

(a) Evidence curve for  $k \in \{1, 2, \dots, 20\}$  = (b) Sign of the discriminant function over the scalar feature space for  $k=2, 5,$  and  $11$ .

Figure K-Nearest Neighbors

## 1 $k$ -Nearest-Neighbor

a. Use the  $k$ -nearest-neighbor method to classify the data (DHS second edition section 4.5.4). As before, the parameter  $k$  is undetermined and must be estimated using an evidence curve  $v(k)$ . For a particular value of  $k$ , compute  $v(k)$  as follows:

- Break the training data set into two parts,  $A$  and  $B$ , where  $B$  contains only a single sample.
- Determine if  $B$  is correctly classified by its  $k$ -nearest-neighbors estimate based on  $A$ .
- Repeat this process for every possible choice of  $B$ . Define  $v(k)$  as the number of times the sample left out was correctly classified.

Plot the evidence curve for odd values of  $k$  in the range 1 to 19. Pick three values of  $k$  from different parts of the curve and plot the resulting discriminant function over the scalar feature space; 1 when choosing the first class, -1 otherwise. Intuitively speaking, what is the best value of  $k$  to use?

b. Design a minimum error rate classifier using the  $k$  that maximizes  $v(k)$ . What is its performance on the test data?

The following MATLAB solves this problem. First it does the leave-one-out validation, the resulting evidence curve can be seen in Figure 4(a). Figure 4(b) plots the sign of the discriminant function for three different values of  $k$ . The resulting confusion matrix on the Test data with the best  $k$  value was as follows:

```

Confusion Matrix for the Test Data
      Recognized as Class 1      Recognized as Class 2
Class 1           858           142
Class 2           287           713
Correct Classification = 78.550000

```

```

% given training data from two classes data1 and data2 of 1-dimensional data
data1 = load('/Users/alockerd/projects/mas622j/ps6/data/problem2/training1.dat');

```

```

data2 = load('/Users/alockerd/projects/mas622j/ps6/data/problem2/training2.dat');
train1 = data1';
train2 = data2';

data1T = load('/Users/alockerd/projects/mas622j/ps6/data/problem2/testing1.dat');
data2T = load('/Users/alockerd/projects/mas622j/ps6/data/problem2/testing2.dat');
test1 = data1T';
test2 = data2T';

%We want to use the knn discriminant technique.
%The degree, k, is undetermined, so we need to estimate this with
%leave-one-out validation:

klimit = 31;
evidence = [];
fprintf(1, '\nComputing Evidence ');
for i=1:size(train1,2)
    fprintf(1, '.');
    A_1=train1; A_1(:,i)=[]; A_2=train2;
    B_1=train1(:,i);

    evidence = [evidence; knn(A_1, A_2, B_1, 1, 1:2:klimit)];
end
fprintf(1, '\n');
for i=1:size(train2,2)
    fprintf(1, '.');
    A_1=train1; A_2=train2; A_2(:,i)=[];
    B_2=train2(:,i);

    evidence = [evidence; knn(A_1, A_2, B_2, -1, 1:2:klimit)];
end

%Get the degree with maximum evidence
krange=1:2:klimit;
[maxevidence maxindex]=max(sum(evidence)); k=krange(maxindex);

plot(1:2:klimit, sum(evidence));
xlabel('k'); ylabel('evidence(k)'); title('Class 1 and 2: K-Nearest Neighbour');
drawnow;

% pick three k values from different parts of the evidence curve
% k=5; k=15; k=20
% for each one plot the sign of the discriminant function for the range of
% the scalar feature space of the data

% determine the scalar range of our data

```

```

x = [min(train1) min(train2) min(test1) min(test2)];
mindata = min(x);
x = [max(train1) max(train2) max(test1) max(test2)];
maxdata = max(x);
r = (mindata-1):5:(maxdata+1);

% plot how the discriminant function would classify the whole range, rough
% sketch of the decision bounds
f5 = [];
f17 = [];
f25 = [];
for t = 1:size(r,2)
    f5 = [f5 knn(train1,train2, r(t), 0, 5)];
    f17 = [f17 knn(train1,train2, r(t), 0, 17)];
    f25 = [f25 knn(train1,train2, r(t), 0, 25)];
end

subplot(3,1,1); plot(r,f5); title('k=5');
subplot(3,1,2); plot(r,f17); title('k=17');
subplot(3,1,3); plot(r,f25); title('k=25');
drawnow;

% check the best k on the test data

fprintf(1,'\nTesting...');
correct1=0; correct2=0;
for i=1:size(test1,2)
    if knn(train1,train2, test1(i), 1, k) == 1
        correct1=correct1+1;
    end
    if knn(train1,train2, test2(i), -1, k) == 1
        correct2=correct2+1;
    end
end
end
fprintf(1,'\n Confusion Matrix for the Test Data\n');
fprintf(1,' Recognized as Class 1          Recognized as Class 2\n');
fprintf(1,'Class 1%d%d\n', correct1, size(test1,2)-correct1);
fprintf(1,'Class 2%d%d\n', size(test2,2)-correct2, correct2);
fprintf(1,'Correct Classification = %f\n', 100*(correct1+correct2)/(size(test1,2)+size(test2,2)));

function correct = knn(A1, A2, B, class_B, k)
%Function uses A1 and A2 as training samples to do KNN classification
%A1 = samples from class 1
%A2 = samples from class -1
%B = Test point
%class_B = Actual Class of B

```

```

%k = the parameter for K-nearest neighbour
%Correct = 1 if B is correctly classified
%Correct = 0 if B is incorrectly classified

correct=[];

dist_A1=abs(A1-B);
dist_A2=abs(A2-B);

for i=1:size(k,2)
    r=0;

    k_class_1=0; k_class_2=0;
    while k_class_1 + k_class_2 < k(i)
        r=r+0.01;
        k_class_1=sum(dist_A1<=r);
        k_class_2=sum(dist_A2<=r);
    end

    if k_class_1 >=k_class_2
        reco_class=1;
    else
        reco_class=-1;
    end

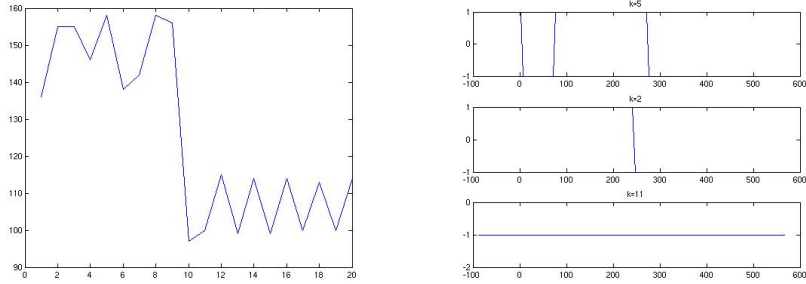
    if class_B==0
        correct=[correct sign(reco_class)];
    elseif reco_class==class_B
        correct=[correct 1];
    else
        correct=[correct 0];
    end
end
end

```

## 2 Generalized Linear Discriminant

- a. Use the generalized linear discriminant (DHS second edition section 5.3) in conjunction with the pseudo-inverse technique (DHS second edition section 5.8.1) to classify the data. That is, for a polynomial of degree  $k$ , let

$$\mathbf{y} = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^k \end{bmatrix}$$



(a) Evidence curve for  $k = \{1, 2, \dots, 20\}$  = (b) Sign of the discriminant function over the scalar feature space for  $k=2, 5,$  and  $11$ .

Figure Generalized Linear Discriminant

be the augmented data vector. Let the margin vector be  $\mathbf{b} = \mathbf{1}$ , so the weight vector  $\mathbf{a}$  can be computed from the pseudoinverse of the data matrix. The degree  $k$  is undetermined and is to be estimated using leave-one-out validation.

To estimate  $k$ , compute the evidence curve  $v(k)$  and then choose the  $k$  that maximizes  $v(k)$ . For a particular value of  $k$ ,  $v(k)$  should be computed as follows:

- Break the training data set into two parts,  $A$  and  $B$ , where  $B$  contains only a single sample.
- Compute the generalized linear discriminant from  $A$  only.
- Determine if  $B$  (the sample left out) is classified correctly.
- Repeat this process for every possible choice of  $B$ . Define  $v(k)$  as the number of times the sample left out was correctly classified.

In this process, it is helpful to multiply the augmented data from the second class by  $-1$  (DHS second edition section 5.4).

Plot the evidence curve for  $k = \{1, 2, \dots, 20\}$ . Pick three values of  $k$  from different parts of the curve and plot the sign of the discriminant function over the scalar feature space; 1 when choosing the first class, -1 otherwise.

- b. Design a minimum error rate classifier using the  $k$  that maximizes  $v(k)$ . What is this classifier's performance on the test data set?

The following MATLAB solves this problem. First it does the leave-one-out validation, the resulting evidence curve can be seen in Figure 5(a). Figure 5(b) plots the sign of the discriminant function for three different values of  $k$ . The resulting confusion matrix on the Test data with the best  $k$  value was as follows:

Confusion Matrix for the Test Data	
Recognized as Class 1	Recognized as Class 2

Class 1	778	222
Class 2	331	669
Correct Classification = 72.350000		

```

% given training data from two classes data1 and data2 of 1-dimensional data
data1 = load('/Users/alockerd/projects/mas622j/ps6/data/problem2/training1.dat');
data2 = load('/Users/alockerd/projects/mas622j/ps6/data/problem2/training2.dat');
train1 = data1';
train2 = data2';
data1T = load('/Users/alockerd/projects/mas622j/ps6/data/problem2/testing1.dat');
data2T = load('/Users/alockerd/projects/mas622j/ps6/data/problem2/testing2.dat');
test1 = data1T';
test2 = data2T';

% We want to use the generalized linear discriminant technique.
% The degree, k, is undetermined, so we need to estimate this with
% leave-one-out validation
degreelimit = 20;
evidence = [];
for i=1:size(train1,2)
    fprintf(1, '.');
    A_1=train1; A_1(:,i)=[]; A_2=train2;
    B_1=train1(:,i);
    evidence=[evidence; linear_discriminant(A_1, A_2, B_1, 1, 1:degreelimit)];
end
    fprintf(1, '\n');
for i=1:size(train2,2)
    fprintf(1, '.');
    A_1=train1; A_2=train2; A_2(:,i)=[];
    B_2=train2(:,i);
    evidence=[evidence; linear_discriminant(A_1, A_2, B_2,-1, 1:degreelimit)];
end

%Get the degree with maximum evidence
degrange=1:degreelimit;
[maxevidence maxindex]=max(sum(evidence));
bestk=degrange(maxindex)

%Plot the evidence curve
figure; plot(1:degreelimit, sum(evidence));
xlabel('Degree of Polynomial'); ylabel('evidence(degree)'); title('Class 1 and 2: Linear Discriminant')
drawnow;

% pick three k values from different parts of the evidence curve
% best: k=5; pretty good: k=2; notso good: k=11
% for each one plot the sign of the discriminant function for the range of

```

```

% the scalar feature space of the data

% determine the scalar range of our data
x = [min(train1) min(train2) min(test1) min(test2)];
mindata = min(x);
x = [max(train1) max(train2) max(test1) max(test2)];
maxdata = max(x);
r = (mindata-1):5:(maxdata+1);

% plot how the discriminant function would classify the whole range,
f5 = [];
f2 = [];
f11 = [];
for t = 1:size(r,2)
    f5 = [f5 linear_discriminant(train1,train2, r(t), 0, 5)];
    f2 = [f2 linear_discriminant(train1,train2, r(t), 0, 2)];
    f11 = [f11 linear_discriminant(train1,train2, r(t), 0, 11)];
end
subplot(3,1,1); plot(r,f5); title('k=5');
subplot(3,1,2); plot(r,f2); title('k=2');
subplot(3,1,3); plot(r,f11); title('k=11');
drawnow;

% check the best k on the test data, k=5
fprintf(1,'\nTesting...');
correct1=0; correct2=0;
for i=1:size(test1,2)
    if linear_discriminant(train1,train2, test1(i), 1, bestk) == 1
        correct1=correct1+1;
    end
    if linear_discriminant(train1,train2, test2(i), -1, bestk) == 1
        correct2=correct2+1;
    end
end
end
fprintf(1,'\n
                Confusion Matrix for the Test Data\n');
fprintf(1,'
                Recognized as Class 1      Recognized as Class 2\n');
fprintf(1,'Class 1  %d  %d\n', correct1, size(test1,2)-correct1);
fprintf(1,'Class 2  %d  %d\n', size(test2,2)-correct2, correct2);
fprintf(1,'Correct Classification = %f\n', 100*(correct1+correct2)/(size(test1,2)+size(test2,2)));

function correct = linear_discriminant(A1, A2, B, class_B, d)
%Function uses A1 and A2 as training samples to compute the linear discriminant
%of the form a0+a1*x+a2*x^2....
%A1 = samples from class 1
%A2 = samples from class -1
%B = Test point

```

```

%class_B = Actual Class of B
% if class_B=0 then correct gives the sign of the discriminant
%d = degree of the polynomial
%Correct = 1 if B is correctly classified
%Correct = 0 if B is incorrectly classified
correct=[];
for i=1:size(d,2)
    %Compute the big Y matrix
    Y=[];
    for n=1:size(A1,2)
        y_t=[1];
        for pwr=1:d(i)
            y_t=[y_t A1(n)^pwr];
        end
        Y=[Y; y_t];
    end
    for n=1:size(A2,2)
        y_t=[-1];
        for pwr=1:d(i)
            y_t=[y_t -A2(n)^pwr];
        end
        Y=[Y; y_t];
    end
    %Compute a using psuedoinverse
    a=pinv(Y)*ones(size(Y,1),1);

    %Test the discriminant on the test point B
    testy_t=[1];
    for pwr=1:d(i)
        testy_t=[testy_t; B^pwr];
    end
    %if class 0, then caller just wants the classification not the test
    if class_B == 0
        correct=[correct sign(testy_t'*a)];
    elseif sign(testy_t'*a)==class_B
        correct=[correct 1];
    else
        correct=[correct 0];
    end
end
end

```