Classification Systems

Daniel Cerdán, Fernando Freire

March 20, 2019

Contents

1 Classification Systems

In this practical, you are asked to compare the prediction error of:

1. The Naive Bayes Classifier
2. LDA
3. QDA
4. Nearest Shrunken Centroids Classifier

On the Breast Cancer dataset provided in the previous notebooks, and the Prostate cancer dataset attached. The details about this last dataset are found in the reference:

Singh, D., Febbo, P., Ross, K., Jackson, D., Manola, J., Ladd, C., Tamayo, P., Renshaw, A., D'Amico, A., Richie, J., Lander, E., Loda, M., Kantoff, P., Golub, T., & Sellers, W. (2002). Gene expression correlates of clinical prostate cancer behavior. Cancer Cell, 1, 203–209.

This dataset is in CSV format and the last column contains the class label. The task of interest is to discriminate between normal and tumor tissue samples.

Importantly:

Use a random split of 2 / 3 of the data for training and 1 / 3 for testing each classifier. Any hyper-parameter of each method should be tuned using a grid-search guided by an inner cross-validation procedure that uses only training data. To reduce the variance of the estimates, report average error results over 20 different partitions of the data into training and testing as described above. Submit a notebook showing the code and the results obtained. Give some comments about the results and respond to these questions:

What method performs best on each dataset? What method is more flexible? What method is more robust to over-fitting?

Script 1.0.1 (python)

```python
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# seaborn is a package for statistical data visualization
import seaborn as sns; sns.set()
import scipy.stats as stats
import scipy as sp
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

Script 1.0.2 (python)

```python
# read the data, 30 features for each record
data = pd.read_csv('./data/wdbc.csv',header=None) #as there is no header in this case
X = data.values[ :, 2:].astype(np.float)
# as a supervised problem, we have an associated label (benign, malign) to each sample on X
y = (data.values[ :, 1 ] == 'B').astype(np.int)
target_names = np.array(['malign', 'benign'], dtype=np.dtype('U10'))
print(target_names)
```

```python

# make a DataFrame with a species column for the first 5 features. Note that the last
↪  element is not included in the selection
X5 = X[ :, 0:5].astype(np.float)
df_cancer = pd.DataFrame(X5)
df_cancer['target'] = target_names[y]

# take a look at df_cancer.
df_cancer.head(5)

# Split dataset between training and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1.0/3, random_state=1)

# Data standardization
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Check standardization
print(np.var(X_train[:,0]))
print(np.var(X_train[:,1]))
for i in range (1, np.size(X_train_scaled,1)):
    assert round(np.var(X_train_scaled[:,0]),3) == round(np.var(X_train_scaled[:,i]),3),\
    "Warning: revise standardization"
```

Output

```
['malign' 'benign']
13.598316715519944
19.71087911250966
[[ 61   5]
 [  7 117]]
Predicion accuracy is: 0.936842
True postive rate is: 0.943548
True negative rate is: 0.924242
```

Script 1.0.3 (python)

```python
# Naive Bayes

# Training
nb = GaussianNB()
nb.fit(X_train_scaled, y_train)

# Evaluation
y_pred = nb.predict(X_test_scaled)

conf = confusion_matrix(y_test, y_pred)

# The matrix conf contains:
# TN | FN
# FP | TP

TN = conf[0][0]
TP = conf[1][1]
FP = conf[0][1]
FN = conf[1][0]

print(conf)
print('Predicion accuracy is: %f' % ((TP + TN) / (TN + TP + FP + FN)))
print('True postive rate is: %f' % (TP / (TP + FN)))
print('True negative rate is: %f\n' % (TN / (TN + FP)))
```

Output

```
[[ 61   5]
 [  7 117]]
Predicion accuracy is: 0.936842
True postive rate is: 0.943548
True negative rate is: 0.924242
```