PCA: Primary Component Analysis

November 27, 2015

1 PCA理论推导

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \dots & \dots & \dots & \dots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{pmatrix} \tag{1}$$

$X \in R^{m\times n}$，式(1)的每列是一个样本，每个样本有m个属性，一共有n个样本。注意，这里的每个样本都经过均值化处理。

X表示一个完整数据集。举例来说，一台设备上有m个测点，这些测点定义了设备的运行特征，采集数据n秒，就形成X。

数据通常是含糊的，有噪声的，不明确的。这种含糊和不明确，体现在它的协方差阵的多数元素都是非零值。比如，X的协方差阵就是:

$$C_X = \frac{1}{n-1}XX^T \tag{2}$$

其中，$\frac{1}{n-1}$是一个实数系数，$C_X \in R^{m\times m}$。

需要从数据找到一个不含糊的，低噪声的方向。这个需求，在本质上就是寻找一个矩阵，用它对X做变换，使得变换后的新矩阵的协方差阵大多数元素的值是零，最好的情况是，只有主对角线非零，其他都是零。令P表示这个矩阵，则：

$$Y = PX \tag{3}$$

其中，$P \in R^{m\times m}$,$Y \in R^{m\times n}$。

把式(3)带入式(2)，令

$$A = XX^T \tag{4}$$

则：

$$\begin{aligned} C_Y &= \frac{1}{n-1}YY^T \\ &= \frac{1}{n-1}(PX)(PX)^T \\ &= \frac{1}{n-1}PXX^TP^T \\ &= \frac{1}{n-1}PAP^T \end{aligned} \tag{5}$$

A是一个对称阵，对它进行对角化，可以写成$A = EDE^T$，其中，D是对角阵，$\{A, E, D\} \in R^{m \times m}$。

如果要将C_Y转化成对角阵，观察上式可知，如果令$P = E^T$，根据矩阵对角化性质可知，$P^{-1} = P^T$，I表示单位阵，则式(5)就变成：

$$\begin{aligned} C_Y &= \frac{1}{n-1} PAP^T \\ &= \frac{1}{n-1} P(P^T DP)P^T \\ &= \frac{1}{n-1} (PP^T)D(PP^T) \\ &= \frac{1}{n-1} (PP^{-1})D(PP^{-1}) \\ &= \frac{1}{n-1} IDI \\ &= \frac{1}{n-1} D \end{aligned} \tag{6}$$

2 均值化

PCA理论推导里的X经过均值化处理的。均值化过程如下：

$$Z = \begin{pmatrix} z_{1,1} & z_{1,2} & \dots & z_{1,n} \\ z_{2,1} & z_{2,2} & \dots & z_{2,n} \\ \dots & \dots & \dots & \dots \\ z_{m,1} & z_{m,2} & \dots & z_{m,n} \end{pmatrix} \tag{7}$$

其中，$Z \in R^{m \times n}$，是原始数据。

令:

$$\overline{z_i} = \frac{1}{n} \sum_{j=1}^{n} z_{i,j} \tag{8}$$

那么，均值化就是：

$$\overline{Z} = \begin{pmatrix} z_{1,1} - \overline{z_1} & z_{1,2} - \overline{z_1} & \dots & z_{1,n} - \overline{z_1} \\ z_{2,1} - \overline{z_2} & z_{2,2} - \overline{z_2} & \dots & z_{2,n} - \overline{z_2} \\ \dots & \dots & \dots & \dots \\ z_{m,1} - \overline{z_m} & z_{m,2} - \overline{z_m} & \dots & z_{m,n} - \overline{z_m} \end{pmatrix} \tag{9}$$

3 实现

用Python2.7，matplotlib和numpy实现pca算法。

```python
#!/usr/bin/env python
#! -*- coding:utf-8 -*-

import matplotlib.pyplot as plt
from numpy import *

#create two data set
def create_dataset(n):
    data_r = random.randn(n, 2)
    #squeez y
    for i in range(data_r.shape[0]):
        data_r[i, 1] = 0.1*data_r[i, 1]
    #rotate
    theta = -0.25*3.14
    tran = zeros((2, 2))
    tran[1, 1] = tran[0, 0] = cos(theta)
    tran[0, 1] = -sin(theta)
    tran[1, 0] = sin(theta)
    data = dot(data_r, tran)
    #move
    data[:, 0] += 3
    data[:, 1] += 1
    return data.transpose()

def do_mean(X):
    means =  zeros((X.shape[0],1))
    means[0,0] = mean(X[0,:])
    means[1,0] = mean(X[1,:])
    for i in range(X.shape[0]):
        X[i,:] -= means[i,0]
    return X

def do_pca(X):
    A = dot(X, X.transpose())
    _ , E = linalg.eig(A)
    P = E.transpose()
    return dot(P, X)

def draw_x(X):
    plt.axis([-3, 6, -6, 6])
    plt.plot([z for z in X[0, :]], [z for z in X[1, :]], 'r.')
    plt.show()

def main():
    X = create_dataset(1000)
    draw_x(X)
    do_mean(X)
    new_X = do_pca(X)
    draw_x(new_X)

if __name__ == "__main__":
    main()
```

该实现创建出一组数据，它近似高斯分布，且主方向在45度角方向的直线上。

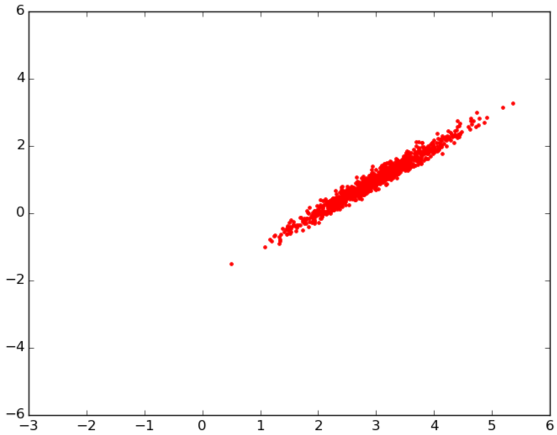

Figure 1: X

经过PCA算法处理后，数据X成为以原点为中心，水平方向是主方向的新数据。究其本质来说，new_X是数据X在以P为基的二维空间的图像。

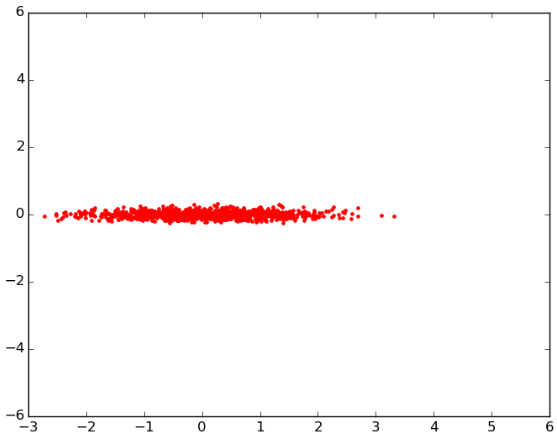

Figure 2: new_X