Introduction to Big Data Analysis
Clustering Analysis

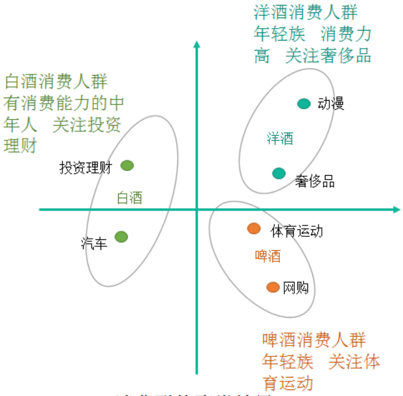
Zhen Zhang

Southern University of Science and Technology

Outlines

Clustering

- Also called data segmentation, group a collection of objects into subsets or “clusters”
- Results : objects in each cluster are more similar to one another than objects in different clusters.
- Example : applications in consumption analysis
- Can be used in data preprocessing



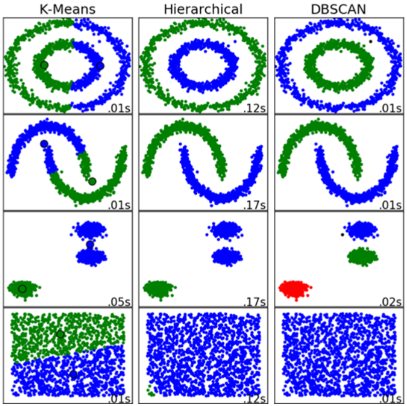
消费群体聚类结果

Concepts in Clustering

- Different from classification : it is unsupervised learning ; no outputs or labels
- Central goal : Optimize the similarity (or dissimilarity) between the individual objects being clustered :
  - Obtain great similarity of samples within cluster
  - Obtain small similarity of samples between clusters
- Cost functions : not related to the outputs, but related to the similarity
- Two kinds of input data :
  - $n \times n$ similarity (dissimilarity) matrix D : only depends on the distances between pairs of samples ; may lose some information on data
  - Original data with features $X \in \mathbb{R}^{n \times d}$

Clustering Methods

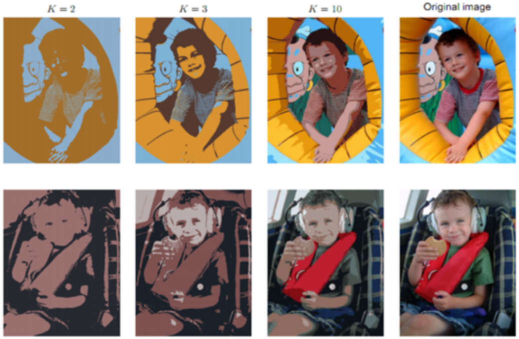
- Clustering process :
  - data preprocessing, especially standadization
  - Similarity matrix
  - Clustering Methods
  - Determine the best number of clusters
- Clustering methods :
  - Partitional clustering :
    - K-means
    - K-Medoids
    - Spectral clustering
    - DBSCAN
  - Hierarchical clustering



Outlines

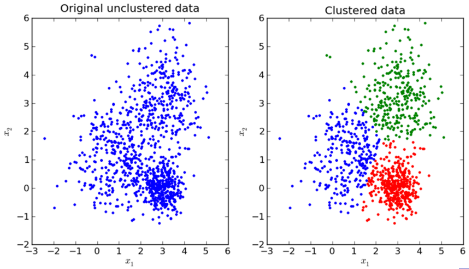
Introduction

- K-means clustering originates from signal processing, it is quite popular in image processing (segmentation)
- Group n samples to k clusters, making each sample belong to the nearest cluster
- In an image, each pixel is a sample



Idea

- Data set $\{\mathbf{x}_i\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$
- Representatives : Mass center of kth-cluster C_k is c_k, $k = 1, \ldots, K$
- Sample x_i belongs to cluster k if $d(x_i, c_k) < d(x_i, c_m)$ for $m \neq k$, where $d(x_i, x_j)$ is dissimilarity function
- Make the mass centers well-located so that the average distance between each sample to its cluster center is as small as possible



Optimization Problem

- Let $C : \{1, \ldots, n\} \to \{1, \ldots, k\}$ be the assignment from the data indices to the cluster indices. $C(i) = k$ means $x_i \in C_k$
- Total point scatter : $T = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} d(x_i, x_j) =$
  $\frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \left(\sum_{C(j)=k} d_{ij} + \sum_{C(j)\neq k} d_{ij} \right) = W(C) + B(C)$
- Loss function : within-cluster point scatter
  $W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(j)=k} d_{ij}$; between-cluster point scatter
  $B(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(j)\neq k} d_{ij}$
- Minimize $W(C)$ is equivalent to maximize $B(C)$

Dissimilarities

- Proximity matrices : $n \times n$ symmetric matrix D with nonnegative entries and zero diagonal elements provides information about dissimilarity between a pair of samples, this is not distance in general
- Dissimilarities based on attributes : $d(x_i, x_j) = \sum_{k=1}^{p} d_k(x_{ik}, x_{jk})$; d_k can be squared distance $d_k(x_{ik}, x_{jk}) = (x_{ik} - x_{jk})^2$, absolute distance $d_k(x_{ik}, x_{jk}) = |x_{ik} - x_{jk}|$
- Weighted average : $d(x_i, x_j) = \sum_{k=1}^{p} w_k d_k(x_{ik}, x_{jk})$ where $\sum_{k=1}^{p} w_k = 1$; setting $w_k \sim 1/\bar{d}_k$ with $\bar{d}_k = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_k(x_{ik}, x_{jk}) = 2\widehat{\mathrm{Var}}(X_k)$ will assign equal influence to all features
- Dissimilarities based on correlation : $d(x_i, x_j) \propto 1 - \rho(x_i, x_j)$

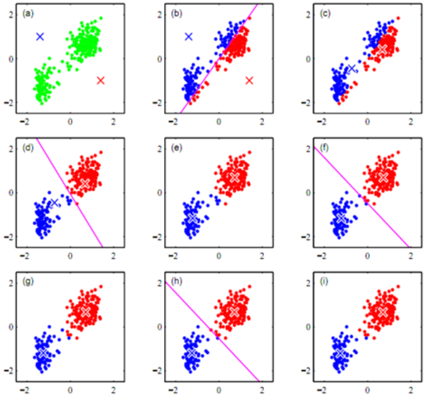
K-Means (as Central Voronoi Tessellation)

- Minimizing $W(C)$ is in general infeasible since this is a greedy algorithm that only works for small data sets
- Taking squared dissimilarity, $W(C) = \sum_{k=1}^{K} n_k \sum_{C(i)=k} \|x_i - \bar{x}_k\|^2$, where $n_k = \sum_{i=1}^{n} I(C(i) = k)$ is the number of samples in cluster k, $\bar{x}_k = \frac{1}{n_k} \sum_{C(j)=k} x_j = \arg\min_{m_k} \sum_{C(j)=k} \|x_j - m_k\|^2$
- $\min_{C} W(C) \iff \min_{C, m_k} \sum_{k=1}^{K} n_k \sum_{C(i)=k} \|x_i - m_k\|^2$
- Alternating minimization :
  1. Given C, solve for $m_k \implies m_k^* = \bar{x}_k$ (choose representatives)
  2. Given m_k, solve for $C \implies C(i) = \arg\min_{1 \leqslant k \leqslant K} \|x_i - m_k\|^2$

     (partitioning, equivalent to Voronoi tessellation for given center m_k)

K-Means Iterations

- The alternating iterations can stop when the mass centers $\{\bar{x}_k\}_{k=1}^{K}$ do not change
- Initial guess :
  - Random guess, try the best one with smallest $W(C)$
  - Base on other clustering methods (e.g., hierarchical clustering), choose the cluster centers as initial guess



How to choose K

- Minimizing Bayesian Information Criterion (BIC) : $BIC(\mathcal{M}|\mathbf{X}) = -2\log \Pr(\mathbf{X}|\hat{\Theta}, \mathcal{M}) + p\log(n)$, where $\mathcal{M}$ indicates the model, $\hat{\Theta}$ is the MLE of the model parameters in $\mathcal{M}$, $\Pr(\mathbf{X}|\mathcal{M})$ is the likelihood function, and p is the number of parameters in model $\mathcal{M}$; trade-off between log-likelihood and model complexity
- Based on Minimum Description Length (MDL) : starting from large K, decreases K until the description length $-\log \Pr(\mathbf{X}|\hat{\Theta}, \mathcal{M}) - \log \Pr(\Theta|\mathcal{M})$ achieves its minimum (similar to MAP)
- Based on Gaussian distribution assumption : starting from $K = 1$, increases K until the points in every cluster follow Gaussian distribution

Pros and Cons

- Where it is good
  - Intuitive, easy to implement
  - Low computational complexity, $O(tnpK)$, where t is the number of iterations
- Disadvantage
  - Need to specify K first (K is tuning parameter)
  - Strong dependence on the initial guess of cluster center
  - Easy to stuck at local minimum
  - Naturally assume ball-shaped data, hard to deal with data which are not ball-shaped
  - Sensitive to outliers

Variant : Bisecting K-means

- Invented to deal with initial guess of center selection
- Idea : sequentially divide the poorest cluster into two sub-clusters
  1. Initially gather all data into one cluster
  2. Repeat :
     2.1 Select the cluster k that maximizes the within-cluster point scatter $\sum_{C(i)=k} \sum_{C(j)=k} \|x_i - x_j\|^2$
     2.2 Use 2-means to divide cluster k into two sub-clusters, with random initial guess of two centers
     2.3 Repeat step 2.2 p times, choose the best pair of clusters that minimizes the within-cluster point scatter
  3. Stop when there are K clusters (Or you can stop any time you like to have a satisfactory clustering result)

Variant : K-medoids

- Invented to overcome the influence of outliers
- Can deal with data of general type, assuming general dissimilarity $d(x_i, x_j)$
- Idea : centers for each cluster are restricted to be one of the observations assigned to that cluster
- Alternating minimization :
  1. Given C, solve for $m_k = x_{i_k^*}$ that minimizes the within-cluster point scatter : $i_k^* = \arg \min_{\{i: C(i)=k\}} \sum_{C(j)=k} d(x_i, x_j)$ (choose the real samples as representatives)
  2. Given m_k, solve for $C \implies C(i) = \arg \min_{1 \leqslant k \leqslant K} d(x_i, m_k)$
- More robust than K-means
- More computational effort when solving for the center in step 1 : $O(n_k^2)$ comparing to $O(n_k)$ in K-means

Other Variants

- K-medians : use Manhattan distance (L^1-distance) instead in K-means ; then the centers are not means, but medians
- K-means++ : designed to select good initial centers that are far away from each other
- Rough-set-based K-means : each sample could be assigned to more than one cluster

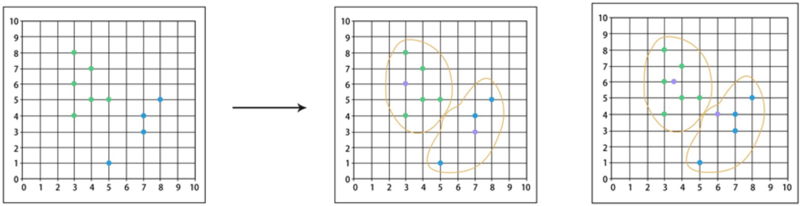


FIGURE: K-medoids

Outlines

Hierarchical Clustering

- Clustering in different hierarchies, generating tree structure
- Two approaches :
    - Agglomerate clustering : bottom-up
    - Divisive clustering : top-down
- Limitation : once merged or divided, the operation cannot be modified

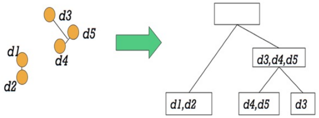


FIGURE: Agglomerate clustering

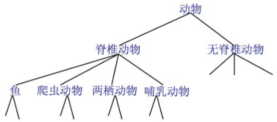


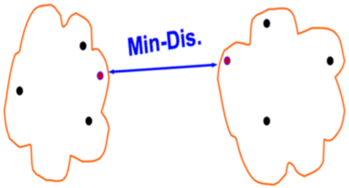
FIGURE: Divisive clustering

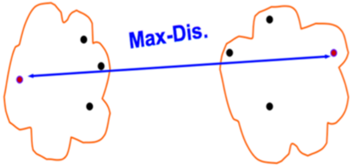
Agglomerate Clustering

- Given n samples and proximity matrix, do the following steps :
  1. Let every observation represent a singleton cluster
  2. Merge the two closest clusters into one single cluster
  3. Calculate the new proximity matrix (dissimilarity between two clusters)
  4. Repeat step 2 and 3, until all samples are merged into one cluster
- Three methods for computing intergroup dissimilarity :
  - Single linkage (SL)
  - Complete linkage (CL)
  - Average linkage (AL)

Intergroup Dissimilarity

- Single linkage : Greatest similarity or least dissimilarity $d_{SL}(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$
- Complete linkage : Least similarity or greatest dissimilarity $d_{SL}(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$
- Average linkage : Average similarity or dissimilarity $d_{AL}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$





Generalized Agglomerative Scheme

- Input : training set $D = \{(x_1), \ldots, (x_n)\}$, dissimilarity function $d(C_i, C_j)$
- Output : A dendrogram containing $\{\mathcal{R}_t\}_{t=0}^{n-1}$, where $\mathcal{R}_t$ is the clustering result at time t

1. Initialize the clustering result $\mathcal{R}_0 = \{\{x_1\}, \{x_2\}, \ldots, \{x_n\}\}$, $t = 0$
2. Do iterations :
   2.1 $t = t + 1$
   2.2 Choose (C_i, C_j) from $\mathcal{R}_{t-1}$ so that $d(C_i, C_j) = \min\limits_{(r,s)} d(C_r, C_s)$
   2.3 $C_q = C_i \bigcup C_j$
   2.4 $\mathcal{R}_t = (\mathcal{R}_{t-1} \setminus \{C_i, C_j\} \bigcup \{C_q\})$
3. Stop at $t = n - 1$ when $|\mathcal{R}_{n-1}| = 1$, return $\{\mathcal{R}_t\}_{t=0}^{n-1}$

Generalized Divisive Scheme

- Input : training set $D = \{(x_1), \ldots, (x_n)\}$, dissimilarity function $d(C_i, C_j)$
- Output : A dendrogram containing $\{\mathcal{R}_t\}_{t=0}^{n-1}$, where $\mathcal{R}_t = \{C_{t,i}\}_{i=1}^{t+1}$ is the clustering result at time t

```
1. Initialize $\mathcal{R}_0 = \{X\}$, $t = 0$
2. Do iterations :
    2.1 $t = t + 1$
    2.2 For $i = 1$ to $t$, do :
        2.2.1 Choose $(C^1_{t-1,i}, C^2_{t-1,i})$ from $C_{t-1,i}$ so that
              $d(C^1_{t-1,i}, C^2_{t-1,i}) = \max_{G \bigcup H = C_{t-1,i}} d(G, H)$
    2.3 Choose $i_{t-1}$ so that $i_{t-1} = \arg\max_i d(C^1_{t-1,i}, C^2_{t-1,i})$
    2.4 $\mathcal{R}_t = (\mathcal{R}_{t-1} \setminus \{C_{t-1,i_{t-1}}\}) \bigcup \{C^1_{t-1,i}, C^2_{t-1,i}\})$
3. Stop at $t = n - 1$ when $|\mathcal{R}_{n-1}| = n$, return $\{\mathcal{R}_t\}_{t=0}^{n-1}$
```

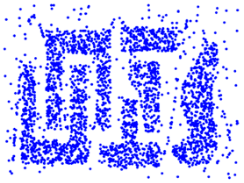
Pros and Cons

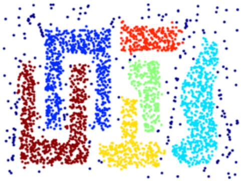
- Where it is good
  - Hierarchical clustering computes tree structure of the whole clustering process in one stroke
  - SL and CL are sensitive to outliers, while AL gives a compromise
    - As $n \to \infty$, $d_{AL}(C_i, C_j) \to \int\int d(x, y)p_i(x)p_j(y)\mathrm{d}x\mathrm{d}y$, the expected dissimilarity w.r.t. the two densities $p_i(x)$ and $p_j(x)$
    - In contrast, $d_{SL}(C_i, C_j) \to 0$ and $d_{CL}(C_i, C_j) \to \infty$ independent of $p_i(x)$ and $p_j(x)$
- Disadvantage
  - Computationally intensive
  - Once a sample is incorrectly grouped into a branch, it will stay in the clusters corresponding to that branch no matter how you threshold the tree

Outlines

Density-based Clustering

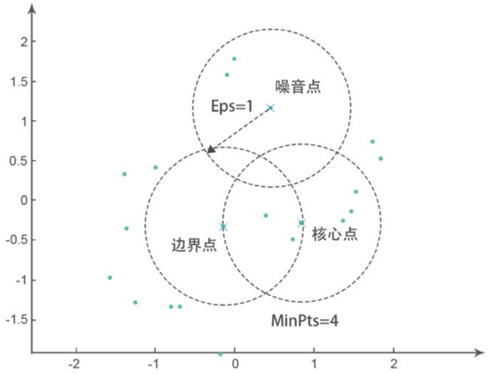
- Limitations of hierarchical clustering and K-means clustering : tend to discover convex clusters
- Density-based Clustering : looks for high-density regions separated by low-density regions, could discover clusters of any shape
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)


Original Points


Clusters

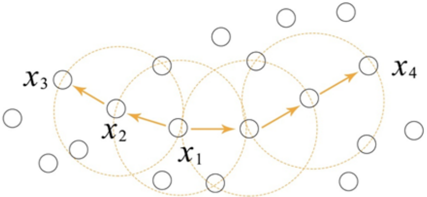
Concepts

- Three types of points :
  - Core point : # of samples in its ϵ-neighborhood $\geqslant$ MinPts
  - Boundary point : it lies in the ϵ-neighborhood of some core point, # of samples in its ϵ-neighborhood $<$ MinPts
  - Noise point : neither core point nor boundary point, it lies in the sparse region



Concepts

- ϵ-neighborhood : for each sample $x_i \in D$, $N_\epsilon(x_i) = \{x_j \in D | d(x_i, x_j) \leqslant \epsilon\}$
- Directly density-reachable : if the sample $x_j \in N_\epsilon(x_i)$, and x_i is core point, then x_j is directly density-reachable from x_i
- Density-reachable : for x_i and x_j, if there exist $p_1, \dots, p_m$, s.t. $p_1 = x_i, p_m = x_j$, and p_{k+1} is directly density-reachable from p_k, then x_j is density-reachable from x_j
- Density-connected : if there exists p, s.t. both x_i and x_j are density-reachable from p, then x_i and x_j are density-connected

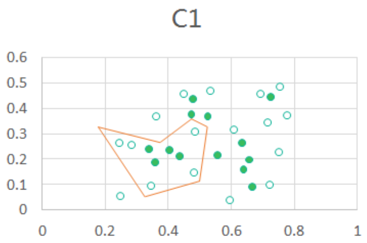


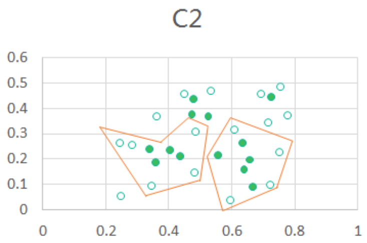
DBSCAN Algorithm

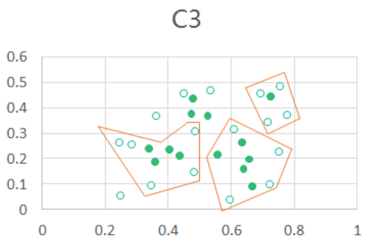
- Input : training set $D = \{(x_1), \ldots, (x_n)\}$, dissimilarity function $d(C_i, C_j)$, parameters $MinPts, \epsilon$
- Output : a set of clusters $\{C_t\}$

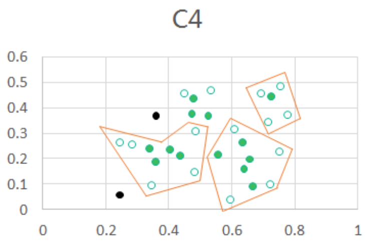
```
1. Mark all samples in $D$ as non-processed
2. For each sample $p \in D$, do :
   2.1 If $p$ has been grouped into some cluster or marked as noise point, go to check next sample
   2.2 Else, if $|N_\epsilon(p)| < MinPts$, then mark $p$ as boundary point or noise point
   2.3 Else, mark $p$ as core point, construct cluster $C = N_\epsilon(p)$. For each $q \in N_\epsilon(p)$, do :
       2.3.1 If $|N_\epsilon(q)| \geqslant MinPts$, then put all un-clustered points in $N_\epsilon(q)$ into $C$
3. Stop when all samples in $D$ have been clustered
```

Examples ($\epsilon = 0.11$, $MinPts = 5$)









DBSCAN vs. K-means

DBSCAN

- The clustering result is not a complete partition of original dataset (noise points are excluded)
- Could deal with clusters with any shape and size
- Could deal with noise points and outliers
- The definition of density must be meaningful
- Not efficient when dealing with high-dimensional data
- No implicit assumptions on the sample distribution

K-Means

- The clustering result is a complete partition of original dataset
- The clusters are nearly ball-shaped
- Sensitive to outliers
- The definition of cluster centers must be meaningful
- Efficient to deal with high-dimensional data
- The samples implicitly follow the Gaussian distribution assumption

Pros and Cons

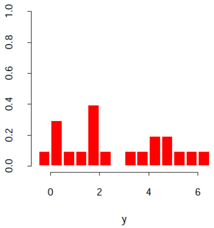
- Computational complexity $O(n \times T)$, where t is the time for searching ϵ-neighborhood) ; in the worst case, $O(n^2)$
- In low-dimensional space, could be improved as $O(n \log n)$ by KD-tree
- Where it is good
  - Fast for clustering
  - Better to deal with noise points
  - Effective for clusters of any shape
- Disadvantage
  - Need large memory
  - Bad performance when the density is not well-distributed and the between-cluster distances are large

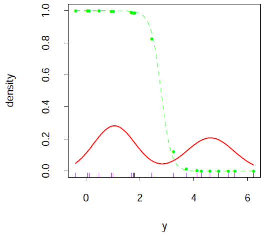
Outlines

Gaussian Mixture Models

- We want to estimate the density of given data set. This is an unsupervised learning problem.
- Commonly used approach is the parametric estimation, such as maximum likelihood estimate (MLE).
- Consider the following set of data points :

-0.39	0.12	0.94	1.67	1.76	2.44	3.72	4.28	4.92	5.53
0.06	0.48	1.01	1.68	1.80	3.25	4.12	4.60	5.28	6.22





Latent Variables

- A single Gaussian family would not be appropriate. A mixture of two Gaussian distributions seems good.
$$Z_1 \sim N(\mu_1, \sigma_1^2), \quad Z_2 \sim N(\mu_2, \sigma_2^2), \quad Z = (1-Y)Z_1 + YZ_2,$$
where $Y \in \{0, 1\}$ with $P(Y = 1) = c$.
- In general, for mixture of K Gaussian distributions, we assume there is a latent variable Y indicating which distribution the data $\mathbf{x}$ is sampled from, i.e., $P(Y = y) = c_y$ with $y \in \{1, \dots, K\}$. Given $Y = y$, the random variable X follows the conditional distribution :
$$P(X = \mathbf{x}|Y = y) = \frac{1}{(2\pi)^{d/2}(\Sigma_y)^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_y)^T\Sigma_y^{-1}(\mathbf{x}-\mu_y)\right).$$
- The density of X is then
$$\begin{aligned} P(X = \mathbf{x}) &= \sum_{y=1}^{K} P(Y = y)P(X = \mathbf{x}|Y = y) \\ &= \sum_{y=1}^{K} c_y \frac{1}{(2\pi)^{d/2}(\Sigma_y)^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_y)^T\Sigma_y^{-1}(\mathbf{x}-\mu_y)\right). \end{aligned}$$

MLE of Gaussian Mixture

- Let $\theta = (c_y, \mu_y, \Sigma_y)_{y=1}^K$. Then the log-likelihood of the sample set $S = \{\mathbf{x}_i\}_{i=1}^n$ is

$$L(\theta) = \sum_{i=1}^n \log P_\theta(X = \mathbf{x}_i) = \sum_{i=1}^n \log \Big(\sum_{y=1}^K P_\theta(X = \mathbf{x}_i, Y = y) \Big).$$

- MLE : $\theta = \arg\max_\theta L(\theta)$ is hard due to the summation inside the log.
- Make a simple assumption : we have known the value of the latent variable for each sample $\mathbf{x}_i$, i.e., $Y_i \in \{1, \ldots, K\}$ is known, then the choice from the Y_i-th Gaussian becomes deterministic, and the log-likelihood is replaced by

$$\begin{aligned}\tilde{L}(\theta) &= \sum_{i=1}^n \log \Big(\sum_{y=1}^K I_{(Y_i=y)} P_\theta(X = \mathbf{x}_i, Y = y) \Big) \\ &= \sum_{i=1}^n \sum_{y=1}^K I_{(Y_i=y)} \log P_\theta(X = \mathbf{x}_i, Y = y).\end{aligned}$$

Expectation-Maximization (EM) Algorithm (Dempster, Laird, and Rubin, 1977')

- But Y_i is indeed random, so that the event $Y_i = y$ happens with a probability $Q_{i,y}$ with $\sum_{y=1}^{k} Q_{i,y} = 1$.
- Consider the modified objective function defined over $Q = (Q_{i,y})_{i=1,\dots,n;y=1,\dots,K}$ and θ

$$F(Q,\theta) = \sum_{i=1}^{n}\sum_{y=1}^{K} Q_{i,y} \log\Big(P_\theta(X = \mathbf{x}_i, Y = y)\Big).$$

- The optimization problem $(Q,\theta) = \arg\max_{Q,\theta} F(Q,\theta)$ can be solved alternatingly :
  1. E-Step : Given $\theta^{(m)}$, solve for $Q^{(m+1)} = E_{\theta^{(m)}}(I_{(Y=y)}|X = \mathbf{x}_i) = P_{\theta^{(m)}}(Y = y|X = \mathbf{x}_i)$;
  2. M-Step : Given $Q^{(m+1)}$, solve for $\theta^{(m+1)} = \arg\max_{\theta} F(Q^{(m+1)},\theta)$ (assume this is tractable).
- Initial values of $Q^{(0)}$ and $\theta^{(0)}$ are chosen randomly.
- Terminate until satisfactory (not always converge to the maximum, but guaranteed convergent).

Illustrative Example (Algorithm)

EM Algorithm for two-component Gaussian Mixture :

1. Take initial guesses for the parameters $\hat{\theta}_1 = (\hat{\mu}_1, \hat{\sigma}_1^2)$, $\hat{\theta}_2 = (\hat{\mu}_2, \hat{\sigma}_2^2)$, $\hat{c}$;
2. E-Step : $\hat{q}_i = P(Y_i = 1 | Z = z_i, \hat{\theta}_1, \hat{\theta}_2) = \frac{\hat{c}\phi_{\hat{\theta}_2}(z_i)}{(1-\hat{c})\phi_{\hat{\theta}_1}(z_i)+\hat{c}\phi_{\hat{\theta}_2}(z_i)}$, for $i = 1, 2, \ldots, n$;
3. M-Step : Compute the weighted means and variances :

$$\hat{\mu}_1 = \frac{\sum_{i=1}^n (1-\hat{q}_i) z_i}{\sum_{i=1}^n (1-\hat{q}_i)} \qquad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^n (1-\hat{q}_i)(z_i - \hat{\mu}_1)^2}{\sum_{i=1}^n (1-\hat{q}_i)}$$

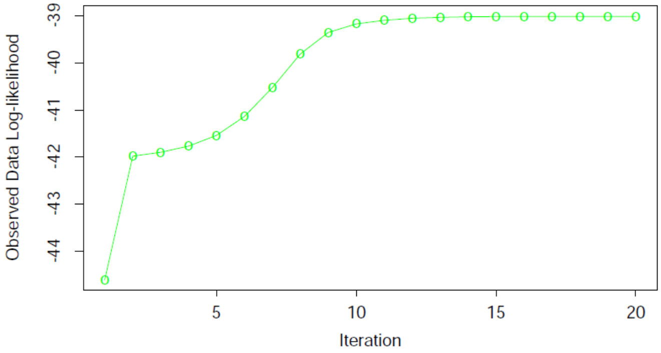
$$\hat{\mu}_2 = \frac{\sum_{i=1}^n \hat{q}_i z_i}{\sum_{i=1}^n \hat{q}_i} \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^n \hat{q}_i (z_i - \hat{\mu}_2)^2}{\sum_{i=1}^n \hat{q}_i}$$

   and the mixing probability $\hat{c} = \frac{1}{n}\sum_{i=1}^n \hat{q}_i$;
4. Iterate between E-Step and M-Step until convergence.

Illustrative Example (Result)

Iteration	1	5	10	15	20
$\hat{c}$	0.485	0.493	0.523	0.544	0.546

The final MLEs are $\hat{\mu}_1 = 4.62$, $\hat{\sigma}_1^2 = 0.87$, $\hat{\mu}_2 = 1.06$, $\hat{\sigma}_2^2 = 0.77$, $\hat{c} = 0.546$.



EM as Maximization-Maximization

- Introduce entropies as penalty to the modified objective function :

$$G(Q,\theta) = F(Q,\theta) - \sum_{i=1}^{n}\sum_{y=1}^{K} Q_{i,y}\log Q_{i,y},$$

where $Q \in \mathbb{Q} = \{Q \in [0,1]^{n,K} : \sum_{y=1}^{K} Q_{i,y} = 1, \forall i\}$

- M-Step is equivalent to : $\theta^{(m+1)} = \arg\max_{\theta} G(Q^{(m+1)},\theta)$
- E-Step is equivalent to : $Q^{(m+1)} = \arg\max_{Q\in\mathbb{Q}} G(Q,\theta^{(m)})$ (Exercise as conditional maximization) : by Jensen's inequality,

$$G(Q,\theta^{(m)}) = \sum_{i=1}^{n}\left(\sum_{y=1}^{K} Q_{i,y}\log\frac{P_{\theta^{(m)}}(X=\mathbf{x}_i, Y=y)}{Q_{i,y}}\right)$$

$$\leqslant \sum_{i=1}^{n}\log\left(\sum_{y=1}^{K} Q_{i,y}\frac{P_{\theta^{(m)}}(X=\mathbf{x}_i, Y=y)}{Q_{i,y}}\right) = L(\theta^{(m)})$$

where "=" iff $\frac{P_{\theta^{(m)}}(X=\mathbf{x}_i, Y=y)}{Q_{i,y}} = C, \forall i,y \Leftrightarrow Q_{i,y} = P_{\theta^{(m)}}(Y=y|X=\mathbf{x}_i)$.

- Monotonicity : $L(\theta^{(m+1)}) \geqslant L(\theta^{(m)})$

EM for Gaussian Mixture as Soft K-Means

- For simplicity, assume $\Sigma_y = I$ for any y.
- E-Step (Partition-Step in K-Means) : $P_{\theta^{(m)}}(Y = y|X = \mathbf{x}_i) = \frac{1}{Z_i} P_{\theta^{(m)}}(Y = y) P_{\theta^{(m)}}(X = \mathbf{x}_i | Y = y) = \frac{1}{Z_i} c_y^{(m)} \exp\Big(-\frac{1}{2} \|\mathbf{x}_i - \mu_y^{(m)}\|^2 \Big)$, where Z_i is a normalization factor.
- M-Step : $\max\limits_{c_y, \mu_y} \sum_{i=1}^n \sum_{y=1}^K P_{\theta^{(m)}}(Y = y|X = \mathbf{x}_i) \Big(\log c_y - \frac{1}{2} \|\mathbf{x}_i - \mu_y\|^2 \Big)$ leads to

$$\mu_y = \sum_{i=1}^n P_{\theta^{(m)}}(Y = y|X = \mathbf{x}_i) \mathbf{x}_i \quad \text{(Mean-Step in K-Means)}$$

$$c_y = \frac{\sum_{i=1}^n P_{\theta^{(m)}}(Y = y|X = \mathbf{x}_i)}{\sum_{y'=1}^K \sum_{i=1}^n P_{\theta^{(m)}}(Y = y'|X = \mathbf{x}_i)} \quad \text{(for partition in next step)}$$

- “Soft” because the partition is done in probabilistic sense instead of deterministic sense and the average is weighted according to the probability.

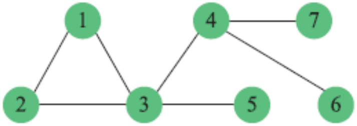
Summary of EM Algorithm

- EM is unsupervised learning, an approach to perform MLE for mixture models with latent variables
- EM is an alternating optimization
- EM can be viewed as soft K-Means
- EM can deal with problems including missing data (treat missing data as latent variables and use Bayes formula, see Section 8.5.2 in the book "Elements of Statistical Learning" for general EM algorithm).
- EM can be used in the framework of Bayesian reasoning (MAP), e.g., Variational Bayesian EM algorithm
- EM is related to generative model, e.g., EM for Gaussian mixture is a population approach to learning the sample distributions, analogous to Gibbs sampling which is sampling approach to learning the distribution.
- EM is a general methodology, even used in natural language processing (e.g., latent dirichlet allocation), deep learning (e.g., restricted Boltzmann machine, deep belief network)

Outlines

Graphs

- A set of data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, similarity s_{ij} or distance d_{ij}
- Graph $G = (V, E)$, where $V = \{v_i\}_{i=1}^n$ with each v_i representing a sample $\mathbf{x}_i$
- v_i and v_j are connected ($w_{ij} > 0$) if $s_{ij} > \epsilon$ where $\epsilon \geqslant 0$ is a threshold ; then the edge is weighted by $w_{ij} = s_{ij}$
- Undirected graph $w_{ij} = w_{ji}$, adjacency matrix $W = \{w_{ij}\}$
- Degree of v_i : $d_i = \sum_{j=1}^n w_{ij}$; $D = \mathrm{diag}(d_1, \ldots, d_n)$



$$W = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Similarity Graphs

- ϵ-neighborhood graph : v_i and v_j are connected if $d(x_i, x_j) < \epsilon$; unweighted graph ; $\epsilon \sim (\log n/n)^p$; difficult to choose ϵ for data on different scales
- k-nearest neighbor graph : connect v_i to v_j if v_j is among the k-nearest neighbors of v_i, directed graph ; connect v_i and v_j if v_i and v_j are among the k-nearest neighbors of each other, mutual k-nearest neighbor graph, undirected ; $k \sim \log n$
- Fully connected graph : connect all points with positive similarity with each other ; model local neighborhood relationships ; Gaussian similarity function $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$, where σ controls the width of neighborhoods ; adjacency matrix is not sparse ; $\sigma \sim \epsilon$

Graph Laplacian

- Unnormalized graph Laplacian : $L = D - W$
  - Has $\mathbf{1}$ as an eigenvector corresponding to the eigenvalue 0
  - Symmetric and positive definite : $\mathbf{f}^T\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2$
  - Non-negative, real-valued eigenvalues $0 = \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$
  - The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}$, where $A_1, \ldots, A_k$ are k connected components in the graph
- Normalized graph Laplacians :
  - Symmetric Laplacian : $L_{sym} = D^{-1/2}LD^{-1/2}$
  - Random walk Laplacian : $L_{rw} = D^{-1}L$
  - Both have similar properties as L

Spectral Clustering

- Graph cut : segment G into K clusters $A_1, \ldots, A_K$, where $A_i \subset V$, this is equivalent to minimize the graph cut function

$$cut(A_1, \ldots, A_K) = \frac{1}{2} \sum_{k=1}^{K} W(A_k, \bar{A}_k)$$

  where $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$. Trivial solution consists of a singleton and its complement

- RatioCut : $RatioCut(A_1, \ldots, A_K) = \frac{1}{2} \sum\limits_{k=1}^{K} \frac{W(A_k, \bar{A}_k)}{|A_k|}$, where $|A|$ is the number of vertices in A
- Normalized cut : $Ncut(A_1, \ldots, A_K) = \frac{1}{2} \sum\limits_{k=1}^{K} \frac{W(A_k, \bar{A}_k)}{vol(A_k)}$, where $vol(A) = \sum_{i \in A} d_i$; it is NP-hard

Relaxation of RatioCut to Eigenvalue Problems with $K = 2$

- $\min_{A \subset V} RatioCut(A, \bar{A})$
- Binary vector $f = (f_1, \ldots, f_n)^T$ as indicator function :
  $$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|}, & \text{if } v_i \in A \\ -\sqrt{||A|/\bar{A}|}, & \text{if } v_i \in \bar{A} \end{cases}$$
- $f^T L f = |V| \cdot RatioCut(A, \bar{A})$, $\sum_{i=1}^{n} f_i = 0$, and $\|f\|_2^2 = n$
- Relax f to be real-valued : $\min_{f \in \mathbb{R}^n} f^T L f$, subject to $f \perp \mathbf{1}$ and $\|f\|_2 = \sqrt{n}$
- By Rayleigh-Ritz theorem, the solution f is the eigenvector corresponding to the second smallest eigenvalue of L
- Cluster $\{f_i\}_{i=1}^n$ to two groups C and $\bar{C}$: $v_i \in A$ if $f_i \in C$, and else $v_i \in \bar{A}$

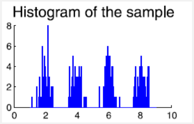
Relaxation of RatioCut and Ncut with general K

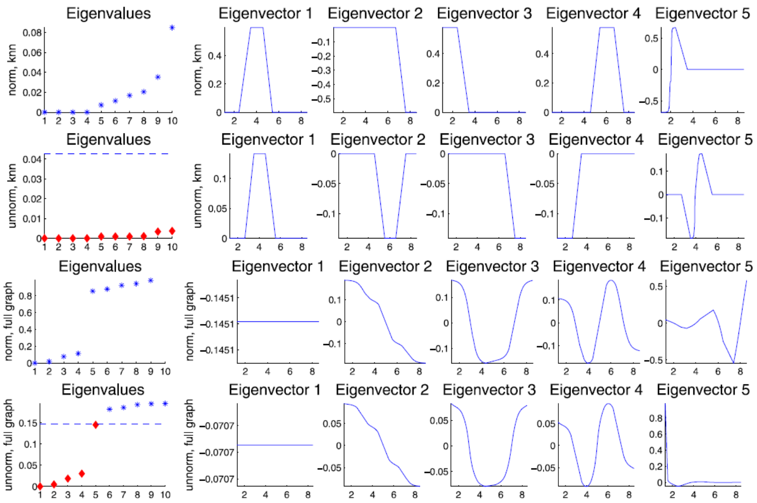
- RatioCut
  - Binary vector $h_j = (h_{1j}, \ldots, h_{nj})^T$, $j = 1, \ldots, K$, as indicator function : $h_{ij} = \begin{cases} 1/\sqrt{|A_j|}, & \text{if } v_i \in A_j \\ 0, & \text{otherwise} \end{cases}$
  - $h_j^T L h_j = Cut(A_j, \bar{A}_j)/|A_j|$, $H = (h_1, \ldots, h_K) \in \mathbb{R}^{n \times K}$, $RatioCut(A_1, \ldots, A_K) = \mathrm{Tr}(H^T L H)$, $H^T H = I$
  - Relax H : $\min\limits_{H \in \mathbb{R}^{n \times K}} \mathrm{Tr}(H^T L H)$, subject to $H^T H = I$
  - Solution : the first K eigenvectors of L as columns
  - Cluster the rows of H to K groups
- Ncut
  - Replacing $|A_j|$ by $vol(A_j)$, the same argument for the relaxation of Ncut : $\min\limits_{H \in \mathbb{R}^{n \times K}} \mathrm{Tr}(H^T L H)$, subject to $H^T D H = I$
  - Solution : the first K eigenvectors of L_{rw} as columns

Spectral Clustering Algorithm

- Input : Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters
- Output : Clusters $A_1, \ldots, A_K$ of indices of vertices
- Algorithm :
  1. Construct a similarity graph $G = (V, E)$ with weighted adjacency matrix W
  2. Compute the unnormalized graph Laplacian L or normalized graph Laplacian L_{sym} or L_{rw}
  3. Compute the first K eigenvectors $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K] \in \mathbb{R}^{n \times K}$
  4. In the case of L_{sym}, normalize the rows of U to norm 1 ; for the other two cases, skip this step
  5. Let $\mathbf{y}_i \in \mathbb{R}^K$ be the i-th row of $\mathbf{U}$, use K-means to cluster the point set $\{\mathbf{y}_i\}_{i=1}^n$ into clusters $C_1, \ldots, C_K$
  6. $A_k = \{i | y_i \in C_k\}$

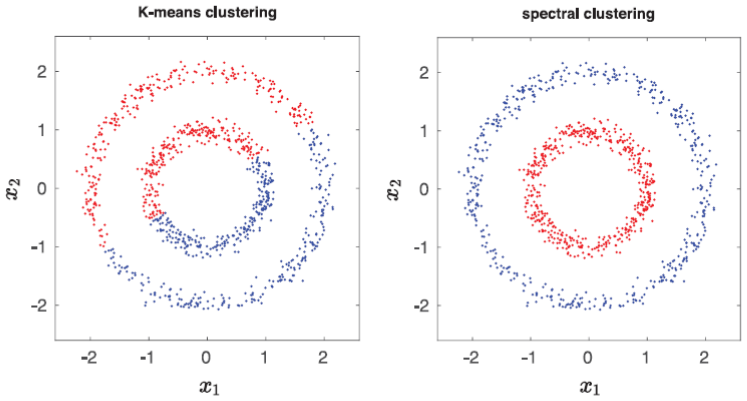
Mixture of 4 Gaussians on $\mathbb{R}$:





Interpretations

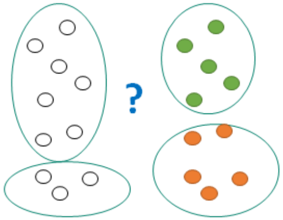
- Usually better than K-means



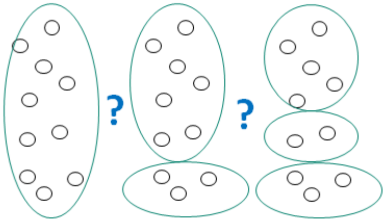
Outlines

Two Types of Indices

- External indices : validate against ground truth (labels), or compare two clusters (how similar)
  - Purity
  - Jaccard coefficient and Rand index
  - Mutual information
- Internal indices : validate without external info, based on the within-cluster similarity and between-cluster distance
  - Davies-Bouldin index (DBI)
  - Silhouette coefficient (SI)



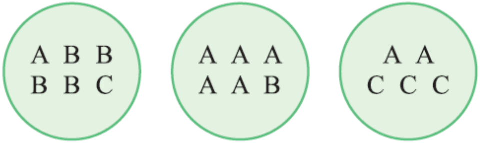
External



Internal

Purity

- Let n_{ij} be the number of samples that belong to label j but were assigned to cluster i
- Then $n_i = \sum_{j=1}^{C}$ is the total number of samples in cluster i
- $p_{ij} = n_{ij}/n_i$ is the probability distribution in cluster i
- Purity of cluster i : $p_i \triangleq \max_j p_{ij}$
- Total purity $\triangleq \sum_i \frac{n_i}{n} p_i$
- Example : purity $= \frac{6}{17}\frac{4}{6} + \frac{6}{17}\frac{5}{6} + \frac{5}{17}\frac{3}{5} = 0.71$
- Naive case : treating each sample as a cluster leads to purity 100%



Confusion Matrix

- SS (True Positive or TP) : # of pairs of samples belonging to the same cluster in both models
- DD (True Negative or TN) : # of pairs of samples belonging to different clusters in both models
- SD (False Positive or FP) : # of pairs of samples belonging to the same cluster in clustering model, but different clusters in reference model
- DS (False Negative or FN) : # of pairs of samples belonging to different clusters in clustering model, but the same cluster in reference model

Clustering model

Reference model

	Same Cluster	Different Cluster
Same class	SS	DS
Different class	SD	DD

Jaccard Coefficient and Rand Index

- Rand index (RI) : $RI = \frac{SS+DD}{SS+SD+DS+DD} \in [0,1]$, similar to the accuracy in classification problems
- Jaccard coefficient (JC) : $JC = \frac{SS}{SS+SD+DS} \in [0,1]$, compare the similarity and diversity of the samples
- Example : # of pairs in the same cluster in clustering model $= SS + SD = C_6^2 + C_6^2 + C_5^2 = 40$, and $SS = \underbrace{C_4^2}_{cluster1} + \underbrace{C_5^2}_{cluster2} + \underbrace{C_3^2 + C_2^2}_{cluster3} = 20$, so $SD = 20$; # of pairs in the same cluster in clustering model $= DS + DD = 6 \times 6 + 6 \times 5 + 6 \times 5 = 96$, and $DS = \underbrace{4 \times 1}_{B} + \underbrace{1 \times 5 + 1 \times 2 + 5 \times 2}_{A} + \underbrace{1 \times 3}_{C} = 24$, so $DD = 72$.

$$RI = \frac{20+72}{20+20+24+72} = 0.68, \qquad JC = \frac{20}{20+20+24} = 0.31$$

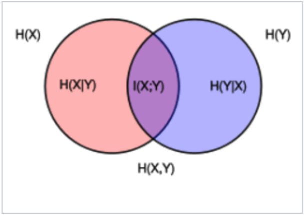
Mutual Information (Wikipedia)

- Mutual information (MI) measures the uncertainty decrement of one random variable given another random variable
- Probability that a sample belongs to both cluster u_i and v_j : $p_{UV}(i,j) = \frac{|u_i \bigcap v_j|}{n}$
- Its marginal probabilities are : $p_U(i) = \frac{u_i}{n}$ and $p_V(j) = \frac{v_j}{n}$
- Mutual information : $I(U,V) = \sum_{i=1}^{R}\sum_{j=1}^{C} p_{UV}(i,j)\log\frac{p_{UV}(i,j)}{p_U(i)p_V(j)}$
- MI attains its maximum $\min\{H(U), H(V)\}$ only when we have many small clusters
- Normalized MI : $NMI(U,V) = \frac{I(U,V)}{(H(U)+H(V))/2}$

- Entropy : $H(X) = -\sum_x p(x)\log p(x)$
- Conditional entropy :

$$H(X|Y) = \sum_y p(y) H(X|Y=y)$$
$$= \sum_y p(y)\Big(-\sum_x p(x|y)\log p(x|y)\Big)$$

- MI : $I(X;Y) = H(X) - H(X|Y)$



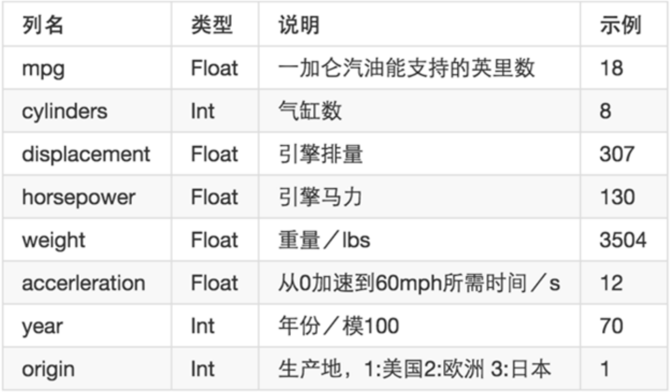
Davies-Bouldin Index and Silhouette Coefficient

- Davies-Bouldin index (DBI) measures both the within-cluster divergence and between-clusters distance
- $DBI = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{div(c_i)+div(c_j)}{d(\mu_i,\mu_j)}\right)$ where $div(c_i)$ represents the average distance of samples within cluster c_i, μ_i is the center of cluster c_i
- Silhouette Coefficient (SC) : $SC = \frac{b_i-a_i}{\max(a_i,b_i)}$, where a_i is average distance between the i-th sample and every other sample in the same cluster, b_i is the minimal distance from the i-th sample to the other clusters ; range is $[-1,1]$
- The smaller the DBI, or the larger the SC, the better the clustering results

Outlines

Case Study

- Use clustering to group the cars with similar performance based on parameters of the cars
- Dataset comes from “Auto” in the R package ISLR.



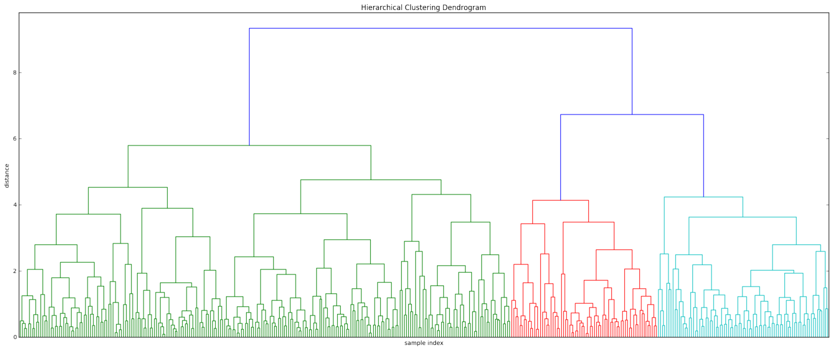
列名	类型	说明	示例
mpg	Float	一加仑汽油能支持的英里数	18
cylinders	Int	气缸数	8
displacement	Float	引擎排量	307
horsepower	Float	引擎马力	130
weight	Float	重量／lbs	3504
accerleration	Float	从0加速到60mph所需时间／s	12
year	Int	年份／模100	70
origin	Int	生产地，1:美国2:欧洲 3:日本	1

Hierarchical Clustering

- Scaling of the feature values : Auto_Scaled
- from scipy.cluster.hierarchy import dendrogram, linkage
- Construct linkage matrix : Z = linkage(Auto_Scaled, method = 'complete', metric = 'euclidean'), possible choice for metric could be 'euclidean', 'cityblock', 'minkowski', 'cosine', 'correlation', 'hamming', 'jaccard', etc.
- Data structure of linkage matrix : in the t-th iteration, clusters C_i with index "Z[i, 0]" and C_j with index "Z[i, 1]" are combined to form cluster C_q with index "n + i" ; "Z[i, 2]" is the distance between C_i and C_j ; "Z[i, 3]" is the number of samples in C_q

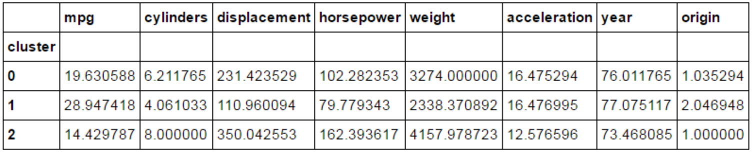
Dendrogram

dendrogram(Z, no_labels = True)



K-Means Clustering

- from sklearn.cluster import KMeans
- clf = KMeans(n_clusters=3, n_init=1, verbose=1)
- clf.fit(Auto_Scaled)
- Cluster 1 : (economy or compact vehicles) high mpg, low horsepower, low weight ; cluster 2 : (luxury vehicles) low mpg, high horsepower, high weight ; cluster 0 : intermediate performance



	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
cluster								
0	19.630588	6.211765	231.423529	102.282353	3274.000000	16.475294	76.011765	1.035294
1	28.947418	4.061033	110.960094	79.779343	2338.370892	16.476995	77.075117	2.046948
2	14.429787	8.000000	350.042553	162.393617	4157.978723	12.576596	73.468085	1.000000

References

- 数据分析导论，博雅大数据学院
- 周志华，机器学习，2016
- T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning : Data mining, Inference, and Prediction, 2nd Edition, 2009
- Arthur, D., Vassilvitskii, S. “k-means++ : the advantages of careful seeding”. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027 – 1035, 2007
- Lingras P, West C, Interval Set Clustering of Web Users with Rough Kmeans, Journal of Intelligent Information Systems 23(1) :5 – 16, 2004