Onur Poyraz
2010401036
23 April 2015

Homework III

In this project, I deal with the Multilayer Perceptron Model Learning Algorithm. It is asked from us that we should implement dynamic MLP structure and test it with three different data sets which are Xor data set, f(x)=1/x data set(x=1….100) and Iris data set respectively. For both of these data sets I use the same dynamic MLP algorithm with user defined number of layers, number neurons per layer and learning factor and momentum coefficient.

I begin the code with reading the inputs and desired outputs form .txt files. (For f(x)=1/x I did not use .txt because it is a function rather than given data set therefore it is more logical that define this data set in the code.) The codes for reading part is given below:

For Xor data set I took values from .txt and I define the input and desired vector. It is important that in .txt file there should be header 'input1, input2, output' for each column respectively.

```
formatSpec = '%f%f%f';
A=readtable('xor.txt','Delimiter',',','Format',formatSpec);
X=A{1:4,{'input1','input2'}};
d=A{1:4,{'output'}};
```

For f(x)=1/x data set I randomly select 85 train inout and 15 test input by using randperm function.

```
indis=randperm(100);
for i=1:100
   if i<=85
      X(i,1)=indis(i);
      d(i,1)=1/indis(i);
   else
      X_test(i-85,1)=indis(i);
      d_test(i-85,1)=1/indis(i);
   end
end
```

For Iris data set I again use the .txt file to read inputs and desired values. In .txt file there should be header same as 'input1, input2, input3, input4, output' for each column in the data set respectively. In here I randomly select 135 data for training and 15 data for test and I

define desired output values from the names in the data set ‘Iris-setosa’, ‘Iris-versicolor’ and ‘Iris-virginica’.

```
formatSpec = '%f%f%f%f%C';
A=readtable('iris.txt','Delimiter',',','Format',formatSpec);
X_all=A{1:150,{'input1','input2','input3','input4'}};
d_name=A{1:150,{'output'}};
for i=1:150
   if d_name(i)=='Iris-setosa'
      d_all(i,:)=[1 0 0];
   elseif d_name(i)=='Iris-versicolor'
      d_all(i,:)=[0 1 0];
   else
      d_all(i,:)=[0 0 1];
   end
end
indis=randperm(150);
for i=1:150
   if i<=135
      X(i,:)=X_all(indis(i),:);
      d(i,:)=d_all(indis(i),:);
   else
      X_test(i-135,:)=X_all(indis(i),:);
      d_test(i-135,:)=d_all(indis(i),:);
      d_result(i-135,1)=d_name(indis(i));
   end
end
```

After defining the inputs I set the parameters which are important for implementation which are learning rate momentum rate and shape of the system. In the nodes matrix the first term represents the number of inputs the last term represent the number of outputs and the other terms represents the hidden layers and the number of neurons on each hidden neurons.

```
lf=0.01;
moment=0;
nodes=[4 5 10 3];
```

After that I define the sigmoid function. In the codes, I use two different function. The first one is the Sigmoid function which are given in class. The other function is tan function. With tan function the algorithm can find outputs more precisely and with less epochs. Therefore I work more with the tank function but I write the code for both.

For tanh function;

```
sigmoid = @(x) tanh(x);
sigmoid_diff= @(x) sech(x);
```

For sigmoid funciton;

```
sigmoid = @(x) 1/(1+exp(-x));
sigmoid_diff= @(x) (1/(1+exp(-x)))*(1-(1/(1+exp(-x))));
```

After that I initialize my weight matrix and my thresholds between [-0.05:0.05]

```
for k=1:layer
   for j=1:nodes(k+1)
      for i=1:nodes(k)
         w(i,j,k)=(rand-0.5)/10;
      end
   end
end
for k=1:layer-1
   for j=1:nodes(k+1)
      teta(k,j)=(rand-0.5)/10;
   end
end
```

In the algorithm part I begin the code with initializing the the values and make randperm for sampler pattern for each epoch. Therefore during training my code is always randomly select the inputs and use all of them equally.

After that I firstly run the forward propagation and I find output, sigma and error functions for the output nodes. In here, I did not use sigmoid function for the output layer. Because when I use sigmoid at the output I can not take correct result.

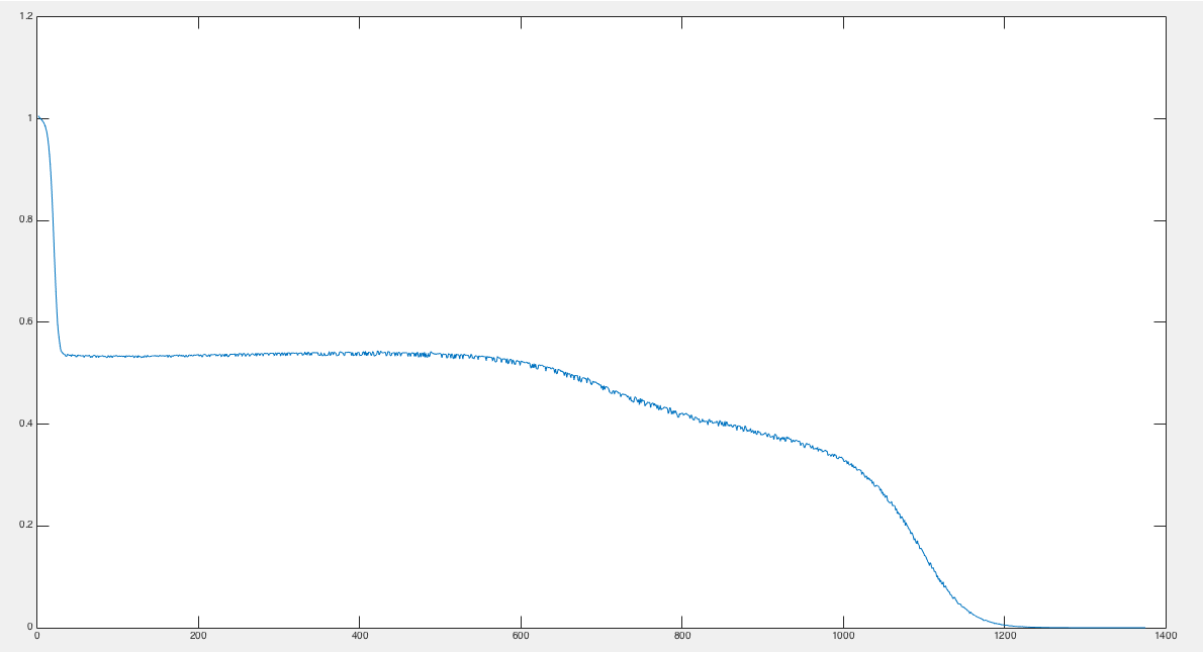
```
for k=1:layer
   for j=1:nodes(k+1)
      for i=1:nodes(k);
         x(k,j)=x(k,j)+input(i)*w(i,j,k);
      end
      if k==layer
         y(k,j)=x(k,j);
         sigma(k,j)=d(s,j)-y(k,j);
         output(s,j)=y(k,j);
         error(1,s)=error(1,s)+(sigma(k,j)^2);
      else
         x(k,j)=x(k,j)+teta(k,j);
         y(k,j)=sigmoid(x(k,j));
         y_diff(k,j)=sigmoid_diff(x(k,j));
      end
      error(1,s)=error(1,s)/2;
   end
   if k~=layer
      input=y(k,:);
   end
end
```

After doing forward propogation, I am done the back propagation. In here I find the sigma values for each neuron and I update the weights from the end to the beginning of the structure.

```
 for k=layer:-1:1
   for i=1:nodes(k)
      for j=1:nodes(k+1)
         if k~=3
```

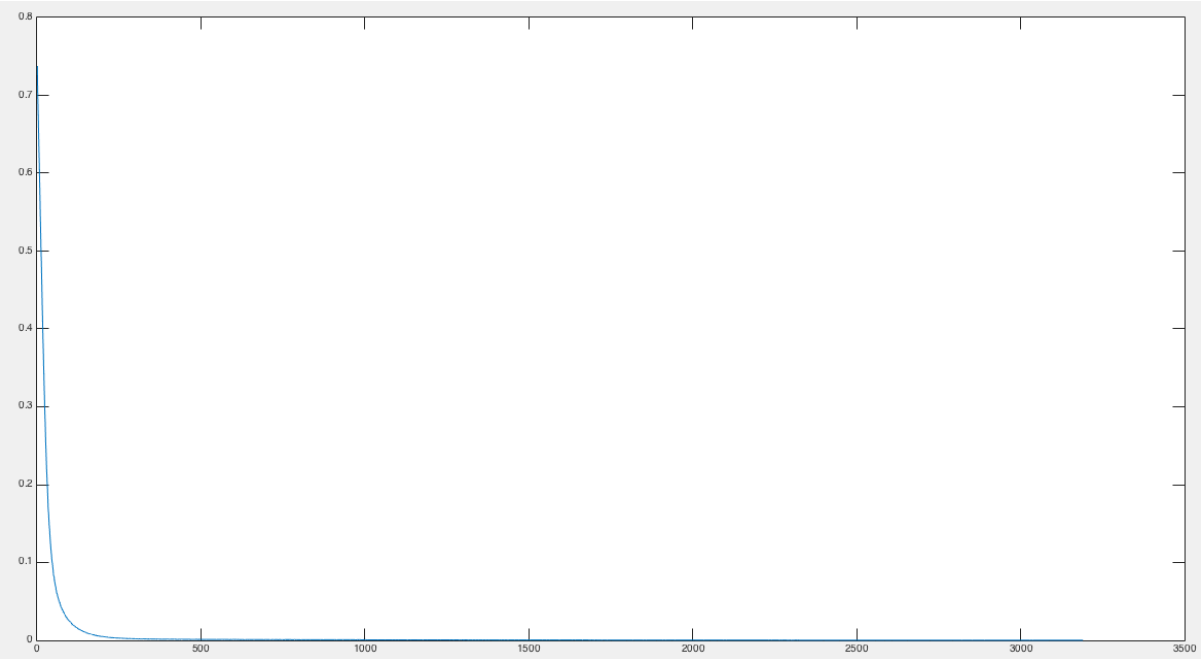
```
            delta_teta(k,j)=lf*sigma(k,j)+moment*delta_teta_old(k,j);
            teta(k,j)=teta(k,j)+delta_teta(k,j);
        end
        if k==1
            delta_w(i,j,k)=lf*sigma(k,j)*X(s,i)+moment*delta_w_old(i,j,k);
            w(i,j,k)=w(i,j,k)+delta_w(i,j,k);
        else
            delta_w(i,j,k)=lf*sigma(k,j)*y(k-1,i)+moment*delta_w_old(i,j,k);
            w(i,j,k)=w(i,j,k)+delta_w(i,j,k);
            sigma(k-1,i)=sigma(k-1,i)+w(i,j,k)*sigma(k,j);
        end
      end
      if k~=1
        sigma(k-1,i)=sigma(k-1,i)*y_diff(k-1,i);
      end
    end
  end
```

From this point of the code, I draw the total cost function per epoch graph and I write the test algorithm code. In the test algorithm there is only forward propagation without calculating sigma and error.



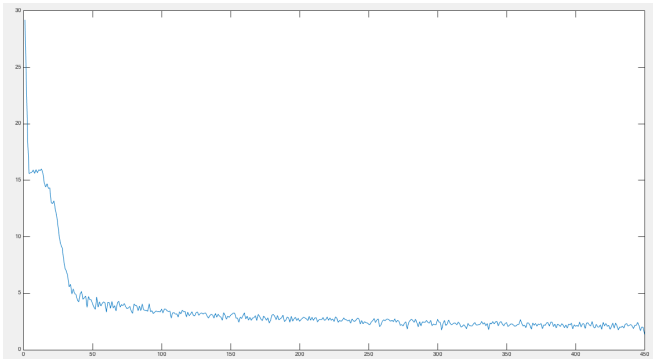
The total cost function for the Xor data set with one hidden layer. After 1200 epoch its value goes to 0.00001.

Desired Output	Actual Output	Percentage Error
0	0.0011	0.1144
1.0000	0.9996	0.0366
1.0000	0.9994	0.0595
0	0.0003	0.0349



The total cost function for f(x)=1/x. In this data set I see that for x=1 and x=2 the percentage error can be high however the other values error is usually small.

Desired Output	Actual Output	Percentage Error
0.0250	0.0247	1.1267
0.0556	0.0550	0.9285
0.0116	0.0112	3.5081
0.0385	0.0374	2.8085
0.0208	0.0207	0.6686
0.0189	0.0187	0.8609
0.0417	0.0406	2.6781
0.0139	0.0135	2.9720
0.0278	0.0273	1.6836
0.0125	0.0121	3.5083
0.5000	0.5385	7.6921
0.0172	0.0170	1.3197
0.0106	0.0103	2.8745
0.0135	0.0131	3.1543
0.0185	0.0183	0.9352



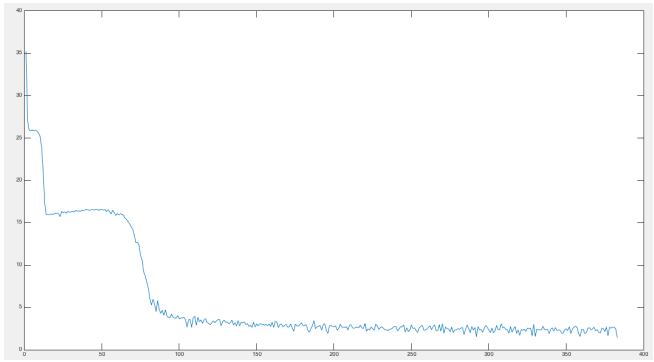


Fig.1 for one hidden layer and Fig.2 for two hidden layer with the Iris data set. In below there is data set for the two hidden layer. With one hide layer there is always same data set.

Desired Output			Actual Output			Percentage Error		
0	1.0000	0	-0.0058	0.9749	0.0338	0.5849	2.5087	3.3769
0	0	1.0000	-0.0001	-0.0383	1.0393	0.0112	3.8349	3.9260
0	0	1.0000	-0.0002	-0.0234	1.0241	0.0179	2.3365	2.4127
0	0	1.0000	-0.0001	-0.0332	1.0341	0.0129	3.3222	3.4060
0	0	1.0000	-0.0001	-0.0384	1.0394	0.0118	3.8448	3.9398
0	1.0000	0	0.0279	0.9849	-0.0193	2.7870	1.5117	1.9315
1.0000	0	0	1.0082	0.0015	-0.0085	0.8239	0.1454	0.8540
1.0000	0	0	0.9883	0.0111	0.0034	1.1673	1.1089	0.3449
0	1.0000	0	-0.0126	1.0123	0.0013	1.2633	1.2300	0.1294
1.0000	0	0	0.9863	0.0131	0.0030	1.3655	1.3069	0.3043
0	0	1.0000	-0.0003	-0.0157	1.0166	0.0268	1.5672	1.6578
0	0	1.0000	-0.0001	-0.0351	1.0360	0.0120	3.5086	3.5953
0	0	1.0000	-0.0001	-0.0327	1.0335	0.0121	3.2664	3.3454
0	1.0000	0	-0.0040	1.0014	0.0034	0.3989	0.1382	0.3383
1.0000	0	0	0.9823	0.0172	0.0037	1.7663	1.7171	0.3715

Desired Name	Actual Name
Iris-versicolor	Iris-versicolor
Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica
Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa
Iris-setosa	Iris-setosa
Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa
Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica
Iris-virginica	Iris-virginica
Iris-versicolor	Iris-versicolor
Iris-setosa	Iris-setosa