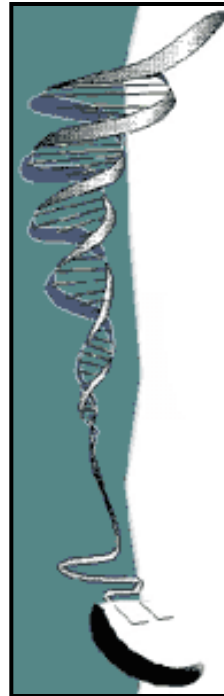
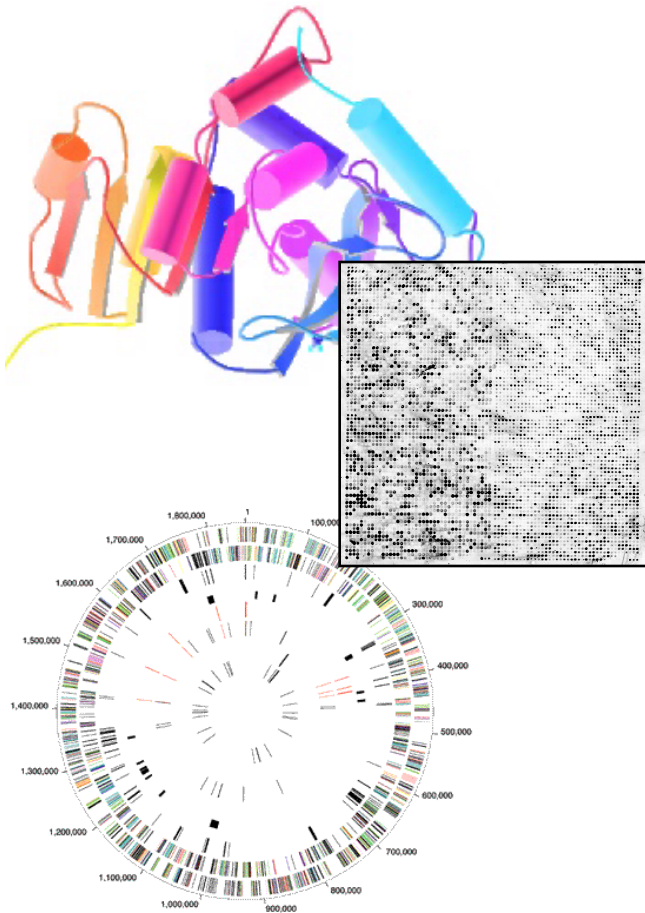


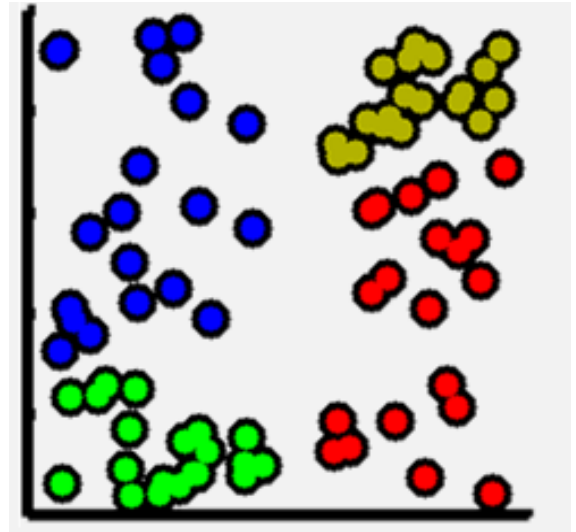
# Bioinformatics: Supervised Datamining



Mark Gerstein, Yale University  
[GersteinLab.org/courses/452](http://GersteinLab.org/courses/452)  
(last edit in spring '16)

# Supervised Learning

- Given: training data
  - The input objects (usually represented as vectors) –  $x, y$  values
  - The labels – colors
- Define: a function that determines the labels given the input objects
- Use the function to predict the labels of new objects



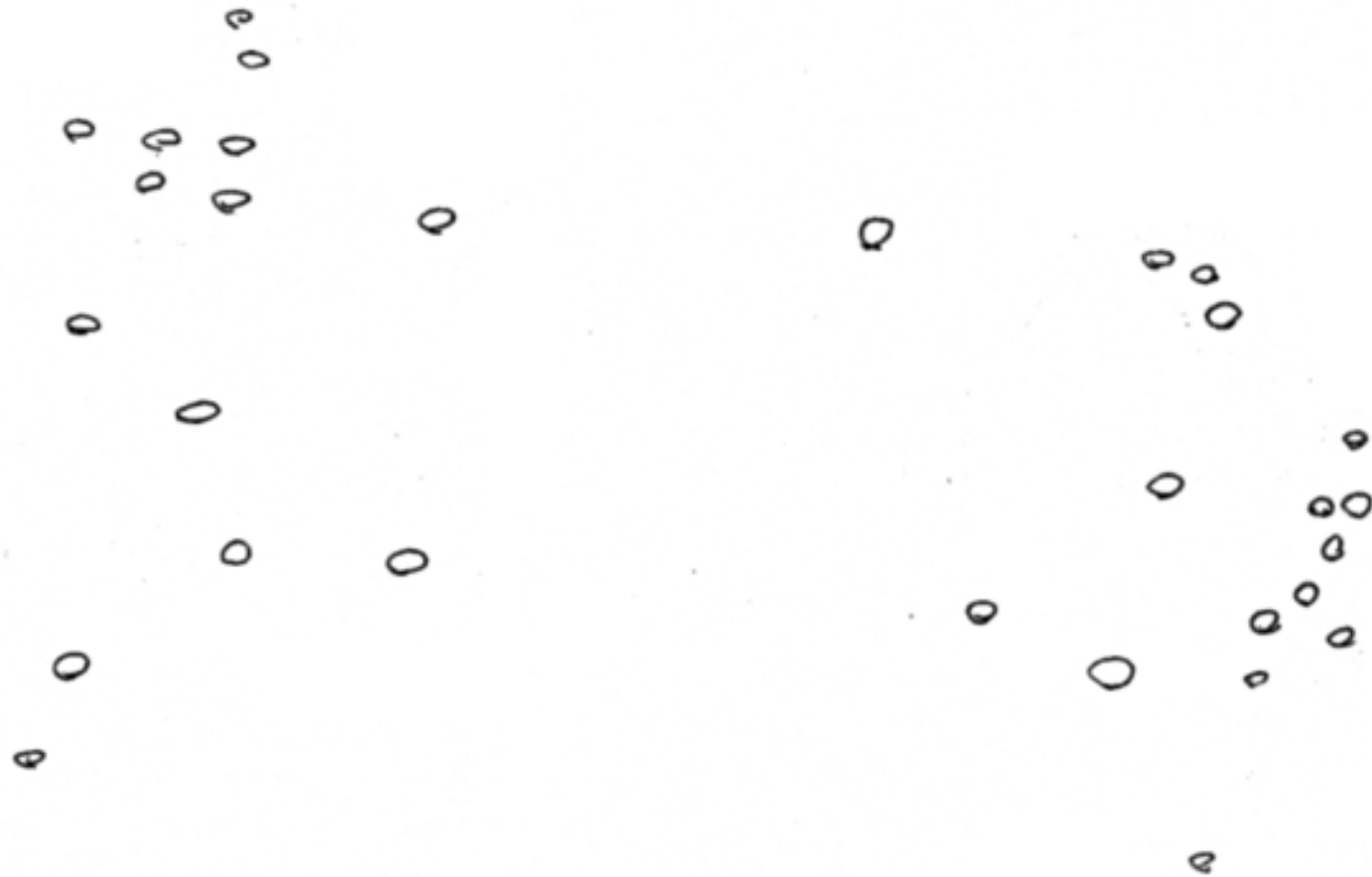
# Distinctions in Supervised Learning

- **Regression vs Classification**
  - Regression: labels are quantitative
  - Classification: labels are categorical
- **Regularized vs Un-regularized**
  - Regularized: penalize model complexity to avoid over-fitting
  - Un-regularized: no penalty on model complexity
- **Parametric vs Non-parametric**
  - Parametric: an explicit parametric model is assumed
  - Non-parametric: otherwise
- **Ensemble vs Non-ensemble**
  - Ensemble: combines multiple models
  - Non-ensemble: a single model

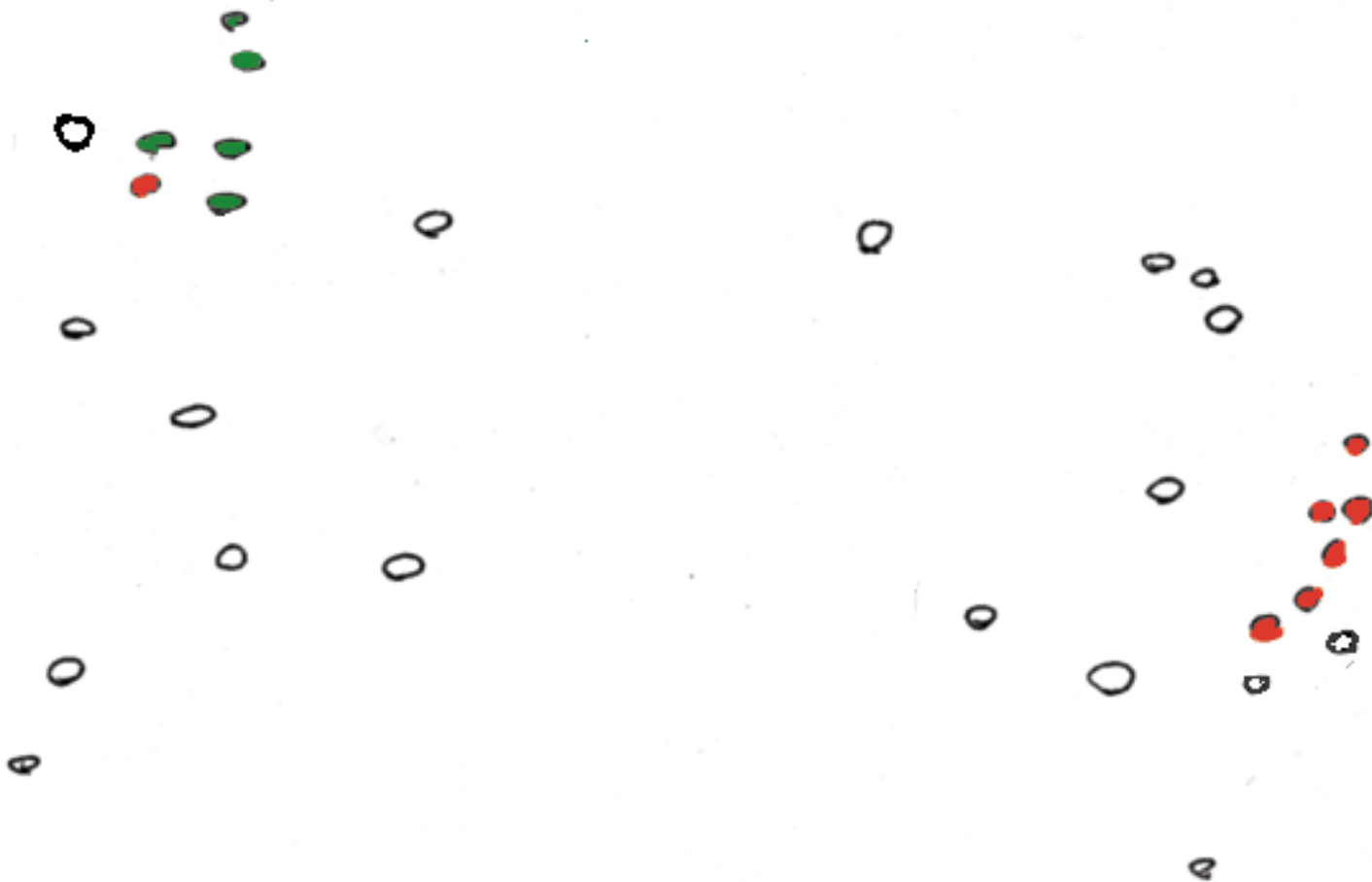
Arrange data in a tabulated form, each row representing an example and each column representing a feature, including the dependent experimental quantity to be predicted.

	predictor1	Predictor2	predictor3	predictor4	response
G1	A(1,1)	A(1,2)	A(1,3)	A(1,4)	Class A
G2	A(2,1)	A(2,2)	A(2,3)	A(2,4)	Class A
G3	A(3,1)	A(3,2)	A(3,3)	A(3,4)	Class B

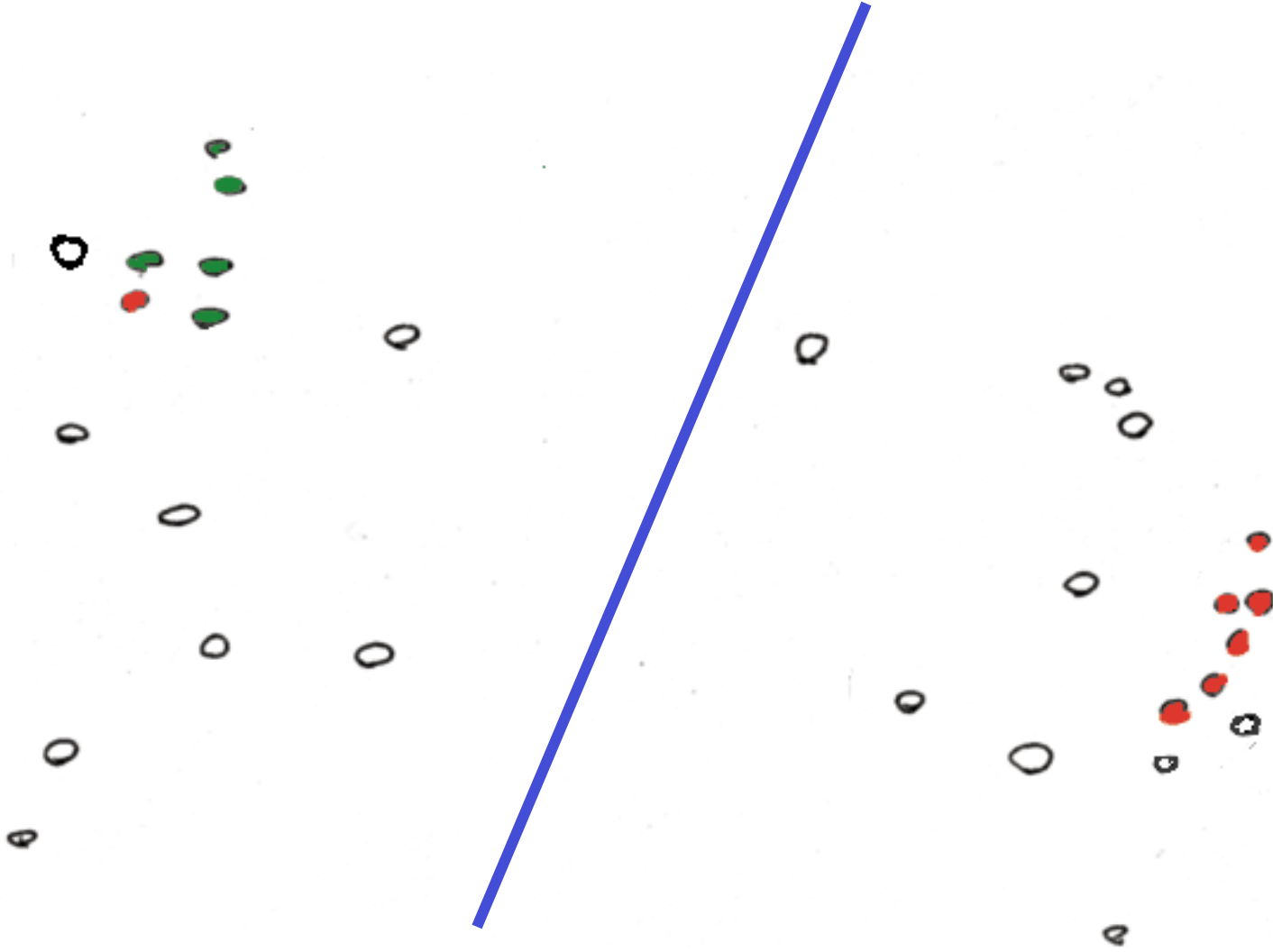
# Represent predictors in abstract high dimensional space



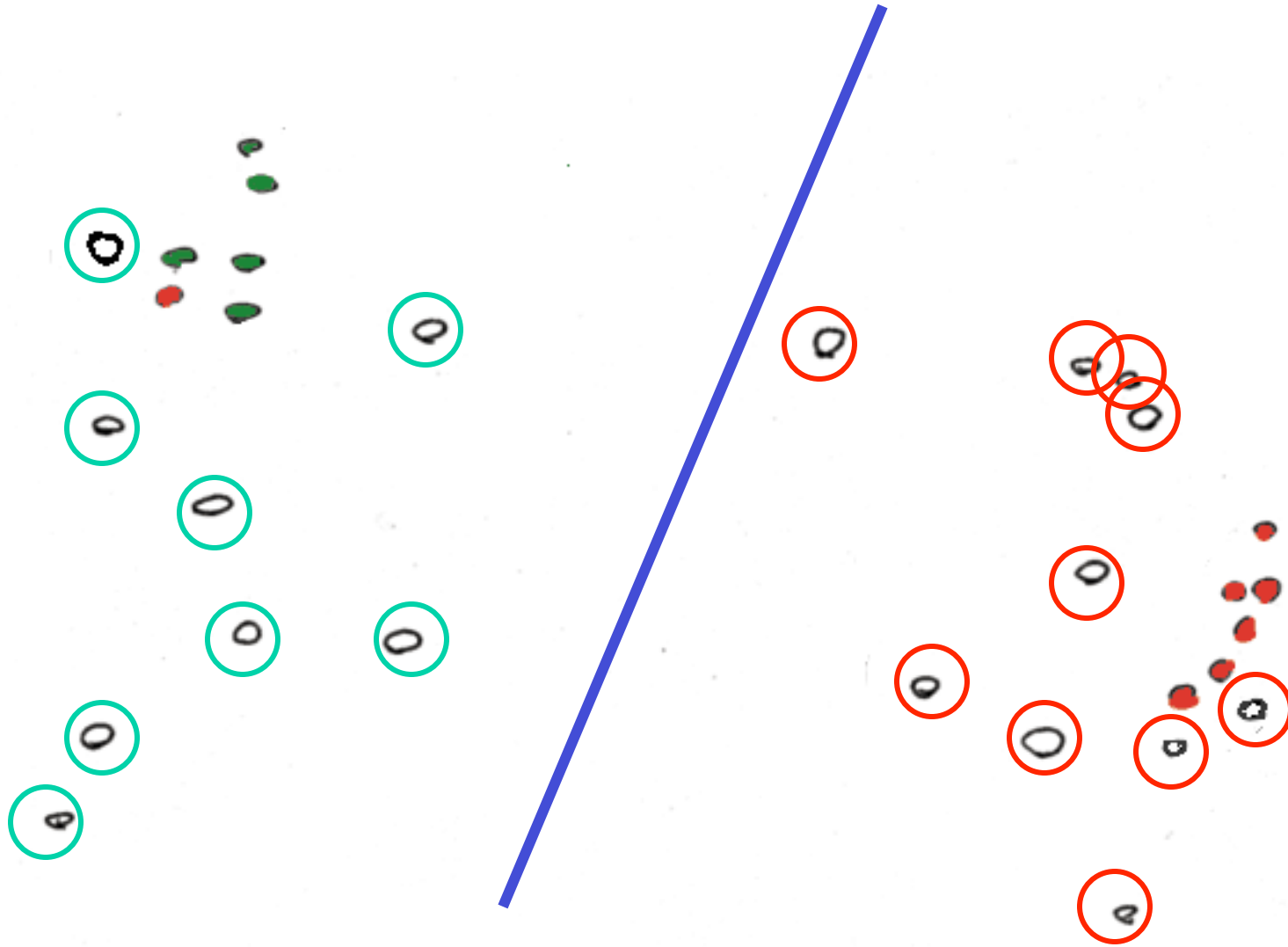
# Tagged Data



# Find a Division to Separate Tagged Points

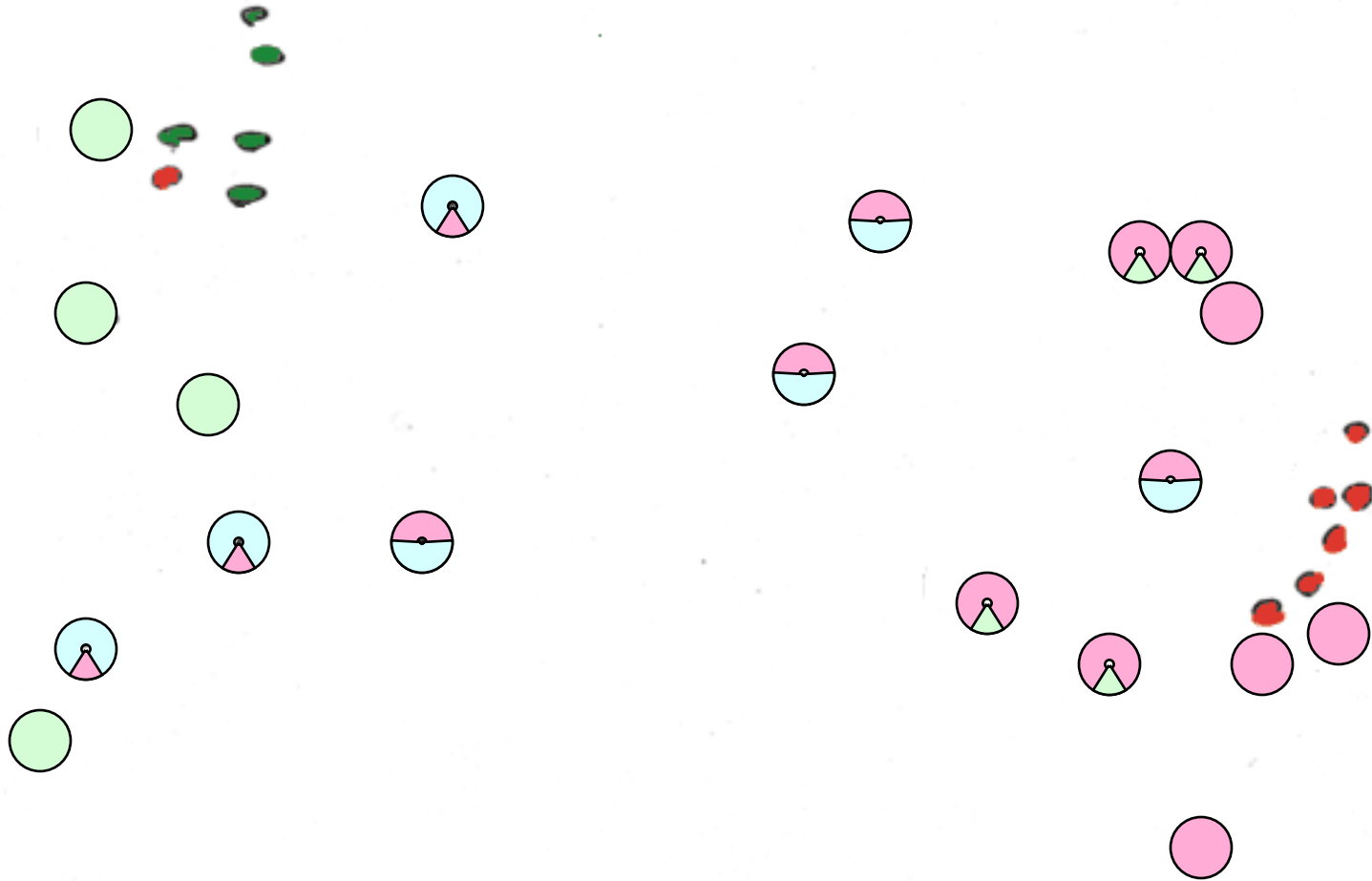


# Extrapolate to Untagged Points

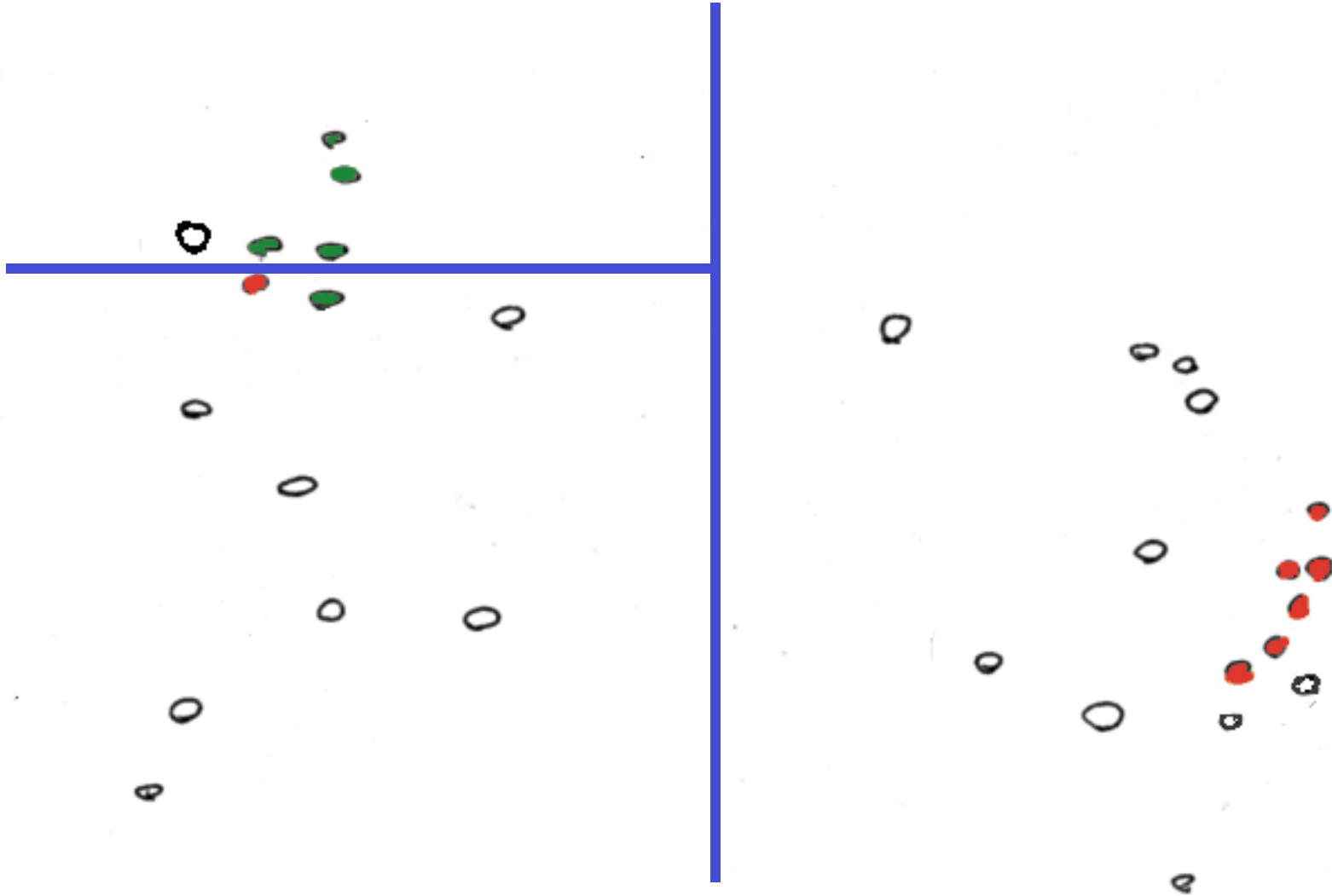




# Probabilistic Predictions of Class



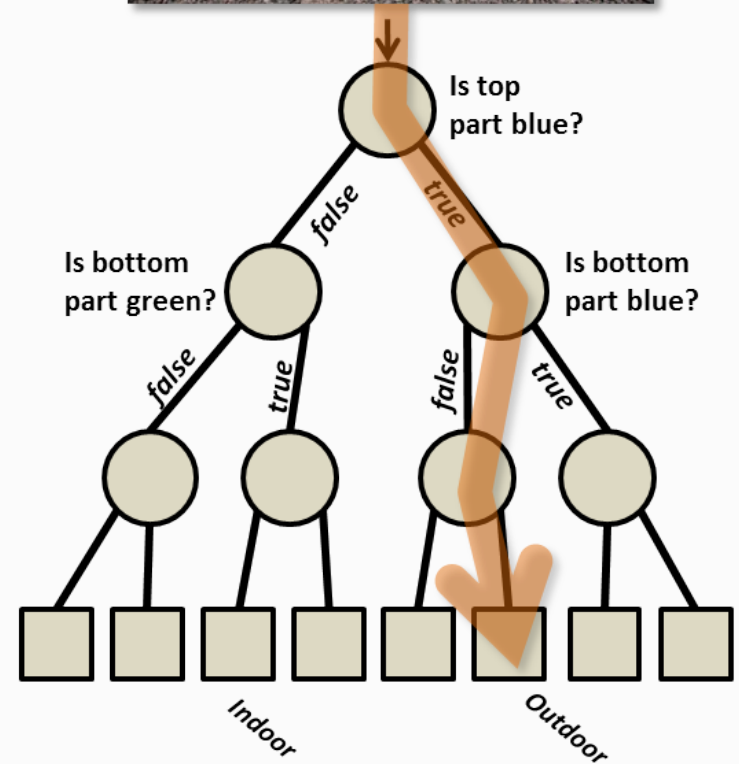
# Find a Division to Separate Tagged Points



# Decision Trees

- **Classify data by asking questions** that divide data in subgroups
- Keep asking questions until subgroups become homogenous
- Use **tree** of questions to make predictions

A decision tree

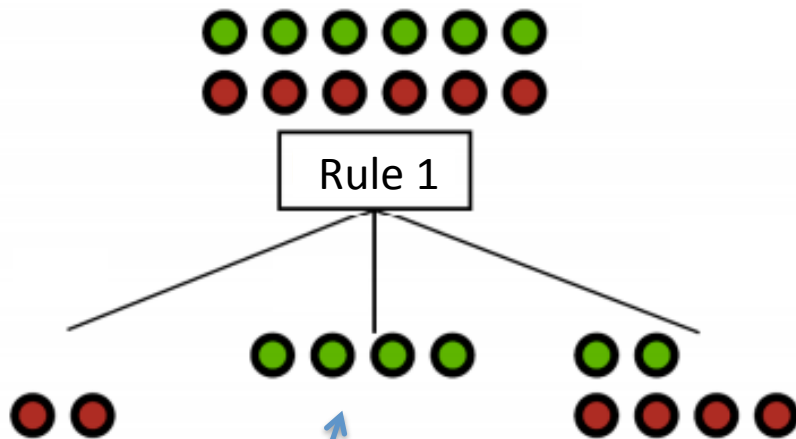


b

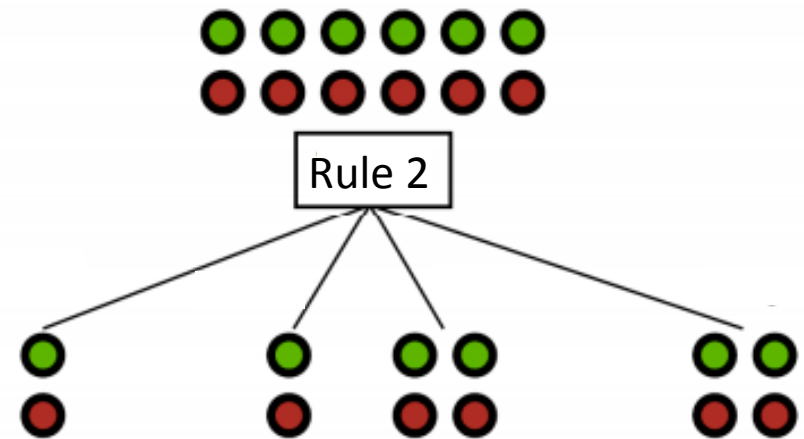
- Example: Is a picture taken inside or outside?

# What makes a good rule?

- Want resulting groups to be as homogenous as possible



2/3 Groups homogenous  
→ Good rule



All groups still 50/50  
→ Unhelpful rule

# Quantifying the value of rules

- Decrease in inhomogeneity

- Most popular metric: Information theoretic entropy

$$S = - \sum_{i=1}^m p_i \log p_i$$

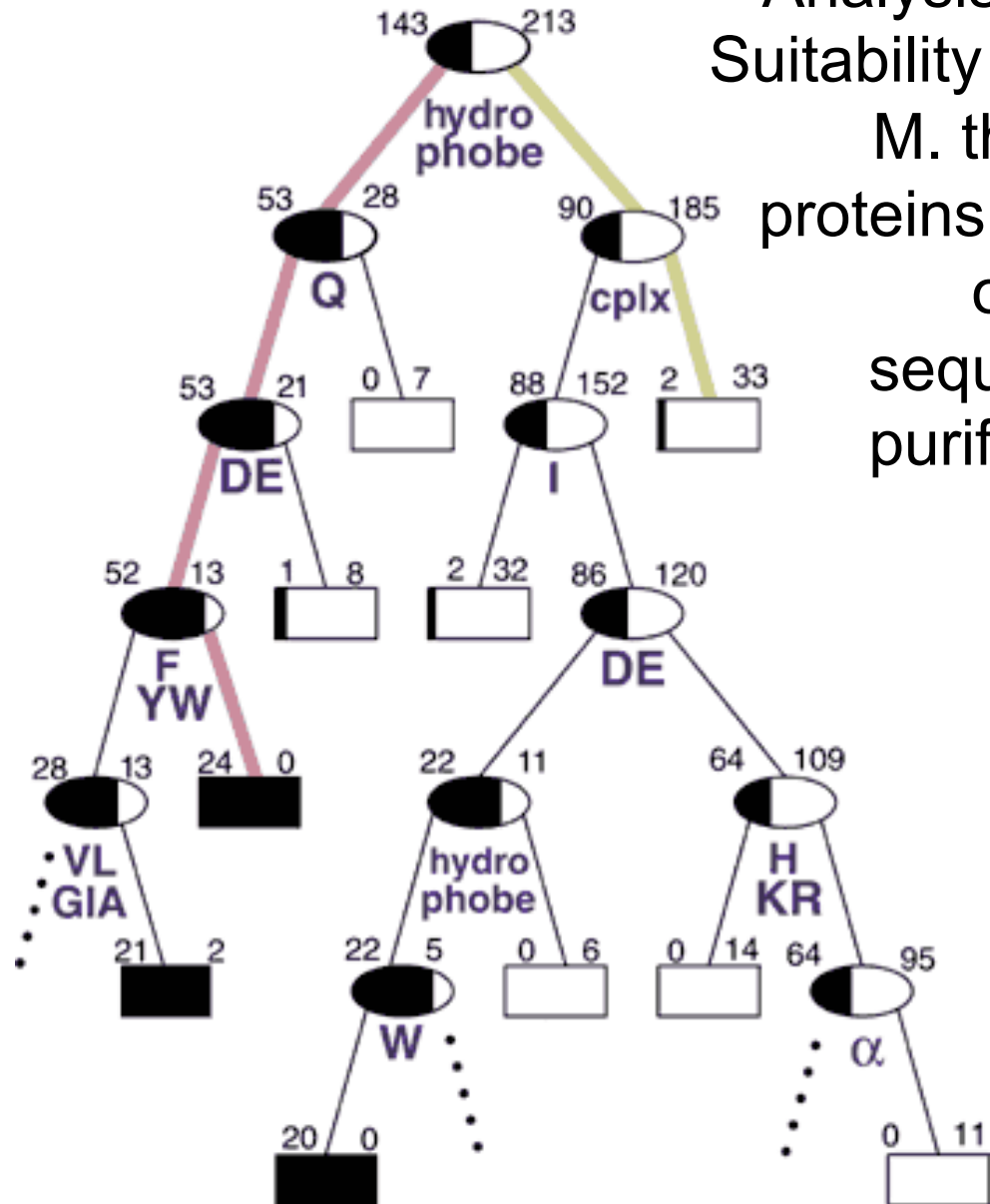
- Use frequency of classifier characteristic within group as probability
- Minimize entropy to achieve homogenous group

# Algorithm

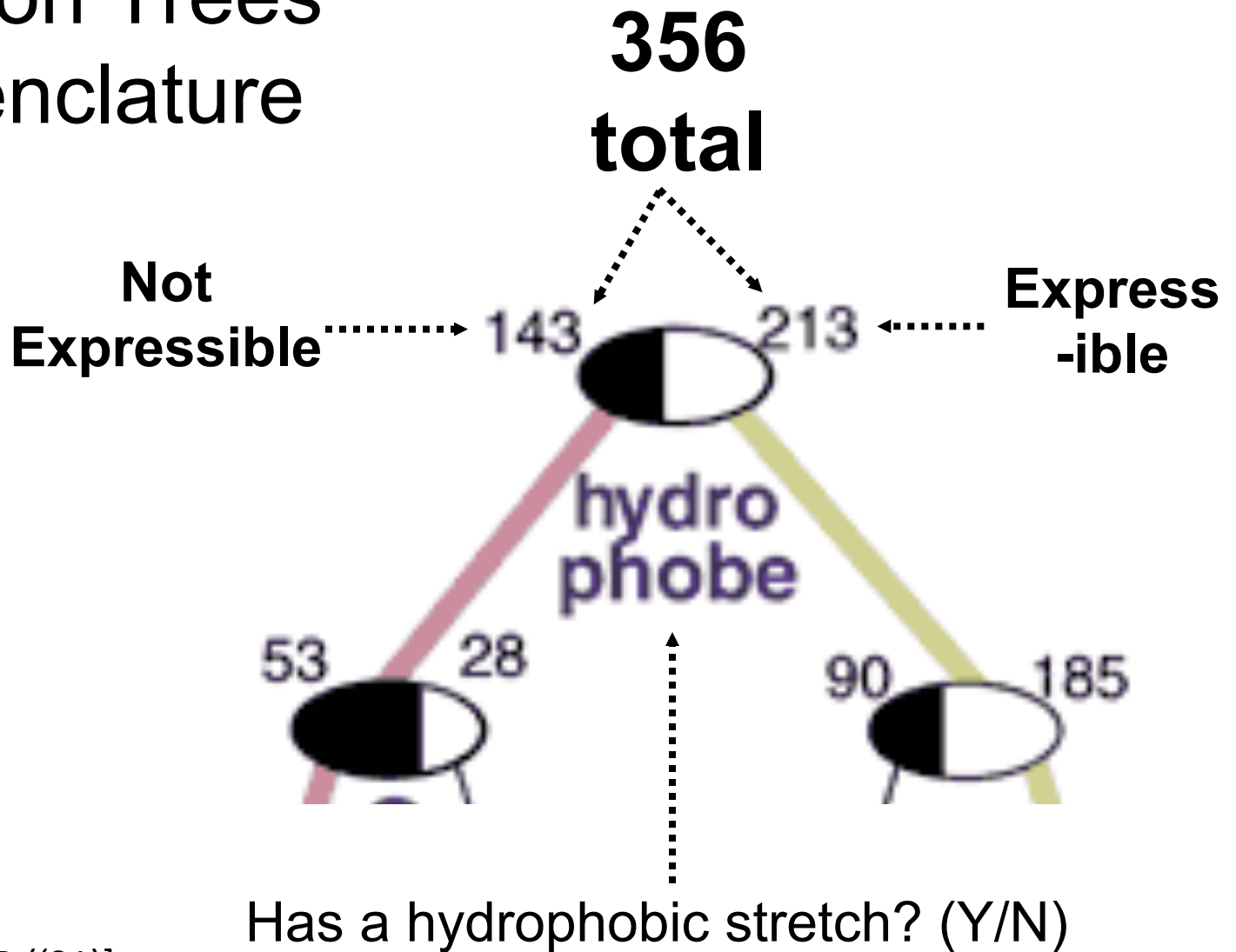
- For each characteristic:
  - Split into subgroups based on each possible value of characteristic
- Choose rule from characteristic that maximizes decrease in inhomogeneity
- For each subgroup:
  - if (inhomogeneity < threshold):
    - Stop
  - else:
    - Restart rule search (recursion)

# Retrospective Decision Trees

Analysis of the Suitability of 500 M. thermo. proteins to find optimal sequences purification

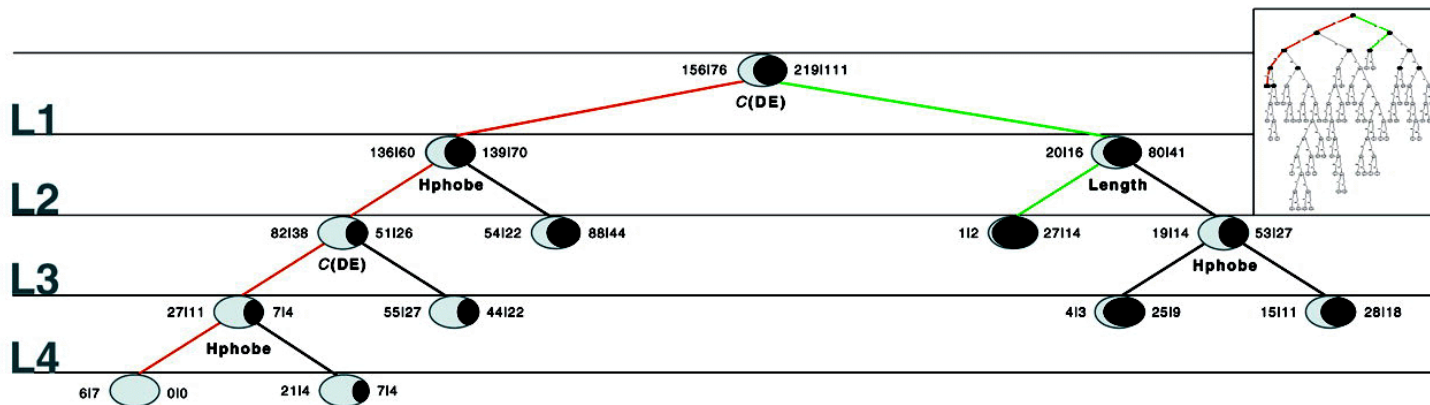
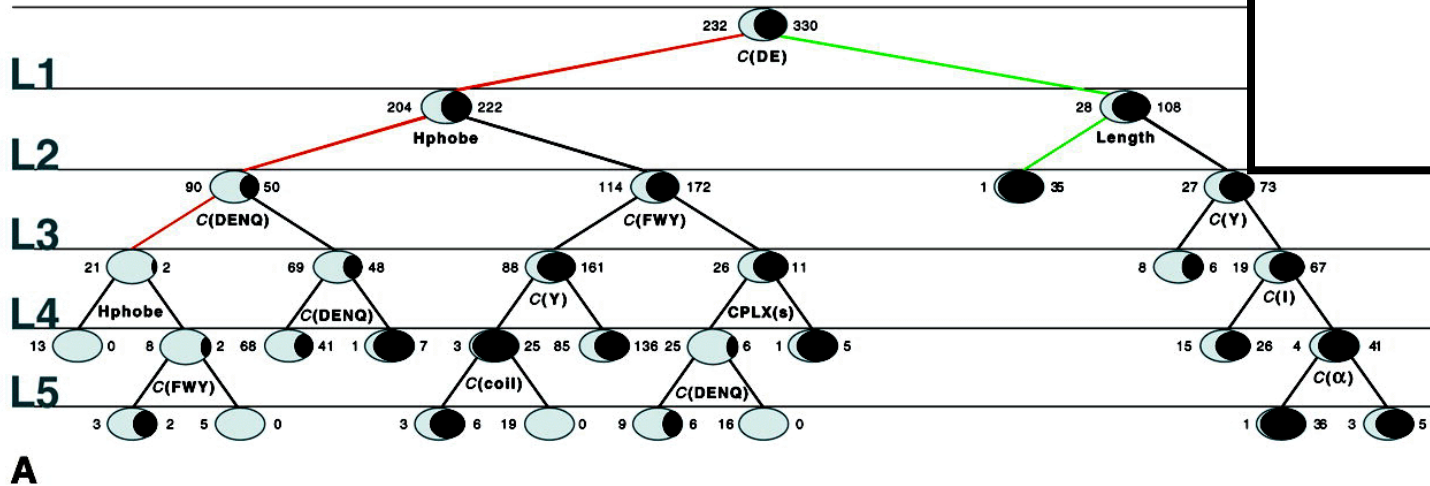
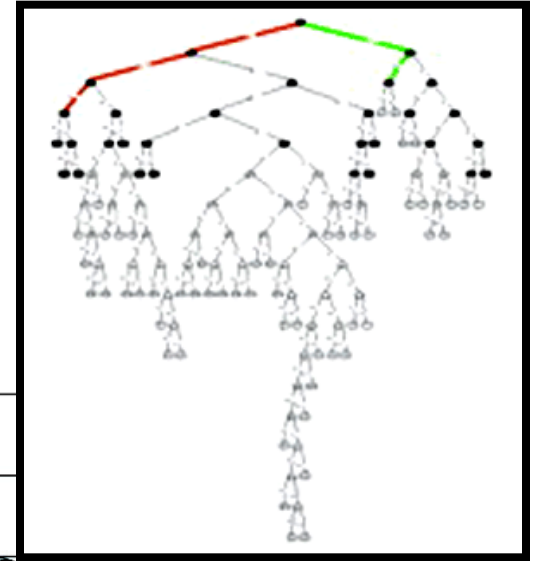


# Retrospective Decision Trees Nomenclature





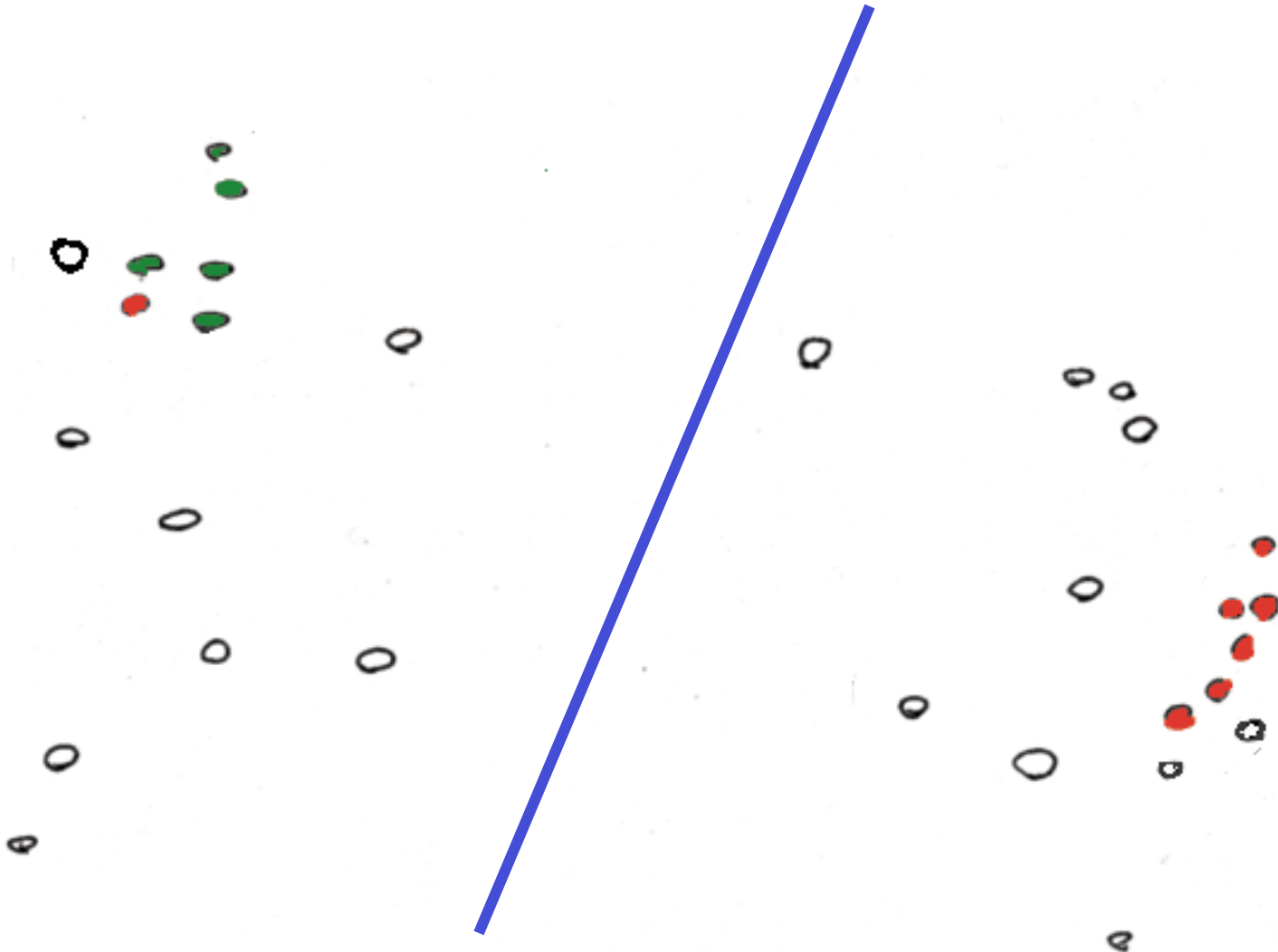
# Overfitting, Cross Validation, and Pruning



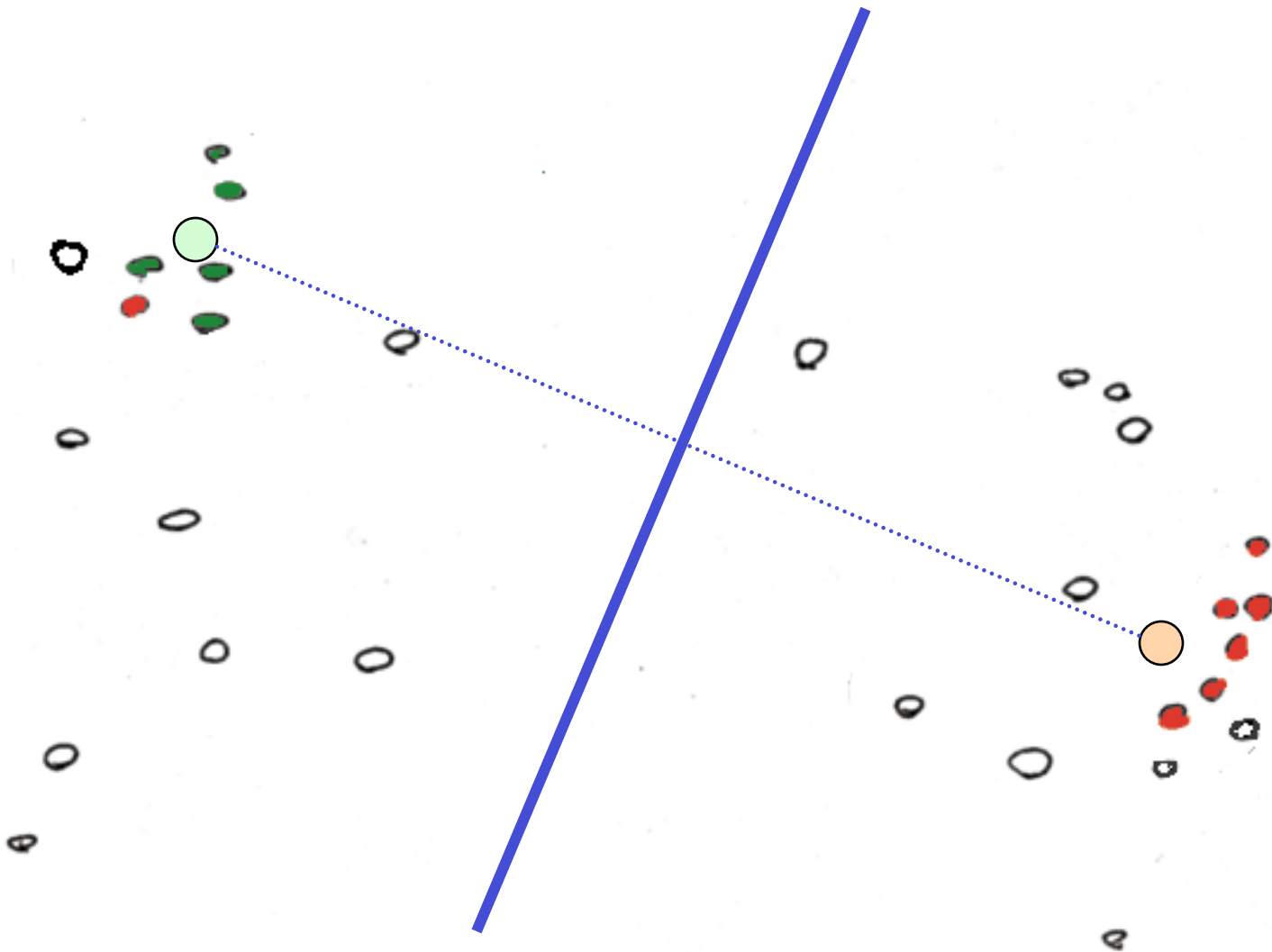
# Extensions of Decision Trees

- Decision Trees method is very sensitive to noise in data
- Random forests is an ensemble of decision trees, and is much more effective.

# Find a Division to Separate Tagged Points



# Discriminant to Position Plane



# Fisher discriminant analysis

- Use the training set to reveal the structure of class distribution by seeking a linear combination
- $y = w_1x_1 + w_2x_2 + \dots + w_nx_n$  which maximizes the ratio of the separation of the class means to the sum of each class variance (within class variance). This linear combination is called the first linear discriminant or first canonical variate. Classification of a future case is then determined by choosing the nearest class in the space of the first linear discriminant and significant subsequent discriminants, which maximally separate the class means and are constrained to be uncorrelated with previous ones.

$$s_i^2 = \sum_{y \in Y_i} (y - m_i)^2$$

$$m_i = \vec{w} \cdot \vec{m}_i$$

Solution of 1<sup>st</sup> variate

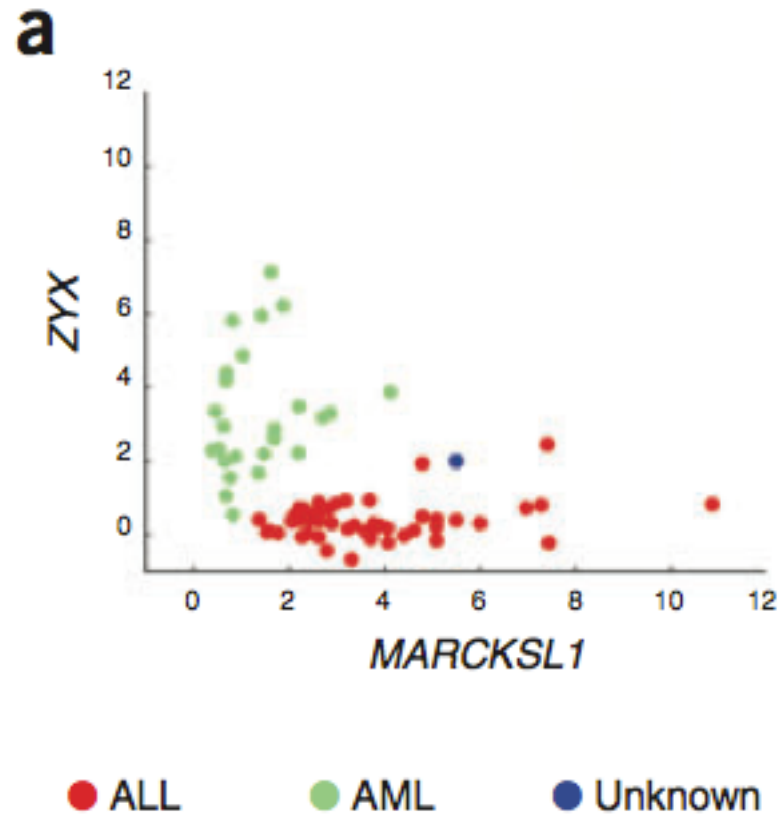
cbb752rd0mg

$$\vec{w} = S_W^{-1} (\vec{m}_1 - \vec{m}_2)$$

# Support Vector Machines

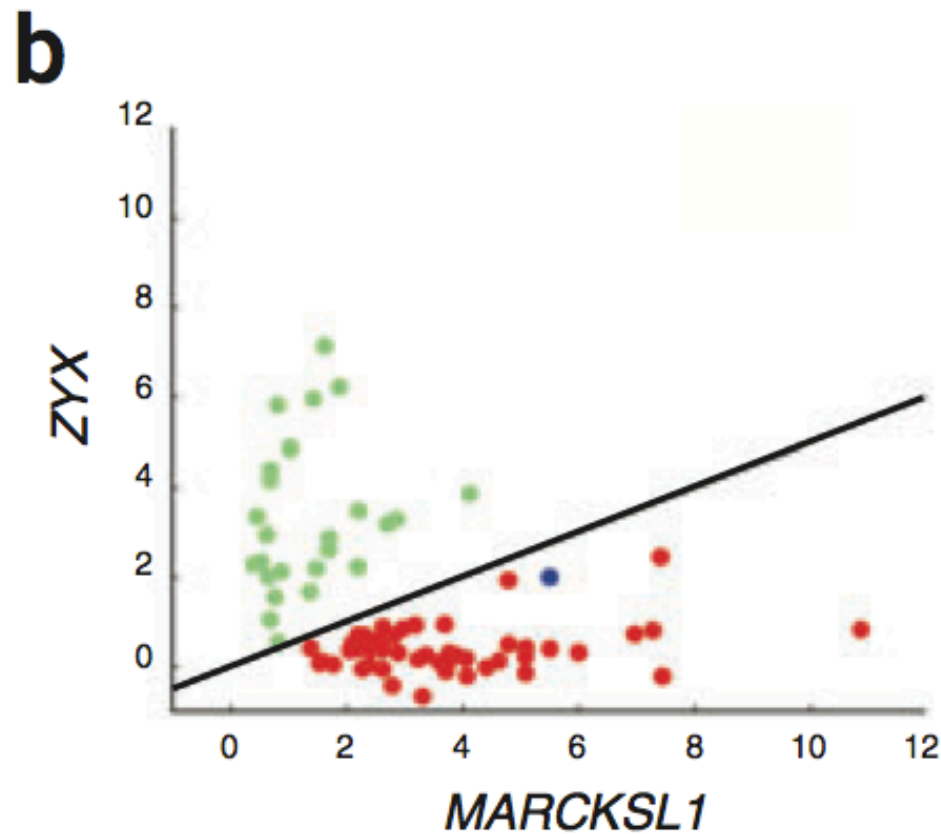
- A very powerful tool for classifications
- Example Applications:
  - Text categorization
  - Image classification
  - Spam email recognition, etc
- It has also been successfully applied in many biological problems:
  - Disease diagnosis
  - Automatic genome functional annotation
  - Prediction of protein-protein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

- Example: Leukemia patient classification



ALL: acute lymphoblastic leukemia

AML: acute myeloid leukemia

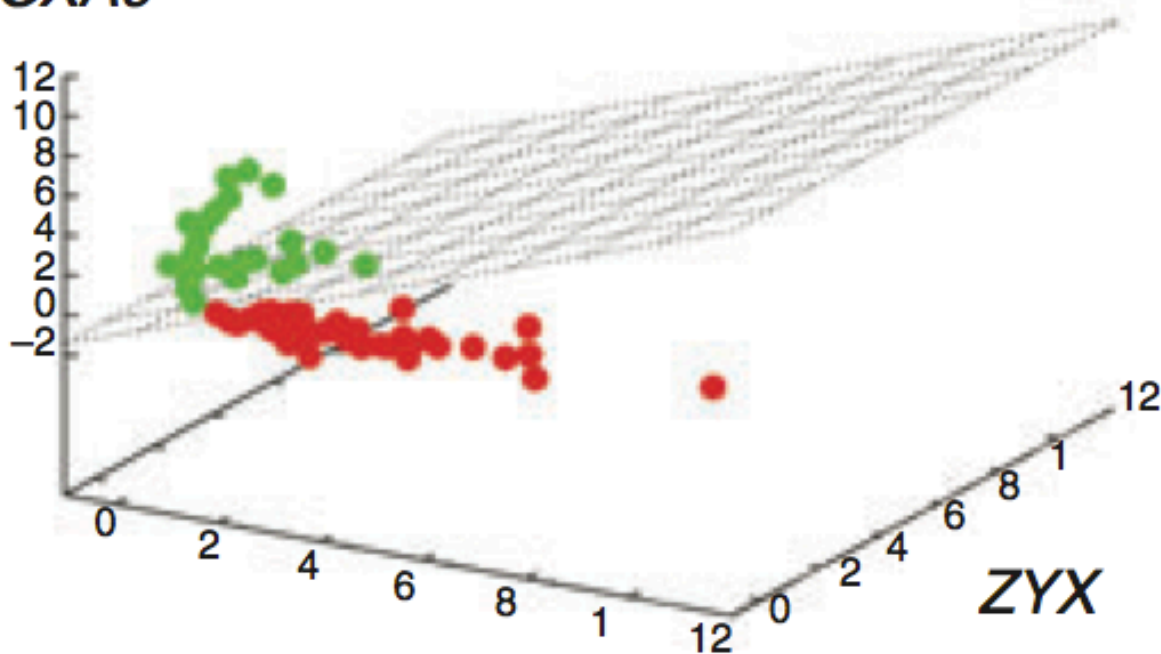


- A simple line suffices to separate the expression profiles of ALL and AML



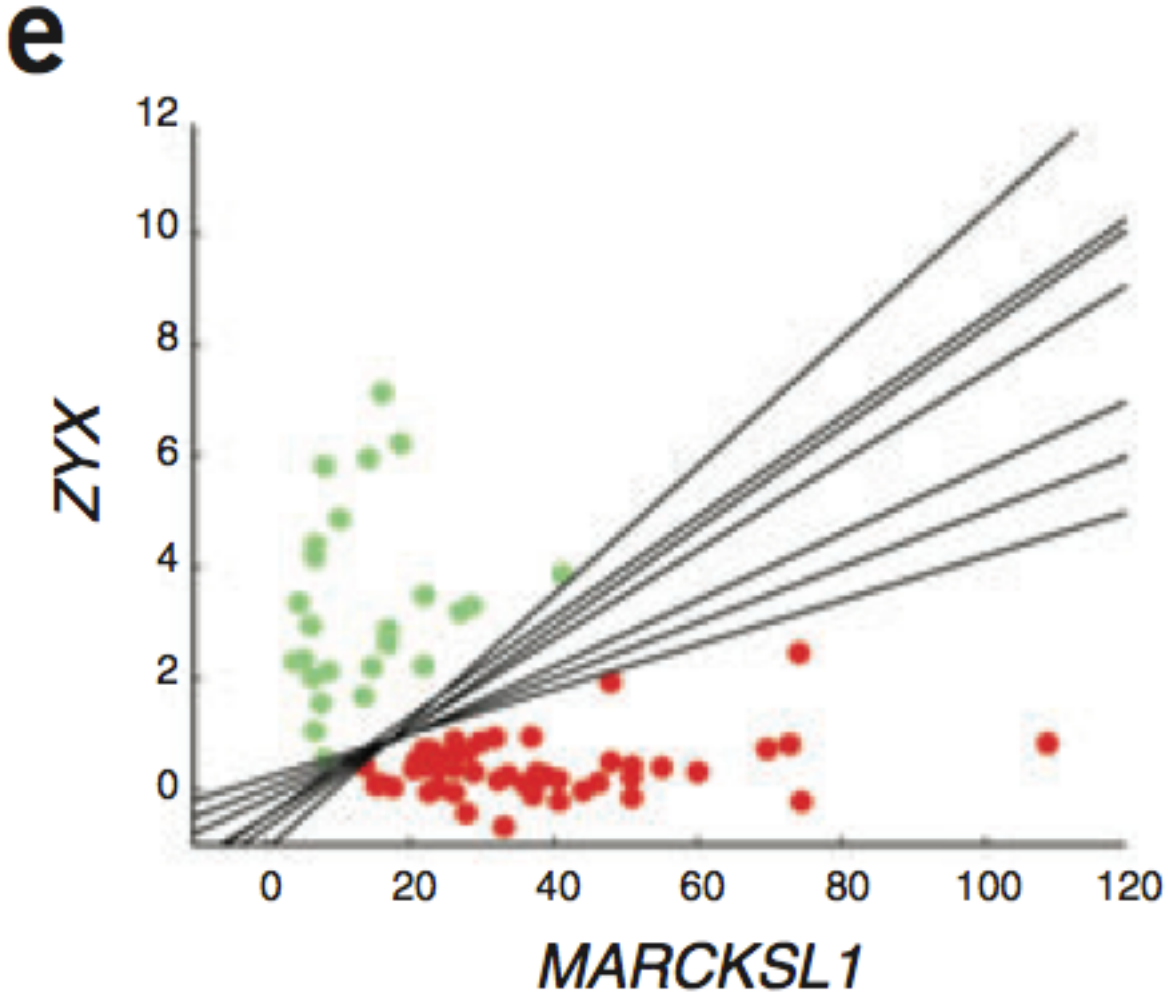
d

*HOXA9*



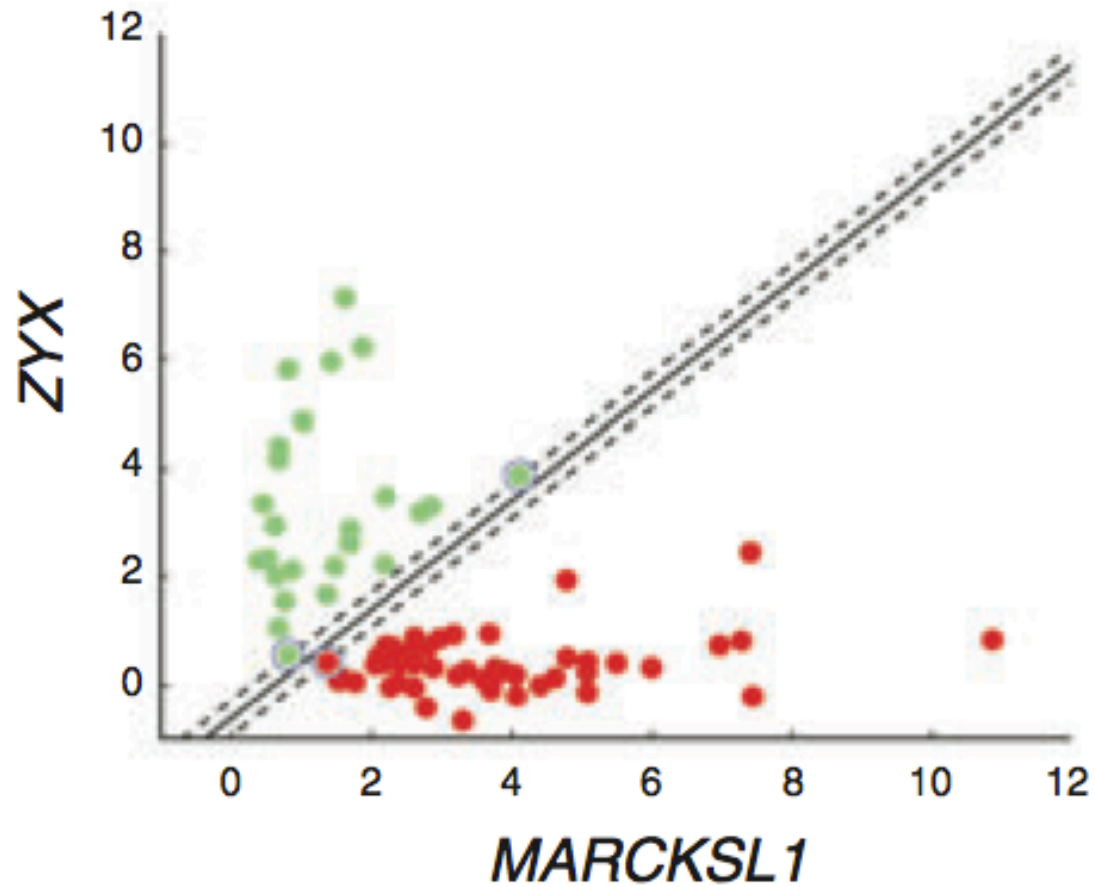
*MARCKSL1*

- 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”



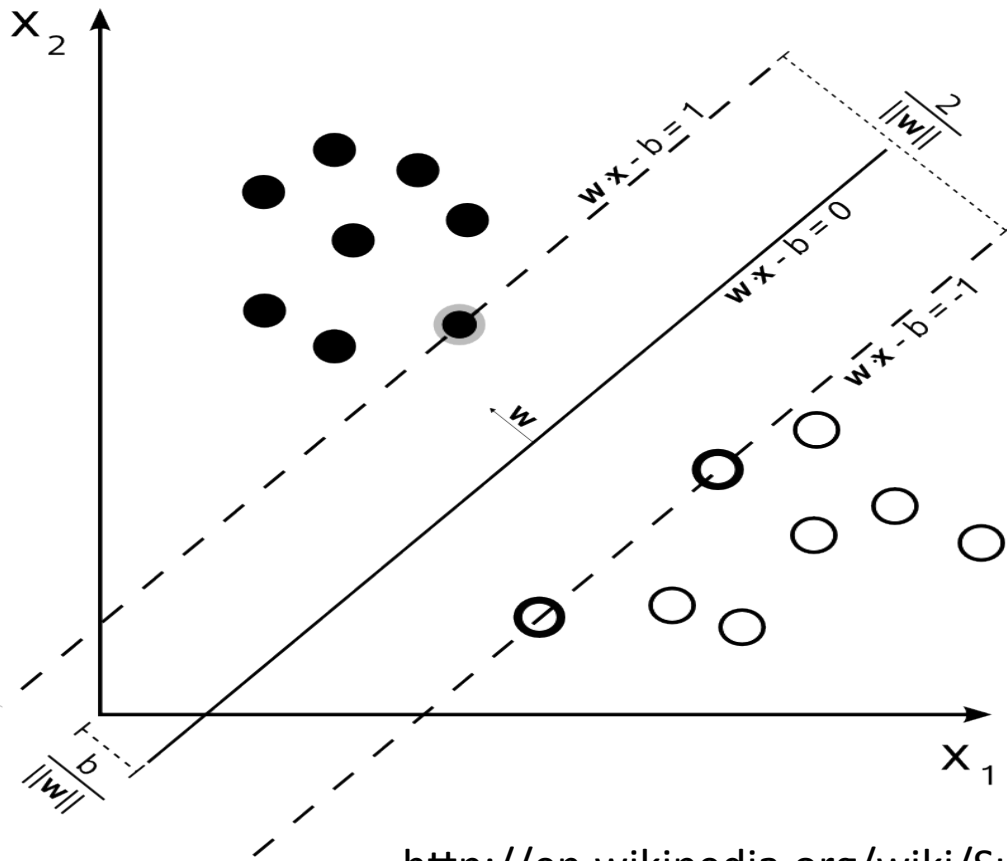
- Is there a “best” line?

**f**



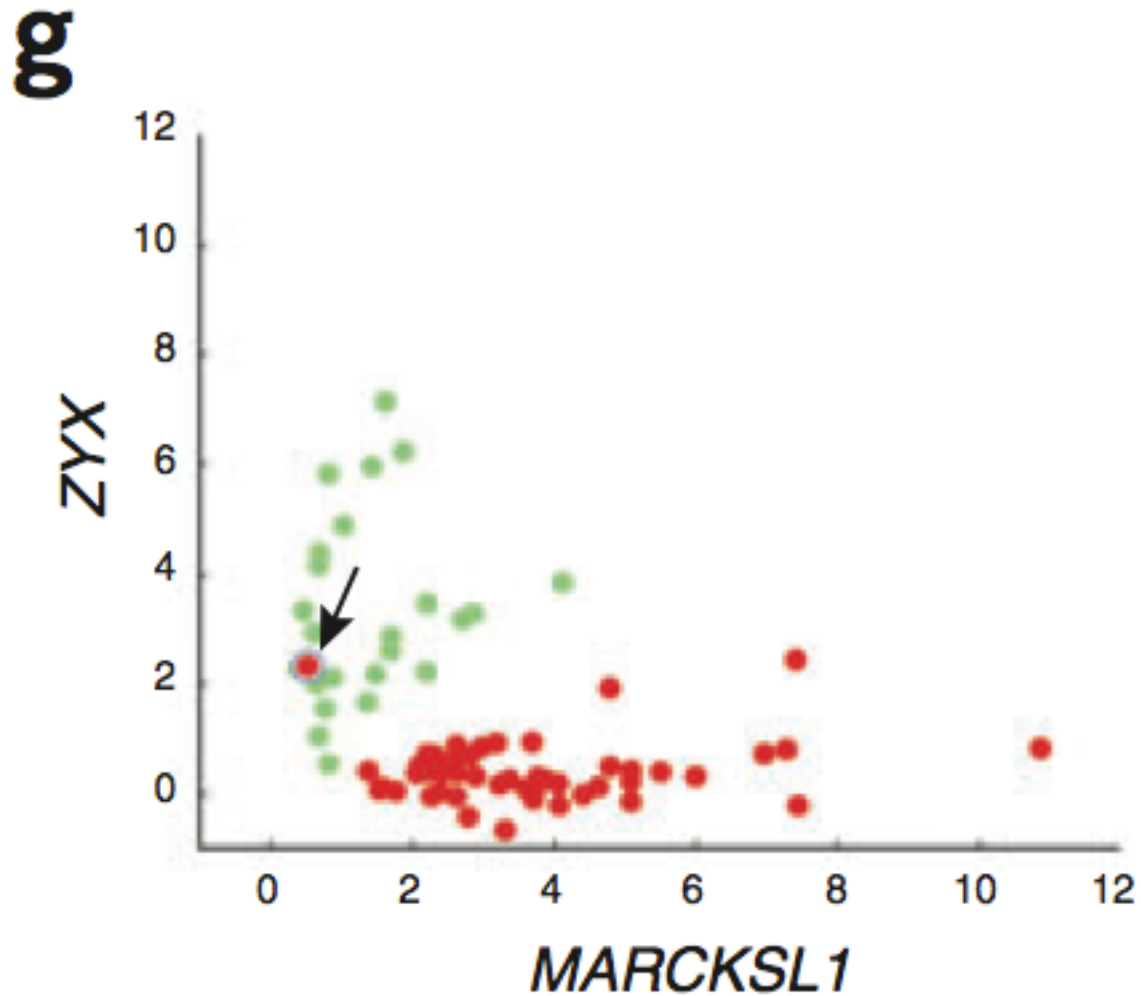
- The **maximum margin hyperplane**

- 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:  $w \cdot x + b = 0$
- The margin-width equals to:  $2 / \|w\|$ ,  $\|w\| = \sqrt{w \cdot w}$

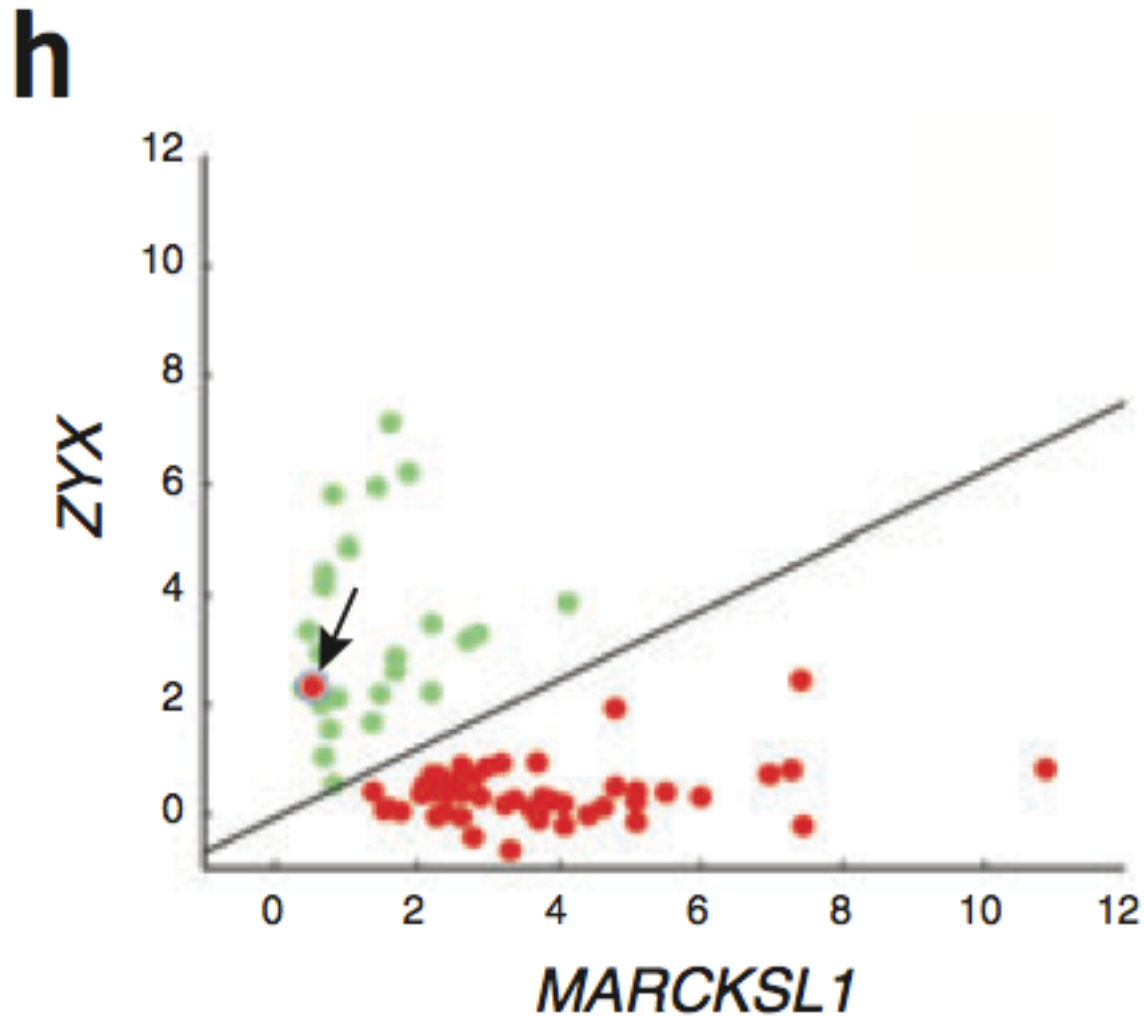


- Find a hyperplane such that:
  - No data points fall between the lines  $w \bullet x + b = -1$  and  $w \bullet x + b = +1$
  - The margin  $2/||w||$  is maximized
- Mathematically,
  - Minimize<sub>w,b</sub>  $1/2 ||w||^2$ , subject to:
  - for  $y_i = 1$ ,  $w \bullet x_i + b \geq 1$
  - for  $y_i = -1$ ,  $w \bullet x_i + b \leq -1$
  - Combining them, for any  $i$ ,  $y_i(w \bullet x_i + b) \geq 1$
- The solution expresses  $w$  as a linear combination of the  $x_i$
- So far, we have been 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  $\frac{1}{2} \|w\|^2 + C \sum_i s_i$   
 $w, b, s$

- subject to, for any  $i$ ,  $y_i(w \cdot x_i + b) \geq 1 - s_i, s_i \geq 0$

- $S_i$  are the slack variables

- $C$  controls the number of tolerated misclassifications  
(It's effectively a regularization parameter on model complexity)

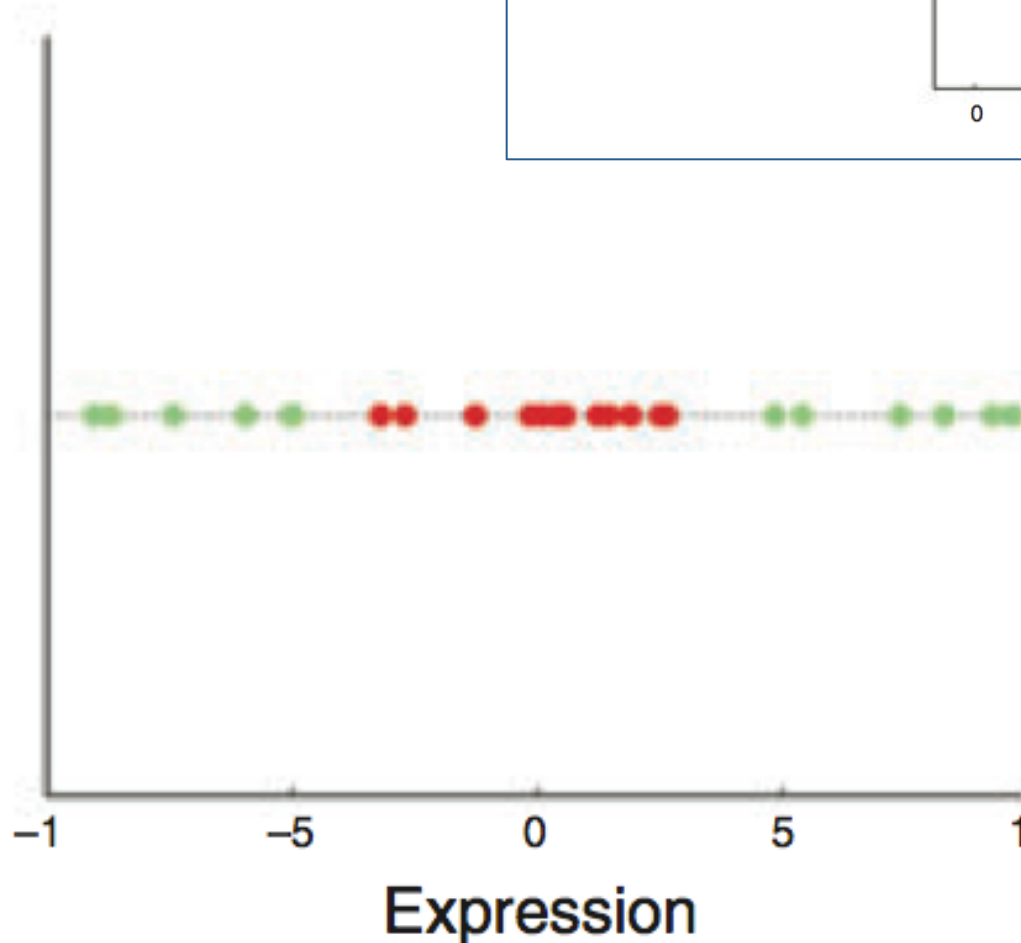
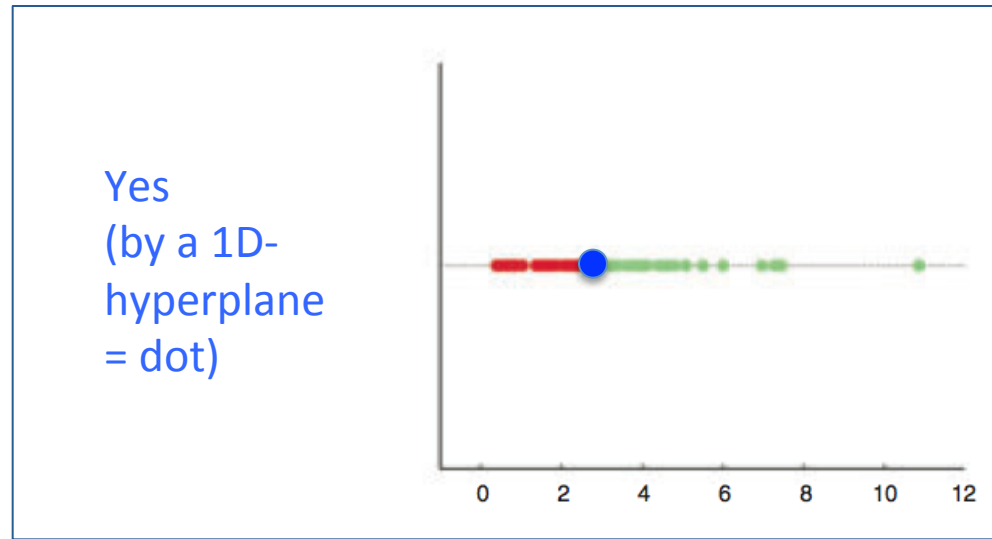
- A small  $C$  would allow more misclassifications

- 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**

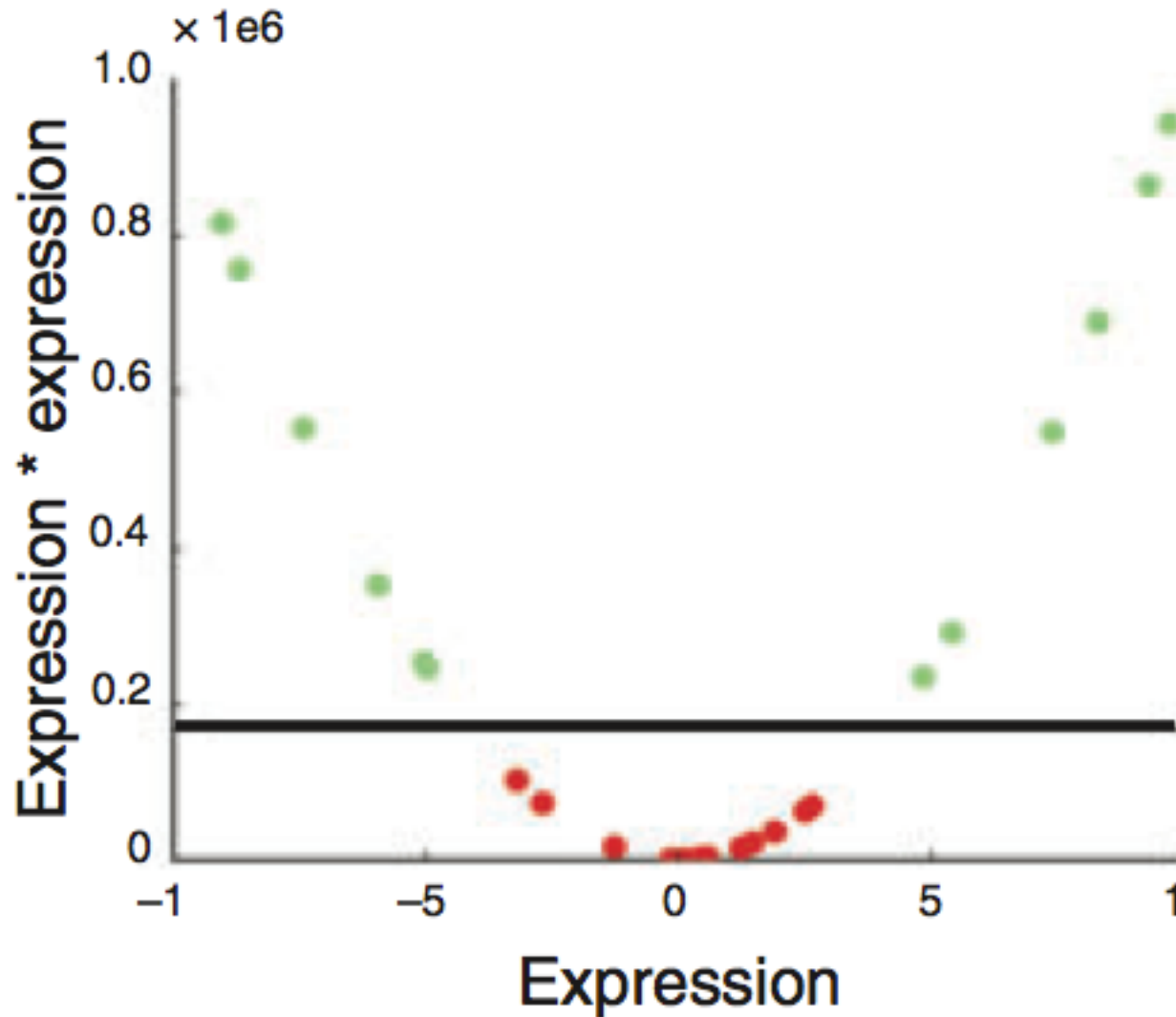


- Are linear separating hyperplanes enough?



**NO**

- 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 important trick** in SVM: to allow for the transformation, we only need to define the “**kernel function**”,  $k(x_i, x_j) = f(x_i) \cdot f(x_j)$
- The above example essentially uses a polynomial kernel

- Math behind the “kernel trick”

- In optimization theory, a constrained optimization problem can be formulated into its dual problem (the original problem is called primal problem)

- The dual formulation of SVM can be expressed as:

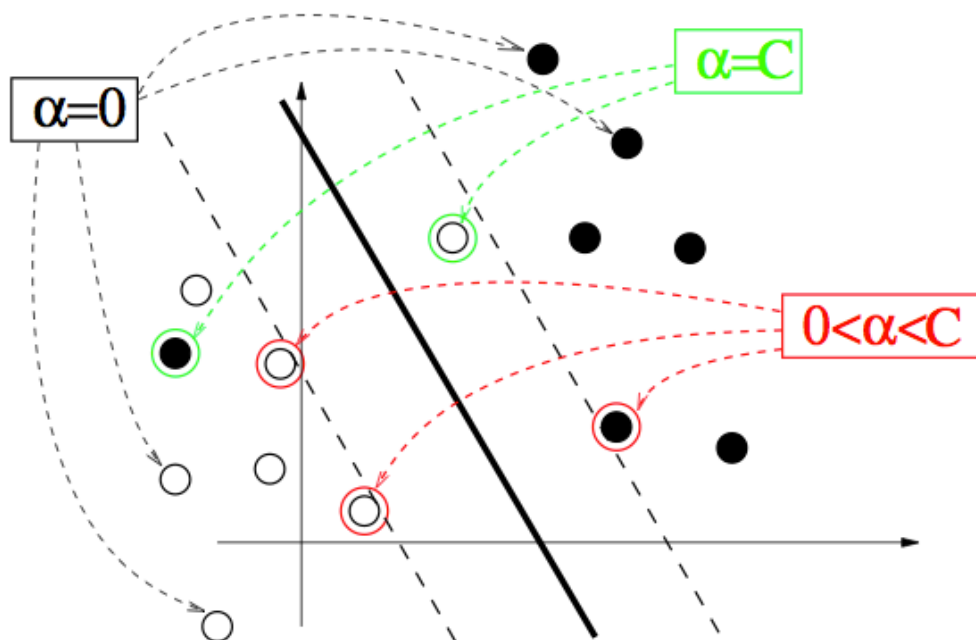
- Maximize  $\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j x_i \bullet x_j$ , subject to

$$\sum_i y_i \alpha_i = 0, 0 \leq \alpha_i \leq C$$

Complicated!

- The “Kernel”:  $x_i \bullet x_j$ , can be replaced by more sophisticated kernels:  $k(x_i, x_j) = f(x_i) \bullet f(x_j)$

- “Support vector machine”, where does the name come from?



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

# Key idea in the **Kernel Trick**

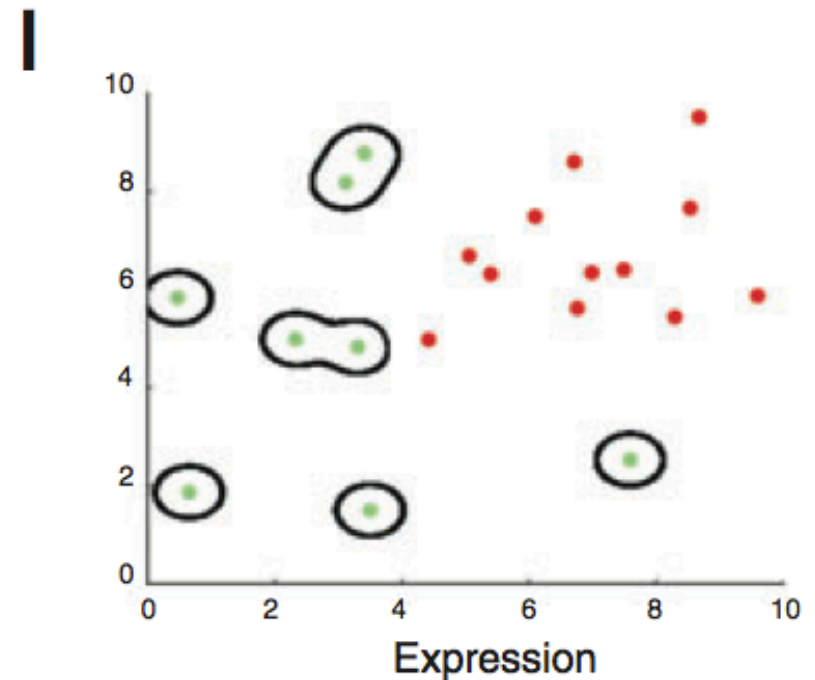
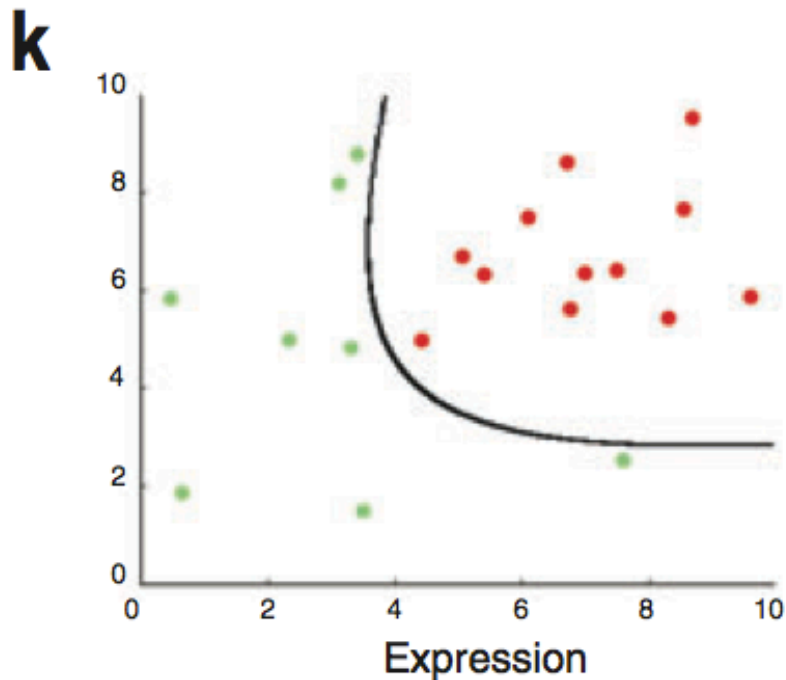
- Original SVM optimization for refining the hyperplane parameters  $w$  &  $b$  in terms of a linear combination of  $x_i$  can be replaced by a different optimization problem using "Lagrange multipliers"  $\alpha_i$ 
  - One only optimizes using the product of  $x_i * x_j$ , now expressing the solution in terms of  $\alpha_i$  which are non-zero for  $x_i$  that function as support vectors
- In a 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
  - This is further formalized in the kernel trick where  $f(x_i) * f(x_j)$  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

- Two commonly used kernels (and there are more)
- Polynomial kernel:
  - $k(x_i, x_j) = (x_i \bullet x_j + a)^d$
  - $a = 1$  (inhomogeneous) or  $0$  (homogeneous)
  - $d$  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

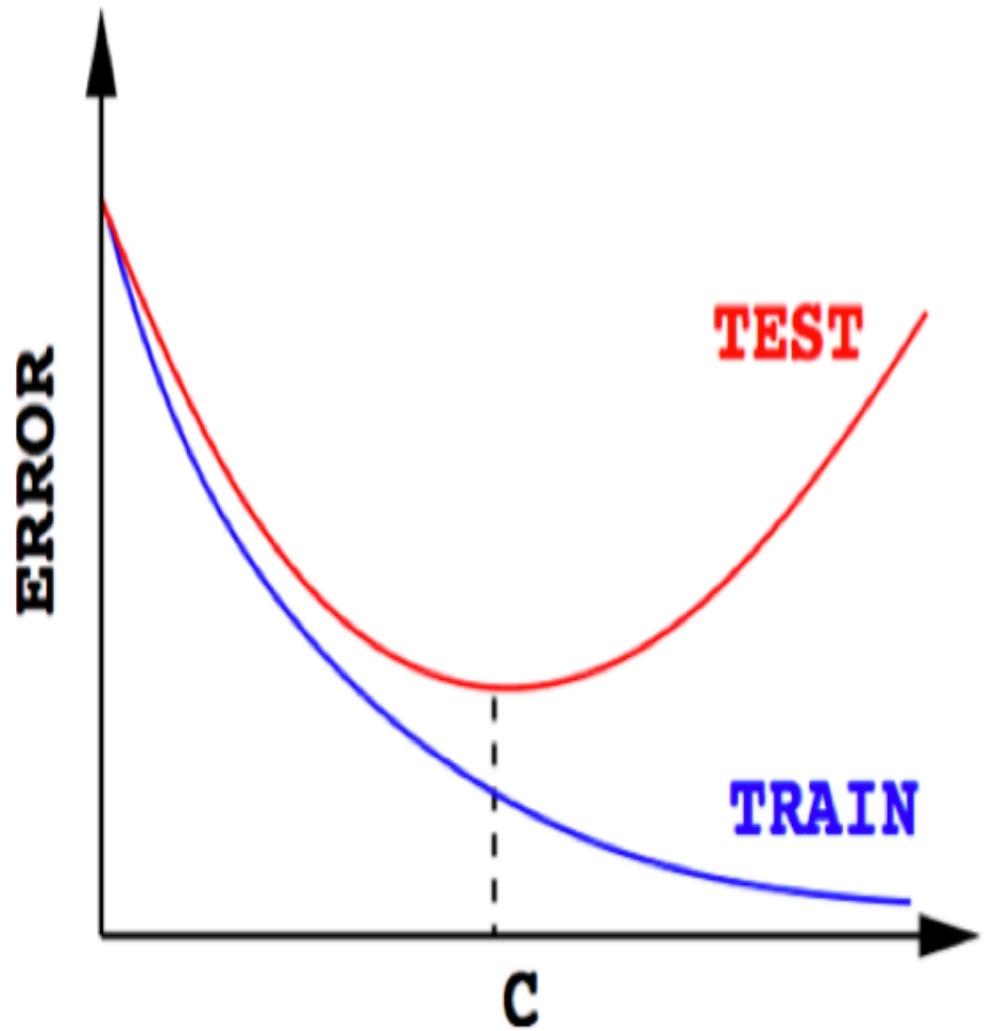
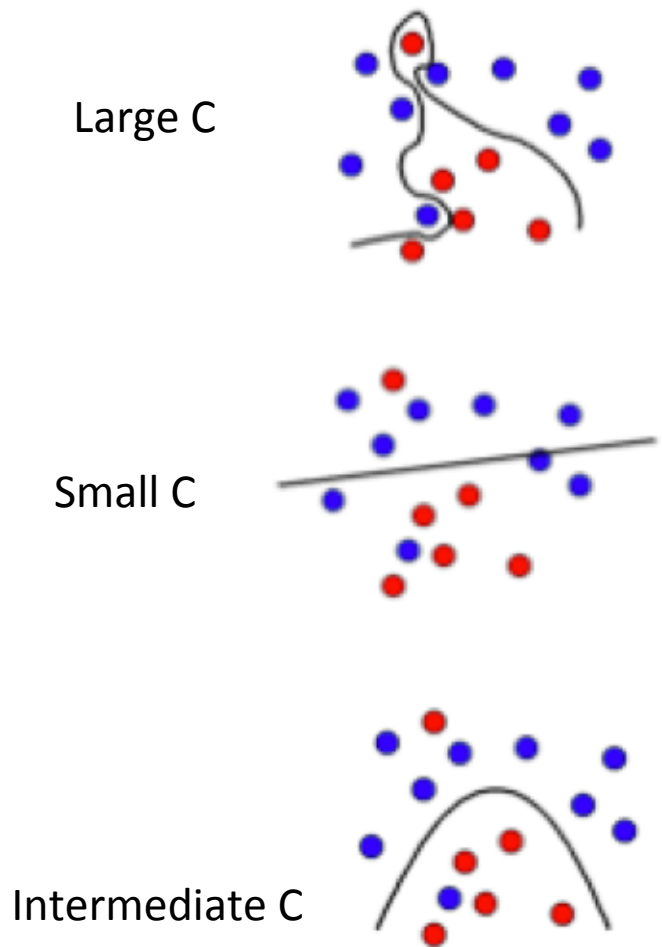
- More about kernels
  - With kernels, non-vector data can be easily handled – we only need to define the kernel function between two objects
  - Examples of non-vector biological data include: DNA and protein sequences (“string kernels”), nodes in metabolic or protein-protein interaction networks, microscopy images, etc
  - Allows for combining different types of data naturally – define kernels on different data types and combine them with simple algebra
- Questions for practitioners: Which kernel to use? How to choose parameters?
  - Trial and error
  - Cross-validation
- 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)

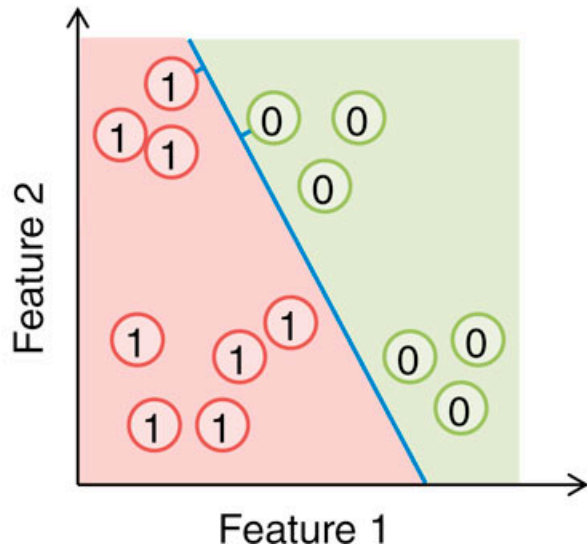


- The parameter  $C$  has a similar role
  - Large  $C$  will make **few classification errors** on the **training data**
  - But this may not generalize to the **testing data**
  - Small  $C$  pursues a **large separating margin** at the expenses of some classification errors on the training data.
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

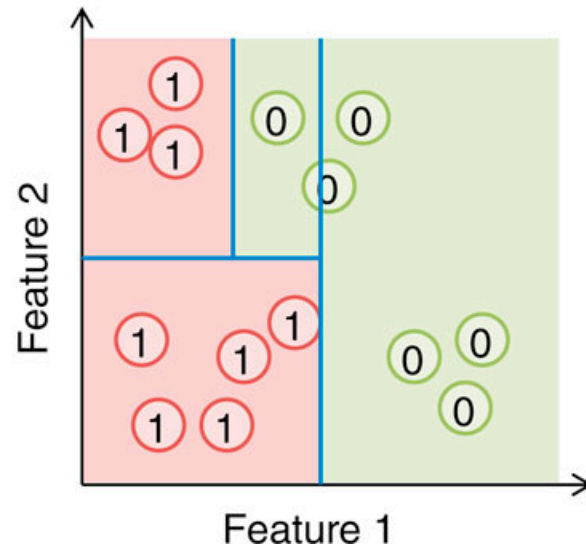


# Decision boundaries: SVM v Tree v Nearest NBR

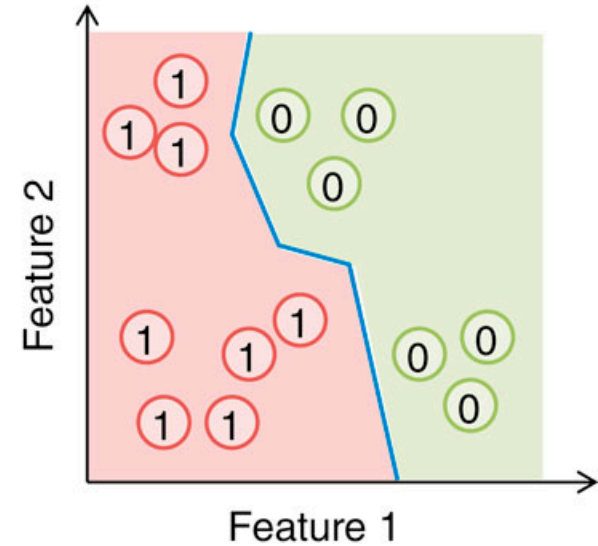
(a)



(b)



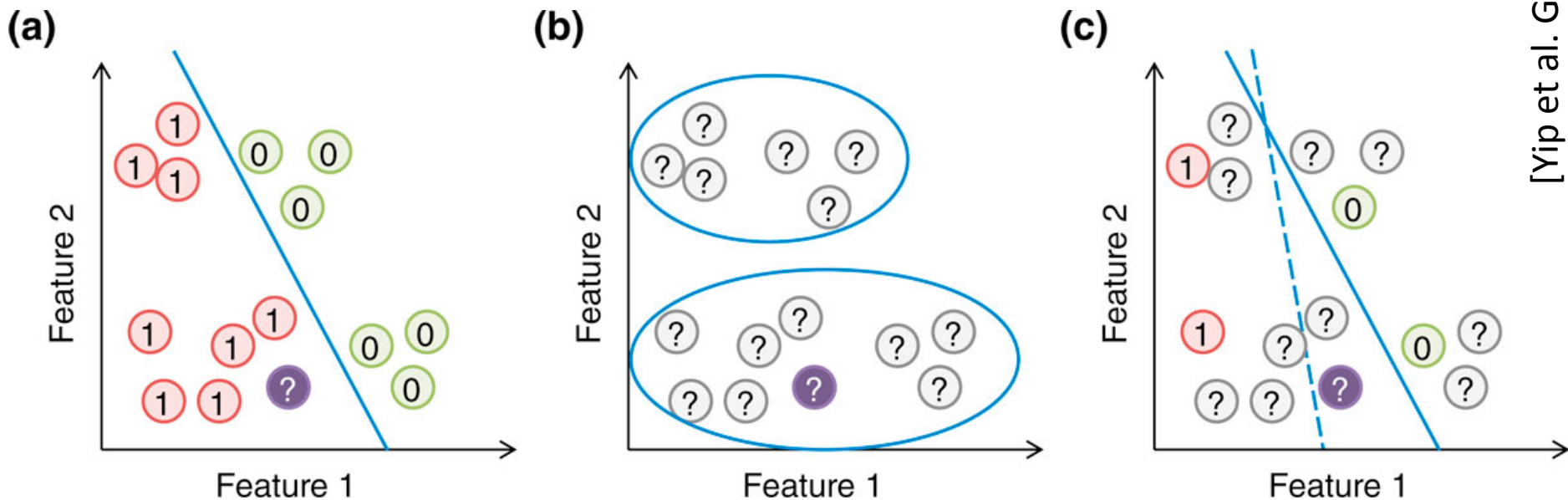
(c)



(a) A support vector machine (SVM) forms an affine decision surface (a straight line in the case of two dimensions) in the original feature space or a vector space defined by the similarity matrix (the kernel), to separate the positive and negative examples and maximize the distance of it from the closest training examples (the support vectors, those with a perpendicular line from the decision surface drawn). It predicts the label of a genomic region based on its direction from the decision surface. In the case a kernel is used, the decision surface in the original feature space could be highly non-linear. (b) A basic decision tree uses feature-parallel decision surfaces to repeatedly partition the feature space, and predicts the label of a genomic region based on the partition it falls within. (c) The one-nearest neighbor (1-NN) method predicts the label of a genomic region based on the label of its closest labeled example. In all three cases, the areas predicted to be positive and negative are indicated by the red and green background colors, respectively.

# Semi-supervised Methods

- Supervised & Unsupervised:  
Can you combine them? YES
  - RHS (c) shows modifying the optimum decision boundary in (a) by "clustering" of unlabeled points



Supervised, unsupervised and semi-supervised learning. (a) In supervised learning, the model (blue line) is learned based on the positive and negative training examples, and the genomic region without a known class label (purple circle) is classified as positive according to the model. (b) In unsupervised learning, all examples are unlabeled, and they are grouped according to the data distribution. (c) In semi-supervised learning, information of both labeled and unlabeled examples is used to learn the parameters of the model. In this illustration, a purely supervised model (dashed blue line) classifies the purple object as negative, while a semi-supervised model that avoids cutting at regions with a high density of genomic regions (solid blue line) classifies it as positive.