Machine Learning

Instructor Mary Yang, PhD
Graduate Assistant: Dan Li

What is machine learning?

- Program systems to automatically learn from data and to improve with experience.

- The core of machine learning
  - representation
  - generalization.

Generalization

Training set (labels known)

Test set (labels unknown)

- How well does a learned model generalize from the data it was trained on to a new test set?

Slide credit: L. Lazebnik

The machine learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

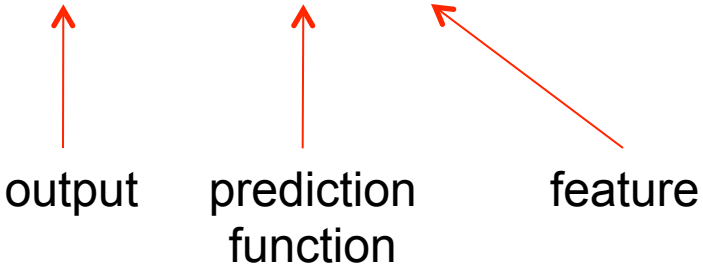
f() = “apple”

f() = “tomato”

f() = “cow”

The machine learning framework

$$y = f(\mathbf{x})$$



- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

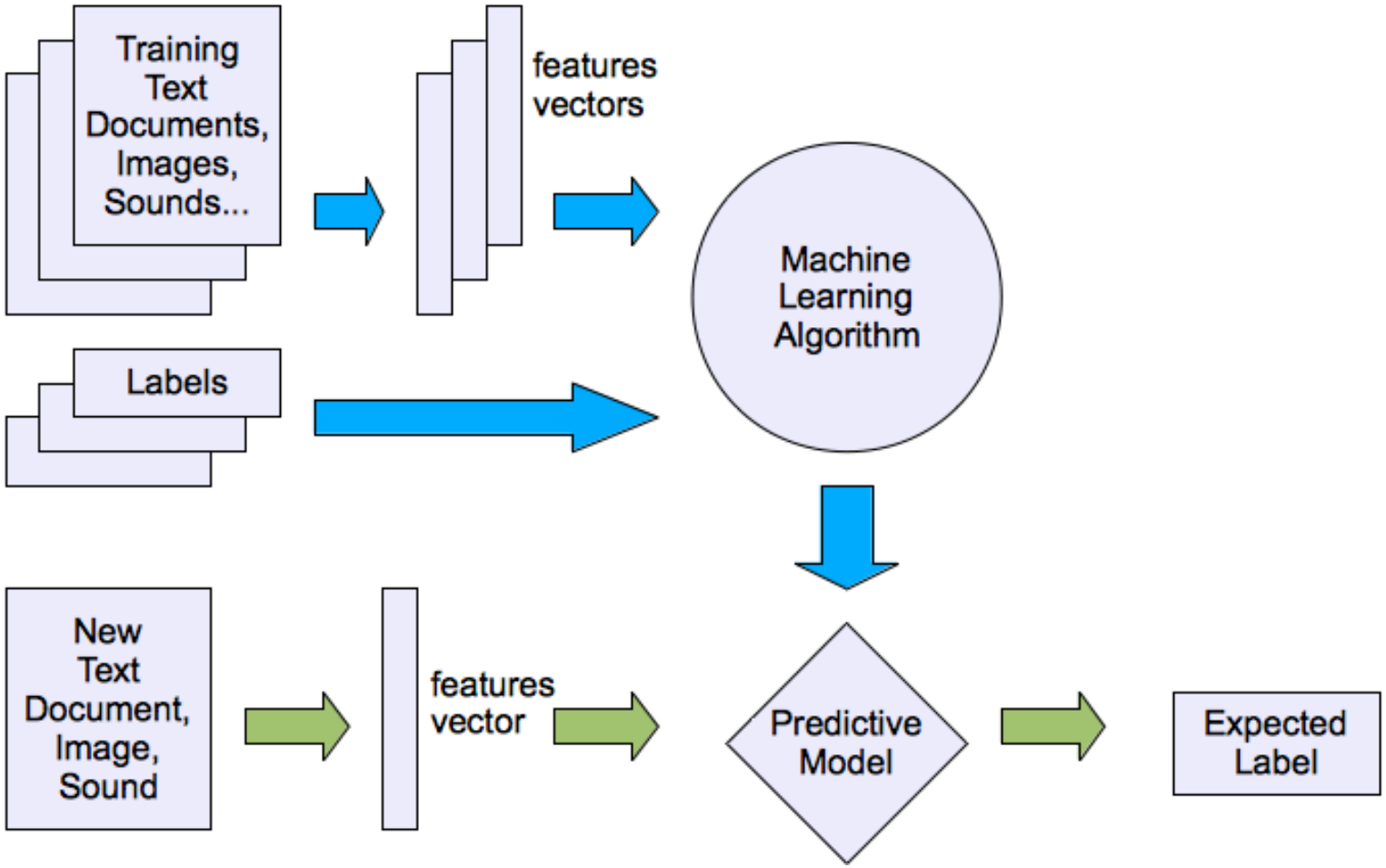
Slide credit: L. Lazebnik

Types of machine learning

- Unsupervised learning
  - Class labels are not provided
  - Identify intrinsic structure in unlabeled data

- Supervised learning
  - Data and class labels are provided
  - Fit model to labeled data
  - Using the model to predict labels of unknown data instances.

Machine learning structure



Unsupervised learning

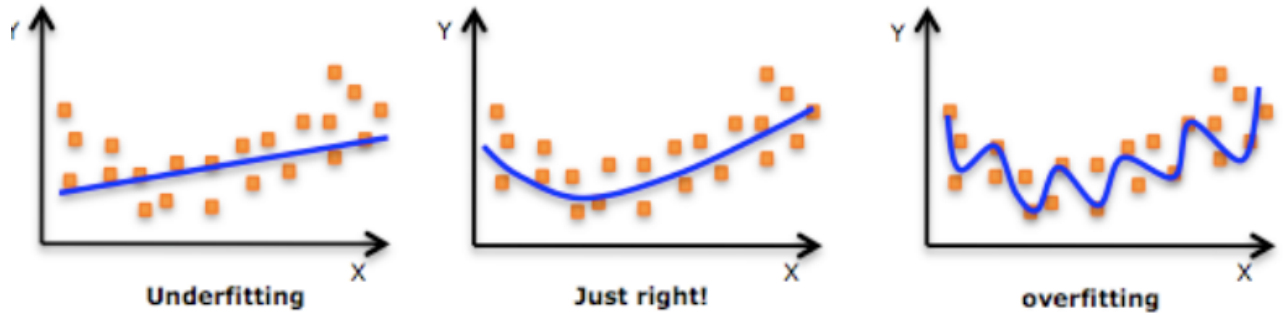
- Approaches
  - Clustering:
    - **K-means**, mixture models, **hierarchical clustering**
  - Blind signal separation using feature extraction for dimensionality reduction
    - Principle component analysis, Independent component analysis

Supervised learning algorithms

- Trees
- **Support Vector Machine(SVMs)**
- Naïve Bayes
- Nearest Neighbor
- Discrimination Analysis
- Regression
  - **Linear regression**

Supervised learning

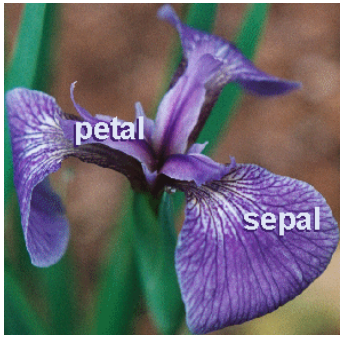
- Procedure
  - Training
    - create a computational model to fit the data; parameters are estimated
  - Validation
    - Estimating parameters in the model.
    - Assess generalization: the ability of the model to produce reasonable outcomes of input data that are not seen during the training.



  - Testing
    - Assess the model using independent data set

R: Hierarchical clustering

- IRIS
  - 3 classes (types of iris flower)
    - 50 setosa
    - 50 Versicolor
    - 50 Virginica
  - 4 numeric attributes
    - Sepal length
    - Sepal width
    - Petal length
    - Petal width



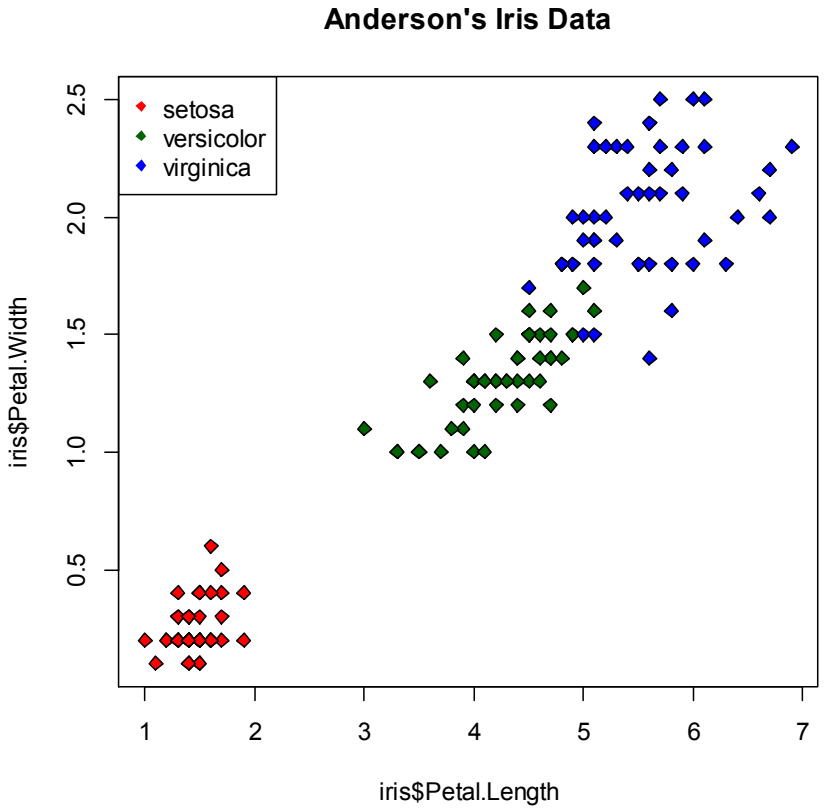
IRIS Dataset

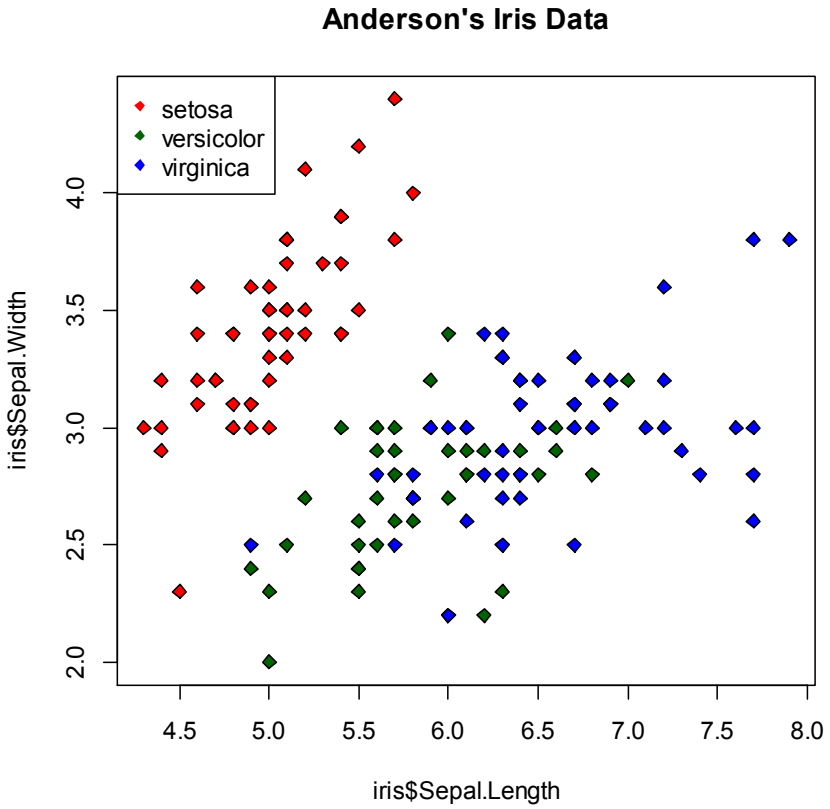
```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> 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 ...
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
> |
```

Take a look inside data

> plot(iris$Petal.Length, iris$Petal.Width, pch=23, bg=c("red","darkgreen","blue")
[unclass(iris$Species)], main="Anderson's Iris Data")

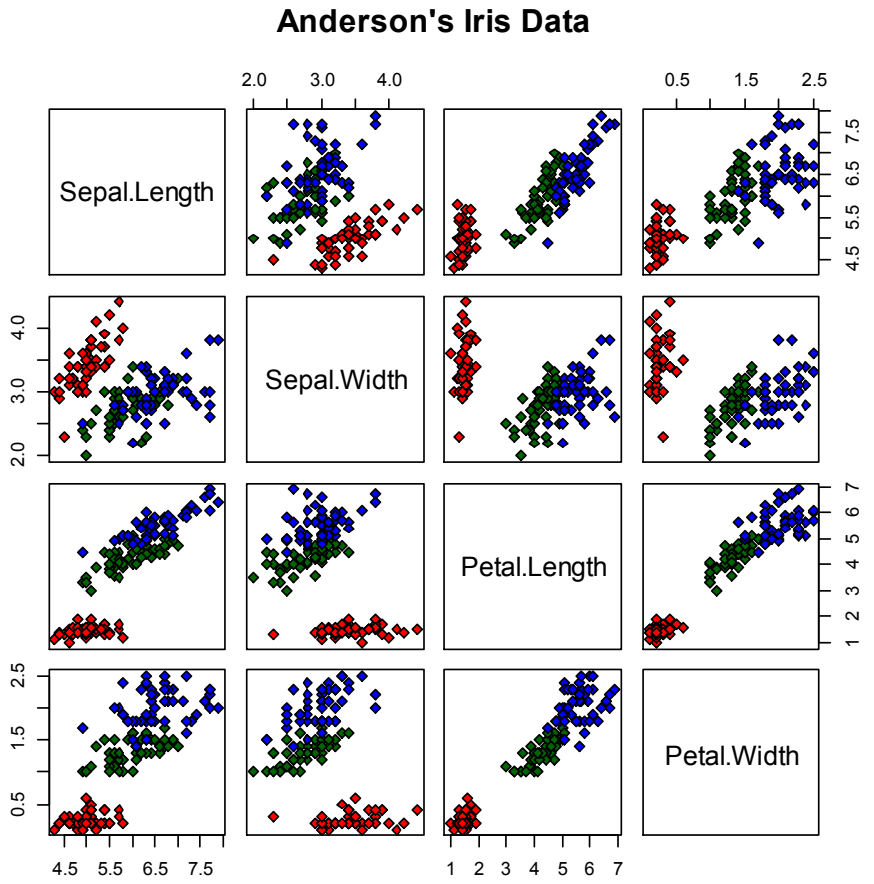
> legend ('topleft', c('setosa', 'versicolor', 'virginica'), pch = rep(18,3), col =
c("red","darkgreen","blue"))





Take a look inside data

```
> pairs(iris[1:4], main = "Edgar Anderson's Iris Data", pch = 23, bg = c("red",
      "darkgreen", "blue")[unclass(iris$Species)])
```



K-means

K-means

- Simple
- Provide the number of expected clusters (K)
- Based on the distance measure between a data point and the cluster center
- Significantly sensitive to the initial randomly selected cluster centers

K-means

Randomly picked K = 3 centers.

K-means (build-in function of R)

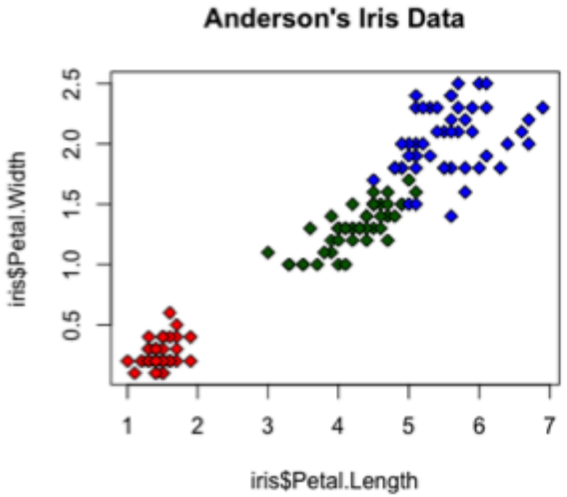
- >?kmeans
- Description
  - Perform k-means clustering on a data matrix.
- Usage
  - kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace=FALSE)

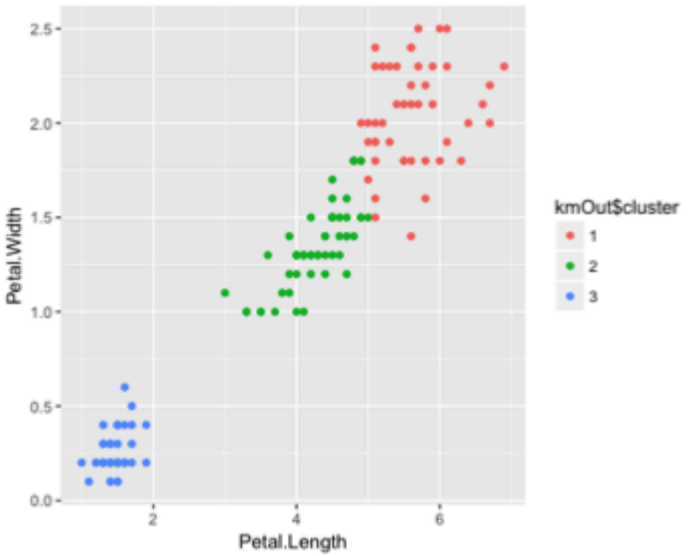
K-means example

```
> mat = iris[,-5]
> head(mat)
  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
> km <- kmeans(mat, 3, iter.max = 20)
> km
K-means clustering with 3 clusters of sizes 38, 50, 62

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.006000    3.428000     1.462000    0.246000
3     5.901613    2.748387     4.393548    1.433871

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [37] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [73] 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1
[109] 1 1 1 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1
[145] 1 1 3 1 1 3

Within cluster sum of squares by cluster:
[1] 23.87947 15.15100 39.82097
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
>
```

K means

```
# Generate left figure
plot(iris$Petal.Length, iris$Petal.Width, pch=23, bg=c("red","darkgreen","blue")[unclass(iris$
      Species)], main="Anderson's Iris Data")

# Generate right figure
set.seed(20)
mat = iris[, -5]
kmOut <- kmeans(mat[,3:4], 3, iter.max = 20)
kmOut$cluster <- as.factor(kmOut$cluster)
ggplot(iris, aes(Petal.Length, Petal.Width, color = kmOut$cluster)) + geom_point()
```

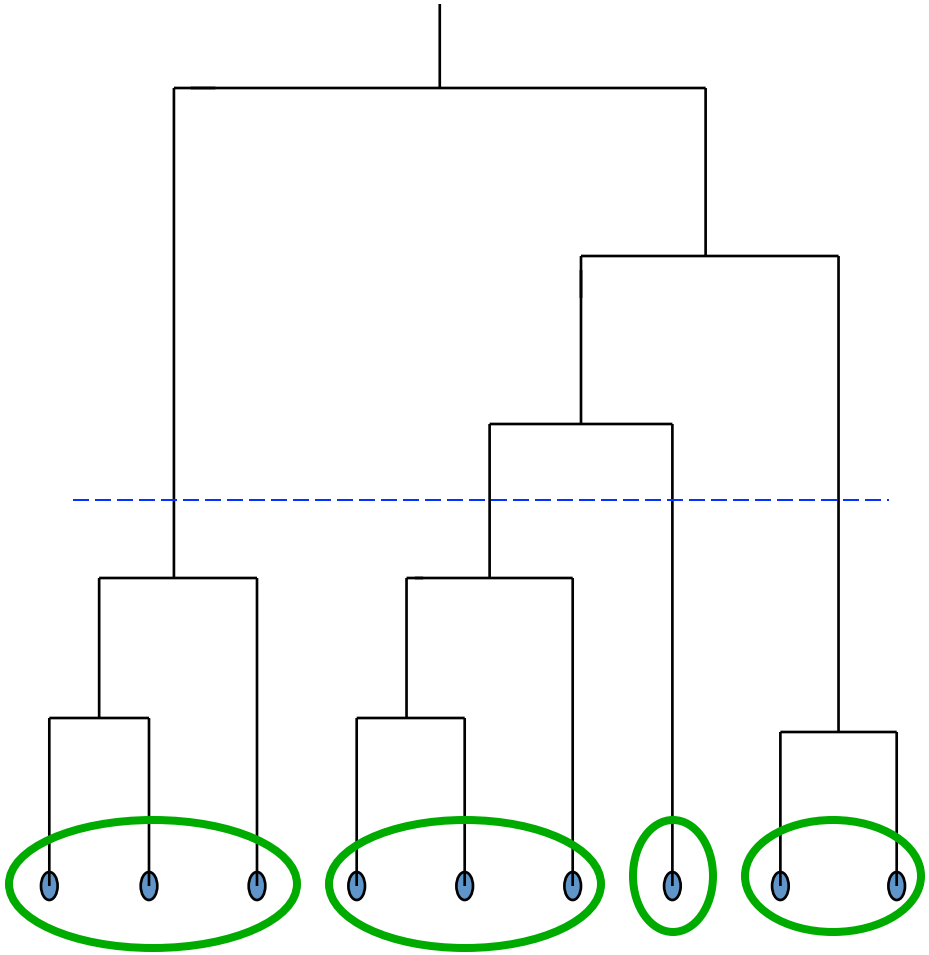




Hierarchical clustering

Dendogram: Hierarchical Clustering

- Clustering obtained by cutting the dendrogram at a desired level: each **connected** component forms a cluster.



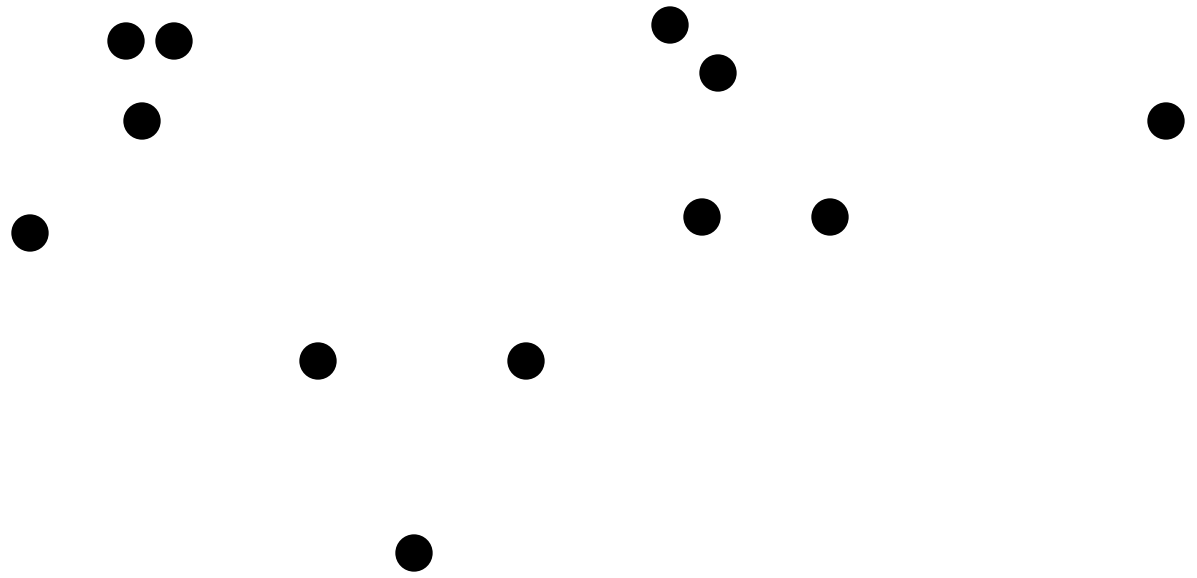
Hierarchical clusters

- **Bottom-up** (agglomerative): start with each component in a separate cluster and merge them accordingly to a given property. Repeat until all clusters are agglomerated into one single cluster.

- **Top down** (division) Starting with all the data in a single cluster, consider each possible way to divide the cluster into two. Choose the best division and recursively operate on both sides.

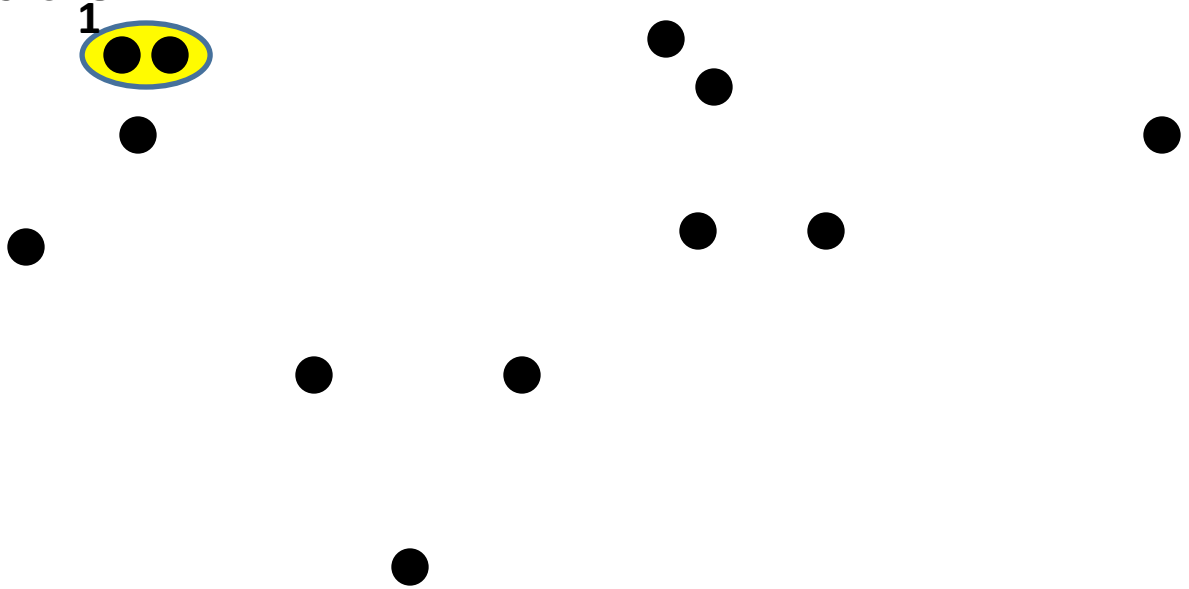
Hierarchical clustering

- Agglomerative (Bottom up)



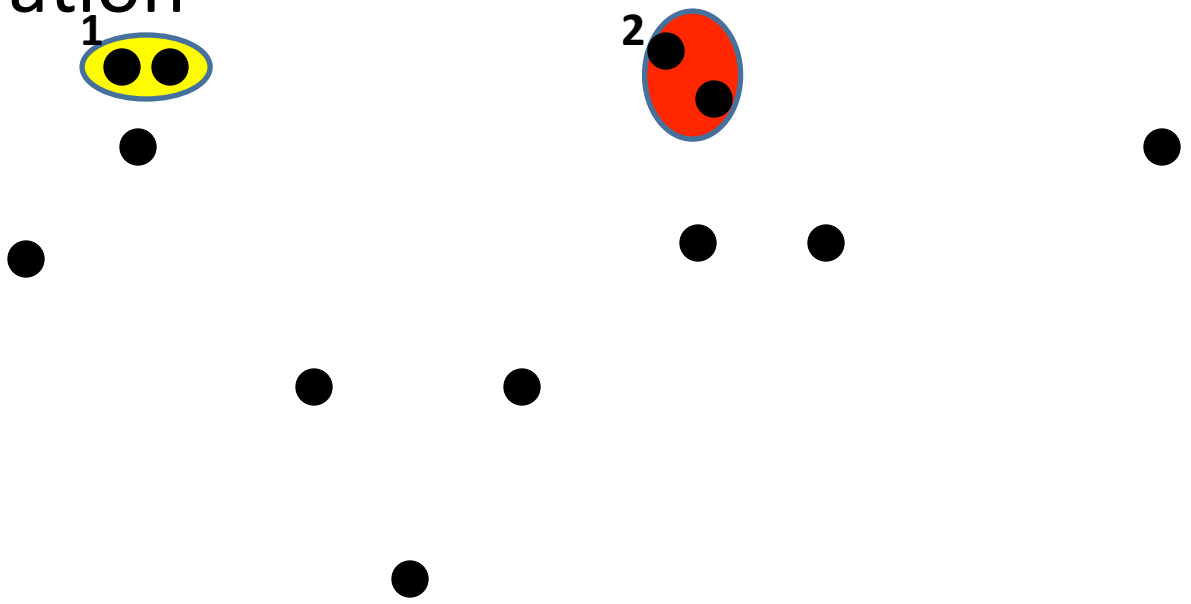
Hierarchical clustering

- Agglomerative (Bottom up)
- 1st iteration



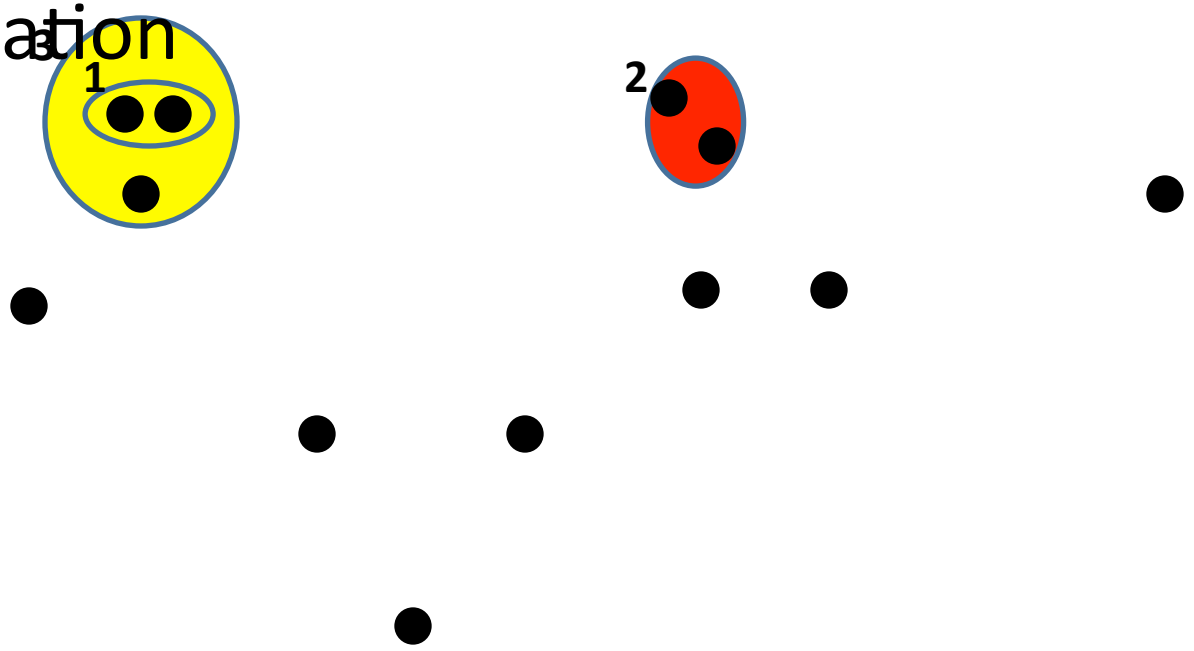
Hierarchical clustering

- Agglomerative (Bottom up)
- 2nd iteration



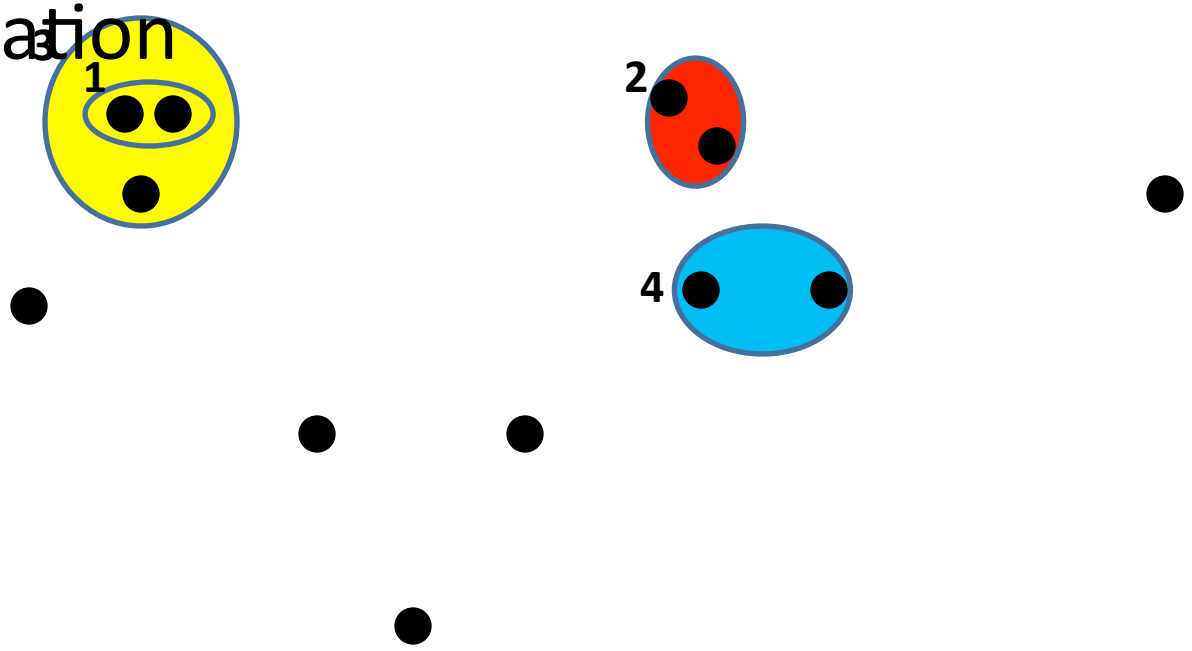
Hierarchical clustering

- Agglomerative (Bottom up)
- 3rd iteration



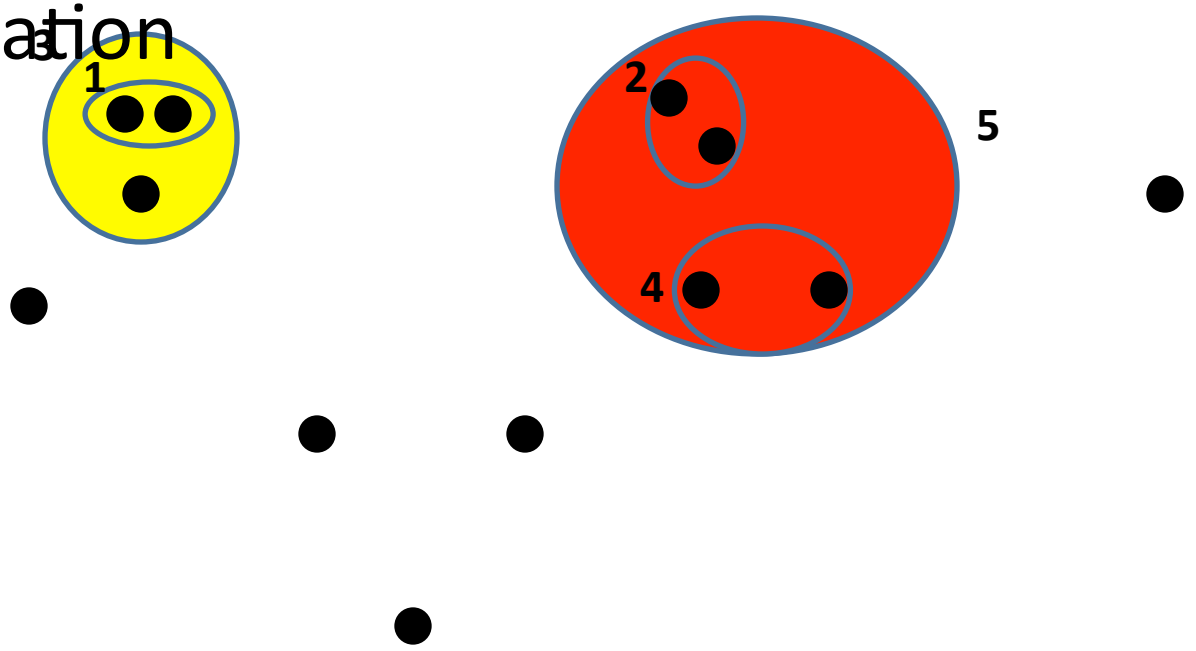
Hierarchical clustering

- Agglomerative (Bottom up)
- 4th iteration



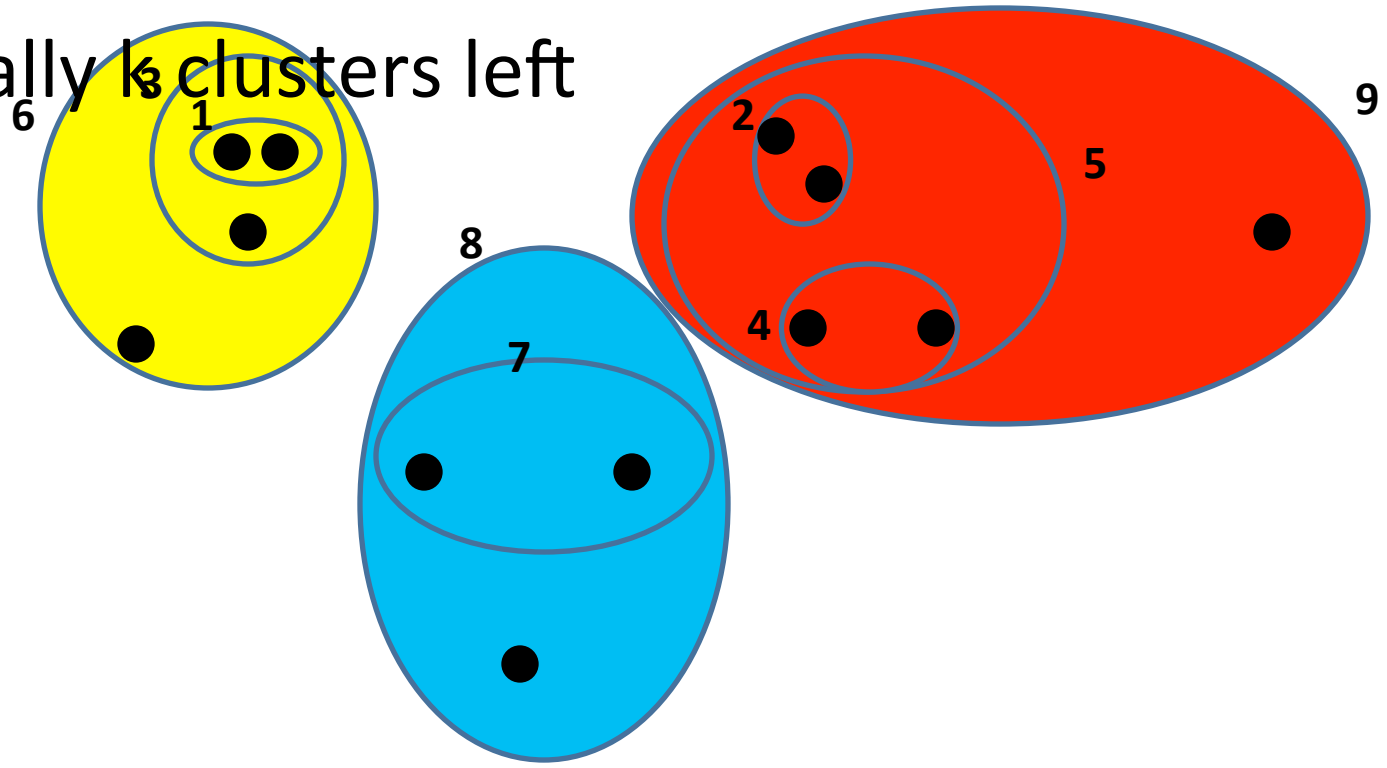
Hierarchical clustering

- Agglomerative (Bottom up)
- 5th iteration

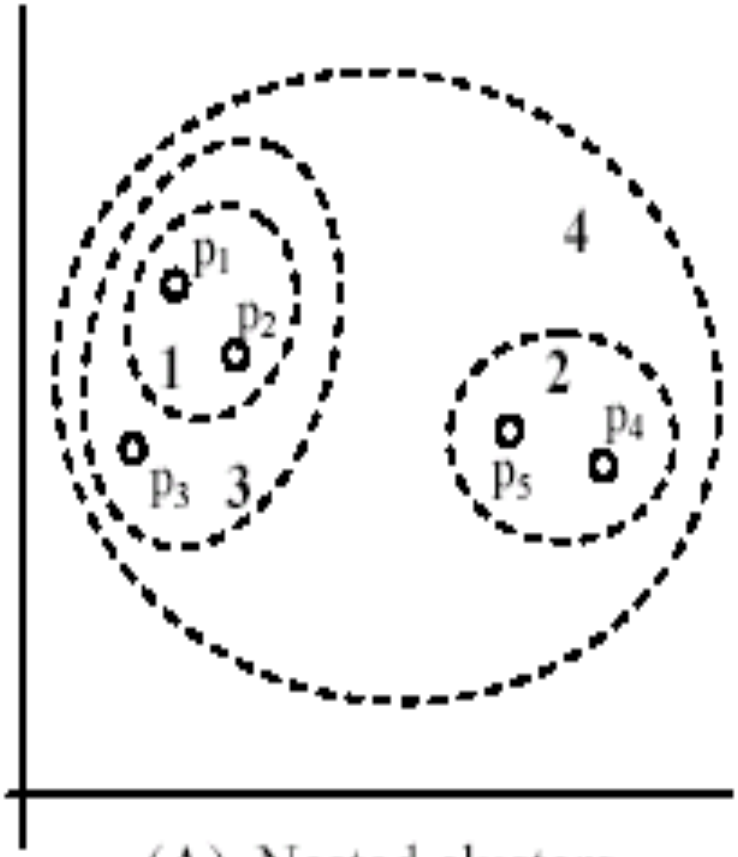


Hierarchical clustering

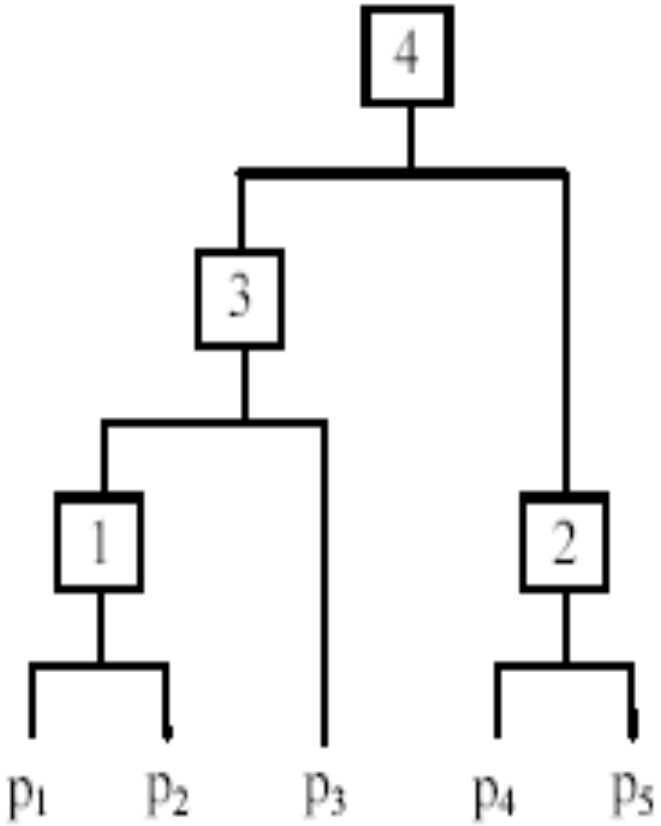
- Agglomerative (Bottom up)
- Finally k clusters left



An example: working of bottom-up algorithm



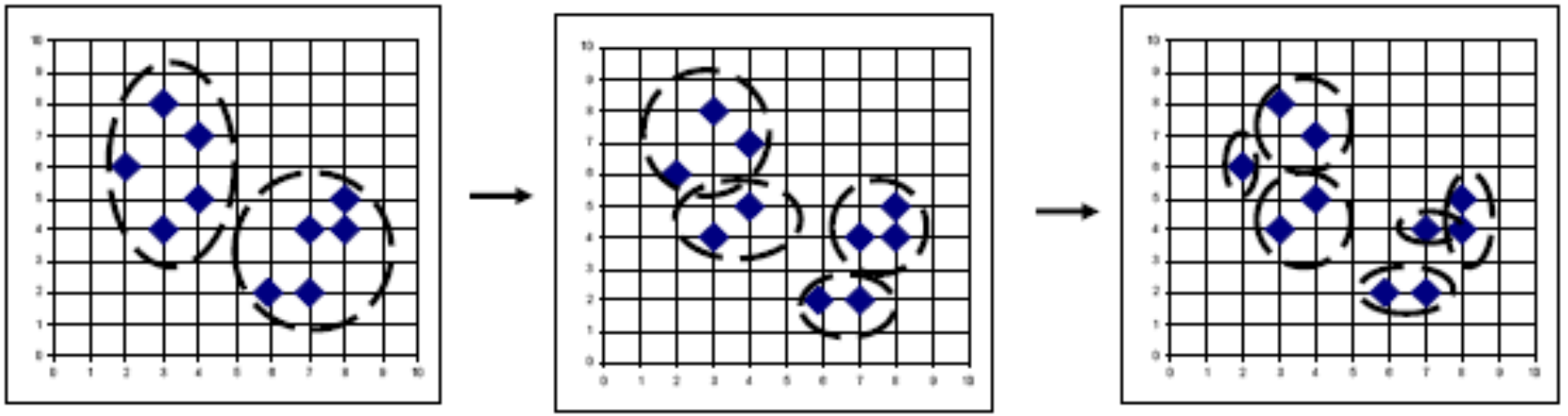
(A). Nested clusters



(B) Dendrogram

Hierarchical clustering

- Divisive (Top-down)



Bottom-up vs. Top-down

- Which one is more accurate?
  - Top-down
  - Bottom-up methods make clustering decisions based on local patterns without initially taking into account the global distribution. These early decisions cannot be undone.
  - Top-down clustering benefits from complete information about the global distribution when making top-level partitioning decisions.

Distances

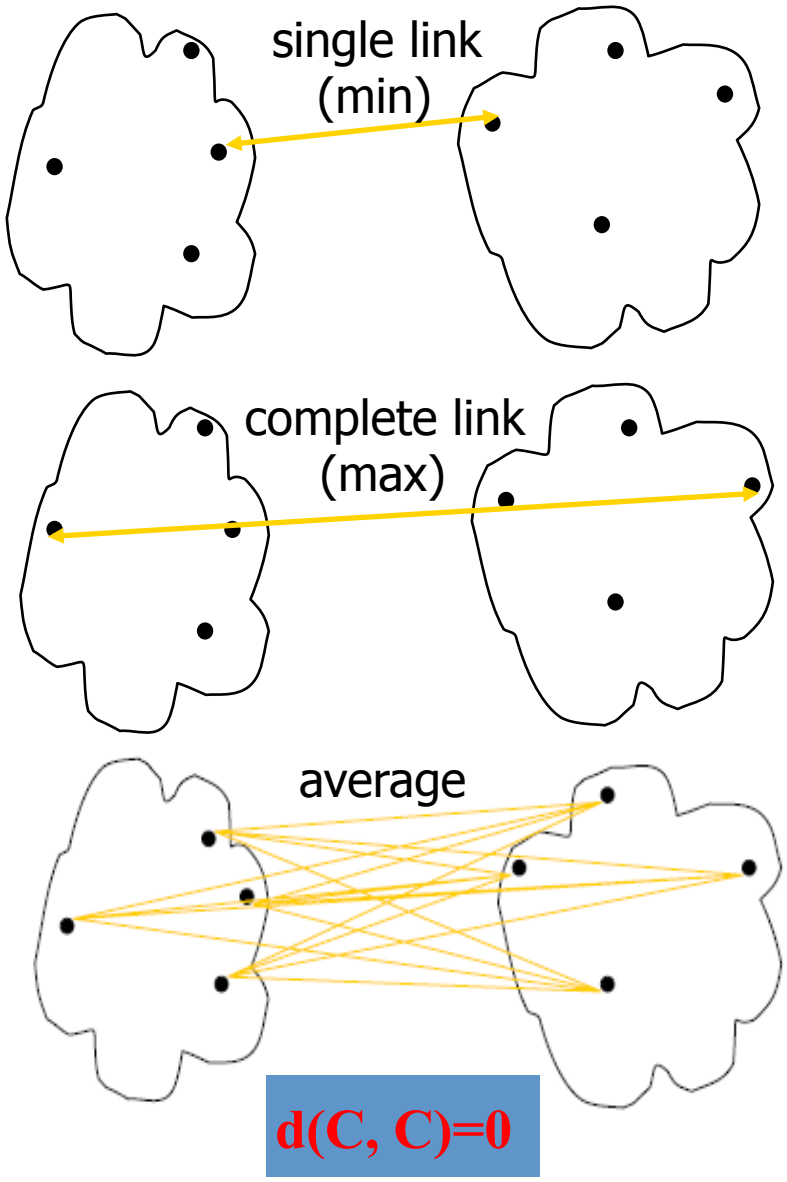
- Determine the similarity between two clusters

Names	Formula
Euclidean distance	$\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
squared Euclidean distance	$\|a-b\|_2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\|a-b\|_1 = \sum_i \|a_i - b_i\|$
maximum distance	$\|a-b\|_\infty = \max_i \|a_i - b_i\|$
Mahalanobis distance	$\sqrt{(a-b)^\top S^{-1}(a-b)}$ where S is the covariance matrix
cosine similarity	$\frac{a \cdot b}{\|a\|\|b\|}$

For text or other non-numeric data, metrics such as the Hamming distance or Levenshtein distance are often used.

Cluster Distance Measures

- Single link: smallest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \min\{d(x_{ip}, x_{jq})\}$
- Complete link: largest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \max\{d(x_{ip}, x_{jq})\}$
- Average: avg distance between elements in one cluster and elements in the other, i.e., $d(C_i, C_j) = \text{avg}\{d(x_{ip}, x_{jq})\}$



Cluster Distance Measures

Example: Given a data set of five objects characterised by a single feature, assume that there are two clusters: C_1: {a, b} and C_2: {c, d, e}.

	a	b	c	d	e
Feature	1	2	4	5	6

1. Calculate the distance matrix.

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

2. Calculate three cluster distances between C_1 and C_2.

Single link

$$\text{dist}(C_1, C_2) = \min\{d(a,c), d(a,d), d(a,e), d(b,c), d(b,d), d(b,e)\}$$
$$= \min\{3, 4, 5, 2, 3, 4\} = 2$$

Complete link

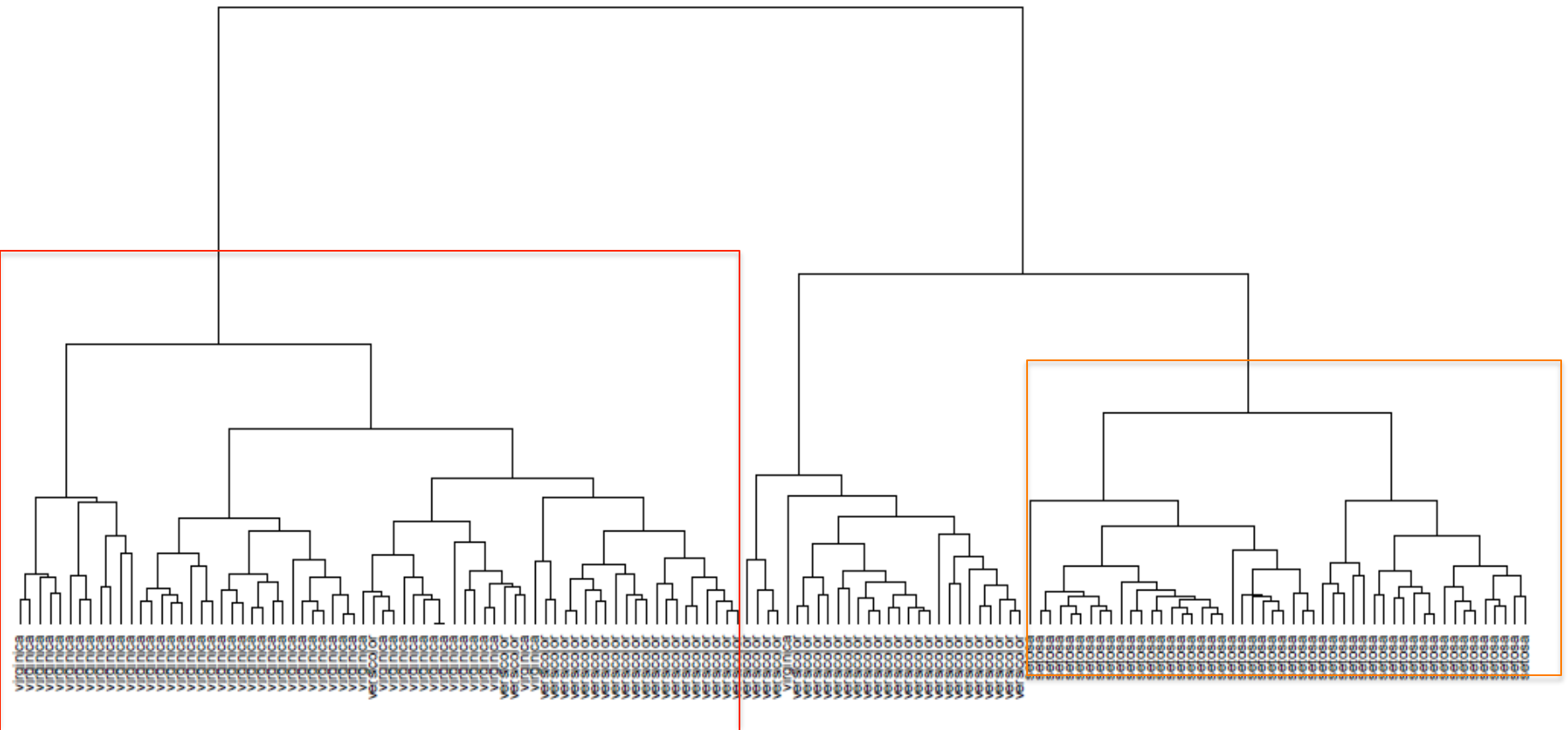
$$\text{dist}(C_1, C_2) = \max\{d(a,c), d(a,d), d(a,e), d(b,c), d(b,d), d(b,e)\}$$
$$= \max\{3, 4, 5, 2, 3, 4\} = 5$$

Average

$$\text{dist}(C_1, C_2) = \frac{d(a,c) + d(a,d) + d(a,e) + d(b,c) + d(b,d) + d(b,e)}{6}$$
$$= \frac{3+4+5+2+3+4}{6} = \frac{21}{6} = 3.5$$

Hierarchical clustering

```
> iris.feature = as.matrix(iris[,1:4])
> rownames (iris.feature) = iris[,5]
# similarity is measured using Euclidean distance determined by group average linkage
> hc <- hclust(dist(iris.feature), "ave")
> plot(hc, hang = -1)
```



Hierarchical clustering

>install.packages (‘gplots’)

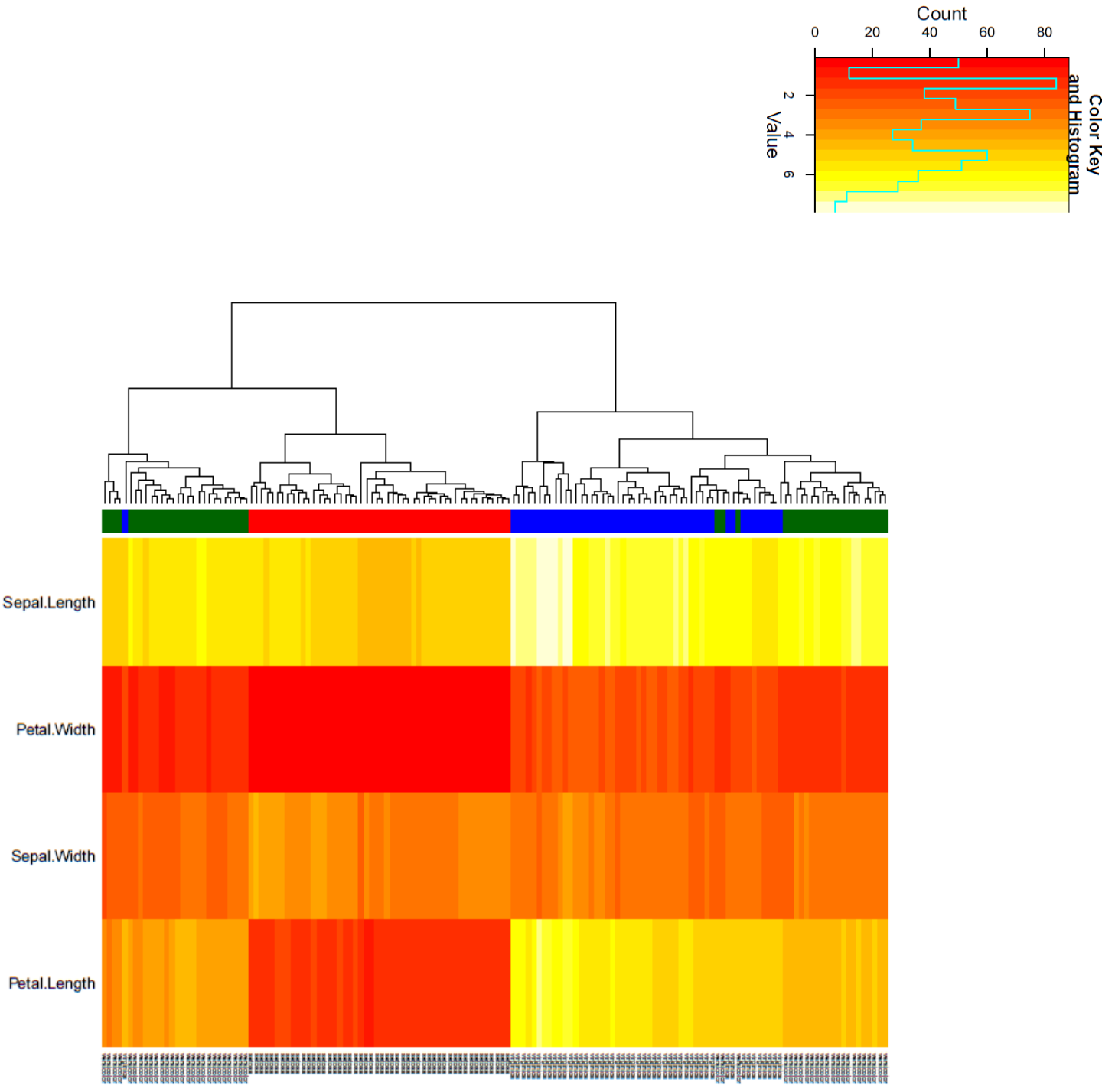
>library (gplots)

>heatmap.2(iris.feature, dendrogram="row", trace = 'none', key = 1)

>heatmap.2(iris.feature, dendrogram="row", trace = 'none', key = 1, cexCol = 1.2, margin = c(8, 5), cexRow = 0.6)

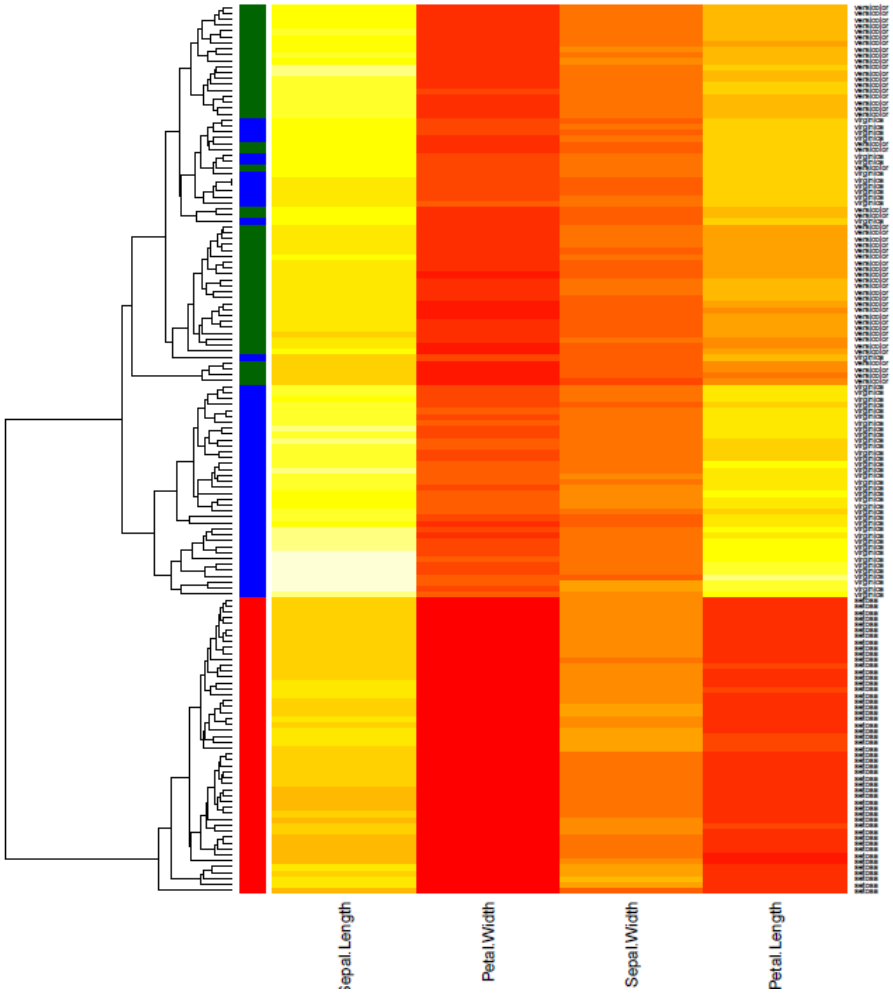
> species.color = c('red', 'darkgreen', 'blue')[unclass(iris$Species)]

> heatmap.2(iris.feature, dendrogram="row", trace = 'none', key = 1, cexCol = 1.2, margin = c(8, 5), cexRow = 0.6, RowSideColors = species.color)


Count
0
20
40
60
80
Value
2
4
6
Color Key
and Histogram
Sepal.Length
Petal.Width
Sepal.Width
Petal.Length

Use different approach to measure cluster distance

> heatmap.2(iris.feature, trace = 'none', dendrogram="row", RowSideColors = species.color, distfun=function (y) dist(y,method = "euclidean"), hclustfun = function(y) hclust(y, ‘ave'), key = 1, margin = c(8, 5), cexCol = 1.3)



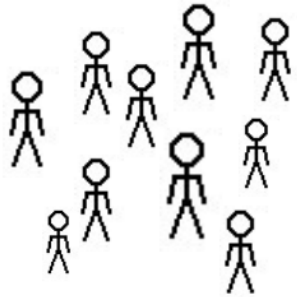
Generate a pdf figure in R

```
>pdf ('myfigure.pdf', height = 18, width = 10, family = 'Helvetica')
> You code
.
.
>dev.off()
```

Linear Regression

Linear Regression

Regression



INPUT: Weight
OUTPUT: Height

Regression →

Estimated function: $\hat{f}$

Weight → $\hat{f}$ → Height

DataCamp

Linear regression

"An approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called simple **linear regression**." -- Wikipedia

Simple linear regression uses a straight line to fit the data.

lm() in R

```
> setosa = iris[iris$Species == "setosa",]
>train = setosa[-1,]
>test = setosa[1,]

>x = train$Sepal.Width  #independent variable
>y = train$Sepal.Length  #dependent variable

>lm.Sepal <- lm(y ~ x)  #formula
>summary(lm.Sepal)
```

lm() in R

```
> summary(lm.Sepal)

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.52378 -0.16179  0.02123  0.14522  0.44522

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.63968    0.31320   8.428 5.89e-11 ***
x            0.69003    0.09085   7.595 1.03e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.241 on 47 degrees of freedom
Multiple R-squared:  0.551,     Adjusted R-squared:  0.5415
F-statistic: 57.69 on 1 and 47 DF,  p-value: 1.03e-09
```

In this case, the regression line is: $y = 2.64 + 0.69 * x$

Predict the length

```
> test
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
> pred = data.frame(x = test$Sepal.Width)
> predict(lm.Sepal, pred, interval = "prediction",level = 0.95)
       fit      lwr      upr
1 5.054778 4.564886 5.544669
>
```

- The width of the test variable is 3.5.
- The predicted length of this variable is 5.054778
- And the 95% confident interval (4.56 ~ 5.54)

Support Vector Machine

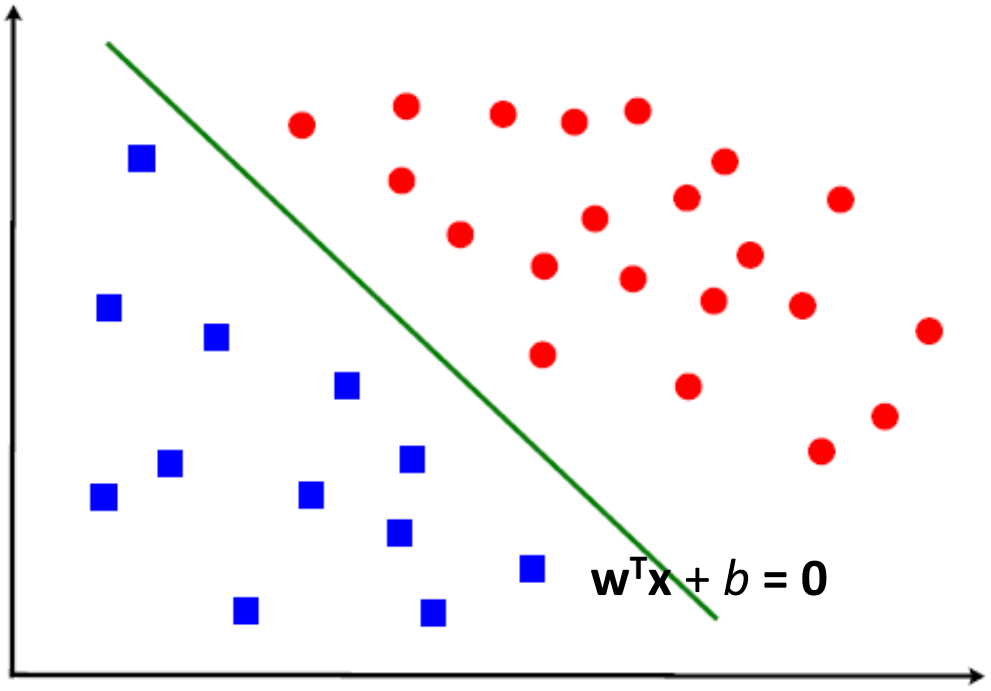
SVM: the main idea

- Find an optimal hyperplane to separate data points, which belong to different classes.
  - Maximize the distance (from closest points) of each class to the separating hyperplane
  - Minimizing the risk of misclassification

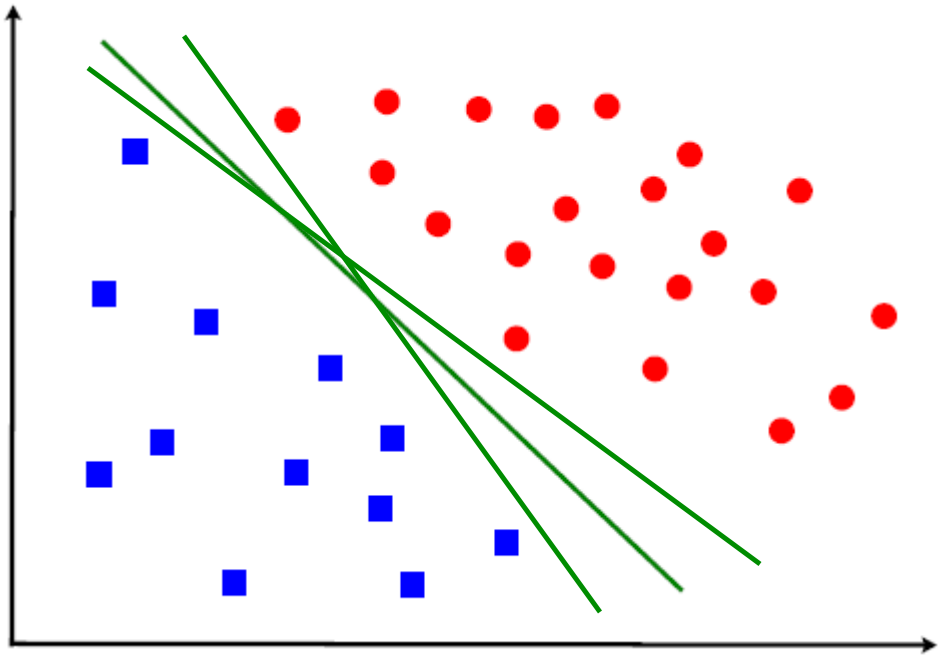
- Strategy: Formulate a constraint-based optimization problem, then solve it using quadratic programming.

Linear classifier

$$Y = \text{sgn}[\mathbf{w}^\top \mathbf{x} + b > 0] = \begin{cases} +1 & \mathbf{w}^\top \mathbf{x} + b > 0 \\ -1 & \mathbf{w}^\top \mathbf{x} + b \leq 0 \end{cases}$$

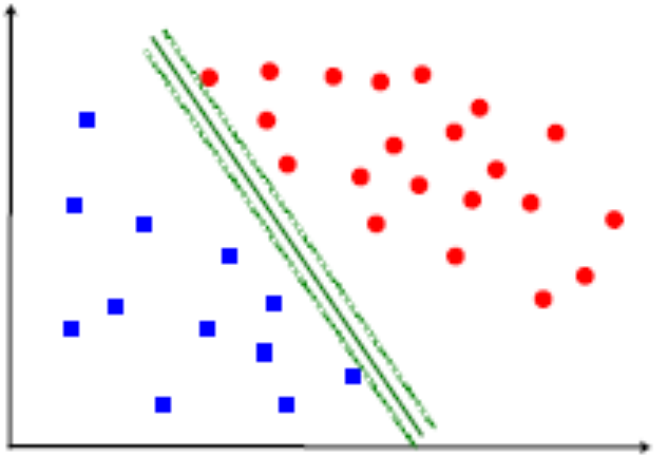


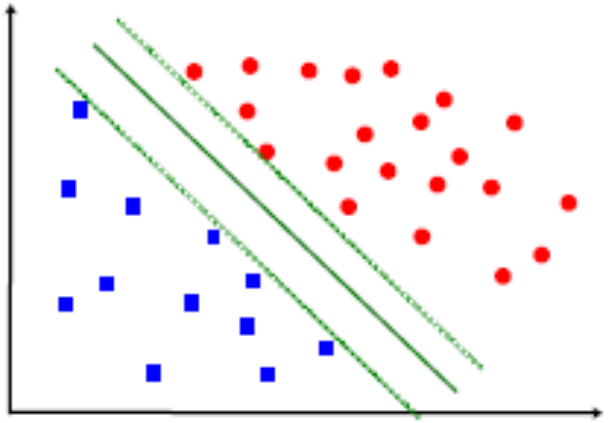
Which linear classifier is the optimal?



Linear classifier

- Margin
  - The distance from a hyperplane to its closest point.
  - Select a hyperplane that can maximize the margin.
    - The decision hyperplane should be as far away from the data of both classes as possible.





Linear SVM

- SVMs maximize the margin around the separating hyperplane.

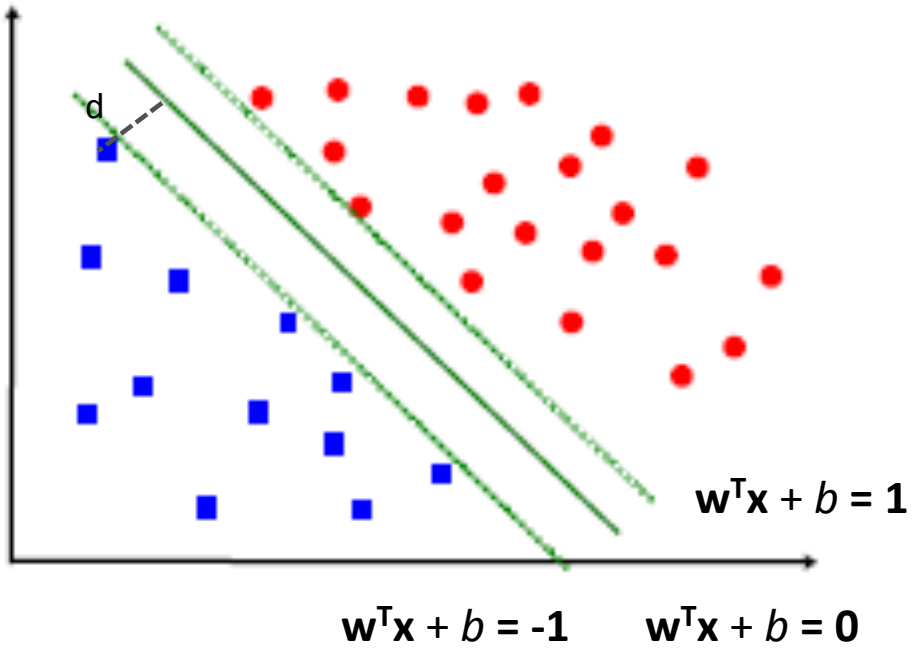
$$d = \frac{|w^T(x - x_i)|}{\|w\|}$$

$$= \frac{|w^T x + b - (w^T x_i + b)|}{\|w\|}$$

$$= \frac{1}{\|w\|}$$

- Then

$$Margin = 2d = \frac{2}{\|w\|}$$



A constrained optimization problem

- Find w and b:
  - Maximize $\frac{2}{\| w \|}$

    Subject to $\min_{i=1,2,..N} | w^T x_i + b | = 1$

  - Equivalent to minimize $\frac{1}{2} w^T w$

    Subject to $y_i (w^T x_i + b) \geq 1$ where i = 1,2, …, N

SVM: quadratic optimization

- Lagrange formulation

$$L(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i (y_i (w^T x_i + b) - 1)$$

$$\nabla_w L = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \Rightarrow \boxed{w = \sum_{i=1}^{N} \alpha_i y_i x_i}$$

$$\frac{\partial L}{\partial b} = \boxed{-\sum_{i=1}^{N} \alpha_i y_i = 0}$$

- Then $L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j$

subject to $\sum_{i=1}^{N} \alpha_i y_i = 0 \qquad \alpha_i \geq 0$

SVM: quadratic optimization

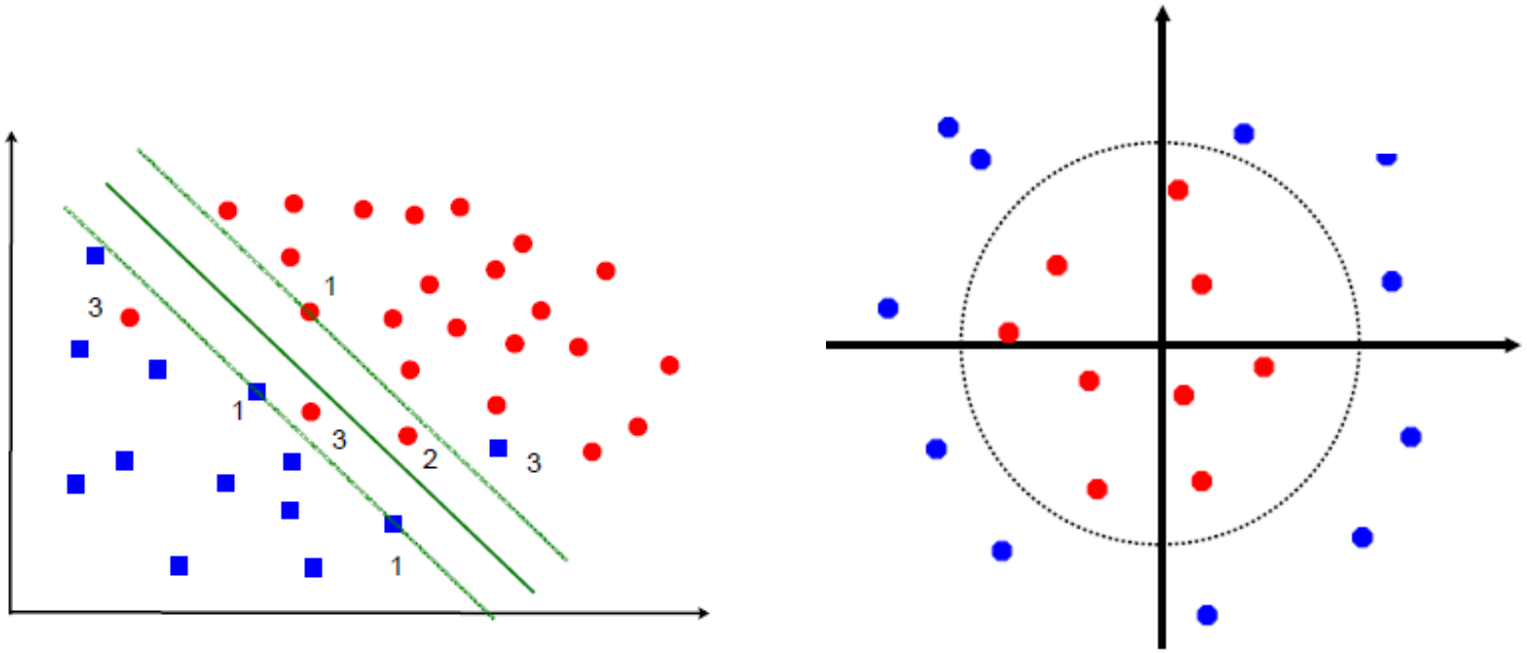
- Then

$$w = \sum_{x_i \in S.V} \alpha_i y_i x_i$$

- Find b using any support vector

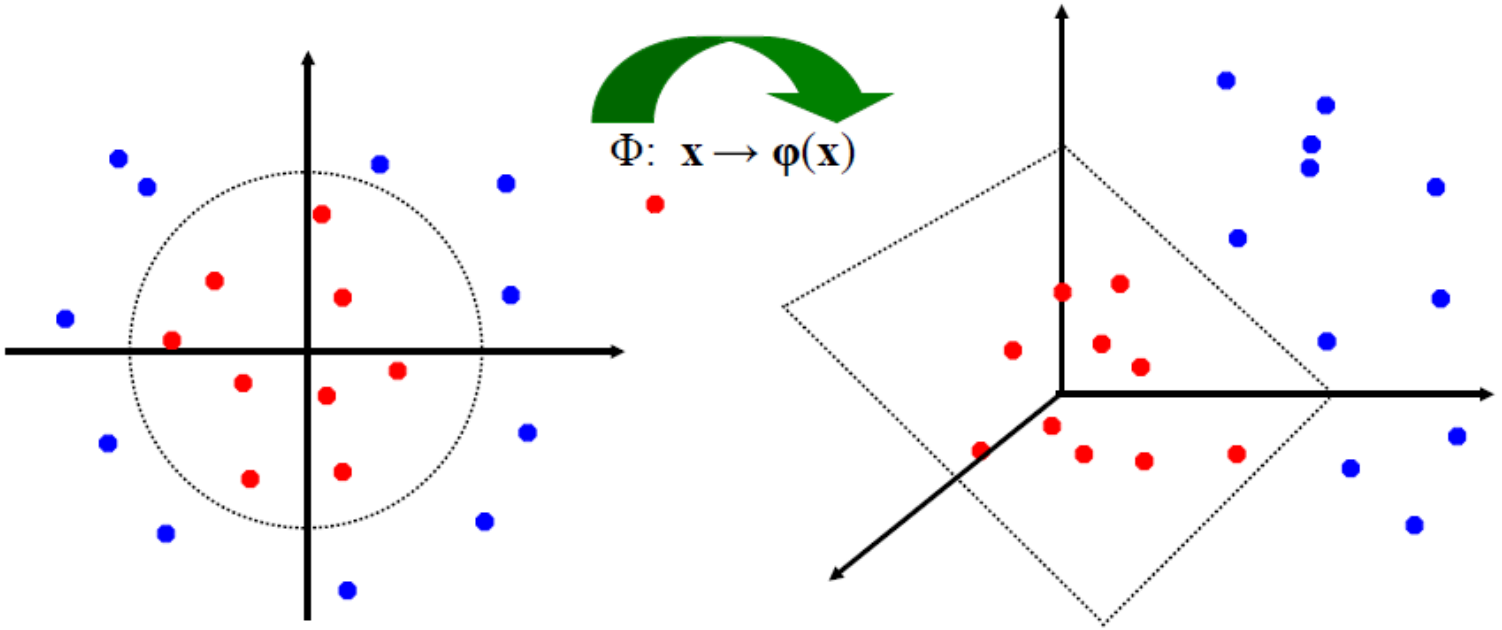
$$y_i (w^T x_i + b) = 1 \qquad x_i \in S.V.$$

Nonlinear separated cases



Nonlinear separated case

Map training data into high dimensional space



Kernel functions

- Why use kernels?
  - Map data into better representation space.
  - Make non-separable case to be separable.
- Common Kernel functions
  - Linear $K(x_i, x_j) = x_i^T x_j$
  - Polynomial $K(x_i, x_j) = (1 + x_i^T x_j)^p$
  - Gaussian $K(x_i, x_j) = e^{\frac{||x_i - x_j||}{2\sigma^2}}$

Non-linear SVMs

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i^T x_j)$$

subject to $\sum_{i=1}^{N} \alpha_i y_i = 0 \qquad \alpha_i \geq 0$

- Build a SVM-based classifier using R
- Classification performance evaluation

Data

- Benign and malignant breast tumor samples
  - 486 training instances
    - 316 benign, 197 malignant
  - 197 testing instances
    - 128 benign, 69 malignant
- Nine numeric attributes
  - Clump Thickness
  - Uniformity of Cell Size
  - Uniformity of Cell Shape
  - Marginal Adhesion
  - Single Epithelial Cell Size
  - Bare Nuclei
  - Bland Chromatin
  - Normal Nucleoli
  - Mitoses

SVMs for classification

- Install a Bioconductor package *e1071*

```
> source (“http://bioconductor.org/biocLite.R")
> biocLite("e1071")
> library (e1071)
```

Load data and build a classifier

```
> train = read.table ('breast-cancer_svm_train.txt', sep = '\t', header = T)
> test = read.table ('breast-cancer_svm_test.txt', sep = '\t', header = T)

> dim(train)
  [1] 486  10
> colnames(train)
[1] "Thickness"      "Unif.Size"      "Unif.Shape"     "Adhesion"       "Epi.Size"       "Bare.Nucle"
 [7] "Bland.Chromatin" "Normal.Nucleoli" "Mitoses"        "Class“

  #build a  computational model using SVM
 > model <- svm(train[, -10], train[, 10], kernel = "linear",scale = TRUE, type =
     "C-classification")
```

A SVM classifier

```
> summary (model)

Call:
svm.default(x = train[, -10], y = train[, 10], scale = TRUE, type =
   "C-classification", kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.1111111

Number of Support Vectors:  31

 ( 16 15 )


Number of Classes:  2

Levels:
 Benign Malignant
```

Cross-validation

```
> model <- svm(train[, -10], train[, 10], kernel = "linear",scale = TRUE, type = "C-
      classification", cross = 10)
> summary (model)

Call:
svm.default(x = train[, -10], y = train[, 10], scale = TRUE, type = "C-classification", kernel = "linear", cross = 10)


Parameters:
   SVM-Type:  C-classification
  SVM-Kernel:  linear
  cost:  1
   gamma:  0.1111111

Number of Support Vectors:  31

 ( 16 15 )


Number of Classes:  2

Levels:
 Benign Malignant


10-fold cross-validation on training data:


Total Accuracy: 97.3251
Single Accuracies:
 100 97.95918 100 97.95918 95.91837 97.91667 100 97.91667 95.91837 89.79592
```

- Build a SVM-based classifier using R
- Classification performance evaluation

Classifier performance evaluation

	Actual negative	Actual positive
Predict negative	TN	FN
Predict positive	FP	TP

$$Sensitivity = \frac{TP}{TP + FN}$$

The ability to detect positive instances

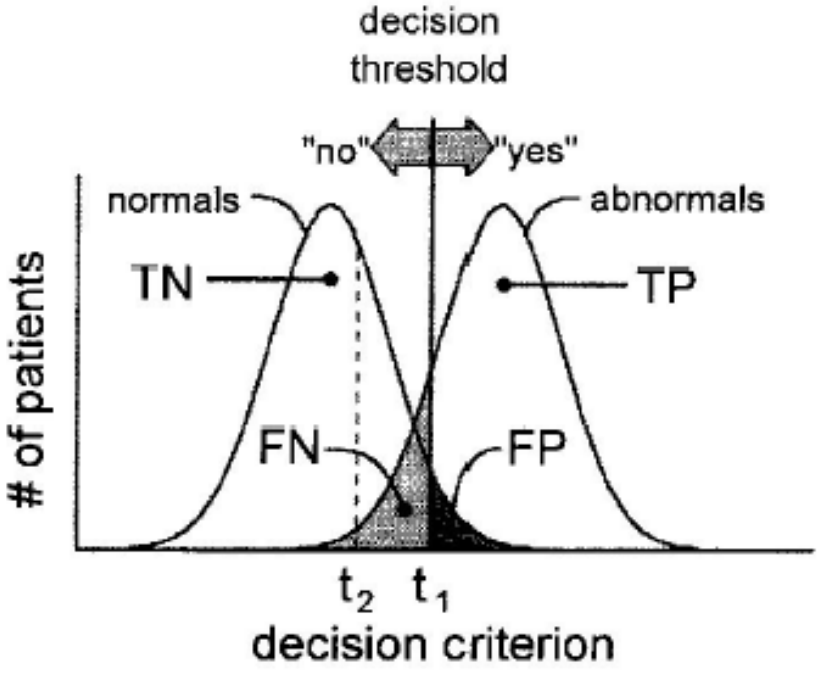
$$Specificity = \frac{TN}{TN + FP}$$

The ability to reject positive instances

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

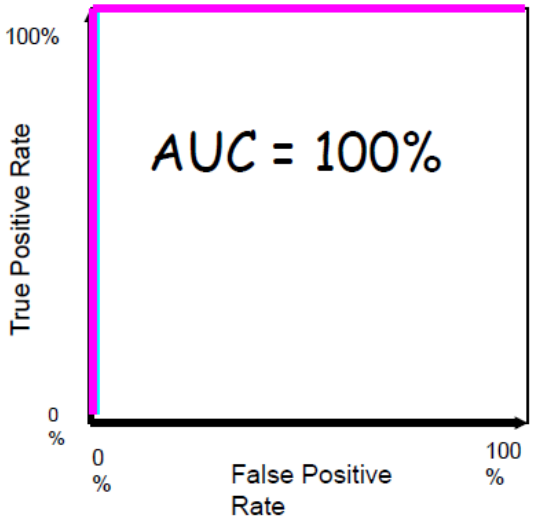
Receiver operating characteristic (ROC) curve

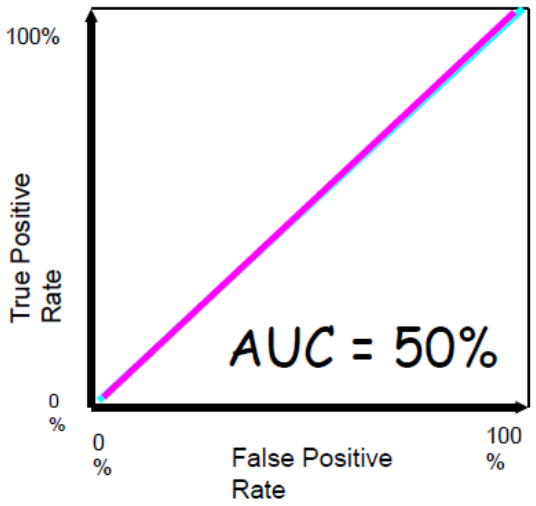
- First used in electronic signal detection theory (1940s - 1950s)
- Commonly used in machine learning to evaluate classifiers

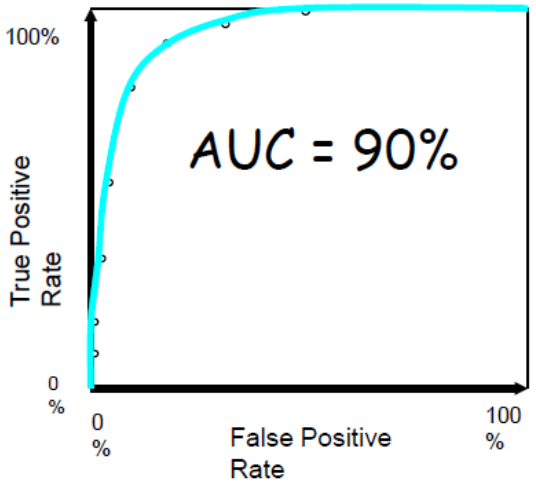


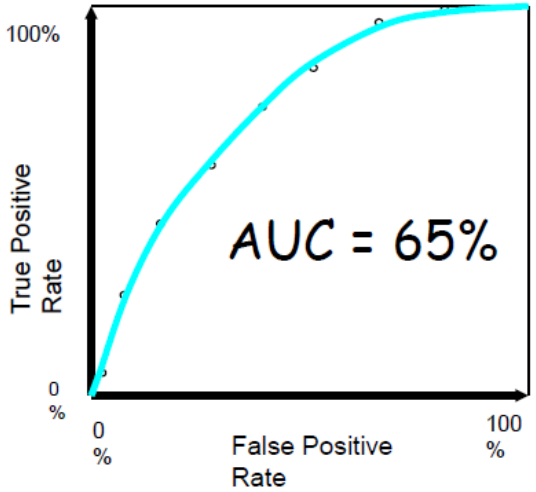
Area under ROC curve (AUC)

- Overall measurement of the performance of a classifier.









Prediction

```
> preds <- predict(model, test[, -10])
> acc <- sum(preds == test[,10])/nrow (test)
> acc = signif(acc, 4)
> acc
[1] 0.9543
> cat("Testing accuracy: ", acc * 100, "%\n", sep =
  "")
 Testing accuracy: 95.43%
```

```
> table(predict = preds, true = test[,10])
            true
predict     Benign Malignant
  Benign       124         5
  Malignant      4        64
```

Benign: negative

Malignant: positive

	Actual negative	Actual positive
Predict negative	TN	FN
Predict positive	FP	TP

Classifier performance evaluation

	Actual negative	Actual positive
Predict negative	TN	FN
Predict positive	FP	TP

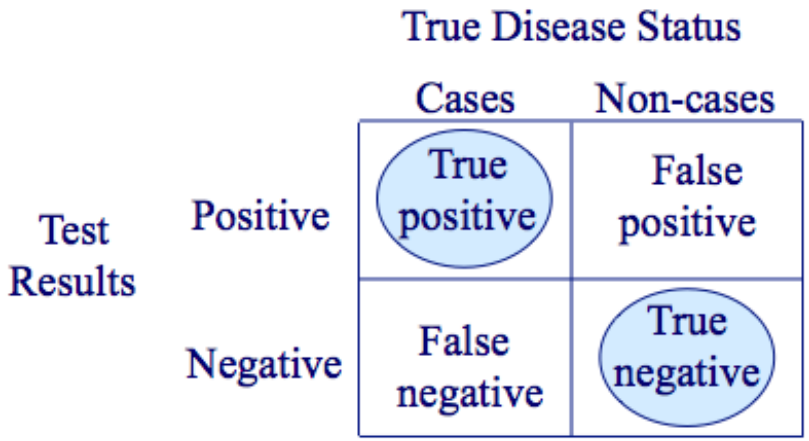
$$Sensitivity = \frac{TP}{TP + FN}$$

The ability to detect positive instances

$$Specificity = \frac{TN}{TN + FP}$$

The ability to reject positive instances

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$



Calculate weight vector w

- The weight vector **w**

$$w = \sum_{x_i \in S.V} \alpha_i y_i x_i$$

```
> w = t(model$coefs) %*% model$SV
> w
     Thickness  Unif.Size   Unif.Shape  Adhesion  Epi.Size  Bare.Nucle Bland.Chromatin  Normal.Nucleoli   Mitoses
[1,] 0.9702354 -0.1605056   0.450314 0.6865762   0.0408258  0.9463005       0.6364409        0.267061 0.6050037

> order(abs(w), decreasing = TRUE)
[1] 1 6 4 7 9 3 8 2 5
```

Weight of features

Use all 9 features

```
> table(predict = preds, true = test[,10])
            true
predict     Benign Malignant
  Benign       124         5
  Malignant      4        64
```

Use top 7 features

```
> model.2 <- svm(train[, -c(2, 5,10)], train[, 10], kernel =
   "linear",scale = TRUE, type = "C-classification")
> new_test = test[,-c(2,5, 10)]
> preds.2 <- predict(model.2, new_test)
> table(predict = preds.2, true = test[,10])
            true
predict     Benign Malignant
  Benign       124         5
  Malignant      4        64
```

Weight of features

Use all 9 features

```
> table(predict = preds, true = test[,10])
```

predict	true Benign	Malignant
Benign	124	5
Malignant	4	64

Exclude top two features

```
> model.3 <- svm(train[, -c(1, 6,10)], train[, 10], kernel =
   "linear",scale = TRUE, type = "C-classification")
> new_test = test[,-c(1,6, 10)]
> preds.3 <- predict(model.3, new_test)
> table(predict = preds.3, true = test[,10])
```

predict	true Benign	Malignant
Benign	124	14
Malignant	4	55

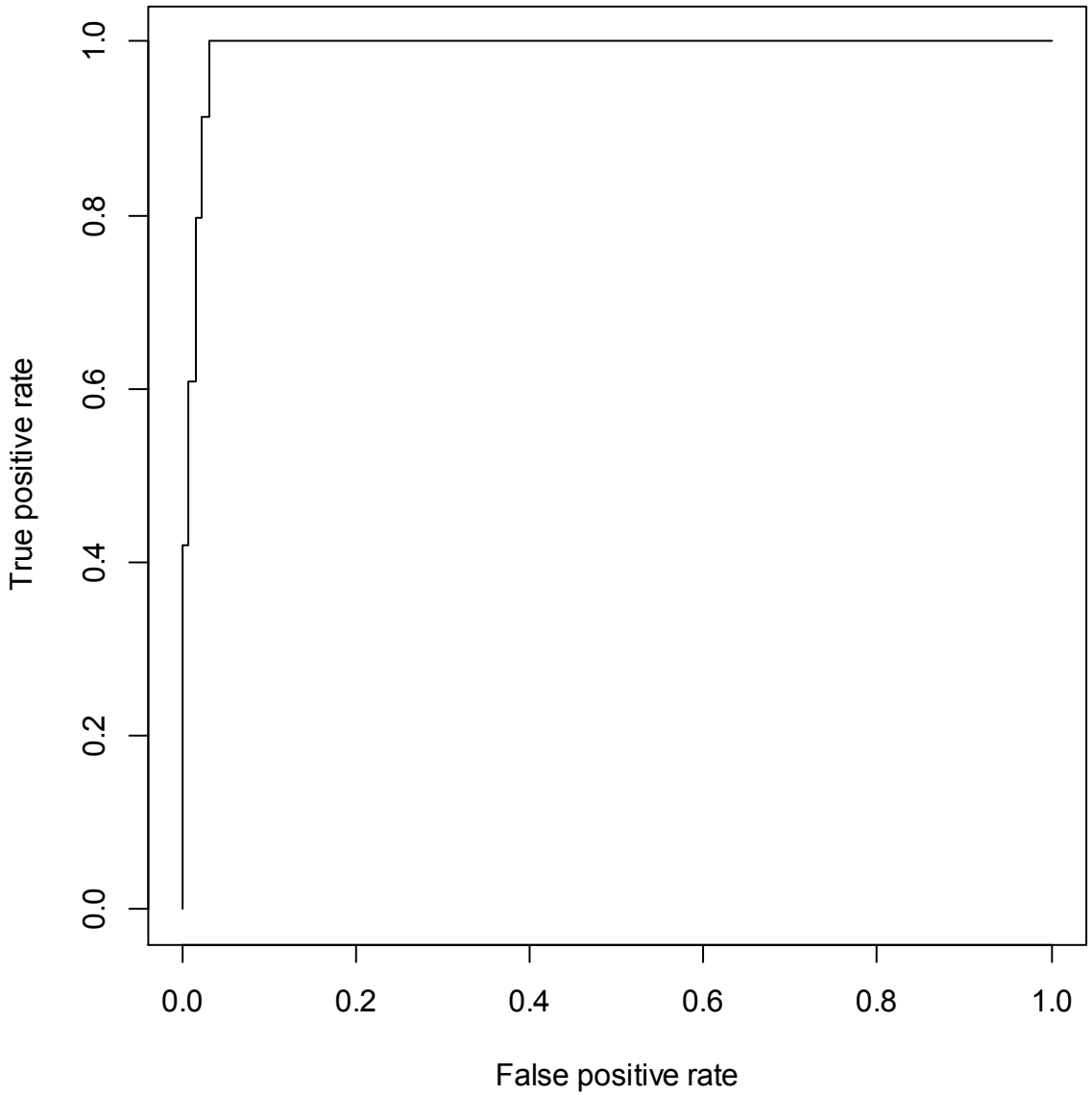
Plot ROC

```
>install.packages ('ROCR')
>library (ROCR)

>model <- svm(train[, -10], train[, 10], kernel = "linear",scale =
    TRUE, type = "C-classification", decision.values = TRUE)
> preds <- predict(model, test[, -10], decision.values = TRUE)

> svm.roc = prediction(attributes(preds)$decision.values, test[,
    10])
```

- `svm.auc <- performance(svm.roc, 'tpr', 'fpr')`
-

```
> plot(svm.auc)
```

ROC


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate