|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Example | 1 | 2 | 3 | 4 | Average |
| X1 | 2 | 3 | 5 | 6 | 4 |
| X2 | 2 | 4 | 4 | 6 | 4 |
| X3 | 4 | 6 | 2 | 4 | 4 |
| X1-u1 | -2 | -1 | 1 | 2 |  |
| X2-u2 | -2 | 0 | 0 | 2 |  |
| X3-u3 | 0 | 2 | -2 | 0 |  |
|  | | | | | Sum |
| (X1-u1)^2 | 4 | 1 | 1 | 4 | 10 |
| (X2-u2)^2 | 4 | 0 | 0 | 4 | 8 |
| (X3-u3)^2 | 0 | 4 | 4 | 0 | 8 |
| (X1-u1)(X2-u2) | 4 | 0 | 0 | 4 | 8 |
| (X1-u1)(X3-u3) | 0 | -2 | -2 | 0 | -4 |
| (X2-u2)(X3-u3) | 0 | 0 | 0 | 0 | 0 |
|  | | | | | |
|  | Cov(x1,x1) | Cov(x1,x2) | Cov(x1,x3) |  | |
| Cov | 10/3 | 8/3 | -4/3 |
| 8/3 | 8/3 | 0. |
| -4/3 | 0. | 8/3 |
|  | | | |

实验环境：python3.6.x+pycharm+window10

代码如下：

def **calculate\_cov**(x):

x=x.T

*'''m个样本n个特征'''*

n,m=x.shape

x=np.array(x)

mean\_x=np.zeros((n,1))

for i in range(n):

mean\_x[i]=np.mean(x[i,:])

mean\_x=np.ones((n,m))\*mean\_x

xT=x-mean\_x

cov=np.dot(xT,xT.T)/(m-1)

return cov

def **cov**(x):

x=x.T

cov=np.cov()

return cov

调试代码：

x=[[2,2,4],[3,4,6],[5,4,2],[6,6,4]]

x=np.array(x)

m,n=x.shape

print(m,*" "*,n)

print(calculate\_cov(x),*'\n'*)

print(np.cov(x.T))

注明：

输入矩阵为：m行n列（m个样本n个特征），本例中有4个样本3维矩阵

Python输出的结果;

编程计算得到的结果：

[[ 3.33333333 2.66666667 -1.33333333]

[ 2.66666667 2.66666667 0. ]

[-1.33333333 0. 2.66666667]]

调用numpy.cov()得出的结果

[[ 3.33333333 2.66666667 -1.33333333]

[ 2.66666667 2.66666667 0. ]

[-1.33333333 0. 2.66666667]

结论：

三种方式的答案完全一样。