

# Machine Learning in Bioinformatics

BMI/CS 776

[www.biostat.wisc.edu/bmi776/](http://www.biostat.wisc.edu/bmi776/)

Spring 2025

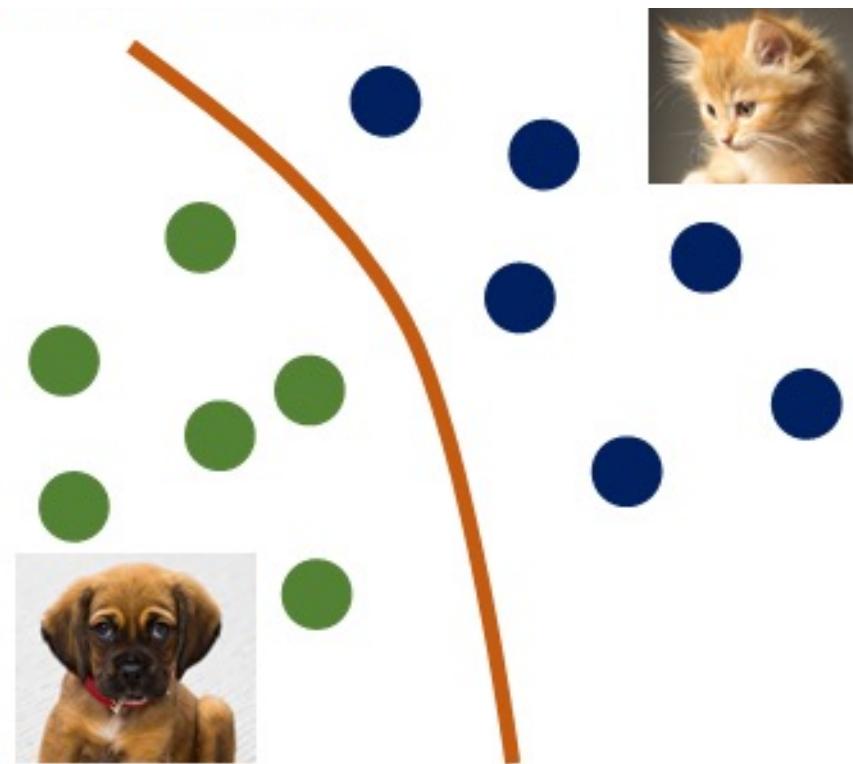
Daifeng Wang

[daifeng.wang@wisc.edu](mailto:daifeng.wang@wisc.edu)

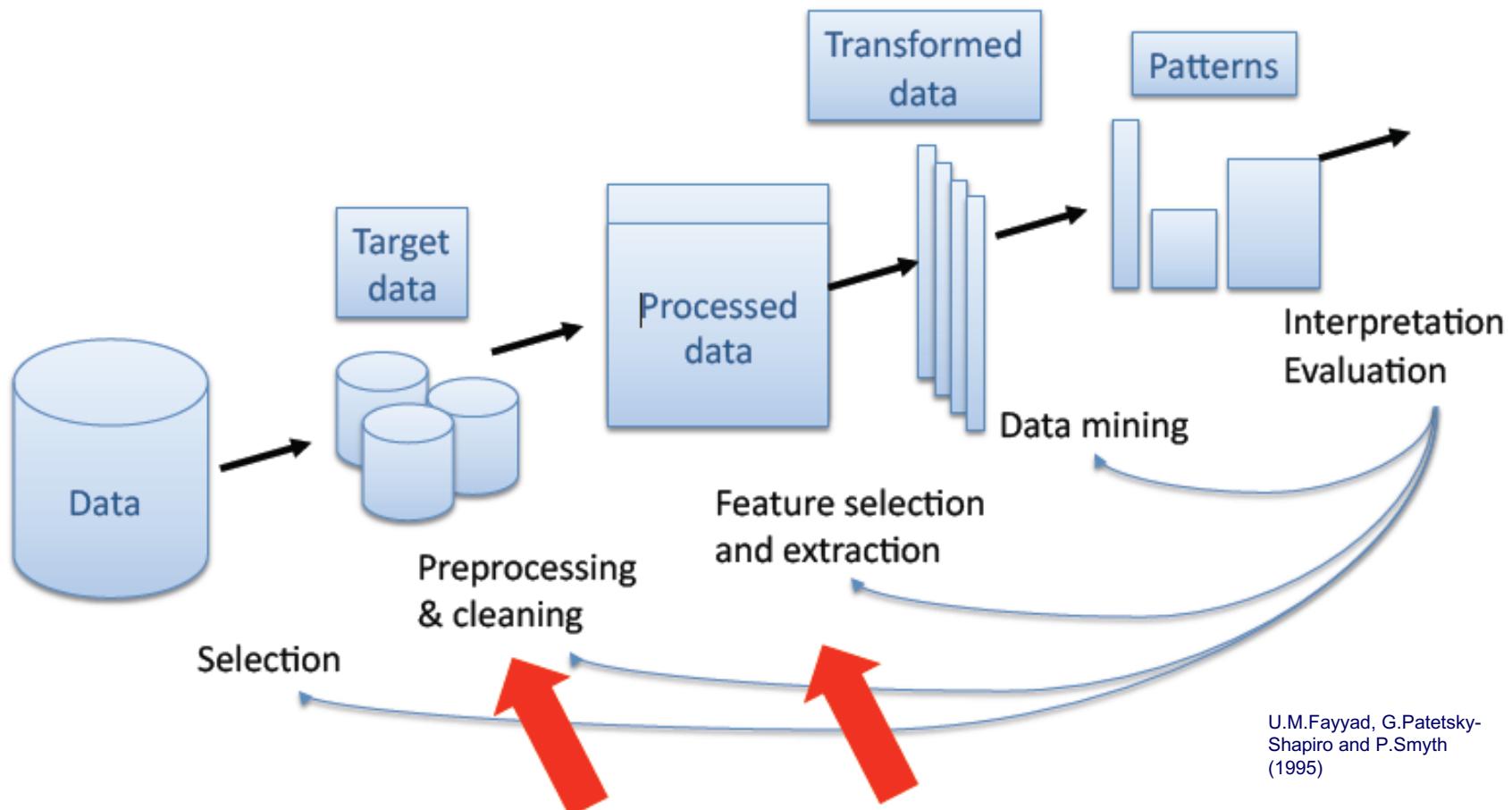
# Goals for lecture

- Basic machine learning ideas
- Feature selection
- Unsupervised learning
  - Partitioning vs. hierarchical clustering
- Supervised learning
  - Classification
- Applications in bioinformatics

# Dog vs Cat

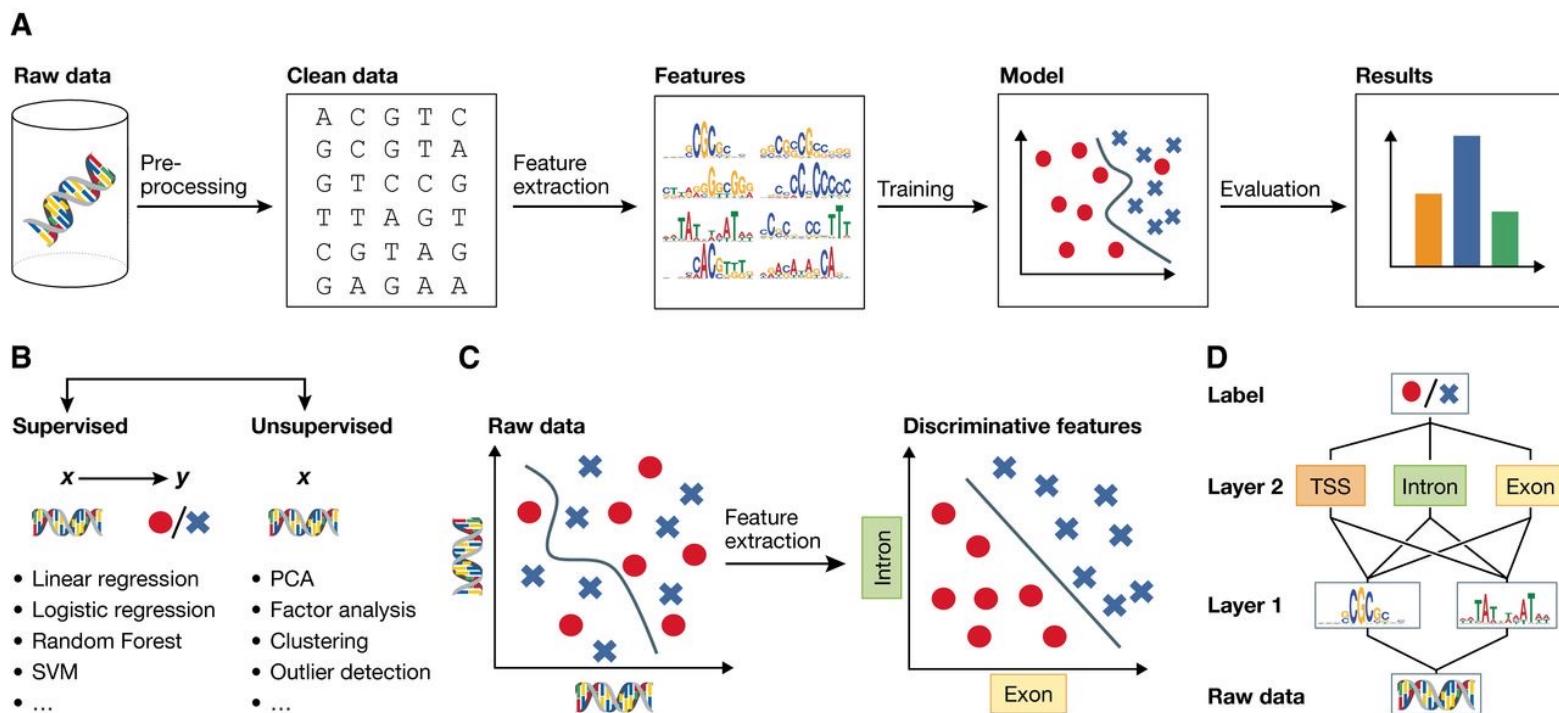


# Knowledge Discovery in Databases (KDD)



U.M.Fayyad, G.Piatetsky-Shapiro and P.Smyth  
(1995)

# Example: Machine learning in genomics



Christof Angermueller et al. Mol Syst Biol 2016;12:878

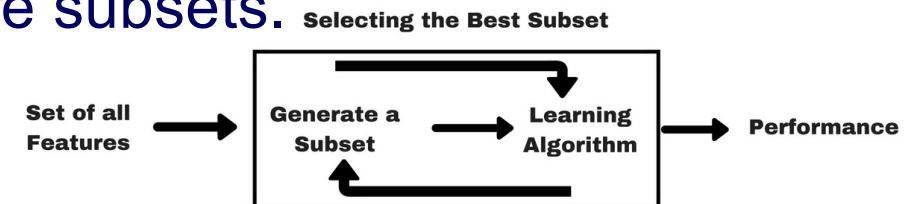


# Feature selection

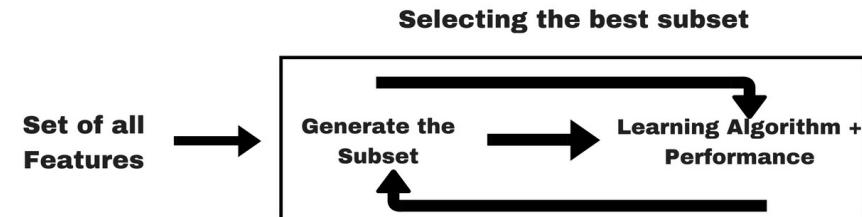
- **Filter approach** scores and ranks features independently of the predictor (classifier).
  - For example, t-test, correlation coefficient



- **Wrapper approach** uses a classifier/predictive model to search (many) best features or feature subsets.
  - Recursive feature elimination



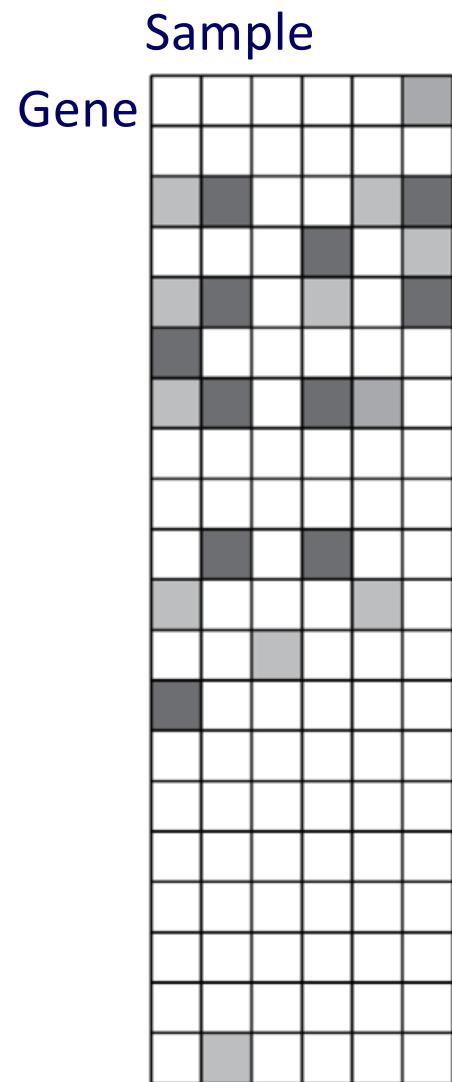
- **Embedded approach** uses a classifier/predictive model to build a (single) model with a subset of features that are internally selected.
  - LASSO regression



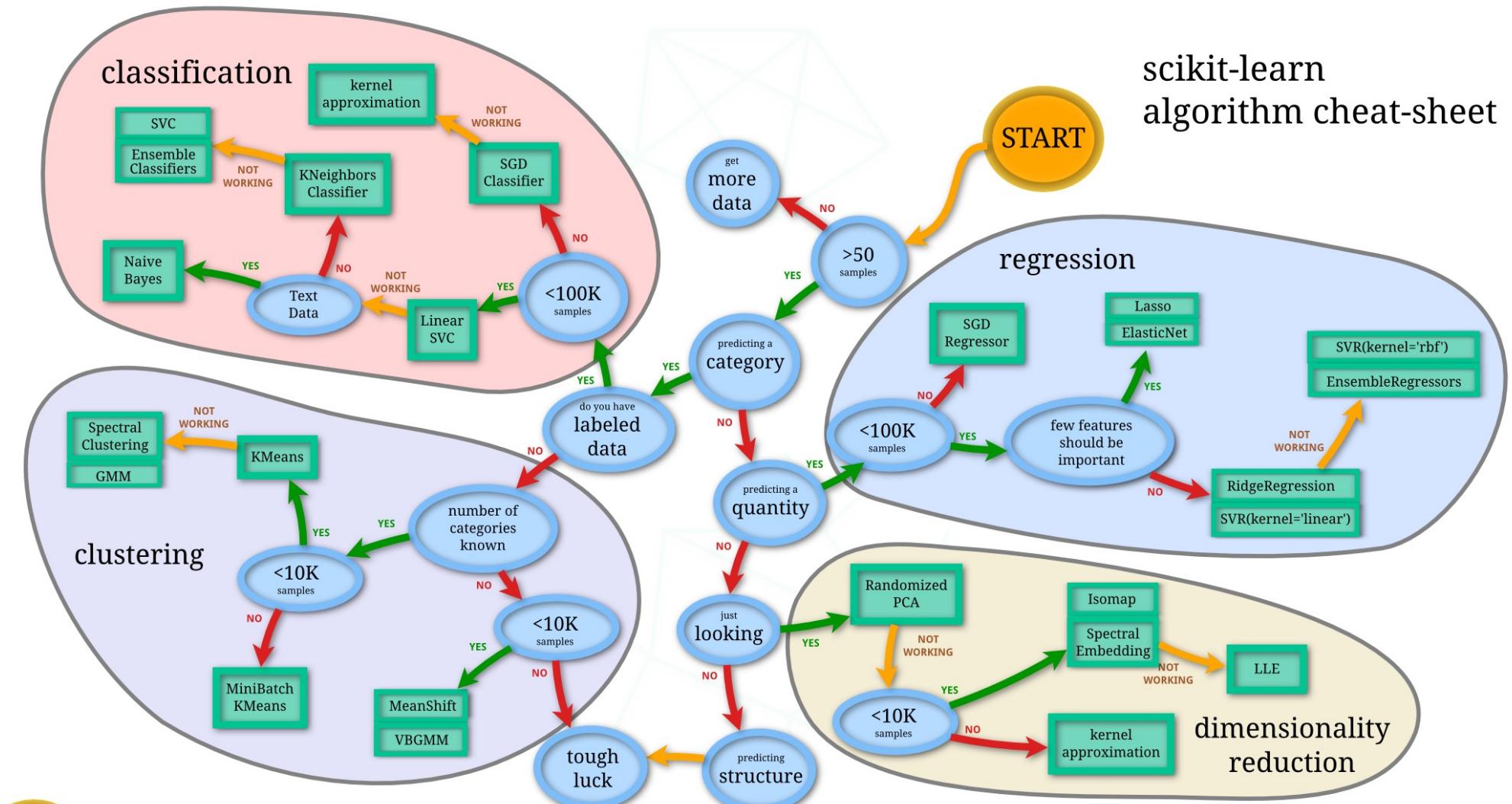
# Differentially expressed genes

- Identify genes with different levels in two conditions
- Examples
  - Highly expressed genes in cancer cells vs. health cells
- Filter method for selecting “feature” genes

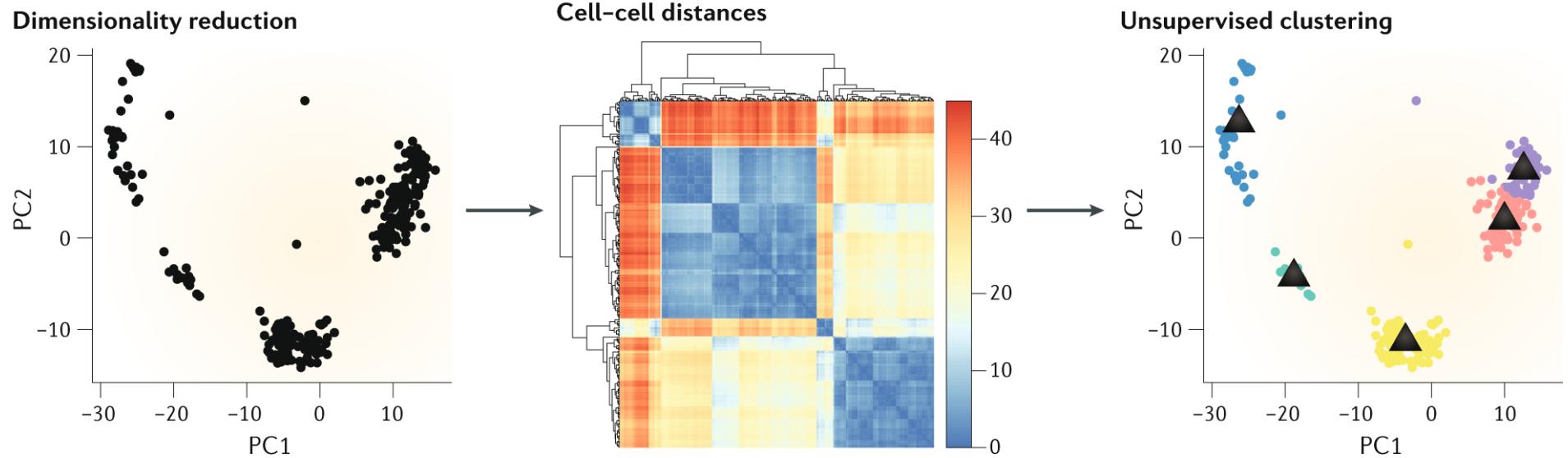
# What can we learn from a data matrix?



# The World of Machine Learning



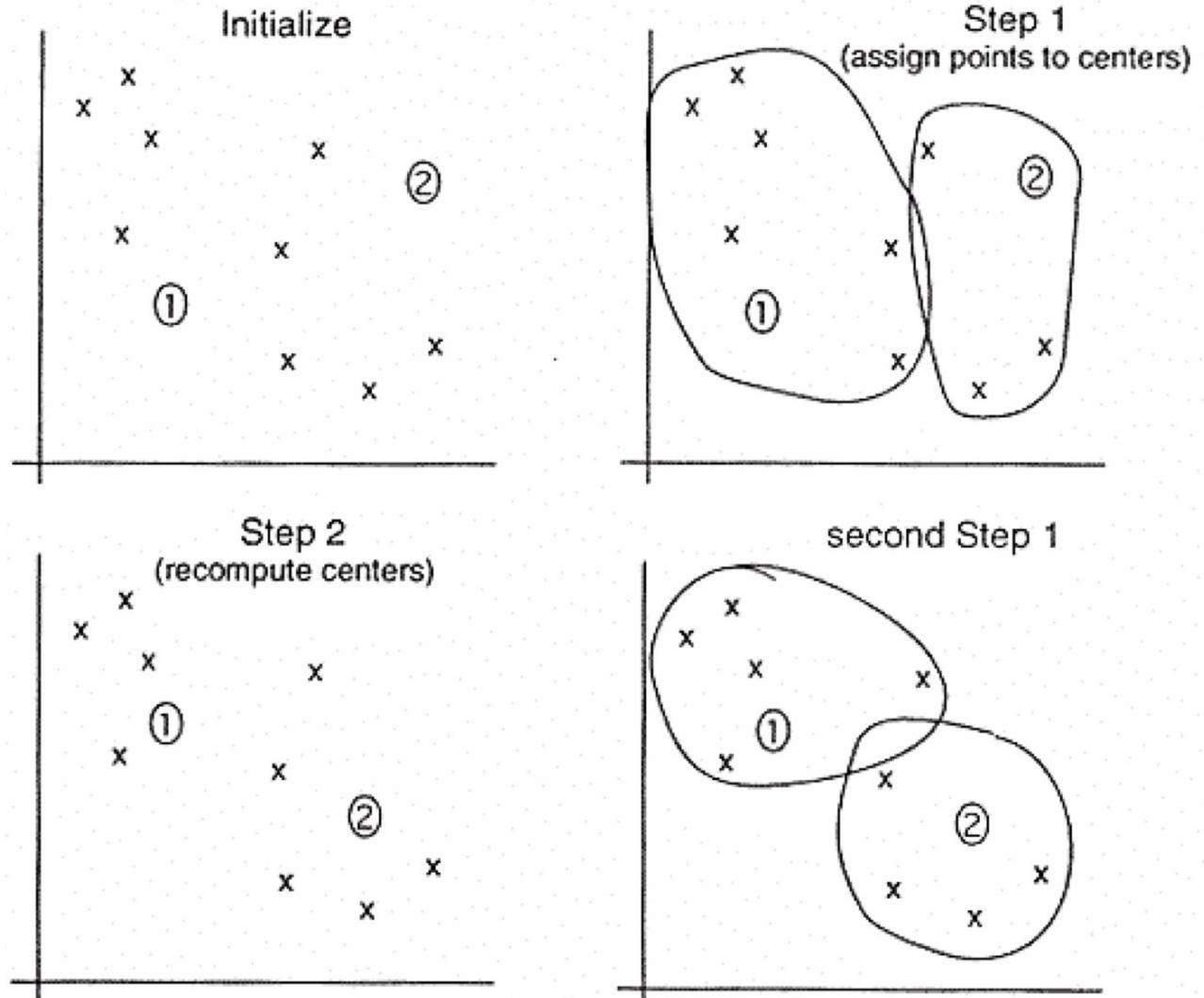
# Clustering cells



- Feature selection and dimensionality reduction extract the most informative genes and represented features from background noise, respectively
- Cell–cell distances are then calculated in the low dimensional space for clustering cells to clusters

# K-means

For example, PCA coefficients over PC1 and PC2



- 1) Pick  $K$  random points as putative cluster centers.
- 2) Group the points to be clustered by the center to which they are closest.
- 3) Then take the mean of each group and repeat, with the means now at the cluster center.
- 4) Stop when the centers stop moving.

# K-means: Setup

- $x_1, \dots, x_N$  are data points or vectors of observations
- Each observation (vector  $x_i$ ) will be assigned to one and only one cluster
- $C(i)$  denotes cluster number for the  $i^{\text{th}}$  observation
- Dissimilarity measure: Euclidean distance metric
- K-means minimizes within-cluster point scatter:

$$W(C) = \frac{1}{2} \sum_{k=1}^K \sum_{C(i)=k} \sum_{C(j)=k} \|x_i - x_j\|^2 = \sum_{k=1}^K N_k \sum_{C(i)=k} \|x_i - m_k\|^2$$

where

$m_k$  is the mean vector of the  $k^{\text{th}}$  cluster

$N_k$  is the number of observations in  $k^{\text{th}}$  cluster

# Within and Between Cluster Criteria

Let's consider total point scatter for a set of  $N$  data points:

$$T = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N d(x_i, x_j)$$

Distance between two points

$T$  can be re-written as:

$$\begin{aligned} T &= \frac{1}{2} \sum_{k=1}^K \sum_{C(i)=k} \left( \sum_{C(j)=k} d(x_i, x_j) + \sum_{C(j) \neq k} d(x_i, x_j) \right) \\ &= W(C) + B(C) \end{aligned}$$

Where,

$$W(C) = \frac{1}{2} \sum_{k=1}^K \sum_{C(i)=k} \sum_{C(j)=k} d(x_i, x_j)$$

Within cluster  
scatter

$$B(C) = \frac{1}{2} \sum_{k=1}^K \sum_{C(i)=k} \sum_{C(j) \neq k} d(x_i, x_j)$$

Between cluster  
scatter

If  $d$  is square Euclidean distance, then

$$W(C) = \sum_{k=1}^K N_k \sum_{C(i)=k} \|x_i - m_k\|^2$$

$$\text{and } B(C) = \sum_{k=1}^K N_k \|m_k - m\|^2$$

Grand mean

Minimizing  $W(C)$  is equivalent to maximizing  $B(C)$

# *K*-means Algorithm

- For a given cluster assignment  $C$  of the data points, compute the cluster means  $m_k$ :

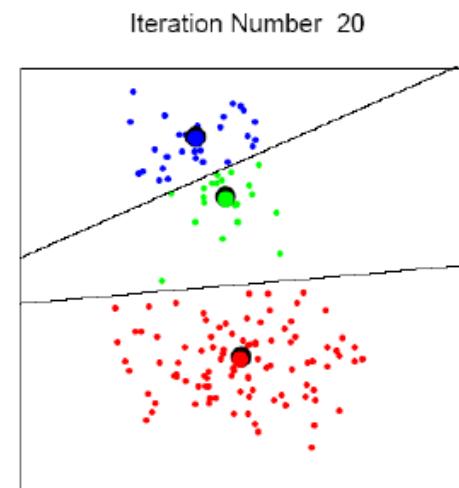
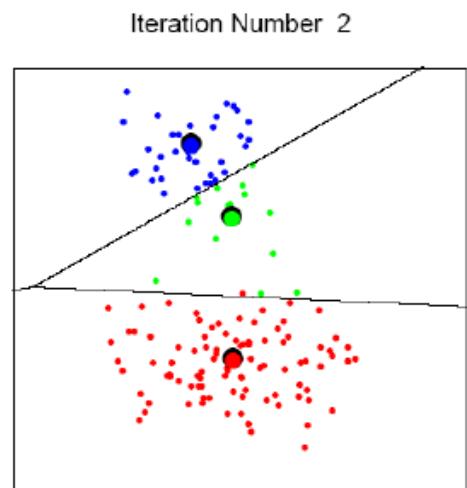
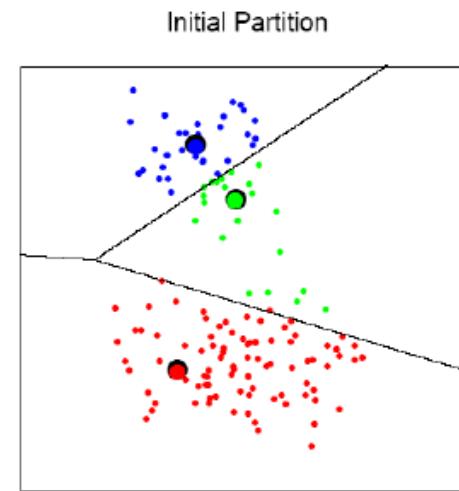
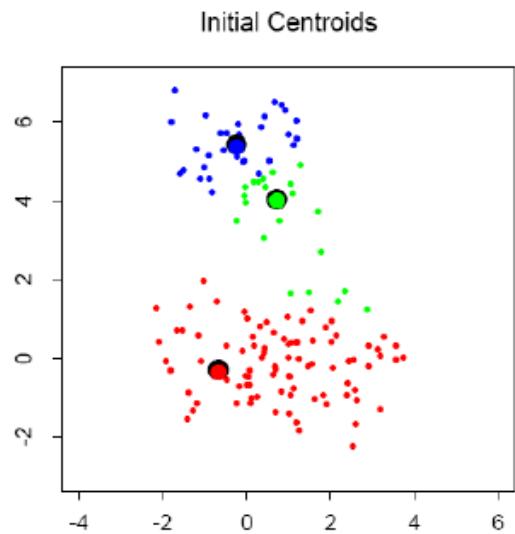
$$m_k = \frac{\sum_{i:C(i)=k} x_i}{N_k}, \quad k = 1, \dots, K.$$

- For a current set of cluster means, assign each observation as:

$$C(i) = \arg \min_{1 \leq k \leq K} \|x_i - m_k\|^2, \quad i = 1, \dots, N$$

- Iterate above two steps until convergence

# K-means clustering example



# *K*-means: summary

- Algorithmically, very simple to implement
- *K*-means converges, but it finds a local minimum of the cost function
- Works only for numerical observations
- *K* is a user input; alternatively BIC (Bayesian information criterion) or MDL (minimum description length) can be used to estimate *K*
- **Outliers can considerable trouble to *K*-means**

# *K*-medoids Clustering

- *K*-means is appropriate when we can work with Euclidean distances
- Thus, *K*-means can work only with numerical, quantitative variable types
- Euclidean distances do not work well in at least two situations
  - Some variables are categorical
  - Outliers can be potential threats
- A general version of *K*-means algorithm called *K*-medoids can work with any distance measure
- *K*-medoids clustering is computationally more intensive

# *K*-medoids Algorithm

- Step 1: For a given cluster assignment  $C$ , find the observation in the cluster minimizing the total distance to other points in that cluster:

$$i_k^* = \arg \min_{\{i: C(i)=k\}} \sum_{C(j)=k} d(x_i, x_j).$$

- Step 2: Assign  $m_k = x_{i_k^*}, k = 1, 2, \dots, K$
- Step 3: Given a set of cluster centers  $\{m_1, \dots, m_K\}$ , minimize the total error by assigning each observation to the closest (current) cluster center:

$$C(i) = \arg \min_{1 \leq k \leq K} d(x_i, m_k), i = 1, \dots, N$$

- Iterate steps 1 to 3

# *K*-medoids Summary

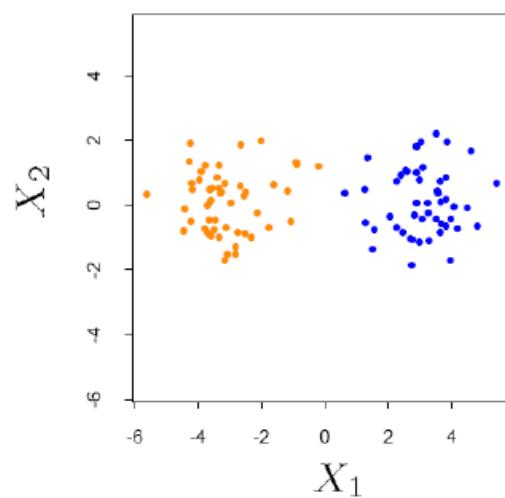
- Generalized *K*-means
- Computationally much costlier than *K*-means
- Apply when dealing with categorical data
- Apply when data points are not available, but only pair-wise distances are available
  - Kernel functions
- Converges to local minimum

# Choice of $K$ ?

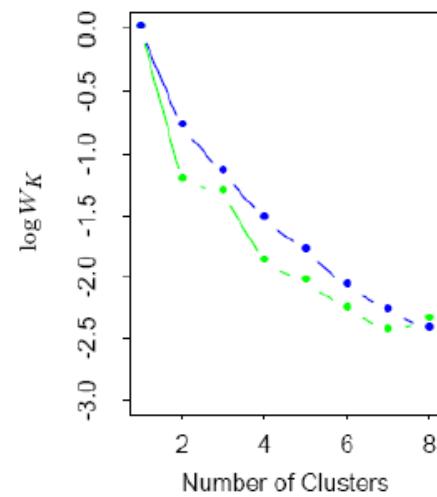
- Can  $W_K(C)$ , i.e., the within cluster distance as a function of  $K$  serve as any indicator?
- Note that  $W_K(C)$  decreases monotonically with increasing  $K$ . That is the within cluster scatter decreases with increasing centroids.
- Instead look for gap statistics (successive difference between  $W_K(C)$ ):

$$\{W_K - W_{K+1} : K < K^*\} \gg \{W_K - W_{K+1} : K \geq K^*\}$$

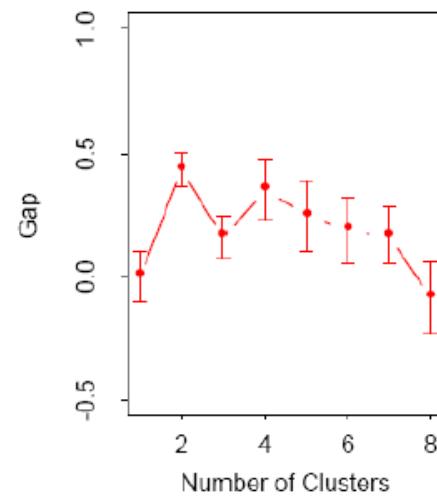
# Choice of $K$ ...



Data points simulated  
from two pdfs



$\log(W_K)$  curve

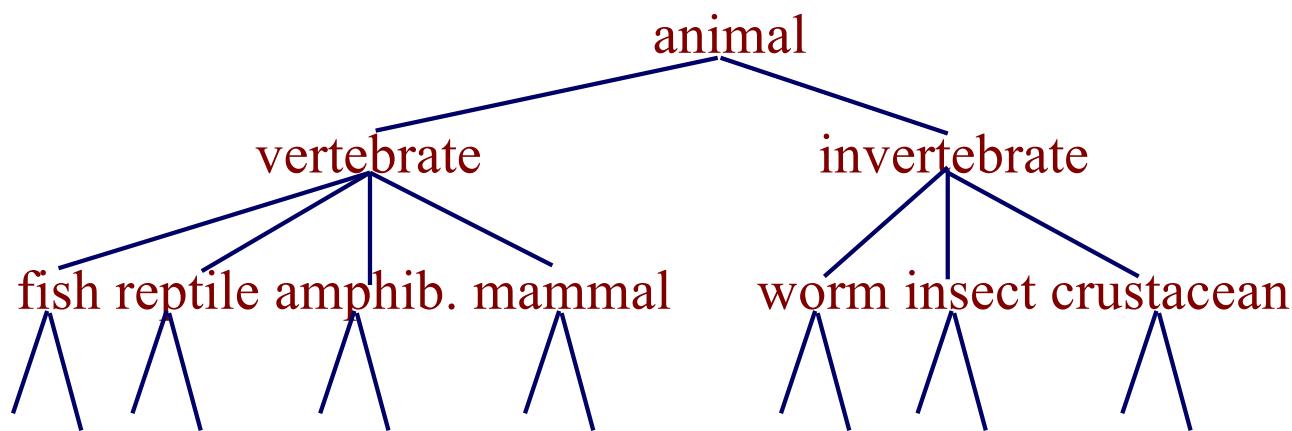


Gap curve

This is essentially a **visual heuristic**

# Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of documents.



How could you do this with K-means?

# Hierarchical Clustering

- **Agglomerative (bottom-up):**
  - Start with each gene being a single cluster.
  - Eventually all genes belong to the same cluster.
- **Divisive (top-down):**
  - Start with all genes belong to the same cluster.
  - Eventually each gene forms a cluster on its own.
  - Could be a recursive application of K-means like algorithms
- Does not require the number of clusters  $K$  in advance
- Needs a termination/readout condition

# Hierarchical Agglomerative Clustering (HAC)

- Start with each gene in a separate cluster
  - then repeatedly joins the closest pair of clusters, until there is only one cluster.
- The history of merging forms a tree or hierarchy.

How to measure distance of clusters??

# Distance Metrics

- properties of metrics

$$\text{dist}(x_i, x_j) \geq 0 \quad (\text{non-negativity})$$

$$\text{dist}(x_i, x_j) = 0 \text{ if and only if } x_i = x_j \quad (\text{identity})$$

$$\text{dist}(x_i, x_j) = \text{dist}(x_j, x_i) \quad (\text{symmetry})$$

$$\text{dist}(x_i, x_j) \leq \text{dist}(x_i, x_k) + \text{dist}(x_k, x_j) \quad (\text{triangle inequality})$$

- some distance metrics

Manhattan  $\text{dist}(x_i, x_j) = \sum_e |x_{i,e} - x_{j,e}|$

Euclidean  $\text{dist}(x_i, x_j) = \sqrt{\sum_e (x_{i,e} - x_{j,e})^2}$

$e$  ranges over the individual measurements for  $x_i$  and  $x_j$

# Correlation distance

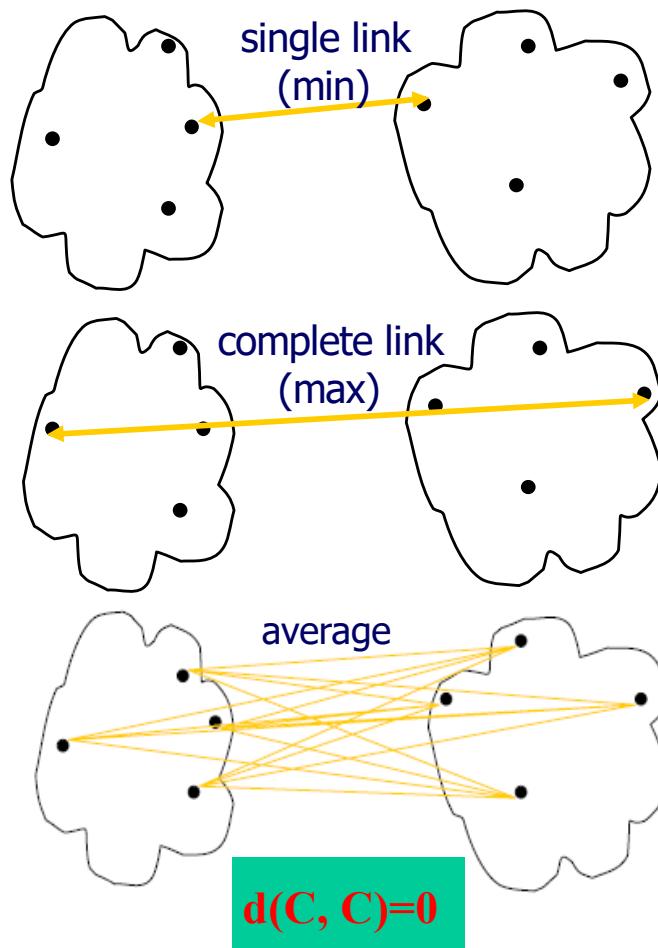
- Correlation distance

$$r_{xy} = \frac{Cov(X, Y)}{\sqrt{Var(X) \cdot Var(Y)}}$$

- Cov(X, Y) stands for covariance of X and Y
  - degree to which two different variables are related
- Var(X) stands for variance of X
  - measurement of a sample differ from their mean

# 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 characterized by a single continuous 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

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

Complete link

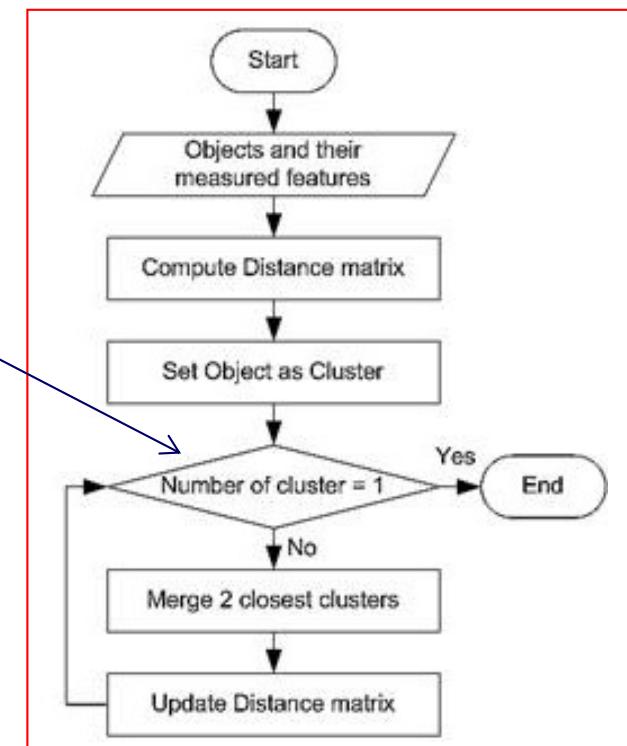
$$\begin{aligned} \text{dist}(C_1, C_2) &= \max\{d(a,c), d(a,d), d(a,e), d(b,c), d(b,d), d(b,e)\} \\ &= \max\{3, 4, 5, 2, 3, 4\} = 5 \end{aligned}$$

Average

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

# Agglomerative Algorithm

- Convert all features (e.g., genes) into a distance matrix
- Set each gene as a cluster (N genes -> N clusters at the beginning)
- Repeat until number of cluster (or known # of clusters)
  - Merge two closest clusters
  - Update “distance matrix”



# Bottom-Up Hierarchical Clustering

given: a set  $X = \{x_1 \dots x_n\}$  of instances

for  $i := 1$  to  $n$  do

$C_i := \{x_i\}$                   // each object is initially its own cluster, and a leaf in tree

$C := \{C_1 \dots C_n\}$

$j := n$

    while  $|C| > 1$

$j := j + 1$

$(c_a, c_b) := \underset{(c_u, c_v)}{\operatorname{argmin}} \operatorname{dist}(c_u, c_v)$                   // find least distant pair in C

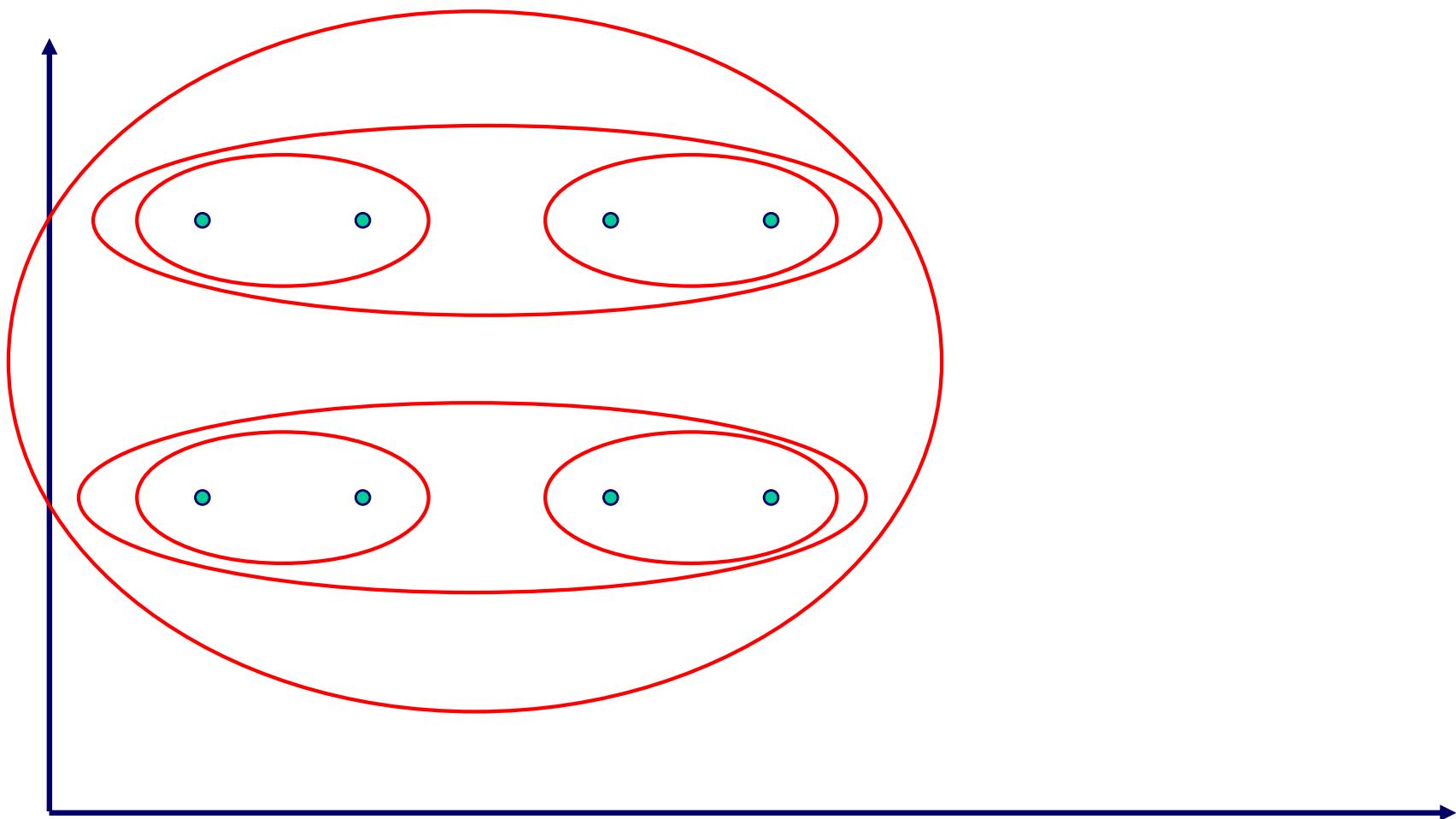
$c_j = c_a \cup c_b$                   // create a new cluster for pair

        add a new node  $j$  to the tree joining  $a$  and  $b$

$C := C - \{c_a, c_b\} \cup \{c_j\}$

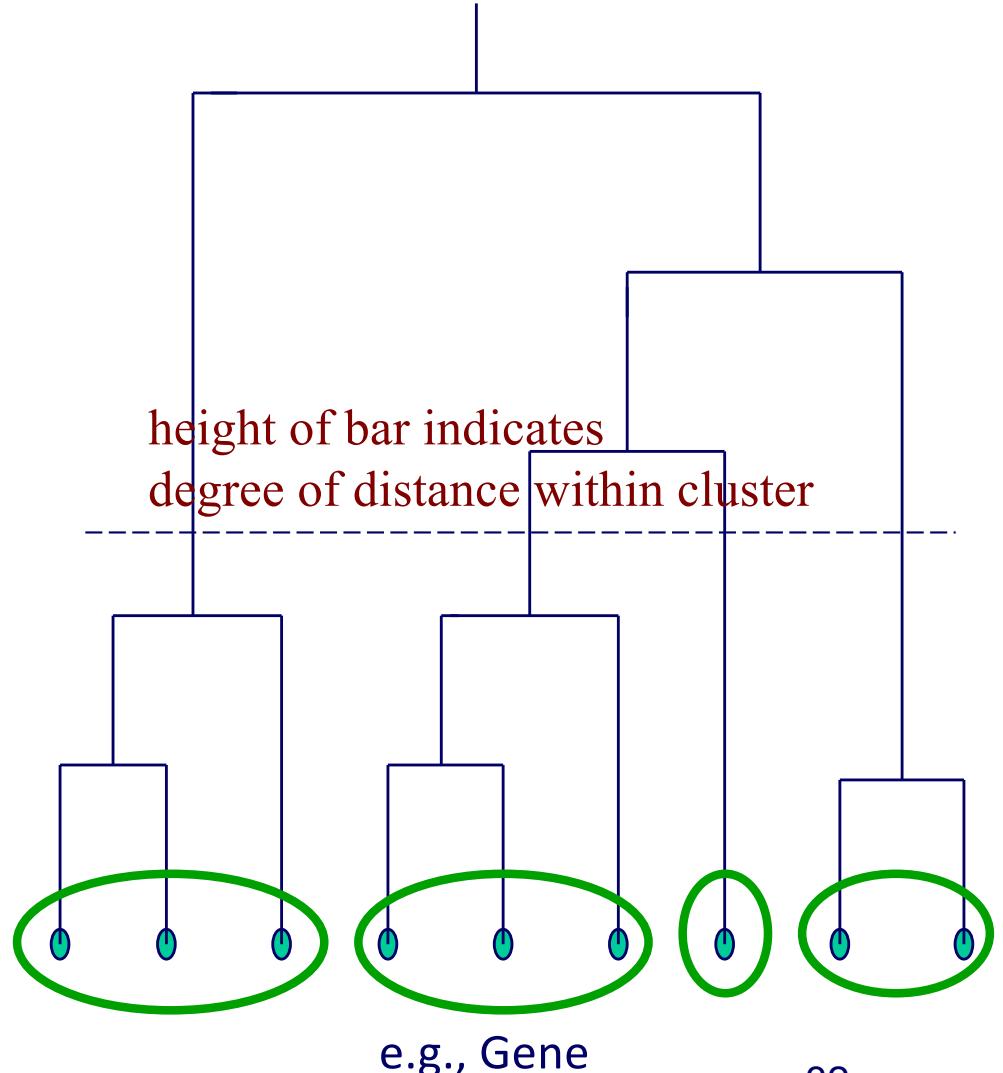
return tree with root node  $j$

# Single Link Example

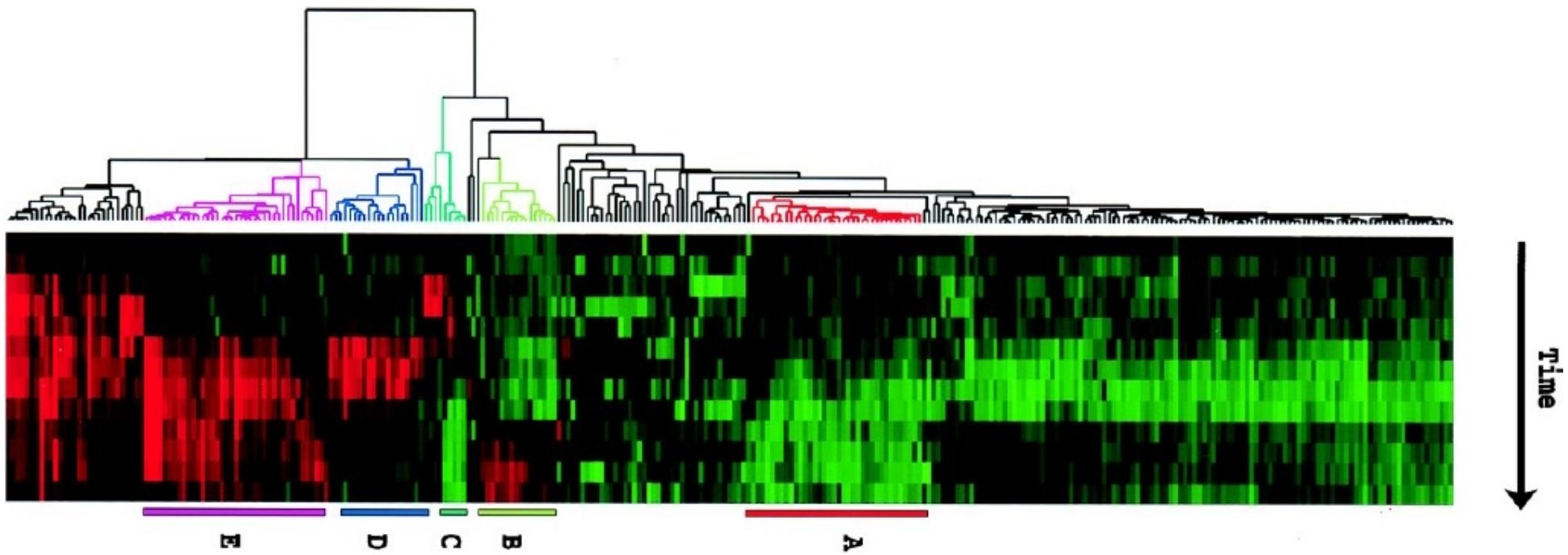


# Dendrogram: Hierarchical Clustering

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



# Hierarchical Clustering of Expression Data



# Partitioning or Hierarchical?

## Partitioning:

### – Advantages

- Optimal for certain criteria.
- Genes automatically assigned to clusters

### – Disadvantages

- Need initial  $k$ ;
- Often slow computation.
- All genes are forced into a cluster.

## Hierarchical

### – Advantages

- Faster computation.
- Visual.

### – Disadvantages

- Unrelated genes are eventually joined
- Rigid, cannot correct later for erroneous decisions made earlier.
- Hard to define clusters.

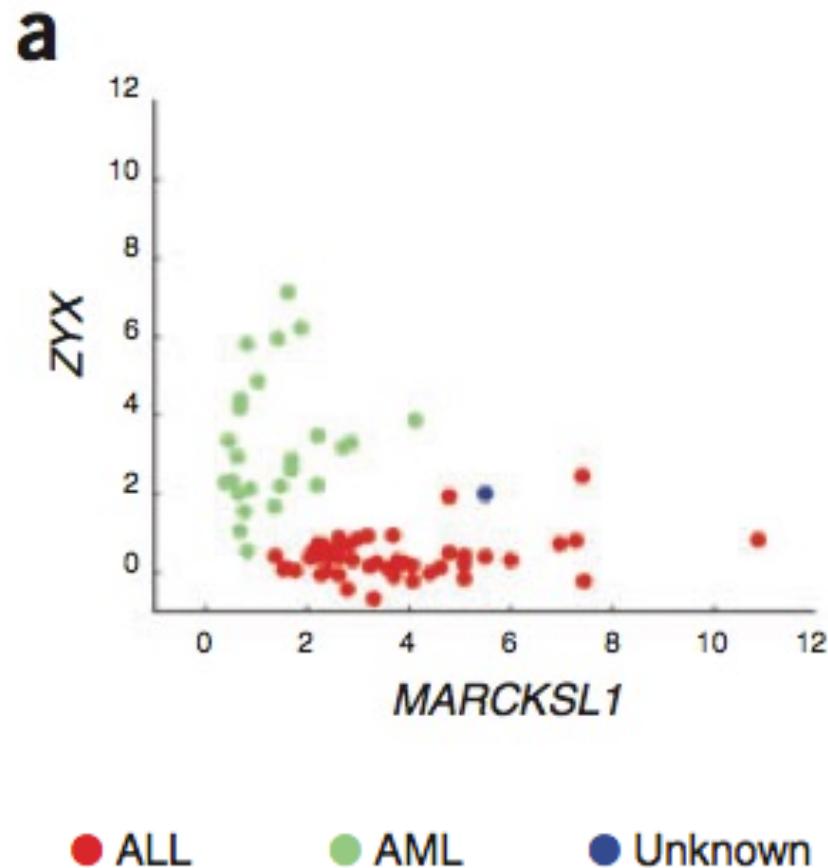
# Reading list

- A. K. Jain and M. N. Murty and P. J. Flynn, Data clustering: a review, ACM Computing Surveys, 31:3, pp. 264 - 323, 1999.
- T. R. Golub et. al, Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring, Science, 286:5439, pp. 531 – 537, 1999.
- Gasch,A.P. and Eisen,M.B. (2002) Exploring the conditional coregulation of yeast gene expression through fuzzy  $k$ -means clustering. *Genome Biol.*, 3, 1–22.
- M. Eisen et. al, Cluster Analysis and Display of Genome-Wide Expression Patterns. *Proc Natl Acad Sci U S A* 95, 14863-8, 1998.

# Support Vector Machines

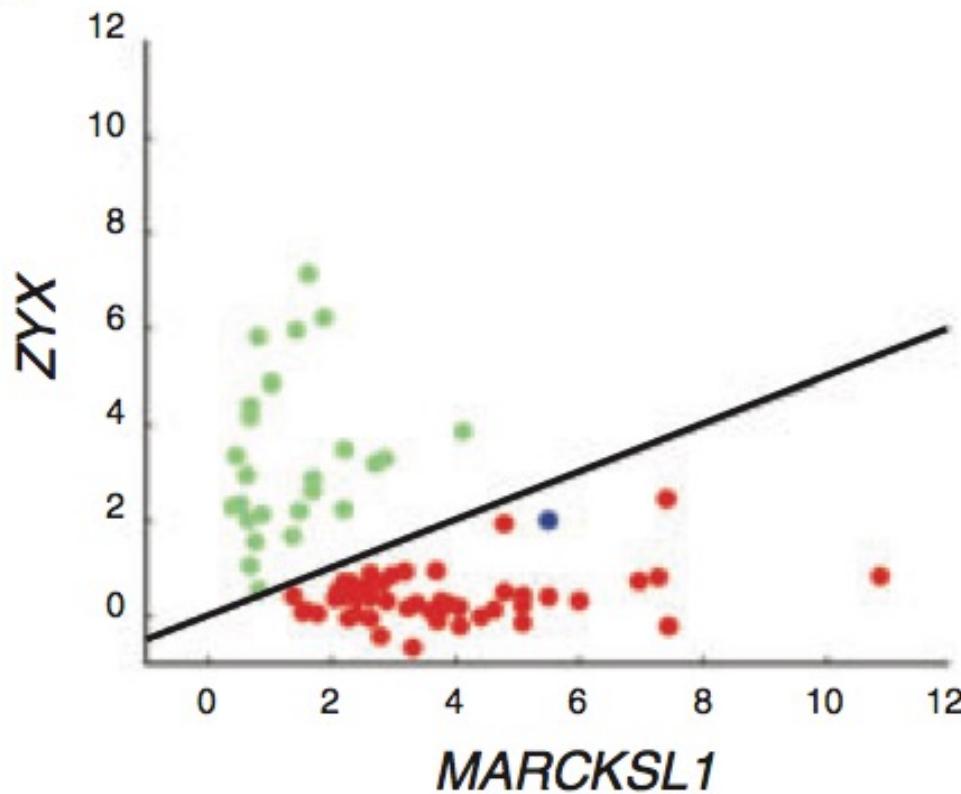
- A very powerful tool for classifications
- Example Applications:
  - Text categorization
  - Image classification
  - Spam email recognition, etc
- It has also been successfully applied in many biological problems:
  - Disease diagnosis
  - Automatic genome functional annotation
  - Prediction of protein-protein interactions
  - and more...

- Example: Leukemia patient classification



ALL: acute lymphoblastic leukemia

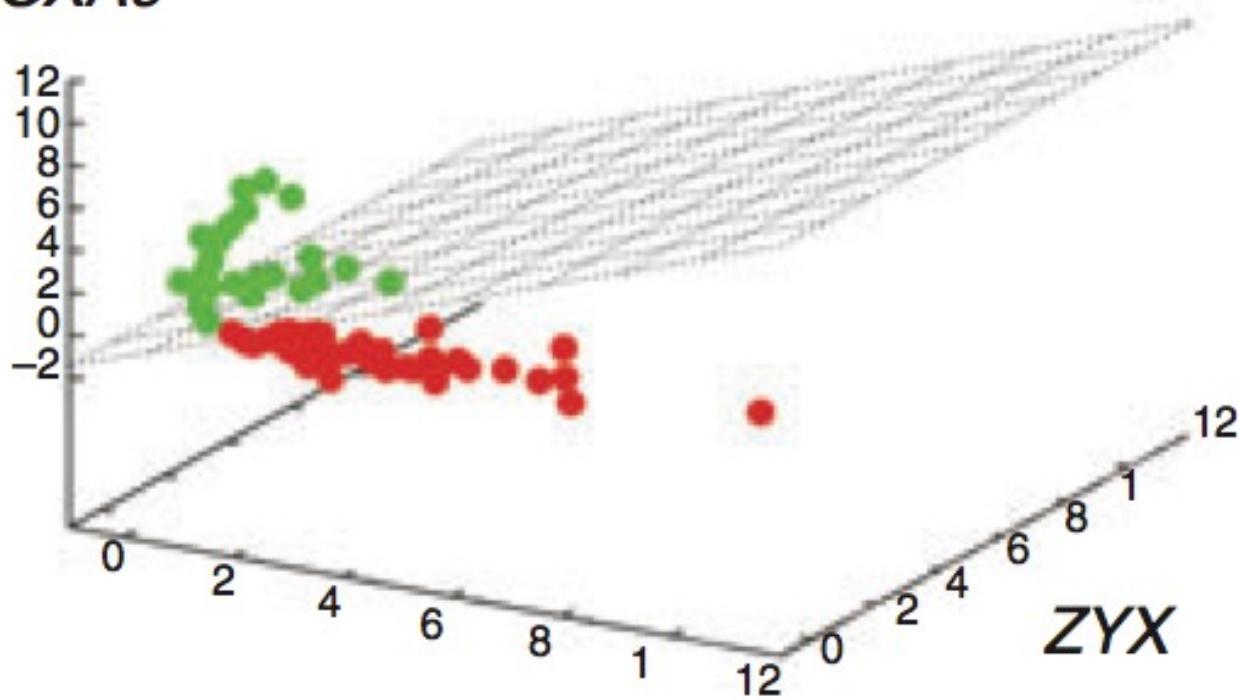
AML: acute myeloid leukemia

**b**

- A simple line suffices to separate the expression profiles of ALL and AML

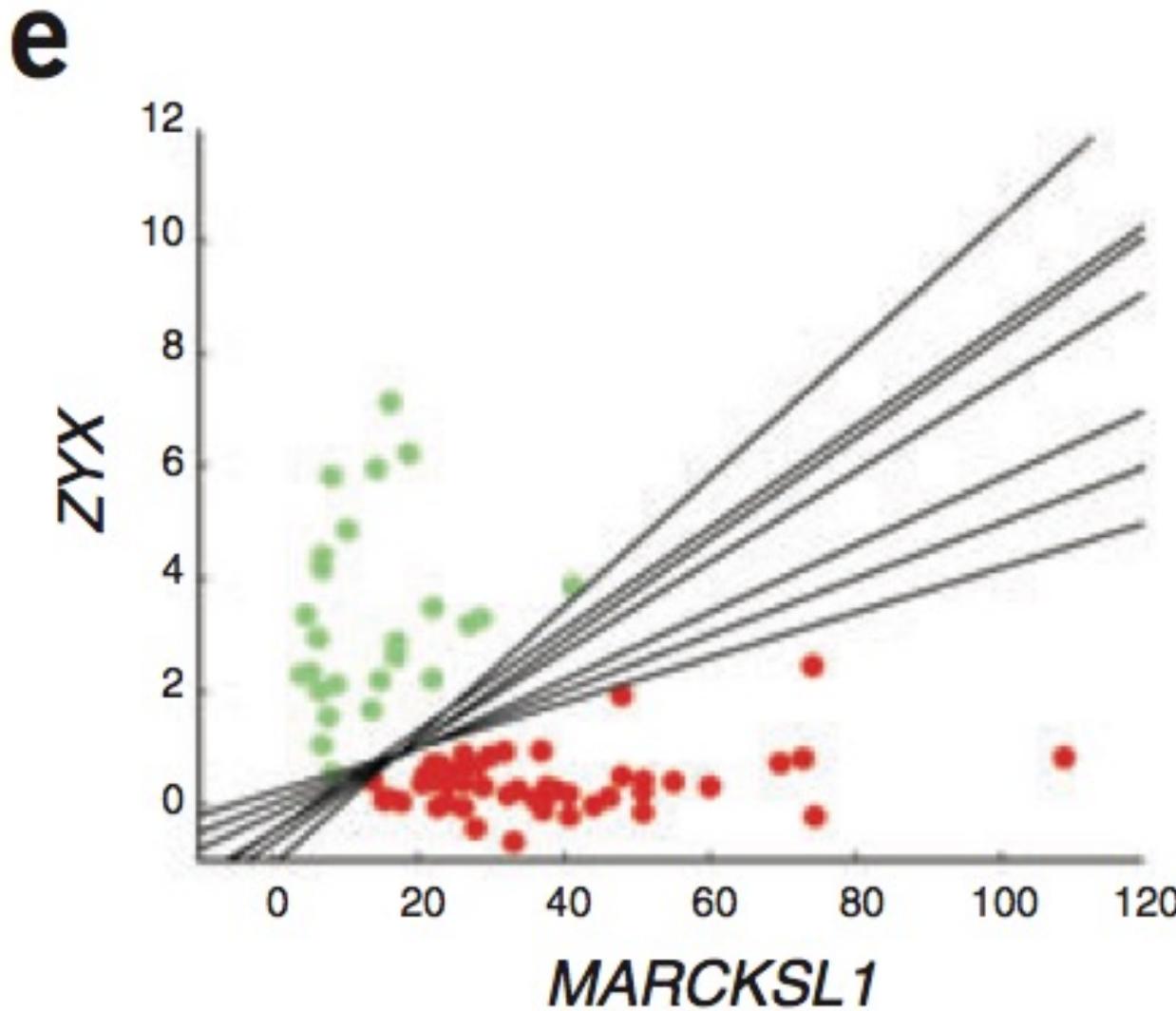
**d**

*HOXA9*



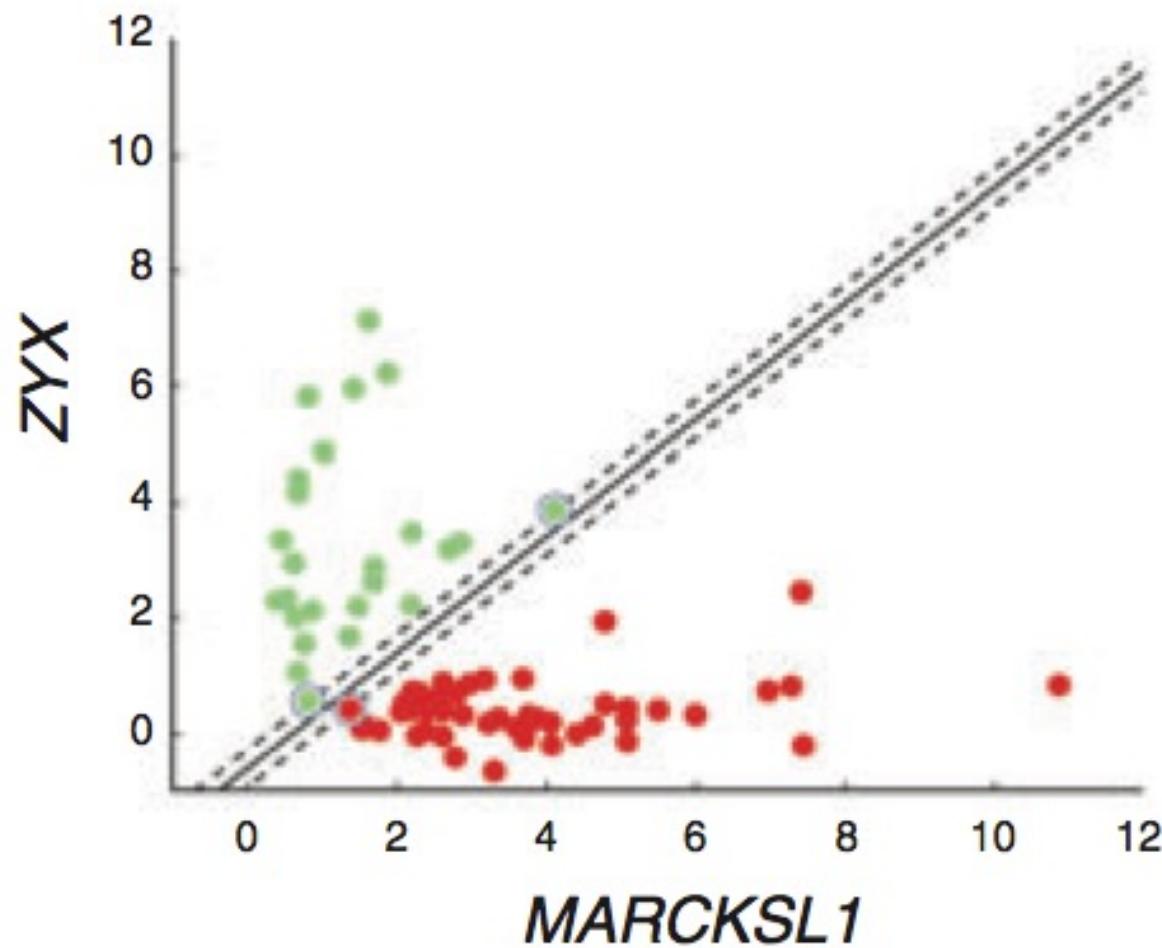
*MARCKSL1*

- In the case of more than two genes, a line generalizes to a plane or “hyperplane”.
- For generality, we refer to them all as “hyperplane”



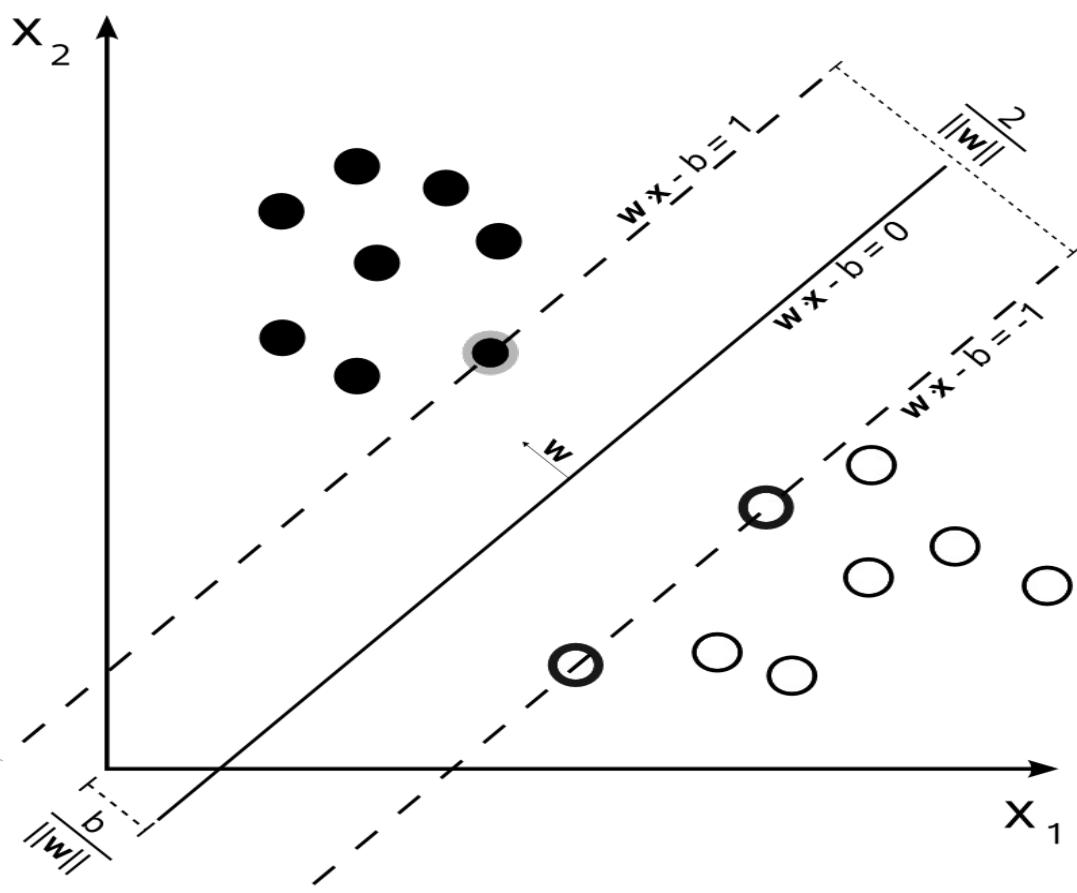
- Is there a “**best**” line?

**f**



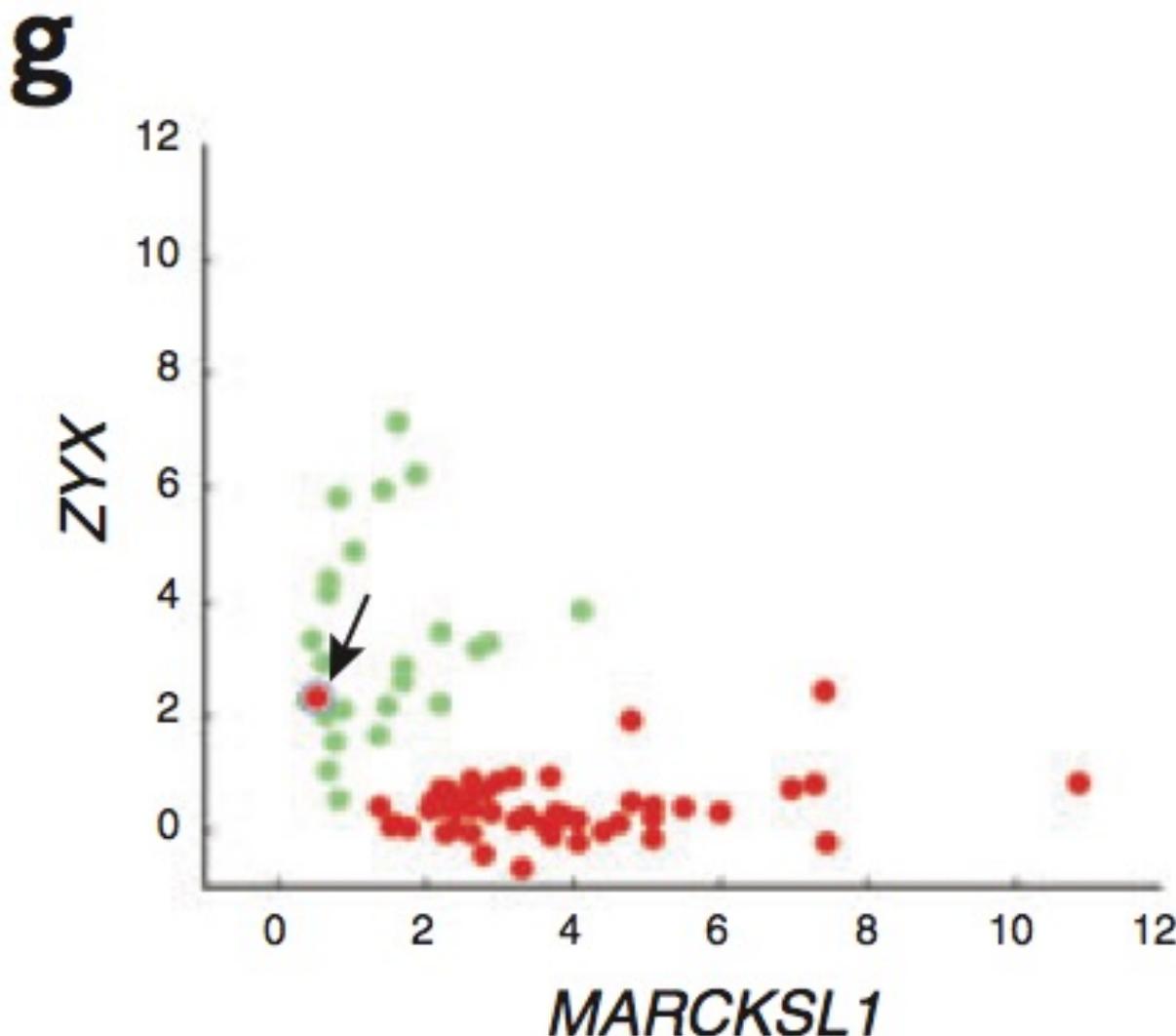
- The maximum margin hyperplane

- Denote each data point as  $(x_i, y_i)$
- $x_i$  is a vector of the expression profiles
- $y_i = -1$  or  $1$ , which labels the class
- A hyperplane can be represented as:  $w^*x + b = 0$
- The margin-width equals to:  $2/\|w\|$ ,  $\|w\| = \sqrt{w \cdot w}$

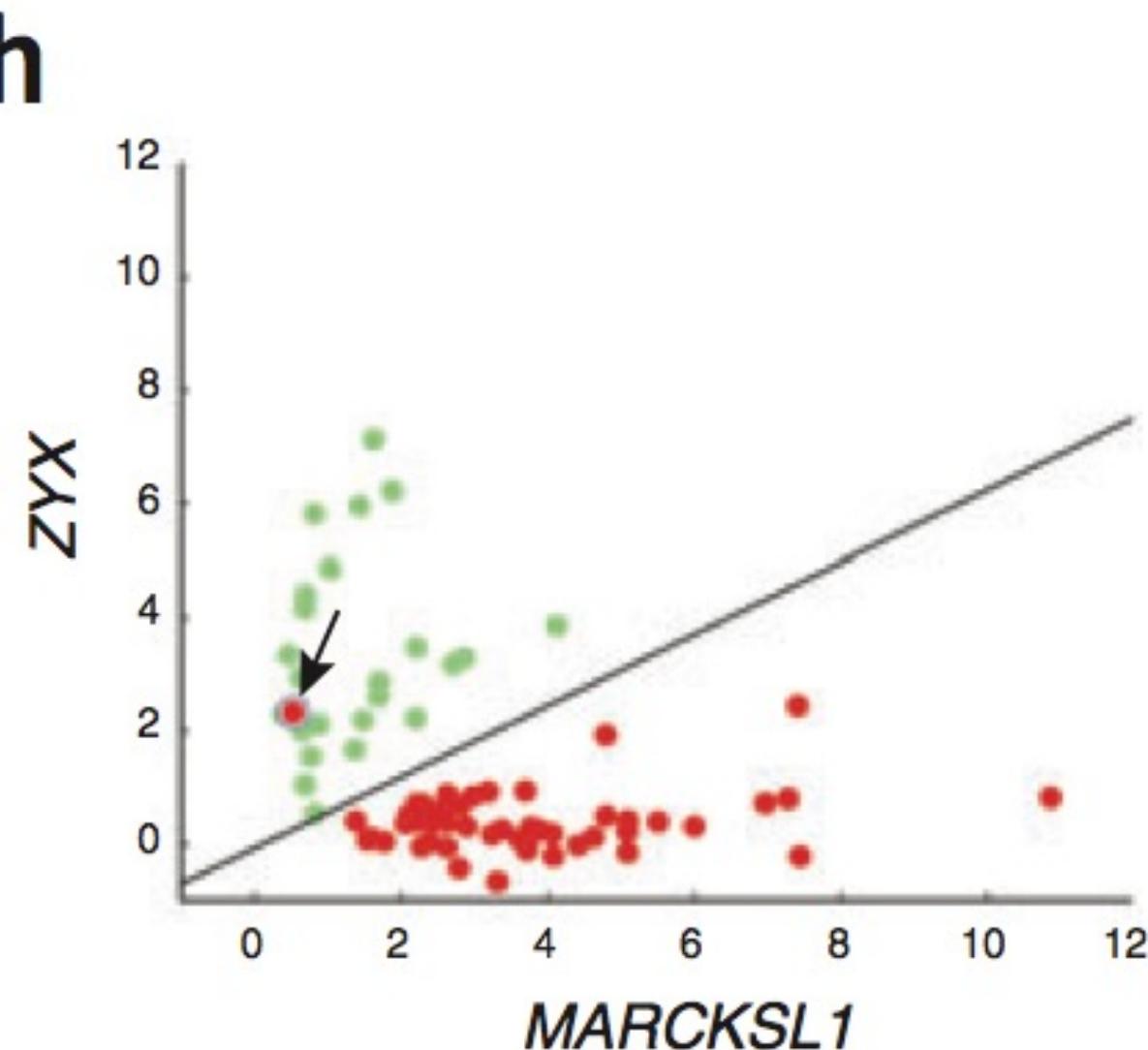


- Find a hyperplane such that:
  - No data points fall between the lines  $w \cdot x + b = -1$  and  $w \cdot x + b = +1$
  - The margin  $2/\|w\|$  is maximized
- Mathematically,
  - Minimize<sub>w,b</sub>  $\frac{1}{2}*\|w\|^2$ , subject to:
  - for  $y_i = 1$ ,  $w \cdot x_i + b \geq 1$
  - for  $y_i = -1$ ,  $w \cdot x_i + b \leq -1$
  - Combining them, for any i,  $y_i(w \cdot x_i + b) \geq 1$
- The solution expresses  $w$  as a linear combination of the  $x_i$
- Assuming that the data points from two classes are always easily **linearly separable**. But that's not always the case

- What if...



- Allow a few anomalous data points

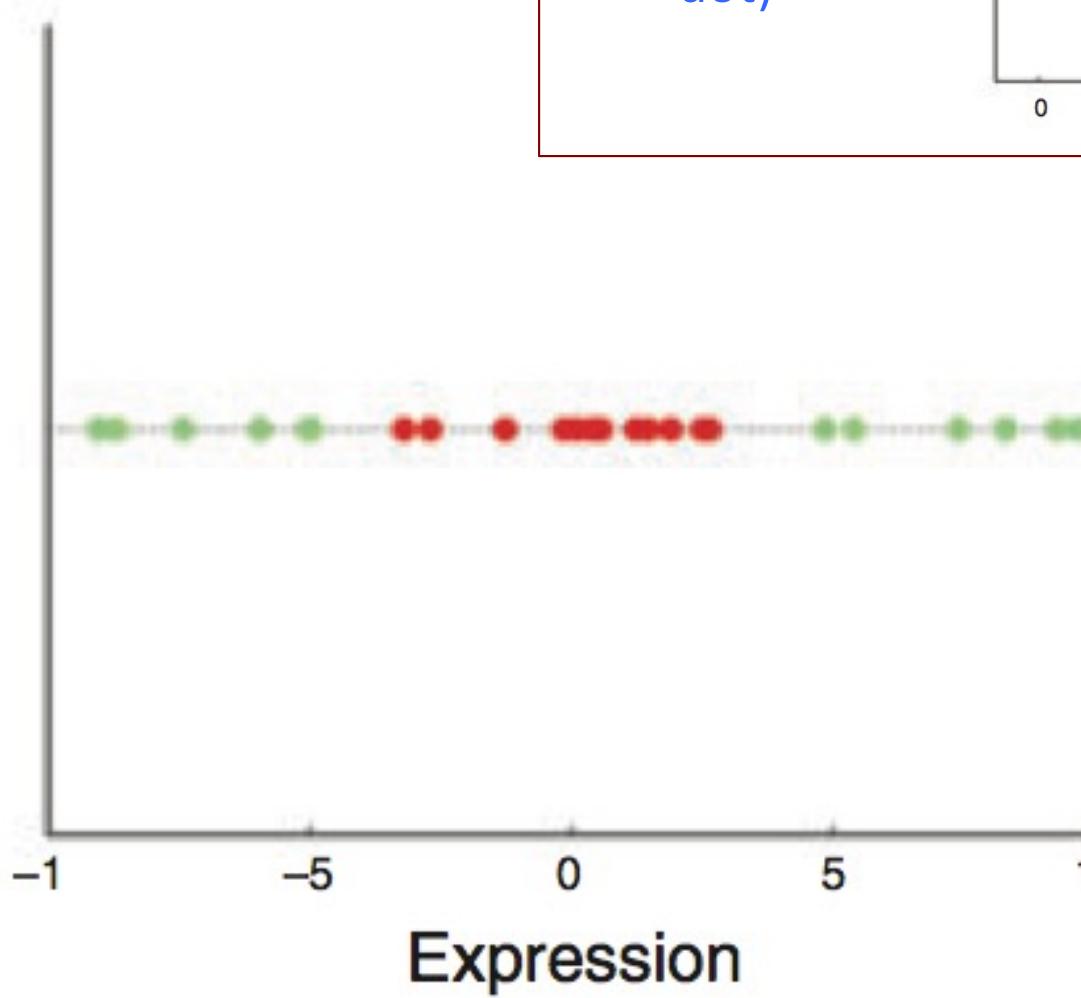
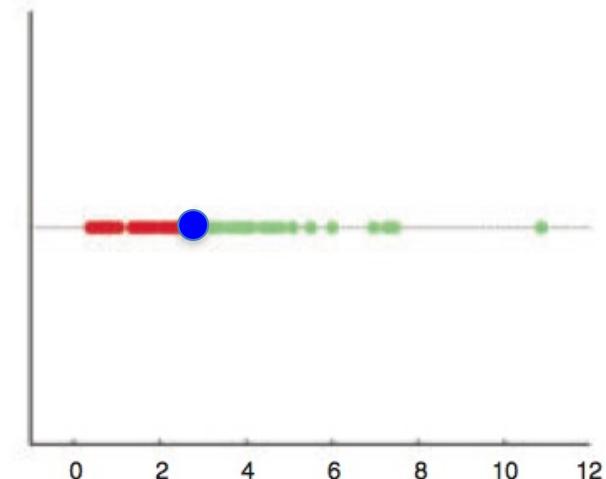


- The soft-margin SVM

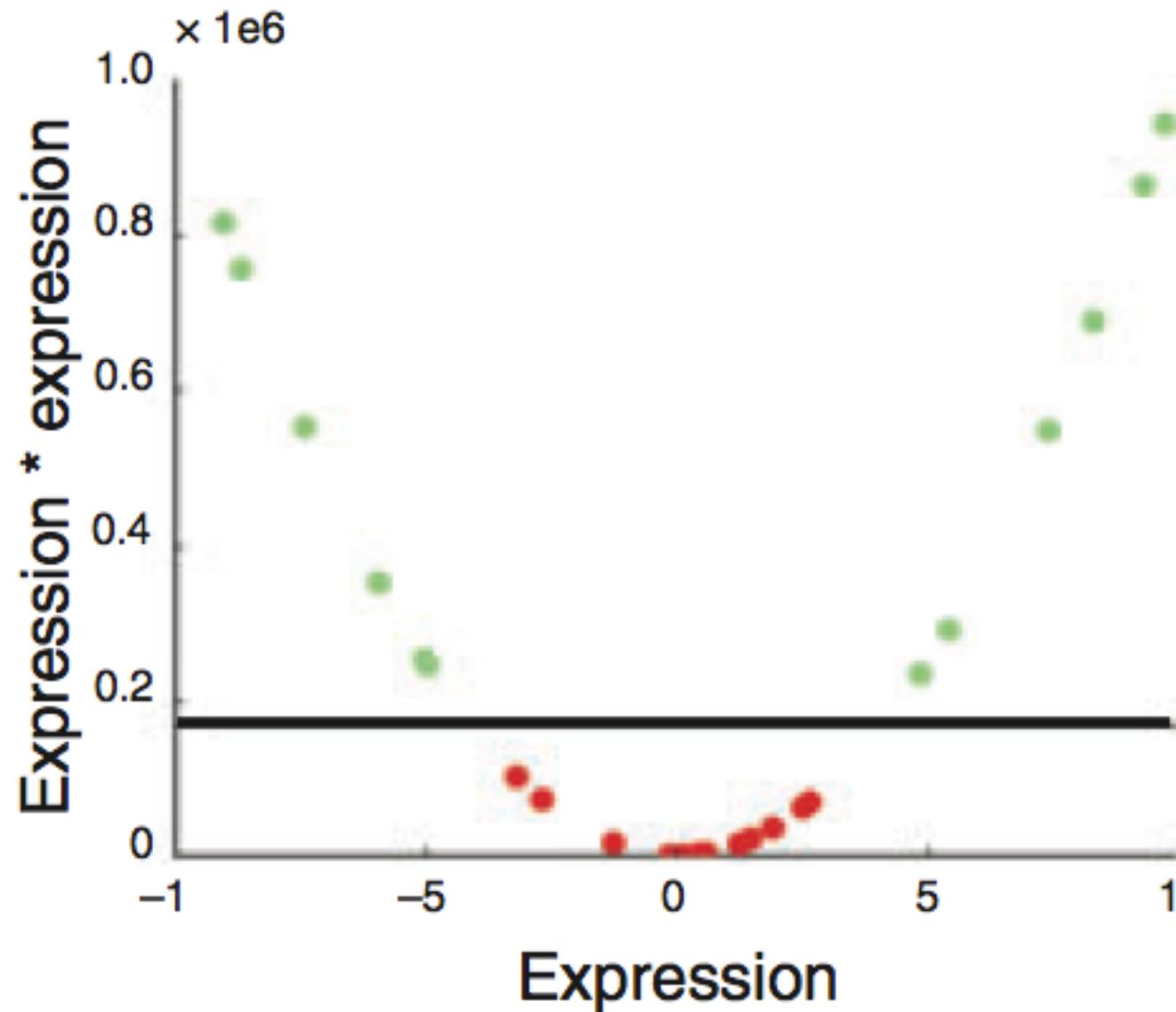
- minimize $\frac{1}{2} \|w\|^2 + C \sum_i s_i$
- subject to, for any  $i$ ,  $y_i(w \bullet x_i + b) \geq 1 - s_i, s_i \geq 0$
- $S_i$  are the slack variables
- $C$  controls the number of tolerated misclassifications
  - (It's effectively a regularization parameter on model complexity)
- A small  $C$  would allow more misclassifications
- A large  $C$  would discourage misclassifications
- Note that even when the data points are linearly separable, one can still introduce the slack variables to pursue a **larger separation margin**

- Are linear separating hyperplanes enough?

Yes  
(by a 1D-  
hyperplane  
= dot)



- Transform  $(x_i)$  into  $(x_i, x_i^2)$



# Non-linear SVM

- In some cases (e.g. the above example), even **soft-margin** cannot solve the non-separable problem
- Generally speaking, we can apply **some function** to the original data points so that different classes become **linearly separable** (maybe with the help of soft-margin)
  - In the above example, the function is  $f(x) = (x, x^2)$
- The **most import trick** in SVM: to allow for the transformation, we only need to define the “**kernel function**”,  $k(x_i, x_j) = f(x_i) \cdot f(x_j)$ 
  - e.g., a polynomial kernel used in above example

# Solving SVM

- Formulation of SVM using Lagrangian multipliers

$$\text{Minimize} \frac{\|w\|^2}{2} + \sum_i \alpha_i (1 - y_i(w^T x_i + b))$$

- The dual formulation of SVM can be expressed as:

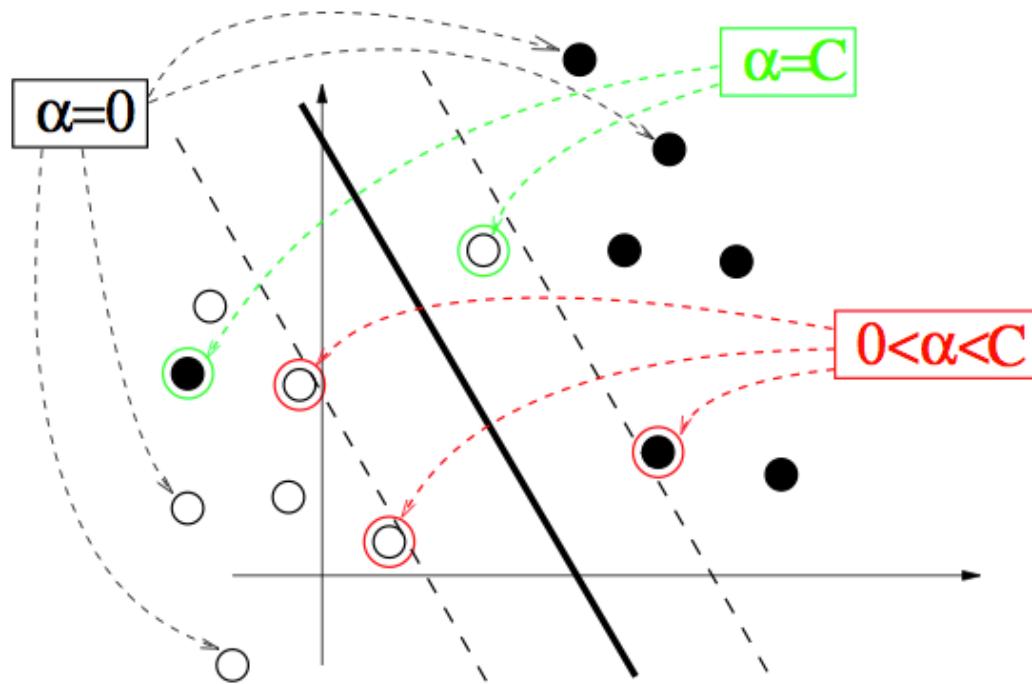
$$\text{Maximize}_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j x_i \bullet x_j , \text{ subject to}$$

$$\sum_i y_i \alpha_i = 0, 0 \leq \alpha_i \leq C \quad \text{no } w \text{ and } b \text{ now}$$

- The “Kernel”:  $x_i \bullet x_j$  can be replaced by more sophisticated **kernel functions**:

$$k(x_i, x_j) = f(x_i) \bullet f(x_j)$$

# Support vectors



- The  $x_i$  for which  $\alpha_i > 0$  are called **support vectors**
- They fall between or right on the separating margins

# Tricks for solving SVM

- Finding optimal  $w$  &  $b$  can be replaced by finding optimal "Lagrange multipliers"  $\alpha_i$ 
  - One only optimizes using the product of  $x_i^*x_j$ , now expressing the solution in terms of positive  $\alpha_i$  for  $x_i$  that function as support vectors
- Non-linear SVM  $x_i^*x_j$  is replaced by  $f(x_i)^*f(x_j)$ , so you don't need to know  $f(x_i)$  itself only the product
  - *Kernel trick*:  $f(x_i)^*f(x_j)$  is just replaced by  $k(x_i, x_j)$ . That is, one only has to know the “distance” between  $x_i$  &  $x_j$  in the high-dimensional space -- not their actual representation

# Kernel functions

- Polynomial kernel:
  - $k(x_i, x_j) = (x_i \bullet x_j + a)^d$
  - $a = 1$  (inhomogeneous) or  $0$  (homogenous)
  - $d$  controls the **degree** of polynomial and henceforth the **flexibility** of the classifier
  - degenerates to linear kernel when  $a = 0$  and  $d = 1$
- Gaussian kernel:
  - $k(x_i, x_j) = (-1/\sigma \|x_i - x_j\|^2)$
  - $\sigma$  controls the **width** of the Gaussian and plays a similar role as  $d$  in the polynomial kernels

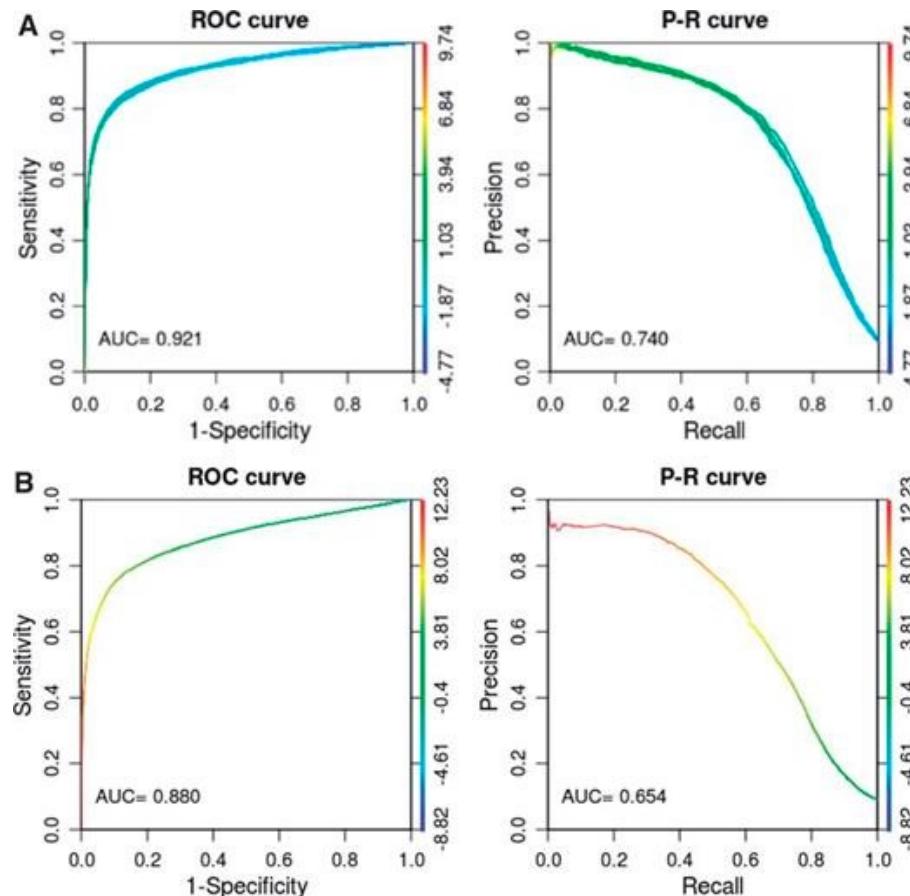
# Kernel functions in computational biology

- "Distance" even for non-vector biological data
  - Protein-protein interactions
  - DNA binding
  - Ben-Hur et al., Support Vector Machines and Kernels for Computational Biology, PLoS Comp. Bio., 2008
- For example, "Spectrum kernels" for sequences
  - k-spectrum of a sequence  $x$  is all possible k-length subsequence
  - Map the sequence to counts on k-spectrum  $c(x)$
  - Spectrum kernel  $K_k(x,y) = \langle c(x), c(y) \rangle$
  - Leslie et al., PSB, 2002

# kmer-SVM for predicting regulatory sequence features

- Fletez-Brant et al., NAR, 2013
- For example, ESRRB binding sites

kmer-SVM



PWM

| 6-mers                 | Revcomp | SVM Scores |
|------------------------|---------|------------|
| <b>Positive 6-mers</b> |         |            |
| AAGGTC                 | GACCTT  | 10.05      |
| AGGTCA                 | TGACCT  | 8.47       |
| ACCTTG                 | CAAGGT  | 5.33       |
| AGGTCG                 | CGACCT  | 5.17       |
| GGTCAA                 | TTGACC  | 4.01       |
| <b>Negative 6-mers</b> |         |            |
| GCAATA                 | TATTGC  | -2.05      |
| TGACCA                 | TGGTCA  | -3.33      |
| AAGGTA                 | TACCTT  | -4.23      |
| AGACCT                 | AGGTCT  | -4.55      |
| AGGTCC                 | GGACCT  | -4.98      |

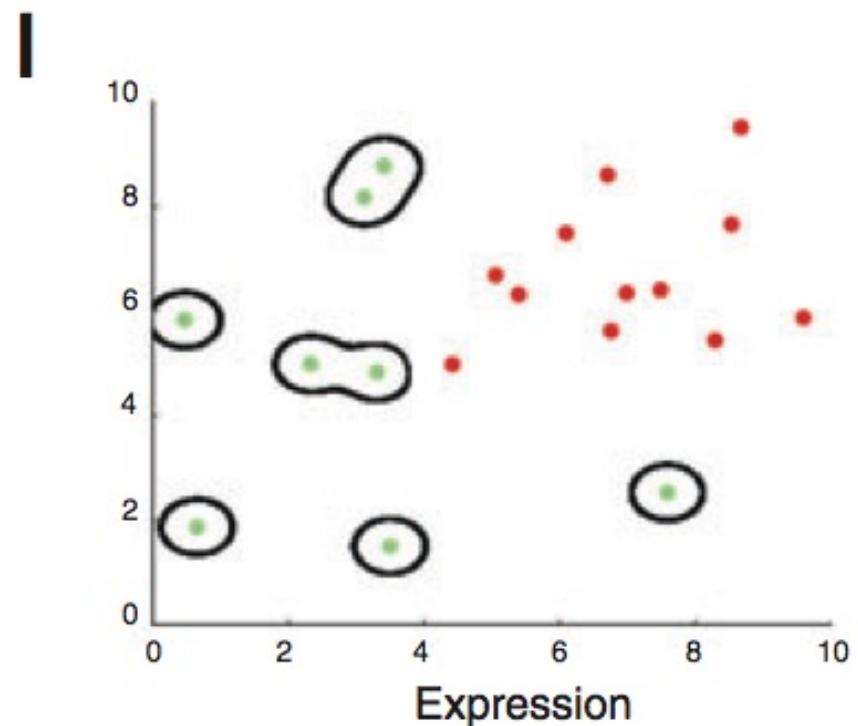
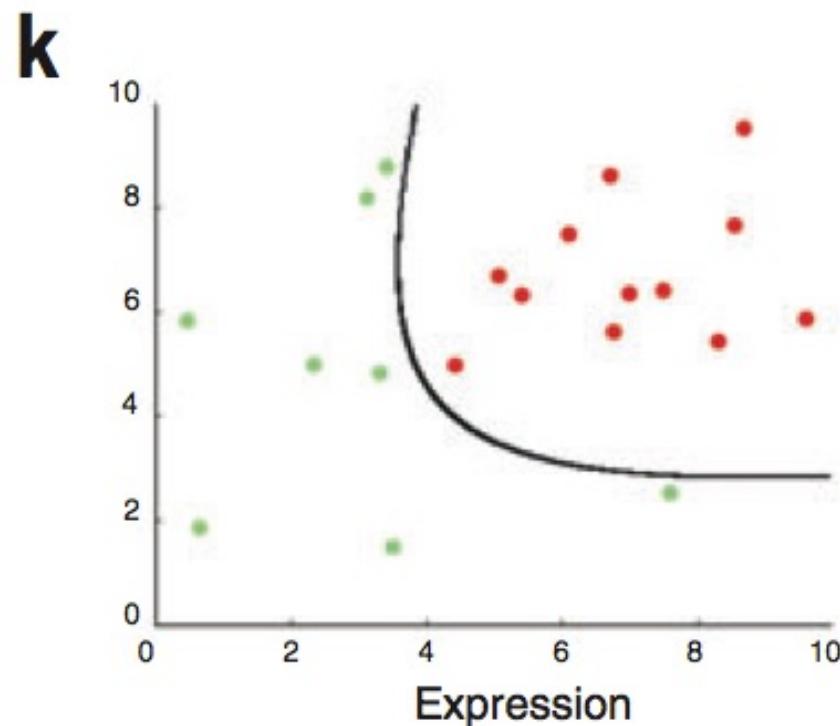
D

Esrrb

# Avoid over-fitting by kernel functions

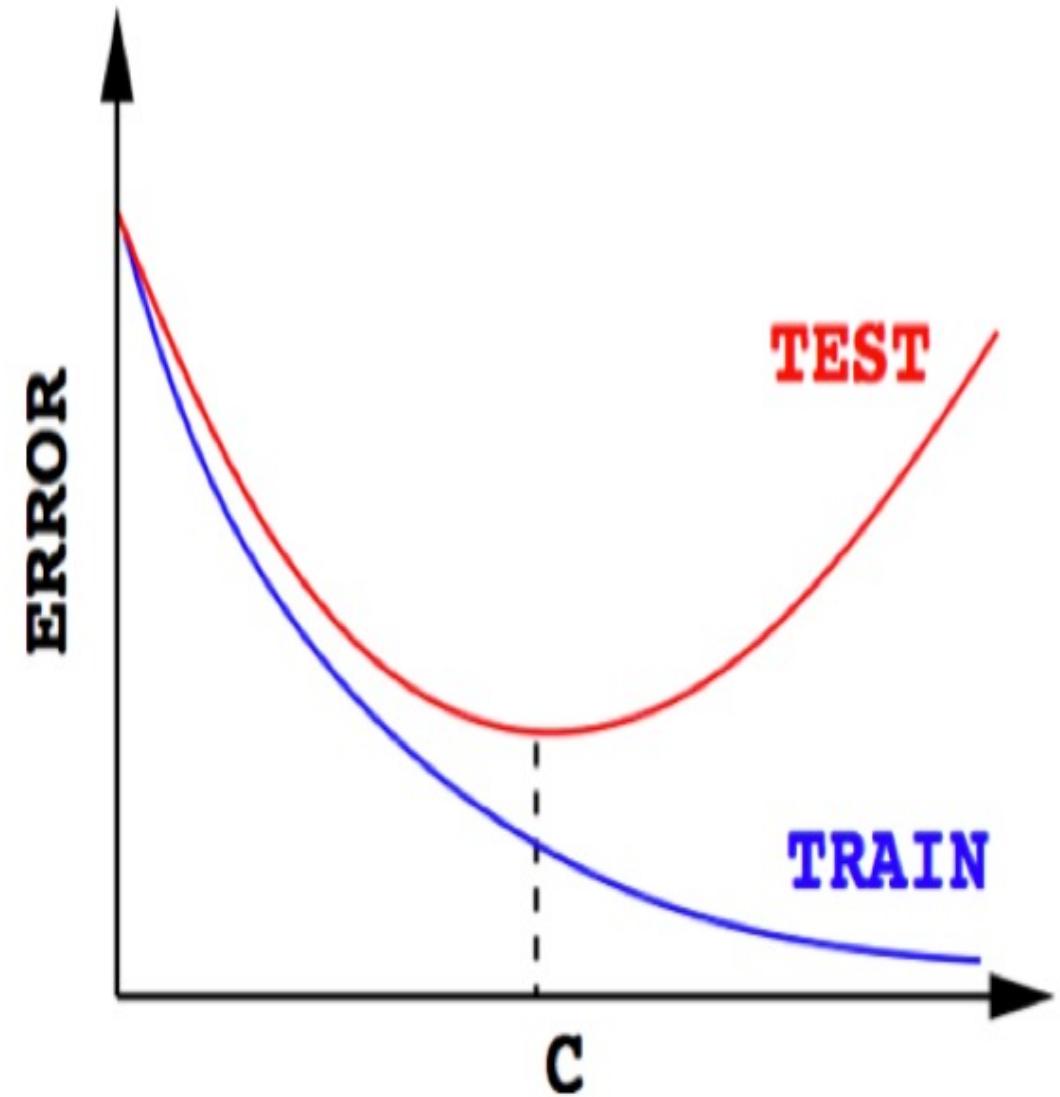
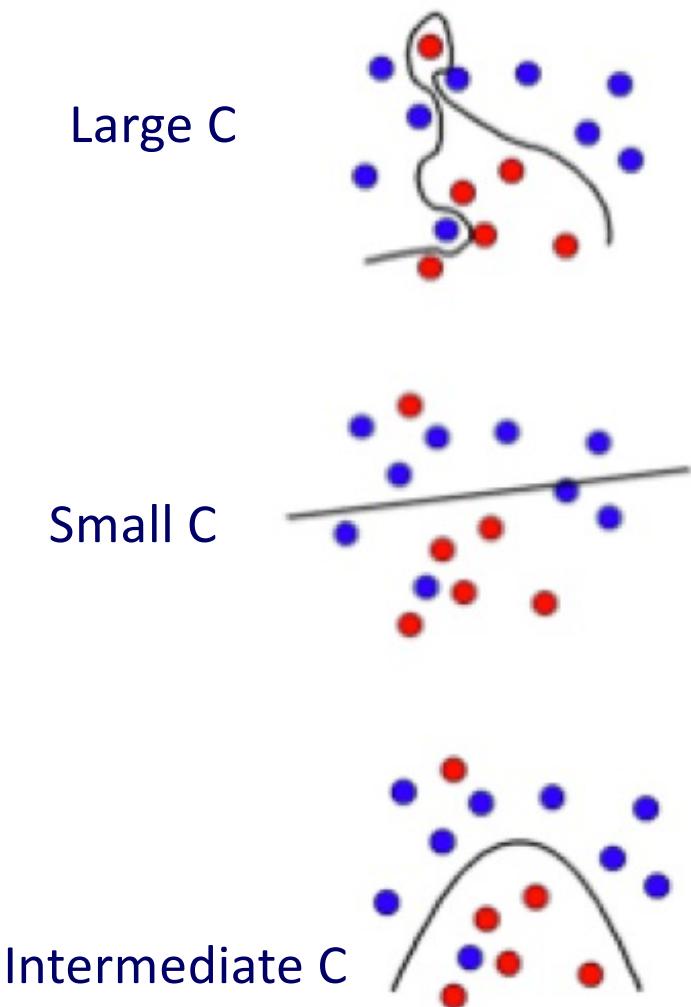
- High-degree kernels always fit the training data well, but at increased risks of over-fitting, i.e. the classifier will not generalize to new data points
- One needs to find a balance between **classification accuracy** on the training data and **regularity** of the kernel (not allowing the kernel to be too flexible)

## A low-degree kernel (left) and an over-fitting high-degree kernel (right)



# The parameter C has a similar role

- Large C will make **few classification errors** on the **training data**
- But this may not generalize to the **testing data**
- Small C pursues a **large separating margin** at the expenses of some classification errors on the training data.
- The accuracy more likely to generalize to testing data

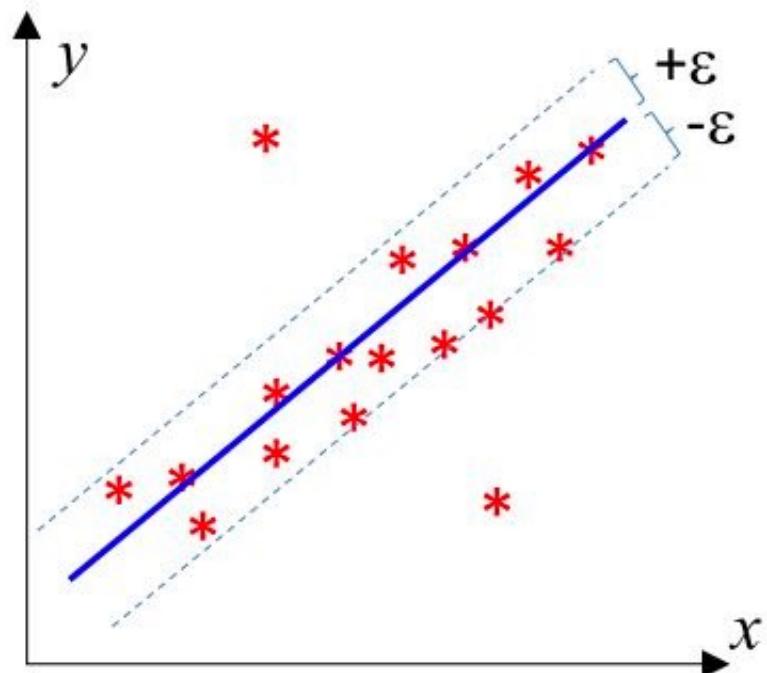


# $\varepsilon$ -Support vector regression ( $\varepsilon$ -SVR)

Given training data:

$$\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N \in R^n$$

$$y_1, y_2, \dots, y_N \in R$$



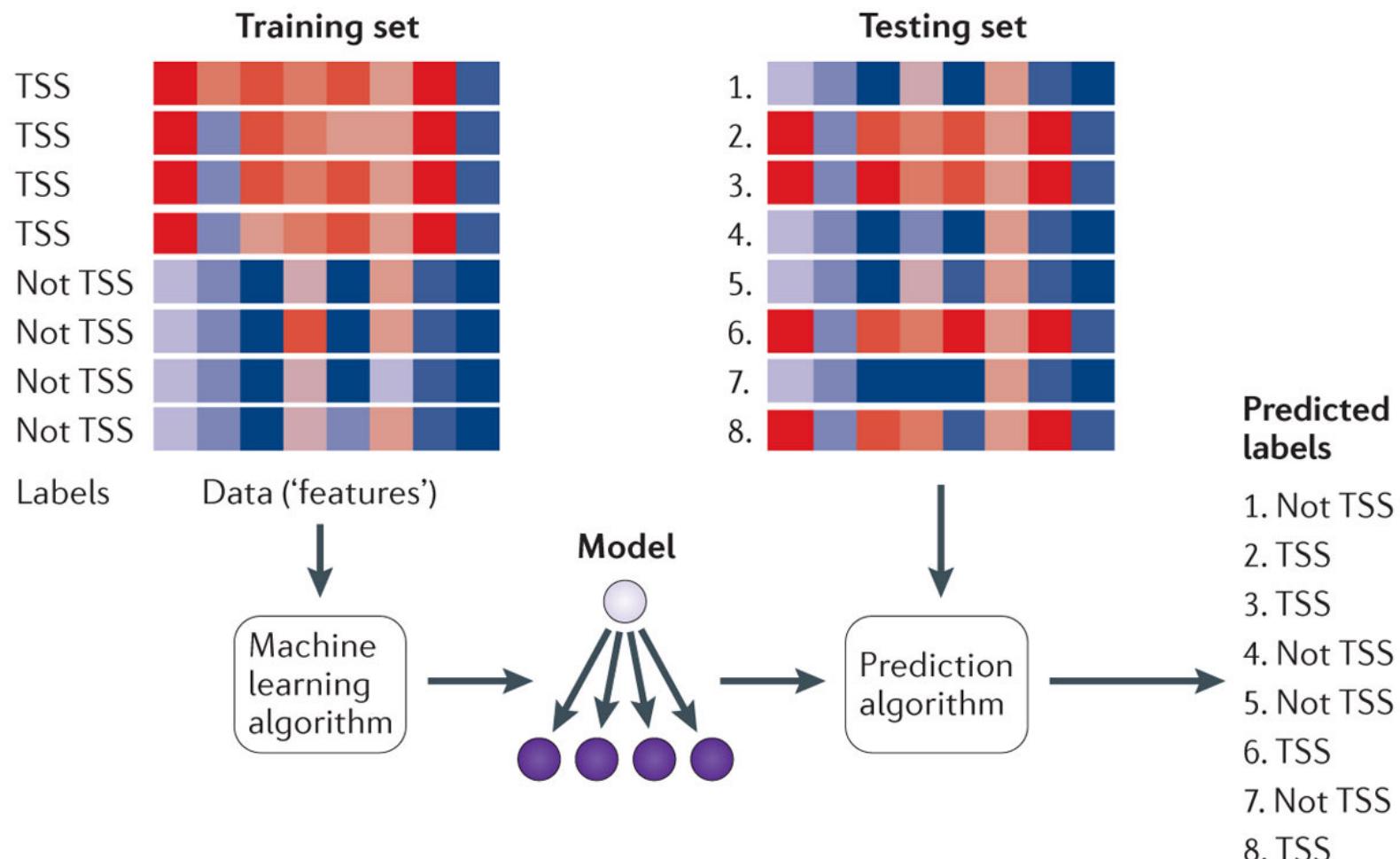
## Main idea:

Find a function  $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$  that approximates  $y_1, \dots, y_N$ :

- it has at most  $\varepsilon$  derivation from the true values  $y_i$
- it is as “flat” as possible (to avoid overfitting)

E.g., build a model to predict survival of cancer patients that can admit a one month error ( $= \varepsilon$  ).

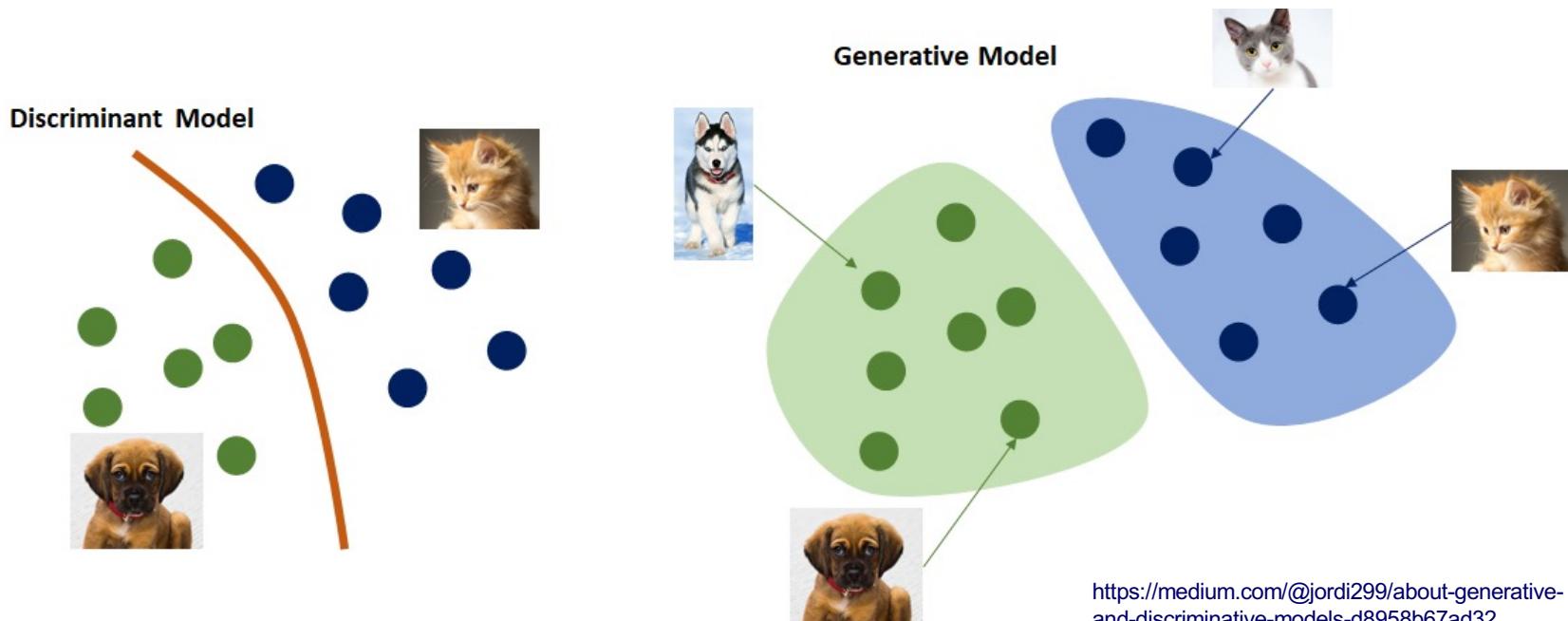
# Training and Testing



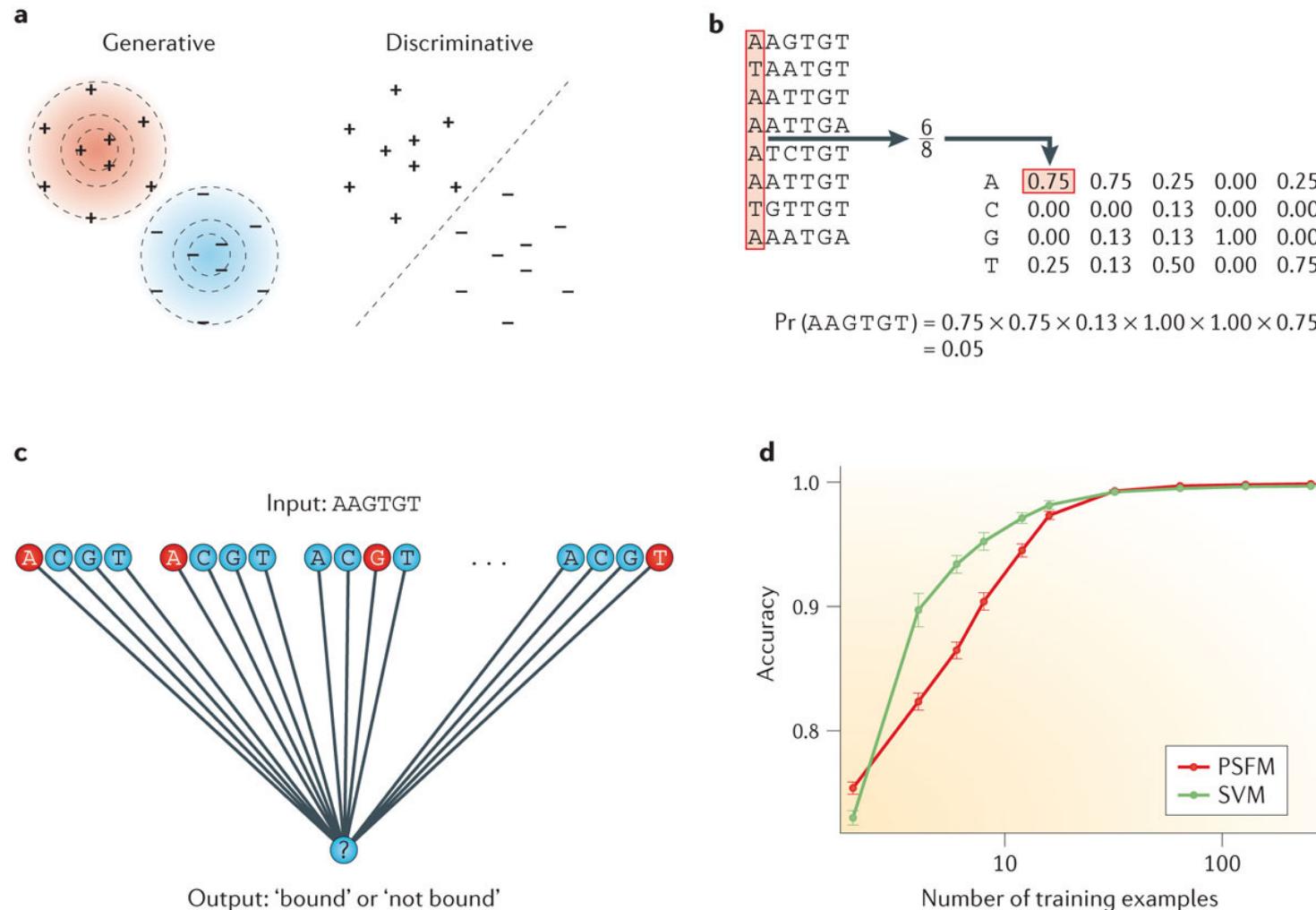
Libbrecht, M., Noble, W. Machine learning applications in genetics and genomics. *Nat Rev Genet* **16**, 321–332 (2015).  
<https://doi.org/10.1038/nrg3920>

# Generative vs. Discriminative models

- Generative approaches model the joint probability  $p(x,y)$  for generating data
- Discriminative approaches directly model  $p(y|x)$  for classification



# Predicting TF binding via Generative vs. Discriminative models



Libbrecht, M., Noble, W. Machine learning applications in genetics and genomics. *Nat Rev Genet* **16**, 321–332 (2015).  
<https://doi.org/10.1038/nrg3920>

# Workshop introducing machine learning to biologists

- ML4BIO workshop from Gitter Lab
- <https://gitter-lab.github.io/ml-bio-workshop/>