北京交通大学
BEIJING JIAOTONG UNIVERSITY




《大数据概论》
大数据分析与挖掘

鲍鹏
软件学院

目录

- 数据理解与特征工程
- 常用数据挖掘算法
  - 无监督学习
    - 关联规则挖掘
    - 聚类分析
  - 监督学习
- 高级数据建模技术
- 数据可视化技术

机器学习

- 机器学习：通过算法构建模型并对模型进行评估，性能若达到要求则利用该模型来测试其它数据，若达不到要求则调整算法重新建立模型，再次进行评估，循环往复，最终获得满意的模型来处理其它数据。

- 根据训练数据是否含有标签，机器学习可分为监督学习、无监督学习和半监督学习。

机器学习

- 开源机器学习框架—— Tensorflow
  - TensorFlow是谷歌基于C++开发的第二代机器学习系统。
  - 开发目的是用于进行机器学习和深度神经网络的研究与应用。
  - Google App的语音识别、Gmail的自动回复功能、Google Photos的图片搜索等都在使用 TensorFlow 。
  - GitHub项目地址：https://github.com/tensorflow/tensorflow

机器学习

- 开源机器学习框架——Scikit-Learn
  - Scikit-Learn是用于机器学习的Python模块，它建立在SciPy之上。
  - 基本功能主要分为六个部分：分类、回归、聚类、数据降维、模型选择、数据预处理。
  - GitHub项目地址：
    https://github.com/scikit-learn/scikit-learn

Caffe

- 开源机器学习框架—— Caffe
  - Caffe是一个兼具表达性、速度和思维模块化的深度学习框架。
  - Caffe是一个基于C++架构的框架，开发者能够利用它自由的组织网络，目前支持卷积神经网络和全连接神经网络（人工神经网络）。
  - 在Linux上，C++可以通过命令行来操作接口，运算上支持CPU和GPU直接无缝切换。
  - GitHub项目地址：

    https://github.com/BVLC/caffe

机器学习

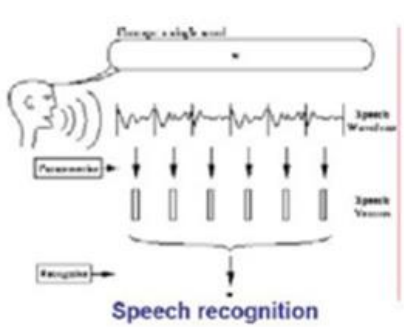
keras makes Deep Learning simple

- 开源机器学习框架—— Keras
  - Keras是基于Python开发的极其精简并高度模块化的神经网络库。
  - Keras在TensorFlow或 Theano上都能够运行，是一个高度模块化的神经网络库，支持GPU和CPU运算。
  - Keras侧重于开发快速实验，用可能最少延迟实现从理念到结果的转变。
  - GitHub项目地址：

    https://github.com/fchollet/keras

机器学习

- 机器学习已广泛应用于数据挖掘、计算机视觉、自然语言处理、生物特征识别、搜索引擎、医学诊断、检测信用卡欺诈、证券市场分析、DNA序列测序、语音和手写识别、战略游戏和机器人等领域。



无监督学习

- 监督学习和无监督学习的区别：训练集的数据是否被标注。
- 在无监督学习中，数据并不被特别标识，学习模型的目标为推断数据的一些内在结构。现实世界中，大多数样本的标签信息未知，无监督学习能够挖掘出无标记数据之间的内在联系并加以利用。
- 常见应用场景包括关联规则的学习以及聚类等。
- 常见算法包括**Apriori**算法和**k-means**算法。

无监督学习-关联规则挖掘

- 关联规则挖掘的基本概念
- 关联规则挖掘的工作过程
- 关联规则的Apriori算法

无监督学习-关联规则挖掘

- 关联规则反映事物之间的相互依存性和关联性。若两个或者多个事物之间存在一定的关联关系，则其中一个事物能够从其它事物中推断出来。

- 典型的关联规则问题是对超市中的货篮数据（**Market Basket**）进行分析。通过发现顾客放入货篮中的不同商品之间的关系来分析顾客的购买习惯。

无监督学习-关联规则挖掘

应用市场：市场货篮分析、交叉销售（Crossing Sale）、部分分类（Partial Classification）、金融服务（Financial Service），以及通信、互联网、电子商务 ……

无监督学习-关联规则挖掘

- 关联规则
  - 由Agrawal等人在1993年的SIGMOD会议上提出。
  - 在事务、关系数据库中的项集和对象中发现频繁模式、关联规则、相关性或者因果结构。
  - 频繁模式: 数据库中频繁出现的项集。

- 目的: 发现数据中的规律
  - 超市中的什么产品会一起购买？— 啤酒和尿布
  - 在买了一台PC之后下一步会购买何种产品?
  - 哪种DNA对这种药物敏感?

无监督学习-关联规则挖掘

- 关联规则（Association rule）
  - 指从事务数据库、关系数据库和其他信息存储中的大量数据的项集之间发现有趣的、频繁出现的模式、关联和相关性。关联规则是支持度和信任度分别满足用户给定阈值的规则。

- 关联分析（Association analysis）
  - 用于发现隐藏在大型数据集中的令人感兴趣的联系。所发现的联系可以用关联规则或者频繁项集的形式表示。关联规则挖掘指从大量的数据中挖掘出描述数据项之间相互联系的有价值的有关知识。

关联规则基本概念

- 项、项集、k-项集与事务
  - 项（Item）：指数据库中不可分割的最小单位。
  - 项集（Itemset）：指多个项的集合，空集是指不包含任何项的项集。
  - k-项集：指由k个项构成的项集组合。
  - 事务：指用户定义的一个数据库操作序列，这些操作序列是一个不可分割的工作单位。
- 频繁项集（Frequent Itemset）
  - 指在所有训练元组中同时出现的次数，超过人工定义的阈值的项集。在关联规则的挖掘过程中，一般只保留候选项集中满足支持度条件的项集。

关联规则基本概念

- 极大频繁项集（Frequent Large Itemset）
  - 指不存在包含当前频繁项集的频繁超集，则当前频繁项集就是极大频繁项集。
- 支持度（Support）
  - 指项集在所有训练元组中同时出现的次数，支持度可表述为：

$$Support(X \rightarrow Y) = |X \cup Y| / |N|$$

其中，$X \cap Y = \Phi$，$|X \cup Y|$表示集合X与Y在一个事务中同时出现的次数，$|N|$表示数据记录。

关联规则基本概念

- 置信度（Confidence）
  - 表示规则在训练数据集中的可信程度。
  - 置信度可表述为：

$$\text{Confidence}(X \rightarrow Y) = |X \cup Y| / |X|$$

$$= \text{Support}(X \rightarrow Y) / \text{Support}(X)$$

其中，$X \cap Y = \Phi$，$|X \cup Y|$表示集合X与Y在一个事务中同时出现的次数，$|X|$ 表示X出现的总次数，$\text{Support}(X \rightarrow Y)$ 表示规则的支持度。

关联规则挖掘

- 关联规则是形如$X \Rightarrow Y$的逻辑蕴含式，其中$X \subset I$，$Y \subset I$，且 $X \cap Y=\varnothing$。如果事务数据库D中有$s\%$的事务包含$X \cup Y$，则称关联规则$X \Rightarrow Y$的支持度为$s\%$。
- 支持度是概率值，置信度为条件概率值。若项集X的支持度记为$support\ (X)$，规则的置信度为$support\ (X \cup Y) / support\ (X)$。

$$support\ (X \Rightarrow Y)=P\ (X \cup Y)$$

$$confidence\ (X \Rightarrow Y)=P\ (Y \mid X)$$

给定事务的集合T，关联规则发现是指找出支持度大于等于minsup，并且置信度大于等于minconf的所有规则，minsup和minconf是预设的支持度和置信度阈值。

关联规则挖掘

Min. support 50%
Min. confidence 50%

交易ID	购买的商品
2000	A, B, C
1000	A, C
4000	A, D
5000	B, E, F

频繁模式	支持度
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

- 针对关联规则 $A \Rightarrow C$:
  - **support = support({A }∪{C}) = 50%**
  - **confidence = support({A }∪{C}) / support({A}) = 66.7%**

关联规则挖掘

- 关联规则挖掘问题可划分为两个子问题：

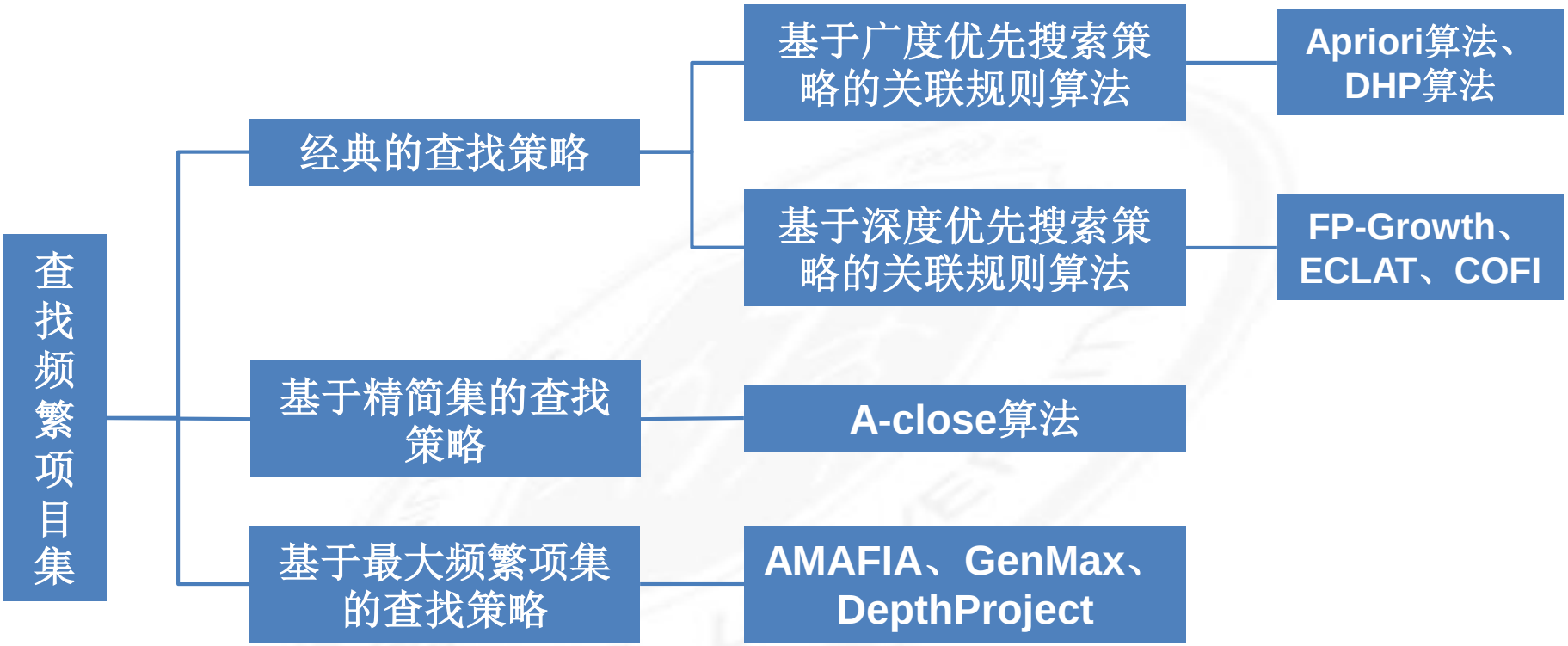
（1）发现频繁项目集

通过用户给定的Min. support，寻找所有频繁项目集，即满足Support不小于Min. support的项目集。这些频繁项目集可能具有包含关系，因此只考虑不被其它频繁项目集所包含的极大频繁项集。极大频繁项集是形成关联规则的基础。

（2）生成关联规则

通过用户给定的Min. confidence，在每个最大频繁项集，寻找Confidence不小于Min. confidence的关联规则。

关联规则挖掘


查找频繁项目集
经典的查找策略
基于精简集的查找策略
基于最大频繁项集的查找策略
基于广度优先搜索策略的关联规则算法
基于深度优先搜索策略的关联规则算法
Apriori算法、DHP算法
FP-Growth、ECLAT、COFI
A-close算法
AMAFIA、GenMax、DepthProject

强关联规则

- 生成关联规则
  - 生成关联规则指通过用户给定的最小置信度，在每个最大频繁项集中，寻找可信度不小于Min. confidence的关联规则。
  - 得到频繁项目集之后，则需要从中找出符合条件的关联规则。
  - 最简单的办法是：遍历所有的频繁项目集，然后从每个项目集中依次取1、2、...k个元素作为后件，该项目集中的其他元素作为前件，计算该规则的置信度进行筛选。
  - 缺点：穷举方式效率很低。

关联规则挖掘—Apriori算法

- **Apriori算法**
  - **Apriori算法是关联规则模型中的经典算法。**
  - **Apriori算法基于频繁项集性质的先验知识，使用由下至上逐层搜索的迭代方法，即从频繁1项集开始，采用频繁k项集搜索频繁k+1项集，直到不能找到包含更多项的频繁项集为止。**

关联规则挖掘—Apriori算法

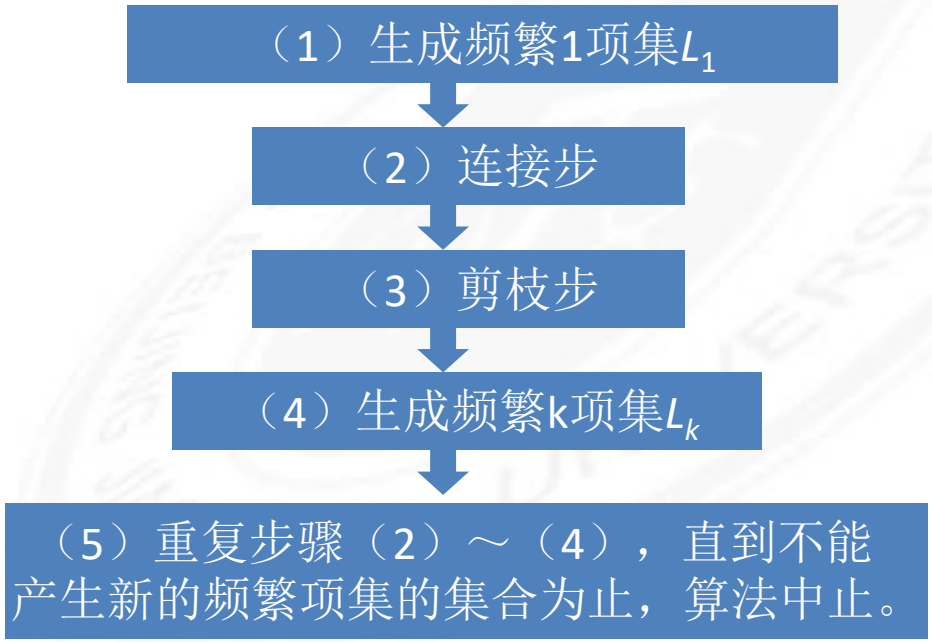
- **Apriori算法的性质**
  - 利用性质1，通过已知的频繁项集构成长度更大的项集，并将其称为潜在频繁项集。潜在频繁k项集的集合C_k是指由有可能成为频繁k项集的项集组成的集合。
  - Apriori算法只需计算潜在频繁项集的支持度，而不必计算所有不同项集的支持度，在一定程度上减少了计算量。

性质1 频繁项集的子集必为频繁项集。

性质2 非频繁项集的超集一定是非频繁的。

关联规则挖掘—Apriori算法

- **Apriori算法的工作步骤**
  - 核心步骤是连接步和剪枝步。



可能产生庞大的候选项集

关联规则挖掘—Apriori算法

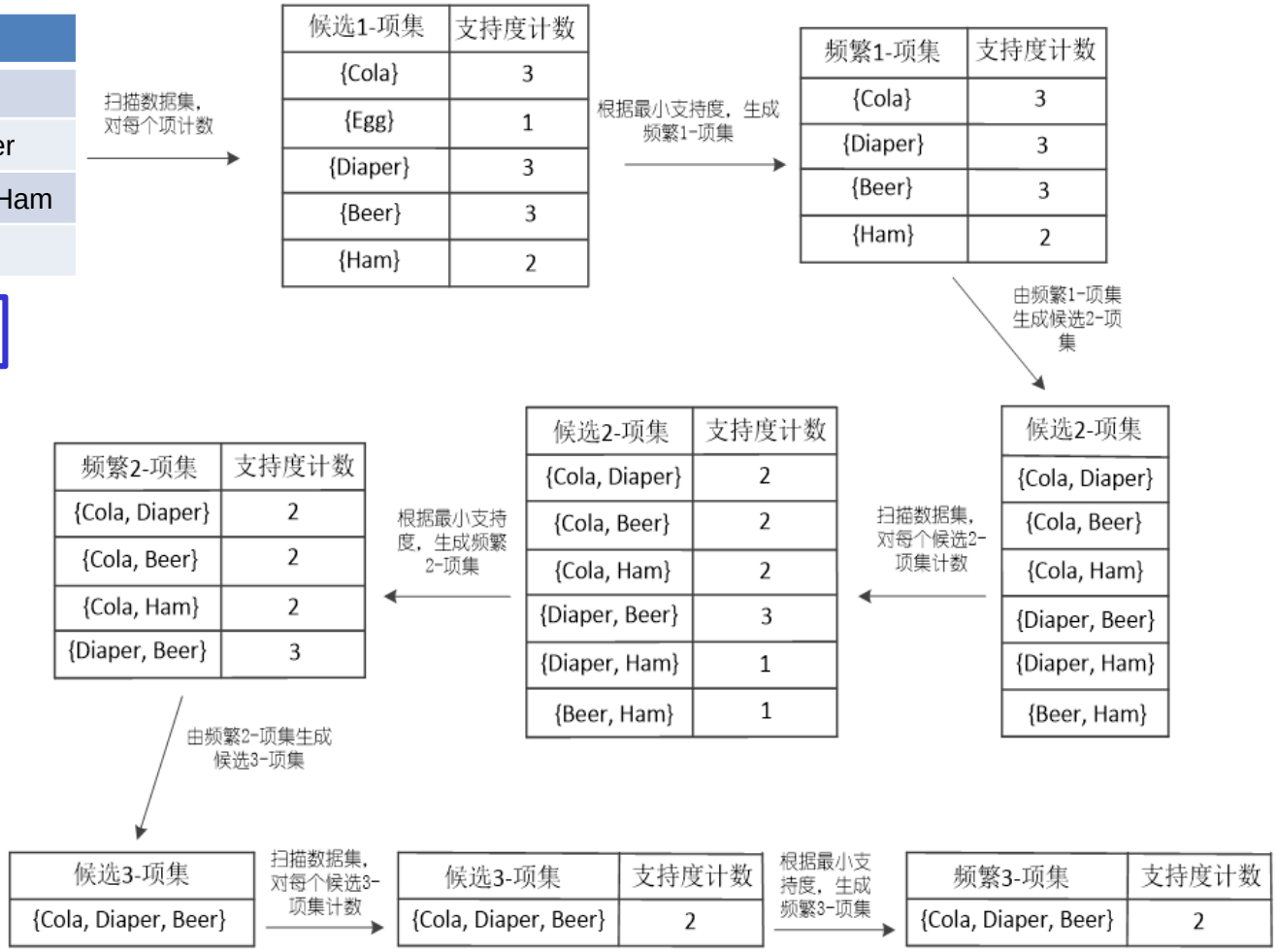
- 连接步
  - 为寻找频繁k项集L_k，将L_{k-1}与自身连接产生候选k项集C_k。L_{k-1}的元素是可连接的。

- 剪枝步
  - 候选k项集C_k是L_k的超集。C_k成员可为频繁项集也可不是频繁的，但所有的频繁项集均包括在C_k中。扫描数据库，确定C_k中每一个候选的计数，根据计数值大于最小支持度的原则确定L_k。
  - 根据性质2，若一个候选k项集的(k-1)项集不在L_k中，则该候选项不可能是频繁的，从而可以从C_k中删除。

关联规则挖掘—Apriori算法

ID	购买的商品
001	Cola, Egg, Ham
002	Cola, Diaper, Beer
003	Cola, Diaper, Beer, Ham
004	Diaper, Beer

支持度计数 ≥ 2



频繁项集的挖掘过程

关联规则挖掘—Apriori算法

- Apriori算法
  - 缺点：候选频繁K项集数量巨大。在验证候选频繁K项集的时候，需要对整个数据库进行扫描，非常耗时。
- Apriori算法的改进方向
  - 通过减少扫描数据库的次数改进I/O的性能；
  - 改进产生频繁项集的计算性能；
  - 寻找有效的并行关联规则算法；
  - 引入抽样技术改进生成频繁项集的I/O和计算性能；
  - 扩展应用领域。比如展开定量关联规则、泛化关联规则及周期性的关联规则的研究。

无监督学习—聚类分析

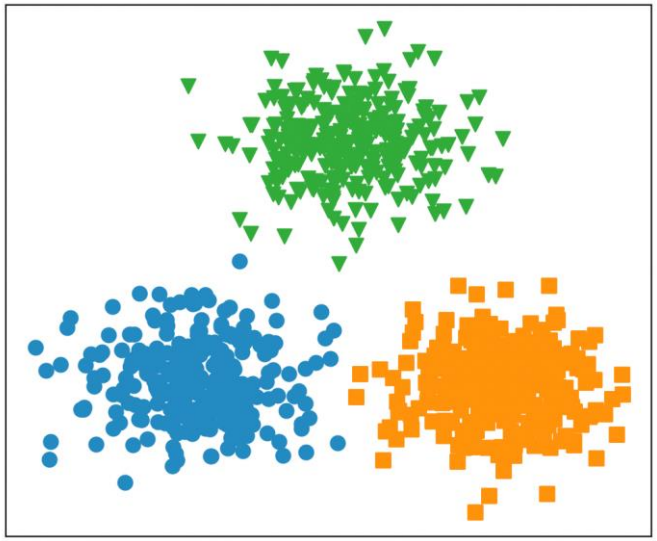
- 聚类分析
- k-means算法原理
- k-means算法求解
- 聚类评估指标

聚类分析

- 聚类分析是数据分析中的一种重要技术，其应用领域极其广泛，如数据挖掘、统计学、机器学习、模式识别、生物学、电子商务等。
- 聚类（Clustering）是对物理的或抽象的对象集合分组的过程。
- 聚类生成的组称为簇（Cluster），簇是数据对象的集合。簇内部的任意两个对象之间具有较高的相似度，而属于不同簇的两个对象间具有较高的相异度。相异度可根据对象的属性值计算，对象间的距离是最常采用的度量指标。

聚类算法思想

- 聚类算法的核心思想是将具有相似特征的事物聚合成组。
  - 如图所示为3种类别的数据样本，其中每一种形状都表示一个类别。聚类算法的目的是将各个类别的样本点分开，将同一种类别的样本点聚在一起。



k-means算法原理

- k-means聚类算法也称k均值聚类。
- 算法的核心思想是通过迭代把数据对象划分到不同的簇中，以求目标函数最小化，从而使生成的簇尽可能地紧凑和独立。

k-means算法原理

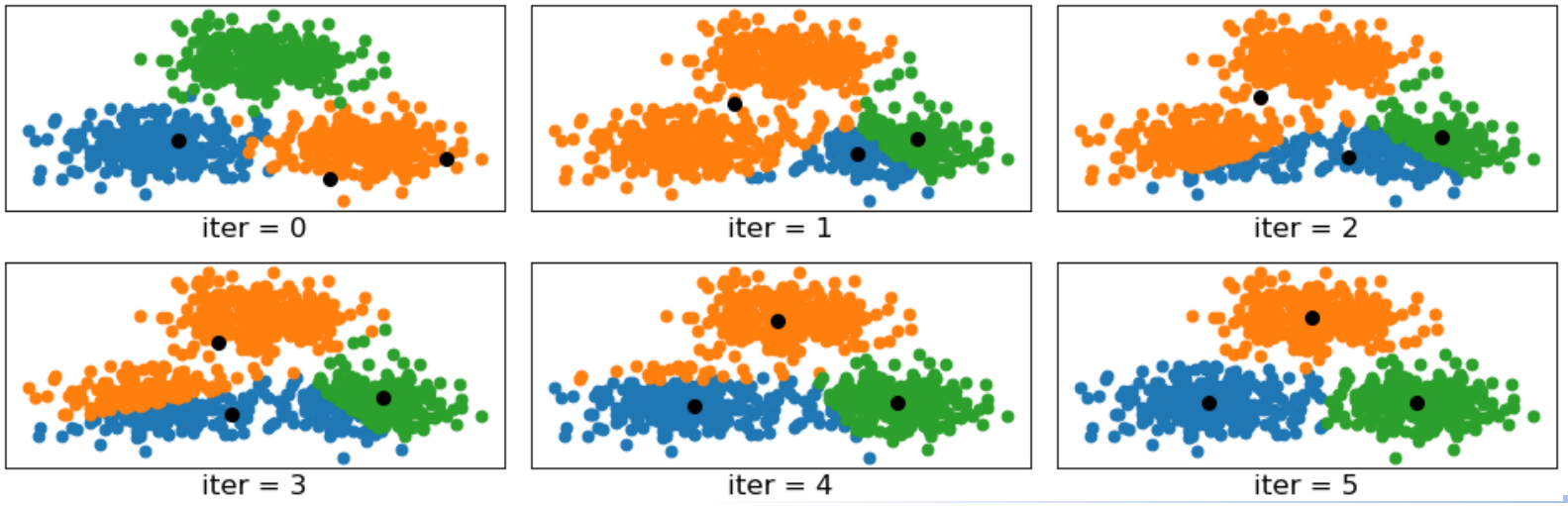
- k-means算法的主要步骤如下：
  - ①首先随机选择k个样本点作为k个簇的初始簇中心；
  - ②然后计算每个样本点与这个k个簇中心的相似度大小，并将该样本点划分到与之相似度最大的簇中心所对应的簇中；
  - ③根据每个簇中现有的样本，重新计算每个簇的簇中心；
  - ④循环迭代步骤②③，直到目标函数收敛，即簇中心不再发生变化。

k-means算法原理

- **k-means算法描述：**
  - 输入：期望得到的簇的数目k，n个对象的数据库。
  - 输出：使得平方误差准则函数最小化的k个簇。
  - 方法：
    - 选择k个对象作为初始的簇的质心；
    - **repeat**
      - 计算对象与各个簇的质心的距离，将对象划分到距离其最近的簇；
      - 重新计算每个新簇的均值；
    - **Until** 簇的质心不再变化。

k-means算法原理

- 如图所示为k-means聚类过程。在图中iter=0表示聚类过程尚未开始，即正确标签下的可视化结果，其中黑色圆点为随机初始化的3个簇中心。
- 聚类算法只能发现哪些样本点属于同一个类别（同一个簇），但无法发现每个簇属于何种类别。

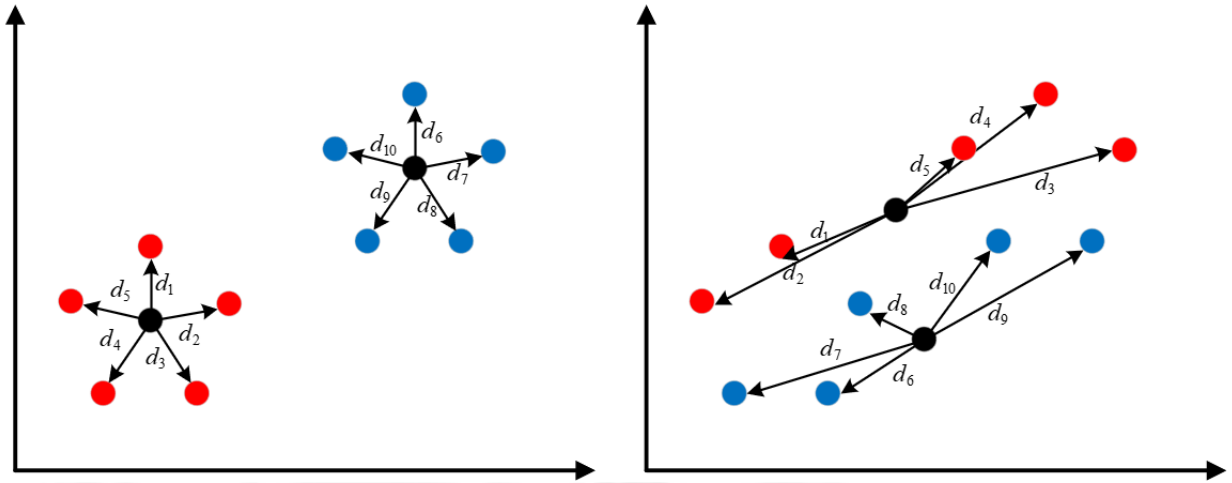


k-means算法求解—目标函数

- 最常见的衡量方法为计算两个样本间的欧式距离，当两个样本点离得越近就代表着两者间的相似度越高。
- 根据上述准则可以将k-means聚类算法的目标函数定义为所有样本点到其对应簇中心距离的总和。

k-means算法求解—目标函数

- 通过最小化目标函数$d = d_1 + d_2 + \cdots + d_{10}$来得到最优解。



如图所示，左右两边为同一数据集的两种不同聚类结果，其中同种颜色表示聚类后被划分到了同一个簇中，黑色圆点为聚类后的簇中心。从可视化结果来看，左图的聚类结果好于右图的聚类结果。

k-means算法求解—目标函数

- 目标函数
  - 设$X=\{X_1,X_2,\ldots,X_n\}$为一个含有n个样本的数据集，其中第i个样本表示为$X_i=\{x_{i1},x_{i2},\ldots,x_{im}\}$， m为样本特征的数量。样本分配矩阵U是一个$n\times k$的**0-1**矩阵，u_{ip}表示第i个样本被分到第p个簇中。$Z=\{Z_1,Z_2,\ldots,Z_k\}$为k个簇中心向量，其中$Z_p=\{z_{pq},z_{p2},\ldots,z_{pm}\}$为第$p$个簇中心。则**k-means**聚类算法的目标函数可写为：

$$P(U,Z)=\sum_{p=1}^{k}\sum_{i=1}^{n}u_{ip}\sum_{j=1}^{m}(x_{ij}-z_{pj})^2$$

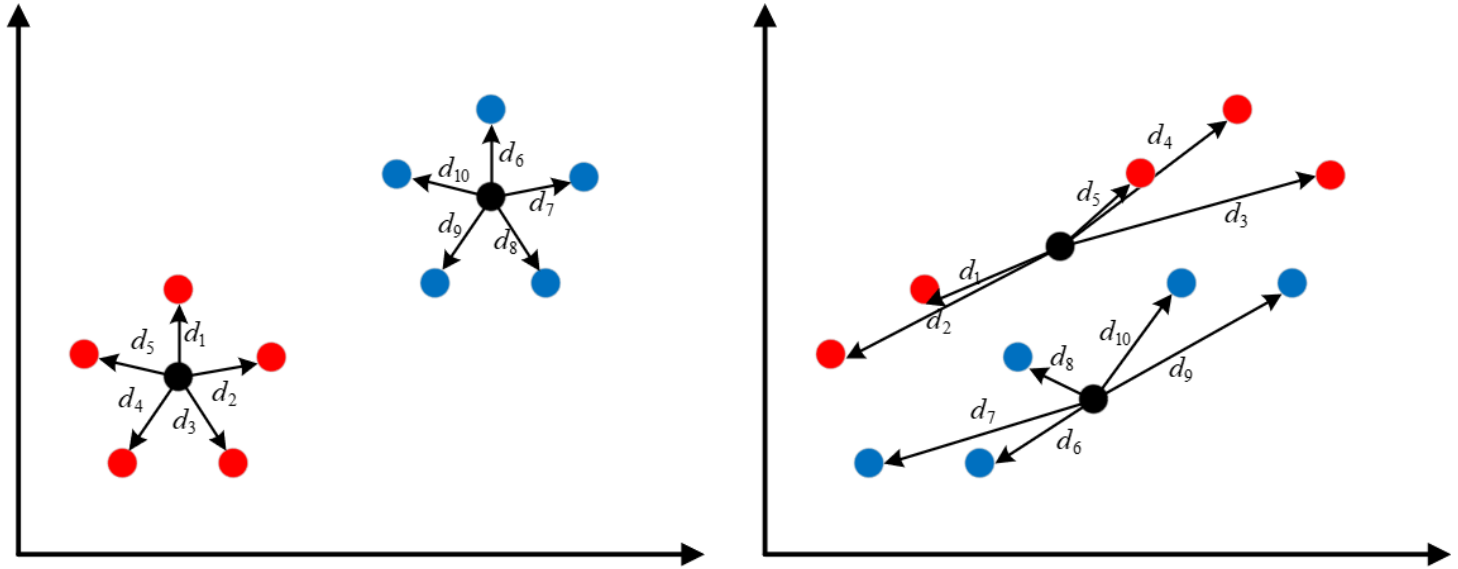
服从约束条件：

$$\sum_{p=1}^{k}u_{ip}=1$$

k-means算法求解—目标函数

- 由下图（左）可得，其对应的簇分配矩阵为

$$U_{[10\times 2]}=\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1\end{bmatrix}^T$$



k-means算法求解—簇中心矩阵Z

- 由目标函数可得，需求解的未知参数包括两个：簇中心矩阵 **Z** 和簇分配矩阵**U**。

$$P(U,Z)=\sum_{p=1}^{k}\sum_{i=1}^{n}u_{ip}\sum_{j=1}^{m}(x_{ij}-z_{pj})^2$$

- 针对目标函数，关于变量z_{pj}求导可得：

$$\frac{\partial P(U,Z)}{\partial z_{pj}}=-2\sum_{i=1}^{n}u_{ip}(x_{ij}-z_{pj})$$

- 进一步，令上式为 **0** 的条件：

$$\sum_{i=1}^{n}u_{ip}(x_{ij}-z_{pj})=0 \Rightarrow \sum_{i=1}^{n}u_{ip}x_{ij}=\sum_{i=1}^{n}u_{ip}z_{pj} \Rightarrow z_{pj}=\frac{\sum_{i=1}^{n}u_{ip}x_{ij}}{\sum_{i=1}^{n}u_{ip}}$$

k-means算法求解—簇中心矩阵Z

- 簇中心的计算公式：

$$\sum_{i=1}^{n} u_{ip}(x_{ij} - z_{pj}) = 0 \Rightarrow \sum_{i=1}^{n} u_{ip}x_{ij} = \sum_{i=1}^{n} u_{ip}z_{pj} \Rightarrow z_{pj} = \frac{\sum_{i=1}^{n} u_{ip}x_{ij}}{\sum_{i=1}^{n} u_{ip}}$$

- 若某个簇种有 3 个样本点[1, 2]，[2, 3]，[4, 6]，则其簇中心为：1/3[(1+2+4), (2+3+6)] 。

k-means算法求解—簇分配矩阵U

- 对每个样本点而言，只需分别计算其与k个簇中心的距离（相似度），然后将其划分到与之相似度最高（距离最近）的簇中，计算公式如下：

$$u_{ip}=\begin{cases}1, & \sum_{j=1}^{m}(x_{ij}-z_{pj})^2 \le \sum_{j=1}^{m}(x_{ij}-z_{tj})^2, \text{for } 1\le t\le k \\ 0, & \text{otherwise}\end{cases}$$

- 上式中，计算每个样本点到所有簇中心的距离，然后将其划分到离它最近的簇中。例如，某个样本点到3个簇中心的距离分别是5，2，8，则簇分配矩阵对应行则为[0, 1, 0]。

聚类评估指标

- 一个好的聚类方法要能产生高质量的聚类结果（簇），簇要具备以下两个特点：
  - 高的簇内相似性
  - 低的簇间相似性

- 聚类结果的好坏取决于该聚类方法采用的相似性评估方法以及该方法的具体实现。
- 聚类方法的好坏还取决于该方法是否能够发现某些或所有的隐含模式。

聚类评估指标

- 在聚类任务中，常见的评价指标有：纯度（Purity）、兰德系数（Rand Index, RI）、F值（F-score）和调整兰德系数（Adjusted Rand Index, ARI）。
- 纯度的计算公式定义为：

$$P=(\Omega,\mathbb{C})=\frac{1}{N}\sum_{k}\max_{j}|\omega_k\cap c_j|$$

其中N表示总的样本数；$\Omega=\{\omega_1,\omega_2,\ldots,\omega_K\}$表示聚类后的所有簇，而$\mathbb{C}=\{c_1,c_2,\ldots,c_J\}$表示正确的类别；$\omega_k$表示聚类后第$k$个簇中的所有样本，$c_j$表示第$j$个类别的真实样本数。$P$的取值范围为[0,1]，$P$的值越大表示聚类效果越好。