Rudra_Ronit_HW1_Practicum_Q3

September 20, 2016

1 Question

Load the iris sample dataset into Python using a Pandas dataframe. Perform a PCA using the Scikit Decomposition component, and provide the percentage of variance explained by the 1st Principal Component. Use Matplotlib to plot the 1st/2nd Principal Components to recreate the scatterplot shown in class, with colored classes for each flower type.

2 Solution

First, we import the necessary python modules for loading up the dataset, performing operations and visualization.

```
In [1]: %matplotlib inline

        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        plt.style.use('ggplot')
        from sklearn.decomposition import PCA
```

Now we import the dataset and store it in a variable. We use the function read_table() in pandas for this. The file extension is .tab which is a tab separated file.

```
In [2]: iris = pd.read_table('iris.tab',sep='\t')
```

After the file is loaded, let us look what the data is actually like.

```
In [3]: print("The dimensions of the data is:",iris.shape)
        print("This is what the column headers are:")
        print(iris.columns)
        print("First few instances of the dataset")
        iris.head(6)
```

```
The dimensions of the data is: (152, 5)
This is what the column headers are:
Index(['sepal length', 'sepal width', 'petal length', 'petal width', 'iris'], dtype='object')
First few instances of the dataset
```

Out[3]:

	sepal length	sepal width	petal length	petal width	iris
0	c	c	c	c	d
1	NaN	NaN	NaN	NaN	class
2	5.1	3.5	1.4	0.2	Iris-setosa
3	4.9	3.0	1.4	0.2	Iris-setosa
4	4.7	3.2	1.3	0.2	Iris-setosa
5	4.6	3.1	1.5	0.2	Iris-setosa

As seen in the auto-mpg dataset, the first two rows are of no use to us anymore. We proceed to remove them and reindex the dataset. Thus, we will have 150 instances with 4 attributes and 1 class label.

```
In [4]: iris = iris[2:]
        iris.index = range(iris.shape[0])
        iris.head(4)
```

Out[4]:

	sepal length	sepal width	petal length	petal width	iris
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa

To perform PCA let's create a data frame containing only the attributes.

```
In [5]: iris_x = iris.iloc[:,0:4]
        iris_x.head(3)
```

Out[5]:

	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2

The next steps perform Principal Component Analysis on the attributes. This involves calculation of covariance matrix, transformation into eigen space and calculation of eigen vectors and eigen values.

```
In [6]: pca = PCA(n_components=4)
        iris_x = pca.fit(iris_x).transform(iris_x)
        print("The variance ratio explained by the components are:",'\n',
            pca.explained_variance_ratio_)
```

```
The variance ratio explained by the components are:
 [ 0.92461621  0.05301557  0.01718514  0.00518309]
```

The Variance Ratio is calculated by calculating the covariance matrix of the principal components and then dividing individual variance by total variance. This is demonstrated easily.

```
In [7]: print("Variance of principal components:",'\n',pca.explained_variance_)
        print("Explained variance ratio:",'\n',
              pca.explained_variance_/(np.sum(pca.explained_variance_)))
```

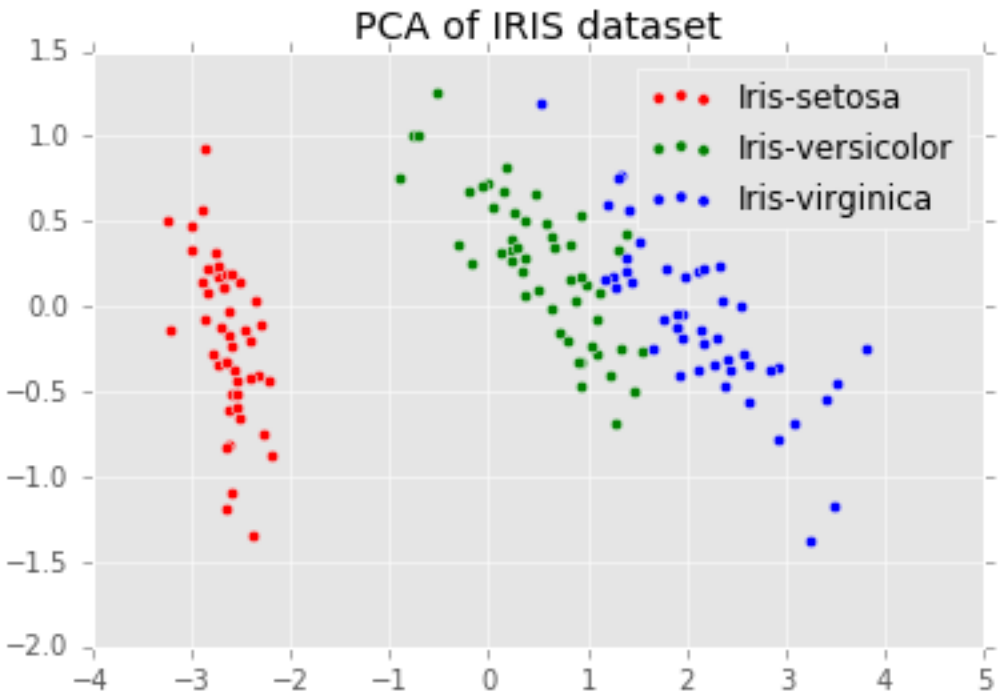
```
Variance of principal components:
 [ 4.19667516  0.24062861  0.07800042  0.02352514]
Explained variance ratio:
 [ 0.92461621  0.05301557  0.01718514  0.00518309]
```

The above result is the same as what was previously displayed. Therefore, the percentage variance explained by the first principal component is 92.46%

We now plot the 1st two principal components.

```
In [8]: target = pd.DataFrame(iris["iris"])
        target_names=np.unique(target)
        iris_x = pd.DataFrame(iris_x)
        iris_pca = pd.concat([iris_x,target],axis=1)
        y = pd.DataFrame(np.repeat([0,1,2],50))
        plt.figure()
        for c, i, target_name in zip("rgb", [0, 1, 2], target_names):
```

```
    plt.scatter(iris_pca.iloc[np.where(y==i)[0],0],
                iris_pca.iloc[np.where(y==i)[0],1], c=c, label=target_name)
plt.legend()
plt.title('PCA of IRIS dataset')
plt.show()
```



The first two principal components provide a very good idea on how to classify the iris dataset. From the plot, Class Setosa can be classified with 100% accuracy while the other two classes are more or less separated. Thus, PCA is a very powerful tool in classification preprocessing as it reduces the dimensionality of the dataset (in this case from 4 to 2).