MATH 444 **Due Date: 03/27/2022**

Assignment 4

Ellis Wright (enw30)

1 Introduciton

This assignment will demonstrate a variety of classifiers based on previous algorithms we have learned in this class to predict abnormal electrocardiogram (ECG) readings. The algorithms investigated below are a distance classifier, K-Nearest-Neighbor (KNN) classifier, Learning Vector Quantifier (LVQ), and Linear Discriminant Analysis (LDA). Each ECG reading has been embedded into $\mathbb{R}^{96}$

2 Classifiers

2.1 Distance and KNN

A standard distance classifier takes an unknown reading and finds which class the closest data point in a training set it belongs to and chooses that as its assignment. The KNN algorithm extends a distance classifier by finding the K nearest points and assigning the class that is the majority of the K points.

2.2 LVQ

The LVQ algorithm trains a Self-Organizing Map (SOM) on each class of the training data to learn the structure. Then you can use a distance classifier with new points and the learned structure (the set of all prototypes).

3 Experimental Results

A distance classifier is a special case of KNN where k = 1. So, I ran KNN with values of k = 1,3,5,7. I was curious how the KNN algorithm would run on the protoypes of LVQ so I included that data aswell. Similarly, it is the same LVQ algorithm when k=1.The confusion matrices used to describe results follow the pattern below.

		Actual	
		1	0
Predicted	1	True Positive	False Positive
	0	False Negative	True Negative

1 = abnormal ECG
0 = normal ECG

k = 1 KNN		Actual 1	0
Predicted	1	29	5
	0	7	59

Accuracy = 88.0%
F1-Score = 82.9%

k = 3 KNN		Actual 1	0
Predicted	1	29	3
	0	7	61

Accuracy = 90.0%
F1-Score = 85.3%

k = 5 KNN		Actual 1	0
Predicted	1	29	3
	0	7	61

Accuracy = 90.0%
F1-Score = 85.3%

k = 7 KNN		Actual 1	0
Predicted	1	28	3
	0	8	61

Accuracy = 89.0%
F1-Score = 83.6%

Now for the same set of results but for the LVQ map.

k = 1 LVQ		Actual 1	0
Predicted	1	31	5
	0	5	59

Accuracy = 90.0%
F1-Score = 86.1%

k = 3 LVQ		Actual 1	0
Predicted	1	31	6
	0	5	58

Accuracy = 89.0%
F1-Score = 84.9%

k = 5 LVQ		Actual 1	0
Predicted	1	31	15
	0	5	49

Accuracy = 80.0%
F1-Score = 75.6%

k = 7 LVQ		Actual 1	0
Predicted	1	29	16
	0	7	48

Accuracy = 77.0%
F1-Score = 71.6%

It seems like k increased the classification got worse. Here are the images of the prototypes. Five prototypes were used for each class(the top row corresponds to normal readings and the bottom row are abnormal readings):

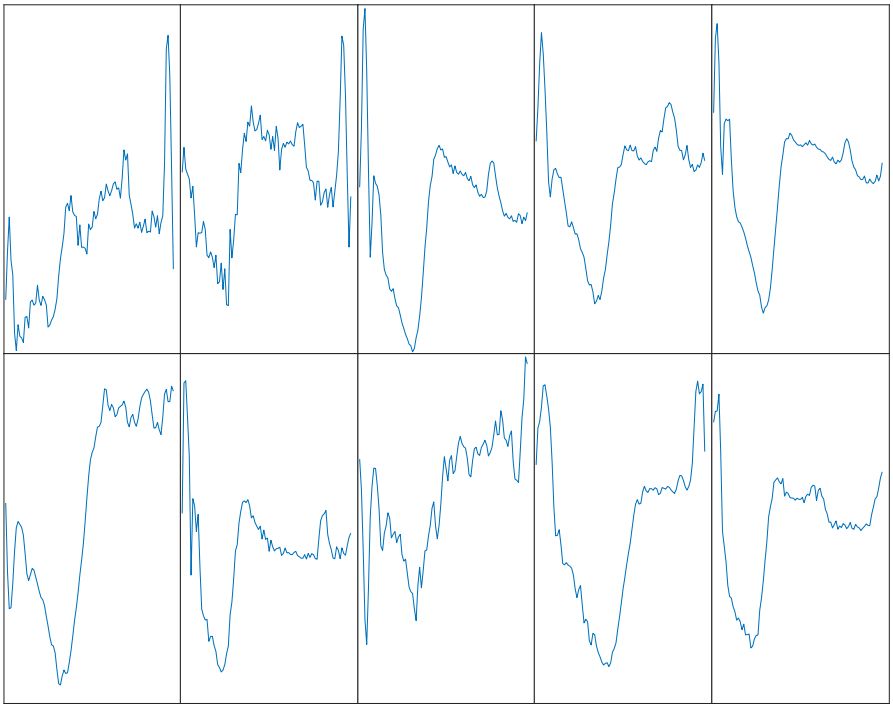

Figure 1: LVQ Prototypes

Finally, I ran the LDA algorithm to find a direction to project the data on for easy classification. Although it looks like the test set was harder to differentiate. Figure 2 shows how well it was able to separate the training set even though there is a lot of overlap in the test set.

As we can see, not all algorithms return the same results– some are better than others depending on how well each algorithm represents the space. The best algorithm was the standard LVQ algorithm

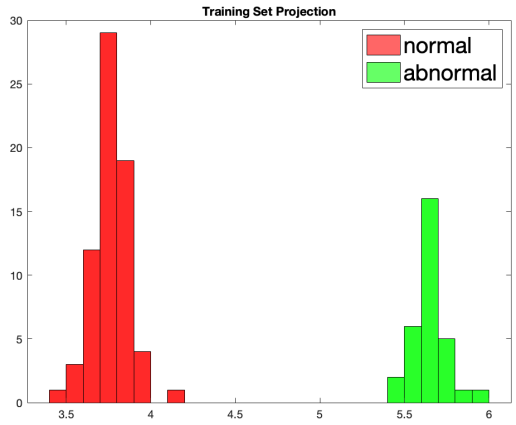

Figure 2: Training Set Projection

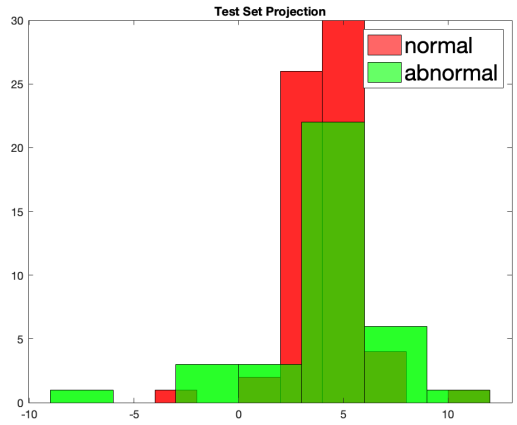

Figure 3: Test Set Projection

with 90% accuracy and an F1 score of 86.1%. There was a second place tie for KNN with k = 3 and 5. Even though the algorithms were not trained and tested on sets with large class imbalances, I thought it would be necessary to include the F1 score. the F1 score is necessary because the impact of missing an abnormal ECG may outweigh the impact of accidentally treating a normal ECG. The vast majority of the time, people have a normal ECG even though the testing data did not reflect this.

4 Code

I used the same code for LDA and SOM as well as the necessary helper functions. Attached is all new code.

```
function class = KNN(data,I,k,unknown)

    %get k indices of smallest distances
    [~,indices] = mink(vecnorm(data - unknown),k);

    %get the classes of the k-nearest neighbors
    neighbor_classes = I(indices);

    %majority vote wins
    class =  mode(neighbor_classes);
end

function [M,Is] = LVQ(data,I,k)

    %get number of classes
```

```
    num_classes = length(unique(I));

    %initizalize set of all prototypes
    M = rand(length(data(:,1)),k*num_classes);
    Is = zeros(1,k*num_classes);

    %for each class generate prototypes
    for i = 1:num_classes

        subs = data(:,I == i-1);
        M(:,(i-1)*k+1:i*k) = subs(:,randsample(length(subs(1,:)),k));
        Is((i-1)*k+1:i*k) = i-1;
    end


    %move each prototype
    for t = 1:1000
        p = randi(length(I));
        a = .9*exp(-t*log(10)/1000);
        point = data(:,p);
        [~,index] = min(vecnorm(M-point));
        if(Is(index) == I(p))
           M(:,index) = M(:,index) + a*(point-M(:,index));
        else
            M(:,index) = M(:,index) - a*(point-M(:,index));
        end

    end

end

function mat = create_conf_mat(predicted,actual)
    TP = 0;
    FP = 0;
    TN = 0;
    FN = 0;
    for i = 1:length(predicted)
        if predicted(i) == 0 && actual(i) == 0
            TN = TN + 1;
        end
        if predicted(i) == 0 && actual(i) == 1
            FN = FN + 1;
        end
        if predicted(i) == 1 && actual(i) == 0
            FP = FP + 1;
```

```
        end
        if predicted(i) == 1 && actual(i) == 1
            TP = TP + 1;
        end
        mat = [TP,FP;FN,TN];
    end
```

And here is the code that generated the images and numbers.

```
%load and format parameters
load("ECG_test.mat")
load("ECG_train.mat")
X_train_abnormal = X_train_abnormal';
X_train_normal = X_train_normal';
X_test_abnormal = X_test_abnormal';
X_test_normal = X_test_normal';
X_train = [X_train_normal,X_train_abnormal];
train_labels = [ones(1,length(X_train_normal(1,:)))-1,ones(1,length(X_train_abnorma
test_labels = [ones(1,length(X_test_normal(1,:)))-1,ones(1,length(X_test_abnormal(
X_test = [X_test_normal,X_test_abnormal];
knn_predictions = zeros(100,4);
lda_predictions = zeros(1,100);
knn_lvq_predictions = zeros(100,4);
t = length(X_test(1,:));

%getting lda direction
lda_direction = LDA(X_train,train_labels);

%getting prototypes
[lvq_prototypes,prototype_labels] = LVQ(X_train,train_labels,5);

%loop through each test point
for i = 1:t

    point = X_test(:,i);

    for k = [1,3,5,7]
        knn_predictions(i,(k+1)/2) = KNN(X_train,train_labels,k,point);
        knn_lvq_predictions(i,(k+1)/2) = KNN(lvq_prototypes,prototype_labels,k,poi
    end
end

knn_conf_1 = create_conf_mat(knn_predictions(:,1),test_labels);
knn_conf_3 = create_conf_mat(knn_predictions(:,2),test_labels);
knn_conf_5 = create_conf_mat(knn_predictions(:,3),test_labels);
knn_conf_7 = create_conf_mat(knn_predictions(:,4),test_labels);
```

```
lvq_conf_1 = create_conf_mat(knn_lvq_predictions(:,1),test_labels);
lvq_conf_3 = create_conf_mat(knn_lvq_predictions(:,2),test_labels);
lvq_conf_5 = create_conf_mat(knn_lvq_predictions(:,3),test_labels);
lvq_conf_7 = create_conf_mat(knn_lvq_predictions(:,4),test_labels);

figure(1)
histogram(lda_direction'*X_train_normal,'FaceColor','r')
title 'Training Set Projection'
hold on
histogram(lda_direction'*X_train_abnormal,'FaceColor','g')
legend('normal','abnormal','FontSize',20)

figure(2)
histogram(lda_direction'*X_test_normal,'FaceColor','r')
title 'Test Set Projection'
hold on
histogram(lda_direction'*X_test_abnormal,'FaceColor','g')
legend('normal','abnormal','FontSize',20)

figure(3)
tiledlayout(2,5,'TileSpacing','none')
for i = 1:10
    nexttile
    plot(lvq_prototypes(:,i))
    set(gca,'XTick',[], 'YTick', [])
end
```