《云数据管理》智能数据分析大作业

组员：

软件61 2015080118 崔殷庇

软件61 2015012351 丁润语

题目1：分类

5分类

读了一篇论文 Deep learning for time series classification: a review，论文中作者比较了9个深度学习模型对时间序列分类任务的效果。研究表明对于时间序列分类问题 Resnet 和 FCN 是最有效的深度学习模型（虽然深度学习方法目前没有超越非深度学习方法），于是我们按照论文实现了 Resnet 和 FCN，最终集成两个模型得到了结果。

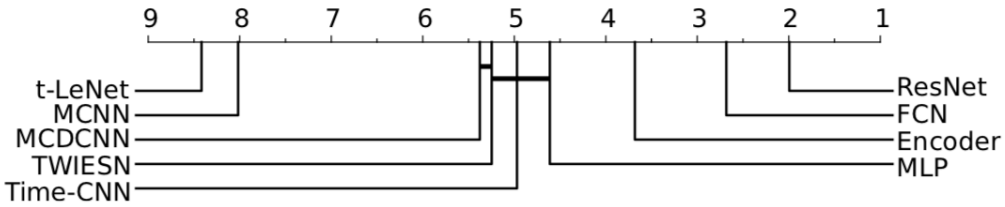


Fig. 7: Critical difference diagram showing pairwise statistical difference comparison of nine deep learning classifiers on the univariate UCR/UEA time series classification archive.

算法

- **模型1- Resnet**

  预处理：z-normalization

  层数：11

  权重初始化方法：glorot's uniform initialization

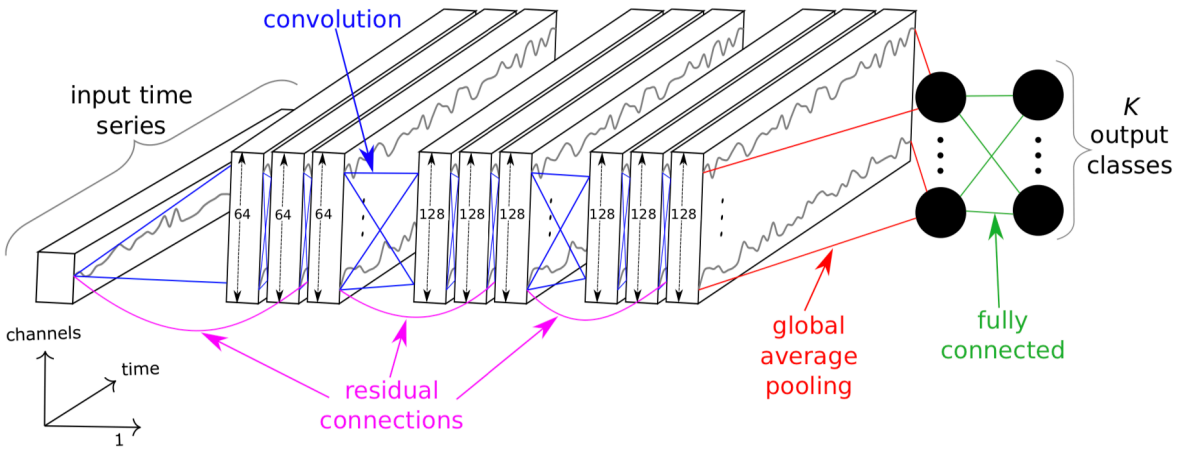
  损失函数：cross-entropy

  Epoch：200

  Optimizer：Adam

  学习率：0.001 + scheduler

  执行方法：`python train.py -m resnet -e 200`



- **模型2 - FCN(Fully Convolutional Neural Network)**

  预处理：z-normalization

  层数：5

  权重初始化方法：glorot's uniform initialization

  损失函数：cross-entropy

  Epoch：500

  Optimizer：Adam

  学习率：0.001 + scheduler

  执行方法：`python train.py -m fcn -e 500`

实验结果

(5折交叉验证，准确率的平均值)

Resnet: **79%**

FCN: **80%**

Ensemble Resnet and FCN: **81%**

分析和评价

跟预期不一样，FCN的准确率比Resnet略高了一点。两个模型都在标签为2和3(index为1和2)的数据分类效果差，precision和recall都比较低，很大程度地拉下了总准确率。若能分类好标签为2和3的数据，准确率会大大提升。最终集成两个模型得到了最高的准确率。

```
                    ## FCN模型的一份报告 ##
[[433   4   1   2   2]
 [  4 316 143   0  16]
 [  0 110 311   0  23]
 [  0   1   0 378  88]
 [  0   8  13  40 407]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       442
```

```
           1       0.72      0.66      0.69       479 #低precision和recall#
           2       0.66      0.70      0.68       444 #低precision和recall#
           3       0.90      0.81      0.85       467
           4       0.76      0.87      0.81       468

    accuracy                           0.80      2300
   macro avg       0.81      0.80      0.80      2300
weighted avg       0.81      0.80      0.80      2300
accuracy:  0.8021739130434783
```

分析一下Resnet和FCN强的原因。在深度学习中时间序列跟计算机视觉一样，较深层的网络比浅层的网络有效，Resnet共有11层，FCN共有5层，是较深的模型。而且Resnet的shortcut residual connection可以有效的解决梯度消失的问题。Resnet和FCN用的GAP层也是一个好的设计，论文的对比实验表明GAP比起FC层和Attention机制效果好。虽然目前的深度学习方法还没有达到非深度学习方法的准确率(HIVE-COTE)，估计即将会超过。

2分类

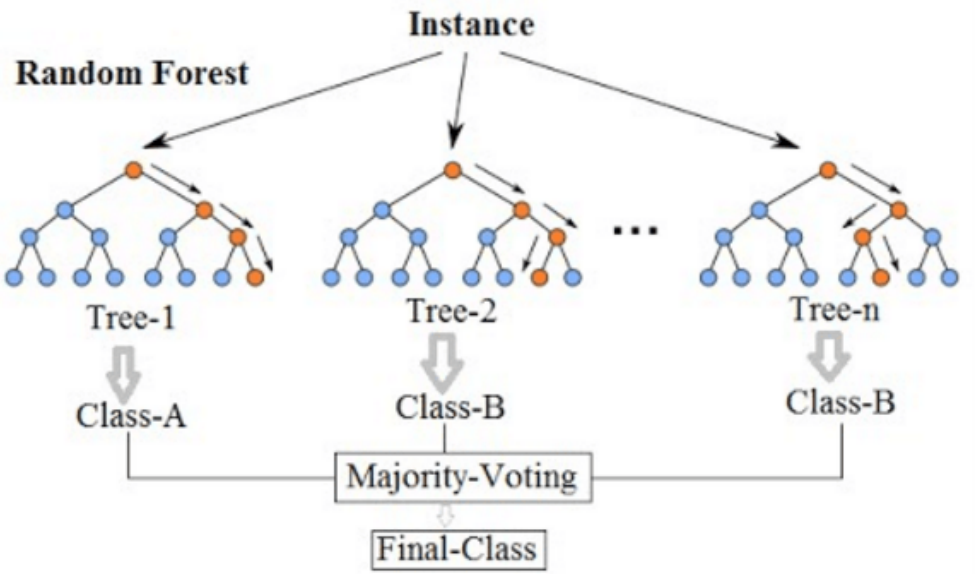
预处理：无

算法

随机森林 Random Forest (with 20 trees)

随机森林是一个包含多个决策树的分类器， 其输出的类别是由个别树输出的类别的众数而定。

随机森林的构建大致如下：首先利用bootstrap方法又放回的从原始训练集中随机抽取n个样本，并构建n个决策树；然后假设在训练样本数据中有m个特征，那么每次分裂时选择最好的特征进行分裂 每棵树都一直这样分裂下去，直到该节点的所有训练样例都属于同一类；接着让每颗决策树在不做任何修剪的前提下最大限度的生长；最后将生成的多棵分类树组成随机森林，用随机森林分类器对新的数据进行分类，按多棵树分类器投票决定最终分类结果。



执行方法：`python random_forest.py`

实验结果

(5折交叉验证，准确率的平均值)

94.1%

分析和评价

随机森林有以下的优点：随机森林的方法由于有了bagging，也就是集成的思想在，实际上相当于对于样本和特征都进行了采样，所以可以处理高维特征，避免产生过拟合，模型训练速度比较快。并且随机森林对数据集的适应能力强，既能处理离散型数据，也能处理连续型数据，数据集无需规范化。随机森林反而对少量数据集和低维数据集的分类不一定可以得到很好的效果。我们的178维时间序列算高维数据，而且数据集也不小，于是很适合用随机森林进行分类，效果也不错。另外，训练速度比深度学习方法明显快。我们实现的随机森林跟sklearn的随机森林(97.1%)比较我们的算法准确率虽然低了一点，但很接近。

题目2：聚类

主要工作

- 实现了KMeans， KMeans++(划分聚类)，ward(层次聚类)，DBSCAN(密度聚类)四种聚类算法。
- 实现类间距，类内矩和轮廓系数指标，评价聚类结果的好坏。对
- 实验结果进行降维可视化，从质量角度评价结果。

算法实现

KMeans算法

给定k个初始划分组，通过不断迭代改变分组，直至准则函数收敛。

其中，K的选取通过实验选择轮廓系数最大的K值。

- 随机选择K个点作为簇族中心，赋予K个标签；
- 对剩余的每个样本点，根据其与各簇中心的距离，将它赋给最近的簇；
- 重新计算每个簇的平均值作为簇族质心；重复2-3步骤，直到准则函数（误差的平方和SSE作为全局的目标函数）收敛，质心不发生变化。

KMeans++

由于KMeans算法中，初始点的选择对于最终的结果至关重要。KMeans++算法优化了这一随机选择过程，以下是具体步骤。

- 随机从m个样本点中选择一个样本作为第一个簇的质心C1；
- 计算所有的样本点到质心C1的欧氏距离；
- 以离质心C1的距离作为权重，**从概率分布中随机选取多个点作为第二个质心C2的候选点集，再通过距离计算选取最合适的点作为第二个质心C2**。其中，离C1越远的点，被选择的概率越大。重新计算各点与质心的距离；
- 重复上述计算，直至初始的K个质心被选择出。

层次聚类

采用自底而上的聚类方法。

- 将所有样本点作为单独的一个初始类，计算两两类之间的链接(ward linkage)距离；

- 根据类与类之间的链接寻找同类，不断合并距离最小的类别，**通过维护一个堆队列加快距离矩阵的更新**。；
- 最后根据需要的类别个数，形成一个最终的簇族。

DBSCAN

- DBSCAN首先根据邻域参数 $(\epsilon, MinPts)$ 确定样本集合D中所有的核心样本点。成为核心样本点的条件：邻域内含有不少于$MinPts$的样本数；
- 从核心对象集合中任意选取一个核心对象作为初始点，**维护栈队列，通过深度优先搜索**找出其密度可达的样本生成聚类簇，构成第一个聚类簇C1；
- 将C1内多有核心对象从P中去除，再从更新后的核心对象集合任意选取下一个种子样本。
- 重复2-3步骤，直到核心对象被全部选择完闭。

评价指标

- 类内矩

  向量到同一簇内其他点**不相似程度**的平均值

  $WSS = \sum_{i} \sum_{x \in C_i} (x - m_i)^2$

- 类间距

  向量到其他簇的平均**不相似程度**的最小值

  $BSS = \sum_{i} |C_i|(m - m_i)^2$

- 轮廓系数：

  类间距和类内距综合考量指标

  计算 a(i) = average(i向量到所有它属于的簇中其它点的距离)

  计算 b(i) = min (i向量到与它相邻最近的一簇内的所有点的平均距离)

  单样本点的轮廓系数：$S(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}}$

  将所有点的轮廓系数求平均，得到该聚类结果的总轮廓系数。

难点与解决方案

- 问题：在使用了KMeans++方法初始化原始样本点时，仍然存在样本点选择不恰当，结果不稳定的现象。相较于kmeans方法，改善不明显。

  解决方法：通过调研文献，使用了渐进求精的初始点选择方法。首先，通过距离的概率分布随机选择2+log(n_clusters) 个点作为候选集，再计算每个候选集如果成为初始点带来了总势变化。选择降低势最多的点作为初始点。这种方法在实验中表现出鲁棒性。

- 问题：在层次聚类方法中，计算类的ESS值计算量大，效率低，导致算法运行慢。

  解决方法：利用Lance-Williams Algorithm，利用简化公式有效降低计算复杂度。

实验结果与评价

KMeans

类别数	类内距	类间距	轮廓系数
4	14.612474	35.867328	0.523039
5	13.692232	34.281603	**0.535546**
6	13.273626	30.659212	0.497575

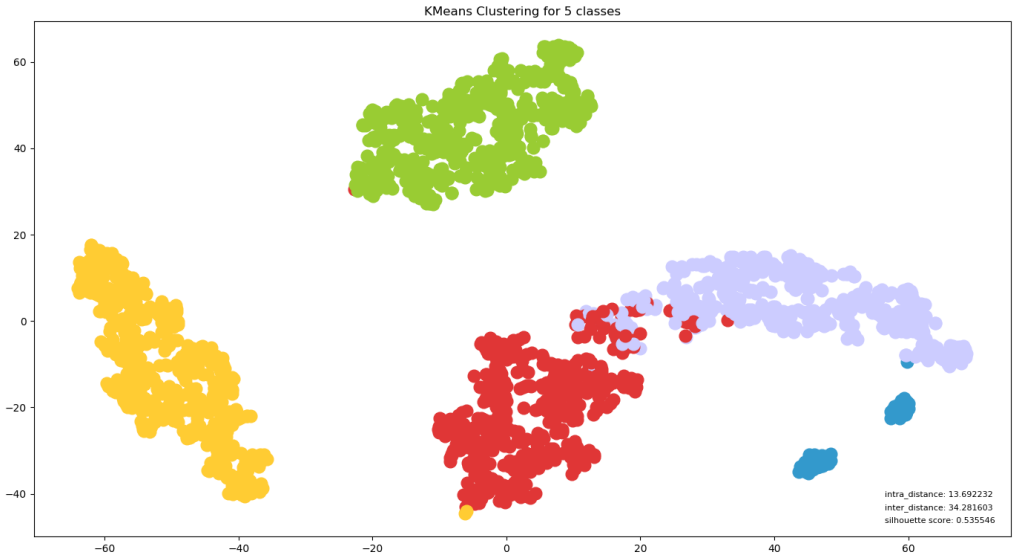
> 由于KMeans方法存在不稳定性，这里列出的是多次实验中的最好结果。

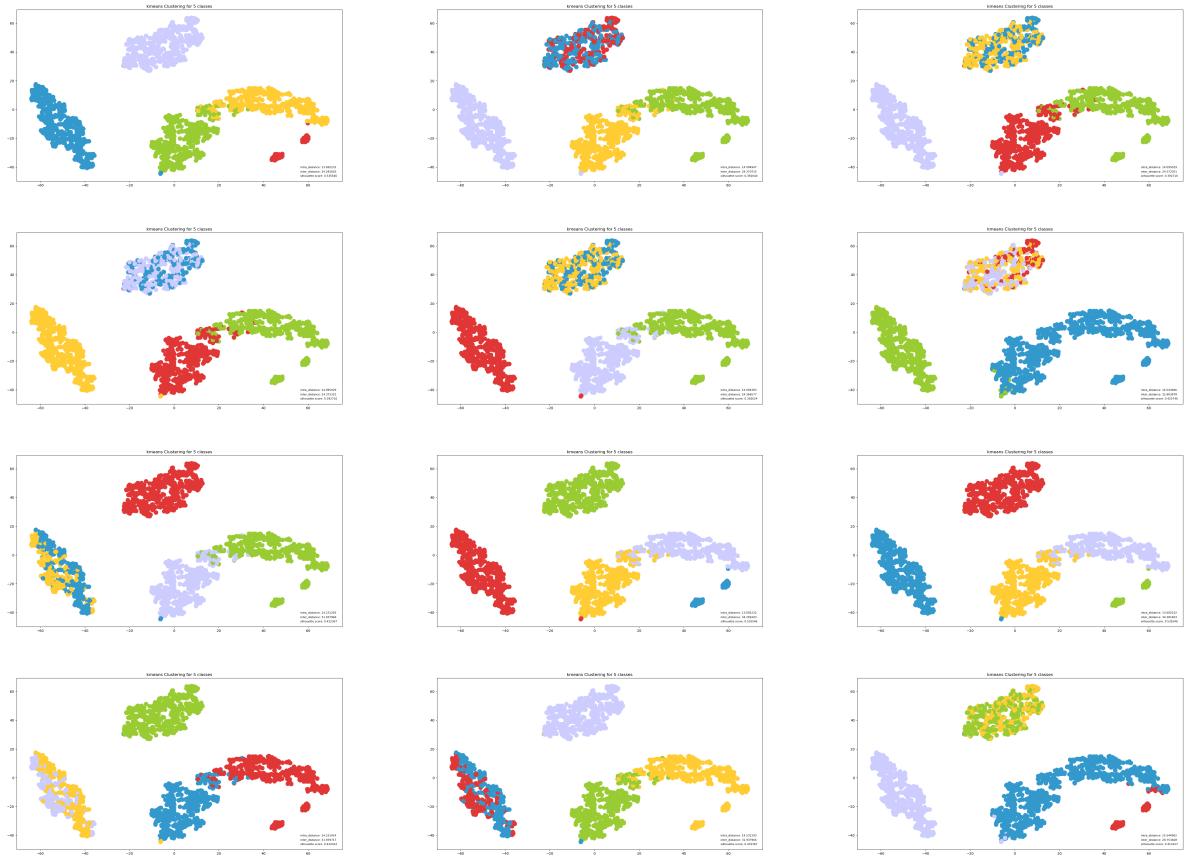
由上表知，类别数为5时，轮廓系数最高，综合表现最好。

此时，各聚类的平均质心为

使用 k=5 作为最好结果，进行**可视化分析（方法：t-SNE）**：

- 从下图图一上可以看到，分类结果较好。存在一些类与类重合的部分，比如红色类和紫色类的部分重叠，有一黄色样本点与红色样本点非常接近。通过多次试验发现，该现象是因为本身数据点之间的类别信息模糊，且降维时由于维度信息缺失，在不同的随机种子和不同方法下，会导致不同的可视化结果，造成了异常点的现象。
- 从下图图二可以看到，在12次的重复实验下，kmeans方法稳定度低，仅有3次产生了最好结果。其余9次分类结果的轮廓系数在0.39-0.42左右，因为初始点的选择问题表现较差，在可视化图中，表现为类间距小，不同类的样本点重合率大。





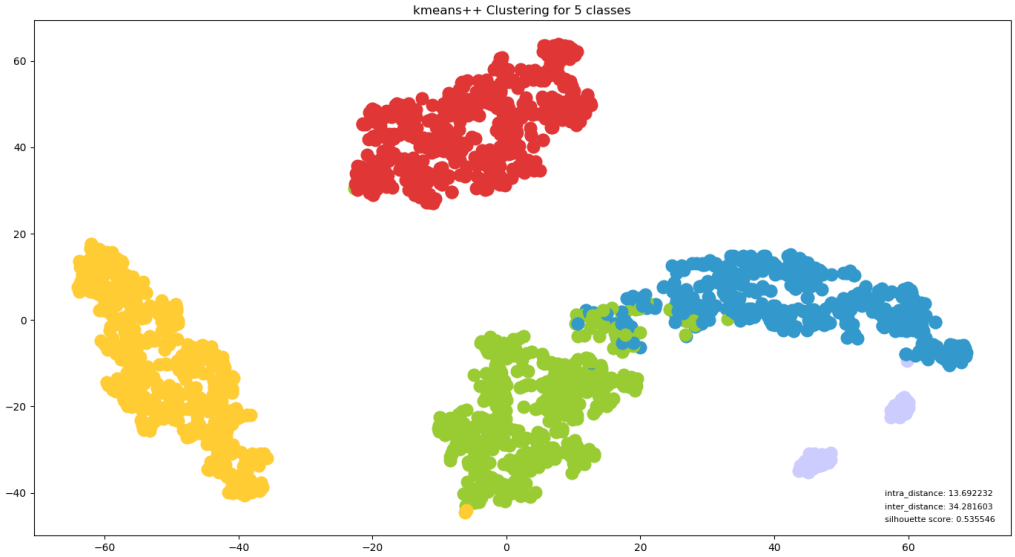
KMeans++

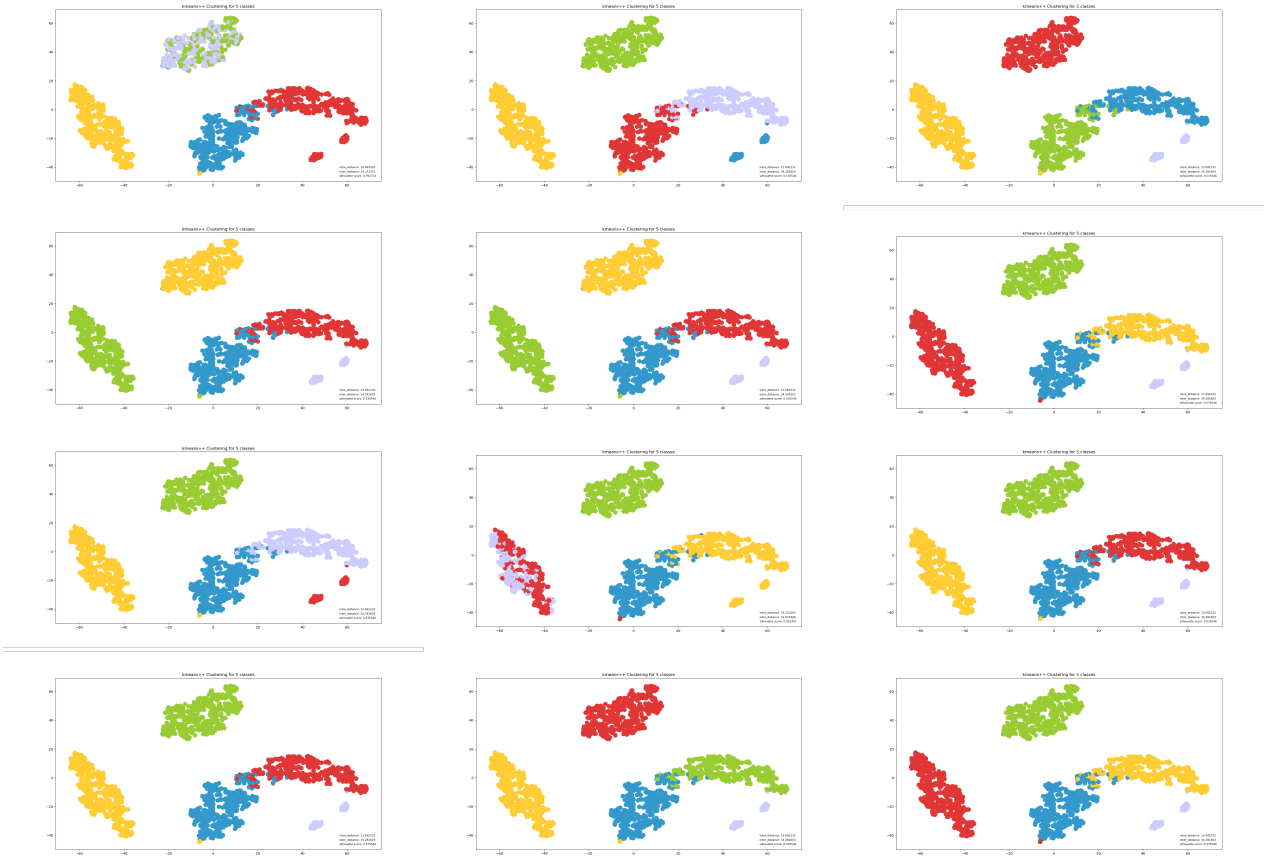
不同类别下的最好结果与KMeans相同。

此时，各类别的质心（患者最可能出现的区域为）：

```
cluster1 [-24.182   , -19.224   , -14.65    , -25.316   , -13.454   , -51.008   ,
-50.99    ]
cluster2 [-21.902934, -37.717832, -37.306997, -22.530474, -49.112866, -54.693002,
-55.435665]
cluster3 [-53.49    , -47.276   , -51.52    , -55.182   , -61.262   , -73.92    ,
-75.026   ]
cluster4 [-22.719836, -27.693251, -25.656441, -23.650306, -36.063394, -54.298568,
-55.353783]
cluster5 [ -0.176470, -40.382352, -40.735294,  -4.970588, -51.926470, -54.779411,
-56.161764]
```

使用 k=5 作为最好结果，进行**可视化分析（方法：t-SNE）**：





结论

- 从上图图一验证，kmeans++的最好结果和kmeans一致。且最好结果优于其他两种算法。
- 从上图图二知，kmeans++方法的结果稳定度高，在12次重复实验中有10次产生了最好结果，仅有两次产生偏离。说明了初始点选择对于划分聚类方法的重要性和kmeans++初始化方法的有效性。
- 简单高效，时间复杂度、空间复杂度低。
- 由于不可避免的初始化的不稳定性和贪心思想，容易陷入局部最优解。

层次聚类

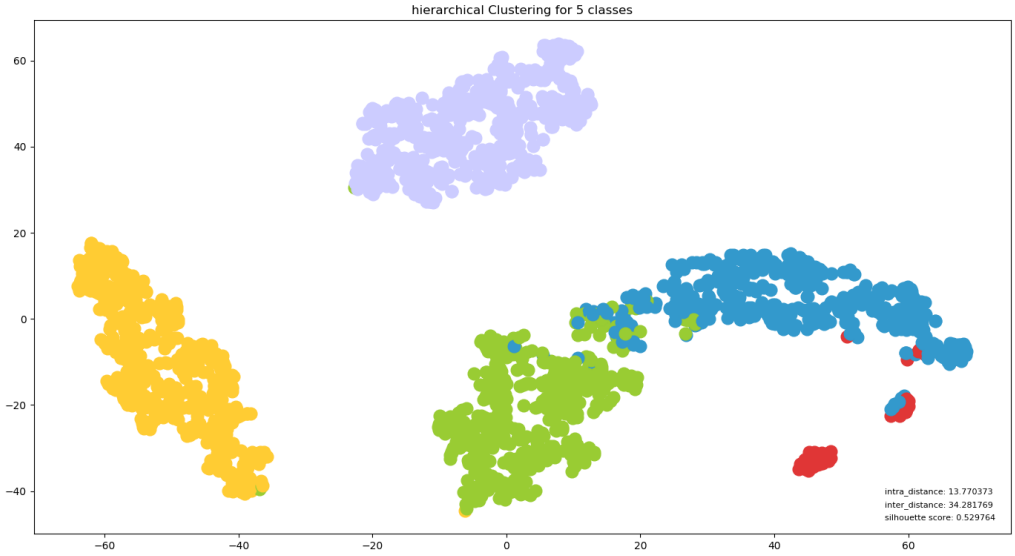
类别数	类内距	类间距	轮廓系数
4	14.660488	35.700898	0.517212
5	13.770373	34.281769	**0.529764**
6	13.373568	23.765849	0.406031

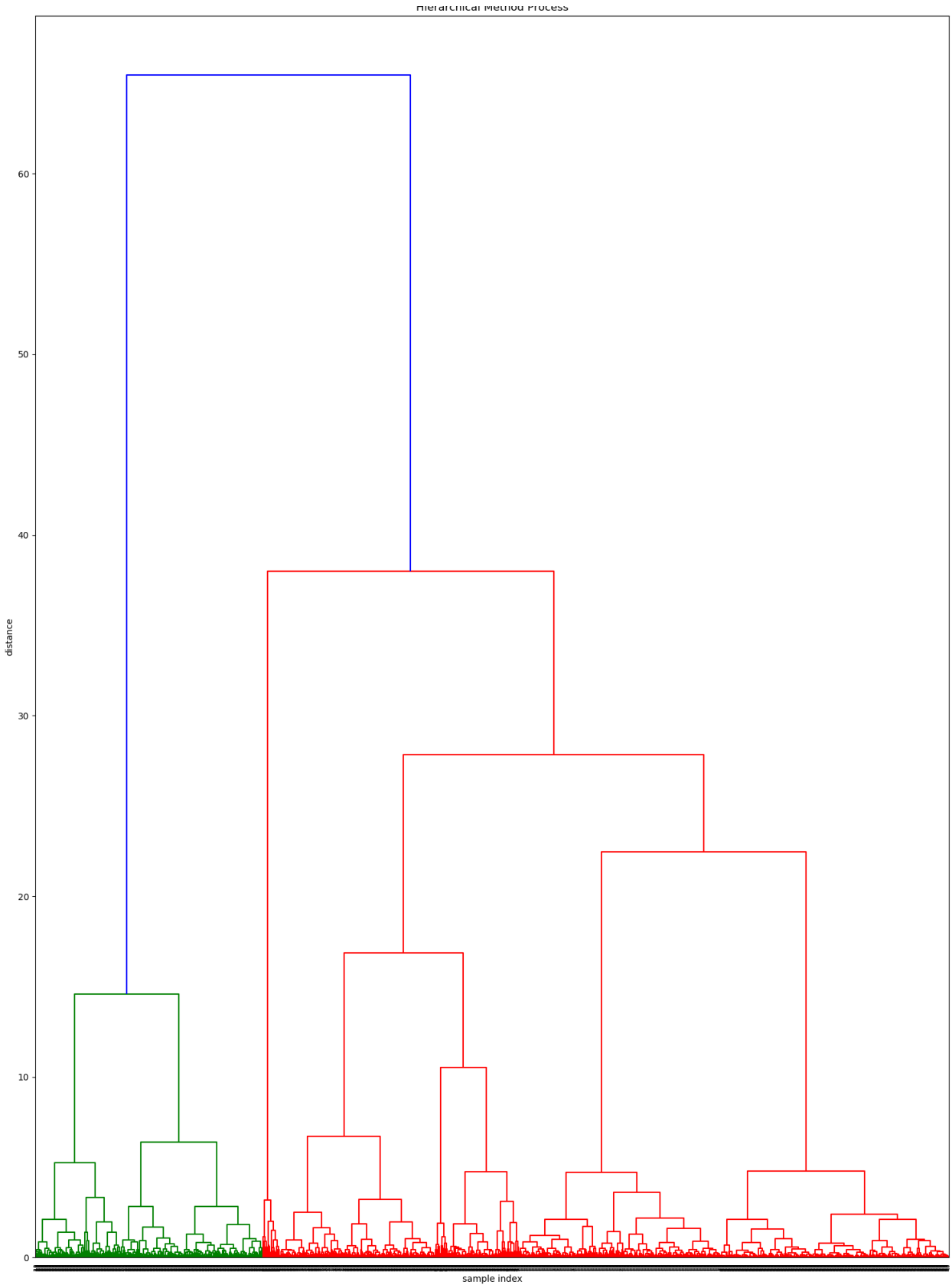
由上表可知，类别数为5时，聚类表现相对较好。

此时，各类别的质心（患者最可能出现的区域为）：

```
cluster1 [  1.421875, -40.6875  , -40.9375  ,  -6.546875, -52.046875, -54.9375  ,
-55.9375  ] cluster2 [-22.752100, -27.447478, -25.514705, -23.655462, -35.825630,
-54.22899 , -55.432773]
cluster3 [-24.169014, -19.219315, -14.631790, -25.305835, -13.382293, -51.012072,
-50.977867]
cluster4 [-21.954643, -37.555075, -37.      , -22.215982, -48.844492, -54.704103,
-55.373650]
cluster5 [-53.49    , -47.276   , -51.52    , -55.182   , -61.262   , -73.92    ,
-75.026   ]
```

使用 k=5 作为最好结果，进行**可视化分析（方法：t-SNE）**：





- 层次聚类方法的轮廓系数较高，可视化结果也比较好。
- 但是，层次聚类算法时间复杂度高啊，$o(m^3)$，使用堆进行改进后，算法复杂度仍有$o(m^2lgm)$。在实验中，明显慢于其他两种方法。
- 且由于贪心算法，容易陷入局部最优解。

DBSCAN

类别数	类内距	类间距	轮廓系数
4	23.413681	46.363273	0.400806
5	16.153147	39.129521	**0.511067**
6	16.654908	38.265790	0.480264

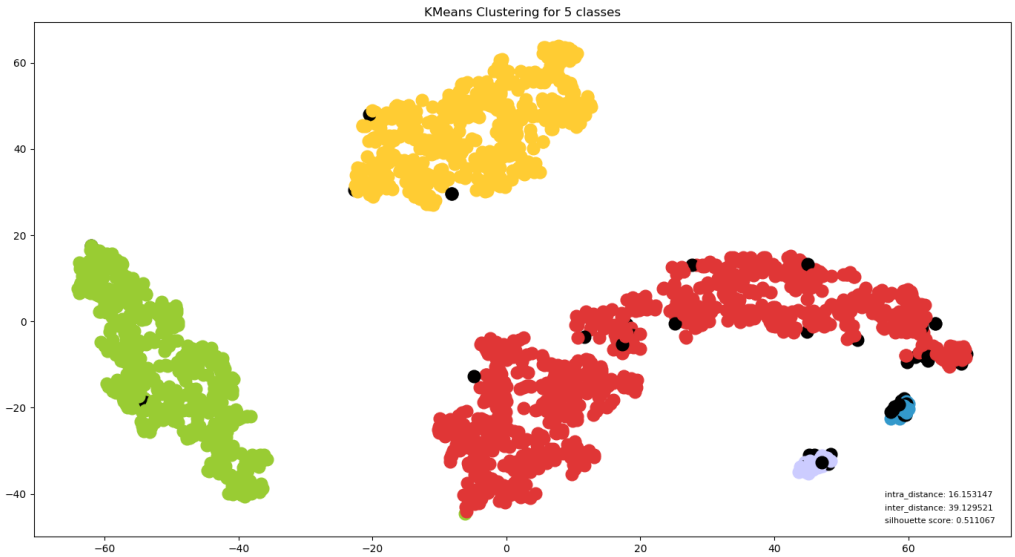
> 注：每一类别下的最好结果，通过调整邻域参数$(\epsilon, MinPts)$得到。

由上表可知，类别数为5时，聚类表现相对较好。此时，$\epsilon = 9, MinPts = 6$。

此时，各类别的质心（患者最可能出现的区域为）：

```
cluster1 [-22.322368, -32.317982, -31.038377, -23.131578, -42.085526, -54.448464,
-55.361842]
cluster2 [-24.175050, -19.231388, -14.659959, -25.287726, -13.426559, -51.002012,
-50.969818]
cluster3 [-53.501006, -47.265593, -51.523138, -55.173038, -61.277666, -73.877263,
-75.030181]
cluster4 [  1.625   , -40.      , -40.3125  , -17.875   , -50.5625  , -55.0625  ,
-55.8125  ]
cluster5 [  3.290322, -40.935483, -41.645161, 1.41935484, -53.806451, -55.806451,
-56.032258]
```

使用 k=5 作为最好结果，进行**可视化分析（方法：t-SNE）**：



其中，黑色点为噪声点。

结论

- DBSCAN方法的轮廓系数较高，可视化结果也比较好。
- 可以搜索出噪声点，排除噪声的干扰，鲁棒性高。
- 对于最优邻域参数的探索比较复杂。

参考文献

[1] Arthur, D. and Vassilvitskii, S."k-means++: the advantages of careful seeding". ACM-SIAM symposium on Discrete algorithms. 2007

[2] Ismail Fawaz, Hassan et al. "Deep Learning for Time Series Classification: a Review." Data Mining and Knowledge Discovery 33.4 (2019): 917–963. Crossref. Web.