Python编程与人工智能实践

算法篇：PCA回归与偏最小二乘回归

Partial Least Squares

于泓
鲁东大学
信息与电气工程学院

2022.12.14

线性回归的问题

- 利用线性回归算法，基于最小均方误差（MSE）的准则，计算得到权重 w。 利用w对观测信号X进行线性加权，得到预测值Y。
- 那么： w中的正数表示，该观测信号对Y的预测起正向作用（正比例）
- w中的负数表示，该观测信号对Y的预测起反向作用（反比例）
- 系数越大，作用越明显

这种理解正确么?

例子：

利用线性回归公式：

$$\mathbf{w}=\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

	Y 血压 SBP	x1 胆固醇 Chol	x2 年龄 Age
1	120	126	38
2	125	128	40
3	130	128	42
4	121	130	42
5	135	130	44
6	140	132	46

可得的： $y = 397 - 3.75x_1 + 5.13x_2$

根据系数可知：年龄对血压有正作用（年纪越大，血压越高）

胆固醇对血压有反作用（胆固醇越高，血压越低）

明显错误的结论

从数据的直接观测上可以看出，血压会随着胆固醇的增加而增加

产生错误的原因 （1）数据量过少
（2）两个参数 x1和x2相关性过大

为了解决上述问题，对观测数据 X 进行PCA降维 通过PCA降维，可以把X中的样本，投影到一系列正交矢量上

得到一组新的观测数据t，且这些数据之间的相关性较小。利用新的数据t 再针对y 进行线性回归

t

	SBP	Chol	Age	PC1
1	120	126	38	105
2	125	128	40	108
3	130	128	42	109
4	121	130	42	111
5	135	130	44	112
6	140	132	46	115

PCA的系数

$$t = 0.589x_1 + 0.808x_2$$

$$y = -83.9 + 1.932t$$

带入可得：$y=-83.9+1.14x_1 +1.56x_2$

主成分回归的主要问题

PCA的系数

$t = 0.589x_1 + 0.808x_2$

$y = -83.9 + 1.932t$

PCA的系数，计算过程中只考虑了观测量X 的分布

其目的是找到X变化最广的成分

在回归分系统，关于数据有2点要求

（1） 观测数据和预测数据，变化范围要大

（2）观测数据和预测数据要有相关性

PCA回归只考虑了其中1点

为了兼顾上述两点，引入偏最小二乘回顾（**PLS**）

PLS的基本步骤

找到**几组**权重**w**，**对X进行加权求和得到t**

$$t_1^{(1)} = w_1^{(1)} x_{11} + w_2^{(1)} x_{12} + ...$$

$$t_2^{(1)} = w_1^{(1)} x_{21} + w_2^{(1)} x_{22} + ...$$

.
.
.

要求：t与y**要相关**，t之间尽量**不相关**

$$t_1^{(2)} = w_1^{(2)} x_{11} + w_2^{(2)} x_{12} + ...$$

$$t_2^{(2)} = w_1^{(2)} x_{21} + w_2^{(2)} x_{22} + ...$$

.
.
.

利用**t**对**y**进行回归

偏

PLS算法的实现

Nonlinear iterative partial least squares (NIPALS)

$\mathbf{X}$ (N,D) 观测数据

Y (N,1) 预测数据

g 从**X**中提取g个成分

$\mathbf{W}$ (D,g) 对**X**进行加权的权重

$\mathbf{T}$ (N,g) 对**X**进行加权求和后的数值

$\mathbf{C}$ (1,g) 对**T**对**Y**的回归系数

P (D,g) 对**T**对**X**的回归系数

$$\hat{\mathbf{X}}=\mathbf{T}\mathbf{P}^{\mathrm{T}}$$

$$\mathbf{X}^{(0)}=\mathbf{X} \qquad \mathrm{Y}^{(0)}=\mathrm{Y}$$

for j in g

$$\mathbf{W}_j=\frac{\left(\mathbf{X}^{(j)}\right)^T\mathbf{Y}}{\left\|\left(\mathbf{X}^{(j)}\right)^T\mathbf{Y}\right\|_2}$$

（**W**的第j列）

$$\mathbf{T}_j=\mathbf{X}^{(j)}\mathbf{W}_j$$

（**T**的第j列）

$$\mathbf{C}_j=\frac{\mathbf{T}_j^T\mathbf{Y}^{(j)}}{\left\|\mathbf{T}_j\right\|_2^2}$$

C_j过小退出

$$\mathbf{P}_j=\frac{\mathbf{X}_j^T\mathbf{T}_j}{\left\|\mathbf{T}_j\right\|_2^2}$$

$$\mathbf{X}^{(j+1)}=\mathbf{X}^{(j)}-\mathbf{T}_j\mathbf{P}_j^T$$

$$\mathbf{Y}^{(j+1)}=\mathbf{Y}^{(j)}-\mathbf{T}_j\mathbf{C}_j$$

$\mathbf{X}^{(0)} = \mathbf{X} \qquad \mathrm{Y}^{(0)} = \mathrm{Y}$

for j in g

$$\mathbf{W}_j = \frac{\left(\mathbf{X}^{(j)}\right)^T \mathbf{Y}}{\left\| \left(\mathbf{X}^{(j)}\right)^T \mathbf{Y} \right\|_2}$$ → 根据X和Y的相关性确定w

$$\mathbf{T}_j = \mathbf{X}^{(j)} \mathbf{W}_j$$ → 对X进行加权取一个成分 T

$$\mathbf{C}_j = \frac{\mathbf{T}_j^T \mathbf{Y}^{(j)}}{\left\| \mathbf{T}_j \right\|_2^2}$$ → 利用T对Y进行回归

$$\mathbf{P}_j = \frac{\mathbf{X}_j^T \mathbf{T}_j}{\left\| \mathbf{T}_j \right\|_2^2}$$ → 利用T对X进行回归

$$\mathbf{X}^{(j+1)} = \mathbf{X}^{(j)} - \mathbf{T}_j \mathbf{P}_j^T$$ → 去除X中的成分j

$$\mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} - \mathbf{T}_j \mathbf{C}_j$$ → 去除Y中的成分j

46

$$\boxed{\mathbf{w} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}}$$ → 线性回归公式

$$\hat{\mathbf{X}} = \mathbf{T}\mathbf{P}^{\mathrm{T}}$$

$$\mathrm{L}_{\mathrm{mse}} = \sum_{i,j}^{N,D} \left(x_{ij} - \hat{x}_{ij}\right)^2 = \mathrm{trac}\left((\mathbf{X} - \hat{\mathbf{X}})^T (\mathbf{X} - \hat{\mathbf{X}})\right)$$

$$= \mathrm{trac}\left((\mathbf{X} - \mathbf{T}\mathbf{P}^{\mathrm{T}})^T (\mathbf{X} - \mathbf{T}\mathbf{P}^{\mathrm{T}})\right)$$

求导可得 $\mathbf{P} = \left(\mathbf{T}^{\mathrm{T}} \mathbf{T}\right)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{T}$

预测过程

（1） 直接预测

$$\mathbf{b}=\mathbf{W}\left(\mathbf{P}^{\mathrm{T}}\mathbf{W}\right)^{-1}\mathbf{C}^{\mathrm{T}}$$

$$\tilde{\mathbf{Y}}=\tilde{\mathbf{X}}\mathbf{b}$$

（2） 迭代预测

$$\tilde{\mathbf{X}}^{(0)}=\tilde{\mathbf{X}} \qquad \tilde{\mathbf{T}}=\mathrm{zeros(N,g)}$$

for j in g

$$\mathrm{T}_j=\mathbf{X}^{(j)}\mathrm{W}_j$$

$$\mathbf{X}^{(j+1)}=\mathbf{X}^{(j)}-\mathrm{T}_j\mathrm{P}_j^T$$

$$\mathbf{Y}=\mathbf{T}\mathbf{C}^{\mathrm{T}}$$

```python
class PLS1():
    # Y 回归目标 维度 Nx1
    # X 观测/训练数据 维度 NxD
    # g 将观测数据 X 变为潜变量T, 维度从D降维到g
    def __init__(self,Y,X,g,bias=False):

        # 最终的潜变量的成分数
        self.components = g

        # 对数据进行预处理,计算均值
        # 不在数据上加偏置, 则需要对数据进行减均值的处理
        self.bias = bias
        if not bias:
            self.mean_X = np.mean(X,axis=0,keepdims=True)
            self.mean_Y = np.mean(Y,axis=0,keepdims=True)

            self.X = X-self.mean_X
            self.Y = Y-self.mean_Y
        else:
            self.X = X
            self.Y =Y

        # 获取特征维度
        self.N,self.D = np.shape(X)
        self.g = g
```

数据预处理

```python
# 获取特征维度
self.N,self.D = np.shape(X)
self.g = g

# 对X进行降维时, g个基的系数
W= np.empty([self.D,self.g])

# 利用潜变量对X进行回归的系数
P = np.empty([self.D, self.g])

# 存储变换后的潜变量
T = np.empty([self.N, self.g])

# 潜变量T 对 Y的 回归系数
c = np.empty([1, self.g])

# 最终的回归系数
b = np.empty((1, self.D))
```

for j in g

$$\mathbf{W}_j = \frac{\left(\mathbf{X}^{(j)}\right)^T \mathbf{Y}}{\left\| \left(\mathbf{X}^{(j)}\right)^T \mathbf{Y} \right\|_2} \longrightarrow$$ 根据X和Y的相关性确定w

$$\mathbf{T}_j = \mathbf{X}^{(j)} \mathbf{W}_j \longrightarrow$$ 对X进行加权取一个成分 T

$$\mathbf{C}_j = \frac{\mathbf{T}_j^T \mathbf{Y}^{(j)}}{\left\| \mathbf{T}_j \right\|_2^2} \longrightarrow$$ 利用T对Y进行回归

$$\mathbf{P}_j = \frac{\mathbf{X}_j^T \mathbf{T}_j}{\left\| \mathbf{T}_j \right\|_2^2} \longrightarrow$$ 利用T对X进行回归

$$\mathbf{X}^{(j+1)} = \mathbf{X}^{(j)} - \mathbf{T}_j \mathbf{P}_j^T \longrightarrow$$ 去除X中的成分j

$$\mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} - \mathbf{T}_j \mathbf{C}_j \longrightarrow$$ 去除Y中的成分j

49

```python
X_j = self.X
Y_j = self.Y
for j  in range(g):
    # 计算X, 每个维度上的特征与Y的相关性
    # 并利用这个相关性作为初始权重
    w_j = X_j.T @ Y_j
    w_j /= np.linalg.norm(w_j, 2)

    # 对 X 进行加权求和得到 t
    t_j = X_j @ w_j
    tt_j = t_j.T @ t_j

    # 利用 t 对 Y 进行回归得到系数 c
    c_j = (t_j.T @ Y_j) / tt_j

    if c_j<1e-6:
        self.components=j
        break

    # 利用t对X 进行回归得到回归系数 P
    p_j = (X_j.T @ t_j) / tt_j

    # 利用 t,P 计算 X的残差
    X_j = X_j - np.outer(t_j, p_j.T)

    # 利用 t,c 计算 Y的残差
    Y_j = Y_j - t_j * c_j
```

```python
self.W = W[ :, 0:self.components]
self.P = P[ :, 0:self.components]
self.T = T[ :, 0:self.components]
self.c = c[ 0:self.components]

# 迭代结束后计算b, Y= Xb^T
# b = W* (P^T*W) ^-1*C^T
b = self.W@ np.linalg.inv(self.P.T @ self.W) @ self.c.T
self.b = b
```

结果收集

直接预测

```python
def prediction(self, Z):
    if not self.bias:
        return self.mean_Y + (Z - self.mean_X) @ self.b

    else:
        return Z @ self.b
```

$$\mathbf{b}=\mathbf{W}\left(\mathbf{P}^{\mathrm{T}}\mathbf{W}\right)^{-1}\mathbf{C}^{\mathrm{T}}$$

$$\tilde{\mathbf{Y}} = \tilde{\mathbf{X}}\mathbf{b}$$

迭代预测

```
def prediction_iterative(self, Z):
    N,_ = np.shape(Z)
    if not self.bias:
        result = self.mean_Y.copy()
        X_j = Z-self.mean_X
    else:
        X_j = Z
        result = np.zeros([N,1])

    t = np.empty((N,self.components))
    for j in range(self.components):
        w_j = np.expand_dims(self.W[:, j],axis=-1)
        p_j = np.expand_dims(self.P[:, j],axis=-1)

        t_j = X_j @ w_j
        X_j = X_j - np.outer(t_j, p_j.T)
        t[:,j]=t_j[:,0]
    result = result +  t @ self.c.T

    return result
```

（2）迭代预测

$$\tilde{\mathbf{X}}^{(0)}=\tilde{\mathbf{X}} \qquad \tilde{\mathbf{T}}=\text{zeros}(\text{N,g})$$

for j in g

$$\mathrm{T}_j=\mathbf{X}^{(j)}\mathrm{W}_j$$

$$\mathbf{X}^{(j+1)}=\mathbf{X}^{(j)}-\mathrm{T}_j\mathrm{P}_j^T$$

$$\mathbf{Y}=\mathbf{TC}^{\mathrm{T}}$$

测试代码

```
if __name__ == "__main__":
    # X = np.array([[126,38],[128,40],[128,42],[130,42],[130,44],[132,46]]).astype(np.float32)
    X = np.array([[1,126,38],[1,128,40],[1,128,42],[1,130,42],[1,130,44],[1,132,46]]).astype(np.float32)
    Y = np.array([[120],[125],[130],[121],[135],[140]]).astype(np.float32)
    m_PLS = PLS1(Y,X,g=1,bias=True)

    result = m_PLS.prediction(X)
    print(m_PLS.b)
    print(result)


    result2 = m_PLS.prediction_iterative(X)
    print(result2)
```

参考文献

`Overview and Recent Advances in Partial Least Squares`| Roman Rosipal and Nicole Krämer| SLSFS 2005, LNCS 3940, pp. 34–51, 2006

参考代码

https://github.com/jhumphry/regressions