ECE 662

PATTERN RECOGNITION

HW assignment 2

Question 1)

To find Wo using Fisher linear discriminant, we usually use the cost function,

$$J(W) = \frac{W^T S_B W}{W^T S_W W}, \quad W_o = S_w^{-1}(m_1 - m_1) \text{ --- (1)}$$

The given problem is that what is the result if we set up Sw = identity matrix, and find Wo by $W_o = (m_1 - m_1)$.

In equation 1, Sw represents scatter within classes. To answer this problem, here I presented 2 sample classes.

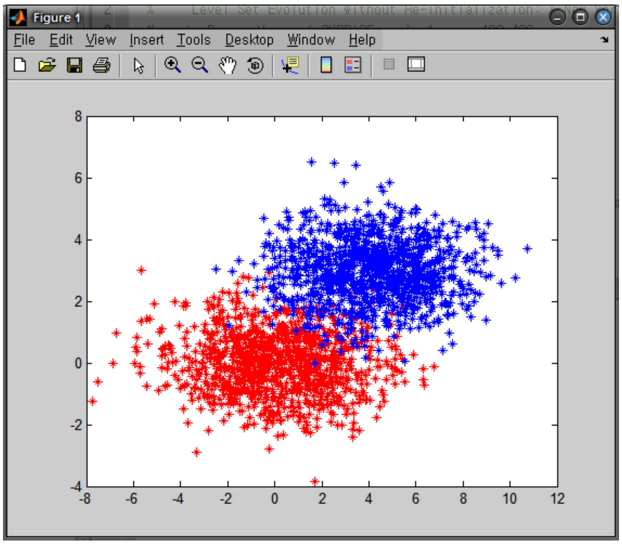
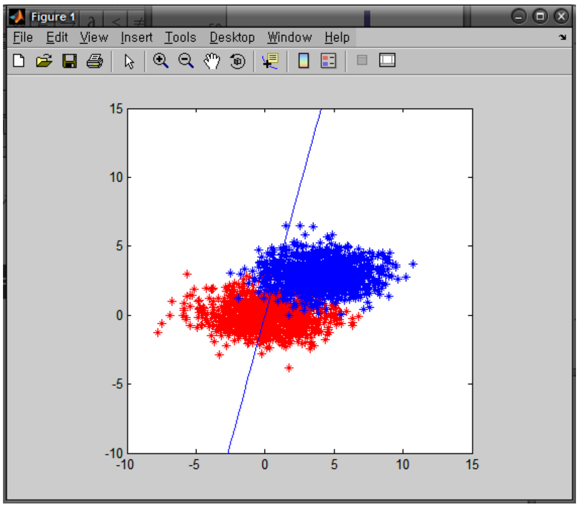


Fig.1 2 Sample classes

To compare the effect of Sw, I calculated Wo using equation (1) and using only (m1-m2).



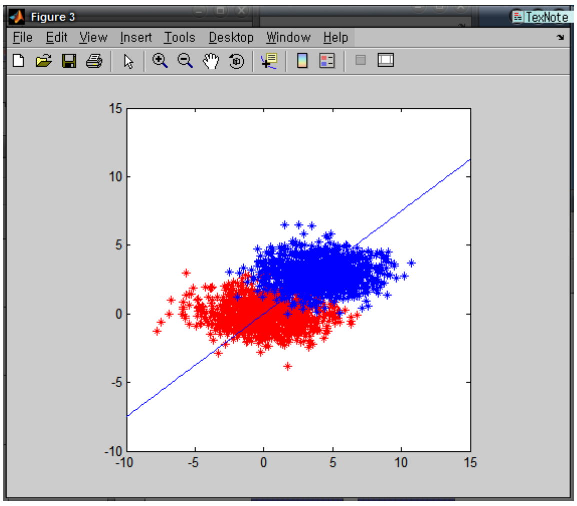


Fig.2 derived Wo by $W_o = S_w^{-1}(m_1 - m_1)$

Fig.3 derived Wo by (m1-m2)

As We can see from above Fig 2.and 3, the resultant Wo direction is different. To see how two different approaches classify well, we can check the histogram along the Wo direction.

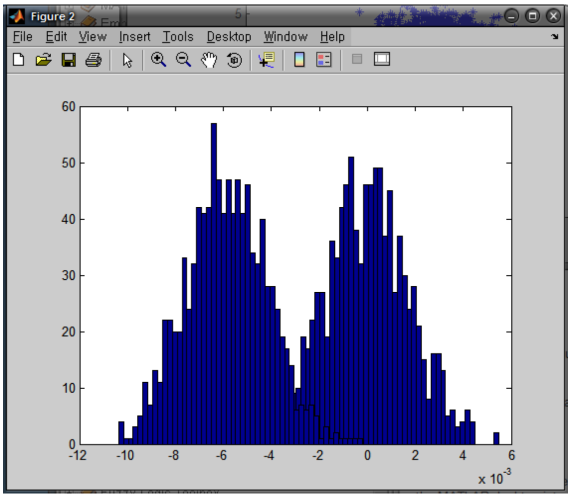


Fig.3 Histogram of Projected onto Wo

by $W_o = S_w^{-1}(m_1 - m_1)$

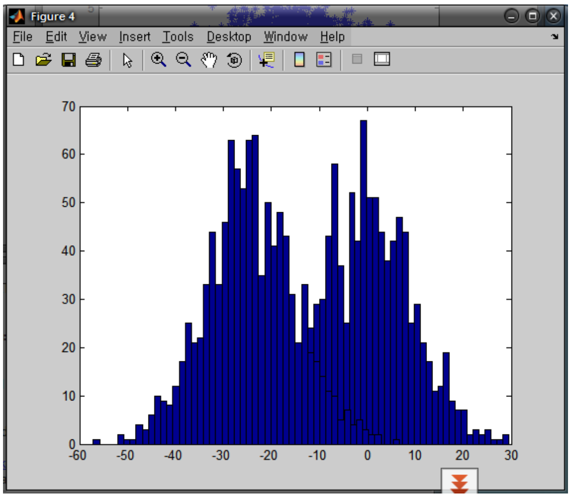


Fig.4 Histogram of Projected onto Wo

by (m1-m2)

As you can see form Fig 3 and 4, the resultant Wo derived using $W_o = S_w^{-1}(m_1 - m_1)$ can divide two classes better than the other method. We can easily check that there exists more overlapping region in Fig 4. If we setup Sw = Idendity matrix, we came to consider only scatter between classes, and find Wo only by using the distance between sample means. In some cases (like this example), this can lead to unreasonable result. We can check from Fig3, that the resultant Wo direction slant to the direction maximizing only the difference between 2 sample means, and this direction also increase the scatter within classes.

Question 2)

To answer the given question 2 (also for Question 3), I downloaded sample data and Package for support vector machine and neural network from website. The following is information about dataset and program package I used.

Data Description

- Webpage : http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#a1a
- # of dimension(feature) : 123
- # of training data : 1605
- # of testing data : 30956
- # of classes : 2(binary), each class identified as -1/+1.
-

Package Description

- LIBSVM(A Library for Support Vector Machine) : http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- FANN(Fast Artificial Neural Network) : http://leenissen.dk/fann/
-

Question 2-a) Neural Network classifier

- The following is the process for this problem.
  - Download and compilation of source code(FANN libraries).
  - Creation of simple C programs for training and classifying data(train / fann_predict).
  - Trains training data.
  - Classification and accuracy report.

- I performed various experiments using different input parameters.
- The following table summarized the various parameters I used for this problem.

# of layers	# of hidden neurons	Desired errors	Maximum epoch number	Accuracy
3	10	0.001	1000	78.7278%
3	100	0.001	1000	79.4288%
3	100	0.001	3000	79.2608%

- As you can see from the above table, when we applied different parameters for Neural Network classifier, there was no significant change of classification accuracy.
-

Question 2-b) Support Vector Machine

- The following is the process for this problem.
  - Download and compilation of source code(LIBSVM).
  - Determination of parameters (C and r)
  - Training and classification using default and derived parameters.
- I used python script "easy.py to find optimal parameters C and r and test

Scaling training data...
Cross validation...
Warning: empty z range [81.8069:81.8069], adjusting to [80.9888:82.625]
Notice: Cannot contour non grid data. Please use "set dgrid3d".
Warning: empty z range [81.8069:81.8069], adjusting to [80.9888:82.625]
Notice: Cannot contour non grid data. Please use "set dgrid3d".
Best c=8192.0, g=3.0517578125e-05 CV rate=83.8006
Training...
Output model: a1a.train.model
Scaling testing data...
Testing...
Accuracy = 83.8287% (25950/30956) (classification)

- Using derived parameters(C = 8192.0 and g = 3.05175), the resultant classification accuracy was 83.8287%
- I also applied default parameters to train and classify data.
-

optimization finished, #iter = 495
4
nu = 0.460268
obj = -673.031393, rho = -0.628569
nSV = 754, nBSV = 722
Total nSV = 754
Accuracy = 83.5864% (25875/30956) (classification)

- With default parameters, the resultant classification accuracy was slightly less than derived optimal parameters.

Question 2-c) Comparison between Support Vector Machine & Artificial Neural Network.

From above experiment, I found the following information.

- Support Vector Machine produced slightly better classification accuracy than ANN approach.

Question 3-b) Classifier using K-nearest neighbor

- The following is process for this problem.
  - Data format conversion from the LIBSM sample data into matrix format using Matlab.
  - Implementation K-nearest neighbor
  - Experiments using the given training and sample dataset
  - Experiments using different K-values.

K	Classification Accuracy	# of correctly classified data out of 30956
2	80.2719 %	24849
3	80.6693 %	24972
4	81.1183 %	25111
5	81.6987 %	25291

- As you can see from the above table, we can easily recognize that the accuracy of KNN classifier increases as K values increases. I attached Matlab code for K-nearest neighbor.

Question 3-c) Classifier using nearest neighbor
For the experiment of nearest neighbor classifier, I just could use same process and same code as K-nearest neighbor, because that nearest neighbor is the special case(K = 1) of K-nearest neighbor. As you can see from below table, the resultant classification accuracy is less than K-nearest neighbor applied using K=2.

K	Classification Accuracy	# of correctly classified data out of 30956
1	78.4339 %	24280

Question 3-d) Comparison between 3 different approaches

From above experiment, I found the following information.
- The classification accuracy increased as K values increased, as I expected.
- The overall time for classification also slightly increased as K values increased.
- The nearest neighbor classifier is the special case for K nearest neighbor(K=1), and leaded to the lower classification accuracy than K nearest neighbor using K=2.

```
%*******************************************************************
%       ECE 662 - Pattern Recognition
%       Homework #2, Prob #1
%       Desc : Implementatio & analysis of Fisher Linear
%                   Discriminant
%*******************************************************************
clear all; clc; format long g;

% numbrt of samples
nSample = 1000;

%===================================
% case 1
% 1. 2 classes having same variance
% 1. apply Sw = I and Sw != I
%===================================

% creation of 2 classes based on normal distribution
%mu1 = [0; 0]; cov1 = [1 0; 0 1];
mu1 = [0; 0]; cov1 = [5 0 ; 0 1];
w1 = mvnrnd(mu1,cov1,nSample);

%mu2 = [6; 3]; cov2 = [1 0; 0 1];
mu2 = [4; 3]; cov2 = [5 0; 0 1];
w2 = mvnrnd(mu2,cov2,nSample);

% Calculation of w0 for Fisher Linear Discriminant
S1 = zeros(2,2); S2 = zeros(2,2);
Sw = zeros(2,2);
for i=1:nSample
        S1 = S1 + (w1(i,:)' - mu1)*(w1(i,:)' - mu1)';
        S2 = S2 + (w2(i,:)' - mu2)*(w2(i,:)' - mu2)';
end
Sw = S1 + S2;

% Calculation of w0 (Sw != I)
w0_1 = inv(Sw)*(mu1-mu2);

% Calculation of w0 (Sw = I)
w0_2 = (mu1-mu2);

% Projection into Line using w0_1
plot(w1(:,1),w1(:,2),'r*');
hold on; plot(w2(:,1),w2(:,2),'b*');
% Draw w0_1 line
wx1 = linspace(-10,15,100);
wy1 = wx1 * w0_1(2,1)/w0_1(1,1);
plot(wx1,wy1);
axis equal;
axis([-10 15 -10 15]);
y1 = w0_1'*w1';
figure,hist(y1,50);
y2 = w0_1'*w2';
hold on, hist(y2,50);


% Projection into Line using w0_2
figure,plot(w1(:,1),w1(:,2),'r*');
hold on; plot(w2(:,1),w2(:,2),'b*');
% Draw w0_2 line
wx2 = linspace(-10,15,100);
wy2 = wx2 * w0_2(2,1)/w0_2(1,1);
plot(wx2,wy2);
axis equal;
axis([-10 15 -10 15]);
y3 = w0_2'*w1';
```

```
figure,hist(y3,50);
y4 = w0_2'*w2';
hold on, hist(y4,50);
```

```
%************************************************************
%       ECE 662 - Pattern Recognition
%       Homework #2, Prob #3
%       Desc : Implementation of KNN
%************************************************************
clear all; clc; format long g;
nSample = 1605;
nTSample = 30956;
% nTSample = 1605;
nDim = 123;

% load training data & test data
fid1 = fopen('D:\Working\a1a.trn');
fid2 = fopen('D:\Working\a1a.tst');

% initialization of matrix for input training & test data
trSample = zeros(nSample, nDim);
tsSample = zeros(nTSample, nDim);

% initialization of matrix for class
trLabel = zeros(nSample,1);
tsLabel = zeros(nTSample,1);

% make training sample matrix from the given LIBSVM sample data set
for i=1:nSample
    line = fgets(fid1);
    % set up class
    class = str2double(line(1:2));
    trLabel(i,1) = class;
    idx = findstr(line,':');
    for j=1:size(idx,2)
        if(line(idx(j)-1) ~= ' ' && line(idx(j)-2) == ' ')
            tmp = str2double(line(idx(j)-1:idx(j)-1));
            trSample(i,tmp) = 1;
        elseif(line(idx(j)-2) ~= ' ' && line(idx(j)-3) == ' ')
            tmp = str2double(line(idx(j)-2:idx(j)-1));
            trSample(i,tmp) = 1;
        elseif(line(idx(j)-3) ~= ' ' && line(idx(j)-4) == ' ')
            tmp = str2double(line(idx(j)-3:idx(j)-1));
            trSample(i,tmp) = 1;
        end
    end
end

% make test sample matrix from the given LIBSVM sample data set
for i=1:nTSample
    line = fgets(fid2);
    % set up class
    class = str2double(line(1:2));
    tsLabel(i,1) = class;
    idx = findstr(line,':');
    for j=1:size(idx,2)
        if(line(idx(j)-1) ~= ' ' && line(idx(j)-2) == ' ')
            tmp = str2double(line(idx(j)-1:idx(j)-1));
            tsSample(i,tmp) = 1;
        elseif(line(idx(j)-2) ~= ' ' && line(idx(j)-3) == ' ')
            tmp = str2double(line(idx(j)-2:idx(j)-1));
```

```
            tsSample(i,tmp) = 1;
        elseif(line(idx(j)-3) ~= ' ' && line(idx(j)-4) == ' ')
            tmp = str2double(line(idx(j)-3:idx(j)-1));
            tsSample(i,tmp) = 1;
        end
    end
end

% Call K-nearest neighbor(K=1->nearest neighbir) function
classM = wkKnn(trSample',trLabel',tsSample',3);

% accuracy test
nCorrect = 0;
for i=1:nTSample
    if((tsLabel(i,1)-classM(1,i)) == 0)
        nCorrect = nCorrect + 1;
    end
end

disp(nCorrect/nTSample * 100);

fclose(fid1);
fclose(fid2);
```

```
%**************************************************************
%       Train.c for FANN
%**************************************************************

#include <stdio.h>
#include <stdlib.h>
#include "fann.h"

int main(int argc, char *argv[])
{
   /* Syntax: train trainFile outFile numInput numOutput numLayers
                numNeuronsHidden desiredError maxEpoch epochBetweenReport */
   unsigned int num_input = atoi(argv[3]);
   unsigned int num_output = atoi(argv[4]);
   unsigned int num_layers = atoi(argv[5]);
   unsigned int num_neurons_hidden = atoi(argv[6]);
   float desired_error = atof(argv[7]);
   unsigned int max_epochs = atoi(argv[8]);
   unsigned int epochs_between_reports = atoi(argv[9]);

   struct fann *ann = fann_create_standard(num_layers, num_input, num_neurons_hidden, num_output);

   fann_set_activation_function_hidden(ann, FANN_SIGMOID_SYMMETRIC);
   fann_set_activation_function_output(ann, FANN_SIGMOID_SYMMETRIC);

   fann_train_on_file(ann, argv[1], max_epochs, epochs_between_reports, desired_error);

   fann_save(ann, argv[2]);
   fann_destroy(ann);

   return 0;
}

%**************************************************************
%       fann_predict.c for FANN
%**************************************************************

/*
** fann_predict.c
**
** Made by (Jinha Jung)
** Login     <jinha@jinha-laptop.ecn.purdue.edu>
**
** Started on    Thu Mar 27 17:30:56 2008 Jinha Jung
** Last update Sun May 12 01:17:25 2002 Speed Blue
*/

#include <stdio.h>
#include <stdlib.h>
#include "floatfann.h"

/* Syntax
     fann_predict testFile netFile */
int main(int argc, char *argv[])
{
   int lengthTestData;
   int numCorrect = 0;
```

```
int class;
float temp;
int i;
fann_type *calc_out;

/* Create FANN structure */
struct fann *ann = fann_create_from_file(argv[2]);
struct fann_train_data *testData = fann_read_train_from_file(argv[1]);

lengthTestData = fann_length_train_data(testData);

//printf("Length of Test data = %d\n", lengthTestData);

for (i = 0; i < lengthTestData; i++) {
        /* Initialize the result class */
        class = 0;
        /* Go through fann */
        calc_out = fann_run(ann, testData->input[i]);
        /* If result is positive class = 1, otherwise class=-1 */
        if (calc_out[0] > 0) {
            class = 1;
        } else {
            class = -1;
        }
        //printf("%d data is classified to %d\n", i, class);

        /* Check whether it is correctly classified */
        if (class == testData->output[i][0]) {
            numCorrect = numCorrect + 1;
            //printf ("Number of correct = %d\n", numCorrect);
        }
}

printf("Testing finished.\n");
printf("Number of test data = %d\n", lengthTestData);
printf("Number of correctly classified data = %d\n", numCorrect);
printf("Accuracy = %f percent\n",
                (float)numCorrect/(float)lengthTestData*100);

fann_destroy_train(testData);
fann_destroy(ann);

return 0;
```