RS322: Pattern Recognition
Programming Assignment 3

Atma Anand
SC12B156

Date: April 26, 2015 Department: Physical Sciences

1. K-Nearest Neighbour Classifier

Aim: To classify the given data set using k-Nearest Neighbour Classifier (in MATLAB).

Data Set Used: *Iris Data* – 3 classes, 4 parameters

Theory and Procedure:

(Also refer to the comments written alongside the code)

1. Input the given data and append the original class of the samples to the data.
2. Randomly create two subsets of the data: first the training set and second the test set, for the algorithm. The sets should be non-overlapping (disjoint) and contain equal representation from all classes.
3. Find the 'k' nearest neighbours from the training data for each test sample. This is done by maintaining an ascending order sorted array and comparing the distance of the test sample from each training data set. We assume each parameter to have equal weightage.
4. The no. of samples from each class in the k nearest neighbours is then counted. The data is classified to the class having the largest no. in the k-NN. In case of a tie the data may be classified to any of the tied class.
5. The confusion matrix is prepared and the accuracy of the classification is found out. Confusion matrix states the classified and misclassified test samples in a matrix representation.

The overall algorithm is very simple to implement and execute.

Code: Classifying Iris Data Using k-Nearest Neighbour Classifier

```
% Reading the Iris data
data1 = xlsread('D:\Workspace\Pattern_Recognition\Iris3\irisdata.xls');
class = ones(150,1);
class(51:100) = 2;
class(101:150) = 3;
data = [data1 class];
% Randomly selecting 15 Training data points from each of the 3 Classes
train = zeros(1,45);
for i=1:3
    k=1;
    while k <= 15
        r = (i-1)*50 + randi(50,1);
        f=0;
        for l=(i-1)*15+1:i*15
            if(train(l)==r)
                f=1;
                break;
            end
        end
        if f==0
            train((i-1)*15+k)=r;
            k=k+1;
        end
    end
end
% Randomly selecting 15 Test samples from each of the 3 Classes
% (non-overlapping with the Training Set)
test = zeros(1,45);
for i=1:3
    k=1;
    while k <= 15
        r = (i-1)*50 + randi(50,1);
        f=0;
        for l=(i-1)*15+1:i*15
            if(train(l)==r || test(l)==r)
                f=1;
                break;
            end
        end
        if f==0
            test((i-1)*15+k)=r;
            k=k+1;
        end
    end
end

k = input('Enter value of k for k-NN Classifier: ');

res = [zeros(1,45);zeros(1,45)];
for i=1:45
    res(1,i)=data(test(i),5);
    c1=0; c2=0; c3=0;
    dist = 100*ones(1,k);
    pos = ones(k);
    % Computing the k Nearest Neighbours
    for l=1:45
        d = sum((data(train(l),1:4)- data(test(i),1:4)).^2); %distance
        for m=1:k
            if d<dist(m)
                for n=k:-1:m+1
                    dist(n)=dist(n-1);
                    pos(n)=pos(n-1);
```

```
                  end
                  dist(m)=d;
                  pos(m)=train(l);
                  break;
              end
          end
      end
      % Counting Neighbours from each class
      for m=1:k
          if data(pos(m),5)==1
              c1 = c1+1;
          else if data(pos(m),5)==2
                  c2 = c2+1;
              else
                  c3 = c3+1;
              end
          end
      end
      % Classification
      if c1>c2 && c1>c3
          res(2,i)=1;
      else if c2>c1 && c2>c3
              res(2,i)=2;
          else
              res(2,i)=3;
          end
      end
  end

  % Making the Confusion Matrix
  conf = zeros(3,3);
  for i=1:45
      conf(res(1,i),res(2,i)) = conf(res(1,i),res(2,i))+1;
  end
  display('The Confusion Matrix is:');
  display(conf);
  acc = trace(conf)/45*100; %accuracy
  display(sprintf('Classification Accuracy = %g', acc));
```

Output:

Sample Run: (Different for each Execution)

1.
   ```
   Enter value of k for k-NN Classifier: 1
   The Confusion Matrix is:
   conf =
     15    0    0
      0   15    0
      0    1   14
   Classification Accuracy = 97.7778
   ```

2.
   ```
   Enter value of k for k-NN Classifier: 3
   The Confusion Matrix is:
   conf =
     15    0    0
   ```

```
     0    14     1
     0     1    14
```

3. Classification Accuracy = 95.5556
   ```
   Enter value of k for k-NN Classifier: 6
   The Confusion Matrix is:
   conf =
     15     0     0
      0    15     0
      0     1    14
   Classification Accuracy = 97.7778
   ```

Result:

The given data set was classified into the 3 classes using all four parameters with an accuracy of above 90%.

2. Support Vector Machine:

Aim: To classify the given data set using k-Nearest Neighbour Classifier (in MATLAB).

Data Set Used: *Iris Data* – 3 classes (2 at a time), 2 out of 4 parameters

Theory and Procedure:

(Also refer to the comments written alongside the code)

1. Input the given data and append the original class of the samples to the data.
2. Randomly create two subsets of the data: first the training set and second the test set, for the algorithm. The sets should be non-overlapping (disjoint) and contain equal representation from all classes.
3. Train the SVM using the training data to arrive at parameters using the built in *svmtrain* function.
4. Classify the data from the obtained parameters using the *svmclassify* function. Plot the results.
5. The confusion matrix is prepared and the accuracy of the classification is found out.
6. Confusion matrix states the classified and misclassified test samples in a matrix representation.

The overall algorithm is very simple to implement and execute. Built in functions were used to more easily implement the algorithm.

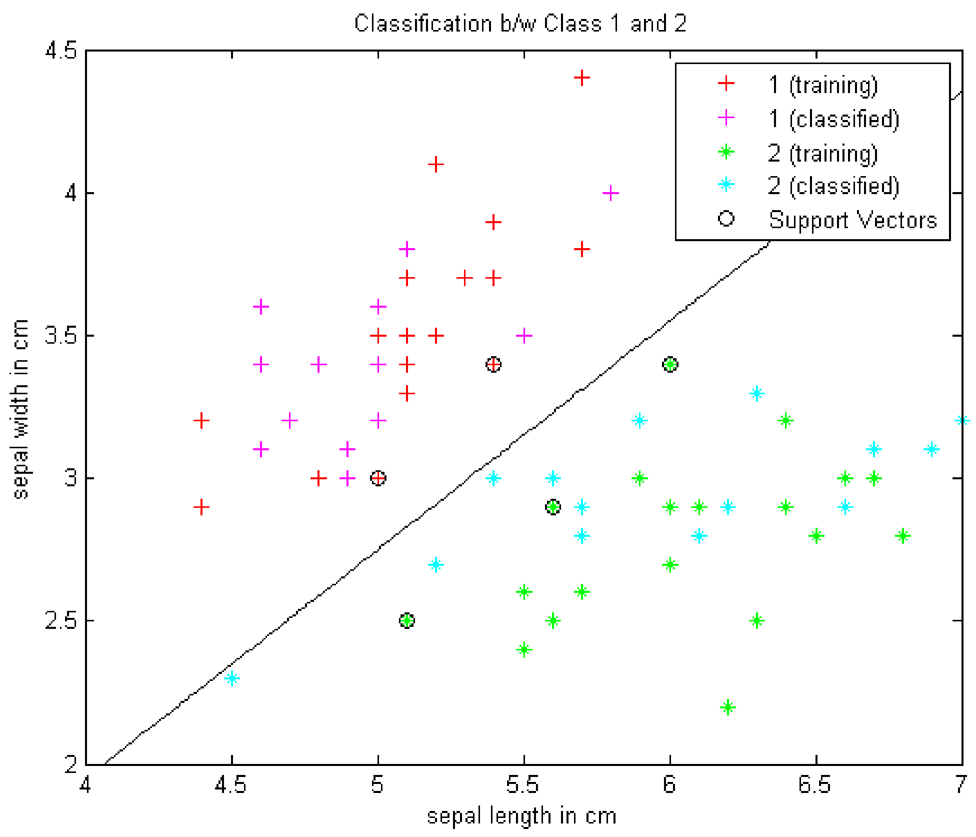
Support Vector Machine

```
% Reading the Iris data
data1 = xlsread('D:\Workspace\Pattern_Recognition\Iris3\irisdata.xls');
class = ones(150,1);
class(51:100) = 2;
class(101:150) = 3;
data = [data1 class];

train = zeros(1,75); test = zeros(1,45);
train1 = zeros(75,5); test1 = zeros(45,5);
% Randomly selecting 25 Training data points from each of the 3 Classes
train = zeros(1,75);
for i=1:3
    k=1;
    while k <= 25
        r = (i-1)*50 + randi(50,1);
        f=0;
        for l=(i-1)*25+1:i*25
            if(train(l)==r)
                f=1;
                break;
            end
        end
        if f==0
            train((i-1)*25+k)=r;
            train1((i-1)*25+k,:)=data(r,:);
            k=k+1;
        end
    end
end
% Randomly selecting 15 Test samples from each of the 3 Classes
% (non-overlapping with the Training Set)
for i=1:3
    k=1;
    while k <= 15
        r = (i-1)*50 + randi(50,1);
        f=0;
        for l=(i-1)*15+1:i*15
            if(train(l)==r || test(l)==r)
                f=1;
                break;
            end
        end
        for l=(i-1)*25+16:i*25
            if(train(l)==r)
                f=1;
                break;
            end
        end
        if f==0
            test((i-1)*15+k)=r;
            test1((i-1)*15+k,:)=data(r,:);
            k=k+1;
        end
    end
end
```

```
% Class 1 and Class 2
% Traing the SVM
svmS = svmtrain(train1(1:50,1:2), train1(1:50,5), 'showplot', true);
% Classifying data based on trained SVM
svmC = svmclassify(svmS, test1(1:30,1:2), 'showplot', true);
xlabel('sepal length in cm');
ylabel('sepal width in cm');
title('Classification b/w Class 1 and 2');
disp('Confusion Matrix:');
disp(confusionmat(test1(1:30,5),svmC));
acc = sum(test1(1:30,5)==svmC)/length(svmC);
display(sprintf('Classification Accuracy = %g', 100*acc));

% Class 2 and Class 3
% Traing the SVM
figure,
svmS = svmtrain(train1(26:75,1:2), train1(26:75,5), 'showplot', true);
% Classifying data based on trained SVM
svmC = svmclassify(svmS, test1(16:45,1:2), 'showplot', true);
xlabel('sepal length in cm');
ylabel('sepal width in cm');
title('Classification b/w Class 2 and 3');
disp('Confusion Matrix:');
disp(confusionmat(test1(16:45,5),svmC));
acc = sum(test1(16:45,5)==svmC)/length(svmC);
display(sprintf('Classification Accuracy = %g', 100*acc));
```

Output:

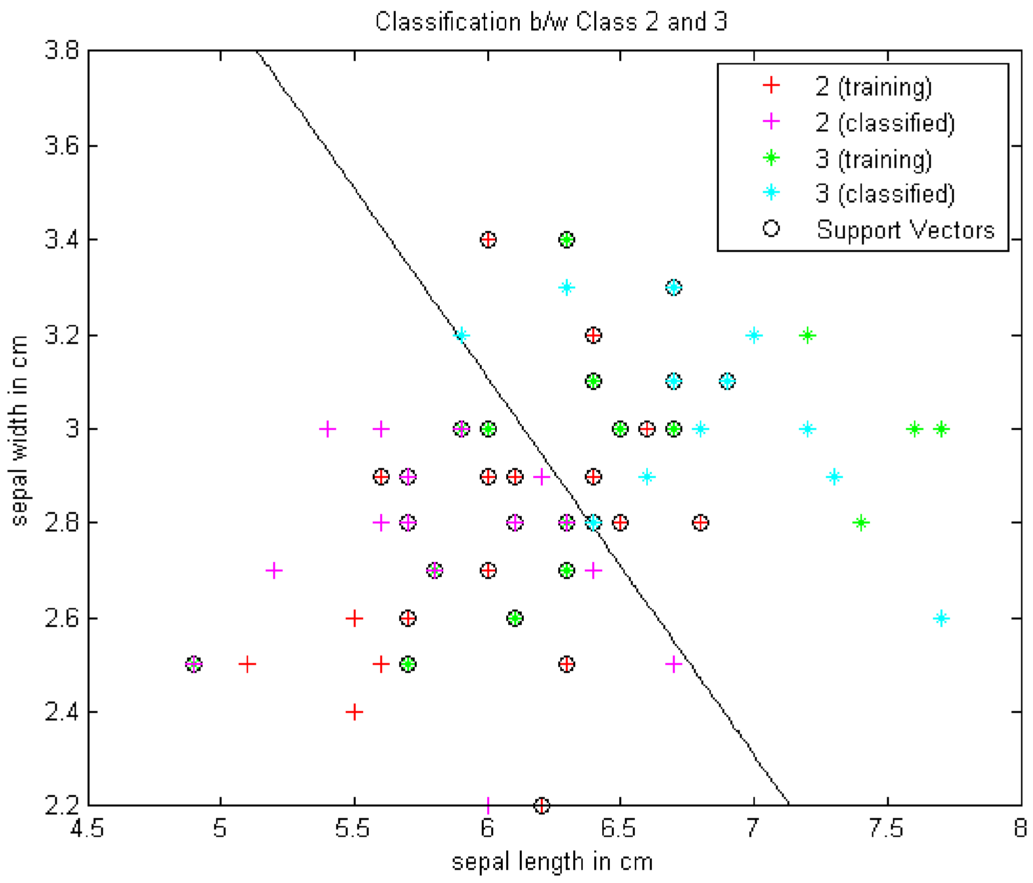
Sample Run: (Different for each Execution)



Confusion Matrix:

```
15   0
 0  15
```

Classification Accuracy = 100%



Confusion Matrix:

```
9   6
3  12
```

Classification Accuracy = 70%

Result:

We note that SVM can only classify two classes and that also having 2 features. Hence its applicability is limited, though it is very efficient and convenient to use in the cases it can be actually applied.

As we can see from the above plots that class 2 and class 3 are not linearly separable while class 1 and class 2 are. The support vectors can also be seen in the figure. The main advantage of using SVM is the margin provided which gives an indication of seperability of classes and accuracy. This is also apparent from the above figures.