HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

WEB MINING

LECTURE 02: MACHINE LEARNING (3/3)

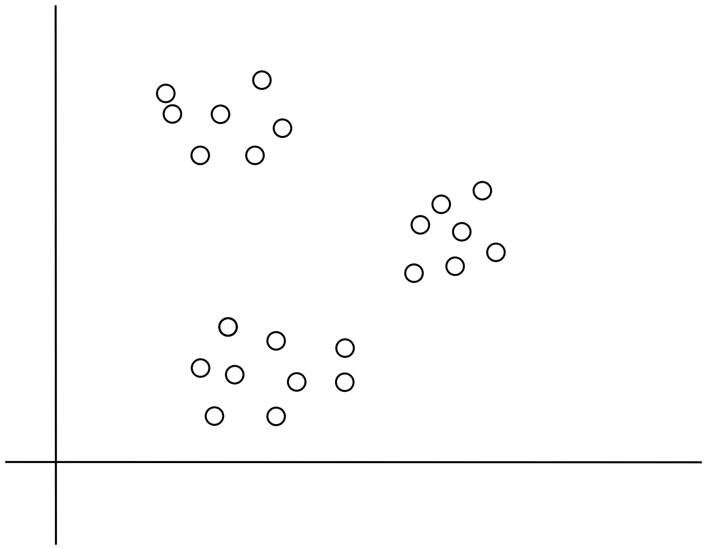
ONE LOVE. ONE FUTURE.

Content

1. Basic Concepts
2. K-means algorithm
3. Cluster representation
4. Hierarchical clustering
5. Distance function
6. Normalize data
7. Handling multiple attribute types
8. Evaluation method
9. Explore holes and data areas
10. Learning LU
11. Learning PU

1. Basic Concept

- Clustering is the process of organizing data elements into groups in which the members have similar properties. Each cluster consists of data elements that are similar and different from data elements belonging to other groups
- Application: clustering customer groups based on interests to design marketing strategies; customer clustering based on body mass index to arrange clothing production; clustering articles to synthesize news; ...



2. *k-means* Algorithm

```
Algorithm k-means(k, D)
    select k data points as centroid (center of cluster)
    repeat
        for x ∈ D do
            calculate the distance from x to each centroid;
            assign x to the nearest centroid  // a centroid represents a cluster
        endfor
        recalculate centroids based on current clusters
    until the stopping criterion is met
```

Convergence Condition:
1. The number of reassigned data points is less than a threshold
2. The number of centroids changed is less than a threshold
3. The sum of squares of error is less than a threshold

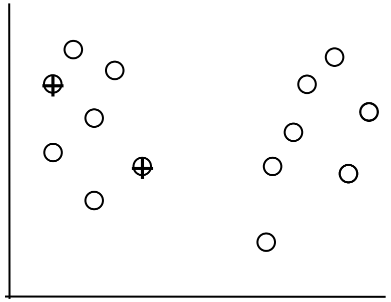
$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2$$

- *k: cluster number*
- C_j : *jth* cluster
- $\mathbf{m}_j$ is centroid of C_j (average vector of example in C_j)
- dist($\mathbf{x}$, $\mathbf{m}_j$) distance between $\mathbf{x}$ and $\mathbf{m}_i$

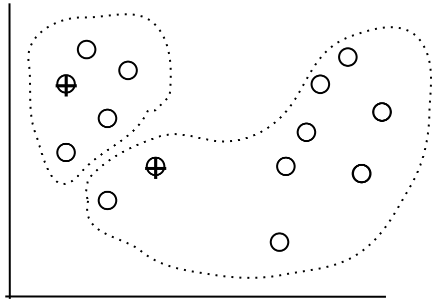
$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$$

$$dist(\mathbf{x}_i, \mathbf{m}_j) = \| \mathbf{x}_i - \mathbf{m}_j \|$$
$$= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \ldots + (x_{ir} - m_{jr})^2}$$

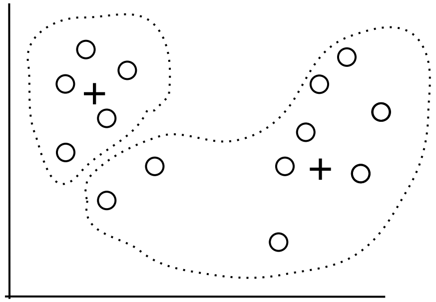
(A) Randomly
select k centroid



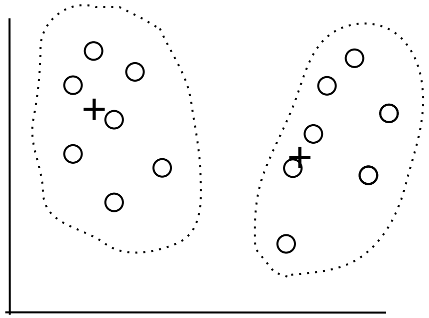
Loop 1:
(B) Cluster
assignment



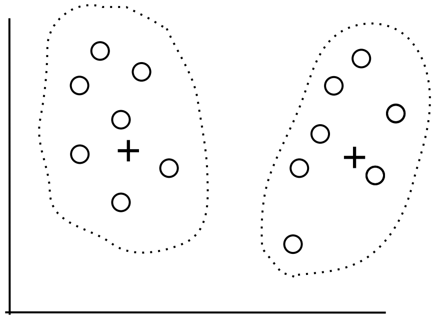
(C) Recalculate
centroid



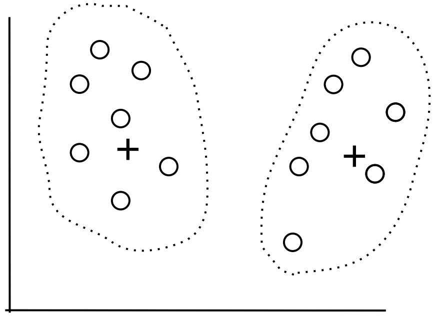
Loop 2:
(D) Cluster
assignment



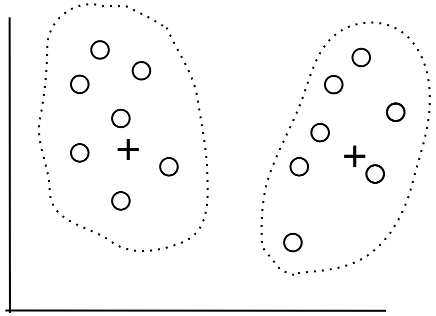
(E) Recalculate
centroid



Loop 3:
(F) Cluster
assignment



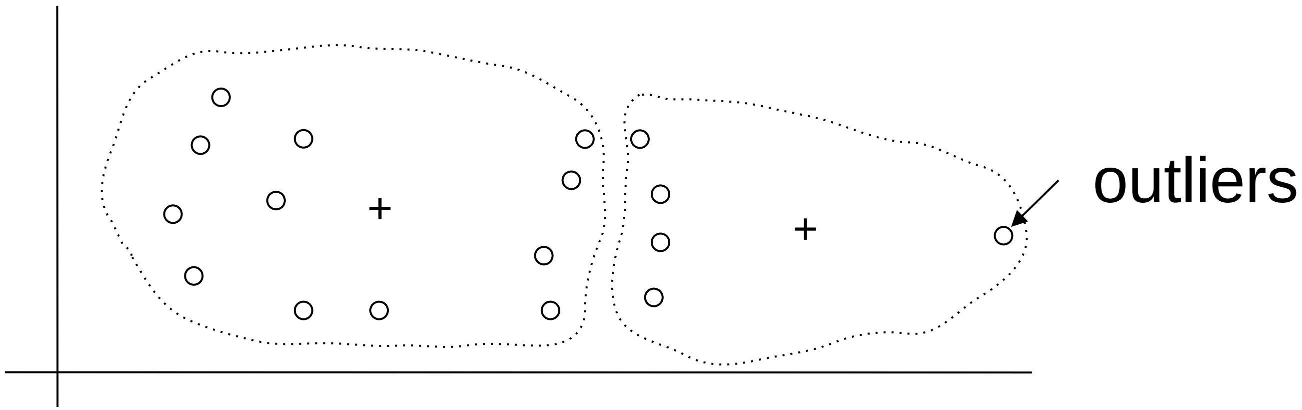
(G) Recalculate
centroid



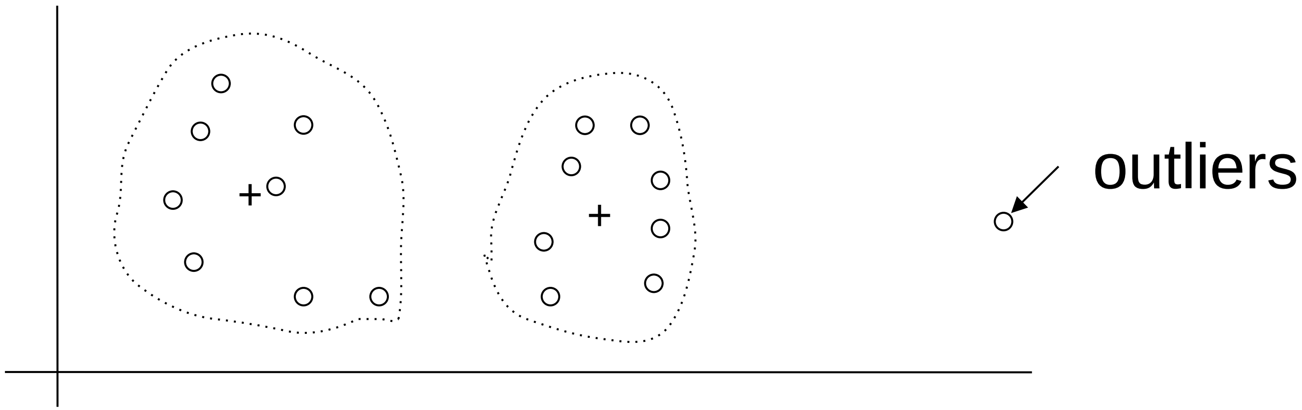
Algorithm disk-k-means(k, D)

Choose k data point as centroid $\boldsymbol{m}_{\text{j}}$, j = 1, ..., k;
repeat
    initialization $s_j \leftarrow 0$, j = 1, ..., k;// 0 is a vector with components equal to 0
    initialization $n_j \leftarrow 0$, j = 1, ..., k;// n_j is the number of points in the cluster j
    for $\boldsymbol{x} \in D$ **do**
        $j \leftarrow$ argmin dist($\boldsymbol{x}$,$\boldsymbol{m}_i$);
        Assign $\boldsymbol{x}$ to cluster j;
        $s_j \leftarrow s_j + x$;
        $n_j \leftarrow n_j + 1$;
    endfor
    $\boldsymbol{m}_{\text{j}} \leftarrow s_j / n_j$, j = 1, ..., k;
until stopping condition is met

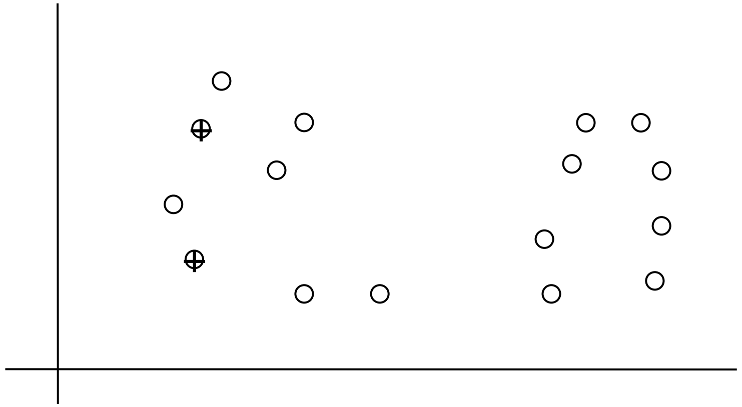
- O(tkn) where t is the number of iterations, k is the number of clusters, n is the number of examples in the training data.
- Only apply to data where mean exists, for discrete data, apply *k-modes algorithm*
- Given k values
- Sensitive to outliers (points located far from the rest of the data set)
- Sensitization to initialization (often approaching local extremes)
- Not suitable for clusters with super sphere shape



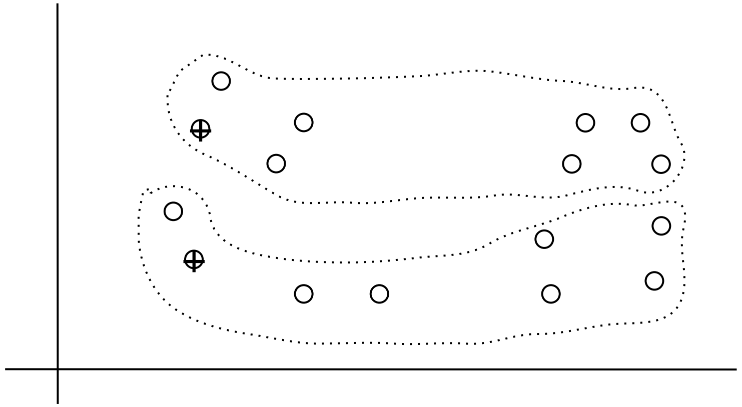
Unwanted clustering



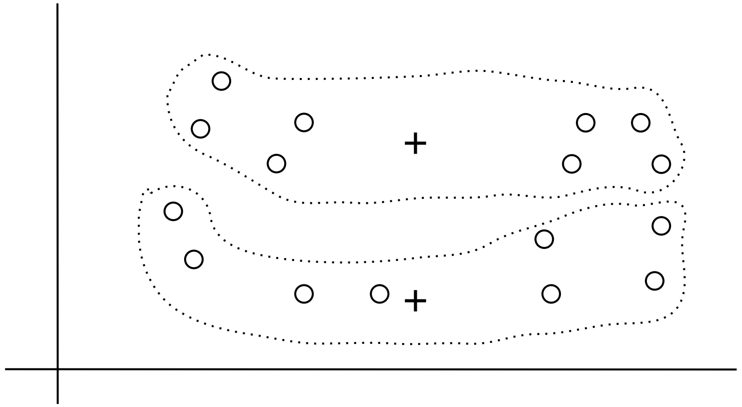
Ideal clustering



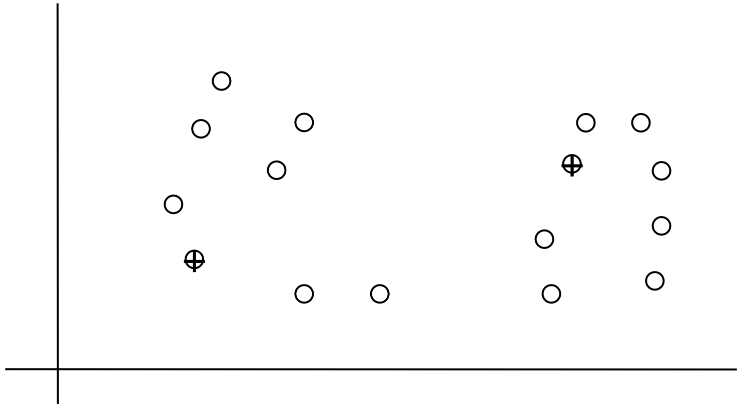
(A) Random initialization



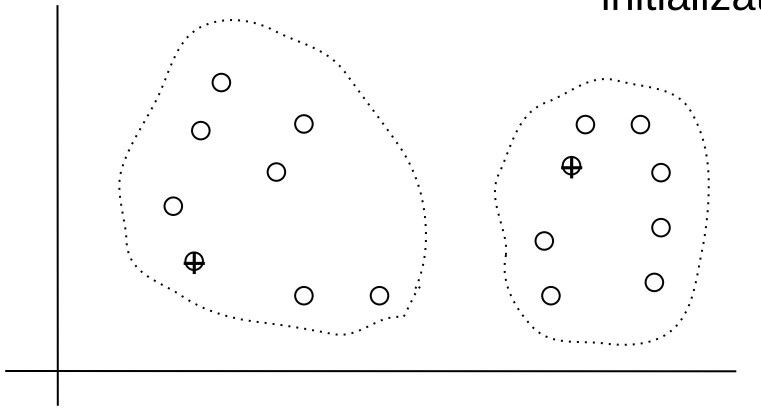
(B) Loop 1



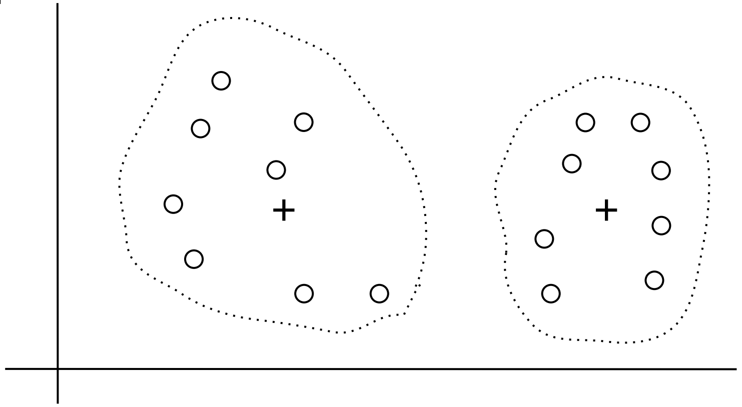
(C) Loop 2



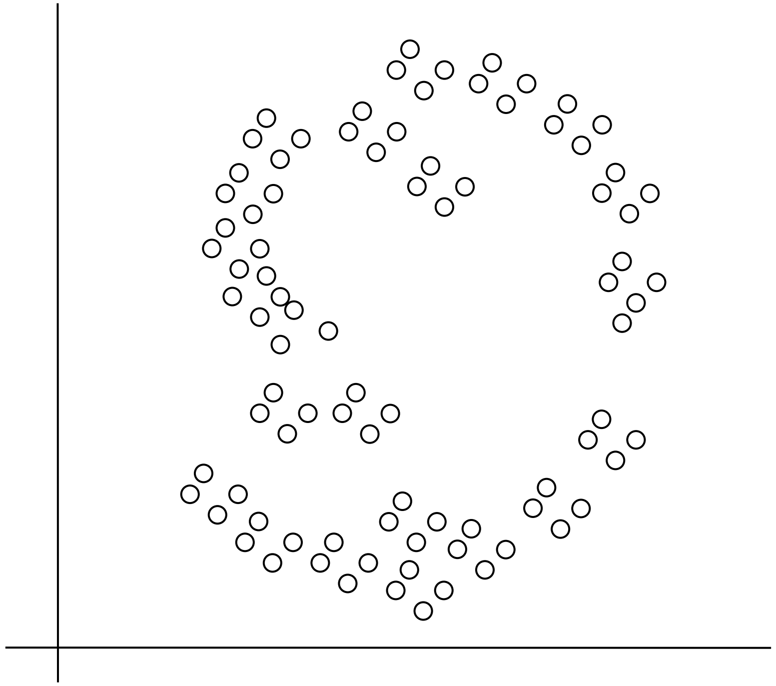
(A) Random initialization



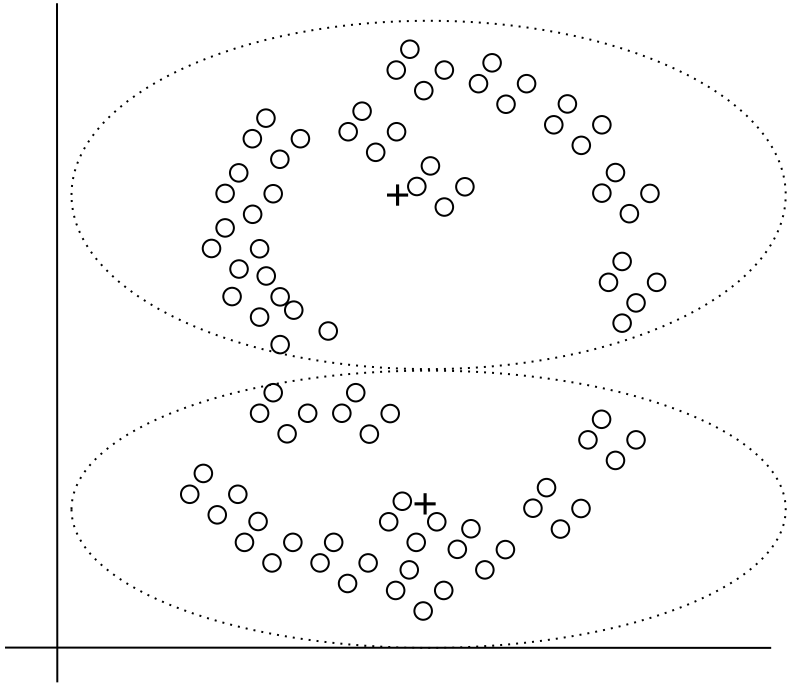
(B) Loop 1



(C) Loop 2



(A) 2 clusters of natural super spheres

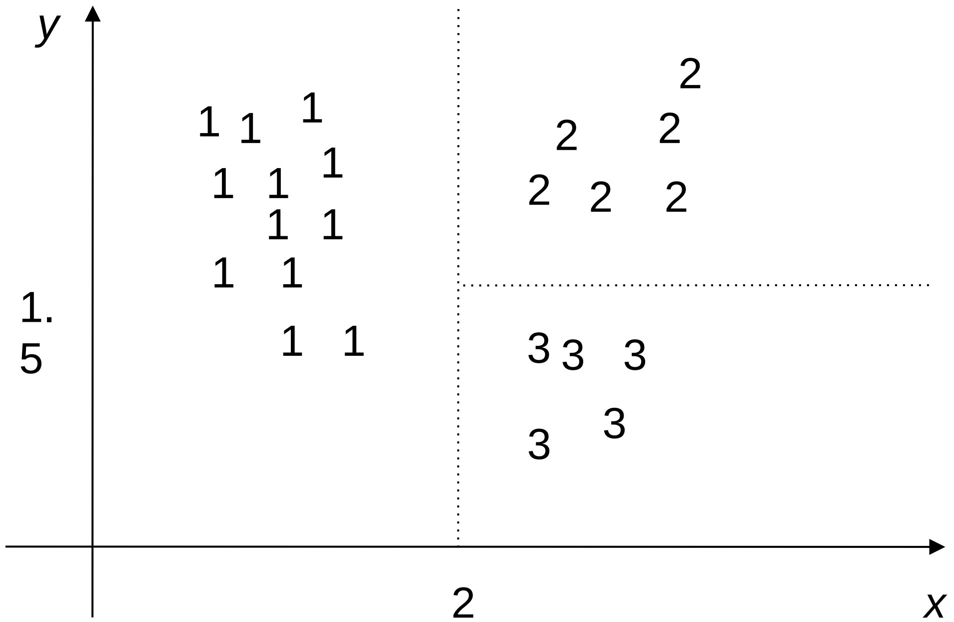


(A) Result of *k-means* (*k* = 2)

3. Cluster representation

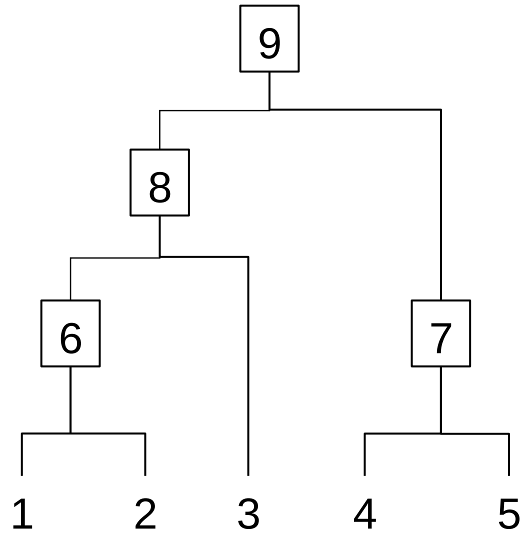
- Typical representation:
  - Centroid-based: Fits elliptical or spherical clusters
  - Based on the classification model: Assume each cluster corresponds to a class with the members of the cluster having the corresponding class label
  - Based on common values in the cluster: Fits discrete values, including text

x ≥ 2 → cluster 1
X >2, y > 1.5 → cluster 2
X > 2, y ≤ 1.5 → cluster 3



4. Hierarchical clustering

- The data is divided into a series of nested clusters in a tree structure (dendrogram).
- The leaves of the tree are data points, the root contains a single cluster, the intermediate nodes contain the sub-cluster nodes
- Bottom Up Clustering: A pair of closest clusters at each level is pooled at the next level. The process repeats until only one cluster remains
- Top-down clustering: An initial cluster containing all the data. This cluster is divided into sub-clusters. A subcluster is recursively divided until only one element remains



Algorithm Agglomerative(D)

Consider each data point in D as a cluster,,

Calculate distance pairs of $x_1, x_2, \ldots, x_n \in D$;

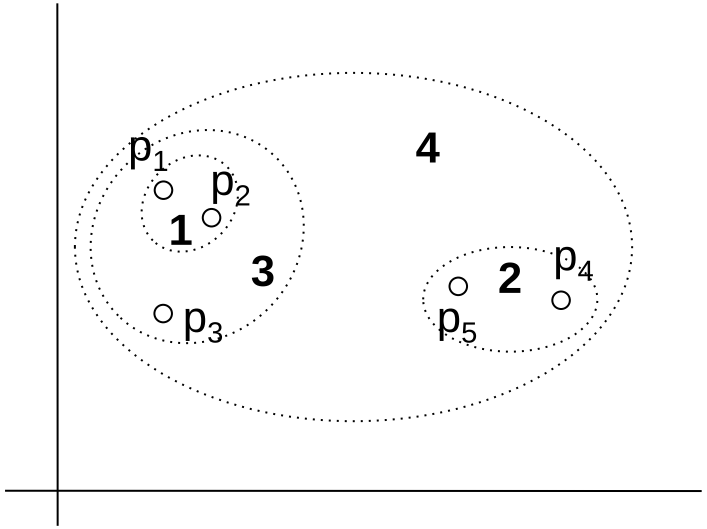
repeat

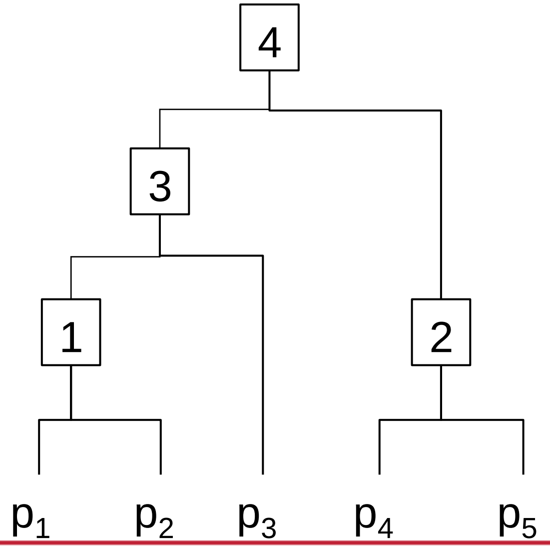
    find the two closest clusters;

    combine the two clusters into a new cluster c;

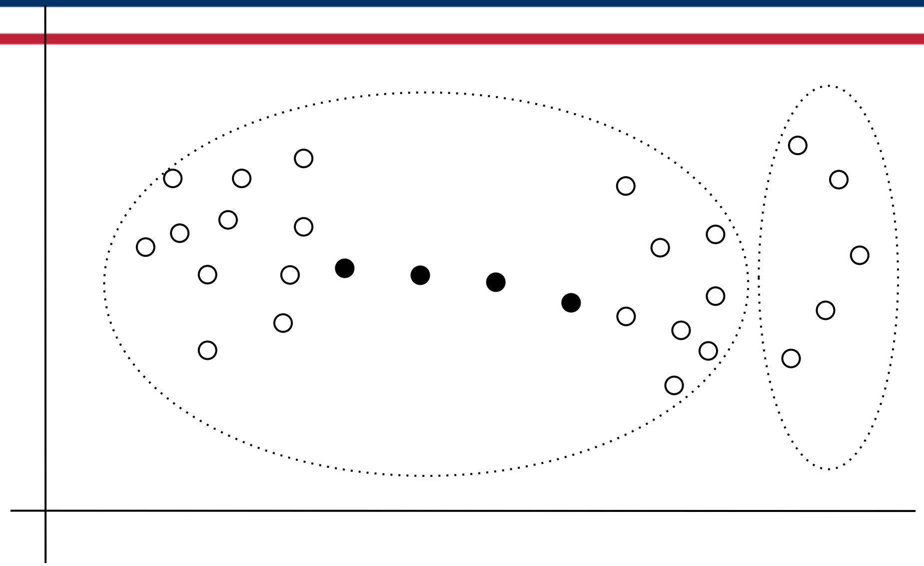
    calculate the distance from c to other clusters;

until only one cluster remains

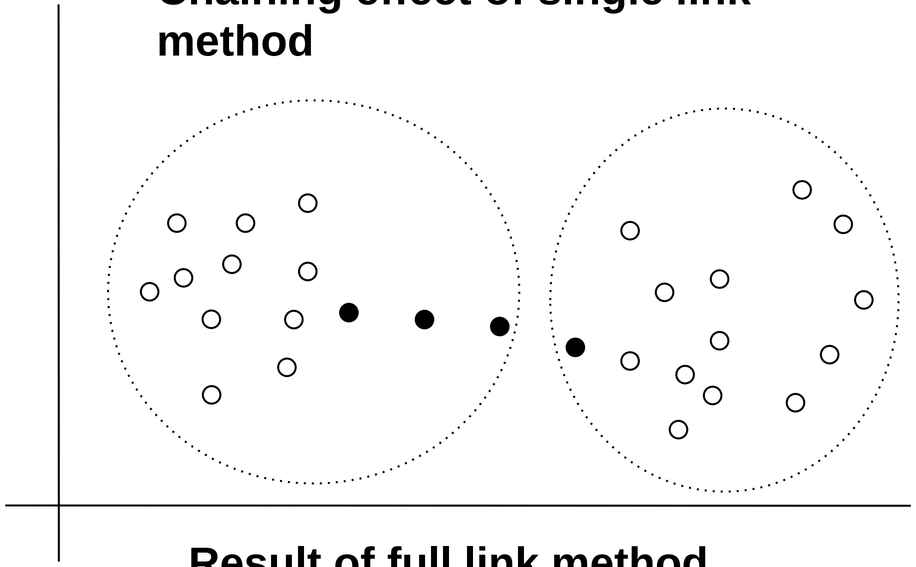




- Single link method:
  - The distance between two clusters is the shortest distance between two data points of each cluster
  - Fits clusters without ellipses
  - Can cause chaining effects due to noise in the data
  - Computational complexity: O(n^2)
- Full link method:
  - The distance between two clusters is the maximum distance between two data points of each cluster
  - No chaining effects but sensitive to outliers
  - Computational complexity O($n^2 \log n$)
- Medium link method:
  - The distance between two clusters is the average distance between two data points of each cluster
  - Computational complexity: O($n^2 \log n$)



Chaining effect of single link method



Result of full link method

- There are also other methods, for example:
  - The distance between two clusters is the distance between their two centroids
  - *Ward* method: the distance between two clusters is the increase in the sum of squared errors from those two clusters to the new cluster (if) combined
- Advantages: Hierarchical clustering allows the generation of clusters depending on the level of the tree
- Disadvantage: Hierarchical clustering has high computational and storage costs

5. Distance function/ 5.1 Continuous attribute

Minkowski($\boldsymbol{x}_i$, $\boldsymbol{x}_j$) = $(|x_{i1}-x_{j1}|^h + |x_{i2}-x_{j2}|^h + \ldots + |x_{ir}-x_{jr}|^h)^{\frac{1}{h}}$

Euclidean($\boldsymbol{x}_i$, $\boldsymbol{x}_j$) = $\sqrt{(x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \ldots + (x_{ir}-x_{jr})^2}.$

Manhattan($\boldsymbol{x}_i$, $\boldsymbol{x}_j$) = $|x_{i1}-x_{j1}| + |x_{i2}-x_{j2}| + \ldots + |x_{ir}-x_{jr}|$

Weighted_Euclidean($\boldsymbol{x}_i$, $\boldsymbol{x}_j$) = $\sqrt{w_1(x_{i1}-x_{j1})^2 + w_2(x_{i2}-x_{j2})^2 + \ldots + w_r(x_{ir}-x_{jr})^2}.$

Squared_Euclidean($\boldsymbol{x}_i$, $\boldsymbol{x}_j$) = $(x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \ldots + (x_{ir}-x_{jr})^2$

Chebychev($\boldsymbol{x}_i$, $\boldsymbol{x}_j$) = $\max(|x_{i1}-x_{j1}|, |x_{i2}-x_{j2}|, \ldots, |x_{ir}-x_{jr}|)$

5.2 Binary & Discrete Attribute

		Data point $\boldsymbol{x}_j$		
		1	0	
	1	a	b	a + b
Data point $\boldsymbol{x}_i$	0	c	d	c + d
		a +c	b + d	a + b + c + d

Ambiguous matrix of two data points containing only binary attribute

Symmetric property: Two binary values of equal importance

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{b + c}{a + b + c + d}$$

Eg:

$\boldsymbol{x}_1$ 1 1 1 0 1 0 0

$\boldsymbol{x}_2$ 0 1 1 0 0 1 0

$$\longrightarrow \text{dist}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{2 + 1}{2 + 2 + 1 + 2} = 3/7 = 0.429$$

Asymmetric attribute : Two binary values of different importance

$$\text{Jaccard}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{b + c}{a + b + c}$$

General discrete attribute :

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{r - q}{r}$$

- r: number of attributes
- q: number of matches between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

6. Normalize data

Eg:
Attribute 1 has a value in the range [0, 1], attribute 2 has a value in the range [0, 1000]
$\boldsymbol{x}_i$: (0.1, 20), $\boldsymbol{x}_j$: (0.9, 720)

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(0.9-0.1)^2 + (720-20)^2} = 700.000457,$$

After normalizing attribute 2 to the interval [0, 1]
$\boldsymbol{X}_i$: (0.1, 0.02), $\boldsymbol{x}_j$: (0.9, 0.72) → dist($\boldsymbol{x}_i$, $\boldsymbol{x}_j$) = 1.063

Linear continuous attribute:

$$rg(x_{if}) = \frac{x_{ij} - \min(f)}{\max(f) - \min(f)}$$

$$\sigma_f = \sqrt{\frac{\sum_{i=1}^{n}(x_{if} - \mu_f)^2}{n-1}}, \quad \mu_f = \frac{1}{n}\sum_{i=1}^{n} x_{if}. \qquad z(x_{if}) = \frac{x_{ij} - \mu_f}{\sigma_f}$$

- Exponential continuum attributes: logarithmic

VD: Ae^{Bt}

- Discrete attributes with no order (e.g. fruits): Can be converted to binary
- Ordered discrete attribute (e.g. age): similarly normalize linear continuous attribute

7. Handling multiple type attributes

- Data contains many types of properties: symmetric binary, asymmetric binary, linear continuum, nonlinear continuum, discrete, ordered discrete
- Convert all to the most common property type (e.g. linear continuous)
- Calculate distance on each attribute and sum it up

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{f=1}^{r} \delta_{ij}^f d_{ij}^f}{\sum_{f=1}^{r} \delta_{ij}^f}.$$

r is the number of attributes in data, d_{ij}^f is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in term of f, $\delta_{ij}^f = 1$ if the attribute f exists in both $\mathbf{x}_i$ và $\mathbf{x}_j$ and $\delta_{ij}^f = 0$ otherwise

8. Evaluation method

- User-Based: Based on a panel of experts. The final rating is the average of the group. Fits some data type (text)
- Based on classification: Use classified data. Each class corresponds to a cluster. Using categorical measures

$$entropy(D_i) = -\sum_{j=1}^{k} \Pr_i(c_j)\log_2 \Pr_i(c_j),$$

$$entropy_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times entropy(D_i)$$

$$purity(D_i) = \max_j(\Pr_i(c_j))$$

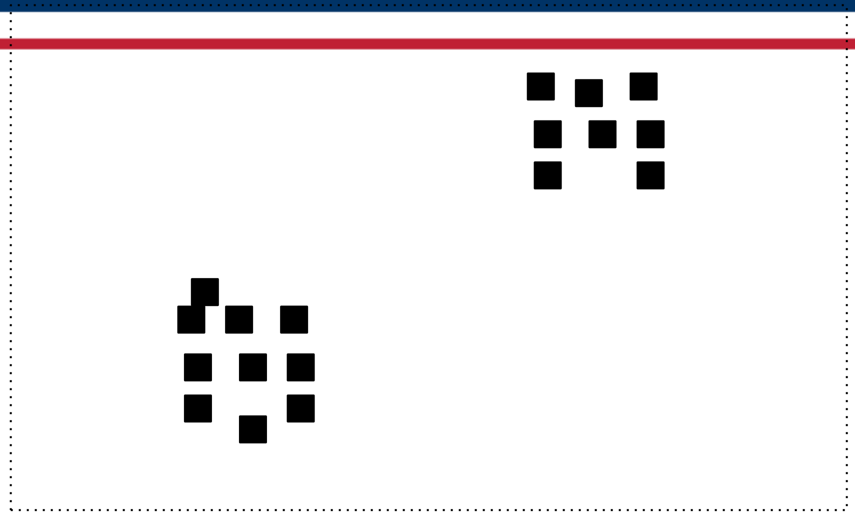
$$purity_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times purity(D_i).$$

- Compression metric: Shows the concentration of data points in a cluster around the centroid (e.g. sum squared error)
- Isolation metric: Expresses the degree of separation of clusters through the distance between centroids
- Indirect evaluation: Clustering is used as an intermediate task → evaluate the clustering technique through evaluation of the end task. E.g. user clustering is applied in recommender system (product)

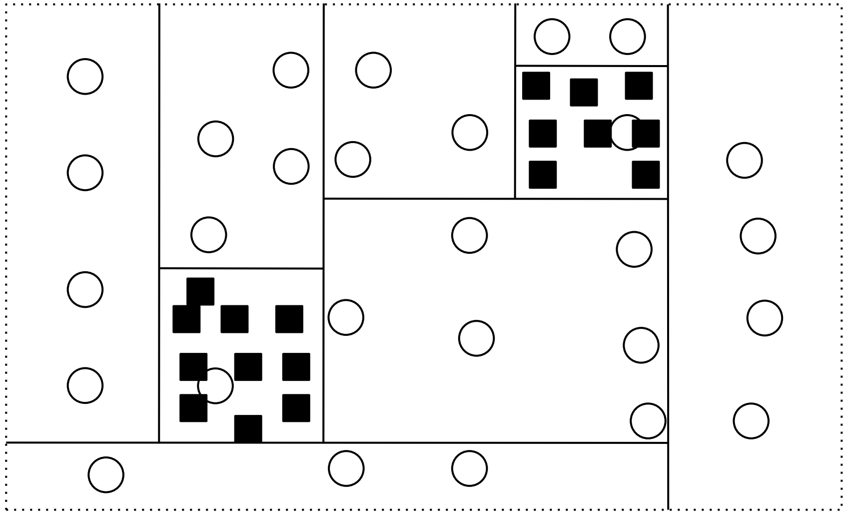
9. Exploring holes and data areas

- The data has areas of concentration of data points and regions that contain no or little data (holes).
- The discovery of data holes is important in a number of applications
- Classification problem:

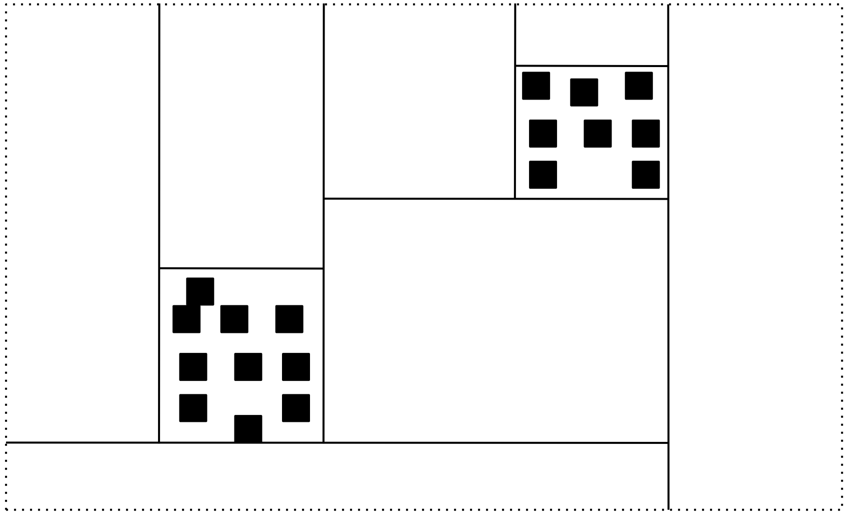
1. Assume already existing data points have label Y. Add new data points randomly and label N
2. Use a classification technique (e.g. decision tree) to classify the data
3. Obtain a classification model with interest label N



(A) Original data space



(B) Partitioning with additional data points



(C) Partition on original data

10. Learning LU

- Supervised learning gives high accuracy but requires a lot of labeled data. Data labeling is manual work, requiring a lot of time and effort. In applications such as web document classification, the data labels are constantly changing.
- LU learning (labeled and unlabeled examples) builds a classifier on a small amount of labeled data and uses a large amount of unlabeled data to improve the classifier
- For example, a text classifier uses 'assignment' as a feature to classify texts on educational topics. Based on unlabeled data, it is possible to discover that 'assignment' often co-occurs with 'lecture', thereby adding 'lecture' to characterize the classifier

10.1 EM algorithm

- EM (Expectation Maximization) is an iterative algorithm to maximize the probability estimate for missing data. The expectation step fills in the missing data based on the estimate of the current parameter. The maximization step re-estimates the parameter with the goal of maximizing the probability.
- EM+ classification model:
- 1) the data labeled L is used to build the classifier f
- 2) Use f to classify data that are not labeled U
- 3) Re-update f based on L and U; back 2), repeat until convergence

Algorithm EM(L, U)

Learning classifier NB f from dataset labeled L

repeat

    // Step E

    for each d_i in U **do**

        Use f to calculate $Pr(c_j|d_i)$

    endfor

    // Step M

    learns the classifier f from L and U (calculates $Pr(c_j)$ and $Pr(w_t|c_j)$)

until the classifier parameters stabilize

Given D_o include labeled examples, D_u include unlabeled examples
The log likelihood function has the form:

$$\log \Pr(D_o; \Theta) = \log \sum_{D_u} \Pr(D_o, D_u; \Theta)$$

Instead of maximizing log likelihood, we maximize the expectation of full log likelihood:

$$\sum_{D_u} E(D_u \mid D_o; \Theta^{T-1}) \log \Pr(D_o, D_u; \Theta)$$

Conditional probability of a document knowing its class:

$$\Pr(d_i \mid c_j; \Theta) = \Pr(|d_i|)\, |d_i|! \prod_{t=1}^{|V|} \frac{\Pr(w_t \mid c_j; \Theta)^{N_{ti}}}{N_{ti}!}$$

Assuming the texts are generated independently, the likelihood function can be written as:

$$\prod_{i=1}^{|D|} \Pr(d_i \mid c_{(i)}; \Theta) \Pr(c_{(i)}; \Theta) = \prod_{i=1}^{|D|} \Pr(|d_i|)\, |d_i|! \prod_{t=1}^{|V|} \frac{\Pr(w_t \mid c_{(i)}; \Theta)^{N_{ti}}}{N_{ti}!} \Pr(c_{(i)}; \Theta)$$

Form log likelihood function:

$$\sum_{i=1}^{|D|}\sum_{t=1}^{|V|} N_{ti} \log \Pr(w_t \mid c_{(i)};\Theta) + \sum_{i=1}^{|D|} \log \Pr(c_{(i)};\Theta) + \phi$$

Use indicator h_{ki}, $h_{ki} = 1$ when document i has label k, form log likelihood:

$$\sum_{i=1}^{|D|}\sum_{t=1}^{|V|}\sum_{k=1}^{|C|} h_{ik} N_{ti} \log \Pr(w_t \mid c_k;\Theta) + \sum_{i=1}^{|D|}\sum_{k=1}^{|C|} h_{ik} \log \Pr(c_k;\Theta) + \phi$$

Expectations of full log likelihood

$$\sum_{i=1}^{|D|}\sum_{t=1}^{|V|}\sum_{k=1}^{|C|} \Pr(c_k \mid d_i;\Theta^{T-1}) N_{ti} \log \Pr(w_t \mid c_k;\Theta)$$
$$+\sum_{i=1}^{|D|}\sum_{k=1}^{|C|} \Pr(c_k \mid d_i;\Theta^{T-1}) \log \Pr(c_k;\Theta) + \phi$$

Lagrangian:

$$\sum_{i=1}^{|D|}\sum_{t=1}^{|V|}\sum_{k=1}^{|C|}\Pr(c_k \mid d_i;\Theta^{T-1})N_{ti}\log\Pr(w_t \mid c_k;\Theta)$$
$$+\sum_{i=1}^{|D|}\sum_{k=1}^{|C|}\Pr(c_k \mid d_i;\Theta^{T-1})\log\Pr(c_k;\Theta)$$
$$+\lambda\left(1-\sum_{k=1}^{|C|}\Pr(c_k;\Theta)\right)+\sum_{t=1}^{|V|}\sum_{k=1}^{|C|}\lambda_{tk}\left(1-\sum_{t=1}^{|V|}\Pr(w_t \mid c_k;\Theta)\right)+\phi$$

Taking the derivative with respect to λ, we have

$$\sum_{k=1}^{|C|}\Pr(c_k;\Theta)=1$$

Taking the derivative with respect to $\Pr(c_k;\Theta)$:

$$\sum_{i=1}^{|D|}\Pr(c_k \mid d_i;\Theta^{T-1})=\lambda\Pr(c_k;\Theta)$$ với $k = 1, 2, \ldots, |C|$

Sum by k, we have:

$$\lambda=\sum_{i=1}^{|D|}\sum_{k=1}^{|C|}\Pr(c_k \mid d_i;\Theta^{T-1})=|D|$$

Update Pr(c_j| Θ):

$$\Pr(c_j;\Theta^T) = \frac{\sum_{i=1}^{|D|} \Pr(c_j \mid d_i;\Theta^{T-1})}{|D|}$$

Update Pr(w_t| c_j, Θ):

$$\Pr(w_t \mid c_j;\Theta^T) = \frac{\sum_{i=1}^{|D|} N_{ti} \Pr(c_j \mid d_i;\Theta^{T-1})}{\sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j \mid d_i;\Theta^{T-1})}$$

10.2 Co-training

- Suppose the set of attributes can be divided into two sets X_1 and X_2 to build two classifiers f_1 and f_2
- Assumption 1: The classifier built on the entire attribute f has the same classification result as f_1 and f_2
- Assumption 2: X_1 and X_2 are independent of each other for class labels

Algorithm co-training(L, U)

repeat

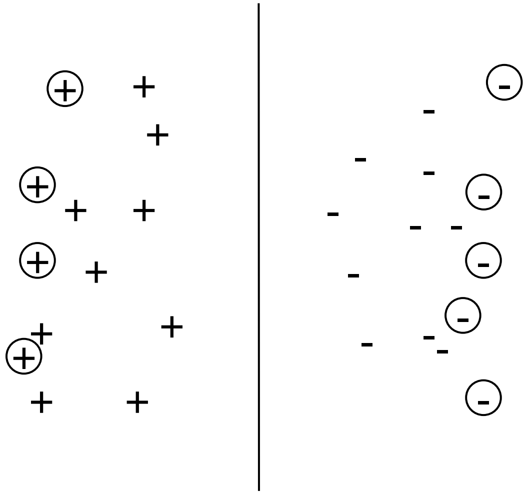
Build classifier f_1 using L based on attribute set X_1

Build classifier f_2 using L based on attribute set X_2

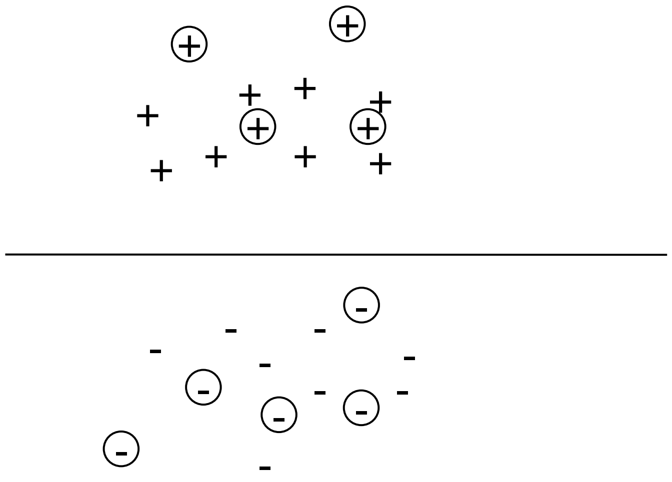
Use f_1 to classify the examples in U, for each class c_i, choose n_i examples f_1 is most confident in and add L.

Use f_2 to classify the examples in U, for each class c_i, choose n_i examples f_2 is most confident in and add L.

until U is empty (or after a certain number of loops)



(A) Data classified by f_1 over U



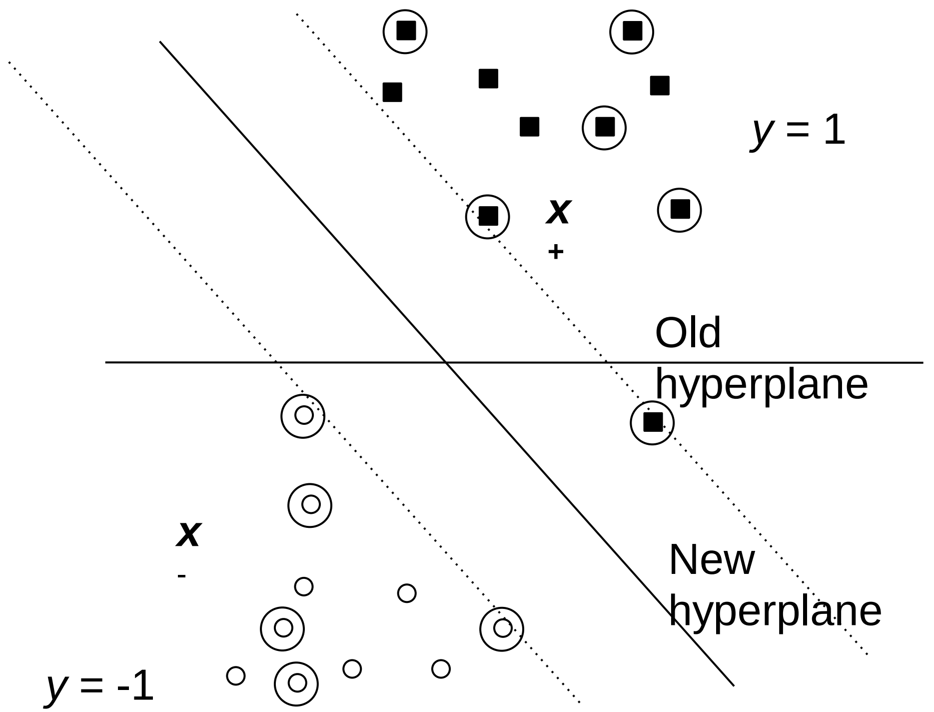
(B) Additional data by f_1 for f_2

10.3 Self - training

- The classifier *f* is built on the set L
- The classifier *f* is used to classify examples in the set U
- The examples with the highest confidence are added to the L
- The process repeats until the end (e.g. U is empty)

10.4 Inference on SVM

- Hyperplane selection for margin maximization based on unlabeled examples



10.5 Graph-based methods

- From *L* and *U*, construct a graph of vertices as examples, and weighted edges as similarity between examples.
  - From a vertex, choose k nearest neighbors
  - Select similarity above a minimum threshold
  - Use a fully connected graph with exponential similarity
- Label the examples in *U* based on the labeled examples in *L* such that similar vertices have the same label

- Mincut: Labeled vertices belonging to L are assigned the value {0,1} depending on the positive or negative class; weighted edges w_{ij}; find a way to label unlabelled vertices of U such that $\Sigma_{(i,j) \in E} w_{ij}|v_i - v_j|$ smallest
- Find the division of the vertex set V into two non-intersecting sets V_+ và V_- such that the sum of the weights of the edges joining the two sets is minimal. V_+ contains positively labeled vertices of L and vertices of U; V_- contains negative labeled vertices in L and vertices in U
- Maximum flow algorithm with $O(|V|^3)$ omputational complexity

- Gaussian field: Minimizing $\Sigma_{(i,j) \in E}\, w_{ij}|v_i - v_j|^2$ with values in [0, 1]
- Spectral graph: Minimize cut (V_+, V_-) / $(|V_+||V_-|)$ o balance the number of vertices of two sets because mincuts tend to choose two unbalanced sets

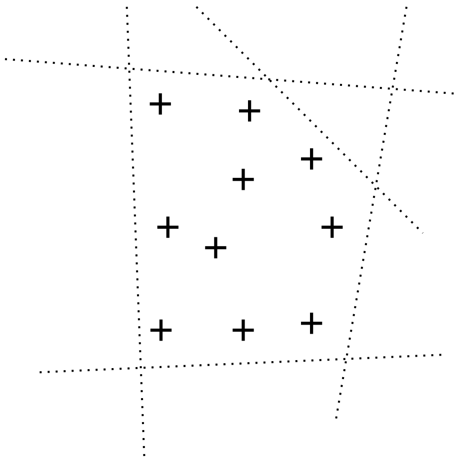
11. Learning PU

- In some applications the user is only interested in one layer of text (positive) and not in the other text (negative).
- Given the set P consisting of positive documents and the set U consisting of unlabeled documents (including positive and negative texts), it is necessary to build a classifier that allows identifying positive documents.

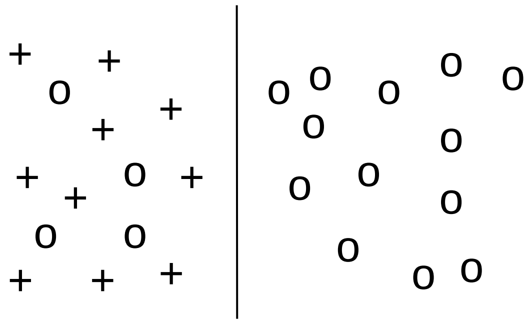
11.1 Applications of learning PU

- For example: It is necessary to build a database of scientific works in the field of data mining.
  - First, use the works of conferences and journals on data mining as positive documents
  - Next, find works on data mining in conferences and journals on databases and artificial intelligence
- Learn with a variety of unlabelled data sources: e.g., find printer websites
  - Find positive pages from amazon.com
  - Use PU learning to find positive pages in other sites
- Learning with negative data is not reliable
- Expanding the dataset
- Covariate shift

11.2 Theoretical background



Classification based only on positive examples



Classification based on positive and negative example

- $(\boldsymbol{x}_i, y_i)$ are random variables taken from the probability distribution $D_{(\boldsymbol{x}i,\, yi)}$, $y \in \{1, -1\}$ is a conditional random variable that we need to estimate when we know $\boldsymbol{x}$, $D_{\boldsymbol{x}|y=1}$ is the conditional distribution on which positive examples are generated, $D_{\boldsymbol{x}}$ is the marginal distribution that produces unlabeled examples.
- The objective is to build a classifier *f* để phân loại văn bản positive và negative, to classify positive and negative documents, the classifier with the smallest probability of error generation $\Pr(f(\boldsymbol{x}) \neq y)$

$Pr(f(\boldsymbol{x}) \neq y) = Pr(f(\boldsymbol{x}) = 1$ và $y = -1) + Pr(f(\boldsymbol{x}) = -1$ và $y = 1)$

We have:

$Pr(f(\boldsymbol{x}) = 1$ và $y = -1)$ $= Pr(f(\boldsymbol{x}) = 1) - Pr(f(\boldsymbol{x}) = 1$ và $y = 1)$

$= Pr(f(\boldsymbol{x}) = 1) - (Pr(y = 1) - Pr(f(\boldsymbol{x}) = -1$ và $y = 1))$

->:

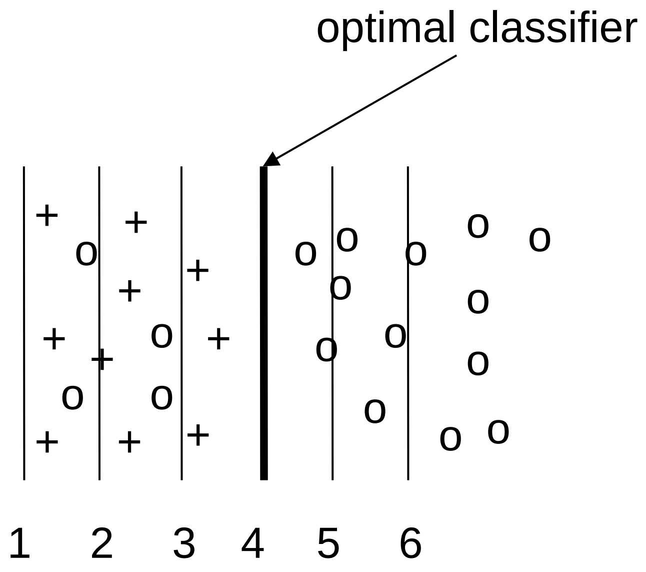
$Pr(f(\boldsymbol{x}) \neq y) = Pr(f(\boldsymbol{x}) = 1) – Pr(y = 1) + 2Pr(f(\boldsymbol{x}) = -1| y = 1)Pr(y = 1)$

Since $Pr(y = 1)$ = const, minimize:

$Pr(f(\boldsymbol{x}) = 1) + 2Pr(f(\boldsymbol{x}) = -1| y = 1)Pr(y = 1)$

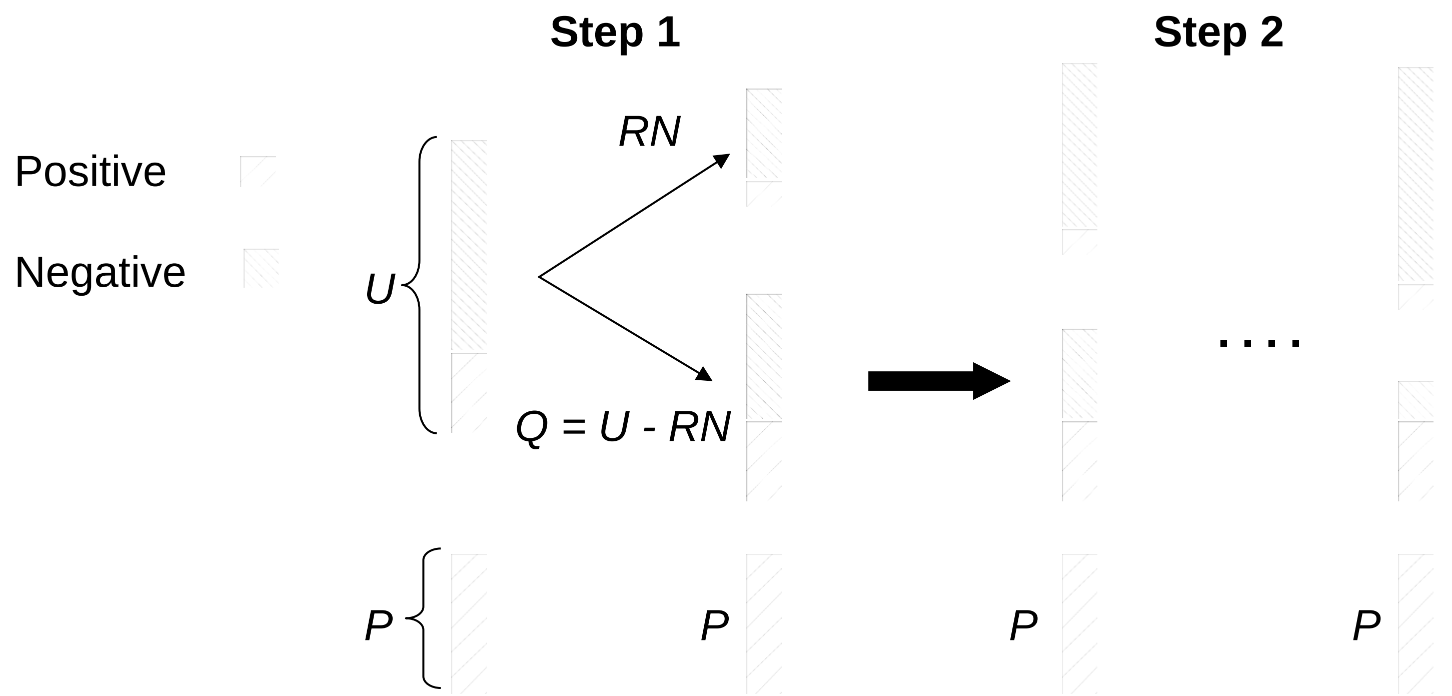
if $Pr(f(\boldsymbol{x}) = -1| y = 1)$ <<, approximation to minimize $Pr(f(\boldsymbol{x}) = 1)$

approximation to minimize $Pr_U(f(\boldsymbol{x}) = 1)$ with $Pr_P(f(\boldsymbol{x}) = 1) \geq r$ (recall)


optimal classifier
1
2
3
4
5
6

11.3 Classifier construction: 2 - steps

- Step 1: Determine the set of reliable negative texts *RN* from U
- Step 2: Build a classifier from P, RN, and U-RN



Step 1:

- Spying Techniques:

1. Randomly select a set S from P and put it in U (line 2-3). Texts in S allow to find positive texts in U

2. Build a classifier NB with PS as the positive set and Us as the negative set (line 3-7). Use the classifier NB to determine the probabilities Pr(1|d) for the documents in Us

3. Use probability Pr(1|d) with the texts in S to find the threshold t. Texts with probability Pr(1|d) < t are included in the set *RN* (line 10-14).

```
Algorithm spy(P, U)
    RN ← ∅;
    S ← sample(P, s%);
    Us ← U ∪ S;
    Ps ← P – S;
    Assign each text in Ps to layer 1;
    Assign each text in Us to layer -1;
    NB(Us, Ps);                          // build NB model
    classifies documents in Us using the NB classifier;
    Determine the probability threshold t based on S;
    for d ∈ Us do
        if probability Pr(1|d) < t then
            RN ← RN ∪ {d};
        endif
    endfor
```

- Cosine-Rocchio Technique:
- Step 1: Determine the potential negative set PN
  - Representing documents into vectors according to TF-IDF
  - Construct the feature vector $\boldsymbol{v}_P$ for the set *P*. Calculate the cosine similarity between the documents in *P* with $\boldsymbol{v}_P$ and arrange them in descending order.
  - A threshold is defined to get all the positive texts hidden in *U* and get the minimum of negative texts.
- Step 2: Build *Rocchio f* classifier based on *P* and *PN*. The documents in *U* are classified as negative because *f* is included in the set *RN*

Algorithm CR(P, U)

$PN = \emptyset$; $RN = \emptyset$;

Represent each document $\boldsymbol{d} \in P$ and U as vector TF-IDF;

$$\mathbf{v}_P = \frac{1}{|P|}\sum_{\mathbf{d}\in P}\frac{\mathbf{d}}{\|\mathbf{d}\|};$$

Calculate cos($\boldsymbol{v}_p$, $\boldsymbol{d}$) for each document $\boldsymbol{d} \in P$;

Sort all texts $\boldsymbol{d} \in P$ based on cos($\boldsymbol{v}_p$, $\boldsymbol{d}$) in descending order;

ω = cos($\boldsymbol{v}_p$, $\boldsymbol{d}$) where $\boldsymbol{d}$ is placed at $(1- l)*|P|$;

for $\boldsymbol{d} \in U$ **do**

  if cos($\boldsymbol{v}_p$, $\boldsymbol{d}$) $< \omega$ **then**

    $PN = PN \cup \{\boldsymbol{d}\}$

$$\mathbf{c}_P = \frac{\alpha}{|P|}\sum_{\mathbf{d}\in P}\frac{\mathbf{d}}{\|\mathbf{d}\|} - \frac{\beta}{|PN|}\sum_{\mathbf{d}\in PN}\frac{\mathbf{d}}{\|\mathbf{d}\|};$$

$$\mathbf{c}_{PN} = \frac{\alpha}{|PN|}\sum_{\mathbf{d}\in PN}\frac{\mathbf{d}}{\|\mathbf{d}\|} - \frac{\beta}{|P|}\sum_{\mathbf{d}\in P}\frac{\mathbf{d}}{\|\mathbf{d}\|};$$

for $\boldsymbol{d} \in U$ **do**

  if cos($\boldsymbol{c}_{PN}$, $\boldsymbol{d}$) > cos($\boldsymbol{c}_P$, $\boldsymbol{d}$) **then**

    $RN = RN \cup \{\boldsymbol{d}\}$

- *1DNF* technique: Word in P and U are collected to build vocabulary (line 1). The built positive PF feature set contains words occurs that is more common in P than U (line 2-7). Documents in U that contain no attributes in PF are included in the RN set (line 8-13).
- NB technique: Use an NB classifier to build a set of RNs from U
- *Rocchio* Technique: Using a *Rocchio* classifier to build RN sets from U

Algorithm 1DNF(P, U)

```
Suppose the vocabulary set V = {w_1, ..., w_n}, w_i ∈ U ∪ P;
Initialize the positive attribute set PF ← ∅;
for each w_i ∈ V do                // freq(w_i, P): number times w_i P
    if (freq(w_i, P) / |P| > freq(w_i, U) / |U|) then
        PF ← PF ∪ {w_i};
    endif
endfor
RN ← U;
for each document d ∈ U do
    if exists w_j such that freq(w_j, d) > 0 and w_j ∈ PF then
        RN ← RN – {d}
    endif
endfor
```

Algorithm NB(P, U)

```
Assign the text in  P: class label 1;
Assign the text in U: class  label -1;
Build classifier NB using P and U;
Use classifier to classify documents in U. Texts are classified negative type is included in the set RN
```

Step 2:

- EM + NB: The expectation step calculates label probabilities of the documents in U – RN. The maximization step re-estimates the classifier parameter NB
- SVM: At each iteration, SVM classifier f is built from *P* and *RN*. The classifier f is used to classify documents in *Q*. Texts classified as negative are removed from *Q* and added to *RN*. The iteration ends when there are no more documents in Q that are classified as negative.

Algorithm EM(*P*, *U*, *RN*)

Each text in *P* is assigned a class label: 1;
Each text in the *RN* is assigned a class label: -1;
Learning a classifier *NB f* from *P* and *RN*
repeat
  for each d_i in *U* - *RN* **do**
    Use the current *f* classifier to calculate $\Pr(c_j | d_i)$
  endfor
  learns the new classifier *NB: f* from *P*, *RN* and *U* - *RN* by computing $\Pr(c_j)$ and $\Pr(w_t | c_j)$
until the classifier parameters are stable

```
Algorithm I-SVM(P, RN, Q)
    Each document in P is assigned a class label 1;
     Each document in RN is assigned a class: –1;
    loop
        Use P and RN to train SVM classifier f;
        Classify Q use f;
        W is the set of documents in Q that are classified as negative;
        if W = ∅ then exit-loop          // converge
        else Q ← Q – W;
            RN ← RN ∪ W;
        endif
```

11.4 Create classifier: SVM

- Given train dataset $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)\ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ where $\mathbf{x}_i$ is vectors and $y_i \in \{1, -1\}$. Assuming the first k examples $\in P$ have a positive label ($y = 1$), the following examples $\in U$ have a negative label ($y = -1$)
- TH1: Set P does not contain errors, theoretically when the number of samples is large enough, a good enough classifier can be obtained if the unlabeled documents are minimized to be classified as positive while the positive example constraint is good classification

Minimize $$\frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\sum_{i=k}^{n} \xi_i$$

constraints : $\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1,\ i = 1, 2, \ldots, k\text{-}1$

$-1(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i,\ i = k, k+1, \ldots n$

$\xi_i \geq 0,\ i = k, k+1, \ldots, n$

- Case 2: The set P contains a negative example (due to noise in the data)

Minimize: $$\frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C_+ \sum_{i=1}^{k-1} \xi_i + C_- \sum_{i=k}^{n} \xi_i$$

Constraint: $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i,\ i = k, k+1, \ldots n$

$\xi_i \geq 0,\ i = k, k+1, \ldots, n$

Where C+ and C- are the weights of positive and negative errors, respectively, determined based on the validation set by measure

$$\frac{r^2}{\Pr(f(\mathbf{x}) = 1)}$$

$r = \Pr(f(x) = 1 | y = 1)$

$p = \Pr(y = 1 | f(\mathbf{x}) = 1)$

We have:

$$\Pr(f(\boldsymbol{x}) = 1 | y = 1)\Pr(y = 1) = \Pr(y = 1 | f(\boldsymbol{x}) = 1) \Pr(f(\boldsymbol{x}) = 1)$$

->

$$\frac{r}{\Pr(f(\boldsymbol{x}) = 1)} = \frac{p}{\Pr(y = 1)} \Rightarrow \frac{r^2}{\Pr(f(\boldsymbol{x}) = 1)} = \frac{pr}{\Pr(y = 1)}$$

Similar to F-score

11.5 Classifier Construction: Probability Estimation

- Example $\boldsymbol{x}$ and label $y \in \{1, -1\}$, $s = 1$ if $\boldsymbol{x}$ is labeled and $s = 0$ if $\boldsymbol{x}$ is unlabel, we have $\Pr(s = 1| \mathbf{x}, y = -1) = 0$
- Objective: Build classification function $f(\boldsymbol{x})$ such that $f(\boldsymbol{x})$ is closest to $\Pr(y = 1| \boldsymbol{x})$
- Totally Random Selection Hypothesis: labeled positive examples are chosen completely randomly from positive examples $\Pr(s = 1| \boldsymbol{x}, y = 1) = \Pr(s = 1| y = 1) = c$
- If P and U are used to build classification function $g(\boldsymbol{x})$ then $g(\boldsymbol{x}) = \Pr(s = 1| \boldsymbol{x})$, then $f(\boldsymbol{x}) = g(\boldsymbol{x}) / c$

$$
\begin{aligned}
g(\boldsymbol{x}) &= \Pr(s = 1 | \boldsymbol{x}) \\
&= \Pr(y = 1 \text{ và } s = 1 | \boldsymbol{x}) \\
&= \Pr(y = 1 | \boldsymbol{x}) \Pr(s = 1 | y = 1, \boldsymbol{x}) \\
&= \Pr(y = 1 | \boldsymbol{x}) \Pr(s = 1 | y = 1) \\
&= f(\boldsymbol{x}) \Pr(s = 1 | y = 1)
\end{aligned}
$$

Estimate c using the validation set V, where V_P là tập các ví dụ positive trong V:

$$\hat{c} = \frac{1}{|V_P|} \sum_{x \in V_P} g(x)$$

$$
\begin{aligned}
g(\boldsymbol{x}) = \Pr(s = 1 | \boldsymbol{x}) \\
&= \Pr(s = 1 | \boldsymbol{x}, y = 1) \Pr(y = 1 | \boldsymbol{x}) + \Pr(s = 1 | \boldsymbol{x}, y = -1) \Pr(y = -1 | \boldsymbol{x}) \\
&= \Pr(s = 1 | \boldsymbol{x}, y = 1) \times 1 + 0 \times 0 \text{ do } \boldsymbol{x} \in V_P \\
&= \Pr(s = 1 | y = 1).
\end{aligned}
$$

Note: If it is only necessary to rank **x** according to the probability of being in positive class, g . can be used directly

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !