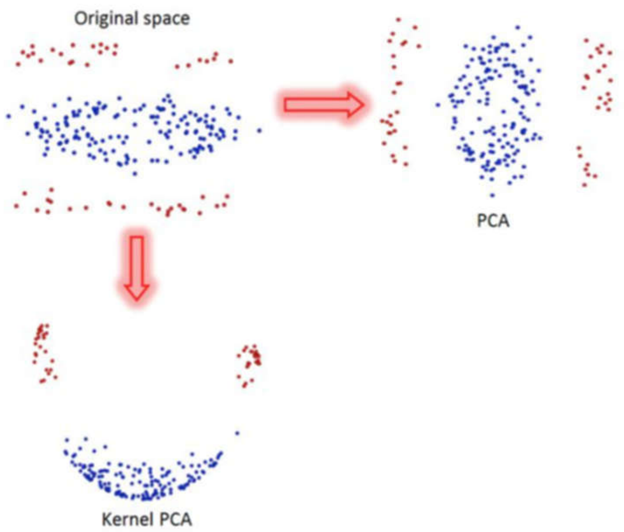
Python编程与人工智能实践

算法篇：数据降维-Kernel-PCA



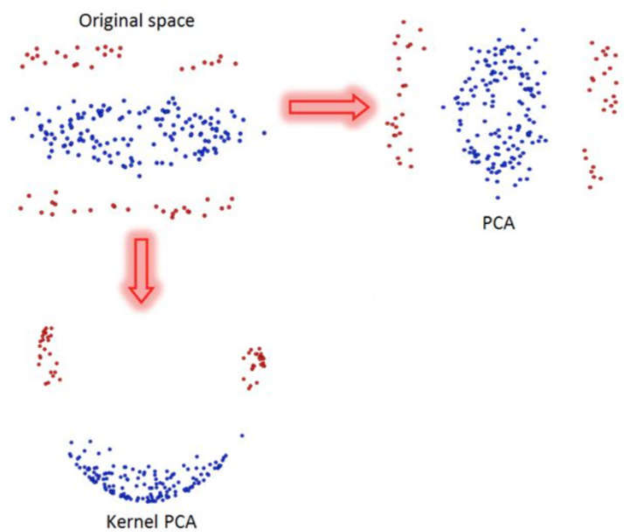
于泓
鲁东大学
信息与电气工程学院

2021.9.27

Kernel-PCA（核PCA）

- PCA降维是一种线性变换，在高维空间的样本点如果线性不可分的话，到了低维空间依旧线性不可分。为了能够让样本在低维空间线性可分引入了KPCA(Kernel PCA)

KPCA基本思想：对源空间的点进行，**非线性高维映射**，在映射后的后的空间内再进行**PCA**降维



非线性高维映射： **（方便表达：后继所有的矢量都是行向量）**

例：

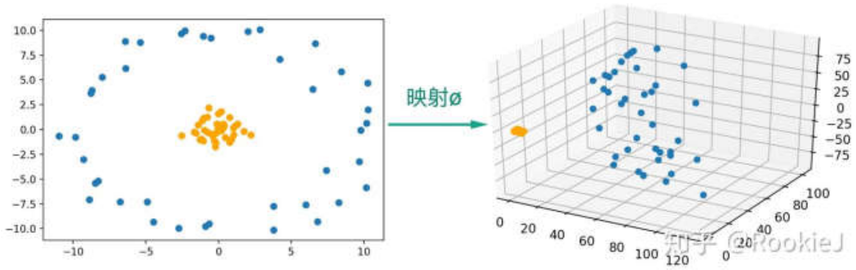
$$\mathrm{x}=[x_1, x_2]$$ **（低维）**

$$\mathrm{X}=\phi(\mathrm{x})=[x_1^2, x_2^2, \sqrt{2}x_1x_2]$$ **（高维）**

通过映射函数 $\phi(\mathrm{x})$

将一个**2维**的样本点x

映射为**3 维**的样本点X



在实际的应用中$\phi(\mathrm{x})$往往未知，已知的通常为核函数

$$\mathrm{k}(\mathrm{x}_1,\mathrm{x}_2)=\mathrm{X}_1\mathrm{X}_2^{\mathrm{T}}=\phi(\mathrm{x}_1)\phi(\mathrm{x}_2)^{\mathrm{T}}$$

即两个高维空间点之间的相关性

常见的核函数:

多项式核

$$\mathrm{k}(\mathrm{x}_1,\mathrm{x}_2)=(a\mathrm{x}_1\mathrm{x}_2^T+\mathrm{c})^d$$

高斯核/RBF(径向基)

$$\mathrm{k}(\mathrm{x}_1,\mathrm{x}_2)=\exp\left(-\gamma\left\|\mathrm{x}_1-\mathrm{x}_2\right\|^2\right)$$

Sigmoid核

$$\mathrm{k}(\mathrm{x}_1,\mathrm{x}_2)=\tanh\left(a\mathrm{x}_1\mathrm{x}_2^{\mathrm{T}}+\mathrm{r}\right)$$

这些核函数都相当于 $\phi(\mathrm{x})$ 映射的特征维度为无穷多维

一般PCA的步骤:

（1）对输入的数据 $\mathbf{x}$ （维度 $[N,D]$,N个样本，每个D维度）

进行中心化处理 $\mathbf{x}=\mathbf{x}-\bar{\mathbf{x}}$

（2）计算相关矩阵 $\mathbf{C}=\frac{1}{\mathrm{N}}\mathbf{x}^{\mathrm{T}}\mathbf{x}$ $\quad \mathbf{C}\in[D,D]$

（特征点各个维度之间的相关性）

（3）对 $\mathbf{C}$ 进行特征值分解，获取d个较大的特征值，所对应的特征向量 $\mathbf{v}$ $\quad \mathbf{v}\in[d,D]$

（4）降维: $\mathbf{x}'=\mathbf{x}\,\mathbf{v}^{\mathrm{T}}$ $\quad \mathbf{x}'\in[N,\ d]$

在KPCA中

$$\mathbf{C}=\frac{1}{\mathrm{N}}\mathbf{X}^{\mathrm{T}}\mathbf{X} \qquad \mathbf{C}\in[D,D]$$

由于映射函数 $\phi(\mathrm{x})$ 通常未知

所以无法求 $\mathbf{C}$ 无法进行特征值分解

已知:

$$\mathbf{K}=\mathbf{X}\mathbf{X}^{T} \qquad \mathbf{K}\in[N,N]$$

各个特征点之间的相关性

设法利用**K**求**C**的特征向量

对 $\mathbf{K}=\mathbf{X}\mathbf{X}^T \quad \mathbf{K}\in[N,N]$ 进行特征值分解

$$\left(\mathbf{X}\mathbf{X}^T\right)\mathrm{u}^{\mathrm{T}}=\lambda\mathrm{u}^{\mathrm{T}} \qquad \mathrm{u}\in[1,N] \quad \mathbf{X}\in[N,D]$$

$$\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)\mathrm{u}^{\mathrm{T}}=\lambda\mathbf{X}^T\mathrm{u}^{\mathrm{T}}$$

$$\left(\mathbf{X}^T\mathbf{X}\right)(\mathrm{u}\mathbf{X})^T=\lambda(\mathrm{u}\mathbf{X})^T$$

对 $N\mathbf{C}=\mathbf{X}^{\mathrm{T}}\mathbf{X} \quad \mathbf{C}\in[D,D]$

进行特征值分解

$$\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)\mathrm{v}^{\mathrm{T}}=\lambda\mathrm{v}^{\mathrm{T}} \qquad \mathrm{v}\in[1,D]$$

得到C的特征向量量就是 $\mathrm{v}=\mathrm{u}\mathbf{X} \quad \mathrm{v}\in[1,D]$

对特征矢量 $\mathrm{v}=\mathrm{u}\mathbf{X}$ $\mathrm{v}\in[1,D]$ $\mathbf{X}\in[N,D]$ $\mathrm{u}\in[1,N]$ 进行归一化

$$\mathrm{v}=\frac{1}{\|\mathrm{u}\mathbf{X}\|}\mathrm{u}\mathbf{X}=\frac{1}{\sqrt{\mathrm{u}\mathbf{X}(\mathrm{u}\mathbf{X})^{\mathrm{T}}}}\mathrm{u}\mathbf{X}=\frac{1}{\sqrt{\mathrm{u}(\mathbf{X}\mathbf{X}^{\mathrm{T}})\mathrm{u}^{\mathrm{T}}}}\mathrm{u}\mathbf{X}$$

$$(\mathbf{X}\mathbf{X}^{T})\mathrm{u}^{\mathrm{T}}=\lambda\mathrm{u}^{\mathrm{T}}$$

$$=\frac{1}{\sqrt{\mathrm{u}\lambda\mathrm{u}^{\mathrm{T}}}}\mathrm{u}\mathbf{X}\quad=\frac{1}{\sqrt{\lambda}}\mathrm{u}\mathbf{X}=\mathrm{u}'\mathbf{X}$$

未知

收集特征值较大的d个 v 组成 $\mathbf{v}$ $\mathbf{v}\in[d,D]$

$$\mathbf{v}=\mathbf{u}'\mathbf{X}\qquad \mathbf{u}'\in[d,N]$$

降维:

$$\mathbf{x}'=\mathbf{X}\,\mathbf{v}^{\mathrm{T}}\qquad \mathbf{x}'\in[N,\ d]$$

$$=\mathbf{X}\,\mathbf{X}^{\mathrm{T}}\mathbf{u}'^{\mathrm{T}}=\mathbf{K}\mathbf{u}'^{\mathrm{T}}$$

$$\mathbf{K}\in[N,N]$$

中心化

$$\tilde{\mathrm{X}}_i = \mathrm{X}_i - \bar{\mathrm{X}}$$

写成矩阵的形式

NxN全1矩阵

$$\tilde{\mathbf{X}} = \mathbf{X} - \frac{1}{N}\mathbf{I}_{N\times N}\mathbf{X} = \mathbf{X} - \mathbf{I}_N\mathbf{X}$$

$\mathrm{X}_i = \phi(\mathrm{x}_i)$ 未知，转而求 $\tilde{\mathbf{K}}$

$$\tilde{\mathbf{K}} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}} = \left(\mathbf{X} - \mathbf{I}_N\mathbf{X}\right)\left(\mathbf{X} - \mathbf{I}_N\mathbf{X}\right)^{\mathrm{T}}$$

$$= \left(\mathbf{X} - \mathbf{I}_N\mathbf{X}\right)\left(\mathbf{X}^{\mathrm{T}} - \mathbf{X}^T\mathbf{I}_N^T\right) = \mathbf{X}\mathbf{X}^{\mathrm{T}} - \mathbf{I}_N\mathbf{X}\mathbf{X}^{\mathrm{T}} - \mathbf{X}\mathbf{X}^T\mathbf{I}_N^T + \mathbf{I}_N\mathbf{X}\mathbf{X}^T\mathbf{I}_N^T$$

$$= \mathbf{K} - \mathbf{I}_N\mathbf{K} - \mathbf{K}\mathbf{I}_N^T + \mathbf{I}_N\mathbf{K}\mathbf{I}_N^T$$

[N,D]:[3,2]

例如：

$$\tilde{\mathbf{X}} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \\ X_{31} & X_{32} \end{bmatrix} - \frac{1}{3}\begin{bmatrix} X_{11}+X_{21}+X_{31} & X_{12}+X_{22}+X_{32} \\ X_{11}+X_{21}+X_{31} & X_{12}+X_{22}+X_{32} \\ X_{11}+X_{21}+X_{31} & X_{12}+X_{22}+X_{32} \end{bmatrix}$$

$$= \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \\ X_{31} & X_{32} \end{bmatrix} - \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \\ X_{31} & X_{32} \end{bmatrix}$$

Kernel-PCA 的一般步骤

（1）给定样本 $\mathbf{x}$ （维度 $[N,D]$,N个样本，每个D维）

（2）利用核函数 计算样本两两之间的关系矩阵

$\mathbf{K}$ $\quad \mathbf{K} \in [N, N]$ 其中 $\mathrm{K}_{ij} = \mathrm{k}(\mathrm{x}_i, \mathrm{x}_j)$

（3）对 $\mathbf{K}$ 进行中心化

$$\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{I}_N \mathbf{K} - \mathbf{K} \mathbf{I}_N^T + \mathbf{I}_N \mathbf{K} \mathbf{I}_N^T$$

（4）对 $\tilde{\mathbf{K}}$ 进行特征值分解

获取较大的d个特征值，λ_i 以及对应的特征向量u_i

并对 u_i 进行正则化

$$\mathrm{u}_i' = \frac{1}{\sqrt{\lambda_i}} \mathrm{u}_i$$

（5）收集d个 u_i' 生成降维矩阵 $\mathbf{u}'$

$$\mathbf{u}' \in [d, N]$$

进行降维

$\mathbf{x}' = \mathbf{K} \mathbf{u}'^{\mathrm{T}} \quad \mathbf{x}' \in [N, d]$

代码实现：

```
import numpy as np
import matplotlib.pyplot as plt

def sigmoid(x1, x2,a = 0.25,r=1):
    x = np.dot(x1, x2)
    return np.tanh(a*x+r)

def linear(x1,x2,a=1,c=0,d=1):
    x = np.dot(x1, x2)
    x = np.power((a*x+c),d)
    return x

def rbf(x1,x2,gamma = 10):
    x = np.dot((x1-x2),(x1-x2))
    x = np.exp(-gamma*x)
    return x
```

多项式核

$$k(x_1,x_2)=(ax_1x_2^T+c)^d$$

高斯核/RBF(径向基)

$$k(x_1,x_2)=\exp\left(-\gamma\left\|x_1-x_2\right\|^2\right)$$

Sigmoid核

$$k(x_1,x_2)=\tanh\left(ax_1x_2^T+r\right)$$

```python
def kpca(data, n_dims=2, kernel = rbf):

    N,D = np.shape(data)
    K = np.zeros([N,N])

    # 利用核函数计算K
    for i in range(N):
        for j in range(N):
            K[i,j]=kernel(data[i],data[j])

    # 对K进行中心化
    one_n = np.ones((N, N)) / N
    K = K - one_n.dot(K) - K.dot(one_n) + one_n.dot(K).dot(one_n)

    #计算特征值和特征向量
    eig_values, eig_vector = np.linalg.eig(K)
    idx = np.argsort(-eig_values)[:n_dims] # 从大到小排序

    # 选取较大的特征值
    eigval = eig_values[idx]
    eigvector = eig_vector[:, idx]   #[N,d]

    # 进行正则
    eigval = eigval**(1/2)
    u = eigvector/eigval.reshape(-1,n_dims) # u [N,d]

    # 进行降维
    data_n = np.dot(K, u)
    return data_n
```

$$\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{I}_N\mathbf{K} - \mathbf{K}\mathbf{I}_N^T + \mathbf{I}_N\mathbf{K}\mathbf{I}_N^T$$

$$\mathbf{u}_i' = \frac{1}{\sqrt{\lambda_i}}\mathbf{u}_i$$

```python
def draw_pic(datas,labs):
    plt.cla()
    unque_labs = np.unique(labs)
    colors = [plt.cm.Spectral(each)
              for each in np.linspace(0, 1,len(unque_labs))]
    p=[]
    legends = []
    for i in range(len(unque_labs)):
        index = np.where(labs==unque_labs[i])
        pi = plt.scatter(datas[index, 0], datas[index, 1], c =[colors[i]] )
        p.append(pi)
        legends.append(unque_labs[i])

    plt.legend(p, legends)
    plt.show()

if __name__ == "__main__":

    # 加载数据
    data = np.loadtxt("iris.data",dtype="str",delimiter=',')
    feas = data[:,:-1]
    feas = np.float32(feas)
    labs = data[:,-1]
    kpca(feas)

    # 进行降维
    data_2d = kpca(feas, n_dims=2,kernel=rbf)


    #绘图
    draw_pic(data_2d,labs)
```