机器学习导论第二次作业

韦俊林 (201928016029023)

2020 年 4 月 22 日

1 简述题

1.1 请简述谱聚类的算法思想，写出一种谱聚类算法的计算步骤，指出哪些因素会影响谱聚类的结果。

我们习惯于将聚类看成由一个一个孤立的点组成，而谱聚类是从图切割的角度，找到一种合理的方式分割图，即连接不同子图的边的权重尽可能小，子图内部权重尽可能大。这样分割成若干个子图，就是若干个类。

谱聚类算法核心步骤：

1. 利用点对之间的相似性相互连接构建图，抽象成亲和度矩阵；
2. 利用亲和度矩阵构建拉普拉斯矩阵；
3. 求解拉普拉斯矩阵最小的特征值对应的特征向量，通常舍弃零特征所对应的分量全相等的特征向量；
4. 由这些特征向量构成样本点的新特征，采用 K-means 等聚类方法完成最后的聚类。

影响谱聚类结果的因素主要有：

a. 根据数据集构造的图是谱聚类算法的核心，图的边以何种规则确定其权重（定量的确定相似性），图如果采用局部连接的方式构建，其邻近阈值的选择。

b. 谱聚类算法涉及到计算特征值、特征向量，如果矩阵规模太大，或是构造的矩阵是病态的，由计算引起的误差也会极大的影响结果。

c. 由特征向量构成的样本点的新特征之后采用何种聚类算法完成最后的聚类。

d. 聚类的数目本身就是一个问题，因此也会影响谱聚类的结果。

1.2 请指出数据聚类存在哪些挑战性问题。

聚类算法主要存在以下几个方面的挑战：

a. 可伸缩性，聚类算法无论是对数据集的大小或是对类别的多少都应有效。

b. 具有处理不同数据类型的能力，不同数据类型是指数值型数据与非数值型数据，连续数据与离散数据，抑或是混合类型的数据。

c. 能对任意形态的数据进行聚类，除了对传统的星云状离散的数据有效外，还得对任意形状的簇，形如球状、同一流形上的数据能够有效的聚类。例如，经典的 K 均值算法无法将以某点为圆心，不同半径围成的圆为不同类别的数据进行聚类。

d. 处理高维数据，在高维空间，随着维数的增加，具有相同举例的两个样本其相似程度可能相差很远，尤其是高维稀疏数据。

e. 对噪声的鲁棒性，在实践中绝大多数样本集都包含噪声、空缺、部分未知属性、孤立点、甚至错误的数据，这些“野点”对结果往往造成极大的影响。

f. 具有约束的聚类，在实际应用中，通常需要在某种约束条件下进行聚类，既满足约束条件，以希望有高聚类精度。

g. 对初始输入参数的鲁棒性，具有自适应的类别数的判定能力。

h. 希望聚类的结果能被用户所理解，并能带来经济效益。

1.3 关于前向神经网络，请描述误差反向传播算法的原理；结合三层前向神经网络，给出权重更新公式。

误差反向传播算法 (BP) 基本思想是利用输出后的误差估计输出层的的前一层的误差，在用这一层的误差估计更前一层的误差，依此类推，从而获取所有其他各层的误差估计。

对于三层前向神经网络，第 k 个样本，从输入层到输出层节点 j 的输出有：

$$z_j^k = f\left(\sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)\right)$$

其中，$f(*)$ 为激活函数，i 为遍历输入层结点，h 为遍历隐含层结点。w 为下标所连接两个结点的边的权重，x_i 为输入层输入数据。

假设第 k 个样本输出层结点 j 输出的真实值为 t_j^k，则误差函数有：

$$E(w)^k = \frac{1}{2}\sum_{j,k}\left\{t_j^k - f\left(\sum_h w_{hj} f\left(\sum_i w_{ih} x_i^k\right)\right)\right\}^2$$

从而有隐含层到输出层的连接权重调节量：

$$\Delta w_{hj} = -\eta\frac{\partial E}{\partial w_{hj}} = \eta\sum_k \delta_j^k y_h^k$$

$$\delta_j^k = -\frac{\delta E}{\delta net_j^k} = f'(net_j^k)\Delta_j^k$$

$$net_j^k = \sum_h w_{hj} f(\sum_i w_{ih} x_i^k)\,, \quad \Delta_j^k = t_j^k - z_j^k$$

输入层到隐含层的连接权重调节量：

$$\Delta w_{ih} = -\eta\frac{\partial E}{\partial w_{ih}} = \eta\sum_k \delta_h^k x_i^k$$

$$\delta_h^k = f'(net_h^k)\Delta_h^k$$

$$net_h^k = \sum_i w_{ih} x_i^k \ , \quad \Delta_h^k = \sum_j w_{hj} \delta_j^k$$

总结上述，BP 算法任意层连接结点 i 与结点 j 的权重修正量为：

$$\Delta w_{ij} = \eta \sum_{all\ samples} \delta_j y_i$$

其中 δ_j 表示结点 j 从后一层收集到的误差，y_i 表示结点 i 向结点 j 传输的数据，上述的 η 为学习率。

1.4 请描述 CNN 网络结构，并结合前向计算过程描述网络待学习的参数。

CNN 网络结构包括输入层、输出层以及若干个隐含层，而隐含层由若干个卷积层、池化层以及全连接层构成，一般卷积层与池化层成对组成，全连接层放在最后面。卷积层主要功能在于特征提取，池化层可以降低数据维度，可以避免过拟合而且还能减少计算量，增强局部感受野，从而更容易获取抽象的特征。全连接层用于最后的分类。因此整个 CNN 网络待学习的参数有卷积核、全连接层的连接权重。

1.5 请给出多维缩放 (MDS) 算法的问题描述（Problem Formulation）、学习准则和详细的算法推导过程。

MDS 是假定 m 维空间中的 n 个样本 $\{x_1, x_2, ..., x_n\} \subset R^m$ 在原始空间的距离矩阵维 $D \in R^{n \times n}$，目标是获得这 n 个样本在 d $(d < m)$ 维空间中的表示 $Z = \{z_1, z_2, ..., z_n\} \subset R^{d \times n}$。学习准则为假定降维后的样本仍保持两两之间的距离。

令降维后的样本是零均值化的，即

$$\sum_{i=1}^{n} z_i = 0 \subset R^d$$

根据学习准则有：

$$d_{ij}^2 = \|z_i - z_j\|_2^2 = \|z_i\|_2^2 + \|z_j\|_2^2 - 2 z_i^T z_j = b_{ii} + b_{jj} - 2bij$$

$$b_{ij} = z_i^T z_j \ , \quad i, j = 1, 2, ..., n$$

其中 d_{ij} 表示样本 x_i 到 x_j 的距离。

因此，元素距离平方矩阵有：

$$D_2 = \begin{pmatrix} d_{11}^2 & d_{12}^2 & \dots & d_{1n}^2 \\ d_{21}^2 & d_{22}^2 & \dots & d_{2n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1}^2 & d_{n2}^2 & \dots & d_{nn}^2 \end{pmatrix} = \begin{pmatrix} \|z_1 - z_1\|_2^2 & \|z_1 - z_2\|_2^2 & \dots & \|z_1 - z_n\|_2^2 \\ \|z_2 - z_1\|_2^2 & \|z_2 - z_2\|_2^2 & \dots & \|z_2 - z_n\|_2^2 \\ \vdots & \vdots & \ddots & \vdots \\ \|z_n - z_1\|_2^2 & \|z_n - z_2\|_2^2 & \dots & \|z_n - z_n\|_2^2 \end{pmatrix}$$

展开得；

$$D_2 = \begin{pmatrix} \|z_1\|_2^2 & \|z_1\|_2^2 & \dots & \|z_1\|_2^2 \\ \|z_2\|_2^2 & \|z_2\|_2^2 & \dots & \|z_2\|_2^2 \\ \vdots & \vdots & \ddots & \vdots \\ \|z_n\|_2^2 & \|z_n\|_2^2 & \dots & \|z_n\|_2^2 \end{pmatrix} + \begin{pmatrix} \|z_1\|_2^2 & \|z_2\|_2^2 & \dots & \|z_n\|_2^2 \\ \|z_1\|_2^2 & \|z_2\|_2^2 & \dots & \|z_n\|_2^2 \\ \vdots & \vdots & \ddots & \vdots \\ \|z_1\|_2^2 & \|z_2\|_2^2 & \dots & \|z_n\|_2^2 \end{pmatrix} - 2 \begin{pmatrix} z_1^T z_1 & z_1^T z_2 & \dots & z_1^T z_n \\ z_2^T z_1 & z_2^T z_2 & \dots & z_2^T z_n \\ \vdots & \vdots & \ddots & \vdots \\ z_n^T z_1 & z_n^T z_2 & \dots & z_n^T z_n \end{pmatrix}$$

令向量 c 为样本点新的表示的模的平方所组成的向量，B 为降维后样本的内积矩阵，即

$$c = \left(\|z_1\|_2^2, \|z_2\|_2^2, \dots, \|z_n\|_2^2\right)^T \in R^n$$

$$B = Z^T Z \in R^{n\times n}$$

从而元素距离平方矩阵化简得：

$$D_2 = ce^T + ec^T - 2B$$

其中 e 是 $d \times n$ 全为 1 的矩阵。
引入数据中心化矩阵：

$$H = I - \frac{1}{n} ee^T \in R^{n\times n}$$

进一步构造 $H^T D_2 H$，再经过化简得：

$$H^T D_2 H = H^T (ce^T + ec^T - 2B)H = -2B \quad \Rightarrow \quad B = -\frac{1}{2} H^T D_2 H$$

再对矩阵 B 进行特征值分解，最终得：

$$B = Z^T Z = U\Lambda U^T \quad \Rightarrow \quad Z = \Lambda_d^{1/2} U_d^T \in R^{d\times n}$$

$$\Lambda_d^{1/2} = diag(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_d}) \in R^{d\times d},\ U_d = (u_1, u_2, \ldots, u_d) \in R^{n\times d}$$

其中，Λ_i 表示矩阵 B 的由大到小排序的第 i 个特征值，同理 u_i 表示对应的特征向量。因此 $\Lambda_d^{1/2}$ 表示由矩阵 B 的前 d 个最大的特征值开根号后对应的对角矩阵，U_d 表示由前 d 个最大的特征值所对应的特征向量组成的矩阵。

1.6 请给出 LLE、LE 和 Isomap 算法的异同点，在此基础上给出流形学习的统一学习模型，并指出流形学习所存在的挑战性问题。

在数学上，流形用于描述一个几何形体，它在局部具有欧氏空间的性质，LLE、LE 和 Isomap 算法都围绕着流形在局部具有欧氏空间性质这个特性而设计。具体说来，给定数据集，通过最近邻等方式构造一个数据图 (data graph)，希望在点对之间在高维空间中的某种度量在低维空间中也得以保持。三个算法的不同点在于，LLE 侧重点在于高维空间中的样本线性重构关系在低维空间中均得以保持；Isomap 则是计算任意两点之间的测地距离，对于所有的任意点对，期望在低维空间中也得以保持；LE 算法是在每一个局部区域内，计算点与点之间的亲和度，期望点对亲和度在低维空间中也得到保持。

综上，流形学习统一形式化的学习模型，目标在于给定高维数据 $\{x_i\}_{i=1}^n \subset R^m$ 寻找其低维表示。
学习模型：

$$\{y_i\}_{i=1}^n \subset R^d \ (d < m)$$

$$\min_Y \ trace(YMY^T)$$

$$s.t. \quad YY^T = I \quad (Y = [y_1, y_2, \ldots, y_n] \in R^{d\times n})$$

任务是构造与数据图构造和局部描述紧密相关的矩阵 M。

当前流形学习所存在挑战性的问题主要有：(1) 低纬本质维数的确定；(2) 如何构建一个好的数据图；(3) 如何将新样本嵌入到已有的低维度结构中；(4) 超大规模的计算量。

1.7 请简述局部投影保持 (LPP) 算法的思想，写出算法的计算步骤。

局部保持投影是一个线性降维算法，是拉普拉斯映射 (LE) 的线性近似，但同时具备流形学习方法和线性降维方法的优点。其算法思想是构建原空间中各样本点对之间的亲和度关系，并在线性投影中保持这种关系比，在降维的同时保留原空间中样本的局部领域结构，即尽量避免样本集在投影空间中发散，保持原来的近邻结构，具体就是在低维空间中最小化近邻样本间的距离加权平方和。具体计算步骤是：

1. 输入样本 $X = [x_1, x_2, \ldots, x_n] \subset R^{m \times n}$，近邻参数 k，低维空间维度 d；

2. 确定 x_i 的 k 个近邻，确定亲和度矩阵 W，计算度矩阵 D；

3. 求解模型 $\min\limits_{V} tr(V^T X(D-W)X^T V), \quad s.t.\ V^T V = I$

4. 采用第 1 至第 d 个最小的特征值对应的特征向量组成低维度投影矩阵 W；

5. 输出 $V \in R^{m \times d}$。

2 计算机编程

2.1 第一题

首先，生成如图 1 数据集，现在假定 x_1 中的 200 个点属于第一类，x_2 中的 200 个点属于第二类，x_3 中的 200 个点属于第三类，x_4 中的 200 个点属于第四类，x_5 中的 200 个点属于第五类。

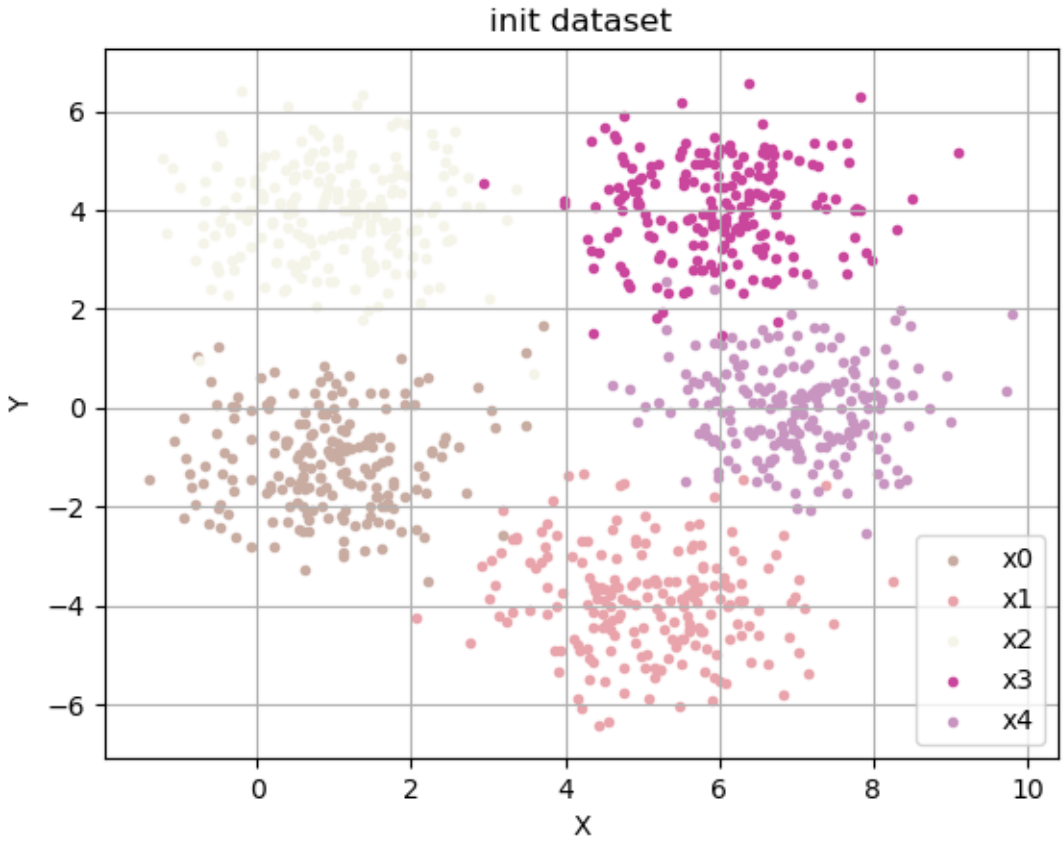

图 1: dataset

请写一个 K 均值聚类算法（给出代码），完成对数据的聚类，报告聚类精度和 NMI 指标。

实验报告：

(a) 基于 python 实现 PPT 上第一个 K 均值聚类算法，如图 2；

ML_model.py > K_Means

```
30
31      def v1(self,savefig,save_path):
32          print('starting K-Means clustering version 1.')
33          center=nd.array(self.center,ctx=self.ctx)
34          center_temp=center.zeros_like()
35          data=nd.array(self.x_data,ctx=self.ctx)
36          #y_pred=self.y_true.zeros_like()
37          data=nd.concat(data,nd.zeros((self.data_num,1),ctx=self.ctx),dim=1)
38          df=pd.DataFrame(columns=['x','y','label'])
39          print('-----------------')
40          print('init center:')
41          print(center)
42          itera=1
43          while(itera<=100):
44              center_temp[:]=center
45              for i in range(self.data_num):
46                  #delta_x^2 delta_y^2
47                  delta_xy=(center-data[i,0:-1])**2
48                  #delta_x^2+delta_y^2
49                  d_array=delta_xy[:,0]+delta_xy[:,1]
50                  #return min index
51                  data[i,2]=nd.argmin(d_array,axis=0)
52                  #y_pred=nd.argmin(d_array,axis=0)
53              #TODO
54              df=pd.DataFrame(data.asnumpy(),columns=['x','y','label'])
55              for i in range(self.labels_num):
56                  cluster_temp=df.loc[df['label']==i].values
57                  center[i]=nd.array(cluster_temp[:,:-1].mean(axis=0),ctx=self.ctx)
58              #----------
59              itera+=1
60              if (center==center_temp).sum().asscalar()==2*self.labels_num: break
61          print('pred center:')
62          print(center)
63          print('iteration:',format(itera))
64          print('-----------------')
65          if savefig : self.draw(pred_data=df,save_path=save_path)
66          self.y_pred=data[:,2].asnumpy()
67          return center.asnumpy()
```

图 2: K-Means code

(b) 实验结果对比图，图 3；

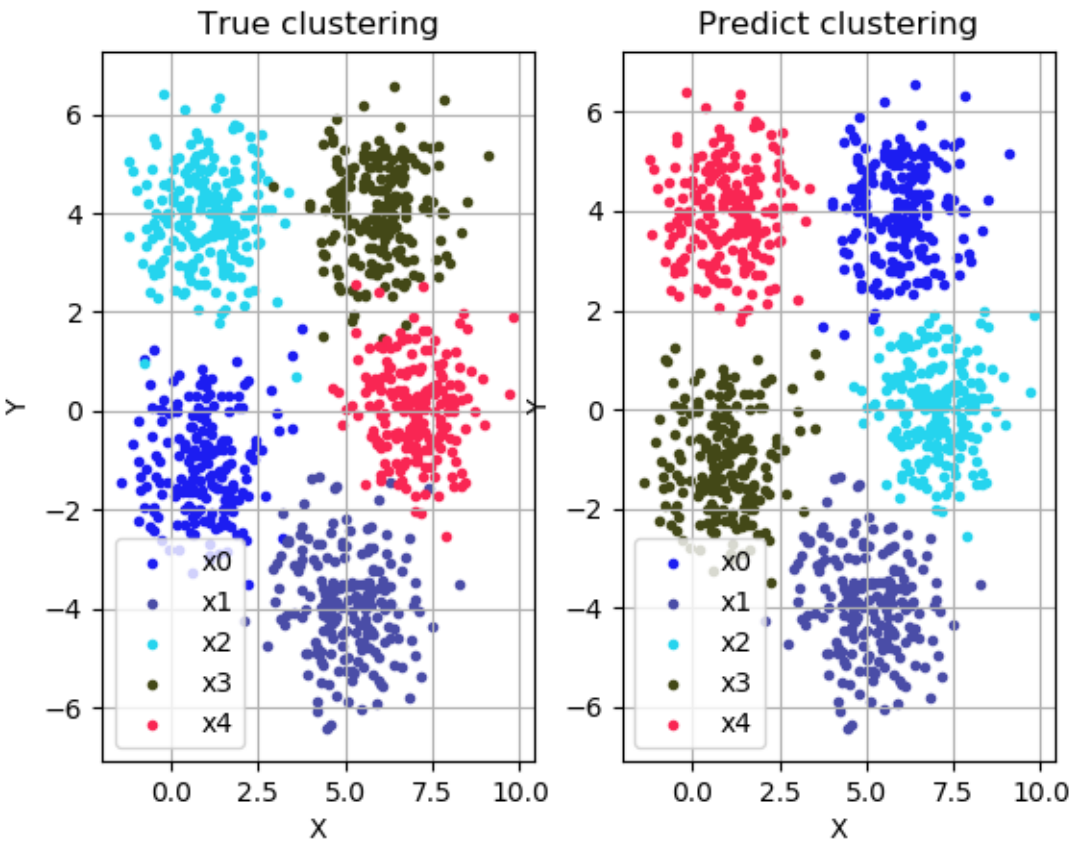

图 3: K-Means result

(c) 聚类评价，计算 ACC 与 NMI 代码与结果，图 4，图 5。

```
def ACC(self):
    y_true = self.y_data.astype(np.int64)
    y_pred = self.y_pred.astype(np.int64)
    assert y_pred.size == y_true.size
    D = max(y_pred.max(), y_true.max()) + 1
    w = np.zeros((D, D), dtype=np.int64)
    for i in range(y_pred.size):
        w[y_pred[i], y_true[i]] += 1
    row_ind, col_ind=linear_sum_assignment(w.max() - w)
    self.acc = sum(w[row_ind, col_ind]) / self.data_num
    print('ACC:',format(self.acc))

def NMI(self):
    self.nmi = metrics.normalized_mutual_info_score(self.y_data.reshape((-1)),
    self.y_pred.reshape((-1)))
    print('NMI:',format(self.nmi))
```

图 4: ACC NMI code

```
starting K-Means clustering version 1.
----------------
init center:

[[ 5.9653277   0.13503915]
 [ 4.386073   -3.6930618 ]
 [ 6.743887    0.17071302]
 [ 1.5714124  -2.3001785 ]
 [ 1.7804744   2.7299232 ]]
<NDArray 5x2 @gpu(0)>
pred center:

[[ 5.989775    4.031954  ]
 [ 5.0627375  -3.9580462 ]
 [ 7.016518   -0.03145027]
 [ 0.90337056 -1.0712457 ]
 [ 0.9712837   3.9588006 ]]
<NDArray 5x2 @gpu(0)>
iteration: 6
----------------
ACC: 0.986
NMI: 0.9510837695525012
****************
```

图 5: ACC NMI result

2.2 第二题

数据同第一题，现在假定 x_1 中的 200 个点属于第一类，x_2 中的 200 个点属于第二类，x_3 中的 200 个点属于第三类，x_4 中的 200 个点属于第四类，x_5 中的 200 个点属于第五类。

请编写一个程序，采用三层前向神经网络来实现分类。

要求：

(1) 每类随机采用 60% 的数据做训练样本，采用剩余的 40% 样本作测试。

(2) 当结构固定时，画出测试精度随学习率 η 变化的曲线。

(3) 画出测试精度随隐含层结点个数变化的曲线。

实验报告：

(a) 基于 maxnet 框架实现三层前向神经网络，隐含层的激活函数使用 sigmoid 函数，损失函数使用 softmax 交叉熵损失函数，优化算法为小批量随机梯度下降算法，具体代码图 6；

```python
class nn_3_layer(model):
    def __init__(self, x_data, y_data, labels_num, train_percent, ctx):
        super().__init__(x_data, y_data, labels_num, ctx)
        self.loss_func=gloss.SoftmaxCrossEntropyLoss()

        self.x_train,self.y_train,self.x_test,self.y_test=_final_dataset(self.x_data,self.y_data,train_percent)

    def net(self,l1_num,l2_num):
        net=nn.Sequential()
        net.add(nn.Dense(l1_num,activation='sigmoid'),nn.Dense(l2_num,activation='sigmoid'),nn.Dense(5))
        net.collect_params().initialize(force_reinit=True, ctx=self.ctx)
        net.initialize(force_reinit=True,ctx=self.ctx,init=init.Xavier())
        return net
```

图 6: forward neural network

(b) 设定第一个隐含层 4 个结点，第二个隐含层 3 个结点的网络结构，测试精度随学习率变化结果，图 7，图 8；

```
start training hyperparameters.
training on gpu(0)
learning rate: 0.150
train acc: 0.875, test acc: 0.973, time: 13.00
learning rate: 0.300
train acc: 0.795, test acc: 0.970, time: 12.88
learning rate: 0.450
train acc: 0.812, test acc: 0.973, time: 12.75
learning rate: 0.600
train acc: 0.847, test acc: 0.968, time: 12.75
learning rate: 0.750
train acc: 0.939, test acc: 0.968, time: 12.94
learning rate: 0.900
train acc: 0.922, test acc: 0.968, time: 12.85
learning rate: 1.050
train acc: 0.924, test acc: 0.960, time: 12.81
learning rate: 1.200
train acc: 0.823, test acc: 0.973, time: 12.75
learning rate: 1.350
train acc: 0.899, test acc: 0.975, time: 12.86
learning rate: 1.500
train acc: 0.948, test acc: 0.960, time: 12.81
train lr done
the best learning rate: 1.3499999999999999
```

图 7: ACC leraning-rate

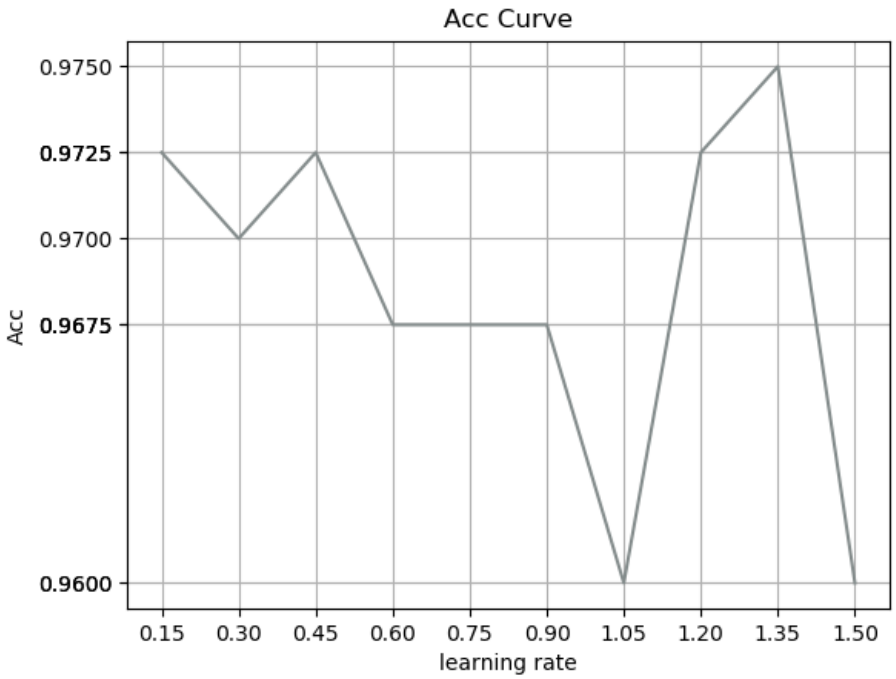

图 8: ACC leraning-rate

(c) 根据隐含层结点变化的精度曲线，先确定第二层为 3 个结点，测试第一层不同的结点数；然后选出第一层最优结点数，再测试第二层不同的结点数，图 9，图 10。

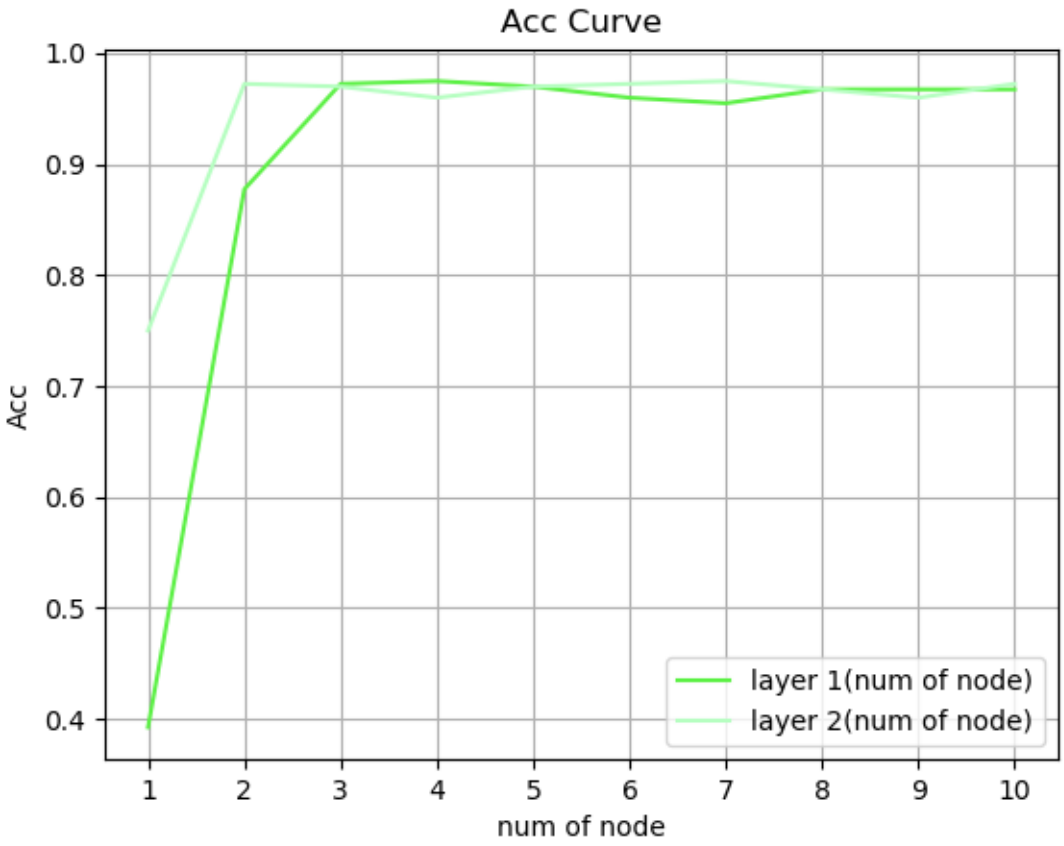

图 9: number of node in each layer

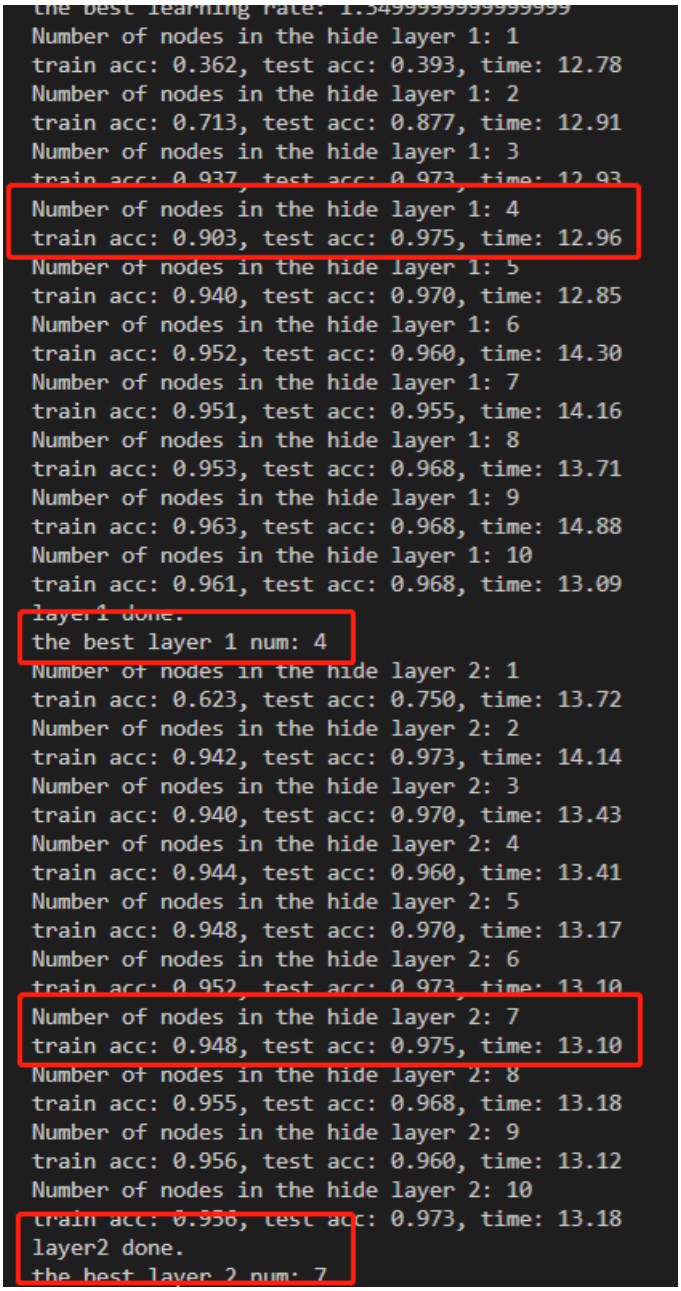

图 10: number of node in each layer

2.3 第三题

本次编程主要涉及到流形学习方法的实现。首先生成一个瑞士蛋卷数据集如图 11，它包含 1000 个三维样本点。

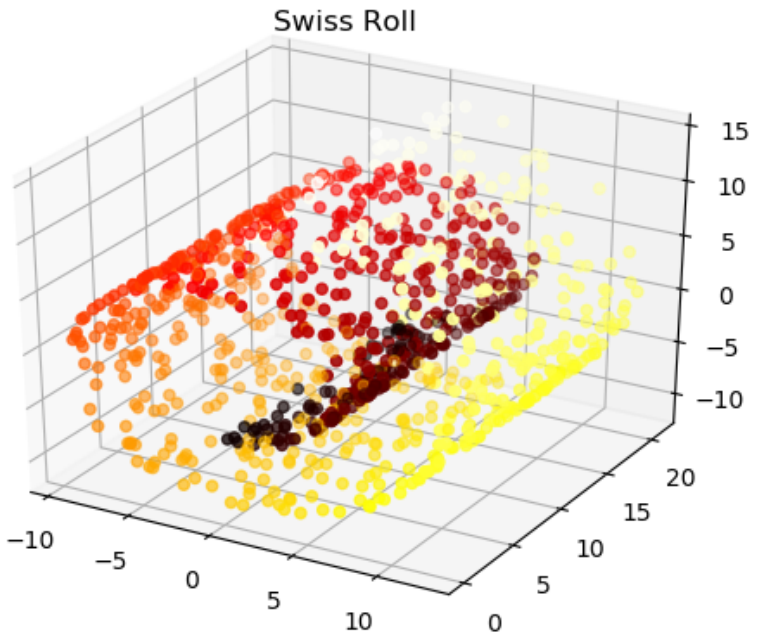

图 11: Swiss Roll

请编程实现 lsomap、LLE 和 LE 算法，并将上述数据降低至二维空间，其中，数据图采用 K-近邻方式进行构造。请显示出 K 取不同值时上述三种方法所对应的降维结果，并给出总结性评述。

实验报告：

(a) 首先生成瑞士蛋卷数据，代码见图 12；

```
def v2_2(self):
    N=1000
    noise=0.001*np.random.randn(N)
    tt=(3*np.pi/2)*(1+2*np.random.rand(N))
    height=21*np.random.rand(N)
    X=np.array([(tt+noise)*np.cos(tt),height,(tt+noise)*np.sin(tt)])
    self.x_data=X
    self.y_data=None
    self.corlor_list=np.squeeze(tt)
    if self.result_path is not None :
        #TODO
        ax1=plt.axes(projection='3d')
        ax1.scatter3D(X[0,:], X[1,:], X[2,:],c=np.squeeze(tt), cmap=plt.cm.hot)
        plt.title('Swiss Roll')
        plt.savefig(self.result_path+'\\dataset-2.2.png')
        plt.close()
```

图 12: Swiss Roll code

(b) 调用 sklearn 包分别实现 LE、LLE 和 Isomap 算法，代码见图 13；

```
plt.title('Laplacian Eigenmapping')
for i in range(1,5):
    trans_data_LE=manifold.SpectralEmbedding(n_components=2,n_neighbors=i*5).fit_transform(dataset.x_data.T)

    plt.subplot(2,2,i)
    plt.scatter(trans_data_LE[:, 0], trans_data_LE[:, 1], c=dataset.corlor_list,marker='o',cmap=plt.cm.hot)
plt.savefig(dataset.result_path+'\\le_reult.png')
plt.close()

plt.title('Locally linear embedding')
for i in range(1,5):
    trans_data_LLE = manifold.LocallyLinearEmbedding(n_components = 2,n_neighbors =i*5,method='standard').fit_transform(dataset.x_data.T)

    plt.subplot(2,2,i)
    plt.scatter(trans_data_LLE[:, 0], trans_data_LLE[:, 1], c=dataset.corlor_list,marker='o',cmap=plt.cm.hot)
plt.savefig(dataset.result_path+'\\lle_result.png')
plt.close()

plt.title('Isometric feature mapping')
for i in range(1,5):
    trans_data_ISOMAP=manifold.Isomap(n_components=2,n_neighbors=i*5).fit_transform(dataset.x_data.T)

    plt.subplot(2,2,i)
    plt.scatter(trans_data_ISOMAP[:, 0], trans_data_ISOMAP[:, 1], c=dataset.corlor_list,marker='o',cmap=plt.cm.hot)
plt.savefig(dataset.result_path+'\\isomap_result.png')
plt.close()
```

图 13: LE LLE Isomap

(c) LE 算法选取不同的近邻值结果，图 14。左上 $k = 5$，右上 $k = 10$，左下 $k = 15$，右下 $k = 20$；

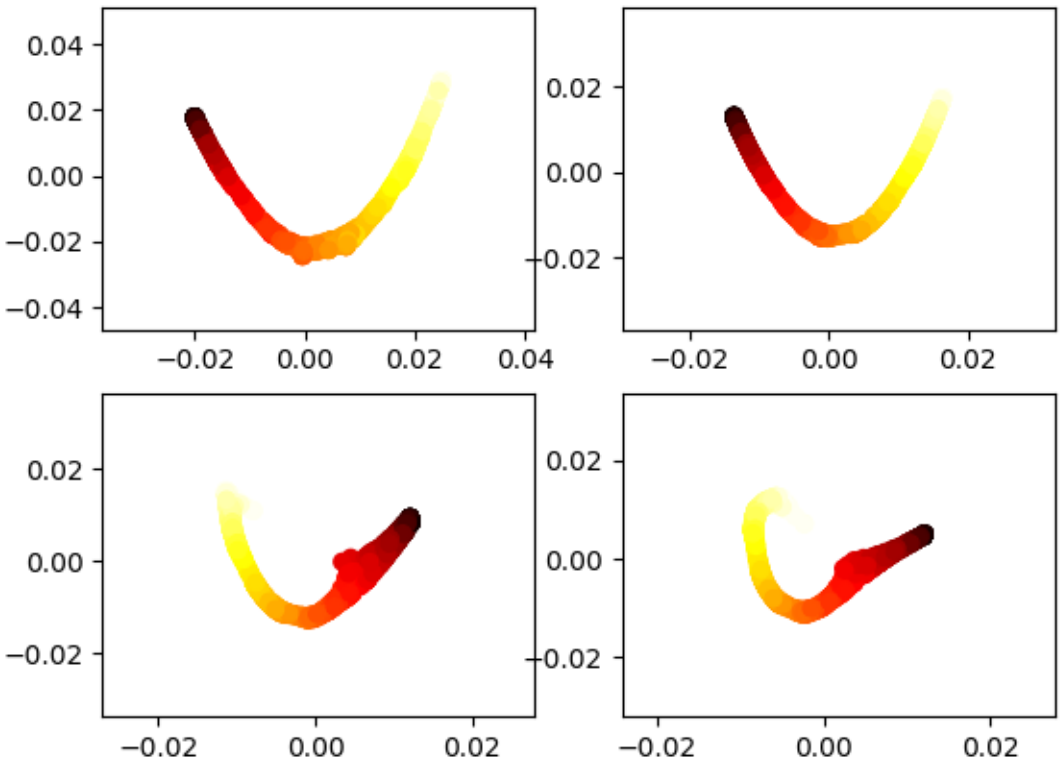

图 14: Laplacian Eigenmapping Result

(d) LLE 算法选取不同的近邻值结果，图 15。左上 $k = 5$，右上 $k = 10$，左下 $k = 15$，右下 $k = 20$；

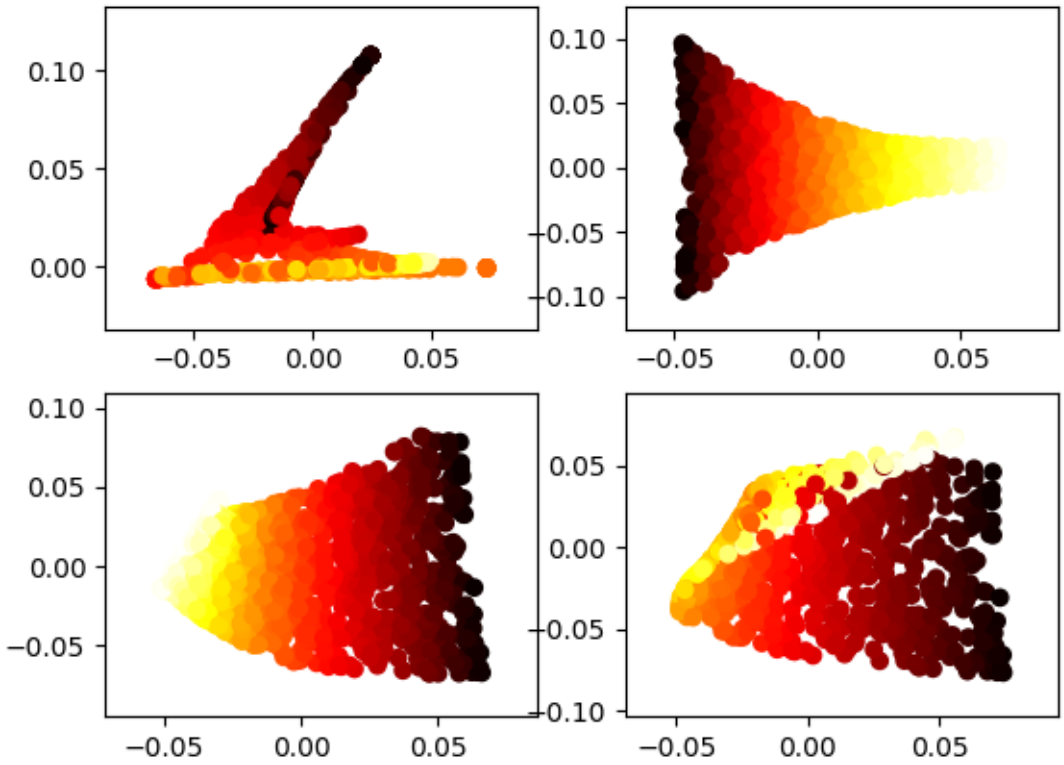

图 15: Locally Linear Embedding Result

(e) Isomap 算法选取不同的近邻值结果，图 16。左上 $k = 5$，右上 $k = 10$，左下 $k = 15$，右下 $k = 20$；

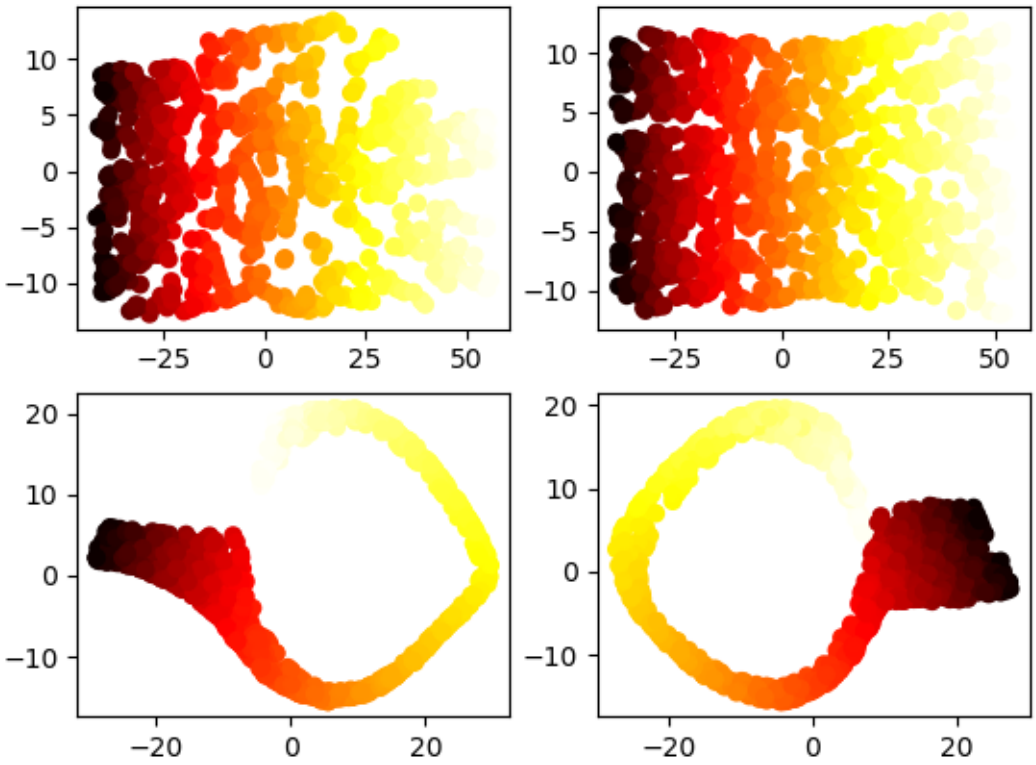

图 16: Isometric Feature Mapping Result

(f) 总结上述实验结果发现，对于瑞士卷这样的高维数据，只要选取得当的 k 值，Isomap 算法实现效果最好，而 LE 算法效果最差。