PCA(Pricipal component analysis) Unsupervised Learning

Covariance Matrix Computation

In [5]:
```
import numpy as np
```

In [6]:
```
marks = np.array([[90,90,60,60,30],[60,90,60,60,30],[90,30,60,90,30]])
```

In [7]:
```
mean_marks = np.mean(marks, axis=1)
```

In [8]:
```
mean_marks
```

Out[8]:
```
array([66., 60., 60.])
```

In [9]:
```
covMat =np.cov(marks,bias=True)
```

In [10]:
```
covMat
```

Out[10]:
```
array([[504., 360., 180.],
       [360., 360.,   0.],
       [180.,   0., 720.]])
```

Compute Eigenvalue and Eigenvector

In [11]:
```
eig_val, eig_vec = np.linalg.eig(covMat)
```

In [12]:
```
eig_val
```

Out[12]:
```
array([ 44.81966028, 910.06995304, 629.11038668])
```

In [13]:
```
eig_vec
```

Out[13]:
```
array([[ 0.6487899 , -0.65580225, -0.3859988 ],
       [-0.74104991, -0.4291978 , -0.51636642],
       [-0.17296443, -0.62105769,  0.7644414 ]])
```

Sort Eigenvalue Choose k Eigenvector

In [14]:
```
eig_pairs = [(np.abs(eig_val[i]),eig_vec[:,i]) for i in range(len(eig_v
```

In [15]:
```
eig_pairs.sort(key=lambda x:x[0], reverse=True)
```

In [16]:
```
for i in eig_pairs:print(i[0])
```

```
910.0699530410361
629.1103866763253
44.81966028263878
```

In [17]:
```
matrix_w = np.hstack((eig_pairs[0][1].reshape(3,1),eig_pairs[1][1].resh
```

In [18]:
```
print('Matrix W:\n',matrix_w)
```

```
Matrix W:
 [[-0.65580225 -0.3859988 ]
 [-0.4291978  -0.51636642]
 [-0.62105769  0.7644414 ]]
```

Transform the value in new subspace

In [19]:
```
transformed = matrix_w.T.dot(marks-mean_marks.reshape(3,1))
```

In [20]:
```
transformed
```

Out[20]:
```
array([[-34.37098481,  -9.98345733,   3.93481353, -14.69691716,
         55.11654576],
       [ 13.66927088, -47.68820559,   2.31599277,  25.24923474,
          6.45370719]])
```

Calculate using sklearn

In [21]:
```
from sklearn.decomposition import PCA as sklearnPCA
```

In [22]:
```
sklearn_pca = sklearnPCA(n_components =2)
```

In [23]:
```
sklearn_transf = sklearn_pca.fit_transform(marks.T)
```

In [24]:
```
sklearn_transf
```

Out[24]:
```
array([[-34.37098481, -13.66927088],
       [ -9.98345733,  47.68820559],
       [  3.93481353,  -2.31599277],
       [-14.69691716, -25.24923474],
       [ 55.11654576,  -6.45370719]])
```

Uniqueness Of Principal Components

In [25]:
```
sklearn_pca.components_
```

Out[25]:
```
array([[-0.65580225, -0.4291978 , -0.62105769],
       [ 0.3859988 ,  0.51636642, -0.7644414 ]])
```

Proportion of Variance Explained

In [27]:
```
eig_val[::-1].sort()
```

In [28]:
```
eig_val
```

Out[28]:
```
array([910.06995304, 629.11038668,  44.81966028])
```

In [31]:
```
eig_val/eig_val.sum()
```

Out[31]:
```
array([0.57453911, 0.39716565, 0.02829524])
```

In [29]:
```
sklearn_pca.explained_variance_ratio_
```

Out[29]:
```
array([0.57453911, 0.39716565])
```

In [ ]:
```
#Cumulative variance explained is
```

In [30]:
```
sklearn_pca.explained_variance_ratio_.cumsum()
```

Out[30]:
```
array([0.57453911, 0.97170476])
```

Deciding the number of components

In [32]:
```
marks1 = np.array([[90,90,60,60,30],[60,90,60,60,30],[90,30,60,90,30],[
```

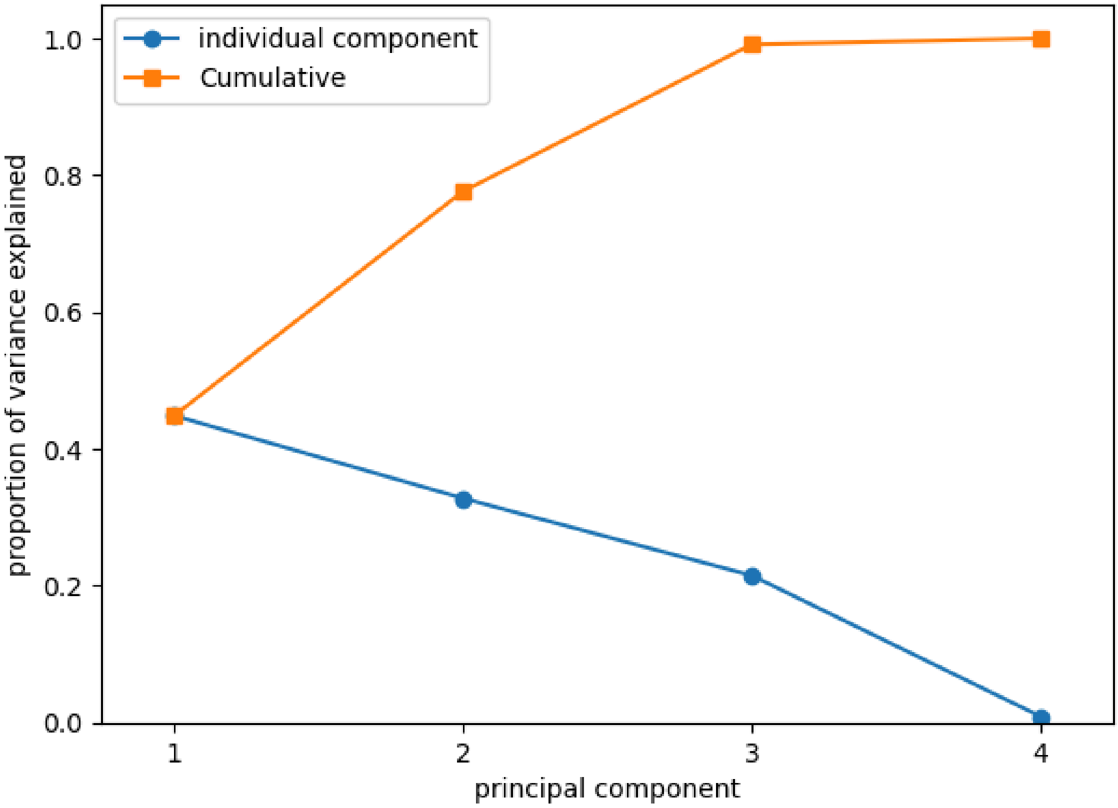
In [33]:
```
sklearn_pca = sklearnPCA(n_components=4)
```

In [34]:
```
sklearn_transf = sklearn_pca.fit_transform(marks1.T)
```

In [35]:
```
from matplotlib import pyplot as plt
```

In [37]:

```
plt.figure(figsize=(7,5))
plt.plot([1,2,3,4],sklearn_pca.explained_variance_ratio_,'-o',label='ir
plt.plot([1,2,3,4],np.cumsum(sklearn_pca.explained_variance_ratio_), '-
plt.ylabel('proportion of variance explained')
plt.xlabel('principal component')
plt.xlim(0.75,4.25)
plt.ylim(0,1.05)
plt.xticks([1,2,3,4])
plt.legend(loc=2)
plt.show()
```



In [ ]:

```

```