Homework 3 – Machine Learning

Saeid Hosseinipoor

Problem 1 – Mean, Covariance, and PCA

Data were produced according to the given instruction in the problem statement. The following figure illustrates the generated random points.

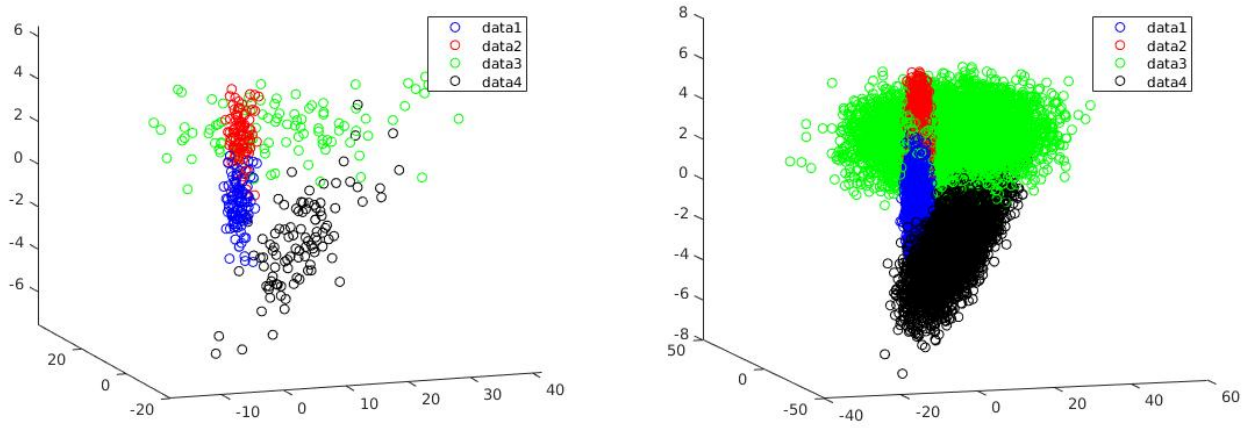


Figure 1: 100(a) and 10000(b) Random Data Points

Question 1:

The following table summarizes the mean and covariance values for four sets of each run:

variable	n	mean	covariance
data	100	[-0.1225 0.0505 0.0370]	[1.2659 0.0748 0.0314 0.0748 0.9181 0.0858 0.0314 0.0858 1.1896]
	10000	[-0.0013 0.0191 -0.0172]	1.0099 -0.0176 0.0086 -0.0176 0.9940 -0.0260 0.0086 -0.0260 1.0232
data1	100	[0.8775 2.0505 3.0370]	[1.2659 0.0748 0.0314 0.0748 0.9181 0.0858 0.0314 0.0858 1.1896]

	10000	[0.9987 2.0191 2.9828]	[1.0099 -0.0176 0.0086 -0.0176 0.9940 -0.0260 0.0086 -0.0260 1.0232]
data2	100	[8.7753 6.1516 3.0370]	[126.5855 2.2434 0.3142 2.2434 8.2632 0.2573 0.3142 0.2573 1.1896]
	10000	[9.9867 6.0572 2.9828]	[100.9874 -0.5282 0.0864 -0.5282 8.9461 -0.0780 0.0864 -0.0780 1.0232]
Data3	100	[10.6406 2.5819 -2.0436]	[62.8396 58.4226 8.4999 58.4226 70.7219 9.3696 8.4999 9.3696 2.4737]
	10000	[11.3720 3.5372 -1.8651]	[49.0915 45.1804 6.9153 45.1804 59.7814 7.5256 6.9153 7.5256 2.0819]

Question 2:

Vector V_2, the directions of maximum variance for data2, and V_3, the directions of maximum variance for data3, are in different direction as expected. V_3 is the rotated form of V_2 by R.

$$[0.6609 \quad 0.7434 \quad 0.1028]^T = R * [\ 1.0000 \quad -0.0057 \quad 0.0009]^T$$

Question 3:

The original data set is randomly distributed. We applied three transformation on the original data set. The first transformation is just a translation in 3D space. The second transformation stretches the data in each dimension with different scales. We expect to have higher variance in first dimension which has bigger multiplier. Therefore the first principle component for data2 is expected to be $[1\ 0\ 0]^T$. The third and the last transformation is a rotation operation which rotates the principle component. So we expect to have

$$R * [1\ 0\ 0]^T = [0.6651 \quad 0.7250 \quad 0.1035]^T$$

which is very close to the calculated value $[0.6609 \quad 0.7434 \quad 0.1028]^T$ for 10000 data. More data results better estimation respect to theory.

Question 4:

Data1 through data3 are generated from original random data by linear operations. Therefore I have expected to see the same transformation between the statistical parameters. The observation and experiments agree with theory. As examples I expected to see the following correlations:

mean of data = $[0\pm\varepsilon_1\ 0\pm\varepsilon_2\ 0\pm\varepsilon_3]$

mean of data1 = $[1\ 2\ 3]^T$ + mean of data

mean of data2 = diag([10 3 1]) + mean of data1

mean of data3 = R * mean of data2

covariance of data = $[1\pm\varepsilon_{11}\ 0\pm\varepsilon_{12}\ 0\pm\varepsilon_{13};\ 0\pm\varepsilon_{21}\ 1\pm\varepsilon_{22}\ 0\pm\varepsilon_{23};\ 0\pm\varepsilon_{31}\ 0\pm\varepsilon_{32}\ 1\pm\varepsilon_{33}]$

covariance of data1 = $[1\ 2\ 3]^T$ + covariance of data

covariance of data2 = diag([10 3 1]).^2 + covariance of data1

covariance of data3 = R * covariance of data2

PC of data depends on the distribution likely $[1\pm\varepsilon_{11}\ 0\pm\varepsilon_{12}\ 0\pm\varepsilon_{13};\ 0\pm\varepsilon_{21}\ 1\pm\varepsilon_{22}\ 0\pm\varepsilon_{23};\ 0\pm\varepsilon_{31}\ 0\pm\varepsilon_{32}\ 1\pm\varepsilon_{33}]$

PC of data1 = PC of data

PC of data2 = $[1\ 0\ 0]^T$

PC of data3 = R * PC of data2

Problem 2 – PCA

To calculate principal component for a data set, we can follow two different ways; first use pca() function in MATLAB or other scripting languages, or find the mean removed covariance of the data set and find the eigen value and vectors, then pick the vectors related to the higher eigen values first until take enough dimensions. I have implemented the difficult way and checked it with the first approach and found that results are almost (but not exactly) the same. I chose these vectors:

0.0020	0.0226	-0.0225
-0.0978	0.9722	0.1434
-0.0161	0.1419	-0.9225
-0.0608	-0.0579	-0.3070
-0.9931	-0.0946	0.0210
-0.0140	0.0470	-0.1324
-0.0005	0.0008	-0.0006
0.0036	0.1402	-0.1255

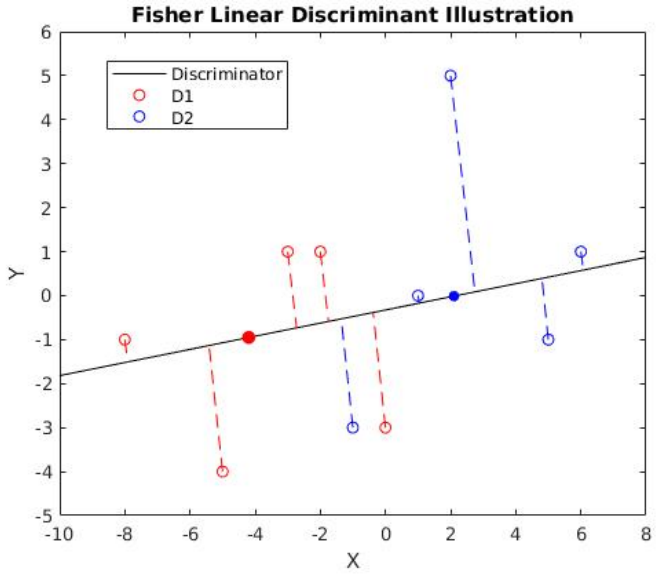
Then I mapped the original 8D data to the new 3D space. I have implemented PCA to this problem by reducing the dimensions using training data and then used the calculated vector for test data to map it into the new space.

The mean accuracy was around %74 and variance of accuracy was 0.02 which is very good.

Problem 3 – Fisher Linear Discriminant

I have applied the procedure in slides 21-22 Lecture 6 and all the steps are in the MATLAB file. The following figure also illustrates the results:

Figure



One point ($[-1\ -3]^T$) is misclassified as could be seen in the figure.

Problem 4 – FLD and MLE

For this section all the features in Pima Indians Diabetes data base have been used. Half of data separated randomly as training data set and FLD method was applied to establish the model. The training set trained by MLE to estimate the transformed data. Then the discernment line vector was used to map the test data set and test them by MLE classifier. As expected, the results were slightly better than MLE on features before FLD transformation. The mean accuracy for FLD was %77 versus %74 for pure MLE.

Problem 5 - Perceptron

Part (a)

If we have a linear discriminant function in form of $g(x) = w^T y$; then $g(x) > 0$ for all y in class 1, and $g(x) < 0$ for all y in class 2 before normalization. By multiplying observations in the second class by negative unit number, we can reduce the problem to $g(x) >$ for all y.

Part (b)

The problem was solved as requested in the problem statement and the step by step solution is available in MATLAB file.

Appendix 1 – MATLAB Code

```
clear

close all
clc

TheRoutineFor(100)
TheRoutineFor(10000)

function TheRoutineFor(n)

    data = randn(3,n);
    m = [1 2 3]';
    data1 = data + repmat(m,1,n);
    data2 = diag([10 3 1]) * data1;
    R = [0.6651 0.7427 0.0775; 0.7395 -0.6696 0.0697; 0.1037 0.0109 -0.9946];
    data3 = R * data2;

    m_data = mean(data')
    cv_data = cov(data'-repmat(m_data,n,1))

    m_data1 = mean(data1')
    cv_data1 = cov(data1'-repmat(m_data1,n,1))

    m_data2 = mean(data2')
    cv_data2 = cov(data2'-repmat(m_data2,n,1))

    m_data3 = mean(data3')
    cv_data3 = cov(data3'-repmat(m_data3,n,1))


    [coef,~,latent] = pca(data');
    [coef1,~,latent1] = pca(data1');
    [coef2,~,latent2] = pca(data2');
    [coef3,~,latent3] = pca(data3');

    figure; hold on
    plot3(data(1,:), data(2,:), data(3,:),'bo')
    plot3(data1(1,:), data1(2,:), data1(3,:),'ro')
    plot3(data2(1,:), data2(2,:), data2(3,:),'go')
    plot3(data3(1,:), data3(2,:), data3(3,:),'ko')

end




clear

close all
clc

%rng(1000);
```

```
tic

data = dlmread('pima-indians-diabetes.data.txt');
data = reshape(data,[],9);

% parameters
split_train_set = 0.5;
ML_accuracy = zeros(1,10);

% pick the new features
active_feat = 1:3;

% Maximum likelihood method
for i = 1:10

    % Random selection
    rp = randperm(length(data));
    data=data(rp,:);

    train_data = data(1:floor(length(data) * split_train_set),:);
    test_data = data(floor(length(data) * split_train_set)+1:end,:);



    % PCA transformation for training set
    X = train_data(:,1:end-1);
    m = mean(X);
%     cv = cov(X -repmat(m,length(X),1));
    cv = (X -repmat(m,length(X),1))' * (X -repmat(m,length(X),1));
    [V,D] = eig(cv);
    [B,I] = sort(diag(D),'descend');
    pcaVectors = V(:,I(1:8));
    %pcaVectors = pca(train_data(:,1:end-1));
    train_data = [train_data(:,1:end-1) * pcaVectors(:,1:3) ...
        train_data(:,end)];

     % PCA transformation for test set
    test_data = [test_data(:,1:end-1) * pcaVectors(:,1:3) ...
        test_data(:,end)];



    % Maximum likelihood method
    [correct, wrong] = ...
        MaximumLikelihood (train_data, test_data, active_feat);
    ML_accuracy(i) = correct / (correct+wrong);

end

ML_mean = mean(ML_accuracy)
ML_SD = std(ML_accuracy)


toc
```

```
function [correct, wrong] = MaximumLikelihood (train_data, test_data, active_feat)

    % training
    mean_0 = mean(train_data(train_data(:,end)==0,active_feat))';
    mean_1 = mean(train_data(train_data(:,end)==1,active_feat))';
    covar_0 = cov(train_data(train_data(:,end)==0,active_feat));
    covar_1 = cov(train_data(train_data(:,end)==1,active_feat));
    prior0tmp = length(train_data(train_data(:,end)==0));
    prior1tmp = length(train_data(train_data(:,end)==1));
    prior_0 = prior0tmp./(prior0tmp+prior1tmp);
    prior_1 = prior1tmp./(prior0tmp+prior1tmp);

    % testing
    d = length(active_feat);
    lklhood_0 = exp(-0.5.*(test_data(:,active_feat)'-mean_0)' * inv(covar_0) ...
        * (test_data(:,active_feat)'-mean_0))' ...
        ./ ((2.*pi).^(d/2).*det(covar_0)^0.5); %...
    lklhood_0 = diag(lklhood_0);
    lklhood_1 = exp(-0.5.*(test_data(:,active_feat)'-mean_1)' * inv(covar_1) ...
        * (test_data(:,active_feat)'-mean_1))' ...
        ./ ((2.*pi).^(d/2).*det(covar_1)^0.5); %...
    lklhood_1 = diag(lklhood_1);

    post_0 = prod(lklhood_0.*prior_0, 2);
    post_1 = prod(lklhood_1.*prior_1, 2);

    correct = sum((post_0(test_data(:,end) == 0) > post_1(test_data(:,end) == 0)));
    wrong = sum(~(post_0(test_data(:,end) == 0) > post_1(test_data(:,end) == 0)));

    correct = correct + ...
        sum((post_1(test_data(:,end) == 1) >= post_0(test_data(:,end) == 1)));
    wrong = wrong + ...
        sum(~(post_1(test_data(:,end) == 1) >= post_0(test_data(:,end) == 1)));

end




clear

close all
clc

rng(1000);
tic

D1 = {[-2 1], [-5 -4], [-3 1], [0 -3], [-8 -1]};
D2 = {[2 5], [1 0], [5 -1], [-1 -3], [6 1]};

n1 = length(D1);
n2 = length(D2);
```

```
X1 = zeros(n1,2);
X2 = zeros(n2,2);

for i = 1:n1
    X1(i,:) = D1{i};
end

for i = 1:n2
    X2(i,:) = D2{i};
end

fprintf('Class 1 means is: ')
mu1 = mean(X1)
fprintf('Class 2 means is: ')
mu2 = mean(X2)

S1 = (n1-1) * cov(X1);
S2 = (n2-1) * cov(X2);

fprintf('\n\nWithin-class scatter is: ')
Sw = S1 + S2

fprintf('\n\nBetween-class scatter is: ')
mu = mean([X1;X2]);
SB = n1 * (mu1 - mu)' * (mu1 - mu) + n2 * (mu2 - mu)' * (mu2 - mu)


fprintf('\n\nThe optimal line direction is: ')
V = Sw \ (mu1 - mu2)'

if sum(X1*V) < 0
    fprintf('\n\nThe first set class discriminats should be negative: ')
    X1*V
    fprintf('\n\nThe second set class discriminats should be positive: ')
    X2*V
else
    fprintf('\n\nThe first set class discriminats should be positive: ')
    X1*V
    fprintf('\n\nThe second set class discriminats should be negative: ')
    X2*V
end


% This part is just for illustration and was not asked in problem statement
V = V / norm(V) * 10;

plot([-10 10], [-V(1)/V(2)*(-10) -V(1)/V(2)*(10)], 'k')
plot([-V(1)+mu(1) V(1)+mu(1)], [-V(2)+mu(2) V(2)+mu(2)], 'k')

hold on
plot(X1(:,1), X1(:,2),'ro')
plot(X2(:,1), X2(:,2),'bo')

for i = 1:length(X1)
    plot([X1(i,1) (X1(i,:)-mu(1))*V/norm(V)^2*V(1)+mu(1)], ...
```

```
        [X1(i,2) (X1(i,:)-mu(2))*V/norm(V)^2*V(2)+mu(2)], 'r--')
end

for i = 1:length(X2)
    plot([X2(i,1) (X2(i,:)-mu(1))*V/norm(V)^2*V(1)+mu(1)], ...
        [X2(i,2) (X2(i,:)-mu(2))*V/norm(V)^2*V(2)+mu(2)], 'b--')
end

c1 = mean(X1*V);
c2 = mean(X2*V);
plot(c1/norm(V)^2*V(1)+mu(1), c1/norm(V)^2*V(2)+mu(2), ...
    'r.','MarkerSize',25)
plot(c2/norm(V)^2*V(1)+mu(1), c2/norm(V)^2*V(2)+mu(2), ...
    'b.', 'MarkerSize',20)


toc




clear

close all
clc

%rng(1000);
tic

data = dlmread('pima-indians-diabetes.data.txt');
data = reshape(data,[],9);

% parameters
split_train_set = 0.5;
ML_accuracy = zeros(1,10);
FLD_accuracy = zeros(1,10);


% pick the new features
active_feat = 1:8;

% Maximum likelihood method
for i = 1:10

    % Random selection
    rp = randperm(length(data));
    data=data(rp,:);

    train_data = data(1:floor(length(data) * split_train_set),:);
    test_data = data(floor(length(data) * split_train_set)+1:end,:);


    % Maximum likelihood method
    [correct, wrong] = ...
        MaximumLikelihood (train_data, test_data, active_feat);
```

```
    ML_accuracy(i) = correct / (correct+wrong);


    % Fisher Linear Discriminant
    [correct, wrong] = ...
        FisherLinearDiscreminant (train_data, test_data, active_feat);
    FLD_accuracy(i) = correct / (correct+wrong);

end

ML_mean = mean(ML_accuracy)
ML_SD = std(ML_accuracy)

FLD_mean = mean(FLD_accuracy)
FLD_SD = std(FLD_accuracy)


toc




function [correct, wrong] = MaximumLikelihood (train_data, test_data, active_feat)

    % training
    mean_0 = mean(train_data(train_data(:,end)==0,active_feat))';
    mean_1 = mean(train_data(train_data(:,end)==1,active_feat))';
    covar_0 = cov(train_data(train_data(:,end)==0,active_feat));
    covar_1 = cov(train_data(train_data(:,end)==1,active_feat));
    prior0tmp = length(train_data(train_data(:,end)==0));
    prior1tmp = length(train_data(train_data(:,end)==1));
    prior_0 = prior0tmp./(prior0tmp+prior1tmp);
    prior_1 = prior1tmp./(prior0tmp+prior1tmp);

    % testing
    d = length(active_feat);
    lklhood_0 = exp(-0.5.*(test_data(:,active_feat)'-mean_0)' * inv(covar_0) ...
        * (test_data(:,active_feat)'-mean_0))' ...
        ./ ((2.*pi).^(d/2).*det(covar_0)^0.5); %...
    lklhood_0 = diag(lklhood_0);
    lklhood_1 = exp(-0.5.*(test_data(:,active_feat)'-mean_1)' * inv(covar_1) ...
        * (test_data(:,active_feat)'-mean_1))' ...
        ./ ((2.*pi).^(d/2).*det(covar_1)^0.5); %...
    lklhood_1 = diag(lklhood_1);

    post_0 = prod(lklhood_0.*prior_0, 2);
    post_1 = prod(lklhood_1.*prior_1, 2);

    correct = sum((post_0(test_data(:,end) == 0) > post_1(test_data(:,end) == 0)));
    wrong = sum(~(post_0(test_data(:,end) == 0) > post_1(test_data(:,end) == 0)));

    correct = correct + ...
        sum((post_1(test_data(:,end) == 1) >= post_0(test_data(:,end) == 1)));
    wrong = wrong + ...
        sum(~(post_1(test_data(:,end) == 1) >= post_0(test_data(:,end) == 1)));
```

```
end
```

```
function [correct, wrong] = FisherLinearDiscreminant (train_data, test_data, active_feat)

    % training
    X0 = train_data(train_data(:,end)==0,active_feat);
    X1 = train_data(train_data(:,end)==1,active_feat);

    n1 = length(X0);
    n2 = length(X1);

    mu0 = mean(X0);
    mu1 = mean(X1);

    S0 = (n1-1) * cov(X0);
    S1 = (n2-1) * cov(X1);

    Sw = S0 + S1;
    %SB = n1 * cov(mu1) + n2 * cov(mu2);

    V = Sw \ (mu0 - mu1)';


    % testing

    mean_0 = mean(X0*V);
    mean_1 = mean(X1*V);
    covar_0 = cov(X0*V);
    covar_1 = cov(X1*V);


    prior0tmp = length(X0);
    prior1tmp = length(X1);

    prior_0 = prior0tmp./(prior0tmp+prior1tmp);
    prior_1 = prior1tmp./(prior0tmp+prior1tmp);

    X = test_data(:,active_feat)*V;

    d = length(active_feat);
    lklhood_0 = exp(-0.5.*(X'-mean_0)' * inv(covar_0) ...
        * (X'-mean_0))' ./ ((2.*pi).^(d/2).*det(covar_0)^0.5); %...
    lklhood_0 = diag(lklhood_0);
    lklhood_1 = exp(-0.5.*(X'-mean_1)' * inv(covar_1) ...
```

```
        * (X'-mean_1))' ./ ((2.*pi).^(d/2).*det(covar_1)^0.5); %...
    lklhood_1 = diag(lklhood_1);


    post_0 = prod(lklhood_0.*prior_0, 2);
    post_1 = prod(lklhood_1.*prior_1, 2);

    correct = sum((post_0(test_data(:,end) == 0) > post_1(test_data(:,end) == 0)));
    wrong = sum(~(post_0(test_data(:,end) == 0) > post_1(test_data(:,end) == 0)));

    correct = correct + ...
        sum((post_1(test_data(:,end) == 1) >= post_0(test_data(:,end) == 1)));
    wrong = wrong + ...
        sum(~(post_1(test_data(:,end) == 1) >= post_0(test_data(:,end) == 1)));

end




clear

close all
clc

%rng(1000);
tic

X = [1 1 -1 0 2;
    0 0 1 2 0;
    -1 -1 1 1 0;
    4 0 1 2 1;
    -1 1 1 1 0;
    -1 -1 -1 1 0;
    -1 1 1 2 1];

Y = [2 1 2 1 1 1 2]';

data = [Y X];

fprintf('\n\nThe initial weight vector:\n')
w = [3 1 1 -1 2 -7]

C1 = data(data(:,1)==1,:);
C1(:,1) = 1;
C2 = data(data(:,1)==2,:);
C2(:,1) = 1;

y = [C1; -C2];

notConverged = true;
iteration = 0;
max_iteration = 100;

while notConverged || iteration > max_iteration
```

```
    iteration = iteration + 1;
    fprintf('\n\nIteration %d:\n', iteration)
    notConverged = false;

    for i = 1:size(y,1)
        if (w*y(i,:)'<= 0)
            notConverged = true;
            fprintf('\tThe perceptrone %d classification is wrong.\n ', i)
            fprintf('\n\tThe new weight vector is:')
            w = w + y(i,:)
        else
            fprintf('\tThe perceptrone %d classification is right.\n ', i)
        end
    end

end


fprintf('\n\nThe final weight vector:\n')
        w

toc
```