

# Artificial Intelligence

## Lecture 13: Ensemble & Clustering

Credit: Ansaf Salleb-Aouissi, and “Artificial Intelligence: A Modern Approach”, Stuart Russell and Peter Norvig, and “The Elements of Statistical Learning”, Trevor Hastie, Robert Tibshirani, and Jerome Friedman, and “Machine Learning”, Tom Mitchell.

# Ensemble

# Majority Voting

- A randomly chosen hyperplane has an expected **error of 0.5**.
- Many random hyperplanes combined by **majority vote** will still be random.
- Suppose we have  $m$  classifiers, performing slightly better than random, that is ***error* =  $0.5 - \epsilon$** .
- Combine: make a decision based on majority vote?
- What if we combined these  $m$  **slightly-better-than-random** classifiers? Would majority vote be a good choice?

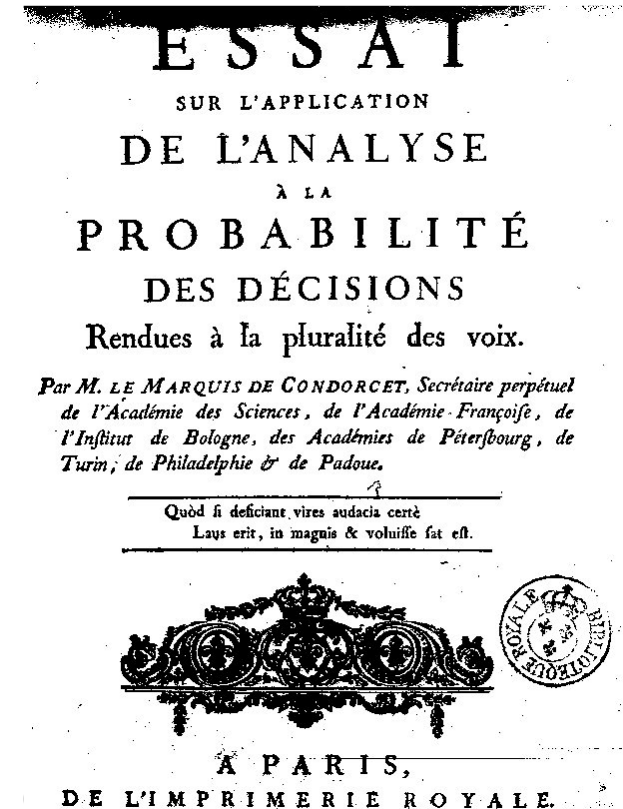
# Condorcet's Jury Theorem

Marquis de Condorcet Application of Analysis to the Probability of Majority Decisions. 1785.

## Assumptions:

1. Each individual makes the right choice with a probability  $p$ .
2. The votes are independent. 只能做到相对独立


If  $p > 0.5$ , then adding more voters increases the probability that the majority decision is correct. if  $p < 0.5$ , then adding more voters makes things worse.



# Ensemble Methods

- An **Ensemble Method** combines the predictions of many individual classifiers by majority voting.
- Such individual classifiers, called **weak learners**, are required to perform slightly better than random.

**How do we produce independent weak learners using the same training data?**

- Use a **strategy** to obtain relatively independent weak learners!
- Different methods:
  1. Boosting
  2. Bagging 
  3. Random Forests

# Boosting

- First ensemble method.
- One of the most powerful Machine Learning methods.
- Popular algorithm: AdaBoost.
- Simple algorithm.
- Weak learners can be trees, perceptrons, decision stumps, etc.
- **Idea:**  
Train the weak learners on <sup>自适应加权</sup> **weighted** training examples.

决策树推：只有一个root和2个leave的决策树

# Boosting

- The predictions from all of the  $G_m$ ,  $m \in \{1, \dots, M\}$  are combined with a weighted majority voting.
- $\alpha_m$  is the contribution of each weak learner  $G_m$ .
- Computed by the boosting algorithm to give a weighted importance to the classifiers in the sequence.
- The decision of a highly-performing classifier in the sequence should weight more than less important classifiers in the sequence.
- This is captured in:

$$G(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$$

控制分类器的重要性

# Boosting

The error rate on the training sample:

$$err := \frac{\sum_{i=1}^n 1\{y_i \neq G(x_i)\}}{n}$$

The error rate on each weak learner:

$$err_m := \frac{\sum_{i=1}^n w_i 1\{y_i \neq G_m(x_i)\}}{\sum_{i=1}^n w_i}$$

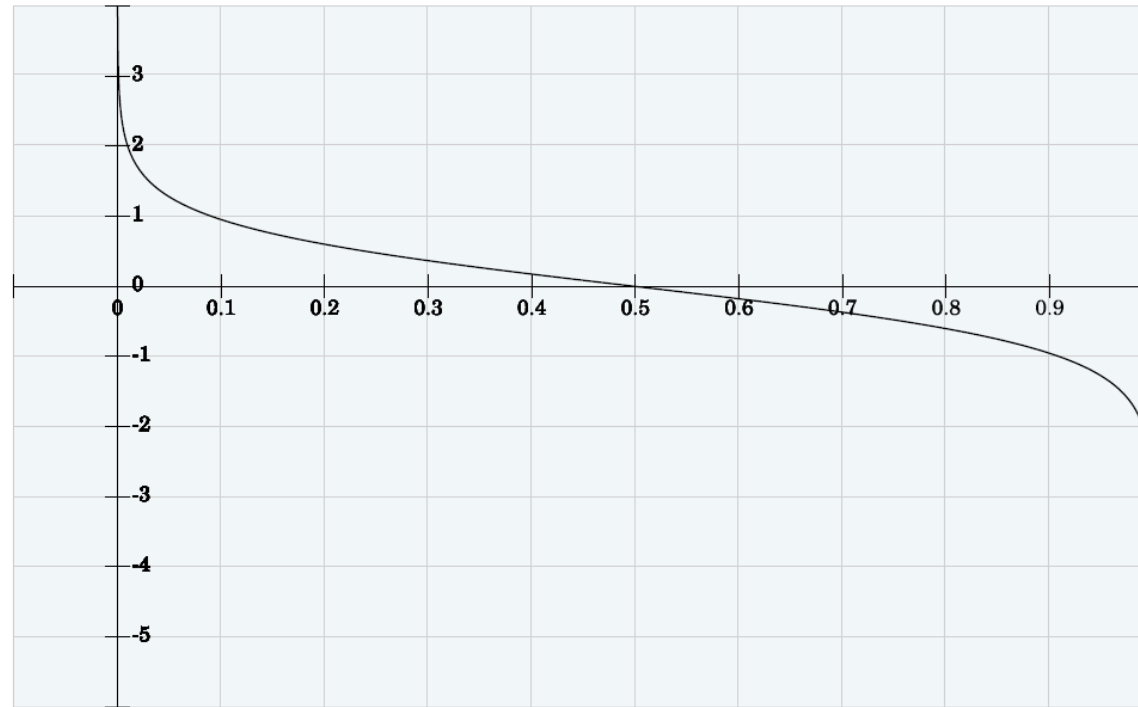
**Intuition:**

- Give large weights for hard examples.
- Give small weights for easy examples.



# Boosting

For each weak learner  $m$ , we associate an error  $err_m$ .



$$\alpha_m = \frac{1}{2} \ln\left(\frac{1-err_m}{err_m}\right)$$

# AdaBoost

1. Initialize the example weights,  $w_i = 1/n$ ,  $i = 1, \dots, n$ .
2. For  $m = 1$  to  $M$  (number of weak learners)
  - (a) Fit a classifier  $G_m(x)$  to training data using the weights  $w_i$ .

(b) Compute

$$err_m := \frac{\sum_{i=1}^n w_i \mathbf{1}\{y_i \neq G_m(x_i)\}}{\sum_{i=1}^n w_i}$$

(c) Compute

$$\alpha_m = \frac{1}{2} \log\left(\frac{1 - err_m}{err_m}\right)$$

(d) Compute

$$w_i \leftarrow w_i \cdot \exp[-\alpha_m \underbrace{y_i G_m(x_i)}_{\text{猜对} > 1, \text{猜错} < 1}] \quad \text{for } i = 1, \dots, n.$$

$$w_i = \frac{w_i'}{\sum w_i'} \quad \text{归一化}$$

3. Output

$$G(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$$

# Digression: Decision Stumps

**This is an example of very weak classifier**

A simple 2-terminal node decision tree for binary classification.

$$f(\mathbf{x}) = s(x_k > c)$$

Where  $c \in \mathbb{R}$ ,  $k \in \{1, \dots, d\}$ ,  $s \in \{-1, 1\}$ .

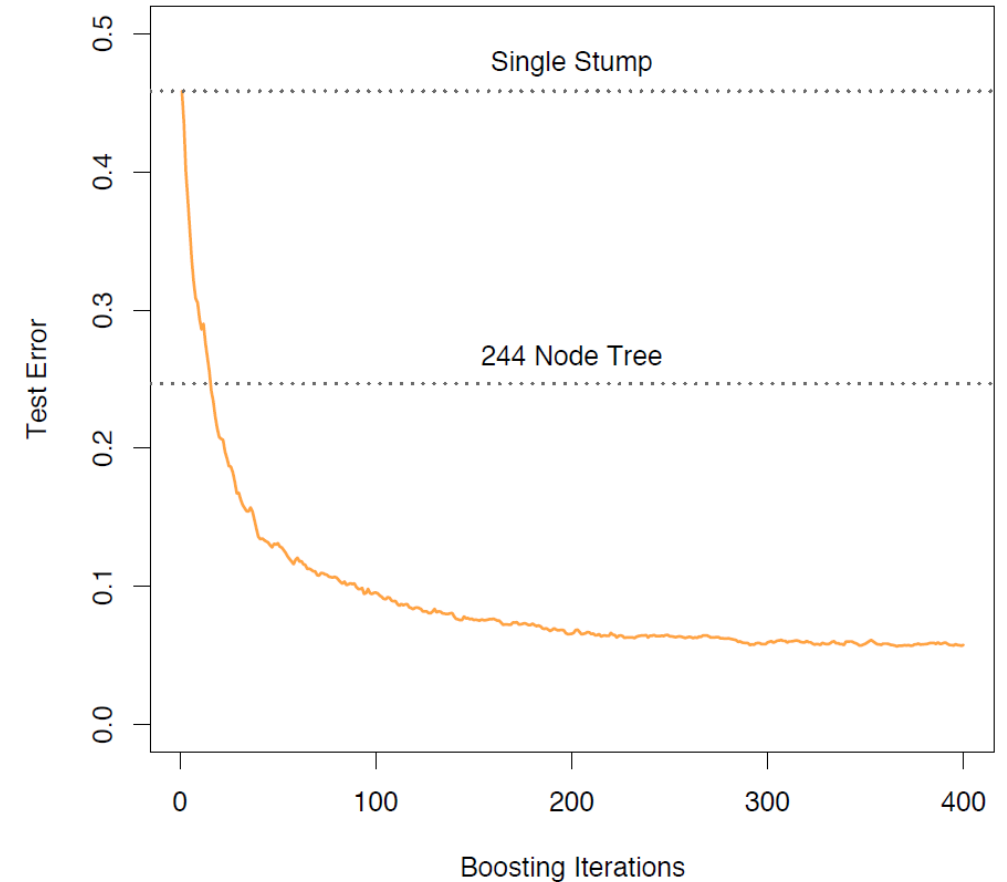
A decision stump is often trained by brute force: <sup>离散化</sup>discretize the real numbers from the smallest to the largest value in the training set, enumerate all possible classifiers, and pick the one with the lowest training error.

**Example:** A dataset with 10 features, 2,000 examples training and 10,000 testing.

# AdaBoost Performance

## Error rates:

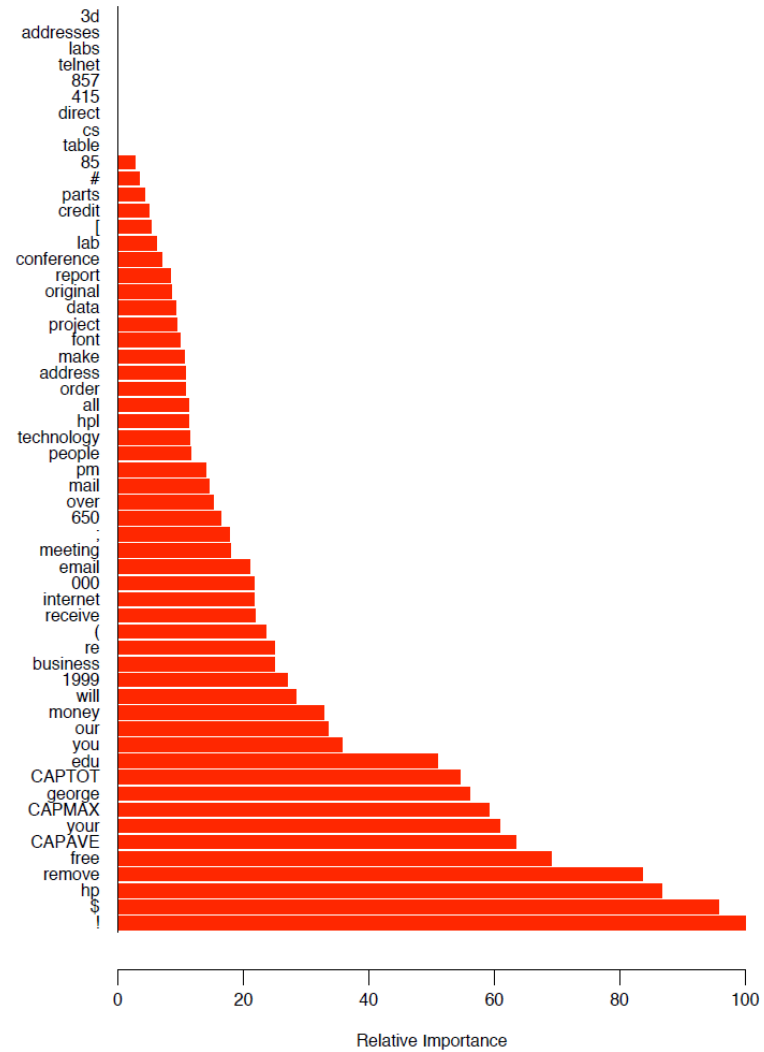
- Random: 50%.
- Stump: 45.8%.
- Large classification tree: 24.7%.
- AdaBoost with stumps: 5.8% after 400 iterations!



特征提取

AdaBoost with Decision stumps lead to a form of: **feature selection**

# AdaBoost-Decision Stumps



# Bagging & Bootstrapping

有放回采样

- Bootstrap is a re-sampling technique  $\equiv$  sampling from the empirical distribution.
- Aims to improve the quality of estimators.
- Bagging and Boosting are based on bootstrapping.
- Both use re-sampling to generate weak learners for classification.
- **Strategy:** Randomly distort data by re-sampling.
- Train weak learners on re-sampled training sets.
- Bootstrap **agg**regation  $\equiv$  Bagging.

# Bagging

## Training

For  $b = 1, \dots, B$

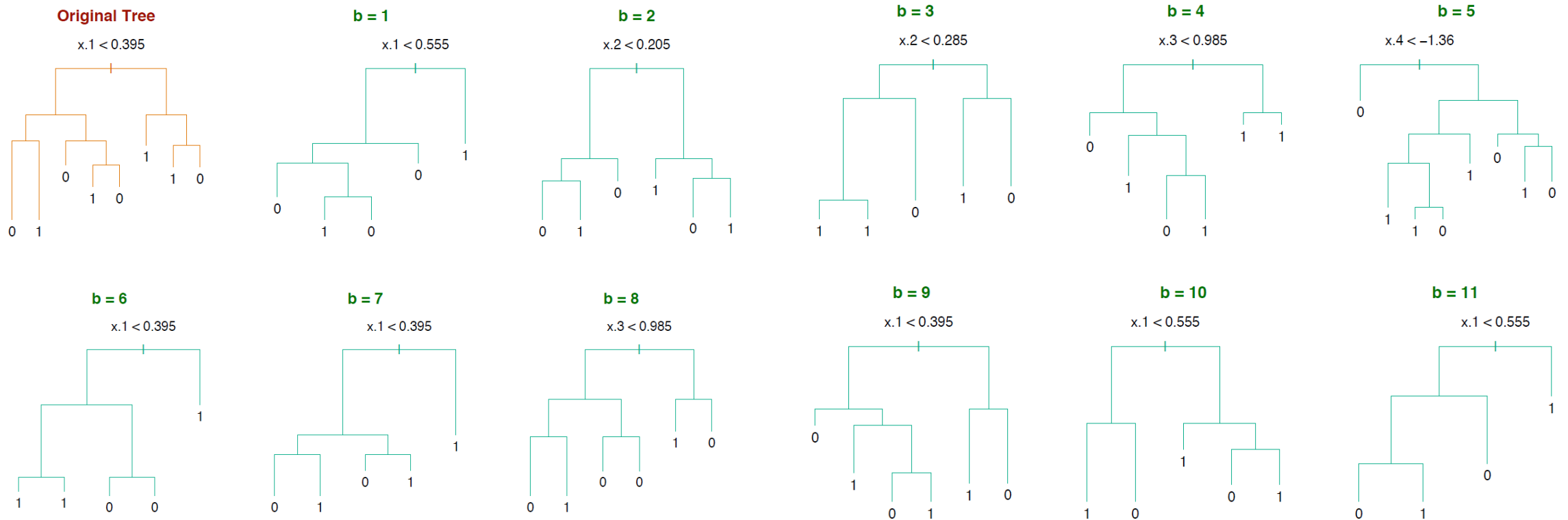
1. Draw a bootstrap sample  $B_b$  of size  $L$  from training data.
2. Train a classifier  $f_b$  on  $B_b$ .

**Classification:** Classify by majority vote among the  $B$  classifiers:

$$f_{avg} := \frac{1}{B} \sum_{b=1}^B f_b(x)$$

# Bagging

Bagging works well for trees:





# Random Forests

1. Random forests: modifies bagging with trees to reduce correlation between trees.
2. Tree training optimizes each split over all dimensions.
3. But for Random forests, **choose a different subset of dimensions at each split**. Number of dimensions chosen  $m$ .
4. Optimal split is chosen within the subset.
5. The subset is chosen at random out of all dimensions  $1, \dots, d$ .
6. Recommended  $m = \sqrt{d}$  or smaller.

# Clustering

# Unsupervised Learning

Training data: “examples”  $x$ .

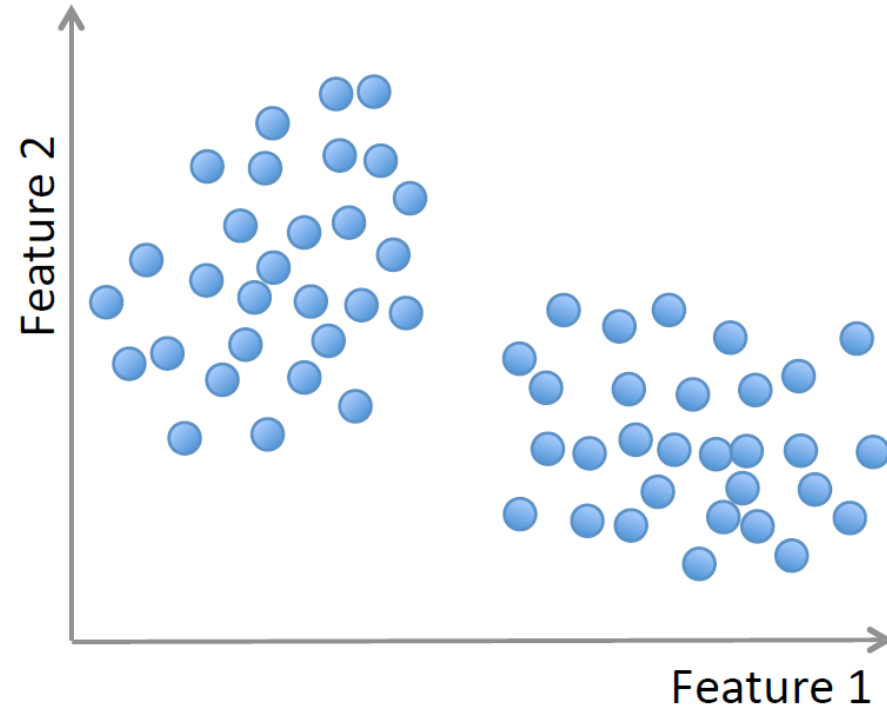
$$x_1, \dots, x_n, \quad x_i \in X \subset \mathbb{R}^d$$

- **Clustering/segmentation:**

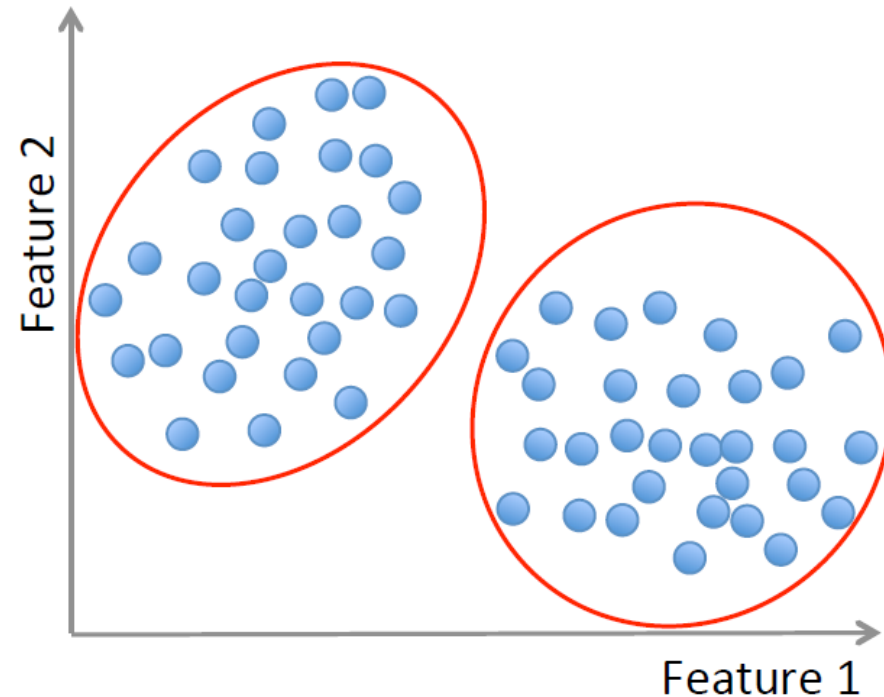
$$f: \mathbb{R}^d \rightarrow \{C_1, \dots, C_k\} \text{ (set of clusters).}$$

Example: Find clusters in the population, fruits, species.

# Unsupervised Learning



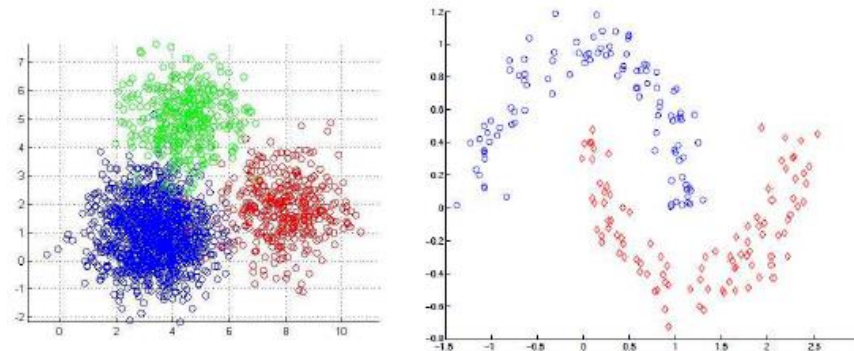
# Unsupervised Learning



**Methods:** K-means, Gaussian mixtures, hierarchical clustering, spectral clustering, etc.

# Notion of Similarity

- Choice of **similarity** measure very important for clustering
- Similarity is inversely related to **distance**
- Different ways to measure distances:
  - Euclidean distance:  $d(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|_2$
  - Manhattan distance:  $d(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{i=1}^d |\mathbf{x}_i - \tilde{\mathbf{x}}_i|$
  - Kernelized distance:  $d(\mathbf{x}, \tilde{\mathbf{x}}) = \|\phi(\mathbf{x}) - \phi(\tilde{\mathbf{x}})\|$



# Similarity is Subjective

- Similarity is often hard to define



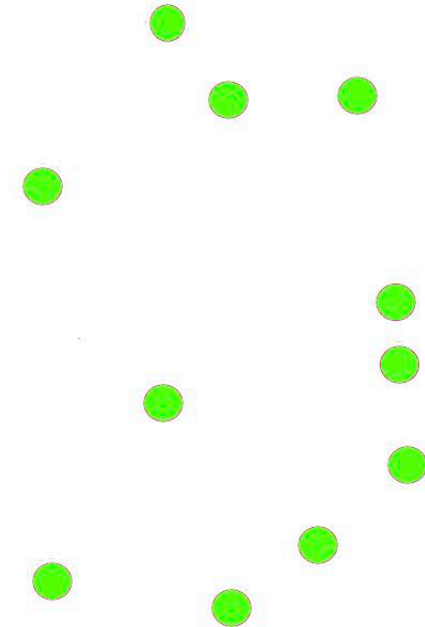
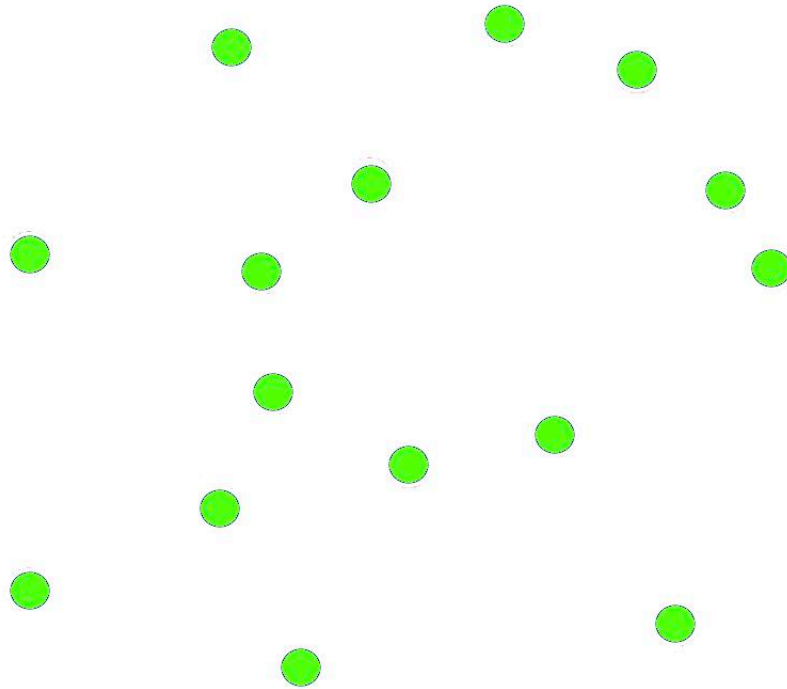
- Different similarity criteria can lead to different clusterings

# Clustering examples

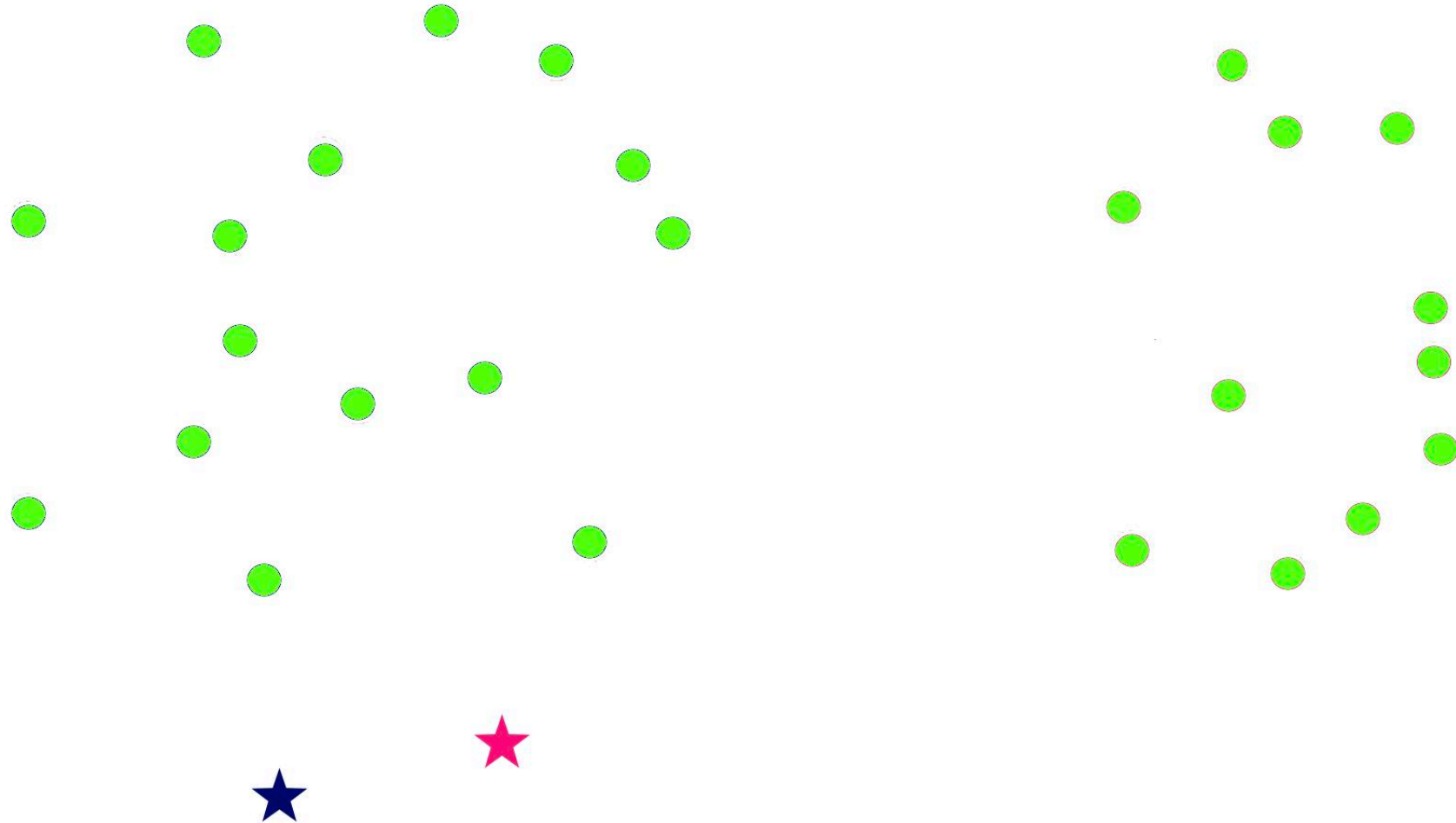
- Clustering of the population by their demographics.
- Clustering of geographic objects (mineral deposits, houses, etc.)
- Clustering of stars
- Audio signal separation. Example?
- Image segmentation. Example?



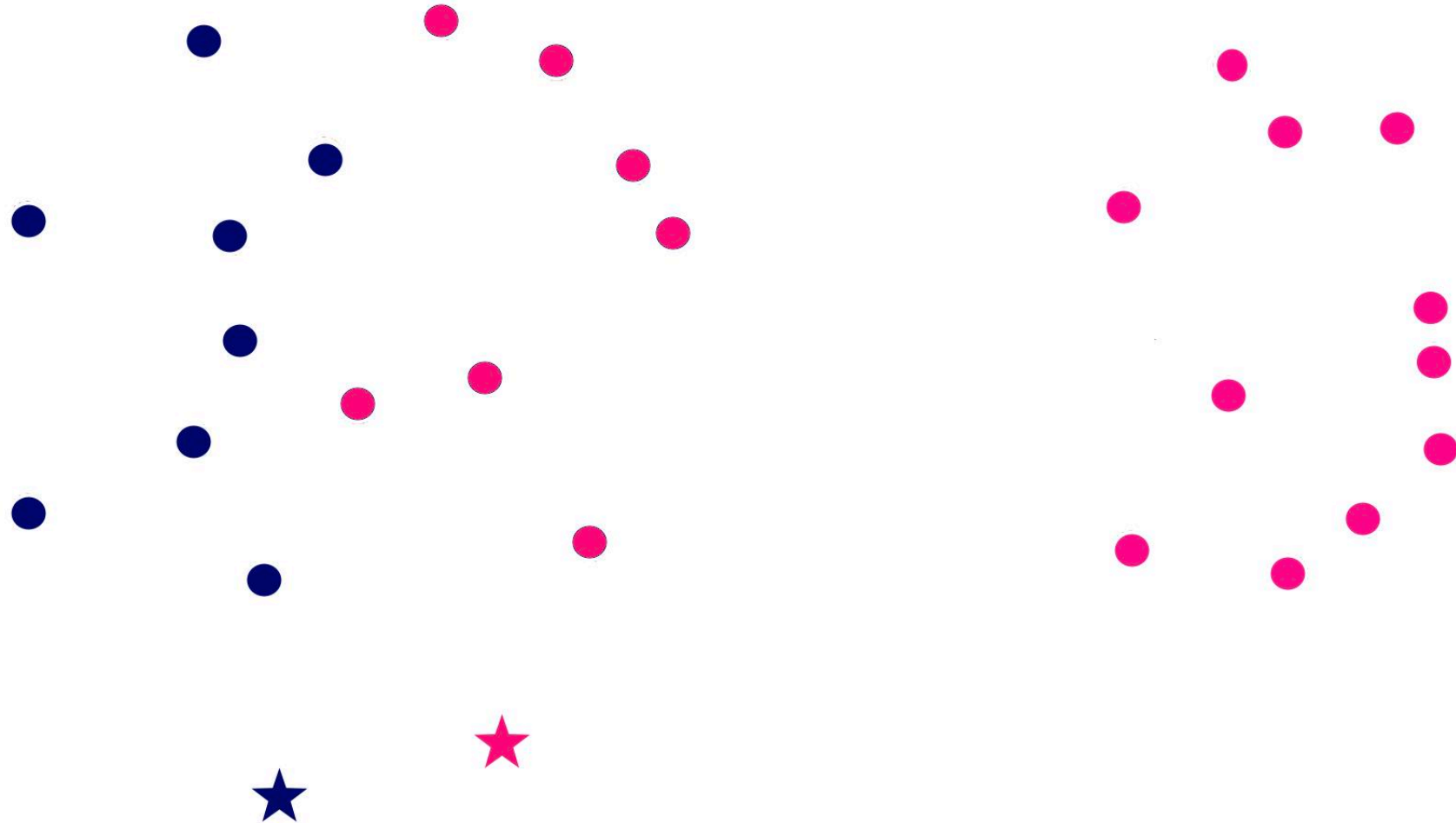
# K-Means: example



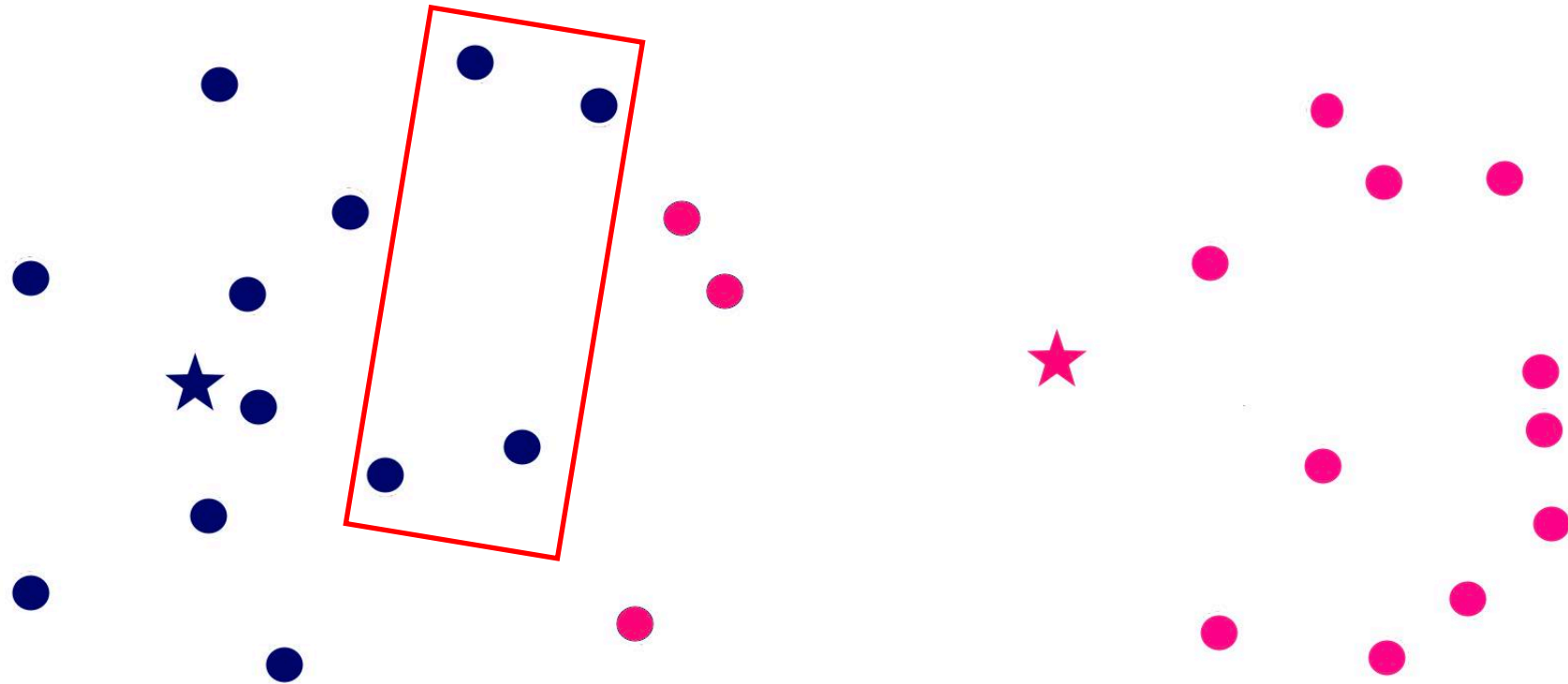
# K-Means: example



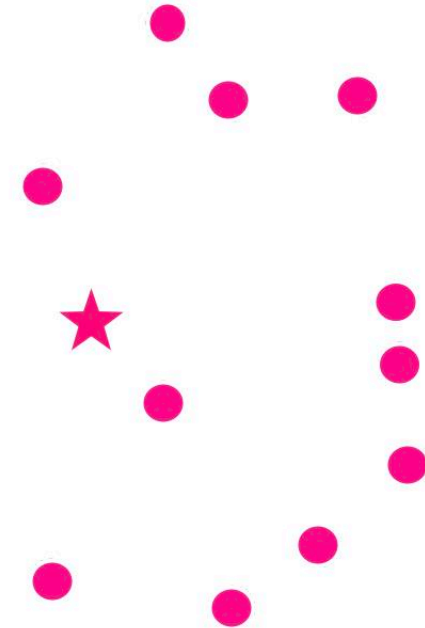
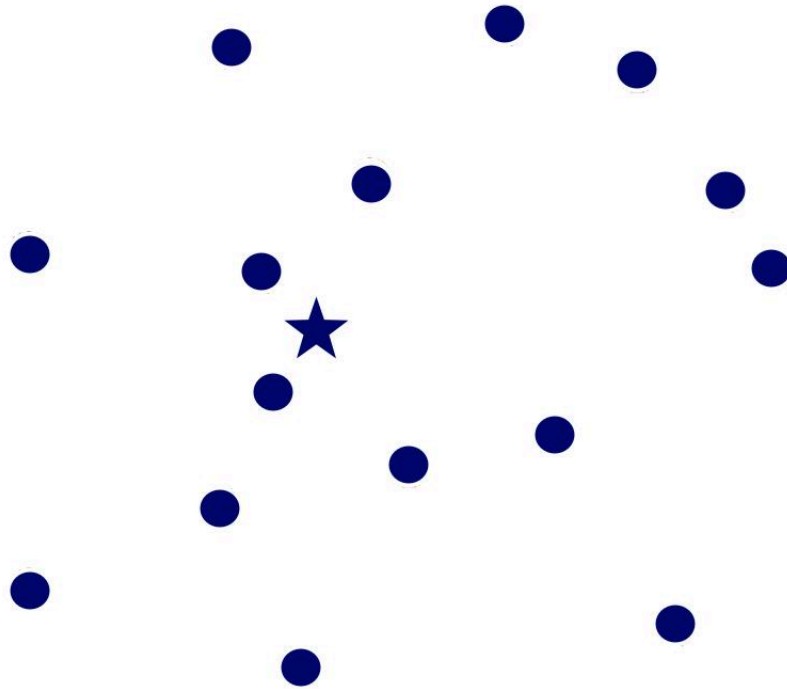
# K-Means: example



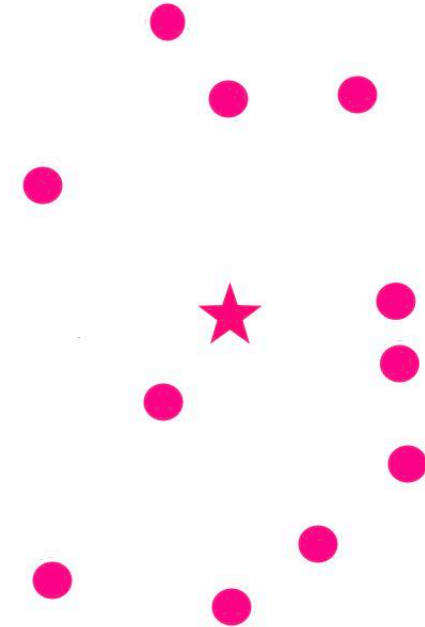
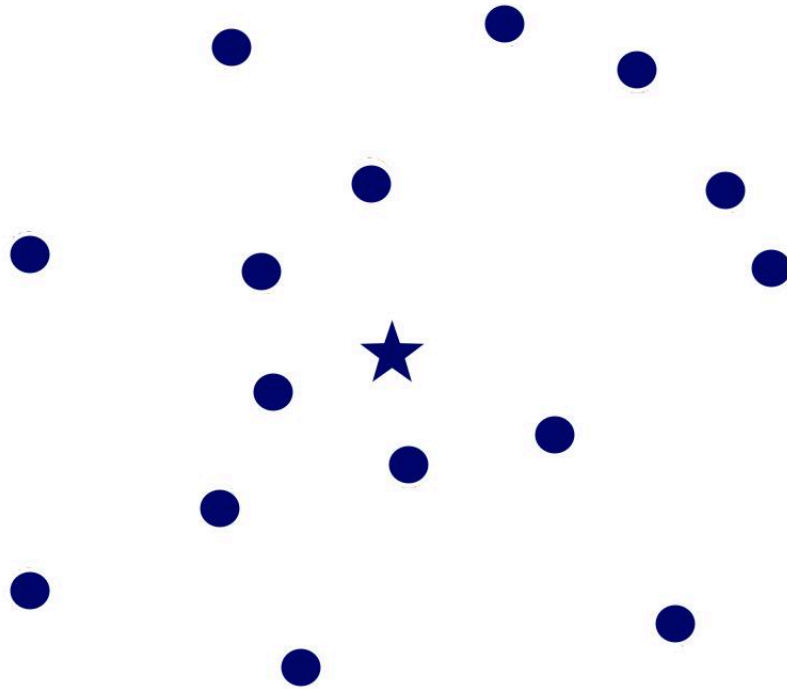
# K-Means: example



# K-Means: example



# K-Means: example



# Clustering: K-Means

## Algorithm K-Means:

Initialize randomly  $\mu_1, \dots, \mu_k$

Repeat

Assign each point  $x_i$  to the cluster with the closest  $\mu_j$ .

Calculate the new mean for each cluster as follows:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

Until convergence\*.

\*Convergence: Means no change in the clusters OR maximum number of iterations reached.

# Clustering: K-Means

- **Goal:** Assign each example  $(x_1, \dots, x_n)$  to one of the  $k$  clusters  $\{C_1, \dots, C_k\}$ .
- $\mu_j$  is the mean of all examples in the  $j^{\text{th}}$  cluster.
- **Minimize:**

类内距离

$$J = \sum_{j=1}^k \sum_{x_i \in C_j} ||x_i - \mu_j||^2$$

- Exact optimization of K-means objective is NP-hard. The K-means algorithm is a heuristic that converges to a local optimum



# K-Means: pros and cons

+ Easy to implement

BUT...

- Need to know K
- Suffer from the curse of dimensionality
- No theoretical foundation

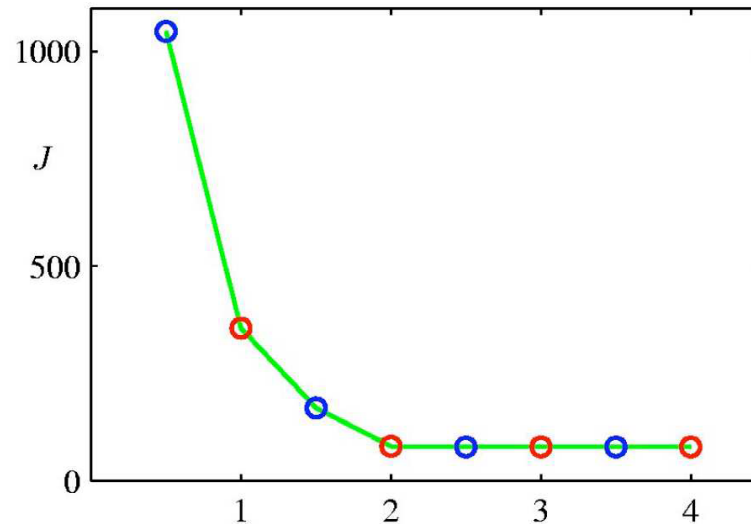
# K-Means: questions

1. How to set  $k$  to optimally cluster the data?
2. How to evaluate your model?
3. How to cluster non circular shapes?
4. How to initialize cluster centers?
5. Other limitations...

# K-Means: questions 1

## How to set $k$ to optimally cluster the data?

- One way to select  $K$  is to try different values of  $K$ , plot K-means objective versus  $K$ , and look at the “elbow-point” in the plot.



- For the above plot,  $K = 2$  is the elbow point

拐点

# K-Means: questions 1

**How to set  $k$  to optimally cluster the data?**

## **G-means** algorithm

1. Initialize  $k$  to be a small number
2. Run k-means with those cluster centers, and store the resulting centers as  $C$
3. Assign each point to its nearest cluster
4. Determine if the points in each cluster fit a Gaussian distribution (Anderson-Darling test).
5. For each cluster, if the points seem to be normally distributed, keep the cluster center. Otherwise, replace it with two cluster centers.
6. Repeat this algorithm from step 2. until no more cluster centers are created.

# K-Means: questions 2

## How to evaluate your model?

- Not trivial (as compared to counting the number of errors in classification).
- Internal evaluation: using same data. high intra-cluster similarity (documents within a cluster are similar) and low inter-cluster similarity. E.g., Davies-Bouldin index that takes into account both the distance inside the clusters and the distance between clusters. The lower the value of the index, the wider is the separation between different clusters, and the more tightly the points within each cluster are located together.
- External evaluation: use of ground truth of external data. E.g., mutual information, entropy, adjusted rand index, etc.

# K-Means: questions 3

## How to cluster non circular shapes?

There are other methods: spectral clustering, kernelized K-means, DBSCAN, BIRCH, etc. that handle other shapes.

# K-Means: questions 4

## How to initialize cluster centers?

- K-means is **extremely sensitive to cluster center initialization**
- Bad initialization can lead to
  - Poor convergence speed
  - Bad overall clustering
- Safeguarding measures
  - Choose first center as one of examples, second which is the farthest from the first, third which is the farthest from both, and so on
- **Try multiple initializations** and choose the best result

# K-Means: questions 5

## Other limitations

- Makes **hard assignments** of points to clusters
  - A point either completely belongs to a cluster or not belongs at all
  - No notion of a **soft assignment** (i.e., probability of being assigned to each cluster: say  $K = 3$  and for some point  $x_i$ 
    - $p_1 = 0.7; p_2 = 0.2; p_3 = 0.1$ ) 高斯混合模型
  - **Gaussian mixture models** allows soft-assignments
- Sensitive to **outlier examples**
  - **K-median** algorithm is a more robust alternative for data with outliers
  - Reason: Median is more robust than mean in presence of outliers



To be continued