ECEN 649 Pattern Recognition - Spring 2017
Computer Project 2

Geethik Narayana Kamineni
UIN - 825008990

Introduction

The project involved determining the best gene feature sets using techniques of error estimation, feature selection and classifier design which best discriminate the two prognosis classes in the breast tumor biopsies data from the following cancer study:
van de Vijver,M.J., He,Y.D., van't Veer,L.J., et al. (2002), "A gene-expression signature as a predictor of survival in breast cancer." $New\ Eng.\ J.\ Med.$, 347, 1999-2009.

Procedure

The process to obtain the best gene features consisted of implementing the two classifier techniques $DLDA$ and $3NN$ along with the feature selection methods of $Exhaustive\ Search$ and $Sequential\ Forward\ Search$ on the data considering the determining error estimate to be the resubstitution error.

DLDA

Diagonal linear discriminant analysis ($DLDA$) is a special case of the LDA wherein the sample covariance matrix is constrained to be only along the diagonal. This is implemented in $Matlab$ using the $fitcdiscr$ function using the following convention:

$$mdl = fitcdiscr\left(trainX, trainY, 'DiscrimType', 'diaglinear'\right)$$

3NN

$k-$Nearest Neighbor rule (KNN) where in $k = 3$ is implemented as one another strategy in finding the best gene sets. Its implementation in $Matlab$ is through the function $fitcknn$ as follows:

$$mdl = fitcknn\left(trainX, trainY, 'NumNeighbors', 3\right)$$

Error criterion

The resubstitution error which is the apparent error of the designed classifier on the training data is given by:

$$\varepsilon_n^r = \frac{1}{n}\sum_{i=0}^{n} \mid Y_i - \psi\left(X_i\right) \mid$$

is implemented in $Matlab$ on the predictions made by the $predict$ function as follows:

$$label_{pred} = predict\left(mdl, trainX\right)$$
$$\varepsilon_n^r = {}^{sum\left(abs\left(label_{pred} - trainY\right)\right)}/_{size(trainY,1)}$$

The test set error estimate on the test data $\left(x_i^{(m)}, y_i^{(m)}\right)$ which is given by:

$$\varepsilon_{n,m} = \frac{1}{m}\sum_{i=0}^{m} \mid Y_i^{(m)} - \psi\left(X_i^{(m)}\right) \mid$$

is even implemented in a similar fashion to the resubstitution error but instead on the test data.

Feature Selection

For the case of determining the feature sets with size from 1 to 3 using $Exhaustive\ Search$, all the combinations are being generated using the $combnk$ combinatorial function in $Matlab$ and looped over each time to find the best discriminating gene set for each classifier.

In the case of $Sequential\ Forward\ Search$ to determine the top eight features among the 70-gene sets $sequentialfs$ function in $Matlab$ is being used whose implementation is as follows:

$$[fs, history] = sequentialfs(criterion, trainX, trainY, 'cv', 'resubstitution', 'nfeatures', 8);$$

where the fs stores the logical vector of the gene sets obtained after the final iteration whereas $history$ holds the logical vectors in each iteration and also the criterion values of each iteration. The criterion function is being based on the $DLDA$ and $3NN$ classifiers. The code for the analysis implementation is as follows:

```
% Computer Project 2 Code
%Get data

```

```
trainObj = importdata("train.txt",'\t');
train_x = trainObj.data;
train_y = train_x(:,72);
train_x = train_x(:,1:71);
trainHeader = trainObj.colheaders;

testObj = importdata("test.txt",'\t');
test_x = testObj.data;
test_y = test_x(:,72);
test_x = test_x(:,1:71);
testHeader = testObj.colheaders;

%Exhaustive search
fprintf("**************Exhaustive for size-1***********\n");

%size-1 feature set
%3NN
for i=2:1:size(train_x,2)
    new_train = [train_x(:,i)];
    mdl_knn = fitcknn(new_train,train_y,'NumNeighbors',3);
    new_train_label = mdl_knn.predict(new_train);

    if i>2
        if res_error > sum(abs(new_train_label - train_y))/size(train_x,1)
            res_error = sum(abs(new_train_label - train_y))/size(train_x,1);
            best_knn_1 = i;
        end
    else
        res_error = sum(abs(new_train_label - train_y))/size(train_x,1);
        best_knn_1 = 2;
    end
end

best1 = char(trainHeader(best_knn_1));
fprintf('Best set of size-1 for 3NN is {%s} and the resubsitution error is %f\n',best1,res_error)

%DLDA
for i=2:1:size(train_x,2)
    new_train = [train_x(:,i)];
    mdl_dlda = fitcdiscr(new_train,train_y,'DiscrimType','diaglinear');
    new_train_label = predict(mdl_dlda, new_train);

    if i>2
        if res_error_dlda > sum(abs(new_train_label - train_y))/size(train_x,1)
            res_error_dlda = sum(abs(new_train_label - train_y))/size(train_x,1);
            best_dlda_1 = i;
        end
    else
        res_error_dlda = sum(abs(new_train_label - train_y))/size(train_x,1);
        best_dlda_1 = 2;
    end
end

best1_dlda = char(trainHeader(best_dlda_1));
fprintf('Best set of size-1 for DLDA is {%s} and the resubsitution error is %f\n',best1_dlda,res_error_dlda)

%Test Set error calculation
best_train_knn = [train_x(:,best_knn_1)];
best_mdl_knn = fitcknn(best_train_knn, train_y,'NumNeighbors',3);
best_test_knn = [test_x(:,best_knn_1)];
test_label_knn = best_mdl_knn.predict(best_test_knn);
test_error_knn = sum(abs(test_label_knn - test_y))/size(test_x,1);
fprintf('Test set error for 3NN on best set {%s} of size-1 is %f\n',best1,test_error_knn);

best_train_dlda = [train_x(:,best_dlda_1)];
best_mdl_dlda = fitcdiscr(best_train_dlda, train_y,'DiscrimType','diaglinear');
best_test_dlda = [test_x(:,best_dlda_1)];
test_label_dlda = predict(best_mdl_dlda, best_test_dlda);
test_error_dlda = sum(abs(test_label_dlda - test_y))/size(test_x,1);
fprintf('Test set error for DLDA on best set {%s} of size-1 is %f\n',best1_dlda,test_error_dlda);

fprintf("****************************************\n");
fprintf("**************Exhaustive for size-2***********\n");

%-------------------------------------------------------------------------%
%size-2 feature set
C = combnk(2:71,2);

%3NN
for i=1:1:size(C,1)
```

```
    new_train = [train_x(:,C(i,:))];
    mdl_knn = fitcknn(new_train,train_y,'NumNeighbors',3);
    new_train_label = mdl_knn.predict(new_train);

    if i>1
        if res_error > sum(abs(new_train_label - train_y))/size(train_x,1)
            res_error = sum(abs(new_train_label - train_y))/size(train_x,1);
            best_knn_2 = i;
        end
    else
        res_error = sum(abs(new_train_label - train_y))/size(train_x,1);
        best_knn_2 = 1;
    end
end

best1 = char(trainHeader(C(best_knn_2,1)));
best2 = char(trainHeader(C(best_knn_2,2)));
fprintf('Best set of size-2 for 3NN is {%s, %s} and the resubstitution error is %f\n',best1,best2,res_error)

%DLDA
for i=1:1:size(C,1)
    new_train = [train_x(:,C(i,:))];
    mdl_dlda = fitcdiscr(new_train,train_y,'DiscrimType','diagLinear');
    new_train_label = predict(mdl_dlda, new_train);

    if i>1
        if res_error_dlda > sum(abs(new_train_label - train_y))/size(train_x,1)
            res_error_dlda = sum(abs(new_train_label - train_y))/size(train_x,1);
            best_dlda_2 = i;
        end
    else
        res_error_dlda = sum(abs(new_train_label - train_y))/size(train_x,1);
        best_dlda_2 = 1;
    end
end

best1_dlda = char(trainHeader(C(best_dlda_2,1)));
best2_dlda = char(trainHeader(C(best_dlda_2,2)));
fprintf('Best set of size-2 for DLDA is {%s, %s} and the resubsitution error is %f\n',best1_dlda,best2_dlda,res_error_dlda)

%Test Set error calculation
best_train_knn = [train_x(:,C(best_knn_2,:))];
best_mdl_knn = fitcknn(best_train_knn, train_y,'NumNeighbors',3);
best_test_knn = [test_x(:,C(best_knn_2,:))];
test_label_knn = best_mdl_knn.predict(best_test_knn);
test_error_knn = sum(abs(test_label_knn - test_y))/size(test_x,1);
fprintf('Test set error for 3NN on best set {%s, %s} of size-2 is %f\n',best1,best2,test_error_knn);

best_train_dlda = [train_x(:,C(best_dlda_2,:))];
best_mdl_dlda = fitcdiscr(best_train_dlda, train_y,'DiscrimType','diaglinear');
best_test_dlda = [test_x(:,C(best_dlda_2,:))];
test_label_dlda = predict(best_mdl_dlda, best_test_dlda);
test_error_dlda = sum(abs(test_label_dlda - test_y))/size(test_x,1);
fprintf('Test set error for DLDA on best set {%s, %s} of size-2 is %f\n',best1_dlda,best2_dlda,test_error_dlda);

fprintf("*****************************************\n");
fprintf("**************Exhaustive for size-3***********\n");

%-------------------------------------------------------------------------%
%size-3 feature set
C = combnk(2:71,3);

%3NN
for i=1:1:size(C,1)
    new_train = [train_x(:,C(i,:))];
    mdl_knn = fitcknn(new_train,train_y,'NumNeighbors',3);
    new_train_label = mdl_knn.predict(new_train);

    if i>1
        if res_error > sum(abs(new_train_label - train_y))/size(train_x,1)
            res_error = sum(abs(new_train_label - train_y))/size(train_x,1);
            best_knn_3 = i;
        end
    else
        res_error = sum(abs(new_train_label - train_y))/size(train_x,1);
        best_knn_3 = 1;
    end
end

best1 = char(trainHeader(C(best_knn_3,1)));
```

```
best2 = char(trainHeader(C(best_knn_3,2)));
best3 = char(trainHeader(C(best_knn_3,3)));
fprintf('Best set of size-3 for 3NN is {%s, %s, %s} and the resubstitution error is %f\n',best1,best2,best3,res_error)

%DLDA
for i=1:1:size(C,1)
    new_train = [train_x(:,C(i,:))];
    mdl_dlda = fitcdiscr(new_train,train_y,'DiscrimType','diaglinear');
    new_train_label = predict(mdl_dlda, new_train);

    if i>1
        if res_error_dlda > sum(abs(new_train_label - train_y))/size(train_x,1)
            res_error_dlda = sum(abs(new_train_label - train_y))/size(train_x,1);
            best_dlda_3 = i;
        end
    else
        res_error_dlda = sum(abs(new_train_label - train_y))/size(train_x,1);
        best_dlda_3 = 1;
    end
end

best1_dlda = char(trainHeader(C(best_dlda_3,1)));
best2_dlda = char(trainHeader(C(best_dlda_3,2)));
best3_dlda = char(trainHeader(C(best_dlda_3,3)));
fprintf('Best set of size-3 for DLDA is {%s, %s, %s} and the resubsitution error is %f\n',best1_dlda,best2_dlda,best3_dlda,
   res_error_dlda)

%Test Set error calculation
best_train_knn = [train_x(:,C(best_knn_3,:))];
best_mdl_knn = fitcknn(best_train_knn, train_y,'NumNeighbors',3);
best_test_knn = [test_x(:,C(best_knn_3,:))];
test_label_knn = best_mdl_knn.predict(best_test_knn);
test_error_knn = sum(abs(test_label_knn - test_y))/size(test_x,1);
fprintf('Test set error for 3NN on best set {%s, %s, %s} of size-3 is %f\n',best1,best2,best3,test_error_knn);

best_train_dlda = [train_x(:,C(best_dlda_3,:))];
best_mdl_dlda = fitcdiscr(best_train_dlda, train_y,'DiscrimType','diaglinear');
best_test_dlda = [test_x(:,C(best_dlda_3,:))];
test_label_dlda = predict(best_mdl_dlda, best_test_dlda);
test_error_dlda = sum(abs(test_label_dlda - test_y))/size(test_x,1);
fprintf('Test set error for DLDA on best set {%s, %s, %s} of size-3 is %f\n',best1_dlda,best2_dlda,best3_dlda,test_error_dlda);

fprintf("****************************************\n");

%------------------------------------------------------------------------%
%Sequential forward Search
opts = statset('display','iter');

fprintf("*************Iteration for 3NN***********\n");
[fs,history] = sequentialfs(@my_crit_knn,train_x,train_y,'cv','resubstitution','nfeatures',8,'options',opts,'direction','
   forward');

fprintf("****************************************\n"); fprintf("*************Iteration for DLDA***********\n");
[fs1,history1] = sequentialfs(@my_crit_dlda,train_x,train_y,'cv','resubstitution','nfeatures',8,'options',opts,'direction','
   forward');

fprintf("****************************************\n");

%Test set estimate
%3NN
for i=1:1:size(history.In,1)
    new_train = [train_x(:,history.In(i,:)==1)];
    fprintf("{")
    features_selected = trainHeader(history.In(i,:)==1);
    for j=1:1:i-1
        fprintf("%s, ",char(features_selected(j)));
    end
    fprintf("%s} and the resub-error is %f\n",char(features_selected(i)),history.Crit(i));
    mdl_knn_fs  = fitcknn(new_train,train_y,'NumNeighbors',3);

    test_knn_fs = [test_x(:,history.In(i,:)==1)];
    test_error_knn_fs = sum(abs(mdl_knn_fs.predict(test_knn_fs)- test_y))/size(test_x,1);
    fprintf("Test error of KNN with set size-%d is %f\n\n",i,test_error_knn_fs);
end

fprintf("****************************************\n");

%Test Set estimate
%DLDA
for i=1:1:size(history1.In,1)
```

```
    new_train = [train_x(:,history1.In(i,:)==1)];
    fprintf("{")
    features_selected_dlda = trainHeader(history1.In(i,:)==1);
    for j=1:1:i-1
        fprintf("%s, ",char(features_selected_dlda(j)));
    end
    fprintf("%s} and the resub-error is %f\n",char(features_selected_dlda(i)),history1.Crit(i));
    mdl_dlda_fs  = fitcdiscr(new_train,train_y,'DiscrimType','diaglinear');

    test_dlda_fs = [test_x(:,history1.In(i,:)==1)];
    test_error_dlda_fs = sum(abs(predict(mdl_dlda_fs,test_dlda_fs)- test_y))/size(test_x,1);
    fprintf("Test error of DLDA with set size-%d is %f\n\n",i,test_error_dlda_fs);
end

function val_knn = my_crit_knn(xT,yT,xt,yt)
    mdl_knn = fitcknn(xT,yT,'NumNeighbors',3);
    val_knn = sum(abs(predict(mdl_knn,xt) - yt)) ;
end

function val_dlda = my_crit_dlda(xT,yT,xt,yt)
    mdl_dlda = fitcdiscr(xT,yT,'DiscrimType','diaglinear');
    val_dlda = sum(abs(predict(mdl_dlda,xt) - yt)) ;
end
```

The results obtained from the program above when tabulated are as follows for $3NN$

$3NN$ **gene sets**	**Resubstitution error**	**True error**
{CENPA}	0.100000	0.225532
{G5, CENPA}	0.033333	0.212766
{G1, FGF18, ORC6L}	0.016667	0.191489
{CENPA}	0.100000	0.225532
{G5, CENPA}	0.033333	0.212766
{G5, PECI.1, CENPA}	0.016667	0.221277
{G5, G15, PECI.1, CENPA}	0.016667	0.246809
{G5, G15, COL4A2, PECI.1, CENPA}	0.016667	0.238298
{G5, G8, G15, COL4A2, PECI.1, CENPA}	0.016667	0.238298
{G5, G8, DC13, G15, COL4A2, PECI.1, CENPA}	0.016667	0.221277
{G4, G5, G8, DC13, G15, COL4A2, PECI.1, CENPA}	0.033333	0.204255

The results for the $DLDA$ obtained are:

$DLDA$ **gene sets**	**Resubstitution error**	**True error**
{ORC6L}	0.133333	0.170213
{G3, G5}	0.100000	0.204255
{MMP9, IGFBP5.1, CENPA}	0.050000	0.195745
{ORC6L}	0.133333	0.170213
{G4, ORC6L}	0.116667	0.170213

{G4, FLT1, ORC6L}	0.116667	0.178723
{G4, ALDH4, FLT1, ORC6L}	0.116667	0.174468
{G4, ALDH4, KIAA1442, FLT1, ORC6L}	0.116667	0.148936
{G4, ALDH4, KIAA1442, FLT1, ORC6L, IGFBP5.1}	0.100000	0.136170
{G4, ALDH4, KIAA1442, FLT1, MMP9, ORC6L, IGFBP5.1}	0.083333	0.127660
{G2, G4, ALDH4, KIAA1442, FLT1, MMP9, ORC6L, IGFBP5.1}	0.083333	0.127660

From the results we can infer that,

- the resubstitution error produced in the case of gene sets obtained from the exhaustive search are less than those obtained from the sequential forward search. Also, we can observe that the apparent error converges to zero early in exhaustive search method when compared to sequential forward search. However, in terms of computational efficiency sequential feature selection is more preferred due to its practical applicability as the maximum number of computations done is $O\left(n^2\right)$ whereas its counterpart takes $O\left(n!\right)$ time.

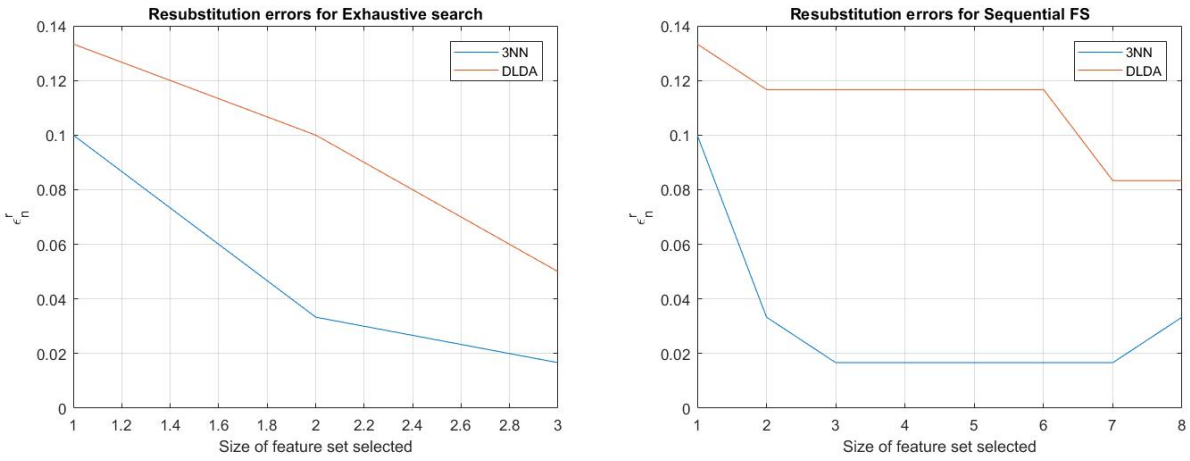

Figure 1: Plot of ε_n^r for Exhaustive search for 3NN and DLDA

Figure 2: Plot of ε_n^r for Sequential forward search for 3NN and DLDA

- From the plot in $Figure - 1$, above, we can also observe the training error is monotonically decreasing in the exhaustive case while this is not represented in the case of sequential feature selection, this can be attributed due to the fact that since once a feature is selected, it gets

frozen even though a better combination exists. However, this monotonic behavior is not observed for the true error which in the case of exhaustive search, which might imply that the exhaustive search was over-fitting the model using the training data.

- From both the figures above, the resubstitution error in the case of $3NN$ are less than those obtained from $DLDA$, whereas the test set estimates are lower for $DLDA$. This can explained from the fact that $3NN$ might be over-fitting a complex model which $DLDA$ fails to do irrespective the data. So, we can conclude that, $DLDA$ appears to be a better classifier in this case of the data compared to $3NN$.

- In order to build better models, we can implement strategies like Plus-l take-r search which might perform well as there is no freezing of a feature as in the case of sequential search. Also implementing a complex classification algorithm like neural networks and support vector machines we can perform well along with increasing the number of samples considered in the training set in designing the classifier.