Assignment 1 Assignment 1

1 Supervised learning: LVQ1

Using the code given in the appendix we created the scatterplot in figure 1.

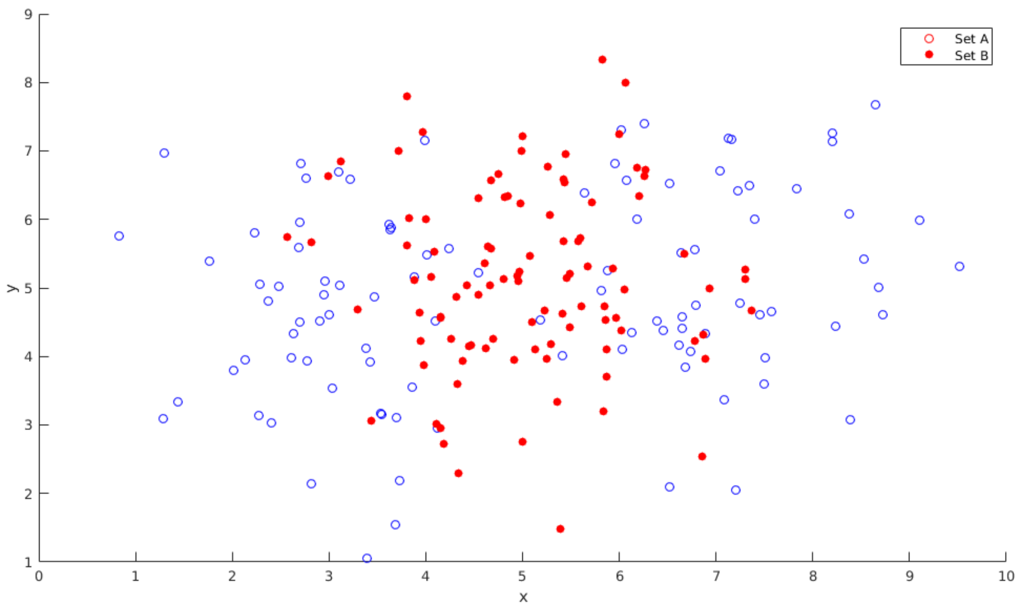


Figure 1: Scatterplot for the two classes.

The plot shows that there are at least three prototypes needed to approach a fairly well classification of these data. Two for set A, which should probably be located around $(3, 4.5)$ and $(7.5, 5.5)$, and one for set B somewhere around $(5, 5)$.

2

The code in the appendix shows our implementation of the LVQ1 algorithm. We acquired the following results for the different settings.

a 1 Prototype for class A and 1 prototype for class B

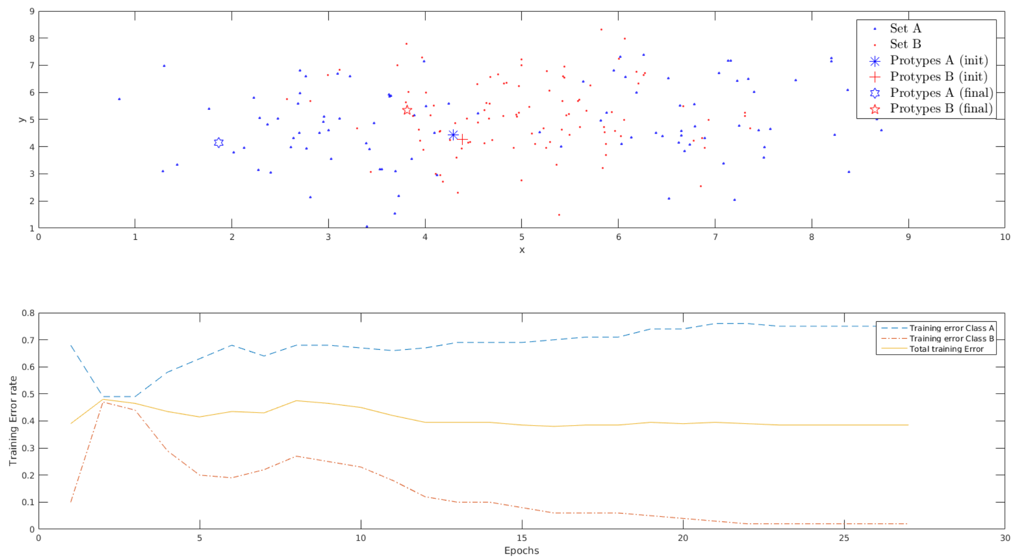


Figure 2: Classification for 1 prototype for class A and 1 prototype for class B. Plot two gives the corresponding error rates (Class 1 is A and class 2 is B).

As is expected with only one prototype per class, figure 2 shows that the prototype for class B is formed quite well, allowing it to correctly classify at least the data points that belong to class B (an error stabilising at around 0.03). However, since class A is distributed in two groups, with B in between them, the prototype for class A is formed at of one of the two outer clusters, which means it can only correctly classify a part of that cluster correctly. The other cluster will be incorrectly classified as belonging to class B, which follows from its error rate which stabilises at around 0.77, meaning that on average 77 out of 100 data points are incorrectly classified, which is quite high. Ultimately this results in a total error rate that stabilises at around 0.4.

b 1 Prototype for class A and 2 prototype for class B

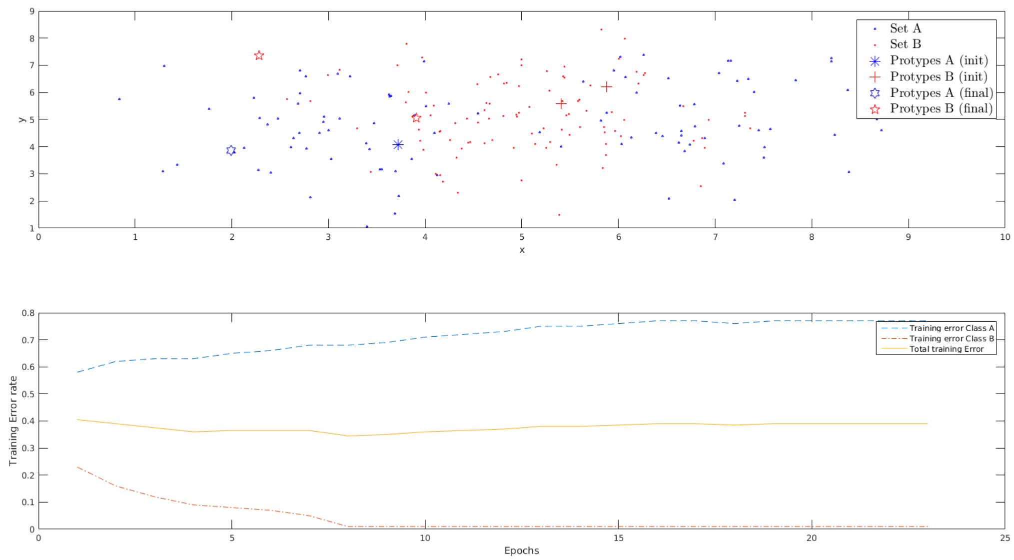
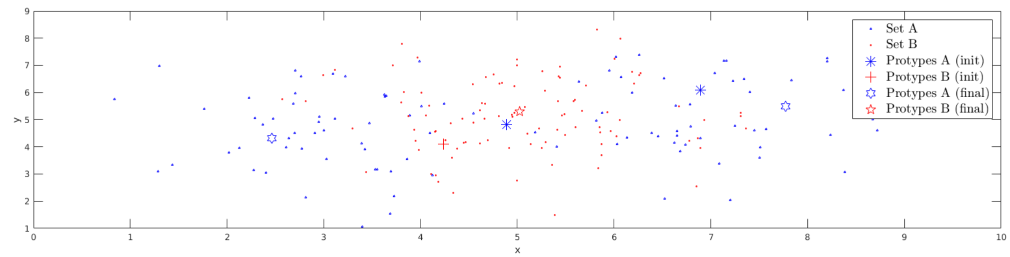


Figure 3: Classification for 1 prototype for class A and 2 prototypes for class B. Plot two gives the corresponding error rates (Class 1 is A and class 2 is B).

In this case figure 3 shows the same thing happening with the prototype for class A: it ends up in one of the two separated A-clusters. Its error is therefore also similar to that in the previous case. On the other hand we see that the two prototypes for class B, even though the data points belonging to class B are clustered in one big cluster, we see that a second prototype slightly improves the result, giving an error that stabilises near 0. This might be the case because it 'catches' some of the last data points that were mixed in with the left-side class A cluster, meaning that it also causes some of the class-B data points in that cluster to be misclassified. This explains the fact that the error of class A has become slightly higher, resulting in a total error that is approximately the same as in the previous case.

c 2 Prototype for class A and 1 prototype for class B



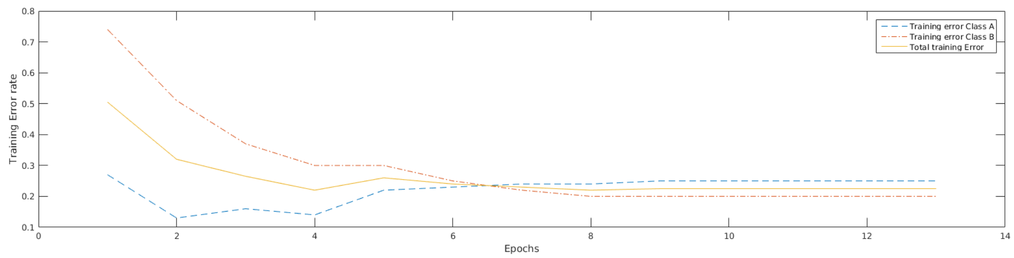


Figure 4: Classification for 2 prototypes for class A and 1 prototype for class B. Plot two gives the corresponding error rates (Class 1 is A and class 2 is B).

In this case figure 4 shows a much lower error rate than in the previous two cases. This is due to the fact that there are now two prototypes for class A, which can classify the seperated data points much better, since they are grouped into two clusters (one prototype for the left cluster, one for the right). Ultimately this results in a total error that stabilises around 0.22, which is significantly lower than in the previous two cases. However, there are also cases in which this amount of prototypes does not result in this error rate. This is the case when the initialization is such that both prototypes for class A begin somewhere near the same cluster of class A, meaning that they will never get to the other cluster, because they get ‘pushed away’ by the data points belonging to class B. In such a case the result yields about the same error rate as the case discussed in the previous paragraph.

d 2 Prototype for class A and 2 prototype for class B

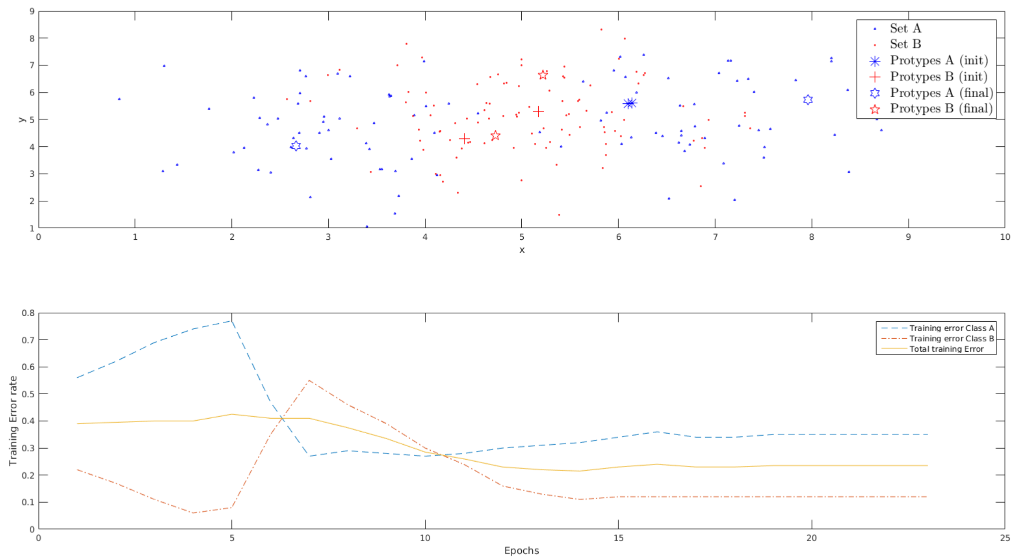


Figure 5: Classification for 2 prototypes for class A and 2 prototypes for class B. Plot two gives the corresponding error rates (Class 1 is A and class 2 is B).

Finally in the case of two prototypes for each class, figure 5 shows slightly worse results than for the previous case, with a total error stabilising at approximately 0.23. This has probably to do with the two prototypes for class B now being divided over the one cluster of data points, which causes them to be slightly nearer to the data points that belong to class A. This causes some of those data points to be misclassified, resulting in a slightly higher error.

Overall we can state that the case for two prototypes for class A and one for class B yields the best results for this particular data set. This is of course also the intuitve explanation, since any human can immediately see that the clusters are distributed in such a way that this distribution of prototypes is needed.

e Learning rate

3 Cross validation

The code in the appendix gives our implementation for this assignment. The computed mean of the 10 values of the classification error is 0.0365. This is quite a low error rate compared to the error rates that were yielded in assignment 1.2(c), which ended up at a total error of around 0.22. This is explained by the fact that for every training phase the algorithm only tests a subset of 20 to the complete set. The probability of errors occuring in that subset is much lower than in the complete training set. Therefore the total average of the error is smaller as well.

4 Relevance LVQ

Appendix

../Code/Ass1_1.m

```
hold off;
hold on;
scatter(matA(:,1),matA(:,2), 'blue');
scatter(matB(:,1),matB(:,2), 'red', 'filled');
xlabel('x'); ylabel('y'); legend('Set A', 'Set B');
```

../Code/Ass1_2.m

```
load('data_lvq_A') % matA
load('data_lvq_B') % matB

close all
subplot(2,1,1)
plot(matA(:,1),matA(:,2), 'b^', 'markersize', 2);
hold on;
plot(matB(:,1),matB(:,2), 'rp', 'markersize', 2);
xlabel('x'); ylabel('y');

data = [matA ; matB];
data_labels = (floor( (0:length(data)-1) * 2 / length(data))).';
data = [data data_labels];

% The prototypes
w_A = 2;
w_B = 2;
w = zeros(w_A + w_B, ndims(data)+1);

eta = 0.01;
nrEpochs = 500;

E_1 = zeros(1,nrEpochs);
E_2 = zeros(1,nrEpochs);

% Randomly initialize the prototypes between the minimum and maximum values
% last value being their class
for i = 1 : size(w,1)
    if i <= w_A
        w(i,:) = [mean(matA) + rand()*2*std(matA)-std(matA) 0];
    else
        w(i,:) = [mean(matB) + rand()*2*std(matB)-std(matB) 1];
    end
end

plot(w(1:w_A,1), w(1:w_A,2), 'b*', 'markersize', 12);
plot(w(w_A+1:size(w,1),1), w(w_A+1:size(w,1),2), 'r+', 'markersize', 12);

for epoch = 1:nrEpochs
    % Training
    for point = 1 : size(data,1)

         % Find the row with the nearest prototype
        rowMin = find(pdist2(data(point,1:2), w(:,1:2)) == min(pdist2(data(point
            ,1:2), w(:,1:2))),1);
        % If the classes of the data point and the nearest prototype are the same
        if w(rowMin,end) == data(point, end)
```

```
                % Move the row closer to the data point
                w(rowMin,1:2) = w(rowMin,1:2) + eta * (data(point,1:2) - w(rowMin,1:2));
            else
                w(rowMin,1:2) = w(rowMin,1:2) - eta * (data(point,1:2) - w(rowMin,1:2));
            end
        end

        % Testing
        for point = 1 : size(data,1)
            % Find the row with the nearest prototype
            rowMin = find(pdist2(data(point,1:2), w(:,1:2)) == min(pdist2(data(point,1:2), w(:,1:2))),1);
            if w(rowMin,end) ~= data(point, end)
                if point <= size(matA,1)
                    E_1(epoch) = E_1(epoch) + 1;
                else
                    E_2(epoch) = E_2(epoch) + 1;
                end
            end
        end
        E = E_1 + E_2;

        if (epoch > 10 && var(E(:,epoch-4:epoch)) < 0.05)
            E_1(:,epoch+1:end) = [];
            E_2(:,epoch+1:end) = [];
            E(:,epoch+1:end) = [];
            break
        end
end

plot(w(1:w_A,1), w(1:w_A,2), 'bh', 'markersize', 12);
plot(w(w_A+1:size(w,1),1), w(w_A+1:size(w,1),2), 'rP', 'markersize', 12);
lgnd = legend('Set A', 'Set B','Protypes A (init)', 'Protypes B (init)','Protypes A (final)', 'Protypes B (final)');
set(lgnd, 'interpreter','latex', 'fontsize', 15);

subplot(2,1,2)
plot(E_1/100, '--')
hold on;
plot(E_2/100, '-.');
plot(E/200);
legend('Training error Class A', 'Training error Class B', 'Total training Error');
xlabel('Epochs')
ylabel('Training Error rate')
```

../Code/Ass2.m

```
load('data_lvq_A') % matA
load('data_lvq_B') % matB

K = 10;

%10 FOLD Cross validation
data = [matA ; matB];
data_labels = (floor( (0:length(data)-1) * 2 / length(data))).';
data = [data data_labels];
```

```
indices = 1:length(data)/K:length(data)+(length(data)/K);
E_K = zeros(1,K);

for fold = 1:K
    train_data = data;
    train_data(indices(fold):indices(fold+1)-1,:) = [];
    test_data  = data(indices(fold):indices(fold+1)-1,:);

    % The prototypes
    w_A = 2;
    w_B = 1;
    w = zeros(w_A + w_B, ndims(data)+1);

    eta = 0.01;
    nrEpochs = 500;

    E_1 = zeros(1,nrEpochs);
    E_2 = zeros(1,nrEpochs);

    % Randomly initialize the prototypes between the minimum and maximum values
    % last value being their class
    for i = 1 : size(w,1)
        if i <= w_A
            w(i,:) = [mean(matA) + rand()*2*std(matA)-std(matA) 0];
        else
            w(i,:) = [mean(matB) + rand()*2*std(matB)-std(matB) 1];
        end
    end

    for epoch = 1:nrEpochs
        % Training
        for point = 1 : size(train_data,1)
            % Find the row with the nearest prototype
            rowMin = find(pdist2(train_data(point,1:2), w(:,1:2)) == min(pdist2(
                train_data(point,1:2), w(:,1:2))),1);
            % If the classes of the train_data point and the nearest prototype are
                the same
            if w(rowMin,end) == train_data(point, end)
                % Move the row closer to the train_data point
                w(rowMin,1:2) = w(rowMin,1:2) + eta * (train_data(point,1:2) - w(
                    rowMin,1:2));
            else
                w(rowMin,1:2) = w(rowMin,1:2) - eta * (train_data(point,1:2) - w(
                    rowMin,1:2));
            end
        end

        % Testing
        for point = 1 : size(train_data,1)
            % Find the row with the nearest prototype
            rowMin = find(pdist2(train_data(point,1:2), w(:,1:2)) == min(pdist2(
                train_data(point,1:2), w(:,1:2))),1);
            if w(rowMin,end) ~= train_data(point, end)
                if point <= size(matA,1)
                    E_1(epoch) = E_1(epoch) + 1;
                else
```

```
                        E_2(epoch) = E_2(epoch) + 1;
                    end
                end
            end
            E = E_1 + E_2;

            if (epoch > 10 && var(E(:,epoch-4:epoch)) < 0.05)
                fold
                E_1(:,epoch+1:end) = [];
                E_2(:,epoch+1:end) = [];
                E(:,epoch+1:end) = [];
                break
            end
        end

        % Now test on the test set
        % Testing
        for point = 1 : size(test_data,1)
            % Find the row with the nearest prototype
            rowMin = find(pdist2(test_data(point,1:2), w(:,1:2)) == min(pdist2(test_data(point,1:2), w(:,1:2))),1);
            if w(rowMin,end) ~= test_data(point, end)
                E_K(fold) = E_K(fold) + 1;
            end
        end
end

% The mean error rate over the 10 folds
mean(E_K)/200
```

../Code/Ass3.m

```
load('data_lvq_A') % matA
load('data_lvq_B') % matB

close all
subplot(3,1,1)
plot(matA(:,1),matA(:,2), 'bp', 'markersize', 2);
hold on;
plot(matB(:,1),matB(:,2), 'rp', 'markersize', 2);
xlabel('x'); ylabel('y');

data = [matA ; matB];
data_labels = (floor( (0:length(data)-1) * 2 / length(data))).';
data = [data data_labels];

% The prototypes
w_A = 2;
w_B = 1;
w = zeros(w_A + w_B, ndims(data)+1);
lambda = [0.5 0.5];

eta = 0.01;
etaL = 0.01;
nrEpochs = 200;

```

```matlab
E_1 = zeros(1,nrEpochs);
E_2 = zeros(1,nrEpochs);
lambdaHist = zeros(2,nrEpochs);

% Randomly initialize the prototypes between the minimum and maximum values
% last value being their class
for i = 1 : size(w,1)
    if i <= w_A
        w(i,:) = [mean(matA) + rand()*2*std(matA)-std(matA) 0];
    else
        w(i,:) = [mean(matB) + rand()*2*std(matB)-std(matB) 1];
    end
end

plot(w(1:w_A,1), w(1:w_A,2), 'b*', 'markersize', 12);
plot(w(w_A+1:size(w,1),1), w(w_A+1:size(w,1),2), 'r+', 'markersize', 12);

for epoch = 1:nrEpochs
    % Save the old lambda values
    lambdaHist(:,epoch) = lambda.';
    % Training
    for point = 1 : size(data,1)
        % Calculate the distances for each prototype to the point
        dist = zeros(1,size(w,1));
        for prot = 1:size(w,1)
            for dim = 1:size(matA,2)
                dist(prot) = dist(prot) + (lambda(dim) * (w(prot, dim) - data(point, dim))^2);
            end
        end

        % Find the row with the nearest prototype
        rowMin = find(dist == min(dist),1);
        % If the classes of the data point and the nearest prototype are the same

        if w(rowMin,end) == data(point, end)
            % Move the row closer to the data point
            w(rowMin,1:2) = w(rowMin,1:2) + eta * (data(point,1:2) - w(rowMin,1:2));
            lambda = lambda - etaL * abs((data(point,1:2) - w(rowMin,1:2)));
        else
            w(rowMin,1:2) = w(rowMin,1:2) - eta * (data(point,1:2) - w(rowMin,1:2));
            lambda = lambda + etaL * abs((data(point,1:2) - w(rowMin,1:2)));
        end

        if lambda(1) < 0
            lambda(1) = 0;
        end
        if lambda(2) < 0
            lambda(2) = 0;
        end
        lambda = lambda / sum(lambda);
    end

    % Testing
    for point = 1 : size(data,1)
        % Find the row with the nearest prototype
```

```matlab
        rowMin = find(pdist2(data(point,1:2), w(:,1:2)) == min(pdist2(data(point,1:2), w(:,1:2))),1);
        if w(rowMin,end) ~= data(point, end)
            if point <= size(matA,1)
                E_1(epoch) = E_1(epoch) + 1;
            else
                E_2(epoch) = E_2(epoch) + 1;
            end
        end
    end
    E = E_1 + E_2;

    if (epoch > 10 && var(E(:,epoch-4:epoch)) < 0.05)
        E_1(:,epoch+1:end) = [];
        E_2(:,epoch+1:end) = [];
        E(:,epoch+1:end) = [];
        lambdaHist(:,epoch+1:end) = [];
        break
    end
end

plot(w(1:w_A,1), w(1:w_A,2), 'bh', 'markersize', 12);
plot(w(w_A+1:size(w,1),1), w(w_A+1:size(w,1),2), 'rP', 'markersize', 12);
lgnd = legend('Set A', 'Set B','Protypes A (init)', 'Protypes B (init)','Protypes A (final)', 'Protypes B (final)');
set(lgnd, 'interpreter','latex', 'fontsize', 15);

subplot(3,1,2)
plot(E_1/200)
hold on;
plot(E_2/200);
plot(E/200);
legend('Training Error Class 1', 'Training Error Class 2', 'Total Training Error');
xlabel('Epochs')
ylabel('Training Error rate')

subplot(3,1,3)
plot(lambdaHist(1,:));
hold on
plot(lambdaHist(2,:));
legend('Lambda 1', 'Lambda 2');
xlabel('Epochs')
ylabel('Lambda value')
```

Pattern Recognition Practical 4

Group 24: Maikel Withagen (s1867733) Steven Bosch (s1861948)

October 8, 2015