自然语言处理

文本聚类

软件学院

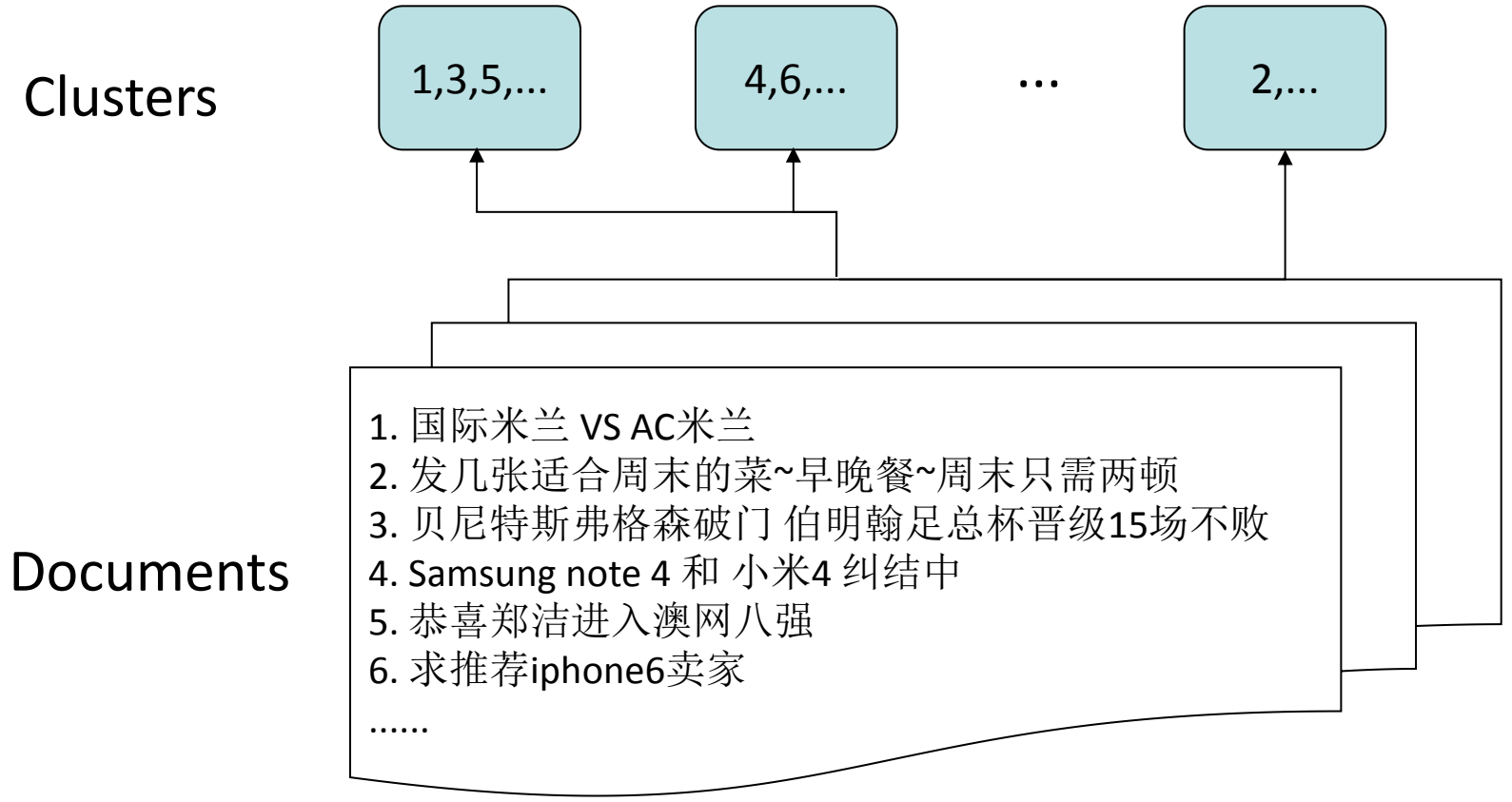
Overview

- 目前为止：分类问题
  - 应用：文本主题分类、语言识别、词义消歧等
  - “有监督”学习框架：学习时需要类别标签
    - 生成模型：e.g., Naïve Bayes
      - 模型=一组分布P(x,y;θ), $\theta \in \Theta$
      - 选择一种最可能从中抽样得到训练数据的分布 $P(\mathrm{x}, \mathrm{y}; \hat{\theta})$
      - 根据 $\hat{y} = \arg\max_y P(\mathrm{x}, \mathrm{y}; \hat{\theta})$ 对新样本x进行分类
    - 判别模型：e.g., maximum entropy models (a.k.a. logistic regression)
      - 模型=一组分类函数F
      - 选择一种可以对训练样本正确分类的函数 $\hat{f} \in F$
      - 根据 $\hat{y} = sign(\hat{f}(x))$ 对新样本x进行分类

Overview

- 接下来：聚类问题
  - “无监督”学习框架：学习时不需要类别标签
  - **Magic：挖掘数据内部的隐藏模式（结构、类别等）**
  - 很多NLP任务都关乎聚类问题：IR, recommendation system, exploratory data analysis

例子：标题聚类

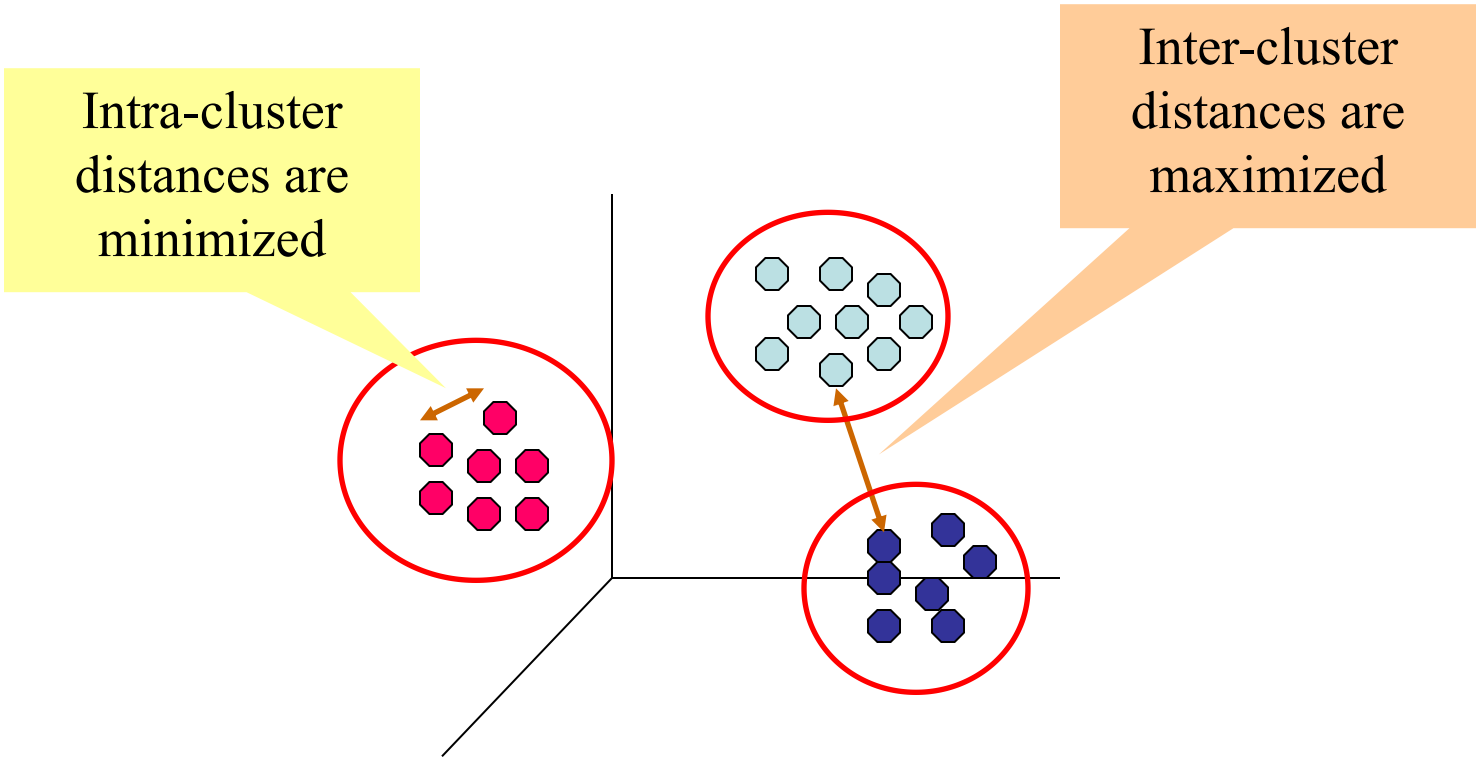

Clusters
1,3,5,...
4,6,...
...
2,...
Documents
1. 国际米兰 VS AC米兰
2. 发几张适合周末的菜~早晚餐~周末只需两顿
3. 贝尼特斯弗格森破门 伯明翰足总杯晋级15场不败
4. Samsung note 4 和 小米4 纠结中
5. 恭喜郑洁进入澳网八强
6. 求推荐iphone6卖家
.....

例子：web搜索优化

- 用户提交的query本身往往包含歧义
  - E.g., Jaguar, NLP, Paris Hilton

  - 前10页几乎没有关于”animal”词义下的结果
- 仅靠词义消歧通常不能很好的解决web搜索的歧义
  - 需要交由用户选择相关的搜索结果
  - 将搜索结果按照query的语义进行分组
  - Jaguar or Jaguar car

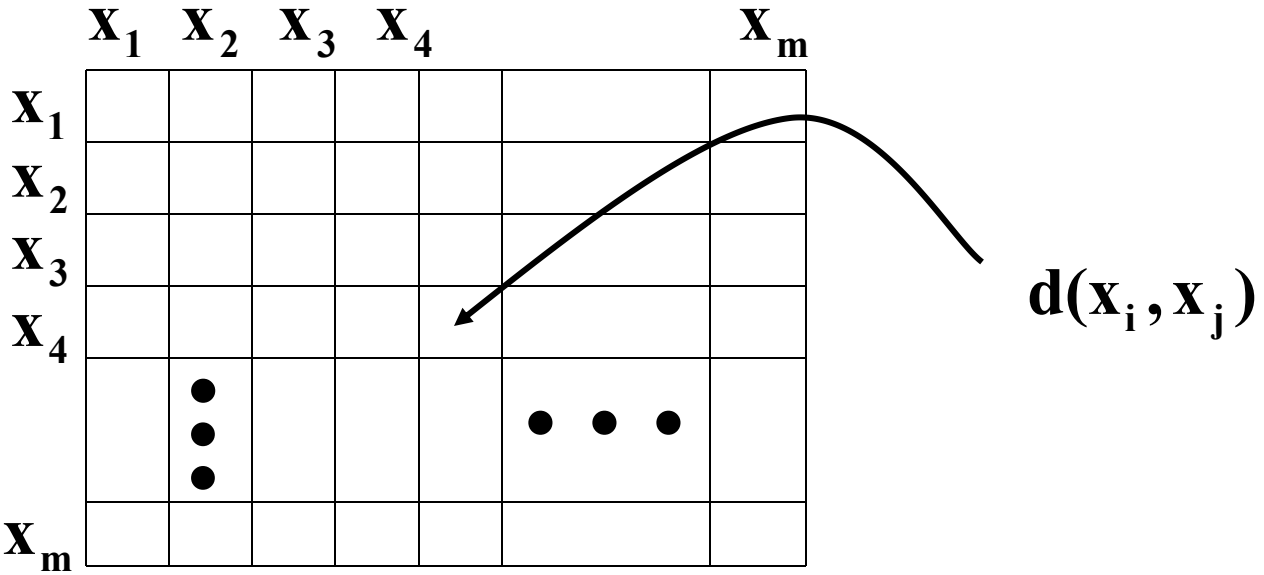
聚类



- 如何定义样本间的距离？
- 如何找到这样一组划分？

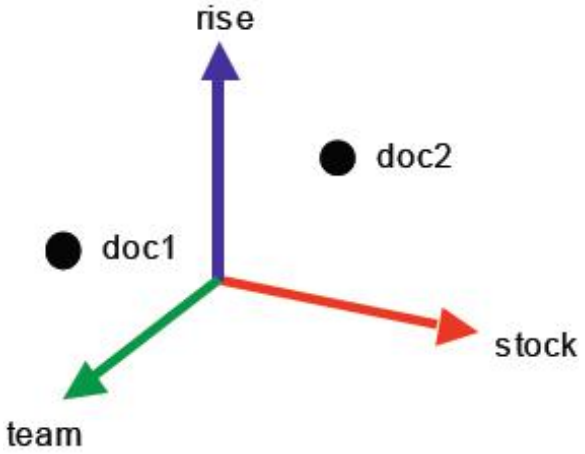
距离测度

- 假设每个样本可以看成高维空间内的一个点，则可以采用距离测度表示样本间的距离
- 则聚类问题的输入可以看成由样本距离张成的一个矩阵：

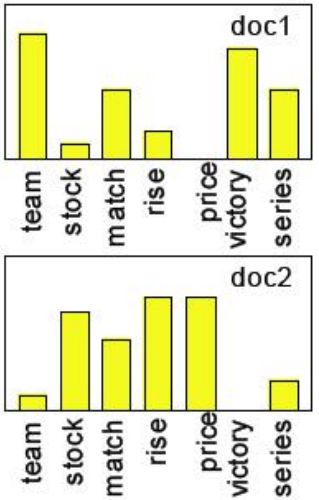


距离测度

- 向量空间表示
  - 一个文档可以看成高维向量空间内的一个点
  - Distance: Euclidean / cosine



- 概率表示
  - 一个文档可以看做从某个经验分布抽样得到的词汇的集合
  - Distance: divergences



距离测度

- 传统意义上，距离测度可以表示为一个函数，满足：

$$1.\ \mathbf{d(x,y) \geq 0,\ d(x,y) = 0 \Leftrightarrow x = y}$$

$$2.\ \mathbf{d(x,y) + d(y,z) \geq d(x,z)}$$

$$3.\ \mathbf{d(x,y) = d(y,x)}$$

- 但对于聚类问题，采用的距离测度（或相似度）可以不严格满足以上三个约束

距离测度

Examples:

- Euclidean Distance:

$$d(x,y) = \sqrt{(x-y)^2} = \sqrt{(x-y)^T(x-y)} = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$$ L_2

- Manhattan Distance:

$$d(x,y) = |x-y| = \sum_{i=1}^{d}|x_i - y_i|$$ L_1

- Chebyshev Distance:

$$d(x,y) = \max_{1\le i\le d}|x_i - y_i|$$ L_∞

•注意: 对于欧氏距离d(x,y)， $d^2(x,y)$ 不是一个测度（metric）但可用于距离度量（measure）（不满足三角不等式）

距离测度

Examples:

- Mahalanobis Distance:

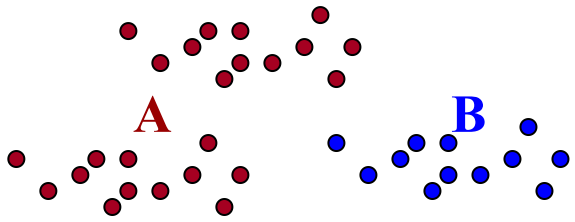
$$\mathbf{d(x,y)} = \sqrt{(\mathbf{x}-\mathbf{y})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mathbf{y})}$$

其中 Σ 表示协方差矩阵

距离测度

- 样本点 x 和样本点集合 A 之间的距离：

$$d(x,A)=\frac{1}{|A|}\sum_{y\in A} d(x,y)$$



- 两个样本点集合A, B之间的距离：

$$d(A,B)=\frac{1}{|A||B|}\sum_{x\in A,y\in B} d(x,y)$$

- 注意，还有其它多种不同的刻画方式

聚类模型

- K-means
- Hierarchical Clustering
- Density-based Clustering
- Gaussian Mixture Model
- Spectral Clustering
- ……

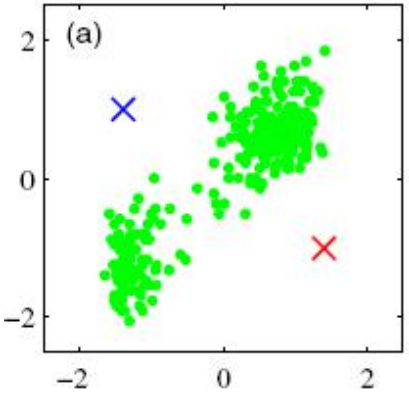
K-means

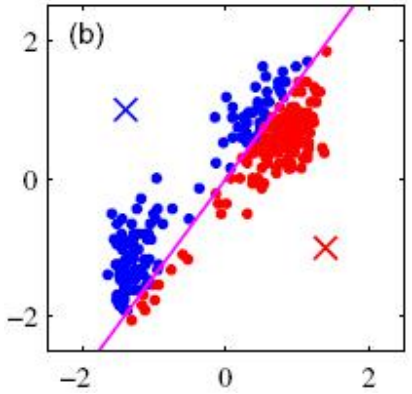
- 先确定簇的个数，K
- 假设每个簇都有一个中心点（centroid）
- 将每个样本点划分到距离它最近的中心点所属的簇中
- 迭代过程：

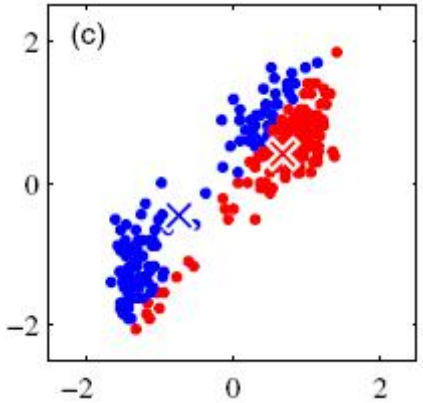
The basic algorithm:

1: select K points as the initial centroids
2: **repeat**
3: Form K clusters by assigning all points to the *closest centroid*
4: Recompute the centroid of each cluster
5: **until** the centroids don't change

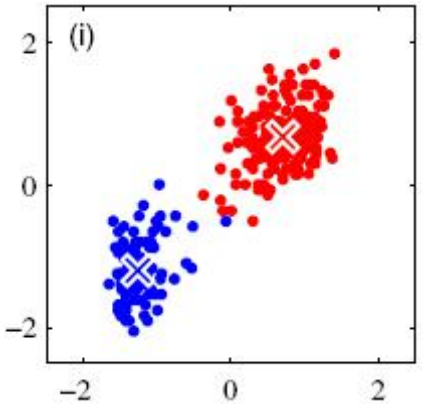
K-means Clustering







...



K-means

- 目标函数：定义为每个样本与其簇中心点的距离的平方和（the Sum of Squared Error, SSE）

$$SSE = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\,\mathrm{dist}(x_n - \mu_k)$$ e.g., $\mathrm{dist}(x_n - \mu_k) = ||x_n - \mu_k||^2$

  - μ_k 表示簇C_k 的中心点（或其它能代表C_k的点）
  - 若x_n被划分到簇C_k则r_{nk}=1，否则r_{nk}= 0

- **目标**：找到簇的中心点μ_k及簇的划分r_{nk}使得目标函数SSE最小

K-means

- 1: choose some initial values for the μ_k (k=1,...,K)
- 2: **repeat**
- 3: minimize SSE with respect to the r_{nk} (n=1,...,N)

$$r_{nk} = \begin{cases} 1 \text{ if } k = \arg\min_j ||x_n - \mu_j||^2 \\ 0 \text{ otherwise} \end{cases}$$

- 4: minimize SSE with respect to the m_k

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

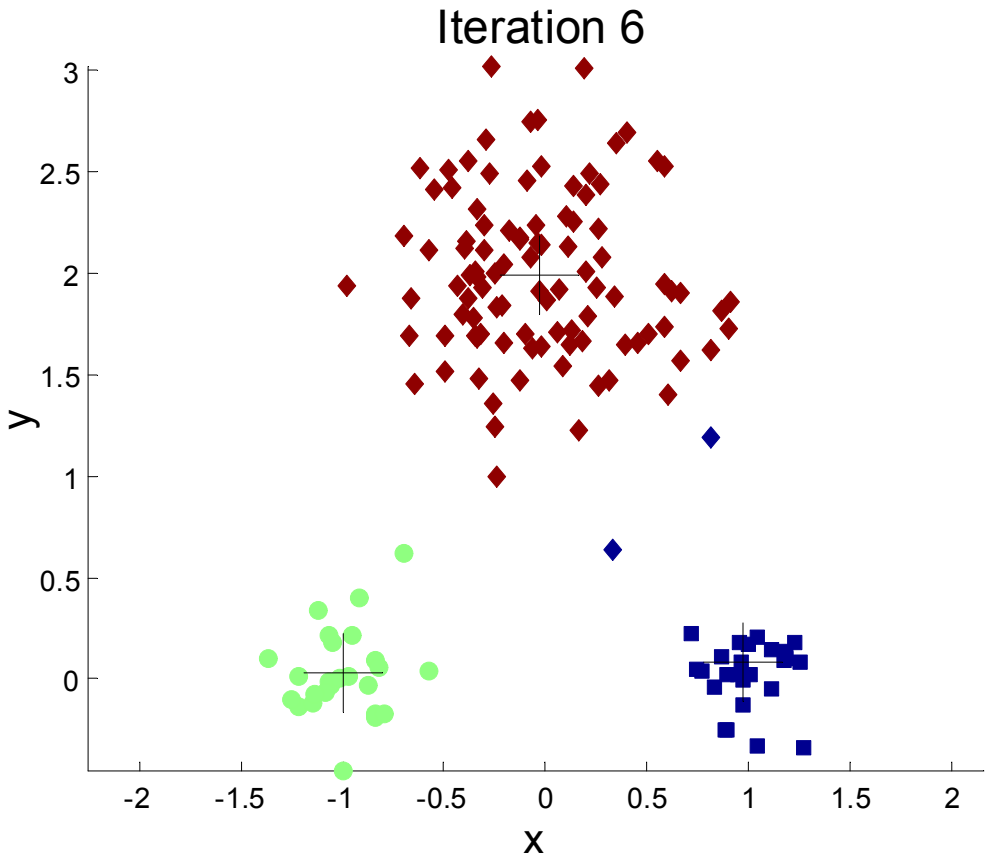
- 5: **until** convergence

K-means – Details

- 初始中心点通常是随机选取的
  - 产生的簇可能和上一次迭代相差很大
- 中心点通常是当前簇中所有样本点的均值
  - K-medoids：中位数
- 算法复杂度：$O(n \times d \times K \times I)$
  - n=样本点个数，d = 样本特征为度
  - K = 类的个数，I = 迭代次数
- 需要预先确定K!

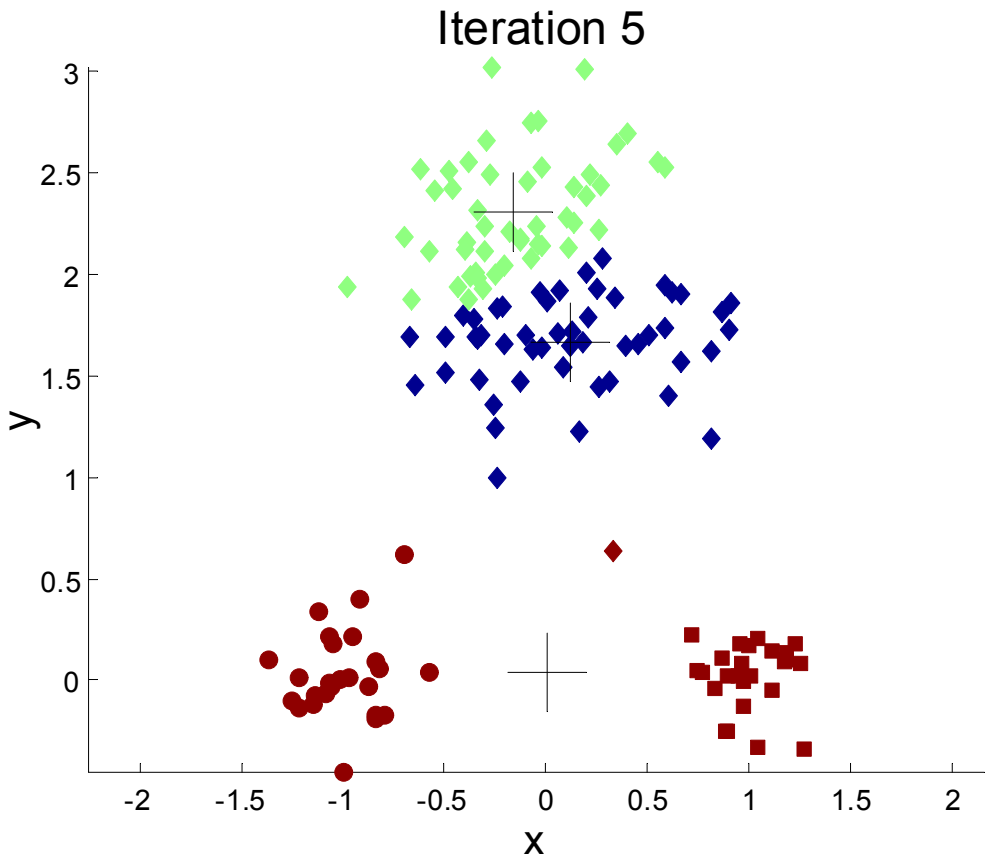
K-means – Details

不同初始中心点的选取对聚类结果的影响：



K-means – Details

不同初始中心点的选取对聚类结果的影响：



K-means – Details

类别本身可能存在歧义：

有多少个类？

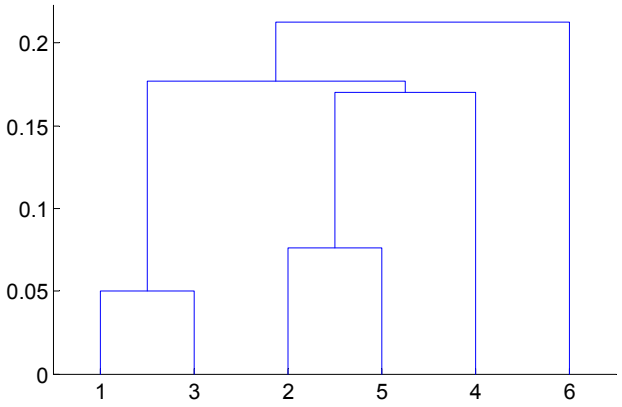
Six Clusters

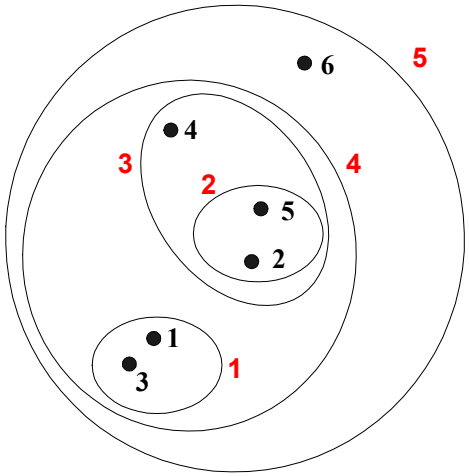
Two Clusters

Four Clusters

Hierarchical Clustering

- 为数据集输出一个嵌套的、层次化的类别树：dendrogram
  - 树结构记录了簇的合并或拆分
  - 自底向上（agglomerative）
  - 自顶向下（divisive）





Hierarchical Agglomerative Clustering (HAC)

- 层次凝聚聚类：一种很常用的聚类模型

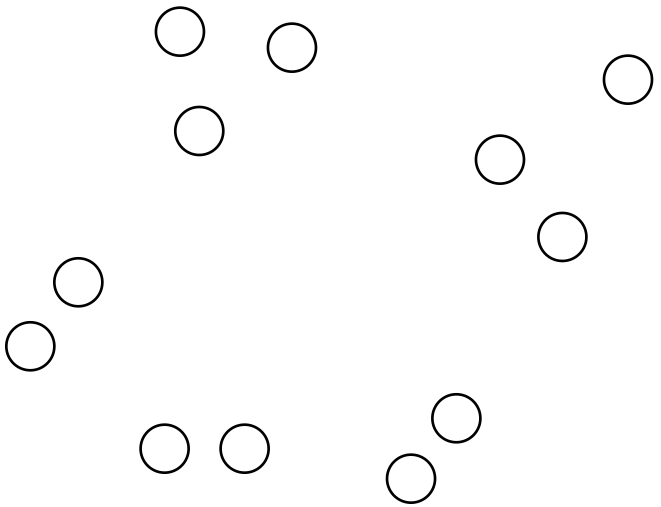
Basic algorithm:

1. Compute the proximity matrix
2. Let each data point be a cluster
3. **Repeat**
4. **Merge** the two closest clusters
5. Update the proximity matrix
6. **Until** only a single cluster remains

- 关键步骤：计算两个簇的相似度
  - 不同的度量两个簇的相似度方法，区分了不同的聚类算法

初始状态

- 初始状态：每个点代表一个簇以及一个相似度矩阵(proximity matrix)

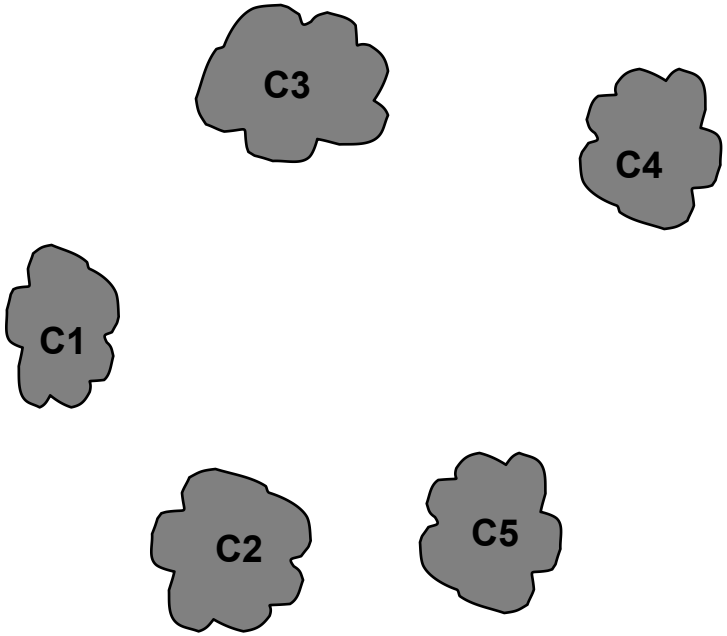


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

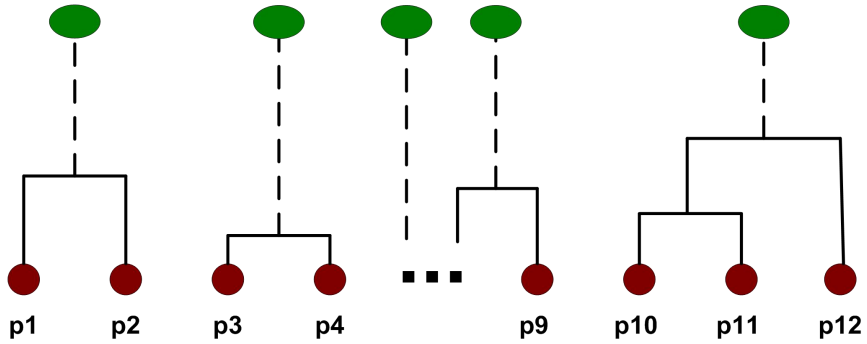
中间状态

- 经过几次合并操作，得到一些簇以及一个相似度矩阵



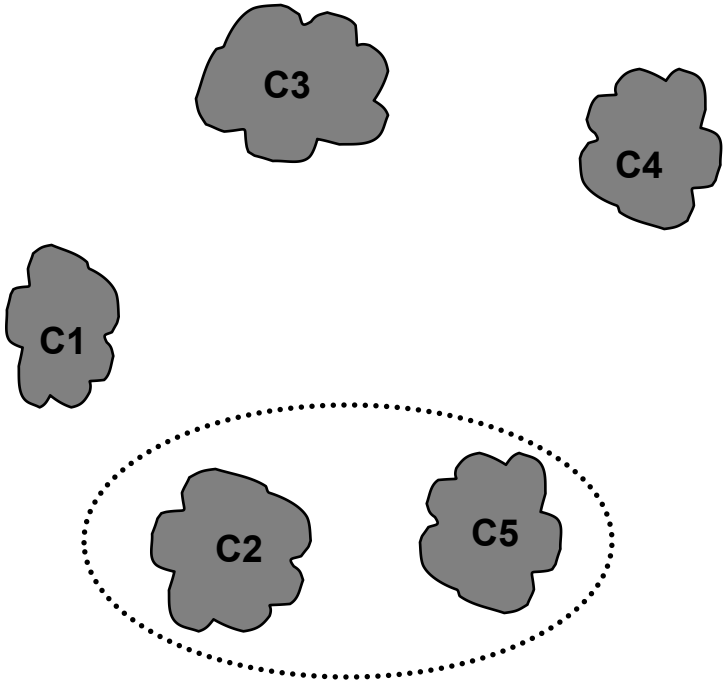
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

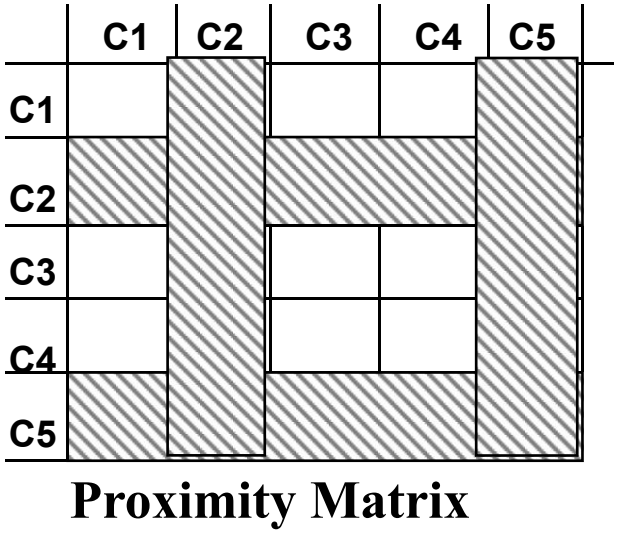
Proximity Matrix



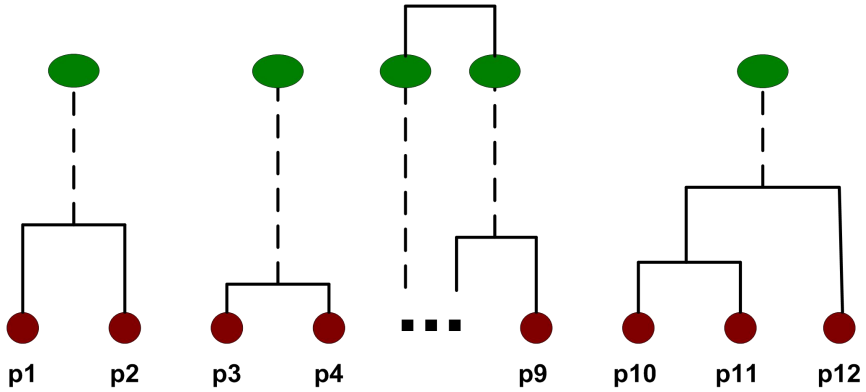
合并操作

- 合并操作：合并最近的两个簇(C2 and C5)，同时更新相似度矩阵



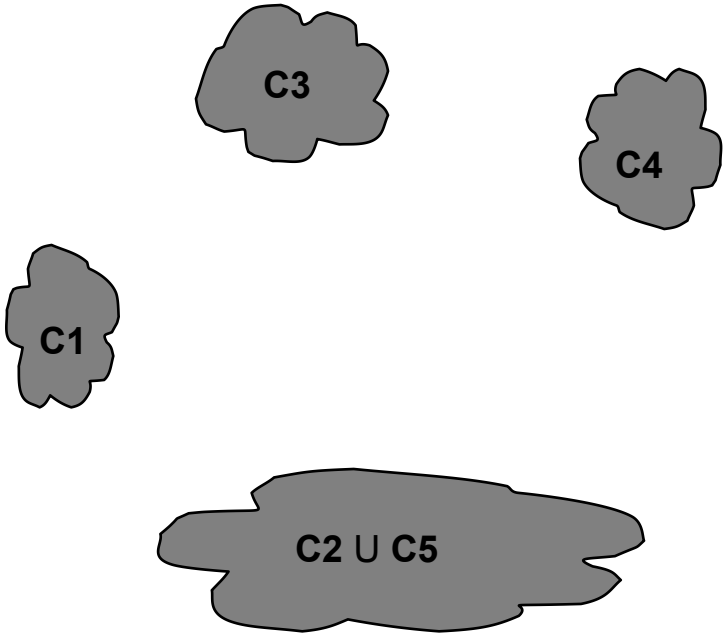


Proximity Matrix



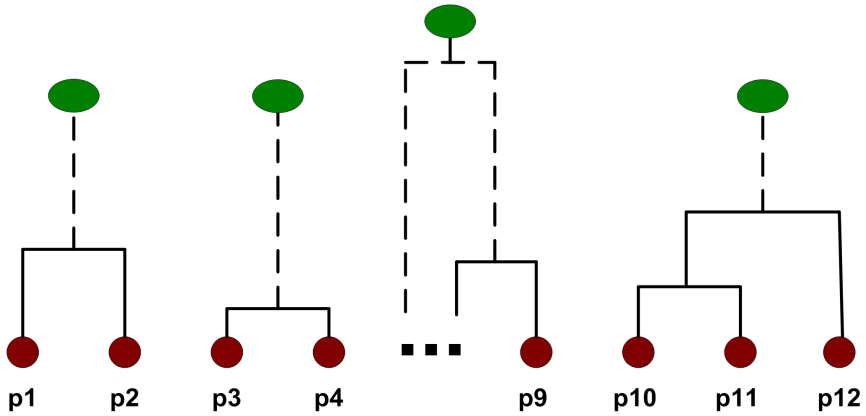
合并操作

- 问题：如何更新相似度矩阵？

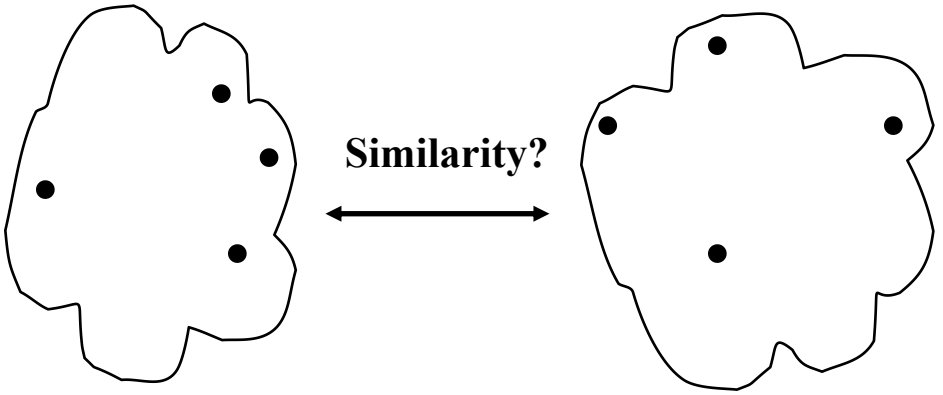


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



Inter-Cluster Similarity

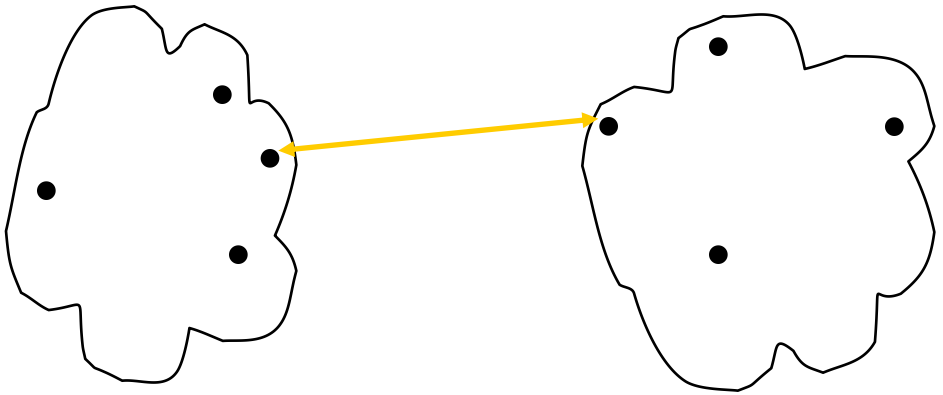


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- 两个不相交的簇G和H，其间的相似度D(G, H)可以通过点点间的相似度(pairwise similarities) D(i, j), i∈G, j∈H, 计算得到

Inter-Cluster Similarity

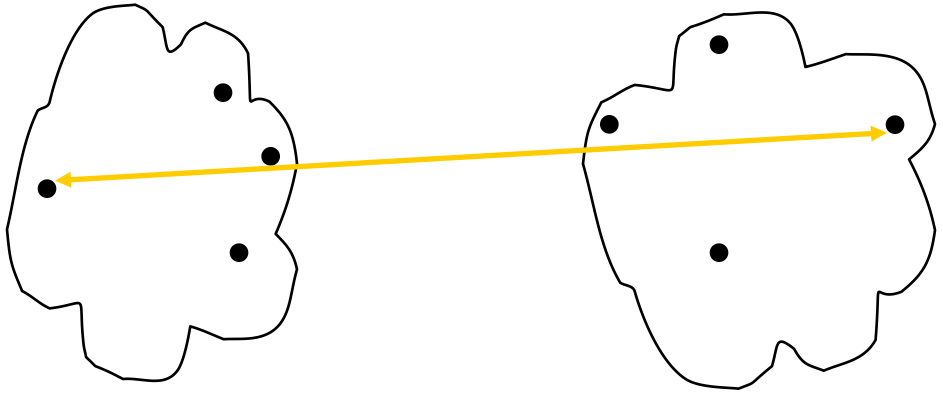


- MIN (single linkage)
- MAX (complete linkage)
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

Inter-Cluster Similarity

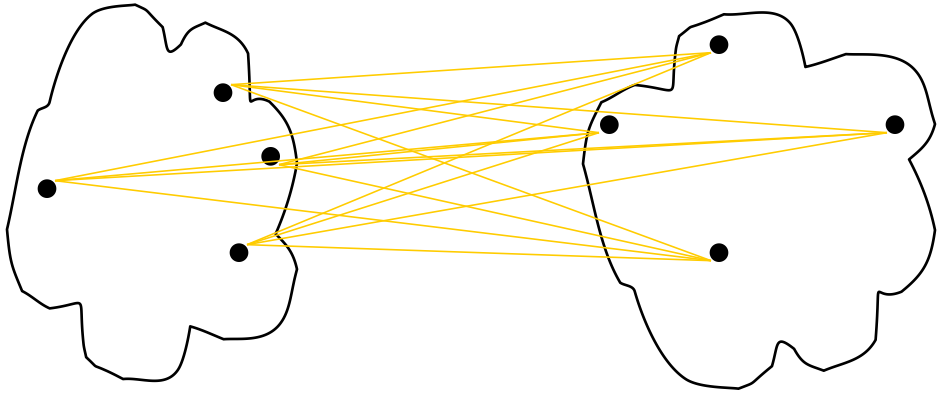


- MIN (single linkage)
- MAX (complete linkage)
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

How to Define Inter-Cluster Similarity

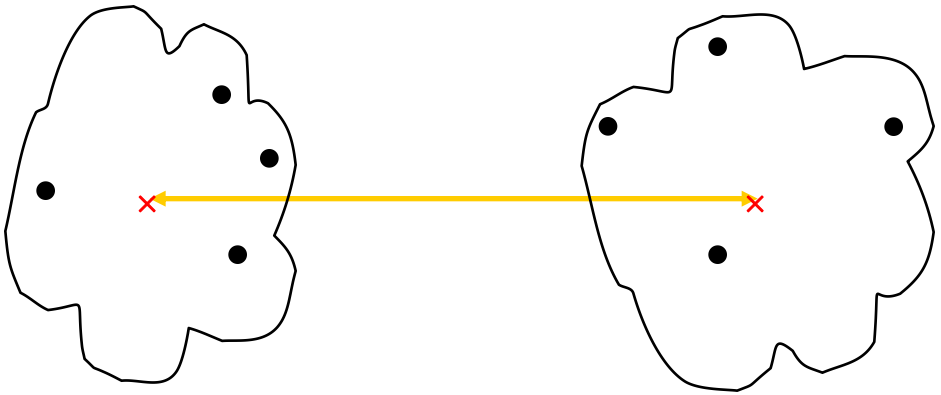


- MIN (single linkage)
- MAX (complete linkage)
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

Inter-Cluster Similarity

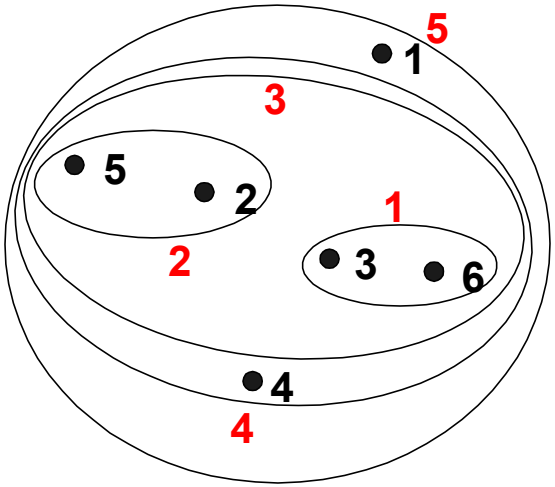


- MIN (single linkage)
- MAX (complete linkage)
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

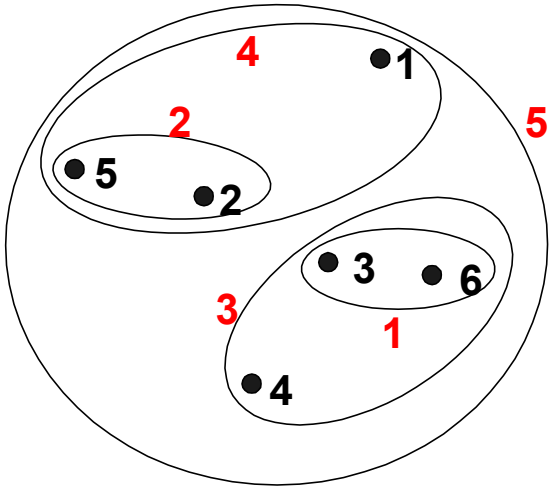
Proximity Matrix

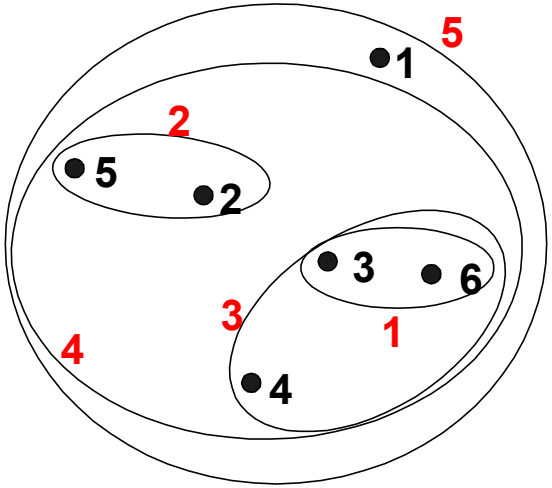
Hierarchical Clustering: Comparison



MIN

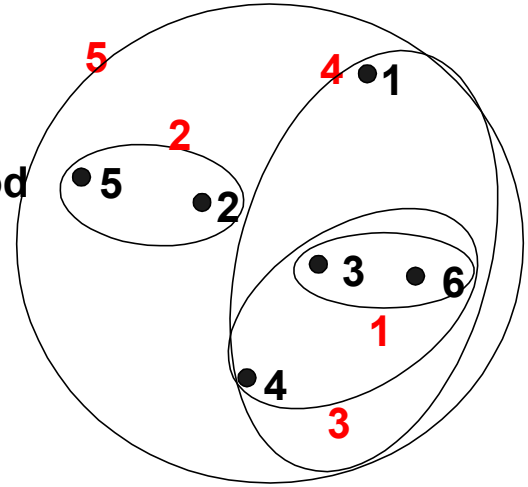
MAX





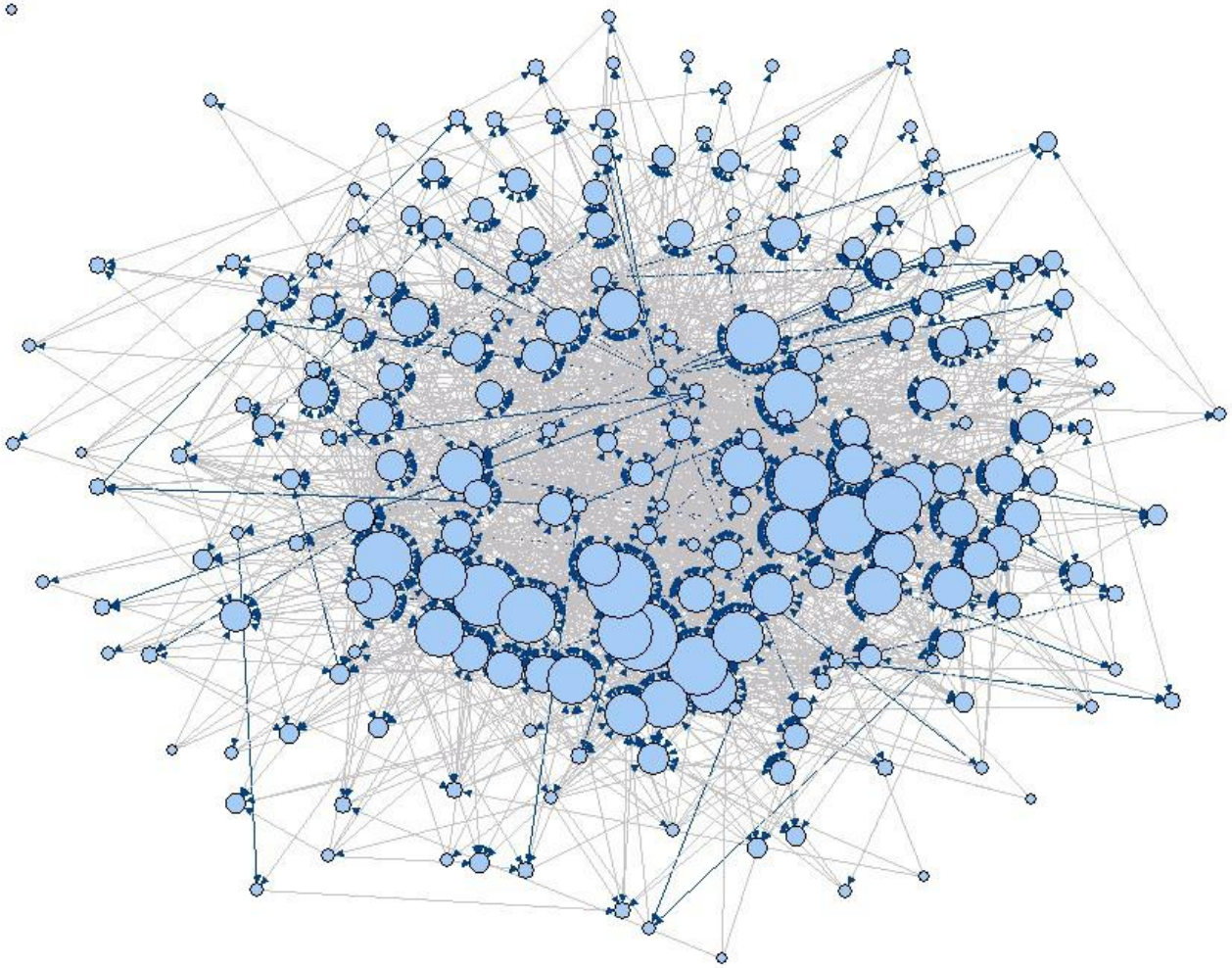
Group Average

Ward's Method



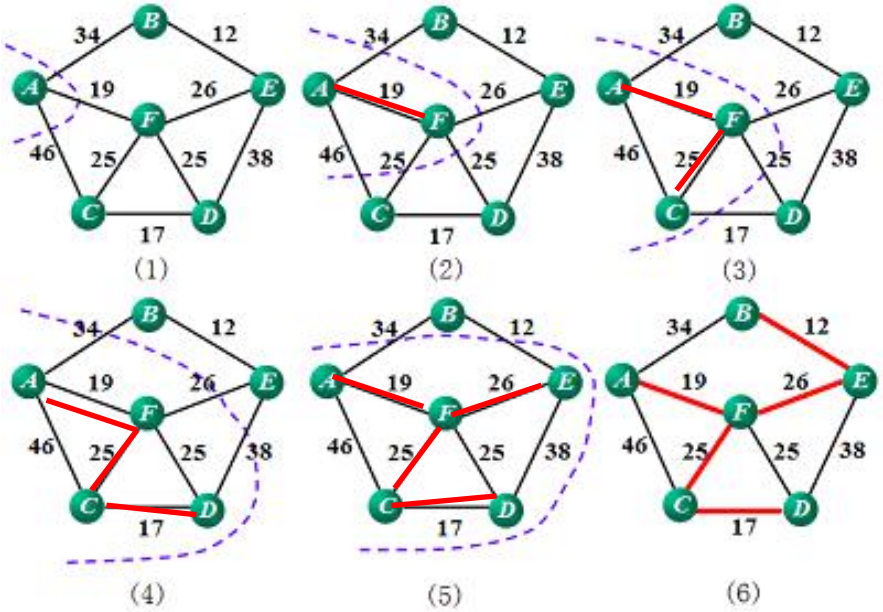
Divisive Hierarchical Clustering

Social Network Graphs



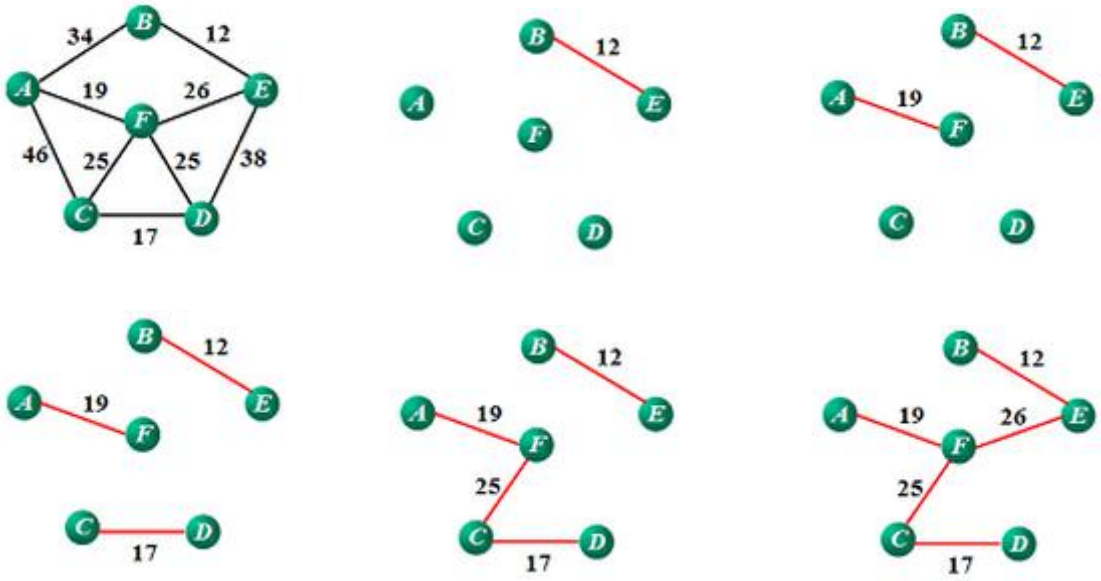
Divisive Hierarchical Clustering

- E.g., in a MST approach
- 构建最小生成树 (Minimum Spanning Tree)
  - Prime算法



Divisive Hierarchical Clustering

- E.g., in a MST approach
- 构建最小生成树 (Minimum Spanning Tree)
  - Kruskal 算法



Hierarchical Clustering: Strengths

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, ...)

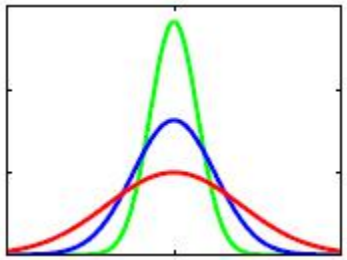
Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

Probabilistic Clustering

- Represent the probability distribution of the data as a *mixture model*
  - captures uncertainty in cluster assignments
  - gives model for data distribution
  - *Bayesian mixture model allows us to determine K*
- Consider mixtures of *Gaussians*

The Gaussian Distribution



- 单高斯模型(Gaussian Single Model)
  - 一个随机变量x服从高斯分布时，概率密度函数为：

$$\mathcal{N}(x|\mu,\sigma^2)=\frac{1}{(2\pi\sigma^2)^{1/2}}\exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

    - μ：模型均值，σ^2为模型方差

  - 多维变量x服从高斯分布时，概率密度函数为：

$$\mathcal{N}(\mathrm{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})=\frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}(\mathrm{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathrm{x}-\boldsymbol{\mu})\right\}$$

    - x是维度为D的列向量，μ：模型均值，Σ为D*D的协方差矩阵

似然函数

- 数据集：

$$D=\{x_i\},\ i=1,\ \ldots,\ N$$

- 考虑单个Gaussian模型
- 假设观测样本点由单个Gaussian独立等分布地抽样得到：

$$p(D|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu},\boldsymbol{\Sigma})$$

- 可以被看做模型参数的函数，因此被称为似然函数

极大似然估计

- 求解使得似然函数取最大值时对应的模型参数
- 等价地，极大化log似然函数：

$$\begin{aligned}\ln p(D|\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{Nd}{2}\ln(2\pi) \\ &\quad -\frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu})\end{aligned}$$

极大似然估计

- 相对于均值求似然最大化，得到样本均值：

$$\boldsymbol{\mu}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

- 相对于方差求似然最大化，得到样本方差：

$$\boldsymbol{\Sigma}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{\mathrm{ML}})(\mathbf{x}_n - \boldsymbol{\mu}_{\mathrm{ML}})^{\mathrm{T}}$$

Gaussian混合模型

- Gaussian Mixture Model (GMM)：多个Gaussian模型的线性混合：

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- 混合系数：

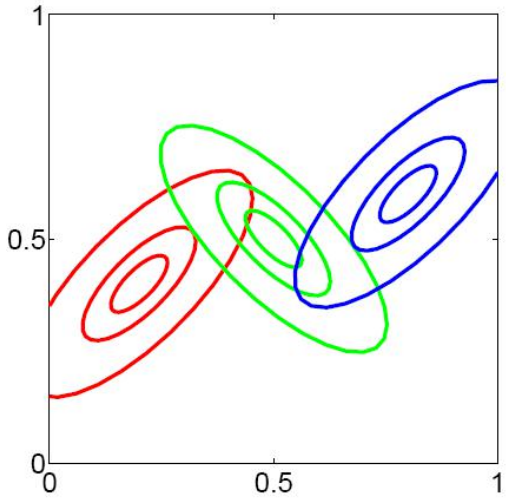
$$\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leqslant \pi_k \leqslant 1$$

- 可以看做一种先验概率：

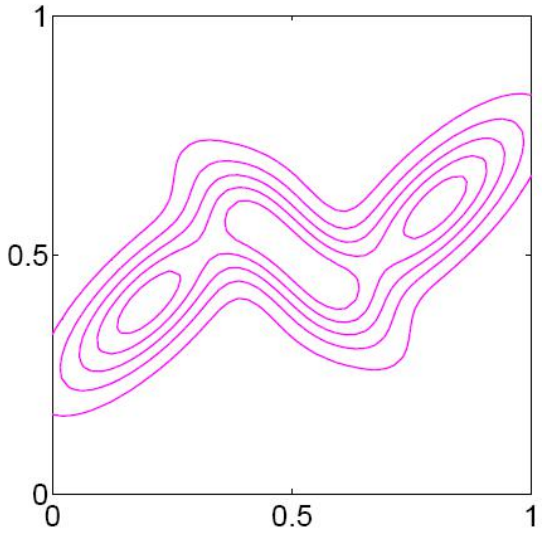
$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k)$$

举例

3个Gaussian的混合：

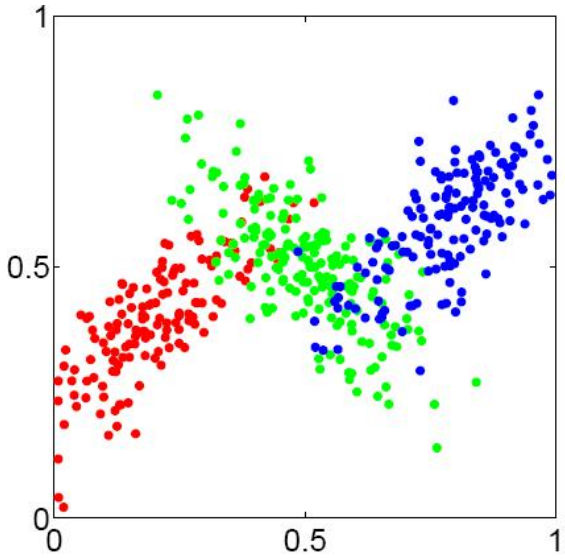


概率分布的等高线：



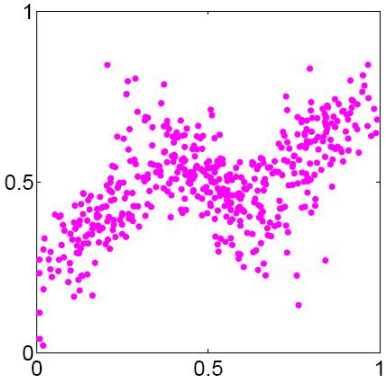
从Gaussian混合分布中抽样得到数据

- 样本点x_n的生成过程：
  - 首先，以概率π_k选择一个混合成分；
  - 接着，从该混合成分中抽样得到样本点x_n
- 对于每个样本点，重复以上两个步骤



从数据中估计Gaussian混合分布的参数

- 求解上述过程的逆过程 – 给定样本点，估计相应的
  - 混合系数
  - 均值
  - 协方差



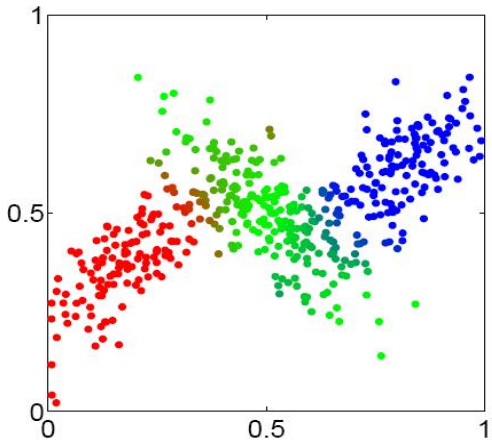
- 如果知道每个样本点由哪个成分抽样得到，则通过极大似然方法可以得到每个类对应的Gaussian模型的参数
- 问题：数据集缺少类别标注
- 因此类别labels可以看做是隐变量（latent/hidden variable）

后验概率

- 可以把混合系数看做每个成分的先验概率
- 给定一个类别标记k，可以估计相应的后验概率（posterior probabilities，或*responsibilities*）
- 可以通过Bayes' theorem得到：

$$\gamma_k(\mathbf{x}) \equiv p(k|\mathbf{x}) = \frac{p(k)p(\mathbf{x}|k)}{p(\mathbf{x})}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$



GMM的极大似然估计

- Log似然函数：

$$\ln p(D|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- 注意：sum出现在*log*内
- 没有封闭解！

GMM的极大似然估计

- 对log似然函数简单求导
- 令 ln $p(X|\pi,\mu,\Sigma)$ 相对于第k个Gaussian的均值μ_k 的倒数为零，得：

$$0 = -\sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

$$\longrightarrow \quad \mu_k = \frac{1}{N_k}\sum_{i=1}^{N} \gamma_k(x_i)x_i$$

$N_k = \sum_{i=1}^{N} \gamma_k(x_i)$ 类别 k 中所属的有效样本个数

GMM的极大似然估计

- 类似的，求解协方差：

$$\Sigma_k = \frac{1}{N_k}\sum_{i=1}^{N}\gamma_k(x_i)(x_i-\mu_k)(x_i-\mu_k)^T$$

- 根据Lagrange multiplier求解混合系数：

$$\pi_k = \frac{1}{N}\sum_{i=1}^{N}\gamma_k(x_i)$$

EM Algorithm

- 上述解构不成封闭形式，因为变量之间互为耦合
- 采用一种迭代的方式求解：
  - 给参数一个初始值
  - 通过下述两个步骤更新参数：
    - E-step: 估计后验概率或responsibilities
    - M-step: 根据MLE的结果更新参数

- 每一次 EM 循环都能保证likelihood值增大

EM Algorithm

- E-step: 估计responsibilities

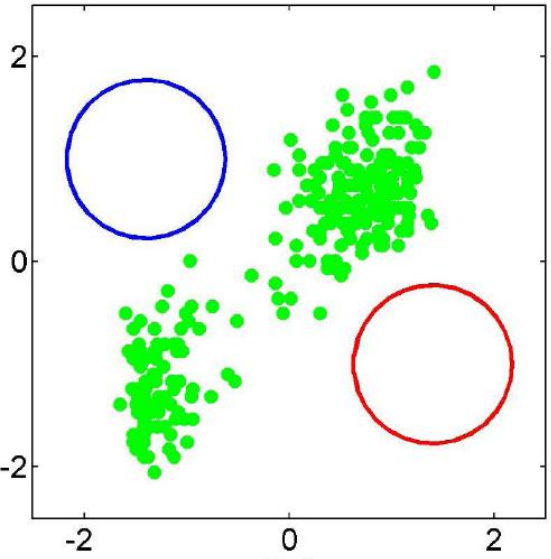
$$\gamma_k(x_i) = \frac{\pi_k N(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_i \mid \mu_j, \Sigma_j)}$$

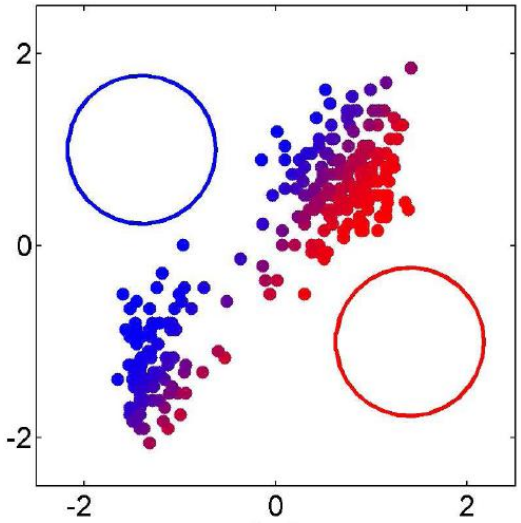
- M-step: 采用MLE估计更新参数

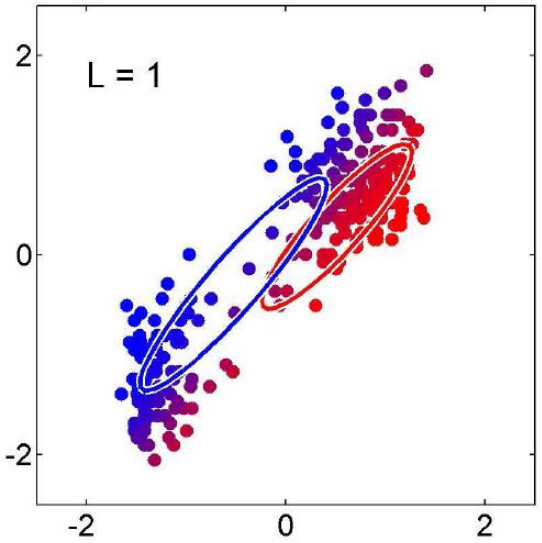
$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_k(x_i)$$

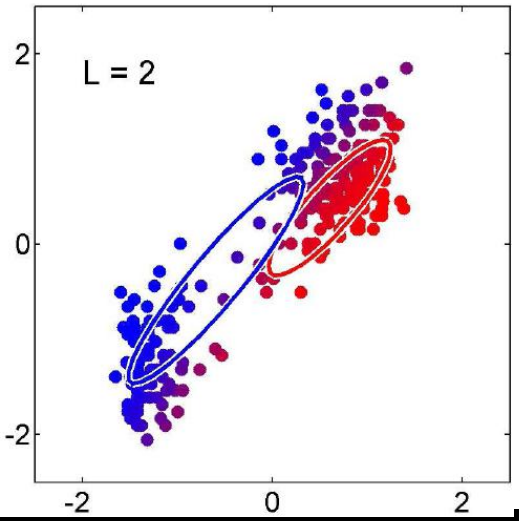
$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_k(x_i)(x_i - \mu_k)(x_i - \mu_k)^T$$

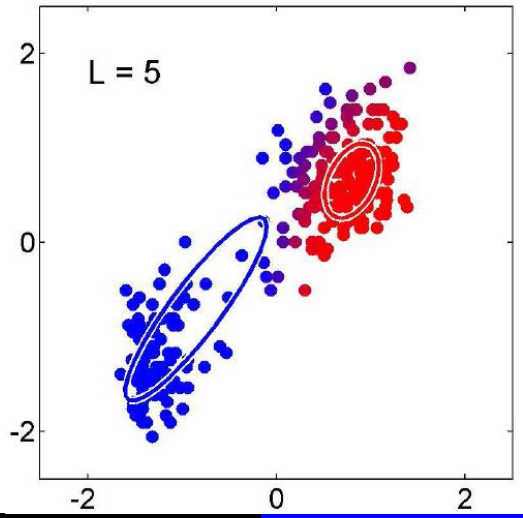
Example

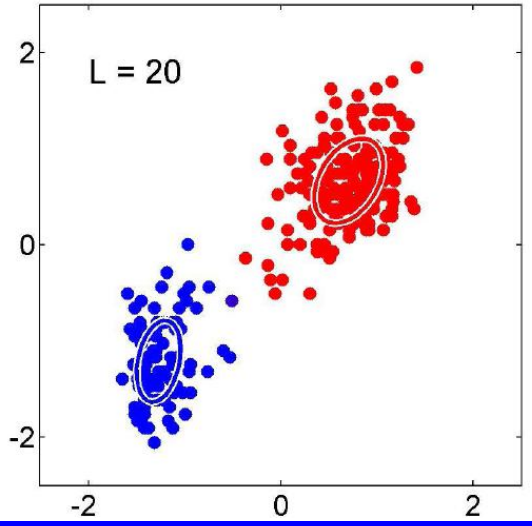












GMM应用于分类

- 分类时：
  - 每个混合成分（类）的参数μ和Σ已知
  - 把数据点x带入到每个混合成分C_k中

$$N(x|\mu_k,\Sigma_k)$$

  - 当概率大于一定阈值时便认为x属于C_k类

与 K-means之间的关系

- 考虑GMM的协方差为一个常数 ε
- 令极限 ε→0
- Responsibilities取两个值：

$$\gamma_i(\mathbf{x}_n) = \frac{\pi_i \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_i\|^2/2\epsilon\right\}}{\sum_j \pi_j \exp\left\{-\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2/2\epsilon\right\}} \to r_{ni} \in \{0, 1\}$$

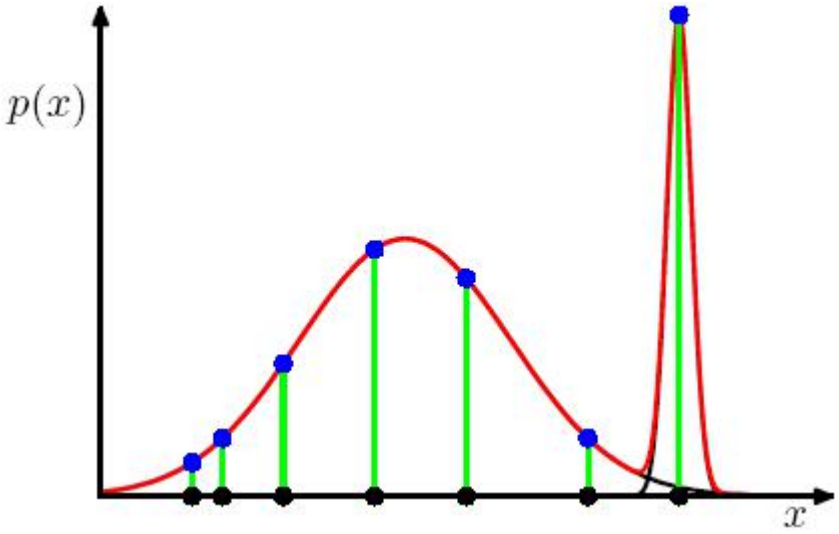
- 此时， EM algorithm与K-means等价

Other issues: Over-fitting in Gaussian Mixture Models

- 假设：某个混合成分仅包括一个样本点

$\Sigma_j=\sigma^2 I$, $\mu_j = x_n$

$$\mathcal{N}(\mathbf{x}_n|\mathbf{x}_n,\sigma_j^2\mathbf{I}) = \frac{1}{(2\pi)^{1/2}}\frac{1}{\sigma_j}$$



- 考虑情形$\sigma^2 \to 0$
- 此时log似然函数趋于无穷大，则极大化似然估计不是一个良态问题

Spectral Clustering

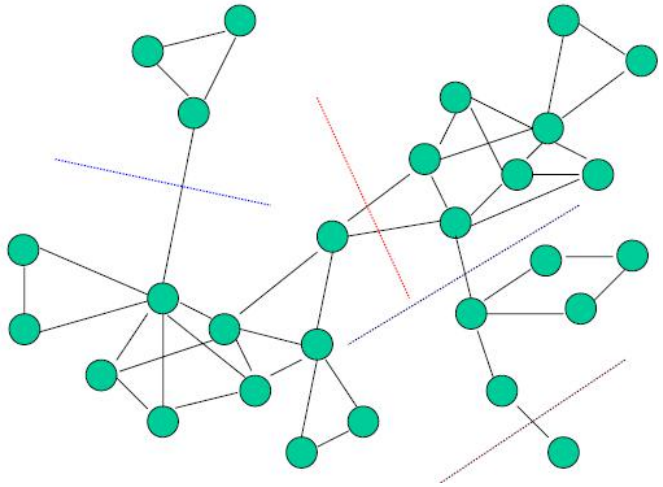
- Recall that the intuitive goal of clustering is
  - to divide the data points into several groups such that points in the same group are similar and points in different groups are dissimilar to each other

- Suppose the data points $\{x_1, ..., x_n\}$ are organized by similarity graph $G = (V, E, W)$
  - Each vertex v_i in this graph represents a data point x_i
  - Each edge e_{ij} is weighted by similarity s_{ij} (or w_{ij}) between v_i and v_j

- The graph construction depends on the application

Spectral Clustering

- The problem of clustering can now be reformulated using the similarity graph:
  - to find a partition of the graph such that the edges between different groups have very low weights and the edges within a group have high weights



Graph partitioning

- Clustering partitions the vertices of the graph.
- A good clustering places dissimilar vertices in different partitions.
- The loss function for a partition of $(A, \bar{A})$ (or the weighted adjacency of $(A,\bar{A})$) is given by the cut:

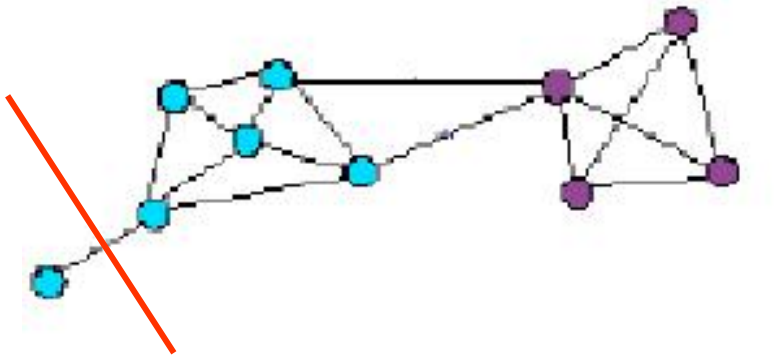
$$cut(A, \bar{A}) = \sum_{i \in A,\, j \in \bar{A}} w_{ij}$$

- For a given number *k* of subsets, the mincut approach simply consists in choosing a partition $A_1, \ldots, A_k$ which minimizes

$$cut(A_1, \ldots, A_k) = \sum_{i \in Ai,\, j \in \bar{A}i} cut(A_i, \bar{A}_i)$$

- Find a partition that minimizes the cut (Mincut criterion) -- **HOW**?

- In many cases, the solution of mincut simply separates one individual vertex from the rest of the graph.



e.g., 2-way Partitioning...

- Of course this is not what we want to achieve in clustering
- Explicitly request that the sets A_1, ..., A_k are "reasonably large"

Two ways of measuring the "size" of a subset A:
- |A| := the number of vertices in A
- vol(A) :=∑di, for all i in A

Graph partitioning

- A loss function that favors such clusters is Normalized cut

$$Ncut = \frac{1}{2}\sum_{i=1}^{k}\frac{W(A_i, \overline{A}_i)}{vol(A_i)} = \sum_{i=1}^{k}\frac{cut(A_i, \overline{A}_i)}{vol(A_i)}$$

  - vol(A) = $\sum_{i \in A} d_i$, $d_i = \sum_{j=1,\ldots n} w_{ij}$

- **A good partition** should separate dissimilar vertices and should produce balanced clusters
- Minimizing normalized cut is NP-hard.
- One way of approximately optimizing normalized cuts leads to spectral clustering

2-way Spectral Graph Partitioning

- Partition membership indicator:

$$q_i = \begin{cases} 1 & \text{if } i \in A \\ -1 & \text{if } i \in B \end{cases}$$

$$J = CutSize = \frac{1}{4}\sum_{i,j} w_{ij}[q_i - q_j]^2$$

$$= \frac{1}{4}\sum_{i,j} w_{ij}[q_i^2 + q_j^2 - 2q_i q_j] = \frac{1}{2}\sum_{i,j} q_i[d_i\delta_{ij} - w_{ij}]q_j$$

$$= \frac{1}{2} q^T (D - W) q$$

$$D = \text{diag}(d_1, \ldots, d_n),\ d_i = \sum_{j=1,\ldots n} w_{ij}$$

- Relax indicators q_i from discrete values to continuous values, the solution for *min* $J(q)$ is given by the eigenvectors of

$$(D\text{-}W)q = \lambda q$$

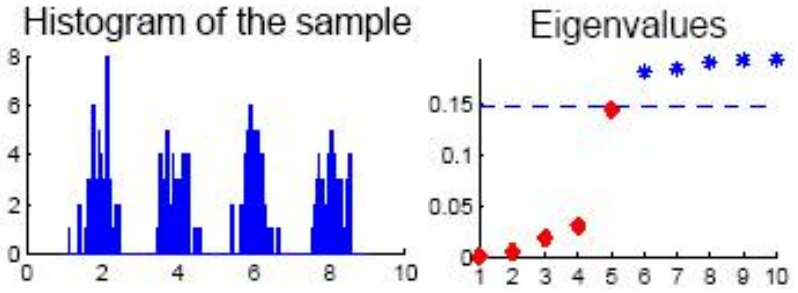
Properties of Graph Laplacian (Proof omitted)

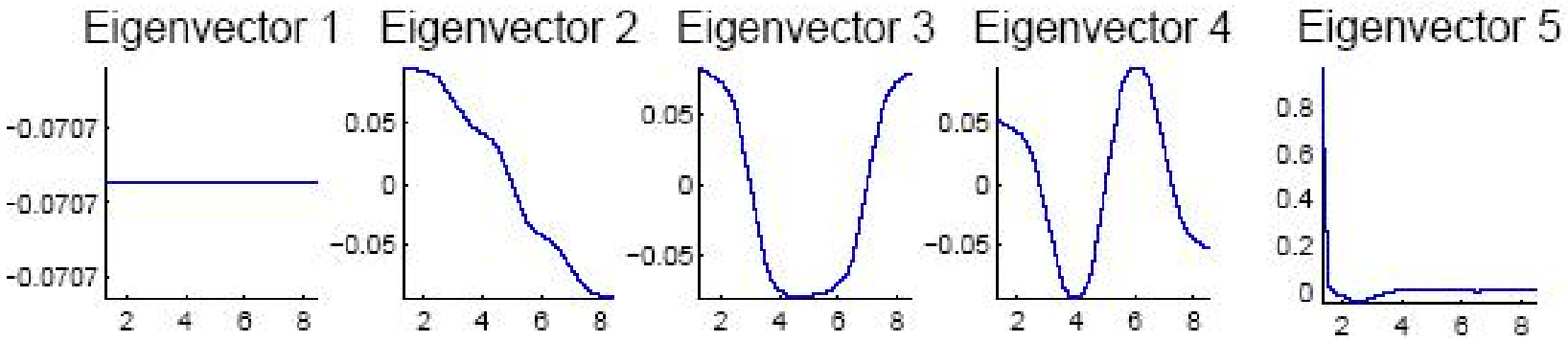
- Laplacian matrix of the Graph: $L = D-W$
- L is semi-positive definite $x^T Lx \geq 0$ for any x.
- Assume eigenvalues will always be ordered increasingly
  - First eigenvector is $q_1=(1,...,1)^T$ with $\lambda_1=0$.
  - The eigenvectors with small eigenvalues provide important information for clustering
    - Second eigenvector q_2 can serve as the desired solution
    - Higher eigenvectors are also useful

Recovering Partitions

- From the definition of cluster indicators, partitions A, B are determined by:

$$A = \{i \mid q_2(i) < 0\},\ B = \{i \mid q_2(i) \geq 0\}$$





Spectral Clustering

- In most common view:
  - Build a weighted graph G = (V, E, W).
  - Construct a Laplacian matrix L=f (W) (different variants of spectral clustering result from different functions f).
  - Compute the eigenvectors of k smallest eigenvalues of L. These provide a new representation of the original data points.
  - Cluster the points in this new representation (e.g. using k-means).

Spectral Clustering

Spectral clustering algorithm

Input: Similarity matrix $S \in R^{n \times n}$, number k of clusters to construct.

1: Construct a similarity graph G=(V, E). Let W be its weighted adjacency matrix.

2: Compute the Laplacian matrix L.

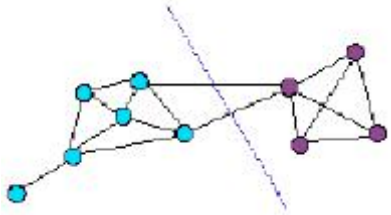
3: Compute the k eigenvectors $u_1, \ldots, u_k$ with k smallest eigenvalues of L.

3: Let $U \in R^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.

4: For i = 1, ..., n, let $y_i \in R^k$ be the vector corresponding to the i-th row of U. (for i = 1, ..., n, x_i can be represented by $y_i = [U_{i1}, \ldots, U_{ik}]$).

5: Cluster the points $(y_i)_{i=1, \ldots, n}$ in R^k with the k-means algorithm into clusters $C_1, \ldots, C_k$.

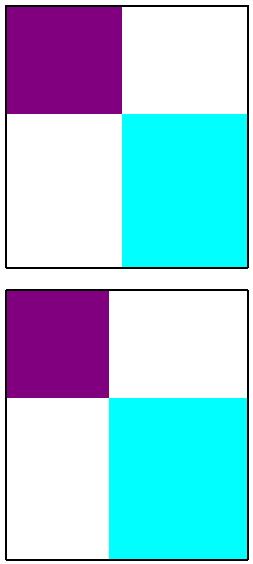
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.



similarity matrix
W

transition matrix
L= D-W

first k eigenalue of L
$l_1, \ldots, l_k$



$l_1, \ldots, l_k$

cosine similarity

normalize rows

spectral mapping

clustering in R^k

Summary

- Clustering is cool
- It's easy to find the most salient pattern
- It's quite hard to find the pattern you want
- It's hard to know how to fix when broken
- EM is a useful optimization technique you should understand well if you don't already