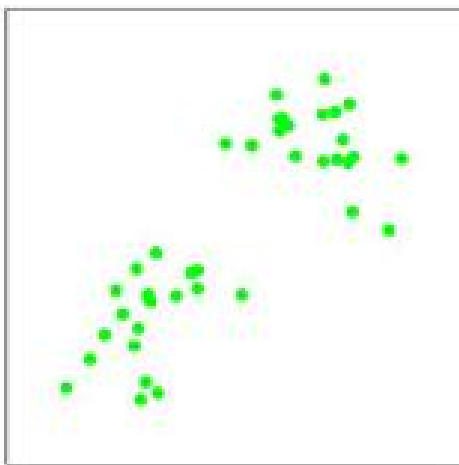


# 人工智能之机器学习

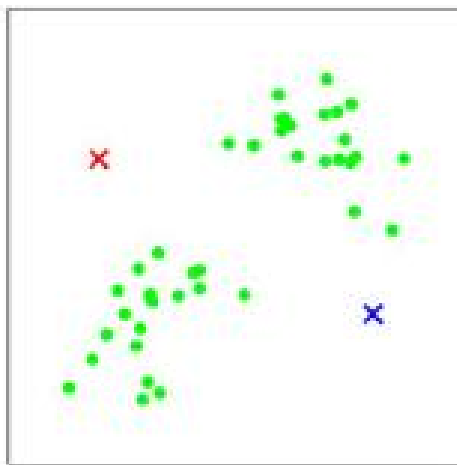
## 聚类算法

主讲人：李老师

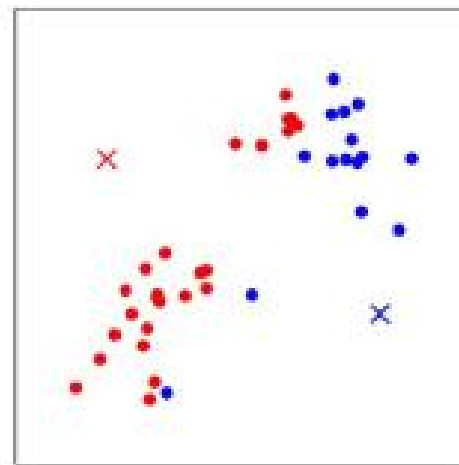
# K-means直观理解



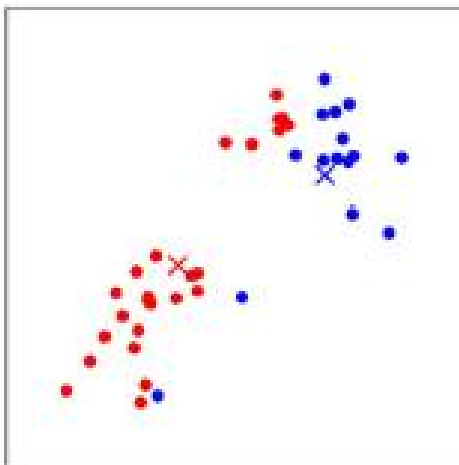
(a)



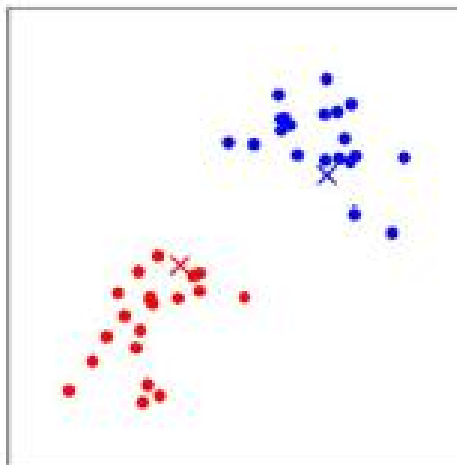
(b)



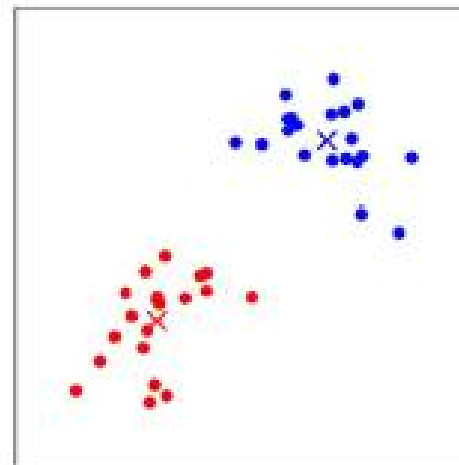
(c)



(d)



(e)



(f)

Kmeans演示: <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

# K均值聚类 (K-means) 介绍

- 历史渊源

- 虽然其思想能够追溯到1957年的Hugo Steinhaus, 术语 “k-均值” 于1967年才被James MacQueen首次使用。标准算法则是在1957年被Stuart Lloyd作为一种脉冲码调制的技术所提出, 但直到1982年才被贝尔实验室公开出版。在1965年, E.W.Forgy发表了本质上相同的方法, 所以这一算法有时被称为Lloyd-Forgy方法。更高效的版本则被Hartigan and Wong提出 (1975/1979)

- 介绍

- K-Means算法是**无监督的聚类算法**, 算法简单, 聚类效果好, 即使是在巨大的数据集上也非常容易部署实施。正因为如此, 它在很多领域都得到的成功的应用, 如市场划分、机器视觉、地质统计学、天文学和农业等。K-Means算法有大量的变体, 包括初始化优化K-Means++以及大数据应用背景下的k-means||和Mini Batch K-Means

## K均值聚类 (K-means) 介绍

- K-Means算法的思想很简单，对于给定的样本集，按照样本之间的距离大小，将样本集划分为K个簇。让簇内的点尽量紧密的连在一起，而让簇间的距离尽量的大。
- 如果用数学表达式表示，假设簇划分为 $(C_1, C_2, \dots, C_k)$ ，则我们的目标是最小化平方误差E：

$$E = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|_2^2$$

其中  $\mu_i$  是簇  $C_i$  的均值向量，有时也称为质心，表达式为：

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

- 开头的那组图就可以形象描述K-Means的迭代求解过程。

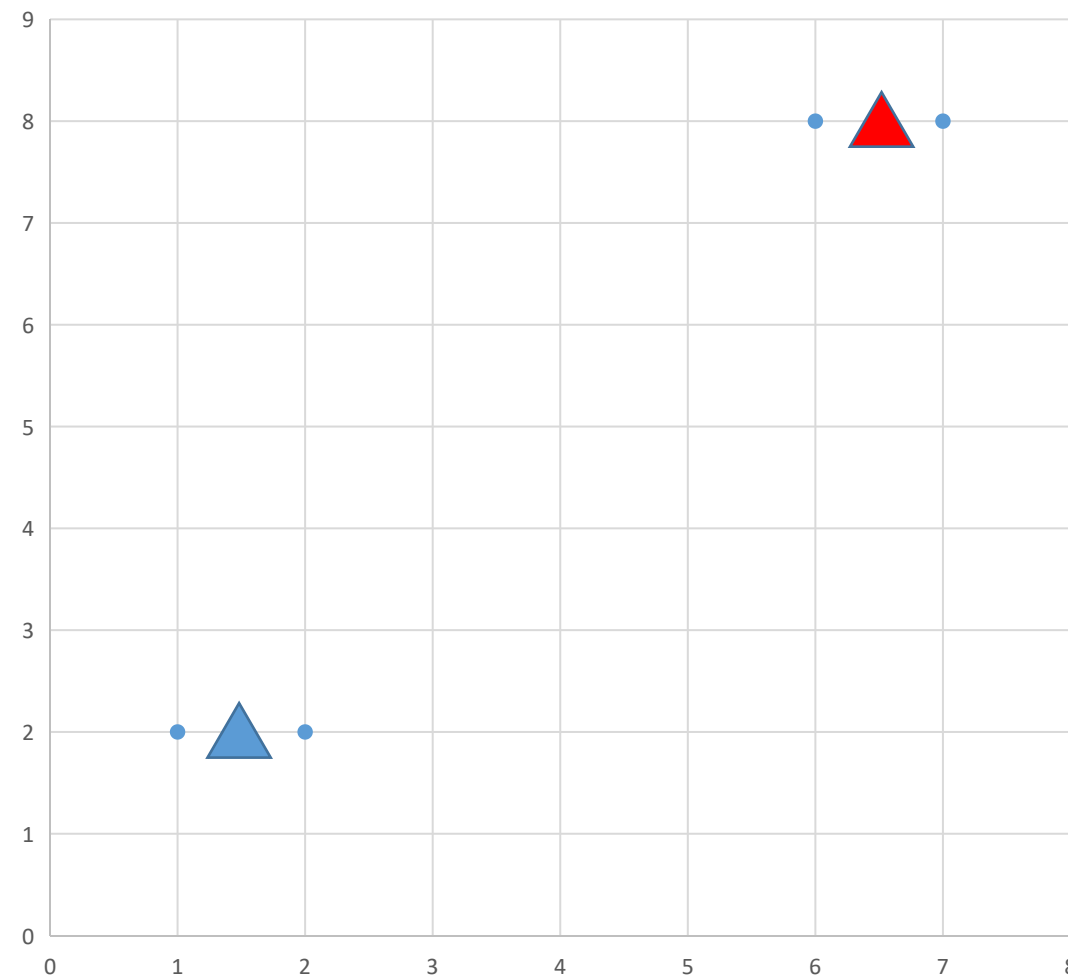
## K-means算法流程

- 输入是样本集 $D=\{x_1, x_2, \dots, x_N\}$ , 聚类的簇数 $k$ , 最大迭代次数 $T$
- 输出是簇划分 $C=\{C_1, C_2, \dots, C_k\}$
- 1) 从数据集 $D$ 中随机选择 $k$ 个样本作为初始的 $k$ 个质心向量:  $\{\mu_1, \mu_2, \dots, \mu_k\}$ , 将每个簇初始化为空集
- 2) 对于 $t=1, 2, \dots, T$ 
  - a) 对于 $i=1, 2, \dots, N$ , 计算样本  $x_i$  和各个质心向量  $\mu_j$ ,  $j=1, 2, \dots, k$  的欧式距离, 将  $x_i$  划分到最近的簇中, 即更新 $C_j=C_j \cup \{x_i\}$
  - b) 对于 $j=1, 2, \dots, k$ , 对 $C_j$ 中所有的样本点重新计算新的质心
  - c) 如果所有的 $k$ 个质心向量都没有发生变化, 则转到步骤3)
- 3) 输出簇划分 $C=\{C_1, C_2, \dots, C_k\}$

$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$

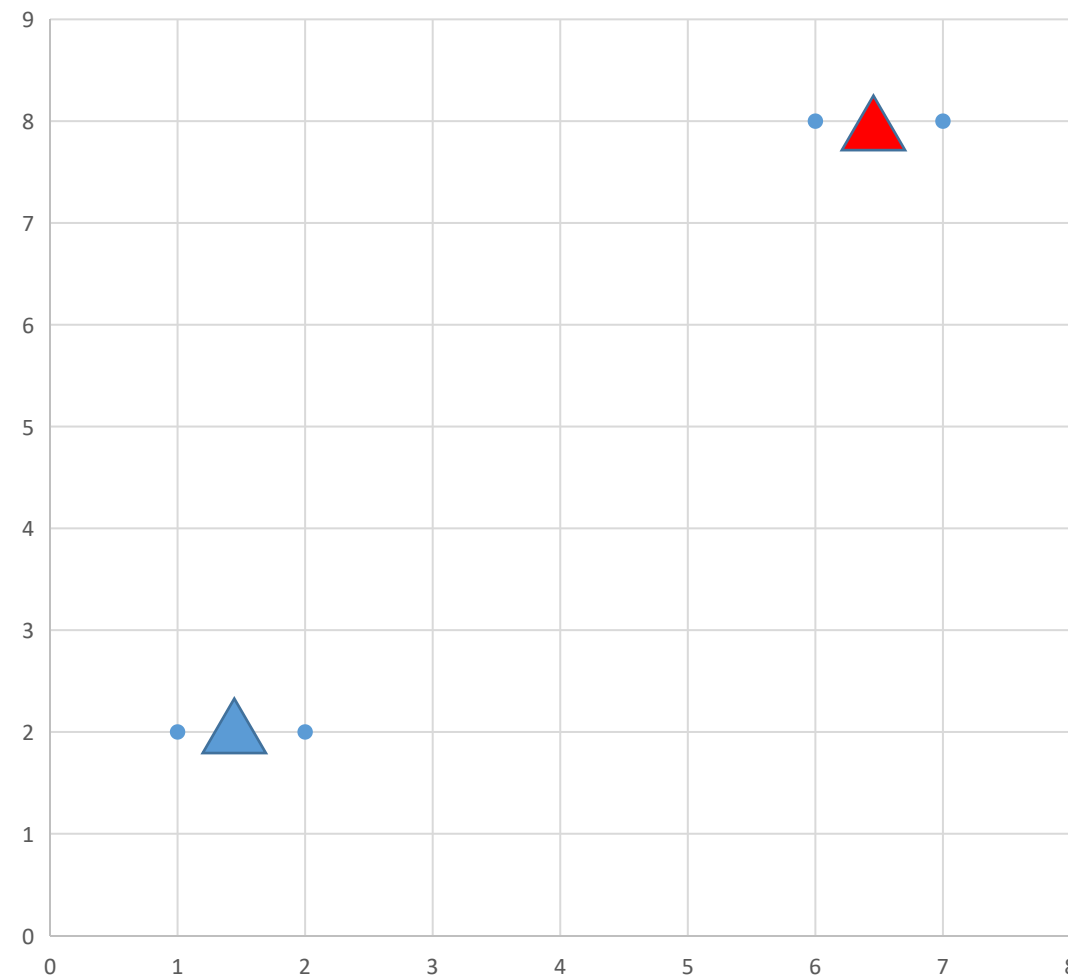
# 编程K-means

- 对数据集  $X = \text{np.array}([[1, 2], [2, 2], [6, 8], [7, 8]])$  聚类,  $K=2$ , 初始聚类中心为  $C = \text{np.array}([[1, 2], [2, 2]])$
- 要求: 打印每次迭代聚类中心的位置



## 调库——编程K-means

- 对数据集  $X = \text{np.array}([[1, 2], [2, 2], [6, 8], [7, 8]])$  聚类,  $K=2$ , 初始聚类中心为  $C = \text{np.array}([[1, 2], [2, 2]])$
- 要求: 打印每次迭代聚类中心的位置



## Kmeans算法的缺点

- 缺点一：聚类中心的个数K需要事先给定，但在实际中K值的选定是非常困难的，很多时候我们并不知道给定的数据集应该聚成多少个类别才最合适
- 缺点二：k-means算法需要随机地确定初始聚类中心，不同的初始聚类中心可能导致完全不同的聚类结果，有可能导致算法收敛很慢甚至出现聚类出错的情况
- 针对第一个缺点：很难在k-means算法以及其改进算法中解决，一般来说，我们会根据对数据的先验经验选择一个合适的k值，如果没有什么先验知识，则可以通过“肘方法”选择一个合适的k值
- 针对第二个缺点：可以通过k-means++算法来解决



## 第一个缺点解决方案之一——肘方法

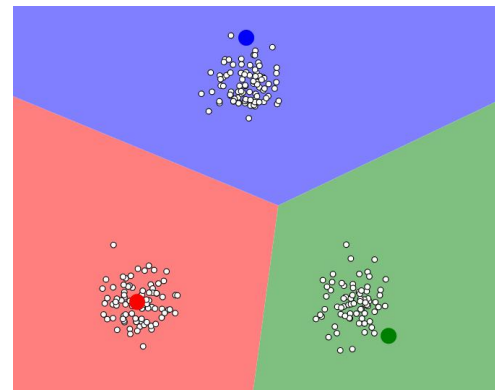
- We then compute Within Set Sum of Squared Error (WSSSE). You can reduce this error measure by increasing  $k$ . In fact the optimal  $k$  is usually one where there is an “elbow” in the WSSSE graph.

肘方法 (elbow method) 基于如下观察：增加簇数有助于降低每个簇的簇内方差之和。这是因为有更多的簇可以捕获更细的数据对象簇，簇中对象之间更为相似。然而，如果形成太多的簇，则降低簇内方差和的边缘效应可能下降，因为把一个凝聚的簇分裂成两个只引起簇内方差和的稍微降低。因此，一种选择正确的簇数的启发式方法是，使用簇内方差和关于簇数的曲线的拐点。

严格地说，给定  $k > 0$ ，我们可以使用一种像  $k$  - 均值这样的算法对数据集聚类，并计算簇内方差和  $var(k)$ 。然后，我们绘制  $var$  关于  $k$  的曲线。曲线的第一个（或最显著的）拐点暗示“正确的”簇数。

## 第二个缺点解决方案之一——K-means++

- K-Means++的对于初始化质心的优化策略，如下：
  - a) 从输入的数据点集合中随机选择一个点作为第一个聚类中心  $\mu_1$
  - b) 对于数据集中的每一个点 $x_i$ ，计算它与已选择的聚类中心中最近聚类中心的距离 $D$
  - c) 选择一个新的数据点作为新的聚类中心，选择的原则是： $D$ 较大的点，被选取作为聚类中心的概率较大。
  - d) 重复b和c直到选择出 $k$ 个聚类质心
  - e) 利用这 $k$ 个质心来作为初始化质心去运行标准的K-Means算法
- 思考：
  - 1.c步骤如何实现呢？（编程）
  - 2.c步为什么不直接取最大的，而是要按照概率来取？



如果最远的点是噪声点，就会选到错误的聚类中心，造成聚类出错

## K-means++的缺点

- 虽然k-means++算法可以确定地初始化聚类中心，但是从可扩展性来看，它存在一个缺点？？。
- 解决方案：针对这种缺陷，k-means||算法提供了解决方法。

它内在的有序性特性：  
下一个中心点的选择依  
赖于已经选择的中心点

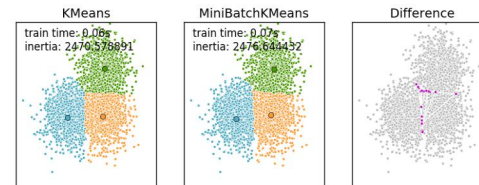
## K-means||

- k-means||是k-means++的变体，k-means||在初始化中心点时对kmeans++的缺点做了规避，主要体现在不需要根据k的个数严格地寻找k个点，突破了算法在大规模数据集上的应用瓶颈，同时初始化的中心点更加健壮
- 参考论文
- <http://theory.stanford.edu/~sergei/papers/vldb12-kmpar.pdf>

## K-Means++算法

- 解决K-Means++算法缺点而产生的一种算法；主要思路是改变每次遍历时候的取样规则，并非按照K-Means++算法每次遍历只获取一个样本，而是每次获取K个样本，重复该取样操作 $O(\log n)$ 次，然后再将这些抽样出来的样本聚类出K个点，最后使用这K个点作为K-Means算法的初始聚簇中心点。实践证明：一般5次重复采用就可以保证一个比较好的聚簇中心点。

# Mini Batch K-Means\*

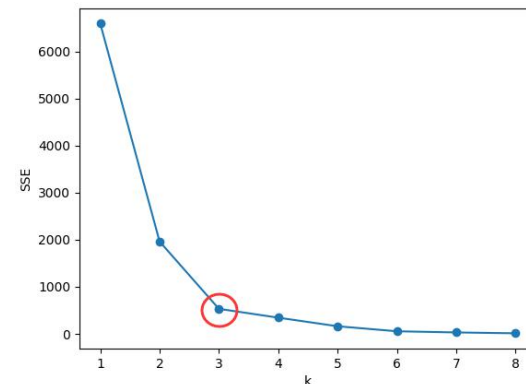


- 在传统的K-Means算法中，要计算所有的样本点到所有的质心的距离。如果样本量非常大，比如达到10万以上，特征有100以上，此时用传统的K-Means算法非常的耗时，就算加上elkan K-Means优化也依旧。在大数据时代，这样的场景越来越多。此时Mini Batch K-Means应运而生。
- 顾名思义，Mini Batch，也就是用样本集中的一部分的样本来做传统的K-Means，这样可以避免样本量太大时的计算难题，算法收敛速度大大加快。当然此时的代价就是我们的聚类的精确度也会有一些降低。一般来说这个降低的幅度在可以接受的范围之内。
- 在Mini Batch K-Means中，我们会选择一个合适的批样本大小batch size，我们仅仅用batch size个样本来做K-Means聚类。那么这batch size个样本怎么来的？一般是通过无放回的随机采样得到的。
- 为了增加算法的准确性，我们一般会多跑几次Mini Batch K-Means算法，用得到不同的随机采样集来得到聚类簇，选择其中最优的聚类簇。

## 评估方法——肘方法

- 手肘法的核心指标是SSE(sum of the squared errors, 误差平方和),

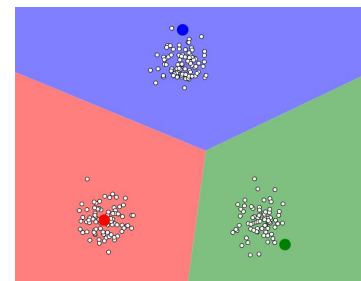
$$SSE = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2$$



- 其中， $C_i$ 是第*i*个簇， $x$ 是 $C_i$ 中的样本点， $\mu_i$ 是 $C_i$ 的质心（ $C_i$ 中所有样本的均值），SSE是所有样本的聚类误差，代表了聚类效果的好坏。
- 手肘法的核心思想是：随着聚类数 $k$ 的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和SSE自然会逐渐变小。并且，当 $k$ 小于真实聚类数时，由于 $k$ 的增大会大幅增加每个簇的聚合程度，故SSE的下降幅度会很大，而当 $k$ 到达真实聚类数时，再增加 $k$ 所得到的聚合程度回报会迅速变小，所以SSE的下降幅度会骤减，然后随着 $k$ 值的继续增大而趋于平缓，也就是说SSE和 $k$ 的关系图是一个手肘的形状，而这个肘部对应的 $k$ 值就是数据的真实聚类数。当然，这也是该方法被称为手肘法的原因。

## 评估方法——轮廓系数法

- **簇内不相似度**：计算样本 $i$ 到同簇其它样本的平均距离为 $a_i$ ， $a_i$ 越小，表示样本 $i$ 越应该被聚类到该簇，簇 $C$ 中的所有样本的 $a_i$ 的均值被称为簇 $C$ 的凝聚度。
- **簇间不相似度**：计算样本 $i$ 到其它簇 $C_j$ 的所有样本的平均距离 $b_{ij}$ ， $b_i = \min\{b_{i1}, b_{i2}, \dots, b_{ik}\}$ ； $b_i$ 越大，表示样本 $i$ 越不属于其它簇。
- **轮廓系数**： $s_i$ 值取值范围为 $[-1, 1]$ ，越接近1表示样本 $i$ 聚类越合理，越接近-1，表示样本 $i$ 应该分类到另外的簇中，近似为0，表示样本 $i$ 应该在边界上；所有样本的 $s_i$ 的均值被称为聚类结果的轮廓系数。



$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$
$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)}, & a(i) < b(i) \\ 0, & a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1, & a(i) > b(i) \end{cases}$$

