# Computer Assignment 2

Haresh Karnan, UIN : 625007719

May 4, 2018

## 1 Table

| Classification Rule | Feature Selection | Gene-Set Name | Resubstitution Error | Test-set Error |
|---|---|---|---|---|
| LDA | Exhaustive-search | LOC51203,Contig55377_RC | 0.20833333 | 0.29142857 |
| | SFS 3 | LOC51203,Contig51464_RC,IGFBP5.1 | 0.18333333 | 0.26285714 |
| | SFS 4 | LOC51203,Contig51464_RC,Contig55725_RC,IGFBP5.1 | 0.18333333 | 0.28 |
| | SFS 5 | LOC51203,AL137718,Contig51464_RC Contig55725_RC,IGFBP5.1 | 0.18333333 | 0.27428571 |
| | None | ALL | 0 | 0.09142857 |
| Linear SVM | Exhaustive-search | LOC51203,IGFBP5.1 | 0.20833333 | 0.25714286 |
| | SFS 3 | Contig46218_RC,LOC51203,IGFBP5.1 | 0.21666667 | 0.26285714 |
| | SFS 4 | Contig46218_RC,LOC51203,Contig51464_RC,IGFBP5.1 | 0.18333333 | 0.28571429 |
| | SFS 5 | AL080059,Contig46218_RC,LOC51203 Contig51464_RC,IGFBP5.1 | 0.18333333 | 0.26857143 |
| | None | ALL | 0.01666667 | 0.08571429 |
| NL-SVM, RBF | Exhaustive-search | Contig63649_RC,IGFBP5.1 | 0.25 | 0.26857143 |
| | SFS 3 | AL080059,Contig63649_RC,Contig46218_RC | 0.26666667 | 0.26857143 |
| | SFS 4 | AL080059,Contig63649_RC,Contig46218_RC,LOC51203 | 0.26666667 | 0.26857143 |
| | SFS 5 | AL080059,Contig63649_RC,Contig46218_RC LOC51203,AA555029_RC | 0.26666667 | 0.26857143 |
| | None | ALL | 0.26666667 | 0.26857143 |
| NN | Exhaustive-search | CENPA,MMP9 | 0.11666667 | 0.22857143 |
| | SFS 3 | Contig32125_RC,KIAA1442,DCK | 0.16666667 | 0.24 |
| | SFS 4 | Contig32125_RC,KIAA1442,ECT2,DCK | 0.175 | 0.26285714 |
| | SFS 5 | Contig32125_RC,KIAA1442,ECT2 WISP1,DCK | 0.14166667 | 0.26285714 |
| | None | ALL | 0 | 0 |

## 2 Python Code

```
from sklearn import *
from numpy import *
import time
from inspect import getargspec
from itertools import chain, combinations
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neural_network import MLPClassifier
from mlxtend.feature_selection import SequentialFeatureSelector as SFS

def totuple(a):
    try:
        return tuple(totuple(i) for i in a)
    except TypeError:
        return a

def resub_error(X,Y,clf):
```

```
    n,m = X.shape
    x = list()
    for i in range(n):
        x.append(totuple(X[i,:]))
    clf.fit(x, Y)
    err = 0.0
    for i in range(n):
        err = err + (1.0 / n) * abs(Y[i] - clf.predict([X[i,:]]))
    return err


def get_subset(X,Y,clf,searchtype,ngenes):
    if searchtype==1 and ngenes==2:
        n, m = X.shape
        maxscore = -Inf
        for i in range(0, m):
            for j in range(0, m):
                if i != j:
                    pair = concatenate((reshape(X[:,i],(n,1)),\
                    reshape(X[:,j],(n,1))),1)
                    resubstitution_error_score = 1 - resub_error(pair,Y, clf)
                    if resubstitution_error_score > maxscore:
                        maxscore = resubstitution_error_score
                        bestsubset = (i, j)
                        error_estimate = 1 - resubstitution_error_score
                    elif resubstitution_error_score == maxscore:
                        if i < bestsubset[0]:
                            maxscore = resubstitution_error_score
                            bestsubset = [i, j]
                            error_estimate = 1 - resubstitution_error_score
    elif searchtype==2:
        n, m = X.shape
        sfs1 = SFS(clf, k_features=ngenes, forward=True, floating=False)
        sfs1 = sfs1.fit(X,Y)
        # maxscore = sfs1.k_score_
        bestsubset = sfs1.k_feature_idx_
        pairlist = list()
        for i in range(0,len(bestsubset)):
            pairlist.append(reshape(X[:, bestsubset[i]], (n, 1)))
        pair = concatenate(pairlist, 1)
        error_estimate = resub_error(pair,Y,clf)
        maxscore = 1-error_estimate
    return list(bestsubset),1-maxscore

if __name__ == '__main__':
    X = loadtxt('/home/haresh/PycharmProjects/patternrecognition/ \
    data/Training_Data.txt', skiprows=1)
    Y = list(X[:,-1])
    X = X[:,1:-1]
    label = loadtxt('/home/haresh/PycharmProjects/patternrecognition/data/ \
    Training_Data.txt', dtype=basestring)
    label = label[0, 1:-1]

    # select the classification_rule here
    # 0: None
    # 1: LDA , p= 0.75
    # 2: SVM ,C = 1
    # 3:SVM with Gaussian RBF Kernel , C = 1
```

```
# 4 : NN with 5 neurons in one hidden layer
classifier_rule = 1
#################  LDA CLASSIFIER ##################
if classifier_rule==1:
    clf = LinearDiscriminantAnalysis(priors=[0.25,0.75])
#################  LINEAR SVM   ####################
elif classifier_rule==2:
    clf = svm.LinearSVC(C=1)
################# NL SVM ############################
elif classifier_rule==3:
    clf = svm.SVC(C=1,kernel='rbf')
################# NN ################################
elif classifier_rule==4:
    clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes = (5,) \
    ,activation='logistic',random_state=1)

############### GENERATE THE GENE SETS ################
# 1: Exhaustive Search :)
subset_exhaustive,error_exhaustive = get_subset(X,Y,clf,1,2)
# 2: Sequential Forward Search :)
subset_forward3,error_forward3 = get_subset(X,Y,clf,2,3)
subset_forward4, error_forward4 = get_subset(X, Y, clf, 2, 4)
subset_forward5, error_forward5 = get_subset(X, Y, clf, 2, 5)
# 3 : No Feature Selection :(
subset_everything = resub_error(X,Y,clf)
################ PRINT THE GENES AND THE ERRORS ###############
print subset_exhaustive,subset_forward3,subset_forward4,subset_forward5
print label[subset_exhaustive],label[subset_forward3],label[subset_forward4],\
label[subset_forward5]
print error_exhaustive,error_forward3,error_forward4,error_forward5 \
,subset_everything
label = loadtxt('/home/haresh/PycharmProjects/patternrecognition/data/ \
Training_Data.txt', dtype=basestring)
label = label[0, 1:-1]

################ TEST DATA ####################

XT = loadtxt('/home/haresh/PycharmProjects/patternrecognition/data \
/Testing_Data.txt', skiprows=1)
YT = list(XT[:,-1])
XT = XT[:, 1:-1]

# 1: Exhaustive Search :)
error_exhaustive_test = resub_error(XT[:,[subset_exhaustive[0]\
,subset_exhaustive[1]]],YT,clf)
# 2: Sequential Forward Search :)

error_forward3_test = resub_error(XT[:,[subset_forward3[0],subset_forward3[1]\
,subset_forward3[2]]],YT,clf)
error_forward4_test = resub_error(XT[:,[subset_forward4[0],subset_forward4[1]\
,subset_forward4[2]]],YT,clf)
error_forward5_test = resub_error(XT[:,[subset_forward5[0],subset_forward5[1]\
,subset_forward5[2]]],YT,clf)
# 3 : No Feature Selection :(
subset_everything_test = resub_error(XT, YT, clf)

print '\n'
print error_exhaustive_test,error_forward3_test,error_forward4_test,\
```

```
    error_forward5_test,subset_everything_test
%\end{lstlisting}
```

# 3 Conclusions

The NN is initialised with random states as 1.

Also, the SFS search is begun with a NULL set.

We can notice that the true classification error is greater than the resubstitution error in all the classificiaton rules.

If we have more training data than 120, we can do better.

Based on true error estimates, Neural Network classifier is better than the rest.

Neural Network error estimates are better than the rest if we dont use feature selection.