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

1. 初始化训练数据的权重分布
2. 对于

(2a) 使用具有权值分布的训练数据，学习基本分类器

(2b) 计算在训练数据集上的加权分类错误率

(2c) 计算的贡献系数

(2d) 更新训练数据集的权重分布

1. 构建基本分类器的线性组合
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
5. 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

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、或构造亲和矩阵W、度矩阵D。计算归一化拉普拉斯矩阵
2. 选择前k个最小特征值对应的向量
3. 构建
4. 将每一行向量进行单位化
5. 将的行向量看成维样本，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。
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：

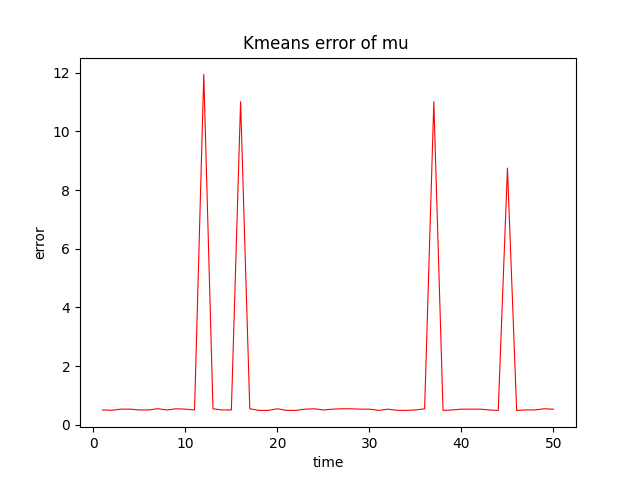
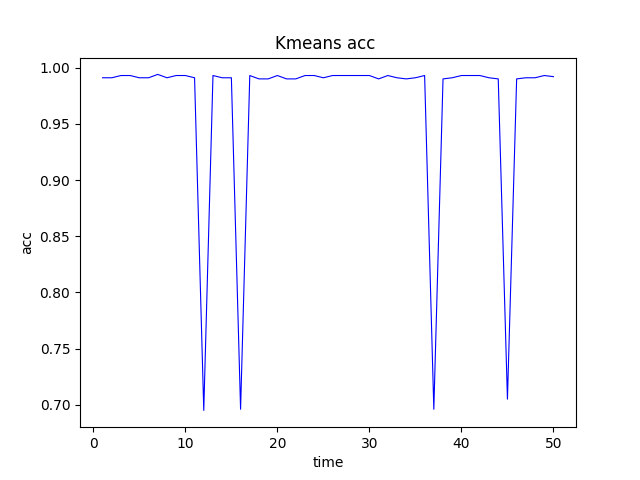


图1.1 聚类性能 图1.2 类中心二范数误差

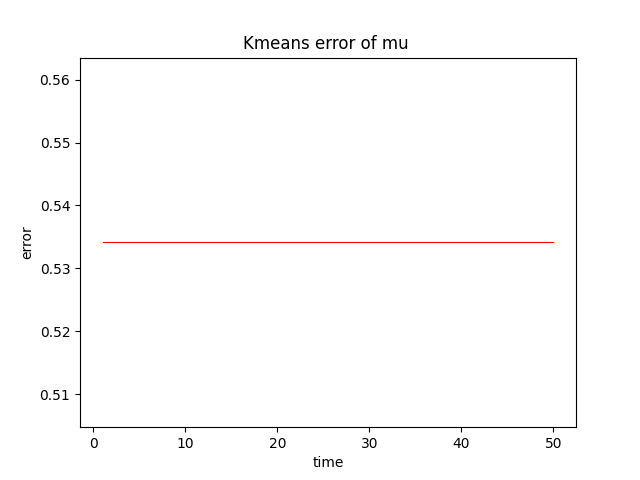
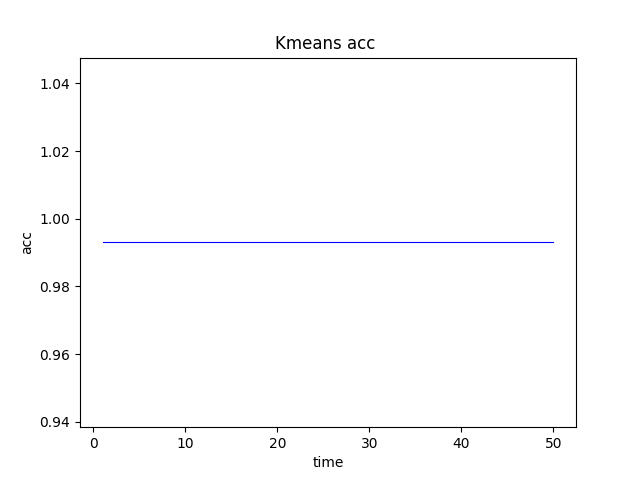
通过上图可以发现，在大多数情况下，聚类性能为0.99以上，最高为0.993，类中心二范数累计误差为0.54左右。但是存在个别情况，使得分类性能急剧下降到0.7左右，通过可视化图，发现出现这种情况是因为有两个类中心初始到同一个簇里，导致实际上只有4个聚类中心，聚类结果出现偏差。

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：

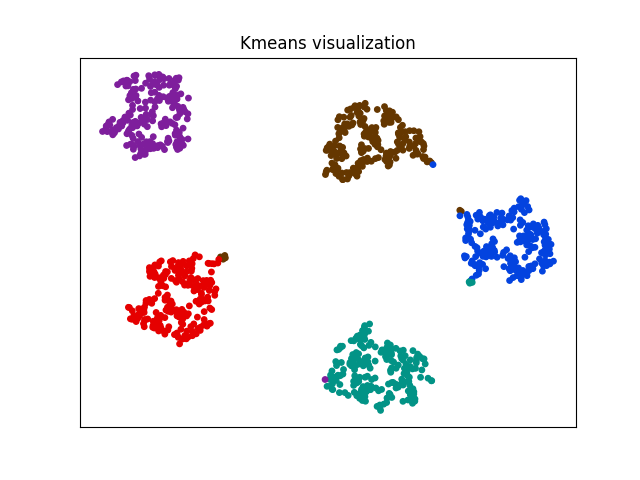
[9.0361641 , 0.01224488] [0.89604309, 4.0497816 ] [ 1.0203202 , -1.08880343] [ 5.55820209, -4.59115453] [6.1568884 , 4.40885642]

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，10近邻构图

，阈值的聚类性能。

|  |  |  |  |
| --- | --- | --- | --- |
| 算法\构图方式 | Knn | 全连图 |  |
| 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构图以及不同的对聚类性能的影响。分析时knn的k从3增长到18，步长为1。从0.1增长到5.0，步长为0.1。图2.1展示了

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。

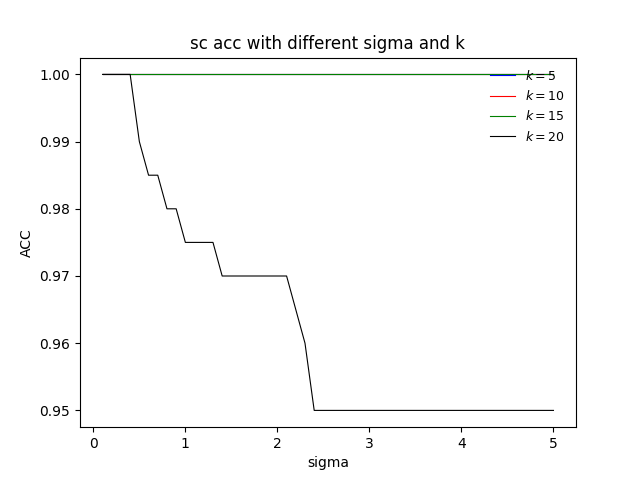
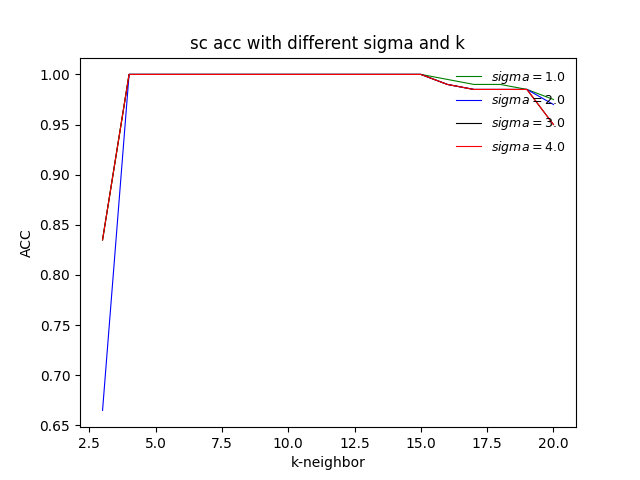


图2.1 聚类性能随k变化 图2.2 聚类性能随sigma变化

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，随着变化，聚类性能为1不发生改变。当k取20时，随着的增大，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

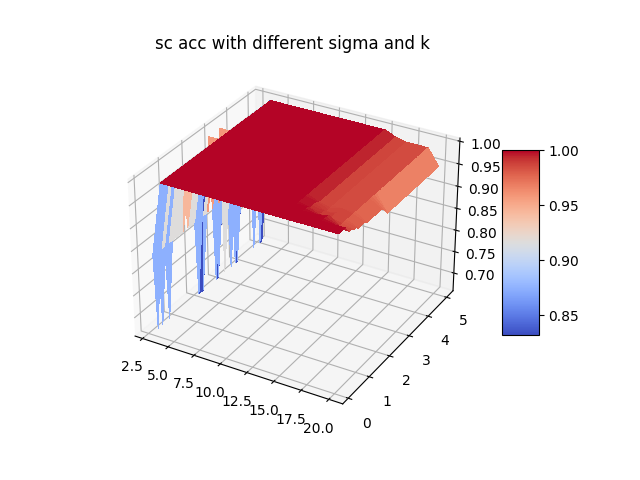


图3 性能图，x为k，y为sigma

数情况下，聚类性能比较好。在k比较小的时候，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。