IRIS data set

March 22, 2016

IRIS data set

Download the IRIS data set from:

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

This is a data set of 150 points in R4, with three classes; refer to the website for more details of the features and classes.

(a) Use a PCA projection to 2d to visualize the entire data set. You should plot different classes using different colors/shapes. Do the classes seem well-separated from each other?

(b) Now build a classifier for this data set, based on a generative model.

```
In [24]: %matplotlib inline
         import pandas as pd
         import matplotlib.pyplot as plt
         import numpy as np
         import random
         from sklearn.decomposition import PCA
         from scipy.stats import multivariate_normal
```

0.1 Importing iris dataset

```
In [25]: !wget https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data -O iris.txt
```

```
--2016-03-13 22:30:19--  https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.249
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.249|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4551 (4.4K) [text/plain]
Saving to: 'iris.txt'

iris.txt            100%[====================>]   4.44K  --.-KB/s   in 0s

2016-03-13 22:30:19 (72.3 MB/s) - 'iris.txt' saved [4551/4551]
```

0.2 (a) Read Data and Perform PCA to 2 components

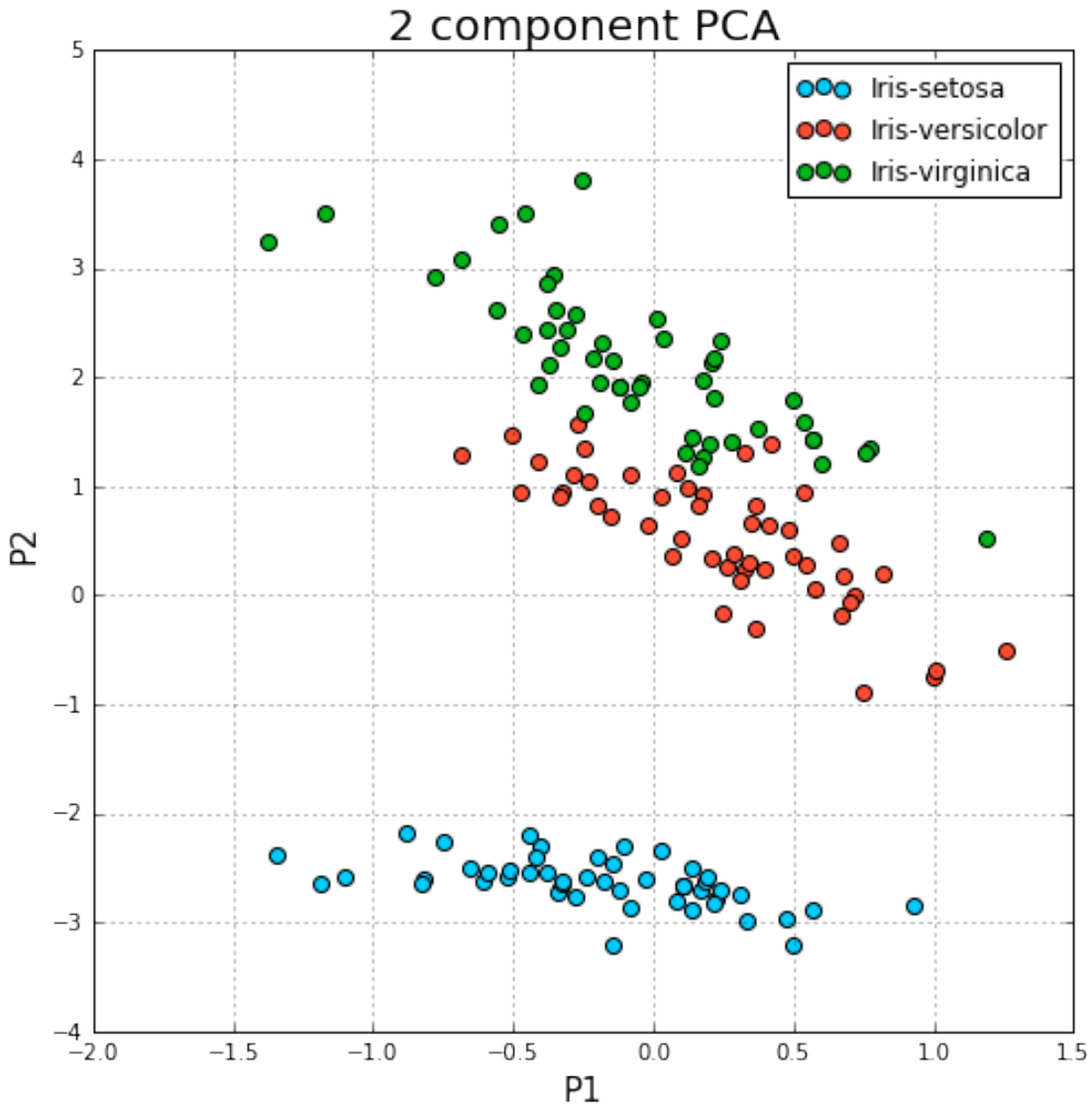
```
In [35]: iris_df = pd.read_csv('iris.txt', names=['sepal_len','sepal_wid','petal_len','petal wid','class
         pca = PCA(n_components = 2)
         pca.fit(iris_df.ix[:,0:4].values)
         X = pca.transform(iris_df.ix[:,0:4].values)
         iris_df['P1'] = X[:,0]
         iris_df['P2'] = X[:,1]
         iris_df.head(5)
```

Out[35]:

	sepal_len	sepal_wid	petal_len	petal wid	class	P1	P2
0	5.1	3.5	1.4	0.2	Iris-setosa	-2.684207	-0.326607

```
1        4.9        3.0        1.4        0.2  Iris-setosa -2.715391  0.169557
2        4.7        3.2        1.3        0.2  Iris-setosa -2.889820  0.137346
3        4.6        3.1        1.5        0.2  Iris-setosa -2.746437  0.311124
4        5.0        3.6        1.4        0.2  Iris-setosa -2.728593 -0.333925
```

In [27]:
```
classes = iris_df['class'].unique()
colors = [(0,.8,1),(1,.3,.2),(0,.7, .1)]
plt.figure(figsize = (8,8))
plt.xlabel('P1', fontsize = 15)
plt.ylabel('P2', fontsize = 15)
plt.title('2 component PCA', fontsize = 20)
for cl, color in zip(classes,colors):
    P1 = iris_df[iris_df['class'] == cl]['P1'].values
    P2 = iris_df[iris_df['class'] == cl]['P2'].values
    plt.scatter(P2, P1, c = color, s = 50)
plt.legend(classes)
plt.grid()
```



The three classes appear to be well separated!
iris-virginica and iris-versicolor could be better separated, but it is good!

0.3 (b) Train with Multivariate Gaussian Classifier

```
In [28]: df_train = iris_df[iris_df['class'] == classes[0]][0:35]
         for c in classes[1:]:
             df_train = pd.concat([df_train, iris_df[iris_df ['class'] == c][0:35]])

         df_test = iris_df[iris_df['class'] == classes[0]][35:]
         for c in classes[1:]:
             df_test = pd.concat([df_test, iris_df[iris_df['class'] == c][35:]])
```

```
In [29]: def classify(sample_df, valid_df):

             prob = []

             for i in xrange(3):
                 cond = sample_df['class'] == classes[i]
                 mean = np.mean(sample_df[cond].ix[:,0:4].values, axis = 0)
                 cov = np.cov(np.transpose(sample_df[cond].ix[:,0:4].values))
                 func = multivariate_normal(mean =mean, cov=cov)
                 prob.append(func.logpdf(valid_df.ix[:,0:4]))
             max_prob = np.argmax(prob, axis = 0)
             tf_number_error = [classes[i] != j for i,j in zip(max_prob, valid_df['class'])]
             error_percent = np.sum(tf_number_error)/len(valid_df)
             return error_percent
```

0.4 Train the dataset

```
In [30]: def find_error(sample_df, valid_df):

             prob, label, flower = [], [], []

             for i in xrange(3):
                 cond = sample_df['class'] == classes[i]
                 mean = np.mean(sample_df[cond].ix[:,0:4].values, axis = 0)
                 cov = np.cov(np.transpose(sample_df[cond].ix[:,0:4].values))
                 func = multivariate_normal(mean, cov)
                 prob.append(func.logpdf(valid_df.ix[:,0:4]))

             max_prob = np.argmax(prob, axis = 0)

             prob = np.matrix(prob)
             for i,j in zip(max_prob, valid_df['class']):
                 if  classes[i] != j:
                     flower.append(j)
                     label.append(classes[i])
             return [flower, label]
```

```
In [31]: #Create dataframe of flowers and predictions
         [flower, label] = find_error(df_train, df_train)
```

```
error_df1 = pd.DataFrame({'prediction': label, 'flower': flower})
error_df1.head()
```

Out[31]:

	flower	prediction
0	Iris-versicolor	Iris-virginica
1	Iris-virginica	Iris-versicolor

0.5 Test Dataset

In [33]:
```
error_percent = classify(df_train, df_test)

print 'Error: ', error_percent
```

```
Error:  0
```

100% accuracy with 0 percent error on our test dataset.