Introduction to Machine Learning CANB 7640

Aik Choon Tan, Ph.D.
Associate Professor of Bioinformatics
Division of Medical Oncology
Department of Medicine
aikchoon.tan@ucdenver.edu
9/10/2018
http://tanlab.ucdenver.edu/labHomePage/teaching/CANB7640/

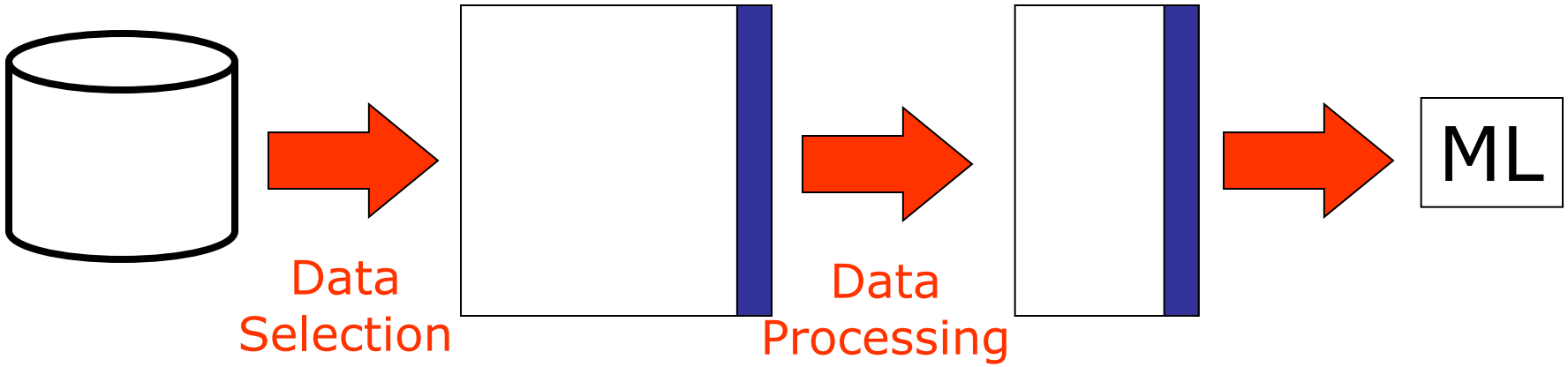
Outline

- Introduction
- Data, features, classifiers, Inductive learning
- (*Selected*) Machine Learning Approaches
  - Decision trees
  - Naïve Bayes
  - Support Vector Machines
  - Clustering
- Model evaluation

Steps in Class prediction problem

- Data Preparation
- Feature selection
  - Remove irrelevant features for constructing the classifier (but may have biological meaning)
  - Reduce search space in H, hence increase speed in generating classifier
  - Direct the learning algorithm to focus on “informative” features
  - Provide better understanding of the underlying process that generated the data
- Selecting a machine learning method
- Generating classifier from the training data
- Measuring the performance of the classifier
- Applying the classifier to unseen data (test)
- *Interpreting the classifier*

Data Preparation



"Big Data"

Focussed
Data

Processed
Data

Data: Samples and Features

Samples →

features ↓

Features	Sample 1	Sample 2	…	Sample *m*
feature 1	Feature_ value	Feature_ value	…	Feature_ value
feature 2	Feature_ value	Feature_ value	…	Feature_ value
…	…	…	…	…
feature *n*	Feature_ value	Feature_ value	…	Feature_ value

Cancer Classification Problem

REPORTS

Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring

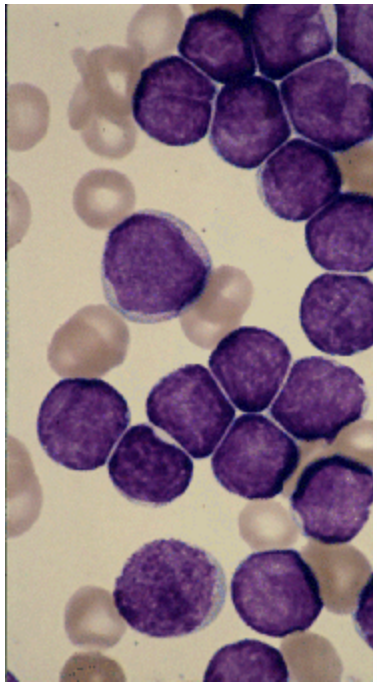
T. R. Golub,[1,2*†] D. K. Slonim,[1†] P. Tamayo,[1] C. Huard,[1] M. Gaasenbeek,[1] J. P. Mesirov,[1] H. Coller,[1] M. L. Loh,[2] J. R. Downing,[3] M. A. Caligiuri,[4] C. D. Bloomfield,[4] E. S. Lander[1,5*]

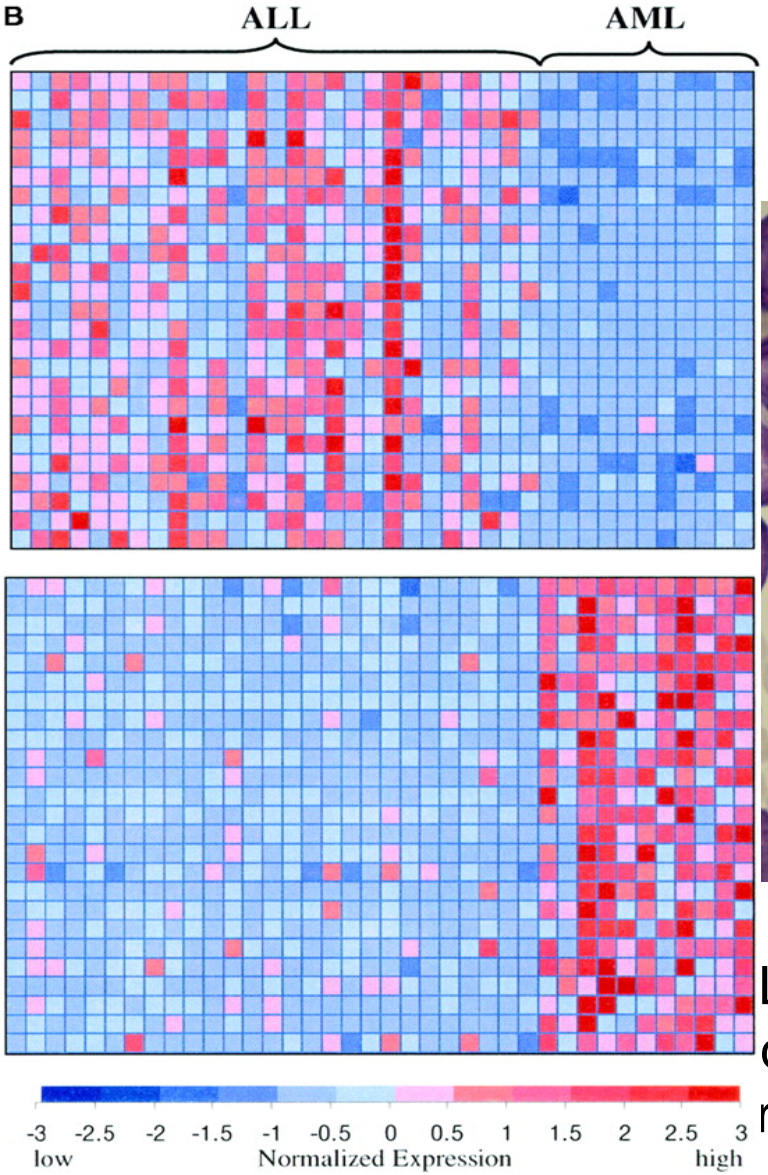
(Science 286:531-537, 1999)

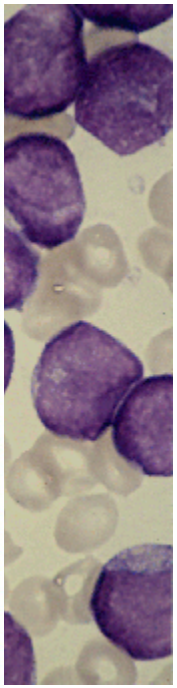
Although cancer classification has improved over the past 30 years, there has been no general approach for identifying new cancer classes (class discovery) or for assigning tumors to known classes (class prediction). Here, a generic approach to cancer classification based on gene expression monitoring by DNA microarrays is described and applied to human acute leukemias as a test case. A class discovery procedure automatically discovered the distinction between acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL) without previous knowledge of these classes. An automatically derived class predictor was able to determine the class of new leukemia cases. The results demonstrate the feasibility of cancer classification based solely on gene expression monitoring and suggest a general strategy for discovering and predicting cancer classes for other types of cancer, independent of previous biological knowledge.

Cancer Classification Problem

(Golub et al 1999)







ALL
acute lymphoblastic
(lymphoid precu

L
d leukemia
recursor)

Cancer Classification Problem

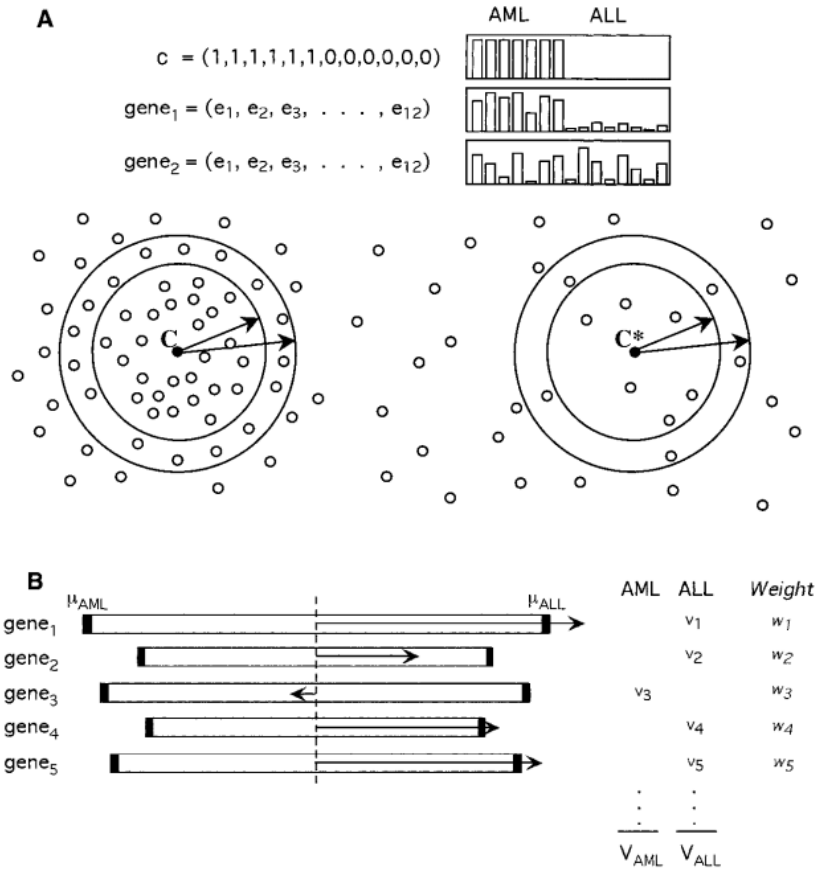


Fig. 1. Schematic illustration of methodology. (**A**) Neighborhood analysis. The class distinction is represented by an "idealized expression pattern" *c*, in which the expression level is uniformly high in class 1 and uniformly low in class 2. Each gene is represented by an expression vector, consisting of its expression level in each of the tumor samples. In the figure, the data set is composed of six AMLs and six ALLs. Gene g_1 is well correlated with the class distinction, whereas g_2 is poorly correlated. Neighborhood analysis involves counting the number of genes having various levels of correlation with *c*. The results are compared to the corresponding distribution obtained for random idealized expression patterns *c**, obtained by randomly permuting the coordinates of *c*. An unusually high density of genes indicates that there are many more genes correlated with the pattern than expected by chance. The precise measure of distance and other methodological details are described in (*16*, *17*) and on our Web site (www.genome.wi.mit.edu/MPR). (**B**) Class predictor. The prediction of a new sample is based on "weighted votes" of a set of informative genes. Each such gene g_i votes for either AML or ALL, depending on whether its expression level x_i in the sample is closer to μ_{AML} or μ_{ALL} (which denote, respectively, the mean expression levels of AML and ALL in a set of reference samples). The magnitude of the vote is $w_i v_i$, where w_i is a weighting factor that reflects how well the gene is correlated with the class distinction and $v_i = |x_i - (\mu_{AML} + \mu_{ALL})/2|$ reflects the deviation of the expression level in the sample from the average of μ_{AML} and μ_{ALL}. The votes for each class are summed to obtain total votes V_{AML} and V_{ALL}. The sample is assigned to the class with the higher vote total, provided that the prediction strength exceeds a predetermined threshold. The prediction strength reflects the margin of victory and is defined as $(V_{win} - V_{lose})/(V_{win} + V_{lose})$, where V_{win} and V_{lose} are the respective vote totals for the winning and losing classes. Methodological details are described in (*19*, *20*) and on the Web site.

(Science 286:531-537, 1999)

Gene Expression Profile

m samples

n genes

Geneid	Condition 1	Condition 2	…	Condition *m*
Gene1	103.02	58.79	…	101.54
Gene2	40.55	1246.87	…	1432.12
…	…	…	…	…
Gene *n*	78.13	66.25	…	823.09

A (very) Brief Introduction to Machine Learning

To Learn

“ … to acquire *knowledge* of (a subject) or skill in (an art, etc.) as a result of *study*, *experience*, or *teaching*… ”
(OED)

What is Machine Learning?

“ … a computer program that can learn from *experience* with respect to some class of *tasks* and *performance measure* … ”
(Mitchell, 1997)

Broader context

- What is learning?
  - Memorizing?
  - Prediction?

Key Steps of Learning

- Learning task
  - what is the learning task?
- Data and assumptions
  - what data is available for the learning task?
  - what can we assume about the problem?
- Representation
  - how should we represent the examples to be classified
- Method and estimation
  - what are the possible hypotheses?
  - how do we adjust our predictions based on the feedback?
- Evaluation
  - how well are we doing?
- Model selection
  - can we rethink the approach to do even better?

Learning Tasks

- Classification – Given positive and negative examples, find hypotheses that distinguish these examples. It can extends to multi-class classification.

- Characterisation – Given positive examples, find hypotheses that describe these examples.

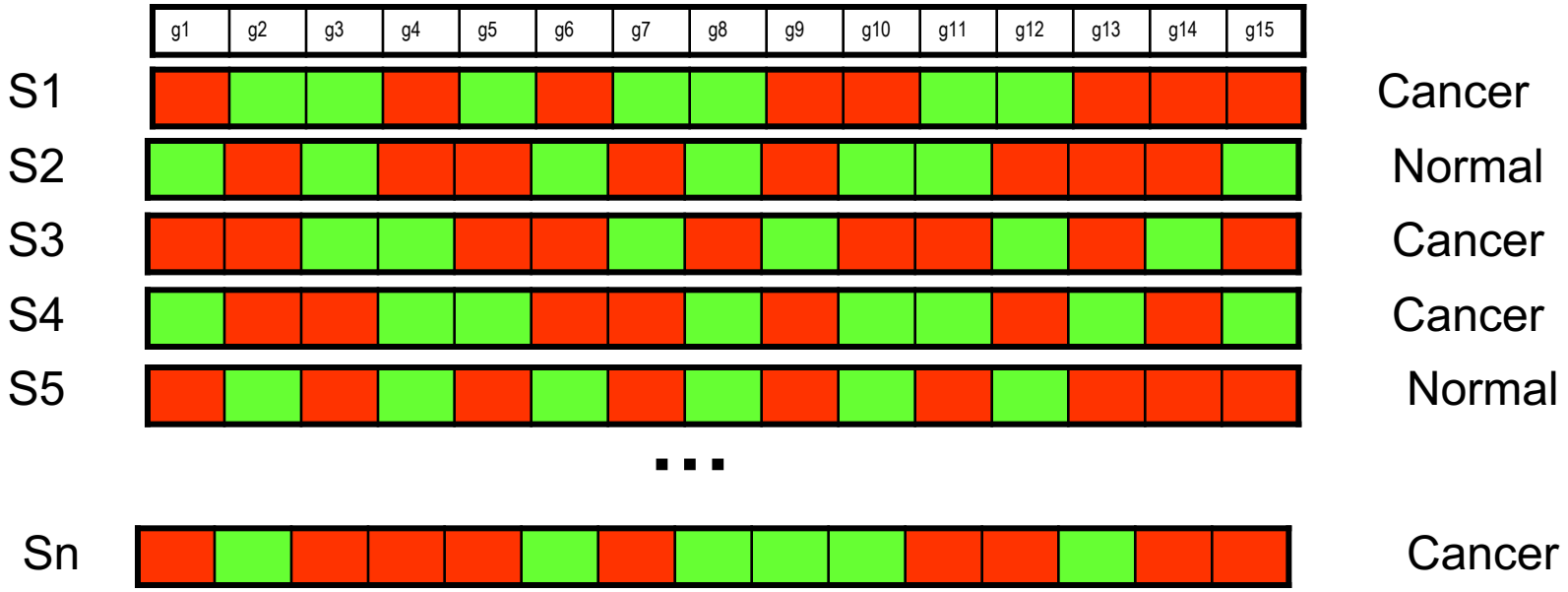
- Clustering – Given a set of unlabelled examples, find clusters for these examples (unsupervised learning)

Learning Approaches

- Supervised approach – given *predefined* class of a set of positive and negative examples, construct the classifiers that distinguish between the classes <**x**, *y*>

- Unsupervised approach – given the *unassigned* examples, group together the examples with similar properties <**x**>

Data and assumptions

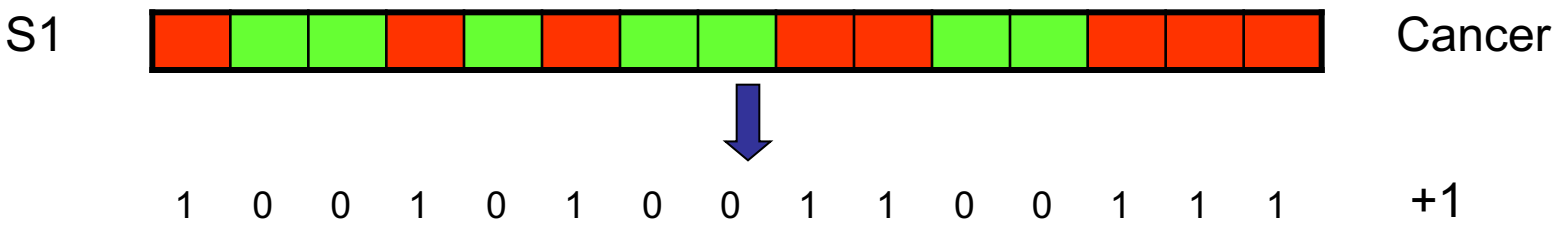


-Is this a classification problem?

-How does the data/label generated?

Representation

There are many ways to represent the same information



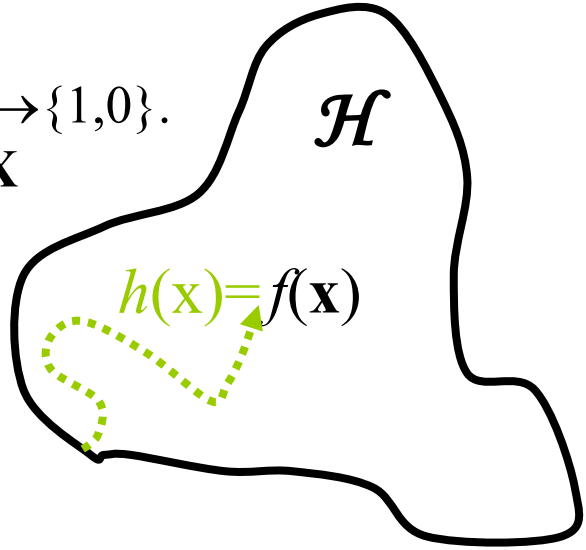
S = (**x**,y)
Where **x** $\in$ {1, 0} (red, green)
And y $\in$ {-1, +1} (Normal, Cancer)

The choice of representation may determine whether the learning task is very easy or very difficult

Concept Learning

Given a set of training examples S = {(x1,y1),…,(xm,ym)} where **x** is the instances usually in the form of tuple <x1,…,xn> and y is the class label, the function y = f(x) is unknown and finding the f(x) represent the essence of concept learning.

For a binary problem $y \in \{1,0\}$, the unknown function f:$\mathbf{X} \rightarrow \{1,0\}$.
The learning task is to find a hypothesis h(x) = f(x) for $\mathbf{x} \in \mathbf{X}$



Training examples <**x**, f(**x**)> where:
f(**x**) = 1 are Positive examples,
f(**x**) = 0 are Negative examples.

$\mathcal{H}$ is the set of all possible hypotheses, where h:$\mathbf{X} \rightarrow \{1,0\}$

A machine learning task:
Find hypothesis, $h(\mathbf{x}) = c(\mathbf{x})$; $\mathbf{x} \in \mathbf{X}$.
(in reality, usually ML task is to approximate $h(\mathbf{x}) \cong c(\mathbf{x})$)

Inductive Learning

- Given a set of observed examples
- Discover concepts from these examples
  - class formation/partition
  - formation of relations between objects
  - patterns

Learning paradigms

- Discriminative (model Pr(y|**x**))
  - only model decisions given the input examples; no model is constructed over the input examples
- Generative (model Pr(**x**|y))
  - directly build class-conditional densities over the multidimensional input examples
  - classify new examples based on the densities

Decision Trees

- Widely used - simple and practical
- Quinlan - ID3 (1986), C4.5 (1993) & See5/C5 (latest)
- Classification and Regression Tree (CART by Breiman et.al., 1984)
- Given a set of instances (with a set of properties/attributes), the learning system constructs a tree with internal *nodes* as an *attribute* and the *leaves* as the *classes*
- Supervised learning
- Symbolic learning, give interpretable results

Information Theory - Entropy

Entropy – a measurement commonly used in information theory to characterise the (im)purity of an arbitrary collection of examples

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

where *S* is a collection of training examples with *c* classes and p_i is the proportion of examples *S* belonging to class *i*.

Example:
If S is a set of examples containing positive (+) and negative (-) examples (c $\in$ {+,-}), the entropy of S relative of this boolean classification is:

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Entropy(*S*) =
- 0 if all members of *S* belong to the same class
- 1 if *S* contains an equal number of positive (+) and negative (-) examples

Note: Entropy ↓ Purity ↑*

ID3 (Induction of Decision Tree)

- Average entropy of attribute A

$$\hat{E}_A = \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

v = all the values of attribute A, S = training examples, S_v = training examples of attribute A with value v

E_A = { 0 if all members of *S* belong to the same value v

1 if *S* contains an equal number of value v examples

Note: Entropy ↓ Purity ↑*

Splitting rule of ID3 (Quinlan, 1986)

- Information Gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Note: Gain↑ Purity ↑*

Decision Tree Algorithm

Function *Decision_Tree_Learning* (*examples*, *attributes*, *target*)
Inputs: *examples* = set of training examples
attributes = set of attributes
target = class label

1. if *examples* is empty **then return** *target*
2. else if all *examples* have the same *target* **then return** *target*
3. else if *attributes* is empty then return most common value of *target* in *examples*
4. **else**
5. *Best* ←the attribute from *attributes* that best classifies *examples*
6. *Tree* ←a new decision tree with root attribute *Best*
7. **for** each value v_i of *Best* **do**
8. $examples_i$←{elements with *Best* = v_i}
9. *subtree*←*Decision_Tree_Learning* (*examples*, *attributes-best*, *target*)
10. add a branch to *Tree* with label v_i and subtree *subtree*
11. end
12. return *Tree*

Training Data

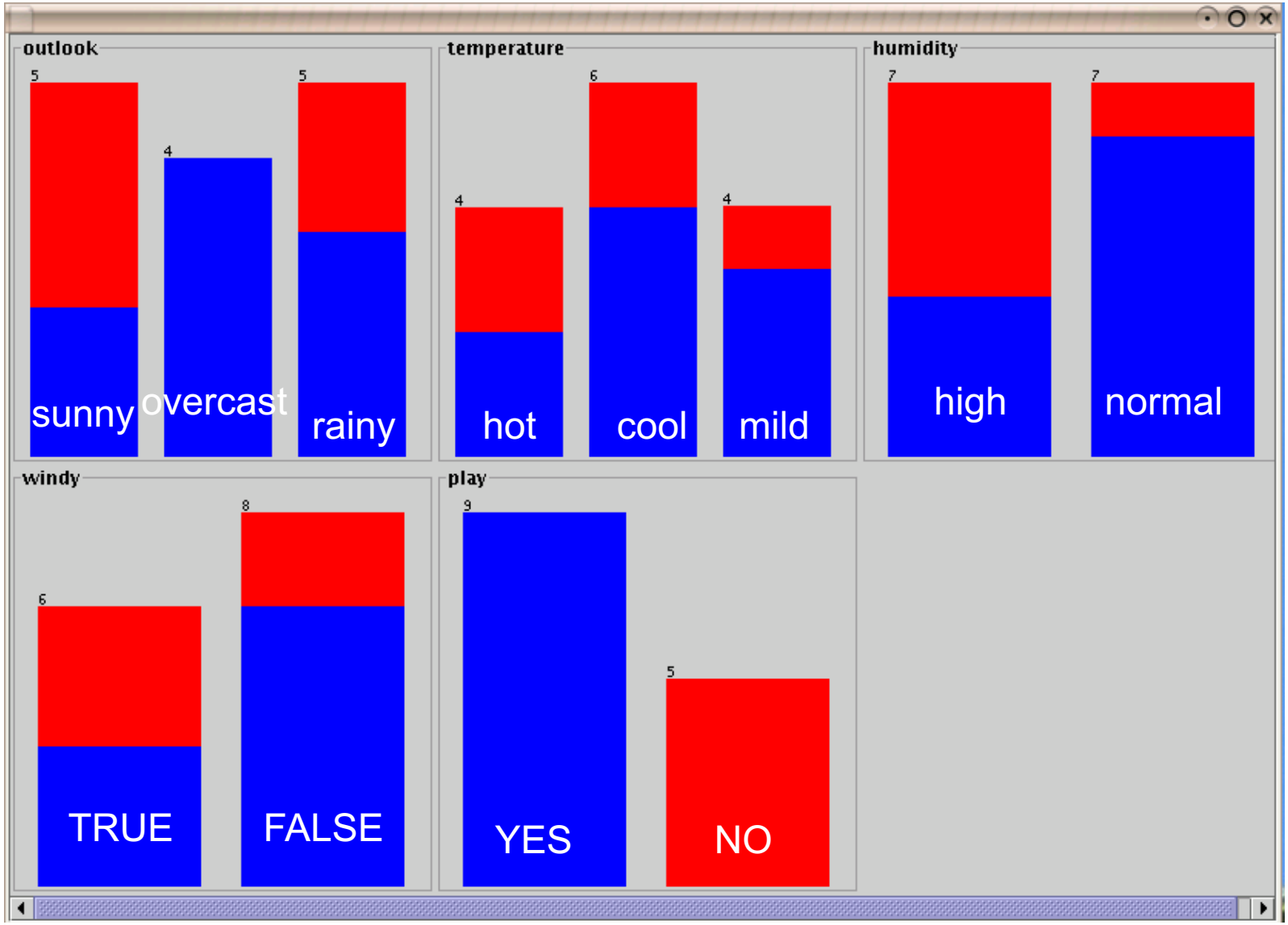
Decision attributes (dependent)

Independent condition attributes

Day	outlook	temperature	humidity	windy	play
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Today	sunny	cool	high	TRUE	?

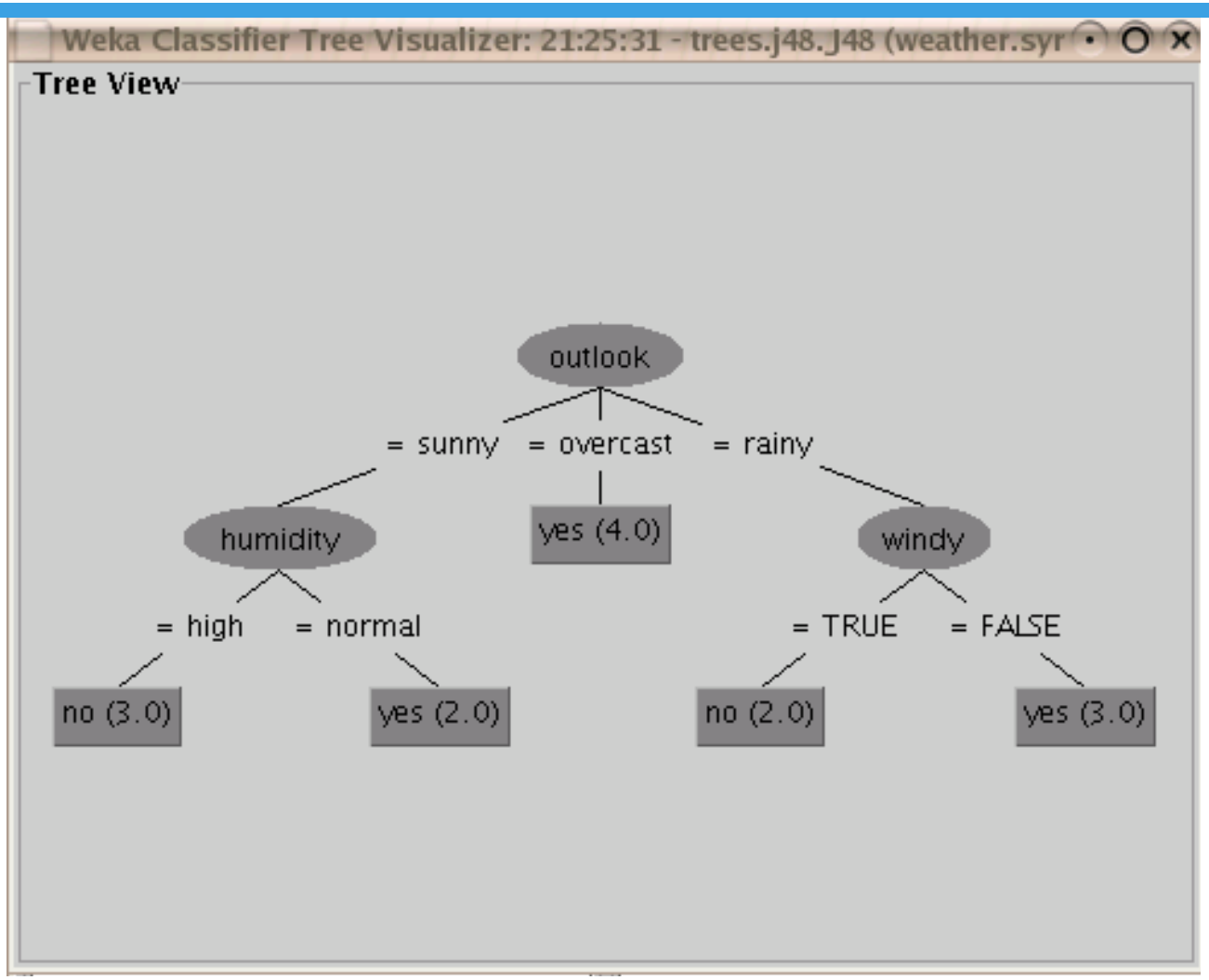
Entropy(S)



YES

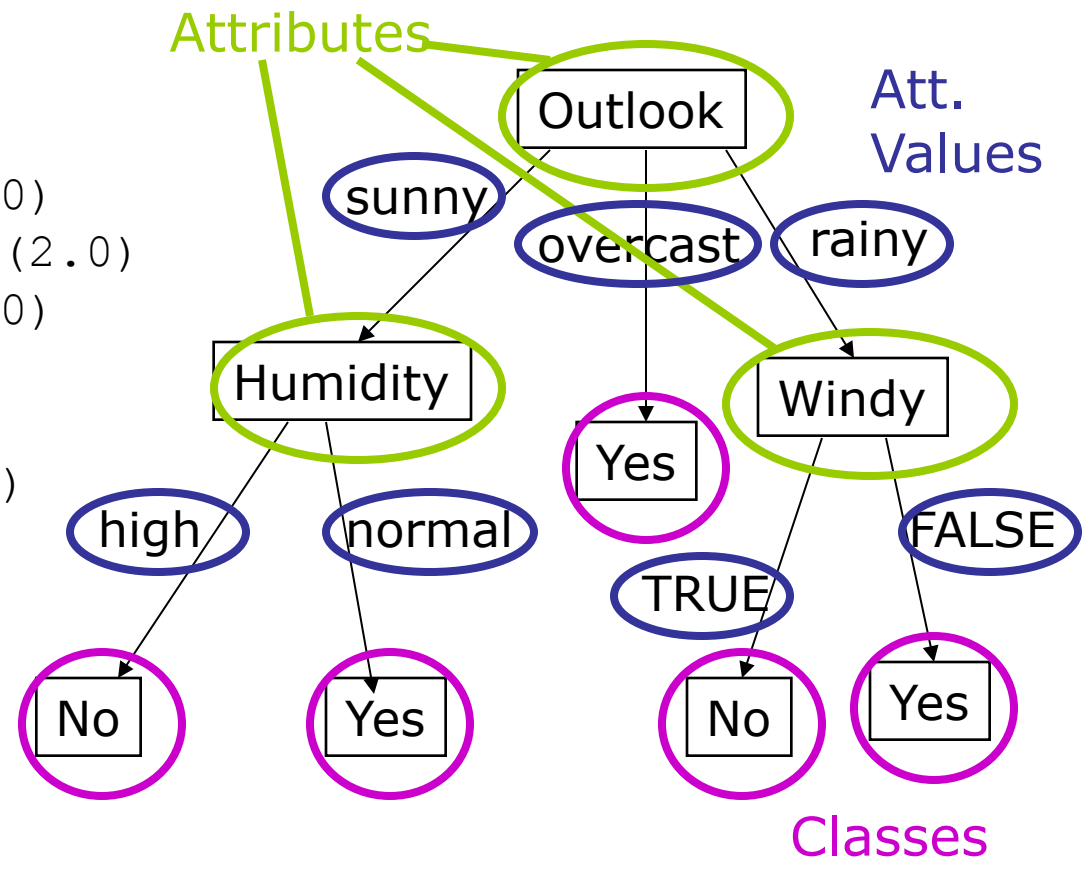
NO

Decision Tree



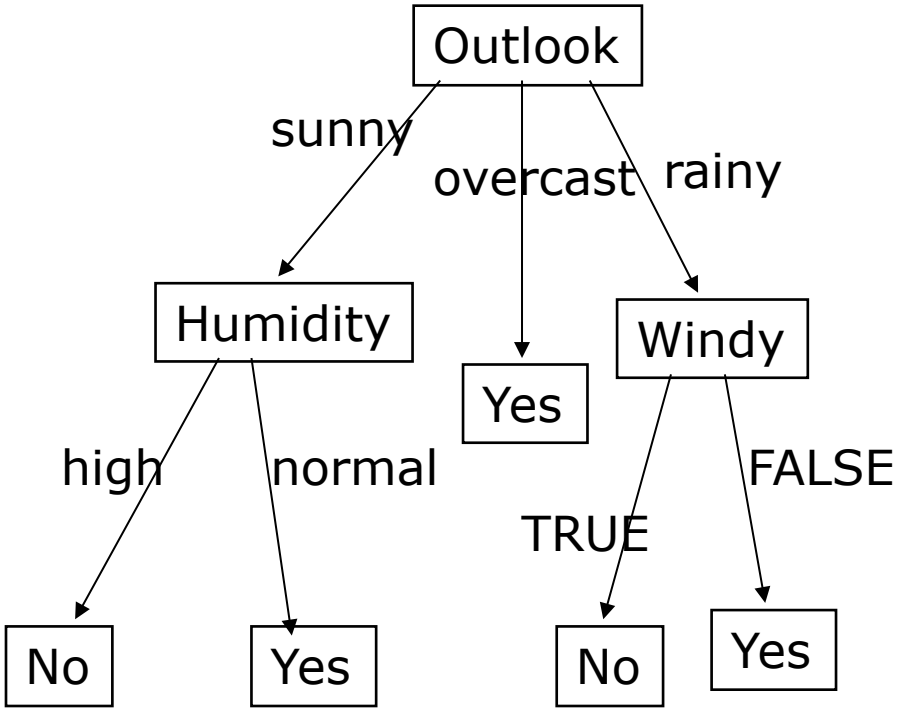
Decision Trees (Quinlan, 1993)

```
J48 pruned tree
------------------

outlook = sunny
|   humidity = high: no (3.0)
|   humidity = normal: yes (2.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  : 5

Size of the tree :  8
```




```
Time taken to build model: 0.05 seconds
Time taken to test model on training data: 0 seconds
```

Converting Trees to Rules



R1: IF Outlook = sunny $\wedge$ Humidity = high THEN play = No

R2: IF Outlook = sunny $\wedge$ Humidity = normal THEN play = Yes

R3: IF Outlook = overcast THEN play = Yes

R4: IF Outlook = rainy $\wedge$ Windy = TRUE THEN play = No

R5: IF Outlook = rainy $\wedge$ Windy = FALSE THEN play = Yes

Bayes Theorem

In machine learning we are interested to determine the best hypothesis *h(**x**)* from space *H*, based on the observed training data ***x***.

Best hypothesis = *most probable* hypothesis, given the data ***x*** with any initial knowledge about the prior probabilities of the various hypothesis in *H*.

Bayes theorem provides a way to calculate

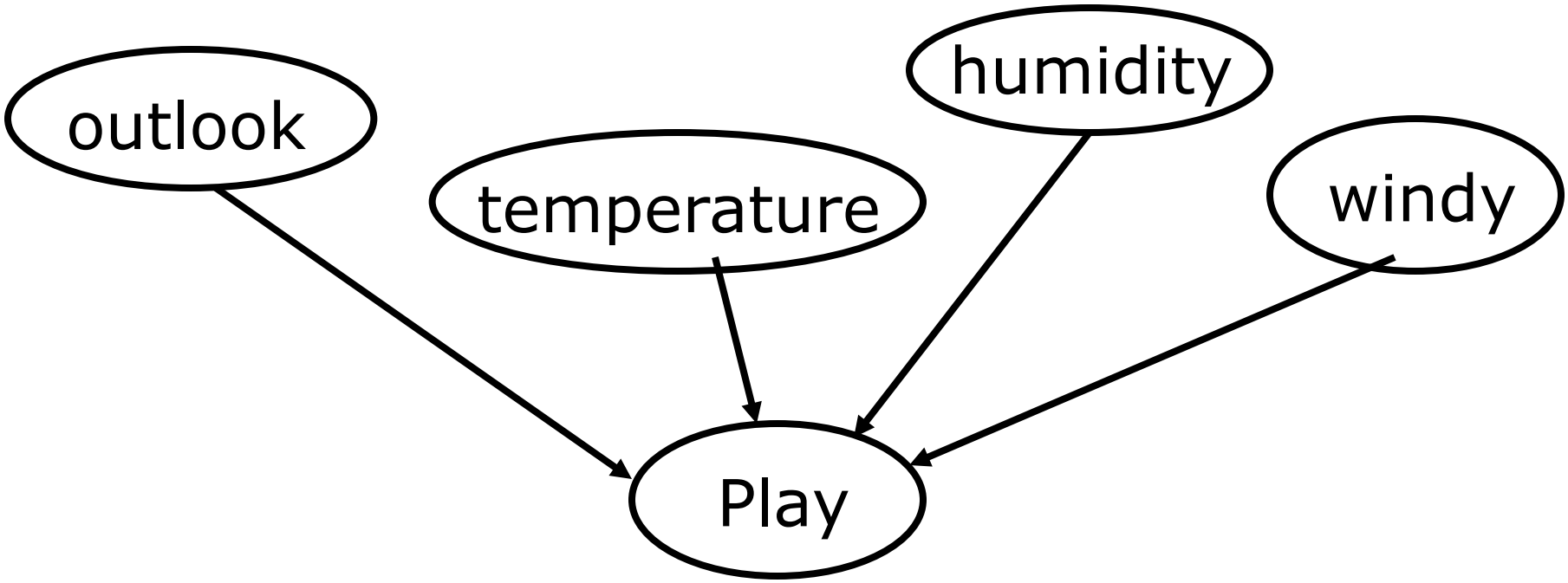
(i) the probability of a hypothesis based on its prior probability Pr(*h(**x**)*)
(ii) the probabilities of the observing various data given the hypothesis Pr(***x***|h)
(iii) the probabilities of the observed data Pr(***x***)

We can calculate the posterior probability *h(**x**)* given the observed data ***x***, Pr(*h(**x**)*|***x***) using *Bayes theorem*.

$$\Pr(h(x) \mid x) = \frac{\Pr(x \mid h(x)) \Pr(h(x))}{\Pr(x)}$$

Naïve Bayes

(John & Langley, 1995)



To use all attributes and allow them to make contributions to the decision that are *equally important* and *independent* of one another, given the class.

Naïve Bayes Classifier

$$v_{NB} = \arg\max_{v_j \in V} \Pr(v_j) \prod_i \Pr(a_i \mid v_j)$$

Where v_{NB} denotes the target value output by the naïve Bayes classifer, $Pr(v_j)$ is the probability of target value v_j occurs in the training data, $Pr(a_i|v_j)$ is the conditionally independant probability of a_i given target value v_j.

Summary:

- The naïve Bayes learning method involves a learning step in which the various $Pr(v_j)$ and $Pr(a_i|v_j)$ terms are estimated, based on their frequencies over the training data.
- The set of these estimates corresponds to the learned hypothesis h(x).
- This hypothesis is then used to classify each new instance by applying the above rule.
- There is no explicit search through the space of possible hypothesis, instead the hypothesis is formed simply by counting the frequency of various data combinations within the training examples.

Naïve Bayes example

Today	sunny	cool	high	TRUE	?

Pr(Play = yes) = 9/14 = 0.64
Pr(Play = no) = 5/14 = 0.36

Pr(Outlook=sunny|Play = yes) = 2/9 = 0.22
Pr(Outlook=sunny|Play=no) = 3/5 = 0.60

Pr(Temperature = cool|Play = yes) =3/9 = 0.33
Pr(Temperature =cool|Play =no) =1/5 = 0.20

Pr(Humidity = high|Play = yes) = 3/9 =0.33
Pr(Humidity = high|Play = no) = 4/5 =0.80

Pr(Wind = TRUE|Play = yes) = 3/9 = 0.33
Pr(Wind = TRUE|Play = no) = 3/5 = 0.60

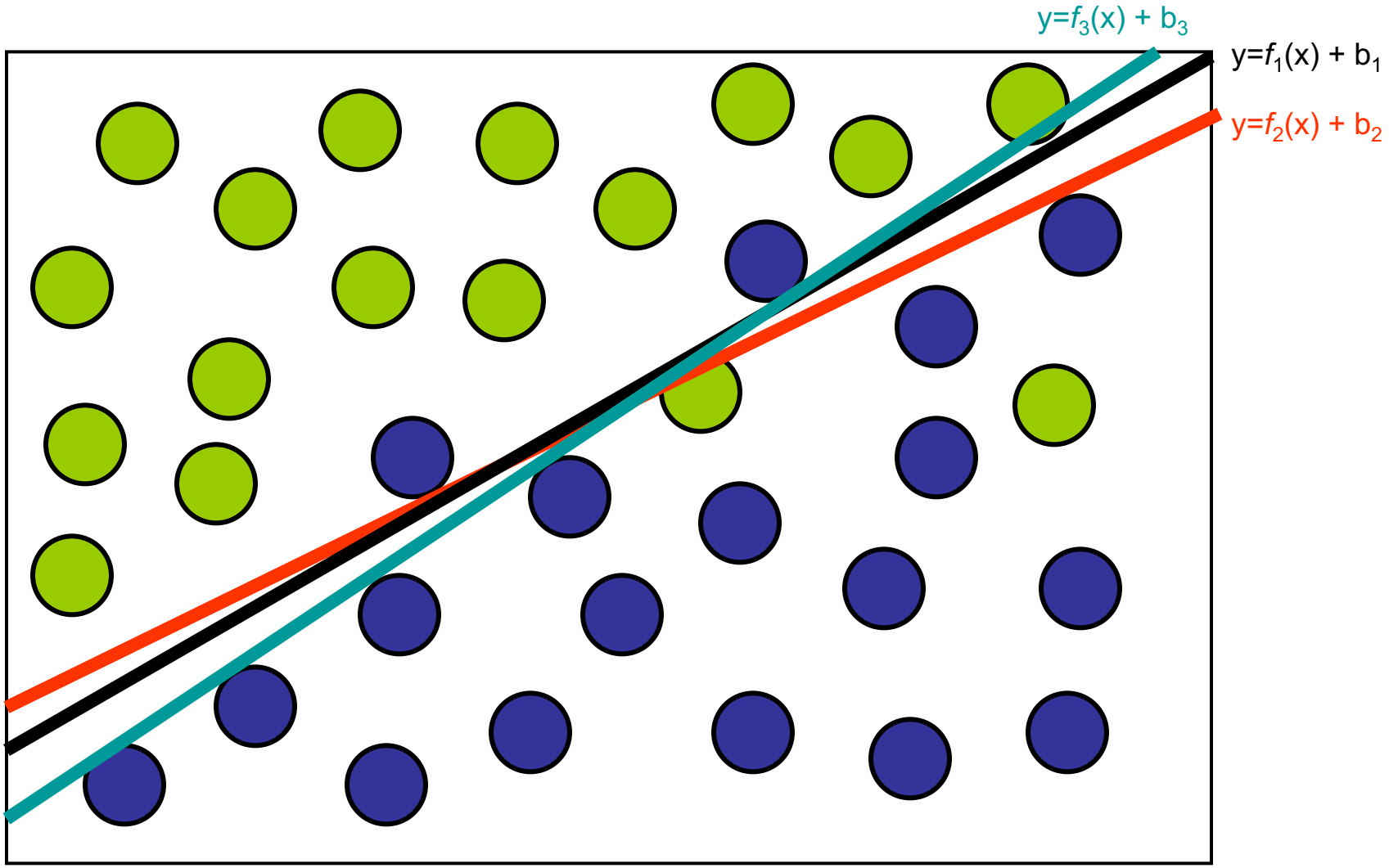
Pr(yes)Pr(sunny|yes)Pr(cool|yes) Pr(high|yes)Pr(TRUE|yes)= 0.64*0.22*0.33*0.33*0.33 = 0.0051

Pr(no)Pr(sunny|no)Pr(cool|no) Pr(high|no)Pr(TRUE|no)= 0.36*0.60*0.20*0.80*0.60 = 0.0207

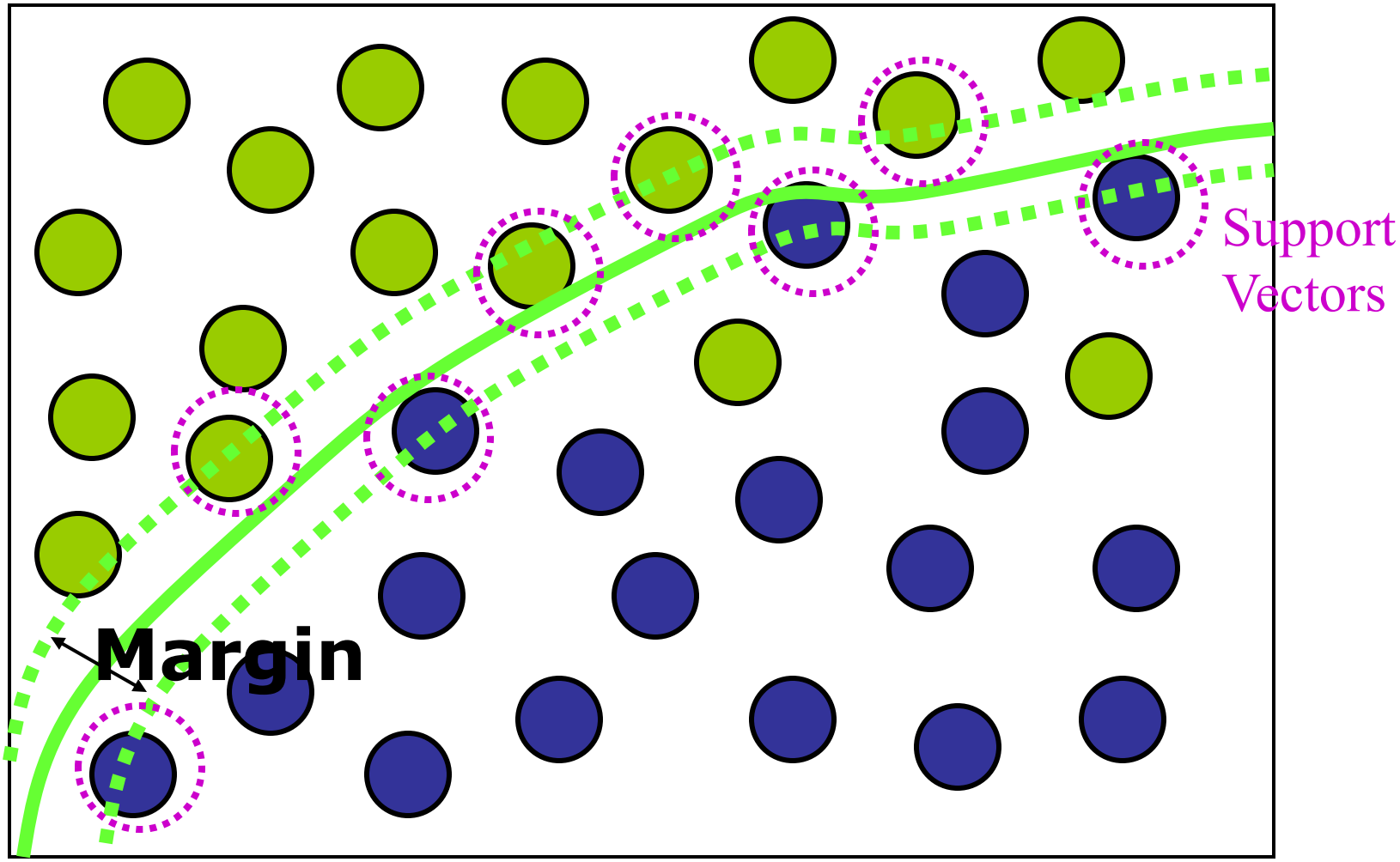
Play = NO

Probability = 0.0207/(0.0207+0.0051) =0.80 (80%)

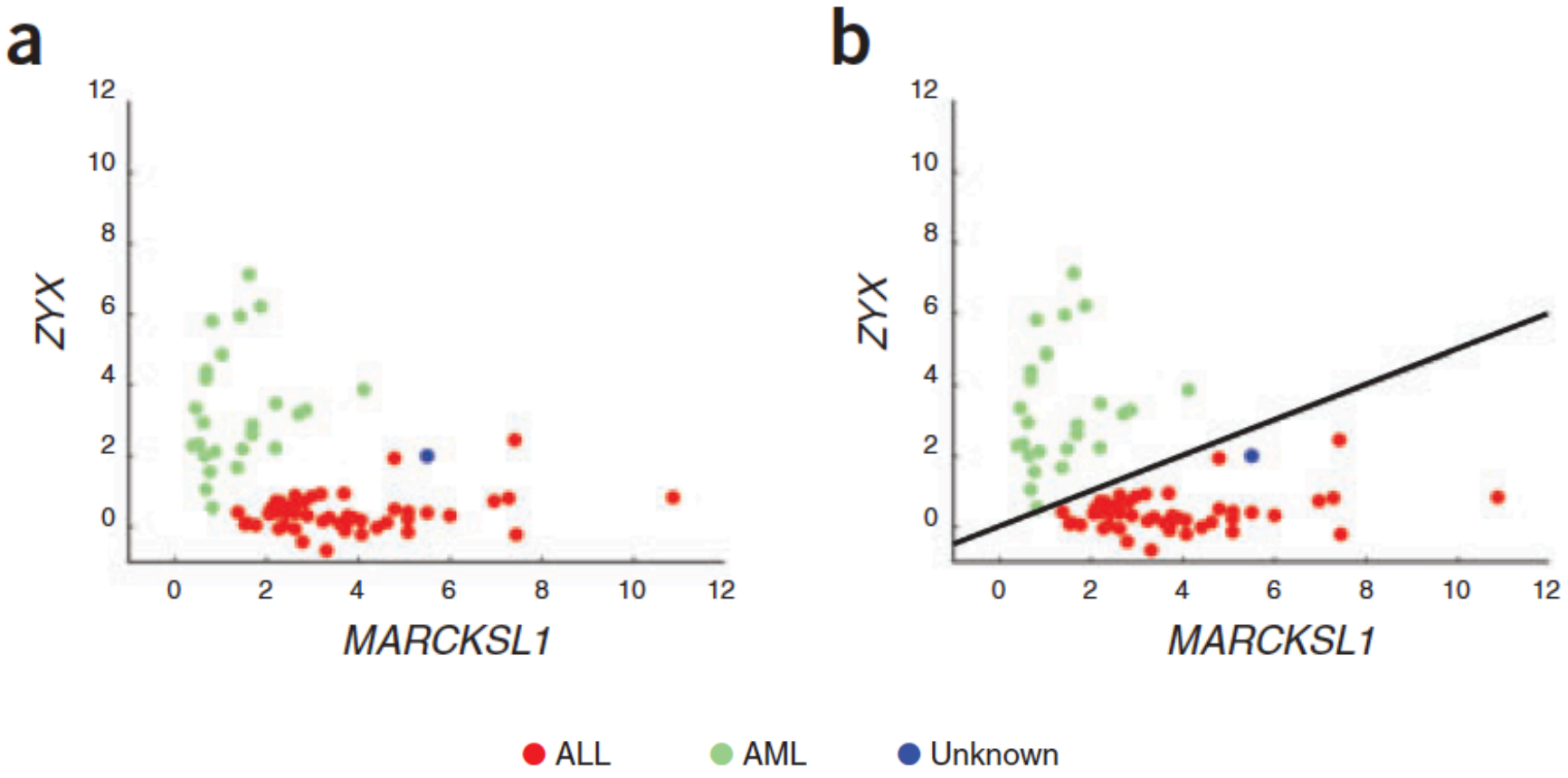
Linear Model



Support Vector Machines



Straight Line as Classifier in 2D Space



Support Vector Machines (SVM)

Key concepts:

Separating hyperplane – straight line in high-dimensional space

Maximum-margin hyperplane - the distance from the separating hyperplane to the nearest expression vector as the margin of the hyperplane Selecting this particular hyperplane maximizes the SVM's ability to predict the correct classification of previously unseen examples.

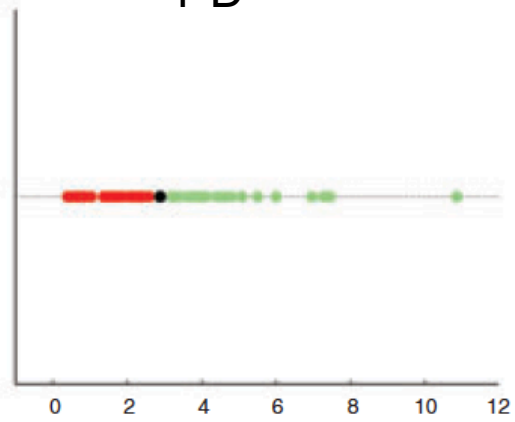
Soft margin - allows some data points ("soften") to push their way through the margin of the separating hyperplane without affecting the final result. User-specified parameter.

Kernel function - mathematical trick that projects data from a low-dimensional space to a space of higher dimension. The goal is to choose a good kernel function to separate data in high-dimensional space.

(Adapted from Noble 2006)

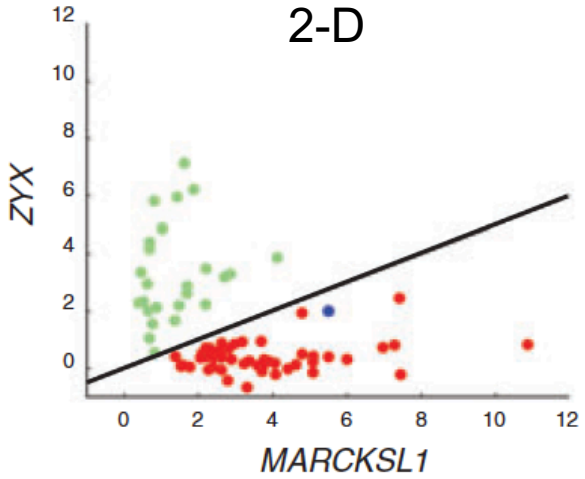
Separating Hyperplane

1-D



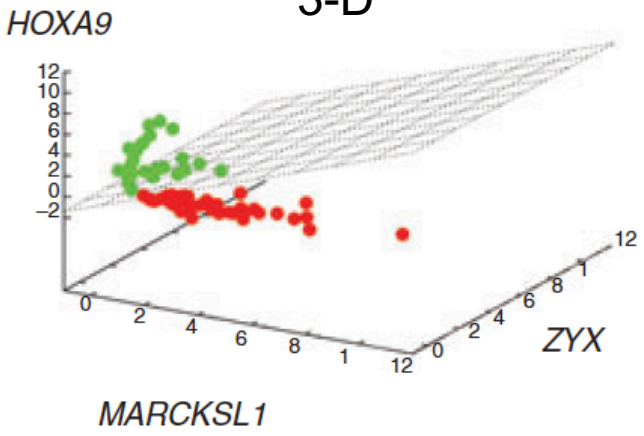
Separating hyperplane = dot

2-D



Separating hyperplane = line

3-D

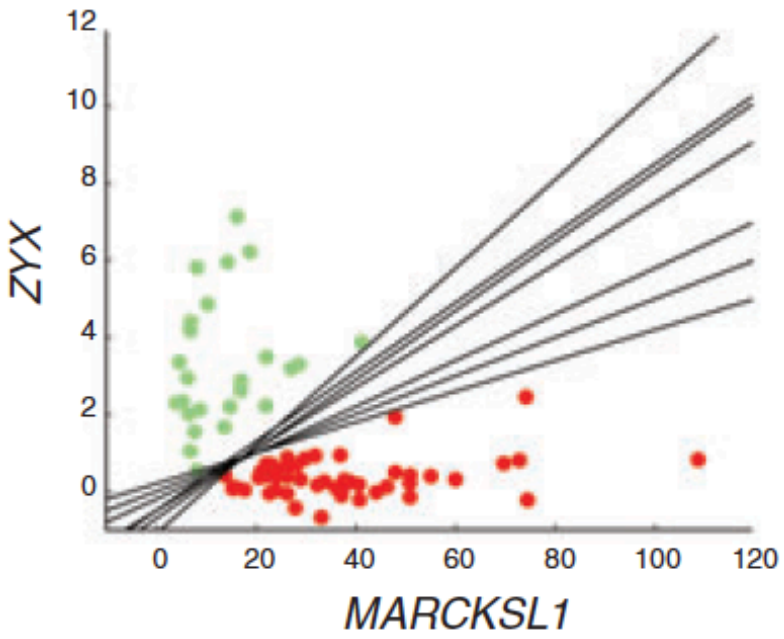


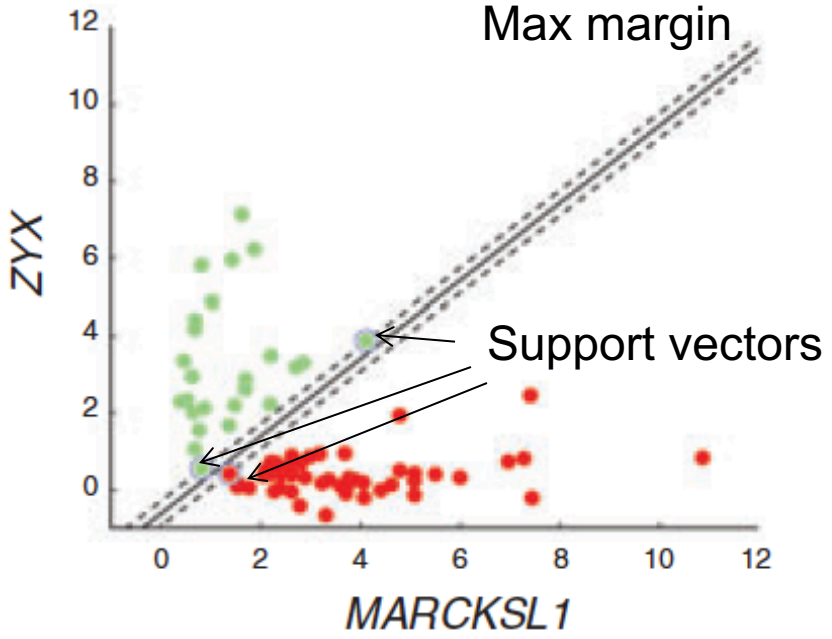
Separating hyperplane = hyperplane

ALL AML Unknown

(Adapted from Noble 2006)

Maximum-margin Hyperplane

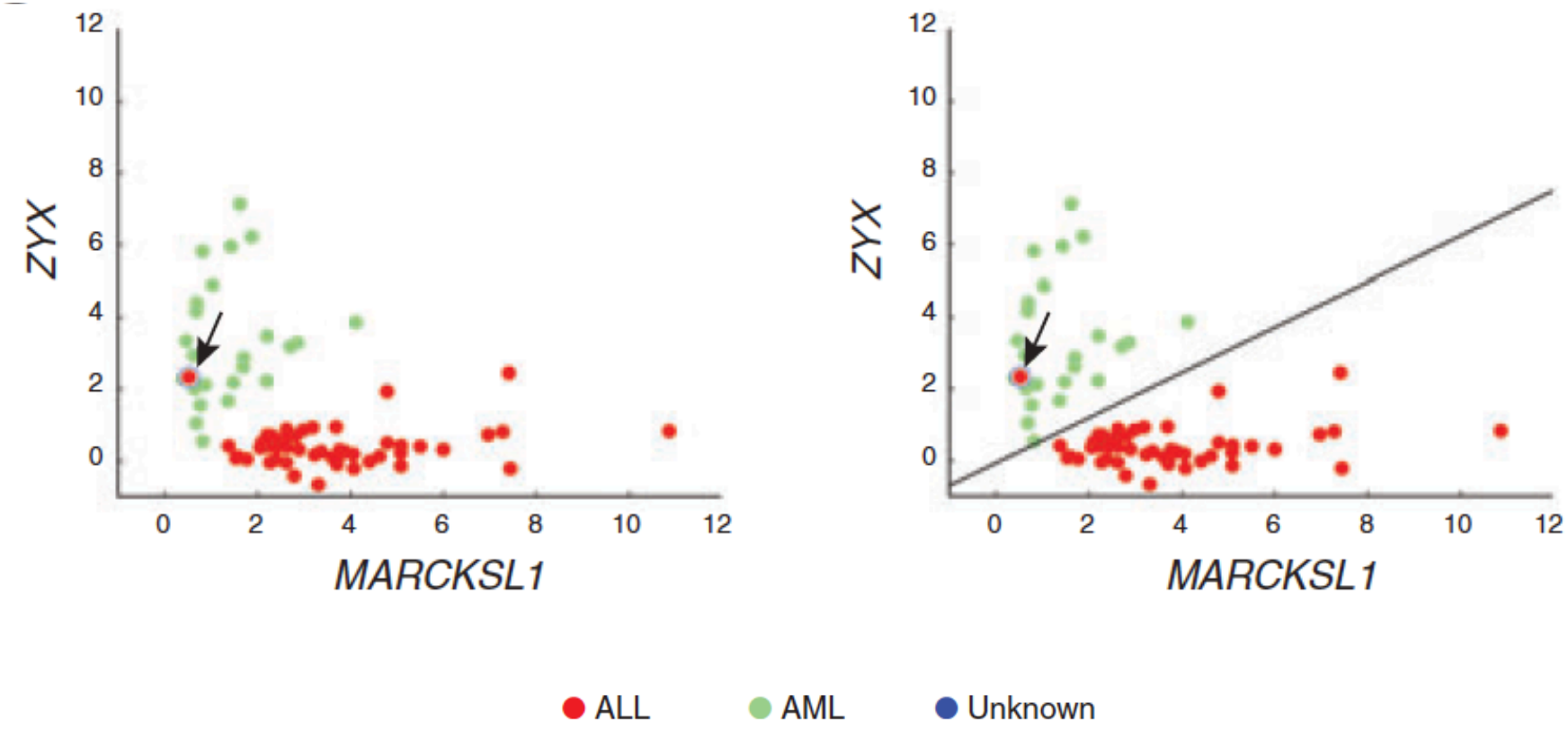




ALL AML Unknown

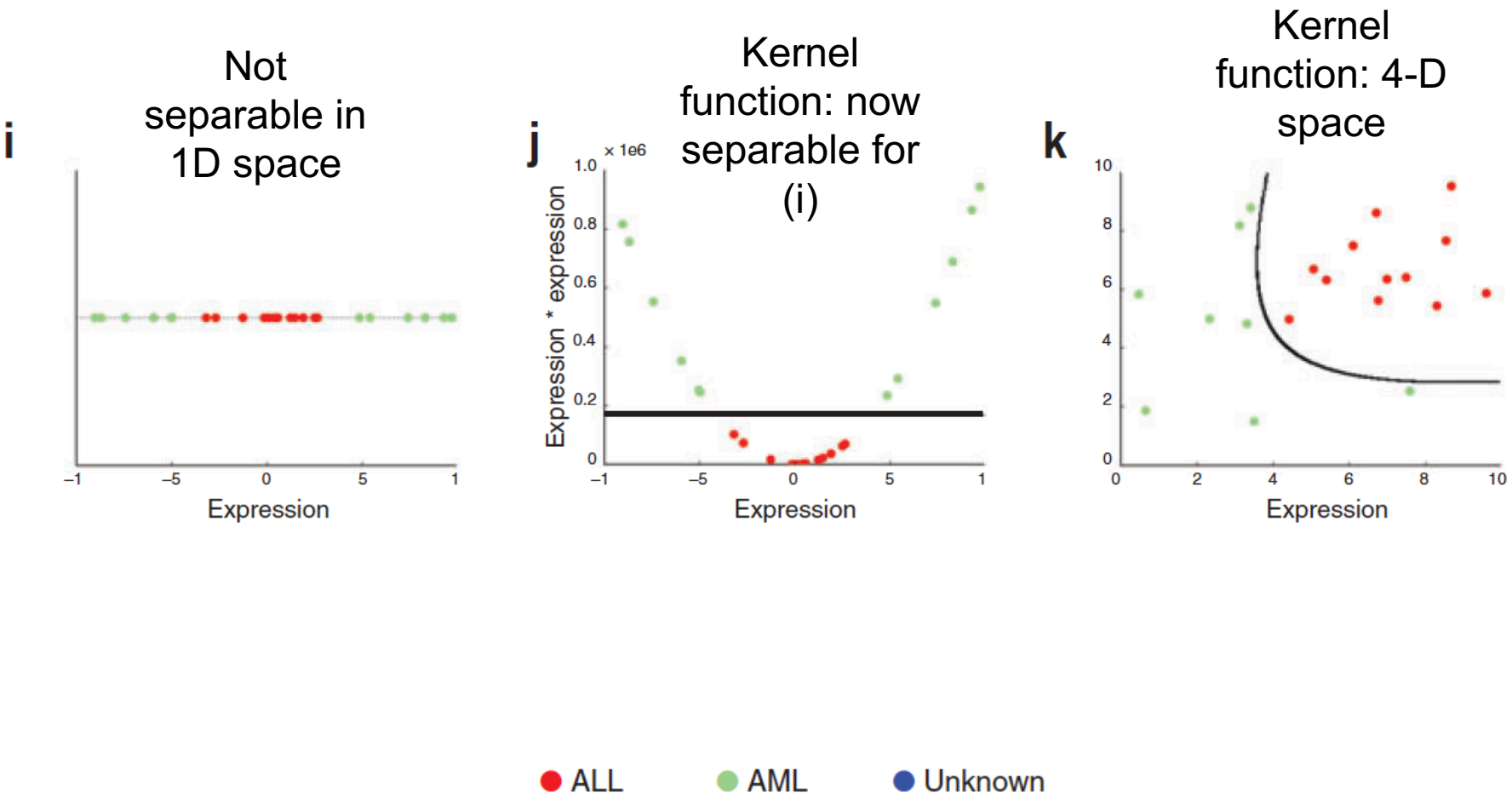
(Adapted from Noble 2006)

Soft Margin



(Adapted from Noble 2006)

Kernel Function



(Adapted from Noble 2006)

Kernal: Linear SVM = Linear Regression

Predictive Accuracy = 50%

Support Vectors = 0

```
SMO

Classifier for classes: yes, no

BinarySMO

Machine linear: showing attribute weights, not support vectors.

   0.8440785904115866 * outlook=sunny
 + -0.9533207559861846 * outlook=overcast
 + 0.10924216557459787 * outlook=rainy
 + 0.5276359628579281 * temperature
 + 0.7712122046533554 * humidity
 + -0.8907578344254022 * windy
 - 0.8688305080362968

Number of kernel evaluations: 66




=== Stratified cross-validation ===

Correctly Classified Instances           7               50      %
Incorrectly Classified Instances         7               50      %
Kappa statistic                         -0.2564
Mean absolute error                      0.5
Root mean squared error                  0.7071
Relative absolute error                105      %
Root relative squared error            143.3236 %
Total Number of Instances               14


=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 5 0 | b = no
```

Kernal: Polynomial (Quadratic Function)

Predictive Accuracy = 78.6%

Support Vectors = 10

```
SMO

Classifier for classes: yes, no

BinarySMO

   -1 * 0.8100635668551557 * K[X(1) * X]
 + -1 * 0.019817568367163058 * K[X(3) * X]
 + -1 * 0.8887836783080866 * K[X(4) * X]
 + -1 * 1.0 * K[X(6) * X]
 + -1 * 0.3534785049716326 * K[X(7) * X]
 + 1 * 0.3727234704263585 * K[X(9) * X]
 + 1 * 0.1796126505860263 * K[X(10) * X]
 + 1 * 1.0 * K[X(11) * X]
 + 1 * 0.7245265999700636 * K[X(12) * X]
 + 1 * 0.7952805975195893 * K[X(13) * X]
 - 0.6275818453891167

Number of support vectors: 10

Number of kernel evaluations: 104



=== Stratified cross-validation ===

Correctly Classified Instances          11               78.5714 %
Incorrectly Classified Instances         3               21.4286 %
Kappa statistic                          0.5116
Mean absolute error                      0.2143
Root mean squared error                  0.4629
Relative absolute error                 45      %
Root relative squared error             93.8273 %
Total Number of Instances               14


=== Confusion Matrix ===

 a b   <-- classified as
 8 1 | a = yes
 2 3 | b = no
```

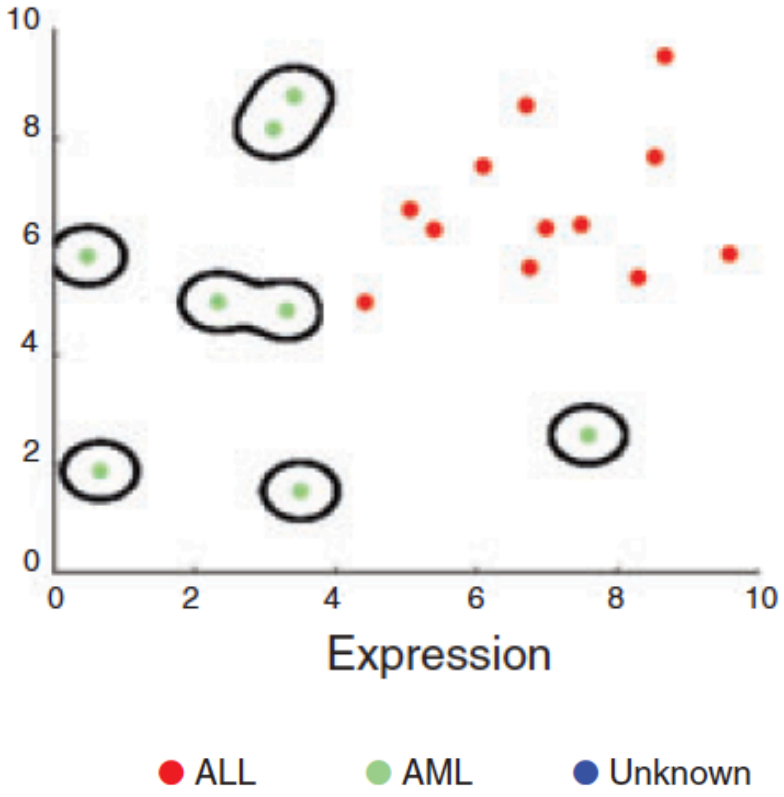
Kernal: Polynomial (Cubic Function)

Predictive Accuracy = 85.7%

Support Vectors = 12

```
SMO

Classifier for classes: yes, no

BinarySMO

   -1 * 0.058598146492685896 * K[X(1) * X]
 + -1 * 0.032159637558211406 * K[X(2) * X]
 + -1 * 0.36378917190737614 * K[X(3) * X]
 + -1 * 0.07019514690769074 * K[X(4) * X]
 + -1 * 0.07107098581642621 * K[X(5) * X]
 + -1 * 0.8766783011511141 * K[X(6) * X]
 + -1 * 0.06751016568226335 * K[X(7) * X]
 + 1 * 0.052713460422677855 * K[X(9) * X]
 + 1 * 0.3664976239753861 * K[X(10) * X]
 + 1 * 1.0 * K[X(11) * X]
 + 1 * 0.027564792859058898 * K[X(12) * X]
 + 1 * 0.0932256782586451 * K[X(13) * X]
 - 0.5679604722895839

Number of support vectors: 12

Number of kernel evaluations: 105



=== Stratified cross-validation ===

Correctly Classified Instances          12               85.7143 %
Incorrectly Classified Instances         2               14.2857 %
Kappa statistic                          0.6585
Mean absolute error                      0.1429
Root mean squared error                  0.378
Relative absolute error                 30      %
Root relative squared error             76.6097 %
Total Number of Instances               14


=== Confusion Matrix ===

 a b   <-- classified as
 9 0 | a = yes
 2 3 | b = no
```

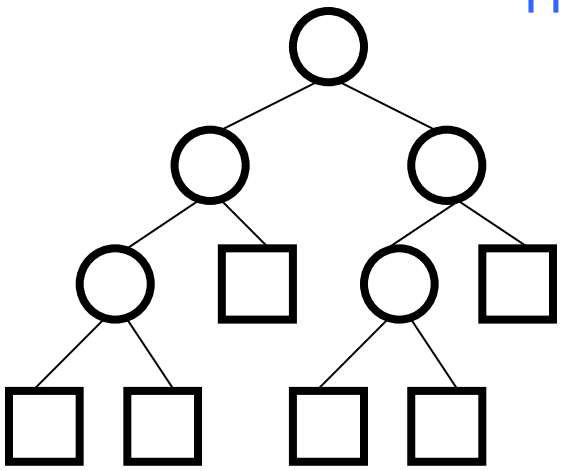
Overfitting in SVM



(Adapted from Noble 2006)

Overfitting

Overfitting : A classifier that performs good on the training examples but poor on unseen instances.



DT1

Low Training-set error: % errors on training data
High Generalisation error: % errors on unseen data

Train and test on same data → good classifier with massive overfitting

To avoid overfitting:

- Pruning the model
- Cross-validation (Computational expensive)
- Simpler model (Occam's razor)

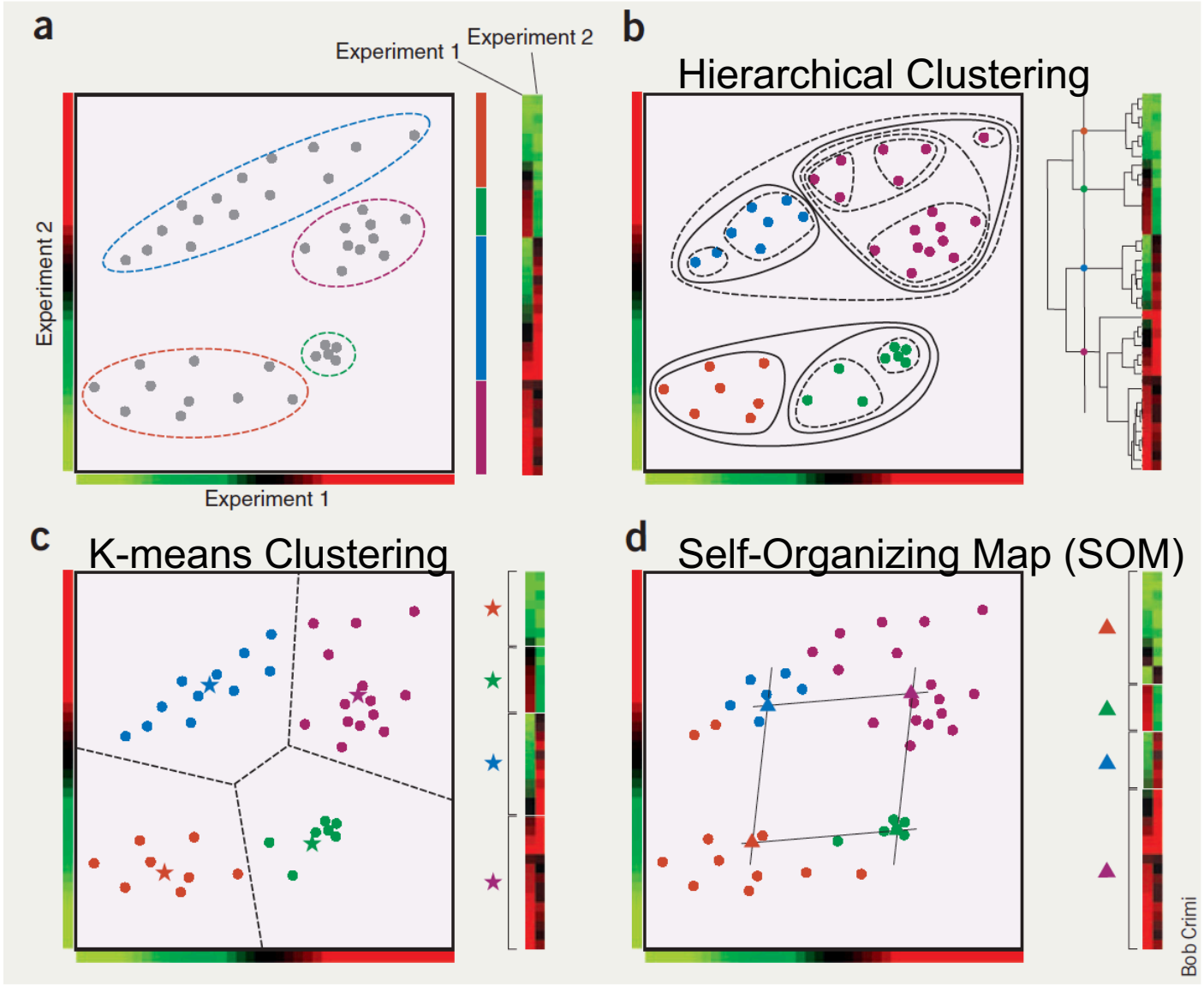
Clustering

- A method of grouping together data / samples that are ***similar*** in some way – based on certain criteria
- ***Unsupervised learning*** – no prior knowledge about the grouping
- Arranging objects into groups according to certain ***properties*** (e.g. expressions, mutations etc)
- Group members share certain properties in common and it is hoped that the resultant classification will provide some insight
- Useful for ***data exploration***
- Could be used to assign new samples into "clusters" –similarities of the new sample to one of the clusters.

Underlying Concepts

- Clustering depends on
  - **Similarity** determines how closely the objects resemble each other. Dissimilarity is the inverse of this, and this is related to the concept of **distance** .
  - **Distance measure** (e.g. Euclidian, correlation, etc)
  - Definition of **distance between clusters** (e.g. single linkage, average linkage etc)
  - **Number of clusters** (user-defined or computationally determined)

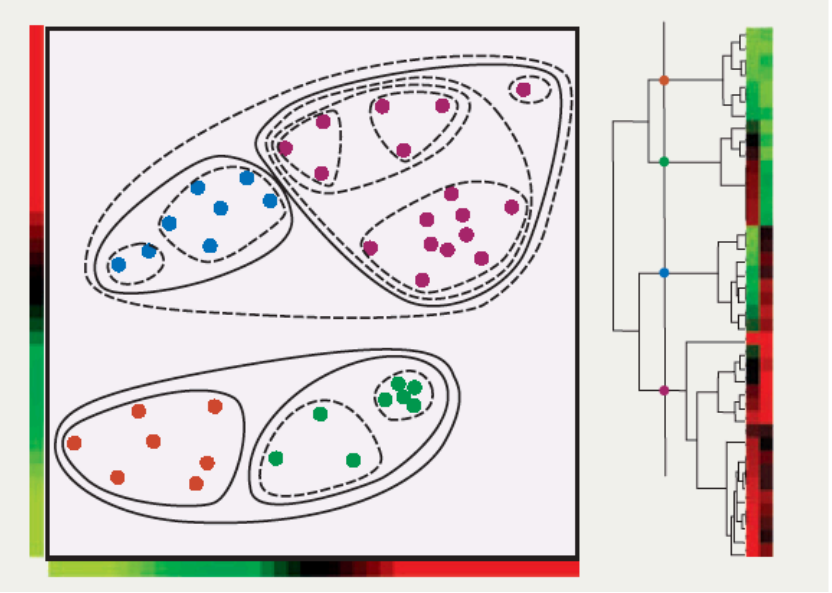
Common Clustering Methods



(Adapted from D'haeseleer 2005)

Hierarchical Clustering

- Step 1: Start every data point in a separate cluster.
- Step 2: Find pairs of data that are similar, merge into one cluster
- Step 3: Repeat Step 2 until one big cluster left



- Hierarchical clustering is a bottom-up or agglomerative method.
- Hierarchical clustering produces a binary tree or dendrogram.
- The final cluster is the root and each data point is a leaf.
- The height of the bars (braches) indicate how close (distance) between clusters
- Learn hierarchical clustering in today's workshop

Similarity Measures

Table 1 Gene expression similarity measures

Manhattan distance (city-block distance, L1 norm)	$d_{fg} = \sum_c \lvert e_{fc} - e_{gc} \rvert$
Euclidean distance (L2 norm)	$d_{fg} = \sqrt{\sum_c (e_{fc} - e_{gc})^2}$
Mahalanobis distance	$d_{fg} = (e_f - e_g)' \Sigma^{-1} (e_f - e_g)$, where Σ is the (full or within-cluster) covariance matrix of the data
Pearson correlation (centered correlation)	$d_{fg} = 1 - r_{fg}$, with $r_{fg} = \frac{\sum_c (e_{fc} - \bar{e}_f)(e_{gc} - \bar{e}_g)}{\sqrt{\sum_c (e_{fc} - \bar{e}_f)^2 \sum_c (e_{gc} - \bar{e}_g)^2}}$
Uncentered correlation (angular separation, cosine angle)	$d_{fg} = 1 - r_{fg}$, with $r_{fg} = \frac{\sum_c e_{fc} e_{gc}}{\sqrt{\sum_c e_{fc}^2 \sum_c e_{gc}^2}}$
Spellman rank correlation	As Pearson correlation, but replace e_{gc} with the rank of e_{gc} within the expression values of gene g across all conditions $c = 1 \ldots C$
Absolute or squared correlation	$d_{fg} = 1 - \lvert r_{fg} \rvert$ or $d_{fg} = 1 - r_{fg}^2$

d_{fg}, distance between expression patterns for genes f and g. e_{gc}, expression level of gene g under condition c.

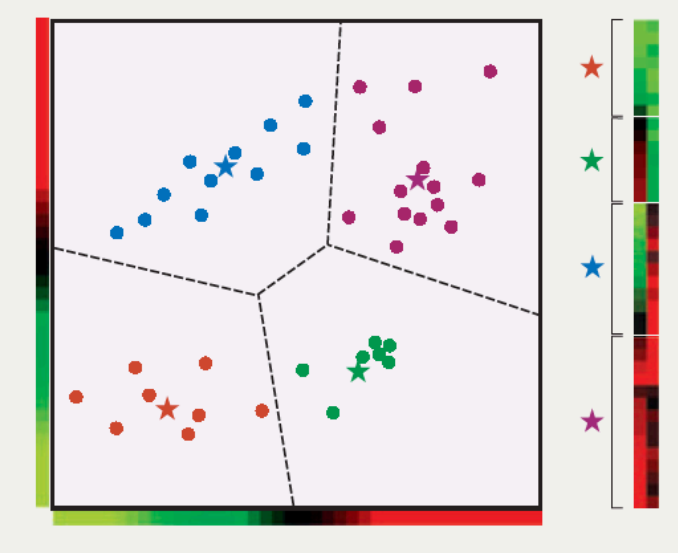
(Adapted from D'haeseleer 2005)

Linkage Methods

Method	Description
Single Linkage	• Minimum of all pairwise distances between points in the two clusters. • Tends to produce long, “loose” clusters.
Complete Linkage	• Maximum of all pairwise distances between points in the two clusters. • Tends to produce very tight clusters.
Average Linkage	• Average of all pairwise distances between point in the two clusters
Centroid Linkage	• Each cluster is associated with a mean vector which is the mean of all the data points in the cluster. • Distances between two mean vectors.

K-means Clustering

- An iterative method that creates K clusters.



Step 1: define number of clusters k

Step 2: initialize cluster centers

  - Pick k data points and set cluster centers to these points
  - Or randomly assign points to clusters and take means of clusters

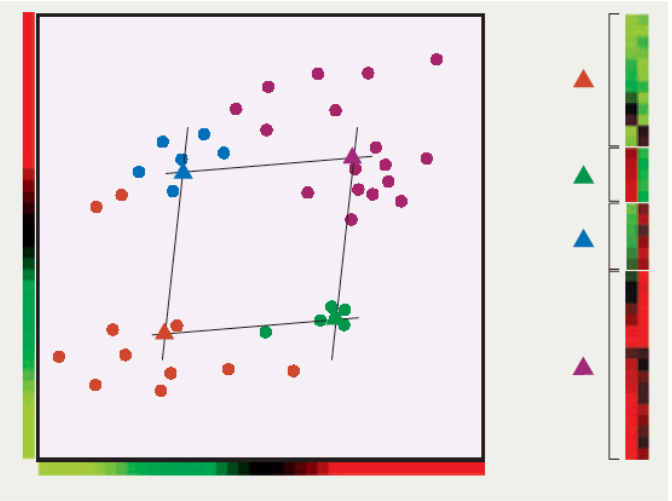
Step 3: For each data point, compute the cluster center closest to it and assign the data point to this cluster

Step 4: Re-compute cluster centers

Stop when there are no new re-assignments.

Self-Organizing Maps

- It requires pre-define number of clusters centroids and pre-specify a topology – a 2D grid that gives the geometric relationships between the clusters.
- For each data point, SOM algorithm moves the cluster centroids to its closest data point, but maintaining the topology specified by the 2D grid.
- At the end of the process, nearby data points tend to map to nearby cluster centroids.



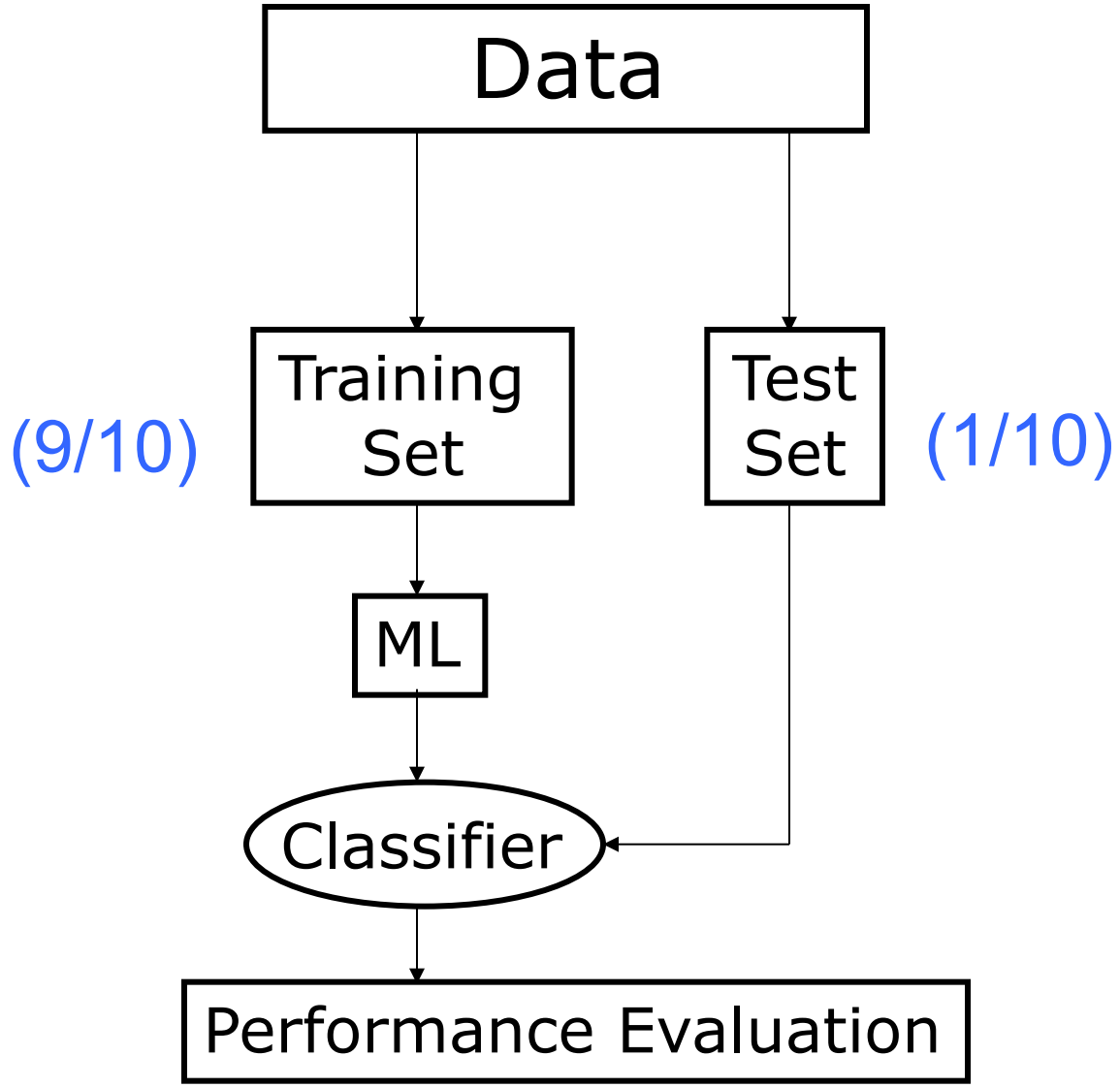
Comparisons of the Clustering Methods

Hierarchical Clustering	K-means Clustering	Self-Organizing Map (SOM)
• Easy to implement • Provide intuitive results (dendrogram) • Hard to decide the stopping criteria	• Easy to implement • Need to pre-specify number of k clusters • Unstable – due to random assignment in different runs	• Complicated and lots of parameters for "tweaking" • Defining the topology in high-dimensional is not obvious • Need to pre-specify number of k clusters

Comparison between classifiers

- Size (Complex? Simple?)
- Sensitivity, specificity?
- Coverage?
- Compression?
- Receiver Operating Characteristic (ROC) Curve

10-Fold Cross-validation



Confusion matrix / Contingency Table

		Predicted		
		Positive	**Negative**	
Actual	**Positive**	**TP**	**FN**	**Positive Examples**
	Negative	**FP**	**TN**	**Negative Examples**

True Positives(TP): $x \in X+$ and $h(x)$ = TRUE
True Negatives(TN): $x \in X-$ and $h(x)$ = FALSE
False Positives(FP): $x \in X-$ and $h(x)$ = TRUE
False Negatives(FN): $x \in X+$ and $h(x)$ = FALSE

Performance measurements

Accuracy

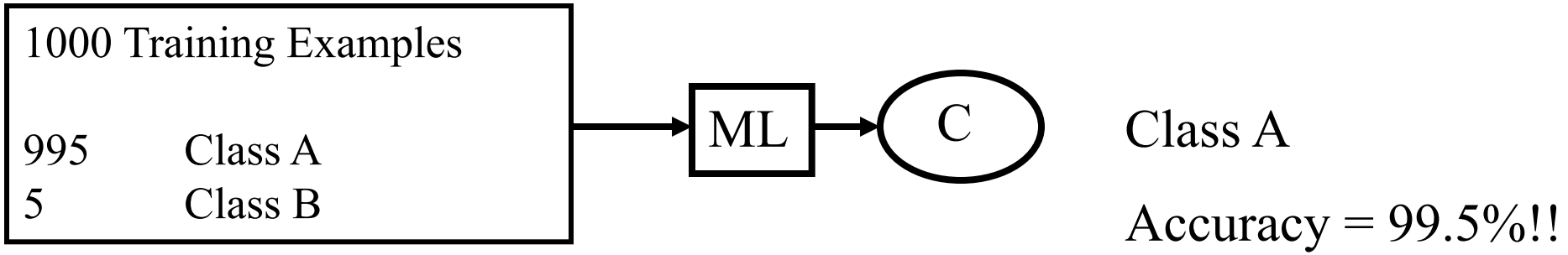
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$0 \leq \text{Accuracy} \leq 1$

Accuracy Error, ε = 1 - Accuracy

NOT the good measurement for evaluating classifier's performance!!

IF the classes are unequally represented in the training examples



Prediction Reliability

Reliability of Positive Prediction (Positive Predicted Value / Precision)

$$PPV = \frac{TP}{TP + FP}$$

$0 \leq \text{PPV} \leq 1$

Reliability of Negative Prediction (Negative Predicted Value)

$$NPV = \frac{TN}{TN + FN}$$

$0 \leq \text{NPV} \leq 1$

More measurements …

TP-rate (Sensitivity / Recall)

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

$0 \leq \text{Sn} \leq 1$

TN-rate (Specificity)

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

$0 \leq \text{Sp} \leq 1$

FP-rate

$$FP - rate = \frac{FP}{FP + TN}$$

$0 \leq \text{FP-rate} \leq 1$

FN-rate

$$FN - rate = \frac{FN}{TP + FN}$$

$0 \leq \text{FN-rate} \leq 1$

Other Statistical Measurements

F – measure (van Rijsbergen)

$$F-measure=\frac{2\times recall\times precision}{recall+precision}=\frac{2TP}{2TP+FP+FN}.$$

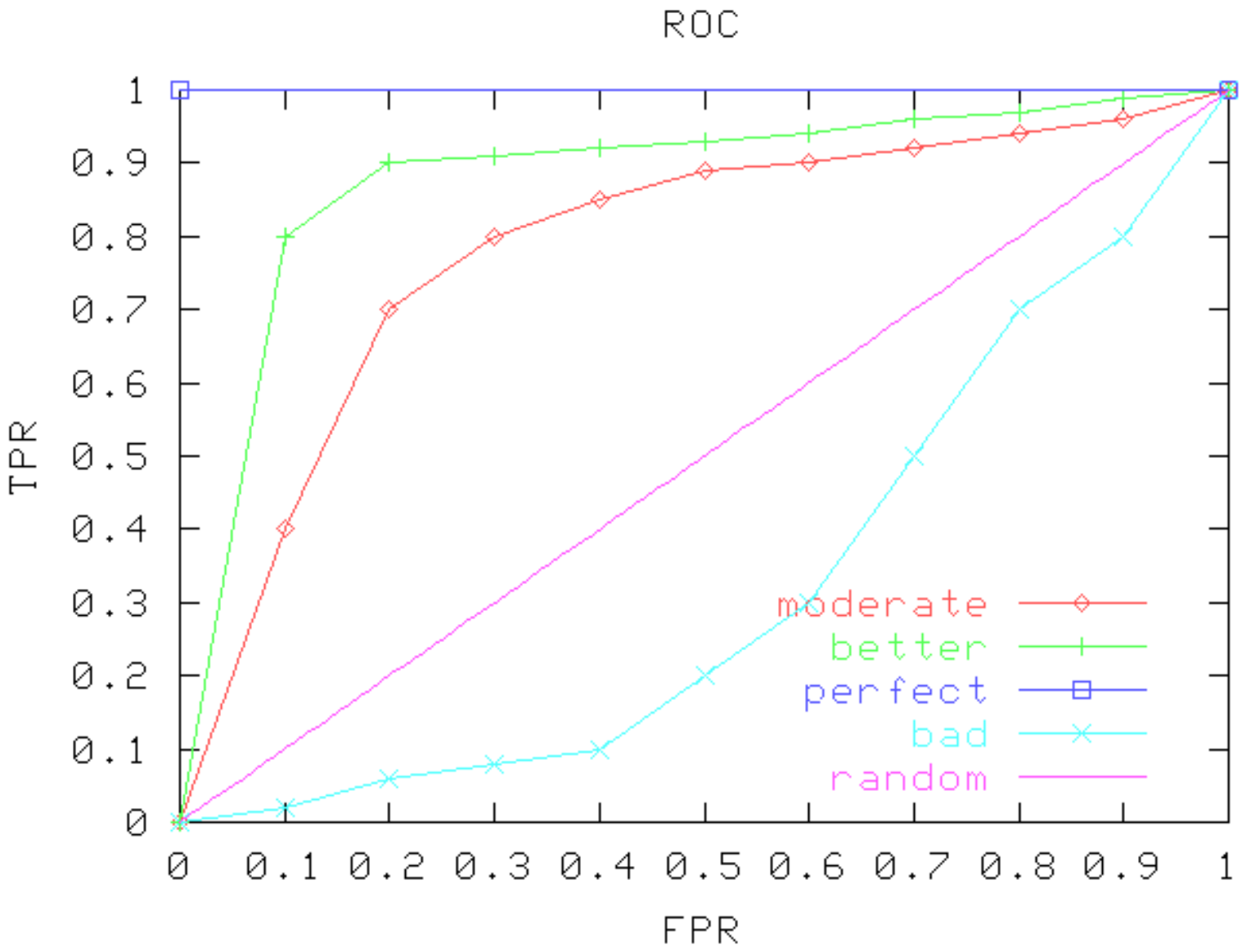
Coefficient Correlation

$$cc=\frac{(TP*TN-FP*FN)}{\sqrt{(TP+FP)*(FP+TN)*(TN+FN)*(FN+TP)}}$$

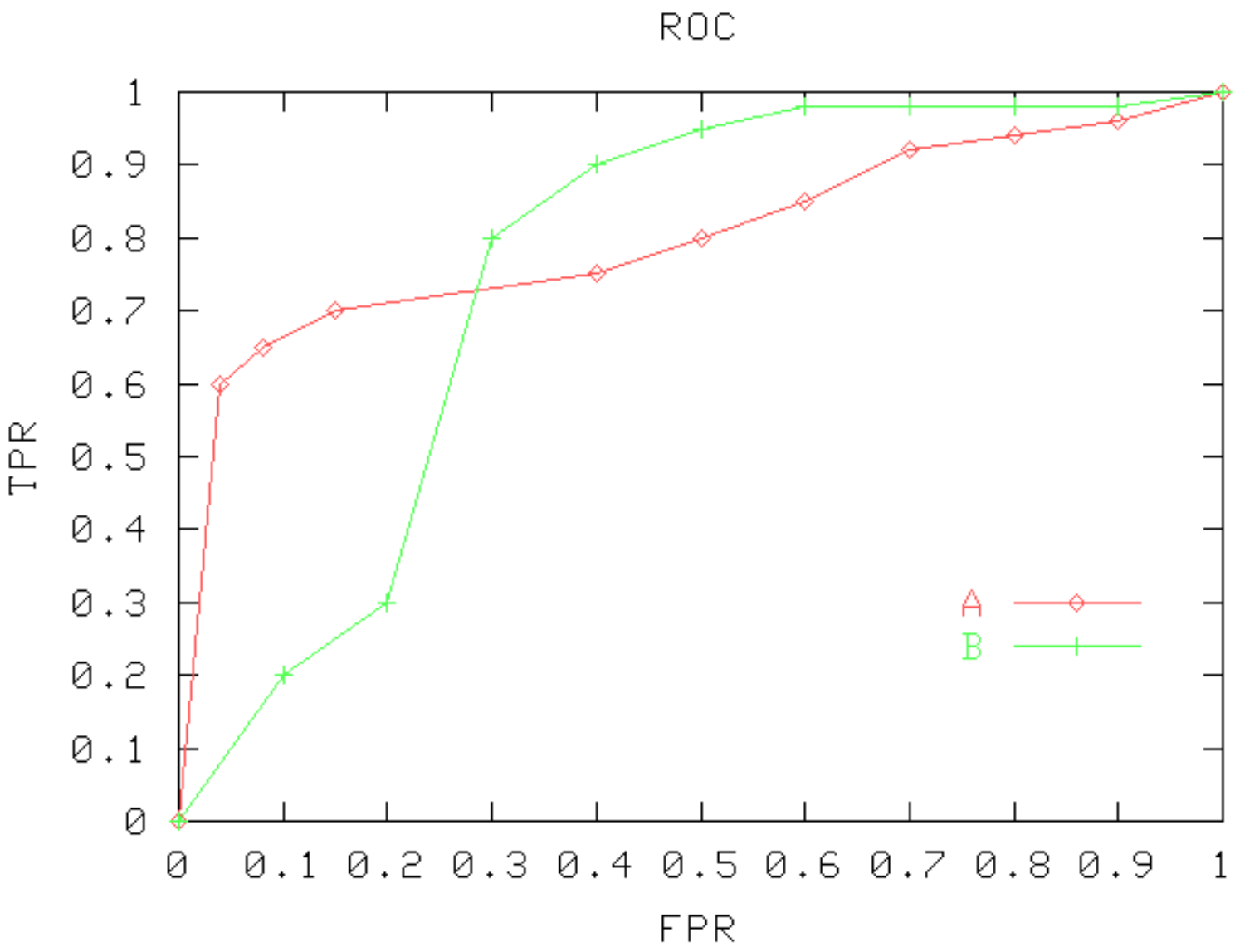
$-1 \le cc \le 1$

cc
- 1.0 no FP or FN
- 0.0 when *f* is random with respect to S+ and S-
- -1.0 only FP and FN

Receiver Operating Curve (ROC)



ROC analysis



Area Under Curve (AUC)

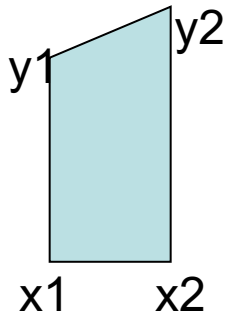
Which classifier performs better?

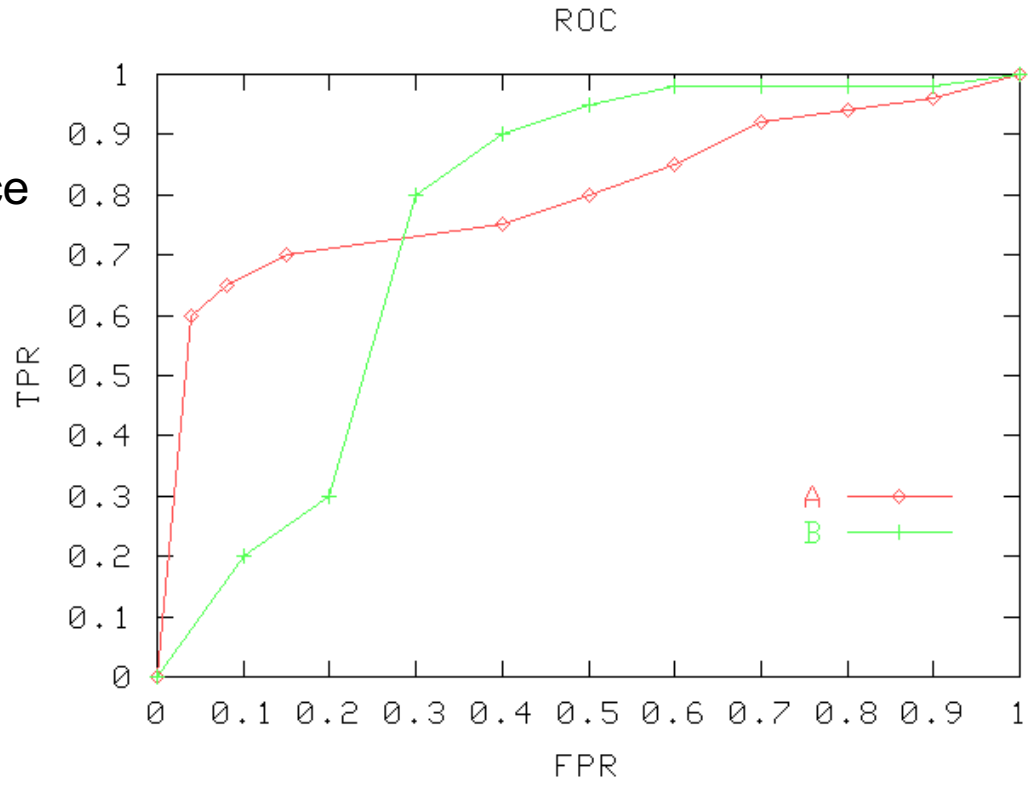
Area Under Curve (AUC) as a Measure of a classifier's performance

Area of trapezoid

The area of a trapezoid is simply the average height times the width of the base.

1. **function** *trap_area*(x1;x2; y1; y2)
2. Base = |x1-x2|
3. $Height_{avg}$ = (y1+y2)/2
4. **return** Base*$Height_{avg}$
5. **end function**





perl AUC.pl -f A -x 0 -y 1

A, AUC = 0.8

perl AUC.pl -f B -x 0 -y 1

B, AUC = 0.757

Take home message

- Machine learning has been widely applied in bioinformatics, especially in the classification and clustering of high-dimensional “omics” data
- Need to understand the “problem” (task) and choose the appropriate machine learning technique
- Do compare with different methods
- The ultimate goal is to interpret the data