Classification

Supervised vs unsupervised clustering

- Cluster analysis: Classes are not known a-priori.
- Classification: Classes are defined a-priori for some objects
  - Sometimes called “supervised clustering”
  - Extract useful features based on known class labels that separate classes in “training set”
  - Assign new objects to classes based on rules developed on the training set

Different Classification methods

- Statistical methods: often aim to classify as well as to identify “marker features (genes)” that characterize different classes
  - Linear discriminant analysis
  - Nearest neighbors
  - Logistic regression
  - Classification and regression tree
- Computer science methods: do not emphasize on parsimony or interpretation
  - Bayesian network
  - Neural network
  - Support vector machine

R packages

- FNN: fast nearest neighbor
- Class: k nearest neighbor, self organizing map
- randomForest
- e1071: support vector machine
- MASS: lda (linear discriminant analysis)
- MLInterface: a useful package that provides an interface to many classification packages

General notation for classification

- Classes (predefined): $C_1, ..., C_K$
- Feature vector (predictor variables) $X = (X_1, ..., X_G) \in \mathcal{X}$
- Response $Y \in \{1, ..., K\}$
- Classifier: $C : \mathcal{X} \rightarrow \{1, ..., k\}$ defines a disjoint partition of feature space $\mathcal{X} = A_1 \cup A_2 \cup ... \cup A_K$
- Classifier is developed (trained) on *learning* or *training* set: $\mathcal{L} = \{(x_1, y_1), , , , (x_n, y_n)\}$. (So $(C(., \mathcal{L})$ is a random variable).
- $X_{G \times n}$ is matrix of gene expression data

Logistic discriminant function

- Binary classification. Logistic regression model $logitp(Y = 1|X = x) = \alpha + \beta' x$.
- Predicted probability: $\hat{p}(1|x) = \frac{e^{\hat{\alpha}+\hat{\beta}' x}}{1+e^{\hat{\alpha}+\hat{\beta}' x}}$ Classification rule based on predicted probability.
- Extension to multiple categories and beyond linearity: $logp(k|x) - logp(1|x) = g_k(x, \theta)$, with $g_1(x, \theta) = 0$.
- $\hat{p}(k|x) = \frac{e^{g_k(x,\hat{\theta})}}{\sum e^{g_k(x,\hat{\theta})}}$
- Bayes rule $C_B(x) = argmax_k \hat{p}(k|x)$

Logistic discriminant function

- Classes C_1=1(yes) C_2=0 (No)
- Feature vector x
- Response Y $Y \in \{0,1\}$
- Learning set $\mathcal{L} = \{(x_1, y_1), , , , (x_n, y_n)\}$
- Classifier C:

$$\hat{p}(1|x) = \frac{e^{\hat{\alpha}+\hat{\beta}' x}}{1+e^{\hat{\alpha}+\hat{\beta}' x}}$$ >0.5 → class 1.

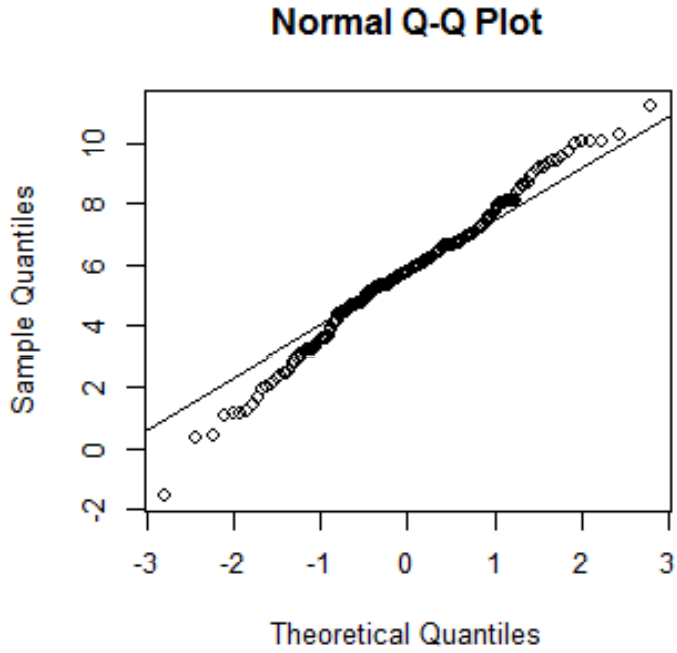
Logistic regression vs logistic discriminant classification

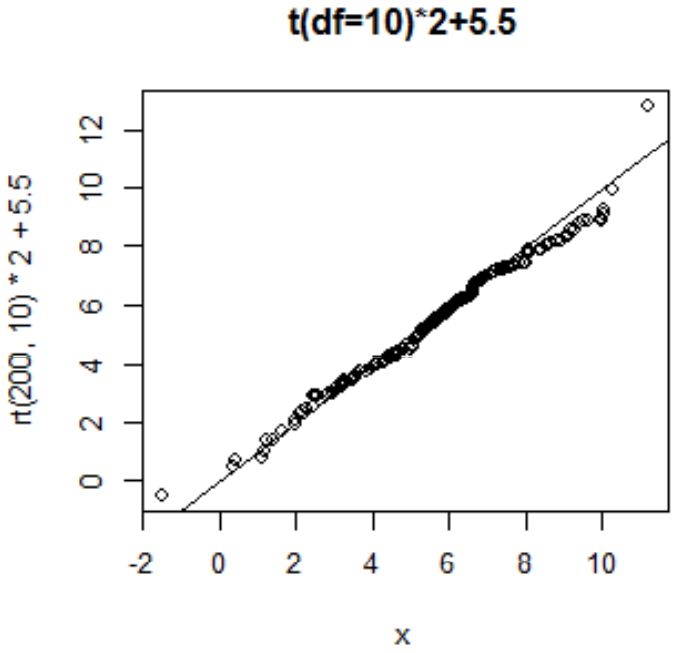
- The purpose of logistic regression is often not to do classification, though you can use it to classify
  - Often you simply want to study if there is an association between the outcome and your covariate
  - Having a statistically significant association does not mean this covariate will be an informative predictor
  - Increasing sample size will improve your statistical power in detecting a real association
  - Increasing the sample size will not necessarily help classification success, if the predictor is weak

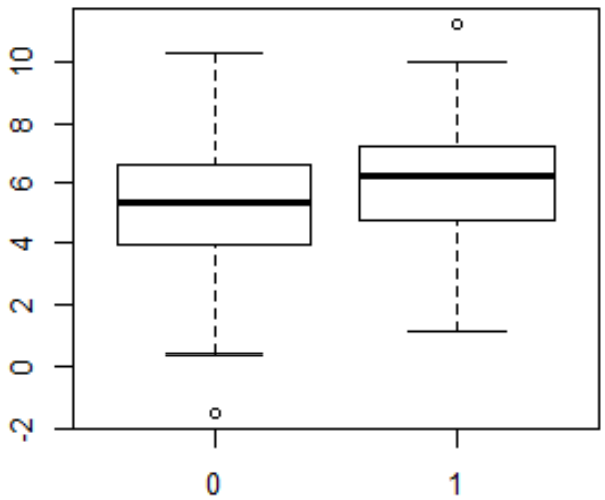
Toy example for logistic regression

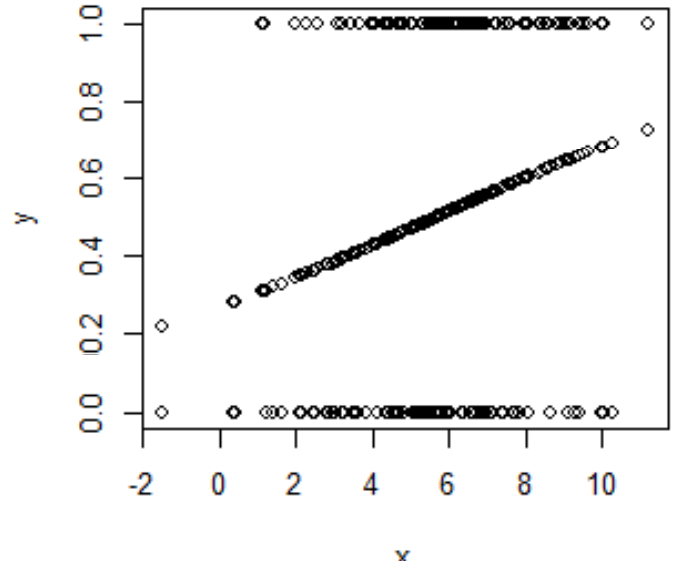
```
>set.seed(2014)
>N1=100;N2=100
>x<-c(rnorm(N1,5,2),rnorm(N2,7,2))
>y<-rep(c(0,1),c(N1,N2))
>par(mfrow=c(2,2))
>qqnorm(x);qqline(x) ## marginally you can’t easily tell
>Mclust(x)
'Mclust' model object:
 best model: univariate normal (X) with 1 components
Warning messages:
1: In summary.mclustBIC(Bic, data, G = G, modelNames = modelNames) :
  best model occurs at the min or max # of components considered
2: In Mclust(x) : optimal number of clusters occurs at min choice
>boxplot(x~y)
>plot(y~x)
>logistic1<-glm(y~x,family="binomial")
>summary(logistic1)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.00258    0.42418  -2.364   0.0181 *
x            0.17647    0.07012   2.517   0.0118 *
---
```


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
-2
0
2
4
6
8
10
-3
-2
-1
0
1
2
3


t(df=10)*2+5.5
rt(200, 10) * 2 + 5.5
x
0
2
4
6
8
10
12
-2
0
2
4
6
8
10


-2
0
2
4
6
8
10
0
1


y
x
0.0
0.2
0.4
0.6
0.8
1.0
-2
0
2
4
6
8
10

How well does it classify?

If we choose 0.5 as a cutoff

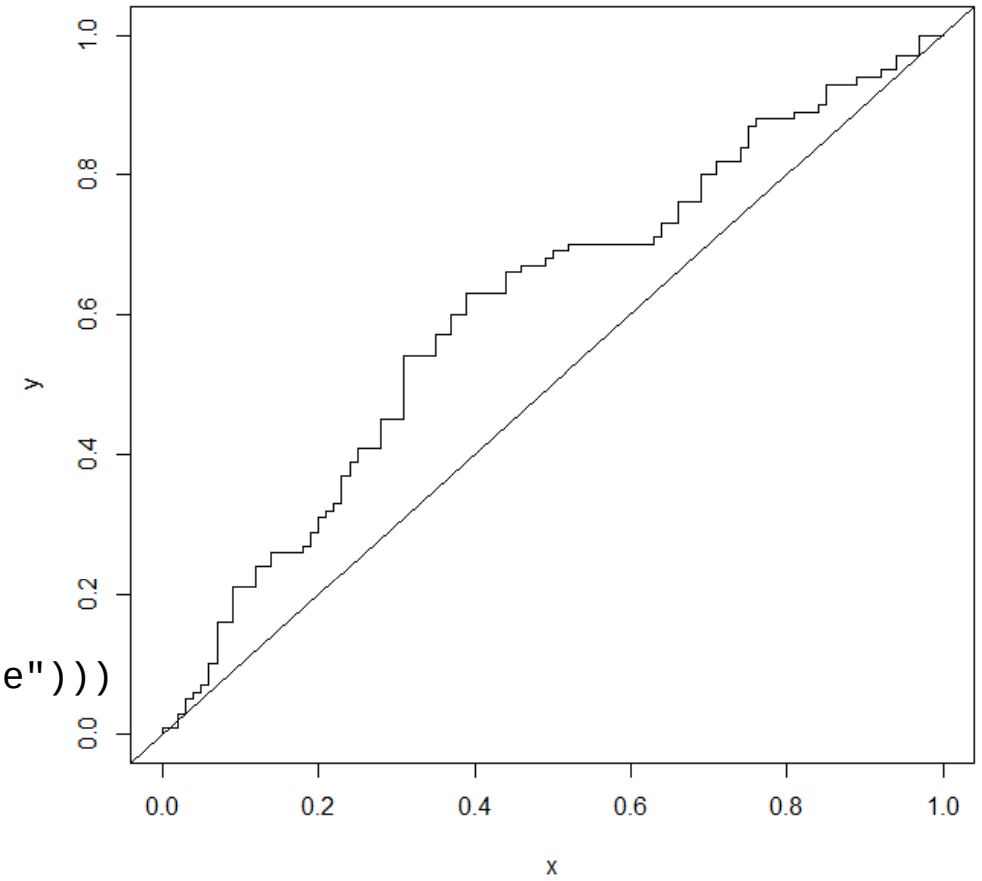
```
y1=as.numeric(predict(logistic1,type="response")>0.5)
table(y,y1)
   y1
y    0  1
  0 61 39
  1 37 63
```

Is it better than random guessing?

What if we choose other cutoffs?

ROC (*Receiver operating characteristic*)
curve

```
library(ROC)
plot(rocdemo.sca(truth=y,
data=predict(logistic1,type="response")))
abline(0,1)
```



– When the classes are well separated, the parameter estimates are unstable
– The link function, *logit*, in logistic regression is popular than other link functions such as *probit* and *clog-log*, because of the nice interpretation of the logit link and because of the symmetry in prospective and retrospective models. It is not chosen for its predictive ability.

Constructing and evaluating classifiers

Training data: for constructing the classifiers

Cross-validation: often cross-validation is used in training process

-Leave one out: asymptotically equivalent to

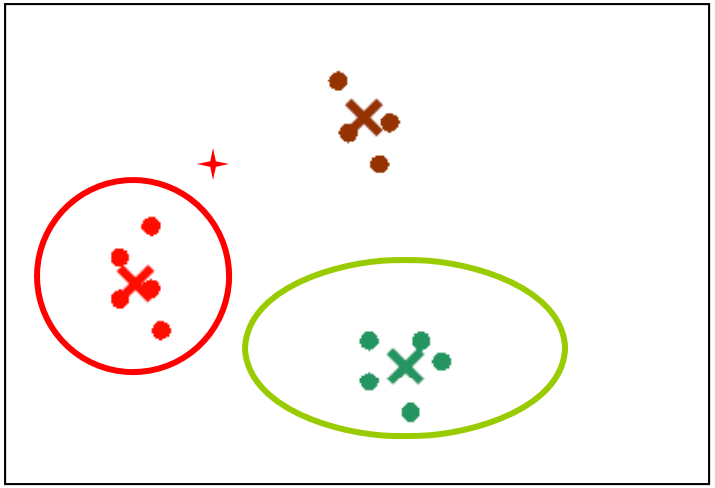
-Leave n_ν out

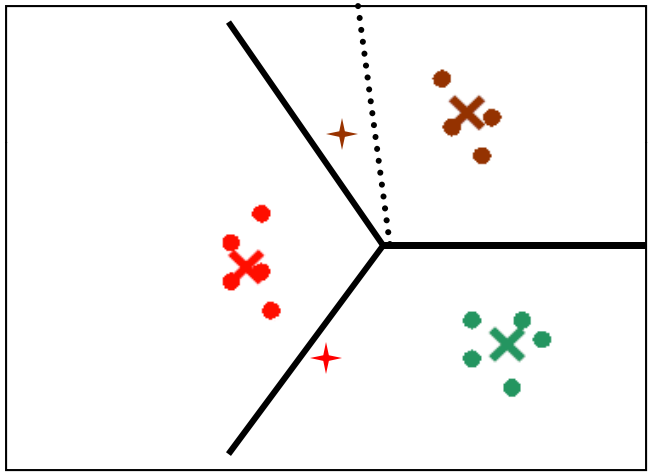
- (see Linear Model Selection by Cross-Validation, Shao J 1993 JASA for details)

Test data: a separate set of data used to evaluate the performance

Toy example

Space: 2 genes, finite range of expression measure.





Decision theory

- Distribution of predictors: $p_k(x) = f(x|Y = k)$.
- Prior: $\pi_k = Pr(Y = k)$
- Posterior $p(k|x) = \frac{\pi_k p_k(x)}{\sum \pi_l p_l(x)}$
- Loss function: $L(k, l) =$ loss if k is erroneously classified as l, where $k, l \in \{1, ..., \}$. Note $L(h, h) = 0$. Common loss function $L(k, l) = 1, if\ k \neq l$.
- Risk function:
  $R(C) = E(L(Y, C(X)) = \sum \pi_k E(L(k, C(X))|Y = k) =$
  $\sum \pi_k \int L(k, C(x)) p_k(x) dx$
- If loss is 0-1 then risk is *misclassification rate*:
  $R(C) = \sum \pi_k \int_{\{C(x) \neq k\}} p_k(x) dx$

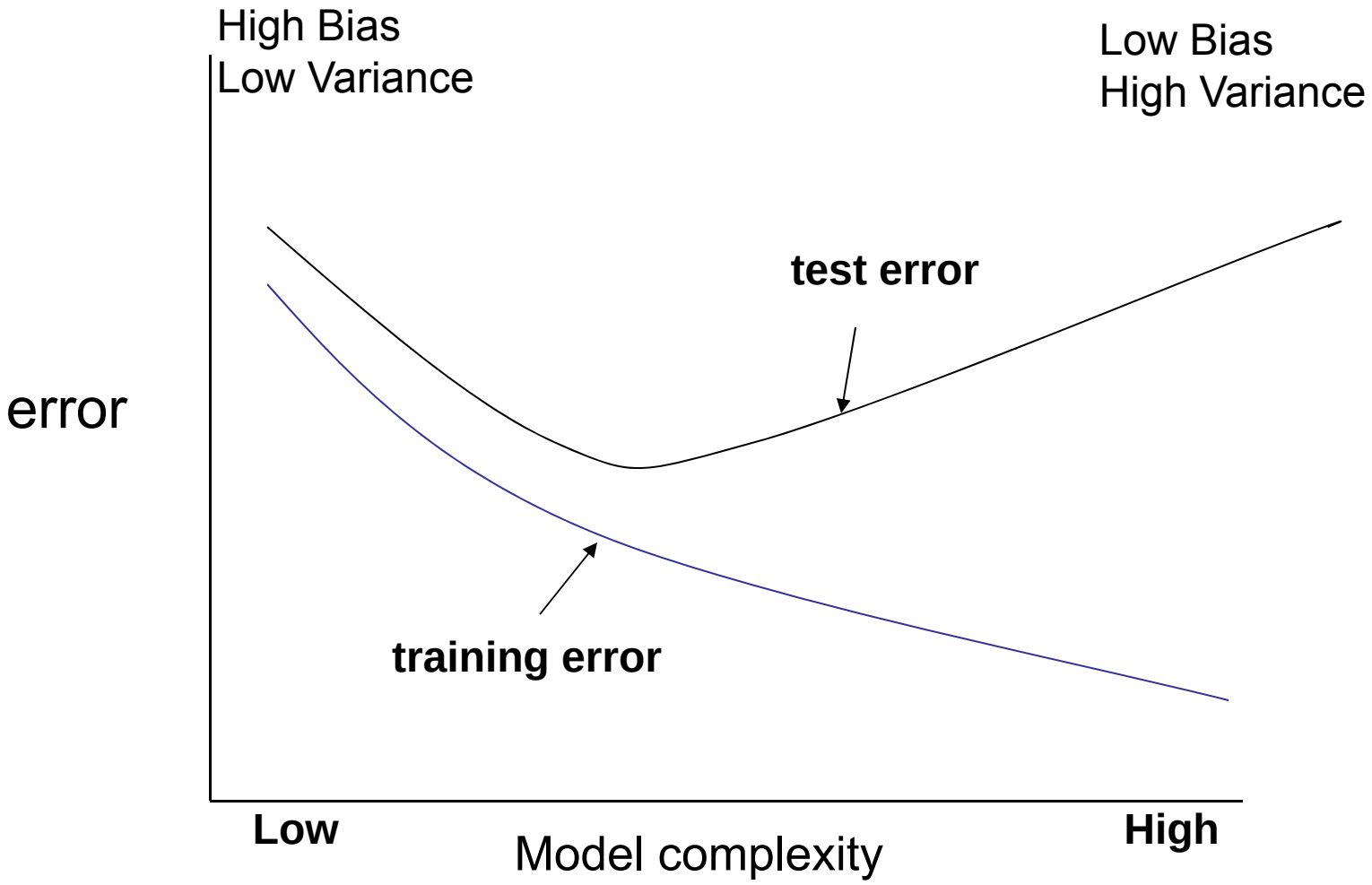
Bayes rules

- Recall posterior $p(k|x) = \frac{\pi_k p_k(x)}{\sum \pi_l p_l(x)}$
- *Bayes risk for action* $l = \sum L(k,l)p(k|l)$
- For 0-1 loss: *Bayes rule*: $C_B(x) = argmax_k p(k|x) =$ class with maximum posterior probability.
- More generally, for $L(k,l) = L_k I(k \neq l)$ then $C_B(x) = argmax_k L_k p(k|x)$

Evaluating classifiers

- *Error* defined conditionally on the training set $\mathcal{L}$ as well as unconditionally over all training sets.
- For particular training set $\mathcal{L}$ *training error* $= e_{train} = (1/n) \sum L(Y_i, C(X_i))$.
- Unconditionally, *test (or generalization) error*= $error_{test} = E(L(Y, C(X)) = \int L(y, C(x)) dF(y, x)$, last intergal taken over joint distribution of Y and X.
- Typically training error less than test error.
- In some situations a *validation* sample is available in addition to training and testing samples.

Bias-variance tradeoff



Nearest-neighbors discriminant rule

- The training set has samples with known classes
- Define a distance measure
  - Euclidean, 1-correlation, Mahalanobis…
- For each sample in a test set, find k closest neighbors
- Predict the class by majority vote
- How to choose k: usually by cross-validation

Example: Iris

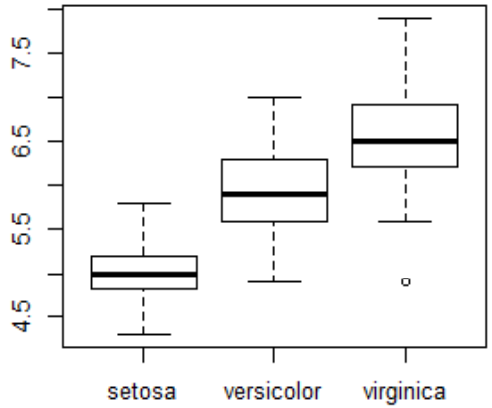
R package FNN **(Fast Nearest Neighbor Search Algorithms and Applications)**

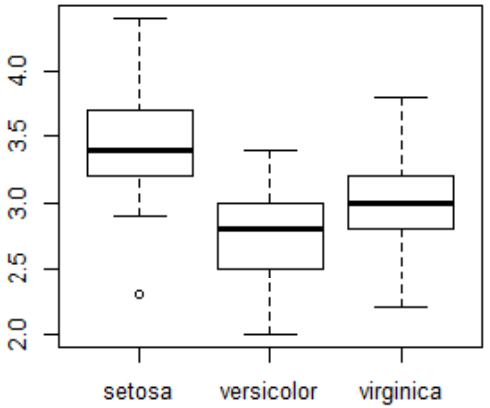
R package “class”

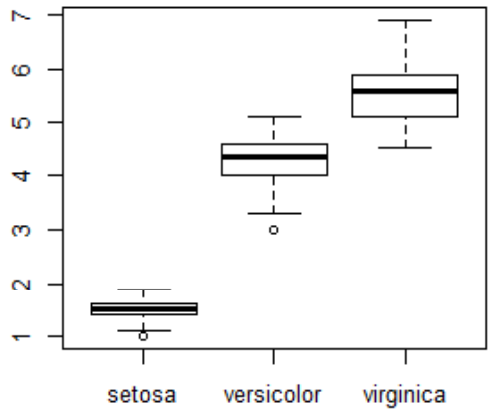
```
#########################################
library("class")
data(iris)
iris[1:5,]
#  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#1          5.1         3.5          1.4         0.2  setosa
#2          4.9         3.0          1.4         0.2  setosa
#3          4.7         3.2          1.3         0.2  setosa
#4          4.6         3.1          1.5         0.2  setosa
#5          5.0         3.6          1.4         0.2  setosa
```

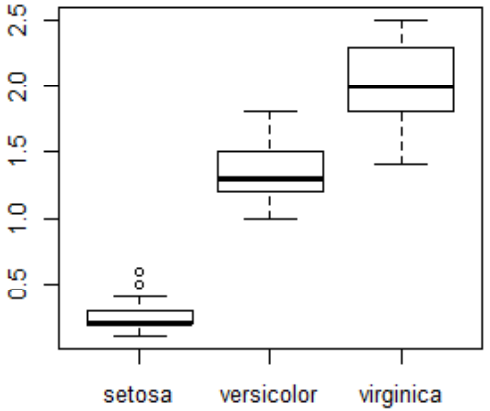
Explore data

```
par(mfrow=c(2,2))
for(i in 1:4)
boxplot(iris[,i]~iris[,5])
```



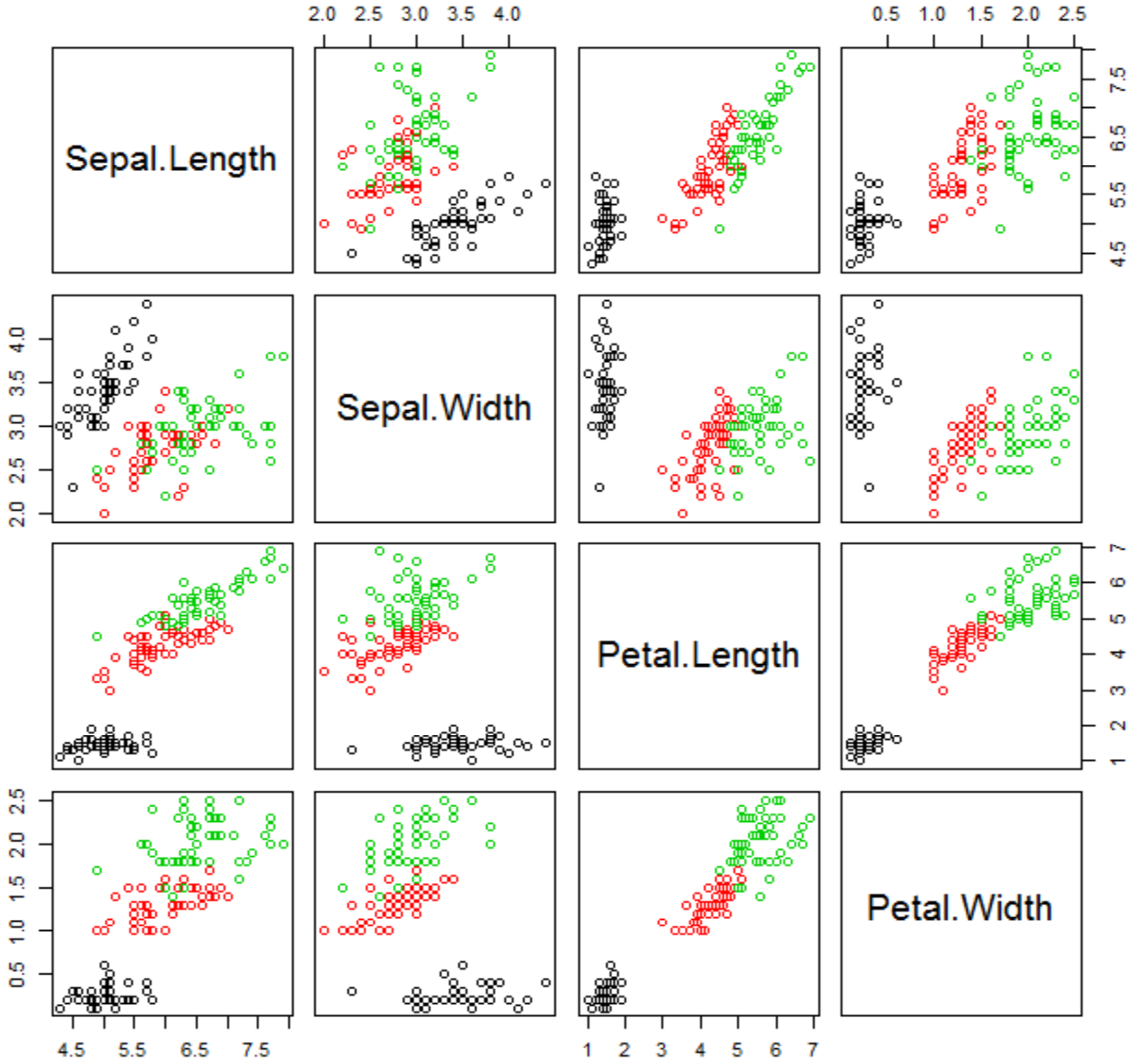






Explore data

```
pairs(iris[,1:4],
col=iris[,5])
```



```
set.seed(2014)
train.id=sample(1:nrow(iris),100)

knn1=knn(iris[train.id,1:4],iris[-
train.id,1:4],cl=iris[train.id,5],k=1)
table(knn1==iris[-train.id,5])
#
#FALSE  TRUE
#    3    47
> knn1=knn(iris[train.id,1:4],iris[-
train.id,1:4],cl=iris[train.id,5],k=5)
> table(knn1==iris[-train.id,5])

FALSE  TRUE
    4    46

> knn1=knn(iris[train.id,1:4],iris[-
train.id,1:4],cl=iris[train.id,5],k=3)
> table(knn1==iris[-train.id,5])

FALSE  TRUE
    3    47
```

```
### lets repeat  it
error1=sapply(1:100,function(i){
 train.id=sample(1:nrow(iris),100)
 knn1=knn(iris[train.id,1:4],iris[-
train.id,1:4],cl=iris[train.id,5],k=1)
 sum(knn1!=iris[-train.id,5])})
error3=sapply(1:100,function(i){
  train.id=sample(1:nrow(iris),100)
  knn1=knn(iris[train.id,1:4],iris[-
train.id,1:4],cl=iris[train.id,5],k=3)
  sum(knn1!=iris[-train.id,5])})
error5=sapply(1:100,function(i){
  train.id=sample(1:nrow(iris),100)
  knn1=knn(iris[train.id,1:4],iris[-
train.id,1:4],cl=iris[train.id,5],k=5)
  sum(knn1!=iris[-train.id,5])})
error7=sapply(1:100,function(i){
  train.id=sample(1:nrow(iris),100)
  knn1=knn(iris[train.id,1:4],iris[-
train.id,1:4],cl=iris[train.id,5],k=7)
  sum(knn1!=iris[-train.id,5])})
```

```
> mean(error1)
[1] 2.23
> mean(error3)
[1] 2.12
> mean(error5)
[1] 1.72
 > mean(error7)
[1] 1.74
```

So it seems k=5 works best.
What is my final classifier when you have a new data point?

Linear Discriminant Analysis

- We start with the simplest toy example:
  - Suppose we only have one feature (gene), and we want to classify the samples into one of two classes.
  - Suppose $f_1(x) \sim N(\mu_1, \sigma^2)$ $f_2(x) \sim N(\mu_2, \sigma^2)$
- Recall the Bayes rule

$$\text{Posterior } p(k|x) = \frac{\pi_k p_k(x)}{\sum \pi_l p_l(x)}$$

We classify an object to class k with the highest posterior probability

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}$$

- To find the k that maximizes

$$\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)$$

- That is, the k that maximizes

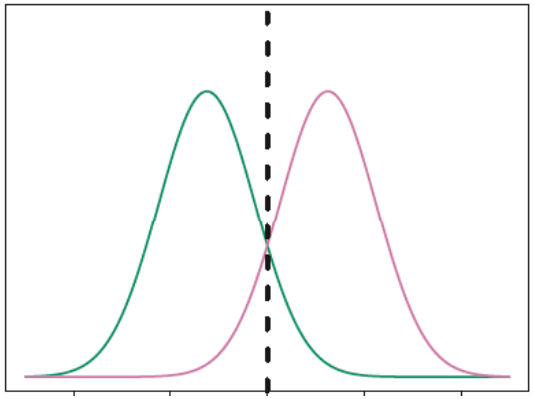
$$x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

- If we have a priori $\pi_1 = \pi_2 = 0.5$

  Then we classify to class 1 when $2x\,(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$

  Suppose we refer to the class with smaller mean "class 1", this means we classify to class 1 when

  x < $\frac{\mu_1 + \mu_2}{2}$



Extending from the simplest toy example

- More than two classes: assigns to the class with the highest discriminant function

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

- Allowing class specific variance
- Allowing multi-dimension for *x*

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

- In practice we often have to estimate the center and covariance. If the prior probability is 50% each, We classify to class 1 if

$$(\bar{X}_1 - \bar{X}_2)' S_{pooled}^{-1} X \geq (1/2)(\bar{X}_1 - \bar{X}_2)' S_{pooled}^{-1} (\bar{X}_1 + \bar{X}_2)$$

2-dimensional toy example

- $\sum = I$
  - Recall with 1 gene (feature) we had class (if $\mu_1 < \mu_2$)

$$x \leq (\mu_1 + \mu_2)/2$$

  - Now in 2-dimensions the parallel is

$$(\mu_1 - \mu_2)'\Sigma^{-1}x \geq \frac{1}{2}(\mu_1 - \mu_2)'\Sigma^{-1}(\mu_1 + \mu_2)$$

$$\sum_g (\mu_{1g} - \mu_{2g})x_g \geq \frac{1}{2}\left(\sum_g \mu_{1g}^2 - \sum_g \mu_{1g}^2\right)$$

  - Simply put: we compute a weighted sum of each genes' score, and we weigh the gene more if the two classes are more separated on this gene(feature)

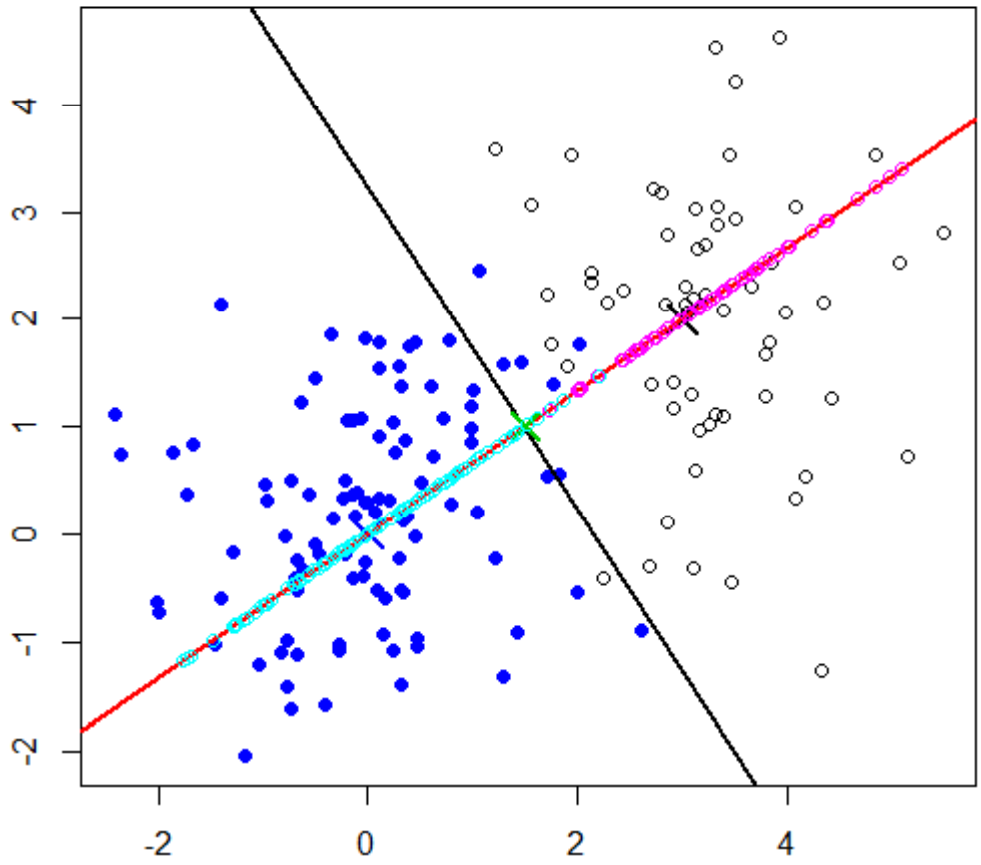
- We call it a linear discriminant because it is based on a linear transformation of ***x*** (a linear combination of the features)

$$(\bar{X}_1 - \bar{X}_2)' S_{pooled}^{-1} X \geq (1/2)(\bar{X}_1 - \bar{X}_2)' S_{pooled}^{-1} (\bar{X}_1 + \bar{X}_2)$$

  - For linear combination $a'x$, $E(a'X) = a'\mu_i$ and $Var(a'X) = a'\Sigma a.$
  - $\frac{SS_{between}}{SS_{within}} = \frac{(a'\mu_1 - a'\mu_2)^2}{a'\Sigma a}$

When $a = \Sigma^{-1}(\mu_1 - \mu_2)$ the ratio of sum of squares is maximized.

The linear combination can be seen as projecting points in a multidimensional space onto a line.


4
3
2
1
0
-1
-2
-2
0
2
4

2-dimensional toy example

- ∑ :diagonal $\begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$
  - The linear combination is

$$(\mu_1 - \mu_2)'\Sigma^{-1}x = \sum_{g=1}^{2} \frac{(\mu_{1g} - \mu_{2g})}{\sigma_g} \frac{x_g}{\sigma_g}$$

  - First we standardize each feature(gene) to have the same variance; then we proceed as before.

2-dimensional in general

- $\sum = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$

$$(\mu_1 - \mu_2)' \Sigma^{-1} x$$

weight for each feature adjusted by the correlation of the two features.

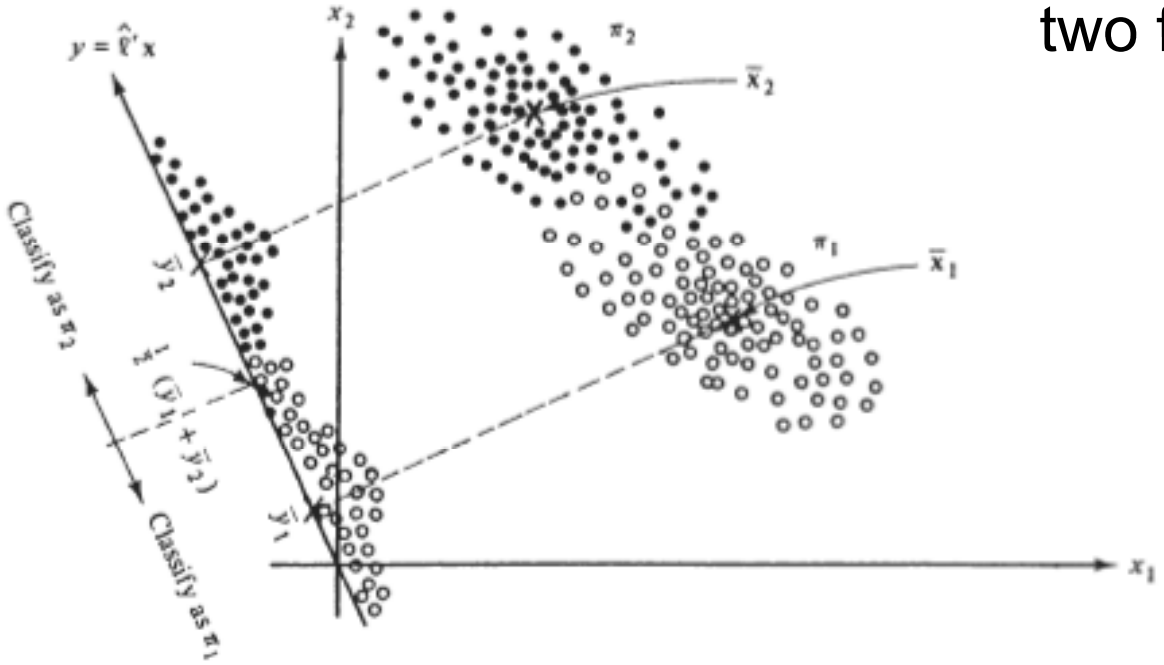


Figure 11.1 A pictorial representation of Fisher's procedure for two populations with $p = 2$.

- In general we have *n* features, we estimate the class centers and pooled variance covariance matrix of the features

$$S_{pooled}=[(N_1-1)S_1+(N_2-1)S_2]/(N_1+N_2-2)$$

- Note that even when we allow each feature to have unequal variance, we still assume that the covariance matrix is the same across classes

Discriminant rule: Assign *x* to Class 1 if

$$(\bar{X}_1 - \bar{X}_2)' S^{-1}_{pooled} X \geq (1/2)(\bar{X}_1 - \bar{X}_2)' S^{-1}_{pooled} (\bar{X}_1 + \bar{X}_2)$$

otherwise to class 2.

With microarray data, S is often singular, and generalized inverse of S, denoted by S^- is often used

Fisher's linear discriminant analysis

- More general *c*>2
  maximize between/within sum of squares

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

$$\mu_c = \frac{1}{N_c} \sum_{i \in c} x_i$$

$$\bar{x} = = \frac{1}{N} \sum_i x_i = \frac{1}{N} \sum_c N_c \mu_c$$

$$S_B = \sum_c N_c (\boldsymbol{\mu}_c - \bar{\mathbf{x}})(\boldsymbol{\mu}_c - \bar{\mathbf{x}})^T$$

$$S_W = \sum_c \sum_{i \in c} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^T$$

FYI

- The problem is equivalent to

$$\begin{aligned} \min_{\mathbf{w}} \quad & -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^T S_W \mathbf{w} = 1 \end{aligned}$$

corresponding to the lagrangian,

$$\mathcal{L}_P = -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} + \frac{1}{2}\lambda(\mathbf{w}^T S_W \mathbf{w} - 1)$$

Solution: find eigen values for

$$S_B^{\frac{1}{2}} S_W^{-1} S_B^{\frac{1}{2}}$$

Use the largest eigne vector v to form $S_B^{-\frac{1}{2}}\mathbf{v}$

FYI

Maximum likelihood discriminant rule

- ML discriminant rule
  - Pr(**x**|y=k)

    arg $\max_k$ pr(**x**|y=k)

  Recall Bayes rule: $p(k|x) = \frac{\pi_k p_k(x)}{\sum \pi_l p_l(x)}$
- Sample ML discriminant rule

$$\mathbf{x}|y=k \sim N(\mu_k, \Sigma_k)$$

$$\arg\min_k \{(\mathbf{x}-\mu_k)\Sigma_k^{-1}(\mathbf{x}-\mu_k)' + \log|\Sigma_k|\}$$

- Bayes rule

$$argmin_k[(x-\mu_k)\Sigma_k^{-1}(x-\mu_k)' + log|\Sigma_k| - 2log\pi_k]$$

Maximum likelihood discriminant rule special cases

- Linear Discriminant analysis

$$\Sigma_k = \Sigma$$

$$\arg\min_k (\mathbf{x} - \mu_k)\Sigma^{-1}(\mathbf{x} - \mu_k)'$$

- Diagonal quadratic discriminant analysis (DQDA): class densities have diagonal covariance matrices

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

$$\arg\min_k \sum_{j=1}^{p} \{(x_j - \mu_{kj})^2/\sigma_{kj}^2 + \log\sigma_{kj}^2\}$$

- Diagonal linear discriminant analysis (DLDA):

$$\Delta = \text{diag}(\sigma_1^2, \ldots, \sigma_p^2)$$

$$\arg\min_k \sum_{j=1}^{p} (x_j - \mu_{kj})^2/\sigma_j^2$$

Weighted gene voting scheme

- Variant of sample ML with same diagonal covariance
  - For two-class case, classify a sample with gene expression profile x=(x_1,x_2,…,x_p), "vote" from each gene *j* is weighted distance $\frac{(x_j - \bar{x}_{2j})^2}{\hat{\sigma}_j^2}$

Classify to class 1 if

$$\sum_{j=1}^{p} \frac{(x_j - \bar{x}_{2j})^2}{\hat{\sigma}_j^2} \geq \sum_{j=1}^{p} \frac{(x_j - \bar{x}_{1j})^2}{\hat{\sigma}_j^2} \quad \text{i.e.,} \quad \sum_{j=1}^{p} \frac{(\bar{x}_{1j} - \bar{x}_{2j})}{\hat{\sigma}_j^2} \left(x_j - \frac{(\bar{x}_{1j} + \bar{x}_{2j})}{2} \right) \geq 0$$

In Golub et al (1999),

$(\bar{x}_{1j} - \bar{x}_{2j})/(\hat{\sigma}_{1j} + \hat{\sigma}_{2j})$ is used instead of $\frac{(\bar{x}_{1j} - \bar{x}_{2j})}{\hat{\sigma}_j^2}$

Logistic discriminant function

- Binary classification. Logistic regression model $logitp(Y = 1|X = x) = \alpha + \beta' x$.
- Predicted probability: $\hat{p}(1|x) = \frac{e^{\hat{\alpha}+\hat{\beta}'x}}{1+e^{\hat{\alpha}+\hat{\beta}'x}}$ Classification rule based on predicted probability.
- Extension to multiple categories and beyond linearity: $logp(k|x) - logp(1|x) = g_k(x, \theta)$, with $g_1(x, \theta) = 0$.
- $\hat{p}(k|x) = \frac{e^{g_k(x,\hat{\theta})}}{\sum e^{g_k(x,\hat{\theta})}}$
- Bayes rule $C_B(x) = argmax_k \hat{p}(k|x)$

Nearest centroid discriminant rule

- Variant of Bayes rule

$$argmin_k[(x-\mu_k)\Sigma_k^{-1}(x-\mu_k)' + log|\Sigma_k| - 2log\pi_k]$$

Ignoring covariance terms and assume same variance matrix for all k,

$$r(x) = argmin_k \sum_{j=1}^{p} \frac{(x_j - \bar{x}_{kj})^2}{s_{kj}^2} - log(\hat{\pi}_k)$$

If prior class probabilities are equal to 1/k, the rule assigns x to the class with the closest mean (centroid)

Q: filter genes or not? How to filter genes?

nearest shrunken centroid method

Prediction Analysis for Microarrays (PAM)

- Centroid distance classification
- Regularize by shrinking the centroids

  gene i (1~G), sample j (1~n, in K classes):

$$d_{ik} = \frac{\bar{x}_{ik} - \bar{x}_i}{m_k \cdot (s_i + s_0)}$$

S_i is pooled within-class standard deviation $s_i^2 = \frac{1}{n-K} \sum_k \sum_{j \in C_k} (x_{ij} - \bar{x}_{ik})^2$

$m_k = \sqrt{1/n_k + 1/n}$ *notice that* $m_k \cdot s_i$ *is the standard error of* $\bar{x}_{ik} - \bar{x}_i$

$$d_j = (\bar{x}_j - x) / [m_j (s + s_0)]$$

Centroid: $$\bar{x}_{ik} = \bar{x}_i + m_k(s_i + s_0)d_{ik}$$

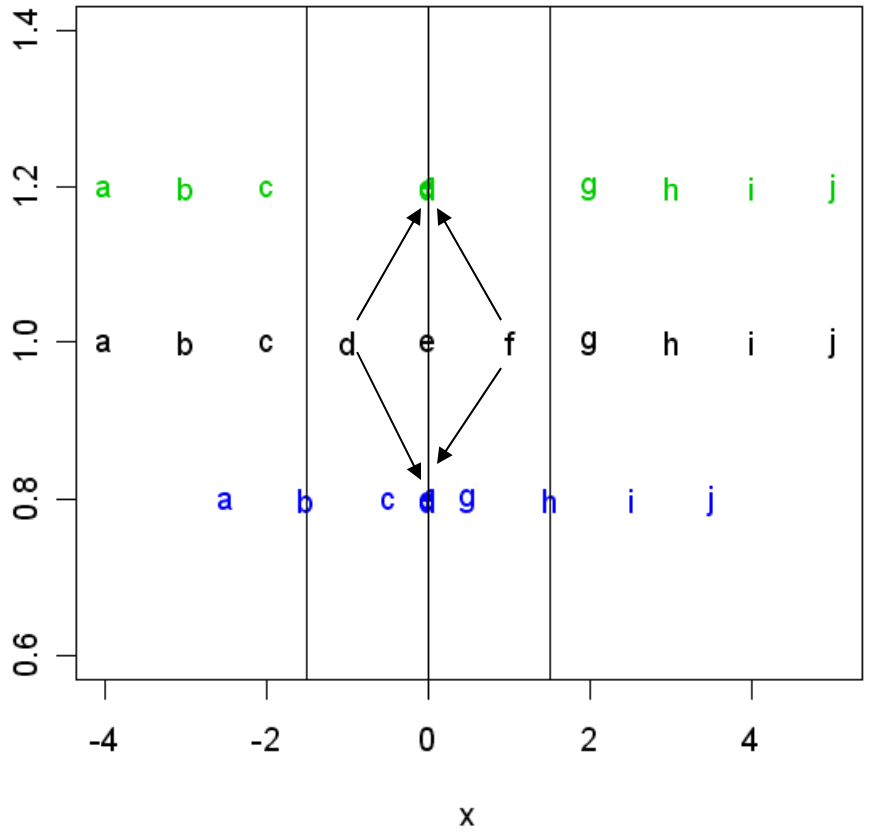
- From overall center, each gene in each class centroid deviates from it
- Some genes are not associated with the classes
- Let's keep gene *i* if its statistic *d* is large enough (larger than Δ)

$$d'_{ik} = \text{sign}(d_{ik})(|d_{ik}| - \Delta)_+$$

i.e., d'=d- Δ if d> Δ; d'=d+ Δ if d<- Δ ; and 0 otherwise

Soft thresholding

Soft thresholding/hard thresholding



- Both shrink the values within threshold to 0
- Direct thresholding leaves other values intact
- Soft thresholding shrinks everything

Centroid: $$\bar{x}_{ik} = \bar{x}_i + m_k(s_i + s_0)d_{ik}$$

- From overall center, each gene in each class centroid deviates from it
- Some genes are not associated with the classes
- Let's keep gene *i* if its statistic *d* is large enough (larger than Δ)

$$d'_{ik} = \text{sign}(d_{ik})(|d_{ik}| - \Delta)_+$$

i.e., d'=d- Δ if d> Δ; d'=d+ Δ if d<- Δ ; and 0 otherwise

Shrunken Centroid: $$\bar{x}'_{ik} = \bar{x}_i + m_k(s_i + s_0)d'_{ik}$$

Shrunken to the global mean if difference is not significant

Lastly: How to choose Δ

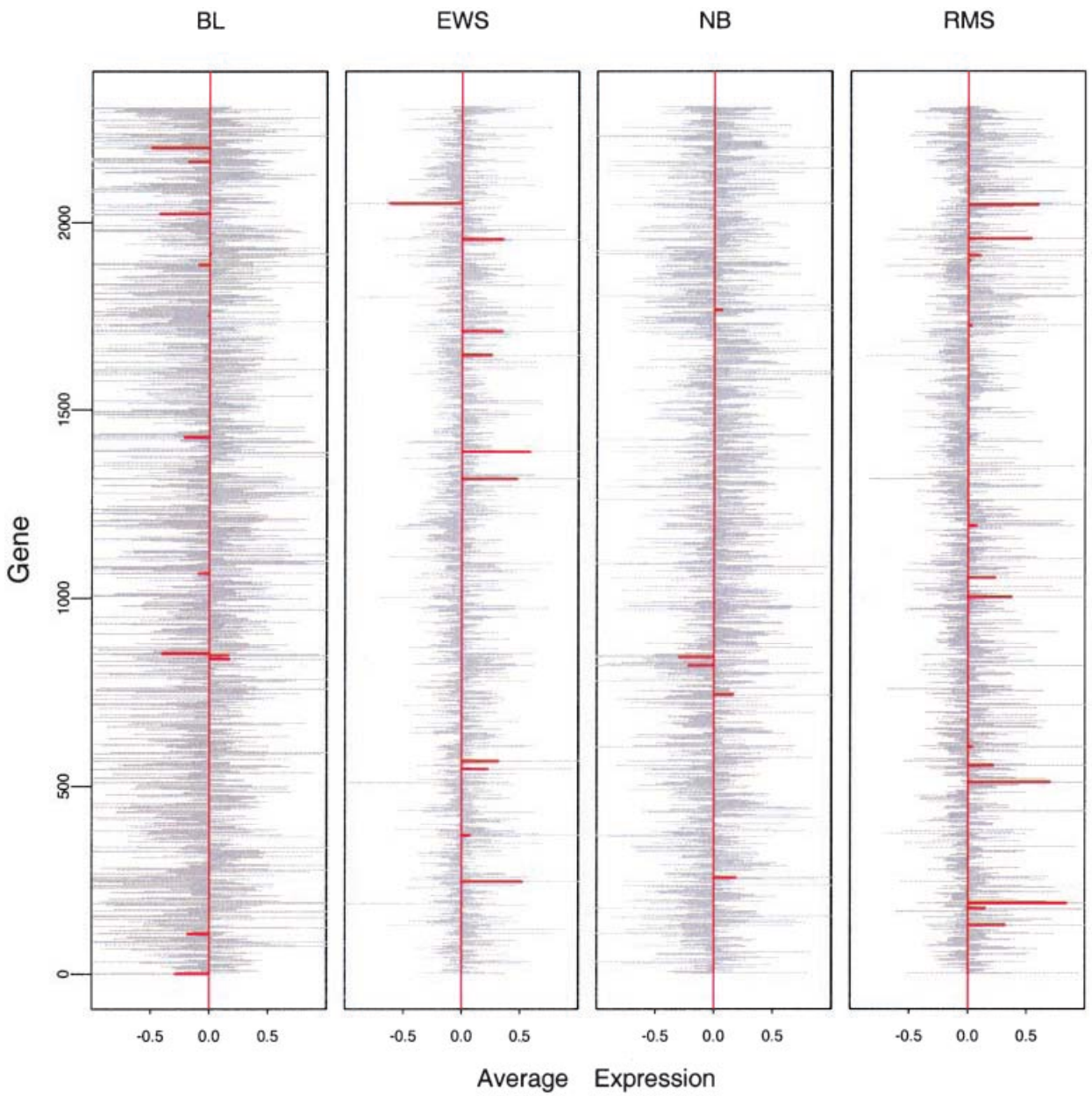


Fig. 1. Centroids (grey) and shrunken centroids (red) for the SRBCT dataset. The overall centroid has been subtracted from the centroid from each class. The horizontal units are log ratios of expression. From left to right, the numbers of training samples for each class are 8, 23, 12, and 20. The order of the genes is arbitrary.

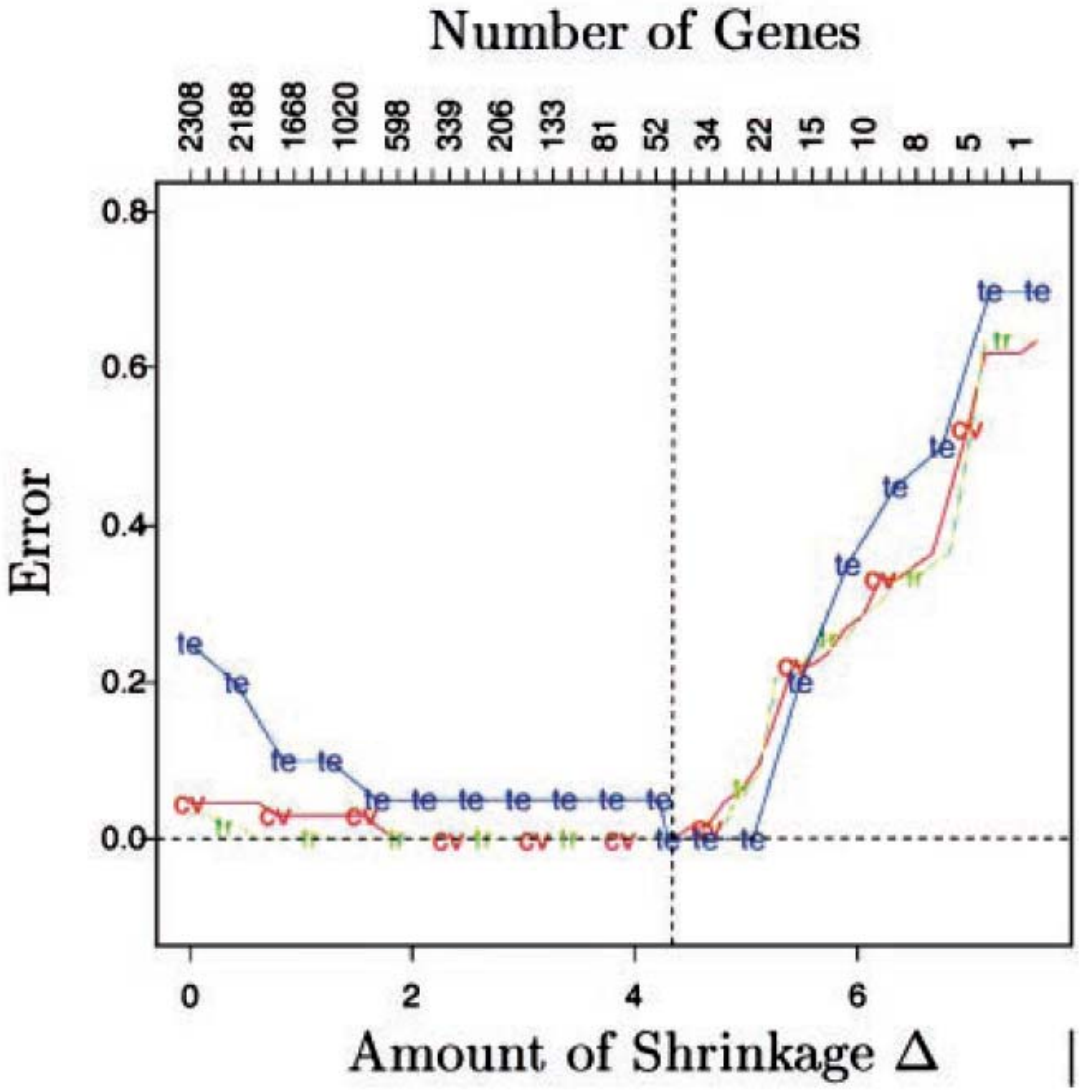


Fig. 2. SBRCT classification: training (tr, green), cross-validation (cv, red), and test (te, blue) errors are shown as a function of the threshold parameter Δ. The value $\Delta = 4.34$ is chosen and yields a subset of 43 selected genes.

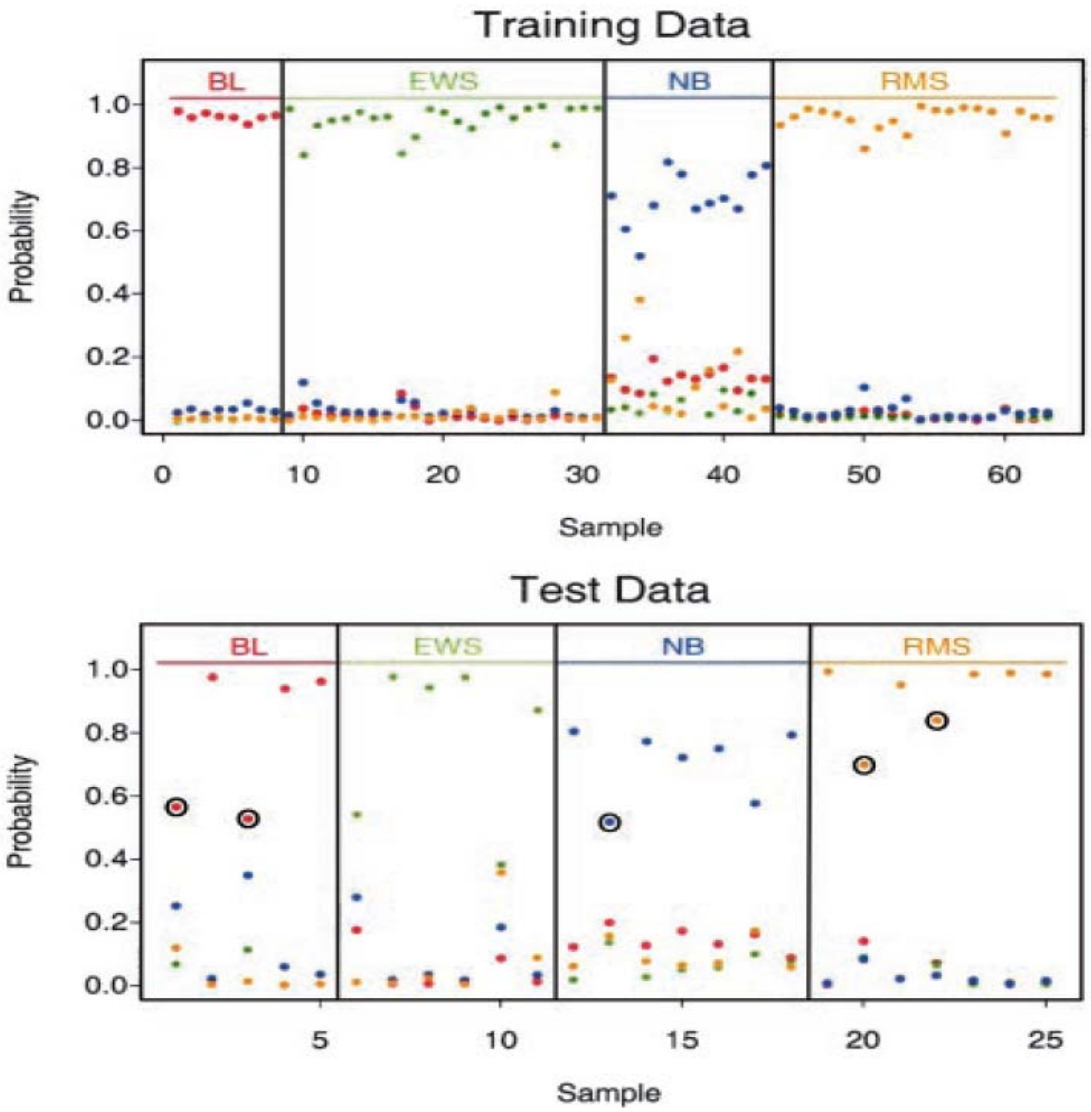
Discriminant rule and probability

For one test sample $x^* = (x_1^*, x_2^*, \ldots x_p^*)$

$$\delta_k(x^*) = \sum_{i=1}^{p} \frac{(x_i^* - \bar{x}'_{ik})^2}{(s_i + s_0)^2} - 2 \log \pi_k$$

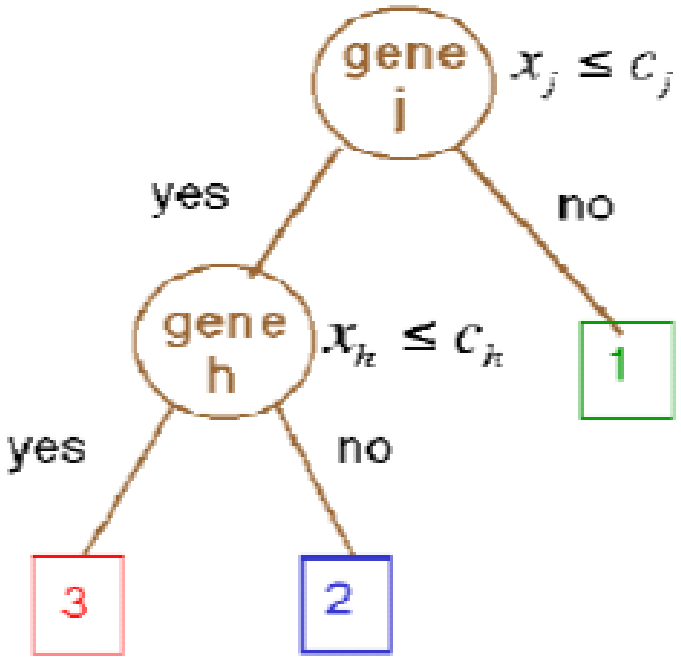
- Recall posterior $p(k|x) = \frac{\pi_k p_k(x)}{\sum \pi_l p_l(x)}$

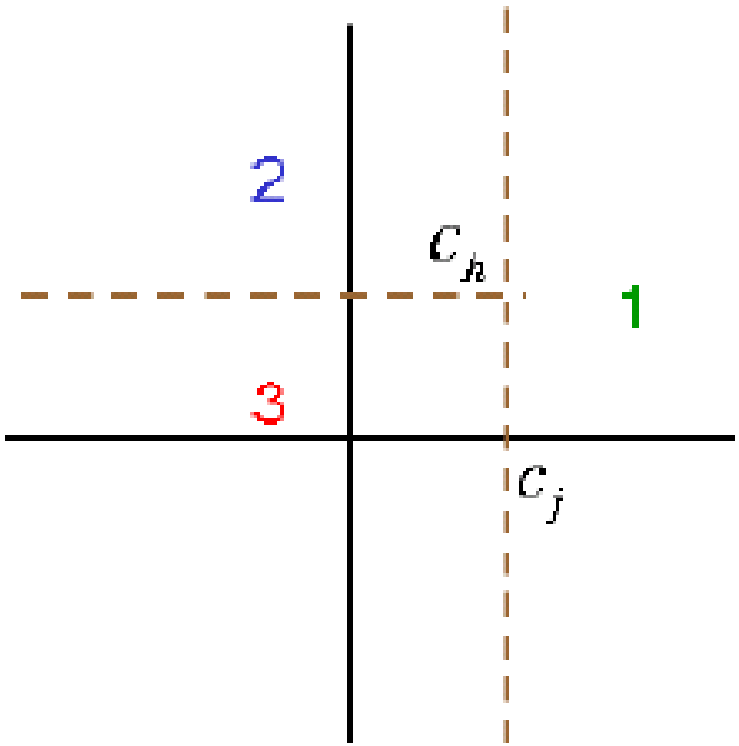
$$\hat{p}_k(x^*) = \frac{e^{-\frac{1}{2}\delta_k(x^*)}}{\sum_{\ell=1}^{K} e^{-\frac{1}{2}\delta_\ell(x^*)}}$$


Training Data
BL
EWS
NB
RMS
Probability
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
Sample
Test Data
BL
EWS
NB
RMS
Probability
1.0
0.8
0.6
0.4
0.2
0.0
5
10
15
20
25
Sample

Classification trees

- *Binary tree structured* classifiers most common. (Breiman et al, Classification and Regression Trees, 1984)
- Each node (subset of feature space) is split into two subsets according to *splitting rule*.
- At each step, decision is made to consider node as "terminal".
- Terminal nodes are assigned to one of two categories.


gene i
$x_j \le c_j$
yes
no
gene h
$x_h \le c_h$
1
yes
no
3
2


2
c_h
1
3
c_j

- Split data using set of binary decisions
- Root node (with all data points) has certain impurity, splitting reduces impurity
  - Highest on root, lowest (0) at leaf node
- Measure of impurities
  - Entropy $-\sum_{class} P(class)\log_2(P(class))$
  - Gini index impurity $1-\sum_{i}(P(class))^2$
- Prune the tree to prevent over fit