Introduction to Artificial Intelligence

Unsupervised Learning

Jianmin Li

Department of Computer Science and Technology
Tsinghua University

Spring, 2024

Outline

1. Introduction
   - What?
   - Why?
2. Clustering
   - Introduction
   - Hierarchical clustering
   - Partitional clustering
3. Unsupervised Dimension reduction
   - Introduction
   - Principle component analysis
   - Kernel PCA

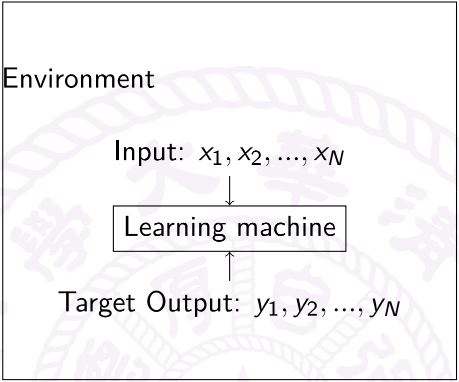
Outline

Settings of learning

- An organism or machine
  - Experiences a series of sensory inputs: $x_1, x_2, \ldots, x_N$
- Supervised learning
  - The machine is also given desired outputs $y_1, y_2, \ldots, y_N$
- Unsupervised learning
  - Nothing else

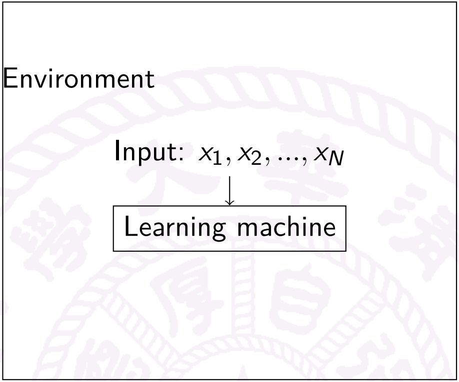
Supervised Learning

- To learn to produce the correct output given a new input.
- Classification
  - The desired outputs y_i are discrete class labels.
- Regression
  - The desired outputs y_i are continuous valued.



Unsupervised Learning

- Build a model or find useful representations of the input for
  - decision making
  - predicting future inputs
  - efficiently communicating the inputs to another machine
  - etc.
- Find patterns (discover the structure) in the data



What can we learn from the unlabeled data?

- Finding clustering
  - partition examples into groups when no pre-defined categories/classes are available
- Dimensionality reduction
  - Reduce the number of variables under consideration
- Outlier detection
  - Identification of new or unknown data or signal that a machine learning system is not aware of during training
- Finding the hidden causes or sources of the data
- Modeling the data density

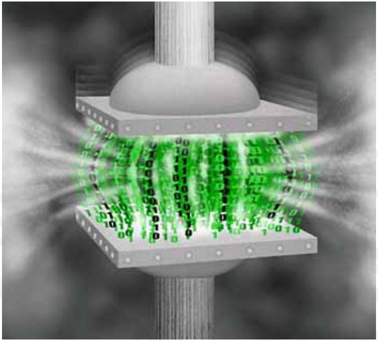
Outline

Importance

- Applications
  - Data compression
  - Intrusion detection
  - Organize search results
  - Segment customer population for targeted marketing
  - Make other learning tasks easier
- A theory of human learning and perception
  - Unsupervised learning is likely to be much more common in the brain than supervised learning

Importance

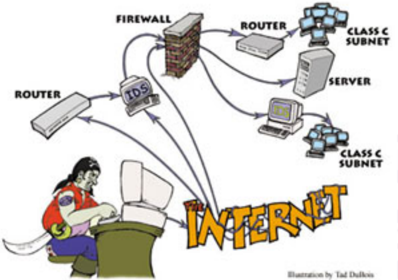
- Applications
  - Data compression
  - Intrusion detection
  - Organize search results
  - Segment customer population for targeted marketing
  - Make other learning tasks easier



- A theory of human learning and perception
  - Unsupervised learning is likely to be much more common in the brain than supervised learning

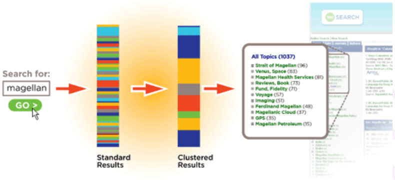
Importance

- Applications
  - Data compression
  - Intrusion detection
  - Organize search results
  - Segment customer population for targeted marketing
  - Make other learning tasks easier
- A theory of human learning and perception
  - Unsupervised learning is likely to be much more common in the brain than supervised learning



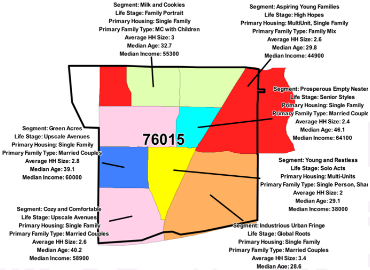
Importance

- Applications
  - Data compression
  - Intrusion detection
  - Organize search results
  - Segment customer population for targeted marketing
  - Make other learning tasks easier
- A theory of human learning and perception
  - Unsupervised learning is likely to be much more common in the brain than supervised learning



Importance

- Applications
  - Data compression
  - Intrusion detection
  - Organize search results
  - Segment customer population for targeted marketing
  - Make other learning tasks easier



- A theory of human learning and perception
  - Unsupervised learning is likely to be much more common in the brain than supervised learning

Importance

- Applications
  - Data compression
  - Intrusion detection
  - Organize search results
  - Segment customer population for targeted marketing
  - Make other learning tasks easier
- A theory of human learning and perception
  - Unsupervised learning is likely to be much more common in the brain than supervised learning

Importance

- Applications
  - Data compression
  - Intrusion detection
  - Organize search results
  - Segment customer population for targeted marketing
  - Make other learning tasks easier
- A theory of human learning and perception
  - Unsupervised learning is likely to be much more common in the brain than supervised learning

Importance

- Applications
  - Data compression
  - Intrusion detection
  - Organize search results
  - Segment customer population for targeted marketing
  - Make other learning tasks easier
- A theory of human learning and perception
  - Unsupervised learning is likely to be much more common in the brain than supervised learning

Outline

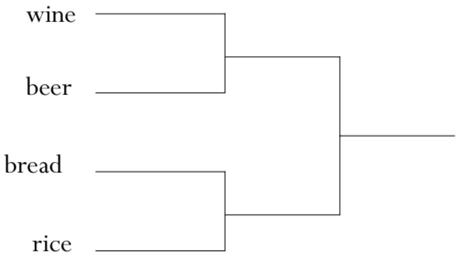
What is Clustering?

- Also known as
  - Cluster analysis, automatic classification, numerical taxonomy, botryology and typological analysis
- Assignment of objects into groups (called clusters) so that:
  - objects within the same cluster are similar
  - objects in different clusters are different
- To help understand the natural grouping or structure in a data set or get insight into data distribution

Types

Hierarchical clustering vs Non-hierarchical clustering

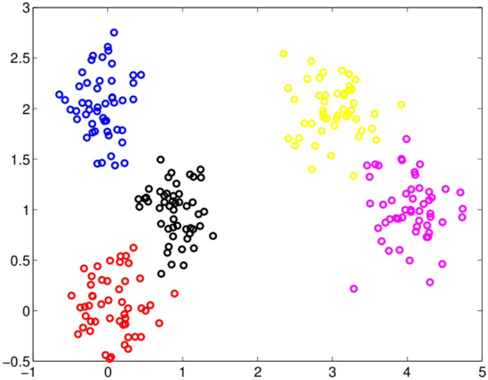
- Hierarchical clustering
  - A hierarchy (tree) of clusters
- Non-hierarchical clustering
  - Flat, one layer



Types

Hierarchical clustering vs Non-hierarchical clustering

- Hierarchical clustering
  - A hierarchy (tree) of clusters
- Non-hierarchical clustering
  - Flat, one layer



Types

Hard clustering vs Soft clustering

- Hard clustering
  - each item can only belong to one cluster
- Soft clustering
  - each item can belong to more than one cluster

	eat	drink	make
wine	0	3	1
beer	0	5	1
bread	4	0	2
rice	4	0	0

Types

Hard clustering vs Soft clustering

- Hard clustering
  - each item can only belong to one cluster
- Soft clustering
  - each item can belong to more than one cluster

	parsing	estimation	prediction	translation
document 1	0	3	2	0
document 2	0	5	1	0
document 3	1	1	1	0
document 4	4	0	0	3

What we need to cluster data?

- Data
- Distance measure or similarity measure for objects
- Evaluation metric for clusters
- Clustering algorithm

Data

- Vector $x \in D_1 \times D_2 \times \cdots D_N$
- Type
  - Real valued
    - $D = R$
  - Binary valued
    - $D = \{v_1, v_2\}$
    - e.g. {Female, Male}
  - Nominal values
    - $D = \{v_1, v_2, \cdots, v_M\}$
    - e.g. {Mon., Tue., Wed., Thu., Fri., Sat., Sun.}
  - Ordinal values
    - $D = R$ or $D = \{v_1, v_2, \cdots, v_M\}$
    - Order is important, e.g., rank

Similarity

Real valued variable

- Similarity
  - Inner product
  - Cosine
  - Kernels
- Minkowski distance
  - Manhattan distance
  - Euclidean distance
  - Chebyshev distance

Similarity

Nominal variable

- Examples
  - {Mon., Tue., Wed., Thu., Fri., Sat., Sun.}
  - {Boston, LA, New York, San Francisco, Seattle}
- Binary rule
  - if $x_i = y_i$ then $sim(x_i, y_i) = 1$, else $sim(x_i, y_i) = 0$
- Underlining semantic property:
  - $$sim(Boston, LA) = \alpha dist(Boston, LA)^{-1}$$
  - $$sim(Boston, LA) = 1 - \alpha \frac{(|size(Boston) - size(LA)|)}{\max(size(cities))}$$
- Similarity matrix

Similarity

Nominal variable

- Examples
  - {Mon., Tue., Wed., Thu., Fri., Sat., Sun.}
  - {Boston, LA, New York, San Francisco, Seattle}
- Binary rule
  - if $x_i = y_i$ then $sim(x_i, y_i) = 1$, else $sim(x_i, y_i) = 0$
- Underlining semantic property:
  - $$sim(Boston, LA) = \alpha dist(Boston, LA)^{-1}$$
  - $$sim(Boston, LA) = 1 - \alpha \frac{(|size(Boston) - size(LA)|)}{\max(size(cities))}$$
- Similarity matrix

Similarity

Nominal variable

- Examples
  - {Mon., Tue., Wed., Thu., Fri., Sat., Sun.}
  - {Boston, LA, New York, San Francisco, Seattle}
- Binary rule
  - if $x_i = y_i$ then $sim(x_i, y_i) = 1$, else $sim(x_i, y_i) = 0$
- Underlining semantic property:
  - $$sim(Boston, LA) = \alpha dist(Boston, LA)^{-1}$$
  - $$sim(Boston, LA) = 1 - \alpha \frac{(|size(Boston) - size(LA)|)}{\max(size(cities))}$$
- Similarity matrix

Similarity

Nominal variable

- Examples
  - {Mon., Tue., Wed., Thu., Fri., Sat., Sun.}
  - {Boston, LA, New York, San Francisco, Seattle}
- Binary rule
  - if $x_i = y_i$ then $sim(x_i, y_i) = 1$, else $sim(x_i, y_i) = 0$
- Underlining semantic property:
  - $$sim(Boston, LA) = \alpha dist(Boston, LA)^{-1}$$
  - $$sim(Boston, LA) = 1 - \alpha \frac{(|size(Boston) - size(LA)|)}{\max(size(cities))}$$
- Similarity matrix

Similarity

Similarity matrix

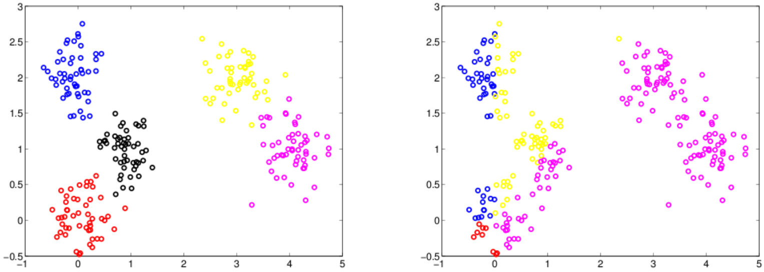
	tiny	little	small	medium	large	huge
tiny	1.0	0.8	0.7	0.5	0.2	0.0
little		1.0	0.9	0.7	0.3	0.1
small			1.0	0.7	0.3	0.2
medium				1.0	0.5	0.3
large					1.0	0.8
huge						1.0

- Diagonal must be 1.0
- No linearity (value interpolation) assumed
- Qualitative Transitive property must hold

Similarity
Ordinal variable

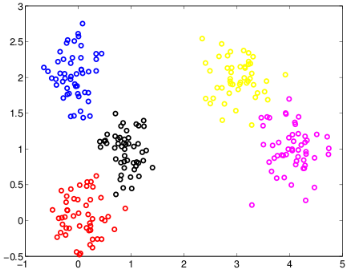
- Convert to real variable on a normalized [0,1] scale
  - max(v)=1, min(v)=0, others interpolate
  - e.g. “small”=0, “medium”=0.33, “large”=0.66, “x-large”=1
- Then use similarity measures for real variable
- Or use similarity matrix

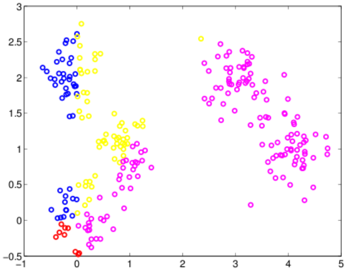
What are good clusters?



- the intra-cluster distance is small
- the inter-cluster distance is large

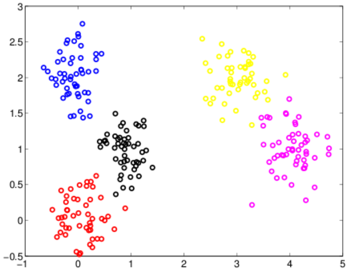
What are good clusters?

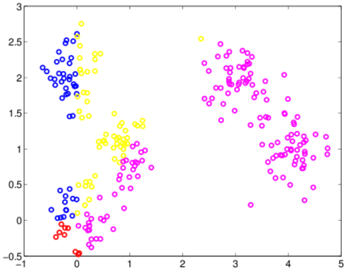




- the intra-cluster distance is small
- the inter-cluster distance is large

What are good clusters?

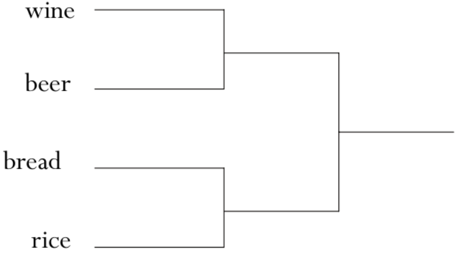




- the intra-cluster distance is small
- the inter-cluster distance is large

Outline

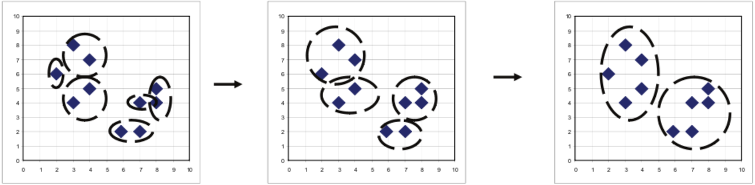
Hierarchical clustering



- Builds a cluster hierarchy, i.e. a tree of clusters
  - Also known as a dendrogram
  - Every cluster node contains child clusters
  - Sibling clusters partition the points covered by their common parent
- Exploring data on different levels of granularity

Hierarchical clustering

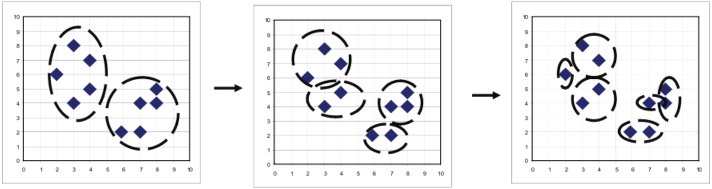
Bottom-up (agglomerative) and Top-down (divisive)



- Bottom up
  - Starts with one-point (singleton) clusters and recursively merges two or more most appropriate clusters.
- Top down
  - Starts with one cluster of all data points and recursively splits the most appropriate cluster.
- The process continues until a stopping criterion is achieved
  - e.g. the requested number k of clusters

Hierarchical clustering

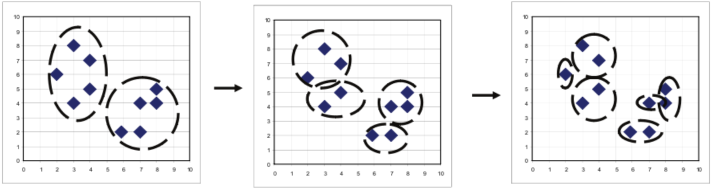
Bottom-up (agglomerative) and Top-down (divisive)



- Bottom up
  - Starts with one-point (singleton) clusters and recursively merges two or more most appropriate clusters.
- Top down
  - Starts with one cluster of all data points and recursively splits the most appropriate cluster.
- The process continues until a stopping criterion is achieved
  - e.g. the requested number k of clusters

Hierarchical clustering

Bottom-up (agglomerative) and Top-down (divisive)



- Bottom up
  - Starts with one-point (singleton) clusters and recursively merges two or more most appropriate clusters.
- Top down
  - Starts with one cluster of all data points and recursively splits the most appropriate cluster.
- The process continues until a stopping criterion is achieved
  - e.g. the requested number k of clusters

Hierarchical Agglomerative Clustering

Algorithm

1. Start with all instances in their own cluster
2. Until there is only one cluster
    1. Among the current clusters, determine two clusters, c_i and c_j, which are most similar
    2. Replace c_i and c_j with a single cluster $c_i \cup c_j$

Hierarchical Agglomerative Clustering
Algorithm

1. Start with all instances in their own cluster
2. Until there is only one cluster
    1. Among the current clusters, determine two clusters, c_i and c_j which are most similar
    2. Replace c_i and c_j with a single cluster $c_i \cup c_j$

Hierarchical Agglomerative Clustering

Algorithm

1. Start with all instances in their own cluster
2. Until there is only one cluster
    1. Among the current clusters, determine two clusters, c_i and c_j which are most similar
    2. Replace c_i and c_j with a single cluster $c_i \cup c_j$

Hierarchical Agglomerative Clustering
Algorithm

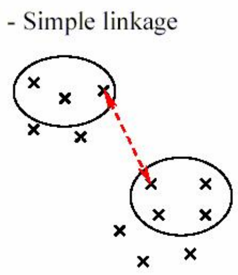
1. Start with all instances in their own cluster
2. Until there is only one cluster
   1. Among the current clusters, determine two clusters, c_i and c_j which are most similar
   2. Replace c_i and c_j with a single cluster $c_i \cup c_j$

Hierarchical Agglomerative Clustering

Cluster Similarity

- Single Linkage
  - Similarity of two most similar members of each cluster

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$



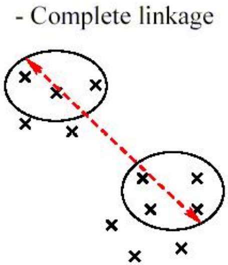
$$sim(c_i \cup c_j, c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

Hierarchical Agglomerative Clustering

Cluster Similarity

- Complete Linkage
  - Similarity of two least similar members of each cluster

$$sim\left(c_i, c_j\right) = \min_{x \in c_i, y \in c_j} sim\left(x, y\right)$$



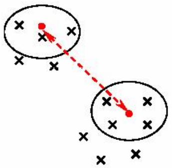
$$sim\left(c_i \cup c_j, c_k\right) = \min\left(sim\left(c_i, c_k\right), sim\left(c_j, c_k\right)\right)$$

Hierarchical Agglomerative Clustering

Cluster Similarity

- Average Linkage
  - Mean similarity between members of each cluster

$$sim\left(c_i, c_j\right) = \frac{1}{|c_i||c_j|} \sum_{x \in c_i, y \in c_j} \sum sim\left(x, y\right)$$



$$sim\left(c_i \cup c_j, c_k\right) = \frac{\left(|c_i| sim\left(c_i, c_k\right) + |c_j| sim\left(c_j, c_k\right)\right)}{\left(|c_i| + |c_j|\right)}$$

Hierarchical Agglomerative Clustering

Cluster Similarity

- Lance-Williams Formula

$$\begin{aligned} sim(c_i \cup c_j, c_k) &= \alpha_i sim(c_i, c_k) + \alpha_j sim(c_j, c_k) \\ &+\beta sim(c_i, c_j) + \gamma |sim(c_i, c_k) - sim(c_j, c_k)| \end{aligned}$$

similarity	α_i	α_j	β	γ
single linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
complete linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
average linkage	$\frac{\|c_i\|}{\|c_i\|+\|c_j\|}$	$\frac{\|c_j\|}{\|c_i\|+\|c_j\|}$	0	0

Hierarchical Agglomerative Clustering

Example: a hierarchical clustering of some cities

- Euclidean distance
- Single linkage

Hierarchical Agglomerative Clustering

Example: a hierarchical clustering of some Italian cities

Distance matrix

	BA	FI	MI	NA	RM	TO
BA	0	662	877	255	412	996
FI	662	0	295	468	268	400
MI	877	295	0	754	564	138
NA	255	468	754	0	219	869
RM	412	268	564	219	0	669
TO	996	400	138	869	669	0

Hierarchical Agglomerative Clustering

Example: a hierarchical clustering of some Italian cities

Distance matrix

	BA	FI	MI/TO	NA	RM
BA	0	662	877	255	412
FI	662	0	295	468	268
MI/TO	877	295	0	754	564
NA	255	468	754	0	219
RM	412	268	564	219	0

Hierarchical Agglomerative Clustering

Example: a hierarchical clustering of some Italian cities

Distance matrix

	BA	FI	MI/TO	NA/RM
BA	0	662	877	255
FI	662	0	295	268
MI/TO	877	295	0	564
NA/RM	255	268	564	0

Hierarchical Agglomerative Clustering

Example: a hierarchical clustering of some Italian cities

Distance matrix

	BA/NA/RM	FI	MI/TO
BA/NA/RM	0	268	564
FI	268	0	295
MI/TO	564	295	0

Hierarchical Agglomerative Clustering

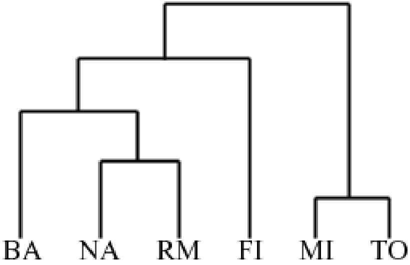
Example: a hierarchical clustering of some Italian cities

Distance matrix

	BA/NA/RM/FI	MI/TO
BA/NA/RM/FI	0	295
MI/TO	295	0

Hierarchical Agglomerative Clustering

Example: a hierarchical clustering of some Italian cities



Hierarchical clustering

Advantages and disadvantages

- Advantages
  - Embedded flexibility regarding the level of granularity
  - Ease of handling of any forms of similarity or distance
  - Consequently, applicability to any attribute types
- Disadvantages
  - Vagueness of termination criteria
  - Most hierarchical algorithms do not revisit once constructed (intermediate) clusters with the purpose of their improvement

Outline

Partitional clustering

- Typically determine all clusters directly
- Partitioning Relocation
  - try to discover clusters by iteratively relocating points between subsets
- Density-Based Partitioning
  - try to discover dense connected components of data
  - A cluster, defined as a connected dense component, grows in any direction that density leads

K-means clustering

- Find K non overlapping clusters $C_1, C_2, \cdots, C_K$, so that
  - Each data point is assigned to a unique cluster
  - The total intra-cluster variance is minimized

$$\sum_{i=1} \sum_{x_j \in C_i} \| x_j - \mu_i \|^2$$

where

$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$$

is the centroid of cluster C_i

K-means clustering

1. Initialize cluster centroids $\mu_1, \mu_2, \cdots, \mu_K$ randomly
2. Repeat until convergence
   1. assign points to clusters whose centers are the closest

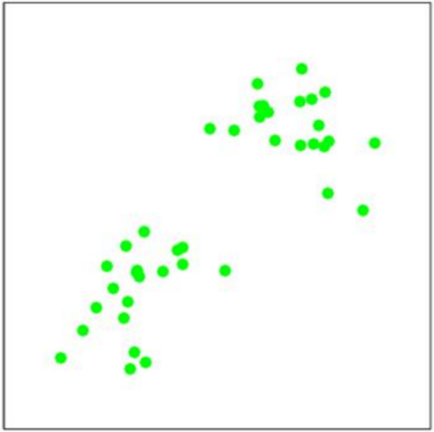
$$c_i := \arg\min_j \| x_i - \mu_j \|^2$$

   2. update cluster centers

$$\mu_i = \frac{1}{| C_i |} \sum_{x_j \in C_i} x_j$$

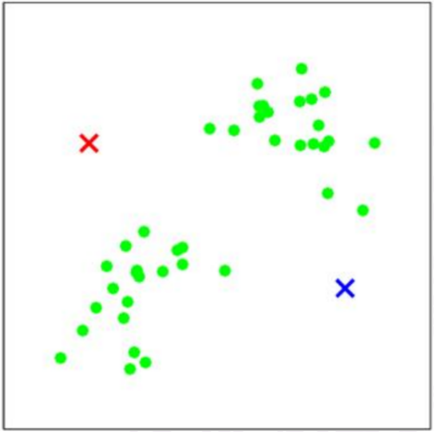
K-means clustering

Example



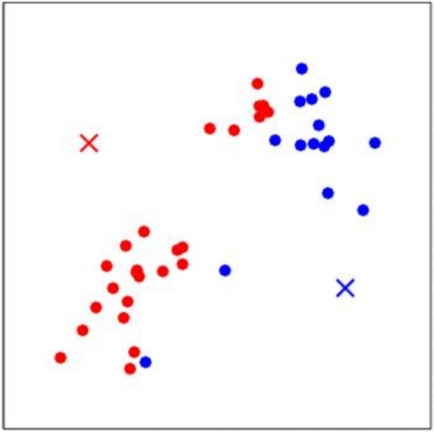
K-means clustering

Example



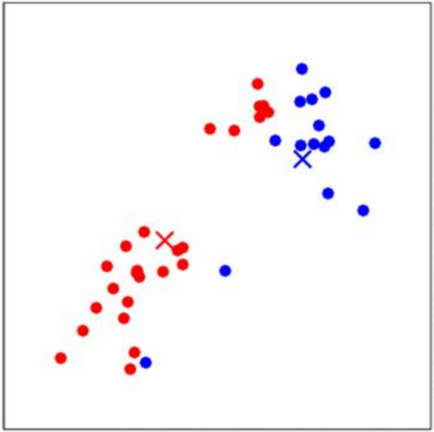
K-means clustering

Example



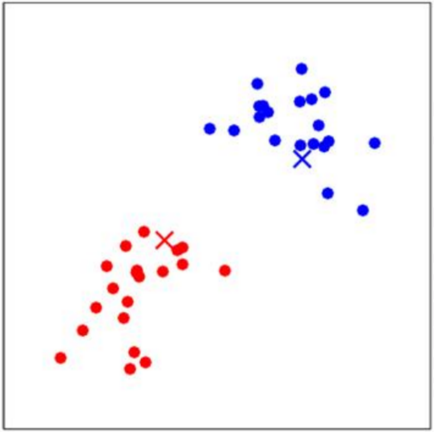
K-means clustering

Example



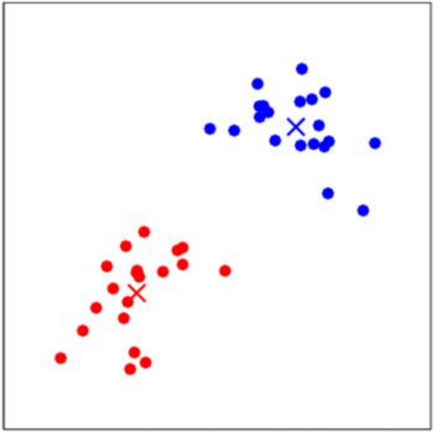
K-means clustering

Example



K-means clustering

Example



K-means clustering

Advantages

- Simplicity
- Converge in a finite number of iterations
- Works well when the clusters are compact clouds that are rather well separated from one another

K-means clustering

Disadvantages

- Significantly sensitive to the initial randomly selected cluster centres
  - Multiple runs
- Local optima
  - Multiple runs
- Very sensitive to noise and outlier points
- Not suitable for discovering clusters with non-convex shapes or clusters with quite different size
- Depend on the value of k

K-means clustering

How to decide K?

- Problem driven
  - The problem itself has the setting of K
- Data driven only when either
  - Data is not sparse
  - Measurement dimensions are not too noisy
- Examine the within cluster dissimilarity W_K
  - a function of K
  - Usually W_K decreases with increasing K
  - a sharp drop at the optimal number of cluster K^*

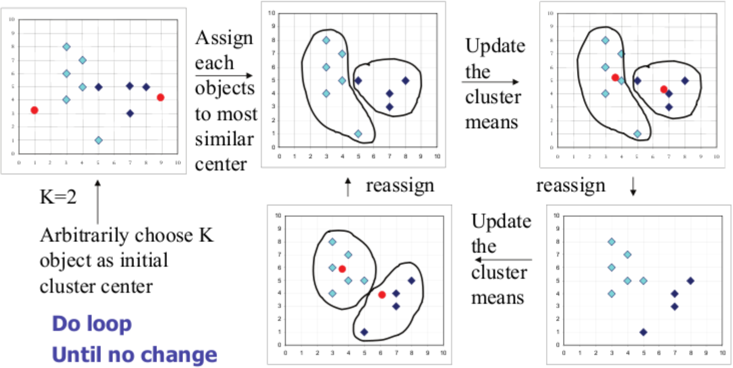
K-medoids clustering

- A reference point of a cluster
  - The medoid – most centrally located object in a cluster
  - Instead of taking the mean value of the objects
- Minimize squared error (the average dissimilarity)
  - the distance between points labeled to be in a cluster and medoid
- Advantages
  - works with an arbitrary matrix of distances between datapoints instead of l_2
  - robust to noise and outliers as compared to k-means

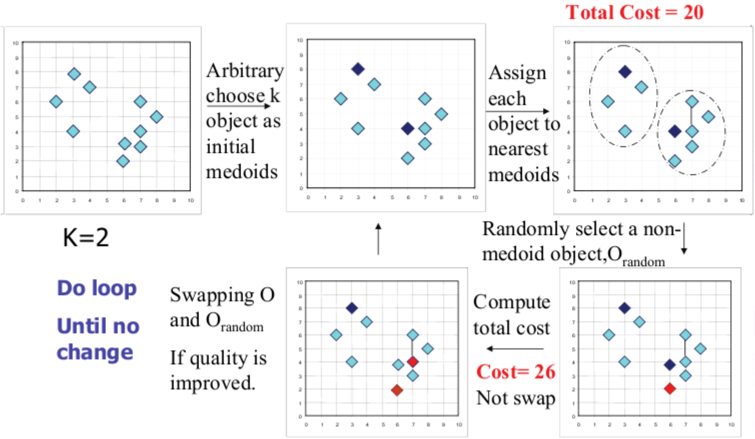
K-medoids clustering

1. Initialize cluster medoids $o_1, o_2, \cdots, o_K$ randomly
2. Repeat until no change in medoids
    1. assign points to clusters whose mediods are the closest
    $$c_k := \underset{j=1,\cdots K}{\arg\min} \parallel x_i - o_j \parallel^2$$
    2. Replace medoid in a cluster that is closest to the other data points in the cluster
    $$o_k := \min_{x_i \in C_k} \sum_{x_j \in C_k} \parallel x_i - x_j \parallel^2$$

K-means clustering (Recall)



K-mediods clustering



Outline

Dimension reduction

- Input data may have thousands of dimensions
- Represent data with fewer dimension
  - Easier learning
    - fewer parameters, accuracy
  - Visualization
    - Display high dimension data in 2-D display and use them for explanatory data analysis
  - discover "intrinsic dimensionality" of data high dimensional data that is truly lower dimensional

Outline

Principle component analysis

- PCA
- Also known as
  - Karhunen-Loeve transform (KLT)
  - Hotelling transform
  - Proper orthogonal decomposition (POD)

- 1901, Invented by Karl Pearson (also Carl Pearson, 1857-1936)
- Transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components (PCs)

Principle component analysis

- Given data points in high dimensional space, project into low dimensional space while preserving as much information as possible
- In particular, choose projection that minimizes the squared error in reconstructing original data

Principle component analysis

Find Projections to Minimize Reconstruction Error

- Assume data is a set of N-dimension, $\left\{x_1, x_2, \cdots, x_M, x_i \in R^N\right\}$
- Represent data with $L \leq N$ orthogonal basis vectors,

$$\hat{x}_i = \bar{x} + \sum_{j=1}^{L} z_{ij} u_j$$

$$\bar{x} = \frac{1}{M} \sum_i x_i$$

$$u_i^T u_j = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

$$u_1, u_2, \cdots, u_L, u_i \in R^N$$

where z_{ij} is the coordinates in low dimension space

Principle component analysis

Find Projections to Minimize Reconstruction Error

- Try to find orthogonal basis vectores to minimize reconstruction error

$$E_L^* = \min_{u_1, u_2, \cdots, u_L} \frac{1}{M} \sum_i \|x_i - \hat{x}_i\|^2$$

$$\begin{aligned} E_L &= \frac{1}{M} \sum_{i=1}^{M} \sum_{j=L+1}^{N} \left[u_j^T (x_i - \bar{x}) \right]^2 \\ &= \sum_{j=L+1}^{N} u_j^T \Sigma u_j \end{aligned}$$

where Σ is covariance matrix

$$\Sigma = \frac{1}{M} \sum_i (x_i - \bar{x})(x_i - \bar{x})^T$$

Principle component analysis

Find Projections to Minimize Reconstruction Error

- E_L is minimized when u_j is eigenvector of Σ, i.e.

$$\Sigma u_j = \lambda_j u_j$$

  where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$, are eigenvalues

- Minimum error $E_L^* = \sum_{j=L+1}^{N} \lambda_j$
- Best coordinates in lower dimensional space defined by dot-products

$$(z_1, z_2, \cdots, z_L)$$
$$z_j = (x - \bar{x}) \bullet u_j$$

Principle component analysis

Algorithm 1

1. Calculate covariance matrix Σ
2. Find eigenvectors and eigenvalues of Σ
3. PCs are L eigenvectors with the largest eigenvalues

Principle component analysis

Algorithm 1

1. Calculate covariance matrix Σ
2. Find eigenvectors and eigenvalues of Σ
3. PCs are L eigenvectors with the largest eigenvalues

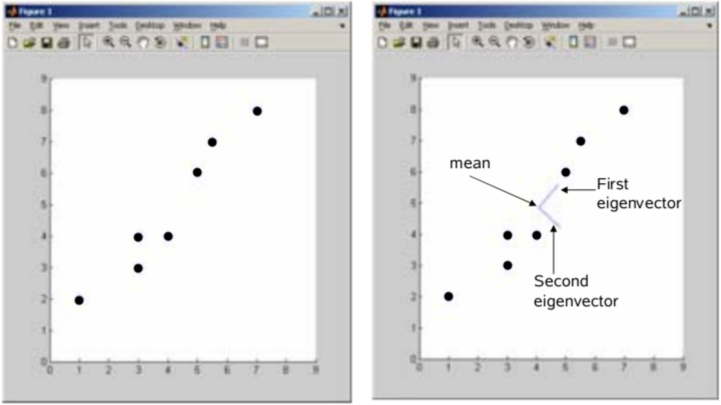
Principle component analysis

Algorithm 1

1. Calculate covariance matrix Σ
2. Find eigenvectors and eigenvalues of Σ
3. PCs are L eigenvectors with the largest eigenvalues

Principle component analysis

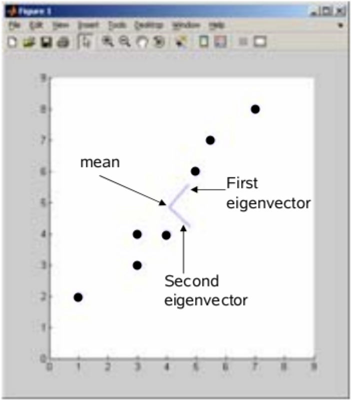
Example

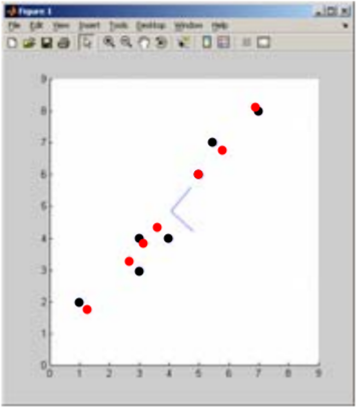


Principle component analysis

Example

- Only use the first principle component





PCA and K-means

Theorem

[Ding, 2004] For K-means clustering where $K = 2$, the continuous solution of the cluster indicator vector is the principle component v_1, i.e., C_1 and C_2 are given by

$$C_1 = \{i | v_1 \leq 0\}, C_2 = \{i | v_1 > 0\}$$

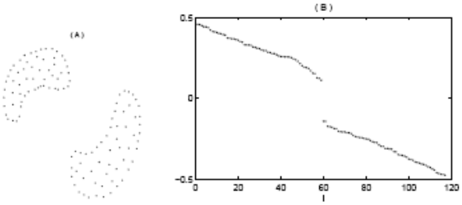
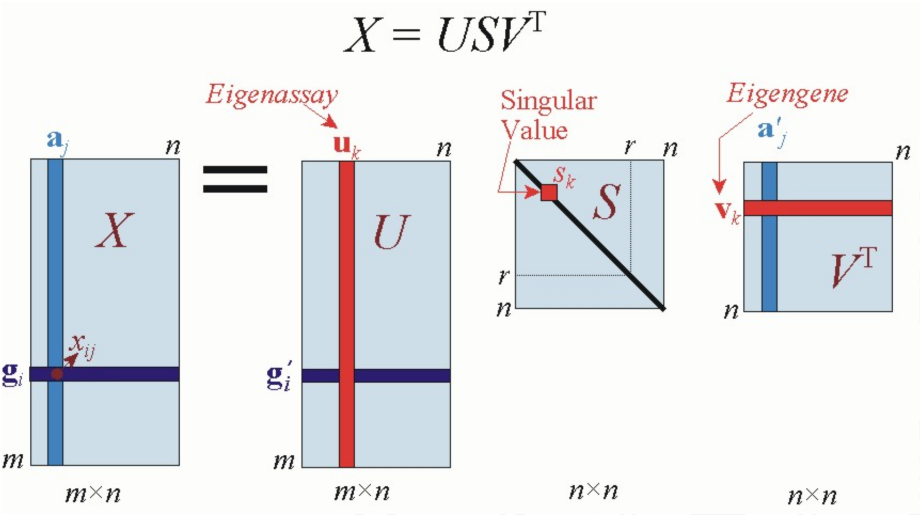


Figure 1. (A) Two clusters in 2D space. (B) Principal component $v_1(i)$, showing the value of each element i.

Very Nice, but...

- Covariance matrix: *NxN*
- What if very large dimensional data?
  - Images ($N \geq 10^4$)
    - finding eigenvectors is very slow
    - no enough memory
  - Use singular value decomposition (SVD)

Singular Value Decomposition



Singular Value Decomposition

- Applications
  - Pseudoinverse
  - Least squares fitting of data
  - Matrix approximation
  - Determining the rank, range and null space of a matrix
  - etc.
- Implementations
  - LAPACK
  - GNU Scientific Library

Principle component analysis

Algorithm with SVD

1. Start from MxN data matrix X, each row is a sample
2. Recenter: substract mean from each row of X

$$X_c = X - \bar{X}$$

3. Call SVD algorithm on X_c – ask for L singular vectors
4. Principal components: L singular vectors with highest singular values (rows of V^T)
5. Project a column vector x into PC coordinates, take the first L coordinates of

$$V^T x$$

Outline

Mercer's theorem

- Any continuous, symmetric, positive semi-definite kernel function $K(x,y)$ can be expressed as a dot product in a high-dimensional space
- Examples
  - Polynomials Kernel
    $$K(x,y) = (< x, y >)^d$$
  - Gauss Kernel
    $$K(x,y) = \exp\left(-\frac{||x-y||^2}{2\sigma^2}\right)$$

Kernels Trick

- The kernel trick transforms any algorithm that solely depends on the dot product between two vectors
- Dot product is replaced with the kernel function
- A linear algorithm can easily be transformed into a non-linear algorithm
- This non-linear algorithm is equivalent to the linear algorithm operating in the range space of ϕ
- Because kernels are used, the ϕ function is never explicitly computed

Kernel PCA

- Covariance matrix

$$C = \frac{1}{N}\sum_i \Phi(x_i)\,\Phi(x_i)^T$$

- Solve eigenvalue equation

$$\lambda v = Cv$$

$$v = \sum_i \alpha_i \Phi(x_i)$$

$$N\lambda\alpha = K\alpha$$

$$K_{ij} =< \Phi(x_i), \Phi(x_j) >$$

Kernel PCA

- Covariance matrix

$$C = \frac{1}{N}\sum_{i}\Phi\left(x_i\right)\Phi\left(x_i\right)^T$$

- Solve eigenvalue equation

$$\lambda v = Cv$$

$$v = \sum_{i}\alpha_i\Phi\left(x_i\right)$$

$$N\lambda\alpha = K\alpha$$

$$K_{ij} = <\Phi\left(x_i\right), \Phi\left(x_j\right)>$$

Kernel PCA

- Select largest L eigenvalue and corresponding eigenvector of Kernel Matrix K

$$\lambda_1 \geq \cdots \geq \lambda_L$$

$$\alpha^1, \cdots, \alpha^L$$

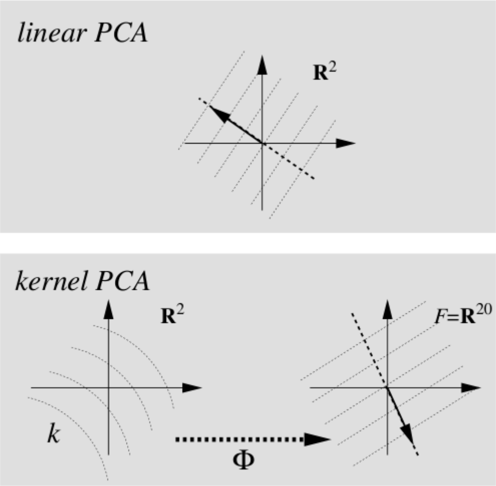
- Largest L eigenvector of Covariance matrix C

$$v^l = \sum_i \alpha_i^l \Phi(x_i)$$

- Projection

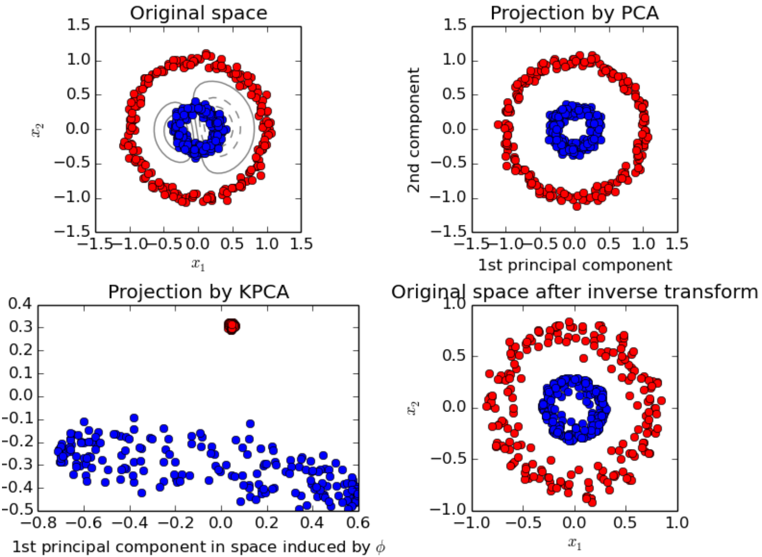
$$< v^l, \Phi(x) >= \sum_i \alpha_i^l < \Phi(x_i), \Phi(x) >$$

Kernel PCA



The dotted lines are contour lines of constant feature value

Kernel PCA



Original space
Projection by PCA
Projection by KPCA
Original space after inverse transform
x_1
x_2
2nd component
1st principal component
1st principal component in space induced by ϕ

Question

How to make data in feature space to be centered without explicit mapping?

i.e., How to "centrize" K?

Summary

- Unsupervised learning
- Hierarchical agglomerative clustering
- K-means clustering
- Principle components analysis
- Kernel PCA