第9课时 数据预处理、降维、特征提取及聚类

主讲教师：欧新宇

February 8, 2020

Outlines

- 数据预处理
  - StandardScaler
  - MinMaxScaler
  - MaxAbsScaler
  - RobustScaler
  - Normalizer
  - Binarizer
- 数据降维
  - PCA主成分分析
- 特征提取
  - PCA特征提取
  - NMF非负矩阵分解
- 聚类
  - K-Means
  - 凝聚聚类
  - DBSCAN

数据预处理

场景一：

我们正在通过一台高清电视而非现场看世界杯足球赛，在电视的纯平显示器上有一个漂亮的足球。在4K的显示器上大概包含了880万像素，而足球则可能是由较少的像素组成，比如说只有1000个像素。在大部分的体育比赛中，我们关注的是给定时刻球的位置。人的大脑想要了解比赛的进展，就需要了解球在运动场中的位置。对于人来说，这一切显得十分的自然，甚至不需要做任何的思考。在这个场景中，人们的大脑会自动地将显示器上数百万的像素转换为一个三维的图像，球就落在这个三维图像的中心，而且其他的区域会被大脑自然虚化，甚至隐藏(这就是传说中的绝技：我的眼里只有你）。该图像给出了运动场上球的位置。

在这个过程中，人的大脑已经自动地将数据从一百万维降至了三维(x,y,z三个轴方向).

数据预处理

场景一：

在上述比赛的例子中，人们面对的原本是百万像素的数据，但是只有球的三维位置才是最重要的，这就称为**降维（Dimensional Reduction）**。

降维的目的是为了简化计算。

和人一样，对于计算机来说，更简单的数据更便于计算.

数据预处理

场景二：

假设我们正在玩一款游戏，比如：王者荣耀或者暗黑破坏神III. 你的人物是一个70级的魔法师，你的血量是56000点，魔法值：54800，力量780，敏捷890，火焰抗性：56%，雷电抗性：21%，身上全套橙色塔拉夏套装，镶嵌了5个红色宝石，3个绿色宝石，4个紫色宝石，最后有一个70级的圣殿骑士的护卫。那么，如果你有一个对手，他是69级的猎魔师，血量是60000点，魔法值：14480，敏捷19040，力量3400，火焰抗性：26%，雷电抗性：45%，身上全套橙色掠夺套装，镶嵌了4个红色宝石，2个绿色宝石，9个紫色宝石，最后有一个70级的魔女护卫。

好了，请问你和这个对手进行PK，你的胜率是多少？

惨了，基本上没法进行进行比较，除非连续打几场，是吧？

数据预处理

场景二：

设想一下，如果我们将所有属性都转换为{0-1}之间的一个数值，并将每个属性按照重要程度给定一个权重值。这样，我们可以将每个人的能力用线性公式来表示：

$$f(x)=w[0]*x[0]+w[1]*x[1]+...+w[n]*x[n]$$

如此，双方的能力就可以用数值计算出来。也就是说，可以直接对比双方的能力了。

*Maybe还要加上一个（或若干个）**约束项**，用来表示双方的操控能力，不过这不在本节课的讨论范围中，我们暂时将其省略。*

在这个例子中，我们将不同参照系的数值进行了调整，将它们**约束到同一个维度**上来进行比较和计算，这种方法称为**归一化、正则化（Normalization）**。归一化的目的是为了让无法**计算**和**比较**的特征可以被计算。

数据预处理

数据集的**标准化**对大多数机器学习算法来说是常见的要求。如果个别特征或多或少看起来不是很像标准正态分布(具有零均值和单位方差)，那么它们的表现力**可能**会较差。

事实上，不仅仅是机器学习，在各种深度学习的算法中，零均值和单位方差也能有效提高整体系统的性能。

- **StandardScaler （Ch0901StandardScaler.py）**
- **MinMaxScaler (Ch0902MinMaxScaler.py)**
- **MaxAbsScaler (Ch0903MaxAbsScaler.py)**
- **RobustScaler (Ch0904RobustScaler.py)**
- **Normalizer (Ch0905Normalizer.py)**
- **Binarizer**

数据预处理 - StandardScaler

StandardScaler 提供了一种将数据进行归一化和标准化的操作，它能够按照输入数据构建一个**均值为0，方差为1**的归一化器，并用这个归一化器实现对训练集数据和测试集数据的归一化处理。这种归一化操作，我们通常称之为**Z-Score**。

具体而言，它会对每个特征/按列分别进行处理，先**去均值**，再**除以方差**。最后生成的数据都会**聚集在0附近，方差为1**。归一化后的数据可正可负，但一般绝对值都不会太大。

StandardScaler方法可以公式化为：

$$X = (x-\mu)/\sigma$$

其中，μ 表示某一列特征的均值 $mean(x_n)$，σ 表示这一列特征的方差。

数据预处理 - MinMaxScaler

利用**MinMaxScaler**进行的归一化是一种线性归一化方法，通常不会对数据分布产生影响，它们将特征缩放到**给定**的最小值和最大值之间。对于 **MinMaxScaler** 来说，通常有两种处理方法：

- 将特征规范化到**0和1之间**，即将特征的最大绝对值定义为单位大小，其他特征值以单位大小进行缩放。
- 将特征规范化到**min和max之间**，即将特征的最大值定义为max，特征的最小值定义为min，其他特征值以单位大小进行缩放。

数据预处理 - MinMaxScaler

如果最大值或最小值不稳定的话，经过**MinMaxScaler**处理的数据结果可能会因此而变得不稳定。但是，对于图像数据，由于像素值范围是[0,255]，因此**MinMaxScaler**类通常在图像处理上比较有效。

默认情况下，我们可以使用 **MinMaxScaler(X)** 将特征约束到[0,1]之间。当给 **MinMaxScaler** 提供一个明确的 feature_range = (min, max)时，即：scaler = MinMaxScaler(feature_range=(min, max)),

它完整的数学表达是:

$$X\_std = (X - X.min)) / (X.max - X.min)$$

$$X\_scaled = X\_std * (max - min) + min$$

事实上，对于默认约束到[0，1]的操作，我们可以定义：MinMaxScaler(feature_range=(0, 1)).

数据预处理 - MaxAbsScaler

MaxAbsScaler 与 **MinMaxScaler** 的操作非常相似，也是将数据落入一定区间，不同的是**MaxAbsScaler**会将特征缩放到[-1，1]的范围内，即将特征**绝对值**最大的特征定义为单位大小（1或-1），其他特征按照比例进行缩放。**MinMaxScaler** 也具有不破坏原有数据分布结构的特点，因此也可以用于稀疏数据，或者稀疏的CSR或CSC矩阵。

这意味着，训练数据应该是已经零中心化或者是稀疏数据。

假设原转换的数据为 x，转换后的新数据为 x'，那么存在：

$x' = \frac{x}{|max|}$，其中 max 为 x 所在列的最大值。

数据预处理 - RobustScaler

某些情况下，假如数据集中有**离群点**，我们可以使用Z-Score进行标准化，但是简单的去均值后的数据并不理想，因为异常点的特征往往在标准化之后容易失去离群特征。同样，我们也可以采用Max-Min标准化来进行处理，但是会发生更严重的问题，如果离群点偏离较大时，当将离群点规范到单位长度时，其他数据点可能会被缩放到接近0，从而失去特征的显著性，致使模型的学习失效。

基于这些原因，我们可以使用**RobustScaler**针对离群点做标准化处理，该方法对数据中心化和数据的缩放鲁棒性有更强的参数控制。

数据预处理 - Normalizer

Normalizer是一个相对特殊的正则化方法，它将每个样本特征缩放到单位范数（即每个样本的范数为1）。此时，所有样本的特征向量的欧式距离 $D_{Euclidean} = 1$。 换句话说，数据分布变成一个半径为 1 的圆（2维空间），或者是一个球（3维空间）。

Normalizer只保留了数据特征向量的方向，而忽略了向量的大小，这对于后面要使用二次型（点积）或其他核方法计算样本相似性的应用会非常有用，比如：文本分类、检索和聚类应用。例如，在检索中对两个TF-IDF倒排检索的特征向量的L2-Norm进行点积，求特征间的余弦相似性。

数据预处理 - Normalizer

Normalizer 的主要思想是对每个样本计算其 p-范数，然后对该样本中每个元素除以该范数，这样的结果是使得每个处理后的样本的p-范数等于1，该类正则化可以公式化为：

$$||x||_p == (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}$$

在sklearn库中，**Normalizer**的参数p-norm有三种取值，分别是 1, 2, ∞ ，相当于将**Normalizer**置于三种*最简形*：

- **norm= 'l1'：** 1-范数， $||x||_1 = (|x_1| + |x_2| + \cdots + |x_n|)$
- **norm= 'l2'：** 2-范数， $||x||_2 == (|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2)^{1/2}$
- **norm='max'：** ∞ -范数， $||x||_\infty = max(|x_1|, |x_2|, \ldots, |x_n|)$

数据预处理 - Binarizer

特征二值化是 将数值特征用阈值过滤得到布尔值 的过程。这对于下游的概率型模型是有用的，它们假设输入数据是多值 伯努利分布(Bernoulli distribution)。

Binarizer 可以实现将所有特征按照阈值进行二次赋值，大于等于阈值的特征设置为1，小于等于阈值的特征设置为0。

这个过程可以公式化为：

$$f(x) = \begin{cases} 1 & x > threshold \\ 0 & x \leq threshold \end{cases}$$

数据预处理 -通过数据预处理提高模型准确率

数据预处理的目的是让模型能够学到更具判别性的特征，从而提高模型的识别能力。因此，数据预处理通常会作为模型训练的前置工作。

训练模型前的数据处理一般包含两个步骤，

- **数据清洗：**让数据更干净，噪声数据更少（但不绝对）；
- **数据预处理：**让数据特征更具显著性，更容易被模型学到数据集内在的知识和信息。

数据降维

数据简化的作用：

- **方便可视化**
- 使得数据集更易使用
- 降低更多算法的计算开销
- 去除噪声
- 使得结果易懂

数据降维 - 常见的数据降维方法

- **成分分析（Principal Component Analysis, PCA）**

在PCA中数据从原来的坐标系转换到了新的坐标系，新坐标系的选择是**由数据本身决定**。

- 第一个新坐标轴选择的是原始数据中方差最大的方向，
- 第二个新坐标轴的选择和第一个坐标轴正交且具有最大方差的方向。
- 该过程一直重复，重复次数为原始数据中特征的数目。

我们会发现，大部分方差都包含在**最前面**的几个坐标轴中。因此，我们可以**忽略余下的坐标轴**，即实现对数据进行**降维**处理。

数据降维

- **因子分析（Factor Analysis）**

在因子分析中，我们假设在观察数据的生成中有一些观察不到的**隐变量（latent variable）**。假设观察数据是这些隐变量和某些噪声的线性组合。那么隐变量的数据可能比观察数据特征的数目少，也就是说通过找到隐变量就可以实现数据的降维。因子分析被广泛应用在社会科学、金融和其他领域。

- **独立成分分析（Independent Component Analysis，ICA）**

ICA假设数据是从N个数据源生成的，这一点和因子分析有点相似。假设数据为多个数据源的混合观察结果，这些数据源之间在统计上是相互独立的，而在PCA中之假设数据是不相关的。同因子分析一样，如果数据源的数量少于观察数据特征的数量，则可以实现降维过程。

数据降维 - PCA主成分分析

- **优点**:

  减低数据的复杂性，识别最重要的多个特征。

- **缺点**:

  不一定需要，且可能损失有用信息。（需要丢弃一些方差不大的特征）

- **适用数据类型**：数值型数据。

数据降维 - PCA主成分分析 – Example1

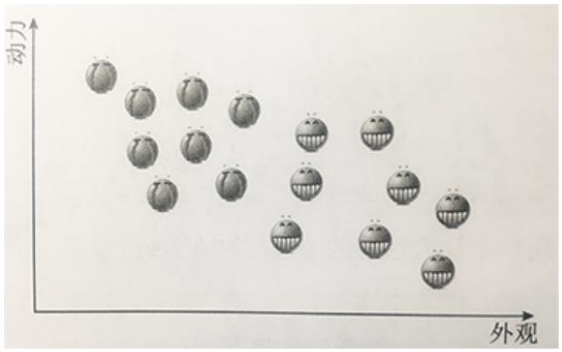


图1 汽车外观和动力对喜好的影响

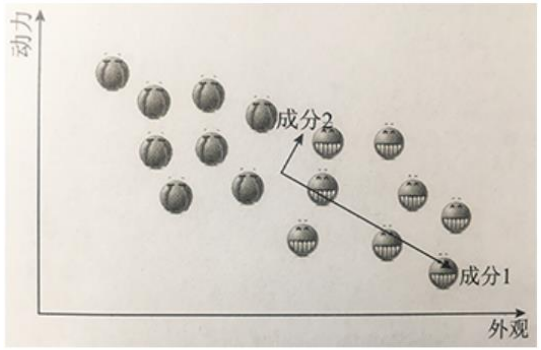


图2 对数据点添加主成分标注

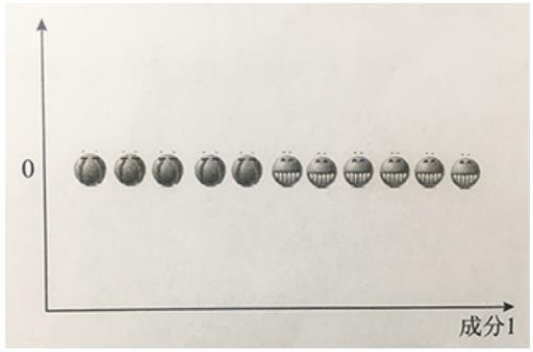


图3 去掉成分2的数据集

PCA降维的基本过程

- 将数据点分布最长的方向标注为成分1
- 与成分1垂直的方向标注为成分2
- 设成分2的取值 = 0， 成分1作为横坐标

可以看出， 表情散点从一个平面变成了一条直线。也就是说，二维的特征变成了一维。特征的维度降低了——**降维。**

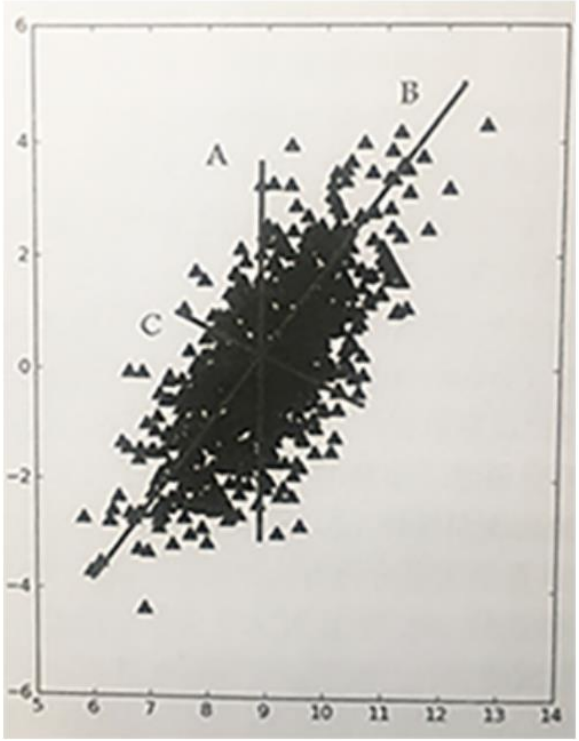
数据降维 - PCA主成分分析 – Example2

- **选择主成分**

考虑图中的大量数据点。如果要求我们画出一条直线，这条直线要尽可能覆盖这些点，那么最长的线可能是哪一条？

直线B

*（在PCA中，我们需要对数据坐标进行旋转，旋转的过程与方向**取决于数据**本身→→我们需要将第一条坐标轴旋转到覆盖数据的**最大方差**位置）*

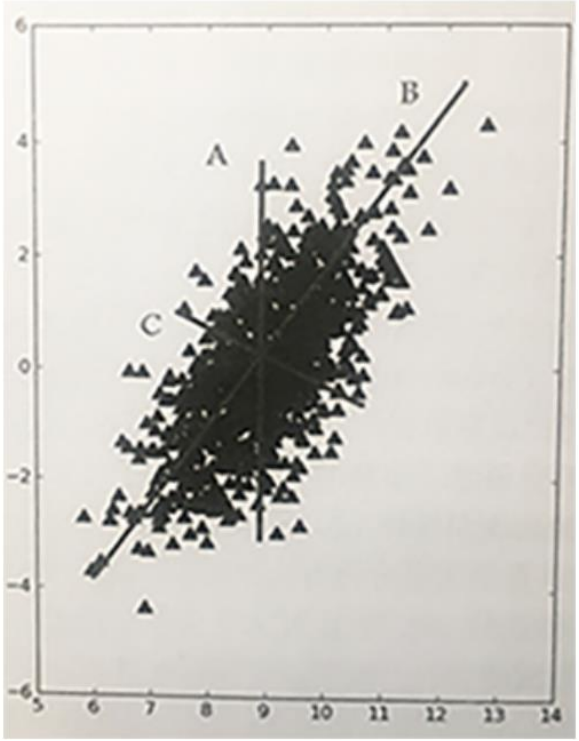


数据的最大方差给出了数据最重要的信息。

数据降维 - PCA主成分分析 – Example2

- **选择主成分**

1. 选择覆盖数据最大差异性的坐标轴
2. 选择第二条坐标轴，该坐标轴与第一条坐标轴垂直，即覆盖数据次大差异性的坐标轴。更严谨的说法就是**正交（orthogonal）**。



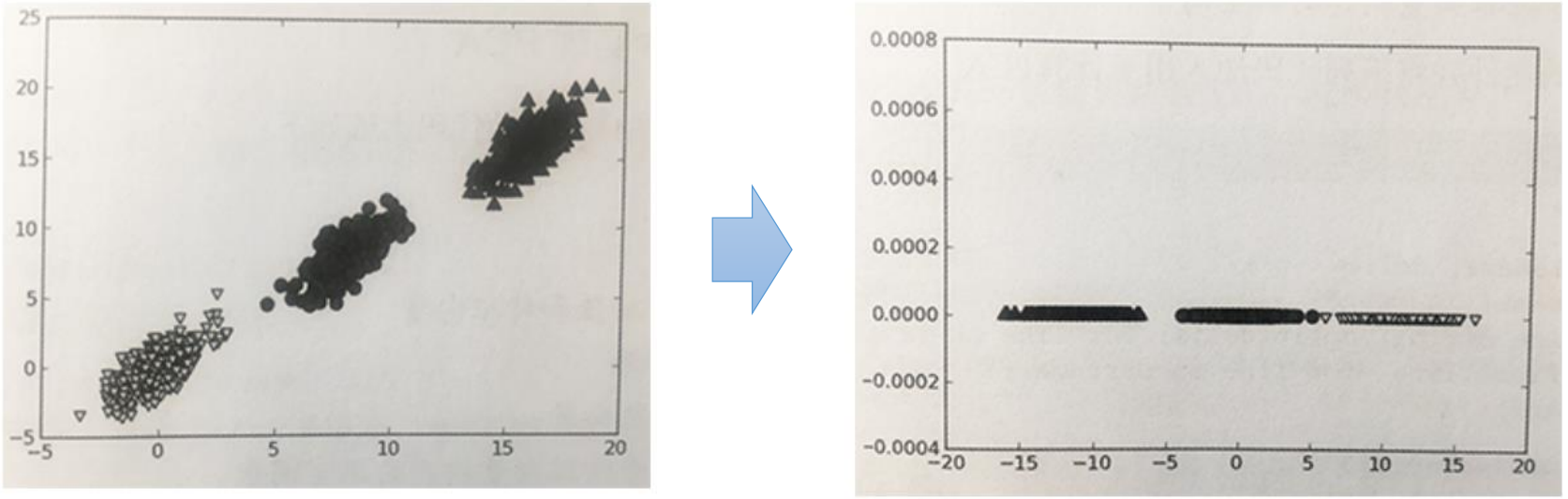
在二维平面下，垂直和正交是一回事。直线C就是第二条坐标轴，即和第一条坐标轴正交的坐标轴。

不难想象，如果主成分不只2个的话，那么我们还需要继续选择第三条坐标轴，第四条坐标轴。只不过第三条、第四条坐标轴需要在高维空间中进行选择。这需要一点空间想象力。

PCA将数据坐标轴旋转到数据角度上的那些最重要的方向。

数据降维 - PCA主成分分析 – Example2

- 降维



从左图到右图，数据从二维降低到了一维，决策面变得简单而且准确。在右图中，我们也只需要一维信息即可，因为另一维信息只是对分类缺乏贡献的噪声数据。

特征提取

特征表达（Feature Representation）：

为了便于处理，将原始数据进行一定的变换，转换为易于处理的数据格式。

- 如何实现比较良好的数据表达？

  特征提取（Feature Extraction）

在机器学习中，特征提取是非常重要的一个过程，选择合适的特征直接关系到后续分类器的判决能力。

值得注意的是，在深度学习中，一个非常重要的区别是：**特征选择是由模型来完成。**

特征提取

- **非负矩阵分解(Nonnegative Matrix Factorization，NMF)**

由Lee和Seung于1999年在自然（Nature）杂志上提出的一种矩阵分解方法，它使分解后的所有分量均为非负值(要求纯加性的描述)，并且同时实现非线性的维数约减。NMF的心理学和生理学构造依据是对整体的感知由对组成整体的部分的感知构成的(纯加性的)，这也符合直观的理解：整体是由部分组成的，因此它在某种意义上抓住了智能数据描述的本质。此外，这种非负性的限制导致了相应描述在一定程度上的稀疏性，稀疏性的表述已被证明是介于完全分布式的描述和单一活跃分量的描述之间的一种有效数据描述形式。

特征提取

相对而言，PCA降维中的主成分的数量 N，是按照方差最大原则进行排序后选择 TopN 获得，选择不同数量的主成分，只需要调整Top中的N即可，不需要重复计算；选择不同数量的NMF的分量时，都需要重新计算。此外，NMF中的成分没有顺序关系，不像PCA中的成分是按照方差大小顺序排列。

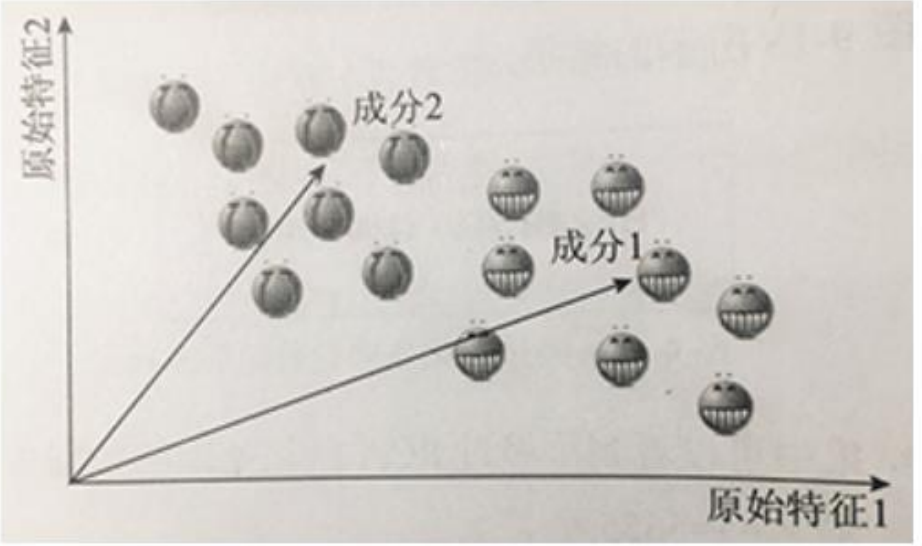


图7 非负矩阵分解NMF

聚类

聚类（Clustering）：根据在数据中发现的描述对象及其关系的信息，将数据对象分组(簇)。聚类的目标是，让组内的对象相互之间尽量相似，而不同组中的对象尽量不同。同一组内相似性越大，组间差别越大，聚类就越好。

下面简要介绍**三种**聚类算法：

1. **K均值聚类（K-Means）**：基于原型的，划分的聚类技术，试图从全部数据对象中发现用户指定个数的簇。
2. **凝聚层次聚类**：开始每个点各成一簇，然后重复的合并两个最近的簇，直到指定的簇个数。
3. **DBSCAN**：一种基于密度的聚类算法。

聚类 - K均值聚类算法（K-Means）

K-means算法是一种聚类算法，所谓聚类，即根据相似性原则，

- 将具有较高相似度的数据对象划分至同一类簇；
- 将具有较高相异度的数据对象划分至不同类簇。

聚类与**分类**最大的区别在于：

- **聚类**，无监督（没有标签 y）过程，待处理数据对象没有任何先验知识；
- **分类**，有监督过程（有标签 y），存在有先验知识的训练数据集。

聚类 - K均值聚类算法（K-Means）

K-means算法中的 *k* 代表类簇个数，means 代表类簇内数据对象的均值（这种均值是一种对类簇中心的描述），因此，k-means算法又称为***k-均值*** 算法。k-means算法是一种基于划分的聚类算法，以距离作为数据对象间相似性度量的标准，即数据对象间的距离越小，则它们的相似性越高，它们越有可能在同一个类簇。

数据对象间距离的计算有很多种，k-means算法通常采用**欧氏距离**来计算数据对象间的距离。算法详细的流程描述如下：

1. 随机选定*n*个聚类中心
2. 计算每个样本与这*n*个聚类中心的距离
3. 将样本划分至最近的聚类中心所在的类
4. 更新所有聚类中心，并计算误差
5. 重新迭代执行2-4

聚类 - K均值聚类算法（K-Means）

k-means算法优缺点分析

- **优点**： 算法简单易实现；
- **缺点**：
  - 需要用户事先指定类簇个数；
  - 聚类结果对初始类簇中心的选取较为敏感；
  - 容易陷入局部最优；
  - 在大规模数据集上收敛较慢

聚类 - 凝聚聚类算法

凝聚：算法初始时，将每个点作为一个簇，每一步合并两个最接近的簇。另外即使到最后，对于噪音点或是离群点也往往还是各占一簇的，除非过度合并。对于这里的最接近，有下面三种定义：

- **单链(MIN)：**定义簇的邻近度为不同两个簇的两个最近的点之间的距离。
- **全链(MAX):**定义簇的邻近度为不同两个簇的两个最远的点之间的距离。
- **组平均：**定义簇的邻近度为取自两个不同簇的所有点对邻近度的平均值。

聚类 - DBSCAN算法

DBSCAN：全称基于密度的有噪声应用空间聚类（**D**ensity-**B**ased **S**ppatial **C**lustering of **A**pplications with **N**oise），它是一种简单的、基于密度的聚类算法。**DBSCAN**使用基于中心的方法，每个数据点的密度通过对以“该点为中心，以边长为2eps”的网格(邻域)内的其他数据点的个数来度量。

简单的说，它会通过对特征空间内的密度进行检测，密度大的地方，DBSCAN认为是一个类；而密度相对较小的区域会认为存在一个分界线。基于这样的工作机制，**DBSCAN**不需要像**K-Means**和**凝聚聚类**算法一样，一开始指定聚类的数量*n_clusters*，而是根据算法的执行*(依据密度参数eps和min_samples)*自动确定类别的数量。

聚类 - DBSCAN算法

DBSCAN根据密度的不同，将数据点分为三类：

- **核心点**：该点在邻域内的密度超过给定的阀值MinPs。
- **边界点**：该点不是核心点，但是其邻域内包含至少一个核心点。
- **噪音点**：不是核心点，也不是边界点。

有了以上对数据点的划分，聚合可以这样进行：

各个**核心点**与其**邻域内**的所有核心点放在同一个簇中，把边界点跟其邻域内的某个核心点放在同一个簇中。

欧老师的联系方式

读万卷书 行万里路 只为最好的修炼

QQ：14777591 (宇宙骑士)

Email：ouxinyu@alumni.hust.edu.cn

Tel：18687840023