COMPUTER PROECT 2
PATTERN RECOGNITION
ECEN 649

SPRING 2018

MAHESH NAIDU
227002187

TEXAS A&M UNIVERSITY
COLLEGE STATION

Result Tables:

LDA p=0.75

LDA			
Feature Selection Method	Gene Set Found	Error estimate	Test Error estimate
Exhaustive,2 features	COL4A2, AKAP2	0.208	0.268
Forward, 3 features	ALDH4, Contig55377_RC, PRC1	0.183	0.263
Forward, 4 features	LOC51203, ALDH4, Contig55377_RC, PRC1	0.175	0.263
Forward, 5 features	LOC51203, ALDH4, Contig55377_RC, PRC1, CFFM4	0.167	0.257
All Features	All	0	0.366

Linear SVM

Linear SVM			
Feature Selection Method	Gene Set Found	Error estimate	Test Error estimate
Exhaustive,2 features	IGFBP5.1, PRC1	0.258	0.268
Forward, 3 features	AL080059, Contig63649_RC, Contig46218_RC	0.267	0.268
Forward, 4 features	AL080059, Contig63649_RC, Contig46218_RC, LOC51203	0.267	0.268
Forward, 5 features	AL080059, Contig63649_RC, Contig46218_RC, LOC51203, IGFBP5	0.25	0.257
All Features	All	0.091	0.286

Non-Linear SVM (Gaussian RBF Kernel)

Non Linear SVM			
Feature Selection Method	Gene Set Found	Error estimate	Test Error estimate
Exhaustive,2 features	Contig63649_RC, IGFBP5.1	0.25	0.263
Forward, 3 features	Contig63649_RC, IGFBP5.1, PRC1	0.25	0.263
Forward, 4 features	Contig63649_RC, IGFBP5.1, PRC1, GNAZ	0.242	0.263
Forward, 5 features	Contig63649_RC, IGFBP5.1, PRC1, GNAZ, Contig55377_RC	0.25	0.268
All Features	All	0.266	0.268

NN (with 5 Neurons)

NN			
Feature Selection Method	Gene Set Found	Error estimate	Test Error estimate
Exhaustive,2 features	CEGP1, PECI.1	0.15	0.326
Forward, 3 features	Contig63649_RC, LOC51203, Contig32125_RC	0.158	0.326
Forward, 4 features	Contig63649_RC, LOC51203, Contig32125_RC, Contig55725_RC	0.125	0.326
Forward, 5 features	Contig63649_RC, LOC51203, Contig32125_RC, Contig55725_RC, IGFBP5	0.058	0.28
All Features	All	0.008	0.268

Programs

Exhaustive Search

In [28]:
```
import itertools
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn import metrics
from sklearn import svm
from sklearn.neural_network import MLPClassifier
```

In [29]:
```
training = pd.read_table(r'C:\Users\DELL\Desktop\Pattern_Recognition_Assignmen
t\Training_Data.txt')
n=training.columns
```

In [30]:
```
testing = pd.read_table(r'C:\Users\DELL\Desktop\Pattern_Recognition_Assignment
\Testing_Data.txt')
```

In [31]:
```
feature_train = np.array(training.iloc[:,1:71])
output_train = np.array(training.iloc[:,-1])
feature_test = np.array(testing.iloc[:,1:71])
output_test = np.array(testing.iloc[:,-1])
```

In [32]:
```
len(feature_train)
```

Out[32]: 120

In [33]:
```
feature_size = 2
def subsets(S,m):
    return set(itertools.combinations(S, m))
```

In [34]:
```
feature_space = subsets(range(0,70),feature_size) # generating all possible co
mbinations of features in the feature space)
feature_space = np.array(list(feature_space))
```

In [35]:
```
b=[]
for i in feature_space:
    x = feature_train[:,i].reshape((feature_train.shape[0],feature_size))

    classifier= LinearDiscriminantAnalysis(priors=[0.25,0.75])
    #classifier = svm.SVC(kernel='linear',C=1.0,random_state=0)
    #classifier=
MLPClassifier(solver='lbfgs',hidden_layer_sizes=(5,),random_s tate=0)
    #classifier = svm.SVC(kernel='rbf',C=1.0,random_state=0)

    #Learning
    classifier.fit(x,output_train)
    a=classifier.score(x, output_train)
    b.append(1-a)
```

In [36]:
```
b=np.array(b)
error_estimate=min(b)
print(error_estimate)
index = np.argmin(b)
print(index)
print(feature_space[index])
```

```
0.208333333333
606
[40 60]
```

In [37]:
```
ip_optimal = feature_train[:,feature_space[index]].reshape((feature_train.shap
e[0],feature_size))

classifier_optimal= LinearDiscriminantAnalysis(priors=[0.25,0.75])
#classifier_optimal= svm.SVC(kernel='linear',C=1.0,random_state=0)
#classifier_optimal= MLPClassifier(solver='lbfgs',hidden_layer_sizes=(5,),rand
om_state=0)
#classifier_optimal= svm.SVC(kernel='rbf',C=1.0,random_state=0)
#Learning
classifier_optimal.fit(ip_optimal,output_train)
```

Out[37]:
```
LinearDiscriminantAnalysis(n_components=None, priors=[0.25, 0.75],
              shrinkage=None, solver='svd', store_covariance=False,
              tol=0.0001)
```

In [38]:
```
x = feature_test[:,feature_space[index]].reshape((feature_test.shape[0],featur
e_size))
output_pred = classifier_optimal.predict(x)
acc= metrics.accuracy_score(output_pred,output_test)
testset_error=1-acc
print(testset_error)
```

```
0.268571428571
```

Sequential Forward Search

In [11]:
```
import itertools
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn import metrics
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
```

In [12]:
```
training = pd.read_table(r'C:\Users\DELL\Desktop\Pattern_Recognition_Assignmen
t\Training_Data.txt')
n=training.columns
```

In [13]:
```
testing = pd.read_table(r'C:\Users\DELL\Desktop\Pattern_Recognition_Assignment
\Testing_Data.txt')
```

In [14]:
```
feature_train = np.array(training.iloc[:,1:71])
output_train = np.array(training.iloc[:,-1])
feature_test = np.array(testing.iloc[:,1:71])
output_test = np.array(testing.iloc[:,-1])
```

In [15]:
```
classifier= LinearDiscriminantAnalysis(priors=[0.25,0.75])
#classifier = svm.SVC(kernel='linear',C=1.0,random_state=0)
#classifier=
MLPClassifier(solver='lbfgs',hidden_layer_sizes=(5,),random_state =0)
#classifier = svm.SVC(kernel='rbf',C=1.0,random_state=0)
```

In [16]:
```
sfs1 = SFS(classifier, k_features=3, forward=True, floating=False, verbose=2,s
coring='accuracy',cv=0)
sfs1 = sfs1.fit(feature_train, output_train)
error_estimate = 1-sfs1.k_score_
print(error_estimate)
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.
0s
[Parallel(n_jobs=1)]: Done  70 out of  70 | elapsed:    0.0s finished

[2018-05-03 13:45:39] Features: 1/3 -- score: 0.766666666667[Parallel(n_jobs=
1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  69 out of  69 | elapsed:    0.0s finished

[2018-05-03 13:45:39] Features: 2/3 -- score: 0.783333333333[Parallel(n_jobs=
1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s

0.183333333333

[Parallel(n_jobs=1)]: Done  68 out of  68 | elapsed:    0.0s finished

[2018-05-03 13:45:39] Features: 3/3 -- score: 0.816666666667
```

In [17]:

```
x = feature_train[:,sfs1.k_feature_idx_]
classifier.fit(x,output_train)
print(sfs1.k_feature_idx_)
```

```
(5, 20, 63)
```

In [18]:

```
y= feature_test[:,sfs1.k_feature_idx_]
output_pred = classifier.predict(y)
acc= metrics.accuracy_score(output_pred,output_test)
testset_error=1-acc
print(testset_error)
```

```
0.262857142857
```

All genes

In [12]:
```
import itertools
import pandas as pd
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn import metrics
from sklearn import svm
from sklearn.neural_network import MLPClassifier
```

In [13]:
```
training = pd.read_table(r'C:\Users\DELL\Desktop\Pattern_Recognition_Assignmen
t\Training_Data.txt')
n=training.columns
```

In [14]:
```
testing = pd.read_table(r'C:\Users\DELL\Desktop\Pattern_Recognition_Assignment
\Testing_Data.txt')
```

In [15]:
```
feature_train = np.array(training.iloc[:,1:71])
output_train = np.array(training.iloc[:,-1])
feature_test = np.array(testing.iloc[:,1:71])
output_test = np.array(testing.iloc[:,-1])
```

In [16]:
```
#making the instance

classifier= LinearDiscriminantAnalysis(priors=[0.25,0.75])
#classifier = svm.SVC(kernel='linear',C=1.0, random_state=0)
#classifier=
MLPClassifier(solver='lbfgs',hidden_layer_sizes=(5,),random_state =0)
#classifier = svm.SVC(kernel='rbf',C=1.0, random_state=0)

#learning
classifier.fit(feature_train, output_train)
a=classifier.score(feature_train, output_train)
error_estimate=1-a
print(error_estimate)
```

```
0.0
```

In [17]:
```
#Prediction
prediction=classifier.predict(feature_test)

#evaluation(Accuracy)
acc= metrics.accuracy_score(prediction,output_test)
testset_error=1-acc
print(testset_error)
```

```
0.365714285714
```

Conclusions

1) In LDA, it is observed that the re-substitution error on an average, irrespective of the no. of features selected is LESS that the test-set estimate of the true classification error. This implies that the resubstitution error is optimistically 'biased'. The model performs well on the training set but does not perform equally well on the test set.
   As the no. of features increase, the performance of the classifier on the training set improves gradually. However, this trend is not followed by the performance of the classifier on the test set where the test-set error estimate reaches 36% while the corresponding resubstitution error is 0, when all the features are taken into account.

2) In Linear SVM, for a few features selected, the resubstitution error and the test-set error estimate are the nearly equal. However, when all the features are considered, the test-set error estimate is much more that the resubstitution error. This implies that the resubstitution error is optimistically 'unbiased' for a lower dimensional feature set but is 'biased' for a higher dimensional feature set.

3) In non-linear SVM (with Gaussian kernel), the resubstitution error is optimistically 'unbiased' as it is nearly equal to the test- set error estimate for all combinations of features selected. The classifier performs equally on the train and the test set.

4) In Neural network, the resubstitution error is significantly less than the test set error estimate implying that it is optimistically 'biased' for all combinations of features selected.

5) All the classifiers while predicting the test set response variable gave errors between 25 to 35 percent. We can speculate that this performance can improve (reduction in error) if there are more number of samples to train on. The no. of samples should be optimal and not too many because too many samples can lead to the problem of overfitting. Tuning of hyperparameters can also help improve the models.

6) We cannot say that increase in the number of features over 70 can lead to further reduction in the test-set error estimate because the test-set error estimate has been more-or-less the same or has not significantly changed for all combinations of features selected (in all classifiers). Except in LDA, for all genes case, the test set error is highest which again supports this argument.

NOTE: All combinations of features selected means all the 5 cases of feature sets.