

# Lab 5

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

Group 16

Pattern Recognition

FMNS • RUG

October 17, 2016

## 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

```
1 function LVQ1ex1()
2     %Load data class_a and class_b
3     load data_lvq_A.mat
4     load data_lvq_B.mat
5
6     %Plot feature2 vs feature1 of both classes
7     scatter(matA(:,1),matA(:,2),20);
8     title('Class distribution scatter plot')
9     xlabel('Feature 1')
10    ylabel('Feature 2')
11    hold on;
12
13    scatter(matB(:,1),matB(:,2),20);
14    legend('Class A','Class B');
15    hold off;
16 end
```

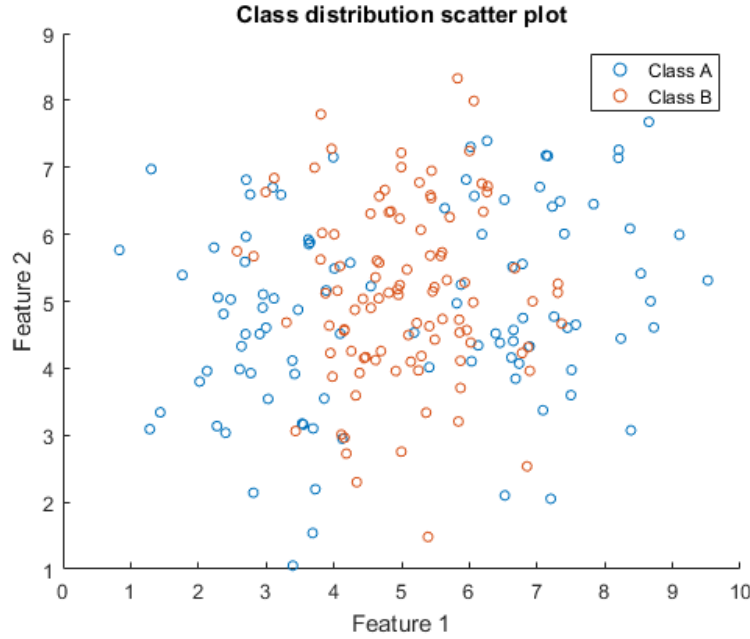


Figure 1: 2 dimension feature map of two different classes

## 1.2 Exercise 2

For class A we are going 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 training we obtain the following prototypes positions (Figure 2). We could observe through different  $\eta$  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  $\eta$  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.

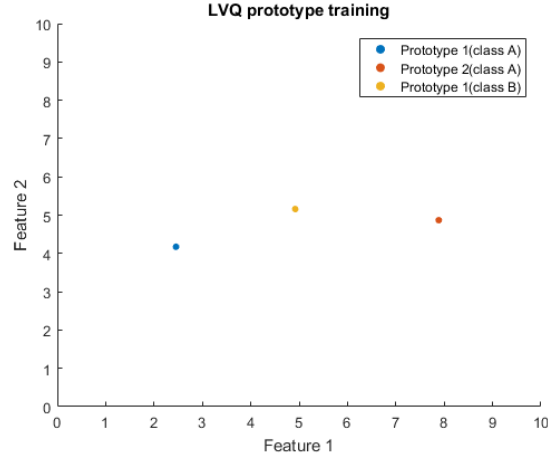


Figure 2: LVQ trained prototypes (2-1)

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.

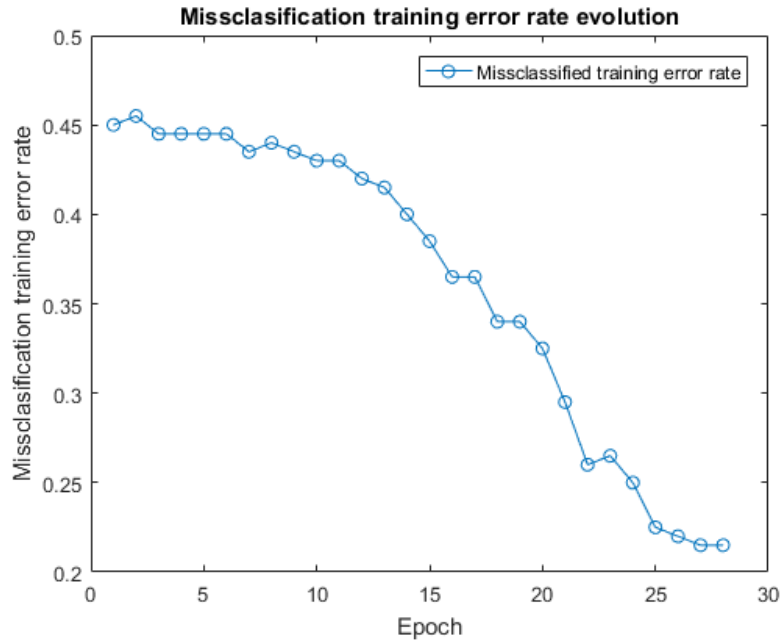


Figure 3: Missclassification error rate during training epochs

### 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 analyzed 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 selection 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.

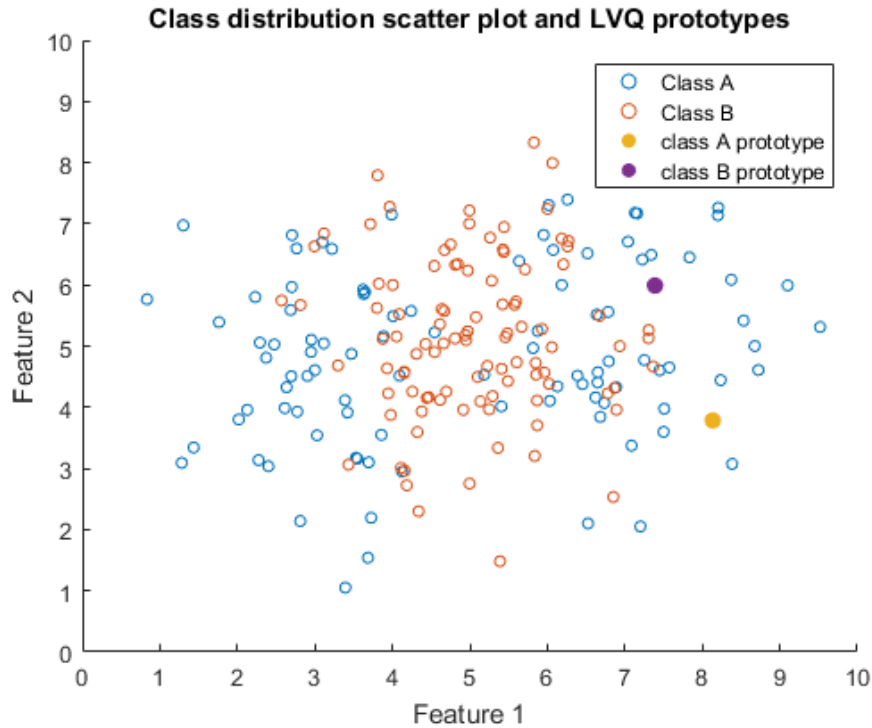


Figure 4: LVQ trained prototypes (1-1) with  $\eta = 0.01$

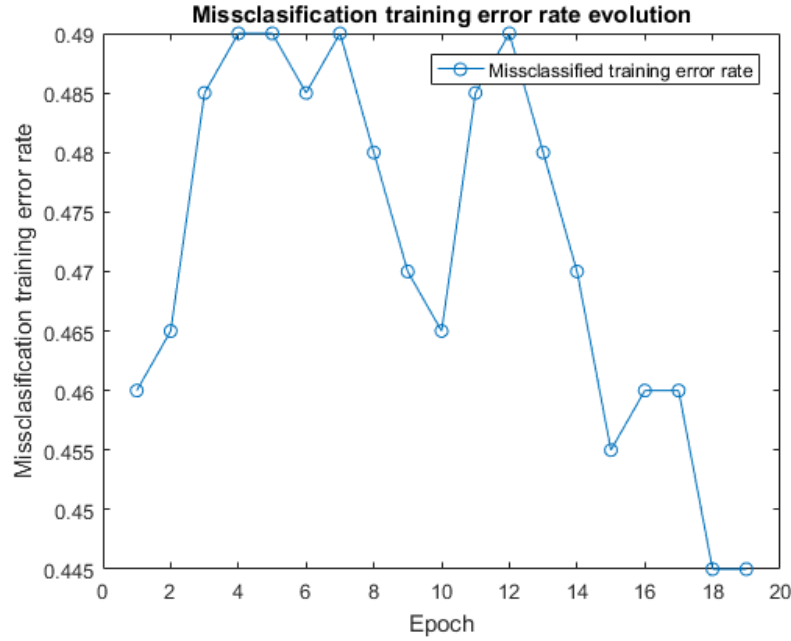


Figure 5: Missclassification error rate during training epochs with  $\eta = 0.01$  (1-1)

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 prototype of class A.

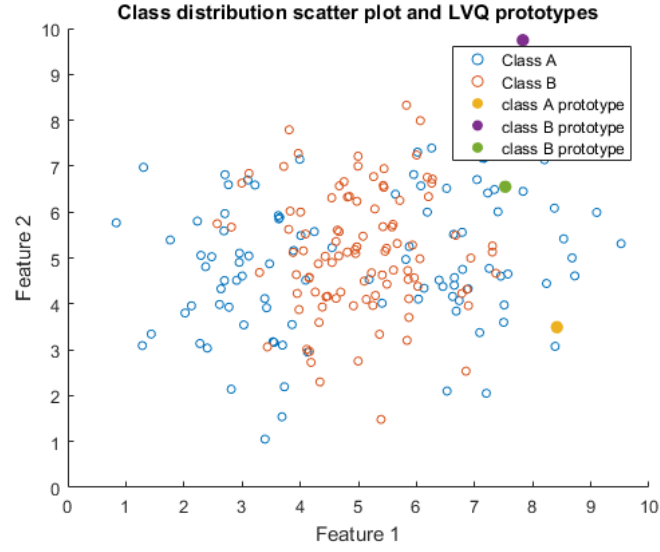


Figure 6: LVQ trained prototypes (1-2) with  $\eta = 0.01$



Figure 7: Missclassification 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 (close to 0.3). We can conclude that this is a working solution for the sample data.

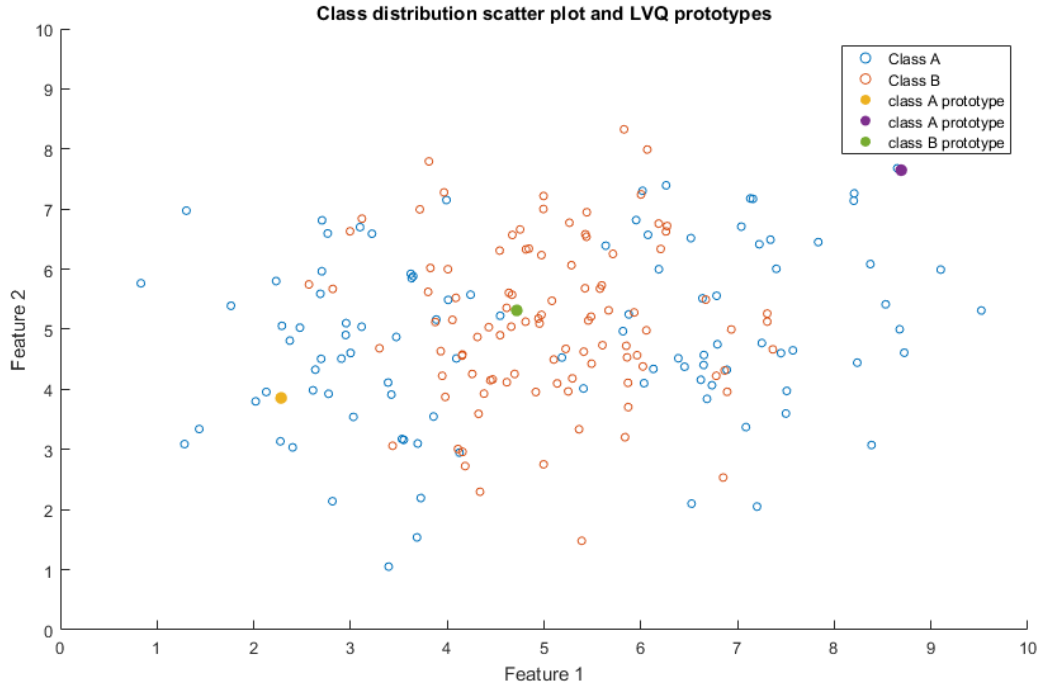


Figure 8: LVQ trained prototypes (2-1) with  $\eta = 0.01$

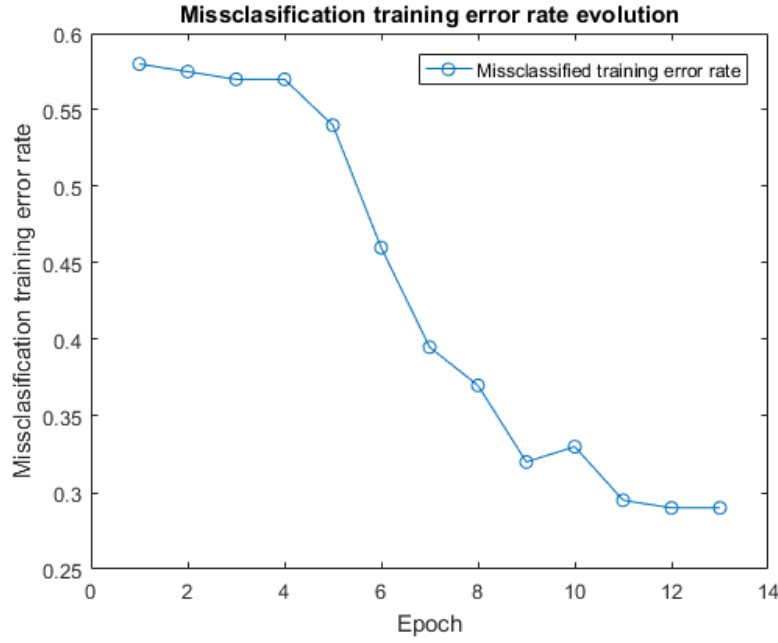


Figure 9: Missclassification error rate during training epochs with  $\eta = 0.01$  (2-1)

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).



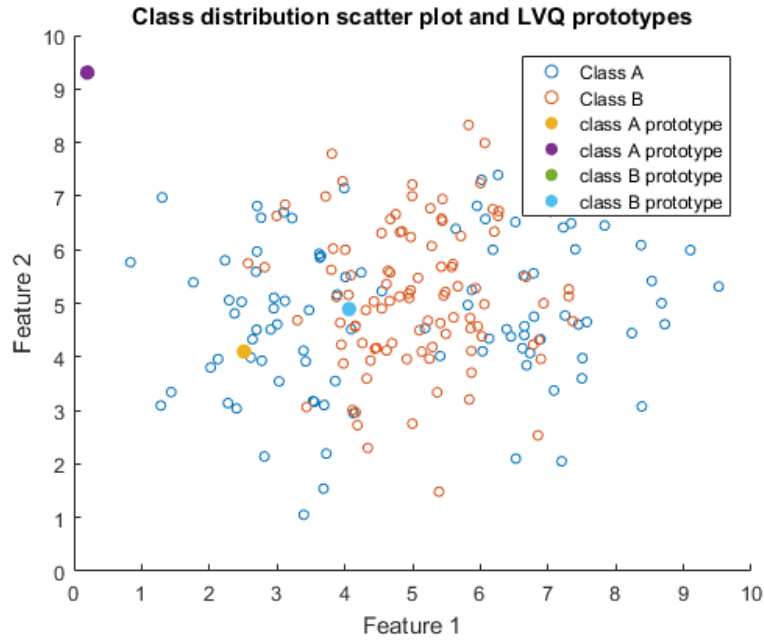


Figure 10: LVQ trained prototypes (2-2) with  $\eta = 0.01$



Figure 11: Missclassification 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  $\eta$  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  $\eta$  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  $\eta = 0.1$

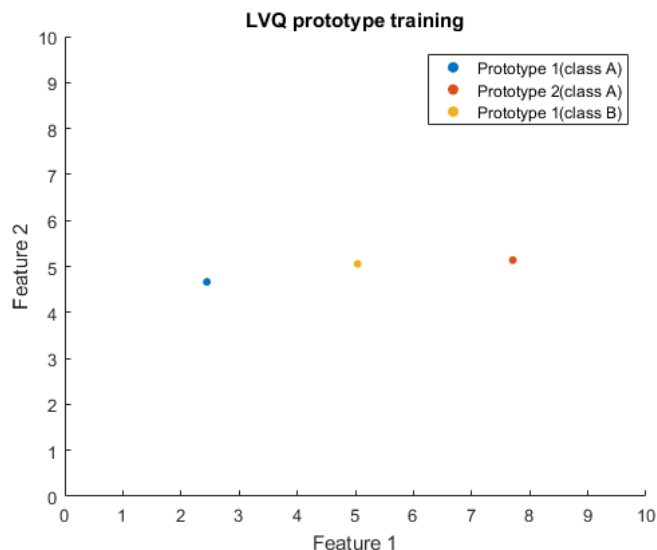


Figure 12: LVQ trained prototypes (2-1) with  $\eta = 0.1$

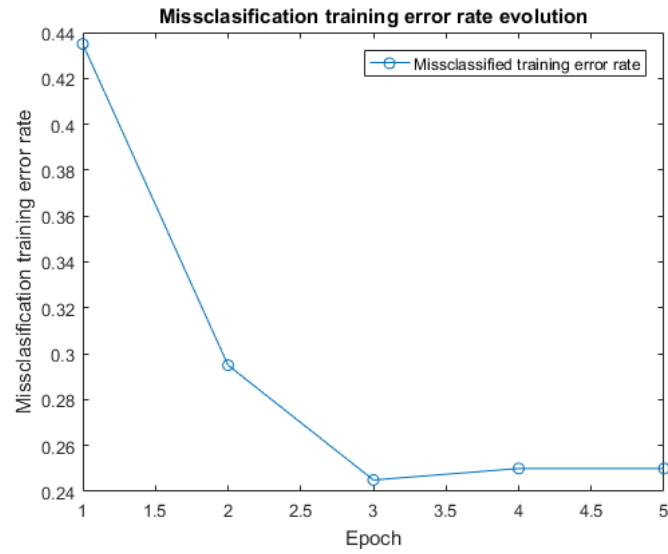


Figure 13: Missclassification error rate during training epochs with  $\eta = 0.1$

And an example using  $\eta = 0.001$

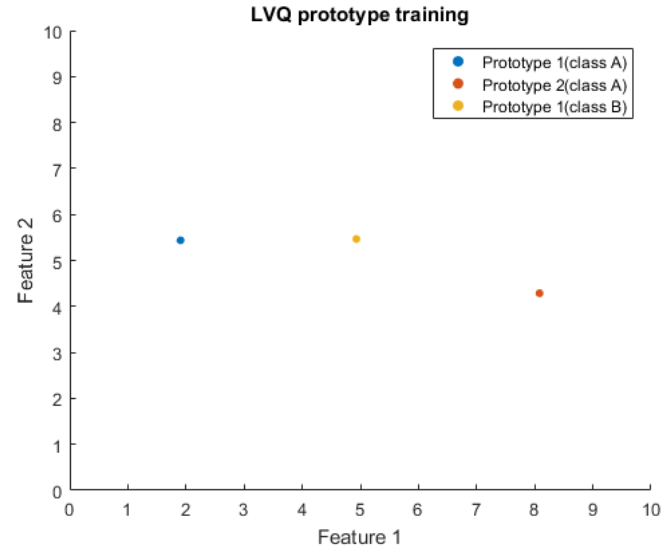


Figure 14: LVQ trained prototypes (2-1) with  $\eta = 0.001$

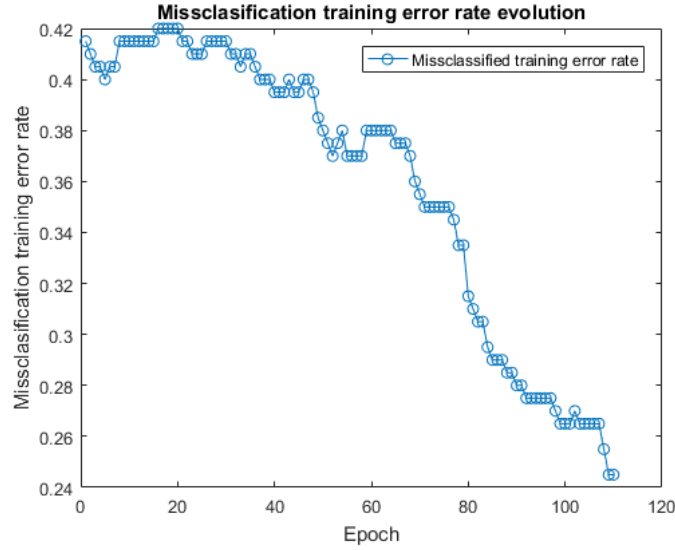


Figure 15: Missclassification error rate during training epochs with  $\eta = 0.001$

## 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).

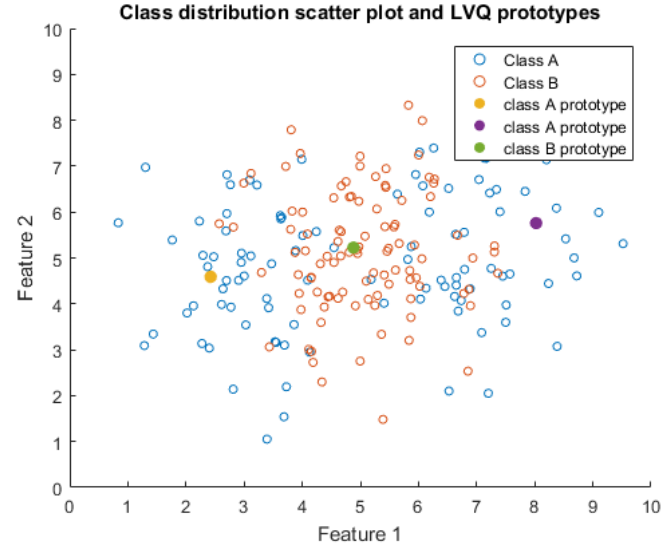


Figure 16: LVQ trained prototypes (2-1) with  $\eta = 0.01$ , using ten folding technique

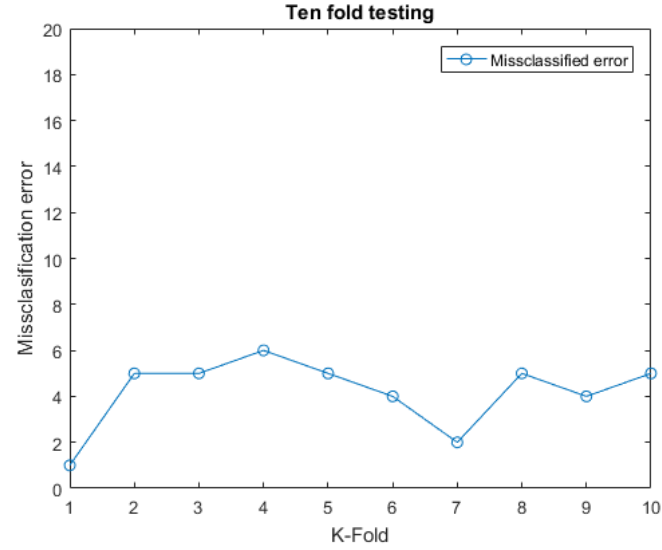


Figure 17: Testing error using ten folding technique, for prototypes (2-1) and  $\eta = 0.01$

We used the matlab function:

```
1 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

```
1 %The test error is the mean of the classification errors
2 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.

```
1 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.

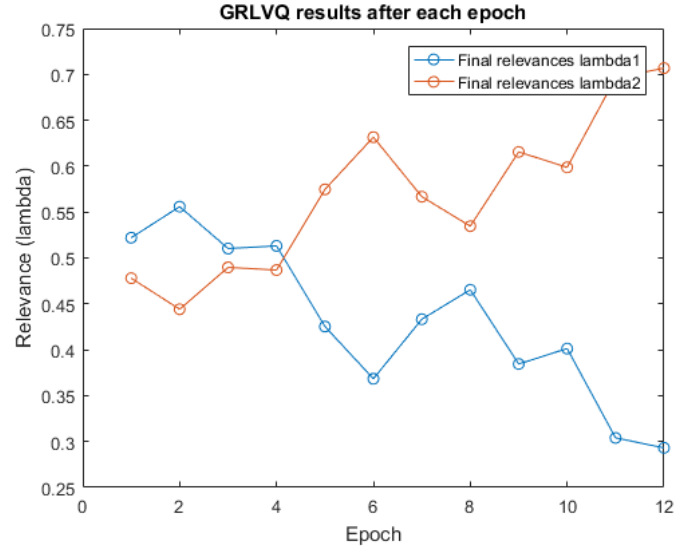


Figure 18: Final relevances (lambda) values after each epoch for prototypes (2-1) and  $\eta = 0.01$

### 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.

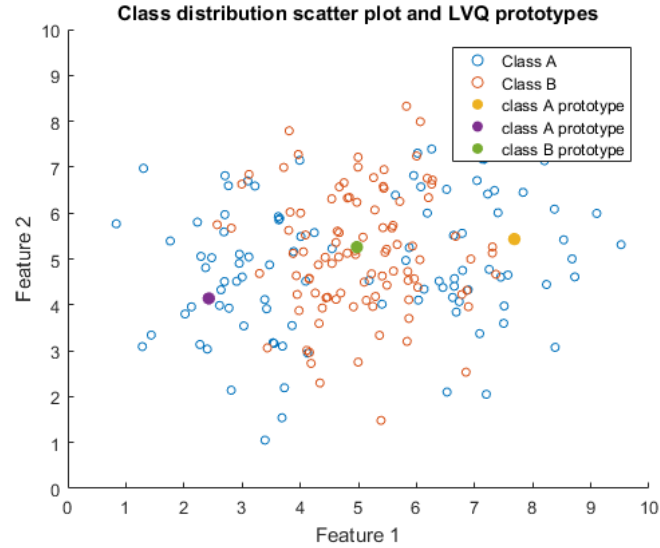


Figure 19: Final prototypes using GRLVQ for (2-1) prototypes and  $\eta = 0.01$

### 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 observe that at the beginning the training started with two initial values well located, that is why only after 12 epochs the training phase finished.

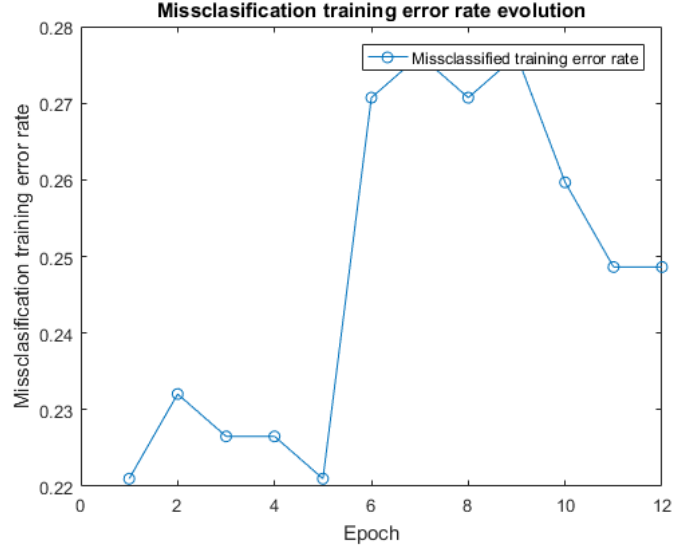


Figure 20: Training error after each epoch using (2-1) prototypes and  $\eta = 0.01$

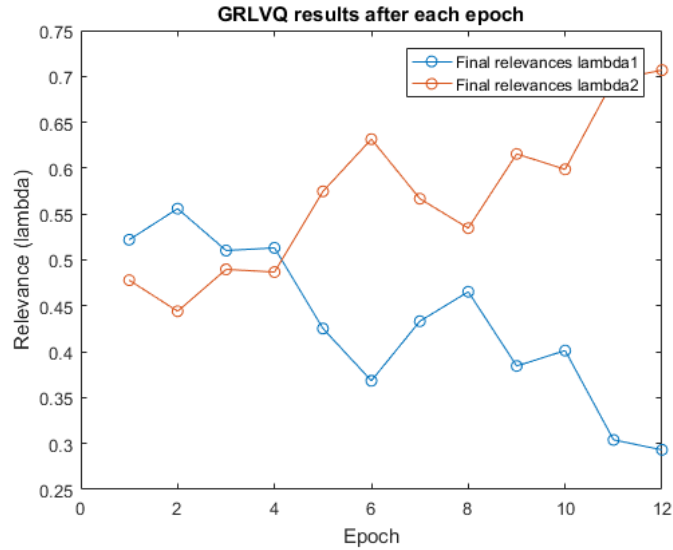


Figure 21: Final relevances (lambda) values after each epoch for prototypes (2-1) and  $\eta = 0.01$



### 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.

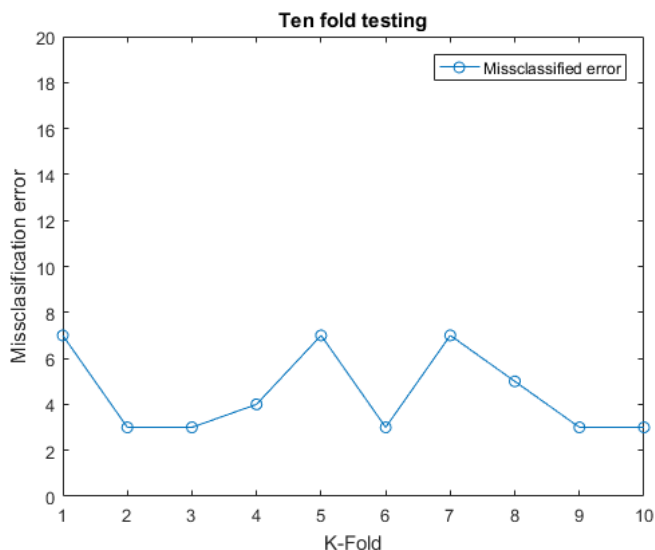


Figure 22: Errors using ten fold validation for (2-1) and  $\eta = 0.01$

### 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

```

1 function LVQlex2(eta)
2 %Load data class_a and class_b
3     load data_lvq_A.mat
4     load data_lvq_B.mat
5
6     %Add category label 0=class A, 1=class B
7     matA(:,3)=0;
8     matB(:,3)=1;
9
10    %Concatenate the two matrices
11    data=vertcat(matA,matB);

```

```

12
13 %Randomly permute the rows, so we have an unbiased training data
14 data2=data(randperm(length(data(:,1))),:);
15
16 %We obtain the final position of the three prototypes after N
17 %epochs, where the variation between epochs becomes smaller than the
18 %threshold
19 [prototype1A, prototype2A, prototype1B, epochNumber, epochError]=LVQeval(data2, eta);
20
21 x=1:length(epochError);
22
23 %Plot of the missclassification training error rate
24 figure
25 plot(x, epochError, '-o');
26 %axis([0 10 0 10]);
27 title('Missclassification training error rate evolution')
28 xlabel('Epoch')
29 ylabel('Missclassification training error rate')
30 legend('Missclassified training error rate');
31 end
32
33 function [prototype1A, prototype2A, prototype1B, epochNumber, epochError]=LVQeval(data, eta)
34 %Randomly generate two prototypes for class A
35 prototype1A=[rand()*4, rand()*9];
36 prototype2A=[6+rand()*4, rand()*9];
37
38 %Randomly generate one prototype for class B
39 prototype1B=[rand()*10, rand()*9];
40
41 %Missclassified training error
42 newMTE=1;
43 MTEdiff=1;
44 variationThreshold=0.0001;
45 errorThreshold=0.25;
46
47 epochNumber=0;
48
49 %Run epochs until the error difference becomes smaller than threshold
50 while(MTEdiff>variationThreshold || newMTE>errorThreshold)
51
52     epochNumber=epochNumber+1;
53     oldMTE=newMTE;
54
55     [newMTE, prototype1A, prototype2A, prototype1B]=epoch(data, prototype1A, prototype2A,
56         prototype1B, eta);
57
58     MTEdiff=abs(oldMTE-newMTE);
59     %We save the missclassified training error for each epoch
60     epochError(epochNumber)=newMTE;
61
62     %Animated plotting
63     scatter(prototype1A(1), prototype1A(2), 20, 'filled');
64     axis([0 10 0 10]);
65     title('LVQ prototype training')
66     xlabel('Feature 1')
67     ylabel('Feature 2')
68     hold on;
69
70     scatter(prototype2A(1), prototype2A(2), 20, 'filled');
71     scatter(prototype1B(1), prototype1B(2), 20, 'filled');
72     legend('Prototype 1(class A)', 'Prototype 2(class A)', 'Prototype 1(class B)');

```

```

72         hold off
73
74         pause(0.10);
75     end
76 end
77
78 function [trainingErrorRate ,prototype1A ,prototype2A ,prototype1B]=epoch(data ,prototype1A ,
    prototype2A ,prototype1B ,eta)
79 %Learning phase
80     mte=0;
81
82     for i=1:1:size(data(:,1))
83
84         point=data(i,1:2);
85
86         %Find the class of the closest prototype
87         closestPrototype=WinnerEuc(point , prototype1A , prototype2A , prototype1B);
88
89         switch closestPrototype
90             case '1A'
91                 %Compare if they belong to the same class
92                 if (data(i,3)==0)
93                     prototype1A=newPosition(prototype1A , point , eta , 1);
94                 else
95                     prototype1A=newPosition(prototype1A , point , eta , -1);
96                     mte=mte+1;
97                 end
98             case '2A'
99                 %Compare if they belong to the same class
100                 if (data(i,3)==0)
101                     prototype2A=newPosition(prototype2A , point , eta , 1);
102                 else
103                     prototype2A=newPosition(prototype2A , point , eta , -1);
104                     mte=mte+1;
105                 end
106             case '1B'
107                 %Compare if they belong to the same class
108                 if (data(i,3)==1)
109                     prototype1B=newPosition(prototype1B , point , eta , 1);
110                 else
111                     prototype1B=newPosition(prototype1B , point , eta , -1);
112                     mte=mte+1;
113                 end
114             end
115         end
116         trainingErrorRate=mte/length(data(:,1));
117     end
118
119 %Return the closest prototype to the point
120 function minD=WinnerEuc(point , prototype1A , prototype2A , prototype1B)
121     d1A=pdist2(point , prototype1A);
122     minD='1A';
123
124     d2A=pdist2(point , prototype2A);
125     if (d2A<d1A)
126         minD='2A';
127     end
128
129     d1B=pdist2(point , prototype1B);
130     if (d1B<d1A & d1B<d2A)
131         minD='1B';

```

```

132     end
133 end
134
135 function w=newPosition(prototype, point, eta, phi)
136     w=prototype+(eta*phi*(point-prototype));
137 end

```

## 4.2 Script for assignment 1, exercise 3

```

1 function LVQ1ex3a(etha,noProtA,noProtB)
2     %Load data class_a and class_b
3     load data_lvq_A.mat
4     load data_lvq_B.mat
5
6     %Add category label 0=class A, 1=class B
7     matA(:,3)=0;
8     matB(:,3)=1;
9
10    %Concatenate the two matrices
11    data=vertcat(matA,matB);
12
13    %Randomly permute the rows, so we have an unbiased training data
14    data2=data(randperm(length(data(:,1))),:);
15
16    %We obtain the final position of the three prototypes after N
17    %epochs, where the variation between epochs becomes smaller than the
18    %threshold
19    [prototype,epochNumber,epochError]=LVQeval(data2, etha,noProtA,noProtB);
20
21    %Plot feature2 vs feature1 of both classes
22    figure
23    scatter(matA(:,1),matA(:,2),20,'DisplayName','Class A');
24    axis([0 10 0 10]);
25    title('Class distribution scatter plot and LVQ prototypes')
26    xlabel('Feature 1')
27    ylabel('Feature 2')
28    hold on;
29
30    scatter(matB(:,1),matB(:,2),20,'DisplayName','Class B');
31
32    for (i=1:length(prototype(:,1)))
33
34        if(prototype(i,3)==0)
35            strLeg='class A prototype';
36        else
37            strLeg='class B prototype';
38        end
39        scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
40    end
41
42    legend('show');
43    hold off;
44
45    x=1:length(epochError);
46
47    %Plot of the missclassification training error rate
48    figure
49    plot(x,epochError,'-o');
50    %axis([0 10 0 10]);
51    title('Missclassification training error rate evolution')

```

```

52 xlabel('Epoch')
53 ylabel('Missclassification training error rate')
54 legend('Missclassified training error rate');
55
56 end
57
58 function [prototype,epochNumber,epochError]=LVQeval(data, etha, noProtA, noProtB)
59 %Randomly generate the prototype matrix, column 3 identifies the class
60 %(A=0, B=1)
61 for (i=1:(noProtA+noProtB))
62     if (i<=noProtA)
63         prototype(i,:)=[rand()*10,rand()*10,0];
64     else
65         prototype(i,:)=[rand()*10,rand()*10,1];
66     end
67 end
68
69 %Missclassified training error
70 newMTE=1;
71 MTEdiff=1;
72 variationThreshold=0.0001;
73 errorThreshold=0.45;
74
75 epochNumber=0;
76
77 %Run epochs until the error difference becomes smaller than threshold
78 %while(MTEdiff>variationThreshold)
79 while(MTEdiff>variationThreshold || newMTE>errorThreshold)
80
81     epochNumber=epochNumber+1
82     oldMTE=newMTE;
83
84     [newMTE,prototype]=epoch(data,prototype,etha);
85     disp(newMTE)
86     MTEdiff=abs(oldMTE-newMTE)
87     %We save the missclassified training error for each epoch
88     epochError(epochNumber)=newMTE;
89
90     strLeg='';
91
92     %Animated plotting
93     for (i=1:length(prototype(:,1)))
94
95         if (prototype(i,3)==0)
96             strLeg='class A';
97         else
98             strLeg='class B';
99         end
100
101         scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
102         axis([0 10 0 10]);
103         title('LVQ prototype training')
104         xlabel('Feature 1')
105         ylabel('Feature 2')
106         hold on;
107
108     end
109
110     legend('show');
111     hold off
112

```

```

113     pause(0.10);
114 end
115 end
116
117 function [trainingErrorRate , prototype]=epoch(data , prototype , etha)
118 %Learning phase
119     mte=0;
120
121     for i=1:1:size(data(:,1))
122
123         point=data(i,1:2);
124
125         %Find the class of the closest prototype
126         closestPrototype=WinnerEuc(point , prototype);
127
128         %Compare if they belong to the same class
129         if(data(i,3)==prototype(closestPrototype,3))
130             prototype(closestPrototype,1:2)=newPosition(prototype(closestPrototype,1:2) ,
131                 point , etha , 1);
132         else
133             prototype(closestPrototype,1:2)=newPosition(prototype(closestPrototype,1:2) ,
134                 point , etha , -1);
135             mte=mte+1;
136         end
137     end
138     trainingErrorRate=mte/length(data(:,1));
139 end
140
141 %Return the closest prototype to the point
142 function closestPrototypeIndex=WinnerEuc(point , prototype)
143     oldD=100;
144     for(i=1:length(prototype(:,1)))
145         d=pdist2(point , prototype(i,1:2));
146         if(d<oldD)
147             closestPrototypeIndex=i;
148             oldD=d;
149         end
150     end
151 end
152
153 function w=newPosition(prototype , point , etha , phi)
154     w=prototype+(etha*phi*(point-prototype));
155 end

```

### 4.3 Script for assignment 2

```

1 function LVQ2(etha , noProtA , noProtB)
2 %Load data class_a and class_b
3     load data_lvq_A.mat
4     load data_lvq_B.mat
5
6     %Add category label 0=class A, 1=class B
7     matA(:,3)=0;
8     matB(:,3)=1;
9
10    %Concatenate the two matrices
11    data=vertcat(matA,matB);
12
13    %Randomly permute the rows, so we have an unbiased training data

```

```

14 data2=data(randperm(length(data(:,1))),:);
15
16
17 %We obtain the final position of the three prototypes after N
18 %epochs, where the variation between epochs becomes smaller than the
19 %threshold
20 [prototype,testingError]=LVQeval(data2, etha,noProtA,noProtB);
21
22 %Plot feature2 vs feature1 of both classes
23 figure
24 scatter(matA(:,1),matA(:,2),20,'DisplayName','Class A');
25 axis([0 10 0 10]);
26 title('Class distribution scatter plot and LVQ prototypes')
27 xlabel('Feature 1')
28 ylabel('Feature 2')
29 hold on;
30
31 scatter(matB(:,1),matB(:,2),20,'DisplayName','Class B');
32
33 for(i=1:length(prototype(:,1)))
34
35     if(prototype(i,3)==0)
36         strLeg='class A prototype';
37     else
38         strLeg='class B prototype';
39     end
40     scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
41 end
42
43 legend('show');
44 hold off;
45
46 x=1:length(testingError);
47
48 %Plot of the missclassification training error rate
49 figure
50 plot(x,testingError,'-o');
51 axis([1 10 0 20]);
52 title('Ten fold testing')
53 xlabel('K-Fold')
54 ylabel('Missclassification error')
55 legend('Missclassified error');
56
57 %The test error is the mean of the classification errors
58 disp(mean(testingError));
59
60 end
61
62 function [prototype,testingError]=LVQeval(data, etha, noProtA, noProtB)
63 %Randomly generate the prototype matrix, column 3 identifies the class
64 %(A=0, B=1)
65 for(i=1:(noProtA+noProtB))
66     if(i<=noProtA)
67         prototype(i,:)=[rand()*10,rand()*10,0];
68     else
69         prototype(i,:)=[rand()*10,rand()*10,1];
70     end
71 end
72
73 indices = crossvalind('Kfold',data(:,3),10);
74 data(:,4)=indices;

```

```

75
76 %Run ten-fold training validatino
77 for (i=1:1:10)
78     trainData=[0 0 0];
79     testData=[0 0 0];
80     for (j=1:1:200)
81         if (data(j,4)~=i)
82             trainData(length(trainData(:,1))+1,:)=data(j,1:3);
83         else
84             testData(length(testData(:,1))+1,:)=data(j,1:3);
85         end
86     end
87
88     [testingError(i),prototype]=epoch(trainData,testData,prototype,etha);
89
90 %Animated plotting
91 for (i=1:length(prototype(:,1)))
92
93     if (prototype(i,3)==0)
94         strLeg='class A';
95     else
96         strLeg='class B';
97     end
98
99     scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
100    axis([0 10 0 10]);
101    title('LVQ prototype training')
102    xlabel('Feature 1')
103    ylabel('Feature 2')
104    hold on;
105
106    end
107
108    legend('show');
109    hold off
110
111    pause(0.10);
112 end
113 end
114
115 function [testingError,prototype]=epoch(data,testData,prototype,etha)
116 %Learning phase
117 testingError=0;
118
119 for i=2:1:size(data(:,1))
120
121     point=data(i,1:2);
122
123     %Find the class of the closest prototype
124     closestPrototype=WinnerEuc(point, prototype);
125
126     %Compare if they belong to the same class
127     if (data(i,3)==prototype(closestPrototype,3))
128         prototype(closestPrototype,1:2)=newPosition(prototype(closestPrototype,1:2),
129             point, etha, 1);
130     else
131         prototype(closestPrototype,1:2)=newPosition(prototype(closestPrototype,1:2),
132             point, etha, -1);
133     end
134 end

```



```

134     for i=2:1:size(testData(:,1))
135         point=testData(i,1:2);
136         %Find the class of the closest prototype
137         closestPrototype=WinnerEuc(point, prototype);
138
139         %Compare if they belong to the same class
140         if(testData(i,3)~=prototype(closestPrototype,3))
141             testingError=testingError+1;
142         end
143     end
144 end
145
146 %Return the closest prototype to the point
147 function closestPrototypeIndex=WinnerEuc(point, prototype)
148     oldD=100;
149     for(i=1:length(prototype(:,1)))
150         d=pdist2(point, prototype(i,1:2));
151
152         if(d<oldD)
153             closestPrototypeIndex=i;
154             oldD=d;
155         end
156     end
157 end
158
159 function w=newPosition(prototype, point, etha, phi)
160     w=prototype+(etha*phi*(point-prototype));
161 end

```

#### 4.4 Script for assignment 3

```

1 function LVQ3(etha, noProtA, noProtB)
2 %Load data class_a and class_b
3     load data_lvq_A.mat
4     load data_lvq_B.mat
5
6 %Add category label 0=class A, 1=class B
7     matA(:,3)=0;
8     matB(:,3)=1;
9
10 %Concatenate the two matrices
11     data=vertcat(matA,matB);
12
13 %Randomly permute the rows, so we have an unbiased training data
14     data2=data(randperm(length(data(:,1))),:);
15
16 %We obtain the final position of the three prototypes after N
17 %epochs, where the variation between epochs becomes smaller than the
18 %threshold
19     [prototype, epochNumber, epochError, testingError, lambda1, lambda2]=LVQeval(data2, etha,
20         noProtA, noProtB);
21
22 %Plot feature2 vs feature1 of both classes
23 figure
24 scatter(matA(:,1),matA(:,2),20,'DisplayName','Class A');
25 axis([0 10 0 10]);
26 title('Class distribution scatter plot and LVQ prototypes')
27 xlabel('Feature 1')
28 ylabel('Feature 2')
29 hold on;

```

```

29 scatter(matB(:,1),matB(:,2),20,'DisplayName','Class B');
30
31
32 for(i=1:length(prototype(:,1)))
33
34     if(prototype(i,3)==0)
35         strLeg='class A prototype';
36     else
37         strLeg='class B prototype';
38     end
39     scatter(prototype(i,1),prototype(i,2),50,'filled','DisplayName',strLeg);
40 end
41
42 legend('show');
43 hold off;
44
45 x=1:length(epochError);
46
47 %Plot of the missclassification training error rate
48 figure
49 plot(x,epochError,'-o');
50 %axis([0 10 0 10]);
51 title('Missclassification training error rate evolution')
52 xlabel('Epoch')
53 ylabel('Missclassification training error rate')
54 legend('Missclassified training error rate');
55
56 %Plot of the classification error
57 x=1:length(testingError);
58 figure
59 plot(x,testingError,'-o');
60 axis([1 10 0 20]);
61 title('Ten fold testing')
62 xlabel('K-Fold')
63 ylabel('Missclassification error')
64 legend('Missclassified error');
65
66 %The test error is the mean of the classification errors
67 disp(mean(testingError));
68
69 %Plot of the final relevances after each epoch
70 x=1:length(lambda1);
71 figure
72 plot(x,lambda1,'-o','DisplayName','Final relevances lambda1');
73 title('GRLVQ results after each epoch')
74 xlabel('Epoch')
75 ylabel('Relevance (lambda)');
76
77 hold on;
78
79 plot(x,lambda2,'-o','DisplayName','Final relevances lambda2');
80 legend('show');
81 hold off;
82
83 end
84
85 function [prototype,epochNumber,epochError,testingError,lambda1E,lambda2E]=LVQeval(data,
86     etha, noProtA, noProtB)
87     %Randomly generate the prototype matrix, column 3 identifies the class
88     %(A=0, B=1)
89     for(i=1:(noProtA+noProtB))

```

```

89         if (i <= noProtA)
90             prototype(i,:) = [rand() * 10, rand() * 10, 0];
91         else
92             prototype(i,:) = [rand() * 10, rand() * 10, 1];
93         end
94     end
95
96     indices = crossvalind('Kfold', data(:, 3), 10);
97     data(:, 4) = indices;
98
99     %Run ten-fold training validation
100    for (i = 1:1:10)
101        trainData = [0 0 0];
102        testData = [0 0 0];
103        for (j = 1:1:200)
104            if (data(j, 4) ~= i)
105                trainData(length(trainData(:, 1)) + 1, :) = data(j, 1:3);
106            else
107                testData(length(testData(:, 1)) + 1, :) = data(j, 1:3);
108            end
109        end
110
111        %Missclassified training error
112        newMTE = 1;
113        MTEdiff = 1;
114        variationThreshold = 0.0001;
115        errorThreshold = 0.25;
116
117        epochNumber = 0;
118
119        %Run epochs until the error difference becomes smaller than threshold
120        %while (MTEdiff > variationThreshold)
121        while (MTEdiff > variationThreshold || newMTE > errorThreshold)
122
123            epochNumber = epochNumber + 1;
124            oldMTE = newMTE;
125
126            [newMTE, prototype, testingError(i), lambda1, lambda2] = epoch(trainData, testData,
127                prototype, etha);
128
129            MTEdiff = abs(oldMTE - newMTE);
130            %We save the missclassified training error for each epoch
131            epochError(epochNumber) = newMTE;
132            lambda1E(epochNumber) = lambda1;
133            lambda2E(epochNumber) = lambda2;
134            strLeg = '';
135
136            %Animated plotting
137            for (i = 1:length(prototype(:, 1)))
138
139                if (prototype(i, 3) == 0)
140                    strLeg = 'class A';
141                else
142                    strLeg = 'class B';
143                end
144
145                scatter(prototype(i, 1), prototype(i, 2), 50, 'filled', 'DisplayName', strLeg);
146                axis([0 10 0 10]);
147                title('LVQ prototype training')
148                xlabel('Feature 1')
149                ylabel('Feature 2')

```

```

149         hold on;
150
151     end
152
153     legend('show');
154     hold off
155
156     pause(0.10);
157 end
158 end
159 end
160
161 function [trainingErrorRate, prototype, testingError, lambda1, lambda2]=epoch(data, testData,
    prototype, etha)
162 %Learning phase
163 mte=0;
164 lambda1=0.5;
165 lambda2=0.5;
166 testingError=0;
167 for i=1:size(data,1)
168
169     point=data(i,1:2);
170
171     %Find the class of the closest prototype
172     closestPrototype=WinnerEuc(point, prototype);
173
174     %Compare if they belong to the same class
175     if(data(i,3)==prototype(closestPrototype,3))
176         [prototype(closestPrototype,1:2), lambda1, lambda2]=newPosition(prototype(
            closestPrototype,1:2), point, etha, 1,lambda1,lambda2,data(i,3));
177     else
178         [prototype(closestPrototype,1:2), lambda1, lambda2]=newPosition(prototype(
            closestPrototype,1:2), point, etha, -1,lambda1,lambda2,data(i,3));
179         mte=mte+1;
180     end
181 end
182 trainingErrorRate=mte/length(data(:,1));
183
184 %Testing phase
185 for i=2:size(testData,1)
186     point=testData(i,1:2);
187     %Find the class of the closest prototype
188     closestPrototype=WinnerEuc(point, prototype);
189
190     %Compare if they belong to the same class
191     if(testData(i,3)~=prototype(closestPrototype,3))
192         testingError=testingError+1;
193     end
194 end
195 end
196
197 %Return the closest prototype to the point
198 function closestPrototypeIndex=WinnerEuc(point, prototype)
199     oldD=100;
200     for(i=1:length(prototype(:,1)))
201         d=pdist2(point, prototype(i,1:2));
202
203         if(d<oldD)
204             closestPrototypeIndex=i;
205             oldD=d;
206         end

```

```

207     end
208 end
209
210 function [w,lambda1,lambda2]=newPosition(prototype , point , etha , phi , lambda1 , lambda2 ,
      class)
211     if ( class==0)
212         w=prototype+(etha*phi*lambda1*(point-prototype));
213         %Calculate the new lambda
214         lambda1=lambda1-etha*phi*pdist2(point , prototype);
215         %Enforce
216         lambda2=1-lambda1;
217     else
218         w=prototype+(etha*phi*lambda2*(point-prototype));
219         %Calculate the new lambda
220         lambda2=lambda2-etha*phi*pdist2(point , prototype);
221         %Enforce
222         lambda1=1-lambda2;
223     end
224 end

```