1. 简述 ababoost 算法的设计思想以及主要的计算步骤

设计思想：

采用一些性能较弱的分类器对训练集进行训练，在每一轮训练结束后，提升本轮错分类样本的权重，降低正确分类样本的权重，同时每一轮分类器根据其分类性能获得不同的分类器权重。这样使得在下一轮分类器学习时，上一轮被错分类的样本被更多地照顾，分类错误率会越来越小。到达终止条件时，得到一系列带权重的分类器，能够对训练集高精度分类。

主要计算步骤：

输入训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

① 初始化训练数据的权重分布

$$D_1 = \{w_{11}, w_{12}, \dots, w_{1n}\}, w_{1i} = 1/n, i = 1, \dots, n$$

② 对于$m = 1,2, \dots, M$

(2a) 使用具有权值分布D_m的训练数据，学习基本分类器

$$G_m(x): X \to \{-1, +1\}$$

(2b) 计算G_m在训练数据集上的加权分类错误率

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{n} w_{mi} I(G_m(x_i) \neq y_i)$$

(2c) 计算G_m的贡献系数

$$\alpha_m = \frac{1}{2} ln \frac{1 - e_m}{e_m}$$

(2d) 更新训练数据集的权重分布

$$D_{m+1} = \{w_{m+1,1}, w_{m+1,2}, \dots, w_{m+1,n}\}$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \times \begin{cases} exp(-\alpha_m), G_m(x_i) = y_i \\ exp(\alpha_m), G_m(x_i) \neq y_i \end{cases}$$

$$Z_m = \sum_{i=1}^{n} w_{mi} \exp\left(-\alpha_m y_i G_m(x_i)\right)$$

③ 构建基本分类器的线性组合

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

2. 从 GMM 的角度，简述 Kmeans 聚类算法的原理；给出 Kmeans 聚类算法的计算步骤；并说明哪些因素会影响 Kmeans 算法的聚类性能。

原理：

GMM 采用 k 个 d 元高斯分布的混合高斯分布来对数据进行聚类。每一个高斯分布都是一个簇。初始情况下随机获得 k 个均值、k 个协方差阵、k 个簇的概率。然后根据求得每一个样本属于每一个簇的概率，最大化似然函数对均值、协方差阵以及簇的概率进行更新。

对应到经典 Kmeans 而言，k 个协方差阵均为单位阵，即 kmeans 采用欧氏距离进行度量。同时，在求每一个样本属于每一个簇的概率时，采用硬编码，即每一个样本硬分类到概率最大的那个簇。因此 Kmeans 是 GMM 硬编码的一种特殊形式。

Kmeans 计算步骤：

① 随机初始化(或根据某种方式)产生 k 个聚类中心

② 计算每一个样本到所有聚类中心的距离，将每一个样本划分到离它最近的簇

③ 按照新的簇划分，更新聚类中心

④ 重复②③步，直至类中心不再改变或者达到最大迭代次数

影响因素：

① 聚类数目 k 值的选择，不同的 k 值将直接导致不同的聚类结果

② 初始聚类中心的选择，如果某些个初始聚类中心在同一个簇或者在边缘点，就会导致聚类性能降低

③ 距离度量的选择，除了欧式距离度量，不同分布的数据使用其他距离度量可能会有更好的结果

④ 聚类中心的表达方式，采用簇均值的聚类中心方式可能不是最佳的，比如 K-Medians 可能会产生更好的结果

3. 简述谱聚类算法的原理，给出一种谱聚类算法的步骤，并指出哪些因素会影响聚类的性能。

原理：

在数据的图表示中，希望通过寻找一个最小代价的切割方式，将图划分为 k 个不连通的子图，两个子图之间的相似性要小，由此构造出目标函数

$$cut(A_1, A_2, \cdots, A_k) \coloneqq \frac{1}{2}\sum_{i=1}^{k} W(A_i, \bar{A_i})$$

将目标函数进行改写，可以推导出划分后的子图的簇，等价为样本数据拉普拉斯矩阵低维谱空间的聚类结果。由此谱聚类是对样本数据拉普拉斯矩阵低维谱空间进行聚类。

Ng 算法计算步骤：

① 通过 Knn、全连、或$\epsilon - neighborhood$构造亲和矩阵 W、度矩阵 D。计算归一化拉普拉斯矩阵$L_{sym} = I - D^{-1/2}WD^{-1/2}$

② 选择L_{sym}前 k 个最小特征值对应的向量$u_1, u_2, \cdots, u_k$

③ 构建$U = [u_1, u_2 \cdots, u_k] \in R^{n\times k}$

④ 将U每一行向量进行单位化

⑤ 将U的行向量看成k维样本，使用聚类算法对其进行聚类，得到聚类结果

影响因素：

① 聚类数目 k

② 初始亲和矩阵的生成方式，是 KNN 还是全连还是$\epsilon - neighborhood$。

③ 点对亲和度的计算，除了高斯相似度，还有其他的相似度比如余弦相似度、马氏距离、Jaccard 相关系数

④ 谱空间聚类算法的选择

编程 1

本例最初所写 Kmeans 初始化样本点的过程如下：

① 随机选择一个样本点作为第一个聚类中心，添加到聚类中心列表 u_list

② 计算所有样本到所有已知聚类中心的欧氏距离

③ min-step 选择每个样本到所有聚类中心欧氏距离的最小值，记为 d-temp

④ max-step 选择 d_temp 最大值对应的样本作为新的聚类中心点，添入 u_list

⑤ 重复②③④，直至 u_list 有 k 个初始聚类中心

设置聚类数为 5，重复运行 50 次，聚类的性能以及所得聚类中心与真实分布均值的二范数累加和如图 1-2：

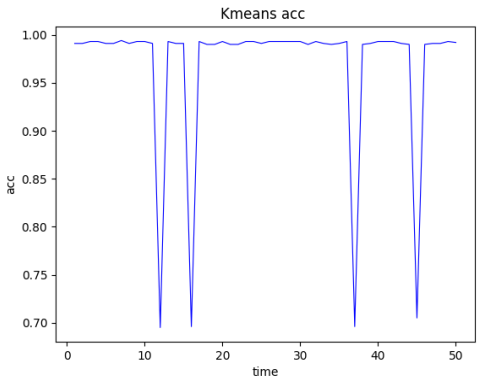


图 1.1 聚类性能

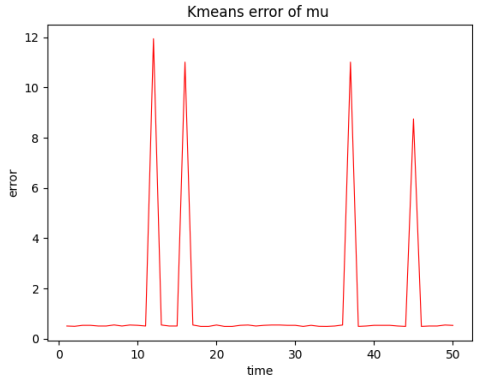


图 1.2 类中心二范数误差

通过上图可以发现，在大多数情况下，聚类性能为 0.99 以上，最高为 0.993，类中心二范数累计误差为 0.54 左右。但是存在个别情况，使得分类性能急剧下降到 0.7 左右，通过可视化图，发现出现这种情况是因为有两个类中心初始到同一个簇里，导致实际上只有 4 个聚类中心，聚类结果出现偏差。

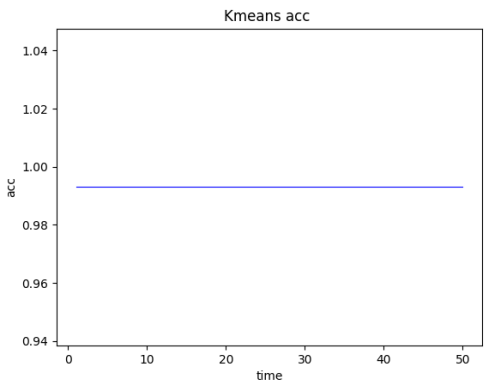
这是因为第一步进行的是随机选择一个样本作为某一个簇的初始聚类中心，尽管有后面的 min-max step，仍会出现初始化的聚类中心在同一个簇里。因此将初始化类别中心进行改良。改良过程其实很简单，由于出错的步骤在于初始化第一个随机聚类中心，因此将①步骤进行如下更改：

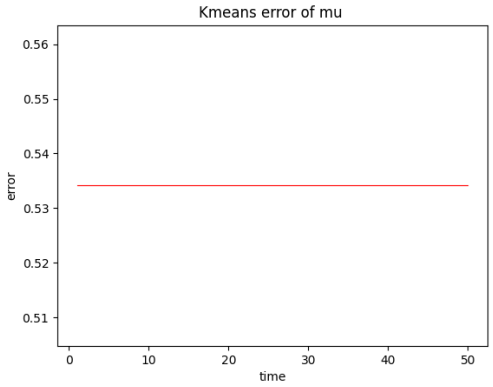
⑴计算样本点对的余弦相似度。

⑵计算每一个样本点的密度：每一个样本点除自身外，最大 10 个余弦相似度的平均值。

⑶选择密度最大的样本点作为第一个初始聚类中心。

上述改良方案考虑的是，密度最大的一个点极有可能是样本的中心点。选取该点作为第一个聚类中心点，再经过 min-max step 就能够生成较好表示的聚类中心点。事实证明，经过本方法生成的初始化中心点，能够较快地完成迭代。下图为改良方案的聚类性能以及类中心累计二范数误差：





可以发现性能十分稳定，聚类准确度为 0.993，类中心二范数累计误差小于 0.54

所获得的聚类中心为：

[9.0361641 , 0.01224488] [0.89604309, 4.0497816] [1.0203202 , -1.08880343]

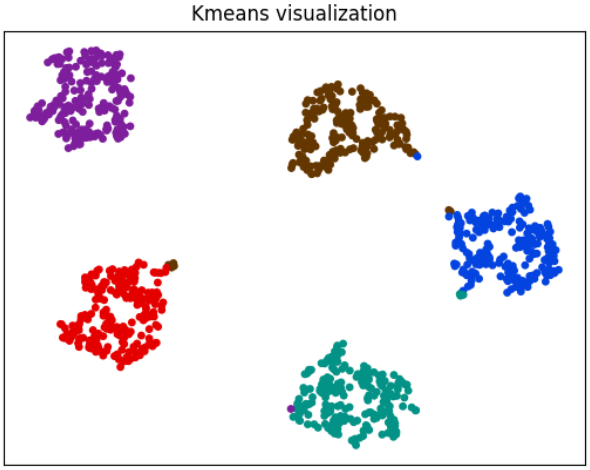
[5.55820209, -4.59115453] [6.1568884 , 4.40885642]

初始聚类中心为：

[9 , 0.0] [1 ,4] [1 , -1]

[5.5 , -4.5] [6 ,4.5]

可以看到经过 kmeans 聚类后，聚类中心非常接近原始的聚类中心。将聚类结果可视化(T-SNE)如下：



可以看到在 T-SNE 中，所有数据都是可分的，这是因为 T-SNE 进行了高斯估计，而源数据正是服从高斯分布的，因此图上展示的可分性程度大。但是本例采用了欧氏距离进行度量，会造成一定的分类误差。

编程 2

本例实现了采用 Knn、全连、或$\epsilon - neighborhood$构造亲和矩阵 W，使用了经典算法、Shi 算法、Ng 算法进行求解。表 1 为聚类数$k = 2$，10 近邻构图$\sigma = 1.0$，阈值$\varepsilon = 0.9$的聚类性能。

算法\构图方式	Knn	全连图	ϵ-neighborhood
经典算法	1.0	0.72	1.0
Shi	1.0	0.73	1.0
Ng	1.0	0.73	1.0

表 1 不同构图方式以及算法聚类性能

随后测试了不同 knn 构图以及不同的σ对聚类性能的影响。分析时 knn 的 k 从 3 增长到 18，步长为 1。σ从 0.1 增长到 5.0，步长为 0.1。图 2.1 展示了σ为 1、2、3、4 时，聚类性能随着 k 的变化。图 2.2 展示了 k 为 5、10、15、20 时，聚类性能随着 signma 的变化。

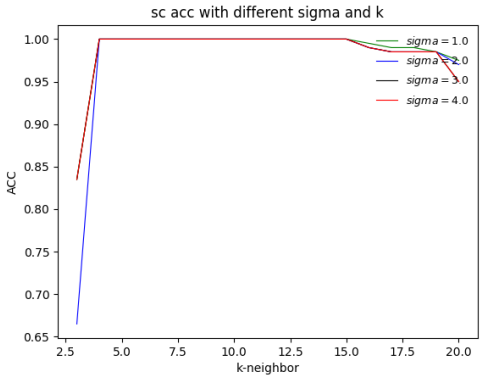


图 2.1 聚类性能随 k 变化

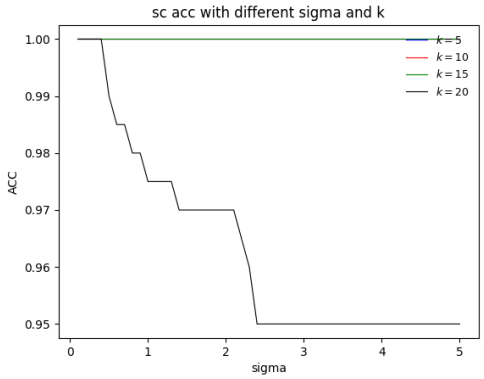


图 2.2 聚类性能随 sigma 变化

从图 2.1 可以看到，在σ固定时，随着 k 的增大，聚类性能总体呈现出先上升后下降的趋势。这是因为，当 k 比较少时，亲和矩阵非常稀疏，较少的包含整幅图的信息，聚类性能不佳。随着 k 的增大，图的信息得以较好保存。当 k 过大，信息冗余，对分类不利。这正好也对应了表 1，表 1 中全连图的性能时最差的，全连图即 k 取样本数减 1。

图 2.2 显示当 k 取 5、10、15 时，随着σ变化，聚类性能为 1 不发生改变。当 k 取 20 时，随着σ的增大，聚类性能将下降。当 k 比较大时，图结构的信息冗余，随着σ的增大，图信息会丢失一部分，这不利于分类。

将测试时生成的所有聚类性能绘制出图 2.3，从图 2.3 中可以看到在绝大多

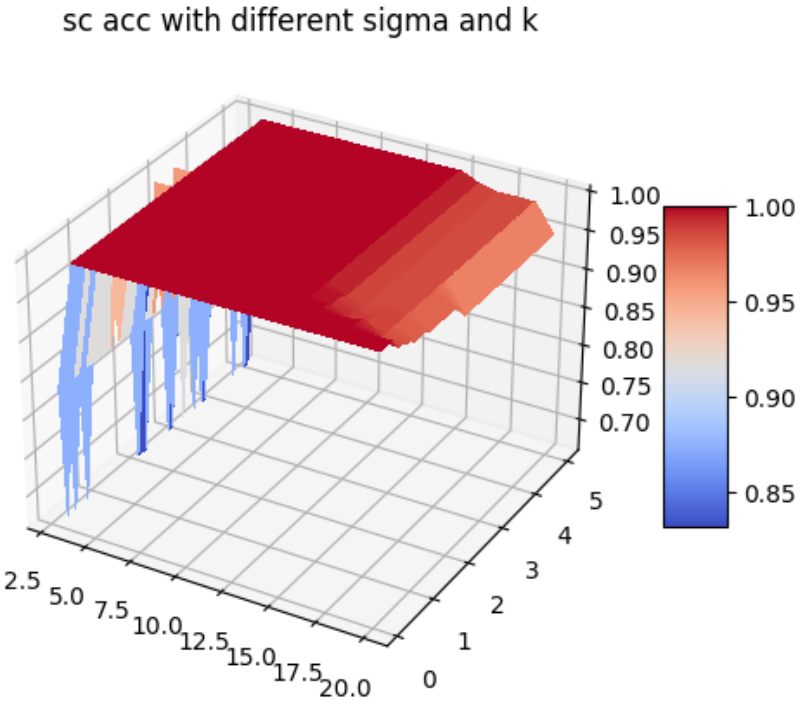


图 3 性能图，x 为 k，y 为 sigma

数情况下，聚类性能比较好。在 k 比较小的时候，σ取得不恰当时，聚类性能会比较低。因此在运用谱聚类算法时，需要考虑选择合适的参数。