Lab 5

Albert Segarra Roca (S3255050), Carlos Humberto Paz Rodríguez (S3040577)

Group 16

Pattern Recognition

FMNS • RUG

November 23, 2017

1 Assignment 1

1.1 Exercise 1

As seen on the code below (Listing 1), we load and plot the two sample data we were provided, we plot feature 2 vs feature 1, and we can see in Figure 1 that the class A is split in two clusters on the left and on the right of the class B, this means that we have to use at least two prototypes for class A and one for class B. We can also see that the distribution of data points is almost equally distributed among the two features, therefore we can use Euclidean Distance to calculate the distance measure.

Listing 1: False acceptance, hit point

```
function LVQ1ex1()
    %Load data class_a and class_b
    load data_lvq_A.mat
    load data_lvq_B.mat

    %Plot feature2 vs feature1 of both classes
    scatter(matA(:,1),matA(:,2),20);
    title('Class distribution scatter plot')
    xlabel('Feature 1')
    ylabel('Feature 2')
    hold on;

    scatter(matB(:,1),matB(:,2),20);
    legend('Class A','Class B');
    hold off;
end
```

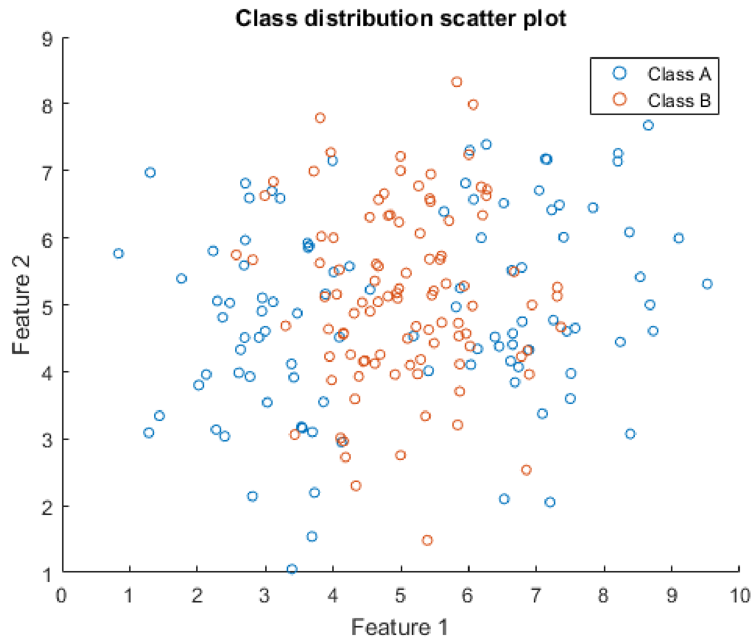

Figure 1: 2 dimension feature map of two different classes

1.2 Exercise 2

For class A we are goint to use two prototypes, since the data in class A is splitted in two clusters, one easy way to choose the starting prototypes is to choose two random points of class A, the first one with value lower than 4 and the second with value bigger than 6 on feature 1; for feature 2, we use a random initial value between the whole space of values (0 10). For class B we use a random value between 0 and 10 for both features.

The complete implementation of this algorithm is included on appendix 4.1, for the training phase we used a $\eta = 0.01$. We also introduced two control thresholds for the training of the algorithm, one for the maximum error rate we would like to have (errorThreshold=0.25) and the second to stop the training when the error rate converges after n epochs (variationThreshold=0.0001).

The code on appendix 4.1 also includes an animation to view the evolution of the prototypes after each epoch.

At the end of the traininig we obtain the following prototypes positions (Figure 2). We could observe through different η values, that if they are high values, sometimes the training converges after few epochs, but sometimes it never converges, furthermore the prototypes position is not too precise; when we use a small η value, the training takes more epochs to converge, but the result is more precise, the cost of this small value is the computational resources needed to compute the training prototypes.

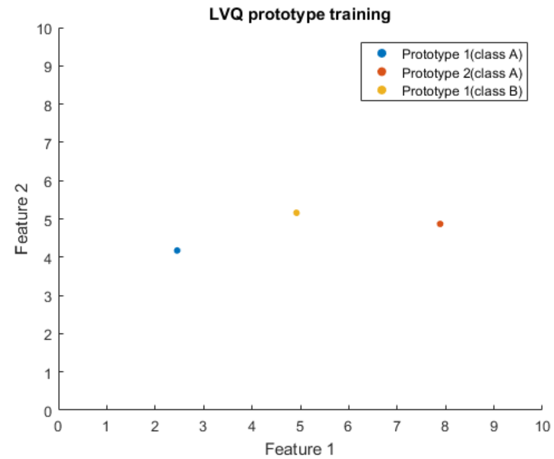

Figure 2: LVQ trained prototypes (2-1)

We also plotted the missclassification training error rate (Figure 3) to visualize how the training evolves after each epoch, due to the small training data, the error rate typically stays on 25%, that is why we added the errorThreshold control variable to this amount, and also because if we only used the variationThreshold between epochs, there were some scenarios where the two error rates were around 50% but since there was no variation in respect to the previous epoch value, the training stopped after few epochs and the prototypes were not fully trained.

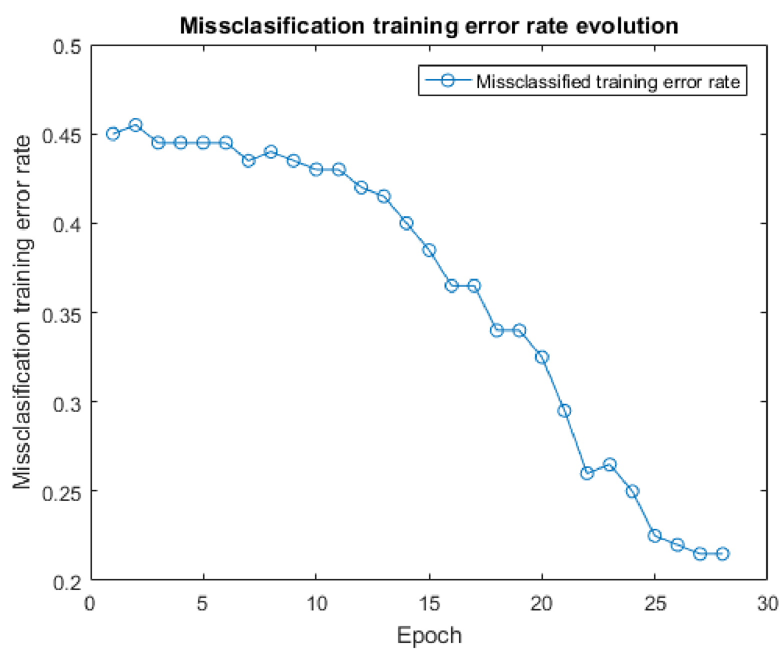

Figure 3: Missclassification error rate during training epochs

1.3 Exercise 3 a)

In this exercise we use one prototype for class A and one for class B, we had to adjust the errorThreshold control variable because the missclassification training error rate, is typically above 0.47, that is because, as we analized on figure 1, the class A is divided in two different clusters, therefore the center of mass of the cluster is similar to the center of mass of the class B cluster, which leads to classification error, and basically we have 0.47 probability of assigning the correct class to the incoming points.

This solution is not optimal because we cannot classify correctly the classes, we have to use at least two prototypes for class A.

These are the resulting prototypes after the training (Figure 4) and missclassification error through epochs (Figure 5).

We observe that when using (1-1) prototypes for class A prototype, it depends on the initial position of the prototype, and the prototype can never cross to the other cluster because class B is pushing it away, so it always stays on the cluster where it began, on the case of class B prototype, the prototype is also affected with this seleccion because when a new point from class A, and from the left cluster is presented, if the closest prototype is the prototype of class B, it gets pushed away, that is why is out of the center of mass of its own class data.

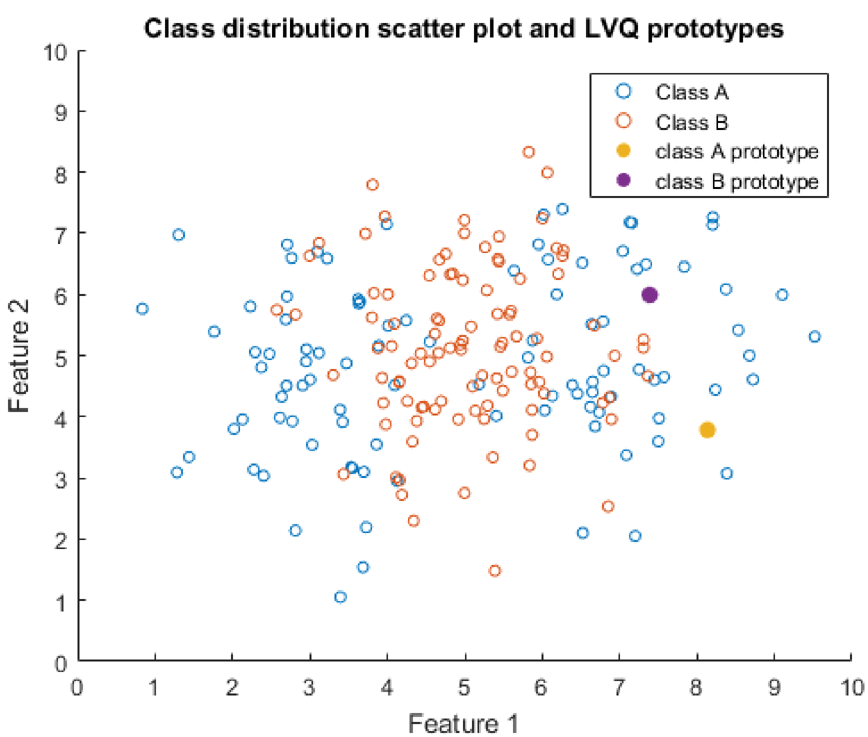

Figure 4: LVQ trained prototypes (1-1) with $\eta = 0.01$

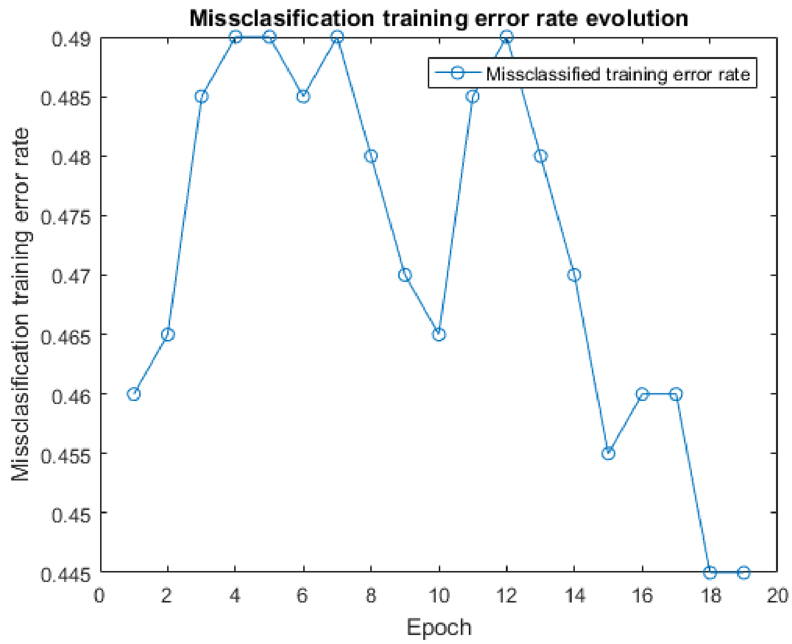

Figure 5: Missclasification error rate during training epochs with $\eta = 0.01$ (1-1)

As we see on figure 5 the missclassification error is too high, and the only way to stop the training is by accepting a big error rate and just waiting for two similar results to converge, but clearly one can see from this plot that along the epochs the missclassification training error rate does not seem to converge.

1.4 Exercise 3 b)

If we take one prototype from class A and two prototypes of class B, we can see that the behaviour is similar to the one in the previous exercise, the prototype of class A is pushed away of the class B prototypes (figure 6), causing that even after 20 epoches, the missclassification error is still high (figure 7), close to 0.5, the main difference now is that with more prototypes of the wrong class, you give more importance to this class and two prototypes push further away the protype of class A.

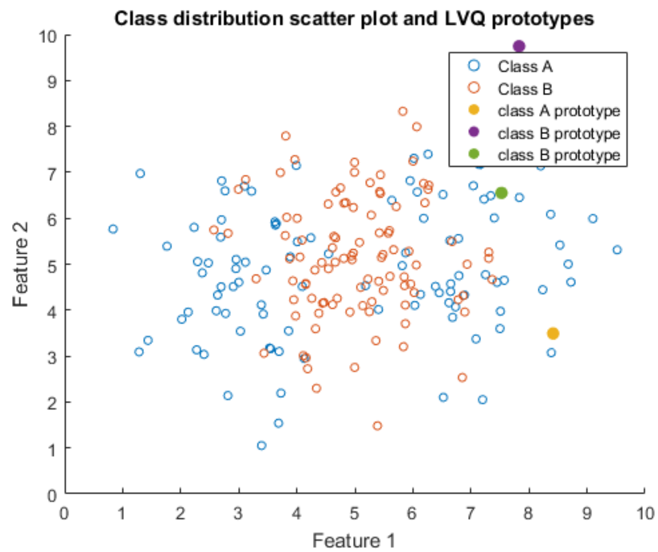

Figure 6: LVQ trained prototypes (1-2) with $\eta = 0.01$

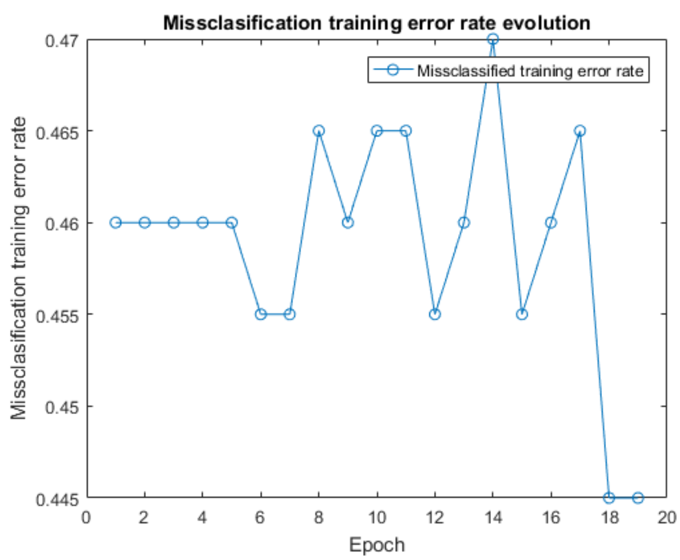

Figure 7: Missclasification error rate during training epochs with $\eta = 0.01$ (1-2)

We still see no traces of convergence (Figure 7).

1.5 Exercise 3 c)

For this exercise we use two class A prototypes and one class B prototype (Figure 8), we are able to see that finally the system is able to converge (Figure 9), it even converges with a smaller missclassification error rate (clse to 0.3). We can conclude that this is a working solution for the sample data.

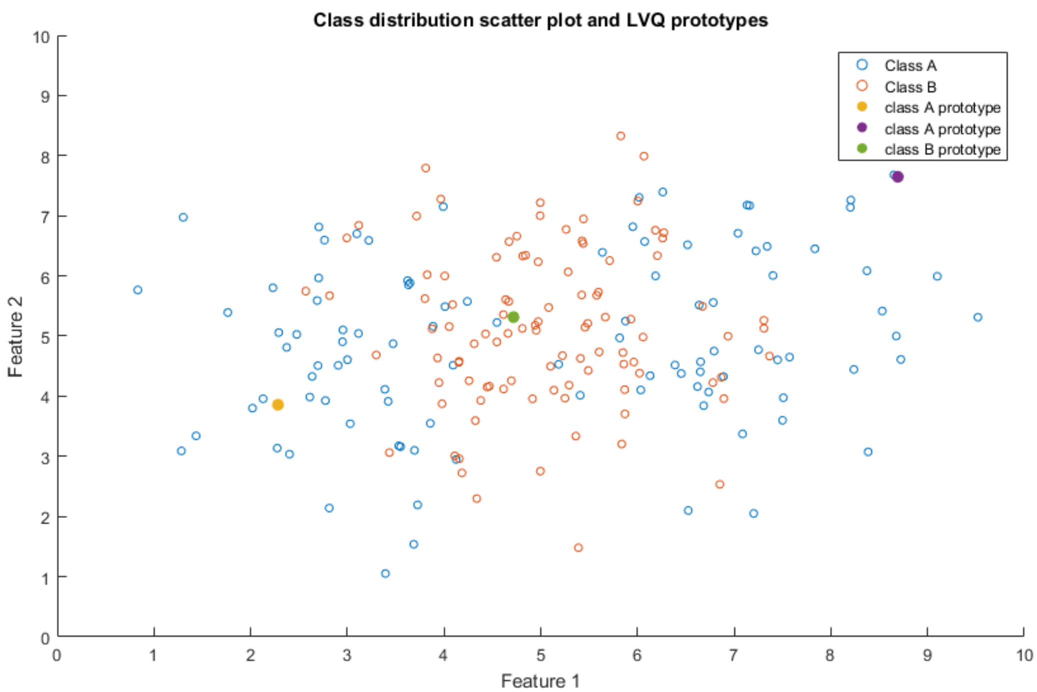

Figure 8: LVQ trained prototypes (2-1) with $\eta = 0.01$

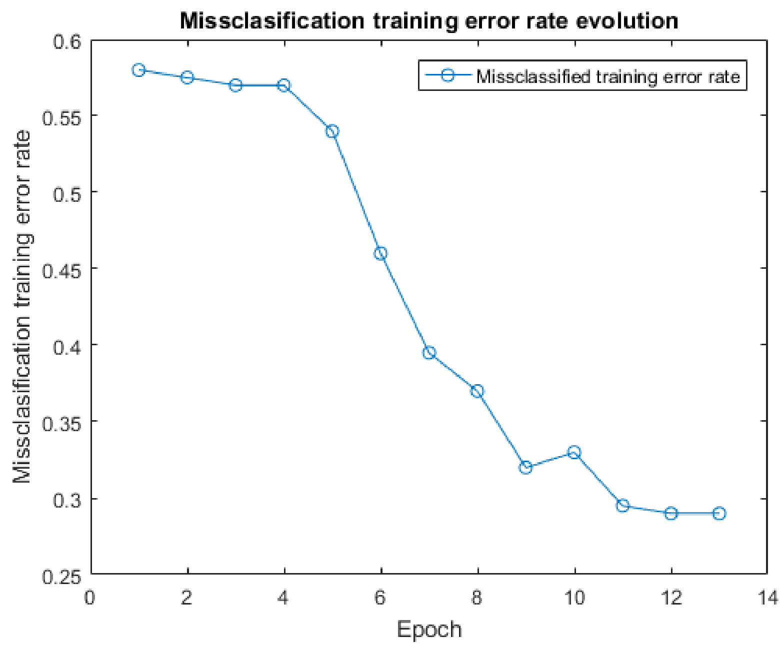

Figure 9: Missclasification error rate during training epochs with $\eta = 0.01$ (2-1)

Of course this solution only works if the two initial prototypes for the class A begin on a position close to each one of the clusters, otherwise they would be pushed away from the class B center and the prototype for class B would have been also moved away of its right position.

1.6 Exercise 3 d)

When using two and two prototypes of each class, we see that the general method still works, because the missclassification error rate tends to decrease and converge at lower error rate values (Figure 11), but not as low as when we used the minimum necessary prototypes number, this is due to the fact that the extra prototype of class A, pushes the class B prototypes a little further from itself while, "steals" some of the corrections that the other prototype would have gotten (Figure 10).

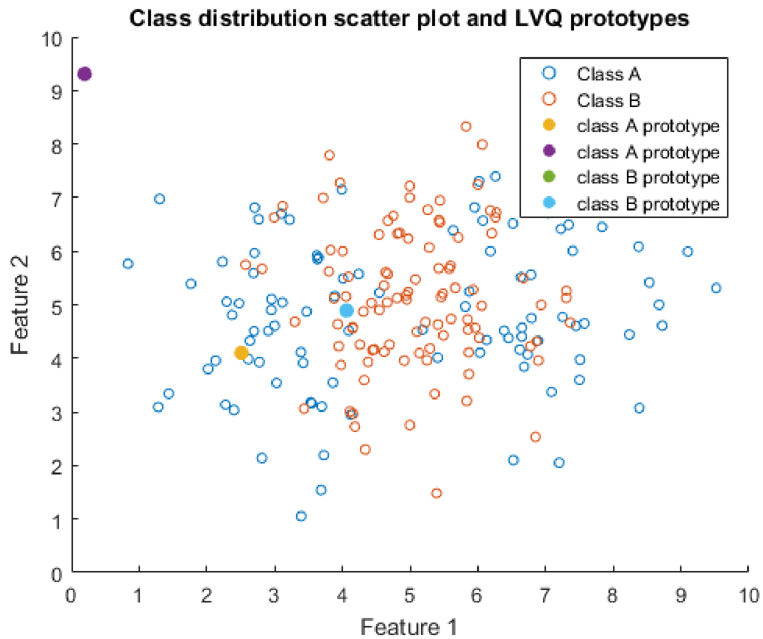

Figure 10: LVQ trained prototypes (2-2) with $\eta = 0.01$

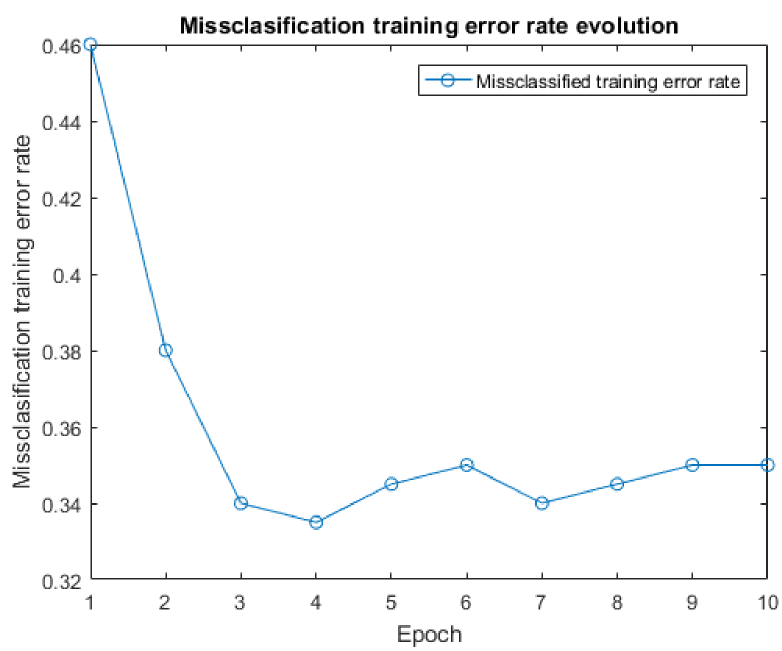

Figure 11: Missclasification error rate during training epochs with $\eta = 0.01$ (2-2)

After all of the tests we can conclude that one has to use the minimal number of prototypes necessary for each class, that way the missclassification error decreases, and the final position of the prototypes is more precise.

The code for generating all the tests for exercise 3, is included in the appendix (Script 4.2)

1.7 Exercise 4 (bonus)

We could observe through different η values, that if they are high values, sometimes the training converges after few epochs, but sometimes it never converges, furthermore the prototypes position is not too precise; when we use a small η value, the training takes more epochs to converge, but the result is more precise, the cost of this small value is the computational resources needed to compute the training prototypes.

Below we can see an example using $\eta = 0.1$

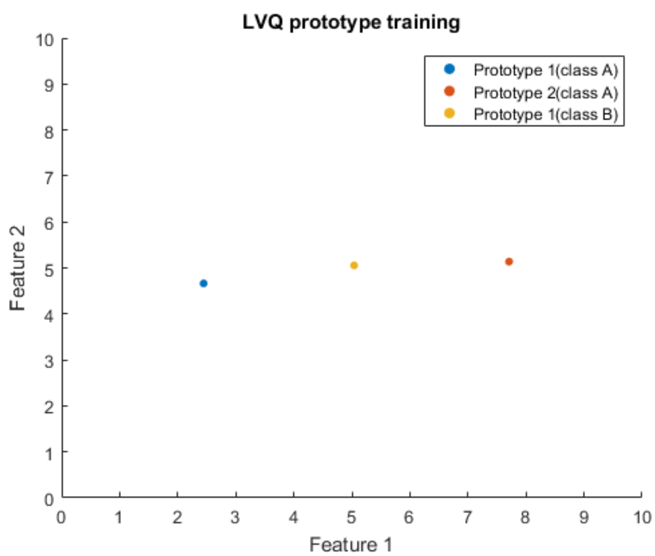

Figure 12: LVQ trained prototypes (2-1) with $\eta = 0.1$

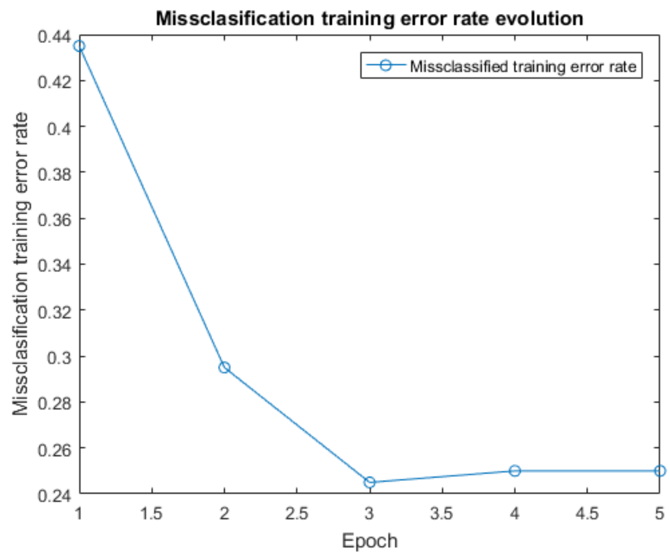

Figure 13: Missclasification error rate during training epochs with $\eta = 0.1$

And an example using $\eta = 0.001$

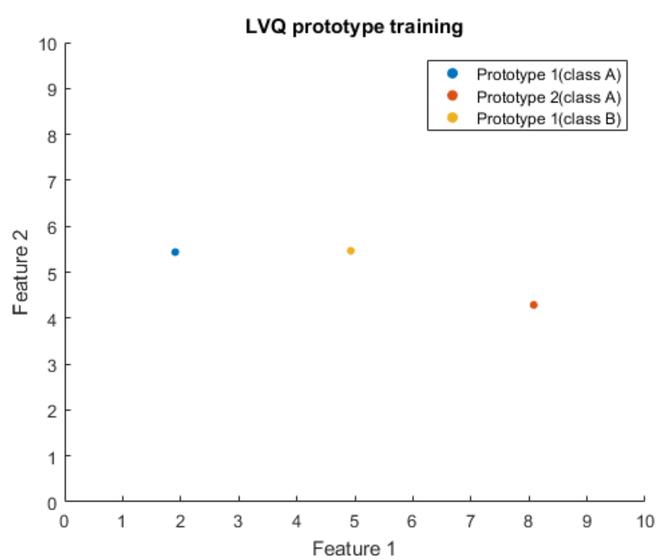

Figure 14: LVQ trained prototypes (2-1) with $\eta = 0.001$

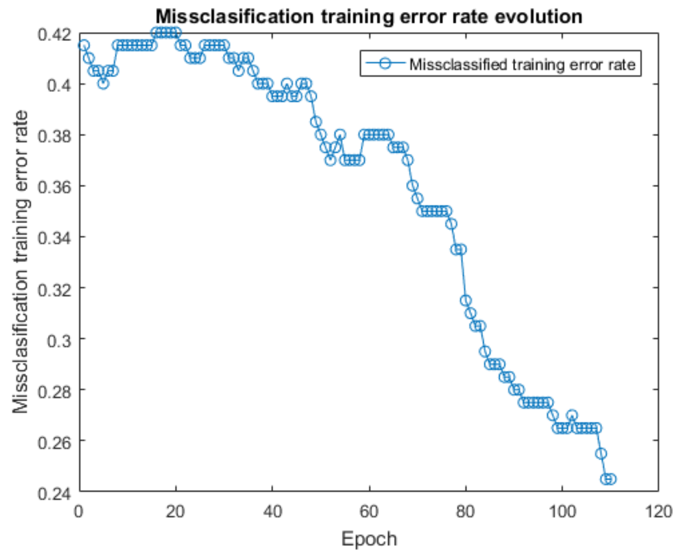

Figure 15: Missclassification error rate during training epochs with $\eta = 0.001$

2 Assignment 2

2.1 Exercise 1

We modified the script from assignment 1 exercise 3, to use ten folding validation method, this means, we divided the sample data into 10 groups and during ten iterations we left one different group for testing the LVQ after training the prototypes with the other 9 groups, we used 2 prototypes for class A and 1 prototype for class B. We obtained the same prototypes positions (Figure 16), and the following classification error for each fold (Figure 17).

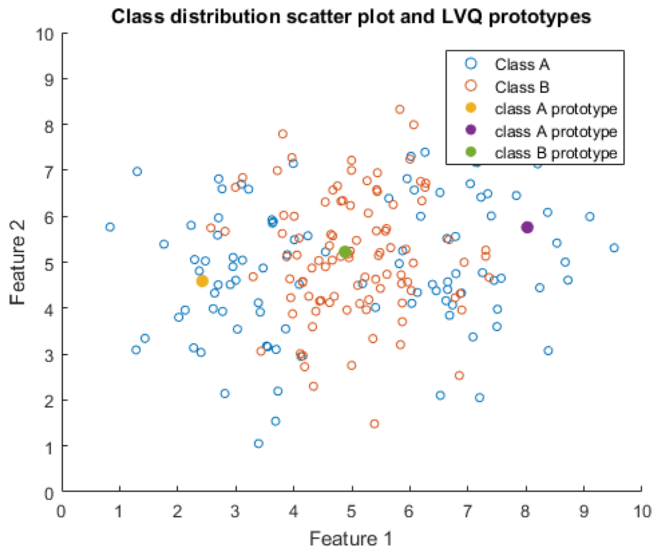

Figure 16: LVQ trained prototypes (2-1) with $\eta = 0.01$, using ten folding technique

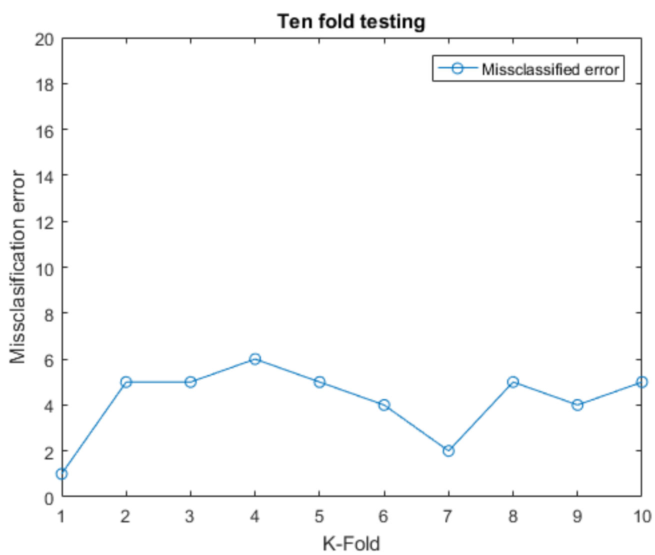

Figure 17: Testing error using ten folding technique, for prototypes (2-1) and $\eta = 0.01$

We used the matlab function:

```
indices = crossvalind('Kfold',data(:,3),10);
```

To randomly split the data into the ten folding samples.

The complete code used is on the appendix section under script 4.3.

2.2 Exercise 2

In order to compute the total test error, we calculated the average of the test errors of each fold (Figure 17) and simply computed the mean with the following instruction, obtaining a result of 4.2

```
%The test error is the mean of the classification errors
mean(testingError);
```

3 Assignment 3

3.1 Exercise 1

We implemented the GRLVQ algorithm using initial lambda values of 0.5, the complete code can be seen on appendix 4.4.

For calculating the new lambda, we used the following formula, using the Euclidean distance.

```
lambda1=lambda1-etha*phi*pdist2(point,prototype);
```

We can see that the distance the prototype is adjusted get determined by how far the prototype is of its final location.

3.2 Exercise 2

The final relevances after each epoch can be seen on the following plot (Figure 18).We can see that they start separating as the prototypes converge.

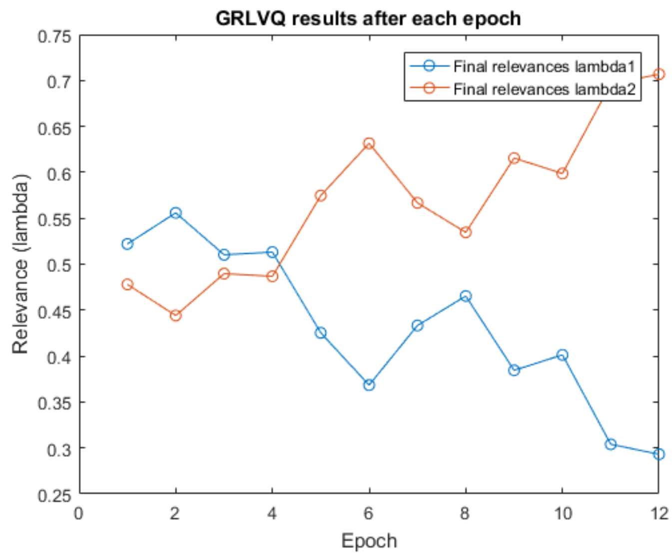

Figure 18: Final relevances (lambda) values after each epoch for prototypes (2-1) and $\eta = 0.01$

3.3 Exercise 3

The final LVQ prototypes are the following (Figure 19), the prototypes always converge, even if the initial prototypes are far from its ideal location.

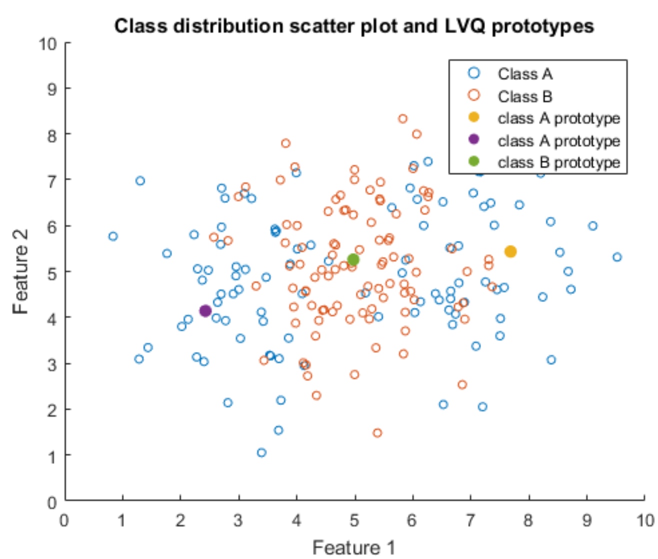

Figure 19: Final prototypes using GRLVQ for (2-1) prototypes and $\eta = 0.01$

3.4 Exercise 4

Below we can see the training error E after each epoch validation (Figure 20) and the relevances as a function of the epochs (Figure 21). We can observer that at the beginning the training started with two initial values well located, that is why only after 12 epochs the training phase finished.

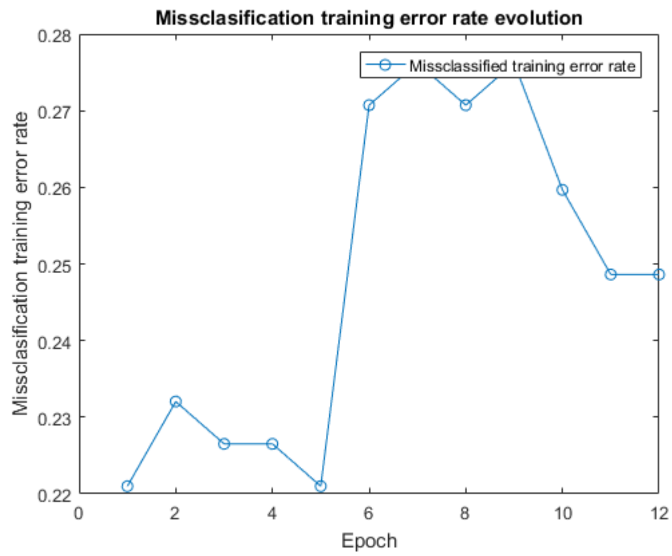

Figure 20: Training error after each epoch using (2-1) prototypes and $\eta = 0.01$

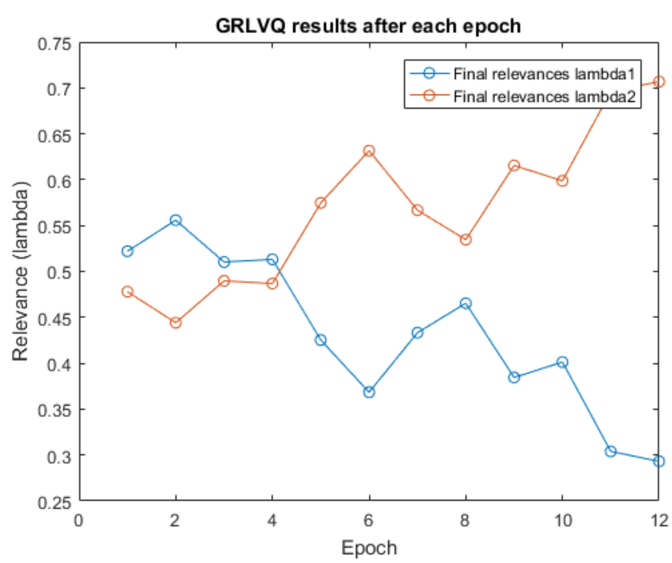

Figure 21: Final relevances (lambda) values after each epoch for prototypes (2-1) and $\eta = 0.01$

3.5 Exercise 5

Using the ten fold cross validation we obtain the following results (Figure 22), with a testing group of 20 samples, we can observe that the error always remain low.

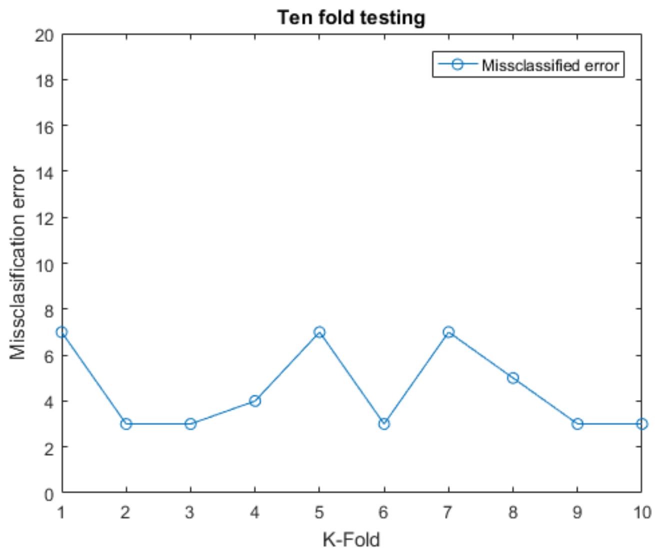

Figure 22: Errors using ten fold validation for (2-1) and $\eta = 0.01$

3.6 Exercise 6

We compute the test error using the mean of the ten fold cross validation errors, and the result is 4.5, the complete code can be seen on the appendix (script 4.4), this means that 4.5 out of 20 test were correct, is a hit rate around 77.5%.

4 Appendix

4.1 Script for assignment 1, exercise 2

```
function LVQ1ex2(eta)
 %Load data class_a and class_b
    load data_lvq_A.mat
    load data_lvq_B.mat

    %Add category label 0=class A, 1=class B
    matA(:,3)=0;
    matB(:,3)=1;

    %Concatenate the two matrices
    data=vertcat(matA,matB);
```

```

    %Randomly permute the rows, so we have an unbiased training data
    data2=data(randperm(length(data(:,1))),:);

    %We obtain the final position of the three prototypes after N
    %epochs,where the variation between epochs becomes smaller than the
    %threshold
    [prototype1A,prototype2A,prototype1B,epochNumber,epochError]=LVQeval(data2, eta);

    x=1:1:length(epochError);

    %Plot of the missclasification training error rate
    figure
    plot(x,epochError,'-o');
        %axis([0 10 0 10]);
    title('Missclasification training error rate evolution')
    xlabel('Epoch')
    ylabel('Missclasification training error rate')
    legend('Missclassified training error rate');
end

function [prototype1A,prototype2A,prototype1B,epochNumber,epochError]=LVQeval(data, eta)
    %Randomly generate two prototypes for class A
    prototype1A=[rand()*4,rand()*9];
    prototype2A=[6+rand()*4,rand()*9];

    %Randomly generate one prototype for class B
    prototype1B=[rand()*10,rand()*9];

    %Missclasiffied training error
    newMTE=1;
    MTEdiff=1;
    variationThreshold=0.0001;
    errorThreshold=0.25;

    epochNumber=0;

    %Run epochs until the eror difference becomes smaller than threshold
    while(MTEdiff>variationThreshold || newMTE>errorThreshold)

        epochNumber=epochNumber+1;
        oldMTE=newMTE;

        [newMTE,prototype1A,prototype2A,prototype1B]=epoch(data,prototype1A,prototype2A,
            prototype1B,eta);

        MTEdiff=abs(oldMTE-newMTE);
        %We save the missclassified training error for each epoch
        epochError(epochNumber)=newMTE;

        %Animated plotting
        scatter(prototype1A(1),prototype1A(2),20,'filled');
        axis([0 10 0 10]);
        title('LVQ prototype training')
        xlabel('Feature 1')
        ylabel('Feature 2')
        hold on;

        scatter(prototype2A(1),prototype2A(2),20,'filled');
        scatter(prototype1B(1),prototype1B(2),20,'filled');
        legend('Prototype 1(class A)','Prototype 2(class A)','Prototype 1(class B)');
```

```
        hold off

       pause(0.10);
    end
end

function [trainingErrorRate,prototype1A,prototype2A,prototype1B]=epoch(data,prototype1A,
    prototype2A,prototype1B,eta)
%Learning phase
    mte=0;

    for i=1:1:size(data(:,1))

        point=data(i,1:2);

        %Find the class of the closest prototype
        closestPrototype=WinnerEuc(point, prototype1A, prototype2A, prototype1B);

        switch closestPrototype
            case '1A'
                %Compare if they belong to the same class
                if(data(i,3)==0)
                    prototype1A=newPosition(prototype1A, point, eta, 1);
                else
                    prototype1A=newPosition(prototype1A, point, eta, -1);
                    mte=mte+1;
                end
            case '2A'
                %Compare if they belong to the same class
                if(data(i,3)==0)
                    prototype2A=newPosition(prototype2A, point, eta, 1);
                else
                    prototype2A=newPosition(prototype2A, point, eta, -1);
                    mte=mte+1;
                end
            case '1B'
                %Compare if they belong to the same class
                if(data(i,3)==1)
                    prototype1B=newPosition(prototype1B, point, eta, 1);
                else
                    prototype1B=newPosition(prototype1B, point, eta, -1);
                    mte=mte+1;
                end
        end
    end
    trainingErrorRate=mte/length(data(:,1));
end

%Return the closest prototype to the point
function minD=WinnerEuc(point, prototype1A, prototype2A, prototype1B)
    d1A=pdist2(point, prototype1A);
    minD='1A';

    d2A=pdist2(point, prototype2A);
    if(d2A<d1A)
        minD='2A';
    end

    d1B=pdist2(point, prototype1B);
    if(d1B<d1A & d1B<d2A)
        minD='1B';
```

```
    end
end

function w=newPosition(prototype, point, eta, phi)
    w=prototype+(eta*phi*(point-prototype));
end
```

4.2 Script for assignment 1, exercise 3

```
function LVQ1ex3a(etha,noProtA,noProtB)
 %Load data class_a and class_b
    load data_lvq_A.mat
    load data_lvq_B.mat

    %Add category label 0=class A, 1=class B
    matA(:,3)=0;
    matB(:,3)=1;

    %Concatenate the two matrices
    data=vertcat(matA,matB);

    %Randomly permute the rows, so we have an unbiased training data
    data2=data(randperm(length(data(:,1))),:);

    %We obtain the final position of the three prototypes after N
    %epochs,where the variation between epochs becomes smaller than the
    %threshold
    [prototype,epochNumber,epochError]=LVQeval(data2, etha,noProtA,noProtB);

    %Plot feature2 vs feature1 of both classes
    figure
    scatter(matA(:,1),matA(:,2),20,'DisplayName','Class A');
    axis([0 10 0 10]);
    title('Class distribution scatter plot and LVQ prototypes')
    xlabel('Feature 1')
    ylabel('Feature 2')
    hold on;

    scatter(matB(:,1),matB(:,2),20,'DisplayName','Class B');

    for(i=1:length(prototype(:,1)))

        if(prototype(i,3)==0)
            strLeg='class A prototype';
        else
            strLeg='class B prototype';
        end
        scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
    end

    legend('show');
    hold off;

    x=1:1:length(epochError);

    %Plot of the missclasification training error rate
    figure
    plot(x,epochError,'-o');
        %axis([0 10 0 10]);
    title('Missclasification training error rate evolution')
```

```
    xlabel('Epoch')
    ylabel('Missclasification training error rate')
    legend('Missclassified training error rate');

end

function [prototype,epochNumber,epochError]=LVQeval(data, etha, noProtA, noProtB)
    %Randomly generate the prototype matrix, column 3 identifies the class
    %(A=0, B=1)
    for(i=1:(noProtA+noProtB))
        if(i<=noProtA)
            prototype(i,:)=[rand()*10,rand()*10,0];
        else
            prototype(i,:)=[rand()*10,rand()*10,1];
        end
    end

    %Missclasiffied training error
    newMTE=1;
    MTEdiff=1;
    variationThreshold=0.0001;
    errorThreshold=0.45;

    epochNumber=0;

    %Run epochs until the eror difference becomes smaller than threshold
    %while(MTEdiff>variationThreshold)
    while(MTEdiff>variationThreshold || newMTE>errorThreshold)

        epochNumber=epochNumber+1
        oldMTE=newMTE;

        [newMTE,prototype]=epoch(data,prototype,etha);
        disp(newMTE)
        MTEdiff=abs(oldMTE-newMTE)
        %We save the missclassified training error for each epoch
        epochError(epochNumber)=newMTE;

        strLeg='';

        %Animated plotting
        for(i=1:length(prototype(:,1)))

            if(prototype(i,3)==0)
                strLeg='class A';
            else
                strLeg='class B';
            end

            scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
            axis([0 10 0 10]);
            title('LVQ prototype training')
            xlabel('Feature 1')
            ylabel('Feature 2')
            hold on;

        end

        legend('show');
        hold off

```

```
        pause(0.10);
    end
end

function [trainingErrorRate,prototype]=epoch(data,prototype,etha)
%Learning phase
    mte=0;

    for i=1:1:size(data(:,1))

        point=data(i,1:2);

        %Find the class of the closest prototype
        closestPrototype=WinnerEuc(point, prototype);

        %Compare if they belong to the same class
        if(data(i,3)==prototype(closestPrototype,3))
            prototype(closestPrototype,1:2)=newPosition(prototype(closestPrototype,1:2),
                point, etha, 1);
        else
            prototype(closestPrototype,1:2)=newPosition(prototype(closestPrototype,1:2),
                point, etha, -1);
            mte=mte+1;
        end
    end
    trainingErrorRate=mte/length(data(:,1));
end

%Return the closest prototype to the point
function closestPrototypeIndex=WinnerEuc(point, prototype)
    oldD=100;
    for(i=1:length(prototype(:,1)))
        d=pdist2(point, prototype(i,1:2));

        if(d<oldD)
            closestPrototypeIndex=i;
            oldD=d;
        end
    end
end

function w=newPosition(prototype, point, etha, phi)
    w=prototype+(etha*phi*(point-prototype));
end
```

4.3 Script for assignment 2

```
function LVQ2(etha,noProtA,noProtB)
 %Load data class_a and class_b
    load data_lvq_A.mat
    load data_lvq_B.mat

    %Add category label 0=class A, 1=class B
    matA(:,3)=0;
    matB(:,3)=1;

    %Concatenate the two matrices
    data=vertcat(matA,matB);

    %Randomly permute the rows, so we have an unbiased training data
```

```
    data2=data(randperm(length(data(:,1))),:);


    %We obtain the final position of the three prototypes after N
    %epochs,where the variation between epochs becomes smaller than the
    %threshold
    [prototype,testingError]=LVQeval(data2, etha,noProtA,noProtB);

    %Plot feature2 vs feature1 of both classes
    figure
    scatter(matA(:,1),matA(:,2),20,'DisplayName','Class A');
    axis([0 10 0 10]);
    title('Class distribution scatter plot and LVQ prototypes')
    xlabel('Feature 1')
    ylabel('Feature 2')
    hold on;

    scatter(matB(:,1),matB(:,2),20,'DisplayName','Class B');

    for(i=1:length(prototype(:,1)))

        if(prototype(i,3)==0)
            strLeg='class A prototype';
        else
            strLeg='class B prototype';
        end
        scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
    end

    legend('show');
    hold off;

    x=1:1:length(testingError);

    %Plot of the missclasification training error rate
    figure
    plot(x,testingError,'-o');
    axis([1 10 0 20]);
    title('Ten fold testing')
    xlabel('K-Fold')
    ylabel('Missclasification error')
    legend('Missclassified error');

    %The test error is the mean of the classification errors
    disp(mean(testingError));

end

function [prototype,testingError]=LVQeval(data, etha, noProtA, noProtB)
    %Randomly generate the prototype matrix, column 3 identifies the class
    %(A=0, B=1)
    for(i=1:(noProtA+noProtB))
        if(i<=noProtA)
            prototype(i,:)=[rand()*10,rand()*10,0];
        else
            prototype(i,:)=[rand()*10,rand()*10,1];
        end
    end

    indices = crossvalind('Kfold',data(:,3),10);
    data(:,4)=indices;
```

```

    %Run ten-fold training validatino
    for(i=1:1:10)
        trainData=[0 0 0];
        testData=[0 0 0];
        for(j=1:1:200)
            if(data(j,4)~=i)
                trainData(length(trainData(:,1))+1,:)=data(j,1:3);
            else
                testData(length(testData(:,1))+1,:)=data(j,1:3);
            end
        end

        [testingError(i),prototype]=epoch(trainData,testData,prototype,etha);

        %Animated plotting
        for(i=1:length(prototype(:,1)))

            if(prototype(i,3)==0)
                strLeg='class A';
            else
                strLeg='class B';
            end

            scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
            axis([0 10 0 10]);
            title('LVQ prototype training')
            xlabel('Feature 1')
            ylabel('Feature 2')
            hold on;

        end

        legend('show');
        hold off

       pause(0.10);
    end
end

function [testingError,prototype]=epoch(data,testData,prototype,etha)
%Learning phase
    testingError=0;

    for i=2:1:size(data(:,1))

        point=data(i,1:2);

        %Find the class of the closest prototype
        closestPrototype=WinnerEuc(point, prototype);

        %Compare if they belong to the same class
        if(data(i,3)==prototype(closestPrototype,3))
            prototype(closestPrototype,1:2)=newPosition(prototype(closestPrototype,1:2),
                point, etha, 1);
        else
            prototype(closestPrototype,1:2)=newPosition(prototype(closestPrototype,1:2),
                point, etha, -1);
        end
    end

```

```
    for  i=2:1:size(testData(:,1))
        point=testData(i,1:2);
        %Find the class of the closest prototype
        closestPrototype=WinnerEuc(point, prototype);

        %Compare if they belong to the same class
        if(testData(i,3)~=prototype(closestPrototype,3))
            testingError=testingError+1;
        end
    end
end

%Return the closest prototype to the point
function closestPrototypeIndex=WinnerEuc(point, prototype)
    oldD=100;
    for(i=1:length(prototype(:,1)))
        d=pdist2(point, prototype(i,1:2));

        if(d<oldD)
            closestPrototypeIndex=i;
            oldD=d;
        end
    end
end

function w=newPosition(prototype, point, etha, phi)
    w=prototype+(etha*phi*(point-prototype));
end
```

4.4 Script for assignment 3

```
function LVQ3(etha,noProtA,noProtB)
 %Load data class_a and class_b
    load data_lvq_A.mat
    load data_lvq_B.mat

    %Add category label 0=class A, 1=class B
    matA(:,3)=0;
    matB(:,3)=1;

    %Concatenate the two matrices
    data=vertcat(matA,matB);

    %Randomly permute the rows, so we have an unbiased training data
    data2=data(randperm(length(data(:,1))),:);

    %We obtain the final position of the three prototypes after N
    %epochs,where the variation between epochs becomes smaller than the
    %threshold
    [prototype,epochNumber,epochError,testingError,lambda1,lambda2]=LVQeval(data2, etha,
        noProtA,noProtB);

    %Plot feature2 vs feature1 of both classes
    figure
    scatter(matA(:,1),matA(:,2),20,'DisplayName','Class A');
    axis([0 10 0 10]);
    title('Class distribution scatter plot and LVQ prototypes')
    xlabel('Feature 1')
    ylabel('Feature 2')
    hold on;
```

```
    scatter(matB(:,1),matB(:,2),20,'DisplayName','Class B');

    for(i=1:length(prototype(:,1)))

        if(prototype(i,3)==0)
            strLeg='class A prototype';
        else
            strLeg='class B prototype';
        end
        scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
    end

    legend('show');
    hold off;

    x=1:1:length(epochError);

    %Plot of the missclasification training error rate
    figure
    plot(x,epochError,'-o');
        %axis([0 10 0 10]);
    title('Missclasification training error rate evolution')
    xlabel('Epoch')
    ylabel('Missclasification training error rate')
    legend('Missclassified training error rate');

    %Plot of the clasification error
    x=1:1:length(testingError);
    figure
    plot(x,testingError,'-o');
    axis([1 10 0 20]);
    title('Ten fold testing')
    xlabel('K-Fold')
    ylabel('Missclasification error')
    legend('Missclassified error');

    %The test error is the mean of the classification errors
    disp(mean(testingError));

    %Plot of the final relevances after each epoch
    x=1:1:length(lambda1);
    figure
    plot(x,lambda1,'-o','DisplayName','Final relevances lambda1');
    title('GRLVQ results after each epoch')
    xlabel('Epoch')
    ylabel('Relevance (lambda)')

    hold on;

    plot(x,lambda2,'-o','DisplayName','Final relevances lambda2');
    legend('show');
    hold off;

end

function [prototype,epochNumber,epochError,testingError,lambda1E,lambda2E]=LVQeval(data,
    etha, noProtA, noProtB)
    %Randomly generate the prototype matrix, column 3 identifies the class
    %(A=0, B=1)
    for(i=1:(noProtA+noProtB))
```

```
        if(i<=noProtA)
            prototype(i,:)=[rand()*10,rand()*10,0];
        else
            prototype(i,:)=[rand()*10,rand()*10,1];
        end
    end

    indices = crossvalind('Kfold',data(:,3),10);
    data(:,4)=indices;

    %Run ten-fold training validatino
    for(i=1:1:10)
        trainData=[0 0 0];
        testData=[0 0 0];
        for(j=1:1:200)
            if(data(j,4)~=i)
                trainData(length(trainData(:,1))+1,:)=data(j,1:3);
            else
                testData(length(testData(:,1))+1,:)=data(j,1:3);
            end
        end

        %Missclasiffied training error
        newMTE=1;
        MTEdiff=1;
        variationThreshold=0.0001;
        errorThreshold=0.25;

        epochNumber=0;

        %Run epochs until the eror difference becomes smaller than threshold
        %while(MTEdiff>variationThreshold)
        while(MTEdiff>variationThreshold || newMTE>errorThreshold)

            epochNumber=epochNumber+1;
            oldMTE=newMTE;

            [newMTE,prototype,testingError(i),lambda1,lambda2]=epoch(trainData,testData,
                prototype,etha);

            MTEdiff=abs(oldMTE-newMTE);
            %We save the missclassified training error for each epoch
            epochError(epochNumber)=newMTE;
            lambda1E(epochNumber)=lambda1;
            lambda2E(epochNumber)=lambda2;
            strLeg='';

            %Animated plotting
            for(i=1:length(prototype(:,1)))

                if(prototype(i,3)==0)
                    strLeg='class A';
                else
                    strLeg='class B';
                end

                scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
                axis([0 10 0 10]);
                title('LVQ prototype training')
                xlabel('Feature 1')
                ylabel('Feature 2')
```

```
                hold on;

            end

            legend('show');
            hold off

           pause(0.10);
        end
    end
end

function [trainingErrorRate,prototype,testingError,lambda1,lambda2]=epoch(data,testData,
    prototype,etha)
%Learning phase
    mte=0;
    lambda1=0.5;
    lambda2=0.5;
    testingError=0;
    for i=1:1:size(data(:,1))

        point=data(i,1:2);

        %Find the class of the closest prototype
        closestPrototype=WinnerEuc(point, prototype);

        %Compare if they belong to the same class
        if(data(i,3)==prototype(closestPrototype,3))
            [prototype(closestPrototype,1:2),lambda1,lambda2]=newPosition(prototype(
                closestPrototype,1:2), point, etha, 1,lambda1,lambda2,data(i,3));
        else
            [prototype(closestPrototype,1:2),lambda1,lambda2]=newPosition(prototype(
                closestPrototype,1:2), point, etha, -1,lambda1,lambda2,data(i,3));
            mte=mte+1;
        end
    end
    trainingErrorRate=mte/length(data(:,1));

%Testing phase
    for i=2:1:size(testData(:,1))
        point=testData(i,1:2);
        %Find the class of the closest prototype
        closestPrototype=WinnerEuc(point, prototype);

        %Compare if they belong to the same class
        if(testData(i,3)~=prototype(closestPrototype,3))
            testingError=testingError+1;
        end
    end
end

%Return the closest prototype to the point
function closestPrototypeIndex=WinnerEuc(point, prototype)
    oldD=100;
    for(i=1:length(prototype(:,1)))
        d=pdist2(point, prototype(i,1:2));

        if(d<oldD)
            closestPrototypeIndex=i;
            oldD=d;
        end
```

```
    end
end

function [w,lambda1,lambda2]=newPosition(prototype, point, etha, phi, lambda1, lambda2,
    class)
    if(class==0)
        w=prototype+(etha*phi*lambda1*(point-prototype));
        %Calculate the new lambda
        lambda1=lambda1-etha*phi*pdist2(point,prototype);
        %Enforce
        lambda2=1-lambda1;
    else
        w=prototype+(etha*phi*lambda2*(point-prototype));
        %Calculate the new lambda
        lambda2=lambda2-etha*phi*pdist2(point,prototype);
        %Enforce
        lambda1=1-lambda2;
    end
end
```