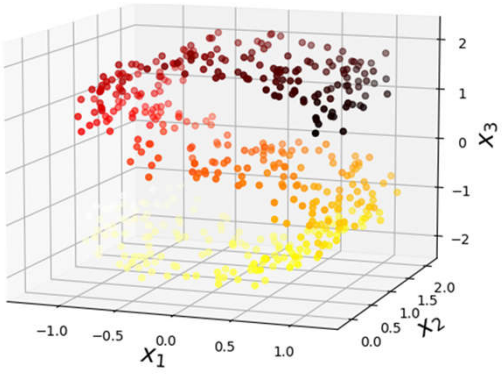
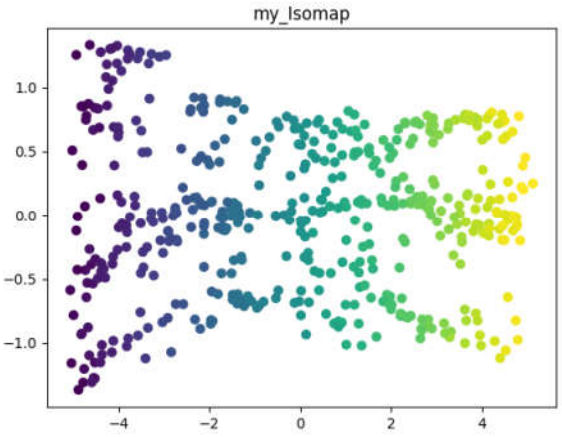
Python编程与人工智能实践

算法篇：数据降维-ISOMAP（等距特征映射）





于泓
鲁东大学
信息与电气工程学院

2021.9.29

ISOMAP 等距特征映射

ISOMAP是 MDS算法在流形结构上的一种应用。

MDS算法

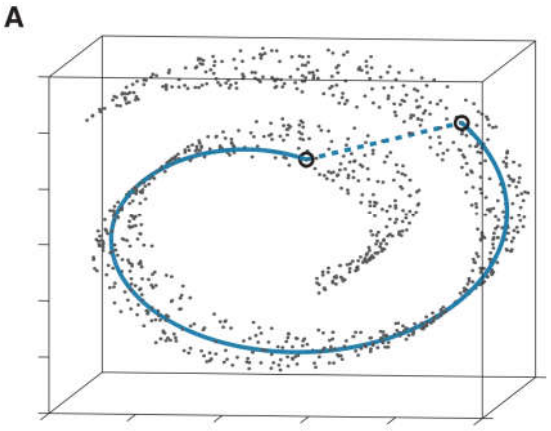
$$\min \sum \left(\left\| z_i - z_j \right\| - d_{ij} \right)^2$$

低维点之间的欧式距离

高维点间的距离

高维点的距离通常也是用**欧式距离**来描述。

但在一些非线性的**流形（Manifold）**上，欧式距离无法真正反应两点间的实际距离

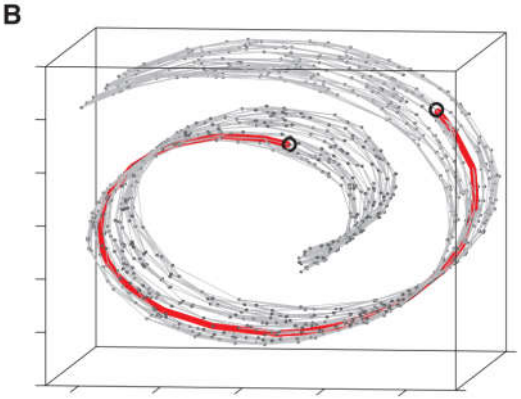


蓝色曲线表示“测地距离”（真实距离）

蓝色虚线表示欧式距离

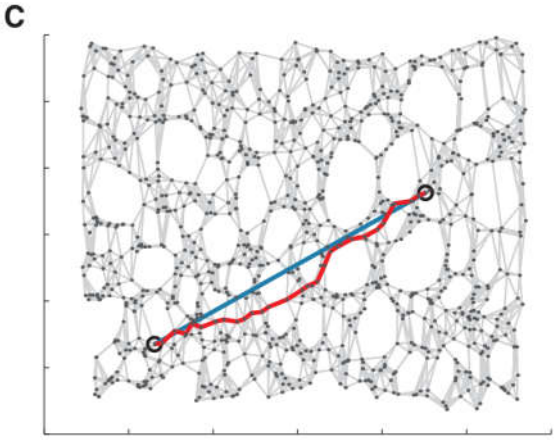
真实的数据分布被**嵌入**到一个“瑞士卷”**流形**上

蓝色曲线的长度才是两点间真实的距离



这个距离并不好求

采用路径近似的方法代替测地线距离



流形展开后用**红色折线**

近似代替

蓝色直线

ISOMAP具体步骤：

步骤1：构建邻接图G （**获取距离矩阵**）

对高维空间中的点，两两计算距离

对每个点，只保留距离其最近的k个点
其他点，认为距离无穷大

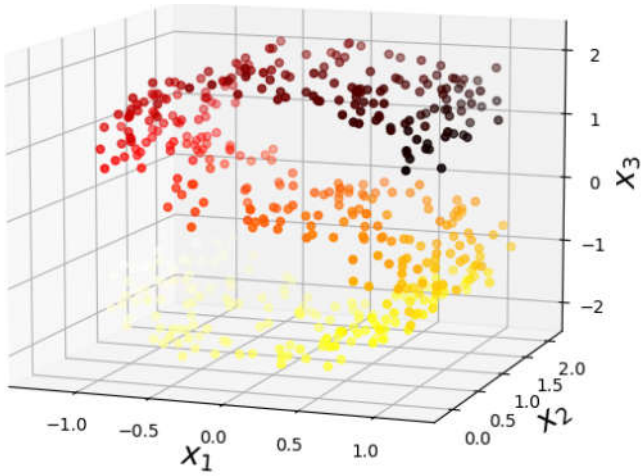
步骤2：计算所有点对之间的最短路径
（**重新填充距离矩阵**）

即对于所有点 i->j

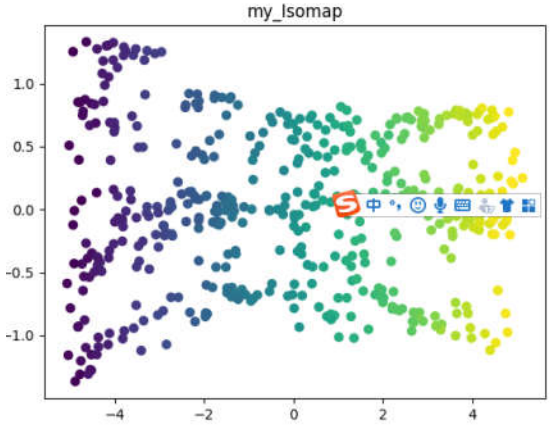
查看是否能找到一条更短路径i->k->j进行替代

即：若 $d_{ij} > d_{ik}+d_{kj}$ 则 $d_{ij} = d_{ik}+d_{kj}$

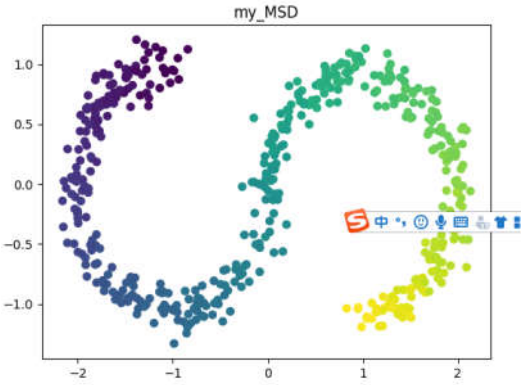
步骤3：利用重构的距离矩阵
结合 MDS算法，进行降维



数据点间的
相互位置关系
得以保留



ISOMAP

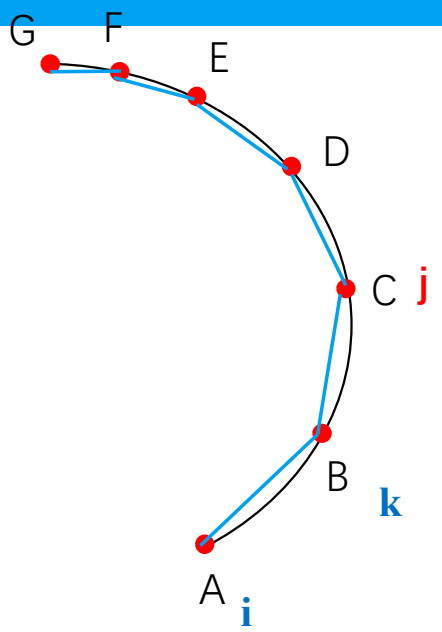


MDS

步骤1，2的实现方法：

Floyd-Warshall算法
(弗洛伊德算法)

```
def floyd(D,n_neighbors=15):
    Max = np.max(D)*1000
    n1,n2 = D.shape
    k = n_neighbors
    D1 = np.ones((n1,n1))*Max
    D_arg = np.argsort(D,axis=1)
    for i in range(n1):
        D1[i,D_arg[i,0:k+1]] = D[i,D_arg[i,0:k+1]]
    for k in range(n1):
        for i in range(n1):
            for j in range(n1):
                if D1[i,k]+D1[k,j]<D1[i,j]:
                    D1[i,j] = D1[i,k]+D1[k,j]
    return D1
```



假设每个点取1个最近点

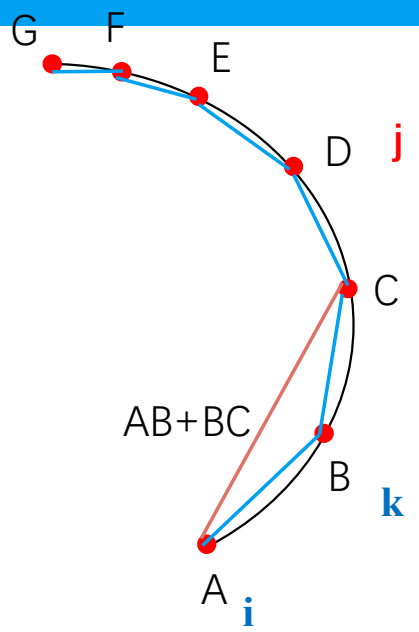
设 A = i B =k **j移动**

j=C AC = ∞ ; AC>AB+BC ; AC =AB+BC

步骤1，2的实现方法：

Floyd-Warshall算法
(弗洛伊德算法)

```
def floyd(D,n_neighbors=15):
    Max = np.max(D)*1000
    n1,n2 = D.shape
    k = n_neighbors
    D1 = np.ones((n1,n1))*Max
    D_arg = np.argsort(D,axis=1)
    for i in range(n1):
        D1[i,D_arg[i,0:k+1]] = D[i,D_arg[i,0:k+1]]
    for k in range(n1):
        for i in range(n1):
            for j in range(n1):
                if D1[i,k]+D1[k,j]<D1[i,j]:
                    D1[i,j] = D1[i,k]+D1[k,j]
    return D1
```



假设每个点取1个最近点

设 A = i B =k **j移动**

j=C AC = ∞ ; AC>AB+BC ; AC =AB+BC

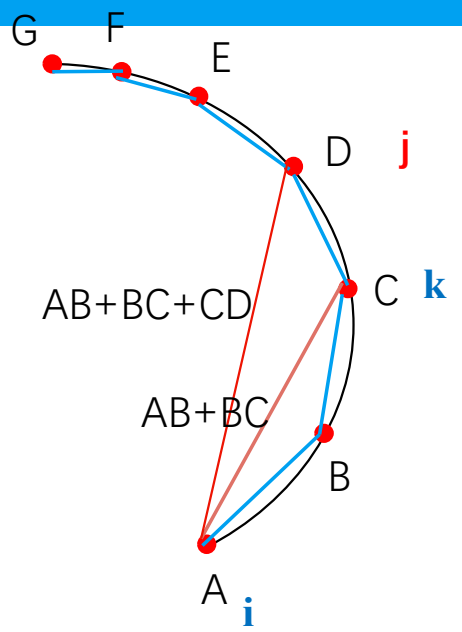
j=D AD = ∞ ; AD<AB+BD ; AD 不变

j=E AE = ∞ ; AE<AB+BE ; AE 不变

步骤1，2的实现方法：

Floyd-Warshall算法
(弗洛伊德算法)

```
def floyd(D,n_neighbors=15):
    Max = np.max(D)*1000
    n1,n2 = D.shape
    k = n_neighbors
    D1 = np.ones((n1,n1))*Max
    D_arg = np.argsort(D,axis=1)
    for i in range(n1):
        D1[i,D_arg[i,0:k+1]] = D[i,D_arg[i,0:k+1]]
    for k in range(n1):
        for i in range(n1):
            for j in range(n1):
                if D1[i,k]+D1[k,j]<D1[i,j]:
                    D1[i,j] = D1[i,k]+D1[k,j]
    return D1
```



假设每个点取1个最近点

设 A = i C = k **j移动**

j=B AB<AC+CB AB不变

j=C AC=AC+CC; AC 不变

j=D AD= ∞ AD>AC+CD ; AD = AC+CD = AB+BC+CD

```python
from sklearn.manifold import Isomap
from tqdm import tqdm

# x 维度 [N,D]
def cal_pairwise_dist(x):

    N,D = np.shape(x)

    dist = np.zeros([N,N])

    for i in range(N):
        for j in range(N):
            # dist[i,j] = np.dot((x[i]-x[j]),(x[i]-x[j]).T)
            dist[i,j] = np.sqrt(np.dot((x[i]-x[j]),(x[i]-x[j]).T))
            # dist[i,j] = np.sum(np.abs(x[i]-x[j]))

    #返回任意两个点之间距离
    return dist
```

```python
# dist N*N 距离矩阵样本点两两之间的距离
# n_dims 降维
# 返回 降维后的数据
def my_mds(dist, n_dims):
    n,n = np.shape(dist)
    dist[dist < 0 ] = 0

    dist = dist**2

    T1 = np.ones((n,n))*np.sum(dist)/n**2
    T2 = np.sum(dist, axis = 1, keepdims=True)/n
    T3 = np.sum(dist, axis = 0, keepdims=True)/n

    B = -(T1 - T2 - T3 + dist)/2

    eig_val, eig_vector = np.linalg.eig(B)
    index_ = np.argsort(-eig_val)[:n_dims]
    picked_eig_val = eig_val[index_].real
    picked_eig_vector = eig_vector[:, index_]

    return picked_eig_vector*picked_eig_val**(0.5)
```

```python
def my_Isomap(D,n=2,n_neighbors=30):

    D_floyd=floyd(D, n_neighbors)
    data_n = my_mds(D_floyd, n_dims=n)
    return data_n
```

```python
def scatter_3d(X, y):
    fig = plt.figure()
    ax = plt.axes(projection='3d')
    ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y, cmap=plt.cm.hot)
    ax.view_init(10, -70)
    ax.set_xlabel("$x_1$", fontsize=18)
    ax.set_ylabel("$x_2$", fontsize=18)
    ax.set_zlabel("$x_3$", fontsize=18)
    plt.show(block=False)
```

```
if __name__ == "__main__":

    X, Y = make_s_curve(n_samples = 500,
                        noise = 0.1,
                        random_state = 42)
    scatter_3d(X,Y)

    # 计算距离
    dist = cal_pairwise_dist(X)


    # MDS 降维
    data_MDS = my_mds(dist, 2)

    plt.figure()
    plt.title("my_MSD")
    plt.scatter(data_MDS[:, 0], data_MDS[:, 1], c = Y)
    plt.show(block=False)


    # ISOMAP 降维
    data_ISOMAP = my_Isomap(dist, 2, 10)

    plt.figure()
    plt.title("my_Isomap")
    plt.scatter(data_ISOMAP[:, 0], data_ISOMAP[:, 1], c = Y)
    plt.show(block=False)

    #利用sklearn 计算 ISOMAP
    data_ISOMAP2 = Isomap(n_neighbors = 10, n_components = 2).fit_transform(X)

    plt.figure()
    plt.title("sk_Isomap")
    plt.scatter(data_ISOMAP2[:, 0], data_ISOMAP2[:, 1], c = Y)
    plt.show(block=False)

    plt.show()
```

实验结果