discriminant_analysis_2

September 30, 2019

[1]:
```
# src

# https://towardsdatascience.com/
 ↪linear-discriminant-analysis-in-python-76b8b17817c2
```

[2]:
```
# LDA steps
# 1. Compute the within class and between class scatter matrices
# 2. Compute the eigenvectors and corresponding eigenvalues for the scatter␣
 ↪matrices
# 3. Sort the eigenvalues and select the top k
# 4. Create a new matrix containing eigenvectors that map to the k eigenvalues
# 5. Obtain the new features (i.e. LDA components) by taking the dot product of␣
 ↪the data and the matrix from step 4
```

[3]:
```
%matplotlib inline
from matplotlib import pyplot as plt
import seaborn as sns

import numpy as np

import pandas as pd

from sklearn.datasets import load_wine
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

[4]:
```
wine_dataset = load_wine()

X = pd.DataFrame( wine_dataset.data, columns=wine_dataset.feature_names )
y = pd.Categorical.from_codes( wine_dataset.target, wine_dataset.target_names )

wine_df = X.join( pd.Series( y, name='class' ) )
```

[5]:
```
display( y.value_counts() )
```

```
class_0    59
```

```
class_1    71
class_2    48
dtype: int64
```

[6]:
```python
# For every class, create a vector with the means of each feature

class_feature_means = pd.DataFrame(columns=wine_dataset.target_names)

for c, rows in wine_df.groupby('class'):
    class_feature_means[c] = rows.mean()
```

[7]:
```python
# Obtain within class scatter matrix

num_of_features = X.shape[1]

within_class_scatter_matrix = np.zeros( (num_of_features, num_of_features) )

for c, rows in wine_df.groupby('class'):
    rows = rows.drop( ['class'], axis=1 )
    s = np.zeros( (num_of_features, num_of_features) )
    for index, row in rows.iterrows():
        x = row.values.reshape(13,1)
        mc = class_feature_means[c].values.reshape(13,1)
        s += (x - mc).dot((x - mc).T)
    within_class_scatter_matrix += s
```

[8]:
```python
within_class_scatter_matrix.shape
```

[8]:
```
(13, 13)
```

[9]:
```python
# Obtain between class scatter matrix

feature_means = wine_df.mean()

num_of_features = X.shape[1]

between_class_scatter_matrix = np.zeros( (num_of_features, num_of_features) )

for c in class_feature_means:
    n = wine_df.loc[wine_df['class'] == c].index.size
    mc, m = class_feature_means[c].values.reshape(13,1), feature_means.values.
 ↪reshape(13,1)
    between_class_scatter_matrix += n * (mc - m).dot((mc - m).T)
```

[10]:
```python
between_class_scatter_matrix.shape
```

[10]:
```
(13, 13)
```

[11]:
```python
# Solve the generalized eigenvalue problem to obtain the linear discriminants
```

```
SW = within_class_scatter_matrix
SWinv = np.linalg.inv(SW)
SB = between_class_scatter_matrix

eigen_values, eigen_vectors = np.linalg.eig(
    SWinv.dot(SB)
)
```

[12]:
```
eigen_values.shape, eigen_vectors.shape
```

[12]: ((13,), (13, 13))

[13]:
```
# The eigenvectors with the highest eigenvalues carry the most information
# about the distribution of the data.
# Thus, we sort the eigenvalues from highest to lowest and select
# the first k eigenvectors.

pairs = [
    (np.abs(eigen_values[i]), eigen_vectors[:,i])
    for i in range( len(eigen_values) )
]

pairs = sorted(pairs, key=lambda x: x[0], reverse=True)
```

[14]:
```
# Print how much of the variance is explained by each component

print( [pair[0] for pair in pairs] )
```

```
[9.081739435042469, 4.128469045639493, 8.301164656845475e-16,
6.220260092356696e-16, 6.220260092356696e-16, 5.149249584290688e-16,
5.149249584290688e-16, 1.8931182654809792e-16, 1.8066202080581507e-16,
7.487157321002816e-17, 7.487157321002816e-17, 6.316361304006824e-18, 0.0]
```

[15]:
```
# Print how much of the variance is explained by each component in pct

eigen_values_sum = sum( eigen_values )
print( [ (pair[0] / eigen_values_sum).real for pair in pairs ] )
```

```
[0.6874788878860778, 0.3125211121139222, 6.283901324483062e-17,
4.708676703666665e-17, 4.708676703666665e-17, 3.8979321119880345e-17,
3.8979321119880345e-17, 1.433072209457855e-17, 1.3675940169302205e-17,
5.667705647455687e-18, 5.667705647455687e-18, 4.781424391025773e-19, 0.0]
```

[16]:
```
# Create a matrix W with the first 2 eigenvectors

w_matrix = np.hstack((
    pairs[0][1].reshape(13,1),
    pairs[1][1].reshape(13,1)
)).real
```

[17]:
```
w_matrix.shape
```

[17]: (13, 2)

[18]:
```
# Create Y
# Y = X * W
# Y is composed of the LDA components == is the new feature space

X_lda = np.array( X.dot(w_matrix) )
```

[19]:
```
# Visualize results of LDA

sns.scatterplot(
    X_lda[:, 0], X_lda[:, 1],
    hue=wine_df['class']
)
plt.show()
```