ME5404/EE5904R NEURAL NETWORKS Homework 2 Cheng Feng A0129146E

Question 1. Rosenbrock's Valley Problem?

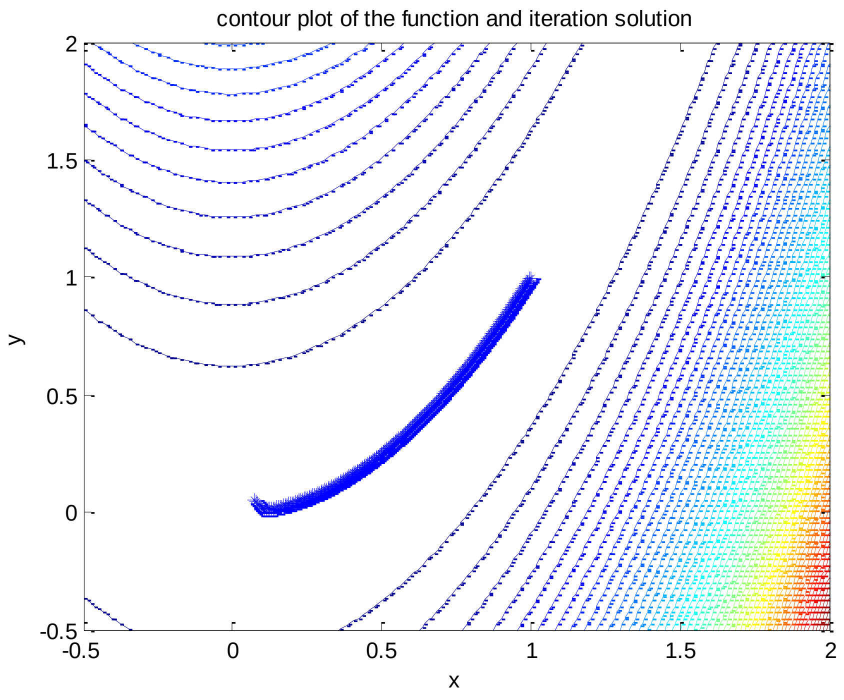
(a). Steepest (Gradient) descent method

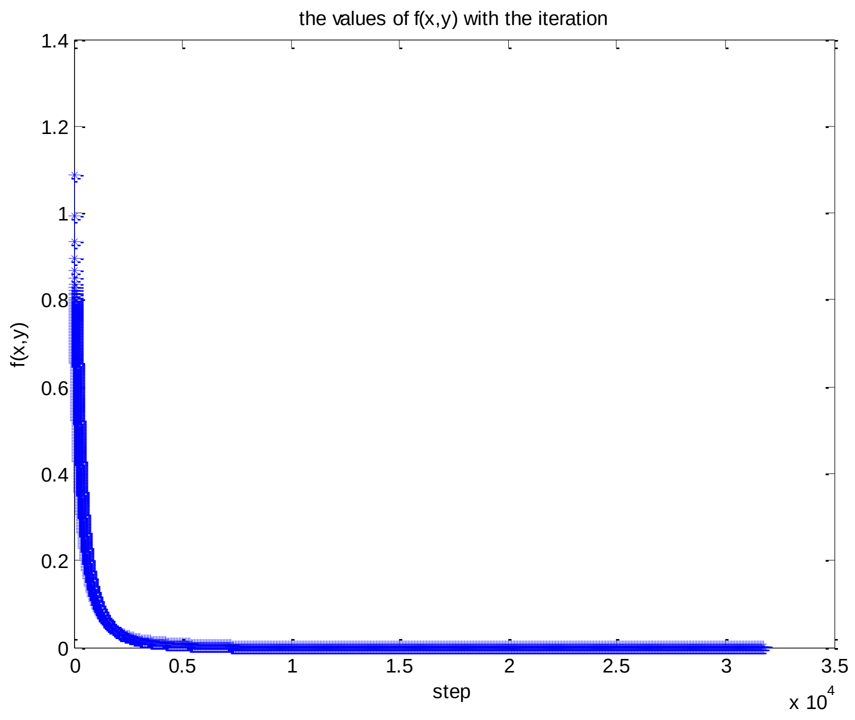
With learning rate η = 0.001. Record the number of iterations when f(x,y) converges to (or very close to) 0 and plot out the trajectory of (x,y) in the 2-dimensional space. Also plot out the function value as it approaches the global minimum. What would happen if a larger learning rate, say η = 0.1, is used?

Answer:

When η = 0.001, error tolerance = `1e-6`.

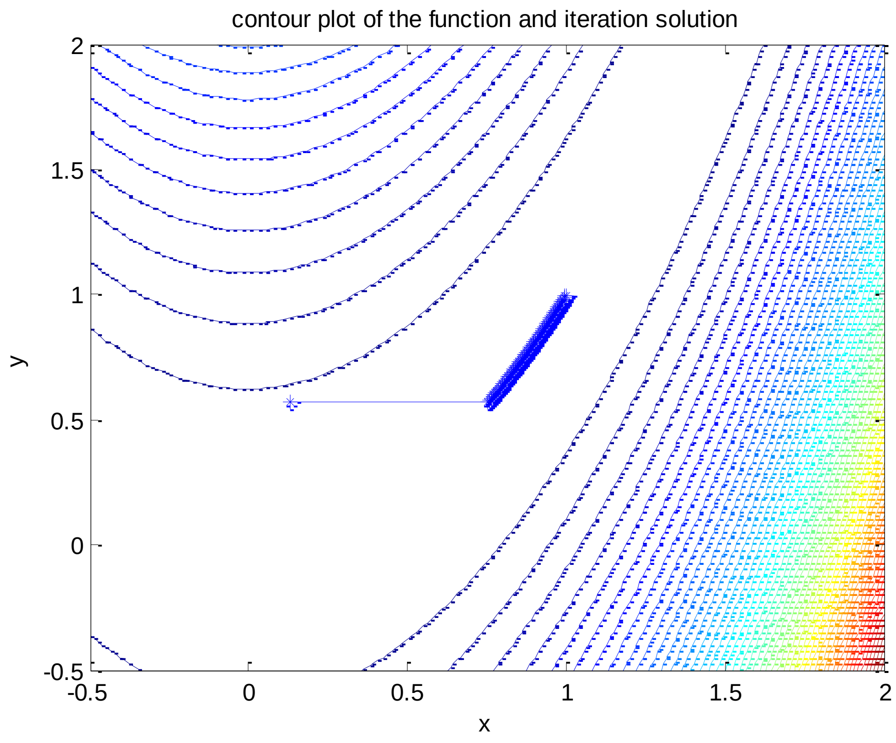
The solution is (x,y) =(0.9990,0.9980). The number of iteration steps is **31784**.

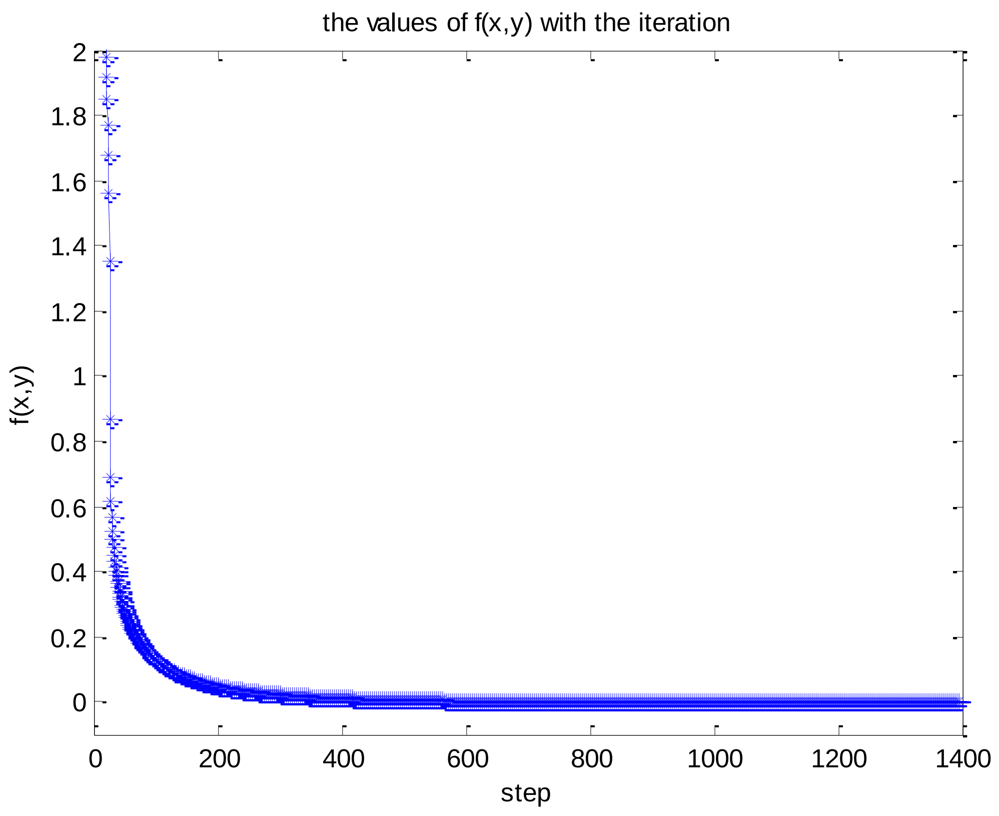




When η = 0.1, error tolerance = `1e-6`.

The solution is (x,y) =(0.9990,0.9981). The number of iteration steps is **1222**.





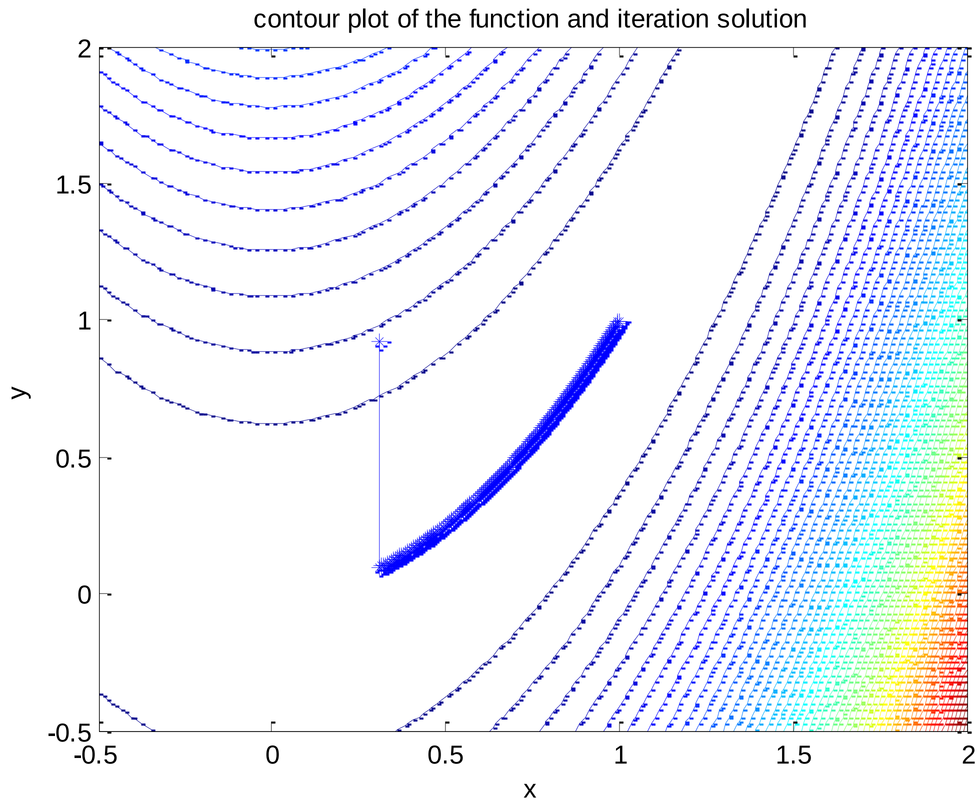
Above all, we can see the speed of the convergence increase if the learning rate η increase.
(The solution code is attached at the end of the report)

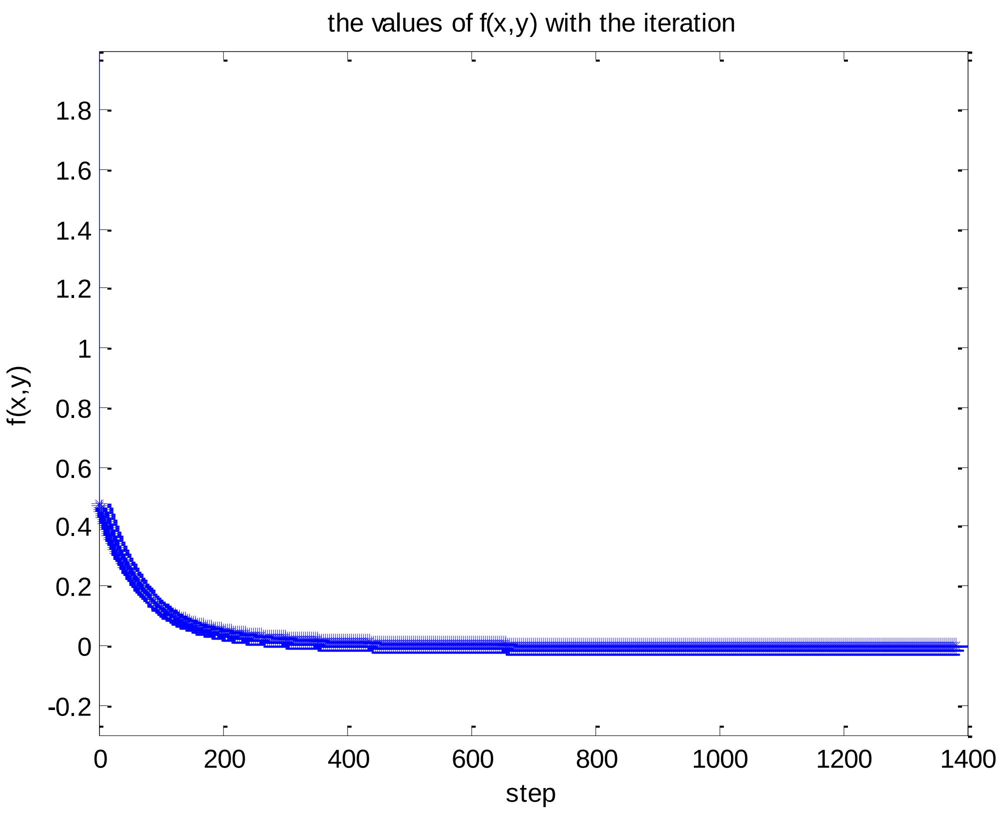
(b). Newton's method

Record the number of iterations when f(x,y) converges to (or very close to) 0 and plot out the trajectory of (x,y) in the 2-dimensional space. Also plot out the function value as it approaches the global minimum.

When error tolerance = `1e-6`.

The solution is (x,y) =(0.9990,0.9981). The number of iteration steps is **1222**.





(The solution code is attached at the end of the report)

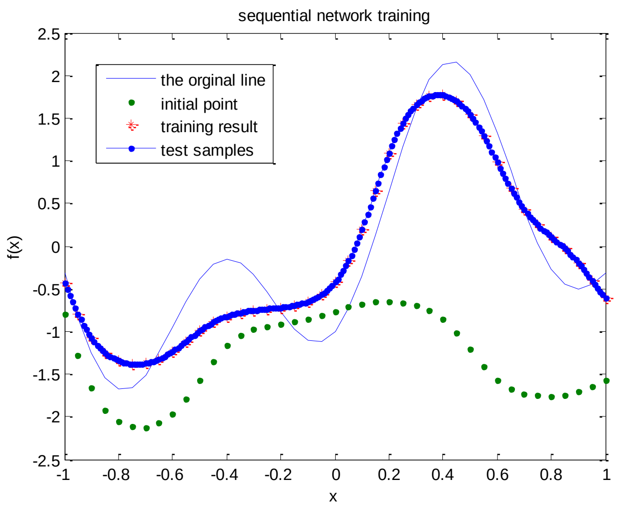
ME5404/EE5904R NEURAL NETWORKS Homework 2 Cheng Feng A0129146E

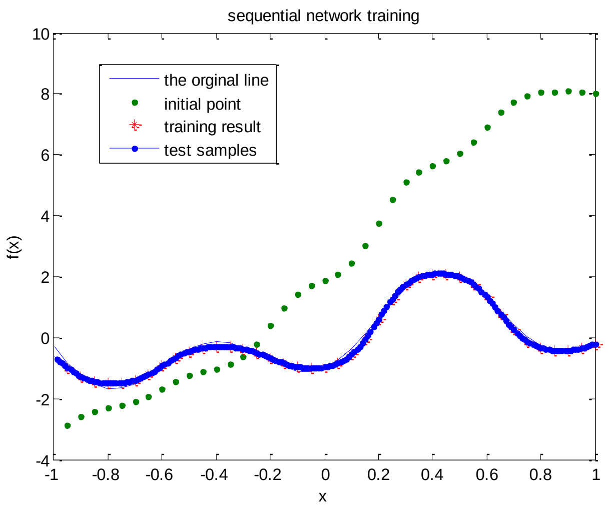
Question 2. Function Approximation?

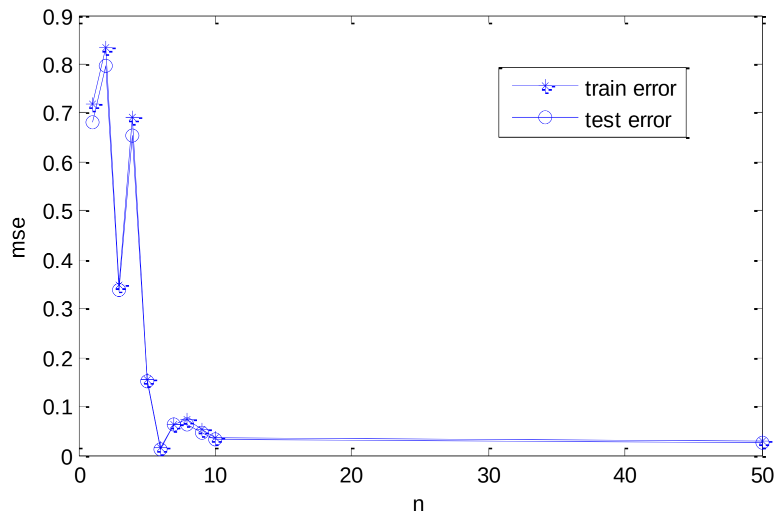
(a). Use the sequential mode with BP algorithm and experiment with the following different structures of the MLP.

When epoch_s = 1000, `net = newff(P,T,n,{'tansig' 'purelin'},'trainlm');`

I conduct the experiment when the number of neuron is 1-10, 50 respectively, and compute the both training error and testing error. The following charts (1, 2) show when the number of neuron is 5 and 6 and the plot of training results, testing results and the initial model. The last chart show the error plot when we choose different numbers of neuron. I find that when n = 6, then the error is the minimum.







So, if we choose different n, we compute the outputs of the MLP when x = -1.5 and 1.5, and compare with the real values. When x =-1.5, the real values is 0.8910; when x =1.5, the real values is -1.509

N	1	2	3	4	5	6	7	8	9	10	11
X=-1.5	-1.29	-1.79	1.10	-0.40	-0.74	1.19	-1.22	-0.02	-1.05	0.70	-0.03
X= 1.5	0.80	-0.52	-1.17	-1.26	0.24	1.12	2.62	-1.39	0.58	-0.63	0.80
Error(-1.5)	-2.18	-2.68	0.21	-1.29	-1.63	0.29	-2.11	-0.91	-1.94	-0.19	-0.92
Error(1.5)	2.31	0.99	0.34	0.24	1.75	2.63	4.13	0.12	2.09	0.87	2.32

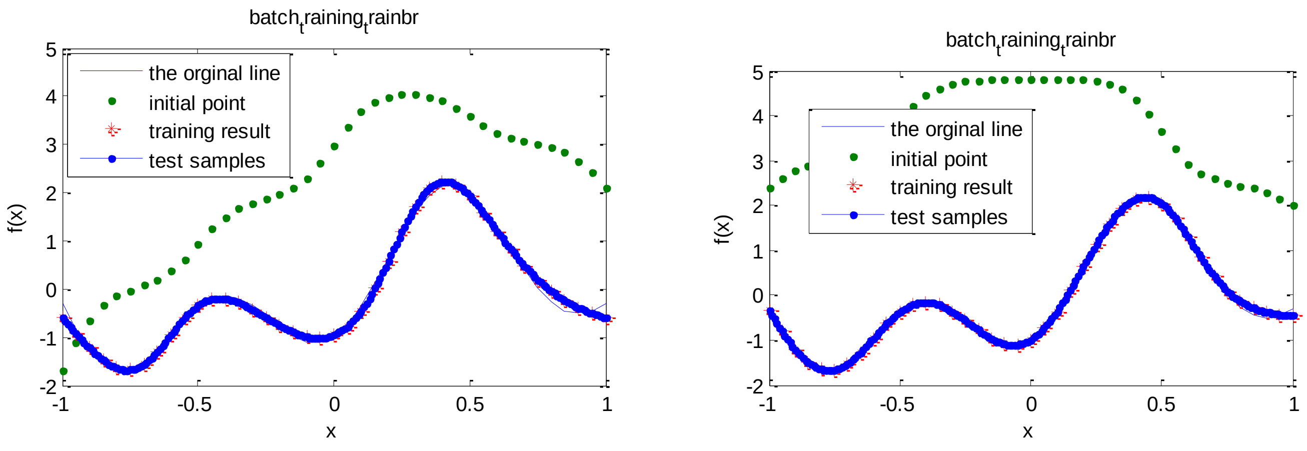
The predictions for the values outside of the range of the input are not reasonable.

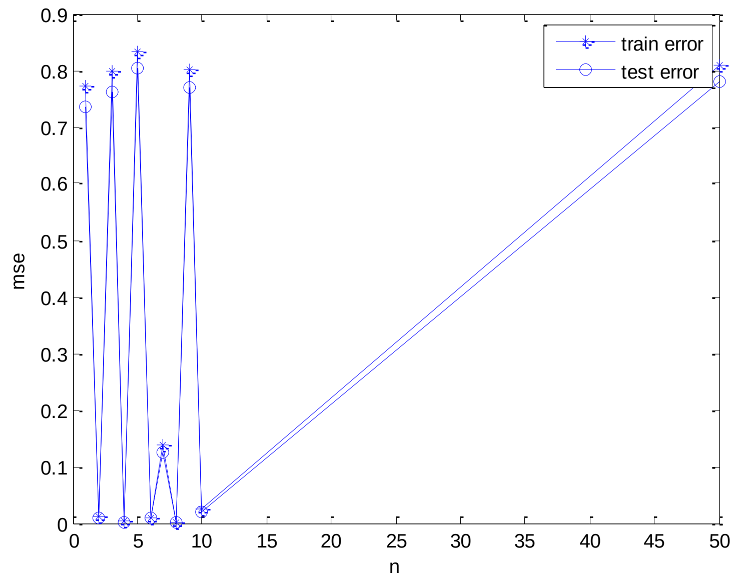
(The solution code is attached at the end of the report)

(b). Use the batch mode with trainlm algorithm to repeat the above procedure.

When epoch_s = 1000, `net = newff(P,T,n,{ },'trainlm');`

I conduct the experiment when the number of neuron is 1-10, 50 respectively, and compute the both training error and testing error. The following charts (1, 2) show when the number of neuron is 5 and 6 and the plot of training results, testing results and the initial model. The last chart show the error plot when we choose different numbers of neuron. I find that when n is even numbers (such as 2,4,6), then the error is the minimum.





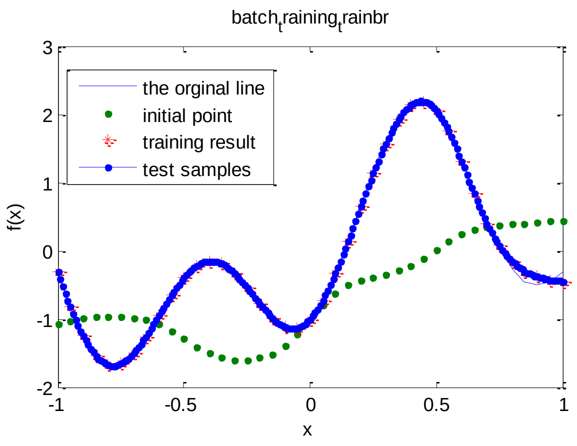
N	1	2	3	4	5	6	7	8	9	10	11
X=-1.5	-1.36	1.32	-1.27	3.86	-1.15	2.24	-1.24	2.87	-1.40	2.17	-0.97
X= 1.5	0.99	-0.83	1.07	-0.50	0.81	-0.54	-1.37	1.21	0.74	-2.15	0.79
Error(-1.5)	-2.25	0.43	-2.16	2.97	-2.04	1.35	-2.13	1.98	-2.29	1.28	-1.86
Error(1.5)	2.50	0.68	2.58	1.01	2.32	0.97	0.14	2.72	2.25	-0.64	2.30

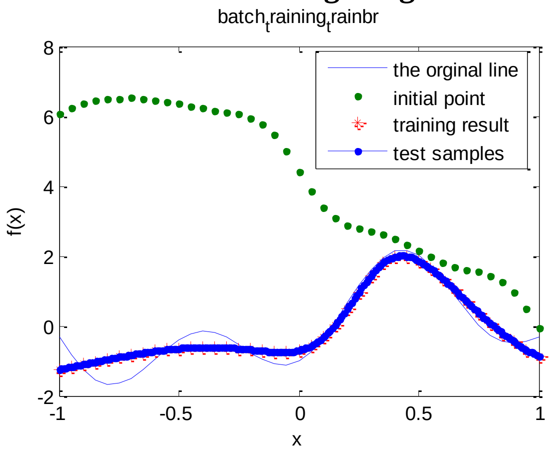
The predictions for the values outside of the range of the input are not reasonable.

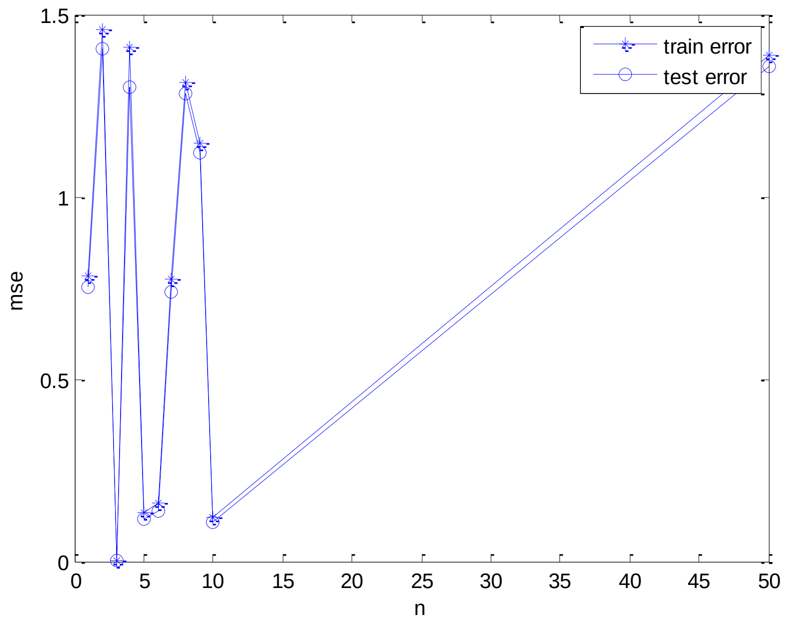
(The solution code is attached at the end of the report)

(c). Use the batch mode with trainbr algorithm to repeat the above procedure.
When epoch_s = 1000, `net = newff(P,T,n,{ },'trainbr');`
I conduct the experiment when the number of neuron is 1-10, 50 respectively, and compute the both training error and testing error. The following charts (1, 2) show when the number of neuron is 5 and 6 and the plot of training results, testing results and the initial model. The last chart show the error plot when we choose different numbers of neuron. I find that when n is 3, then the error is the minimum.
(The solution code is attached at the end of the report)

ME5404/EE5904R NEURAL NETWORKS Homework 2 Cheng Feng A0129146E





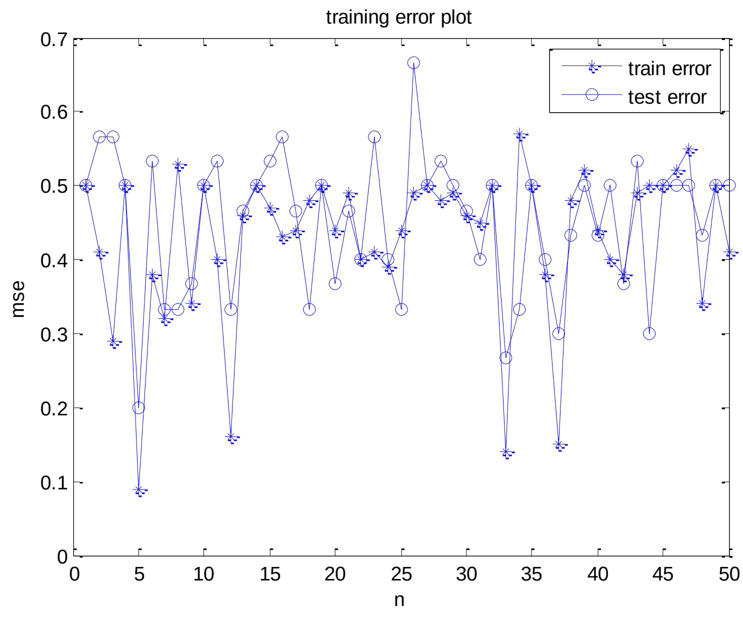


N	1	2	3	4	5	6	7	8	9	10	11
X=-1.5	-0.27	2.86	-1.10	-1.16	2.79	4.52	-1.40	-0.06	-0.66	0.18	-1.58
X= 1.5	0.08	-1.16	0.52	0.57	-0.50	-0.69	0.93	0.01	0.20	0.08	1.05
Error(-1.5)	-1.16	1.97	-1.99	-2.05	1.90	3.63	-2.30	-0.95	-1.55	-0.71	-2.47
Error(1.5)	1.59	0.35	2.03	2.08	1.01	0.82	2.43	1.52	1.71	1.59	2.55

The predictions for the values outside of the range of the input are not reasonable. The minimum of training error may not lead to a smaller prediction error.

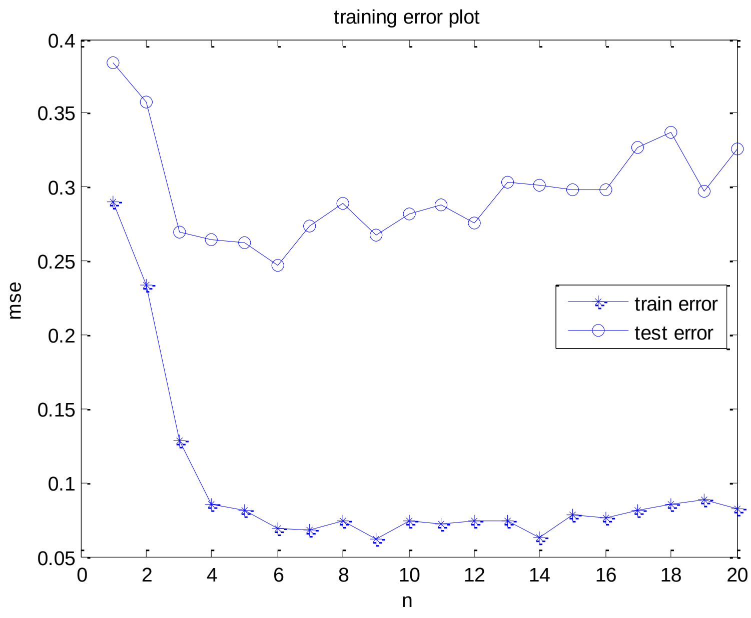
ME5404/EE5904R NEURAL NETWORKS Homework 2 Cheng Feng A0129146E
Question 3. Facial Image Based Glasses Wearing Recognition

(a). Apply the Rosenblatt's perceptron (**single layer perceptron**) to the glass-wearing recognition problem. After the training procedure, calculate the recognition errors for both the training set and the test set, and evaluate the performance of the network.



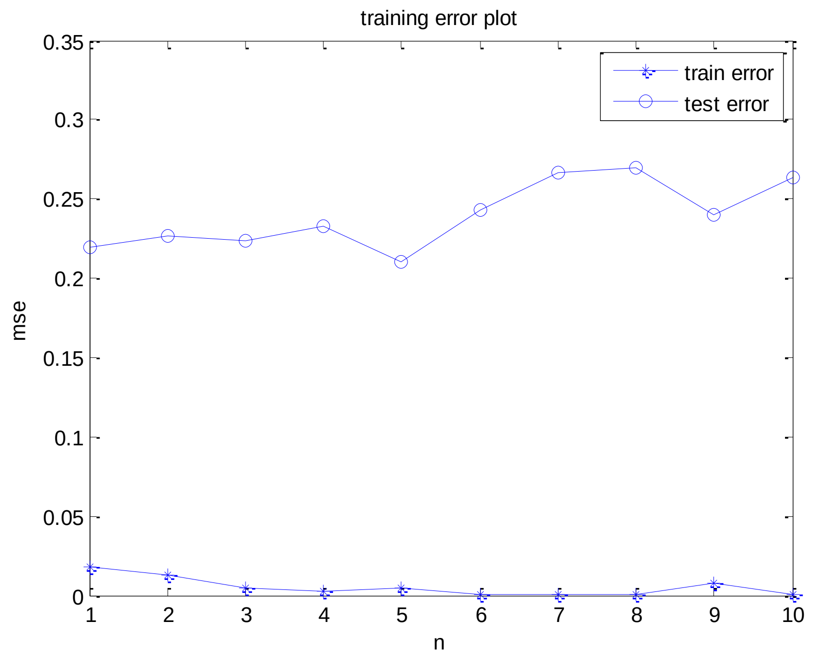
When apply single layer perceptron, both the training error and testing error are very high. And each training processes have different results, which means this problem is n0t linearly separable.
After 20 times experiments, the average of the training error is 42%.
And the average of testing error is 45%. ***(The solution code is attached at the end of the report)***

(b). Apply **the Multi-layer** perceptron to the glass-wearing recognition problem **using batch mode of learning**. After the training procedure, calculate the recognition error for both training set and test set, and evaluate the performance of the network.
I conduct 50 times experiments, and record the training error and testing error. The following chart is the plot of the training error and testing error.



(The solution code is attached at the end of the report)

(c). Apply **the Multi-layer** perceptron to the glass-wearing recognition problem **using sequential mode of learning**. After the training procedure, calculate the recognition error for both training set and test set, and evaluate the performance of the network. Which one is the best in terms of test error?



The result of the sequential learning is better than the batch learning. But, the sequential mode learning takes 10 times time cost than other methods. Because of the stochastic gradient descent direction, the function same times can reach better optimization results.

(The solution code is attached at the end of the report)

Appendix.

Q1

Steepest (Gradient) descent method

```
function home2_1()
% steepest gradient descent method and newton method
xy = rand(1,5); % initial point
k = 1; error = 1e-6; ita = 0.001;%learning rate
fxy = @(x,y) 100*(y - x.^2).^2 + (1 - x).^2;
x = xy(1);y=xy(2); xy(1,3) = fxy(x,y);
%% steepest gradient descent method
df=@(x,y)df(x,y);
while(abs(fxy(x,y))>error & k<50000 )
    pd = df(x,y);
    dx = x - ita*pd(1);
    [x, fx] = fminbnd(@(x) fxy(x,y), min(x,dx), max(x,dx));
    pd = df(x,y);
    dy = y - ita*pd(2);
    [y, fy] = fminbnd(@(y) fxy(x,y), min(y,dy), max(y,dy));
    k = k+1; % counter
    xy(k,1:2)=[x,y];
    xy(k,3) = fxy(x,y); %record the values of the f(x,y)
    xy(k,4:5) = df(x,y);
end
%% plot trajectory of (x,y) in the 2-dimensional space
plotme(); hold on;
x = xy(:,1)';  y = xy(:,2)';
plot(x,y,'-*');
title('contour plot of the function and iteration solution');
xlabel('x');ylabel('y');
%% plot the f(x,y) converges 0
figure(2)
plot(xy(:,3),'-*');
title('the values of f(x,y) with the iteration');
xlabel('step'); ylabel('f(x,y)');
ylim([-0.1,2]);
function fxy = df(x,y) %% differentiation
dfx = 2*(x-1)-400*(y-x^2)*x;
dfy = 200*(y-x^2); fxy = [dfx,dfy];
function plotme()
x = linspace(-0.5,2); y = linspace(-0.5,2);
[X,Y] = meshgrid(x,y); Z = (1-X).^2+100*(Y-X.^2).^2;
figure(1)
contour(X,Y,Z,50) %,'ShowText','on'
```

Q1

newton method

```
function home2_12()
xy = rand(1,5); % initial point
k = 1; error = 1e-6; ita = 1;%learning rate
fxy = @(x,y) 100*(y - x.^2).^2 + (1 - x).^2;
x = xy(1);y=xy(2); xy(1,3) = fxy(x,y);
%% newton method
df=@(x,y)df(x,y); hessian=@(x,y)Hessian(x,y);
while(abs(fxy(x,y))>error & k<10000 )
    inhessian = inv(hessian(x,y));
    pd = df(x,y)*inhessian;
    dx = x - ita*pd(1);
    [x, fx] = fminbnd(@(x) fxy(x,y), min(x,dx), max(x,dx));
    inhessian = inv(hessian(x,y));
    pd = df(x,y)*inhessian;
    dy = y - ita*pd(2);
    [y, fy] = fminbnd(@(y) fxy(x,y), min(y,dy), max(y,dy));
    k = k+1;
    xy(k,1:2)=[x,y];    xy(k,3) = fxy(x,y);    xy(k,4:5) = df(x,y);
end
%% plot trajectory of (x,y) in the 2-dimensional space
plotme(); hold on;
x = xy(:,1)'; y = xy(:,2)'; plot(x,y,'-*');
title('contour plot of the function and iteration solution');
xlabel('x');
ylabel('y')
%% plot the f(x,y) converges 0
```

```
figure(2)
plot(xy(:,3),'-*');
ylim([-0.3,2]);
title('the values of f(x,y) with the iteration');
xlabel('step');
ylabel('f(x,y)')
function fxy = df(x,y) %% differentiation
dfx = 2*(x-1)-400*(y-x^2)*x;
dfy = 200*(y-x^2);
fxy = [dfx,dfy];

function H = Hessian(x,y) %%
H = [2+1200*x-400*y,-400*x;-400*x,200];

function plotme()
x = linspace(-0.5,2);
y = linspace(-0.5,2);
[X,Y] = meshgrid(x,y);
Z = (1-X).^2+100*(Y-X.^2).^2;
figure(1)
contour(X,Y,Z,50) %,'ShowText','on'
```

Q2

```
Function Approximation
function home2_22()
fx = @(x)(1.2*sin(pi*x)-cos(2.4*pi*x));
x = [-1:0.01:1]; xtrain = [-1:0.05:1]; xtest = [-1:0.01:1];
P = xtrain; T = fx(xtrain); PP = xtest; TT = fx(xtest);
sizen=[1:10,50]; mse = zeros(2,length(sizen));
testx = zeros(2,length(sizen));
for n = 1:length(sizen)
    [net,net_initial] = batch_training_trainbr(P,T,n);
    initial_Y = sim(net_initial,P);  %
    plot(P,T,P,initial_Y,'.') ; hold on
    Y = sim(net,P);  % training error
    plot(P,Y,'*r');    hold on
    YY = sim(net,PP);  % testing error
    plot(PP,YY,'.-b') ;    title('batch_training_trainbr');
    xlabel('x');ylabel('f(x)');
    train_error = sum((T-Y).^2)/(length(Y)-2);
    test_error = sum((TT-YY).^2)/(length(YY)-2);
    mse(:,n) = [train_error;test_error];
    YX = sim(net,[-1.5,1.5]);     testx(:,n) = YX;
End
```

Sequential_training

```
function [net,net_initial] = sequential_training(P,T,n)
%% P input; T output; n number of the neuron
net = newff(P,T,n,{'tansig' 'purelin'},'trainlm');
net_initial = net;
epoch_s = 1000;  % Specify the number of epoch for sequential training
net.trainParam.epochs = 1;  % Set epoch of batch training to be 1
for i = 1 : epoch_s
    index = randperm(length(P));   % Shuffle the input data every epoch
    net = adapt(net,P(index),T(index));    % Perform sequential learning
end
```

Batch_training_trainbr

```
function [net,net_initial] = batch_training_trainbr(P,T,n)
net = newff(P,T,5,{},'trainbr'); %% create s feed back progation network
net_initial = net;
net.trainParam.epochs = 1000;
net = train(net,P,T);
```

Batch_training_trainlm

```
function [net,net_initial] = batch_training_trainlm(P,T,n)
net = newff(P,T,5,{},'trainlm'); %% create s feed back progation network
net_initial = net;
net.trainParam.epochs = 1000;
net = train(net,P,T);
```

Q3

Function Approximation

```
function homework3_1()
%% Facial Image Based Glasses Wearing Recognition
% Apply the Rosenblatt's perceptron (single layer perceptron)
clc,clear;close all
times =50;
record_error = zeros(times,2);
for i = 1:times
    error =  single_perceptron();
    record_error(i,1) = error(1);
    record_error(i,2) = error(2);
end
plot(record_error(:,1),'*-');hold on
plot(record_error(:,2),'o-');mean(record_error)
```

single_perceptron

```
function error = single_perceptron()
[trainB1,trainB2,testB1,testB2]= readimg();
P = [trainB1,trainB2];
T = [ones(1,50),zeros(1,50)];
net = newff(P,T,1,{'purelin'},'traincgf');
net.trainParam.epochs = 1000;
net = train(net,P,T);
% view(net);close all
trainresult = sim(net,P);
trainer = threshold(trainresult);
trainerr = 1- sum(trainer==T)/length(trainer);
%% test
PP = [testB1,testB2];
TT = [ones(1,15),zeros(1,15)];
testresult = sim(net,PP);
tester = threshold(testresult);
testerr = 1- sum(tester==TT)/length(tester);
error =[trainerr,testerr];
```

readimg()

```
function [trainB1,trainB2,testB1,testB2]= readimg()
path_train_glass = 'E:/courses/Course 2014 semester 2/ME5404-EE5904R NEURAL 
NETWORKS/homework2/Glasswearing/Train/WithGlasses';
path_train_noglass = 'E:/courses/Course 2014 semester 2/ME5404-EE5904R 
NEURAL NETWORKS/homework2/Glasswearing/Train/WithoutGlasses';
path_test_glass = 'E:/courses/Course 2014 semester 2/ME5404-EE5904R NEURAL 
NETWORKS/homework2/Glasswearing/Test/WithGlasses';
path_test_noglass = 'E:/courses/Course 2014 semester 2/ME5404-EE5904R 
NEURAL NETWORKS/homework2/Glasswearing/Test/WithoutGlasses';
dir1 = dir(path_train_glass);dir2 = dir(path_train_noglass);
dir3 = dir(path_test_glass);dir4 = dir(path_test_noglass);
for i = 1:length(dir1)-2
    ind = dir1(i+2);    filename = getfield(ind,'name');
    file =strcat(path_train_glass,'/',filename) ;
    A = double(imread(file));    trainB1(:,i) = A(:);
end
for i = 1:length(dir2)-2
    ind = dir2(i+2);    filename = getfield(ind,'name');
    file =strcat(path_train_noglass,'/',filename) ;
    A = double(imread(file));    trainB2(:,i) = A(:);
end
for i = 1:length(dir3)-2
    ind = dir3(i+2);    filename = getfield(ind,'name');
    file =strcat(path_test_glass,'/',filename) ;
    A = double(imread(file));    testB1(:,i) = A(:);
end
for i = 1:length(dir4)-2
    ind = dir4(i+2);    filename = getfield(ind,'name');
    file =strcat(path_test_noglass,'/',filename) ;
    A = double(imread(file));    testB2(:,i) = A(:);
end
```

```
Multi_perceptron_sequential training
function error = Multi_perceptron_sqtrain(n)
[trainB1,trainB2,testB1,testB2]= readimg();
P = [trainB1,trainB2];T = [ones(1,50),zeros(1,50)];
PP = [testB1,testB2];TT = [ones(1,15),zeros(1,15)];
error = zeros(2,n);epoch_s = 100;
net.trainParam.epochs = 1;
for i = 1:n
    net = newff(P,T,i,{'tansig' 'purelin'},'trainlm');
    for j = 1 : epoch_s
        index = randperm(100); % Shuffle the input data every epoch
        net = adapt(net,P(:,index),T(:,index));
    end
    trainresult = sim(net,P);
    trainer = threshold(trainresult);
    trainerror = 1 - sum(trainer==T)/length(trainer);
    testresult = sim(net,PP);
    tester = threshold(testresult);
    testerror = 1 - sum(tester==TT)/length(tester);
    error(:,i) = [trainerror;testerror];
end
```

```
Multi-layer perceptron batch mode of learning.
function error = Multi_layer_perceptron(n)
[trainB1,trainB2,testB1,testB2]= readimg();
P = [trainB1,trainB2];
T = [ones(1,50),zeros(1,50)];
PP = [testB1,testB2];
TT = [ones(1,15),zeros(1,15)];
error = zeros(2,n);
for i = 1:n
    net = newff(P,T,i,{'tansig' 'purelin'},'traincgf');
    net.trainParam.epochs = 1000;     net = train(net,P,T);
    trainresult = sim(net,P);    trainer = threshold(trainresult);
    trainerror = 1 - sum(trainer==T)/length(trainer);
    testresult = sim(net,PP);
    tester = threshold(testresult);
    testerror = 1 - sum(tester==TT)/length(tester);
    error(:,i) = [trainerror;testerror];
end
```