C03

分類法則
Classification

吳漢銘

國立政治大學 統計學系

http://www.hmwu.idv.tw

本章大綱&學習目標

- Introduction
- Classification of Subjects or Samples (Supervised Learning)
- Performance Measures (評估指標)
- Methods
  - K-Nearest Neighbors (KNN) (*k*最近鄰居法)
  - Linear/Quadratic Discriminant Analysis (LDA/QDA) (區別分析)
  - Classification Tree (Decision Tree) (分類樹、決策樹)
  - Random Forest (隨機森林)
  - Support Vector Machine (SVM) (支持向量機)
  - Artificial Neural Network (ANN) (人工神經網路)
- Ensemble Learning (整合學習)

  http://www.hmwu.idv.tw/web/R/C05-hmwu_R-EnsembleLearning.pdf
  - XGBoost: eXtreme Gradient Boosting (極限梯度提升)

R Packages

CRAN Task View: Machine Learning & Statistical Learning

http://cran.r-project.org/web/views/MachineLearning.html

- Topics: Neural Networks, Recursive Partitioning, Random Forests, Regularized and Shrinkage Methods, Boosting, Support Vector Machines and Kernel Methods, Bayesian Methods, Optimization using Genetic Algorithms, Association Rules, Fuzzy Rule-based System, Model selection and validation, Meta packages, Elements of Statistical Learning.

http://cran.r-project.org/web/packages/package-name

- knn (最近k鄰居分類法)
  - `class`: Functions for Classification
- lda (線性區別分析)
  - `MASS`: Support Functions and Datasets for Venables and Ripley's MASS
- Decision Tree (決策樹)
  - `C50`: C5.0 Decision Trees and Rule-Based Models
  - `rpart`: Recursive Partitioning and Regression Trees
  - `party`: A Laboratory for Recursive Partytioning
  - caret: **Classification and Regression Training**
- SVM (支持向量機)
  - `e1071`: Misc Functions of the Department of Statistics (e1071), TU Wien

What is Classification?

- Classification
  - Clustering (unsupervised learning) (群集分析、非監督式學習)
  - Discriminant Analysis (supervised learning, classification) (區別分析、監督式學習、分類法則)
- Discriminant Analysis
  - It focuses on situations where the different groups (clusters) are known a priori.
  - Decision rules are provided in classifying a multivariate observation into one of the known groups.
  - Classification is the task of learning a target function $\boldsymbol{f}$ that maps each attribute set $\boldsymbol{x}$ to one of the predefined class labels $\boldsymbol{y}$.

Class Prediction Analysis

- **Class prediction analysis** is designed to predict the value, or “class”, of an individual parameter in an uncharacteristic sample or set of samples.

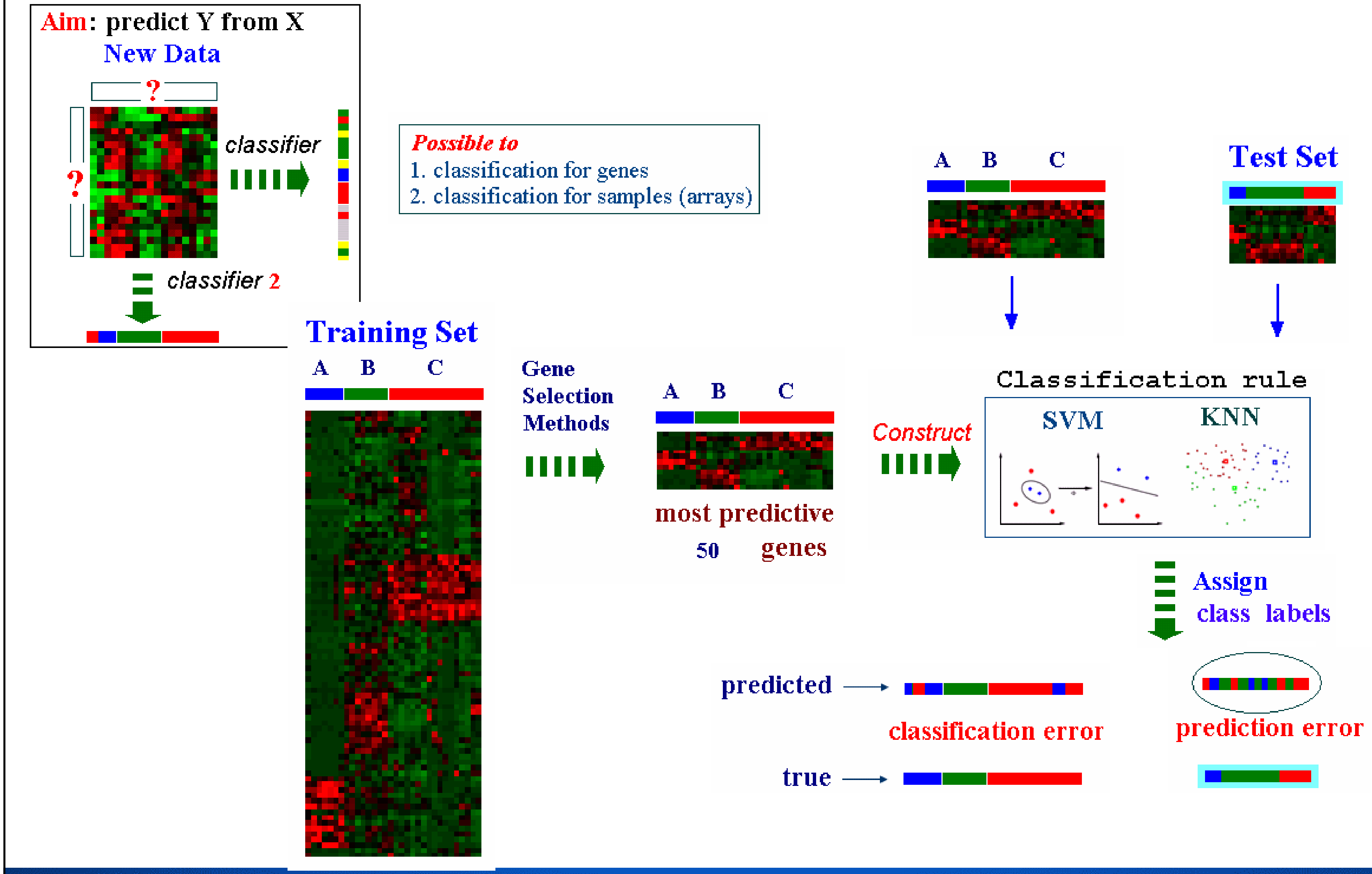
- **Apply classification to microarray data**
  - Predict cancer types using genomic expression profiling.
  - Predict the class/phenotype/parameter of a sample.
  - Identify genes that discriminate well among classes
  - Identify samples that could be potential outliers.

- **Examples of classification task**
  - Classifying credit card transactions as legitimate or fraudulent
  - Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil.
  - Categorizing news stories as finance, weather, entertainment, sports, etc.

- Fernandez-Delgado, M., Cernadas, E., Barro, S. & Amorim, D. **Do we need hundreds of classifiers to solve real world classification problems?** *Journal of Machine Learning Research* 15, 3133–3181 (2014). [被引用 1883 次] (179 classifiers on 121 data sets)

Classification of Genes, Tissues or Samples

Possible to
1. classification for genes
2. classification for samples (arrays)



Aim: predict Y from X

New Data

?

?

classifier

classifier 2

Training Set

A B C

Gene Selection Methods

A B C

most predictive 50 genes

Construct

Classification rule

SVM

KNN

A B C

Test Set

Assign class labels

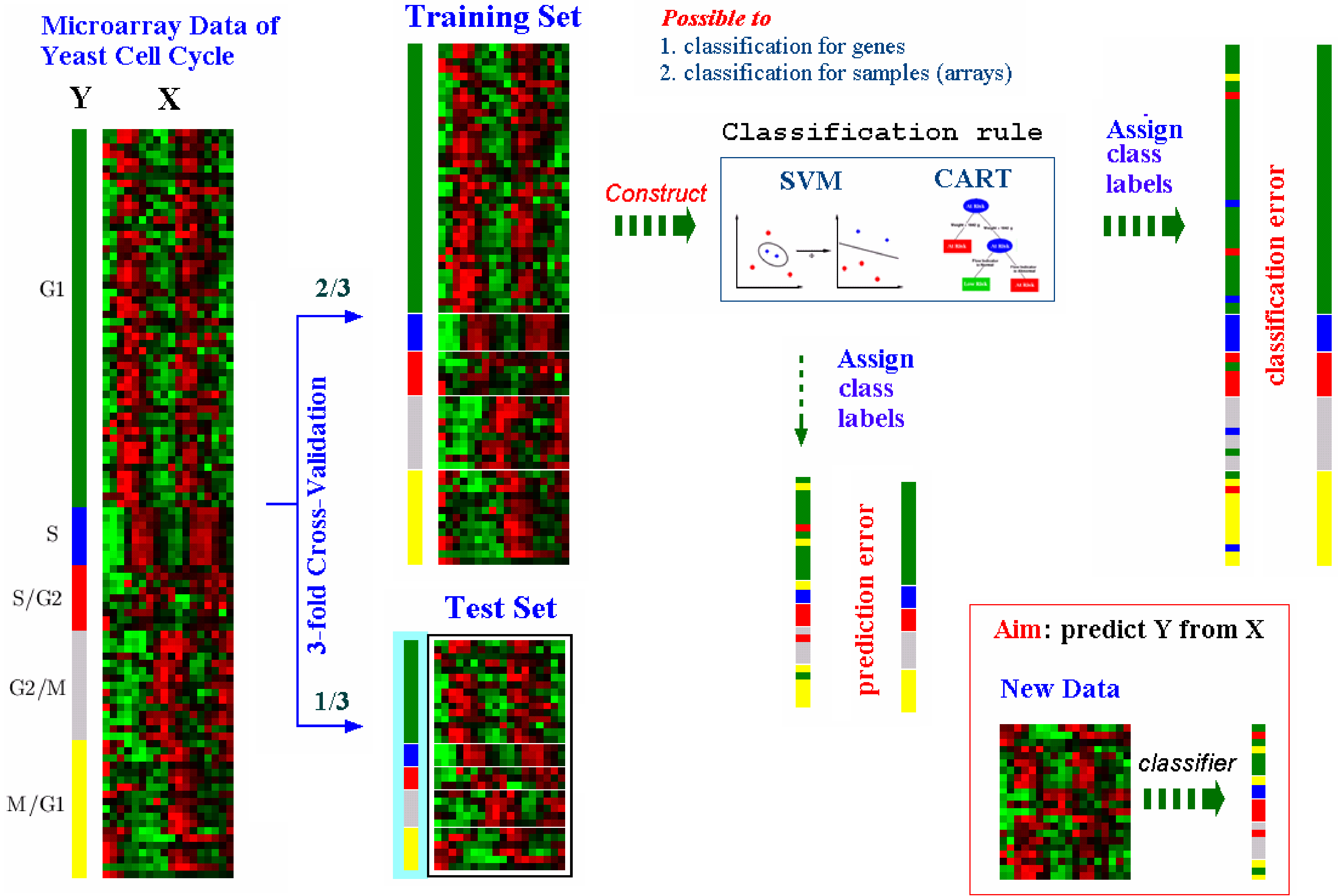
predicted →

classification error

true →

prediction error

n-fold Cross-Validation Error Rates



Microarray Data of Yeast Cell Cycle

3-fold Cross-Validation

Training Set

Test Set

Possible to

1. classification for genes
2. classification for samples (arrays)

Construct

Classification rule

SVM

CART

Assign class labels

classification error

Assign class labels

prediction error

Aim: predict Y from X

New Data

classifier

Split Data into Training Set and Test Set

```
> id <- sample(2, nrow(iris), replace = TRUE, prob = c(0.9, 0.1))
> id
  [1] 1 1 2 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
 [38] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
[112] 1 1 1 1 1 1 1 2 1 2 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 2 2 2 1 1
[149] 1 1
> train.data <- iris[id == 1, ]
> dim(train.data)
[1] 131   5
> test.data <- iris[id == 2, ]
> dim(test.data)
[1] 19  5
```

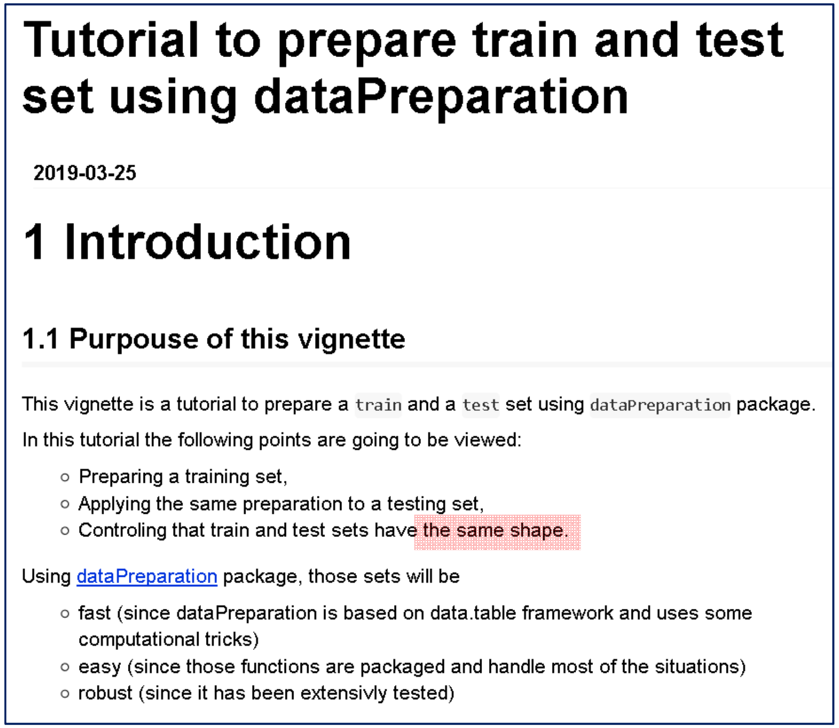
```
> id <- sample(nrow(iris), floor(nrow(iris) * 0.9))
> id
  [1]  39  27  96  33   4  98  12   3  32  48   2  22  18  24 126  93 140  85 110  60
 [21]  62  91 131  35 134 143  29 108 114  50  19  43  45  66  36  90 105  76 127  92
 [41]  68  57  65 147  69  41 130  82  31  20  51  17 149  61 107  70 139   5 115  72
 [61]  78 118 117  38  15  74 120 111 106  11 104  67  13  21 133  42  87 121 122  40
 [81]  84 135 123  77  83  97  52 116  55  88 142  16   7  49 125 112  34  10  56  26
[101]  99  63  37  46 144   9 141  59 138  80 101 132 129 113  73  30  44 136 119  79
[121]  95  64 109 148  28  14  86 150 137  81  94  75 128 102 124
> train.data <- iris[id, ]
> dim(train.data)
[1] 135   5
> test.data <- iris[-id, ]
> dim(test.data)
[1] 15  5
```

Split Data into Training Set and Test Set

```
splitdf <- function(df, train.ratio, seed=NULL) {
        if (!is.null(seed)) set.seed(seed)
        index <- 1:nrow(df)
        id <- sample(index, trunc(length(index)*train.ratio))
        train <- df[id, ]
        test <- df[-id, ]
        list(trainset=train,testset=test)
}
```

```
> splits <- splitdf(iris, 0.9, 12345)
> lapply(splits, dim)
$trainset
[1] 135   5

$testset
[1] 15  5

> iris.training <- splits$trainset
> iris.testing <- splits$testset
```

```
library(dplyr)
iris.train <- sample_frac(iris, 0.9)
id <- as.numeric(rownames(iris.train))
iris.test <- iris[-id, ]
```



Tutorial to prepare train and test set using dataPreparation

2019-03-25

1 Introduction

1.1 Purpouse of this vignette

This vignette is a tutorial to prepare a `train` and a `test` set using `dataPreparation` package.

In this tutorial the following points are going to be viewed:

- Preparing a training set,
- Applying the same preparation to a testing set,
- Controling that train and test sets have the same shape.

Using dataPreparation package, those sets will be

- fast (since dataPreparation is based on data.table framework and uses some computational tricks)
- easy (since those functions are packaged and handle most of the situations)
- robust (since it has been extensivly tested)

https://cran.r-project.org/web/packages/dataPreparation/vignettes/train_test_prep.html

Split Data into Test and Training Set According to Group Labels

```
> library(caTools)
> Y <- iris[,5] # extract labels from the data
> msk <- sample.split(Y, SplitRatio=4/5)
> msk
  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
...
[144]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
> table(Y, msk)
            msk
Y             FALSE TRUE
  setosa         10   40
  versicolor     10   40
  virginica      10   40
> iris.train <- iris[msk, ]
> iris.test <- iris[!msk, ]
> dim(iris.train)
[1] 120   5
> dim(iris.test)
[1] 30  5
```

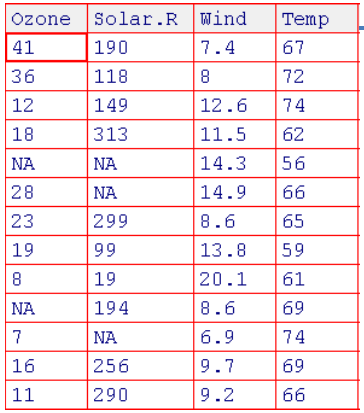
```
> library(caret)
> createFolds(iris$Species, k=3)
$Fold1
 [1]    2    8   15   22   25   27   30   ...

$Fold2
 [1]    5    6    9   10   11   12   17 ...

$Fold3
 [1]    1    3    4    7   13   14   16   20...
```

```
require(caTools)
set.seed(12345)
id <- sample.split(1:nrow(iris), SplitRatio = 0.90)
iris.train <- subset(iris, id == TRUE)
iris.test <- subset(iris, id == FALSE)
```

```
library(caret)
id <- createDataPartition(y=iris$Species, p=0.9, list=FALSE)
iris.train <- iris[id, ]
iris.test <- iris[-id, ]
```

Performance Measures

Binary Classifier



Ozone	Solar.R	Wind	Temp	True Y	Predict Y
41	190	7.4	67	0	0
36	118	8	72	0	0
12	149	12.6	74	0	1
18	313	11.5	62	1	1
NA	NA	14.3	56	1	0
28	NA	14.9	66	1	1
23	299	8.6	65		
19	99	13.8	59	.	.
8	19	20.1	61	.	.
NA	194	8.6	69	.	.
7	NA	6.9	74	.	.
16	256	9.7	69		
11	290	9.2	66		

Confusion Matrix

		Predicted Response	
		True (1)	False (0)
Actual Response	True (1)	Count_{11} (TP)	Count_{01} (FN)
	False (0)	Count_{10} (FP)	Count_{00} (TN)

- **Count:**$_{11}$ The number of observations that were true and predicted to be true (true positives).

- **Count:**$_{00}$ The number of observations that were false and predicted to be false (true negatives).

- **Count:**$_{10}$ The number of observations that were false yet predicted to be true (false positives).

- **Count:**$_{01}$ The number of observations that were true and predicted to be false (false negatives).

Confusion Matrix for Binary Response

Predicted Response

Actual Response	Total: N	True (1)	False (0)
	True (1)	True Positive (TP)	False Negative (FN) **Type II Error**
	False (0)	False Positive (FP) **Type I Error**	True Negative (TN)

$Pr(\hat{Y}=1|Y=1)$

TPR
Recall
Sensitivity

FNR

Prevalence $\frac{\text{TP}+\text{FN}}{\text{N}}$

$\frac{\text{TP}}{\text{TP}+\text{FN}}$	$\frac{\text{FN}}{\text{TP}+\text{FN}}$
$\frac{\text{FP}}{\text{FP}+\text{TN}}$	$\frac{\text{TN}}{\text{FP}+\text{TN}}$

FPR

TNR, Specificity

$Pr(\hat{Y}=0|Y=0)$

PPV, Precision

FOR

$\frac{\text{TP}}{\text{TP}+\text{FP}}$	$\frac{\text{FN}}{\text{FN}+\text{TN}}$
$\frac{\text{FP}}{\text{TP}+\text{FP}}$	$\frac{\text{TN}}{\text{FN}+\text{TN}}$

FDR

NPV

$Pr(Y=0|\hat{Y}=0)$

Accuracy $= \frac{\text{TP+TN}}{\text{N}}$

$Pr(\hat{Y}=Y)$

Misclassification rate $= \frac{\text{FP+FN}}{\text{N}}$

$Pr(\hat{Y}\neq Y)$

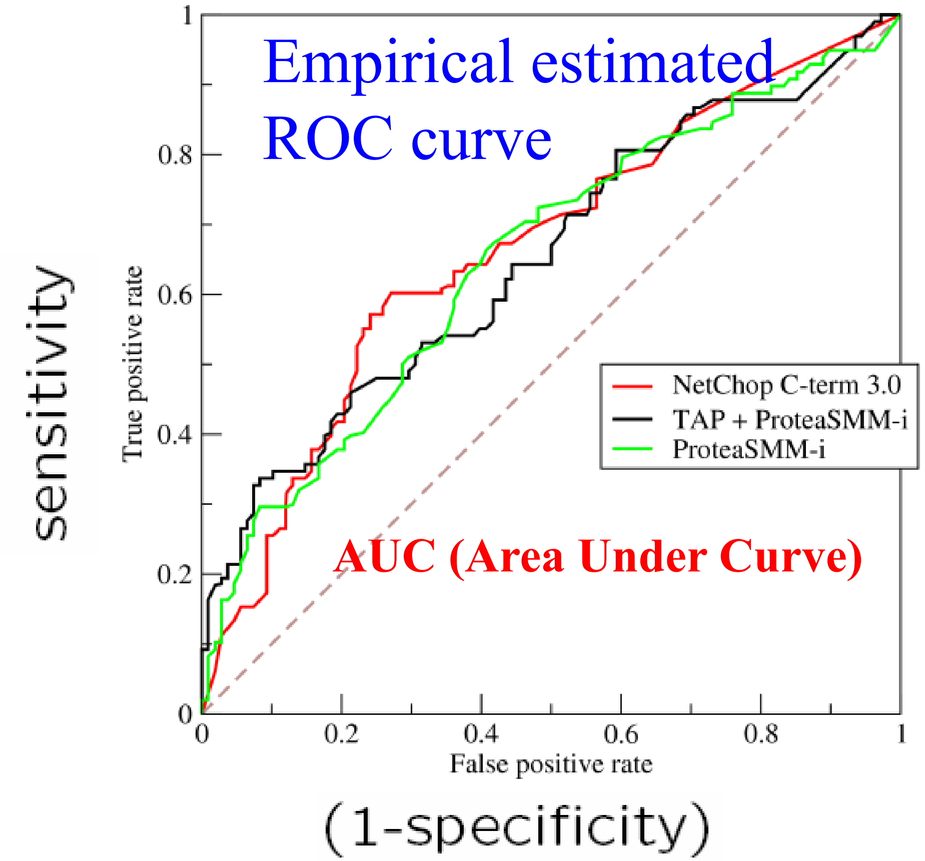
F1 Score $= \frac{2}{\frac{1}{\text{Precision}}+\frac{1}{\text{Recall}}}$

https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Receiver Operating Characteristic (ROC) Curve

A receiver operating characteristic (ROC) curve is a plot of sensitivity as a function of (1-specificity) for the possible cutoffs π_0.

$$\text{sensitivity} = P(\hat{y} = 1 | y = 1), \quad \text{specificity} = P(\hat{y} = 0 | y = 0)$$

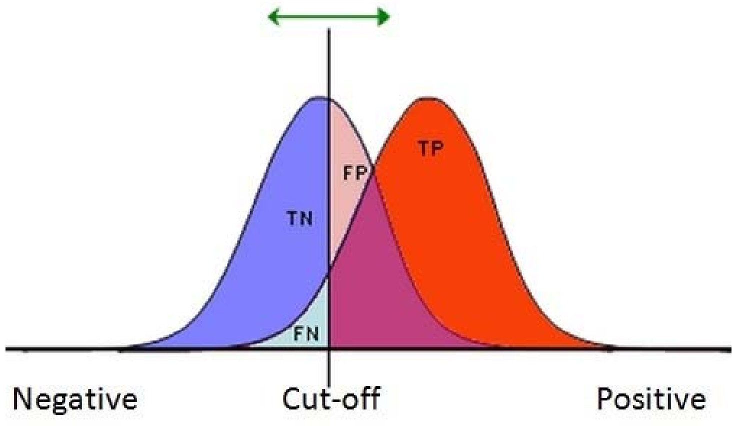


Empirical estimated ROC curve

AUC (Area Under Curve)

http://en.wikipedia.org/wiki/File:Roc.png

- A diagonal line is a baseline, where a random prediction would lie.
- The closer the point is to the upper top left point in the plot, the better the prediction.



$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Some R Packages for ROC Curves

- **ROCR** [2005]: **Visualizing the Performance of Scoring Classifiers**
  - Sing T, Sander O, Beerenwinkel N, Lengauer T. (2005) ROCR: visualizing classifier performance in R. Bioinformatics 21(20):3940-1.
  - https://ipa-tys.github.io/ROCR/
- **pROC** [2010]: Display and Analyze ROC Curves [Multi-class AUC]
  - Robin, X., Turck, N., Hainard, A. et al. pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics 12, 77 (2011).
- **PRROC** [2014]*: Precision-Recall and ROC Curves for Weighted and Unweighted Data
- **plotROC** [2014]*: Generate Useful ROC Curve Charts for Print and Interactive Use
  - Example: https://mlr.mlr-org.com/articles/tutorial/roc_analysis.html
- **precrec** [2015]*: Calculate Accurate Precision-Recall and ROC (Receiver Operator Characteristics) Curves [Multiple models and multiple test sets]
- **multiROC[2018]:** Calculating and Visualizing ROC and PR Curves Across Multi-Class Classifications
- **ROCit** [2019]*: Performance Assessment of Binary Classifier with Visualization

*Vignettes

NOTE: the ROC curves are typically used in binary classification but not for multiclass classification problems.

For **multi-class ROC/AUC:**

- Fieldsend, Jonathan & Everson, Richard. (2005). Formulation and comparison of multi-class ROC surfaces.
- Hand, D.J., Till, R.J. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine Learning* 45, 171–186 (2001). [被引用 1639 次]
- How to plot ROC curves in multiclass classification? https://stats.stackexchange.com/questions/2151/how-to-plot-roc-curves-in-multiclass-classification/2155#2155

ROCR: Visualizing the Performance of Scoring Classifiers

```
> library(ROCR) # ROCR supports only binary classification
> data(ROCR.simple)  # n = 200
> ROCR.simple
$predictions
  [1] 0.612547843 0.364270971 0.432136142 0.140291078 0.384895941 0.244415489 0.970641299
...
  [197] 0.858970526 0.383807972 0.606960209 0.138387070

$labels
  [1] 1 1 0 0 0 1 1 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0
 ...
  [173] 0 1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1 0

> pred <- prediction(ROCR.simple$predictions, ROCR.simple$labels)
> pred
An object of class "prediction"
Slot "predictions":
[[1]]
  [1] 0.612547843 0.364270971 0.432136142 0.140291078 0.384895941 0.244415489 0.970641299
...
[197] 0.858970526 0.383807972 0.606960209 0.138387070


Slot "labels":
[[1]]
  [1] 1 1 0 0 0 1 1 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0
 ...
  [173] 0 1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1 0
Levels: 0 < 1
```

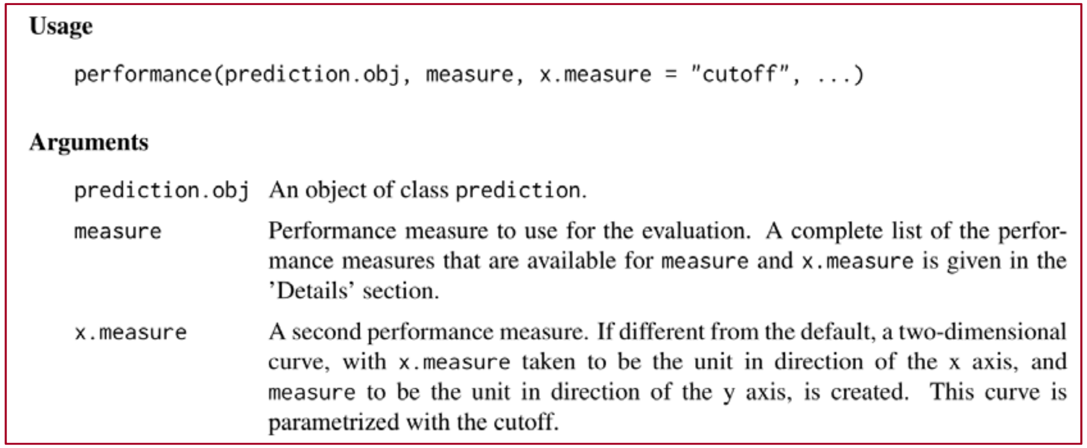
ROCR: Visualizing the Performance of Scoring Classifiers

```
Slot "cutoffs":
[[1]]
  [1]         Inf 0.991096434 0.984667270 0.984599159 0.983494405 0.970641299 0.959417835
  ...
Slot "fp": # a vector of the number of false positives induced by the cutoffs
[[1]]
  [1]   0   0   0   0   1   1   2   3   3   3   3   3   3   3   4   4   4   4   4   ...
Slot "tp":
[[1]]
  [1]  0  1  2  3  3  4  4  4  5  6  7  8  9 10 10 11 12 13 14 15 16 17 17 18 19    ...
Slot "tn":
[[1]]
  [1] 107 107 107 107 106 106 105 104 104 104 104 104 104 104 103 103 103 103 103   ...
Slot "fn":
[[1]]
  [1] 93 92 91 90 90 89 89 89 88 87 86 85 84 83 83 82 81 80 79 78 77 76 76 75 74   ...
Slot "n.pos": # contains the number of positive samples in the  given x-validation run
[[1]]
[1] 93
Slot "n.neg":
[[1]]
[1] 107
Slot "n.pos.pred":
[[1]]
  [1]   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  ...
Slot "n.neg.pred":
[[1]]
  [1] 200 199 198 197 196 195 194 193 192 191 190 189 188 187 186 185 184 183 182 18 ...
```

the 2x2 contingency table consisting of `tp`, `tn`, `fp`,and `fn`, along with the marginal sums `n.pos`, `n.neg`, `n.pos.pred`, `n.neg.pred`.

ROCR: Visualizing the Performance of Scoring Classifiers

```
> perf <- performance(pred, measure = "tpr", x.measure = "fpr") # predictor evaluation
> perf
An object of class "performance"
Slot "x.name":
[1] "False positive rate"

Slot "y.name":
[1] "True positive rate"

Slot "alpha.name":
[1] "Cutoff"

Slot "x.values":
[[1]]
  [1] 0.000000000 0.000000000 0.000000000 0.000000000 0.009345794 0.009345794 0.018691589
...

Slot "y.values":
[[1]]
  [1] 0.00000000 0.01075269 0.02150538 0.03225806 0.03225806 0.04301075 0.04301075
...

Slot "alpha.values":
[[1]]
  [1]         Inf 0.991096434 0.984667270 0.984599159 0.983494405 0.970641299 0.959417835
...
```



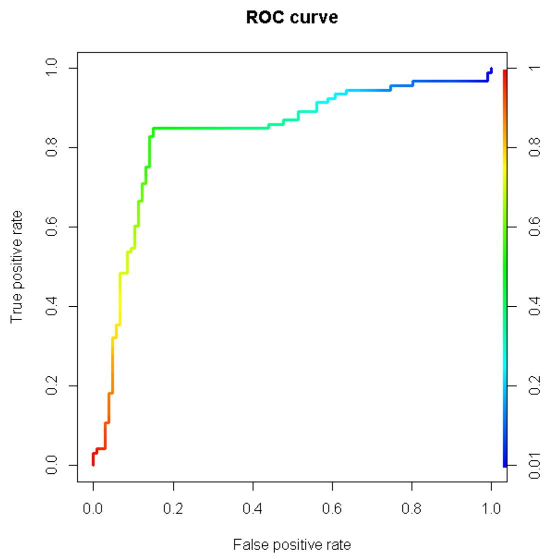
Usage

```
performance(prediction.obj, measure, x.measure = "cutoff", ...)
```

Arguments

`prediction.obj`	An object of class `prediction`.
`measure`	Performance measure to use for the evaluation. A complete list of the performance measures that are available for `measure` and `x.measure` is given in the 'Details' section.
`x.measure`	A second performance measure. If different from the default, a two-dimensional curve, with `x.measure` taken to be the unit in direction of the x axis, and `measure` to be the unit in direction of the y axis, is created. This curve is parametrized with the cutoff.

ROCR: Visualizing the Performance of Scoring Classifiers



```
> perf <- performance(pred, measure="tpr", x.measure="fpr")
> plot(perf, colorize=TRUE, lwd=3, asp=1, main="ROC curve")
```

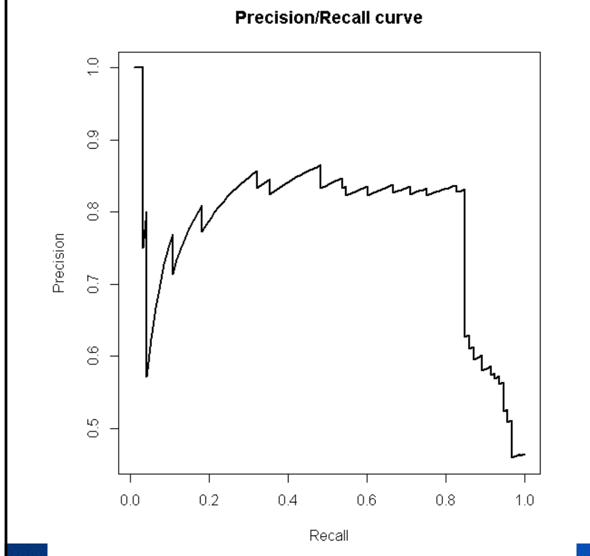
```
abline(a=0, b=1)
```

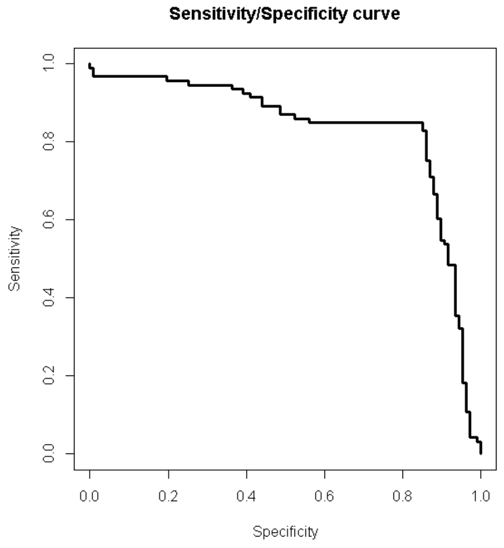
ROC curves: measure="tpr", x.measure="fpr".

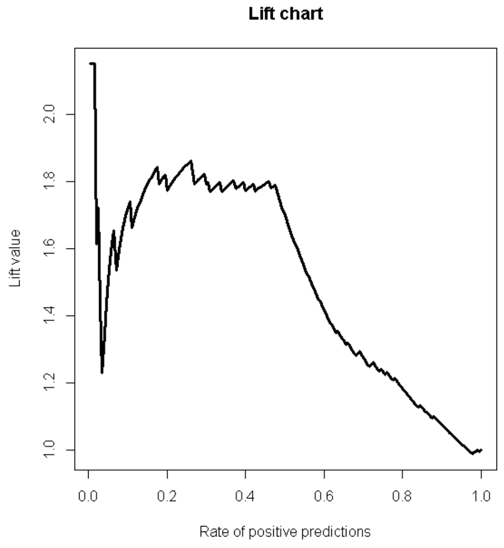
Precision/recall graphs: measure="prec", x.measure="rec".

Sensitivity/specificity plots: measure="sens", x.measure="spec".

Lift charts: measure="lift", x.measure="rpp".







NOTE: use `add = TRUE` to add more curves on the plot e.g., `plot(svm.perf, add = TRUE)`

Plot ROC curves to compare multiple classifiers (using ROCR)
https://rpubs.com/JanpuHou/359286

ROCR: Visualizing the Performance of Scoring Classifiers

```
> # more examples
> demo(ROCR)
```

```
> acc <- performance(pred, measure = "acc")
> str(acc)
Formal class 'performance' [package "ROCR"] with 6 slots
  ..@ x.name      : chr "Cutoff"
  ..@ y.name      : chr "Accuracy"
  ..@ alpha.name  : chr "none"
  ..@ x.values    :List of 1
  .. ..$ : num [1:201] Inf 0.991 0.985 0.985 0.983 ...
  ..@ y.values    :List of 1
  .. ..$ : num [1:201] 0.67 0.675 0.68 0.685 0.68 ...
  ..@ alpha.values: list()
> acc@y.values[[1]]
  [1] 0.670 0.675 0.680 0.685 0.680 0.685 0.680...
```

```
> auc <- performance(pred, measure = "auc")
> str(auc)
> auc@y.values[[1]]
[1] 0.6957259
```

acc: Accuracy.

err: Error rate.

fpr: False positive rate.

tpr: True positive rate.

rec: Recall. Same as tpr.

sens: Sensitivity. Same as tpr.

fnr: False negative rate.

miss: Miss. Same as fnr.

tnr: True negative rate.

spec: Specificity. Same as tnr.

ppv: Positive predictive value.

prec: Precision. Same as ppv.

npv: Negative predictive value.

pcfall: Prediction-conditioned fallout.

pcmiss: Prediction-conditioned miss.

rpp: Rate of positive predictions.

rnp: Rate of negative predictions.

phi: Phi correlation coefficient.

mat: Matthews correlation coefficient.

mi: Mutual information.

chisq: Chi square test statistic.

odds: Odds ratio.

lift: Lift value. $\frac{P(\hat{Y}=\oplus|Y=\oplus)}{P(\hat{Y}=\oplus)}$.

f: Precision-recall F measure

rch: ROC convex hull.

auc: Area under the ROC curve.

aucpr: Area under the Precision/Recall curve.

prbe: Precision-recall break-even point.

cal: Calibration error.

mxe: Mean cross-entropy.

rmse: Root-mean-squared error.

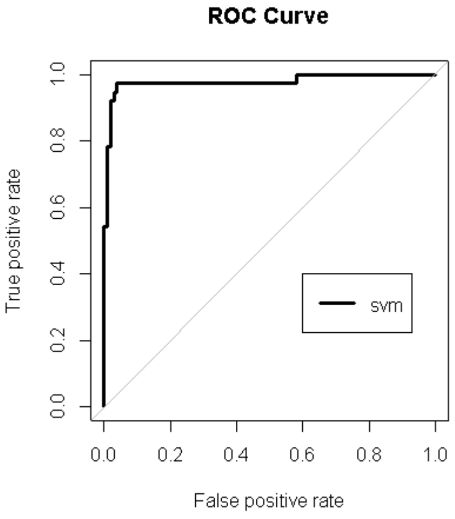
sar: Score combinining performance measures

ecost: Expected cost.

cost: Cost of a classifier

Example: Evaluate svm classifier using ROCR

```
# mlbench: Machine Learning Benchmark Problems
#install.packages("mlbench")
library(mlbench)
data(BreastCancer) # Wisconsin Breast Cancer Database
dim(BreastCancer)  # 699 x 11
levels(BreastCancer$Class) # "benign(良性)", "malignant(惡性)"
head(BreastCancer)
? BreastCancer
summary(BreastCancer)

x <- as.data.frame(lapply(BreastCancer[, -c(1, 7:11)], as.numeric))
dim(x) # 699   5
y <- BreastCancer$Class
n <- nrow(x)
p <- ncol(x)
id <- sample(1:n, n*0.8)
length(id)
x.train <- x[id, ]
x.test <- x[-id, ]
y.train <- y[id]
y.test <- y[-id]

library(e1071)
model <- svm(x.train, y.train)
summary(model)
pred.1 <- predict(model, x.test)
table(y.test, pred.1)
```



```
# compute decision values and probabilities:
pred.2 <- predict(model, x.test, decision.values = TRUE)
str(pred.2)
pred.values <- attr(pred.2, "decision.values")
pred.values

# since Levels: benign < malignant
pred.svm <- prediction(-pred.values, y.test)
perf.svm <- performance(pred.svm, "tpr", "fpr")
plot(perf.svm, lwd=3, main="ROC Curve")
abline(a=0, b=1, col="grey")
legend(0.6, 0.4, "svm", lty=1, lwd=3)
auc <- performance(pred.svm, measure = "auc")
auc@y.values[[1]]
```

More example:
https://rpubs.com/JanpuHou/359286

K-Nearest Neighbors

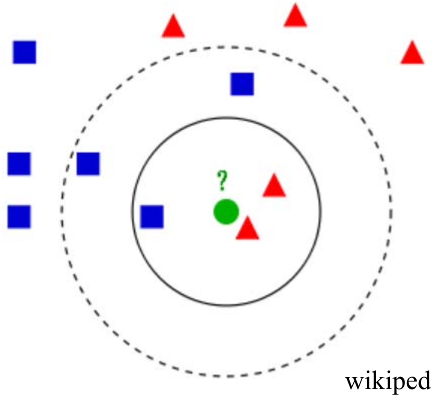
The number of k-nearest neighbors is user-defined.

1. Counts the k-nearest samples (in Euclidean distance) in the training set to the new sample to be classified.

2. Determines the proportion of neighbor samples from each class and makes a 'vote' for each class.

3. Calculates p-values for the likelihood of observed representation of each class.

4. Computes the ratio between the p-value of the most highly represented class and the p-value of the next most highly represented class.



wikipedia

5. Allows "no prediction" result if differential between p-values is above Decision cutoff for P-value ratio.

knn {class}
k-Nearest Neighbour Classification

class: Functions for Classification

Various functions for classification, including k-nearest neighbour, Learning Vector Quantization and Self-Organizing Maps.

```
> library(class)
> sel <- sample(1:50, 30)
> iris.train <- rbind(iris3[sel,,1], iris3[sel,,2], iris3[sel,,3])
> iris.test <- rbind(iris3[-sel,,1], iris3[-sel,,2], iris3[-sel,,3])
> y.train <- factor(c(rep("s", 30), rep("c", 30), rep("v", 30)))
> y.test.true <- factor(c(rep("s", 20), rep("c", 20), rep("v", 20)))
> y.test.pred <- knn(train=iris.train, test=iris.test, cl=y.train, k = 3)
> ct <- table(y.test.true, y.test.pred)
> ct
           y.test.pred
y.test.true  c  s  v
          c 18  0  2
          s  0 20  0
          v  1  0 19
> (accuracy <- sum(diag(ct))/sum(ct))
[1] 0.95
```

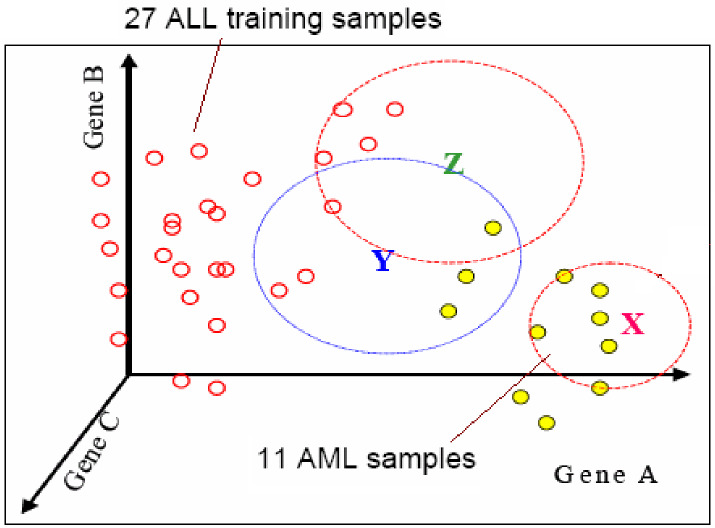
```
> iris3[1:3,,]
, , Setosa

     Sepal L. Sepal W. Petal L. Petal W.
[1,]      5.1      3.5      1.4      0.2
[2,]      4.9      3.0      1.4      0.2
[3,]      4.7      3.2      1.3      0.2

, , Versicolor

     Sepal L. Sepal W. Petal L. Petal W.
[1,]      7.0      3.2      4.7      1.4
[2,]      6.4      3.2      4.5      1.5
[3,]      6.9      3.1      4.9      1.5

, , Virginica

     Sepal L. Sepal W. Petal L. Petal W.
[1,]      6.3      3.3      6.0      2.5
[2,]      5.8      2.7      5.1      1.9
[3,]      7.1      3.0      5.9      2.1
```

Apply KNN to Microarray Data

- *#Samples*: Bone marrow
  - #ALL (acute lymphoblastic leukemia): 27 patients (急性淋巴細胞白血病)
  - #AML (acute myeloid leukemia): 11 patients (急性骨髓性白血病)
  - *#Genes*: 7070 genes.

Number of Predictor genes: 3
Number of neighbors: 6
Decision cutoff P-value ratio: 0.2

Test Sample	ALL vote	ALL p-value	AML vote	AML p-value	P value ratio	Prediction
X	0	1	6	0	0 (0.0 is less than decision cutoff p-value ratio, 0.2 ⇒ predicts sample X as AML)	AML
Y	3	.953**	3	.221**	.232 (0 .232 is higher than decision cutoff p-value ratio, 0.2 ⇒ do not make prediction for sample Y)	Not predicted
Z	5	.429	1	.893	.480 (0.480 is higher than decision cutoff p-value ratio, 0.2 ⇒ do not make prediction for sample Z)	Not predicted



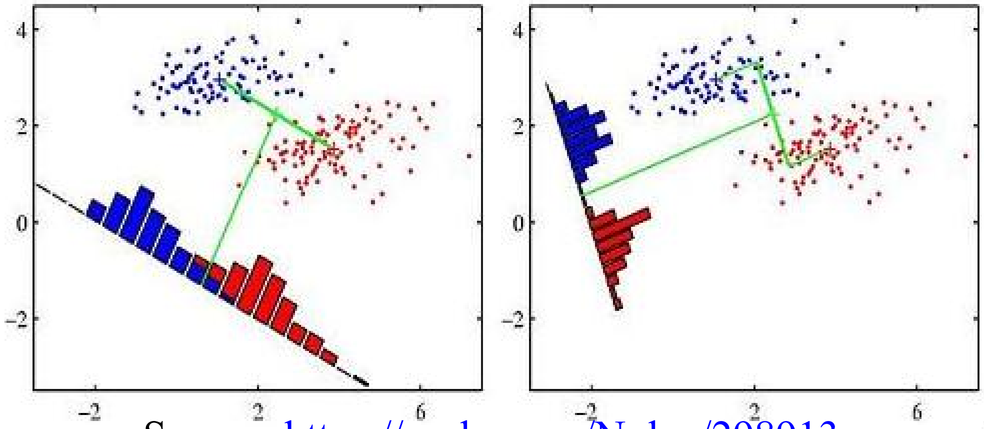
Golub, T.R et al. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286, 531--537. Cancer Genomics Program at Whitehead Institute for Genome Research
http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi
被引用 12815 次

Linear Discriminant Analysis (LDA)

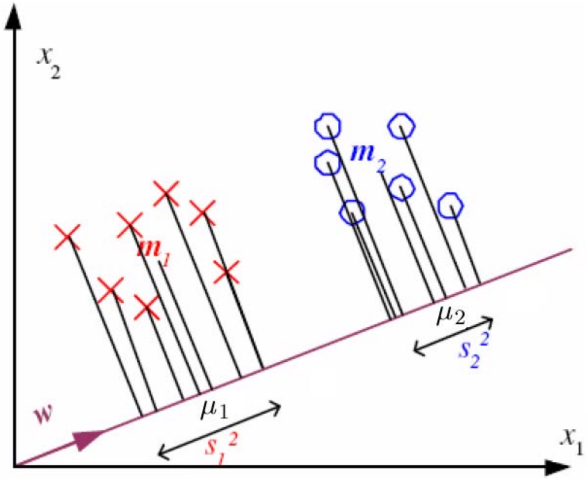
- LDA (Fisher, 1936) finds the linear combinations $w^T x$ of $x = (x_1,\ldots, x_p)$ with large ratios of between-groups to within-groups sum of squares.

- LDA is a supervised method for dimension reduction for classification problem.

- Given samples from two classes *C1* and *C2*, we want to find the direction, as defined by a vector *w*, such that when the data are projected onto *w*, the examples from the two classes are as well separated as possible.



Source: https://rpubs.com/Nolan/298913



LDA: Methodology (1/3)

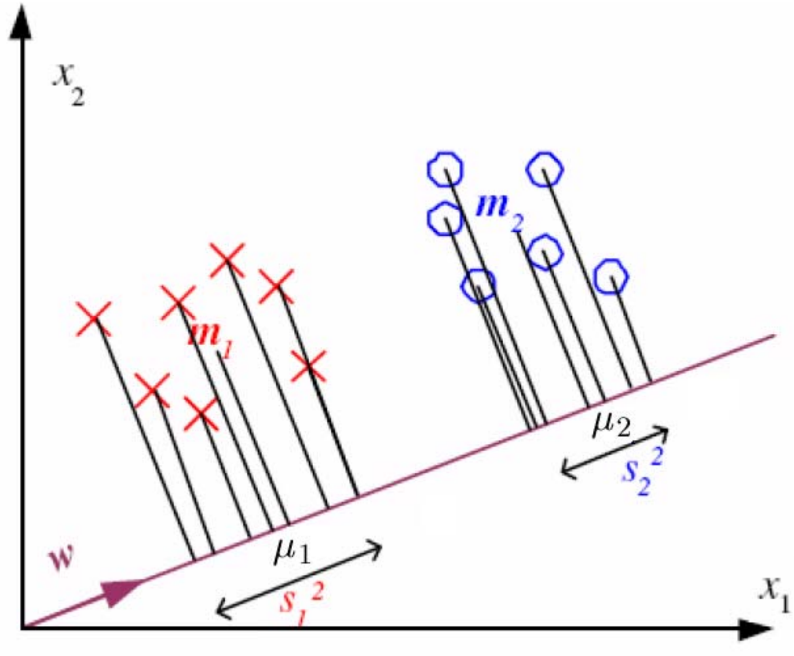
$z = \mathbf{w}^T \mathbf{x}$

the projection of $\mathbf{x}$ onto $\mathbf{w}$ and thus is a dimension reduction from d to 1.

$X = \{\mathbf{x}^t, r^t\}$, $r^t = 1$, if $\mathbf{x}^t \in C_1$ $r^t = 0$, if $\mathbf{x}^t \in C_2$

$\mathbf{m}_1 \in \mathcal{R}^d$: the means of samples from C_1 before projection

$\mu_1 \in \mathcal{R}^1$: the means of samples from C_1 after projection



LDA: Methodology (2/3)

$$\mu_1 = \frac{\sum_t^N \mathbf{w}^T\mathbf{x}^t r^t}{\sum_t^N r^t} = \mathbf{w}^T\mathbf{m}_1 \qquad \mu_2 = \frac{\sum_t^N \mathbf{w}^T\mathbf{x}^t(1-r^t)}{\sum_t^N (1-r^t)} = \mathbf{w}^T\mathbf{m}_2$$

$$s_1^2 = \sum_t (\mathbf{w}^T\mathbf{x}^t - \mu_1)^2 r^t \qquad s_2^2 = \sum_t (\mathbf{w}^T\mathbf{x}^t - \mu_2)^2 (1-r^t)$$

$$\boxed{J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}}$$

Want $|\mu_1 - \mu_2|$ to be large and $s_1^2 + s_2^2$ to be small →

$$\boxed{J(\mathbf{w}) = \frac{(\mu_1-\mu_2)^2}{s_1^2+s_2^2}}$$

$$\begin{aligned}(\mu_1-\mu_2)^2 &= (\mathbf{w}^T\mathbf{m}_1 - \mathbf{w}^T\mathbf{m}_2)^2 \\ &= \mathbf{w}^T(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^T\mathbf{w} \\ &= \mathbf{w}^T\mathbf{S}_B\mathbf{w}\end{aligned}$$

$$\mathbf{S}_B = (\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^T$$

Between-class scatter matrix

$$s_1^2 + s_2^2 = \mathbf{w}^T\mathbf{S}_W\mathbf{w}$$

$$\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$$

Total within-class scatter matrix

$$\begin{aligned}s_1^2 &= \sum_t (\mathbf{w}^T\mathbf{x}^t - \mu_1)^2 r^t \\ &= \sum_t \mathbf{w}^T(\mathbf{x}^t-\mathbf{m}_1)(\mathbf{x}^t-\mathbf{m}_1)^T\mathbf{w}r^t \\ &= \mathbf{w}^T\mathbf{S}_1\mathbf{w}\end{aligned}$$

$$\mathbf{S}_1 = \sum_t r^t(\mathbf{x}^t-\mathbf{m}_1)(\mathbf{x}^t-\mathbf{m}_1)^T$$

Within-class scatter matrix for C_1

$$s_2^2 = \mathbf{w}^T\mathbf{S}_2\mathbf{w}$$

$$\mathbf{S}_2 = \sum_t (1-r^t)(\mathbf{x}^t-\mathbf{m}_2)(\mathbf{x}^t-\mathbf{m}_2)^T$$

LDA: Methodology (3/3)

$$\begin{tikzcd}
J(\mathbf{w}) = \dfrac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2} \arrow[r] \arrow[d] & J(\mathbf{w}) = \dfrac{|\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)|^2}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \arrow[d] \\
J(\mathbf{w}) = \dfrac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} & \dfrac{\partial J}{\partial \mathbf{w}} = \dfrac{\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \left(2(\mathbf{m}_1 - \mathbf{m}_2) - \dfrac{\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \mathbf{S}_W \mathbf{w}\right) = 0
\end{tikzcd}$$

$$\mathbf{w} = c\mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

- Because it is the direction that is important for us and not the magnitude, set ***c=1*** and find ***w***

NOTE: $p(\mathbf{x}|C_i) \sim N(\mu_i, \Sigma)$

Linear discriminant $\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2)$

LDA: More Than 2 Classes

$\mathbf{z} = \mathbf{W}^T\mathbf{x}$ $\quad \mathbf{z} \in \mathcal{R}^k$ $\quad \mathbf{W} = [d \times k]$

The within-class scatter matrix for C_i

$$\mathbf{S}_i = \sum_t r_i^t (\mathbf{x}^t - \mathbf{m}_i)(\mathbf{x}^t - \mathbf{m}_i)^T$$

$r^t = 1$ if $\mathbf{x}^t \in C_i$ and 0 otherwise

The total within-class scatter

$\mathbf{S}_W = \sum_{i=1}^K \mathbf{S}_i$

The between-class scatter matrix

$$\mathbf{S}_B = \sum_{i=1}^{K} N_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

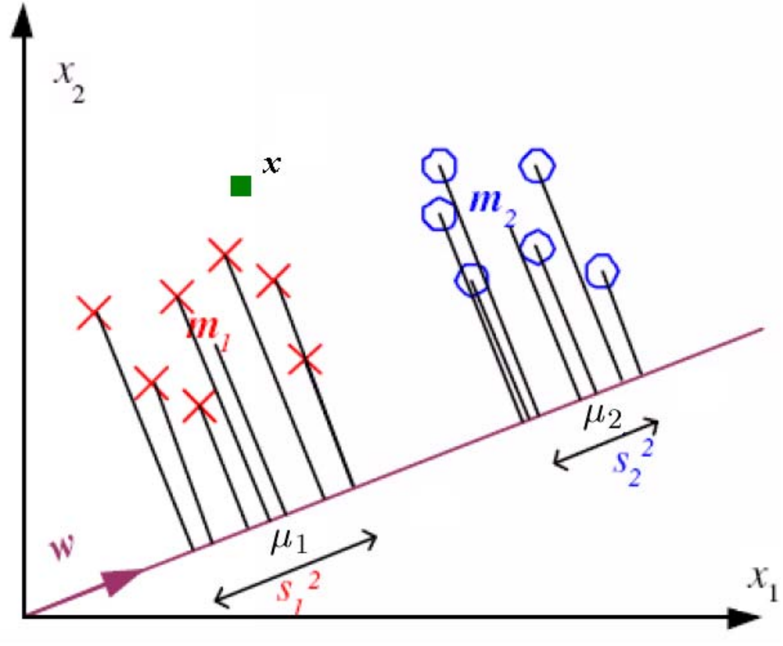
$$\mathbf{m} = \frac{1}{K}\sum_{i=1}^{K} \mathbf{m}_i \qquad N_i = \sum_t r_i^t$$

$$J(\mathbf{W}) = \frac{\mathbf{W}^T \mathbf{S}_B \mathbf{W}}{\mathbf{W}^T \mathbf{S}_W \mathbf{W}}$$

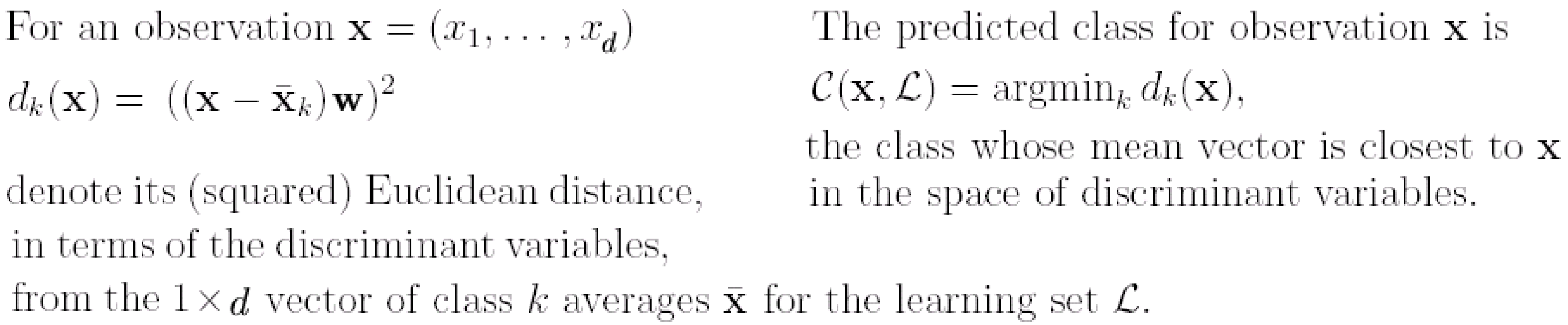
- The largest eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_B$ are the solution.
- $\mathbf{S}_B$ is the sum of K matrices of rank 1, $(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$, and only $K-1$ of them are independent.
- $\mathbf{S}_B$ has a maximum rank of $K-1$ and we take $k = K-1$.
- define a new lower $(K-1)$ dimensional space, where the discriminant is then to be constructed.

LDA: Classification

- Fisher’s linear discriminant is optimal normally distributed.
- After projection, for the two classes to would like the means to be as far apart examples of classes be scatteres in as s possible.



Classification Rules:



For an observation $\mathbf{x} = (x_1, \ldots, x_d)$

$$d_k(\mathbf{x}) = ((\mathbf{x} - \bar{\mathbf{x}}_k)\mathbf{w})^2$$

denote its (squared) Euclidean distance, in terms of the discriminant variables, from the $1 \times d$ vector of class k averages $\bar{\mathbf{x}}$ for the learning set $\mathcal{L}$.

The predicted class for observation $\mathbf{x}$ is

$$\mathcal{C}(\mathbf{x}, \mathcal{L}) = \text{argmin}_k \, d_k(\mathbf{x}),$$

the class whose mean vector is closest to $\mathbf{x}$ in the space of discriminant variables.

LDA Assumptions

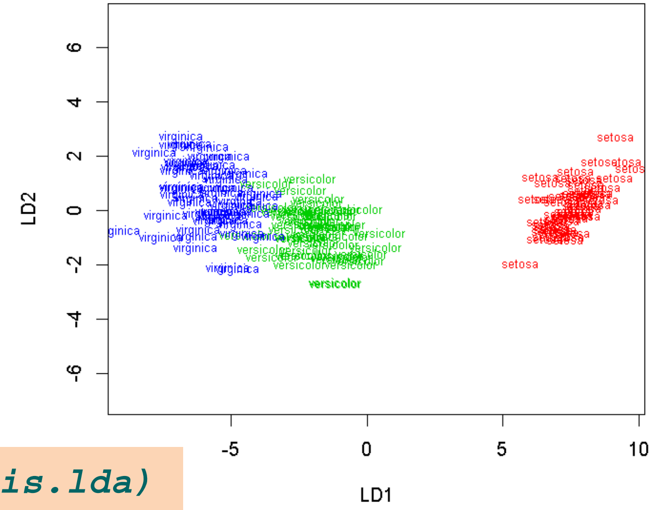
- The predictors are multivariate normal within groups. This assumption implies that the predictors have linear relationships.
- Homogeneity of Covariance (within groups).
- Independence

`aq.plot {mvoutlier}`: Detect Outliers
`shapiro.test {stats}`: Shapiro-Wilk Normality Test
`mshapiro.test {mvnormtes}`: Normality test for multivariate variables
`bartlett.test {stats}`: Test of Homogeneity of Variances

```
# LDA in R
lda {MASS}
partimat {klaR}
train {caret}
LDA {flipMultivariates}
discrimin {ade4}
```

LDA in R: lda {MASS}

```
> library(MASS)
> data <- iris[,1:4]
> class <- iris[,5]
> iris.lda <- lda(x=data, grouping=class)
> # sam as iris.lda <- lda(Species ~ ., iris)
> iris.lda
Call:
lda(data, grouping = class)

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.5

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.8293776  0.02410215
Sepal.Width   1.5344731  2.16452123
Petal.Length -2.2012117 -0.93192121
Petal.Width  -2.8104603  2.83918785

Proportion of trace:
   LD1    LD2
0.9912 0.0088
> plot(iris.lda, col=as.integer(class)+1)
```

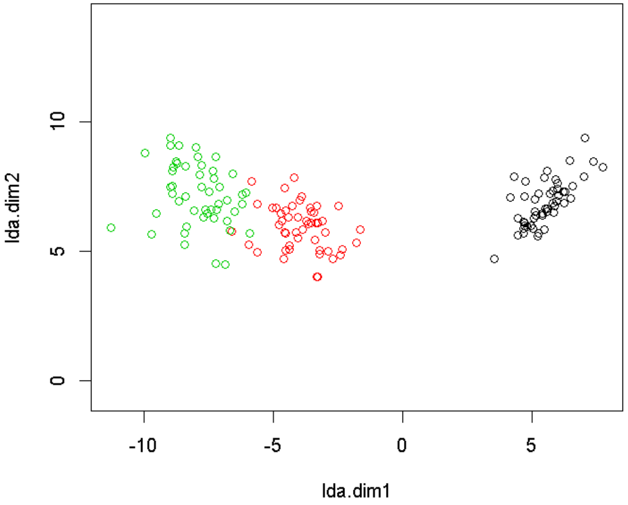


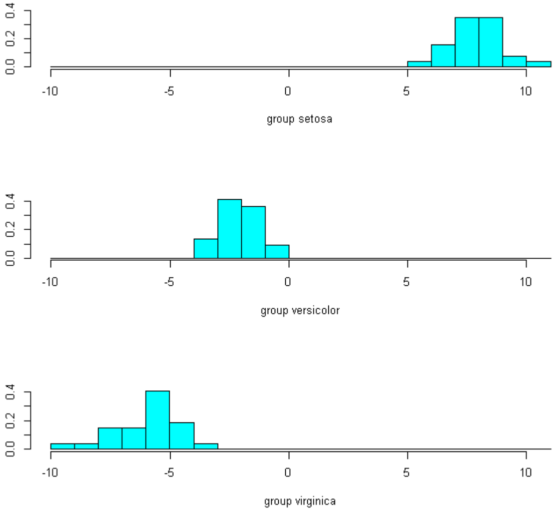
```
> iris.lda.predict <- predict(iris.lda)
```

```
> fit <-lda(x=data, grouping=class, CV=TRUE)
> (ct <- table(class, fit$class))
class        setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          1        49
> diag(prop.table(ct, 1))
    setosa versicolor  virginica
      1.00       0.96       0.98
> # total percent correct
> sum(diag(prop.table(ct)))
[1] 0.98
```

LDA in R: lda {MASS}

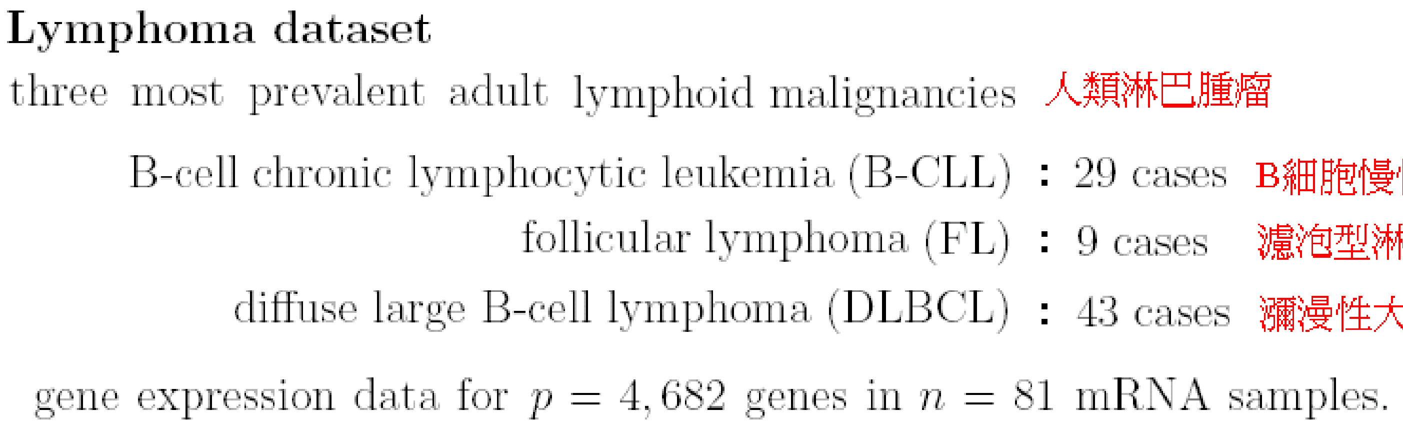
```
> lda.dim1 <- as.matrix(data)%*%iris.lda$scaling[,1]
> lda.dim2 <- as.matrix(data)%*%iris.lda$scaling[,2]
> plot(lda.dim1, lda.dim2, col=class, asp=1)
>
> ## LDA for classification
> set.seed(123456)
> trainingIndex <- sample(1:150, 75)
> trainingSample <- iris[trainingIndex, ]
> testSample <- iris[-trainingIndex, ]
> table(iris$Species[trainingIndex])

    setosa versicolor  virginica
 24          28          23
>
> ldaRule <- lda(Species ~ ., iris,  subset = trainingIndex)
> # plot(ldaRule)
> ldaRule.predict <- predict(ldaRule, testSample)
> names(ldaRule.predict)
[1] "class"      "posterior" "x"
> ldahist(data = ldaRule.predict$x[,1], g=testSample$Species)
> # plot(ldaRule, dimen = 1, type = "b")
>
> table(testSample$Species, ldaRule.predict$class)

             setosa versicolor virginica
  setosa         26          0         0
  versicolor      0         21         1
  virginica       0          1        26
```





練習: Apply LDA to Microaray Data (select genes)

Dudoit S., J. Fridlyand, and T. P. Speed (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *JASA* 97 (457), 77-87.



Lymphoma dataset

three most prevalent adult lymphoid malignancies 人類淋巴腫瘤

B-cell chronic lymphocytic leukemia (B-CLL) : 29 cases B細胞慢性淋巴性白血病

follicular lymphoma (FL) : 9 cases 濾泡型淋巴瘤

diffuse large B-cell lymphoma (DLBCL) : 43 cases 瀰漫性大B細胞淋巴癌

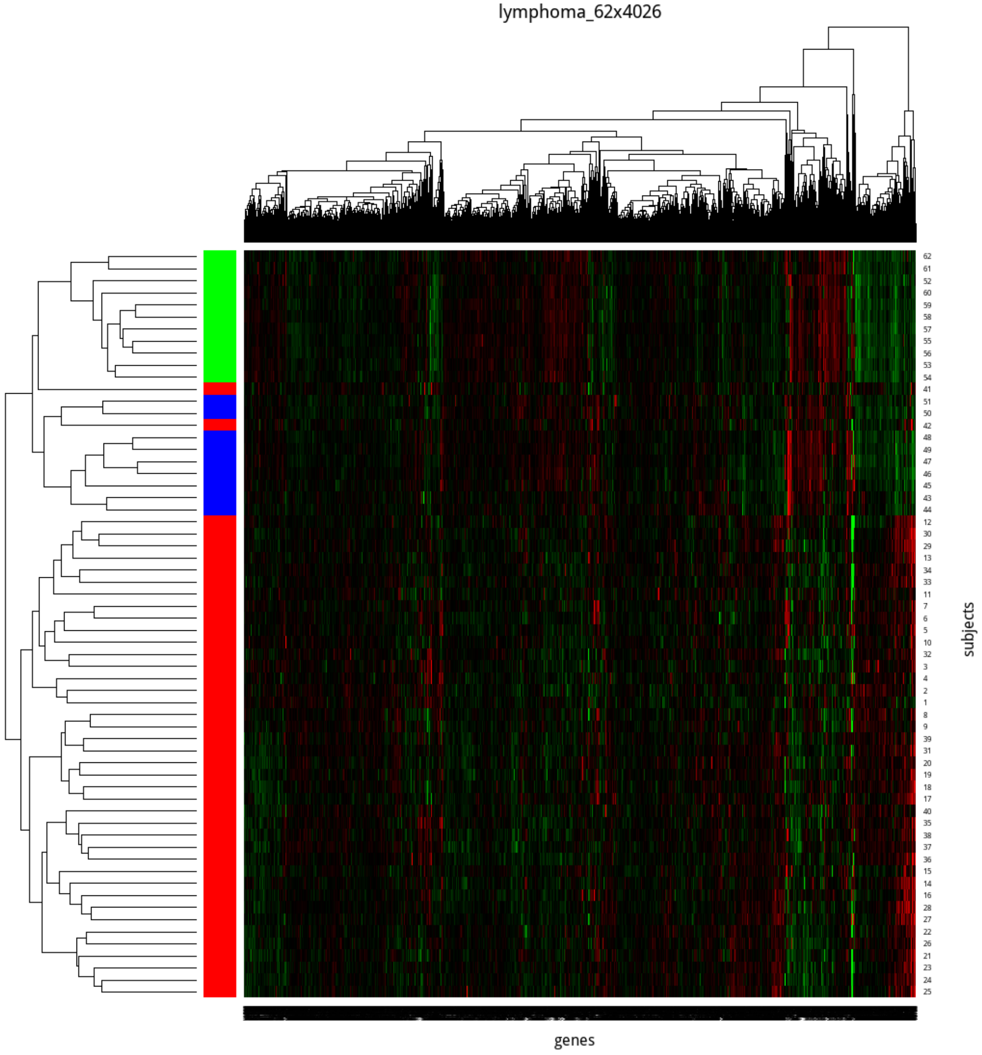
gene expression data for $p = 4,682$ genes in $n = 81$ mRNA samples.

```
> lymphoma <- read.table("lymphoma_62x4026.txt", sep="\t", row.names=1)
> dim(lymphoma)
[1]   62 4027
> lymphoma[1:5, 1:4]
 V2      V3      V4      V5
1  0 -0.3780 -0.7255 -0.5349
2  0 -1.0103 -0.9069 -0.4071
3  0  0.2696  0.1540  0.2696
4  0 -0.6809 -0.9298 -0.6809
5  0 -0.7706 -0.9311 -1.0382
> group <- lymphoma[, 1]
> xdata.orig <- lymphoma[, 2:ncol(lymphoma)]
```

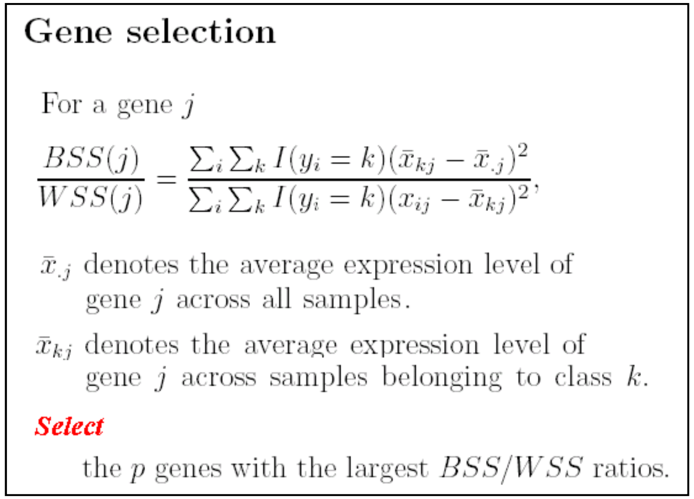
lymphoma_62x4026.txt

Heatmap for Lymphoma Microarray Data

```
> library(fields)
> gbr <- two.colors(start="green",
+                   middle="black",
+                   end="red")
> gcol <- c("red", "blue", "green")
> xdata <- xdata.orig
> range(xdata)
[1] -9.542  9.415
> xdata[xdata > 5] <- 5
> xdata[xdata < -5] <- -5
> heatmap(as.matrix(xdata), col = gbr,
+         Rowv=NULL,
+         RowSideColors = gcol[group+1],
+         margins = c(5,10),
+         xlab = "genes",
+         ylab =  "subjects",
+         main = "lymphoma_62x4026")
```



Gene Selection



Gene selection

For a gene j

$$\frac{BSS(j)}{WSS(j)} = \frac{\sum_i \sum_k I(y_i = k)(\bar{x}_{kj} - \bar{x}_{.j})^2}{\sum_i \sum_k I(y_i = k)(x_{ij} - \bar{x}_{kj})^2},$$

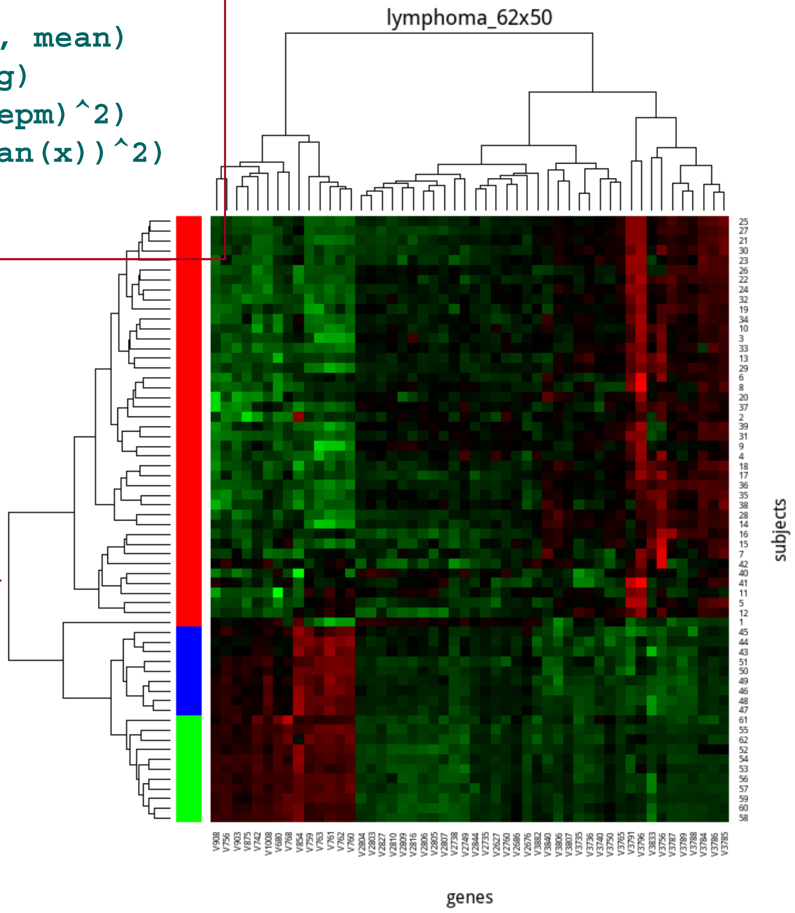
$\bar{x}_{.j}$ denotes the average expression level of gene j across all samples.

$\bar{x}_{kj}$ denotes the average expression level of gene j across samples belonging to class k.

Select

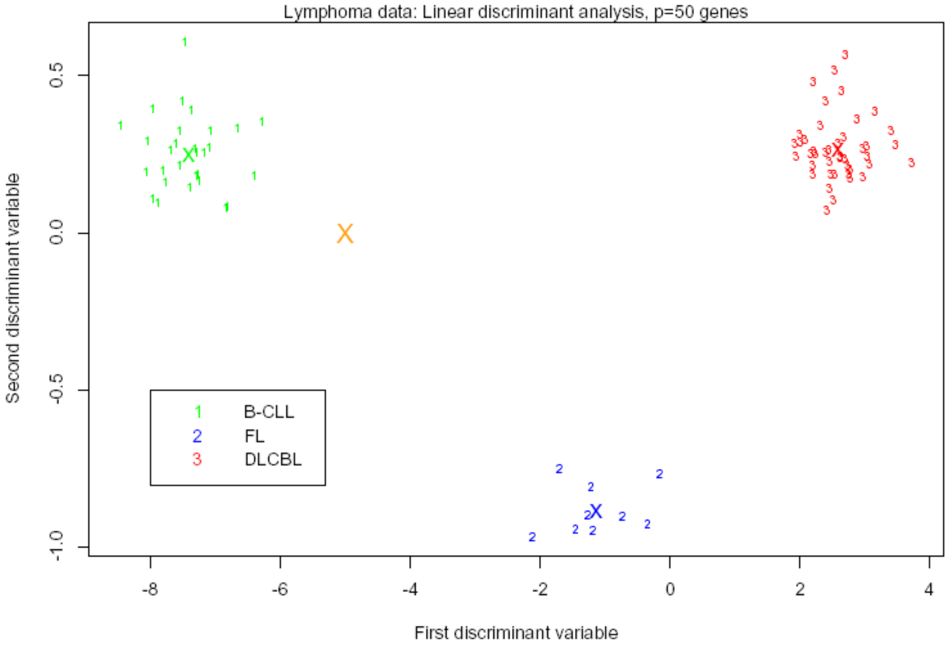
the p genes with the largest BSS/WSS ratios.

```
bw.ratio <- function(x, y){
  tg <- table(y)
  gm <- tapply(x, y, mean)
  repm <- rep(gm, tg)
  wss <- sum((x - repm)^2)
  bss <- sum((gm-mean(x))^2)
  bw <- bss/wss
}
```



```
> bw.values <- apply(xdata.orig, 2, bw.ratio, group)
> top <- 50
> selected.genes <- order(bw.values,
                          decreasing = TRUE)[1:top]
> xdata.selected <- xdata.orig[, selected.genes]
> range(xdata.selected)
[1] -6.127  9.415
> xdata.selected[xdata.selected > 5] <- 5
> xdata.selected[xdata.selected < -5] <- -5
> heatmap(as.matrix(xdata.selected), col = gbr, Rowv=NULL,
+         RowSideColors = c("red", "blue", "green")[group+1],
+         margins = c(5,10),
+         xlab = "genes", ylab =  "subjects", main = "lymphoma_62x50")
```

練習: Apply LDA to Microaray Data



Why Discriminant Analysis: Compare with logistic regression

- When the classes of the response variable Y are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. LDA & QDA do not suffer from this problem.
- If n is small and the distribution of the predictors X is approximately normal in each of the classes, the LDA & QDA models are again more stable than the logistic regression model.
- LDA & QDA are often preferred over logistic regression when we have more than two non-ordinal response classes.
- LDA & QDA have assumptions that are often more restrictive then logistic regression.

Source: http://uc-r.github.io/discriminant_analysis

The Assumptions of LDA & QDA

- Both LDA and QDA assume the the predictor variables X are drawn from a multivariate Gaussian distribution.
- LDA assumes equality of covariances among the predictor variables X across each all levels of Y. This assumption is relaxed with the QDA model.
- LDA and QDA require the number of predictor variables (p) to be less then the sample size (n).
- The performance will severely decline as p approaches n.
- A simple rule of thumb is to use LDA & QDA on data sets where $n \geq 5\times p$.

Source: http://uc-r.github.io/discriminant_analysis

Compare LDA with QDA

- LDA is a much less flexible classifier than QDA, and so has substantially lower variance.

- If LDA's assumption that the predictor variable share a common variance across each Y response class is badly off, then LDA can suffer from high bias.

- LDA tends to be a better bet than QDA if there are relatively few training observations and so reducing variance is crucial.

- In contrast, QDA is recommended if the training set is very large, so that the variance of the classifier is not a major concern, or if the assumption of a common covariance matrix is clearly untenable.

Source: http://uc-r.github.io/discriminant_analysis

HDLSS Problems in LDA

- In the high dimensional setting ($p >> n$) LDA is not appropriate for two reasons:
  - First, the standard estimate for the within-class covariance matrix is singular, and so the usual discriminant rule cannot be applied.
  - Second, when p is large, it is difficult to interpret the classification rule that is obtained from LDA, since the classification rule involves a linear combination of all p features.

- Friedman, J. H. (1989). Regularized discriminant analysis. J. Am. Stat. Assoc. 84, 165–175.
- Hastie, T., Buja, A. and Tibshirani, R. (1995) Penalized discriminant analysis. Ann. Statist., 23, 73–102.
- Guo, Y., Hastie, T. and Tibshirani, R. (2007) Regularized linear discriminant analysis and its application in microarrays. Biostatistics, 8, 86–100.
- Daniela M.Witten and Robert Tibshirani, 2011, Penalized classification using Fisher's linear discriminant, J. R. Statist. Soc. B, 73(5), 753–772.

Classical LDA

$\mathbf{X}$ be an $n \times p$ matrix

$\mathbf{X}_j$ denote feature or column j

the features are centred to have mean 0

$\mathbf{x}_i$ denote observation or row i.

$C_k \subset \{1, \ldots, n\}$ contains the indices of the observations in class k

the *within-class covariance matrix* $\boldsymbol{\Sigma}_{\mathrm{w}}$

$$\hat{\boldsymbol{\Sigma}}_{\mathrm{w}} = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in C_k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^{\mathrm{T}}$$

$\hat{\boldsymbol{\mu}}_k$ is the sample mean vector for class k

the *between-class covariance matrix* $\boldsymbol{\Sigma}_{\mathrm{b}}$

$$\hat{\boldsymbol{\Sigma}}_{\mathrm{b}} = \frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X} - \hat{\boldsymbol{\Sigma}}_{\mathrm{w}} = \frac{1}{n} \sum_{k=1}^{K} n_k \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^{\mathrm{T}}.$$

Fisher's discriminant problem

$$\text{maximize}_{\boldsymbol{\beta}_k \in \mathbb{R}^p} (\boldsymbol{\beta}_k^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_{\mathrm{b}} \boldsymbol{\beta}_k) \text{ subject to } \boldsymbol{\beta}_k^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_{\mathrm{w}} \boldsymbol{\beta}_k \leqslant 1, \boldsymbol{\beta}_k^{\mathrm{T}} \hat{\boldsymbol{\Sigma}}_{\mathrm{w}} \boldsymbol{\beta}_i = 0 \qquad \forall i < k.$$

Positive Definite Estimate

- A matrix is "positive definite" if all of its eigenvalues are positive.
- Sample covariance matrix is always positive semi-definite.
- Sample covariance matrix is positive definite: full rank.
- If the input covariance or correlation matrix being analyzed is not positive definite:
  - Generalized least squares (GLS) estimation requires that the covariance or correlation matrix analyzed must be positive definite, and
  - maximum likelihood (ML) estimation will also perform poorly in such situations.

$$\text{maximize}_{\beta_k \in \mathbb{R}^p} (\beta_k^{\mathrm{T}} \hat{\Sigma}_{\mathrm{b}} \beta_k) \text{ subject to } \beta_k^{\mathrm{T}} \tilde{\Sigma}_{\mathrm{w}} \beta_k \leqslant 1, \beta_k^{\mathrm{T}} \tilde{\Sigma}_{\mathrm{w}} \beta_i = 0 \qquad \forall i < k,$$

$\tilde{\Sigma}_{\mathrm{w}}$ is a positive definite estimate for Σ_{w}

addresses the singularity issue,
but not the interpretability issue.

- Krzanowski, W. J., Jonathan, P., McCarthy, W. V. and Thomas, M. R. (1995) Discriminant analysis with singular covariance matrices: methods and applications to spectroscopic data. Appl. Statist., 44, 101-115.
- Xu, P., Brock, G. and Parrish, R. (2009) Modified linear discriminant analysis approaches for classification of high-dimensional microarray data. Computnl Statist. Data Anal., 53, 1674-1687.

General Form of Penalized Linear Discriminant Analysis

We define the *first penalized discriminant vector* $\hat{\boldsymbol{\beta}}_1$ to be the solution to the problem

$$\text{maximize}_{\boldsymbol{\beta}_1}\{\boldsymbol{\beta}_1^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_{\mathrm{b}}\boldsymbol{\beta}_1 - P_1(\boldsymbol{\beta}_1)\} \quad \text{subject to} \quad \boldsymbol{\beta}_1^{\mathrm{T}}\tilde{\boldsymbol{\Sigma}}_{\mathrm{w}}\boldsymbol{\beta}_1 \leqslant 1,$$

where $\tilde{\boldsymbol{\Sigma}}_{\mathrm{w}}$ is a positive definite estimate for $\boldsymbol{\Sigma}_{\mathrm{w}}$ and where P_1 is a convex penalty function.

Penalized LDA-L_1 method

$$\text{maximize}_{\boldsymbol{\beta}_k}\left(\boldsymbol{\beta}_k^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_{\mathrm{b}}^{k}\boldsymbol{\beta}_k - \lambda_k\sum_{j=1}^{p}|\hat{\sigma}_j\beta_{kj}|\right) \text{subject to } \boldsymbol{\beta}_k^{\mathrm{T}}\tilde{\boldsymbol{\Sigma}}_{\mathrm{w}}\boldsymbol{\beta}_k \leqslant 1.$$

$\hat{\sigma}_j$ is the within-class standard deviation for feature j;

Penalized LDA-FL method

$$\text{maximize}_{\boldsymbol{\beta}_k}\left(\boldsymbol{\beta}_k^{\mathrm{T}}\hat{\boldsymbol{\Sigma}}_{\mathrm{b}}^{k}\boldsymbol{\beta}_k - \lambda_k\sum_{j=1}^{p}|\hat{\sigma}_j\beta_{kj}| - \gamma_k\sum_{j=2}^{p}|\hat{\sigma}_j\beta_{kj} - \hat{\sigma}_{j-1}\beta_{k,j-1}|\right) \quad \text{subject to} \quad \boldsymbol{\beta}_k^{\mathrm{T}}\tilde{\boldsymbol{\Sigma}}_{\mathrm{w}}\boldsymbol{\beta}_k \leqslant 1.$$

fused lasso penalty (Tibshirani *et al.*, 2005)

(least absolute shrinkage and selection operator)

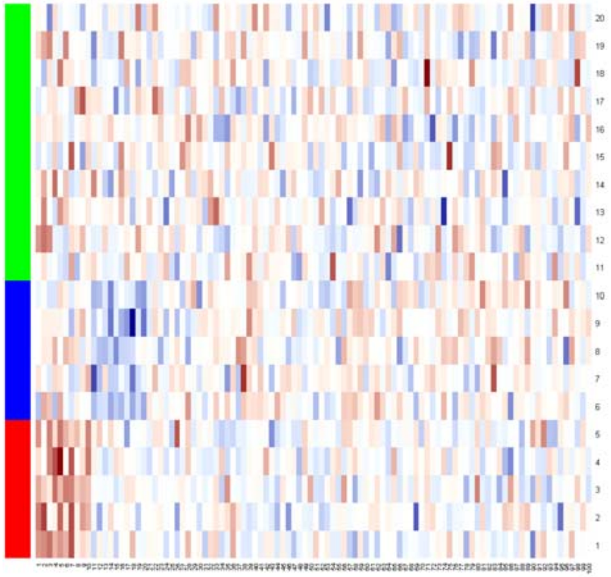
PenalizedLDA {penalizedLDA}

Solve Fisher's discriminant problem in high-dimensions using (a) a diagonal estimate of the within-class covariance matrix, and (b) lasso `(type="standard")` or fused lasso `(type="ordered")` penalties on the discriminant vectors.

Usage

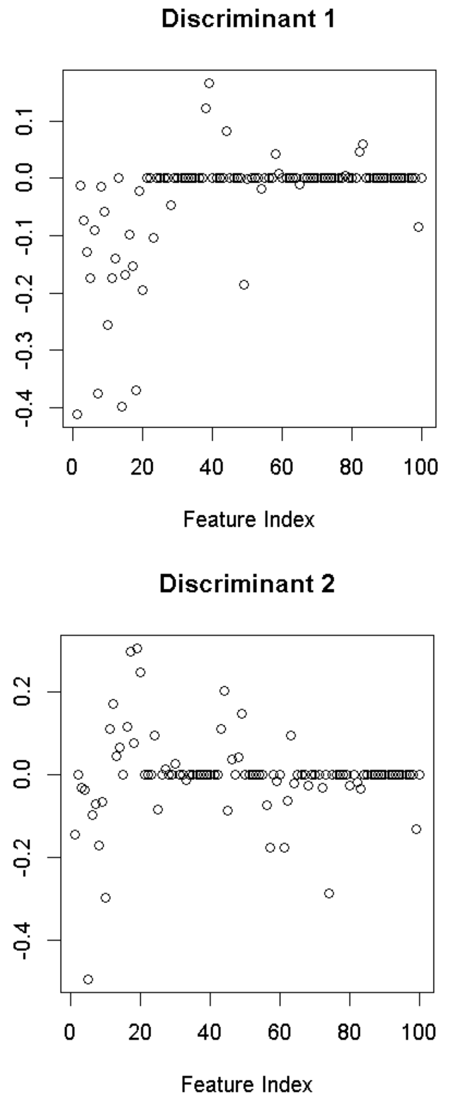
```
PenalizedLDA(x, y, xte=NULL, type = "standard", lambda, K = 2, chrom =
NULL, lambda2 = NULL, standardized = FALSE, wcsd.x = NULL, ymat = NULL,
maxiter = 20, trace=FALSE)
```

```
> library(penalizedLDA)
> library(fields)
>
> set.seed(12345)
> # an example modified from penalizedLDA package
> n <- 20
> p <- 100
> x.train <- matrix(rnorm(n*p), ncol=p)
> x.test <- matrix(rnorm(n*p), ncol=p)
> y <- c(rep(1, 5), rep(2, 5), rep(3, 10))
>
> x.train[y==1, 1:10] <- x.train[y==1, 1:10] + 2
> x.train[y==2, 11:20] <- x.train[y==2, 11:20] - 2
>
> x.test[y==1, 1:10] <- x.test[y==1, 1:10] + 2
> x.test[y==2, 11:20] <- x.test[y==2, 11:20] - 2
>
> heatmap(x.train, Rowv=NA, Colv=NA, RowSideColors=c("red", "blue", "green")[y],
+         col=two.colors(start="darkblue", middle="white", end="darkred"))
```



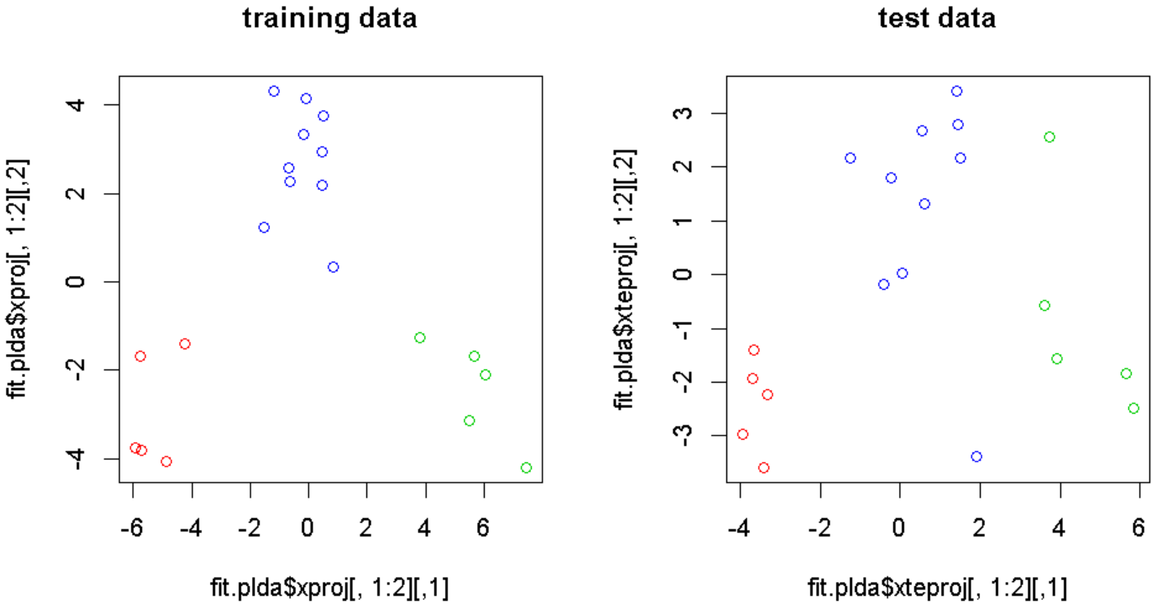
PenalizedLDA {penalizedLDA}

```
> fit.plda <- PenalizedLDA(x.train, y, x.test, lambda=0.14, K=2)
> print(fit.plda)
Number of discriminant vectors:  2
Number of nonzero features in discriminant vector  1 : 34
Number of nonzero features in discriminant vector  2 : 43
Total number of nonzero features:  54

Details:
Type:  standard
Lambda:  0.14
> plot(fit.plda)
> str(fit.plda)
List of 12
 $ ypred  : int [1:20, 1:2] 1 1 1 1 1 2 3 2 2 2 ...
 $ discrim: num [1:100, 1:2] -0.412 -0.012 -0.0738 -0.1294 ...
 $ xproj  : num [1:20, 1:2] -5.74 -4.86 -4.24 -5.71 -5.94 ...
 $ xteproj: num [1:20, 1:2] -3.42 -3.67 -3.94 -3.32 -3.65 ...
 $ K      : num 2
 $ crits  :List of 2
  ..$ : num [1:13] -0.931 3.283 3.335 3.342 3.345 ...
  ..$ : num [1:7] -0.882 0.937 0.954 0.955 0.955 ...
 $ type   : chr "standard"
 $ lambda : num 0.14
 $ lambda2: NULL
 $ wcsd.x : num [1:100] 0.733 1.099 1.252 1.102 0.974 ...
 $ x      : num [1:20, 1:100] 2.59 2.71 1.89 1.55 2.61 ...
 $ y      : num [1:20] 1 1 1 1 1 2 2 2 2 2 ...
 - attr(*, "class")= chr "penlda"
```



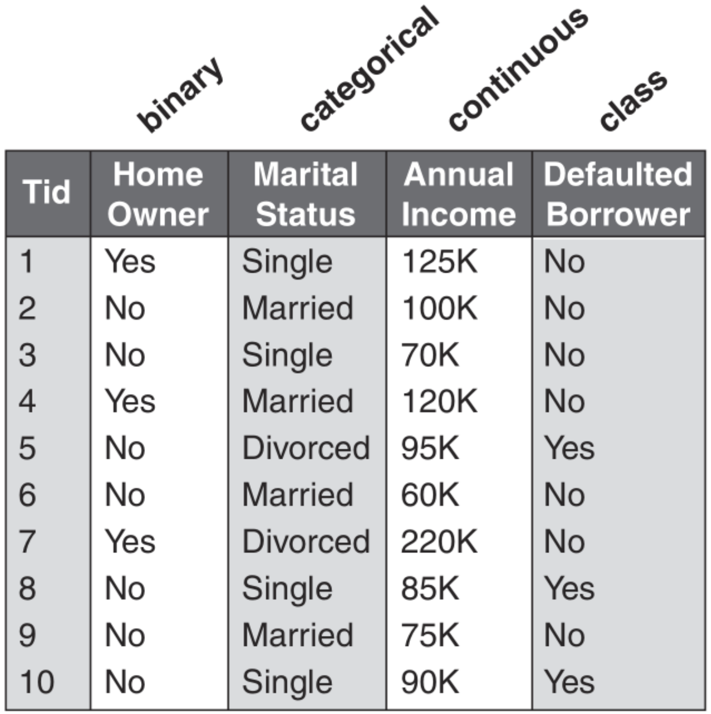
PenalizedLDA {penalizedLDA}

```
> par(mfrow=c(1, 2))
> plot(fit.plda$xproj[,1:2], col=y+1, main="training data")
> plot(fit.plda$xteproj[,1:2], col=fit.plda$ypred+1, main="test data")
> pred.fit.plda <- predict(fit.plda, xte=x.test)
> pred.fit.plda
$ypred
      [,1] [,2]
 [1,]    1    1
 [2,]    1    1
 [3,]    1    1
 [4,]    1    1
 [5,]    1    1
 [6,]    2    2
 [7,]    3    2
...
[18,]    3    3
[19,]    3    3
[20,]    3    3
```

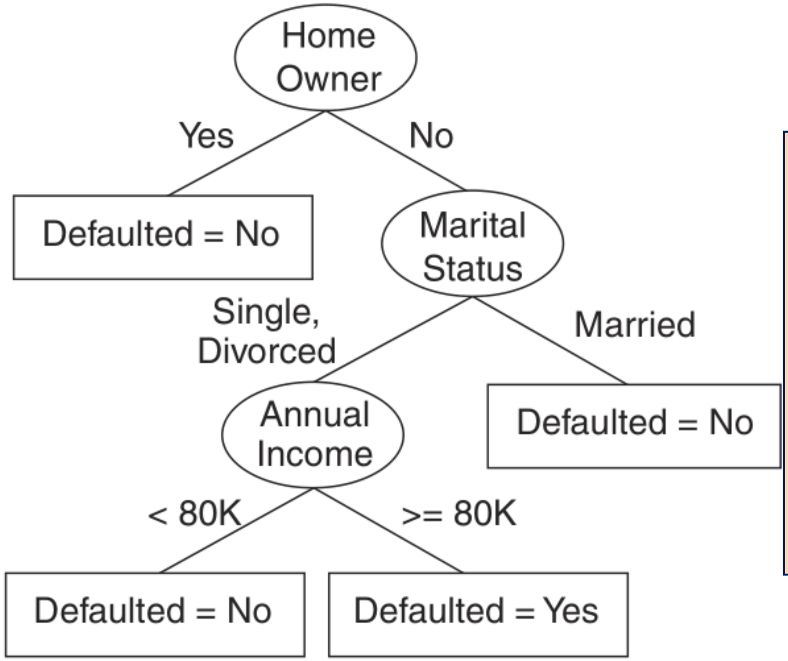


Decision Tree (決策樹)

- A decision trees is a classifier expressed as a **recursive partition** of the instance space.
- A decision tree consists of internal nodes that represent the decisions corresponding to the hyperplanes or split points (i.e., which half-space a given point lies in), and leaf nodes that represent regions or partitions of the data space, which are labeled with the majority class.



	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



To classify a new test point we have to recursively evaluate which half-space it belongs to until we reach a leaf node in the decision tree, at which point we predict its class as the label of the leaf.

Images source: Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining 1st Edition, Publisher: Pearson; 1 edition (May 12, 2005)

Decision Tree (決策樹)

- A decision tree uses an **axis-parallel hyperplane** to split the data space $\boldsymbol{R}$ into two resulting half-spaces or regions, say $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$, which also induces a partition of the input points into $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$, respectively.
- Each of these regions is recursively split via axis-parallel hyperplanes until the points within an induced partition are relatively pure in terms of their class labels.
- The resulting hierarchy of split decisions constitutes the decision tree model, with the leaf nodes labeled with the majority class among points in those regions.

Figure 1. Sample decision tree based on binary target variable Y

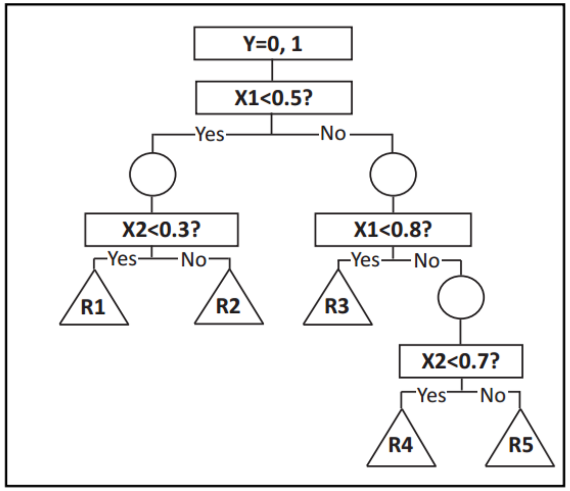
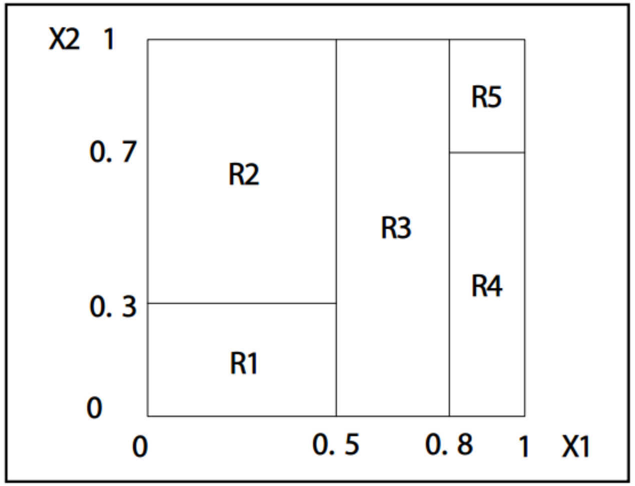


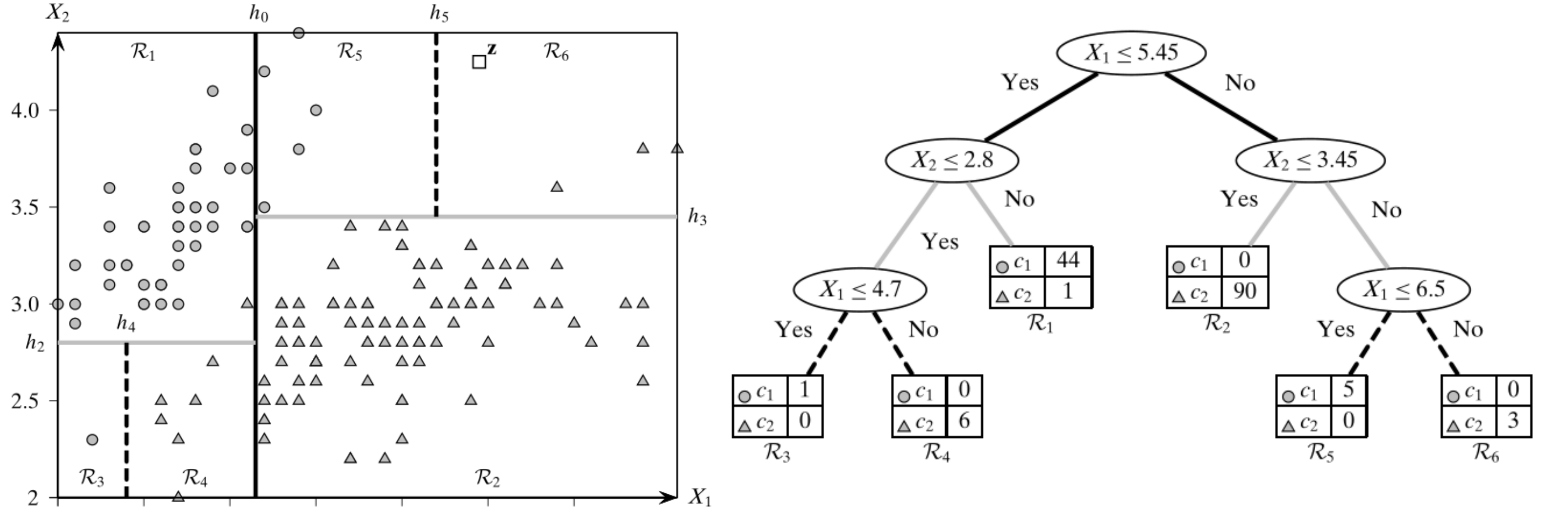
Figure 2. Decision tree illustrated using sample space view



Yan-yan SONG and Ying LU, Decision tree methods: applications for classification and prediction, Shanghai Arch Psychiatry. 2015 Apr 25; 27(2): 130–135.

Decision Rules

- A tree can be read as set of decision rules, with each rule's antecedent comprising the decisions on the internal nodes along a path to a leaf, and its consequent being the label of the leaf node.
- Because the regions are all disjoint and cover the entire space, the set of rules can be interpreted as a set of alternatives or disjunctions.



(b) Decision Tree

(a) Recursive Splits

sepal length (X_1) and sepal width (X_2).

c_1, iris-setosa (in circles),
c_2, the other two types of Irises (in triangles).

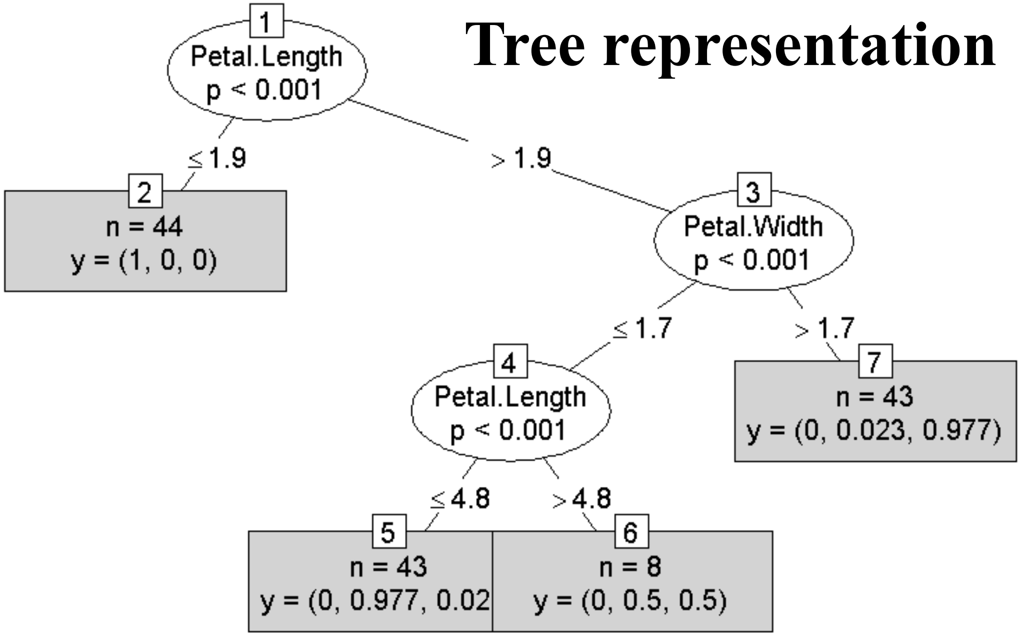
Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, May 2014.

Representation of a Decision Tree

Rule representation

```
 Conditional inference tree with 4 terminal nodes

Response:  Species
Inputs:  Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
Number of observations:  138

1) Petal.Length <= 1.9; criterion = 1, statistic = 128.431
  2)*  weights = 44
1) Petal.Length > 1.9
  3) Petal.Width <= 1.7; criterion = 1, statistic = 63.498
    4) Petal.Length <= 4.8; criterion = 0.999, statistic = 13.638
      5)*  weights = 43
    4) Petal.Length > 4.8
      6)*  weights = 8
  3) Petal.Width > 1.7
    7)*  weights = 43
```

Tree representation



Axis-Parallel Hyperplanes

- Training dataset $\mathbf{D}=\{\mathbf{x}_i, y_i\}$: consist of n points in a d-dimensional space, $\mathbf{x}_i$: numeric variables; y_i: class label.
- A hyperplane $h(\mathbf{x})$ is defined as the set of all points $\mathbf{x}$ that satisfy

$$h(\mathbf{x}) : \mathbf{w}^T\mathbf{x} + b = 0 \qquad \mathbf{w} \in \mathbb{R}^d$$

- $\mathbf{w}$ is a weight vector that is normal to the hyperplane, and b is the **offset** of the hyperplane from the origin.
- A decision tree considers only axis-parallel hyperplanes, that is, the weight vector must be parallel to one of the original dimensions or axes $\mathbf{x}_j$.

Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, May 2014.

Split Points and Data Partition

- A hyperplane specifies a decision or split point because it splits the data space $\boldsymbol{R}$ into two half-spaces. All points $\boldsymbol{x}$ such that $h(x) \leq 0$ are on the hyperplane or to one side of the hyperplane, whereas all points such that $h(x) > 0$ are on the other side.

- The generic form of a split point for a numeric attribute X_j is given as $X_j \leq v$

- The decision or split point $X_j \leq v$ thus splits the input data space $\boldsymbol{R}$ into two regions $\boldsymbol{R}_Y$ and $\boldsymbol{R}_N$, which denote the set of all possible points that satisfy the decision and those that do not.

- A split point of the form $X_j \leq v$ induces the data partition

$$\mathbf{D}_Y = \{\mathbf{x} \mid \mathbf{x} \in \mathbf{D}, x_j \leq v\}$$

$$\mathbf{D}_N = \{\mathbf{x} \mid \mathbf{x} \in \mathbf{D}, x_j > v\}$$

Purity

- The purity of a region R_j is defined in terms of the mixture of classes for points in the corresponding data partition D_j.

- Formally, purity is the fraction of points with the majority label in D_j, that is

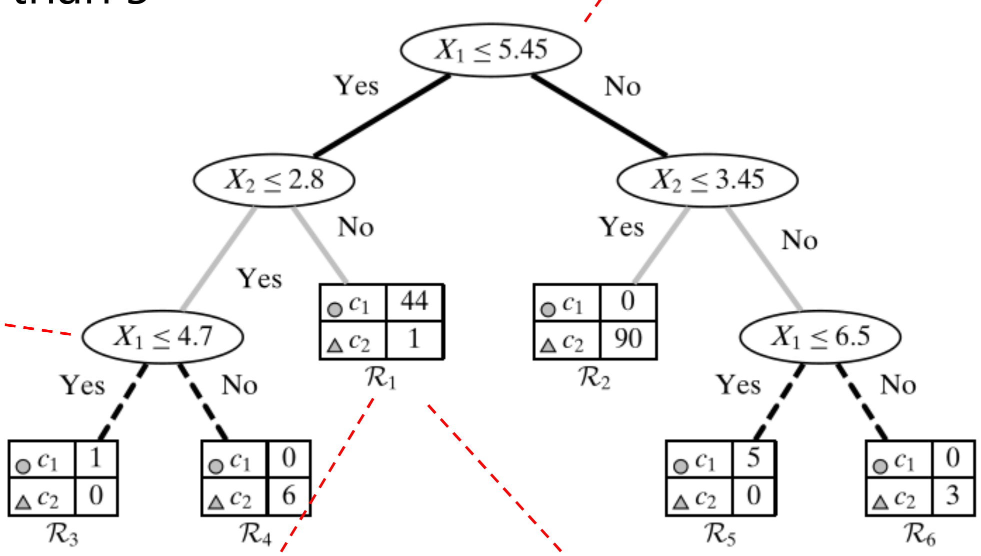
$$purity(\mathbf{D}_j) = \max_i \left\{ \frac{n_{ji}}{n_j} \right\}$$

where $n_j = |D_j|$ is the total number of data points in the region R_j ,and n_{ji} is the number of points in D_j with class label c.

Purity Example

- Use a size threshold of 5 and a purity threshold of 0.95 in this example.
- A region will be split further only if the number of points is more than 5 and the purity is less than 0.9

$h_1(\mathbf{x}) : x_1 - 5.45 = 0$



(b) Decision Tree

The purity is $6/7 = 0.857$.

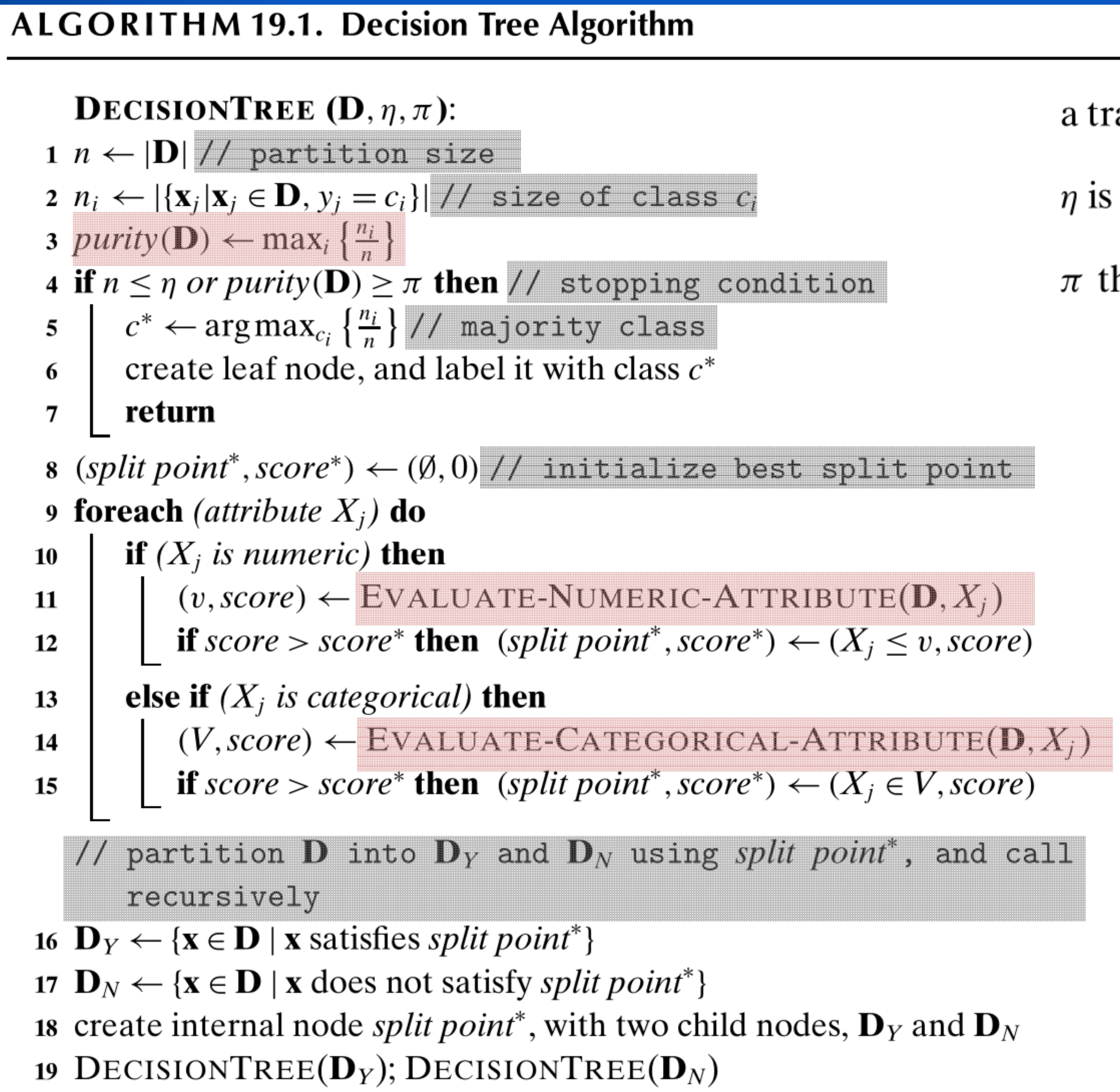
the region has more than five points, and its purity is less than 0.95, it is further split via the hyperplane $h_4(\mathbf{x}) : x_1 - 4.7 = 0$

the probability of misclassification in region $\mathcal{R}_1$ (the error rate for that leaf) is $1/45 = 0.022$

the entire region ($\mathcal{R}_1$) is labeled with the majority class c_1.

Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, May 2014.

Decision Tree Algorithm



ALGORITHM 19.1. Decision Tree Algorithm

```
DECISIONTREE ($\mathbf{D}, \eta, \pi$):
$n \leftarrow |\mathbf{D}|$ // partition size
$n_i \leftarrow |\{\mathbf{x}_j | \mathbf{x}_j \in \mathbf{D}, y_j = c_i\}|$ // size of class $c_i$
$purity(\mathbf{D}) \leftarrow \max_i \left\{\frac{n_i}{n}\right\}$
if $n \leq \eta$ or $purity(\mathbf{D}) \geq \pi$ then // stopping condition
    $c^* \leftarrow \arg\max_{c_i} \left\{\frac{n_i}{n}\right\}$ // majority class
    create leaf node, and label it with class $c^*$
    return
$(split\ point^*, score^*) \leftarrow (\emptyset, 0)$ // initialize best split point
foreach (attribute $X_j$) do
    if ($X_j$ is numeric) then
        $(v, score) \leftarrow$ EVALUATE-NUMERIC-ATTRIBUTE($\mathbf{D}, X_j$)
        if $score > score^*$ then $(split\ point^*, score^*) \leftarrow (X_j \leq v, score)$
    else if ($X_j$ is categorical) then
        $(V, score) \leftarrow$ EVALUATE-CATEGORICAL-ATTRIBUTE($\mathbf{D}, X_j$)
        if $score > score^*$ then $(split\ point^*, score^*) \leftarrow (X_j \in V, score)$
// partition $\mathbf{D}$ into $\mathbf{D}_Y$ and $\mathbf{D}_N$ using $split\ point^*$, and call
   recursively
$\mathbf{D}_Y \leftarrow \{\mathbf{x} \in \mathbf{D} \mid \mathbf{x}$ satisfies $split\ point^*\}$
$\mathbf{D}_N \leftarrow \{\mathbf{x} \in \mathbf{D} \mid \mathbf{x}$ does not satisfy $split\ point^*\}$
create internal node $split\ point^*$, with two child nodes, $\mathbf{D}_Y$ and $\mathbf{D}_N$
DECISIONTREE($\mathbf{D}_Y$); DECISIONTREE($\mathbf{D}_N$)
```

a training dataset $\mathbf{D}$

η is the leaf size

π the leaf purity threshold.

Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, May 2014.

Split Point Evaluation Measures: Entropy

- Entropy measures, $H(\mathbf{D})$, the amount of disorder or uncertainty in a system.
  - In the classification setting, a partition has **lower entropy** (or low disorder) if it is **relatively pure** (points from the same class).
  - A partition has higher entropy (or more disorder) if the class labels are mixed, and there is no majority class as such.

$$\boxed{H(\mathbf{D}) = -\sum_{i=1}^{k} P(c_i|\mathbf{D}) \log_2 P(c_i|\mathbf{D})}$$

$P(c_i|\mathbf{D})$ is the probability of class c_i in $\mathbf{D}$,

- If a region is **pure**, then the entropy is **zero**.
- If the classes are all mixed up, and each appears with equal probability, $P(c_i|\mathbf{D}) = \frac{1}{k}$, then the entropy has the highest value: $H(\mathbf{D}) = \log_2 k$.

Split Entropy and Information Gain

- Assume that a split point partitions $\mathbf{D}$ into $\mathbf{D}_Y$ and $\mathbf{D}_N$. The split entropy is defined as the weighted entropy of each of the resulting partitions.

$$\boxed{H(\mathbf{D}_Y, \mathbf{D}_N) = \frac{n_Y}{n} H(\mathbf{D}_Y) + \frac{n_N}{n} H(\mathbf{D}_N)}$$

$n = |\mathbf{D}|$ is the number of points in $\mathbf{D}$
$n_Y = |\mathbf{D}_Y|, \ n_N = |\mathbf{D}_N|$
are the number of points in $\mathbf{D}_Y$ and $\mathbf{D}_N$.

- To see if the split point results in a reduced overall entropy, define the information gain for a given split point as:

$$\boxed{Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N) = H(\mathbf{D}) - H(\mathbf{D}_Y, \mathbf{D}_N)}$$

- The higher the information gain, the more the reduction in entropy, and the better the split point.
- Given split points and their corresponding partitions, we can score each split point and choose the one that gives the highest information gain.

Gini Index

- Gini index is used to measure the purity of a split point:

$$G(\mathbf{D}) = 1 - \sum_{i=1}^{k} P(c_i|\mathbf{D})^2$$

- If the partition is pure, then the probability of the majority class is 1 and the probability of all other classes is 0, and thus, the Gini index is 0.

- When each class is equally represented, with probability $p(c_i \mid \mathbf{D})=1/k$, then the Gini index has value $(k-1)/k$.

- Higher values of the Gini index indicate more disorder, and lower values indicate more order in terms of the class labels.

Other Measures

- The lower the **Weighted Gini index** value, the better the split point.

$$G(\mathbf{D}_Y, \mathbf{D}_N) = \frac{n_Y}{n} G(\mathbf{D}_Y) + \frac{n_N}{n} G(\mathbf{D}_N)$$

- The **Classification And Regression Trees (CART)** measure prefers a split point that maximizes the difference between the class probability mass function for the two partitions; the higher the CART measure, the better the split point.

$$CART(\mathbf{D}_Y, \mathbf{D}_N) = 2\frac{n_Y}{n}\frac{n_N}{n}\sum_{i=1}^{k}\left|P(c_i|\mathbf{D}_Y) - P(c_i|\mathbf{D}_N)\right|$$

$$\text{Classification error}(\mathbf{D}) \quad = \quad 1 - \max_i P(c_i|\mathbf{D})$$

The Class Probability Mass Function for a Partition

- If X is a numeric attribute, we have to evaluate split points of the form $X \leq v$.
- Consider only the midpoints between two successive distinct values for X in the sample $\mathbf{D}$.
- Let $\{v_1, \ldots, v_m\}$ denote the set of all such midpoints, such that $v_1 < v_2 < \cdots < v_m$.
- For each split point $X \leq v$, we have to estimate the class PMFs:

$$\hat{P}(c_i|\mathbf{D}_Y) = \hat{P}(c_i|X \leq v)$$

$$\hat{P}(c_i|\mathbf{D}_N) = \hat{P}(c_i|X > v)$$

Review: Breslow, L. A.and Aha,D. W. (1997).Simplifyingdecision trees:Asurvey.Knowledge Engineering Review, 12(1): 1–40.

Estimate the Class PMFs

Using the Bayes theorem

$$\hat{P}(c_i|X \le v) = \frac{\hat{P}(X \le v|c_i)\hat{P}(c_i)}{\hat{P}(X \le v)} = \frac{\hat{P}(X \le v|c_i)\hat{P}(c_i)}{\sum_{j=1}^{k} \hat{P}(X \le v|c_j)\hat{P}(c_j)}$$

$$N_{vi} = \sum_{j=1}^{n} I(x_j \le v \text{ and } y_j = c_i)$$

Define N_{vi} as the number of points $x_j \le v$ with class c_i

$$\hat{P}(c_i) = \frac{1}{n}\sum_{j=1}^{n} I(y_j = c_i) = \frac{n_i}{n}$$

The prior probability for each class in **D**

$$\hat{P}(X \le v|c_i) = \frac{\hat{P}(X \le v \text{ and } c_i)}{\hat{P}(c_i)} = \left(\frac{1}{n}\sum_{j=1}^{n} I(x_j \le v \text{ and } y_j = c_i)\right) \Big/ (n_i/n)$$

$$= \frac{N_{vi}}{n_i}$$

Review: Breslow, L. A.and Aha,D. W. (1997).Simplifyingdecision trees:Asurvey.Knowledge Engineering Review, 12(1): 1–40.

Estimate the Class PMFs

$$\boxed{\hat{P}(c_i|\mathbf{D}_Y) = \hat{P}(c_i|X \le v) = \frac{N_{vi}}{\sum_{j=1}^{k} N_{vj}}}$$

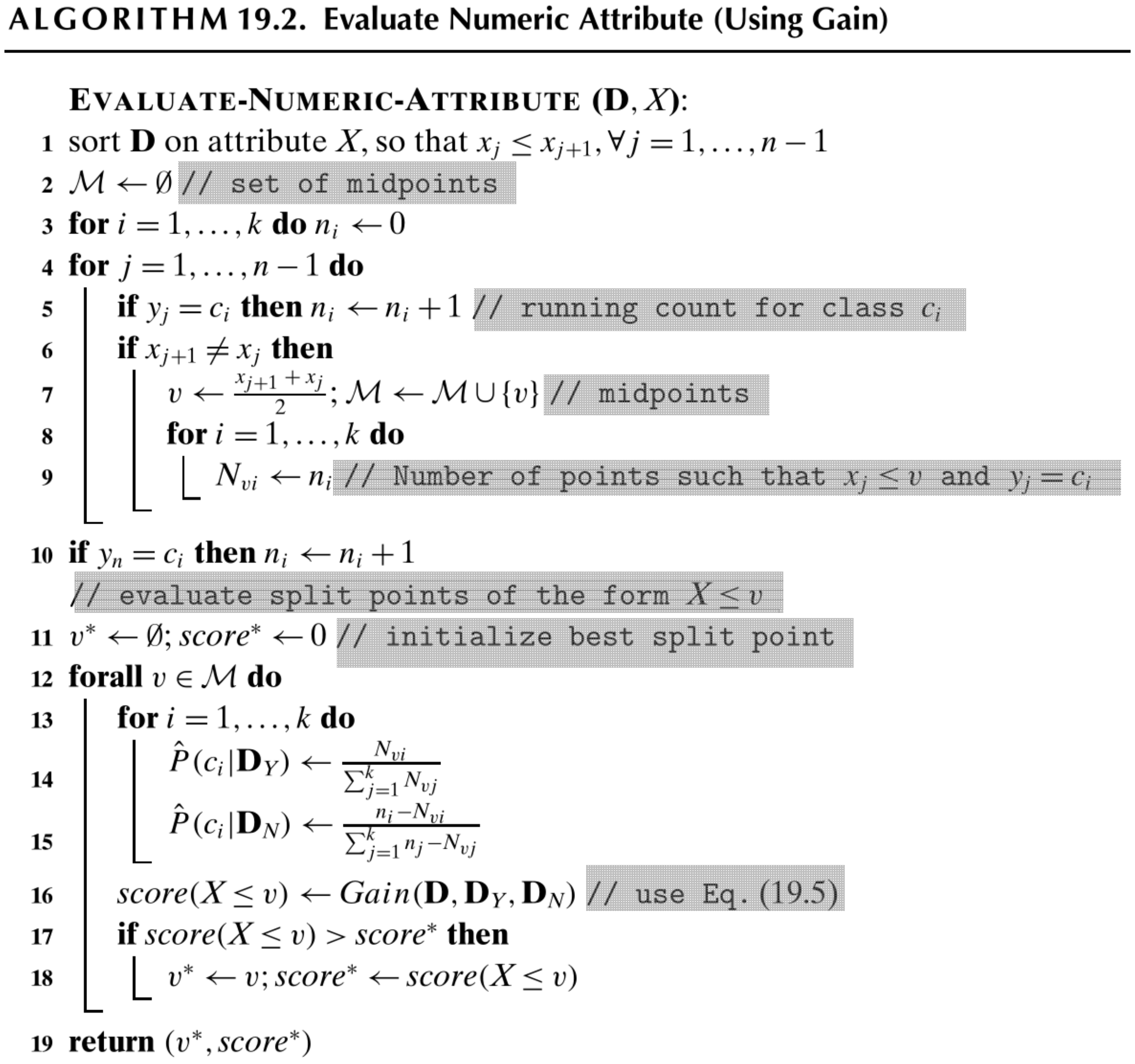
class PMF $\hat{P}(c_i|\mathbf{D}_N)$

$$\hat{P}(X > v|c_i) = 1 - \hat{P}(X \le v|c_i) = 1 - \frac{N_{vi}}{n_i} = \frac{n_i - N_{vi}}{n_i}$$

$$\hat{P}(c_i|\mathbf{D}_N) = \hat{P}(c_i|X > v) = \frac{\hat{P}(X > v|c_i)\hat{P}(c_i)}{\sum_{j=1}^{k} \hat{P}(X > v|c_j)\hat{P}(c_j)} = \frac{n_i - N_{vi}}{\sum_{j=1}^{k}(n_j - N_{vj})}$$

Review: Breslow, L. A.and Aha,D. W. (1997).Simplifyingdecision trees:Asurvey.Knowledge Engineering Review, 12(1): 1–40.

Evaluate Numeric Attribute



ALGORITHM 19.2. Evaluate Numeric Attribute (Using Gain)

```
EVALUATE-NUMERIC-ATTRIBUTE (D, X):
sort D on attribute $X$, so that $x_j \le x_{j+1}, \forall j = 1, \ldots, n-1$
$\mathcal{M} \leftarrow \emptyset$ // set of midpoints
for $i = 1, \ldots, k$ do $n_i \leftarrow 0$
for $j = 1, \ldots, n-1$ do
    if $y_j = c_i$ then $n_i \leftarrow n_i + 1$ // running count for class $c_i$
    if $x_{j+1} \neq x_j$ then
        $v \leftarrow \frac{x_{j+1} + x_j}{2}$; $\mathcal{M} \leftarrow \mathcal{M} \cup \{v\}$ // midpoints
        for $i = 1, \ldots, k$ do
            $N_{vi} \leftarrow n_i$ // Number of points such that $x_j \le v$ and $y_j = c_i$
if $y_n = c_i$ then $n_i \leftarrow n_i + 1$
// evaluate split points of the form $X \le v$
$v^* \leftarrow \emptyset$; $score^* \leftarrow 0$ // initialize best split point
forall $v \in \mathcal{M}$ do
    for $i = 1, \ldots, k$ do
        $\hat{P}(c_i|\mathbf{D}_Y) \leftarrow \frac{N_{vi}}{\sum_{j=1}^{k} N_{vj}}$
        $\hat{P}(c_i|\mathbf{D}_N) \leftarrow \frac{n_i - N_{vi}}{\sum_{j=1}^{k} n_j - N_{vj}}$
    $score(X \le v) \leftarrow Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N)$ // use Eq. (19.5)
    if $score(X \le v) > score^*$ then
        $v^* \leftarrow v$; $score^* \leftarrow score(X \le v)$
return $(v^*, score^*)$
```

Review: Breslow, L. A.and Aha,D. W. (1997).Simplifyingdecision trees:Asurvey.Knowledge Engineering Review, 12(1): 1–40.

Characteristics of Decision Tree

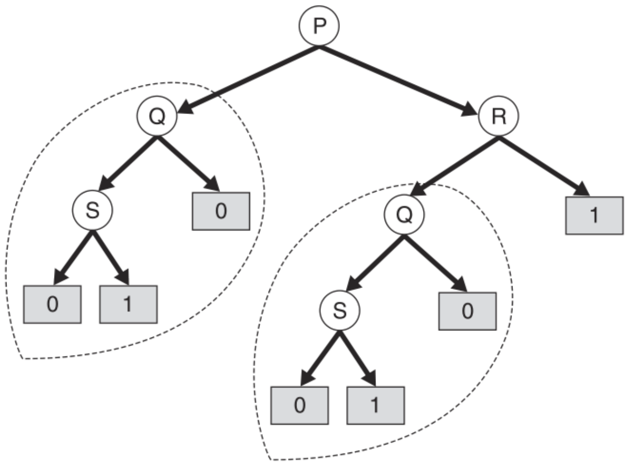
- DT Simplifies complex relationships between input variables and target variables by dividing original input variables into significant subgroups.
    - Easy to understand and interpret.
    - Non-parametric approach without distributional assumptions.
    - Easy to handle missing values without needing to resort to imputation.
    - Easy to handle heavy skewed data without needing to resort to data transformation.
    - Robust to outliers (noise).

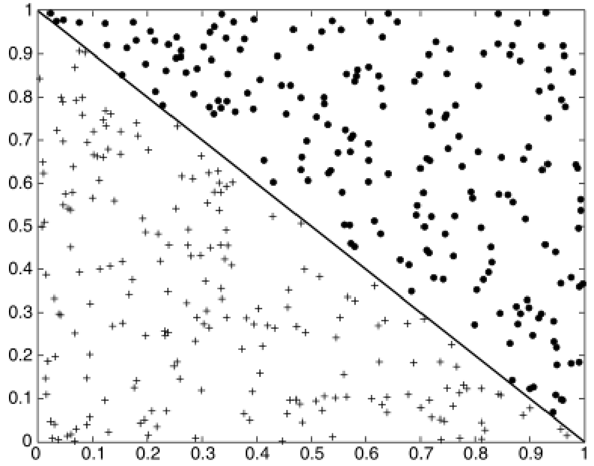
Characteristics of Decision Tree

- Decision trees do not always deliver the best performance, and represent a trade-off between performance and simplicity of explanation.
  - Finding an optimal; decision tree is an NP-complete problem. Many DT algorithms employ a heuristic-based approach to guide their search in the vast hypothesis space.
- Techniques developed for constructing DT are computationally inexpensive.
- DT provides an expressive representation for learning discrete-valued functions.
- The presence of redundant attributes does not adversely affect the accuracy of DTs.
- Data fragmentation problem: at the leaf nodes, the number of records may be too small to make a statistically significant decision about the class representation of the nodes.

Characteristics of Decision Tree

- The choice of impurity measure has little effect on the performance of decision tree. The strategy used to prune the tree has a greater impact on the final tree than the choice of impurity measure.
- The decision tree structure can represent both classification and regression models.
- **The subtree replication problem.**
- The border between two neighboring regions of different classes is known as a decision boundary. A data set may not be classified effectively by a decision tree algorithm that uses test conditions involving only a single attribute a time. (overcame)





Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining 1st Edition, Publisher: Pearson; 1 edition (May 12, 2005)

Model Overfitting

- A good classification model must not only fit the training data well, it must also accurately classify records it has never seen before.
- **Model overfitting**: a model fits the training data too well can have a poorer generalization error than a model with a higher training error.
- Some potential causes of model overfitting:
  - due to presence of noise.
  - due to lack of representative samples.
  - multiple comparison procedure.
- A survey on decision tree-pruning methods to avoid overfitting: Breslow and Aha (1997) and Esposito et al. (1997).

Popular Algorithms

- **AID** (automatic interaction detection), **CHAID** (Chi-Squared Automatic Interaction Detection)
- **ID3** (iterative dichotomizer 3rd) (Quinlan, 1986) , **C4.5** (Quinlan, 1993), **C5**
- **CART** (Classification and Regression Trees) (Breiman et al., 1984).
  - CART and C4.5 perform an exhaustive search over all possible splits maximizing an information measure of node impurity selecting the covariate showing the best split. This approach has two fundamental problems: overfitting and a selection bias towards covariates with many possible splits.
- **QUEST** (Quick, Unbiased, Efficient, Statistical Tree.) Wei-Yin Loh and Yu-Shan Shih (1997)
- **SAS algorithms**: incorporate and extend most of the good ideas discussed for recursive partitioning with univariate splits.

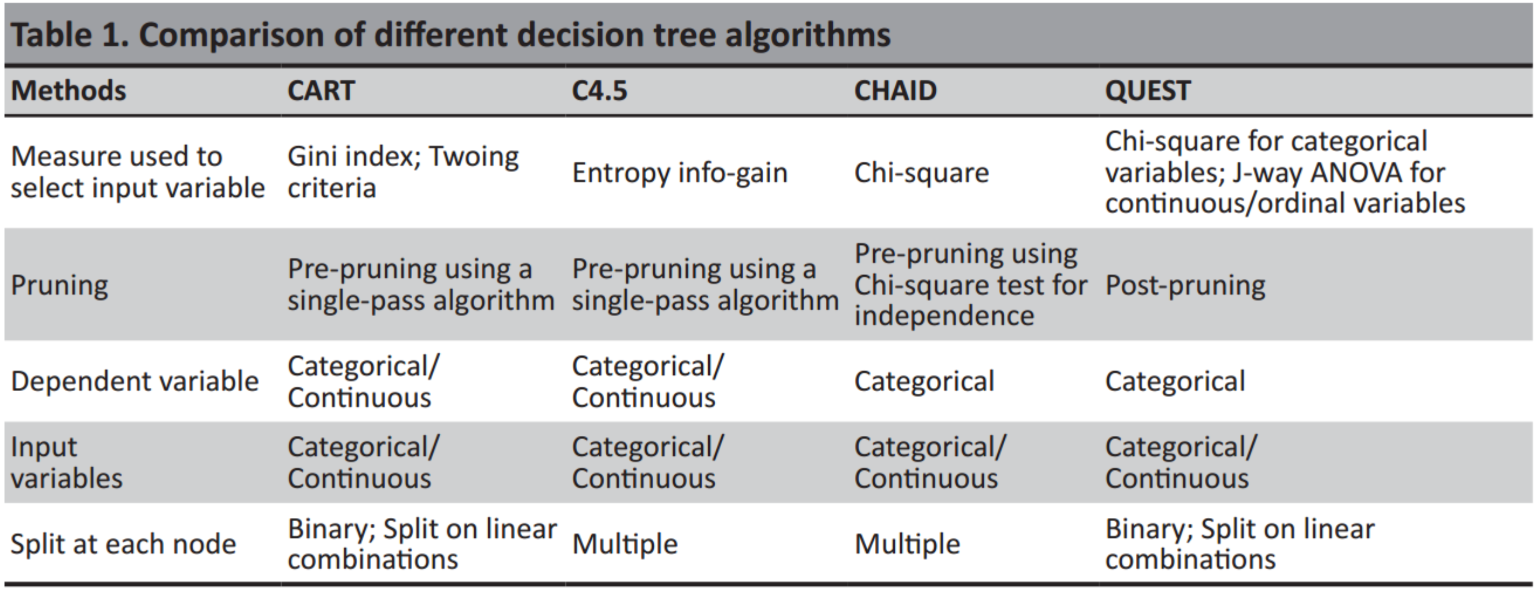


Table 1. Comparison of different decision tree algorithms

Methods	CART	C4.5	CHAID	QUEST
Measure used to select input variable	Gini index; Twoing criteria	Entropy info-gain	Chi-square	Chi-square for categorical variables; J-way ANOVA for continuous/ordinal variables
Pruning	Pre-pruning using a single-pass algorithm	Pre-pruning using a single-pass algorithm	Pre-pruning using Chi-square test for independence	Post-pruning
Dependent variable	Categorical/ Continuous	Categorical/ Continuous	Categorical	Categorical
Input variables	Categorical/ Continuous	Categorical/ Continuous	Categorical/ Continuous	Categorical/ Continuous
Split at each node	Binary; Split on linear combinations	Multiple	Multiple	Binary; Split on linear combinations

Yan-yan SONG and Ying LU, Decision tree methods: applications for classification and prediction, Shanghai Arch Psychiatry. 2015 Apr 25; 27(2): 130–135.

R Package: party

- The `party` package (Hothorn, Hornik, and Zeileis 2006) aims at providing a recursive part(y)itioning laboratory assembling various high- and low-level tools for building tree-based regression and classification models.

- This includes conditional inference trees (`ctree`), conditional inference forests (`cforest`) and parametric model trees (`mob`).
- At the core of the package is `ctree`, an implementation of conditional inference trees which embed tree-structured regression models into a well defined theory of conditional inference procedures.

- This non-parametric class of regression trees is applicable to all kinds of regression problems, including nominal, ordinal, numeric, censored as well as multivariate response variables and arbitrary measurement scales of the covariates.

R Package: party

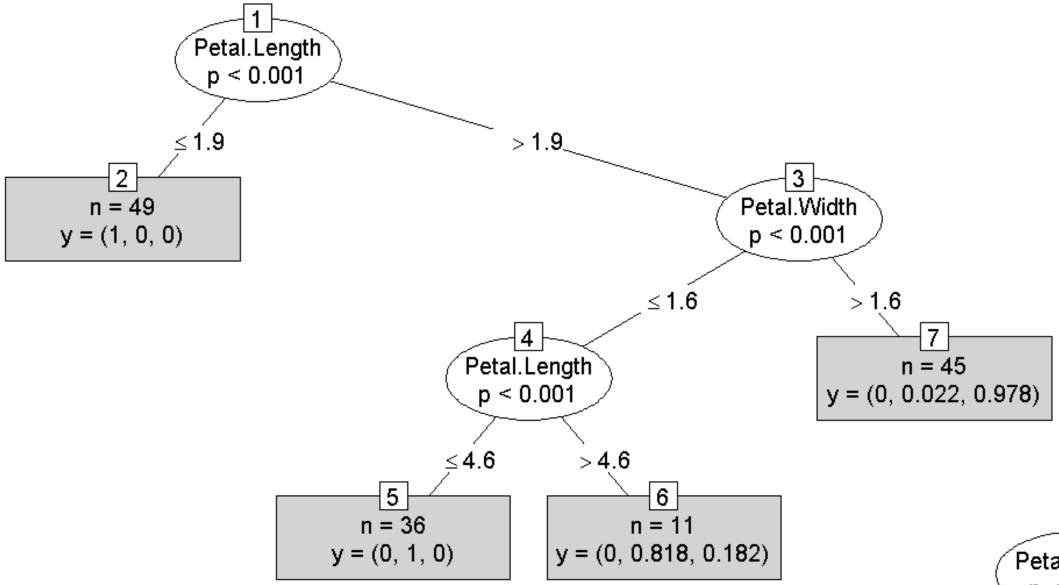
Hothorn, T., Hornik, K., Strobl, C., and Zeileis, A. (2015). Party: A laboratory for recursive partytioning. http://cran.r-project.org/web/packages/party/. R package
version 1.0-23.

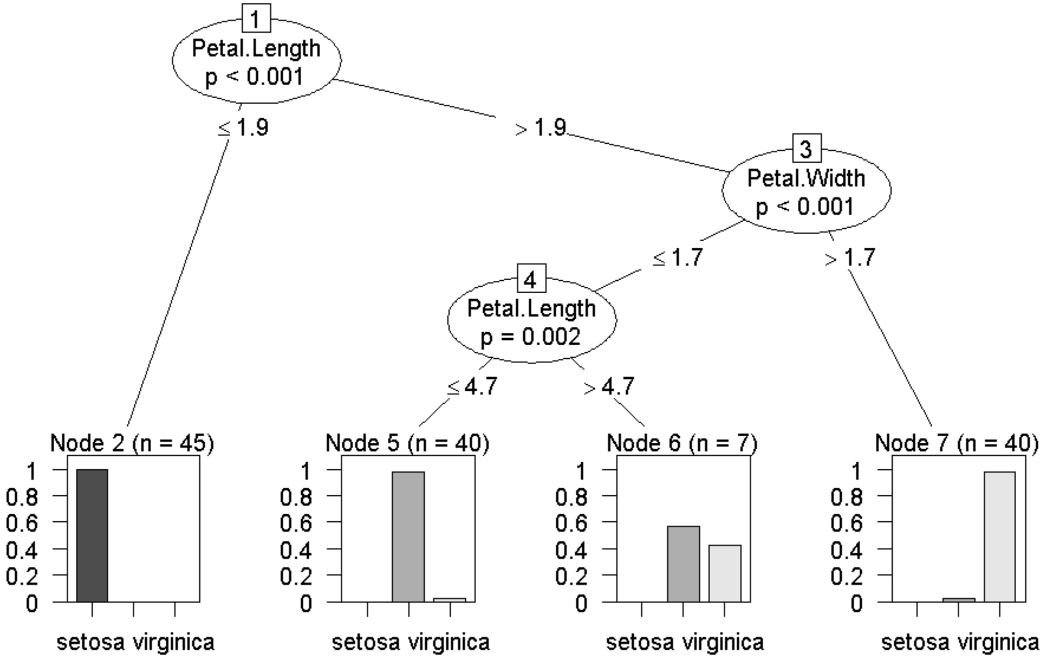
```
> library(party)
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> id <- sample(2, nrow(iris), replace = TRUE, prob = c(0.9, 0.1))
> train.data <- iris[id == 1, ]
> test.data <- iris[id == 2, ]
>
> # method 2
> #id <- sample(1:nrow(iris), nrow(iris)/10)
> #train.data <- iris[-id, ]
> #test.data <- iris[id, ]
>
> myModel <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
> iris.ctree <- ctree(myModel, data=train.data)
> table(predict(iris.ctree), train.data$Species)

             setosa versicolor virginica
  setosa         45          0         0
  versicolor      0         43         4
  virginica       0          1        39
```

R Package: party

```
> print(iris.ctree)
         Conditional inference tree with 4 terminal nodes

Response:  Species
Inputs:  Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
Number of observations:  132

1) Petal.Length <= 1.9; criterion = 1, statistic = 123.873
  2)*  weights = 45
1) Petal.Length > 1.9
  3) Petal.Width <= 1.7; criterion = 1, statistic = 58.5
    4) Petal.Length <= 4.7; criterion = 0.998, statistic = 12.429
      5)*  weights = 40
    4) Petal.Length > 4.7
      6)*  weights = 7
  3) Petal.Width > 1.7
    7)*  weights = 40
>
> plot(iris.ctree)
> plot(iris.ctree, type="simple")

> testPred <- predict(iris.ctree, newdata = test.data)
> table(testPred, test.data$Species)

testPred     setosa versicolor virginica
  setosa          5          0         0
  versicolor      0          6         1
  virginica       0          0         6
```

> plot(iris.ctree)





R Package: rpart

Therneau, T., Atkinson, B., and Ripley, B. (2015). rpart: Recursive Partitioning and Regression Trees.

- Function `rpart()` is used to build a decision tree, and the tree with the minimum prediction error is selected.
- After that, it is applied to new data to make prediction with function `predict()`.

Example: bodyfat {TH.data} Prediction of Body Fat by Skinfold Thickness, Circumferences, and Bone Breadths

```
> install.packages("TH.data")
> data("bodyfat", package = "TH.data")
> ? bodyfat
> dim(bodyfat)
[1] 71 10
> head(bodyfat)
   age DEXfat waistcirc hipcirc elbowbreadth kneebreadth anthro3a anthro3b anthro3c anthro4
47  57  41.68     100.0   112.0          7.1         9.4     4.42     4.95     4.50    6.13
48  65  43.29      99.5   116.5          6.5         8.9     4.63     5.01     4.48    6.37
49  59  35.41      96.0   108.5          6.2         8.9     4.12     4.74     4.60    5.82
50  58  22.79      72.0    96.5          6.1         9.2     4.03     4.48     3.91    5.66
51  60  36.42      89.5   100.5          7.1        10.0     4.24     4.68     4.15    5.91
52  61  24.13      83.5    97.0          6.5         8.8     3.55     4.06     3.64    5.14
```

rpart Example

```
> set.seed(1234)
> id <- sample(2, nrow(bodyfat), replace=TRUE, prob=c(0.7, 0.3))
> bodyfat.train <- bodyfat[id==1,]
> bodyfat.test <- bodyfat[id==2,]
> # train a decision tree
> library(rpart)
> my.model <- DEXfat ~ age + waistcirc + hipcirc + elbowbreadth + kneebreadth
> bodyfat.rpart <- rpart(my.model, data = bodyfat.train, control = rpart.control(minsplit =
10))
> bodyfat.rpart
n= 56

node), split, n, deviance, yval
      * denotes terminal node

 1) root 56 7265.0290000 30.94589
   2) waistcirc< 88.4 31  960.5381000 22.55645
     4) hipcirc< 96.25 14  222.2648000 18.41143
       8) age< 60.5 9   66.8809600 16.19222 *
       9) age>=60.5 5   31.2769200 22.40600 *
     5) hipcirc>=96.25 17  299.6470000 25.97000
      10) waistcirc< 77.75 6   30.7345500 22.32500 *
      11) waistcirc>=77.75 11  145.7148000 27.95818
        22) hipcirc< 99.5 3    0.2568667 23.74667 *
        23) hipcirc>=99.5 8   72.2933500 29.53750 *
   3) waistcirc>=88.4 25 1417.1140000 41.34880
     6) waistcirc< 104.75 18  330.5792000 38.09111
      12) hipcirc< 109.9 9   68.9996200 34.37556 *
      13) hipcirc>=109.9 9   13.0832000 41.80667 *
     7) waistcirc>=104.75 7  404.3004000 49.72571 *
```

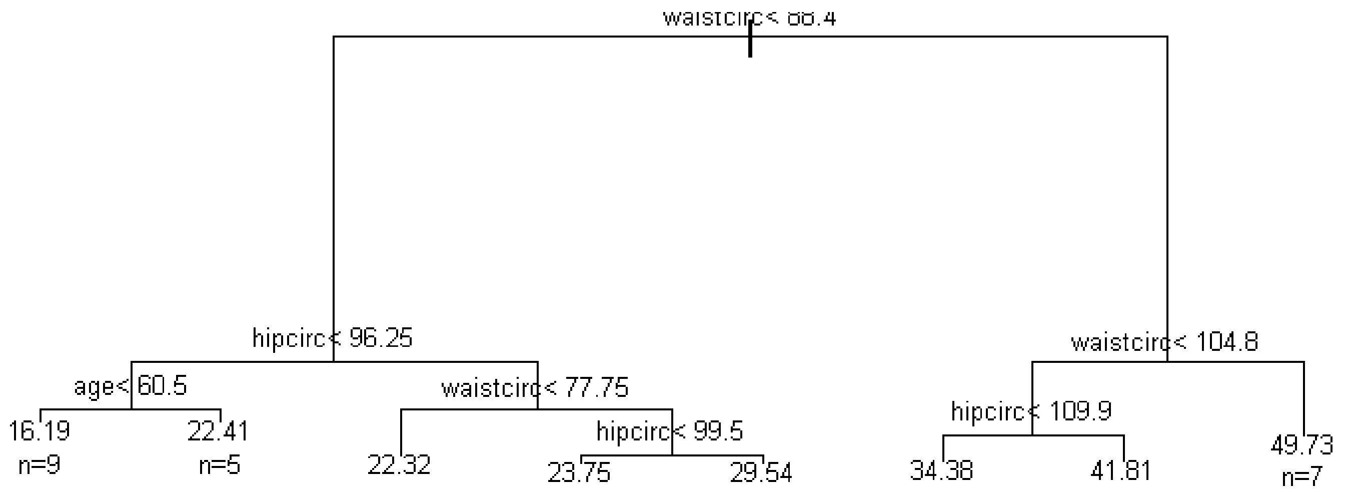
Chapter 4: Decision Tree and Random Forest
http://www.RDataMining.com

rpart Example

```
> attributes(bodyfat.rpart)
$names
 [1] "frame"   "where"    "call"         "terms"    "cptable"  "method"
 [7] "parms"   "control"  "functions"    "numresp"  "splits"   "variable.importance"
[13] "y"       "ordered"

$xlevels
named list()

$class
[1] "rpart"

> str(bodyfat.rpart)
List of 14
 $ frame              :'data.frame':    15 obs. of  8 variables:
  ..$ var       : Factor w/ 4 levels "<leaf>","age",..: 4 3 2 1 1 4 1 3 1 1 ...
  ..$ n         : int [1:15] 56 31 14 9 5 17 6 11 3 8 ...
  ..$ wt        : num [1:15] 56 31 14 9 5 17 6 11 3 8 ...
  ..$ dev       : num [1:15] 7265 960.5 222.3 66.9 31.3 ...
  ..$ yval      : num [1:15] 30.9 22.6 18.4 16.2 22.4 ...
  ..$ complexity: num [1:15] 0.6727 0.0604 0.0171 0.01 0.01 ...
  ..$ ncompete  : int [1:15] 4 4 4 0 0 4 0 4 0 0 ...
  ..$ nsurrogate: int [1:15] 4 4 2 0 0 4 0 2 0 0 ...
 $ where              : Named int [1:56] 14 14 13 7 9 10 13 9 5 9 ...
  ..- attr(*, "names")= chr [1:56] "47" "48" "49" "50" ...
...
 $ ordered            : Named logi [1:5] FALSE FALSE FALSE FALSE FALSE
  ..- attr(*, "names")= chr [1:5] "age" "waistcirc" "hipcirc" "elbowbreadth" ...
 - attr(*, "xlevels")= Named list()
 - attr(*, "class")= chr "rpart"
```

rpart Example

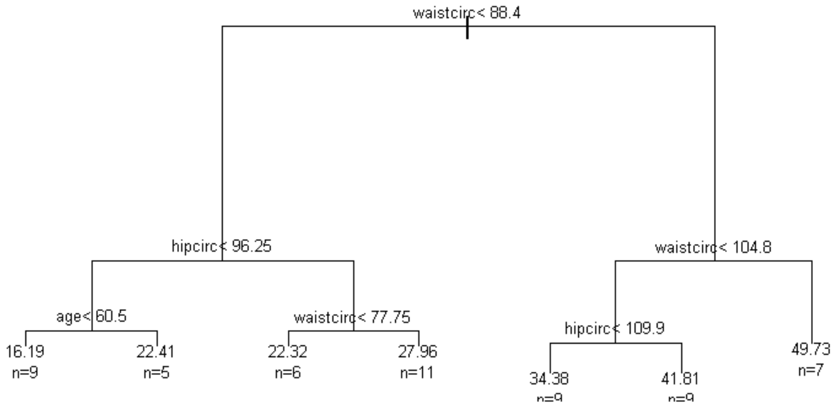
```
> # print the matrix of information on the optimal prunings based on a complexity
parameter.
> print(bodyfat.rpart$cptable)
          CP nsplit  rel error    xerror       xstd
1 0.67272638      0 1.00000000 1.0194546 0.18724382
2 0.09390665      1 0.32727362 0.4415438 0.10853044
3 0.06037503      2 0.23336696 0.4271241 0.09362895
4 0.03420446      3 0.17299193 0.3842206 0.09030539
5 0.01708278      4 0.13878747 0.3038187 0.07295556
6 0.01695763      5 0.12170469 0.2739808 0.06599642
7 0.01007079      6 0.10474706 0.2693702 0.06613618
8 0.01000000      7 0.09467627 0.2695358 0.06620732
>
> plot(bodyfat.rpart)
> text(bodyfat.rpart, use.n=T)
```



rpart Example

```
> #select the tree with the minimum prediction error
> opt <- which.min(bodyfat.rpart$cptable[,"xerror"])
> opt
7
7
> cp <- bodyfat.rpart$cptable[opt, "CP"]
> cp
[1] 0.01007079
> bodyfat.prune <- prune(bodyfat.rpart, cp = cp)
> bodyfat.prune
n= 56

node), split, n, deviance, yval
      * denotes terminal node

 1) root 56 7265.02900 30.94589
   2) waistcirc< 88.4 31  960.53810 22.55645
     4) hipcirc< 96.25 14  222.26480 18.41143
       8) age< 60.5 9   66.88096 16.19222 *
       9) age>=60.5 5   31.27692 22.40600 *
     5) hipcirc>=96.25 17  299.64700 25.97000
      10) waistcirc< 77.75 6   30.73455 22.32500 *
      11) waistcirc>=77.75 11  145.71480 27.95818 *
   3) waistcirc>=88.4 25 1417.11400 41.34880
     6) waistcirc< 104.75 18  330.57920 38.09111
      12) hipcirc< 109.9 9   68.99962 34.37556  *
      13) hipcirc>=109.9 9   13.08320 41.80667 *
     7) waistcirc>=104.75 7  404.30040 49.72571 *
```

```
> plot(bodyfat.prune)
> text(bodyfat.prune, use.n=T, cex=0.7)
```

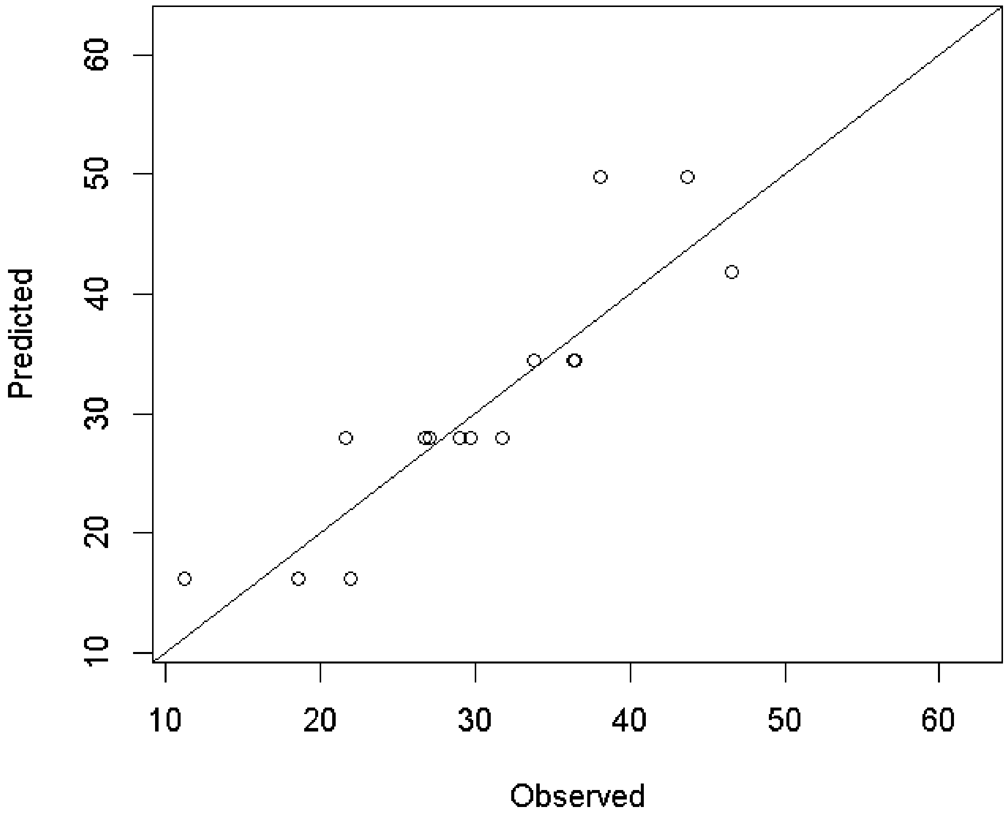


rpart Example

```
> DEXfat.pred <- predict(bodyfat.prune, newdata=bodyfat.test)
> xlim <- range(bodyfat$DEXfat)
> plot(DEXfat.pred ~ DEXfat, data=bodyfat.test, xlab="Observed", ylab="Predicted",
ylim=xlim, xlim=xlim)
> abline(a=0, b=1)
```

- After that, the selected tree is used to make prediction and the predicted values are compared with actual labels. In the code below, function `abline()` draws a diagonal line.

- The predictions of a good model are expected to be equal or very close to their actual values, that is, most points should be on or close to the diagonal line.



Chapter 4: Decision Tree and Random Forest, http://www.RDataMining.com

Apply Classification Tree to Microarray Data

- **Growing a tree (classification rule)**: variable selection, split criterion and tree Pruning

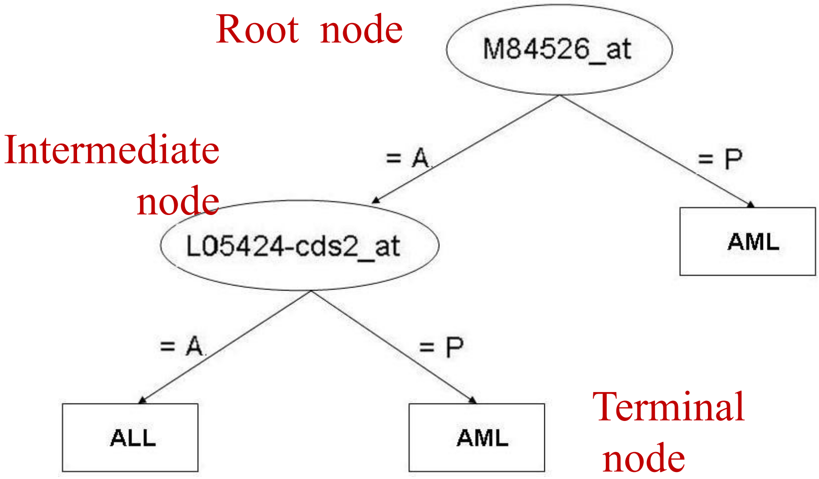
The Leukemia Dataset [Golub et al. 1999].

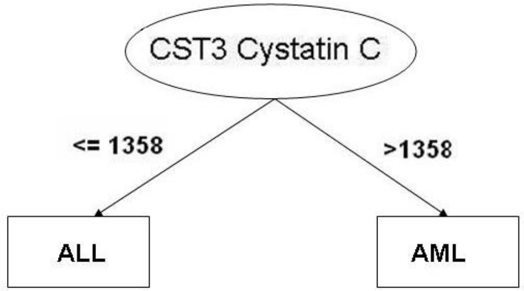
- There are 72 patients (47 ALL ((急性淋巴細胞白血病))and 25 AML(急性骨髓性白血病)).
- Each sample (patient) consists 7219 genes expression values.
- Training set: 38 samples (27 ALL and 11 AML)
- Test set: 34 samples (20 ALL and 14 AML).
- Nominal values:

  A: gene is absent or not expressed;

  P: gene is expressed or present

  M: the level of the expression is marginal among A and P



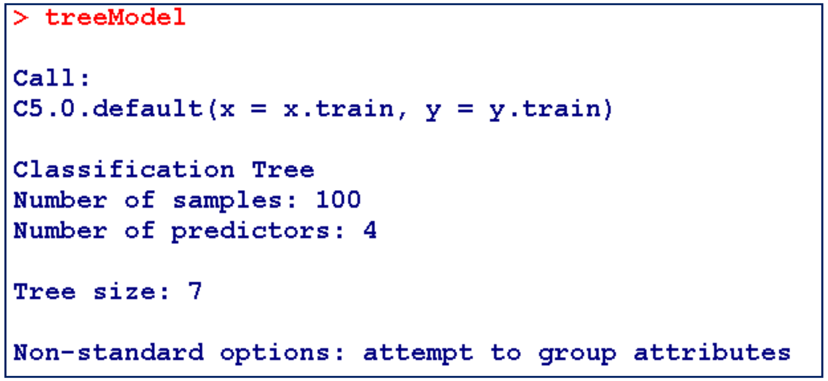


Gene ID	Description	References
M84526_at	Human adipsin/complement factor D	[Dobra 2008] [Schachtner et al. 2007]
L05424-cds2_at	Human cell surface glycoprotein CD44	[Screaton et al. 1992] [Krause et al. 2006]
CST3	Cystatin C (amyloid angiopathy and cerebral hemorrhage)	[Tang et al. 2009] [Sun et al. 2004]

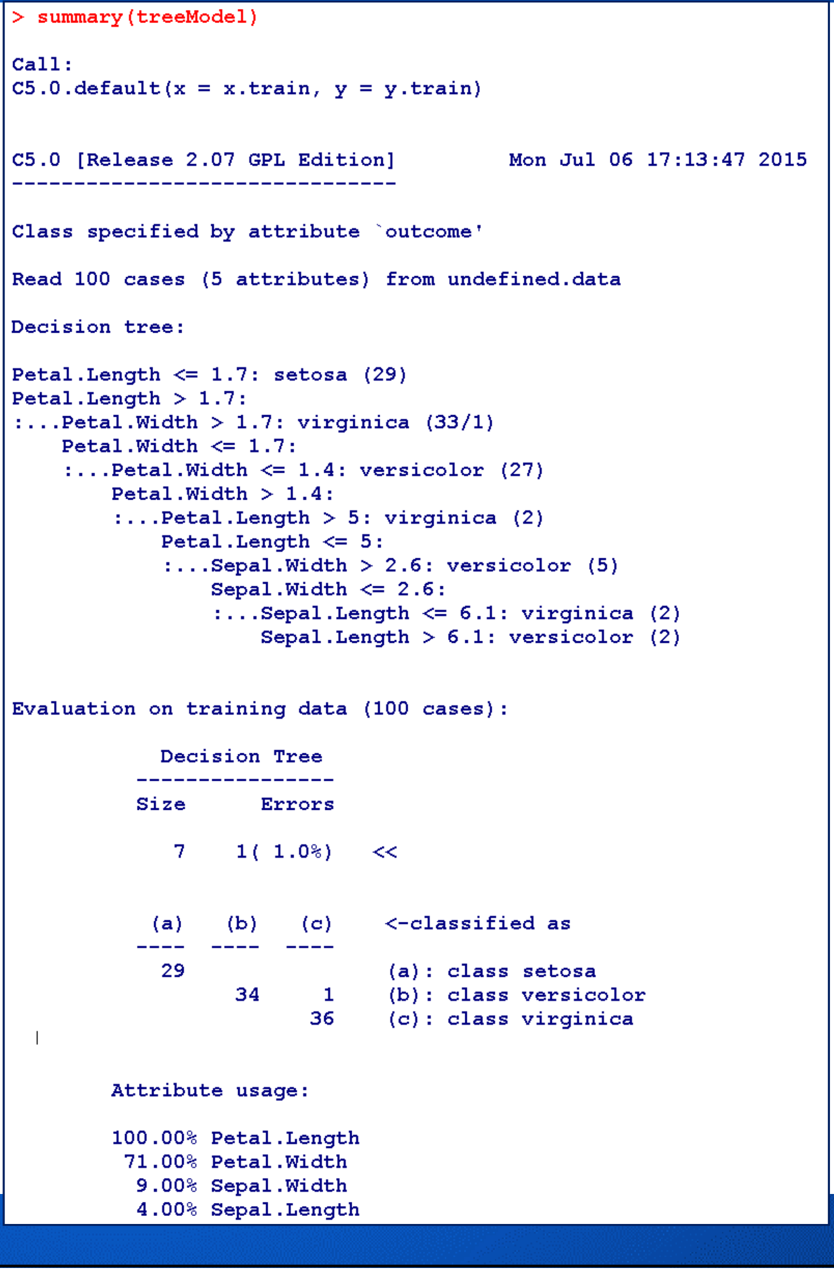
OP Netto et al., Applying Decision Trees to Gene Expression Data from DNA Microarrays: A Leukemia Case Study. Technical report.

Apply C5.0 to Iris Data

```
library(C50)
attach(iris)

# setup the training and testing data
id <- sample(1:nrow(iris),
2*floor(nrow(iris)/3))
x.train <- iris[id, 1:4]
y.train <- Species[id]
x.test <- iris[-id, 1:4]
y.test <- Species[-id]

# C5.0 Decision Tree
treeModel <- C5.0(x.train, y.train)
treeModel
summary(treeModel)
```



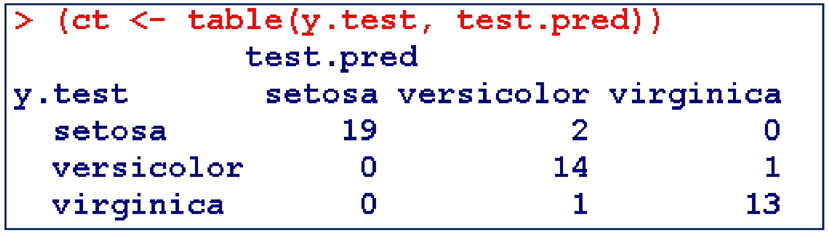
```
> treeModel

Call:
C5.0.default(x = x.train, y = y.train)

Classification Tree
Number of samples: 100
Number of predictors: 4

Tree size: 7

Non-standard options: attempt to group attributes
```



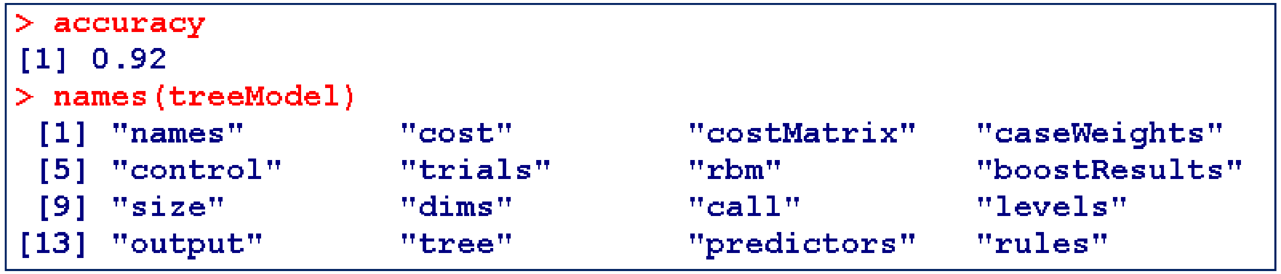
```
> summary(treeModel)

Call:
C5.0.default(x = x.train, y = y.train)


C5.0 [Release 2.07 GPL Edition]         Mon Jul 06 17:13:47 2015
-------------------------------

Class specified by attribute `outcome'

Read 100 cases (5 attributes) from undefined.data

Decision tree:

Petal.Length <= 1.7: setosa (29)
Petal.Length > 1.7:
:...Petal.Width > 1.7: virginica (33/1)
    Petal.Width <= 1.7:
    :...Petal.Width <= 1.4: versicolor (27)
        Petal.Width > 1.4:
        :...Petal.Length > 5: virginica (2)
            Petal.Length <= 5:
            :...Sepal.Width > 2.6: versicolor (5)
                Sepal.Width <= 2.6:
                :...Sepal.Length <= 6.1: virginica (2)
                    Sepal.Length > 6.1: versicolor (2)


Evaluation on training data (100 cases):

            Decision Tree
          ----------------
          Size      Errors

             7    1( 1.0%)   <<


           (a)   (b)   (c)    <-classified as
          ----  ----  ----
            29                (a): class setosa
                  34     1    (b): class versicolor
                        36    (c): class virginica


        Attribute usage:

        100.00% Petal.Length
         71.00% Petal.Width
          9.00% Sepal.Width
          4.00% Sepal.Length
```

Apply C5.0 to Iris Data

```
# Predition and Accuracy
test.pred <- predict(treeModel, x.test)
(ct <- table(y.test, test.pred))
accuracy <- sum(diag(ct))/sum(ct)
accuracy

names(treeModel)
```



```
> (ct <- table(y.test, test.pred))
            test.pred
y.test       setosa versicolor virginica
  setosa         19          2         0
  versicolor      0         14         1
  virginica       0          1        13
```



```
> accuracy
[1] 0.92
> names(treeModel)
 [1] "names"        "cost"         "costMatrix"   "caseWeights"
 [5] "control"      "trials"       "rbm"          "boostResults"
 [9] "size"         "dims"         "call"         "levels"
[13] "output"       "tree"         "predictors"   "rules"
```

課堂練習: 把分類的結果，呈現在資料的前兩維主成份(PCA)空間上。

See also: `caret: Classification and Regression Training`
Misc functions for training and plotting classification and regression models.

Random Forest

- Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
- Random decision forests correct for decision trees' habit of overfitting to their training set.
- **Two limitations:**
  - it cannot handle data with missing values, and users have to impute data before feeding them into the function.
  - Second, there is a limit of 32 to the maximum number of levels of each categorical attribute. Attributes with more than 32 levels have to be transformed first.
- `cforest{party}`: not limited to the maximum levels of each categorical attribute..
- Categorical variables with more levels will make it require more memory and take longer time to build a random forest.

https://en.wikipedia.org/wiki/Random_forest

randomForest {randomForest}

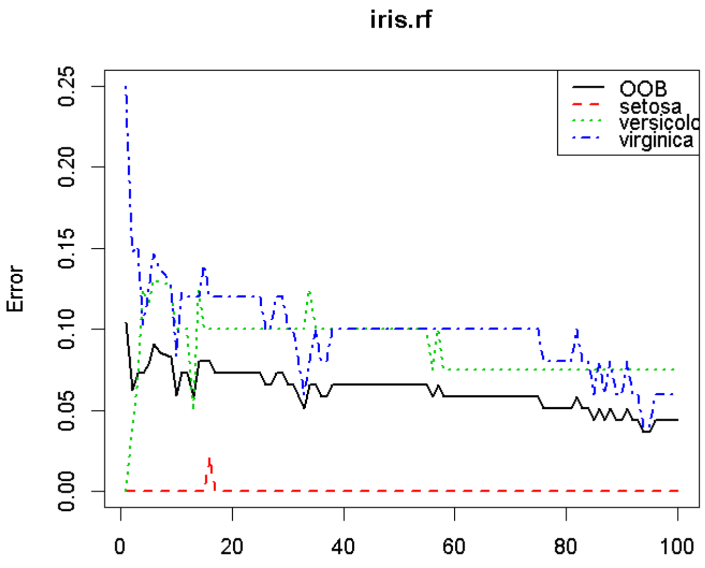
Classification and Regression with Random Forest

proximity: Should proximity measure among the rows be calculated?

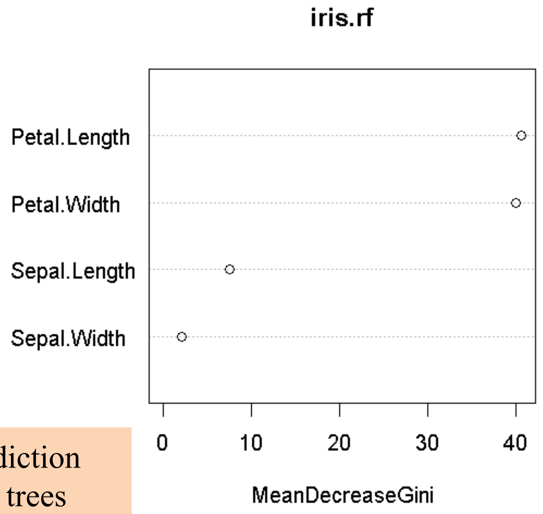
```
> library(randomForest)
> id <- sample(2, nrow(iris), replace=TRUE, prob=c(0.9, 0.1))
> iris.train <- iris[id==1, ]
> iris.test <- iris[id==2, ]
> iris.rf <- randomForest(Species ~ ., data=iris.train, ntree=100, proximity=TRUE)
> iris.rf

Call:
 randomForest(formula = Species ~ ., data = iris.train, ntree = 100, proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4.26%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         48          0         0  0.00000000
versicolor      0         44         3  0.06382979
virginica       0          3        43  0.06521739
> attributes(iris.rf)
$names
 [1] "call"            "type"            "predicted"       "err.rate"        "confusion"
 [6] "votes"           "oob.times"       "classes"         "importance"      "importanceSD"
[11] "localImportance" "proximity"       "ntree"           "mtry"            "forest"
[16] "y"               "test"            "inbag"           "terms"

$class
[1] "randomForest.formula" "randomForest"
```

Apply randomForest to iris data

```
> # plot the error rates with various number of trees.
> table(predict(iris.rf), iris.train$Species)

             setosa versicolor virginica
  setosa         48          0         0
  versicolor      0         44         3
  virginica       0          3        43
> head(iris.rf$err.rate)
            OOB     setosa versicolor  virginica
[1,] 0.06000000 0.00000000 0.06250000 0.12500000
[2,] 0.06172840 0.00000000 0.08333333 0.10000000
[3,] 0.06542056 0.00000000 0.12903226 0.07894737
[4,] 0.05555556 0.00000000 0.10000000 0.06976744
[5,] 0.05343511 0.00000000 0.09756098 0.06666667
[6,] 0.06716418 0.02173913 0.09302326 0.08888889
> plot(iris.rf, lwd=2)
> legend("topright", colnames(iris.rf$err.rate),
+   lty=1:4, lwd=2, col=1:4)
```



```
> #The importance of variables can be obtained
> # with functions importance() and varImpPlot().
> importance(iris.rf)
             MeanDecreaseGini
Sepal.Length         8.870603
Sepal.Width          2.267712
Petal.Length        37.233183
Petal.Width         44.814665
> varImpPlot(iris.rf)
```

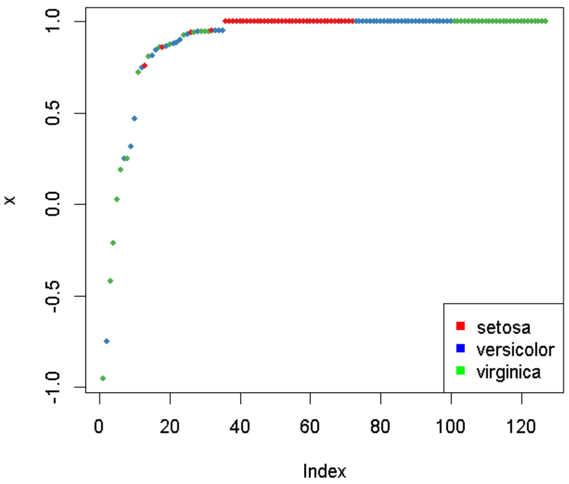


Out-of-bag (OOB) error: OOB is the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.

Apply randomForest to iris data

- Finally, the built random forest is tested on test data, and the result is checked with functions `table()` and `margin()`.
- The margin of a data point is as the proportion of votes for the correct class minus maximum proportion of votes for other classes.
- Generally speaking, positive margin means correct classification.

```
> iris.pred <- predict(iris.rf, newdata=iris.test)
> table(iris.pred, iris.test$Species)
iris.pred    setosa versicolor virginica
  setosa          2          0         0
  versicolor      0          3         0
  virginica       0          0         4
> sort(margin(iris.rf, iris.train$Species)) # 127
 virginica  versicolor   virginica   virginica   virginica   virg
-0.95121951 -0.75000000 -0.42105263 -0.21212121  0.02857143  0.18
...
  virginica   virginica   virginica   virginica   virginica   vir
 1.00000000  1.00000000  1.00000000  1.00000000  1.00000000  1.00
  virginica
 1.00000000
> plot(margin(iris.rf, iris.train$Species))
> legend("bottomright", levels(iris.train$Species), col=c("red","blue","green"), pch=15)
```



margin {randomForest}: Compute or plot the margin of predictions from a randomForest classifier.

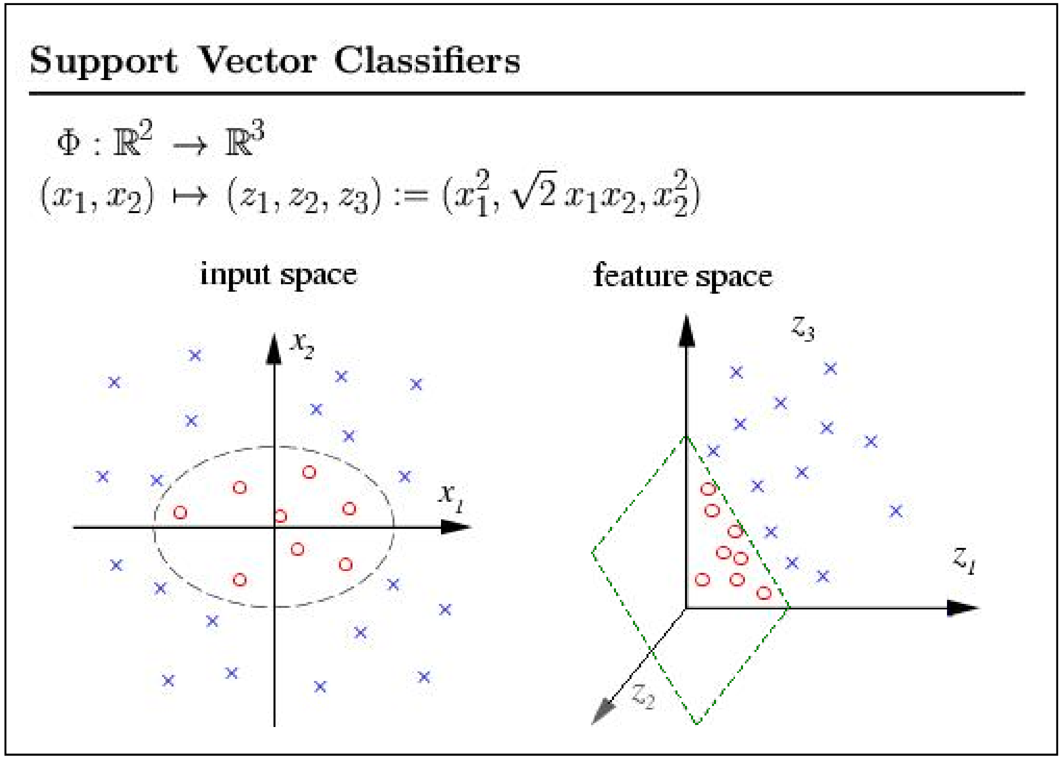
`margin(x, observed, ...)`

`x`: an object of class randomForest

`observed`: the true response corresponding to the data in `x`.

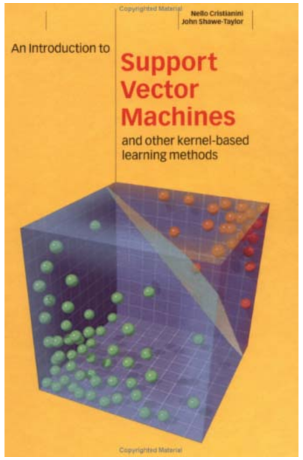
Support Vector Machines (SVMs)

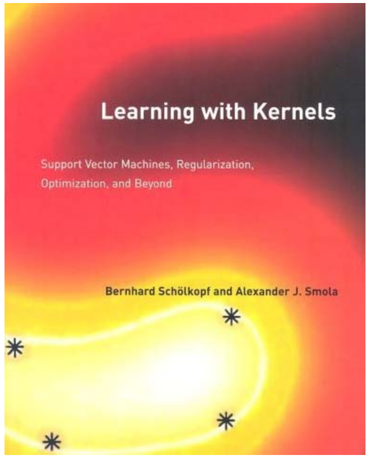
SVMs (Vapnik, 1995) map the data (input space) into high dimensional space (feature space) through a kernel function ϕ and then find a hyperplane $\mathbf{w}$ to separate two groups (binary classification).

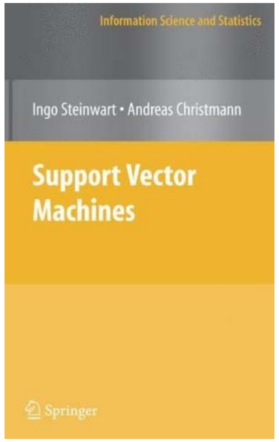


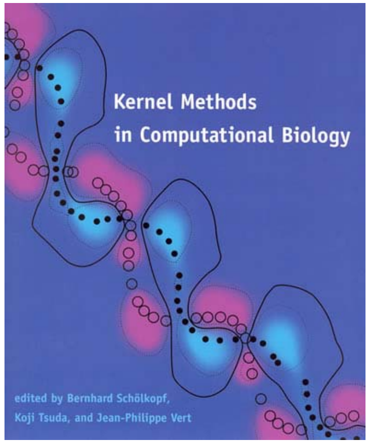
Support Vector Classifiers

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

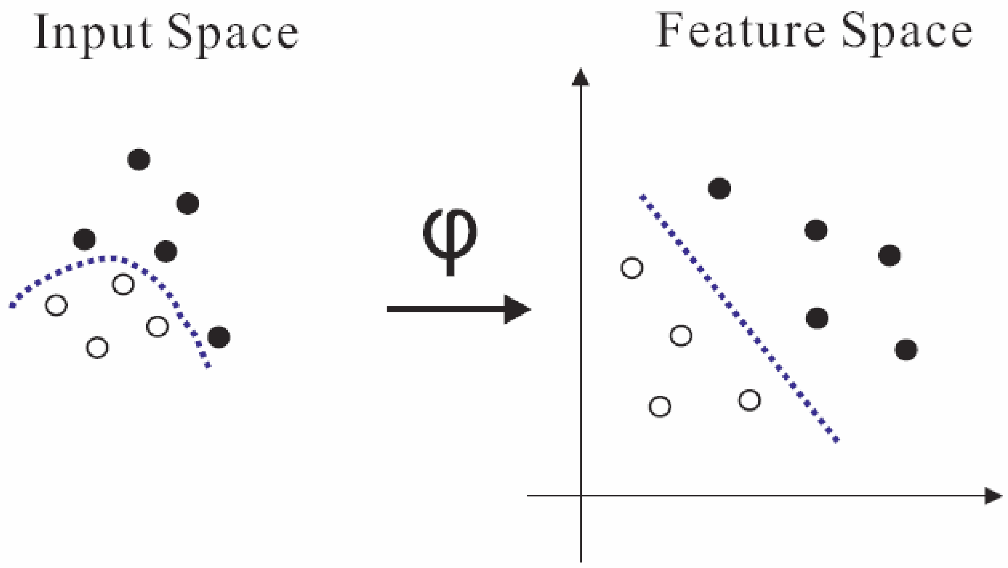



Learning with Kernels
Support Vector Machines, Regularization, Optimization, and Beyond
Bernhard Schölkopf and Alexander J. Smola


Information Science and Statistics
Ingo Steinwart • Andreas Christmann
Support Vector Machines
Springer

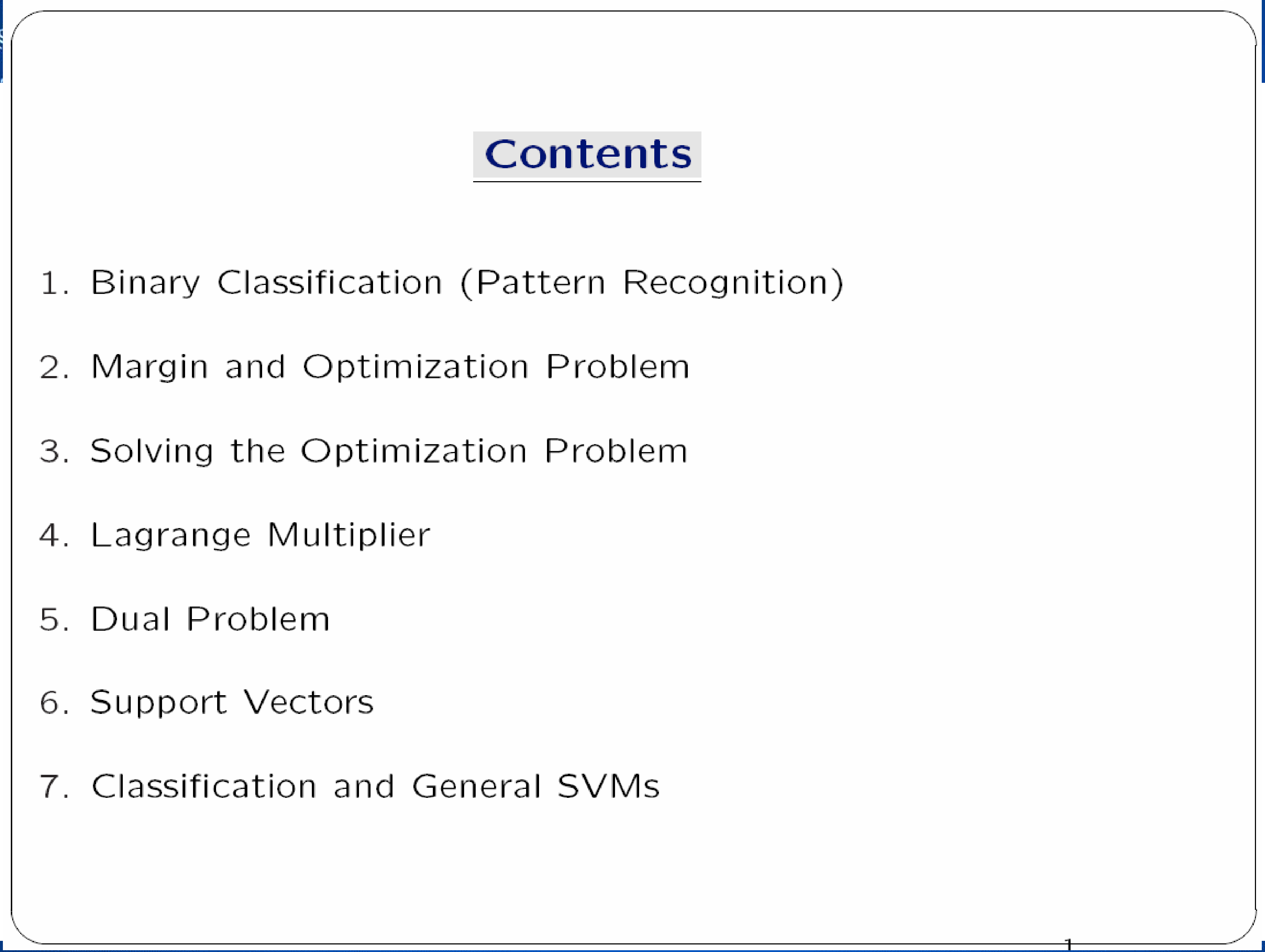

Kernel Methods in Computational Biology
edited by Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert

popular in data mining and machine learning



1. SVMs perform a linear discrimination of a training set of labeled points in the feature space associated with a kernel.

2. The resulting separation can be nonlinear in the orginal space.



Contents

1. Binary Classification (Pattern Recognition)

2. Margin and Optimization Problem

3. Solving the Optimization Problem

4. Lagrange Multiplier

5. Dual Problem

6. Support Vectors

7. Classification and General SVMs

1. Binary Classification (Pattern Recognition)

1. What we have at hand:

   (a) data objects: $x_1, x_2, \cdots, x_n \in \mathcal{X}$.

   (b) class labels: $y_1, y_2, \cdots, y_n \in \mathcal{Y}$.

2. **SVMs** are kernel methods to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, which can be used to predict the label of any new object x by $f(x)$.

3. Binary classification or pattern recognition: each object is classified into one of two classes, indicated by the label $y \in \{-1, +1\}$.

1. Binary Classification (Pattern Recognition) (2)

1. Examples of pattern recognition problems in computational biology:

   (a) predicting whether a tissue is healthy from a gene profiling experiment.

   (b) predicting whether a chemical compound can bind a given target or not from its structure.

2. Prediction (or classificaion)

   (a) A positive prediction is associated with the label $+1$.

   (b) A negative prediction is associated with the label -1.

1. Binary Classification (Pattern Recognition) (3)

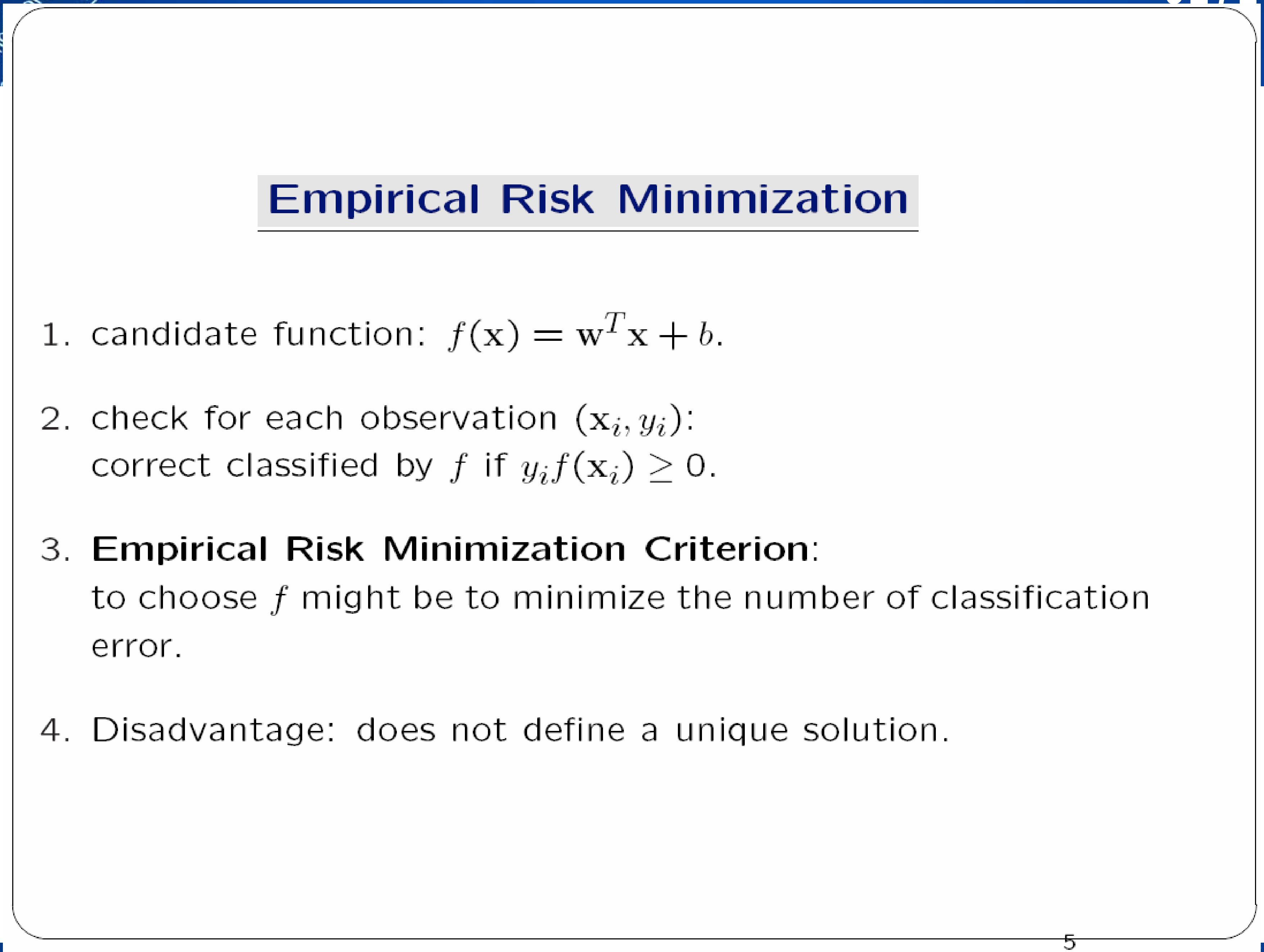
1. SVMs tries to separate the two classes of points using a linear function of the form

$$f(\mathrm{x}) = \mathrm{w}^T\mathrm{x} + b, \quad \mathrm{w} \in R^p, \ b \in R$$

   (a) label(x)$=$ $+1$ when $f(\mathrm{x}) \geq 0$.

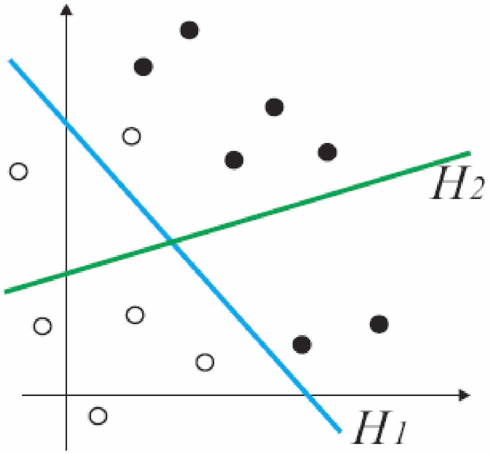
   (b) label(x)$=$ -1 when $f(\mathrm{x}) < 0$.

2. The problem is therefore to learn sunch a function f from a data set of observations.



Empirical Risk Minimization

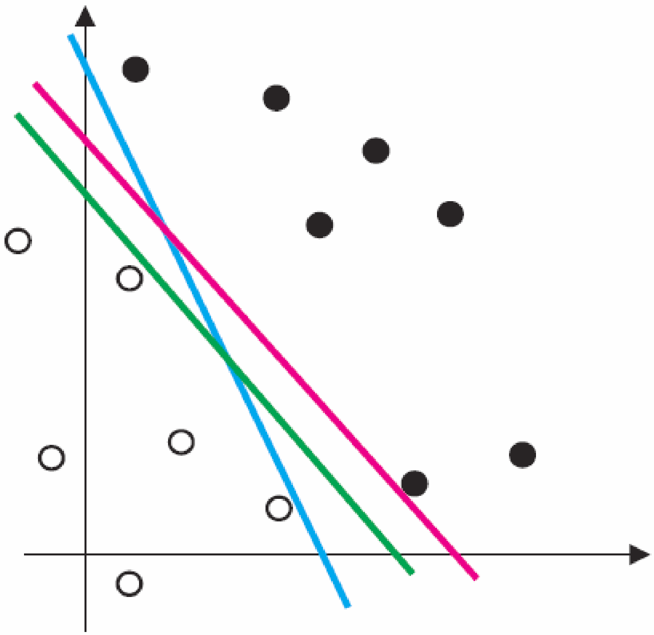
1. candidate function: $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$.
2. check for each observation $(\mathbf{x}_i, y_i)$:
   correct classified by f if $y_i f(\mathbf{x}_i) \geq 0$.
3. **Empirical Risk Minimization Criterion**:
   to choose f might be to minimize the number of classification error.
4. Disadvantage: does not define a unique solution.



1. The hyperplane H_1 discriminates the white circles from the black ones with 1 mistake.

2. The hyperplane H_2 separates these points with 5 mistake.

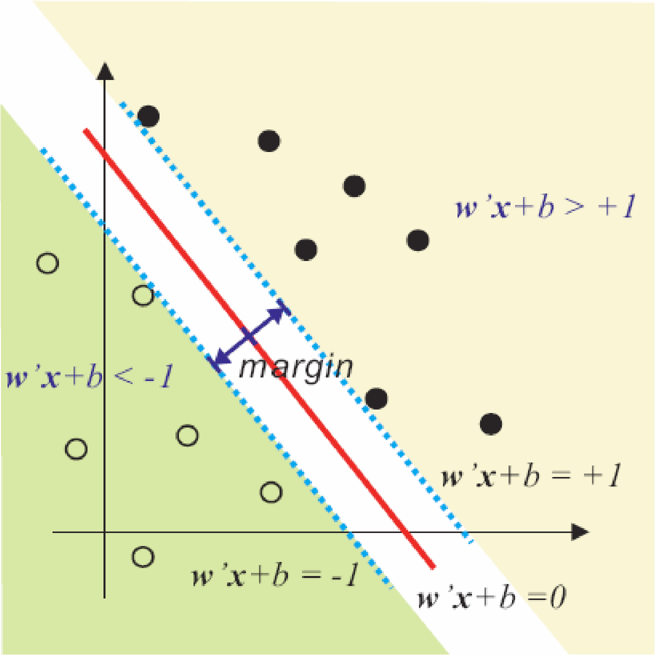
3. The empirical risk minimization principle: one should choose a hrperplane with a minimal number of errors on the training set, whch is H_1 in this case.



Even when the training data are linearly separable, the empirical risk minimization principle does not define a unique solution.

2.1 Margin

1. SVMs focus more on the confidence of the classifications than on the number of misclassifications.
2. $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ defines two half-spaces of points classified positively and negatively with large confidence.
   (a) $h^+ = \{\mathbf{x} : f(\mathbf{x}) \geq 1\}$.
   (b) $h^- = \{\mathbf{x} : f(\mathbf{x}) \leq -1\}$.
3. The distance between these two half-spaces, called the margin. (margin=$\frac{2}{||\mathbf{w}||}$).
4. One would like to find f with largest possible margin $\Rightarrow$ maximizing $\frac{2}{||\mathbf{w}||}$ under the constraints $y_i(\mathbf{w}^T\mathbf{x} + b) \geq 1$ for $i = 1, \cdots, n$.



1. A function $f(\mathrm{x}) = \mathrm{w}^T\mathrm{x} + b$ defines two half-spaces where points are classified with large confidence.

2. The distance between the half-spaces is equal to $1/\|\mathrm{w}\|$.

3. $\mathrm{w}^T\mathrm{x}_1 + b_0 = +1$, $\mathrm{w}^T\mathrm{x}_2 + b_0 = -1$,
$\mathrm{w}^T(\mathrm{x}_1 - \mathrm{x}_2) = 2$, $\dfrac{\mathrm{w}^T}{\|\mathrm{w}\|}(\mathrm{x}_1 - \mathrm{x}_2) = \dfrac{2}{\|\mathrm{w}\|}$,

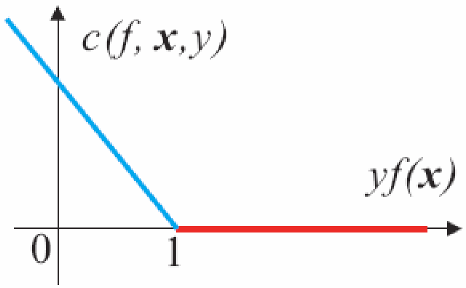
Hinge Loss

1. To deal with the cases when the training set cannot be correctly separated by a linear hyperplane, SVMs soften the constraints using the continuous hinge loss function

$$c(f, \mathrm{x}, y) = \max(0, 1 - yf(\mathrm{x})).$$

   (a) If (x, y) is correctly classified by f with large confidence, the $c(f, \mathrm{x}, y) = 0$.

   (b) Otherwise, $c(f, \mathrm{x}, y)$ increases with the distance from x to the correct half-space of large confidence.

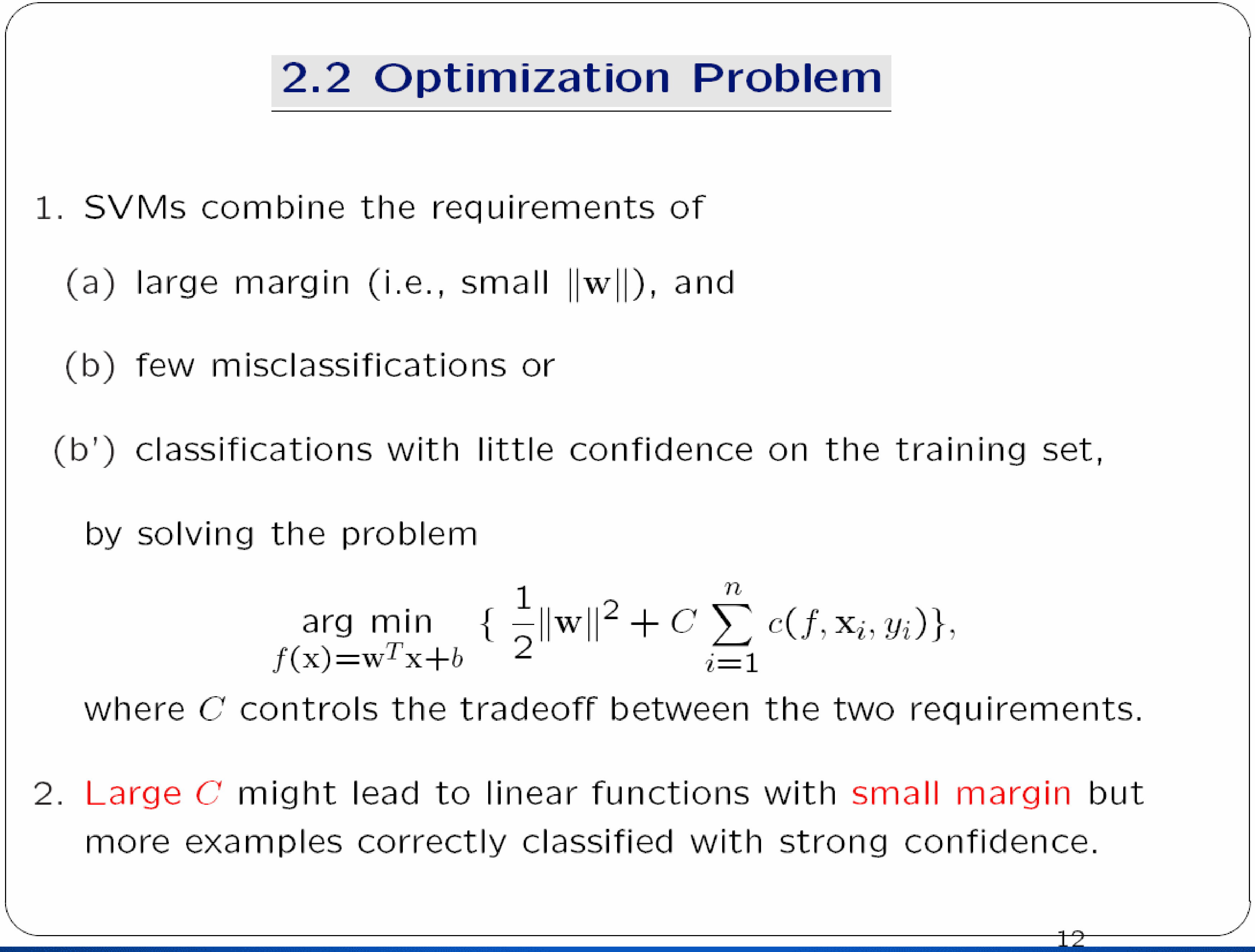


1. As long as $yf(\mathrm{x}) \geq 1$, the point x is correctly classified by f with large confidence and the hinge loss is null.

2. When $yf(\mathrm{x}) < 1$, x is either correctly classified with small confidence $(0 \leq yf(\mathrm{x}) < 1)$, or misclassified $(yf(\mathrm{x} < 0))$.

3. In these cases the hinge loss is positive, and increases as $1 - yf(\mathrm{x})$.

4. SVM find a linear separating function with a large margin and small average hinge loss on the training set.



2.2 Optimization Problem

1. SVMs combine the requirements of

   (a) large margin (i.e., small $\|\mathbf{w}\|$), and

   (b) few misclassifications or

   (b') classifications with little confidence on the training set,

   by solving the problem

$$\underset{f(\mathbf{x})=\mathbf{w}^T\mathbf{x}+b}{\arg\min} \; \{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} c(f, \mathbf{x}_i, y_i)\},$$

   where C controls the tradeoff between the two requirements.

2. Large C might lead to linear functions with small margin but more examples correctly classified with strong confidence.

3. Solving the Optimization Problem

1. The hinge loss function is not differentiable, direct minimization is not straightforward.

2. Let $\xi_1, \cdots, \xi_n$ be the slack variables.

3. Rewrite optimization problem as

$$\underset{\mathbf{w}, b, \xi_1, \cdots, \xi_n}{\arg\min} \ \{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \},$$

   under the constraints $\xi_i \geq c(f, \mathbf{x}_i, y_i)$ for $i = 1, \cdots, n$.

4. Minimization w.r.t ξ_i is obtaned when ξ_i takes its minimal value $c(f, \mathbf{x}_i, y_i)$.

5. $\xi_i \geq c(f, \mathbf{x}_i, y_i) \Leftrightarrow$ (a) $\xi_i \geq 0$ and (b) $\xi_i \geq 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)$.

Quadratic Programming Problem

1. SVMs solve

$$\underset{\mathbf{w},b,\xi_1,\cdots,\xi_n}{\text{arg min}} \ \{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i\},$$

under the constraints:

(a) $\xi_i \geq 0$ and

(b) $\xi_i \geq 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)$.

for $i = 1, \cdots, n$.

4. Lagrange Multipliers

1. The constrained optimization problem:

$$L(\mathbf{w}, b, \xi, \alpha, \beta) =$$

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i - \{\sum_{i=1}^{n} \alpha_i[\xi_i - (1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))]\} - \{\sum_{i=1}^{n} \beta_i\xi_i\}$$

2. Find the unique saddle point of L, wihc is a minimum with respect to $(\mathbf{w}, b, \xi)$ and a maximum with repect to $(\alpha, \beta) \geq 0$.

$$\underset{\mathbf{w},b,\xi,\alpha,\beta}{\text{arg min}}\ L$$

Setting Partial Derivatives to Zero

1. For fixed α, β,

   (a) $$\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \beta)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i = 0.$$

   (b) $$\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \beta)}{\partial b} = \sum_{i=1}^{n} y_i \alpha_i = 0.$$

   (c) $$\frac{\partial L(\mathbf{w}, b, \xi, \alpha, \beta)}{\partial \xi_i} = C - \alpha_i - \beta_i = 0.$$

   for $i = 1, \cdots, n$.

5. Dual Problem

1. We get

$$\mathbf{w} = \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i.$$

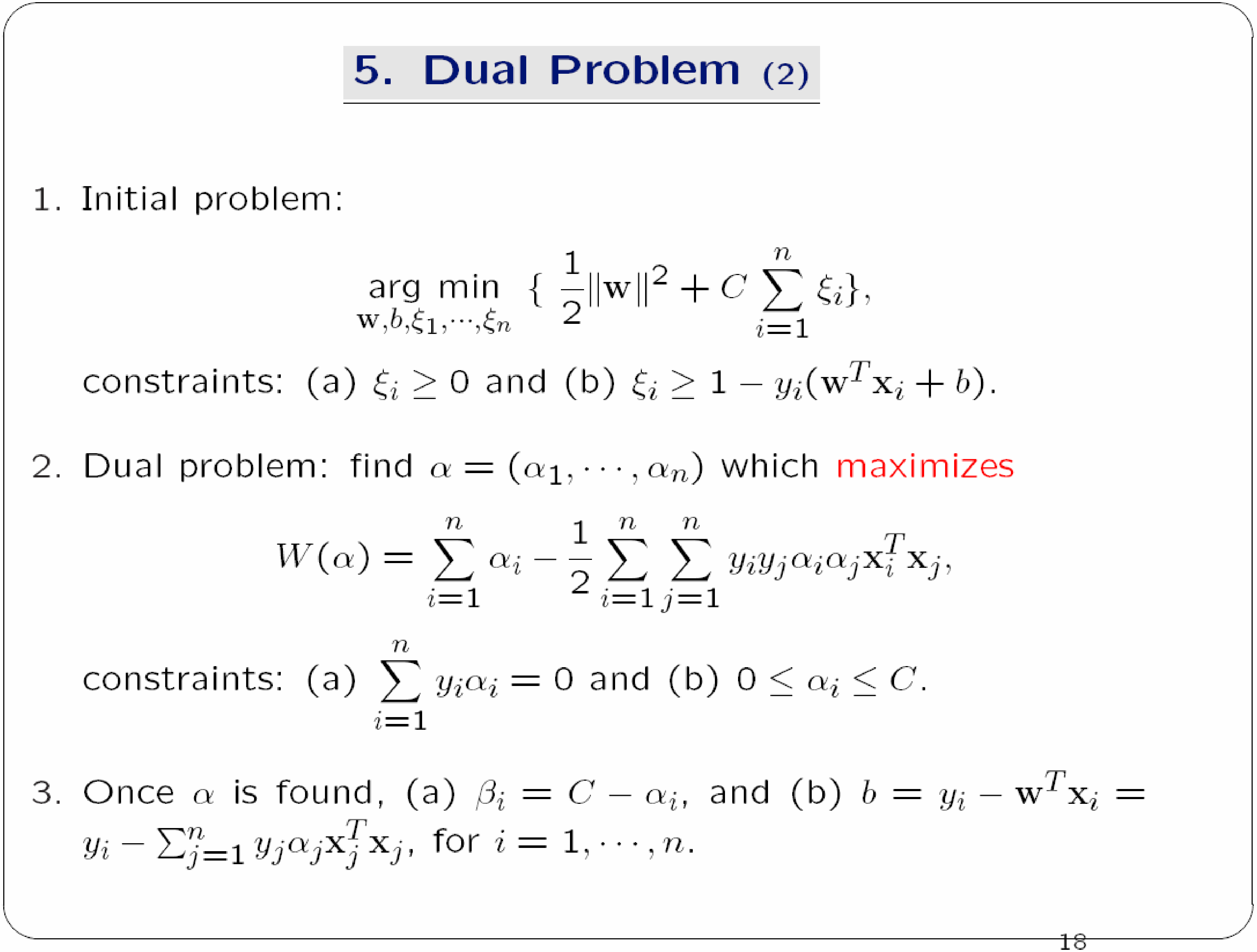
   i.e., $\mathbf{w}$ is a linear combination of the $\mathbf{x}_1, \cdots, \mathbf{x}_n$.

2. The value of the Lagrangian when minimized wrt $(\mathbf{w}, b, \mathbf{x})$ becomes, $\forall(\alpha, \beta) \geq 0$

$$\inf_{\mathbf{w}, b, \xi} L(\mathbf{w}, b, \xi, \alpha, \beta) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j,$$

3. Need to maximize above as a function of α and to check there exists some $\beta > 0$ for which $C - \alpha_i - \beta_i = 0$.

   Note: The infimum (inf) is the greatest lower bound of a set. The supremum (sup) is the least upper bound of the set.



5. Dual Problem (2)

1. Initial problem:

$$\underset{\mathbf{w},b,\xi_1,\cdots,\xi_n}{\text{arg min}} \; \{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i\},$$

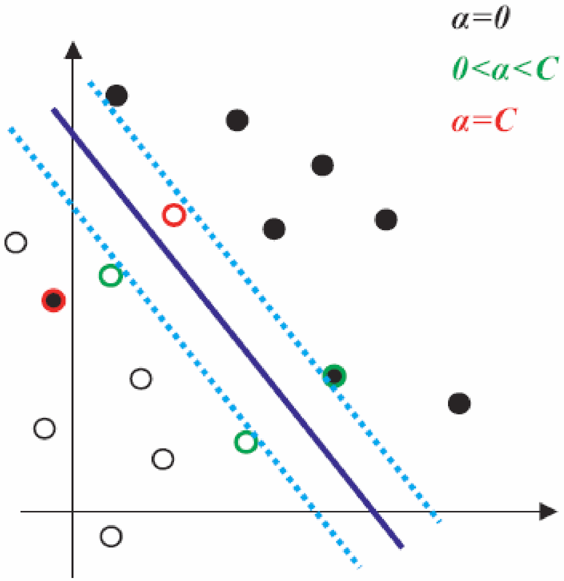
constraints: (a) $\xi_i \geq 0$ and (b) $\xi_i \geq 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)$.

2. Dual problem: find $\alpha = (\alpha_1, \cdots, \alpha_n)$ which maximizes

$$W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j,$$

constraints: (a) $\sum_{i=1}^{n} y_i\alpha_i = 0$ and (b) $0 \leq \alpha_i \leq C$.

3. Once α is found, (a) $\beta_i = C - \alpha_i$, and (b) $b = y_i - \mathbf{w}^T\mathbf{x}_i = y_i - \sum_{j=1}^{n} y_j\alpha_j\mathbf{x}_j^T\mathbf{x}_j$, for $i = 1, \cdots, n$.

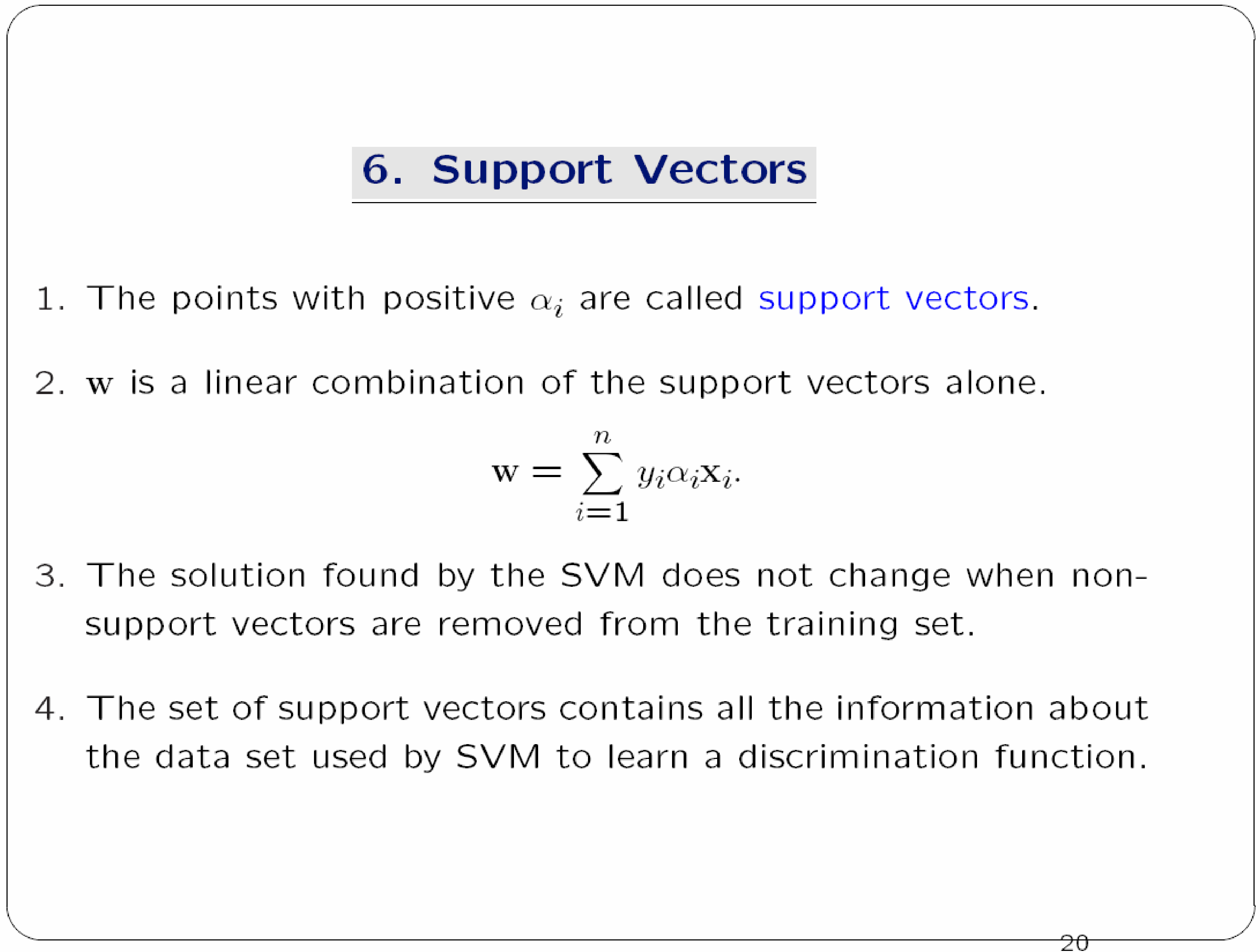


1. Each point correctly classified with large confidence ($yf(\mathbf{x}) > 1$) has $\alpha_i = 0$.

2. Other points are called support vectors.

   (a) they can be on the boundary ($0 \leq \alpha \leq C$), or

   (b) on the wrong side of this boundary ($\alpha_i = C$).



6. Support Vectors

1. The points with positive α_i are called support vectors.

2. $\mathbf{w}$ is a linear combination of the support vectors alone.

$$\mathbf{w} = \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i .$$

3. The solution found by the SVM does not change when non-support vectors are removed from the training set.

4. The set of support vectors contains all the information about the data set used by SVM to learn a discrimination function.

7. Classification and General SVMs

1. For a new $\mathbf{x} \in \mathcal{X}$, linear SVM classifier

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{n} y_i\alpha_i\mathbf{x}_i^T\mathbf{x} + b$$

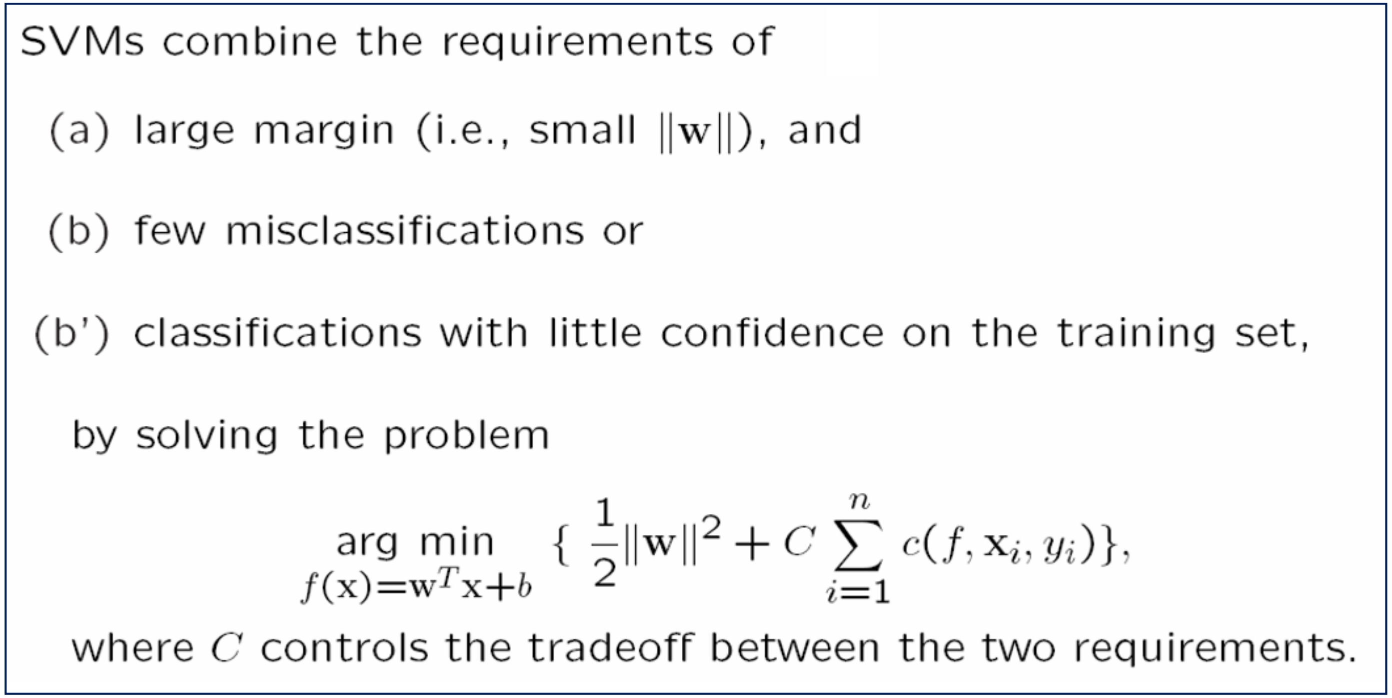
predict the class of $\mathbf{x}$ is -1 or $+1$ depdeding on the sign of the function.

2. Using kernel Trick to perform SVM in the feature space associated with a general kernel:

$$f(\mathbf{x}) = \sum_{i=1}^{n} y_i\alpha_i k(\mathbf{x}_i, \mathbf{x}) + b$$

The function is linear in the feature space associated with the kernel, can of course be nonlinear if the initial space is a vector space and the kernel is nonlinear.

SVM: Optimization Problem



SVMs combine the requirements of

(a) large margin (i.e., small $\|\mathbf{w}\|$), and

(b) few misclassifications or

(b') classifications with little confidence on the training set,

by solving the problem

$$\underset{f(\mathrm{x})=\mathrm{w}^T\mathrm{x}+b}{\text{arg min}} \; \{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} c(f, \mathbf{x}_i, y_i)\},$$

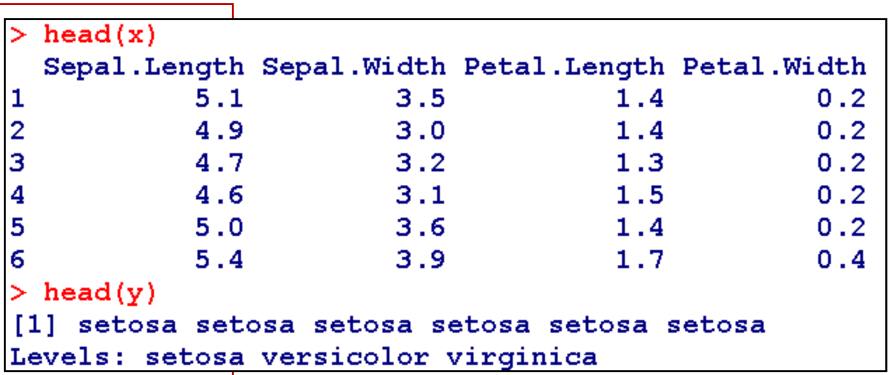
where C controls the tradeoff between the two requirements.

Two approaches for multi-class classification:

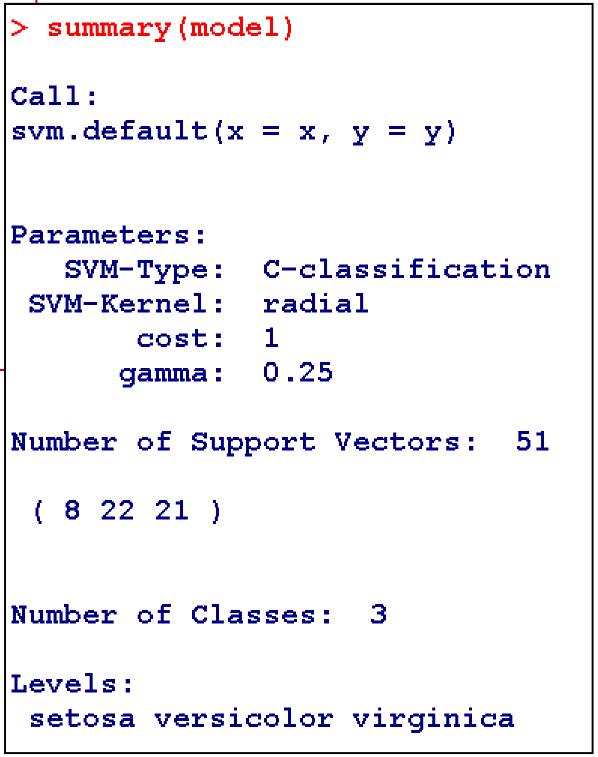
- **one-against-others**: The kth SVM model is constructed with all of the samples in the kth class with one group, and all other samples with the other group.
- **one-against-one**: The SVM trained model is constructed by using any two of classes. Therefore, there are total $K(K-1)/2$ classifiers.

Apply SVM to Iris Data

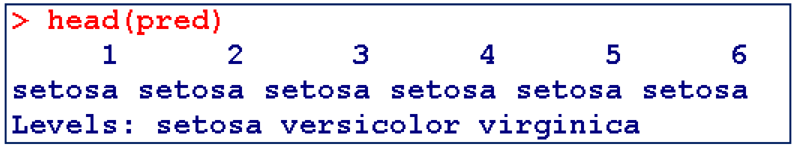
```
> # prepare data
> library(e1071)
> attach(iris)
> x <- subset(iris, select = -Species)
> y <- Species

> # use default setting (?svm)
> model <- svm(x, y)
> print(model)
> summary(model)

> # test with train data and report accuracy
> pred <- predict(model, x)
> table(pred, y)

> sum(diag(table(pred, y)))/length(y)
[1] 0.9733333
```



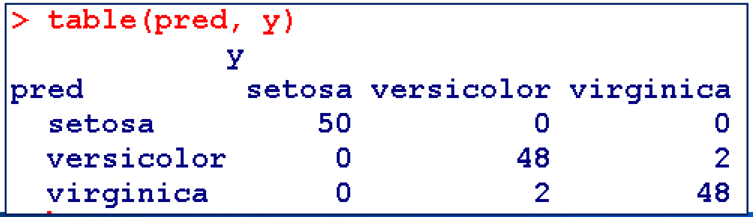
```
> head(x)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
> head(y)
[1] setosa setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica
```



```
> summary(model)

Call:
svm.default(x = x, y = y)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.25

Number of Support Vectors:  51

 ( 8 22 21 )


Number of Classes:  3

Levels:
 setosa versicolor virginica
```



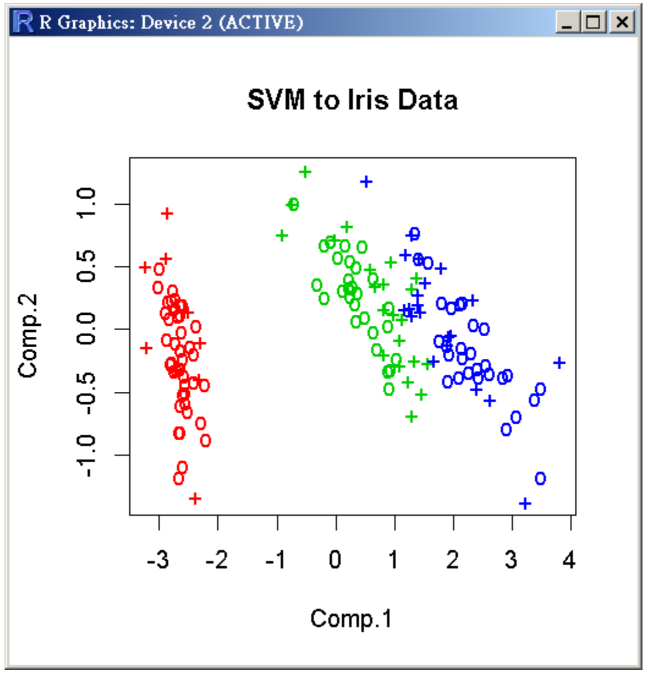
```
> head(pred)
     1      2      3      4      5      6
setosa setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica
```

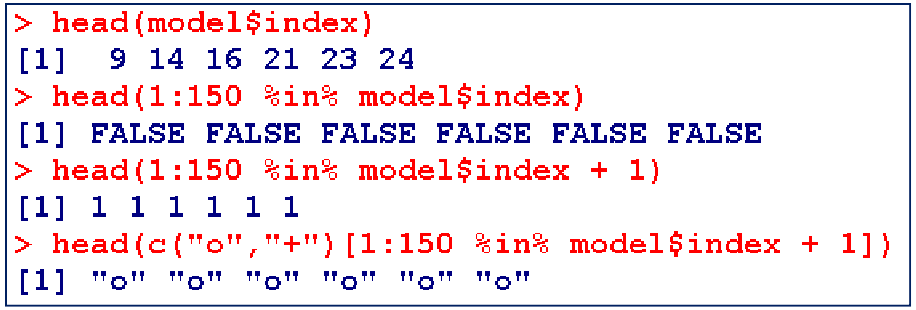


```
> table(pred, y)
            y
pred         setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          2        48
```

Apply SVM to Iris Data

```
# visualize (classes by color, SV by crosses):
pca <- princomp(iris[,-5], scores=TRUE)
plot(pca$scores[,1:2],
     col = as.integer(iris[,5])+1,
     pch = c("o","+")[1:150 %in% model$index + 1],
     main="SVM to Iris Data")
```





```
> head(model$index)
[1]  9 14 16 21 23 24
> head(1:150 %in% model$index)
[1] FALSE FALSE FALSE FALSE FALSE FALSE
> head(1:150 %in% model$index + 1)
[1] 1 1 1 1 1 1
> head(c("o","+")[1:150 %in% model$index + 1])
[1] "o" "o" "o" "o" "o" "o"
```

課堂練習:

(1) Randomly divide iris data into the training set (2/3) and testing set (1/3) and then apply SVM.

(2) Use the 10-fold CV technique.

Apply SVM to Microarray Data

Brown et al. (2000). Knowledge-based Analysis of Microarray Gene Expression Data Using Support Vector Machines, PNAS 97(1), 262-267.

Assume: Genes of similar function yield similar expression pattern.

Data

Yeast Gene Expression [2467x 80] out of [6,221x 80] has accurate functional annotations.

- Tricarboxylic acid
- Respiration
- Ribosome
- Proteasome
- Histone
- Helix-turn-helix

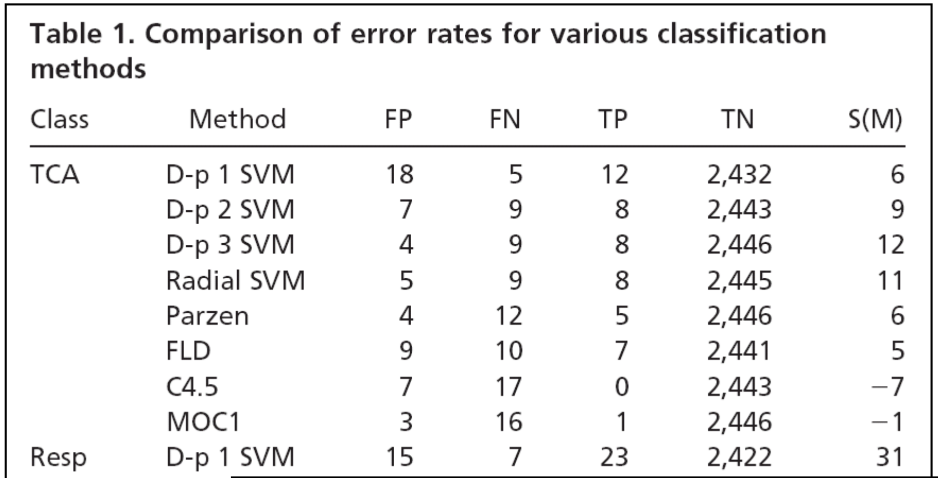


Table 1. Comparison of error rates for various classification methods

Class	Method	FP	FN	TP	TN	S(M)
TCA	D-p 1 SVM	18	5	12	2,432	6
	D-p 2 SVM	7	9	8	2,443	9
	D-p 3 SVM	4	9	8	2,446	12
	Radial SVM	5	9	8	2,445	11
	Parzen	4	12	5	2,446	6
	FLD	9	10	7	2,441	5
	C4.5	7	17	0	2,443	−7
	MOC1	3	16	1	2,446	−1
Resp	D-p 1 SVM	15	7	23	2,422	31
	D-p 2					
	D-p 3					

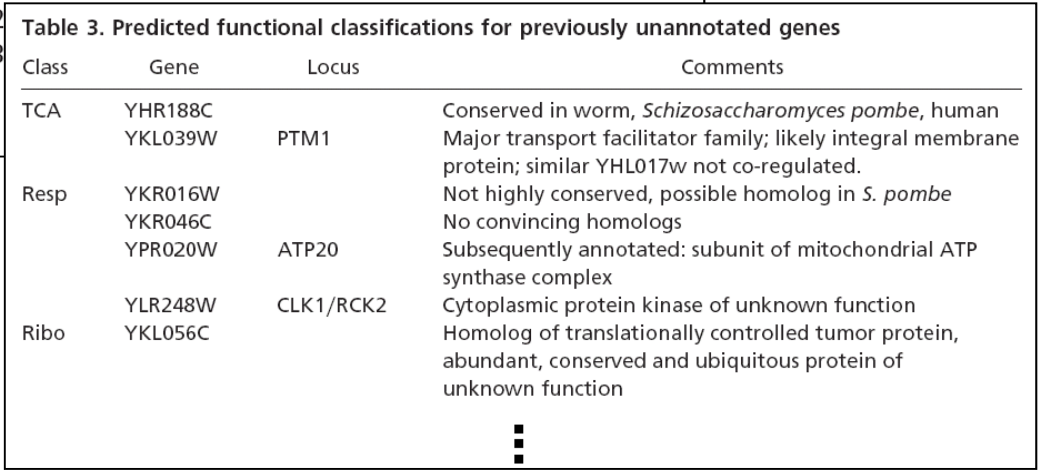
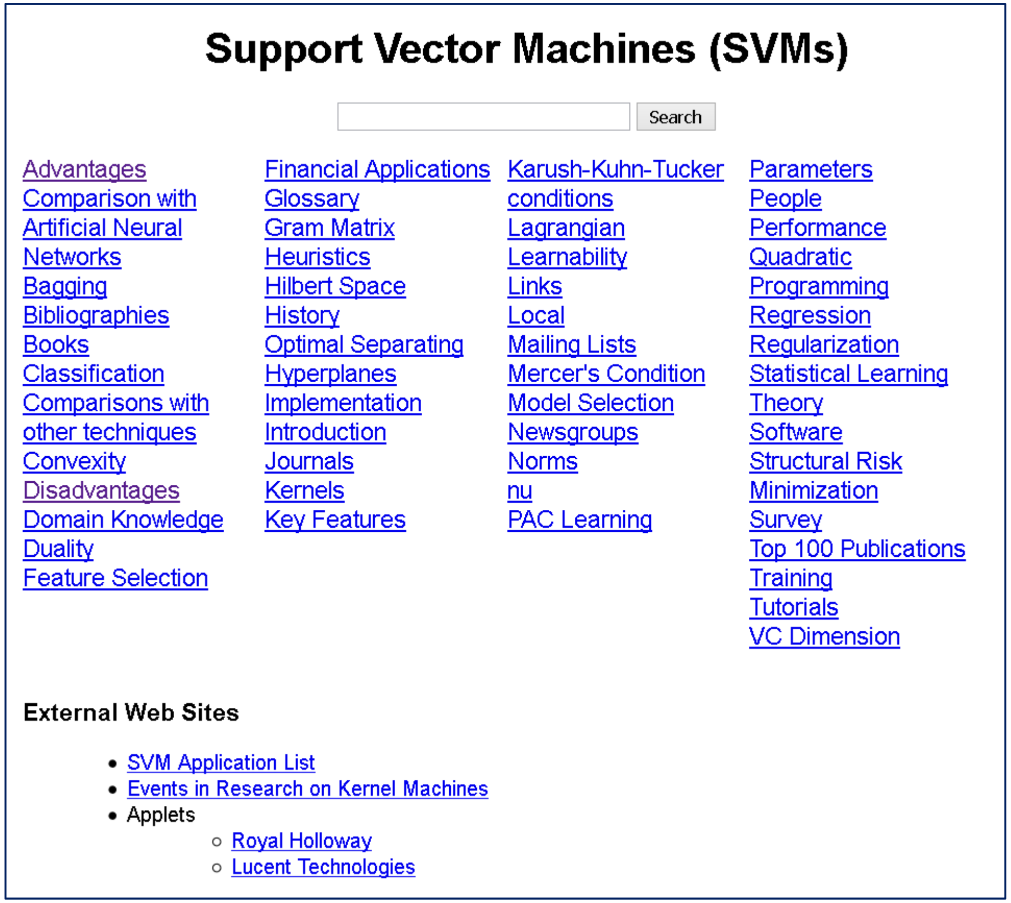


Table 3. Predicted functional classifications for previously unannotated genes

Class	Gene	Locus	Comments
TCA	YHR188C		Conserved in worm, *Schizosaccharomyces pombe*, human
	YKL039W	PTM1	Major transport facilitator family; likely integral membrane protein; similar YHL017w not co-regulated.
Resp	YKR016W		Not highly conserved, possible homolog in *S. pombe*
	YKR046C		No convincing homologs
	YPR020W	ATP20	Subsequently annotated: subunit of mitochondrial ATP synthase complex
	YLR248W	CLK1/RCK2	Cytoplasmic protein kinase of unknown function
Ribo	YKL056C		Homolog of translationally controlled tumor protein, abundant, conserved and ubiquitous protein of unknown function

⋮

http://www.svms.org



Support Vector Machines (SVMs)

Search

- Advantages
- Comparison with Artificial Neural Networks
- Bagging
- Bibliographies
- Books
- Classification
- Comparisons with other techniques
- Convexity
- Disadvantages
- Domain Knowledge
- Duality
- Feature Selection
- Financial Applications
- Glossary
- Gram Matrix
- Heuristics
- Hilbert Space
- History
- Optimal Separating Hyperplanes
- Implementation
- Introduction
- Journals
- Kernels
- Key Features
- Karush-Kuhn-Tucker conditions
- Lagrangian
- Learnability
- Links
- Local
- Mailing Lists
- Mercer's Condition
- Model Selection
- Newsgroups
- Norms
- nu
- PAC Learning
- Parameters
- People
- Performance
- Quadratic Programming
- Regression
- Regularization
- Statistical Learning Theory
- Software
- Structural Risk Minimization
- Survey
- Top 100 Publications
- Training
- Tutorials
- VC Dimension

External Web Sites

- SVM Application List
- Events in Research on Kernel Machines
- Applets
  - Royal Holloway
  - Lucent Technologies

Advantages

- "The key features of SVMs are the use of kernels, the absence of local minima, the sparseness of the solution and the capacity control obtained by optimising the margin." Shawe-Taylor and Cristianini (2004)
- A **regularization parameter** makes the user think about avoiding over-fitting.
- SVM uses the **kernel trick**, so one can build in expert knowledge about the problem via engineering the kernel.
- SVM is defined by a **convex optimization** problem (no local minima) for which there are efficient methods (e.g. SMO).
- There is a substantial body of **theory** behind SVM.

The Disadvantages of SVM

- The theory only really covers the determination of the parameters for a given value of the regularization and kernel parameters and choice of kernel.
- The kernel models can be quite sensitive to over-fitting the model selection criterion.
- How to choose of the kernel with its parameters.
- Computational cost: slow speed and size, both in training and testing.
  - SVMs is the high algorithmic complexity and extensive memory requirements of the required quadratic programming in large-scale tasks.
- Discrete data presents another problem.
- The optimal design for multiclass SVM classifiers is a further area for research.

http://www.svms.org/disadvantages.html

Practical Guide to deal with Imbalanced Classification Problems in R
https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/

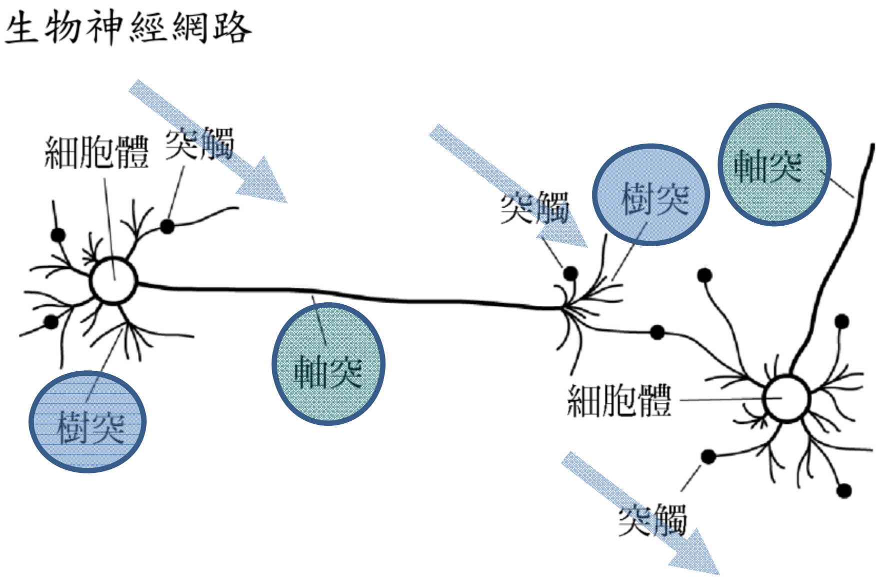
人工神經網路 (Artificial Neural Network，ANN)

- 人工神經網路(Artificial Neural Network，ANN)，簡稱神經網路(Neural Network，NN)或類神經網路，在機器學習和認知科學領域，是一種**模仿生物神經網路**（動物的中樞神經系統，特別是大腦）的結構和功能的**數學模型**或**計算模型**，用於**對函式進行估計或近似**。(Wiki)

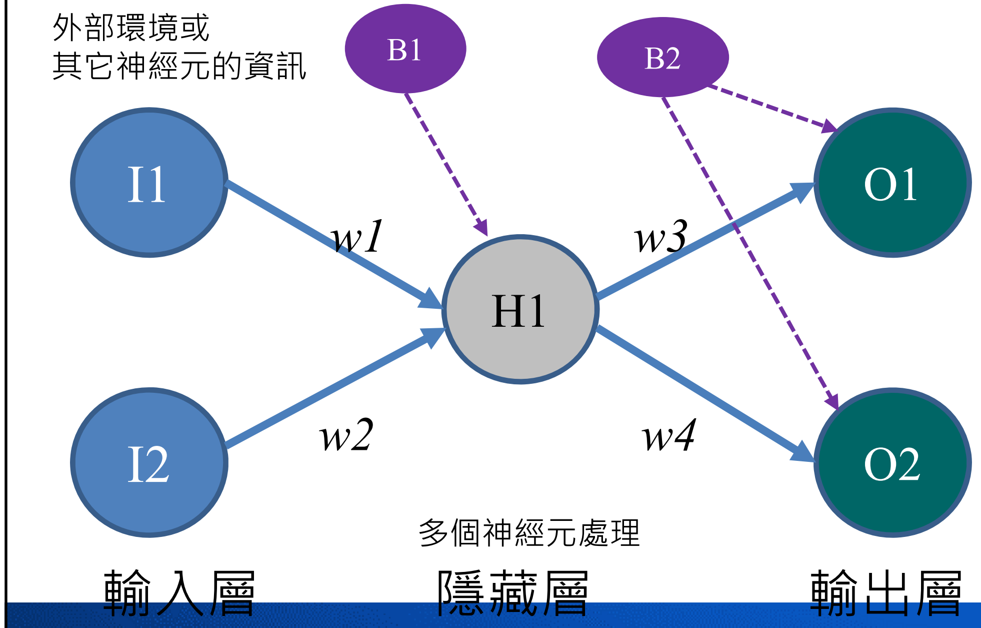
圖書館學與資訊科學大辭典

- ANN起源於1943年McCulloch和Pitts所提出之人工神經元運作模型，經過約20年的(初期)發展後，在1960年代中期因為遇到難以突破的技術瓶頸而沉寂了約20年，直到1986年由McClelland和Rumelhart提出Back-Propagation Algorithm突破技術瓶頸後，才又開始蓬勃發展至今。
- **ANN優點與特色**：
  - （一）自我學習能力：能自行從一堆輸入資料中學習到其中隱藏的規則或模式；
  - （二）容錯能力：其所學習到的規則或模式是散佈儲存於整個網路中，因此小部分的類神經網路損壞，並不會對系統的能力造成太大影響；
  - （三）平行處理能力：適合用來處理許多複雜的非線性問題；
  - （四）自我調整能力：能不斷根據新輸入的資料來調整已學習到的規則或模式。
- **ANN主要缺點**是學習速度緩慢，往往須要運用大量的輸入(或訓練)資料來讓類神經網路學習，而模糊邏輯(fuzzy logic)則有學習速度快但學習能力較為受限的特性，因此近年來的研究經常將人工類神經網路和模糊邏輯合併運用來互補長短。

模仿生物神經網路



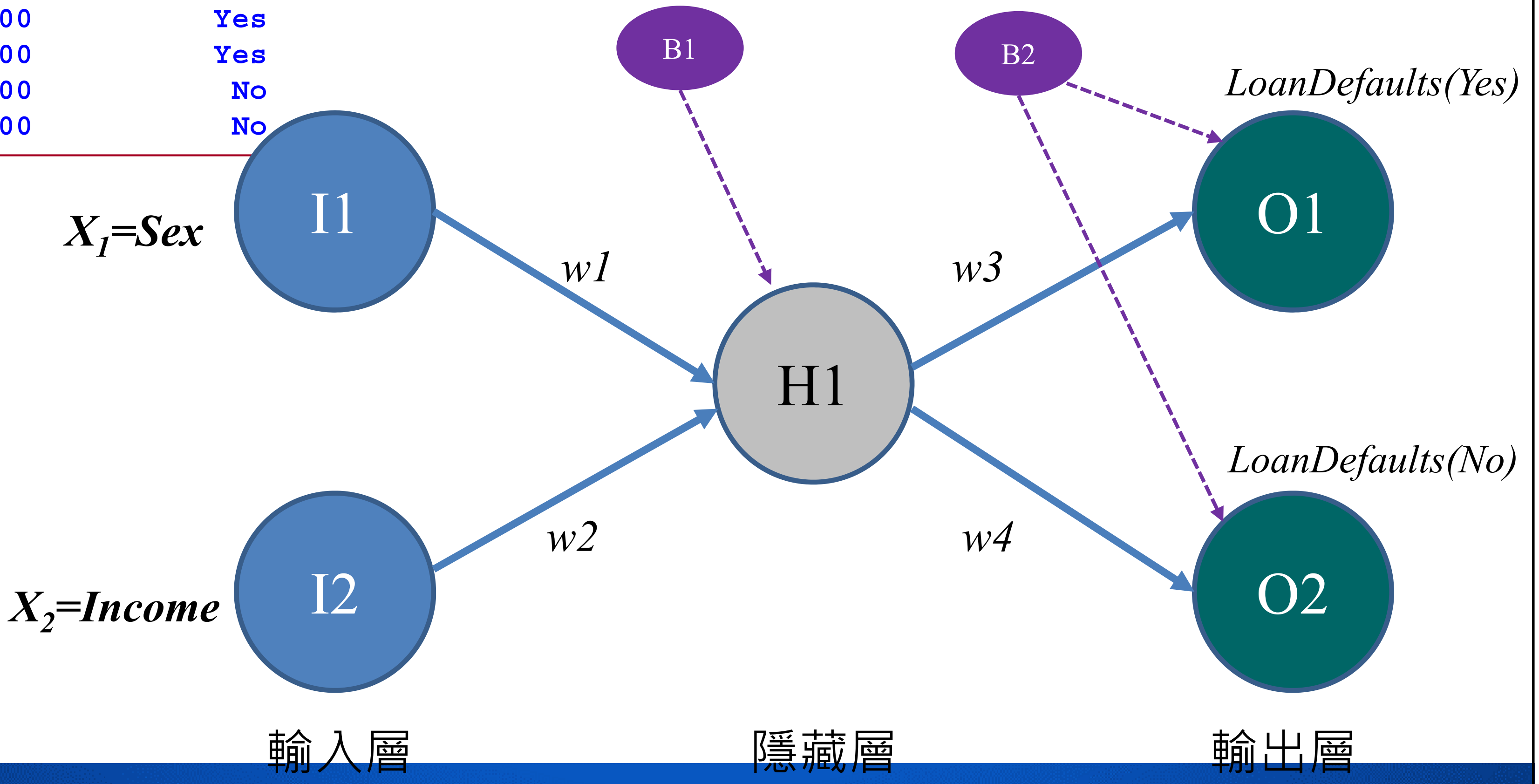
生物神經網路	人工神經網路
細胞體	神經元
樹突	輸入
軸突	輸出
突觸	權重



- W = 模仿生物神經細胞的突觸，稱為鏈結值(weights)，可視為一種加權效果，其值越大，對類神經網路的影響也更大。
- B = 模仿生物神經細胞的細胞核偏權值(bias)，具有偏移的效果。
- f(θ) = 模仿生物神經細胞的細胞核的非線性**轉換函數**，目的是將加權成績和的值做映射得到所需要的輸出。
- O = 輸出傳到外界環境或是其他人工神經元的資料。

Artificial Neural Network (ANN) (人工神經網路)

```
> Loan <- read.table("Loan_training.txt", header=T, sep="\t")
> Loan
   Sex Income LoanDefaults
1    F  23000          Yes
2    F  24000          Yes
3    F  25000           No
4    F  26000           No
5    F  27000           No
6    M  25000          Yes
7    M  26000          Yes
8    M  27000          Yes
9    M  30000           No
10   M  31000           No
```

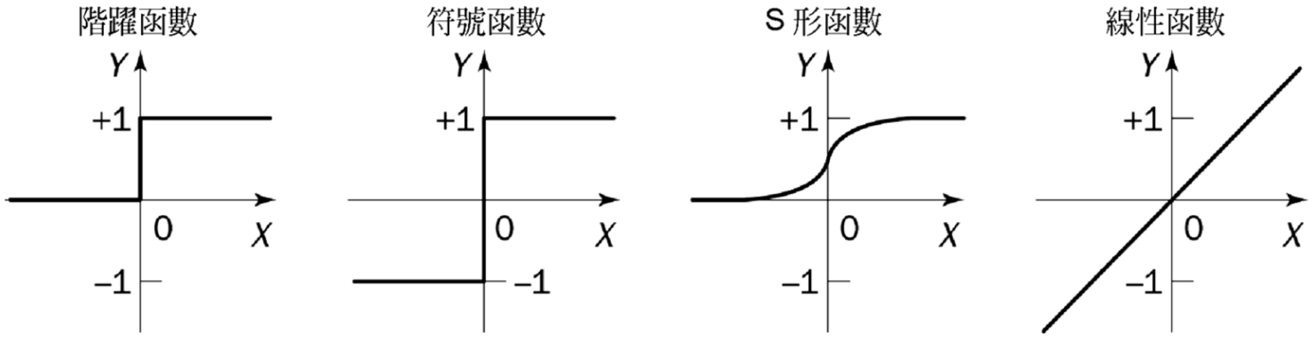


類神經網路主要結構

- 類神經網路主要結構:
  - 輸入層(Input Layer): 輸入的X變數因子。
  - 輸出層(Output Layer): 結果值Y的組合。
  - 中間的隱藏層(Hidden Layer): 負責運算的神經元，通常會以輸入層加上輸出層數目開根號決定單一層隱藏層中的神經元數目。
  - Weights (權重): 代表各個神經元受到刺激後的活躍程度。
  - Bias (偏權值)
- 機器學習就是逐漸調整weights 和bias的過程，使得類神經網路的真實輸出與期望輸出逐漸趨於一致。

神經元如何決定輸出

- 神經元一般透過**轉移函數**將最終的計算結果轉換為輸出，我們通常會依照處理問題的特性來選擇不同的轉移函數。
- 常見的轉移函數包含：步階函數、符號函數、線性函數與S型函數。



$Y^{step} = \begin{cases} 1, \text{若 } X \ge 0 \\ 0, \text{若 } X < 0 \end{cases}$ $Y^{sign} = \begin{cases} +1, \text{若 } X \ge 0 \\ -1, \text{若 } X < 0 \end{cases}$ $Y^{sigmoid} = \dfrac{1}{1 + e^{-X}}$ $Y^{linear} = X$

- 轉換函數 (transform function)
  - - 集成函數 (summation function): Sj = sum_{i=1}^n (Xi*wij) + Bj
  - - 啟動函數 (activation function): F(Sj) = 1/(1 + exp(-Sj))

Xi: 第i個神經元所輸出的資訊
Sj: 第j個神經元所接收資訊的總和
wij: 第i個神經元與第j個神經元之間的權重
Bj: 第j個神經元的Bias(偏權值)

Recoding

```
> income <- (Loan$Income - min(Loan$Income))/(max(Loan$Income)- min(Loan$Income))
> no.class <- length(levels(Loan$LoanDefaults))
> Target <- data.frame(matrix(0, nrow=length(income), ncol=no.class))
> sapply(1:no.class, function(x){
+     Target[which(Loan$LoanDefaults == levels(Loan$LoanDefaults)[x]), x] <<- 1
+ })
[1] 1 1
> colnames(Target) <- c("NO", "YES")
> data.frame(Loan$Sex, income, Target)
   Loan.Sex income NO YES
1         F  0.000  0   1
2         F  0.125  0   1
3         F  0.250  1   0
4         F  0.375  1   0
5         F  0.500  1   0
6         M  0.250  0   1
7         M  0.375  0   1
8         M  0.500  0   1
9         M  0.875  1   0
10        M  1.000  1   0
```

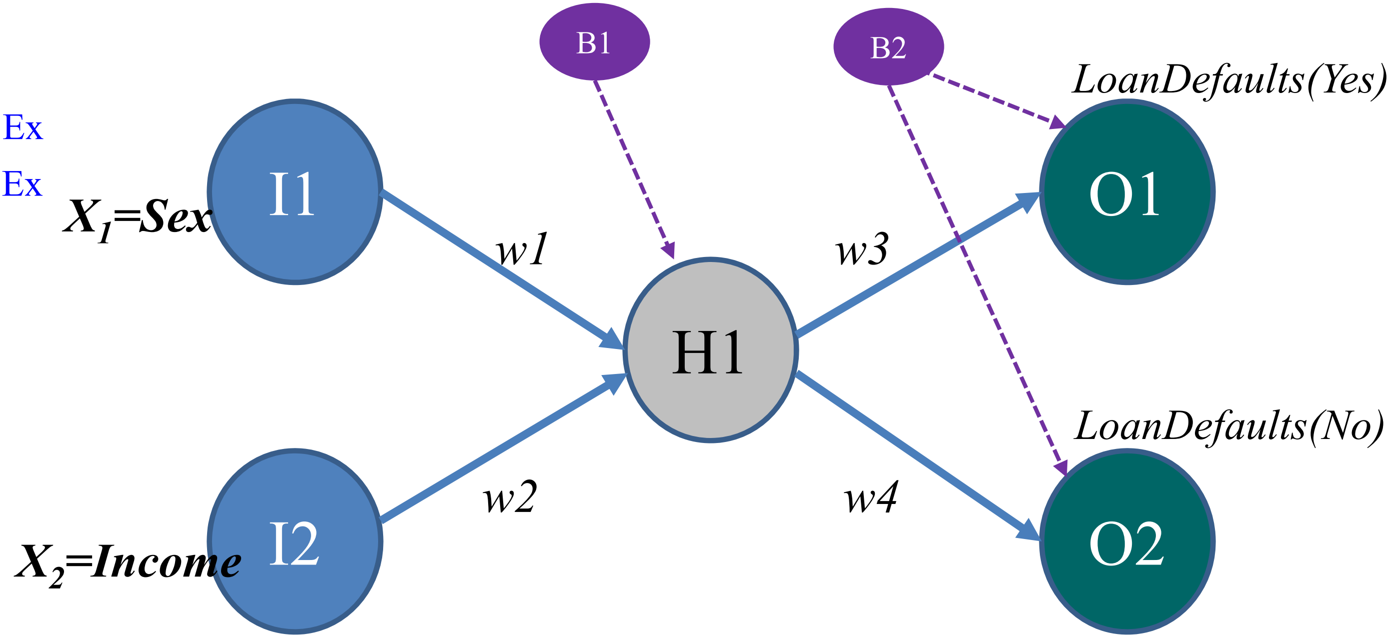
類神經網路演算說明

輸入值 I1	0	
輸入值 I2	0	
期望輸出值 O1	0	O1Ex
期望輸出值 O2	1	O2Ex

初始值

I1 → H1 權值 w1	0.01
I2 → H1 權值 w2	0.01
H1 → O1 權值 w3	0.01
H1 → O2 權值 w4	0.01
偏權值 H1	0.01
偏權值 O1	0.01
偏權值 O2	0.01

學習率	0.1



真實輸出 H1	0.01
	0.5025

真實輸出 O1	0.015025
	0.5037562

真實輸出 O2	0.015025
	0.5037562

O1 誤差	0.1268851
O2 誤差	0.123129
平均誤差	0.5000141

$H1 = I1 * w1 + I2 * w2 + B1H1$

$F(H1) = 1/(1 + \exp(-H1))$

$O1 = F(H1) * w3 + B2O1$

$F(O1) = 1/(1 + \exp(-O1))$

$O2 = F(H1) * w4 + B2O2$

$F(O2) = 1/(1 + \exp(-O2))$

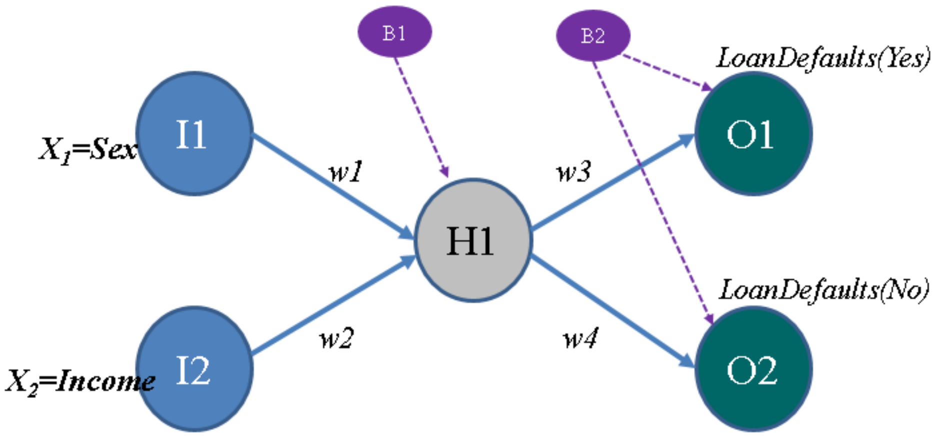
$O1Err = (O1Ex - F(O1))\text{^}2/2$

$O2Err = (O2Ex - F(O2))\text{^}2/2$

$AveErr = \mathrm{sqrt}(O1Err + O2Err)$

權重更新過程

- 誤差項: 節點對誤差的影響
- 輸出層節點對誤差的影響，可看成真實output與期望output的差。
- 隱藏層節點對誤差的影響，可由後面一層的誤差項算出。



O1真實輸出X (1 - O1真實輸出) X (O1期望輸出 - O1真實輸出)

O1 誤差項	-0.1259319
O2 誤差項	0.124054
H1 誤差項	-4.695E-06

O1ErrTerm = F(O1) * (1 - F(O1)) * (O1Ex - F(O1))

O2ErrTerm = F(O2) * (1 - F(O2)) * (O2Ex - F(O2))

H1ErrTerm = F(H1) * (1 - F(H1)) * (O1ErrTerm * w3 + O2ErrTerm * w4)

w1 修正權值	0.01
w2 修正權值	0.01
w3 修正權值	0.0036719
w4 修正權值	0.0162337
H1 修正偏權值	0.0099995
O1 修正偏權值	-0.0025932
O2 修正偏權值	0.0224054

w1(t+1) = w1(t) + alpha * H1ErrTerm * I1

w2(t+1) = w2(t) + alpha * H1ErrTerm * I2

w3(t+1) = w3(t) + alpha * O1ErrTerm * F(H1)(t)

w4(t+1) = w4(t) + alpha * O2ErrTerm * F(H1)(t)

B1H1(t+1) = B1H1(t) + alpha * H1ErrTerm(t)

B2O1(t+1) = B2O1(t) + alpha * O1ErrTerm(t)

B2O2(t+1) = B2O2(t) + alpha * O2ErrTerm(t)

R Packages

- `nnet`: **Feed-Forward Neural Networks and Multinomial Log-Linear Models**
  Software for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models.
  https://cran.r-project.org/web/packages/nnet/

- `neuralnet`: **Training of Neural Networks**
  Training of neural networks using backpropagation, resilient backpropagation with (Riedmiller, 1994) or without weight backtracking (Riedmiller and Braun, 1993) or the modified globally convergent version by Anastasiadis et al. (2005). The package allows flexible settings through custom-choice of error and activation function. Furthermore, the calculation of generalized weights (Intrator O & Intrator N, 1993) is implemented.
  https://cran.r-project.org/web/packages/neuralnet/

- `RSNNS`: **Neural Networks using the Stuttgart Neural Network Simulator (SNNS)**
  The Stuttgart Neural Network Simulator (SNNS) is a library containing many standard implementations of neural networks. This package wraps the SNNS functionality to make it available from within R. Using the 'RSNNS' low-level interface, all of the algorithmic functionality and flexibility of SNNS can be accessed. Furthermore, the package contains a convenient high-level interface, so that the most common neural network topologies and learning algorithms integrate seamlessly into R.
  https://cran.r-project.org/web/packages/RSNNS/

R Package: nnet

```
> # install.packages("nnet")
> library(nnet)
> xdata <- iris[, 1:4]
> ydata <- iris[, 5]
> no.class <- length(unique(ydata))
> T <- matrix(0, nrow=nrow(xdata), ncol=no.class)
> sapply(1:no.class, function(x){
+     T[which(ydata == levels(ydata)[x]), x] <<- 1
+ })
> id <- sample(1:nrow(iris), 100)
> training.x <- xdata[id,]
> training.y <- T[id, ]
> testing.x <- xdata[-id,]
> testing.y <- T[-id, ]
> nn.iris <- nnet(training.x, training.y, size = 2, rang = 0.1,
+             decay = 5e-4, maxit = 200)
# weights:  19
initial  value 73.240851
iter  10 value 37.694980
iter  20 value 32.664756
iter  30 value 31.946509
...
> pred <- max.col(predict(nn.iris, testing.x))
> testing.true <- apply(T[-id, ], 1, which.max)
> table(testing.true, pred)
            pred
testing.true  1  2  3
           1 14  0  0
           2  0 13  2
           3  0  0 21
```

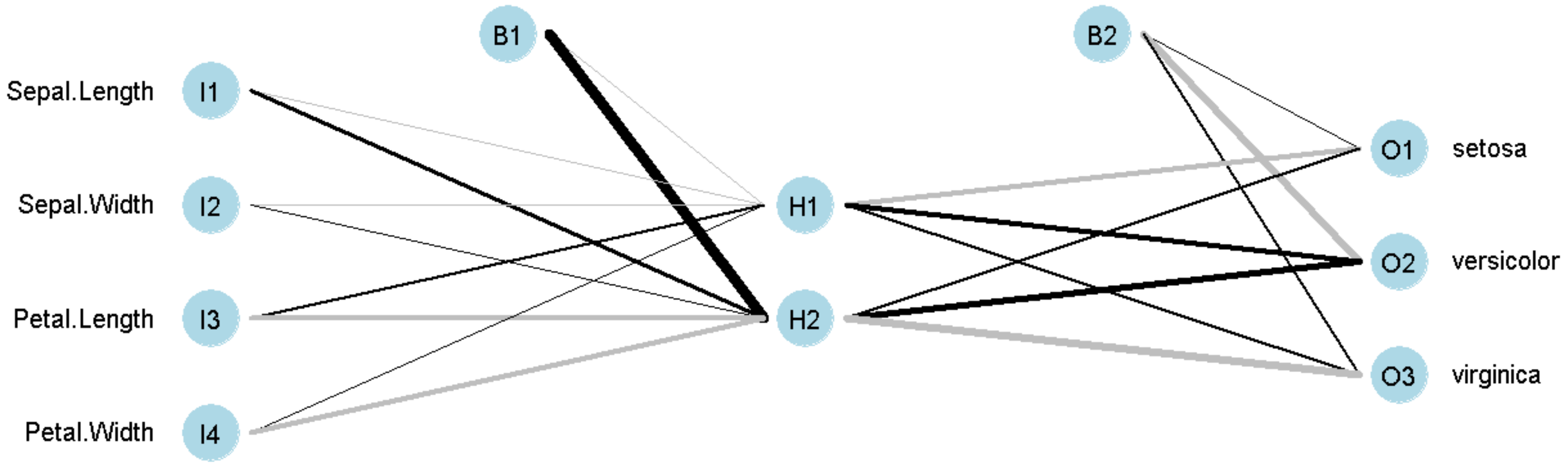
```
> T
       [,1] [,2] [,3]
  [1,]    1    0    0
  [2,]    1    0    0
  [3,]    1    0    0
  [4,]    1    0    0
...
[148,]    0    0    1
[149,]    0    0    1
[150,]    0    0    1
```

- **size**: number of units in the hidden layer.
- **rang**: initial random weights on [-rang, rang]. Value about 0.5 unless the inputs are large, in which case it should be chosen so that rang * max(|x|) is about 1.
- **decay**: parameter for weight decay. Default 0. (It's regularization to avoid over-fitting.)
- **maxit**: maximum number of iterations. Default 100.

R Package: nnet

```
> str(nn.iris)
List of 15
 $ n              : num [1:3] 4 2 3
 ...
$ wts            : num [1:19] -0.209 -0.309 -1.996 2.634 1.349 ...
 $ convergence   : int 1
 $ fitted.values: num [1:100, 1:3] 0.99229 0.009657 0.993522 0.011416 0.000118 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:100] "13" "93" "23" "94" ...
  .. ..$ : NULL
...
> summary(nn.iris)
a 4-2-3 network with 19 weights
options were - decay=5e-04
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1 
 -0.21  -0.31  -2.00   2.63   1.35 
...
 b->o3 h1->o3 h2->o3 
  3.26   3.61 -13.76 
> print(nn.iris)
a 4-2-3 network with 19 weights
options were - decay=5e-04
> coef(nn.iris)
      b->h1       i1->h1       i2->h1       i3->h1       i4->h1        b->h2 
 -0.2088855   -0.3088326   -1.9963053    2.6339182    1.3487065   20.6682103 
     i1->h2       i2->h2       i3->h2       i4->h2        b->o1       h1->o1 
```

R Package: nnet

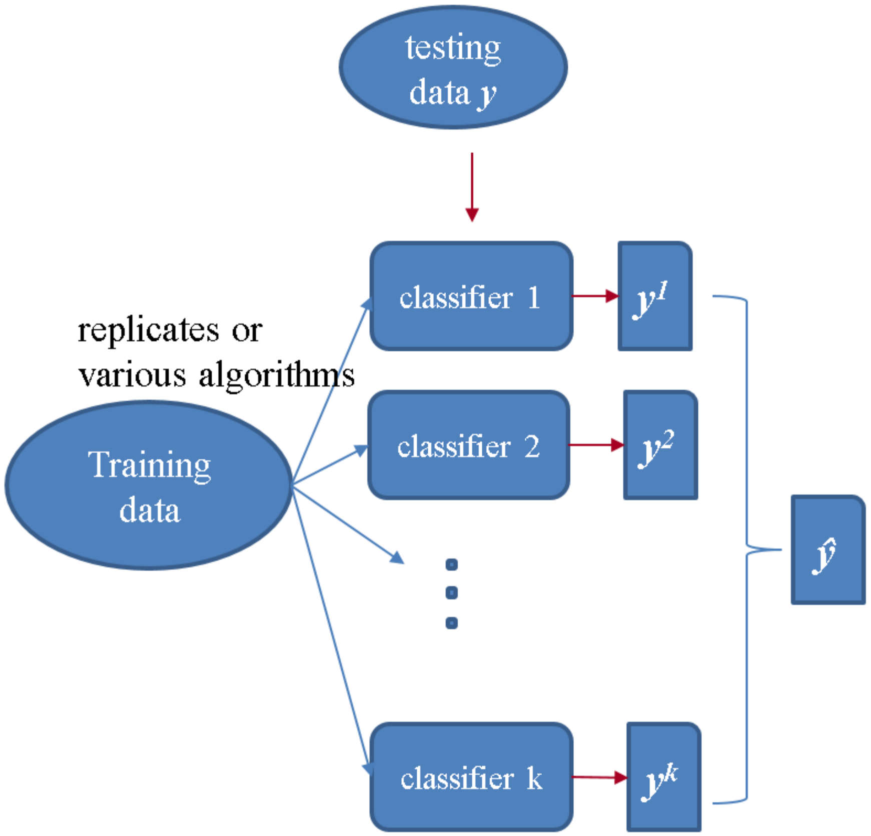


```
# install.packages("NeuralNetTools")
library(NeuralNetTools)
plotnet(nn.iris)
```

```
# or
nn.iris.2 <- nnet(Species ~ ., data = iris, subset = id,
                  size = 2, rang = 0.1,
                  decay = 5e-4, maxit = 200)
table(iris$Species[-id], predict(nn.iris.2, iris[-id,], type = "class"))
plotnet(nn.iris.2)
```

See also: https://rpubs.com/skydome20/R-Note8-ANN

整合學習 (Ensemble Learning)



testing data y

replicates or various algorithms

Training data

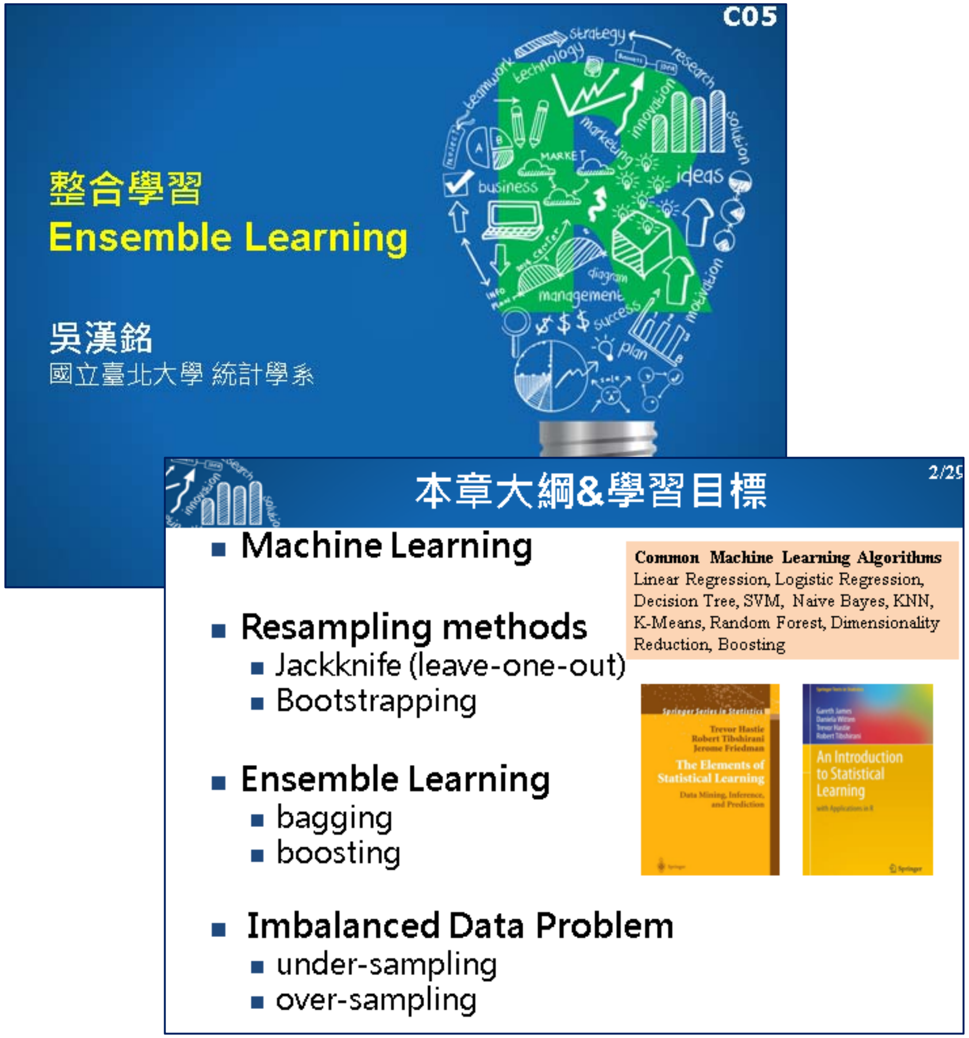
classifier 1 → y^1

classifier 2 → y^2

⋮

classifier k → y^k

$\hat{y}$



C05

整合學習

Ensemble Learning

吳漢銘

國立臺北大學 統計學系

本章大綱&學習目標

2/2

- Machine Learning
- Resampling methods
  - Jackknife (leave-one-out)
  - Bootstrapping
- Ensemble Learning
  - bagging
  - boosting
- Imbalanced Data Problem
  - under-sampling
  - over-sampling

Common Machine Learning Algorithms
Linear Regression, Logistic Regression, Decision Tree, SVM, Naive Bayes, KNN, K-Means, Random Forest, Dimensionality Reduction, Boosting

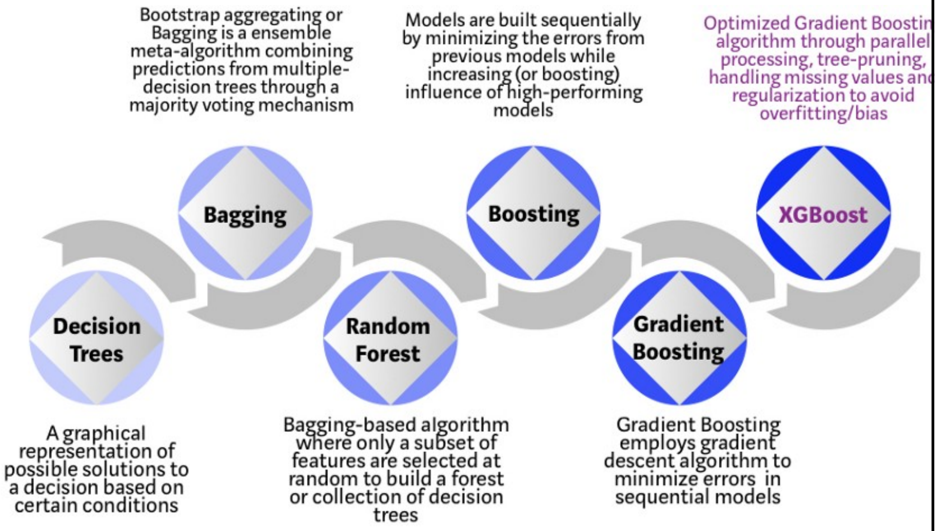
Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
An Introduction to Statistical Learning

http://www.hmwu.idv.tw/web/R/C05-hmwu_R-EnsembleLearning.pdf

XGBoost: Extreme Gradient Boosting

- XGBoost stands for Extreme Gradient Boosting, it is a performant machine learning library based on the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman.
- XGBoost implements a **Gradient Boosting algorithm** based on decision trees.
- Tianqi Chen (陳天奇) and Carlos Guestrin. **XGBoost: A scalable tree boosting system**. In Proceedings of the 22nd ACM SIGKDD InternationalConference on Knowledge Discovery and Data Mining, pages 785–794. ACM, 2016. [被引用 5488 次]
- TianqiChen, **Introduction to Boosted Trees**, Oct. 22 2014. https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

- XGBoost是基於決策樹的集成機器學習算法，使用了梯度提升框架。當分析對象是中小型結構化/表格數據時，基於決策樹的算法被公認是目前同類中最好的。
- NOTE: 若分析對象是非結構化數據(圖像、文本等)，人工神經網絡往往優於所有其他算法或框架。
- XGBoost自發展以來，不僅贏得了衆多的Kaggle競賽，也被認爲是若干前沿行業應用的驅動力。



Decision Trees

A graphical representation of possible solutions to a decision based on certain conditions

Bagging

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism

Random Forest

Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees

Boosting

Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models

Gradient Boosting

Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

XGBoost

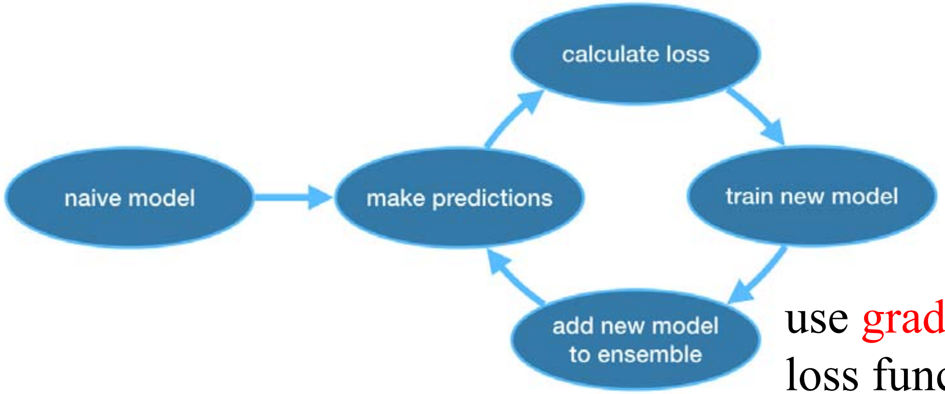
Optimized Gradient Boostin algorithm through parallel processing, tree-pruning, handling missing values anc regularization to avoid overfitting/bias

Evolution of XGBoost Algorithm from Decision Trees

https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

XGBoost特點

- 應用範圍廣泛：可用於解決迴歸、分類、排名和用戶自定義預測問題。
- 可移植性：在Windows、Linux和os x上運行平穩。
- 語言：支持所有主流的編程語言，包括C++、Python、R、java、Scala和Julia。
- 雲集成：支持AWS、Azure和Yarn集羣，與Flink、Spark和其他生態系統配合良好。
- Scalibility : 可水平擴展、平行加速
- Wieghted Quantile：選取特徵切分點的方法
- Sparse Awareness : 特殊處理NA的方式
- Cache Aware Structure : 特殊的資料處理方式，使算法加速
- 靈活性：支持回歸、分類、排名和用戶定義函數
- 跨平台：適用於Windows、Linux、macOS，以及多個雲平台
- 多語言：支持C++, Python, R, Java, Scala, Julia等
- 效果好：贏得許多數據科學和機器學習挑戰。用於多家公司的生產
- 雲端分布式：支持多台計算機上的分布式訓練，包括AWS、GCE、Azure和Yarn集群。可以與Flink、Spark和其他雲數據流系統集成



use gradient descent on the loss function to determine the parameters in this new model.

Image source: https://www.kaggle.com/alexisbcook/xgboost

Boosted Trees (Tianqi Chen)

- Objective function that is everywhere

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

Training Loss measures how well model fit on training data

Regularization, measures complexity of model

- Loss on training data: $L = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$
  - Square loss: $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$
  - Logistic loss: $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$
- Regularization: how complicated the model is?
  - L2 norm: $\Omega(w) = \lambda \|w\|^2$
  - L1 norm (lasso): $\Omega(w) = \lambda \|w\|_1$

- Model: assuming we have K trees

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

- Objective

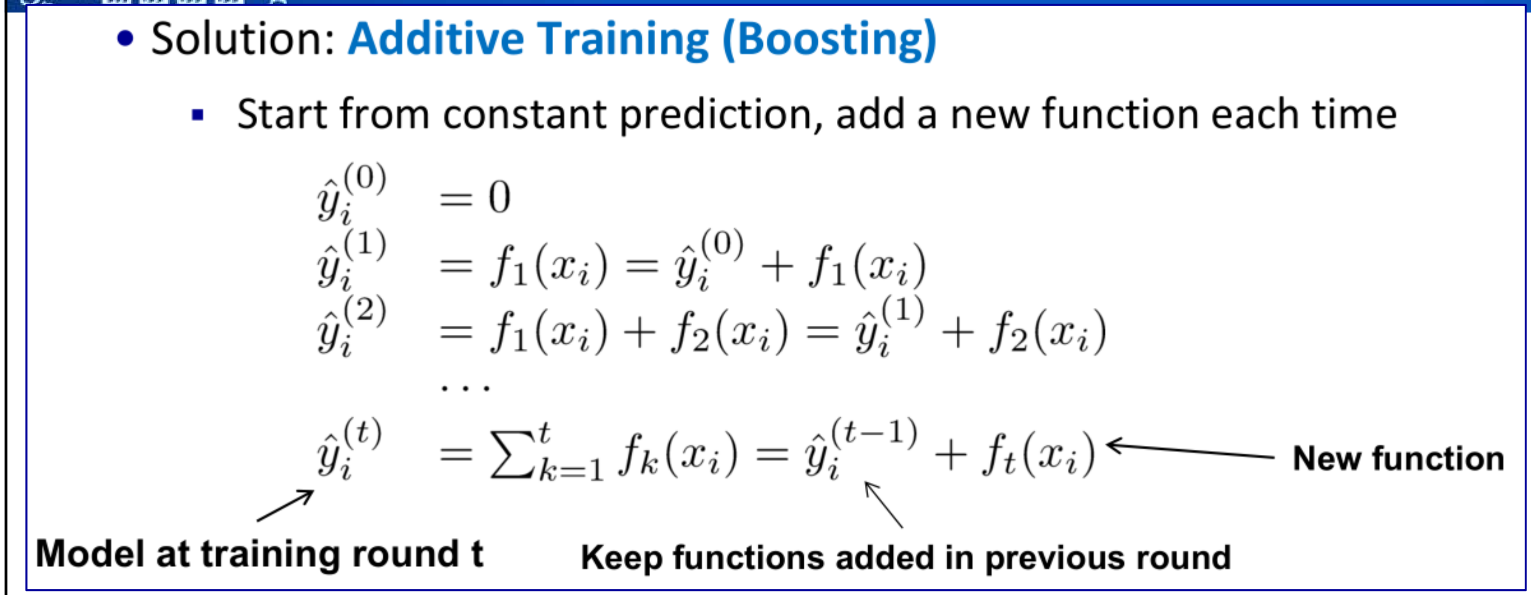
$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training loss

Complexity of the Trees

Source: Tianqi Chen, Introduction to Boosted Trees, Oct. 22 2014.

Boosted Trees (Tianqi Chen)



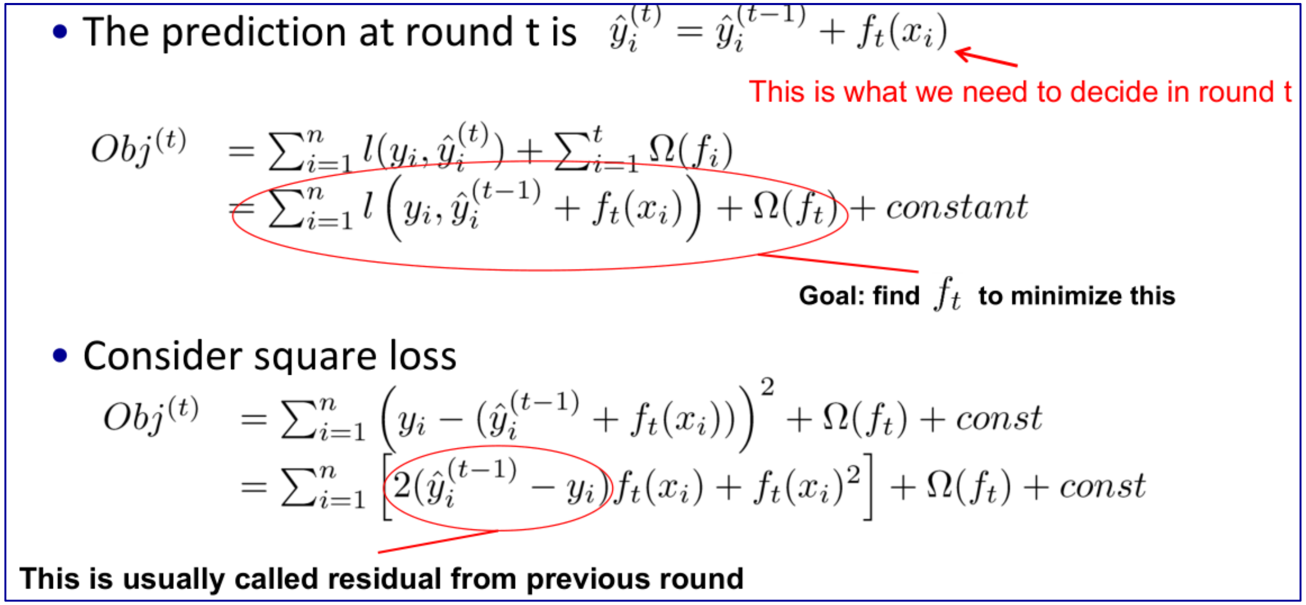
- Solution: **Additive Training (Boosting)**
  - Start from constant prediction, add a new function each time

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0\\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)\\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)\\
&\dots\\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

New function

Model at training round t

Keep functions added in previous round



- The prediction at round t is $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$

This is what we need to decide in round t

$$\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)\\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant
\end{aligned}$$

Goal: find f_t to minimize this

- Consider square loss

$$\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} \left(y_i - (\hat{y}_i^{(t-1)} + f_t(x_i))\right)^2 + \Omega(f_t) + const\\
&= \sum_{i=1}^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t) + const
\end{aligned}$$

This is usually called residual from previous round

Source: Tianqi Chen, Introduction to Boosted Trees, Oct. 22 2014.

Boosted Trees (Tianqi Chen)

Take Taylor expansion of the objective

- Objective, with constants removed

$$\sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t)$$

  - where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

use gradient descent on the loss function to determine the parameters in this new model.

- Define complexity as (this is not the only possible definition)

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

Number of leaves

L2 norm of leaf scores

- We define tree by a vector of scores in leafs, and a leaf index mapping function that maps an instance to a leaf

$$f_t(x) = w_{q(x)}, \quad w \in \mathbf{R}^T, q : \mathbf{R}^d \to \{1, 2, \cdots, T\}$$

The leaf weight of the tree

The structure of the tree

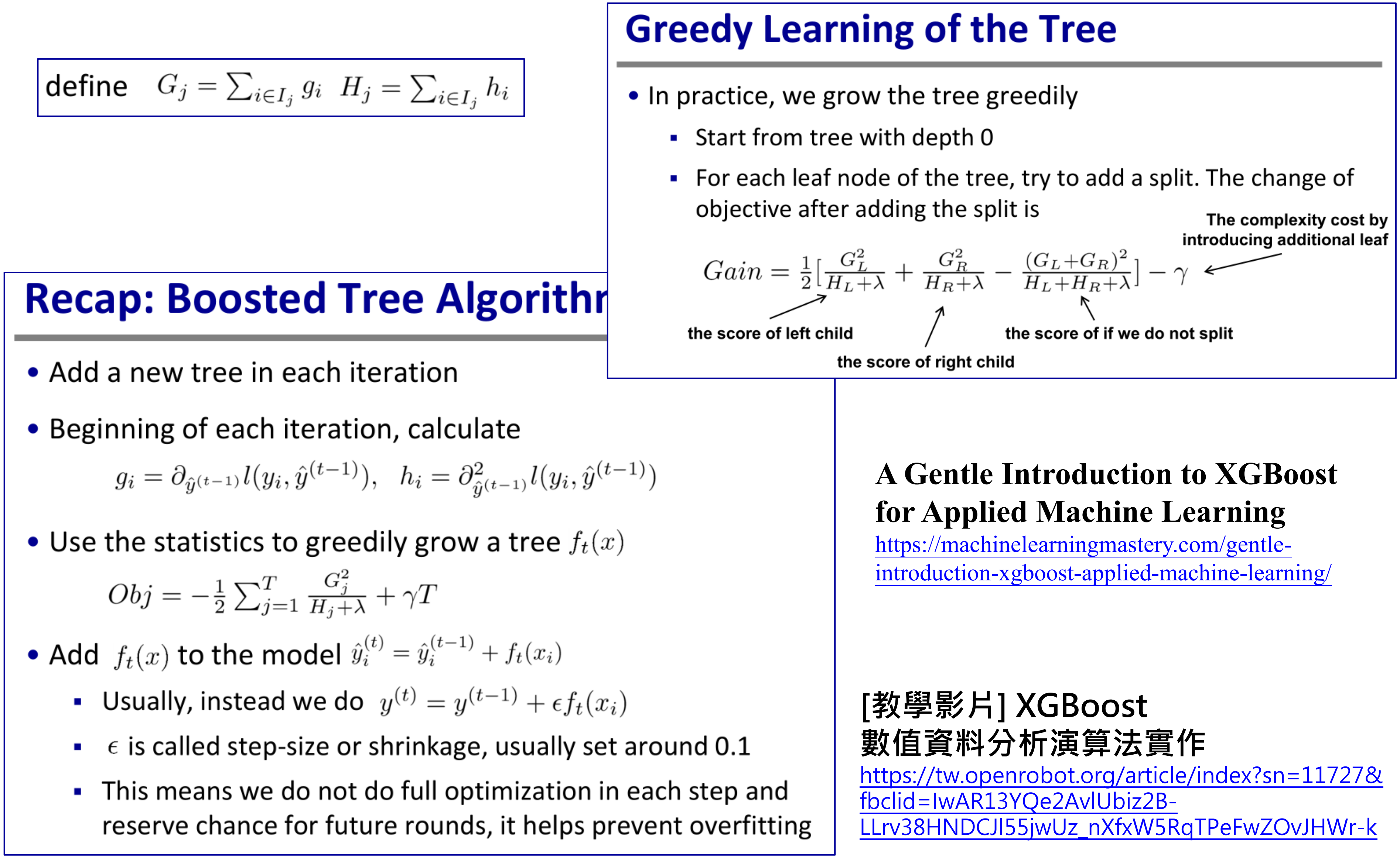
- Regroup the objective by each leaf

$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t) \\ &= \sum_{i=1}^{n}\left[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2\right] + \gamma T + \lambda\frac{1}{2}\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}\left[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2\right] + \gamma T \end{aligned}$$

This is sum of T independent quadratic functions

Source: Tianqi Chen, Introduction to Boosted Trees, Oct. 22 2014.

Boosted Trees (Tianqi Chen)



define $G_j = \sum_{i \in I_j} g_i \quad H_j = \sum_{i \in I_j} h_i$

Greedy Learning of the Tree

- In practice, we grow the tree greedily
  - Start from tree with depth 0
  - For each leaf node of the tree, try to add a split. The change of objective after adding the split is

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

The complexity cost by introducing additional leaf

the score of left child

the score of right child

the score of if we do not split

Recap: Boosted Tree Algorithr

- Add a new tree in each iteration
- Beginning of each iteration, calculate

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

- Use the statistics to greedily grow a tree $f_t(x)$

$$Obj = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j+\lambda} + \gamma T$$

- Add $f_t(x)$ to the model $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$
  - Usually, instead we do $y^{(t)} = y^{(t-1)} + \epsilon f_t(x_i)$
  - ϵ is called step-size or shrinkage, usually set around 0.1
  - This means we do not do full optimization in each step and reserve chance for future rounds, it helps prevent overfitting

A Gentle Introduction to XGBoost for Applied Machine Learning
https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[教學影片] XGBoost 數值資料分析演算法實作
https://tw.openrobot.org/article/index?sn=11727&fbclid=IwAR13YQe2AvlUbiz2B-LLrv38HNDCJl55jwUz_nXfxW5RqTPeFwZOvJHWr-k

Source: Tianqi Chen, Introduction to Boosted Trees, Oct. 22 2014.

xgboost: Extreme Gradient Boosting

```
xgb.train(
  params = list(),
  data,
  nrounds,
  watchlist = list(),
  obj = NULL,
  feval = NULL,
  verbose = 1,
  print_every_n = 1L,
  early_stopping_rounds = NULL,
  maximize = NULL,
  save_period = NULL,
  save_name = "xgboost.model",
  xgb_model = NULL,
  callbacks = list(),
  ...
)
```

```
xgboost(
  data = NULL,
  label = NULL,
  missing = NA,
  weight = NULL,
  params = list(),
  nrounds,
  verbose = 1,
  print_every_n = 1L,
  early_stopping_rounds = NULL,
  maximize = NULL,
  save_period = NULL,
  save_name = "xgboost.model",
  xgb_model = NULL,
  callbacks = list(),
  ...
)
```

https://cran.r-project.org/web/packages/xgboost/

- `booster: gbtree` or `gblinear.`
- `xgb.train` accepts only an `xgb.DMatrix` as the input.
- `xgboost` accepts `matrix`, `dgCMatrix`, or name of a local data file.
- categorical variables should convert to numeric
- could custom objective and evaluation metric
- `xgb.cv {xgboost}`: The cross validation function of xgboost

MUST READ:

- https://cran.r-project.org/web/packages/xgboost/vignettes/discoverYourData.html
- https://cran.r-project.org/web/packages/xgboost/vignettes/xgboostPresentation.html

Example: xgboost for Iris Data

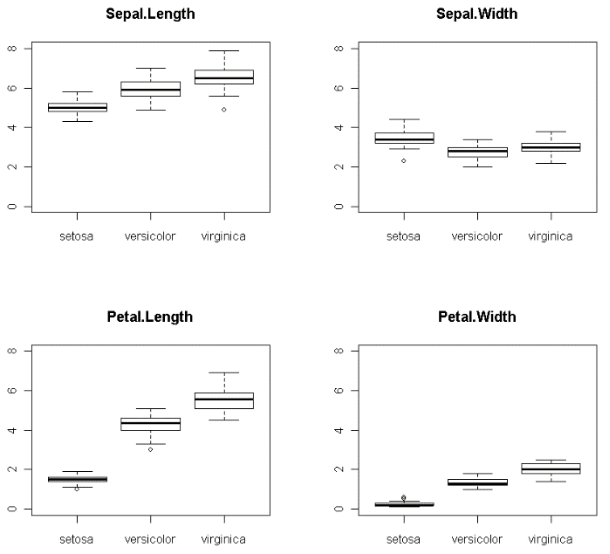
```
> # install.packages("xgboost")
> # install.packages("DiagrammeR")
> library(xgboost)
> library(DiagrammeR); require(Matrix)
>
> # training set and testing test
> id <- sample(2, nrow(iris), replace=TRUE, prob=c(0.8, 0.2))
> iris.train <- iris[id==1, ]
> iris.test <- iris[id==2, ]
> iris.train.x <- as.matrix(iris.train[, -5])
> iris.train.y <- as.integer(iris.train[, 5]) - 1  # 0, 1, ...
> iris.test.x <- as.matrix(iris.test[, -5])
> iris.test.y <- as.integer(iris.test[, 5]) - 1
>
> dim(iris.train.x)
[1] 114   4
> dim(iris.test.x)
[1] 36  4
>
>
> # conver data.frame to DMatrix class for xgboost
> iris.DM.train <- xgb.DMatrix(data=iris.train.x, label=iris.train.y)
> iris.DM.test <- xgb.DMatrix(data=iris.test.x, label=iris.test.y)
> str(iris.DM.train)
Class 'xgb.DMatrix' <externalptr>
 - attr(*, ".Dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
> no.class <- length(unique(iris.train.y))
```

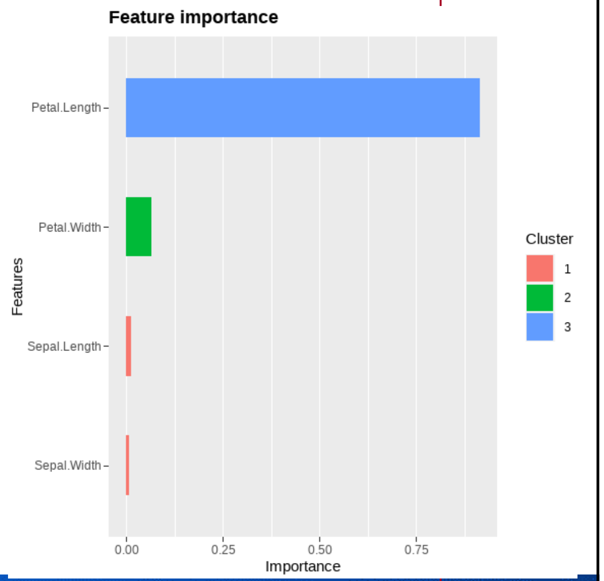
Example: xgboost for Iris Data

```
> # set up parameters
> watchlist <- list(train=iris.DM.train, eval=iris.DM.test)
> param <- list(booster="gbtree", objective="multi:softprob",
+               eval_metric="mlogloss", num_class=no.class)
>
> # build the model using "xgb.train"
> xgb.result <- xgb.train(param, iris.DM.train, nrounds=5, watchlist)
[1]     train-mlogloss:0.743538 eval-mlogloss:0.763476
...
[5]     train-mlogloss:0.234911 eval-mlogloss:0.311803
> str(xgb.result)
List of 9
 $ handle        :Class 'xgb.Booster.handle' <externalptr>
...
 $ feature_names : chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
 $ nfeatures     : int 4
 - attr(*, "class")= chr "xgb.Booster"
>
> # or build the model using "xgboost"
> xgb.model <- xgboost(param=param, data=iris.DM.train, nrounds=20)
[1]     train-mlogloss:0.743538
[2]     train-mlogloss:0.535182
...
[20]    train-mlogloss:0.036144
> # xgb.model <- xgboost(data=iris.train.x, label=iris.train.y,
> #                max_depth=2, eta=1, nthread=2, nrounds=20,
> #                objective="multi:softprob", num_class=3)
```

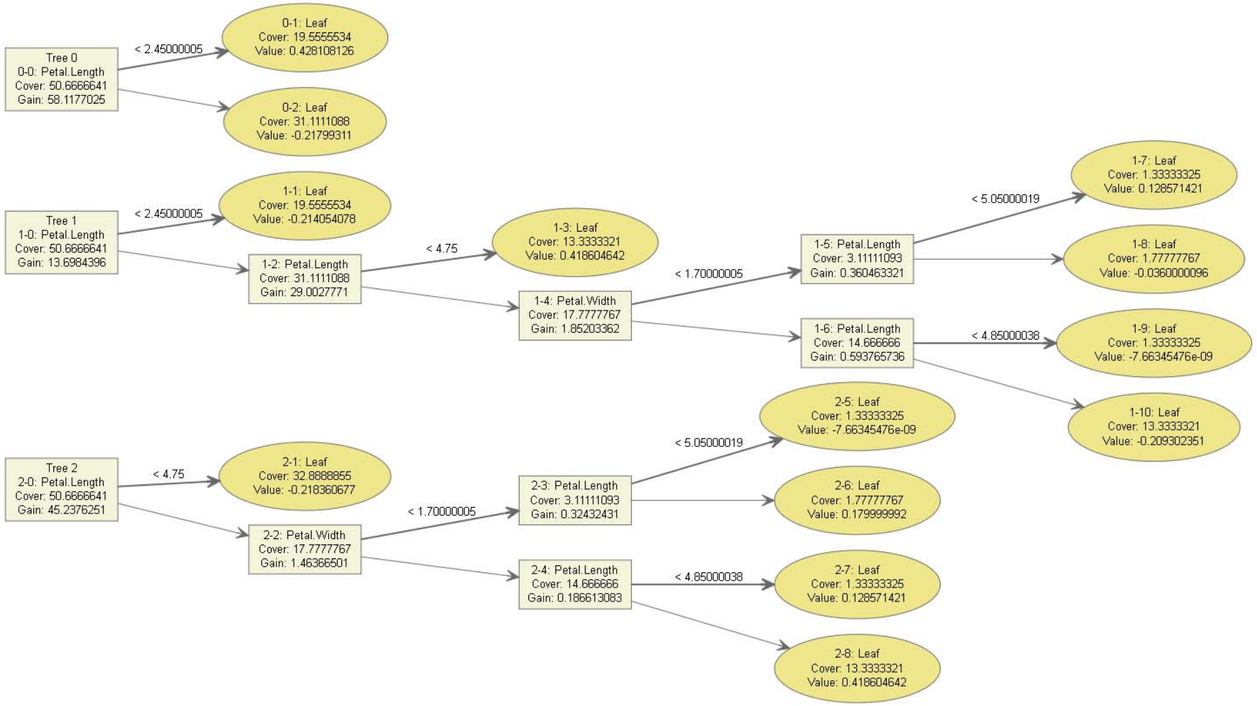
Example: xgboost for Iris Data

```
> # get the prediction for testing data
> Ypred <- predict(xgb.model, iris.DM.test)
> Ypred <- t(matrix(Ypred, no.class, length(Ypred)/no.class))
> at <- apply(Ypred, 1, which.max)
> Ypred <- factor(levels(iris$Species)[at],
> +               levels=levels(iris$Species))
>
> # confusion matrix
> cm <- table(Ytest=iris.test[,5], Ypred)
> cm
            Ypred
Ytest        setosa versicolor virginica
  setosa          6          0         0
  versicolor      0         15         1
  virginica       0          1        13
>
> # accuracy
> sum(diag(cm))/sum(cm)
[1] 0.9444444
> # variable importance
> imp <- xgb.importance(names(iris.train.x), model=xgb.model)
> print(imp)
        Feature        Gain      Cover Frequency
1: Petal.Length 0.915643044 0.73207482 0.5120482
2:  Petal.Width 0.065798318 0.15832632 0.1807229
3: Sepal.Length 0.011720103 0.07458112 0.1807229
4:  Sepal.Width 0.006838536 0.03501775 0.1265060
> xgb.plot.importance(imp)
> xgb.ggplot.importance(imp)
> xgb.plot.tree(feature_names=names(iris[,-5]), model=xgb.model,
+               trees=0:2, show_node_id=TRUE)
```





Example: xgboost for Iris Data



More examples:

- Parameters tuning for caret , xgboost: https://steve-chen.tw/?p=438
- Multiclass Classification with xgboost in R: https://rpubs.com/mharris/multiclass_xgboost
- Run xgboost and plot the ROC curve using Titanic dataset from Kaggle http://jamesmarquezportfolio.com/get_up_and_running_with_xgboost_in_r.html
- Plotting the AUC from an xgboost model in R https://stackoverflow.com/questions/46736934/plotting-the-auc-from-an-xgboost-model-in-r
- How to Use XGBoost for Time Series Forecasting https://machinelearningmastery.com/xgboost-for-time-series-forecasting/
  中文版: 如何使用XGBoost模型進行時間序列預測 (https://bangqu.com/2Y1ftF.html)

Tuning XGBoost Parameters

The XGBoost Advantage:

- Regularization, Parallel Processing, High Flexibility, Handling Missing Values, Tree Pruning, Built-in Cross-Validation, Continue on Existing Model

XGBoost Parameters:

- General Parameters: guide the overall functioning
- Booster Parameters: guide the individual booster (tree/regression) at each step
- Learning Task Parameters: guide the optimization performed

Useful links:

- https://xgboost.readthedocs.io/en/latest/tutorials/param_tuning.html
- https://towardsdatascience.com/doing-xgboost-hyper-parameter-tuning-the-smart-way-part-1-of-2-f6d255a45dde
- https://www.hackerearth.com/zh/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/
- https://www.kaggle.com/general/17120
- https://insightr.wordpress.com/2018/05/17/tuning-xgboost-in-r-part-i/

caret: Classification And REgression Training

http://topepo.github.io/caret/index.html

```
> install.packages("caret")
> library(caret)
> ?train
> names(getModelInfo())
  [1] "ada"              "AdaBag"         "AdaBoost.M1"
  [4] "adaboost"         "amdai"          "ANFIS"
  [7] "avNNet"           "awnb"           "awtan"
...
[232] "widekernelpls"    "WM"             "wsrf"
[235] "xgbDART"          "xgbLinear"      "xgbTree"
[238] "xyf"
```

1. Introduction
2. Visualizations
3. Pre-processing (missing values, standardization, ...)
4. Data spliting
5. Model training and tuning

... more

Max Kuhn. Building predictive models in R using the `caret` package. *Journal of Statistical Software.*, 28(5):1–26, 2008. [被引用 2483 次]

- **Comparing Several Classifiers Using MNIST Data**: SVM, k-nearest neighbors, Random Forest, AdaBoost Classifier, Gradient Boosting, Naive Bayes, LDA, QDA, RBMs, Logistic Regression, RBM + Logistic Regression Classifier, https://martin-thoma.com/comparing-classifiers/
- Fernandez-Delgado, M., Cernadas, E., Barro, S. & Amorim, D. **Do we need hundreds of classifiers to solve real world classification problems?** *Journal of Machine Learning Research* 15, 3133–3181 (2014). [被引用 1883 次] (179 classifiers on 121 data sets)

Quick Tour for caret Package

```
library(AppliedPredictiveModeling)
library(caret)
library(pROC)

featurePlot(x=iris[, 1:4], y=iris$Species,
            plot="ellipse",
            auto.key=list(columns=3))


id <- createDataPartition(iris$Species, p=0.8, list=FALSE, times=1)
iris.train <- iris[id, ]
iris.test  <- iris[-id, ]
dim(iris.train)
dim(iris.test)

my.preProcess <- c("center", "scale")
performance.metric <- "Accuracy"

set.seed(12345)
cv.10fold <- trainControl(method="cv", number=10)

rpart.model <- train(Species ~ ., data=iris.train, method="rpart",
                     metric=performance.metric,
                     trControl=cv.10fold, preProcess=my.preProcess)
rpart.model
iris.rpart.pred <- predict(rpart.model, iris.test)
confusionMatrix(iris.rpart.pred, iris.test$Species)
plot(varImp(rpart.model))
```

`metric`: "RMSE" and "Rsquared" for regression and "Accuracy" and "Kappa" for classification. Custom performance metrics via the summaryFunction argument in trainControl)

Quick Tour for caret Package

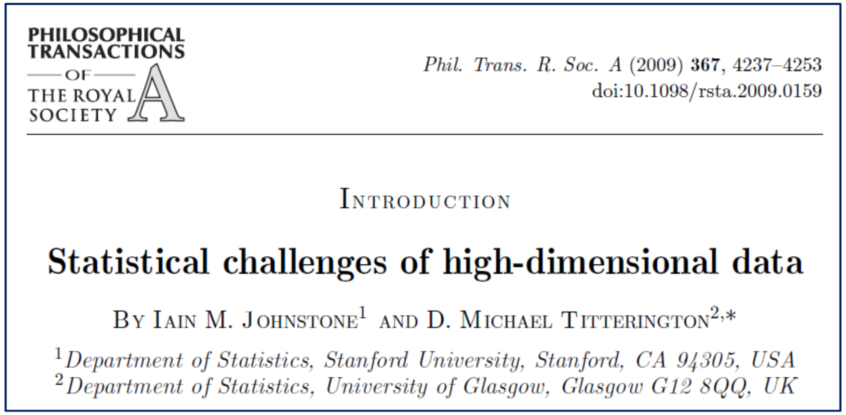
Compare Classifiers

```
rf.model <- train(Species ~ ., data=iris.train, method="rf",
                  metric=performance.metric,
                  trControl=cv.10fold, preProcess=my.preProcess)

svm.model <- train(Species~., data=iris.train, method="svmRadial",
                   metric=performance.metric,
                   trControl=cv.10fold, preProcess=my.preProcess)

results.iris <- resamples(list(rpart=rpart.model, rf=rf.model, svm=svm.model))
summary(results.iris)
dotplot(results.iris)
```

Parameter Tuning

```
# Parameter Tuning: Grid Search: parameter mtry
# mtry: Number of variables randomly sampled as candidates at each split of the tree

control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
tunegrid <- expand.grid(mtry=c(1:15))
rf.gridsearch <- train(Species~., data=iris.train, method="rf",
                       metric=performance.metric, tuneGrid=tunegrid,
                       trControl=control, preProcess=c("center", "scale"))
print(rf.gridsearch)
plot(rf.gridsearch)
```

Classification: large p small n



PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

Phil. Trans. R. Soc. A (2009) **367**, 4237–4253
doi:10.1098/rsta.2009.0159

Introduction

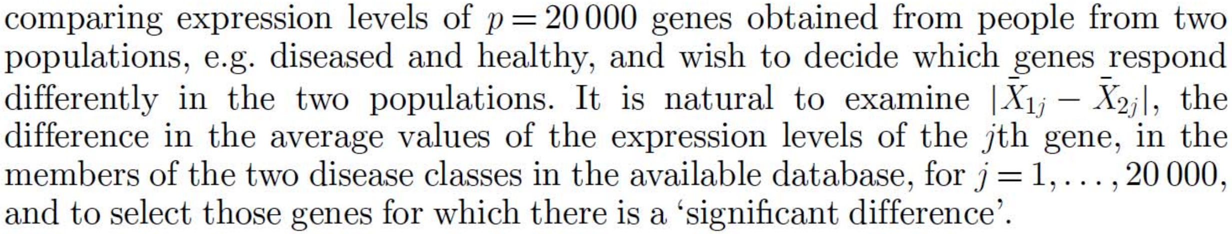
Statistical challenges of high-dimensional data

By Iain M. Johnstone[1] and D. Michael Titterington[2,*]

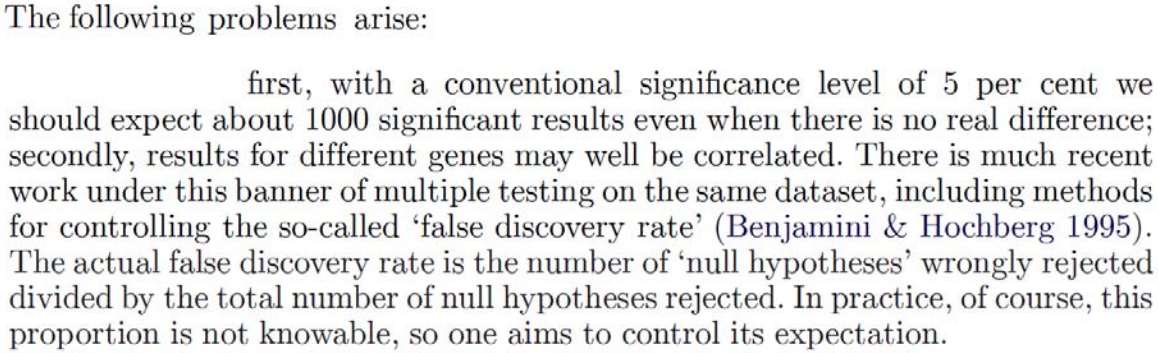
[1] *Department of Statistics, Stanford University, Stanford, CA 94305, USA*
[2] *Department of Statistics, University of Glasgow, Glasgow G12 8QQ, UK*

The large p, small n problem arises here too, and the method can also be described in terms of constrained optimization.

'support vector machine' (SVM)



comparing expression levels of $p = 20\,000$ genes obtained from people from two populations, e.g. diseased and healthy, and wish to decide which genes respond differently in the two populations. It is natural to examine $|\bar{X}_{1j} - \bar{X}_{2j}|$, the difference in the average values of the expression levels of the jth gene, in the members of the two disease classes in the available database, for $j = 1, \ldots, 20\,000$, and to select those genes for which there is a 'significant difference'.



The following problems arise:

first, with a conventional significance level of 5 per cent we should expect about 1000 significant results even when there is no real difference; secondly, results for different genes may well be correlated. There is much recent work under this banner of multiple testing on the same dataset, including methods for controlling the so-called 'false discovery rate' (Benjamini & Hochberg 1995). The actual false discovery rate is the number of 'null hypotheses' wrongly rejected divided by the total number of null hypotheses rejected. In practice, of course, this proportion is not knowable, so one aims to control its expectation.