Ch 10. Decision Tree

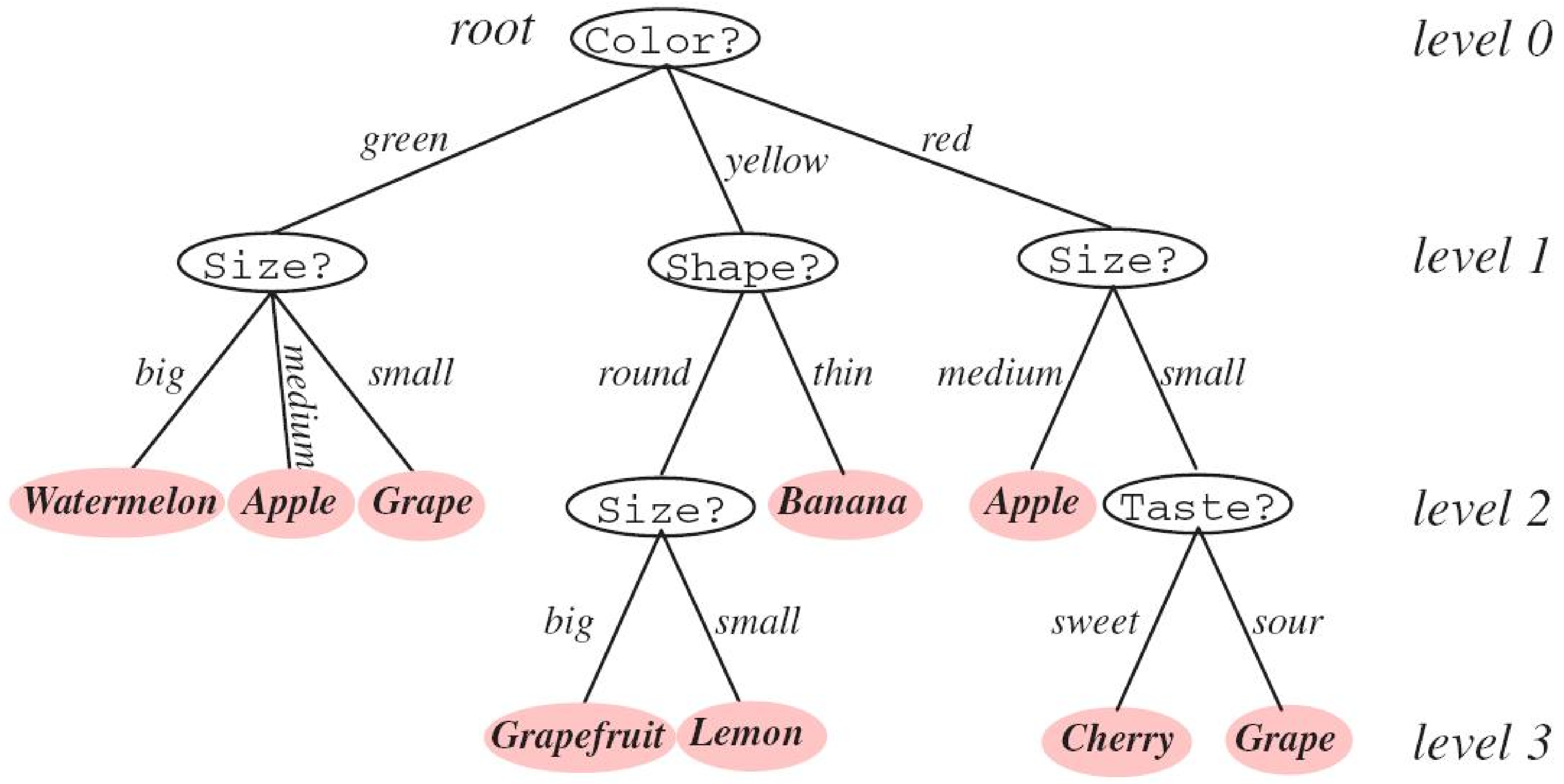
Feature Types

- **Numerical data** (数值数据)
  - e.g. {1.2, 4.5, 3.3}
  - Distance metric can be calculated between patterns
  - Method of pattern classification based on metric
- **Nominal data** (标称数据)
  - e.g. {red，gloss，sweet，small}
  - There is no concept of distance between patterns
  - Non-metric method

Decision Tree

- What is a decision tree?
  - **Decision tree is a tree structure similar to flow chart. Each internal node represents a test(query), each branch of the node represents a result of the test, and each leaf node represents a category.**
- Composition of a decision tree
  - **Root** (根节点)
  - **Branch** (分支)
  - **Leaf node** (叶节点)

Decision Tree

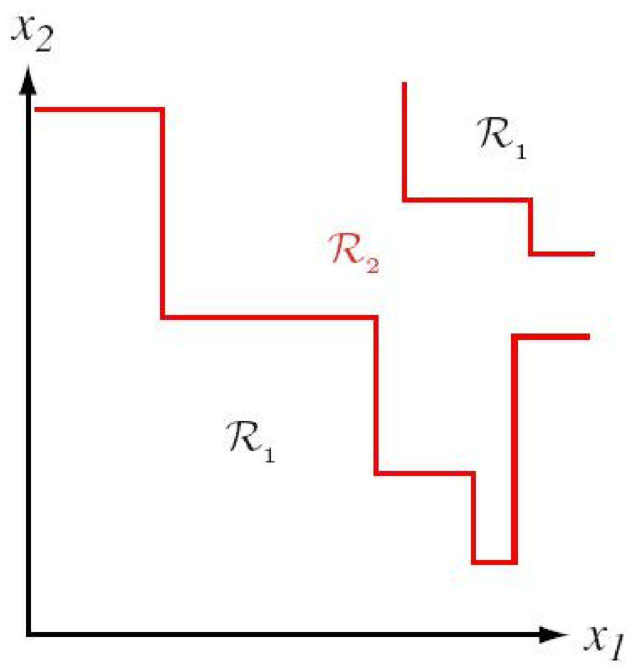


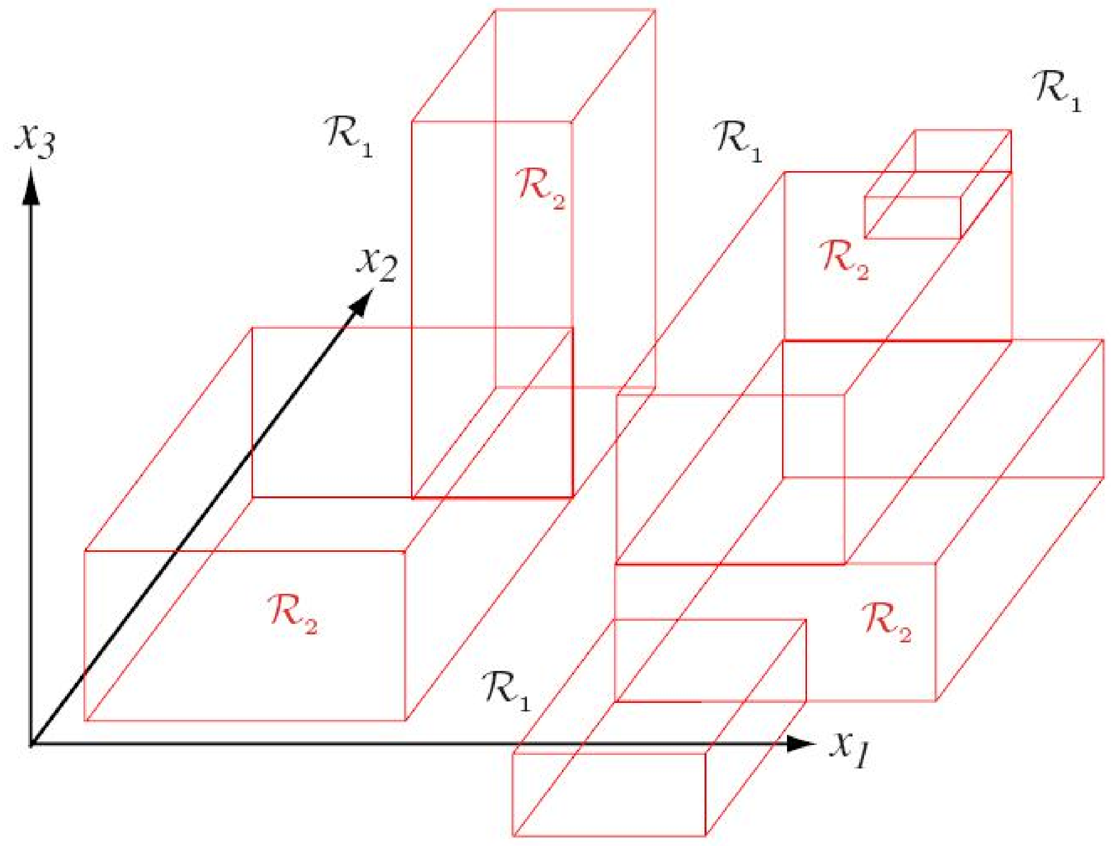
Decision Tree

- Classification process of a decision tree
  - Question the value of a certain attribute from the root
    - **Color?**
  - Different branches connected to the root correspond to different values of this attribute
    - **green; yellow; red;**
  - According to different answers, turn to the corresponding branch
    - **green**
  - Make the same branch decision at the new node
    - **Size?** – **big**.
  - This process continues until reaching a leaf node, then outputs the leaf node's category tag
    - **Watermelon**

Decision Tree

- decision surface of a decision tree





Decision Tree

- Advantages of a decision tree:
  - **Semantic representability**
    - Express from the root to the leaf node as a conjunction
      - （color=yellow）AND（shape=slender） ⇨ banana
    - Obtain a clear description of a class by using conjunction and disjunction
      - apple=（green AND medium-size）OR（red AND medium-size）
  - **Fast classification**
    - A series of simple queries can be used to determine the categories of patterns
  - **The prior knowledge of experts can be embedded naturally**

Decision Tree Learning Algorithm

- Research history of decision tree:
  - **The first decision tree algorithm is called as CLS (Concept Learning System)**
    [E. B. Hunt, J. Marin, and P. T. Stone's book "*Experiments in Induction*"*published by Academic Press in 1966]*
  - **ID3 triggered the upsurge of decision tree research**
    [J. R. Quinlan's paper in a book "*Expert Systems in the Micro Electronic Age*" *edited by D. Michie, published by Edinburgh University Press in 1979]*
  - **C4.5 is the most popular decision tree algorithm**
    [J. R. Quinlan's book "*C4.5: Programs for Machine Learning*" *published by Morgan Kaufmann in 1993]*

Decision Tree Learning Algorithm

- Research history of decision tree:
  - **CART (Classification and Regression Tree) is the general decision tree algorithm**
    [L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone’s book “Classification and Regression Trees” published by Wadsworth in 1984]
  - **Ensemble learning algorithm based on decision tree： Random Forests**
    [L. Breiman’s MLJ’01 paper “Random Forests”]

Construct A Decision Tree

- Basic process
  - From top to bottom, divide-and-conquer (分而治之), recursive growth
  - Initially, all samples are at the root
  - All attributes are nominal(in case of continuous numerical type, discretization is required)
  - All samples are divided recursively according to each selected attribute
    - **The selected attribute is called a split or test or query**
  - Query selection is based on heuristic or statistical characteristics

Construct A Decision Tree

- Basic process
  - If one of the following conditions is met, the partition operation stops
    - **All samples falling into the same node belong to the same category**
      - **This node becomes a leaf node, labeled this category**
    - **No feature can be further used to partition the sample set**
      - **This node becomes a leaf node, and its category label is the category which most samples falling into**
    - **No samples fall into a node**
      - **This node becomes a leaf node, and its category label is the category of most samples falling into the parent node**

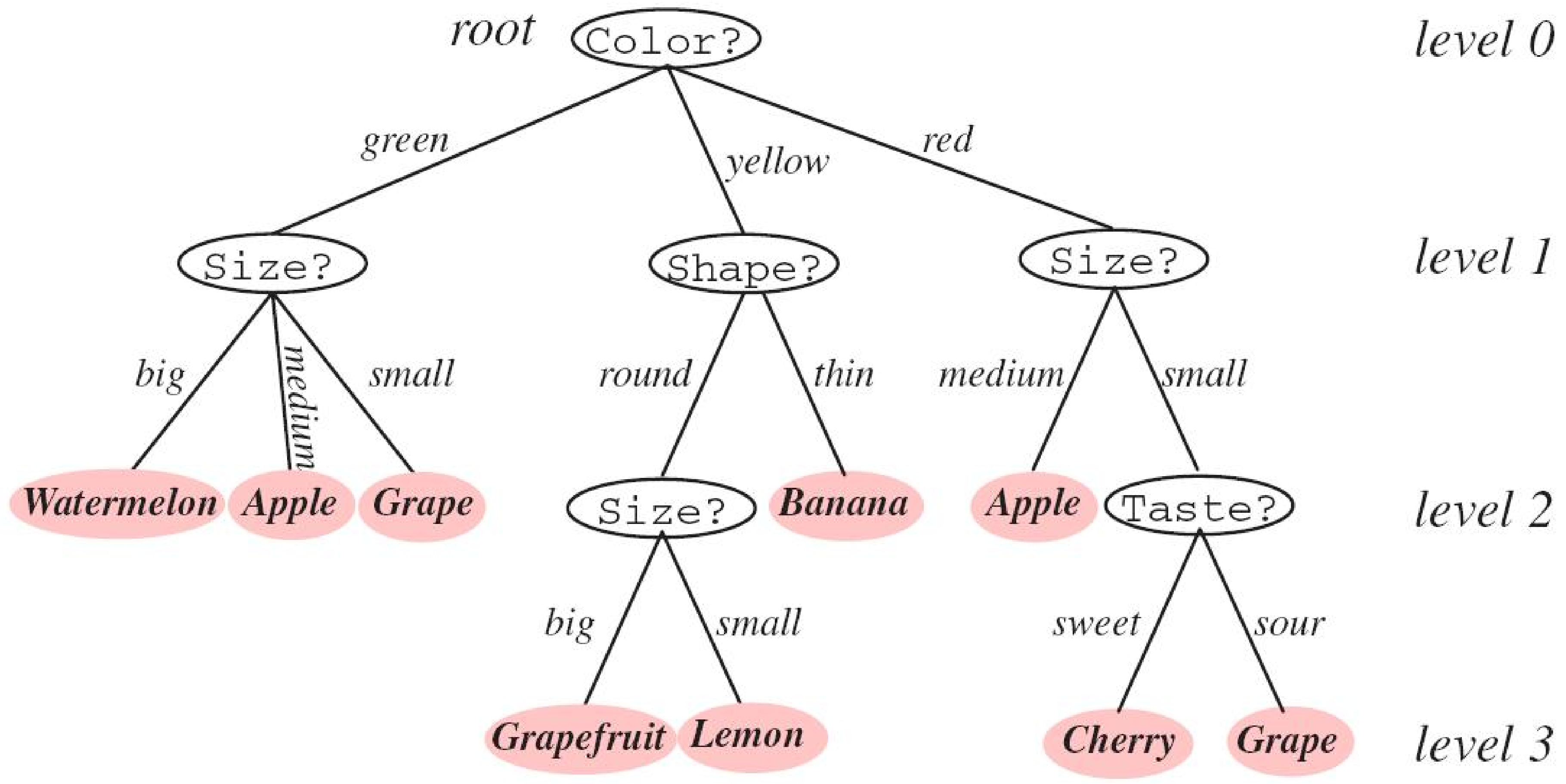
CART

- **Classification And Regression Tree, CART(分类和回归树)**
- CART is a general tree growth algorithm framework, which involves the following problems:
  - **Is the value of the property binary or multivalued? How many branches can a node have?**
  - **How to determine which attribute should be tested at a node?**
  - **When to make a node as a leaf node?**
  - **If the tree grows too large, how to make it smaller and simpler, that is, how to cut branches?**
  - **If not all the samples falling into a leaf node belong to the same category, how to assign a class tag to the leaf node?**

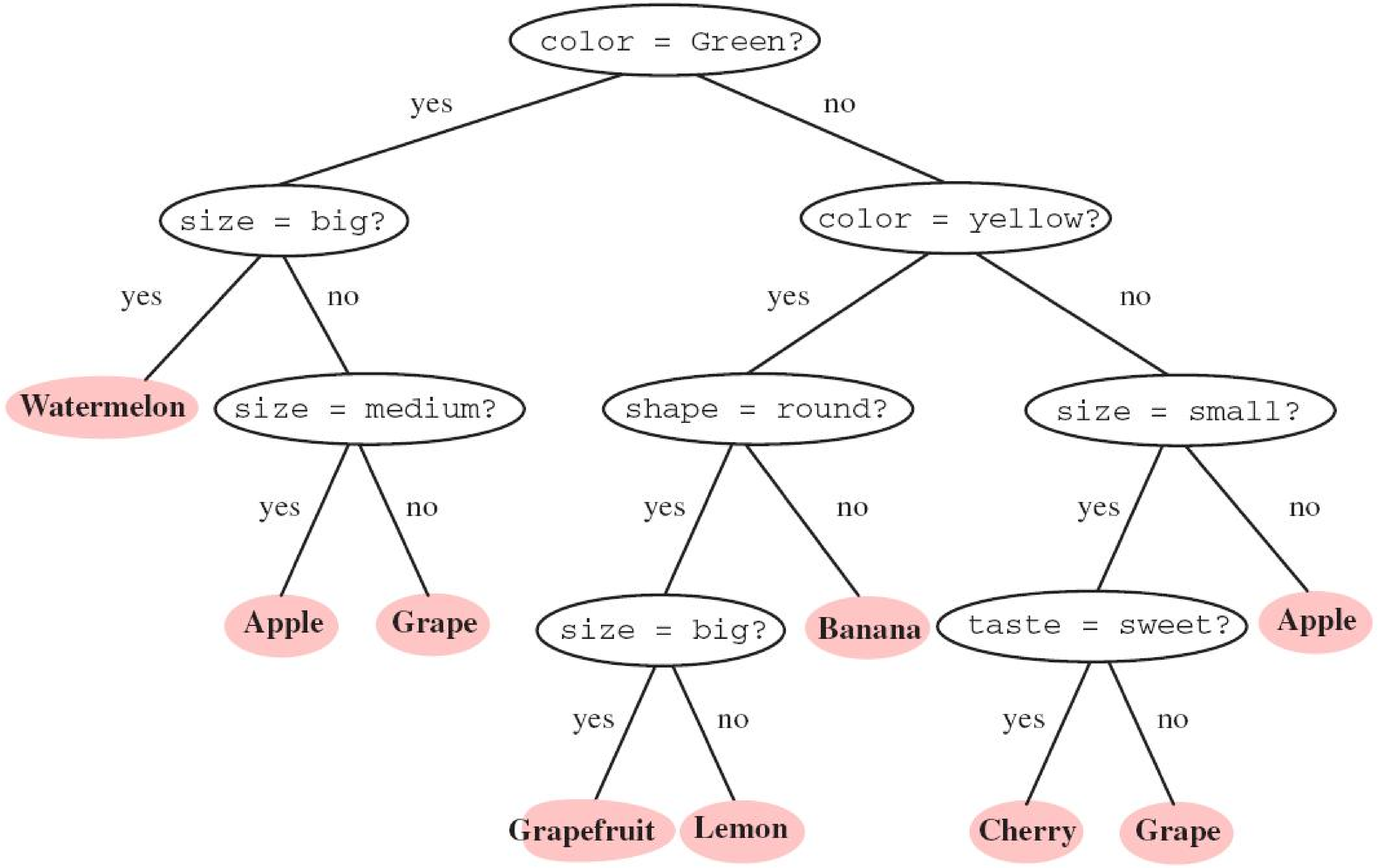
Number of branches

- The number of branches that are divided out of the same node is referred to as the branching ratio（分支率）
- Any decision tree can be represented by a decision tree with a branch factor of 2 (binary tree)
- Binary tree is the most common form of decision tree

Number of branches



Number of branches



Selection of tests

- One of the core issues in decision tree design
- Basic idea:
  Make the data of subsequent nodes as "pure" as possible
- Impurity of node N （不纯度）i(N)
  - When all patterns on N nodes come from the same category, i(N)=0；
  - When the pattern classes on N nodes are evenly distributed， i(N) should be large

Selection of tests

- Common impurity metric
  - **Entropy impurity （熵不纯度）**

$$i(N) = -\sum_{j} P(\omega_j)\log_2 P(\omega_j)$$

$$P(\boldsymbol{\omega}_j) = \frac{\text{Number of samples belonging to } \omega_j}{\text{Total number of samples}}$$

  - **Gini impurity**

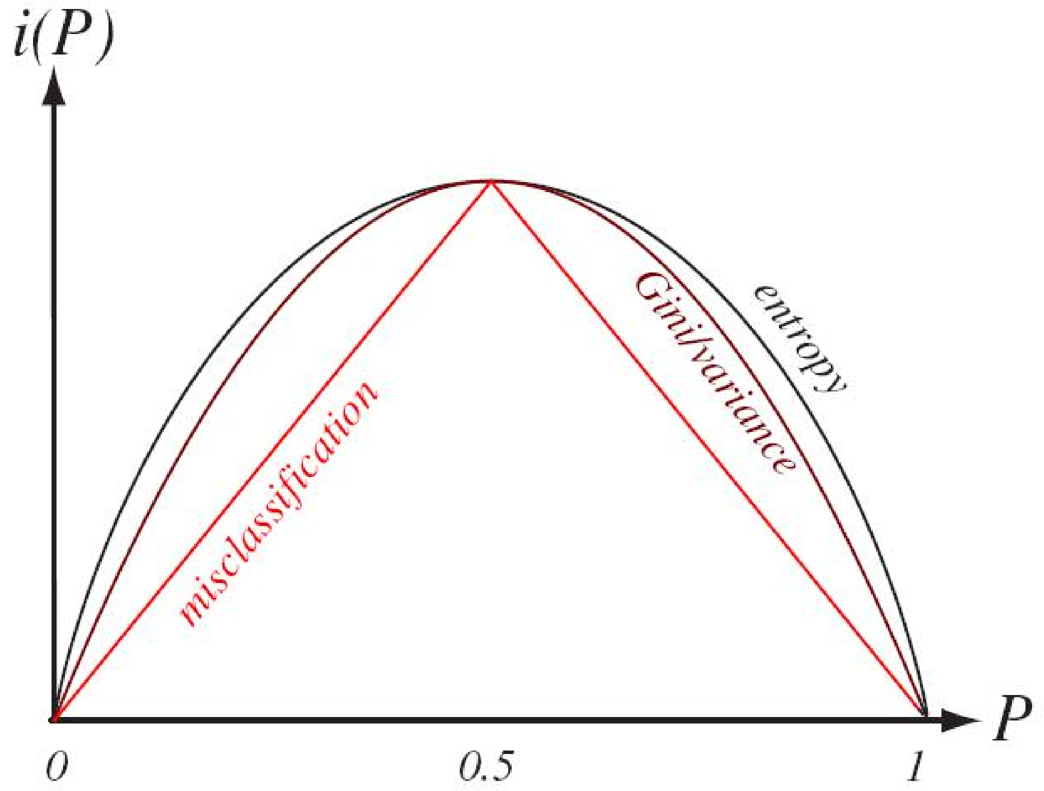
$$i(N) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = 1 - \sum_{j} P^2(\omega_j)$$

  - **Misclassification impurity**

$$i(N) = 1 - \max_{j} P(\omega_j)$$

Selection of tests

- Common impurity metric



Selection of tests

- How to select query for N node?
  - **The query that makes the impurity drop fastest!**

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

- N_L and N_R are left and right child nodes
- $i(N_L)$ and $i(N_R)$ are impurity left and right child nodes
- P_L is the ratio of N node's pattern partition to N_L
- If entropy impure is used, the decrease of impurity is what this query can provide **information gain (信息增益)**

Information gain

Information gain （信息增益）

- S： Total number of samples on node N
- S_i： Number of samples belong to ω_i（i=1,2, …, m）
- a_j： The j-th value of attribute a （j=1,2, …, v）
- Entropy impurity at this node

$$E(S) = -\sum_{i=1}^{m} \frac{S_i}{S} \log_2 \frac{S_i}{S}$$

- Attribute A divides S into V subsets $\{S'_1, S'_2, \ldots, S'_v\}$
- The number of samples belonging to class ω_i in S'_j is S'_{ij}

Information gain

- **Information gain （信息增益）**
  - Using A as a query, the information entropy of V branches is generated:

$$E(A)=\sum_{j=1}^{v}\frac{S'_j}{S}E(S'_j)=\sum_{j=1}^{v}\frac{S'_j}{S}\left(-\sum_{i=1}^{m}\frac{S'_{ij}}{S'_j}\log_2\frac{S'_{ij}}{S'_j}\right)$$

  - Information gain with A as query:

$$Gain(A)=E(S)-E(A)$$

  - **Select the attribute with the largest information gain as the query of N node**

Information gain

- Example
  - Training set

S1: buys_computer= “yes”,
S2: buys_computer= “no”

rid	age	income	student	credit_rating	Class: buys_computer
1	<30	High	No	Fair	No
2	<30	High	No	Excellent	No
3	30-40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	30-40	Low	Yes	Excellent	Yes
8	<30	Medium	No	Fair	No
9	<30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<30	Medium	Yes	Excellent	Yes
12	30-40	Medium	No	Excellent	Yes
13	30-40	High	Yes	Fair	Yes
14	>40	Medium	no	Excellent	No

Information gain

- Entropy impurity on the root

$$E(root) = -\left(\frac{9}{14}\log_2\frac{9}{14} + \frac{5}{14}\log_2\frac{5}{14}\right) = 0.940$$

- Age as the information entropy of query

for age = "< 30":

$S_{11} = 2, S_{21} = 3$ $\quad E(root_1) = -\left(\frac{2}{5}\log_2\frac{2}{5} + \frac{3}{5}\log_2\frac{3}{5}\right) = 0.971$

for age = "30-40":

$S_{12} = 4, S_{22} = 0$ $\quad E(root_2) = 0$

for age = "> 40":

$S_{13} = 3, S_{23} = 2$ $\quad E(root_3) = -\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right) = 0.971$

$$E(age) = \frac{5}{14}E(root_1) + \frac{4}{14}E(root_2) + \frac{5}{14}E(root_3) = 0.694$$

Information gain

- Age as the information entropy of query

$$Gain(age) = E(root) - E(age) = 0.246$$

- Similarly, the information gain of all attributes can be calculated

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

- The information gain of age is the largest, so age is selected as the query of root to partition the training set for the first time

Information gain ratio

- The disadvantages of information gain as query selection criteria are as follows:
  Preference for attributes with more different values
- In order to overcome this shortcoming, J. R. Quinlan used the gain ratio (信息增益率) as the selection criterion in his famous C4.5 algorithm

$$Gain\_ratio(A) = \frac{Gain(A)}{IV(A)}$$

$$IV(A) = -\sum_{j=1}^{v} \frac{S_j'}{S} \log_2 \frac{S_j'}{S}$$

Information gain ratio

- Example

$$IV(age) = -\left(\frac{5}{14}\log_2\frac{5}{14} + \frac{4}{14}\log_2\frac{4}{14} + \frac{5}{14}\log_2\frac{5}{14}\right) = 1.578$$

$$IV(income) = -\left(\frac{4}{14}\log_2\frac{4}{14} + \frac{6}{14}\log_2\frac{6}{14} + \frac{4}{14}\log_2\frac{4}{14}\right) = 1.556$$

$$IV(student) = -\left(\frac{7}{14}\log_2\frac{7}{14} + \frac{7}{14}\log_2\frac{7}{14}\right) = 1.661$$

$$IV(credit\_rating) = -\left(\frac{8}{14}\log_2\frac{8}{14} + \frac{6}{14}\log_2\frac{6}{14}\right) = 0.985$$

$$\boxed{Gain\_ratio(age) = \frac{Gain(age)}{IV(age)} = 0.156}$$

$$Gain\_ratio(income) = \frac{Gain(income)}{IV(income)} = 0.019$$

$$Gain\_ratio(student) = \frac{Gain(student)}{IV(student)} = 0.091$$

$$Gain\_ratio(credit\_rating) = \frac{Gain(credit\_rating)}{IV(credit\_rating)} = 0.049$$

Gini Impurity

- S： Total number of samples on node n
- S_i： The number of samples belonging to class ω_i（i=1,2, …, m）
- a_j： The j-th value of attribute A（j=1,2, …, v）
- Gini impurity at this node

$$Gini(S) = 1 - \sum_{i=1}^{m} \left(\frac{S_i}{S} \right)^2$$

- Attribute A divides S into v subsets $\{S'_1, S'_2, \ldots, S'_v\}$
- The number of samples belonging to class ω_i in S'_j is S'_{ij}

Gini Impurity

- Gini impurity with a query of A resulting in V branches

$$Gini(A) = \sum_{j=1}^{v} \frac{S'_j}{S} Gini(S'_j) = \sum_{j=1}^{v} \frac{S'_j}{S} \left(1 - \sum_{i=1}^{m} \left(\frac{S'_{ij}}{S'_j} \right)^2 \right)$$

- **Select the attribute with the greatest Gini impurity difference (the smallest Gini(A)) as the query for the N node**

Gini Impurity

- Example

$$Gini(age) = \frac{5}{14}\left(1-\left(\left(\frac{2}{5}\right)^2+\left(\frac{3}{5}\right)^2\right)\right)+\frac{4}{14}\left(1-\left(\left(\frac{4}{4}\right)^2+\left(\frac{0}{4}\right)^2\right)\right)$$

$$+\frac{5}{14}\left(1-\left(\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right)\right)=0.343$$

$$Gini(income) = ?$$

$$Gini(student) = ?$$

$$Gini(credit\_rating) = ?$$

Branch Stop Criterion

- If the decision tree continues to grow until all the leaf nodes reach the minimum impurity, then "**over-fitting**" will generally occur.
  - Extreme case: all leaf nodes correspond to only one training sample, in which case the decision tree degenerates to a lookup table
- If the branch stops too early, the training samples are poorly fitted, resulting in poor classification performance
- Common Branch Stop Criteria
  - **Cross-validation**
  - **Preset a threshold for impurity drop**
  - **Monitor if the number of samples represented by each node is less than a threshold**

Branch Stop Criterion

- **Minimize the following metric**

$$\underbrace{\alpha \cdot size}_{\text{Regular Item}} + \sum_{leaf\ nodes} i(N)$$

- **Statistical significant analysis of impurity decrease**
  - **Stop branching if a division does not significantly reduce impurity**

Pruning

- **Pruning（剪枝）to eliminate overfitting**
- **Pre-pruning** and **post-pruning**
  - Pre-pruning refers to the branch-stop technique mentioned earlier, i.e. stopping the division when the tree is growing to a certain condition
  - Post-pruning refers to first letting the tree grow sufficiently until the leaf nodes have the least impurity, then pruning the tree
    - **Cross-validation techniques can be used to determine which branches are cut off**
    - **Cut off branches that minimize impurity growth**
  - **In general, post-pruning performs better, but requires more computation**

Marker of leaf node

- If the corresponding sample of a leaf node comes from the same category, the leaf node is marked with this category
- In general, leaf nodes have pure impurity, and the leaf node is marked with the dominant sample category

ID3

- **ID3**: Interactive Dichotomizer-3（交互式二分法第三版）
- Only for nominal (out of order) data
  If real-value data is involved, it needs to be discretized and then processed as nominal data
- The branching factor for each partition is equal to the number of values of the query attributes
- Using Information Gain Ratio as the Base for Selecting Queries
- The algorithm stops until all leaf nodes have the least impurity or no attributes available for partitioning
- No pruning steps in Standard Edition

C4.5

- **C4.5**: Succession and improvement of ID3 algorithm
- Can process real value data
- The **branching factor** for each partition is equal to the number of values of the query attributes
- Using **Information Gain Rate** as the Base for Selecting Queries
- First allow the tree to grow sufficiently, then use the **statistical significance** of the branches to prune
- **C4.5 is one of the most popular decision tree algorithms at present**

Ch 11. Clustering

Unsupervised Learning

- **Supervised（有监督）** learning
  - Each sample in the training set has a category label
  - All categories are known in advance
  - Commonly used in: classification, regression

- **Unsupervised（无监督）** learning
  - The category label of samples in training set is unknown
  - Given a set of samples, find its intrinsic properties, such as category or clustering
  - Commonly used in: clustering, probability density estimation

Motivation of unsupervised learning

- **Collecting and tagging a large number of patterns is often costly**
  - We hope to train a rough classifier on a small labeled sample set, and then let the classifier run unsupervised on a large sample set
  - Or a large number of unlabeled samples are used to train the classifier to automatically discover the groups in the data, and then labeled these groups with more expensive methods(such as manual)
- **In many applications, the characteristics of patterns change over time**
  - The classification performance will be greatly improved if this feature change can be captured by a classifier running in unsupervised mode

Motivation of unsupervised learning

- **Unsupervised methods can be used to extract features or preprocess existing features to prepare for subsequent pattern recognition problems**
  - e.g. PCA dimensionality reduction
- **In any exploratory work, unsupervised methods can reveal some internal structures and laws of observation data**
  - The inner clustering or grouping of patterns may provide a basis for classifier design

Clustering

- **clustering（聚类）**
  - An independent data clustering is the process of naturally grouping physical or abstract objects so that each group is made up of similar objects
  - Clustering is an **unsupervised learning** (无监督学习) process because training set samples have no category labels
  - A **cluster** (聚类) is a set of samples that are similar to those belonging to the same cluster but not to other clusters.
  - Clustering can be used as:
    - analysis tool for analyzing the intrinsic characteristics of data
    - A data preprocessing method for subsequent pattern recognition services

Similarity metric

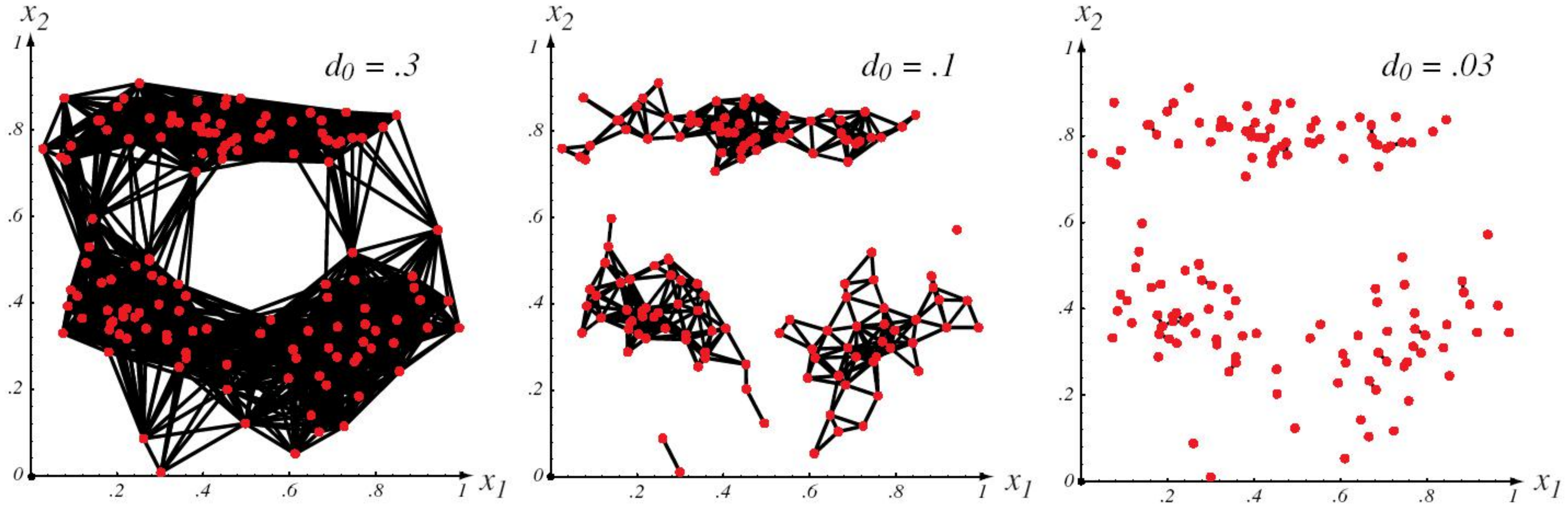
- "**Samples within the same cluster are more similar than those within different clusters**" A metric of similarity (or dissimilarity) between samples based on a definition
- Two main types of similarity (dissimilarity) metric
  - Metric-based distance criteria
  - Nonmetric similarity function
- A metric (a distance function) satisfy the following conditions
  - **Nonnegative**： $d(\mathbf{x}_1, \mathbf{x}_2) \geq 0$
  - **Reflexivity**： $d(\mathbf{x}_1, \mathbf{x}_2) = 0$ if and only $\mathbf{x}_1 = \mathbf{x}_2$
  - **Symmetry**： $d(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_2, \mathbf{x}_1)$
  - **Trigonometric Inequality**： $d(\mathbf{x}_1, \mathbf{x}_2) + d(\mathbf{x}_2, \mathbf{x}_3) \geq d(\mathbf{x}_1, \mathbf{x}_3)$

Distance metric

- Common distance metric
  - The most commonly used distance metric is Euclidean distance as a metric of dissimilarity
  - Next is the Mahalanobis distance considering data distribution
- Clustering samples based on distance
  - Calculate the distance between any two samples
  - If the distance between two samples is less than a threshold d_0, then these two samples belong to the same cluster
    - d_0 is too large, all samples are divided into the same cluster
    - d_0 is too small, each sample has its own cluster

Distance metric

Euclidean distance

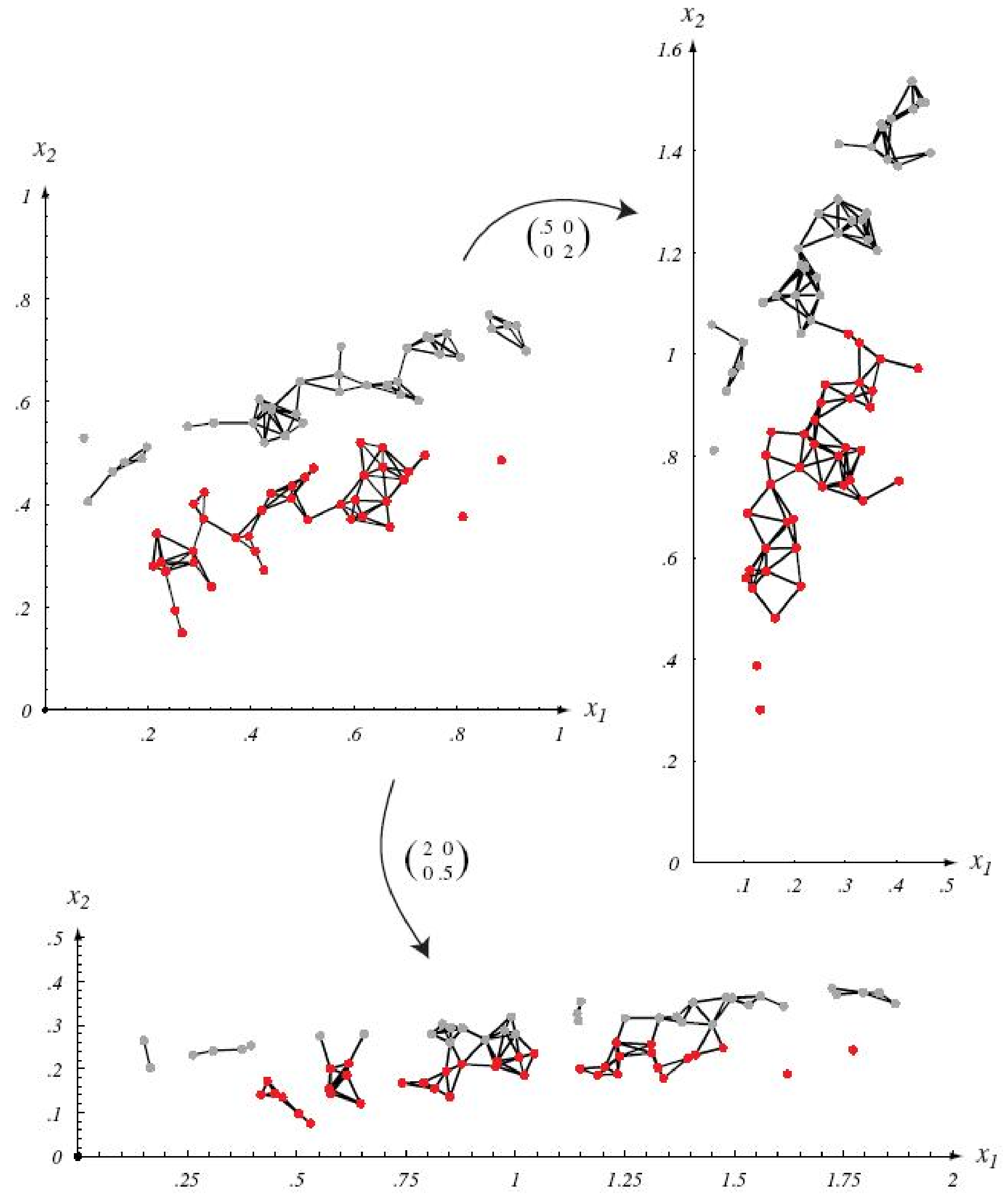


The smaller d_0, the smaller each cluster and the larger the number of clusters

Distance metric

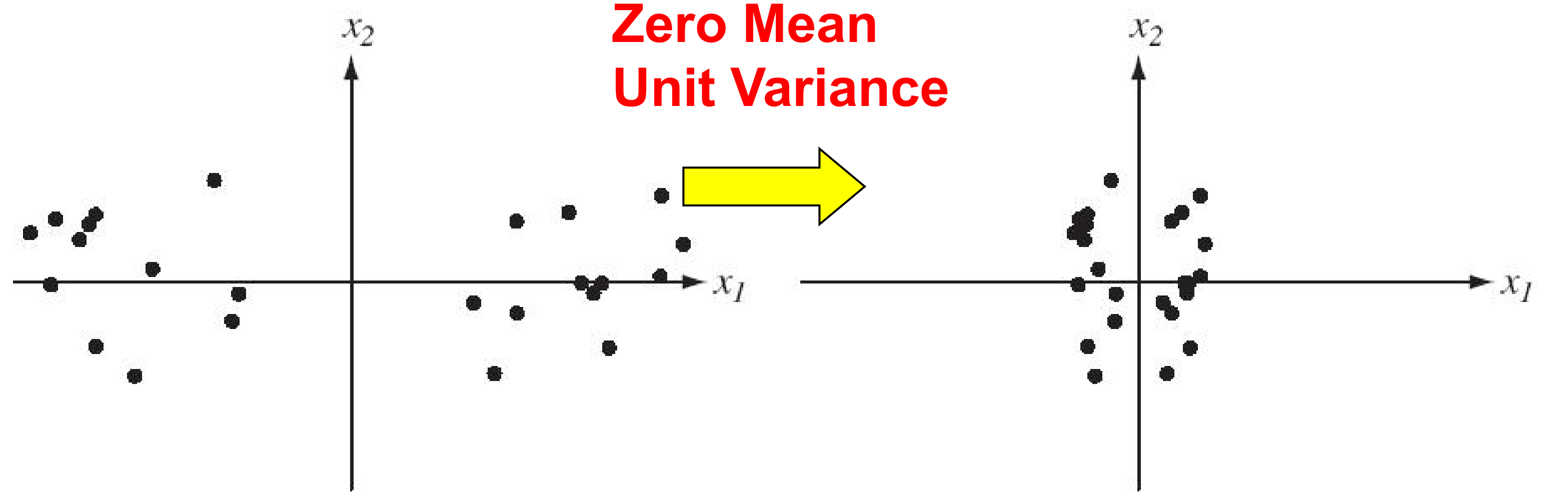
- Scaling coordinate scales

causes changes in clustering results



Normalization

- Raw data can be **normalized** (规格化) before clustering to achieve invariance
  - **Displacement and Scaling Invariance**
    - By shifting and scaling, new features have zero mean and unit variance
  - **Rotation Invariance**
    - Rotate the axis so that it is parallel to the eigenvectors of the sample covariance matrix
- Normalization cannot be abused!



Similarity function

- Non-metric similarity functions can be directly defined without using distance
- Similarity function is a function of measuring similarity between samples
  - Symmetry $s(\mathbf{x}_1, \mathbf{x}_2) = s(\mathbf{x}_2, \mathbf{x}_1)$
  - When two samples have some similarity, the value of the function is larger
- Common Similarity Functions
  - **Normalized inner product (归一化内积)**（cosine of angle between two vectors）
    $s(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\|\mathbf{x}_1\| \|\mathbf{x}_2\|}$
  - For binary features (0-1), the normalized inner product corresponds to the relative count of shared attributes, further simplifying to the proportion of shared attributes
    $s(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{d}$

What is a good cluster?

- A good clustering process produces high-quality clustering
  - **High similarity within clusters**
  - **Low similarity between clusters**

- The quality of clustering results depends on the similarity metric used and the implementation of clustering algorithm
- **Evaluating clustering results is often subjective**

Criterion functions for clustering

- **"One cluster partition is better than another"** is a criterion function based on some kind of clustering
- **Clustering can be viewed as a discrete optimization problem**
  - Criterion functions are used to metric the quality of a partition of a data cluster
  - The goal is to find a partition that minimizes (enlarges) the criterion function
- Different criterion functions may result in different clustering results
- Common Criterion Functions
  - **Square error criterion**
  - **Minimum variance criterion**
  - **Dispersion Criteria**

Square error criterion

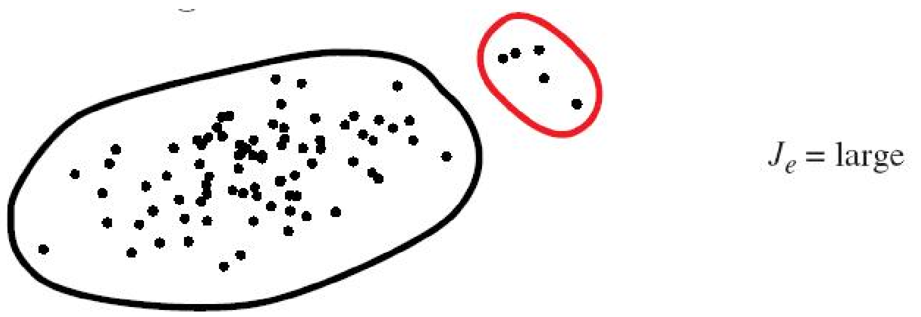
- **Square error criterion** is the simplest and most widely used clustering criterion function

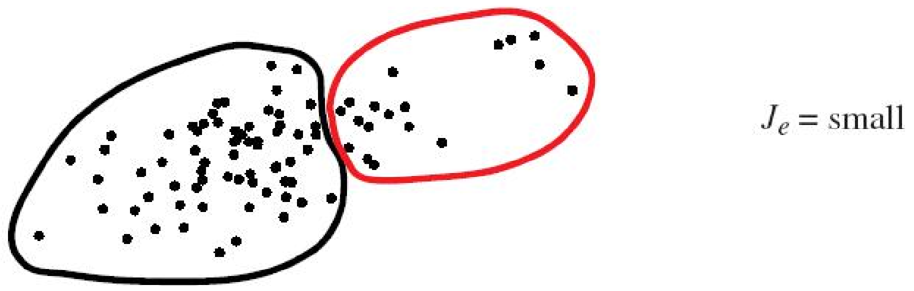
$$J_e = \sum_{i=1}^{c} \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|$$

  - where $\mathbf{m}_i$ is the mean of D_i sample in the first cluster
- When data points can be divided into well-differentiated clusters with dense interiors, **Square error criterion** is applied

Square error criterion

- Possible problems with using **square error criterion**
  - When there is a large difference in the number of samples contained in different clusters, splitting a large cluster may result in a smaller sum of squared errors





Minimum variance criterion

- Because the square of errors criterion metric the variance from the sample point to the cluster mean, it is one of the minimum variance criteria
- Form equivalent to error squared sum rule

$$J_e = \frac{1}{2}\sum_{i=1}^{c} n_i \,\overline{s_i} \quad \text{where} \quad \overline{s_i} = \frac{1}{n_i^2}\sum_{\mathbf{x}\in D_i}\sum_{\mathbf{x}'\in D_i} \|\mathbf{x}-\mathbf{x}'\|^2$$

where n_i is the number of samples in the i-th cluster

- General form of minimum variance criterion

$$J_e = \frac{1}{2}\sum_{i=1}^{c} n_i \,\overline{s_i} \quad \text{where} \quad \overline{s_i} = \frac{1}{n_i^2}\sum_{\mathbf{x}\in D_i}\sum_{\mathbf{x}'\in D_i} s(\mathbf{x},\mathbf{x}')$$

$s(\mathbf{x},\mathbf{x}')$ is a similarity function

Dispersion Criteria

- Mean Vector
  - Mean Vector of the i-th Cluster

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}$$

  - Total Mean Vector

$$\mathbf{m}_i = \frac{1}{n} \sum_{\mathbf{x} \in D} \mathbf{x} = \frac{1}{n} \sum_{i=1}^{c} n_i \, \mathbf{m}_i$$

Dispersion Criteria

Dispersion Matrix

- Dispersion matrix of the i-th cluster

$$\boldsymbol{S}_i = \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$$

- Total dispersion matrix

$$\boldsymbol{S}_T = \sum_{\mathbf{x} \in D} (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^t$$

- Within-cluster dispersion matrix

$$\boldsymbol{S}_W = \sum_{i=1}^{c} \boldsymbol{S}_i$$

Dispersion Criteria

- **Dispersion matrix**
  - Between-cluster dispersion matrix

$$\boldsymbol{S}_B = \sum_{i=1}^{c} n_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^{\boldsymbol{t}}$$

  - The Relationship between the dispersion matrix within -cluster and the dispersion matrix between-cluster

$$\boldsymbol{S}_T = \boldsymbol{S}_W + \boldsymbol{S}_B$$

Dispersion Criteria

- For better clustering quality, we expect smaller intra-cluster and larger inter-cluster dispersion
- The "size" of a scalar metric matrix is required, such as a trace of the matrix （**trace** (迹), the sum of elements on the diagonal of the matrix ）

$$tr[\boldsymbol{S}_W] = \sum_{i=1}^{c} tr[\boldsymbol{S}_i] = \sum_{i=1}^{c} \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = J_e$$

- Since $tr[\boldsymbol{S}_T] = tr[\boldsymbol{S}_W] + tr[\boldsymbol{S}_B]$ and $tr[\boldsymbol{S}_T]$ are independent of how clustering is divided, minimizing $tr[\boldsymbol{S}_W]$ $(= J_e)$ maximizes the trace of the scatter matrix between clusters at the same time

$$tr[\boldsymbol{S}_B] = \sum_{i=1}^{c} n_i \|\mathbf{m}_i - \mathbf{m}\|^2$$

Scalar metric can also use the determinant of a matrix

Iterative optimization

- For a finite sample set, the number of possible partitions is limited, and in theory the optimal solution can be found by the method of exhaustion. However, the method of exhaustion is often too computationally intensive to achieve
- **Iterative optimization methods** are often used to find optimal partitions
  - Start with some reasonable initial division
  - Then move some samples from one cluster to another -- if this improves the criteria function
  - Repeat iteration until no significant improvement is achieved
- This iteration method guarantees convergence to local optimum, but not global optimum

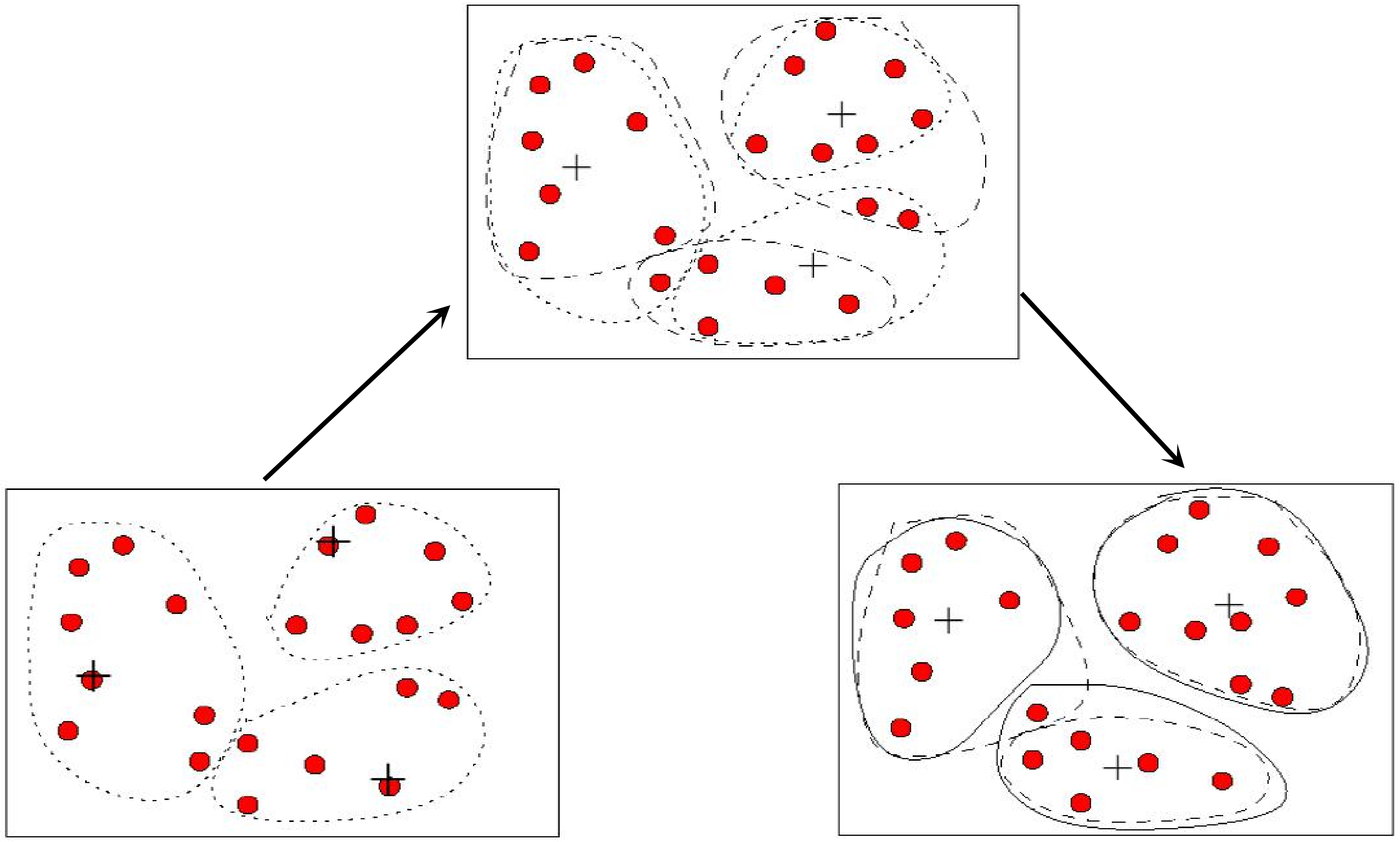
Partition-based clustering method

- Given a dataset, the dataset is divided into k subsets based on partitioning, each of which corresponds to a cluster
- Two scenarios
  - Each cluster is represented by the mean of the sample it contains
  - Each cluster is represented by a sample (center point) close to the cluster center
- Typical algorithm
  - **k-means**
  - **k-medoids**

k-means algorithm

- **Each cluster is represented by the mean value of the samples it contains**
  - Step 1: randomly select k samples as the centers of K clusters
  - Step 2: each remaining sample is divided into the cluster whose center is closest to the sample
  - Step 3: calculate the mean value of each cluster as the new center
  - Step 4: if there is no change in the cluster center, the algorithm will stop. Otherwise, return to step 2

k-means algorithm



K-medoids algorithm

- **Each cluster is represented by a sample near the center of the cluster**
  - Step 1: randomly select k samples as the centers of K clusters
  - Step 2: each remaining sample is divided into the cluster whose center is closest to the sample
  - Step 3: calculate the medoid of each cluster（The sample nearest to the mean）
  - Step 4: if there is no change in the medoid of clustering, the algorithm will stop, otherwise return to step 2

K-medoids algorithm