MATH4280

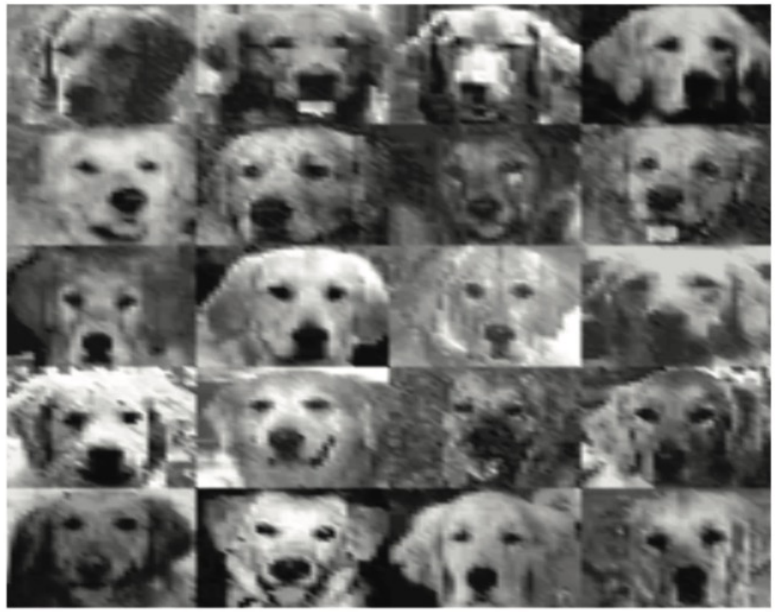
Lecture Notes 5: Basics of machine learning

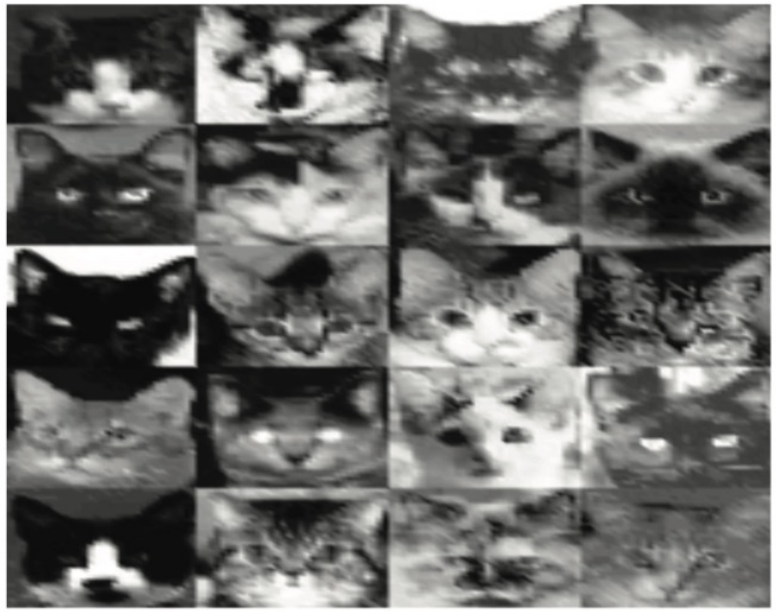
Overview

- Supervised vs unsupervised machine learning
- Supervised learning: labels are given to training data
- Unsupervised learning: no label is given to training data
- Our goal: discuss some common supervised and unsupervised learning
- Key point: finding a low-rank feature space that is informative and interpretable

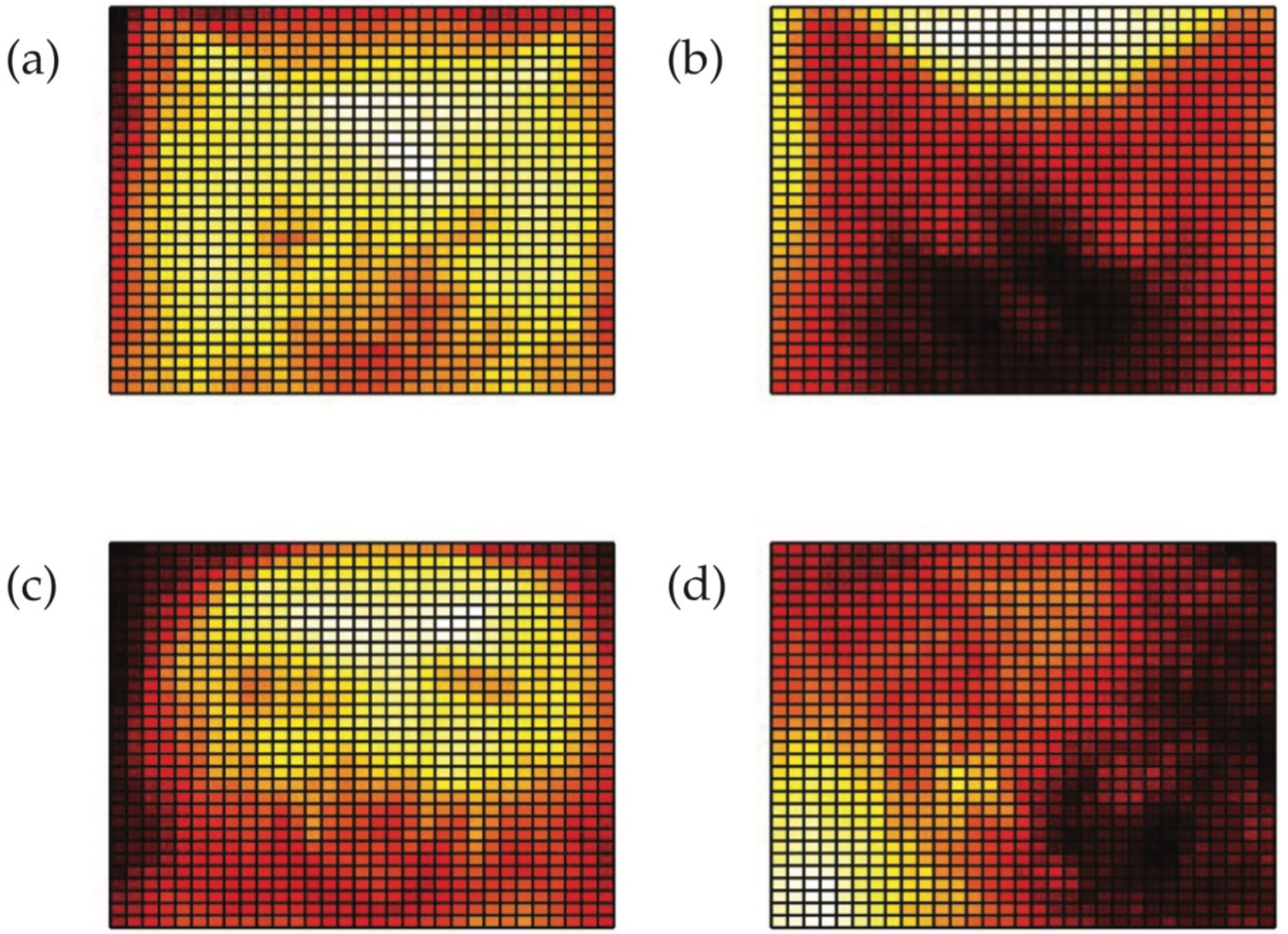
Feature selection

- SVD or PVA can be used to extract main features
- For example, consider a set of dog and cat images
- Perform SVD or PCA as in the eigenface example

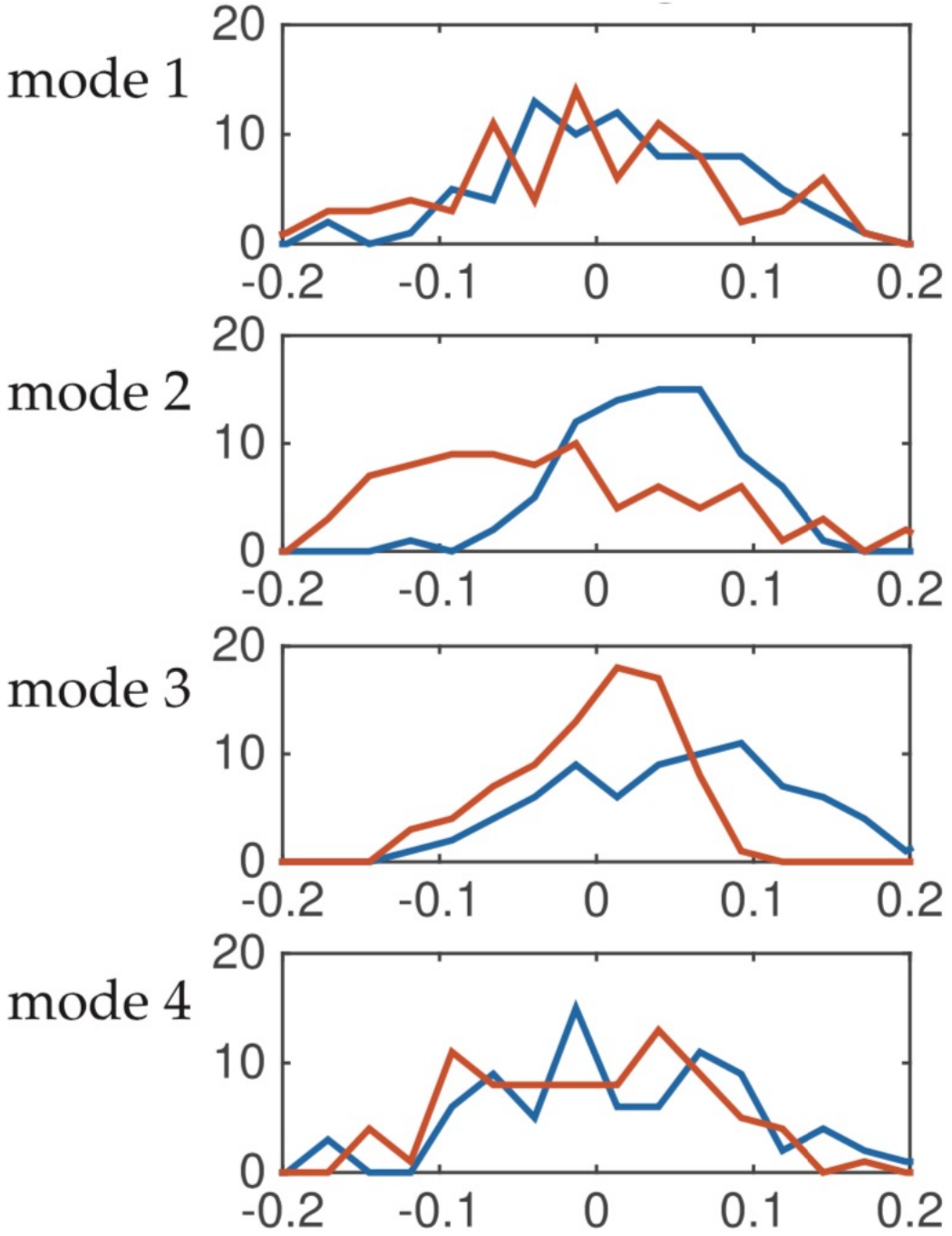




- The first 4 PCA modes (vectors U in SVD)



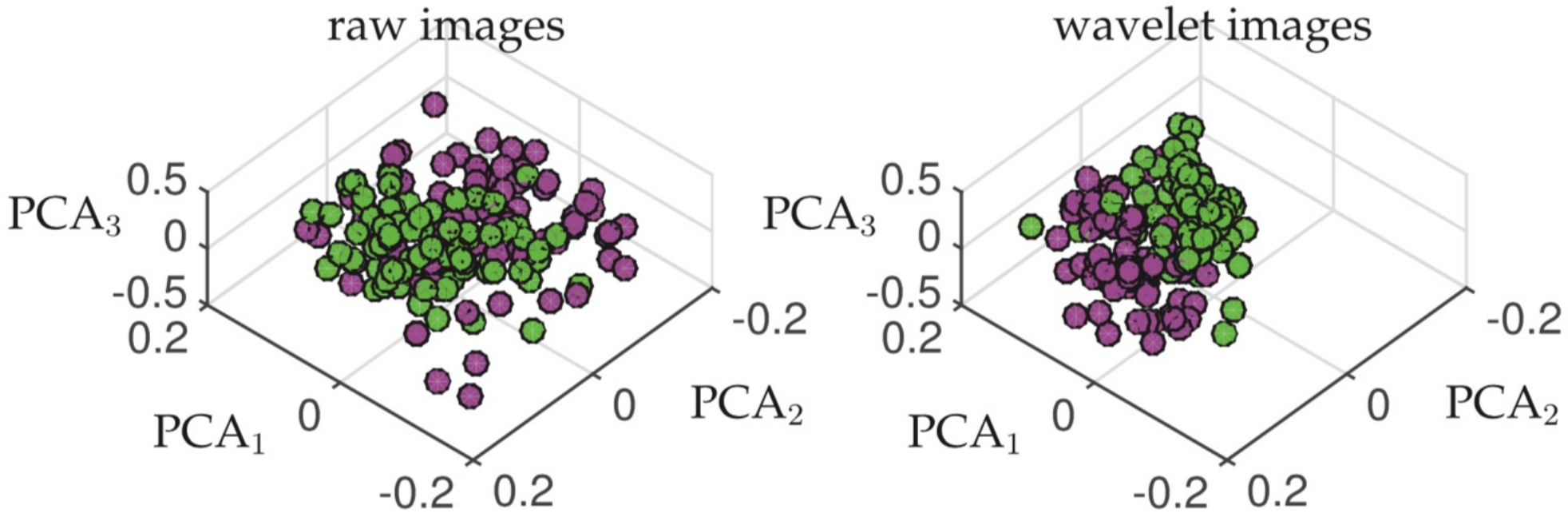
- Note that, each column of the matrix V determines the loading of each feature onto a specific image
- Blue (dogs)
- Red (cats)
- We observe that the second mode can be used to distinguish dogs and cats



Remark on using wavelets

- It is beneficial to preprocess the data using wavelet transforms

Dogs (Green)
Cats (Magneta)



The first 3 PCA modes: more separation is observed when wavelet transform is used

Supervised vs unsupervised learning

- A general mathematical definition: let

$$\mathcal{D} \subset \mathbb{R}^n$$

where $\mathcal{D}$ is an open bounded set

- We also let

$$\mathcal{D}' \subset \mathcal{D}$$

- The goal of classification is to build a classifier labelling for all data in $\mathcal{D}$ given the data from $\mathcal{D}'$

- To be more precise, consider a set of data points $x_j \in \mathbb{R}^n$ and labels y_j for each point, where $j = 1,2, \dots, m$
- The labels can be in many forms. For example, $y_j = \{\pm 1\}$
- Unsupervised learning can be formulated as

Input

data $\left\{\mathbf{x}_j \in \mathbb{R}^n,\ \ j \in Z := \{1, 2, \cdots, m\}\right\}$

Output

labels $\left\{\mathbf{y}_j \in \{\pm 1\},\ \ j \in Z\right\}.$

- Supervised learning is formulated as

Input

data $\left\{\mathbf{x}_j \in \mathbb{R}^n,\ \ j \in Z := \{1, 2, \cdots, m\}\right\}$

labels $\left\{\mathbf{y}_j \in \{\pm 1\},\ \ j \in Z' \subset Z\right\}$

Output

labels $\left\{\mathbf{y}_j \in \{\pm 1\},\ \ j \in Z\right\}$.

- Note that the data used to train the classifier is from $\mathcal{D}'$
- The classifier is then applied (generalize) to $\mathcal{D}$

Example

- In the dogs and cats example, the data and label are defined as

$$\mathbf{x}_j = \{64 \times 64 \text{ image} = 4096 \text{ pixels}\}$$

$$\mathbf{y}_j = \{\text{dog, cat}\} = \{1, -1\}$$

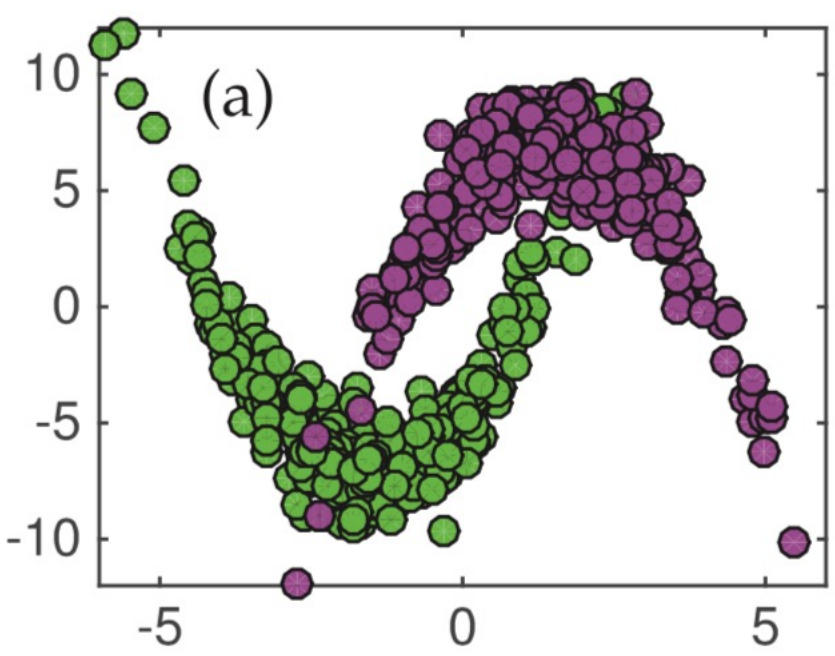
- The data sets are defined as

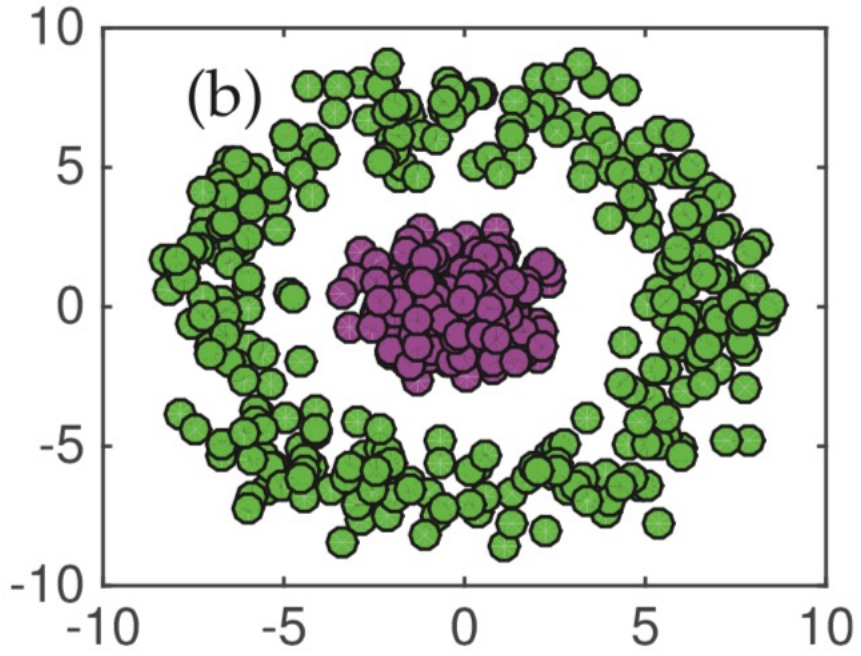
$$\mathcal{D}' \in \{160 \text{ image samples: } 80 \text{ dogs and } 80 \text{ cats}\}$$

$$\mathcal{D} \in \{\text{the universe of dogs and cats}\}$$

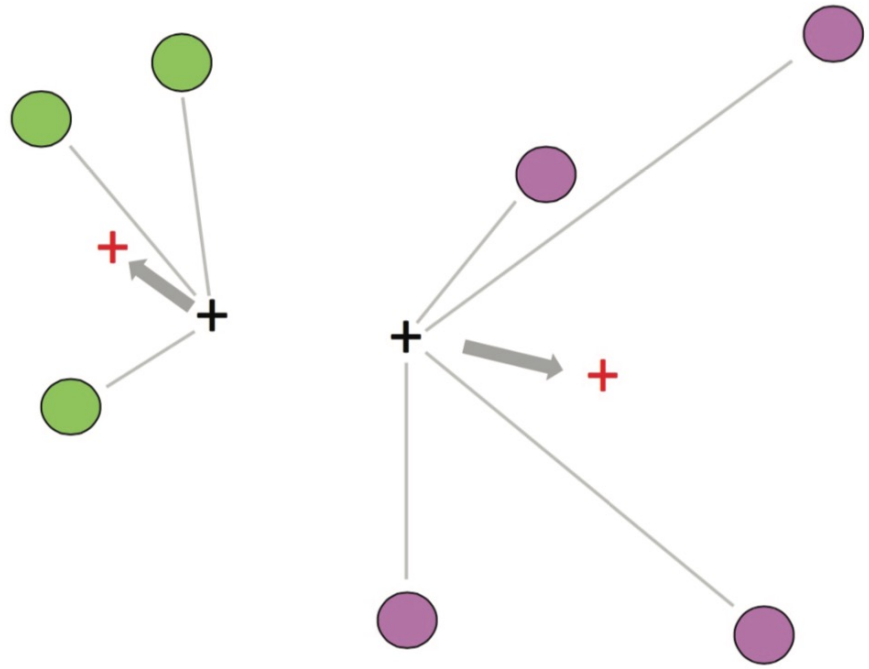
Some difficulties

- If the boundary of data forms a nonlinear manifold, then it is often difficult to characterize the data
- Examples:

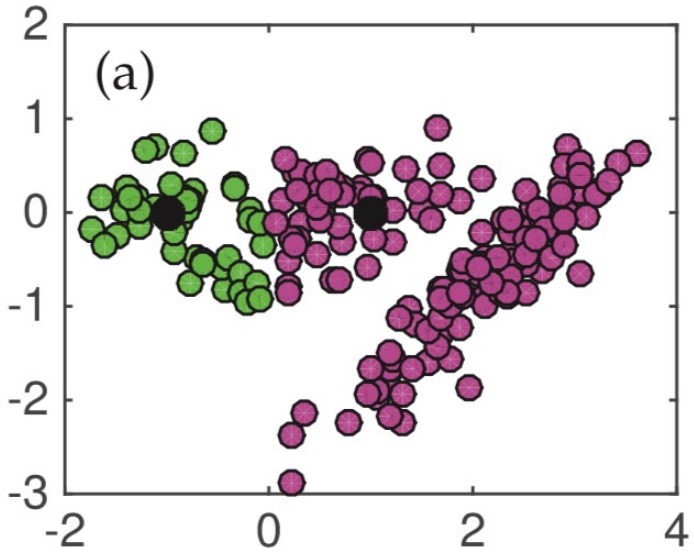


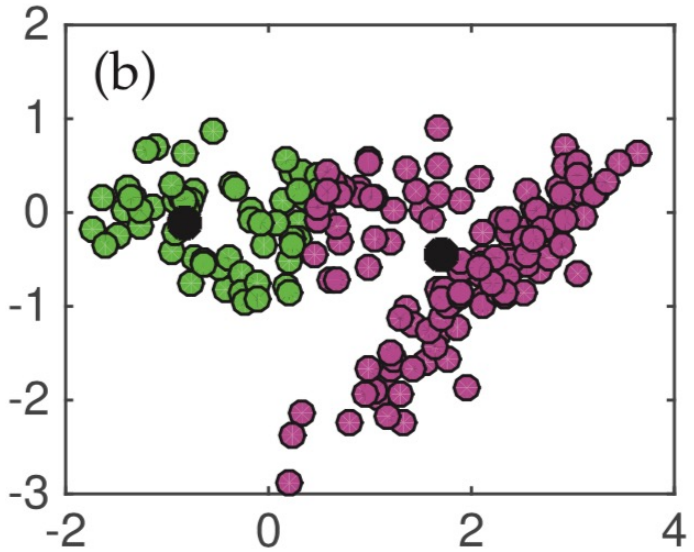


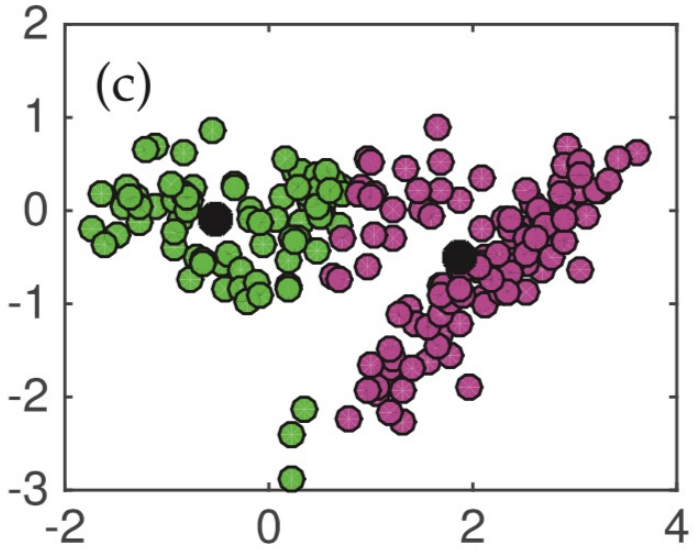
K-means clustering

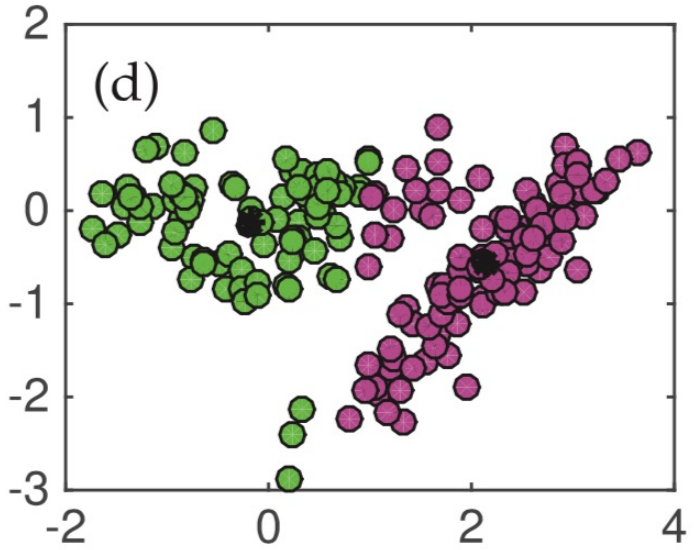


- A popular unsupervised learning algorithm
- Goal: partitioning m observations into k clusters
- Algorithm
  1. Assume initial means of the k clusters
  2. Compute the distance of each observation to each of the k means
  3. Label each observation as belonging to the nearest mean
  4. Find the mean for each cluster
  5. Repeat
- Formally, the problem can be stated as $\underset{\boldsymbol{\mu}_j}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{\mathbf{x}_j \in \mathcal{D}'_j} \|\mathbf{x}_j - \boldsymbol{\mu}_j\|^2$
- A heuristic algorithm





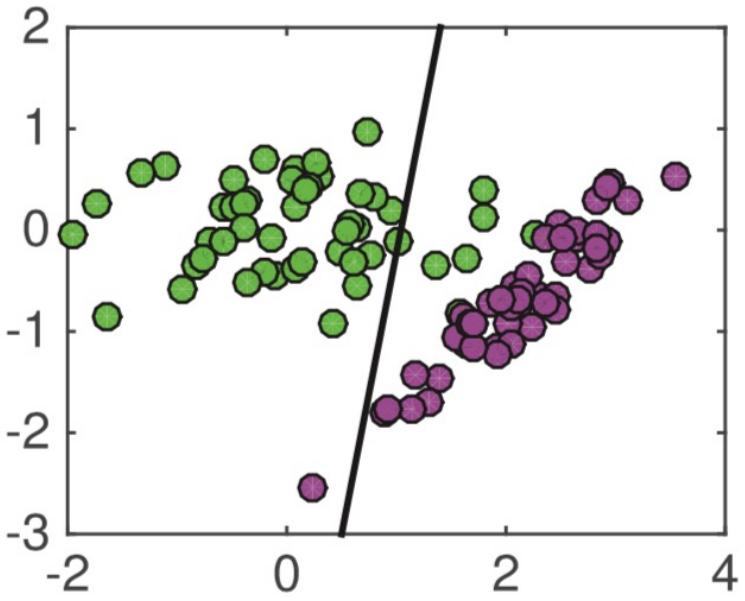




The algorithm converges in 4 iterations

- Data divided into training and testing sets
- Training set is used to form the clusters
- Testing set is used to show the performance of the clustering
- Success based on (1) no supervision is needed (2) fast algorithm

Testing set: some observations are incorrectly predicted



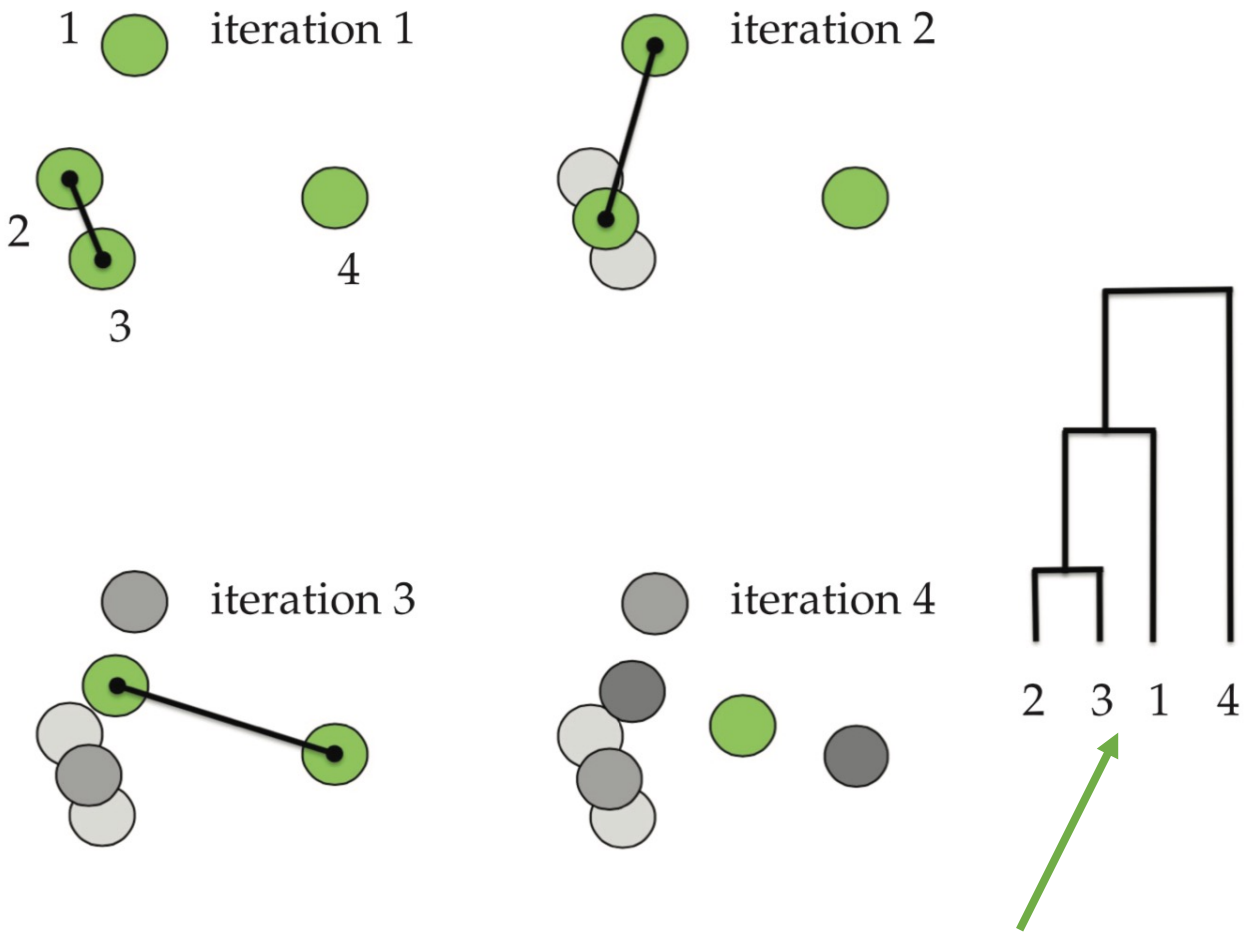
Black line = separation obtained by the k-means algorithm

Dendrogram

- An unsupervised clustering algorithm
- It is an example of hierarchical clustering method
- Two types:
- Agglomerative (bottom-up)
  - Each point is its own cluster initially
  - The data is merged in pairs as one creates a hierarchy of clusters
- Divisive (top-down)
  - All data points form a single cluster
  - The data is divided into smaller clusters

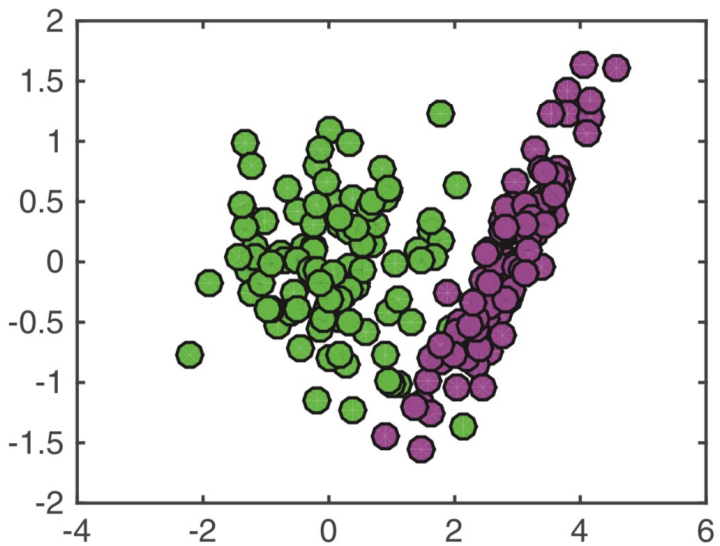
Agglomerative hierarchical clustering

- Distances between all m points are computed
- The closest two points are merged into a single data point midway between them
- Repeat with the new $m - 1$ points
- Resulting in a dendrogram

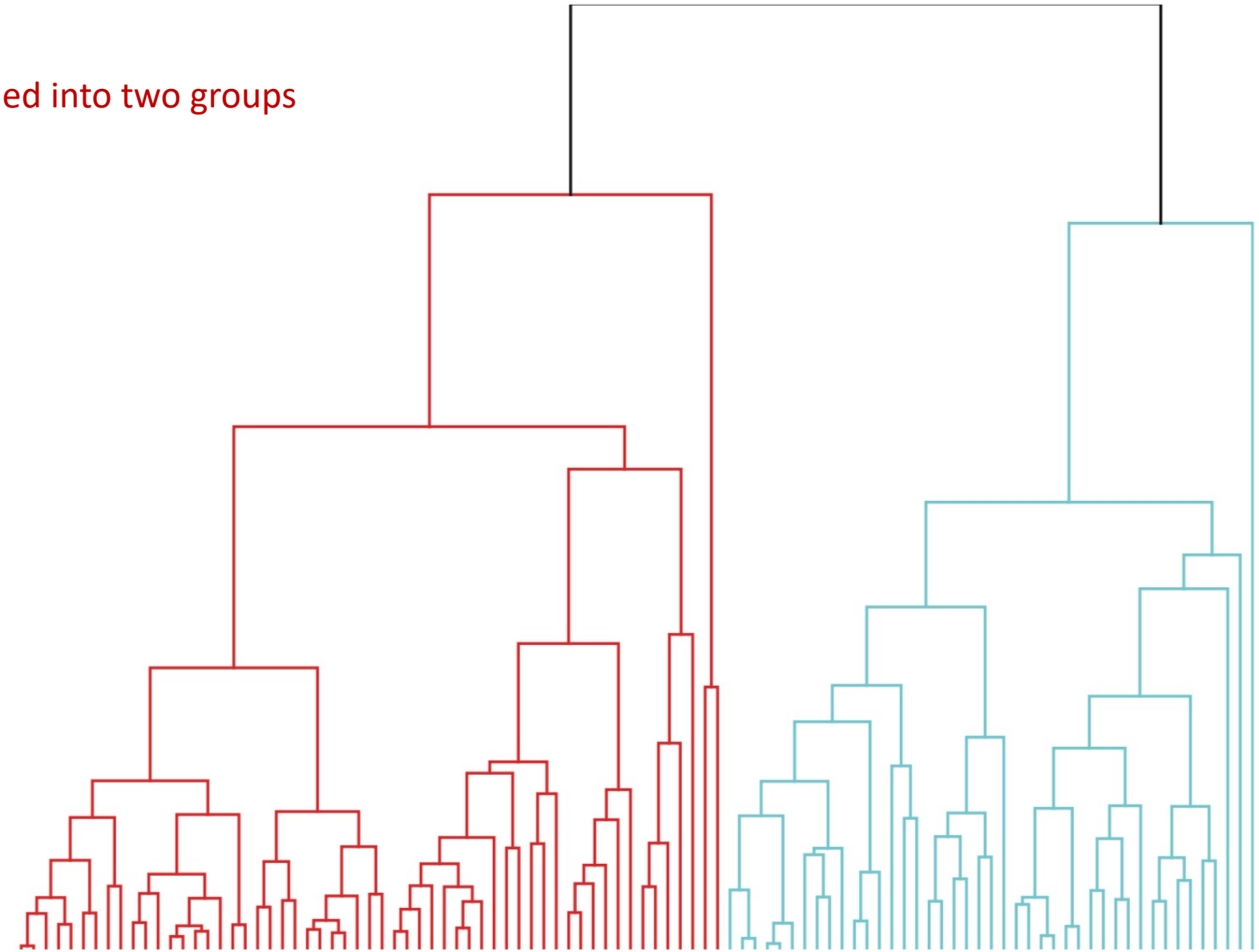


Note: the length of the branches are directly related to the distance between the merged points

The data points are systematically divided into two groups

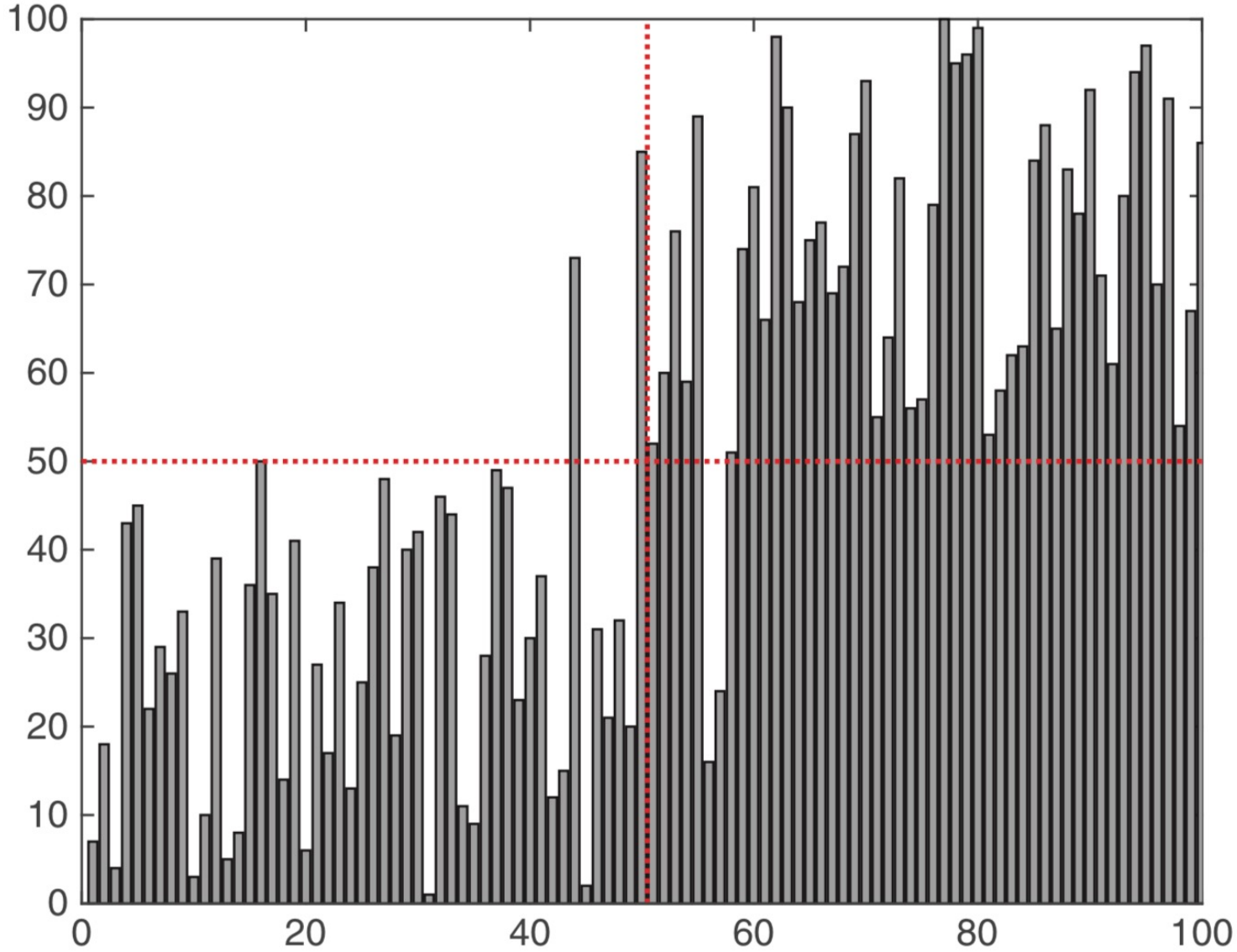


Data points



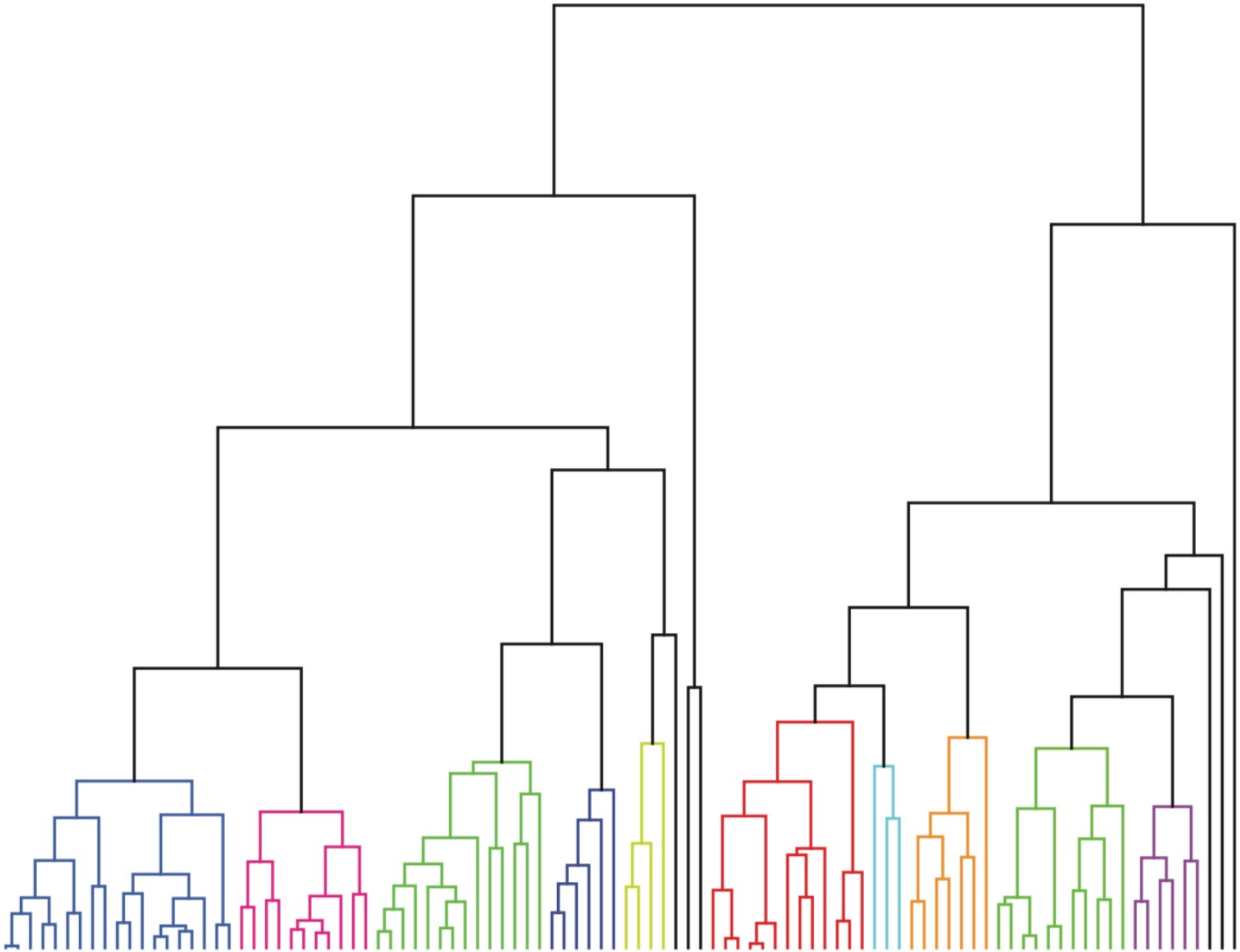
Dendrogram

Permuted index



We observe that some points are incorrectly placed

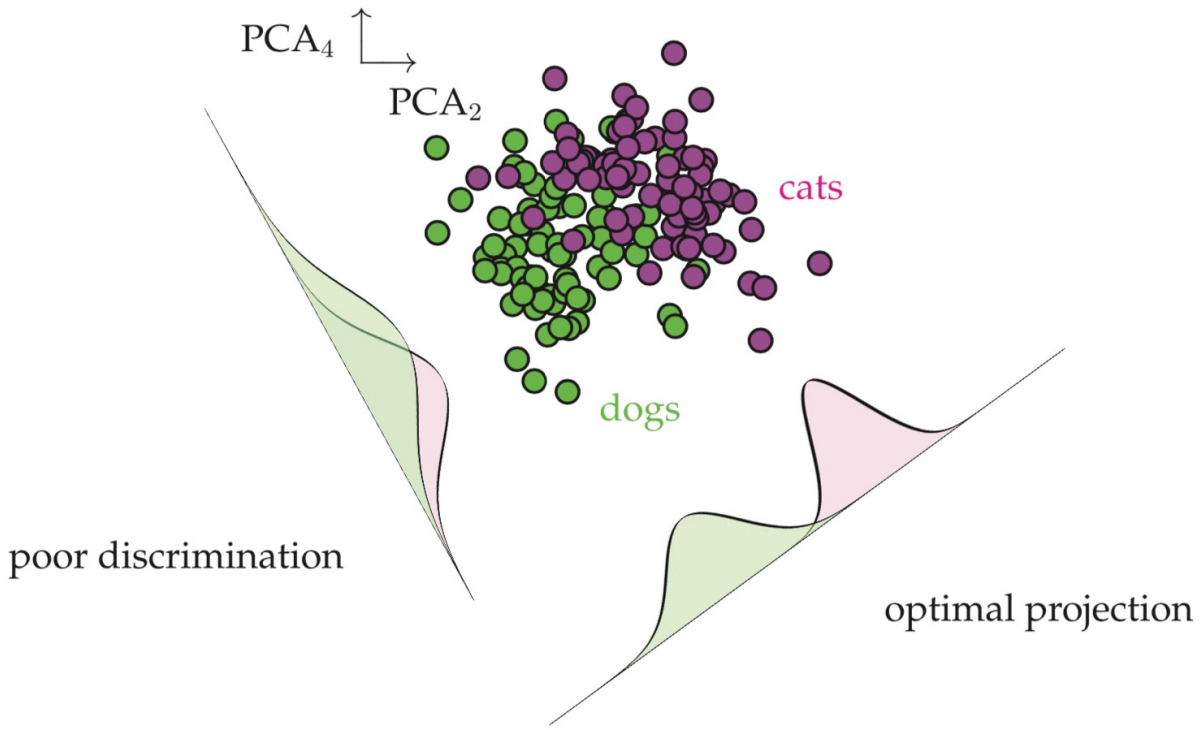
Input index: first 50 points belong to one group



More clusters are obtained if we set the threshold at a different level

Linear discriminant analysis (LDA)

- A supervised algorithm
- The goal is to find a linear combination of features that separate two or more classes of objects
- Use of labelled data
- Find a projection that maximize distance between inter-class data while minimizing intra-class data



- Mathematically, we construct a projection w such that

projection w

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

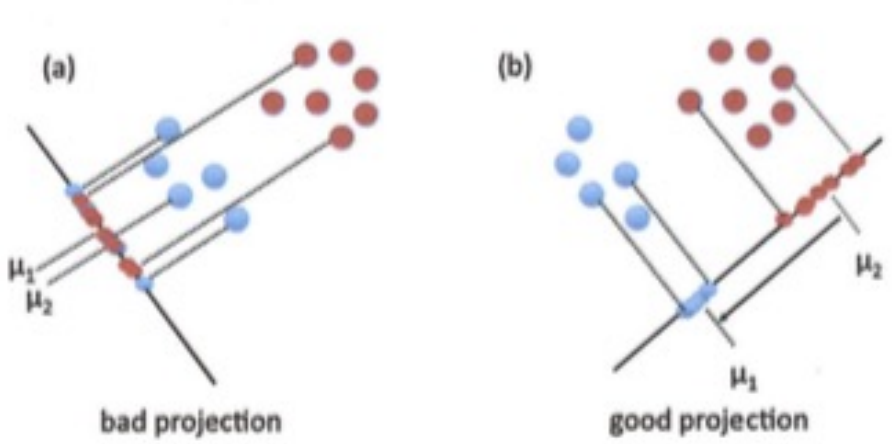
maximize distance between inter-class data

minimize distance between intra-class data

where the between-class S_B and within-class S_W matrices are

$$\mathbf{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T$$

$$\mathbf{S}_W = \sum_{j=1}^{2} \sum_{\mathbf{x}} (\mathbf{x} - \mu_j)(\mathbf{x} - \mu_j)^T$$



- The above is equivalent to the generalized eigenvalue problem

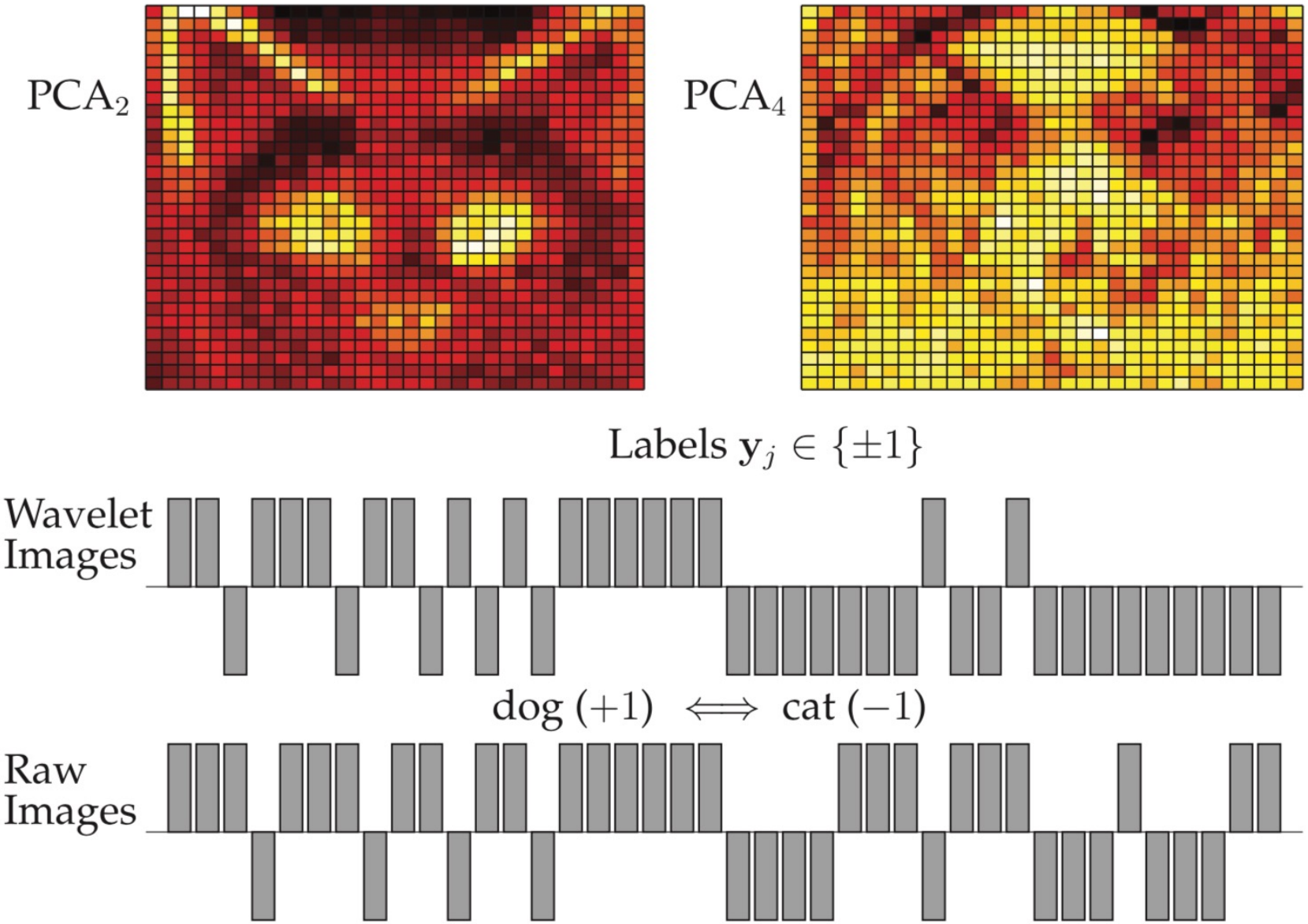
generalized Rayleigh quotient

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w}$$

This gives a set of eigenvalue and eigenvector pairs.
We sort them by magnitude to get the most significant ones.

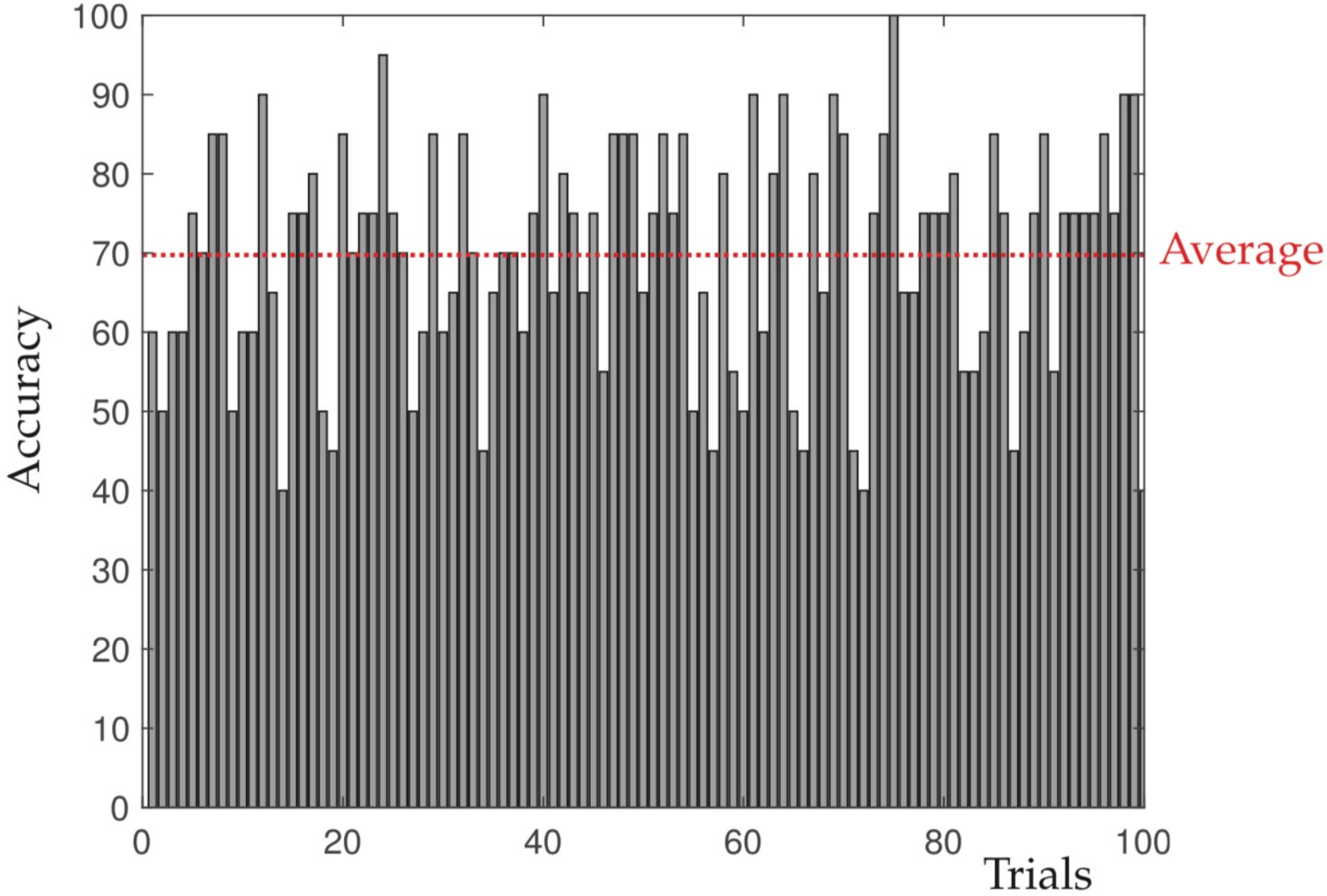
Example

- Use 60 cat/dog images for training, and 20 images for testing
- Use the second and the fourth PCA modes (data points in 2D)
- Consider both the raw images and those wavelet transformed images
- The training data are used for finding S_B and S_B , and the best projection vector w
- For each testing data, project it to the vector w
- Need to find a way to separate the projected data points
- Labels are 1 (dog) or -1 (cat)



Prediction

Accuracy of 100 trials

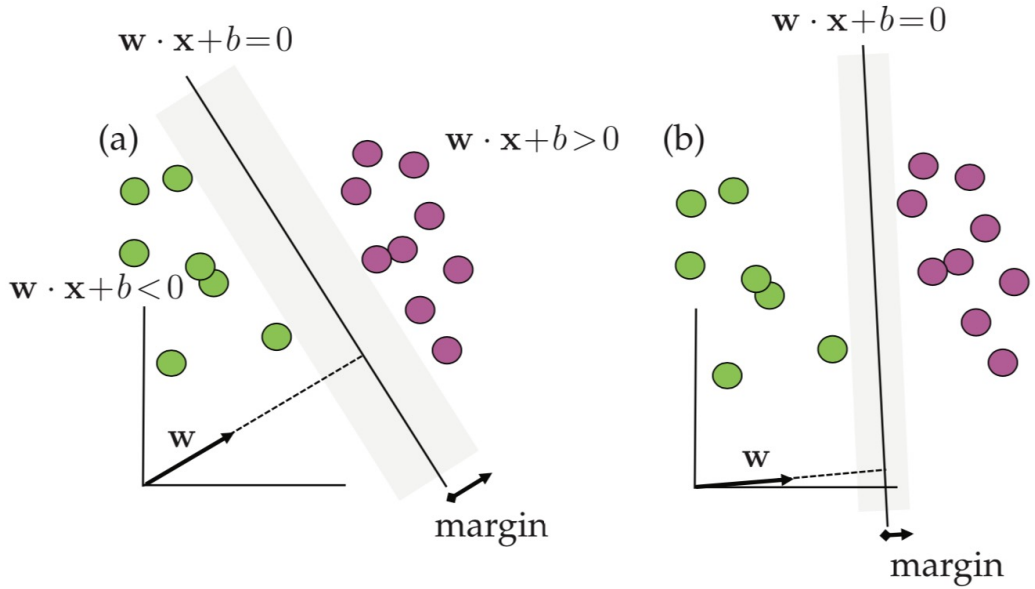


Support vector machines (SVM)

- A popular supervised machine learning method
- The main idea of linear SVM is to find a hyperplane that separate the data points

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

- The hyperplane is parameterized by two vectors w and b
- The SVM optimize the decision line and maximize the margin



- Each data point x_j can be classified by the sign of $w \cdot x_j + b$
- The labels $y_j = \{\pm 1\}$ are defined by

$$\mathbf{y}_j(\mathbf{w} \cdot \mathbf{x}_j + b) = \text{sign}(\mathbf{w} \cdot \mathbf{x}_j + b) = \begin{cases} +1 & \text{magenta ball} \\ -1 & \text{green ball.} \end{cases}$$

- We find the hyperplane by optimizing an objective (loss) function

$$\ell(\mathbf{y}_j, \bar{\mathbf{y}}_j) = \ell(\mathbf{y}_j, \text{sign}(\mathbf{w} \cdot \mathbf{x}_j + b)) = \begin{cases} 0 & \text{if } \mathbf{y}_j = \text{sign}(\mathbf{w} \cdot \mathbf{x}_j + b) \\ +1 & \text{if } \mathbf{y}_j \neq \text{sign}(\mathbf{w} \cdot \mathbf{x}_j + b) \end{cases}$$

- Note that this objective (loss) function satisfies

$$\ell(\mathbf{y}_j, \bar{\mathbf{y}}_j) = \begin{cases} 0 & \text{if data is correctly labeled} \\ +1 & \text{if data is incorrectly labeled} \end{cases}$$

- The goal is to minimize the loss function and make the margin as large as possible
- The optimization problem can be formulated as

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{j=1}^{m} \ell(\mathbf{y}_j, \bar{\mathbf{y}}_j) + \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad \text{Min } |x_j \cdot w + b| = 1$$

why we also add another term 0.5 ||w||^2 in the training process?

why the boundary is subjected to this?

- Note that it is more convenient to replace the discrete loss function by a continuous loss function

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{j=1}^{m} H(\mathbf{y}_j \;\; \bar{\mathbf{y}}_j) + \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad \text{Min } |x_j \cdot w + b| = 1$$

- The loss function $H(z) = \max(0, 1 - z)$ is called the Hinge loss function

Nonlinear SVM

So this practice is trying to generate non-existing parameters for the set of data

- Linear SVM is simple, but with limited practical value

Q: We can have non-linear function on the same dimensional space as well. Why we choose to project the Kernel to orthogonal space?

- The main idea of nonlinear SVM is to map each point x to a higher dimensional space

$$\mathbf{x} \mapsto \mathbf{\Phi}(\mathbf{x})$$

Q1: How can this method obtains unique answer from one training?
Q2: Is converging always happens at the end of training?

- We call $\Phi(x)$ the new observables of the data
- The SVM learns a hyperplane that splits the data in the feature space

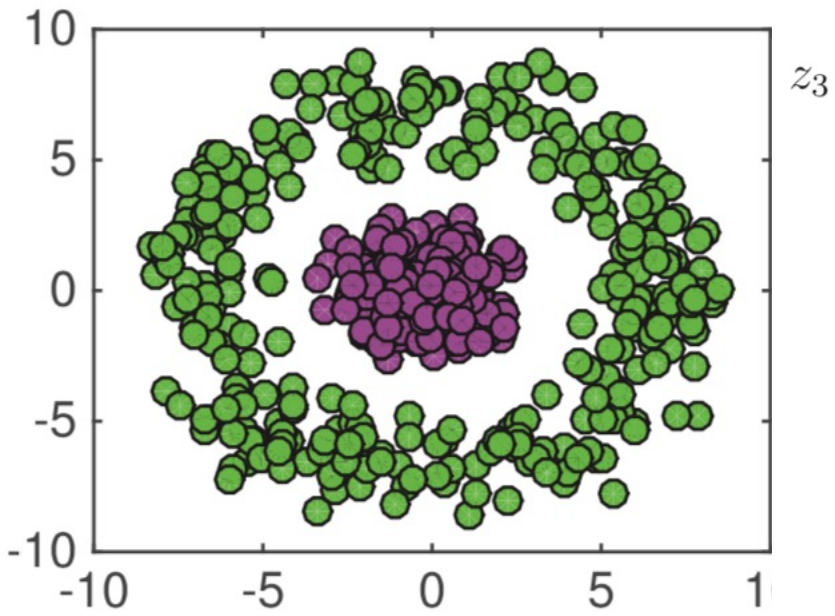
$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}) + b$$

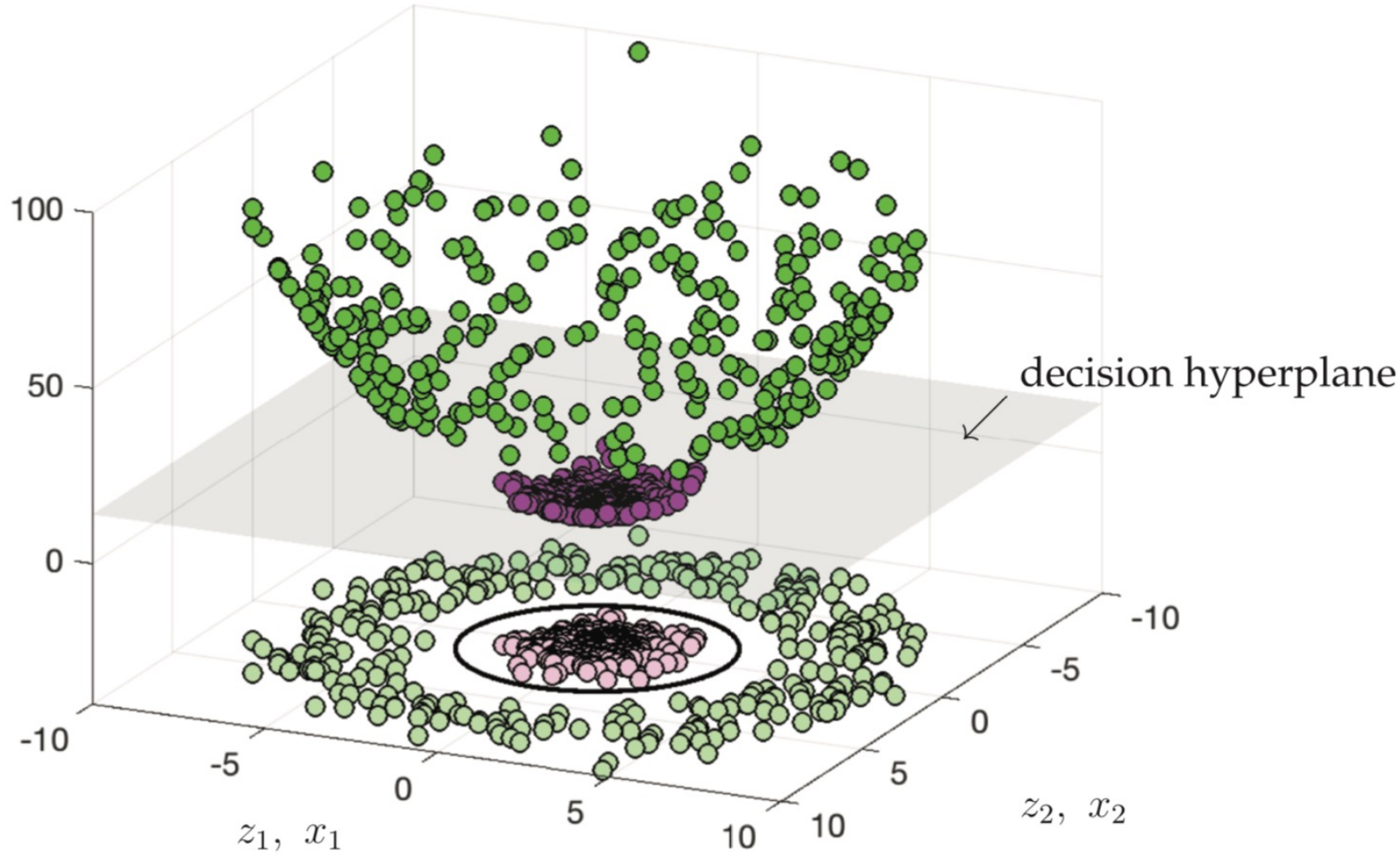
- For example, one can define the following map for the observables

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1, x_2, x_1^2 + x_2^2)$$

Example

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1, x_2, x_1^2 + x_2^2)$$





The feature map is able to separate the data points

Kernel method for SVM

- The vector w lies in a very high dimensional space, computing this vector is expensive
- The main idea of the kernel method is to represent w as follows
- $$\mathbf{w} = \sum_{j=1}^{m} \alpha_j \mathbf{\Phi}(\mathbf{x}_j)$$
- The hyperplane is defined as

$$f(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j \mathbf{\Phi}(\mathbf{x}_j) \cdot \mathbf{\Phi}(\mathbf{x}) + b$$

- We define the kernel function as

$$K(\mathbf{x}_j, \mathbf{x}) = \mathbf{\Phi}(\mathbf{x}_j) \cdot \Phi(\mathbf{x})$$

- The optimization problem is formulated as

Do we need to apply Kernel on this? Or just counting how many data points falls into the boundary line?

$$\underset{\boldsymbol{\alpha},b}{\operatorname{argmin}} \sum_{j=1}^{m} H(\mathbf{y}_j \;\; \bar{\mathbf{y}}_j) + \frac{1}{2}\| \sum_{j=1}^{m} \alpha_j \boldsymbol{\Phi}(\mathbf{x}_j)\|^2 \quad \text{subject to} \quad \text{Min } |x_j \cdot w + b| = 1$$

- The kernel function allows one to operate in an implicit high dimensional space without computing the coordinates of the data in that space
- It computes the inner products between all pairs of data in the feature space
- Some popular choices of kernel functions:

$$K(\mathbf{x}_j, \mathbf{x}) = \exp\left(-\gamma \|\mathbf{x}_j - \mathbf{x}\|^2\right)$$

Question 2: how to determine the coefficients of these Kernel functions?

Radial basis functions

$$K(\mathbf{x}_j, \mathbf{x}) = (\mathbf{x}_j \cdot \mathbf{x} + 1)^N$$

Polynomials

Decision tree

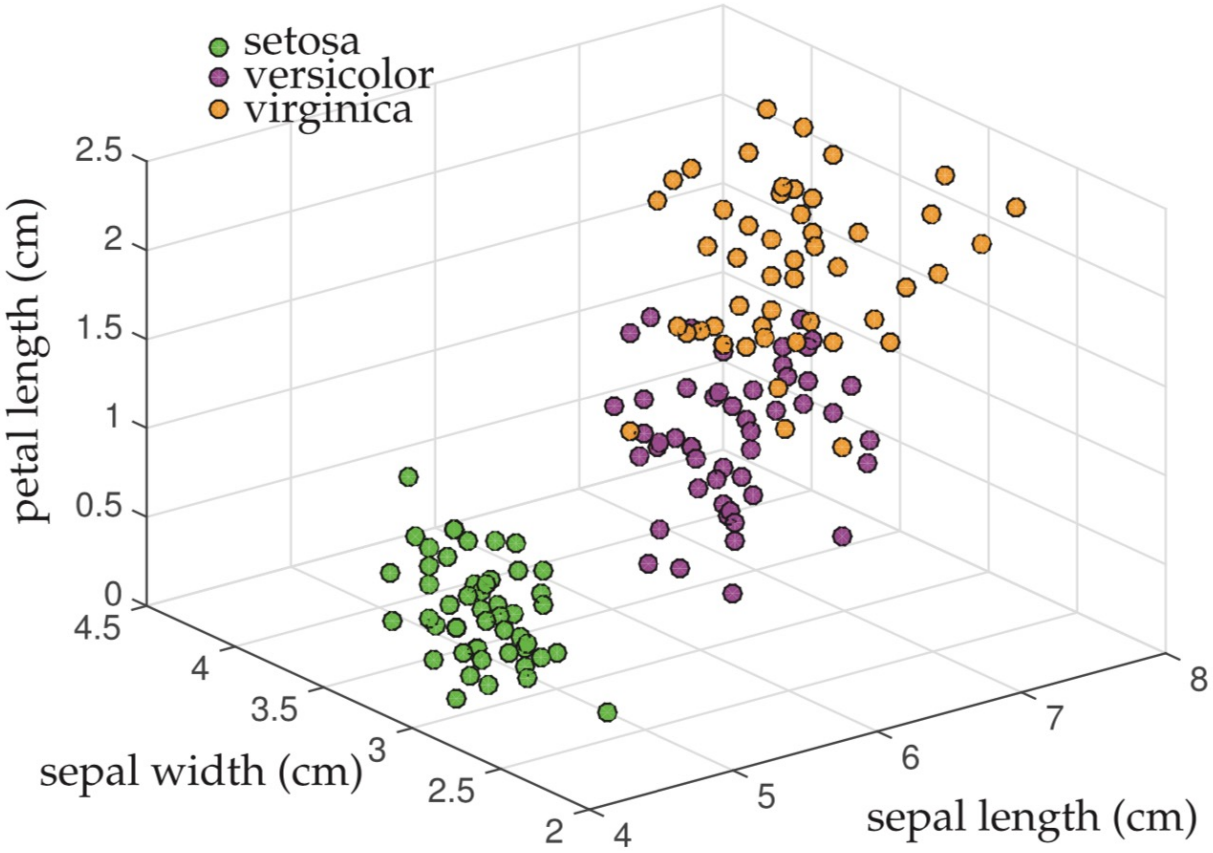
- The decision tree is a hierarchical construct that looks for optimal ways to split the data
- Consider again a set of labelled data

$$\text{data} \quad \left\{\mathbf{x}_j \in \mathbb{R}^n, \;\; j \in Z := \{1, 2, \cdots, m\}\right\}$$
$$\text{labels} \quad \left\{\mathbf{y}_j \in \{\pm 1\}, \;\; j \in Z' \subset Z\right\}.$$

- The basic decision tree algorithm:
  - Scan each component of the data to identify a component and a value that give the best predication of the labels
  - With the two new branches, the above process is repeated for each branch

Example

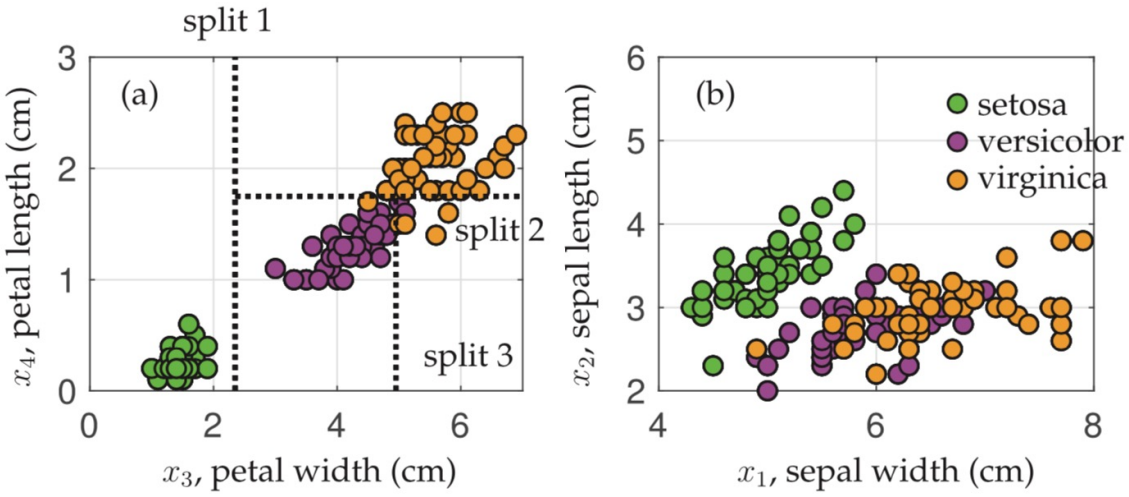
- Consider the MATLAB Fisher iris data set
- 150 irises: setosa, versicolor and virginica; 50 samples each type
- Each sample has 4 measurements: sepal length, sepal width, petal length and petal width

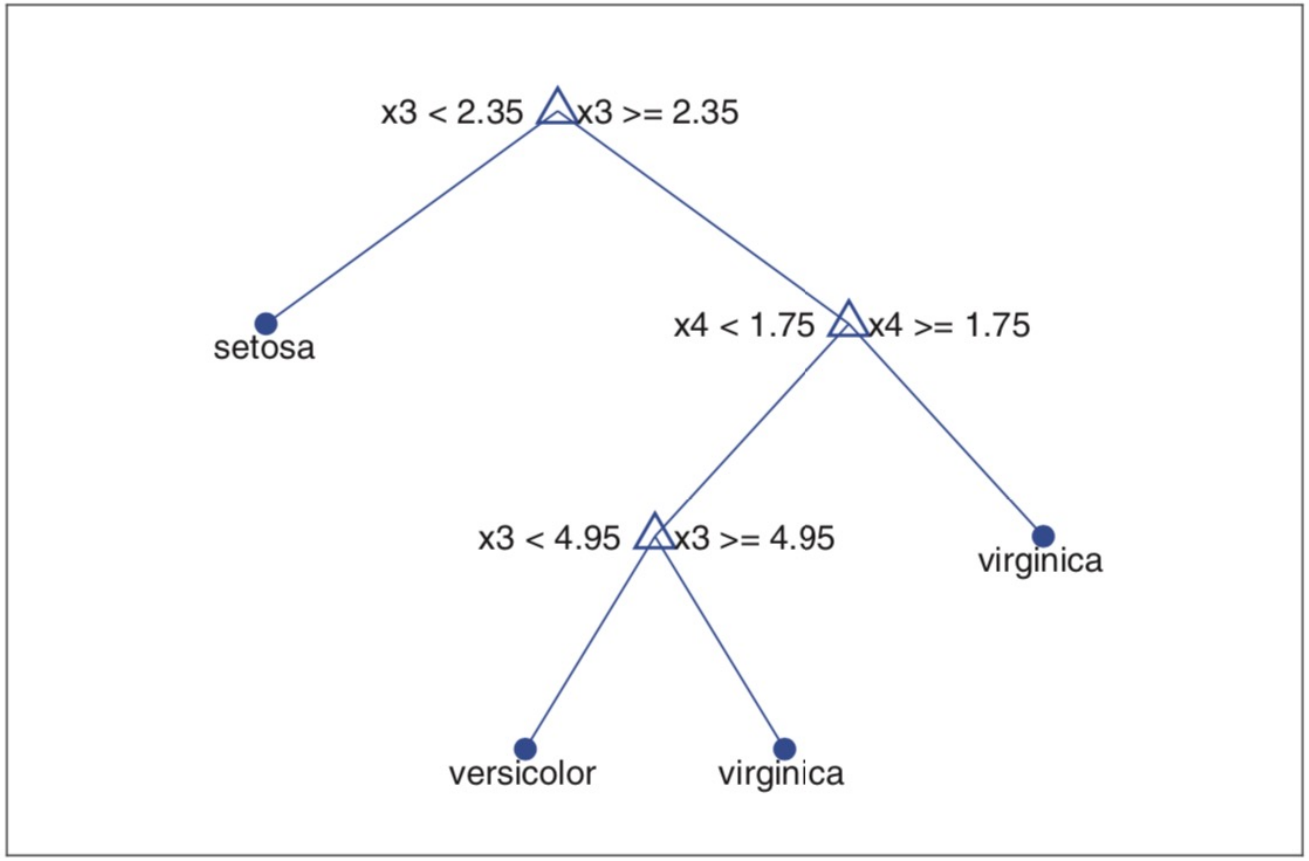



```
load fisheriris;
x1=meas(1:50,:);     % setosa
x2=meas(51:100,:);   % versicolor
x3=meas(101:150,:); % virginica

plot3(x1(:,1),x1(:,2),x1(:,4),'go'), hold on
plot3(x2(:,1),x2(:,2),x2(:,4),'mo')
plot3(x3(:,1),x3(:,2),x3(:,4),'ro')
```

- Each data point has 4 components
- In the split 1, we observe that x_3 gives the best split
- In the split 2, we observe that x_4 gives the best split
- In the split 3, we observe that x_3 gives the best split

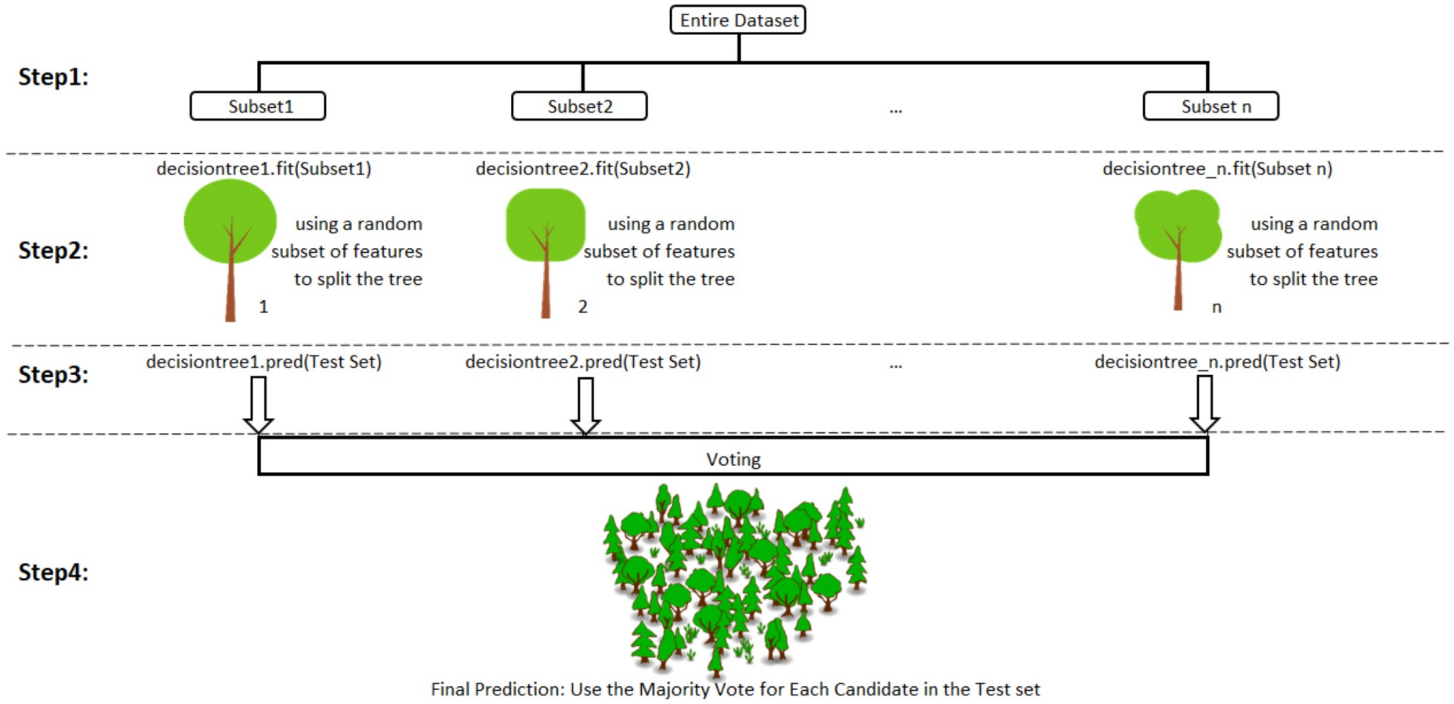




Random forest

Note that there are many other variants of random forest

- Note that, decision tree is not robust, as one can generate two different trees with two subsamples of data
- The random forest is an example of ensemble learning: using multiple models to make a prediction



Top 10 algorithms in 2008

- K-means
- Expectation-Maximization (EM) (unsupervised learning)
- Support vector machines (SVM)
- Classification and Regression tree (CART)
- K-nearest neighbors (kNN)
- Naive Bayes
- AdaBoost (random forest)
- C4.5 (ensemble learning tree)
- Apriori algorithm
- Page rank