Advanced Computational Econometrics: Machine Learning

Chapter 2: Linear and Quadratic Classification

Thierry Denœux

Spring 2023

Overview

1. Introduction to classification
   - Basic notions
   - Bayes classifier
   - Voting K-NN rule
2. Linear and quadratic discriminant analysis
   - Quadratic Discriminant Analysis
   - Simplifying assumptions
   - Case of binary classification
3. Logistic regression and related models
   - Binomial logistic and probit regression
   - Multinomial logistic regression
   - Ordered probit and logit regression

Classification

- In classification problems, the response variable Y is nominal, i.e., it takes values in a finite and unordered set $\mathcal{C}$, 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.
- The elements in $\mathcal{C}$ are called classes. They are arbitrarily numbered $1, 2, \ldots, c$.
- Our goals are to:
  - Build a classifier $C : \mathbb{R}^p \to \mathcal{C}$ that predicts the class a future predictor vector X.
  - Assess the uncertainty in each classification
  - Understand the roles of the different predictors
- In this chapter, we will also see how to handle the case where Y is an ordinal variable, i.e., the elements of $\mathcal{C}$ are ordered. This learning task is called ordinal regression/classification.

Overview

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 specify:
  1. The marginal distribution of Y. We use the notation

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

     and we call π_k the prior probability of class k. We have

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

  2. The conditional probability density functions (pdfs) 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

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

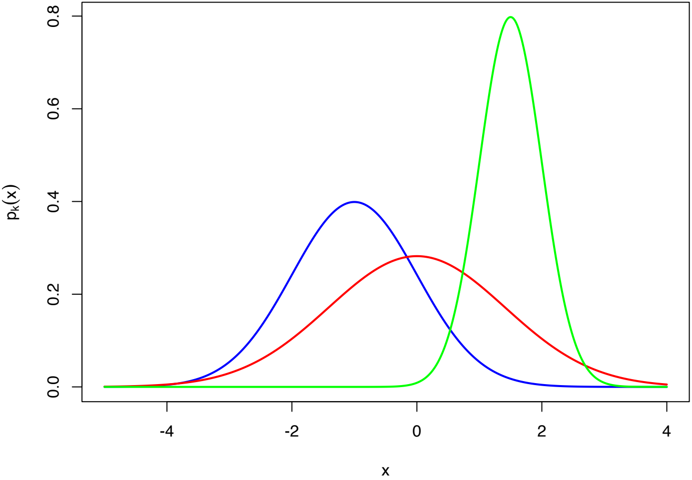
$$p_k(x) = \phi(x; \mu_k, \sigma_k)$$

where ϕ is the normal pdf, with

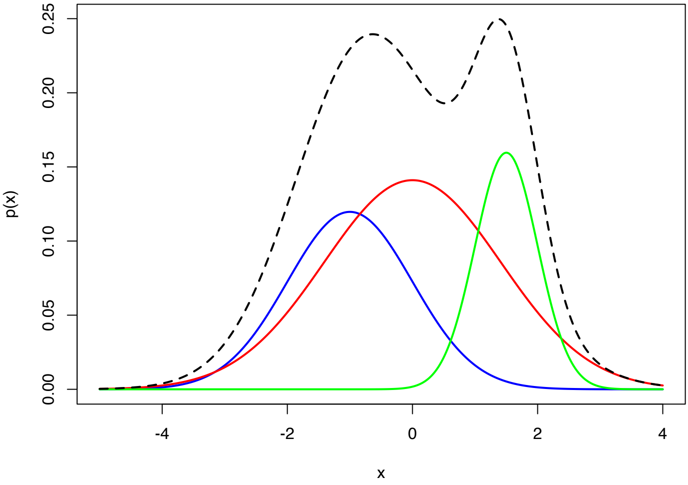
$$\mu_1 = -1, \quad \mu_2 = 0, \quad \mu_3 = 1.5$$

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

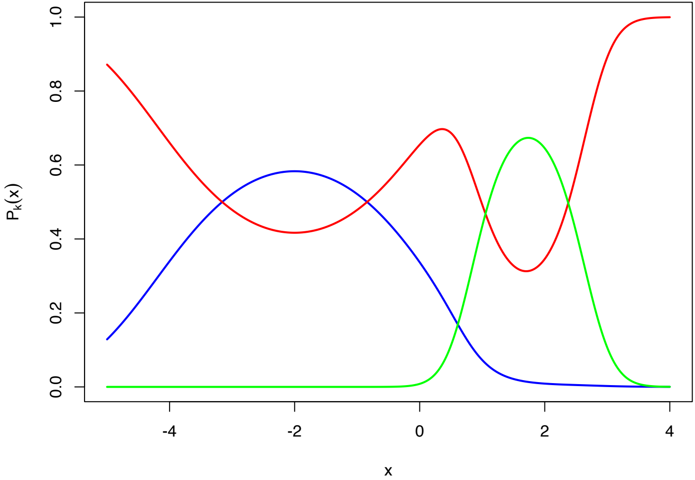
Example: conditional densities $p_k(x)$



Example: marginal density $p(x)$



Example: posterior probabilities $P_k(x)$



Overview

The Bayes classifier

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

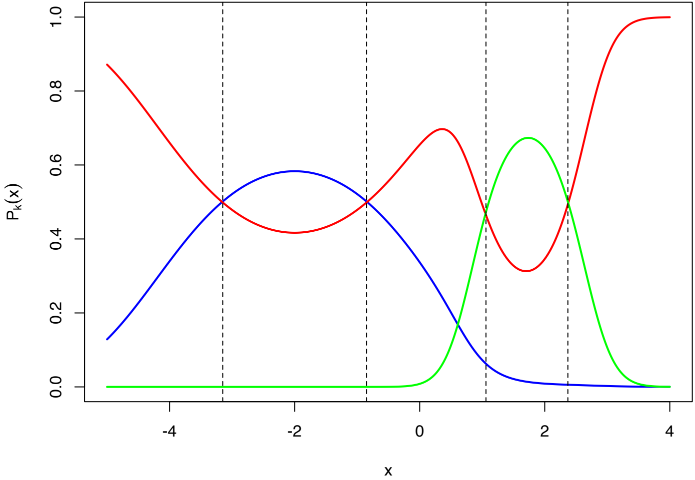
$$\begin{aligned}\mathbb{P}(\text{error} \mid X = x) &= \mathbb{P}(C(X) \neq Y \mid X = x) \\ &= 1 - \mathbb{P}(C(X) = Y \mid X = x)\end{aligned}$$

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

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

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

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

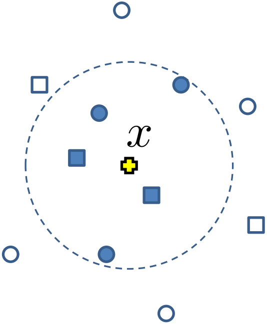
- This probability is called the Bayes error rate. It is the lowest error probability that can be achieved by a classifier. It characterizes the difficulty of the classification task.

Approximating the Bayes classifier

- The Bayes classifier is optimal but theoretical. We need practical methods to learn classifiers that will approximate the Bayes classifier.
- For this, we need to estimate the posterior probabilities $P_k(x)$.
- As in regression, we distinguish between
  - Parametric methods that postulate a model (of the densities $p_k(x)$, the posterior probabilities $P_k(x)$ or the decisions $C(x)$) depending on a limited number of parameters
  - Nonparametric methods, which make minimal assumptions about the distribution of the data.
- A widely used nonparametric method is the voting K nearest neighbor (K-NN) method.

Overview

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:

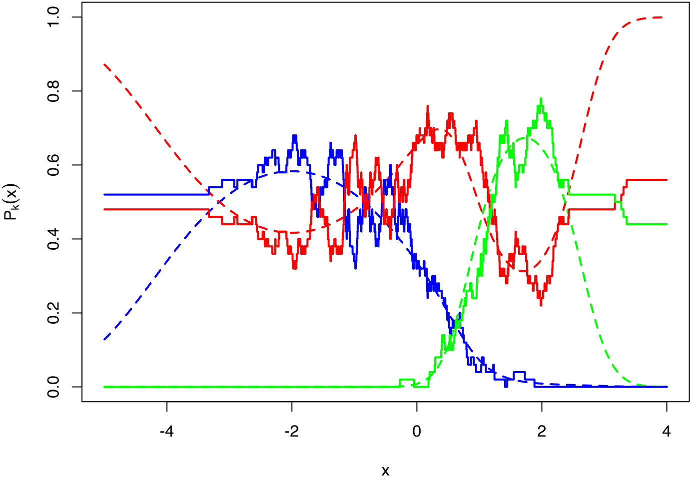
$$\widehat{P}_k(x) = \frac{1}{K}\#\{i \in \{1, \ldots, K\} : y_{(i)} = k\}$$

- 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 $K = 50$



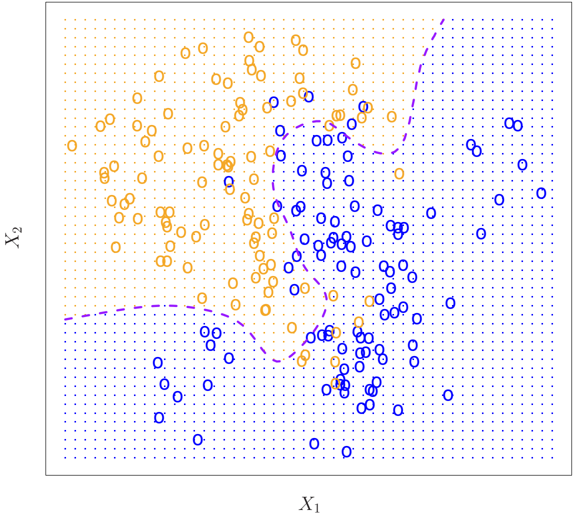
Error probability estimation

- Typically, we estimate the error probability of a classifier $C(X)$ by its error rate on a test set $\mathcal{T} = \{(x_i', y_i')\}_{i=1}^m$:

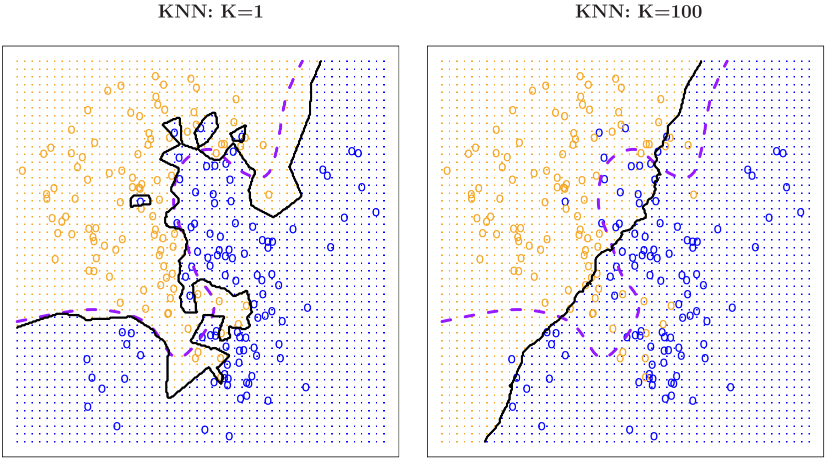
$$\text{Err}_{\mathcal{T}} = \frac{1}{m} \# \left\{ i \in \{1, \ldots, m\} : y_i' \neq C(x_i') \right\}$$

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

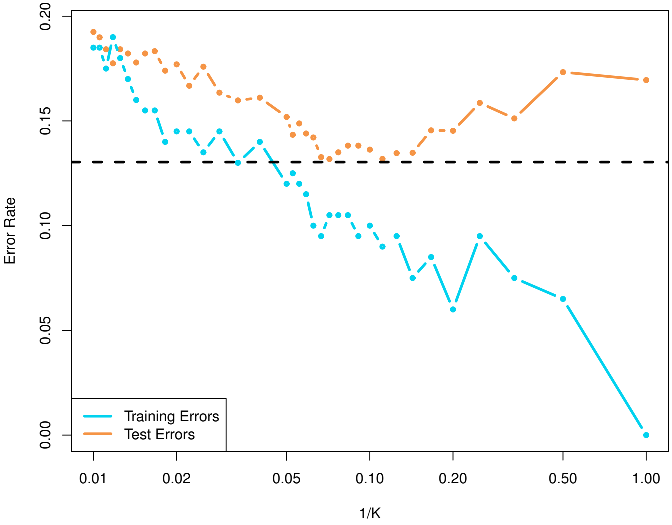
Example: simulated data and Bayes decision boundary



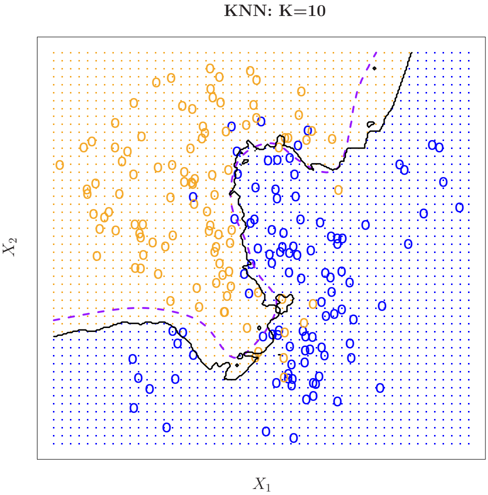
Decision boundaries for $K = 1$ and $K = 100$



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



Decision boundary for the best value of K



Overview

Decision regions and decision boundadries

- Since our classifier $C(X)$ takes values in a finite set $\mathcal{C}$, we can always divide the input space into a collection of decision regions:

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

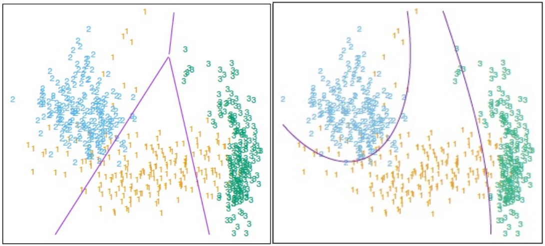
- The boundaries of these regions can be rough or smooth, depending on the prediction function.

Linear/quadratic classification

- For an important class of procedures, these decision boundaries are linear or quadratic, i.e., they have equations of the form

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

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



- Linear and quadratic methods for classification are examples of parametric methods.

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 π_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

Basic assumption

- Quadratic Discriminant Analysis (QDA) is based on the assumption that the class-conditional densities $p_k(x)$ are multivariate normal:

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

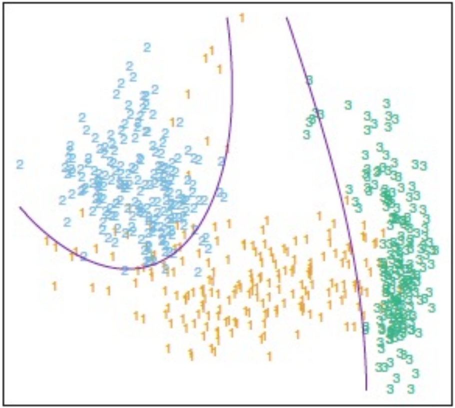
  where $\mu_k = \mathbb{E}(X \mid Y = k)$ and $\boldsymbol{\Sigma}_k = \mathrm{Var}(X \mid Y = k)$.

- The parameters of the model are the class-conditional means μ_k and covariance matrices $\boldsymbol{\Sigma}_k$, as well as the prior probabilities π_k, $k = 1, \ldots, c$.

Optimal decision boundaries

- The boundary between optimal decision regions $\mathcal{R}_k$ and $\mathcal{R}_\ell$ is defined by the equation
$$P_k(x) = P_\ell(x)$$
- Applying the logarithm to both sides and using the Bayes' theorem
$$P_k(x) \propto p_k(x)\pi_k,$$
we get
$$\log p_k(x) + \log \pi_k = \log p_\ell(x) + \log \pi_\ell \tag{1}$$
- Now
$$\log p_k(x) = -\frac{1}{2}(x-\mu_k)^T \boldsymbol{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \text{cst} \tag{2}$$
- From (1) and (2), the boundary equation can be put in the form $x^T Q x + \beta^T x + \beta_0 = 0$: the decision boundary is a quadric.

Example



Estimation of parameters

- Let $\boldsymbol{\theta}$ be the vector of all parameters.
- Assumption: the sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ is independent and identically distributed (iid).
- The likelihood function is

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} p(x_i, y_i) = \prod_{i=1}^{n} \underbrace{p(x_i \mid y_i)}_{\prod_{k=1}^{c} p_k(x_i)^{y_{ik}}} \underbrace{p(y_i)}_{\prod_{k=1}^{c} \pi_k^{y_{ik}}}$$

$$= \prod_{i=1}^{n} \prod_{k=1}^{c} \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k)^{y_{ik}} \pi_k^{y_{ik}}$$

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

Maximum likelihood estimates

- The MLEs are

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

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

with $n_k = \sum_{i=1}^{n} y_{ik}$.

- These estimators are consistent (they converge to the true parameter values when the sample size tends to infinity).

Implementation of QDA

- To implement QDA, we plug-in the parameter estimates into the expressions of the posterior probabilities. The decision rule is then

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

- We actually only need to compute monotonic transformations of the estimated posterior probabilities $\widehat{P}_k(x)$, called discriminant functions (DFs).
- We get quadratic DFs by applying a logarithmic transformation:
  - Case $c = 2$: we only need one DF $\delta(x) = \log \widehat{P}_1(x) - \log \widehat{P}_2(x)$ and the decision rule is

$$C(x) = \begin{cases} 1 & \text{if } \delta(x) > 0 \\ 2 & \text{otherwise.} \end{cases}$$

  - Case $c > 2$: we need c DFs $\delta_k(x) = \log \widehat{P}_k(x)$ and the decision rule is

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

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.

Example (continued)

```
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,]

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

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

0.1166
```

Confusion matrix

```
> print(perf)
```

	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	182	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2	1	0	0	3	0
B	0	165	0	3	2	1	0	4	1	0	0	0	0	0	2	0	0	8	4	0	0	0	0	0	0	0
C	0	0	164	0	2	1	6	0	0	0	5	1	0	0	3	0	0	0	0	0	1	0	0	0	0	0
D	0	3	0	196	0	0	0	0	0	0	1	0	3	0	3	1	1	4	3	3	1	1	0	0	0	0
E	0	1	3	0	163	1	7	1	0	0	1	3	0	0	0	0	1	1	0	0	0	0	0	2	0	3
F	0	1	0	1	1	163	2	1	2	0	0	0	0	4	0	2	0	1	0	6	0	0	1	0	2	0
G	0	3	4	1	0	3	145	0	0	0	3	3	0	0	7	0	1	3	7	0	0	0	0	0	0	0
H	0	3	1	10	2	4	3	108	0	0	14	0	1	0	3	1	0	6	0	0	1	1	0	4	3	0
I	0	1	0	4	1	5	0	0	157	7	1	0	0	1	0	0	0	1	3	0	0	0	0	2	0	1
J	0	0	0	1	0	1	0	3	3	145	0	0	0	1	1	0	2	0	2	0	0	0	0	0	0	2
K	0	0	5	1	1	0	0	5	0	0	161	0	0	0	0	0	0	12	1	0	0	1	0	5	0	0
L	0	0	0	0	1	0	0	3	0	0	0	189	0	0	0	0	1	2	10	0	0	0	0	2	1	0
M	2	1	0	0	0	0	0	0	0	0	0	0	181	0	0	0	0	1	0	0	0	1	3	0	0	0
N	0	0	0	4	0	1	0	2	0	0	1	0	0	163	2	1	0	2	0	0	0	0	1	0	1	0
O	1	0	0	2	0	0	1	3	0	0	0	0	2	0	173	0	9	1	0	0	0	0	1	0	0	0
P	0	0	0	0	0	11	2	1	0	0	1	1	0	0	0	186	2	2	0	0	0	0	0	0	1	0
Q	2	0	0	0	0	0	0	0	0	0	0	1	0	0	9	0	179	1	5	0	0	0	0	0	0	0
R	0	5	1	3	0	0	1	2	0	0	2	0	0	0	0	0	1	180	1	0	1	0	0	0	0	0
S	0	5	0	1	7	4	0	0	1	1	1	0	0	0	0	0	0	0	178	0	0	0	0	0	0	6
T	1	0	0	0	5	3	2	2	0	0	1	1	0	0	0	0	0	3	1	168	5	0	0	1	2	0
U	1	0	0	0	0	0	1	0	0	1	1	0	3	0	3	0	0	0	0	0	210	1	5	0	0	0
V	0	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	181	3	0	3	0
W	0	0	0	0	0	0	0	1	0	0	1	0	4	1	2	0	0	0	0	0	2	0	169	0	1	0
X	0	0	0	0	2	0	0	5	3	1	5	1	0	0	0	0	1	1	1	1	3	0	0	154	1	1
Y	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	2	0	0	1	4	1	29	0	0	162	0
Z	2	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	2	0	4	2	1	0	0	1	0	195

Overview

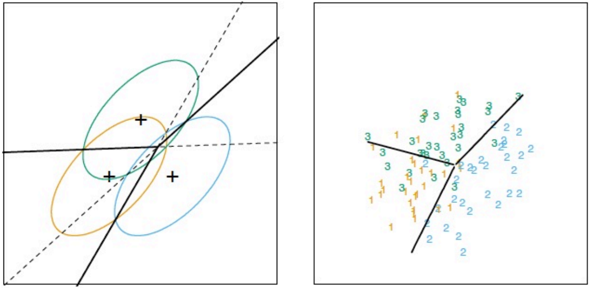
Number of parameters for QDA

- The number of parameters for QDA is $c[p + p(p+1)/2] + c - 1$ (c means, c covariance matrices and $c-1$ prior probabilities).
- This number is quadratic in p: the method becomes impractical when p is large, and QDA may perform badly when p is large and n is small.
- We can decrease the number of parameters to estimate by making simplifying assumptions. We will consider two such assumptions:
  1. Equality of covariance matrices (homoscedasticity) $\rightarrow$ Linear Discriminant Analysis (LDA)
  2. Conditional independence of the predictors X_j given the class variable Y $\rightarrow$ Naive Bayes classifiers

Linear Discriminant Analysis (LDA)

- LDA is based on the assumption that the class-conditional covariance matrices are equal:
$$\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \quad \text{for all } k.$$
- In that case, the equation of the optimal boundary between regions $\mathcal{R}_k$ and $\mathcal{R}_\ell$,
$$\log p_k(x) + \log \pi_k = \log p_\ell(x) + \log \pi_\ell \tag{3}$$
is linear. Indeed, we now have
$$\log p_k(x) = -\frac{1}{2}(x-\mu_k)^T \boldsymbol{\Sigma}^{-1}(x-\mu_k) + \text{cst} \tag{4}$$
$$= \mu_k^T \boldsymbol{\Sigma}^{-1} x - \frac{1}{2}\mu_k^T \boldsymbol{\Sigma}^{-1}\mu_k + \text{cst} \tag{5}$$
- The DF and the decision boundaries are now linear in x.

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 size 30 drawn from each distribution, and the fitted LDA decision boundaries.

Estimation of parameters

- The MLEs are

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

where, as before $\widehat{\boldsymbol{\Sigma}}_k$ is the sample covariance matrix in class k:

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

and $n_k = \sum_{i=1}^{n} y_{ik}$.

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

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

LDA in R

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

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

perf <-table(letter.test$V1,pred.letters.lda$class)
1-sum(diag(perf))/ntst
0.2996
```

Comparing classifiers: McNemar's test

- The test error rate was 0.1166 for QDA, and it is 0.2996 for LDA. Is this difference statistically significant?
- To answer such a question, we typically use McNemar's test for 2×2 contingency tables.
- We consider the following table:

	classifier 2 wrong	classifier 2 correct
classifier 1 wrong	n_{00}	n_{01}
classifier 1 correct	n_{10}	n_{11}

Comparing classifiers: McNemar's test (continued)

- Under the null hypothesis that the error probabilities of the two classifiers are equal, the statistics

$$D^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}}$$

  is distributed approximately as χ^2 with 1 degree of freedom.
- The p-values is $p = \mathbb{P}_{H_0}(\chi^2_1 \geq d^2)$.
- Remark: when comparing more than two classifiers, we have more chance of rejecting the null hypothesis for at least one pair of classifiers. To address this problem, we can use the Bonferroni correction: we reject the null hypothesis at level α for any two classifiers if $p \leq \alpha/m$, where m is the number of classifier pairs.

McNemar's test in R

```
correct.lda<-letter.test$V1==pred.letters.lda$class
correct.qda<-letter.test$V1==pred.letters.qda$class
mcnemar.test(correct.lda,correct.qda)

McNemar's Chi-squared test with continuity correction

data: correct.lda and correct.qda
McNemar's chi-squared = 767.12, df = 1, p-value < 2.2e-16
```

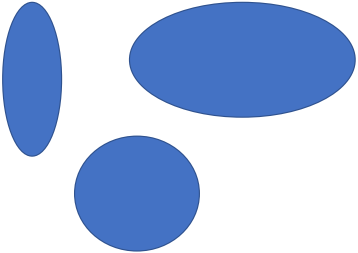
Naive Bayes classifiers

- In LDA and QDA, we need to estimate covariance matrices with $p(p+1)/2$ parameters, which can yield poor results (or can even be unfeasible) when p is very large.
- Starting from the QDA model, we get a simpler model by assuming that the covariance matrices $\boldsymbol{\Sigma}_k$ are diagonal:

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

  where $\sigma_{kj}^2 = \mathrm{Var}(X_j \mid Y = k)$ (see next slide).
- We get a naive QDA classifier, a special kind of naive Bayes classifier.

Naive QDA model



Conditional independence assumption

- The assumption that the covariance matrices are diagonal 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 σ^2_{kj} of X_j conditionally on $Y = k$ is estimated by

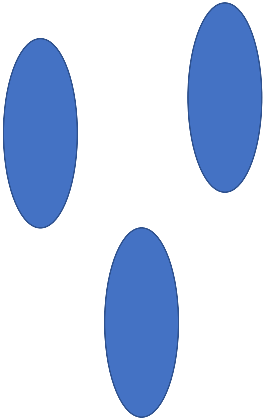
$$\widehat{\sigma}^2_{kj} = \frac{1}{n_k} \sum_{i=1}^{n} y_{ik} (x_{ij} - \widehat{\mu}_{kj})^2.$$

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

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

This model can be called "Naive LDA" (see next slide).

Naive LDA model



Advantages of Naive Bayes classifiers

- In spite of their simplicity, naive Bayes classifiers often (but not always) have very good performances, especially when the number p of predictors is large.
- They can accommodate mixed feature vectors (qualitative and quantitative). If X_j is qualitative, we can 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

0.3554
# Comparison with LDA
correct.naive<-letter.test$V1==pred.letters.naive
mcnemar.test(correct.lda,correct.naive)

McNemar's Chi-squared test with continuity correction

data: correct.lda and correct.naive
McNemar's chi-squared = 83.731, df = 1, p-value < 2.2e-16
```

Comparison of the different models

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	$cp+p+c-1$

- 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 often outperform other methods when p is very large.

Example

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

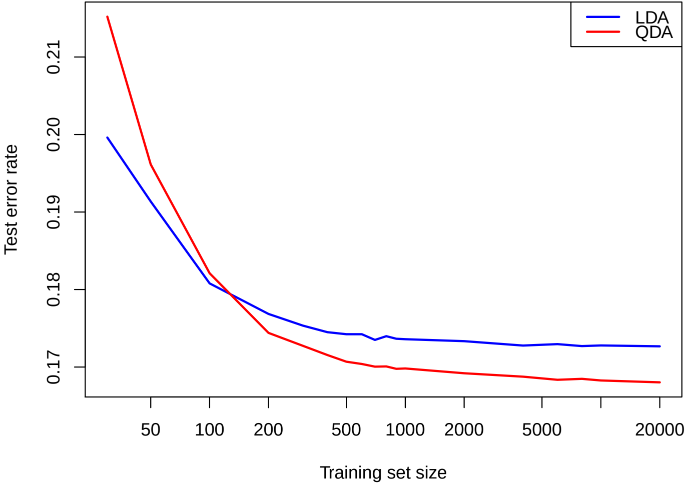
$$\pi_1 = \pi_2 = 0.5$$

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

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

- LDA and QDA classifiers were trained using training sets of different sizes between 30 and 20,000, and their error probability 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

Case $c = 2$: fixing the threshold

- From (5), in the case of $c = 2$ classes, LDA assigns x to class 2 if

$$\log \widehat{P}_2(x) > \log \widehat{P}_1(x) \Leftrightarrow (\widehat{\mu}_2 - \widehat{\mu}_1)^T \widehat{\boldsymbol{\Sigma}}^{-1} x > s,$$

  where the threshold s depends on the estimated parameters, including 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.
- In general, a decision is made by comparing a DF $\delta(x)$ to some threshold s.
- 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	N	P
N	true negative (TN)	false positive (FP)
P	false negative (FN)	true positive (TP)

- 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.

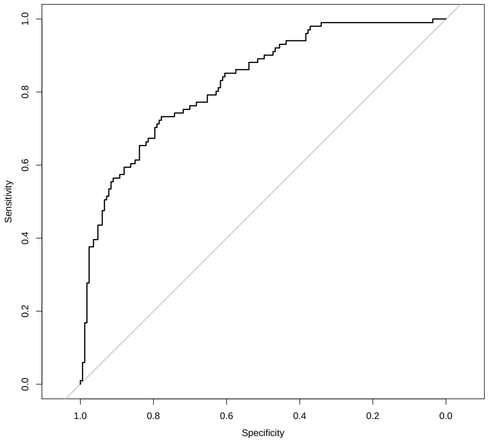
LDA of the Pima dataset

```
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

Overview

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

$$P(x) = \beta_0 x_0 + \beta_1 x_1 + \ldots + \beta_p x_p = \beta^T x,$$

  where x is the augmented feature vector with $x_0 = 1$. However, this is not suitable because $\beta^T x$ can take any value in $\mathbb{R}$, whereas $P(x) \in [0, 1]$.
- A better idea is to assume that Y depends on a latent (unobserved) continuous variable Y^*, which is linearly related to x.

Model

- Assume that

$$Y^* = \beta^T x + \epsilon,$$

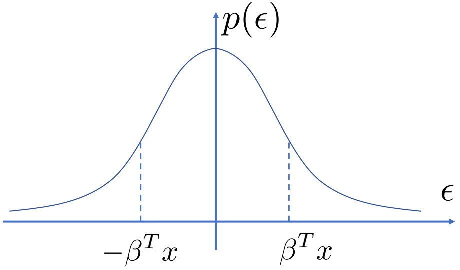
where ϵ is a random error term with 0 mean and cumulative distribution function (cdf) F, and

$$Y = \begin{cases} 1 & \text{if } Y^* > 0 \\ 0 & \text{otherwise} \end{cases}$$

- We then have

$$P(x) = \mathbb{P}(Y = 1 \mid x) = \mathbb{P}(Y^* > 0 \mid x) = \mathbb{P}(\epsilon > -\beta^T x).$$

Model (continued)



- If we assume the distribution of ϵ to be symmetric, then

$$\mathbb{P}(\epsilon > -\beta^T x) = \mathbb{P}(\epsilon \leq \beta^T x) \quad \text{and} \quad \boxed{P(x) = F(\beta^T x)}$$

- Different choices of F give us different models. The decision boundary is linear, with equation

$$P(x) = \frac{1}{2} \Leftrightarrow \beta^T x = F^{-1}(0.5) = 0$$

Logit model

- In the logit model, we assume that ϵ has a standard logistic distribution with cdf

$$F(u) = \Lambda(u) = \frac{\exp(u)}{1+\exp(u)}.$$

- We then have

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

- The log-odds ratio is linear in x:

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

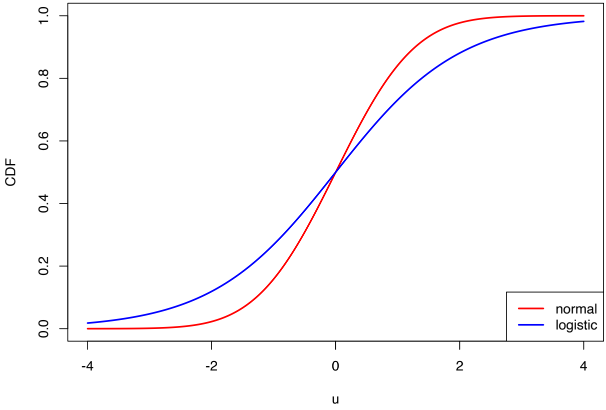
Probit model

- In the probit model, we assume that ϵ has a standard normal distribution with cdf Φ. We then have

$$P(x) = \Phi(\beta^T x).$$

- In practice, the two models usually give very similar results.
- Logistic regression based on the logit model is more popular in ML.

Plot of the logistic and normal cdfs



Conditional likelihood function

- Logit and probit 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

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

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

Conditional log-likelihood (logit model)

- The conditional log-likelihood for the logit model is

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

Proof

- This function is non linear and the score equation $\frac{\partial \ell}{\partial \beta} = 0$ does not have a closed-form solution: we need to use an iterative nonlinear optimization procedure such as the Newton-Raphson algorithm.
- As the log-likelihood function is concave, it has only one maximum and the convergence of the Newton-Raphson algorithm is guaranteed.

Update equation (logit model)

- 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 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 result makes it possible to compute confidence intervals and to test the significance of the coefficients β_j.
- Similar results hold for probit regression.

Binomial logistic regression in R

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

```
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

Number of Fisher Scoring iterations: 5
```

Prediction (logit model)

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

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

    FALSE  TRUE
 0    158    14
 1     41    55
```

The error rate is $(14 + 41)/268 \approx 0.21$.

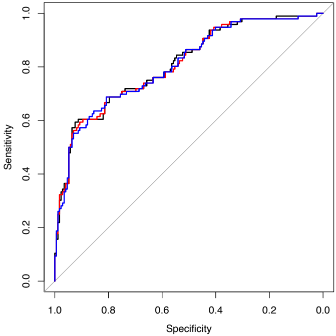
Binomial probit regression in R

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

```
> summary(probit.fit)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5488  -0.7291  -0.3691   0.7558   2.9030

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.0692691  0.5018665 -10.101  < 2e-16 ***
pregnant     0.0674591  0.0247598   2.725  0.00644 **
glucose      0.0189866  0.0025196   7.536 4.86e-14 ***
BP          -0.0107108  0.0042744  -2.506  0.01222 *
skin         0.0038887  0.0050731   0.767  0.44336
insulin     -0.0011300  0.0006979  -1.619  0.10545
bmi          0.0658705  0.0113925   5.782 7.38e-09 ***
diabetes     0.5366801  0.2167489   2.476  0.01328 *
age          0.0108750  0.0071087   1.530  0.12606
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 643.65  on 499  degrees of freedom
Residual deviance: 462.67  on 491  degrees of freedom
AIC: 480.67
```

ROC curves: comparison with LDA

```
logit<-predict(glm.fit,newdata=pima.test,type='link')
probit<-predict(probit.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$class,logit)
roc_probit<-roc(pima.test$class,probit)
plot(roc_glm,add=TRUE,col='red')
plot(roc_probit,add=TRUE,col='blue')
```



Overview

Model

- Multinomial logistic regression extends binomial logistic regression to $c > 2$ by assuming the following model for the posterior probabilities $P_k(x) = \mathbb{P}(Y = k \mid X = x)$:

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

- However, there is indeterminacy in the model, because the probabilities are unchanged if we add a constant vector α to all β_k's:

$$\frac{\exp((\beta_k + \alpha)^T x)}{\sum_{l=1}^{c} \exp((\beta_l + \alpha)^T x)} = \frac{\exp(\beta_k^T x)}{\sum_{l=1}^{c} \exp(\beta_l^T x)}$$

- To remove this indeterminacy, we set $\beta_1 = 0$.

Model (continued)

- We then have
$$P_1(x) = \frac{1}{1 + \sum_{l=2}^{c} \exp(\beta_l^T x)}$$
and
$$P_k(x) = \frac{\exp(\beta_k^T x)}{1 + \sum_{l=2}^{c} \exp(\beta_l^T x)}, \quad k = 2, \ldots, c.$$
- The log-odds ratios for class k vs. class 1 are still linear in x:
$$\boxed{\log \frac{P_k(x)}{P_1(x)} = \beta_k^T x}$$

Learning

- The conditional likelihood for the multinomial model is

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

- 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

0.285
# Comparison with LDA correct.log<-letter.test$V1==pred.letters.log
mcnemar.test(correct.lda,correct.log)

McNemar's Chi-squared test with continuity correction

data: correct.lda and correct.log
McNemar's chi-squared = 6.4881, df = 1, p-value = 0.01086
```

Overview

Ordinal classification/regression

- In ordinal regression/classification, the response Y is an ordinal variable, i.e., it takes values in a finite ordered set.
- For instance, a variable "Customer satisfaction" may take values in the set $\{\text{High}, \text{Medium}, \text{Low}\}$.
- To solve ordinal regression problems, we can still use classification methods, but the results will often not be optimal because the ordering relation between the values of Y is ignored.
- A much better option to use a specific method such as ordered probit or logit regression.

Ordered logit and probit models

- As in the binomial logit and probit models, we assume the existence of a latent variable Y^* linearly related to x:

$$Y^* = \beta^T x + \epsilon$$

- We now assume that Y is determined by Y^* as follows:

$$Y = \begin{cases} 1 & \mu_0 < Y^* \leq \mu_1 \\ 2 & \mu_1 < Y^* \leq \mu_2 \\ \vdots & \\ c & \mu_{c-1} < Y^* < \mu_c \end{cases}$$

where $-\infty = \mu_0 < \mu_1 < \ldots < \mu_{c-1} < \mu_c = +\infty$ are unknown parameters.

Ordered logit and probit models (continued)

- The ordered logit and probit models correspond to different assumptions about the distribution of ϵ: respectively, logistic ($F = \Lambda$) or normal ($F = \Phi$).
- The conditional log-likelihood function is

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik} \log P_k(x) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik} \log\left[F(\mu_k - \beta^T x) - F(\mu_{k-1} - \beta^T x)\right],\end{aligned}$$

  with $\theta = (\beta, \mu_1, \ldots, \mu_{c-1})$.
- The MLE of θ can be found by an iterative nonlinear optimization algorithm.

Example: Housing dataset

- Package `MASS`, 72 rows and 5 variables.
- Variables:

  Sat: Satisfaction of householders with their present housing circumstances (High, Medium or Low, ordered factor).

  Infl: Perceived degree of influence householders have on the management of the property (High, Medium, Low).

  Type: Type of rental accommodation, (Tower, Atrium, Apartment, Terrace).

  Cont: Contact residents are afforded with other residents, (Low, High).

  Freq: Frequencies: the numbers of residents in each class.

Ordered logit regression in R

```
library("MASS")
house.logit <- polr(Sat ~ Infl + Type + Cont, weights = Freq,
data = housing, method = "logistic")
```

```
> summary(house.logit, digits = 3)

Re-fitting to get Hessian

Call:
polr(formula = Sat ~ Infl + Type + Cont, data = housing, weights = Freq,
    method = "logistic")

Coefficients:
               Value Std. Error t value
InflMedium     0.566     0.1047    5.41
InflHigh       1.289     0.1272   10.14
TypeApartment -0.572     0.1192   -4.80
TypeAtrium    -0.366     0.1552   -2.36
TypeTerrace   -1.091     0.1515   -7.20
ContHigh       0.360     0.0955    3.77

Intercepts:
            Value   Std. Error t value
Low|Medium  -0.496  0.125      -3.974
Medium|High  0.691  0.125       5.505

Residual Deviance: 3479.149
AIC: 3495.149
```

Ordered probit regression in R

```
house.probit <- polr(Sat ~ Infl + Type + Cont, weights = Freq,
data = housing, method = "probit")
```

```
> summary(house.probit, digits = 3)

Re-fitting to get Hessian

Call:
polr(formula = Sat ~ Infl + Type + Cont, data = housing, weights = Freq,
    method = "probit")

Coefficients:
               Value Std. Error t value
InflMedium     0.346     0.0641    5.40
InflHigh       0.783     0.0764   10.24
TypeApartment -0.348     0.0723   -4.81
TypeAtrium    -0.218     0.0948   -2.30
TypeTerrace   -0.664     0.0918   -7.24
ContHigh       0.222     0.0581    3.83

Intercepts:
            Value  Std. Error t value
Low|Medium  -0.300  0.076     -3.937
Medium|High  0.427  0.076      5.585

Residual Deviance: 3479.689
AIC: 3495.689
```

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

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

(The quadratic term $x^T \widehat{\boldsymbol{\Sigma}}^{-1} x$ is absorbed in the constant because if does not depend on k).

Back

Discriminant functions of QDA

From

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

we get

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

(The quadratic terms $x^T \widehat{\boldsymbol{\Sigma}}_k^{-1} x$ now depend on k).

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

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

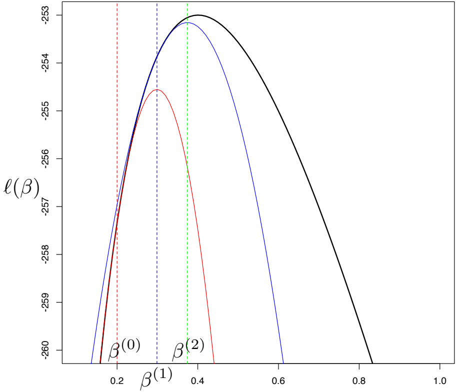
$$\ell(\beta) = \sum_{i=1}^{n} \left\{ y_i \beta^T x_i - \log[1+\exp(\beta^T x)] \right\}$$

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. β, 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 **y** denote the vector of y_i values, **X** the $n \times (p+1)$ matrix of x_i values, **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

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

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