pca_covariance_correlation

COSC 3337

[3]:
```python
import pandas as pd
df = pd.read_csv("iris.csv")
#df.columns=['sepal_len', 'sepal_wid', 'petal_len', 'petal_wid', 'class']
df.dropna(how="all", inplace=True) # drops the empty line at file-end
df.tail()
```

[3]:
```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

[5]:
```python
# split data table into data X and class labels y

X = df.iloc[:,0:4].values
y = df.iloc[:,4].values
```

[6]:
```python
from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(X)
```

[7]:
```python
import numpy as np
mean_vec = np.mean(X_std, axis=0)
cov_mat = (X_std - mean_vec).T.dot((X_std - mean_vec)) / (X_std.shape[0]-1)
print('Covariance matrix \n%s' %cov_mat)
```

```
Covariance matrix
[[ 1.00671141  0.72148618 -0.40039813  0.8886718 ]
 [ 0.72148618  1.00671141 -0.11010327  0.87760486]
 [-0.40039813 -0.11010327  1.00671141 -0.42333835]
 [ 0.8886718   0.87760486 -0.42333835  1.00671141]]
```

[8]:
```python
cov_mat = np.cov(X_std.T)

eig_vals, eig_vecs = np.linalg.eig(cov_mat)

print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

```
Eigenvectors
[[ 0.55318314  0.31153594 -0.77256222 -0.00902118]
 [ 0.51774664  0.48025478  0.56930389 -0.42093567]
 [-0.28847469 -0.16889872 -0.2641027  -0.90471285]
 [ 0.58541369 -0.80235523  0.09638701 -0.06501105]]

Eigenvalues
[2.83122907 0.04725055 0.22729518 0.92107083]
```

[9]:
```python
#Eigendecomposition of the standardized data based on the correlation matrix:
cor_mat1 = np.corrcoef(X_std.T)

eig_vals, eig_vecs = np.linalg.eig(cor_mat1)

print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

```
Eigenvectors
[[ 0.55318314  0.31153594 -0.77256222 -0.00902118]
 [ 0.51774664  0.48025478  0.56930389 -0.42093567]
 [-0.28847469 -0.16889872 -0.2641027  -0.90471285]
 [ 0.58541369 -0.80235523  0.09638701 -0.06501105]]

Eigenvalues
[2.81235421 0.04693554 0.22577988 0.91493036]
```

[10]:
```python
#Eigendecomposition of the raw data based on the correlation matrix:
cor_mat2 = np.corrcoef(X.T)

eig_vals, eig_vecs = np.linalg.eig(cor_mat2)

print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

```
Eigenvectors
[[ 0.55318314  0.31153594 -0.77256222 -0.00902118]
 [ 0.51774664  0.48025478  0.56930389 -0.42093567]
 [-0.28847469 -0.16889872 -0.2641027  -0.90471285]
 [ 0.58541369 -0.80235523  0.09638701 -0.06501105]]

Eigenvalues
```

```
[2.81235421 0.04693554 0.22577988 0.91493036]
```

[11]:
```python
#Singular Value Decomposition (SVD) to improve the computational efficiency
u,s,v = np.linalg.svd(X_std.T)
u
```

[11]:
```
array([[-0.55318314,  0.00902118,  0.77256222, -0.31153594],
       [-0.51774664,  0.42093567, -0.56930389, -0.48025478],
       [ 0.28847469,  0.90471285,  0.2641027 ,  0.16889872],
       [-0.58541369,  0.06501105, -0.09638701,  0.80235523]])
```

[12]:
```python
for ev in eig_vecs.T:
    np.testing.assert_array_almost_equal(1.0, np.linalg.norm(ev))
print('Everything ok!')
```

```
Everything ok!
```

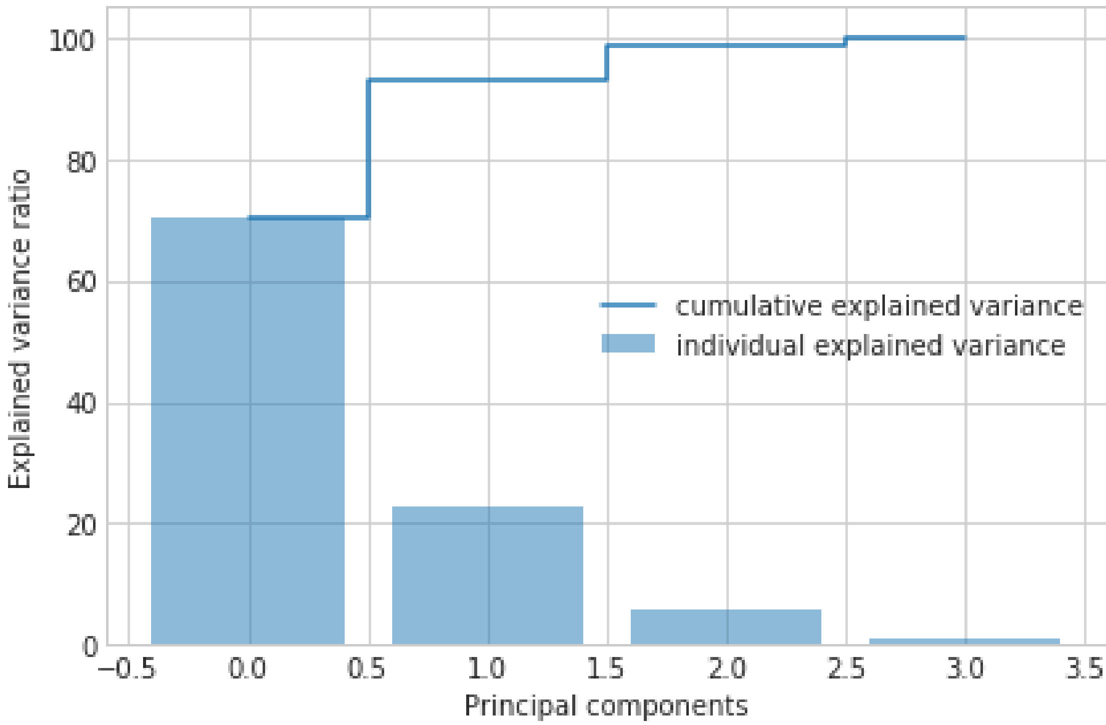
[13]:
```python
# Make a list of (eigenvalue, eigenvector) tuples
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the (eigenvalue, eigenvector) tuples from high to low
eig_pairs.sort(key=lambda x: x[0], reverse=True)

# Visually confirm that the list is correctly sorted by decreasing eigenvalues
print('Eigenvalues in descending order:')
for i in eig_pairs:
    print(i[0])
```

```
Eigenvalues in descending order:
2.8123542144739995
0.9149303606832303
0.22577988052025108
0.04693554432251515
```

[14]:
```python
#how many principal components are we going to choose for our new feature␣
↪subspace?"
tot = sum(eig_vals)
var_exp = [(i / tot)*100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
```

[16]:
```python
import matplotlib.pyplot as plt
with plt.style.context('seaborn-whitegrid'):
    plt.figure(figsize=(6, 4))

    plt.bar(range(4), var_exp, alpha=0.5, align='center',
            label='individual explained variance')
    plt.step(range(4), cum_var_exp, where='mid',
             label='cumulative explained variance')
```

```
    plt.ylabel('Explained variance ratio')
    plt.xlabel('Principal components')
    plt.legend(loc='best')
    plt.tight_layout()
```



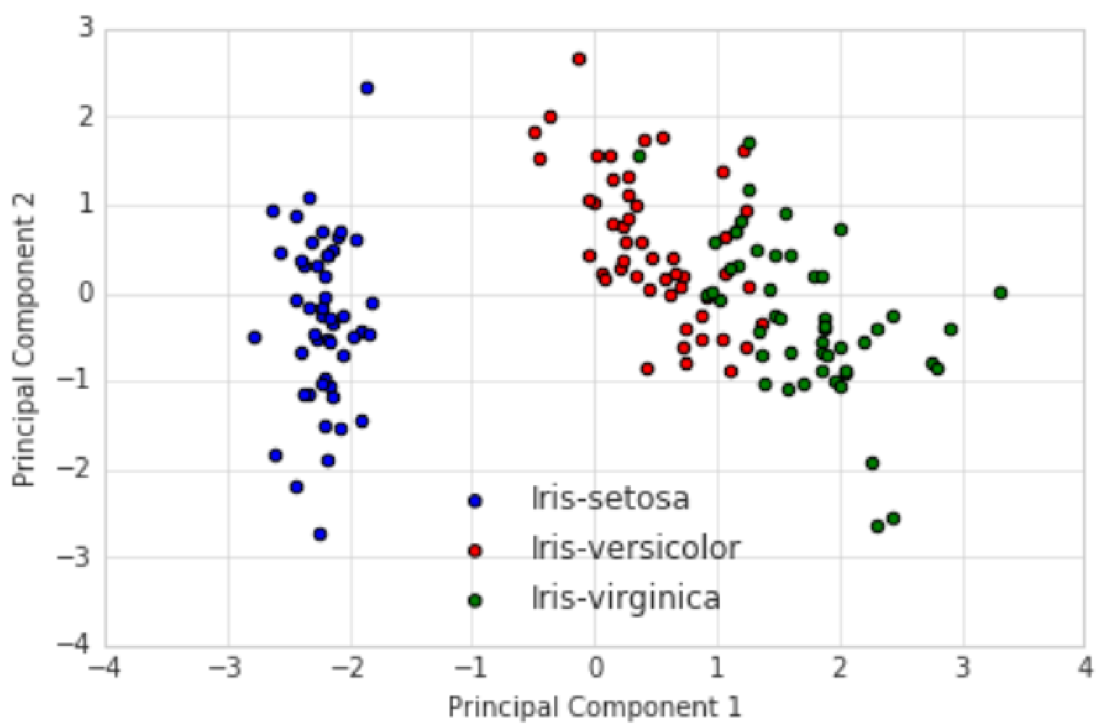
[17]:
```
#reducing the sice from 4 to 2 features
matrix_w = np.hstack((eig_pairs[0][1].reshape(4,1),
                      eig_pairs[1][1].reshape(4,1)))

print('Matrix W:\n', matrix_w)
```

```
Matrix W:
 [[ 0.55318314 -0.00902118]
 [ 0.51774664 -0.42093567]
 [-0.28847469 -0.90471285]
 [ 0.58541369 -0.06501105]]
```

[28]:
```
#Projection Onto the New Feature Space
Y = X_std.dot(matrix_w)
```

[34]:
```
import warnings
with plt.style.context('seaborn-whitegrid'):
    plt.figure(figsize=(6, 4))
```

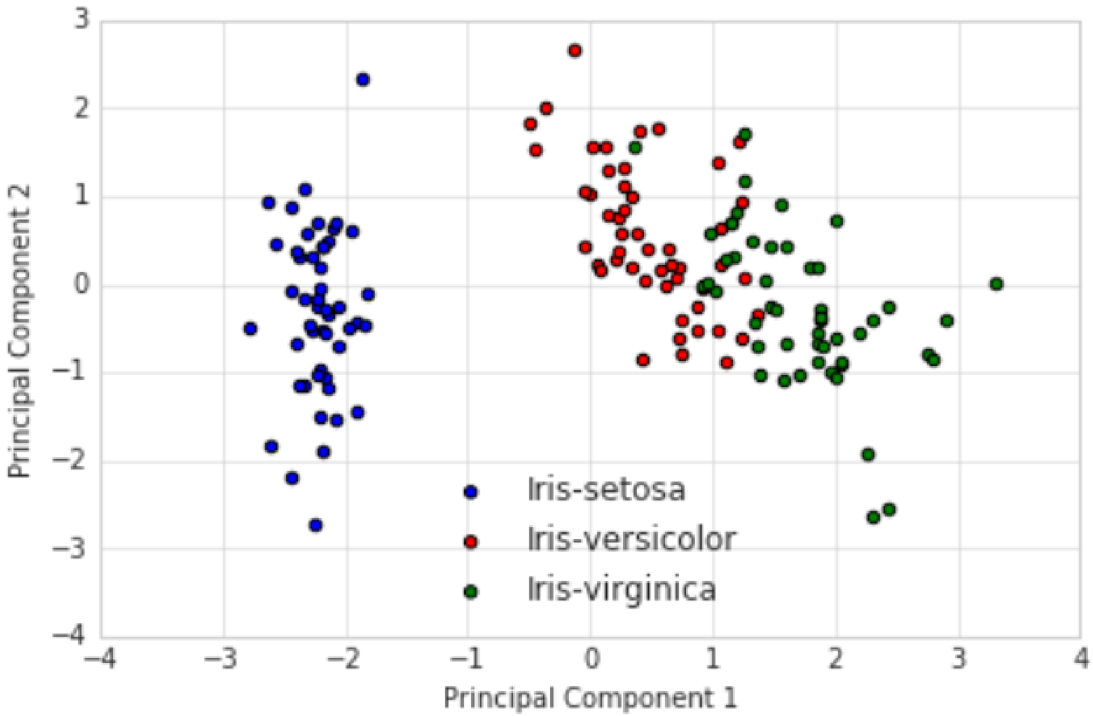
```
    warnings.simplefilter(action='ignore', category=FutureWarning)
    for lab, col in␣
↪zip(('Iris-setosa','Iris-versicolor','Iris-virginica'),('blue', 'red',␣
↪'green')):
        plt.scatter(Y[y==lab, 0],Y[y==lab, 1],label=lab,c=col)
    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    plt.legend(loc='lower center')
    plt.tight_layout()
    plt.show()
```



[35]:
```
from sklearn.decomposition import PCA as sklearnPCA
sklearn_pca = sklearnPCA(n_components=2)
Y_sklearn = sklearn_pca.fit_transform(X_std)
```

[39]:
```
with plt.style.context('seaborn-whitegrid'):
    plt.figure(figsize=(6, 4))
    for lab,col in zip(('Iris-setosa', 'Iris-versicolor',␣
↪'Iris-virginica'),('blue', 'red', 'green')):
        plt.scatter(Y_sklearn[y==lab, 0], Y_sklearn[y==lab, 1], label=lab,c=col)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(loc='lower center')
```

```
plt.tight_layout()
plt.show()
```



[ ]: