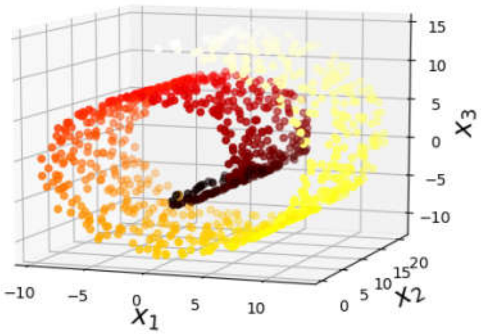
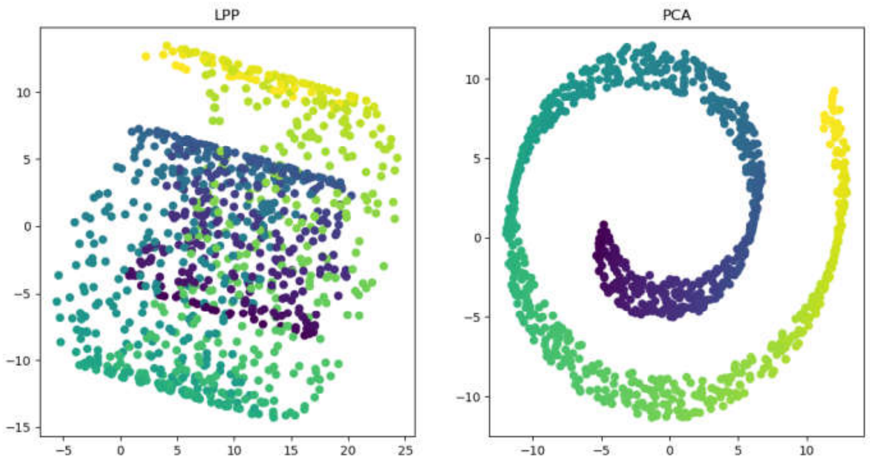
Python编程与人工智能实践



算法篇：数据降维-LPP



于泓
鲁东大学
信息与电气工程学院

2022.9.18

LPP(Locality Preserving Projections) 局部保留投影算法

LPP 可以被看做是PCA（Principal component analysis）的替代，同时与LE（Laplacian eigenmaps）及 LLE（Locally Linear Embedding）有着非常相近的性质。

计算流程与PCA类似，PCA只考虑数据本身的分布，LPP还考虑了样本点间的相对位置关系

文案参考：https://zhuanlan.zhihu.com/p/340121889
代码参考：https://github.com/heucoder/dimensionality_reduction_alo_codes/blob/master/codes/LPP/LPP.py
论文链接：https://papers.nips.cc/paper/2003/file/d69116f8b0140cdeb1f99a4d5096ffe4-Paper.pdf

算法原理

- 将一组高维样本**X** 通过转换矩阵**A**,映射到低维样本**Y**

$$\mathbf{Y}=\mathbf{A}^{\mathrm{T}}\mathbf{X} \qquad \mathbf{Y}\in(d,N),\ \ \mathbf{A}\in(D,d),\mathbf{X}\in(D,N)$$

关系矩阵 **W**,用来描述高维空间样本点之间的关系 $\mathbf{W}\in(N,N)$

$$W_{ij}=e^{-\frac{\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2}{t}}$$

t：常量

当高维点i与点j比较接近时　　*距离越近权重越高*

$$W_{ij}=0$$

当高维点i与点j距离较远近时　　**利用KNN找到接近的点**

$$W_{ii}=0$$

假设d=1　　$\mathbf{y}=\mathbf{a}^{\mathrm{T}}\mathbf{X}$

优化函数：　**最小化** $\sum_{i,j}(y_i-y_j)^2W_{ij}$

高维空间接近：W_{ij} 大，低维空间：$(y_i\text{-}y_j)^2$ 小

高维空间远离：W_{ij} 小，低维空间：$(y_i\text{-}y_j)^2$ 大

$$\frac{1}{2}\sum_{i,j}(y_i-y_j)^2W_{ij}=\sum_i \mathbf{a}^T\mathbf{x}_iD_{ii}\mathbf{x}_i^{\mathrm{T}}\mathbf{a}-\sum_{ij}\mathbf{a}^T\mathbf{x}_iW_{ij}\mathbf{x}_j^{\mathrm{T}}\mathbf{a}$$

$$=\mathbf{a}^T\mathbf{X}(\mathbf{D}-\mathbf{W})\mathbf{X}^T\mathbf{a}=\mathbf{a}^T\mathbf{XLX}^T\mathbf{a}$$

对角线矩阵　$D_{ii}=\sum_j W_{ij}$

假设N=3

$$\frac{1}{2}\sum_{i,j}(y_i - y_j)^2 W_{ij} = (y_1 - y_2)^2 W_{12} + (y_1 - y_3)^2 W_{13} + (y_2 - y_3)^2 W_{23}$$

$$= y_1^2 W_{12} + y_1^2 W_{13} + y_2^2 W_{12} + y_2^2 W_{23} + y_3^2 W_{13} + y_3^2 W_{23}$$

$$-\left(2y_1y_2W_{12} + 2y_1y_3W_{13} + 2y_2y_3W_{23}\right)$$

$$\sum_i \mathrm{y}_i D_{ii} y_i^{\,T} = \mathbf{yDy}^{\mathrm{T}}$$

$$\mathbf{yWy}^{\mathrm{T}}$$

最小化： $\mathbf{a}^T\mathbf{XLX}^T\mathbf{a}$

引入约束： $\mathbf{yDy}^{\mathrm{T}}=1$ 即 $\mathbf{a}^T\mathbf{XDX}^T\mathbf{a}=1$

引入拉格朗日算式 $\min_{\mathbf{a}} F(\mathbf{a})=\mathbf{a}^T\mathbf{XLX}^T\mathbf{a}+\lambda(1-\mathbf{a}^T\mathbf{XDX}^T\mathbf{a})$

求导可得：

$$2\mathbf{XLX}^T\mathbf{a}\text{-}2\lambda\mathbf{XDX}^T\mathbf{a}=0$$

$$\mathbf{XLX}^T\mathbf{a}=\lambda\mathbf{XDX}^T\mathbf{a}$$

$$\left[\left(\mathbf{XDX}^T\right)^{-1}\mathbf{XLX}^T\right]\mathbf{a}=\lambda\mathbf{a}^T$$

当d>1时
A 为d个最小特征值
对应的特征向量

最小特征值对应的特征向量

比较PCA与 LPP

PCA： $$C=\frac{X^T X}{m-1}$$ 取K个**最大特征值**的特征向量

LPP： $$C=\left(\mathbf{XDX}^T\right)^{-1}\mathbf{XLX}^T=\frac{\mathbf{XLX}^T}{\mathbf{XDX}^T}$$ 取d个**最小特征值**的特征向量

代码实现:

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

    N,D = np.shape(X)

    tile_xi = np.tile(np.expand_dims(X,1),[1,N,1])
    tile_xj = np.tile(np.expand_dims(X,axis=0),[N,1,1])

    dist = np.sum((tile_xi-tile_xj)**2,axis=-1)

    #返回任意两个点之间距离
    return dist
```

```python
def cal_rbf_dist(data, n_neighbors = 10, t = 1):

    dist = cal_pairwise_dist(data)
    dist[dist < 0] = 0
    N = dist.shape[0]
    rbf_dist = rbf(dist, t)

    W = np.zeros([N, N])
    for i in range(N):
        index_ = np.argsort(dist[i])[1:1 + n_neighbors]
        W[i, index_] = rbf_dist[i, index_]
        W[index_, i] = rbf_dist[index_, i]

    return W
```

计算K近邻
去掉自己

```python
def rbf(dist, t = 1.0):
    '''
    rbf kernel function
    '''
    return np.exp(-(dist/t))
```

$$W_{ij} = e^{-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{t}}$$

```
def lpp(X,n_dims = 2,n_neighbors = 30, t = 1.0):

    N = X.shape[0]
    W = cal_rbf_dist(X, n_neighbors, t)
    D = np.zeros_like(W)

    for i in range(N):
        D[i,i] = np.sum(W[i])

    L = D - W
    XDXT = np.dot(np.dot(X.T, D), X)
    XLXT = np.dot(np.dot(X.T, L), X)

    eig_val, eig_vec = np.linalg.eig(np.dot(np.linalg.pinv(XDXT), XLXT))

    sort_index_ = np.argsort(np.abs(eig_val))
    eig_val = eig_val[sort_index_]
    print("eig_val[:10]", eig_val[:10])

    j = 0
    while eig_val[j] < 1e-6:
        j+=1

    print("j: ", j)

    sort_index_ = sort_index_[j:j+n_dims]

    eig_val_picked = eig_val[j:j+n_dims]
    print(eig_val_picked)
    A = eig_vec[:, sort_index_]

    Y = np.dot(X, A)

    return Y
```

$$\left(\mathbf{XDX}^T\right)^{-1}\mathbf{XLX}^T$$

特征值从小到大

舍弃过小的特征值

得到变换矩阵A

测试:

```python
if __name__ == '__main__':
    #1 测试瑞士卷数据
    X, Y = make_swiss_roll(n_samples=1000)
    scatter_3d(X, Y)
    n_neighbors = 5

    #2 测试 load_digits 数据
    # X = load_digits().data
    # Y = load_digits().target
    # n_neighbors = 5


    #3 测试  load_iris 数据
    # X = load_iris().data
    # Y = load_iris().target
    # n_neighbors = 5

    dist = cal_pairwise_dist(X)
    max_dist = np.max(dist)

    data_2d_LPP = lpp(X, n_neighbors = n_neighbors, t = 0.01*max_dist)
    data_2d_PCA = PCA(n_components=2).fit_transform(X)


    plt.figure(figsize=(12,6))
    plt.subplot(121)
    plt.title("LPP")
    plt.scatter(data_2d_LPP[:, 0], data_2d_LPP[:, 1], c = Y)

    plt.subplot(122)
    plt.title("PCA")
    plt.scatter(data_2d_PCA[:, 0], data_2d_PCA[:, 1], c = Y)
    plt.show()
```