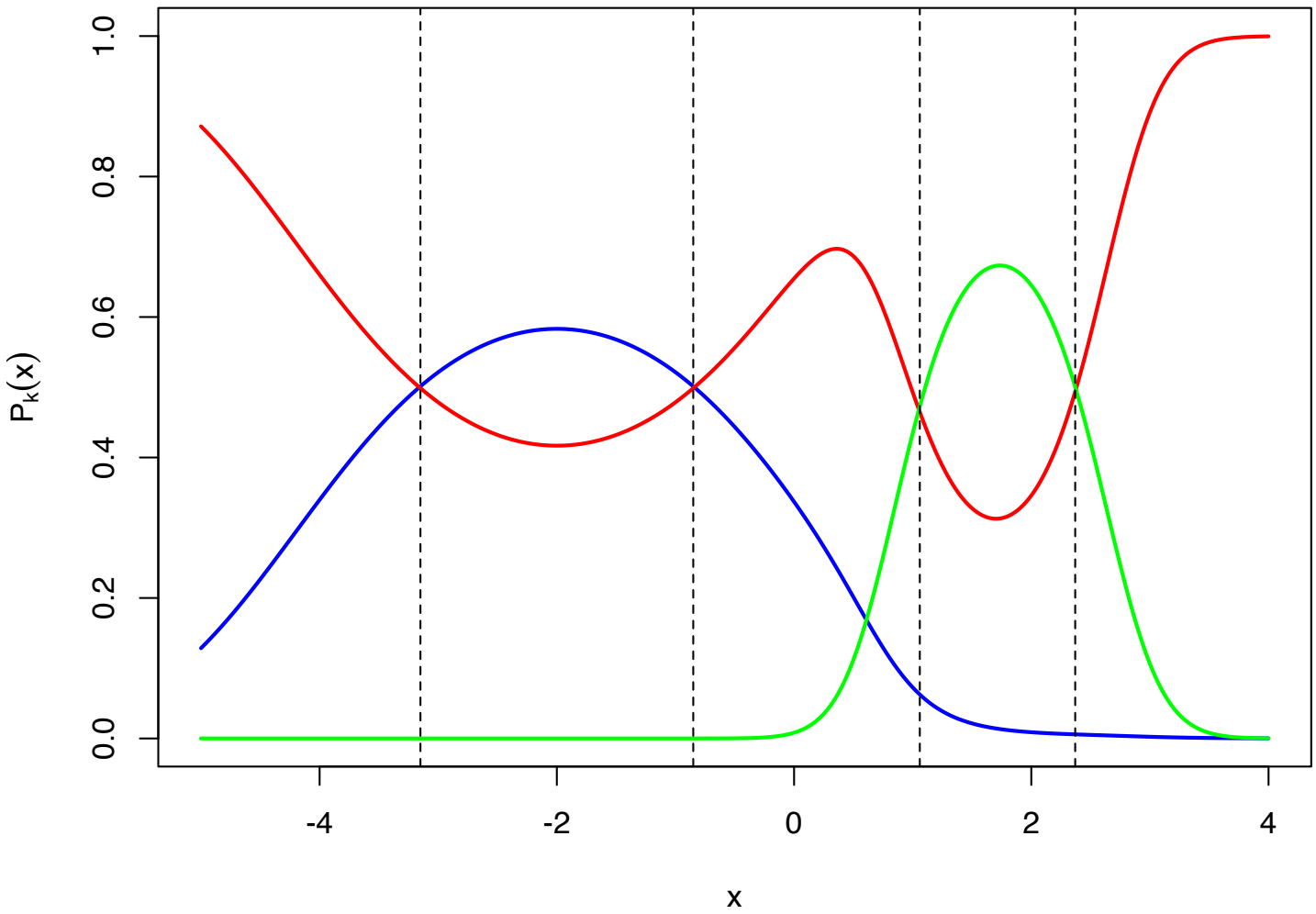
$$\begin{aligned}\mathbb{P}(\text{error} \mid X = x) &= \mathbb{P}(C(X) \neq Y \mid X = x) \\ &= 1 - \mathbb{P}(C(X) = Y \mid X = x)\end{aligned}$$

- 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.

# Example: decision regions of the Bayes classifier



# 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

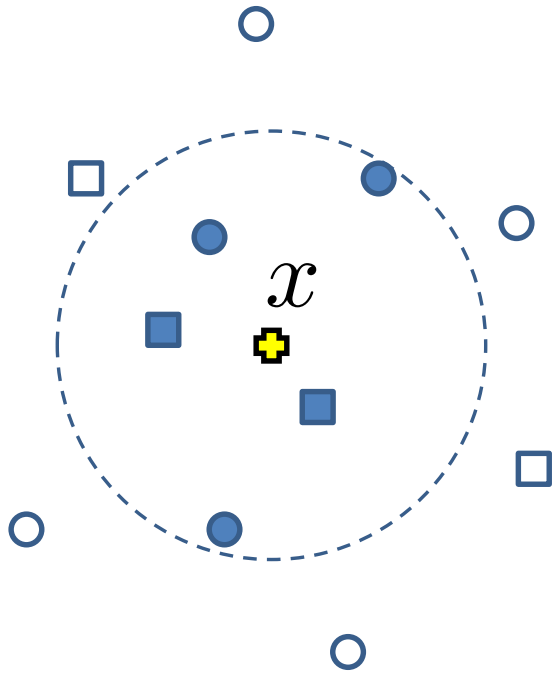
$$\text{Err}_B = \mathbb{E}_X \left[ 1 - \max_k P_k(X) \right] = \int \left[ 1 - \max_k P_k(x) \right] p(x) dx$$

- 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


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# $K$ nearest neighbors




- Nearest-neighbor averaging can be used as in regression.
- Let $x_{(1)}, \ldots, x_{(K)}$ denote the $K$ nearest neighbors of $x$ in the learning set, and $y_{(1)}, \ldots, y_{(K)}$ the corresponding class labels.

# Voting $K$-nearest-neighbor rule

- The posterior probability of class $k$ can be estimated by the proportion of observations from that class among the $K$ nearest neighbors of $x$:

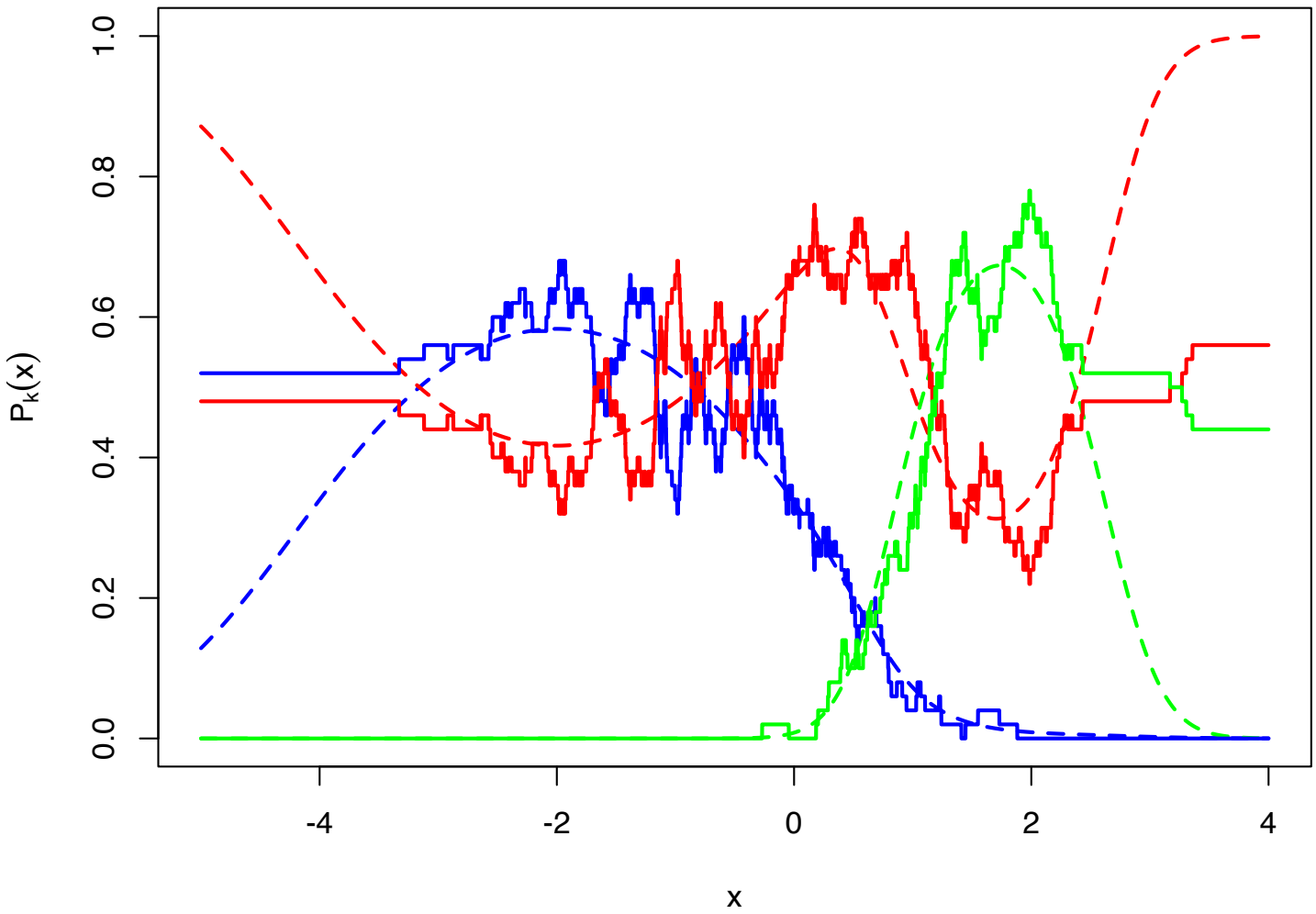
$$\widehat{P}_k(x) = \frac{1}{K} \#\{i \in \{1, \ldots, K\} : y_{(i)} = k\}$$

- Voting $K$-nearest neighbor ($K$-NN) rule: select the majority class among the $K$ nearest neighbors:

$$C_K(x) = \arg\max_k \widehat{P}_k(x).$$

- As in regression, the $K$-NN rule breaks down as dimension grows. However, the impact on $C_K(x)$ is less than that on the probability estimates $\widehat{P}_k(x)$.

# Example: voting $K$-NN rule with $n = 1000$ and $K = 50$



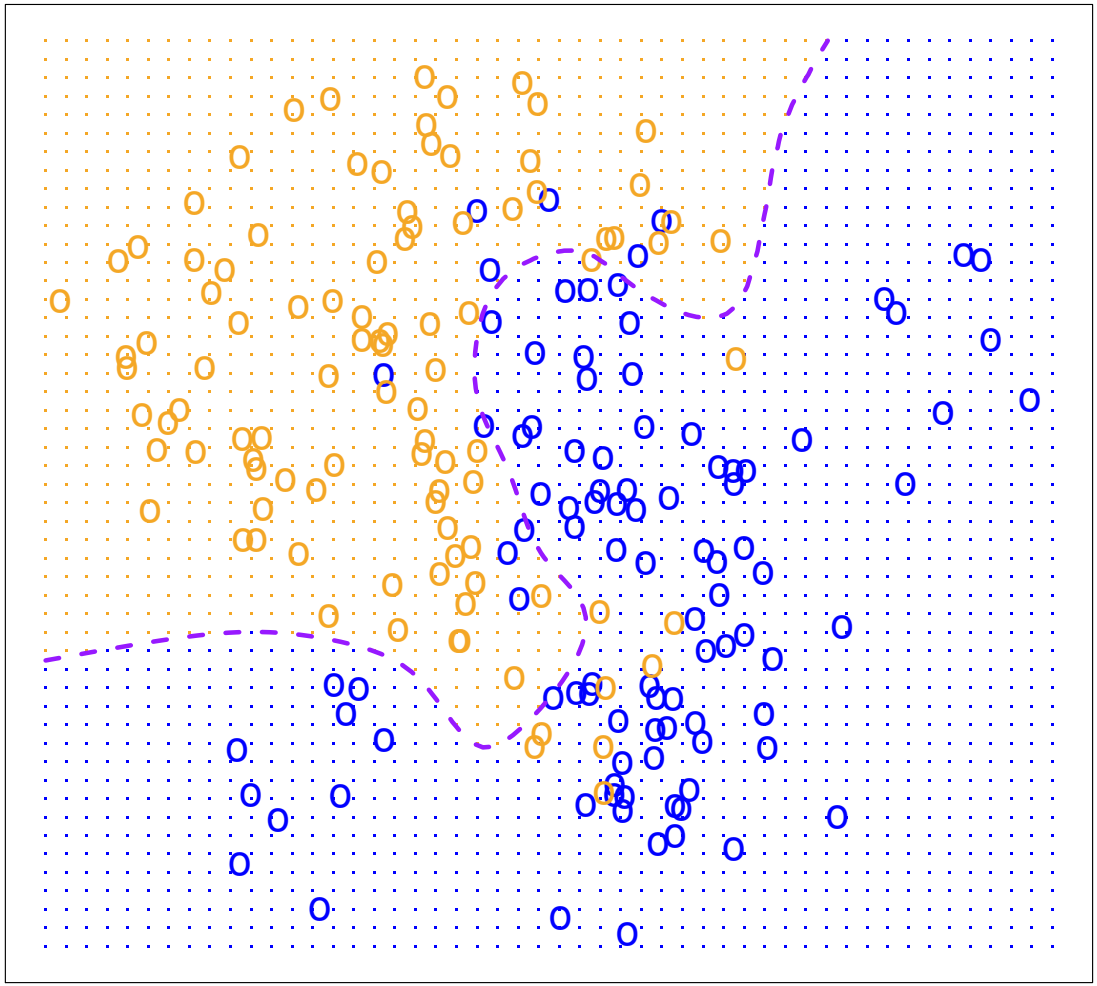
# Error rate estimation

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

$$\text{Err}_{\mathcal{T}} = \frac{1}{m} \# \left\{ i \in \{1, \ldots, m\} : y'_i \neq C(x'_i) \right\},$$

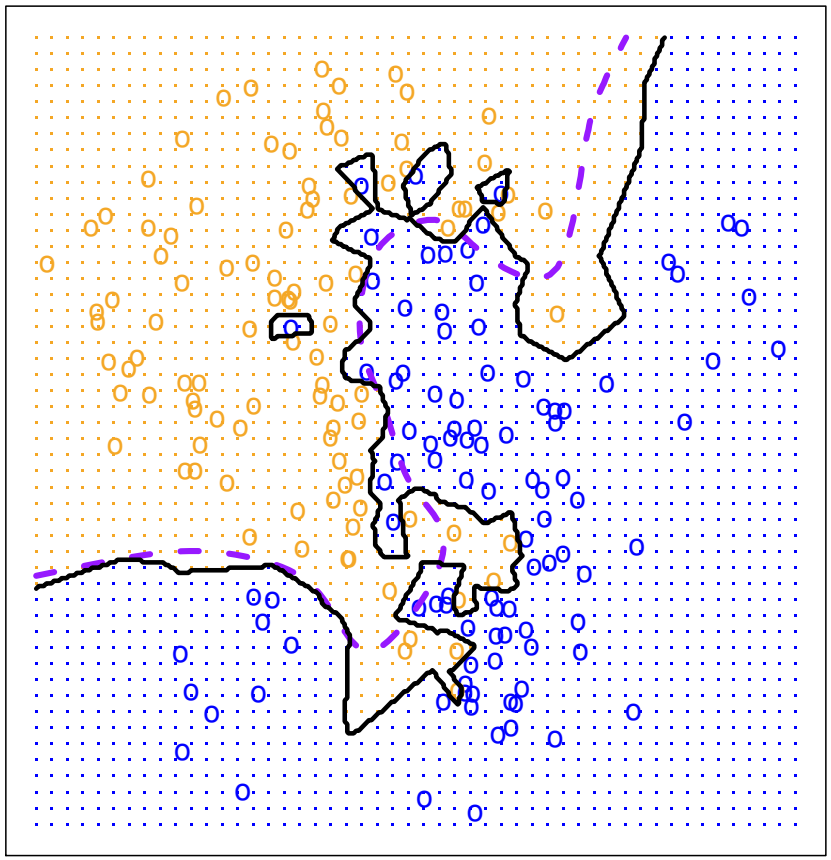
  where $\mathcal{T}$ is a test dataset.
- The test error rate allows us to select the best model in a set of candidate models (more on this later).

# Example: simulated data and Bayes decision boundary

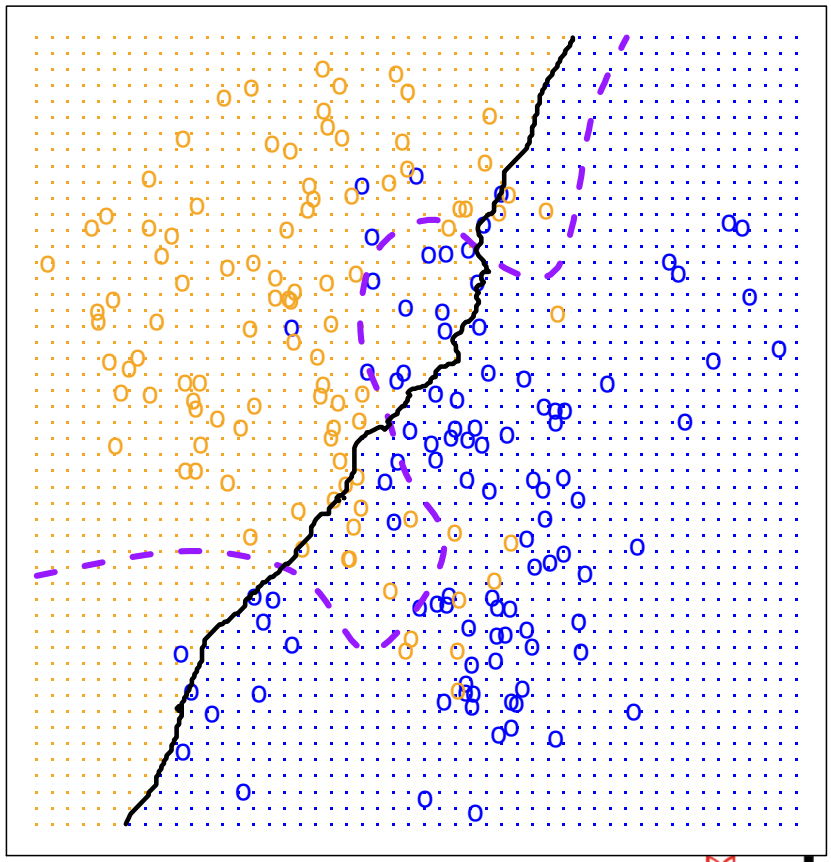


# Decision boundaries for $K = 1$ and $K = 100$

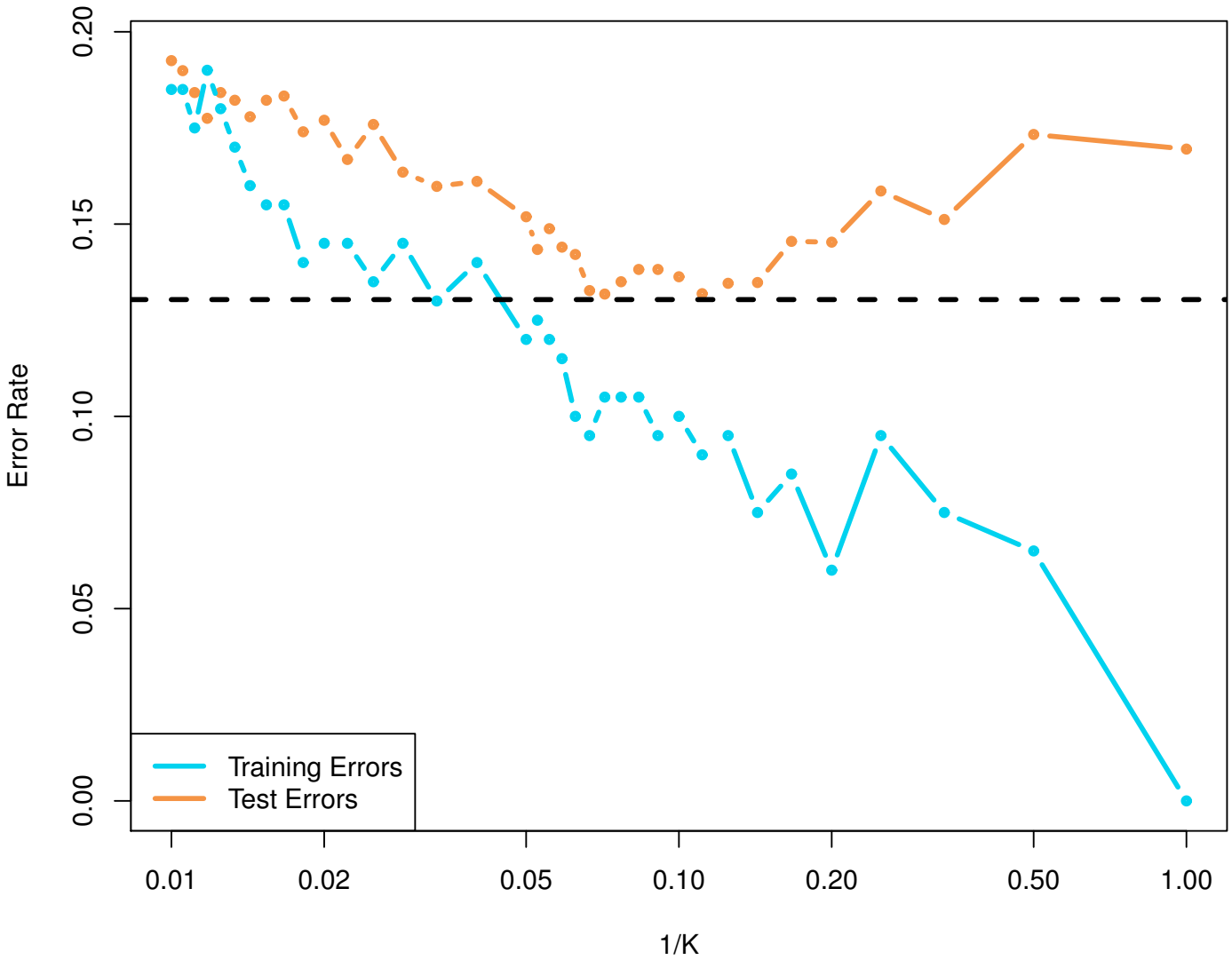
KNN: K=1



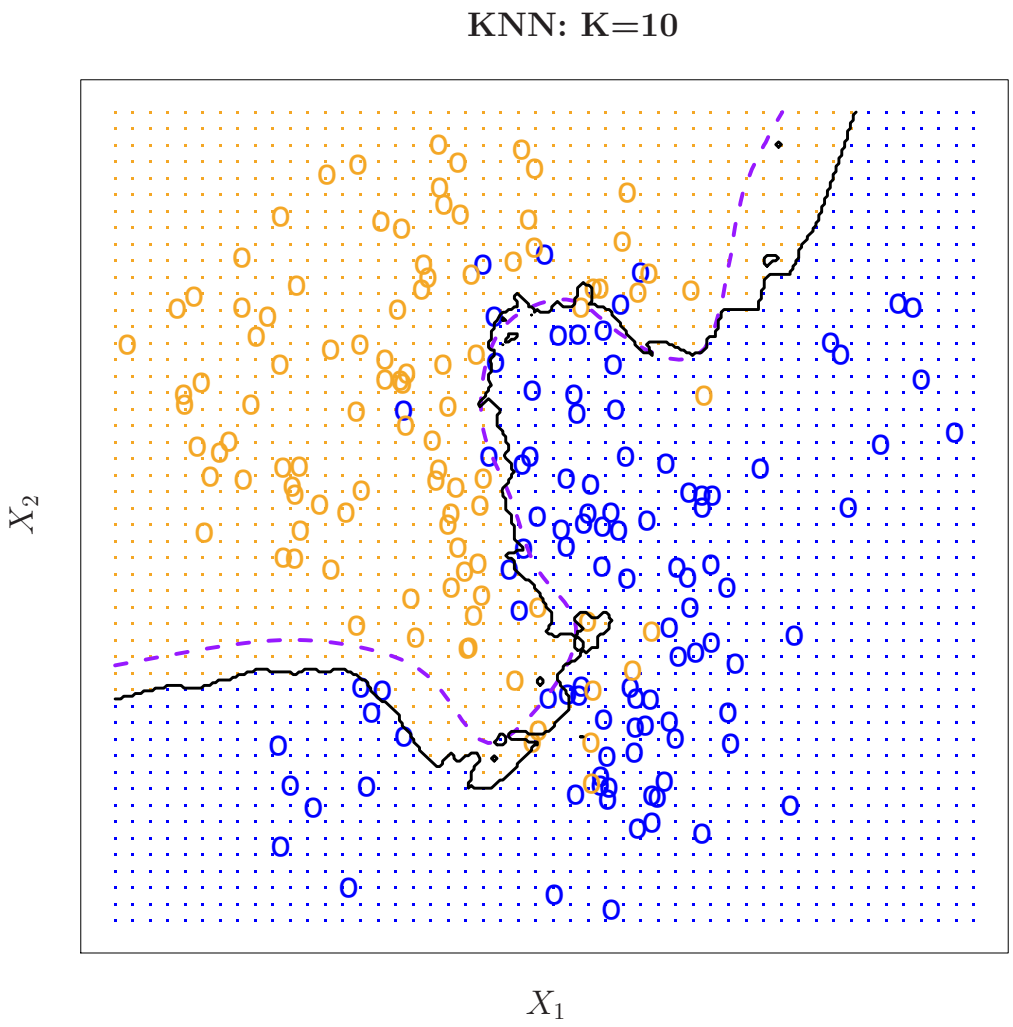
KNN: K=100



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



# Decision boundary for the best value of $K$



# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Linear/quadratic 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:

$$\mathcal{R}_k = \{x \in \mathbb{R}^p : C(x) = k\}, \quad k = 1, \ldots, c.$$

- 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 or quadratic, i.e., they have equations of the form

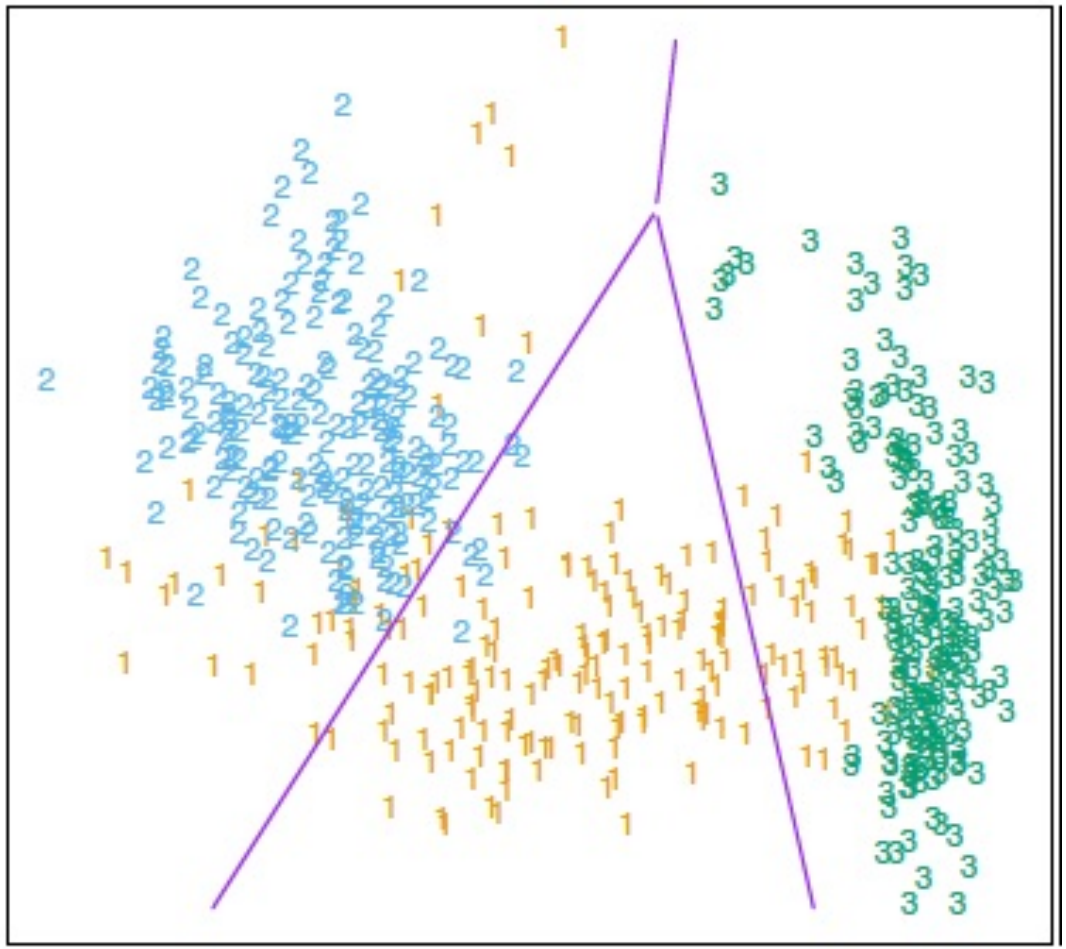
$$\beta^T x + \beta_0 = 0 \quad \text{(linear)}$$

or

$$x^T Q x + \beta^T x + \beta_0 = 0 \quad \text{(quadratic)}$$

- This is what we will mean by linear and quadratic methods for classification.

# Example



# Generative vs. discriminative models

- To approximate Bayes' rule, we need to estimate the posterior probabilities $P_k(x) = P(Y = k \mid X = x)$.
- We can distinguish two kinds of models for classification:

  Generative models represent the conditional pdf's $p_k(x)$ and the prior probabilities $\pi_k$. Using Bayes' theorem, we then get the posterior probabilities $P_k(x)$.

  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 families of classifiers:
  1. Linear and quadratic classifiers based on a generative model: Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA).
  2. A linear classifier based on a discriminative model: Logistic regression.

# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Model

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

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

  where $\mu_k = \mathbb{E}(X \mid Y = k)$ and $\mathbf{\Sigma}_k = \mathrm{Var}(X \mid Y = k)$.
- LDA further assumes that the covariance matrices are equal:

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

- We start with a study of LDA.

# Linear boundaries of the Bayes classifier

## Proposition

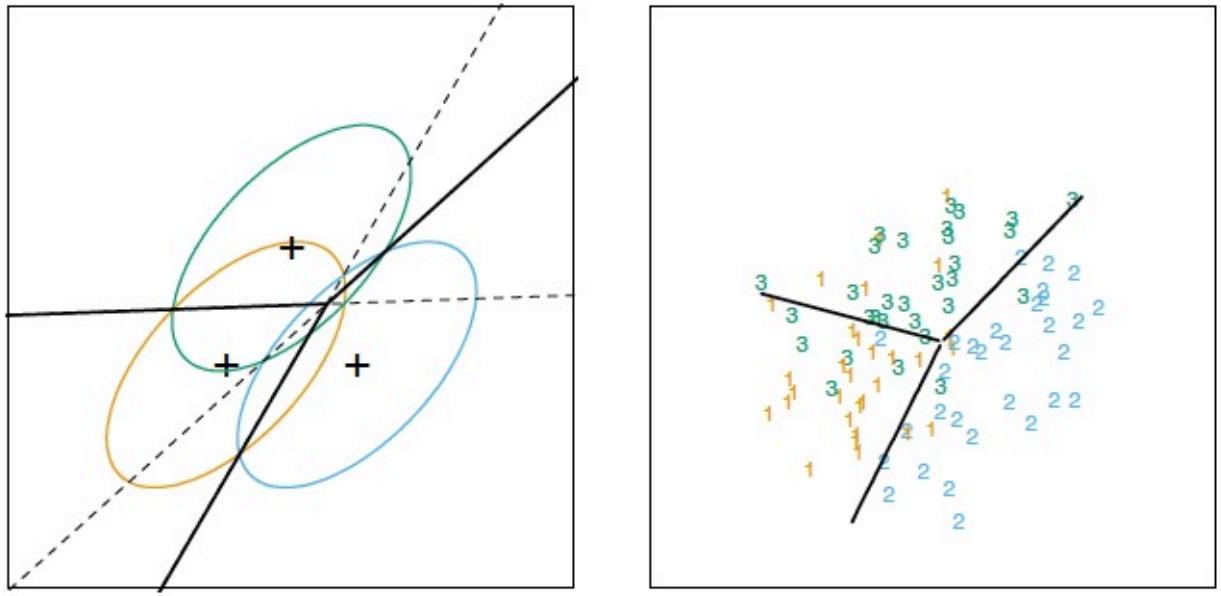
*Under the assumptions of LDA, the boundary between any two decision regions $\mathcal{R}_k$ and $\mathcal{R}_\ell$ of the Bayes classifier is a hyperplane defined by the equation:*

$$(\mu_k - \mu_\ell)^T \mathbf{\Sigma}^{-1} x - \frac{1}{2}(\mu_k + \mu_\ell)^T \mathbf{\Sigma}^{-1}(\mu_k - \mu_\ell) + \log \frac{\pi_k}{\pi_\ell} = 0 \qquad (1)$$

Proof.

See example on next slide.

# 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 drawn from each distribution, and the fitted LDA decision boundaries.

# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Plug-in LDA classifier

- The model parameters are $\pi_k$, $\mu_k$ $(k = 1, \ldots, c)$ and the common covariance matrix $\mathbf{\Sigma}$.
- To implement LDA, we compute the maximum likelihood estimates (MLEs) of these parameters and we "plug in" these estimates in the expressions of the conditional probabilities $P_k(x)$.
- From the consistency of the MLEs, the resulting plug-in classifier will tend to the Bayes classifier as the sample size $n$ tends to infinity (assuming the model to be correct).

# Maximum likelihood estimates

- The MLEs 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 covariance matrix in class $k$:

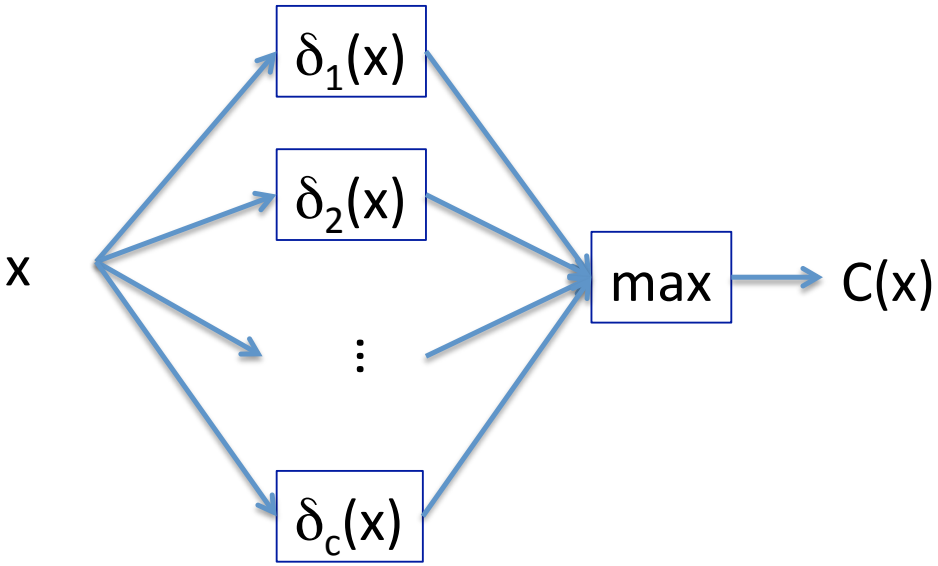
$$\widehat{\mathbf{\Sigma}}_k = \frac{1}{n_k} \sum_{i=1}^{n} y_{ik} (x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^T,$$

  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

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

# Discriminant functions




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

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

- Simple DF's can be obtained by computing the $\log \widehat{P}_k(x)$, and dropping the terms that are constant for all classes. Here, we get (Proof)

$$\delta_k(x) = \widehat{\mu}_k^T \widehat{\mathbf{\Sigma}}^{-1} x - \frac{1}{2} \widehat{\mu}_k^T \widehat{\mathbf{\Sigma}}^{-1} \widehat{\mu}_k + \log \widehat{\pi}_k.$$

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

# Example: Letter recognition dataset

- Source: P. W. Frey and D. J. Slate, *Machine Learning*, Vol 6 #2, March 91.
- Objective: identify 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 instances.
- Each instance was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were scaled to fit into a range of integer values from 0 through 15.

# 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


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Case $c = 2$: fixing the threshold

- From (1), in the case of $c = 2$ classes, LDA assigns $x$ to class 2 if

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

  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 assumptions are not verified, a different threshold may give better results.
- 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 $s$.

# Example: Pima diabetes dataset

- 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. Tested positive (1) or negative (0) for diabetes
- Problem: predict the test result for the 8 predictors.

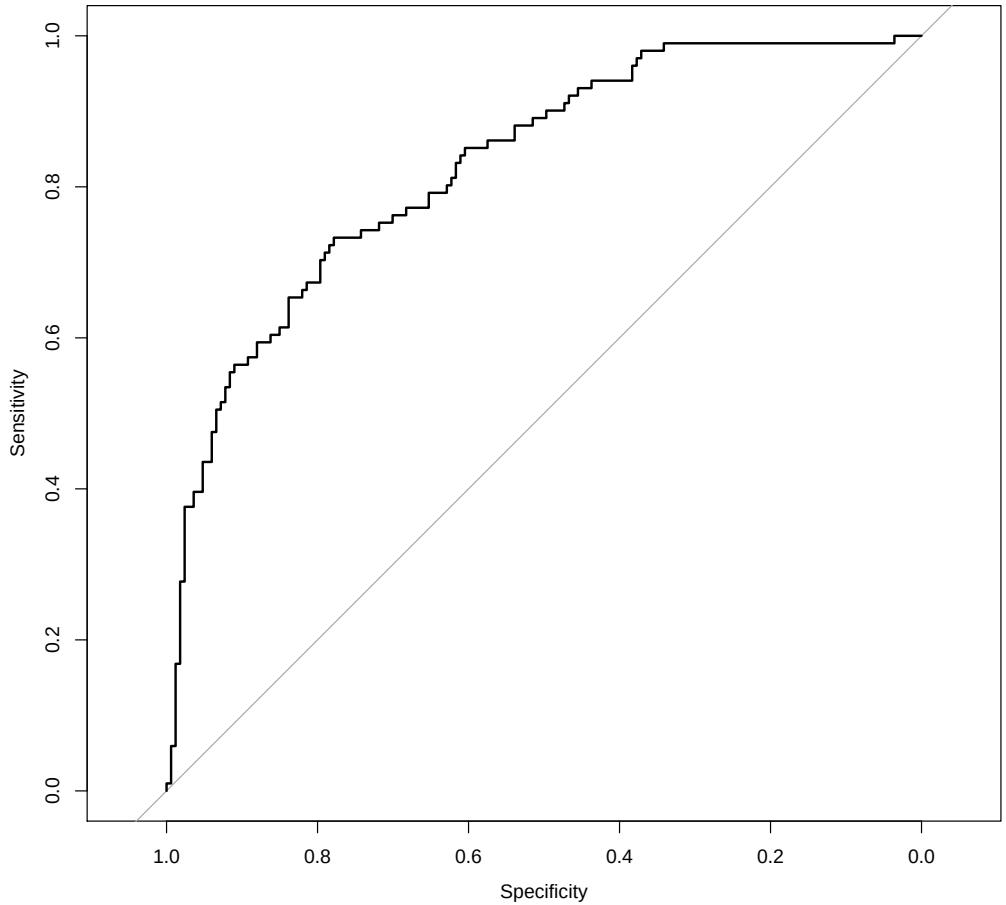
# LDA of the Pima dataset

```
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 $(15 + 45)/268 \approx 0.22$.

# ROC curve for the LDA classifier (Pima dataset)

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



# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

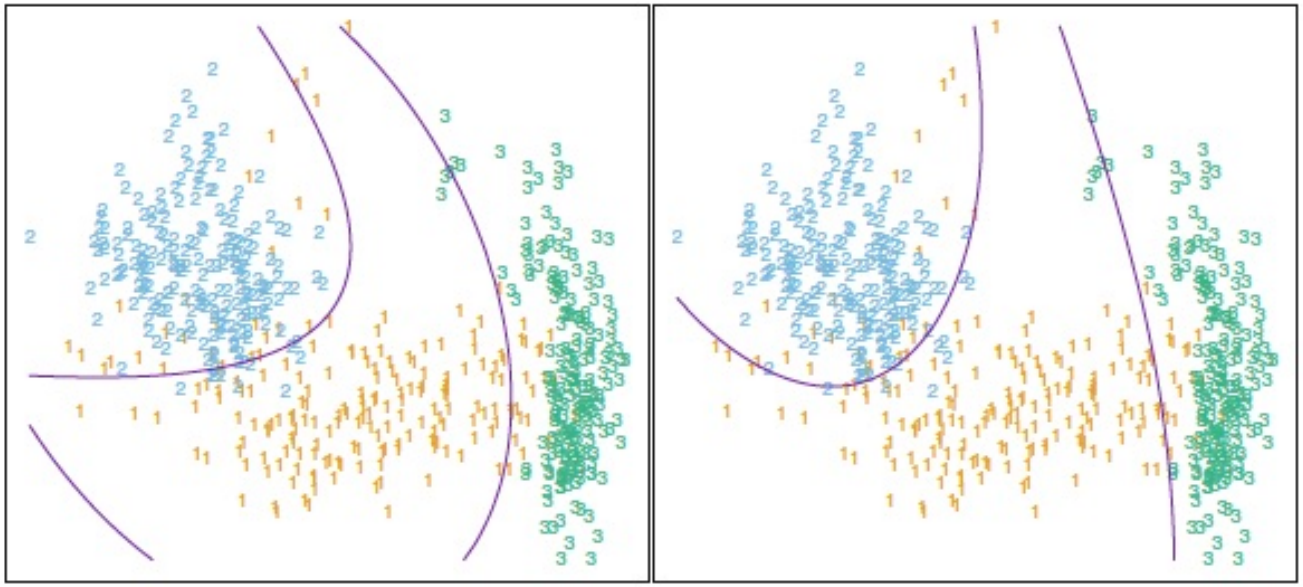
# Quadratic Discriminant Analysis (QDA)

- 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 for the Bayes' rule:

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

- 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 $\mathbf{\Sigma}_k$ 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 terms in $\widehat{\mathbf{\Sigma}}_k$ to 0. The variance $\sigma_{kj}^2$ 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:

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

This model can be called "Naive LDA".

# Advantages of Naive Bayes classifiers

- In spite of their simplicity, naive Bayes classifiers often (but not always) have very good performances.
- They can accommodate mixed feature vectors (qualitative and quantitative). If $X_j$ is qualitative, estimate the probability mass functions $p_{kj}(x_j)$ using histograms 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 largest number of parameters.
- Although LDA also has a number of parameters proportional 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

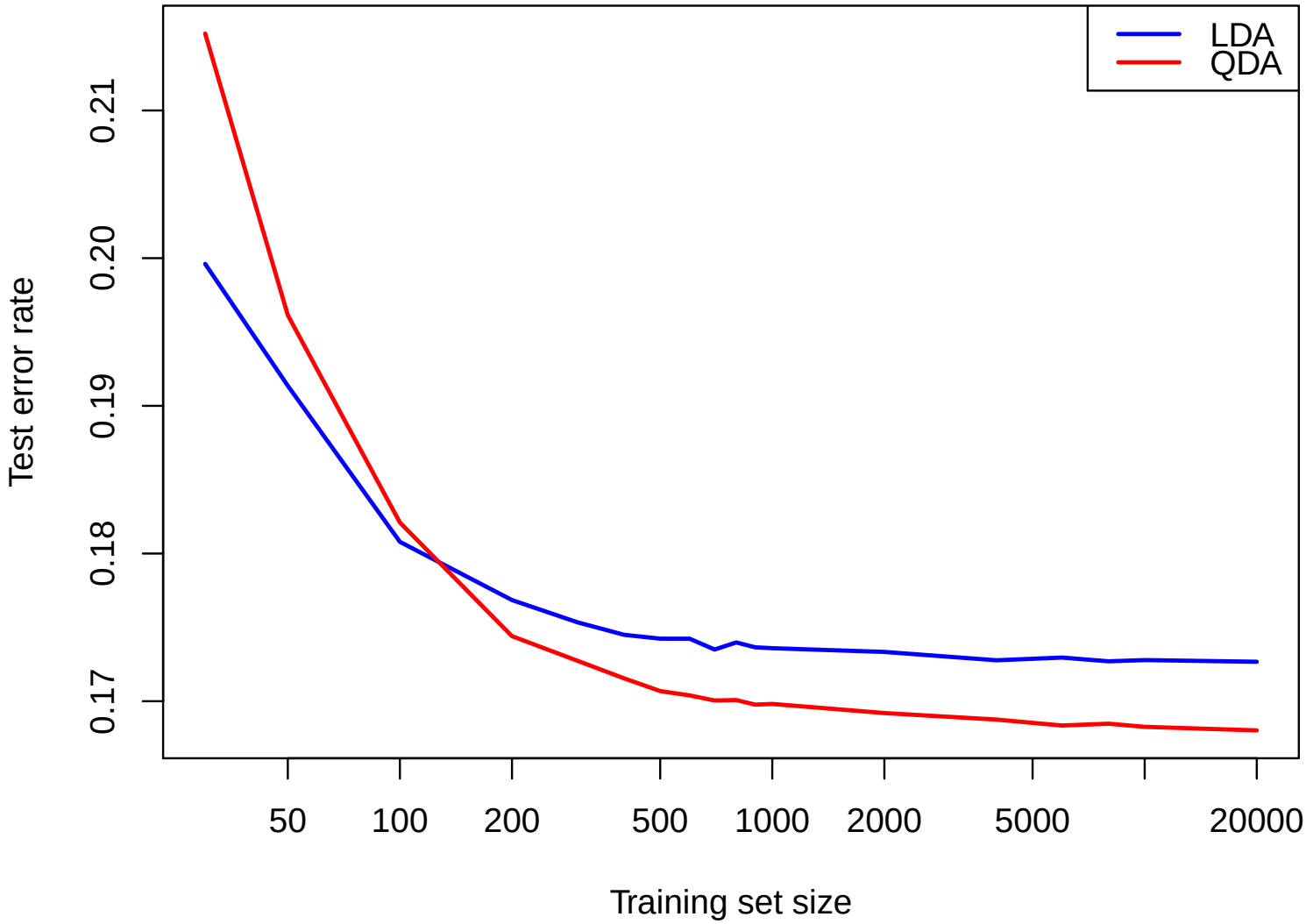
$$\pi_1 = \pi_2 = 0.5$$

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

$$\Sigma_1 = \mathbf{I}_3, \quad \Sigma_2 = 0.7\mathbf{I}_3.$$

- 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 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


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Searching for a linear discriminative model

- 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:

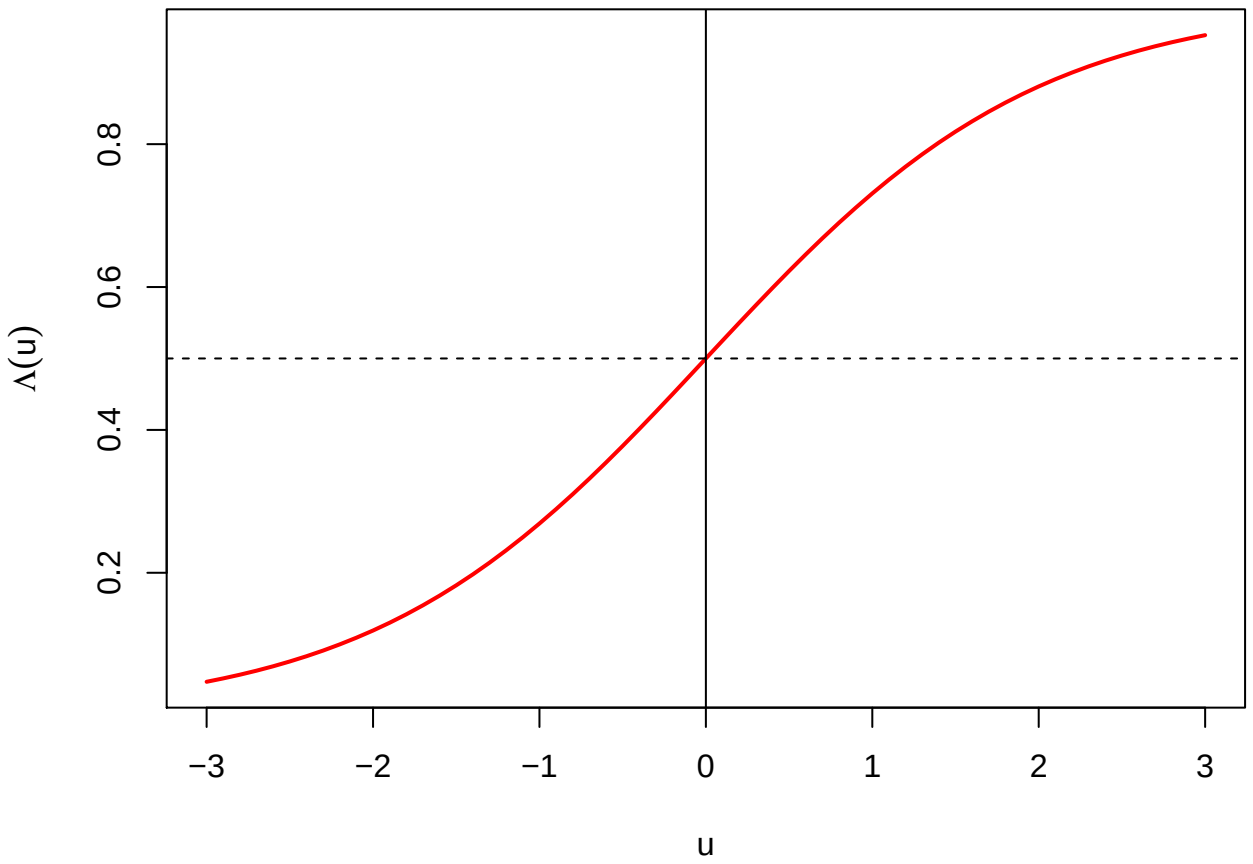
$$\log \frac{P(x)}{1 - P(x)} = \beta_0 + \beta^T x$$

- Equivalently, we can write

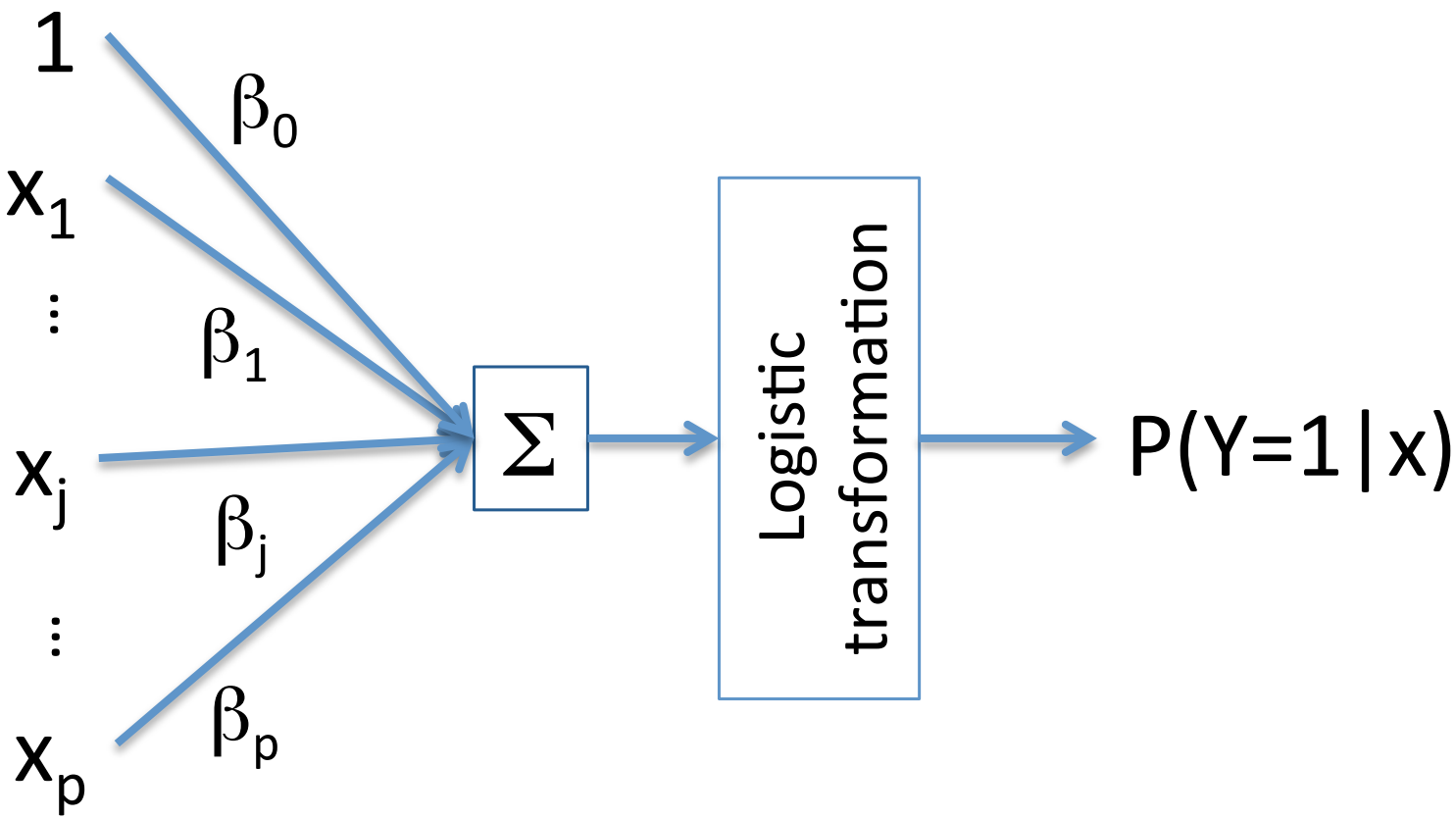
$$P(x) = \frac{1}{1 + \exp[-(\beta_0 + \beta^T x)]} = \Lambda(\beta_0 + \beta^T x),$$

  where $\Lambda(u) = [1 + \exp(-u)]^{-1}$ is called the logistic function.
- Link with neural networks (more on this later in this course).

# Plot of the logistic function



# Graphical representation of binomial logistic regression



# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# 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.
- Assuming $Y_1, \ldots, Y_n$ to be independent conditionally on $X_1 = x_1, \ldots, X_n = x_n$, the conditional likelihood is

$$\begin{aligned} L(\beta) &= \mathbb{P}(Y_1 = y_1, \ldots, Y_n = y_n; X_1 = x_1; \ldots; X_n = x_n) \\ &= \prod_{i=1}^{n} \mathbb{P}(Y_i = y_i | X_i = x_i; \beta) \\ &= \prod_{i=1}^{n} P(x_i; \beta)^{y_i} \left[1 - P(x_i; \beta)\right]^{1-y_i} \end{aligned}$$

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

# Maximization

- The conditional log-likelihood is

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

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

- This function is non linear and the score equation $\frac{\partial \ell}{\partial \beta} = 0$ does not have a closed-form expression: we need to use an iterative nonlinear optimization procedure such as the Newton-Raphson algorithm.

# Update equation

- Let $\mathbf{y}$ 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 $i$-th element $P(x_i; \beta)$.
- The gradient and Hessian of $\ell(\beta)$ can be written as

$$\frac{\partial \ell}{\partial \beta} = \mathbf{X}^T(\mathbf{y} - \mathbf{p}) \quad \text{and} \quad \frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\mathbf{X}^T \mathbf{W} \mathbf{X},$$

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

- The update equation is, thus,

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

# 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 $\beta_j$.

# 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
```

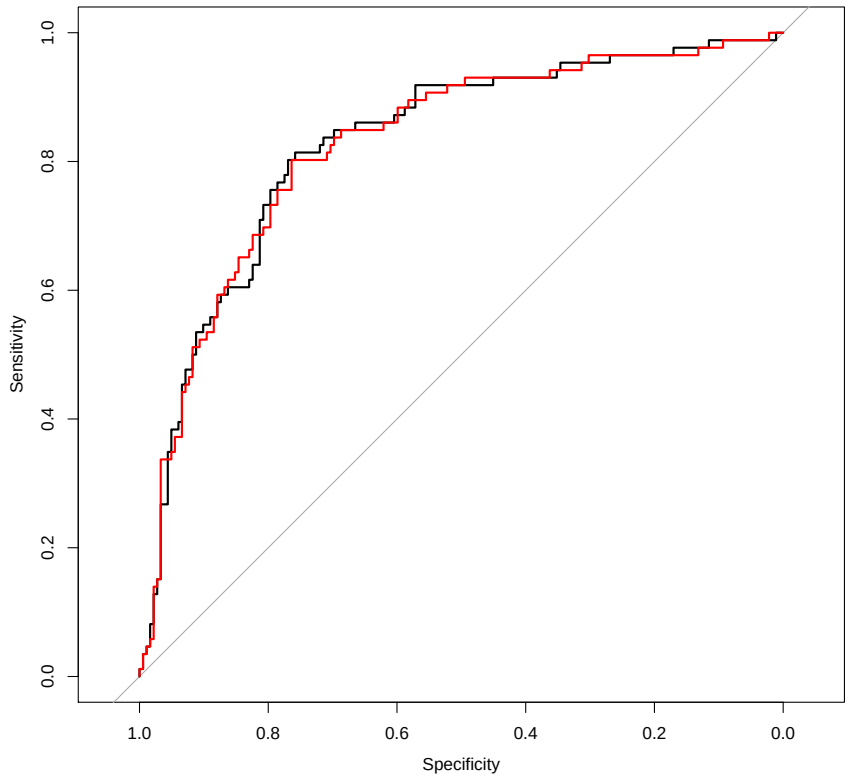
# Prediction

```
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


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Model

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

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

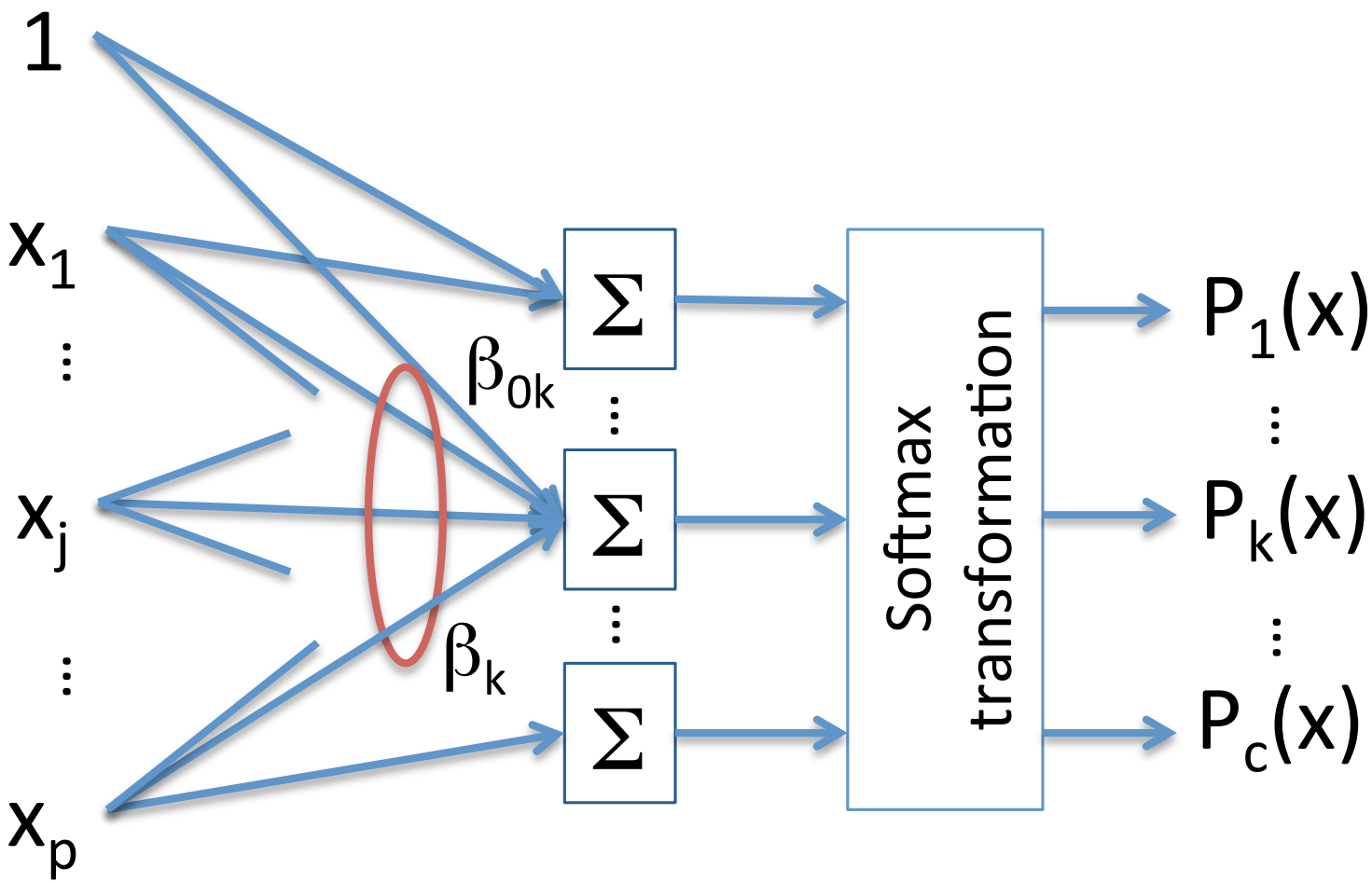
where $P_k(x) = \mathbb{P}(Y = k \mid X = x)$, $\beta_k \in \mathbb{R}^p$, $\beta_{k0} \in \mathbb{R}$ and $\gamma \in \mathbb{R}$ is a constant that does not depend on $k$.

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

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

- 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.

# Overview


1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting $K$-NN rule
2. Linear Discriminant Analysis
   - Model
   - Parameter estimation
   - Case $c = 2$
   - Related models
3. Logistic regression
   - Model (case $c = 2$)
   - Parameter estimation
   - Multinomial logistic regression
4. Appendix

# Decision boundaries of LDA I

- The decision boundary between regions $\mathcal{R}_k$ and $\mathcal{R}_\ell$ is defined by the equation $P_k(x) = P_\ell(x)$, which can be written as

$$\log \frac{P_k(x)}{P_\ell(x)} = \log \frac{p_k(x)\pi_k}{p_\ell(x)\pi_\ell} = \log p_k(x) - \log p_\ell(x) + \log \frac{\pi_k}{\pi_\ell} = 0$$

- Now,

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

so

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

# Decision boundaries of LDA II

- Consequently,

$$\begin{aligned}\log p_k(x) - \log p_\ell(x) &= \mu_k^T \boldsymbol{\Sigma}^{-1} x - \frac{1}{2}\mu_k^T \boldsymbol{\Sigma}^{-1}\mu_k \\ &\quad - \mu_\ell^T \boldsymbol{\Sigma}^{-1} x + \frac{1}{2}\mu_\ell^T \boldsymbol{\Sigma}^{-1}\mu_\ell \\ &= (\mu_k - \mu_\ell)^T \boldsymbol{\Sigma}^{-1} x - \frac{1}{2}\underbrace{\left[\mu_k^T \boldsymbol{\Sigma}^{-1}\mu_k - \mu_\ell^T \boldsymbol{\Sigma}^{-1}\mu_\ell\right]}_{(\mu_k+\mu_\ell)^T \boldsymbol{\Sigma}^{-1}(\mu_k-\mu_\ell)}\end{aligned}$$

Back

# Discriminant functions of LDA

From

$$\widehat{P}_k(x) = \frac{\widehat{p}_k(x)\widehat{\pi}_k}{\widehat{p}(x)}$$

we get

$$\begin{aligned}
\log \widehat{P}_k(x) &= \log \widehat{p}_k(x) + \log \widehat{\pi}_k + \text{cst} \\
&= -\frac{1}{2}(x - \widehat{\mu}_k)^T \widehat{\mathbf{\Sigma}}^{-1}(x - \widehat{\mu}_k) + \log \widehat{\pi}_k + \text{cst} \\
&= \widehat{\mu}_k^T \widehat{\mathbf{\Sigma}}^{-1} x - \frac{1}{2}\widehat{\mu}_k^T \widehat{\mathbf{\Sigma}}^{-1} \widehat{\mu}_k + \log \widehat{\pi}_k + \text{cst}
\end{aligned}$$

Back

# Log-likelihood of binary logistic regression

From

$$P(x_i) = \frac{1}{1+\exp(-\beta^T x_i)} \quad \text{and} \quad 1 - P(x_i) = \frac{\exp(-\beta^T x_i)}{1+\exp(-\beta^T x_i)}$$

we get

$$\ell(\beta) = \sum_{i=1}^{n} \Big\{ -y_i \log[1+\exp(-\beta^T x)] \underbrace{-\beta^T x_i - \log[1+\exp(-\beta^T x_i)]}_{-\log[1+\exp(\beta^T x)]} + y_i \beta^T x_i + y_i \log[1+\exp(-\beta^T x)] \Big\}$$

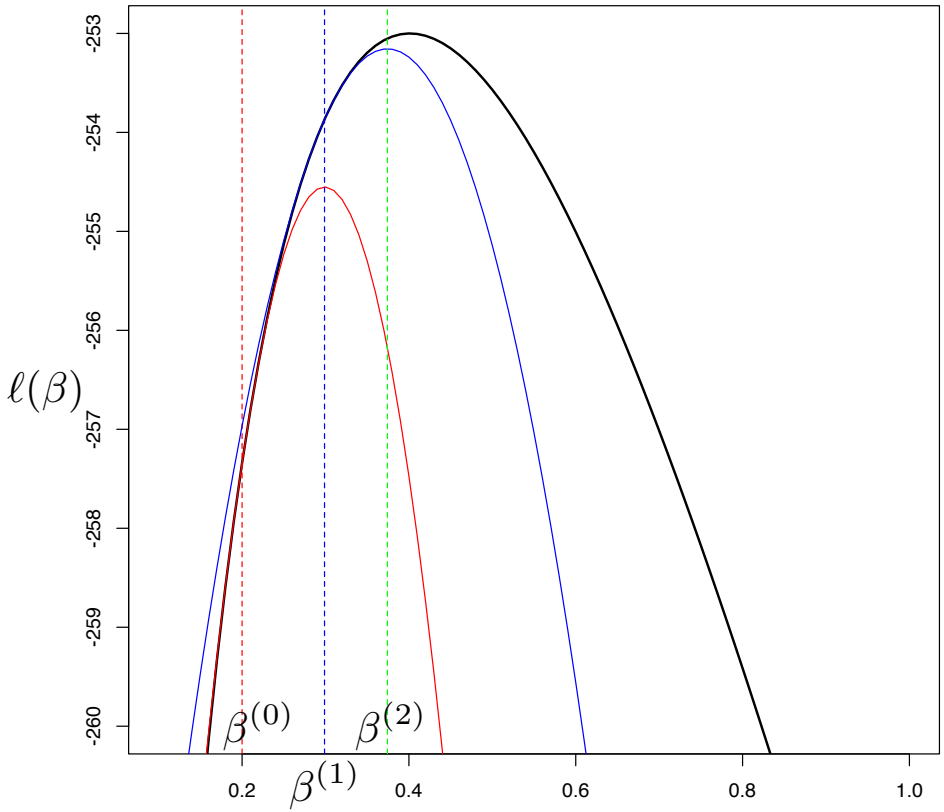
$$\ell(\beta) = \sum_{i=1}^{n} \Big\{ y_i \beta^T x_i - \log[1+\exp(\beta^T x)] \Big\}$$

Back

# The Newton-Raphson algorithm

## Main ideas

- An iterative optimization algorithm.
- Basic idea: at each time step, approximate $\ell(\beta)$ around the current estimate $\beta^{(t)}$ by the second-order Taylor series expansion.



# The Newton-Raphson algorithm

- We have

$$\ell(\beta) \approx \ell(\beta^{(t)}) + (\beta - \beta^{(t)})^T \frac{\partial \ell(\beta^{(t)})}{\partial \beta} + \frac{1}{2}(\beta - \beta^{(t)})^T \frac{\partial^2 \ell(\beta^{(t)})}{\partial \beta \partial \beta^T}(\beta - \beta^{(t)}).$$

- Differentiating both sides w.r.t. $\beta$, we get

$$\frac{\partial \ell(\beta)}{\partial \beta} \approx \frac{\partial \ell(\beta^{(t)})}{\partial \beta} + \frac{\partial^2 \ell(\beta^{(t)})}{\partial \beta \partial \beta^T}(\beta - \beta^{(t)}).$$

- Setting $\frac{\partial \ell}{\partial \beta}(\beta) = 0$, we get the update equation

$$\boxed{\beta^{(t+1)} = \beta^{(t)} - \left( \frac{\partial^2 \ell(\beta^{(t)})}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial \ell(\beta^{(t)})}{\partial \beta}}$$

Back

# Gradient of $\ell(\beta)$

From

$$\ell(\beta) = \sum_{i=1}^{n} \left\{ y_i \beta^T x_i - \log(1 + \exp(\beta^T x_i)) \right\}$$

the gradient is

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{n} y_i x_i - \underbrace{\frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)}}_{P(x_i;\beta)} x_i$$

$$= \sum_{i=1}^{n} x_i \left( y_i - P(x_i; \beta) \right) = \mathbf{X}^T (\mathbf{y} - \mathbf{p})$$

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 $i$-th element $P(x_i; \beta)$.

# Hessian of $\ell(\beta)$ I

- From

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i=1}^{n} x_{ij} \left( y_i - \underbrace{P(x_i;\beta)}_{\Lambda(\beta^T x)} \right)$$

and $\Lambda'(u) = \Lambda(u)[1 - \Lambda(u)]$, we have

$$\frac{\partial^2 \ell}{\partial \beta_j \partial \beta_k} = -\sum_{i=1}^{n} x_{ij} x_{ik} P(x_i;\beta)\left[1 - P(x_i;\beta)\right]$$

# Hessian of $\ell(\beta)$ II

- The Hessian matrix can, thus, be written as

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

where $\mathbf{W}$ an $n \times n$ diagonal matrix of weights with $i$-th diagonal element $P(x_i;\beta)\left[1 - P(x_i;\beta)\right]$.

Back