BACK – PROPAGATION ALGORITHM (Two Layered Network)

1.

- Initialize the weights. Activate the network.
- Compute the response using forward pass technique.

2. Forward pass:

First of all, the actual output of the network is computed. The computation is done as follows:

- $h_j = \Sigma w_{jk}\, x_k$
- $v_j =$ f(h_j) ; where f(x) is the tan-sigmoid function.
- $g_i = \Sigma w_{ij}\, v_j$
- output ,y = k(g_i); where k(x) is the log-sigmoid function.

3. Backward Pass:

- Compute the error e = $y_d - y$
- Compute $\delta_i = y(1 - y)\,(y_d - y)$

δ_i is used to distribute the error at the output unit back to the preceding layers. Update the weights connecting hidden layer to the output layer using the following updation rule.

- $w_{ij}(t+1) = w_{ij}(t) + \eta\delta_i\, v_j$ ($l_r =$ learning rate)
- Compute (δ_i = D =derivative vector)

  $\delta_j = v_j\,(1- v_j)\, w_{jk}\, \delta j\, x_k$

  It is not necessary to propagate the error back to the input layer.δ_j is used to adapt the weights connecting the input layer to the hidden layer. Update the weights connecting input layer to the hidden layer.

- $w_{jk}(t+1) = w_{jk}(t) + \eta\delta i\delta j\, x_k$

After all the δ factors have been determined, the weights for all layers are adjusted simultaneously.

4. Repeat steps 2 and 3 for all given patterns in the given training set.
5. Compute the rms error, given by sqrt($(y_d - y)/n$)where n is the number of patterns.

   If the rms error > $\in$, then repeat steps 2 through 4.Else break.

APPENDIX-2

NEURAL NETWORK TESTING AND TRAINING CODE:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#define nset 60
#define nin 3
#define toler 1.0e-05
#define SEED 123456789123456789123456789
#define neta 0.7
double myrand(void);
void train1(double i1,double i2,double i3,double o1);
void train2(double i1,double i2,double i3,double o2);
void train3(double i1,double i2,double i3,double o3);
void train4(double i1,double i2,double i3,double o4);
void test(double t1,double t2,double t3);
double func(double x);
long iter;
double w11[3][12],w12[12][3],w13[3],out1[60],in[60][3],w21[3][6];
double w22[6][12],w23[12],out2[60];
double w31[3][6],w32[6][12],w33[12],out3[60];
double w41[3][9],w42[9][12],w43[12],out4[60];
FILE *ptr1,*ptr2,*ptr3,*ptr4,*ptr5;
double myrand()
{
double g;
g=rand()/50000.0;
return g;
}
double func(double x)
{
double d;
d=1/(1+exp(-x));
return d;
}
void main(void)
{
int i,j;
double i1,i2,o1,o2,o3,o4,y,t1,t2;
double i3,t3;
clrscr();
```

```
srand(SEED);
ptr1=fopen("o1.dat","r");
ptr2=fopen("in.dat","r");
ptr3=fopen("o2.dat","r");
ptr4=fopen("o3.dat","r");
ptr5=fopen("o4.dat","r");
for(i=0;i<3;i++)
{
for(j=0;j<12;j++)
{
w11[i][j]=myrand();
}
}
for(i=0;i<12;i++)
{
for(j=0;j<3;j++)
{
w12[i][j]=myrand();
}
}
for(i=0;i<3;i++)
{
w13[i]=myrand();
}
for(i=0;i<nset;i++)
{
for(j=0;j<nin;j++)
{
fscanf(ptr2,"%lf",&in[i][j]);
}
}
for(i=0;i<nset;i++)
{
fscanf(ptr1,"%lf",&out1[i]);
}
for(i=0;i<3;i++)
{
for(j=0;j<6;j++)
{
w21[i][j]=myrand();
}
}
for(i=0;i<6;i++)
{
for(j=0;j<12;j++)
{
```

```
w22[i][j]=myrand();
}
}
for(i=0;i<12;i++)
{
w23[i]=myrand();
}
for(i=0;i<nset;i++)
{
fscanf(ptr3,"%lf",&out2[i]);
}
for(i=0;i<3;i++)
{
for(j=0;j<6;j++)
{
w31[i][j]=myrand();
}
}
for(i=0;i<6;i++)
{
for(j=0;j<12;j++)
{
w32[i][j]=myrand();
}
}
for(i=0;i<12;i++)
{
w33[i]=myrand();
}
for(i=0;i<nset;i++)
{
fscanf(ptr4,"%lf",&out3[i]);
}
for(i=0;i<3;i++)
{
for(j=0;j<9;j++)
{
w41[i][j]=myrand();
}
}
for(i=0;i<9;i++)
{
for(j=0;j<12;j++)
{
w42[i][j]=myrand();
}
```

```
}
for(i=0;i<12;i++)
{
w43[i]=myrand();
}
for(i=0;i<nset;i++)
{
fscanf(ptr5,"%lf",&out4[i]);
}
for(i=0;i<nset;i++)
{
j=0;
i1=in[i][j];
i2=in[i][j+1];
i3=in[i][j+2];
i3=log10(i3/100)/log10(500);
o1=out1[i];
train1(i1,i2,i3,o1);
}
for(i=0;i<nset;i++)
{
j=0;
i1=in[i][j];
i2=in[i][j+1];
i3=in[i][j+2];
i3=log10(i3/100)/log10(500);
o2=out2[i];
train2(i1,i2,i3,o2);
}
for(i=0;i<nset;i++)
{
j=0;
i1=in[i][j];
i2=in[i][j+1];
i3=in[i][j+2];
i3=log10(i3/100)/log10(500);
o3=out3[i];
train3(i1,i2,i3,o3);
}
for(i=0;i<nset;i++)
{
j=0;
i1=in[i][j];
i2=in[i][j+1];
i3=in[i][j+2];
i3=log10(i3/100)/log10(500);
```

```
o4=out4[i];
train4(i1,i2,i3,o4);
}
fclose(ptr1);
fclose(ptr2);
fclose(ptr3);
fclose(ptr4);
fclose(ptr5);
printf("Enter salt concentration,PEG concentration and molecular weight of PEG:\n");
scanf("%lf %lf %lf",&t1,&t2,&t3);
t3=log10(t3/100)/log10(500);
test(t1,t2,t3);
getch();
}
void train1(double i1,double i2,double i3,double o1) /*train 1st net*/
{
double err,y=0.0,x,z,fx,sum=0,h11[12],h12[3],o,dh11[12],dh12[3];
int i=0,j,iter=0;
do
{
iter++;
for(j=0;j<12;j++)
{
i=0;
x=i1*w11[i][j]+i2*w11[i+1][j]+i3*w11[i+2][j];
fx=func(x);
h11[j]=fx;
}
for(j=0;j<3;j++)
{
sum=0;
for(i=0;i<12;i++)
{
z=h11[i]*w12[i][j];
sum+=z;
}
fx=func(sum);

h12[j]=fx;
}
sum=0;
for(i=0;i<3;i++)
{
x=h12[i]*w13[i];
sum+=x;
}
```

```
y=func(sum);
err=(o1-y);
if(err<0)
err*=-1;
for(i=0;i<3;i++)
{
o=y*(1-y)*(o1-y);
w13[i]+=neta*h12[i]*o;
}
for(i=0;i<3;i++)
dh12[i]=h12[i]*(1-h12[i])*o*w13[i];
for(i=0;i<12;i++)
{
for(j=0;j<3;j++)
{
w12[i][j]+=neta*h11[i]*dh12[j];
}
}
for(i=0;i<12;i++)
{
dh11[i]=(h11[i])*(1-
h11[i])*(dh12[0]*w12[i][0]+dh12[1]*w12[i][1]+dh12[2]*w12[i][2]);
}
for(i=0;i<3;i++)
{
for(j=0;j<12;j++)
{
w11[i][j]+=neta*i1*dh11[j];
}
}
}while(err>toler);
printf("M");
printf("the salt concentration in the bottom phase is %lf\n",y);
printf("No. of iterations=%ld\n",iter);
}
void test(double t1,double t2,double t3)
{
double z,xt,fxt,sumt=0;
int i=0,j=0;
double th11[12],th12[3],th21[6],th22[12];
double th31[6],th32[12],th41[9],th42[12];
for(j=0;j<12;j++)
{
i=0;
xt=t1*w11[i][j]+t2*w11[i+1][j]+t3*w11[i+2][j];
fxt=func(xt);
```

```
th11[j]=fxt;
}
for(j=0;j<3;j++)
{
sumt=0;
for(i=0;i<12;i++)
{
xt=th11[i]*w12[i][j];
sumt+=xt;
}
fxt=func(sumt);
th12[j]=fxt;
}
sumt=0;
for(i=0;i<3;i++)
{
xt=th12[i]*w13[i];
sumt+=xt;
}
z=func(sumt);
printf("salt concentration in bottom phase is %lf \n",z);
for(j=0;j<6;j++)      /*2nd net test*/
{
i=0;
xt=t1*w21[i][j]+t2*w21[i+1][j]+t3*w21[i+2][j];
fxt=func(xt);
th21[j]=fxt;
}
for(j=0;j<12;j++)
{
sumt=0;
for(i=0;i<6;i++)
{
xt=th21[i]*w22[i][j];
sumt+=xt;
}
fxt=func(sumt);
th22[j]=fxt;
}
sumt=0;
for(i=0;i<12;i++)
{
xt=th22[i]*w23[i];
sumt+=xt;
}
z=func(sumt);
```

```
printf("PEG concentration in bottom phase is %lf \n",z);
for(j=0;j<6;j++)      /*3rd net test*/
{
i=0;
xt=t1*w31[i][j]+t2*w31[i+1][j]+t3*w31[i+2][j];
fxt=func(xt);
th31[j]=fxt;
}
for(j=0;j<12;j++)
{
sumt=0;
for(i=0;i<6;i++)
{
xt=th31[i]*w32[i][j];
sumt+=xt;
}
fxt=func(sumt);
th32[j]=fxt;
}
sumt=0;
for(i=0;i<12;i++)
{
xt=th32[i]*w33[i];
sumt+=xt;
}
z=func(sumt);
printf("salt concentration in top phase is %lf \n",z);
for(j=0;j<9;j++)      /*4th net test*/
{
i=0;
xt=t1*w41[i][j]+t2*w41[i+1][j]+t3*w41[i+2][j];
fxt=func(xt);
th41[j]=fxt;
}
for(j=0;j<12;j++)
{
sumt=0;
for(i=0;i<9;i++)
{
xt=th41[i]*w42[i][j];
sumt+=xt;
}
fxt=func(sumt);
th42[j]=fxt;
}
sumt=0;
```

```
for(i=0;i<12;i++)
{
xt=th42[i]*w43[i];
sumt+=xt;
}
z=func(sumt);
printf("PEG concentration in top phase is %lf \n",z);
}
void train2(double i1,double i2,double i3,double o2) /*train 2nd net */
{
double err,y=0.0,x,z,l,fx,sum=0,h21[6],h22[12],o,dh21[6],dh22[12];
int i=0,j,iter=0;
do
{
i=0;
iter++;
for(j=0;j<6;j++)
{
i=0;
x=i1*w21[i][j]+i2*w21[i+1][j]+i3*w21[i+2][j];
fx=func(x);
h21[j]=fx;
}
for(j=0;j<12;j++)
{
sum=0;
for(i=0;i<6;i++)
{
z=h21[i]*w22[i][j];
sum+=z;
}
fx=func(sum);
h22[j]=fx;
}
sum=0;
for(i=0;i<12;i++)
{
x=h22[i]*w23[i];
sum+=x;
}
y=func(sum);
err=o2-y;
if(err<0)
err*=-1;
for(i=0;i<12;i++)
{
```

```
o=y*(1-y)*(o2-y);
w23[i]+=neta*h22[i]*o;
}
for(i=0;i<12;i++)
{
dh22[i]=h22[i]*(1-h22[i])*o*w23[i];
}
for(i=0;i<6;i++)
{
for(j=0;j<12;j++)
{
w22[i][j]+=neta*h21[i]*dh22[j];
}
}
for(i=0;i<6;i++)
{
l=0;
for(j=0;j<12;j++)
{
l+=dh22[j]*w22[i][j];
}
dh21[i]=(h21[i]*(1-h21[i])*l);
}
for(i=0;i<3;i++)
{
for(j=0;j<6;j++)
{
w21[i][j]+=neta*i1*dh21[j];
}
}
}while(err>toler);
printf("the PEG concentration in bottom phase is %lf\n",y);
printf("No. of iterations=%ld\n",iter);
}
void train3(double i1,double i2,double i3,double o3) /*train 3rd net*/
{
double err,y=0.0,x,z,fx,m,sum=0,h31[6],h32[12],o,dh31[6],dh32[12];
int i=0,j,iter=0;
do
{
i=0;
iter++;
for(j=0;j<6;j++)
{
i=0;
x=i1*w31[i][j]+i2*w31[i+1][j]+i3*w31[i+2][j];
```

```
fx=func(x);
h31[j]=fx;
}
for(j=0;j<12;j++)
{
sum=0;
for(i=0;i<6;i++)
{
z=h31[i]*w32[i][j];
sum+=z;
}
fx=func(sum);
h32[j]=fx;
}
sum=0;
for(i=0;i<12;i++)
{
x=h32[i]*w33[i];
sum+=x;
}
y=func(sum);
err=o3-y;
if(err<0)
err*=-1;
for(i=0;i<12;i++)
{
o=y*(1-y)*(o3-y);
w33[i]+=neta*h32[i]*o;
}
for(i=0;i<12;i++)
dh32[i]=h32[i]*(1-h32[i])*o*w33[i];
for(i=0;i<6;i++)
{
for(j=0;j<12;j++)
{
w32[i][j]+=neta*h31[i]*dh32[j];
}
}
for(i=0;i<6;i++)
{
m=0;
for(j=0;j<12;j++)
{
m+=dh32[j]*w32[i][j];
}
dh31[i]=h31[i]*(1-h31[i])*m;
```

```
}
for(i=0;i<3;i++)
{
for(j=0;j<6;j++)
{
w31[i][j]+=neta*i1*dh31[j];
}
}
}while(err>toler);
printf("the salt concentration in the top phase is %lf\n",y);
printf("No. of iterations=%ld\n",iter);
}
void train4(double i1,double i2,double i3,double o4) /*train 4rd net*/
{
double err,n,y=0.0,x,z,fx,sum=0,h41[9],h42[12],o,dh41[9],dh42[12];
int i=0,j,iter=0;
do
{
iter++;
for(j=0;j<9;j++)
{
i=0;
x=i1*w41[i][j]+i2*w41[i+1][j]+i3*w41[i+2][j];
fx=func(x);
h41[j]=fx;
}
for(j=0;j<12;j++)
{
sum=0;
for(i=0;i<9;i++)
{
z=h41[i]*w42[i][j];
sum+=z;
}
fx=func(sum);
h42[j]=fx;
}
sum=0;
for(i=0;i<12;i++)
{
x=h42[i]*w43[i];
sum+=x;
}
y=func(sum);
err=o4-y;
if(err<0)
```

```
err*=-1;
o=y*(1-y)*(o4-y);
for(i=0;i<12;i++)
{
w43[i]+=neta*h42[i]*o;
dh42[i]=h42[i]*(1-h42[i])*o*w43[i];
}
for(i=0;i<9;i++)
{
for(j=0;j<12;j++)
{
w42[i][j]+=neta*h41[i]*dh42[j];
}
}
for(i=0;i<9;i++)
{
n=0;
for(j=0;j<12;j++)
{
n+=dh42[j]*w42[i][j];
}
dh41[i]=h41[i]*(1-h41[i])*n;
}
for(i=0;i<3;i++)
{
for(j=0;j<9;j++)
{
w41[i][j]+=neta*i1*dh41[j];
}
}
}while(err>toler);
printf("the PEG concentration in the top phase is %lf\n",y);
printf("No. of iterations=%ld\n",iter);
}
```

APPENDIX-4

CONVERGED WEIGHTS

W11[3][12]

.161489	.532561	.055188	.300368	.402127	.407684	.345041	.357798	.451189	.150871	.167958	.095595
.445269	.601901	.414268	.492188	.514507	.504104	.618121	.072478	.089869	.370731	.236958	.064775
.284649	.420041	.329848	.254388	.489567	350224	.221781	.024958	.300309	.516591	.430618	.385935

W12[12][3]

.549683	.474633	.538771
.510652	.640832	.381437
.612604	.149949	.216964
.476216	.203365	.659980
.290842	.324790	.349648
.136760	.526242	.343572
.091809	.116352	.056286
.188197	.150300	.147389
.533081	.475458	.044698
.340513	.291246	.437104
.363486	.488551	.498404
.368184	.150461	.064521

W13[3]

-1.030382	-0.525338	-0.730276

W21[3][6]

.431317	.589983	.587316	.120489	.365797	.628021
.323917	.193783	.125256	.314969	.009677	.466641
.283637	.538463	.320776	.379589	.055537	.421181

W22[6][12]

.495766	.246097	.515551	.509585	.079229	.610499	.495877	.029614	.089303	.342919	614155	.369342
.108506	.339753	.379653	.053812	.246069	.067236	.362662	.607972	.021010	.594426	.243002	.311555
.343298	.636666	.257968	.543627	.316726	.608958	.039522	.089427	.115754	.477501	.388140	.518875
.127928	.571756	.120687	.514661	.158816	.336835	.645248	.048622	.478899	.169528	.550077	.490019
.471081	.088164	.195108	.495327	.313067	.381670	.459979	.368415	.564470	.516963	.522545	.643447
.405757	.247694	.599287	.337457	.182212	.478275	.642553	.640667	.313644	.568284	.252427	.184843

W23[12]

-.508877	-.427694	-.514548	-.610573	-.045034	-.240430	-.504676	-.279229	-.118208

-.381526	-.360561	-.199803

W31[3][6]

.163019	.183814	.596532	.552203	.413421	.187706
.515379	.209794	.601772	.363383	.130021	.184626
.539219	.105434	.100232	.343043	.027261	.128366

W32[6][12]

.368604	.330501	.237998	.569485	.383555	.467477	.536643	.033577	.081445	.614800	.525847	.189184
.566774	.600061	.431857	.339658	.370934	.554183	.500601	.469244	.473391	.392280	.509388	.597173
.237975	.363954	.635018	.575180	.458895	.380549	.479849	.503851	.418481	.434483	.163054	.526486
.622111	.352030	.518893	.513298	.539611	.263266	.134424	.145348	.545877	.435118	.071230	.425301
.033530	.392130	.488641	.287018	.127336	.397342	.361529	.243697	.321381	.498601	.310010	.049841
.338775	.197718	.293880	.062050	.191148	.228474	.445287	.242345	.598872	.388281	.091243	.503740

W33[12]

-.328244	-.233739	-.452365	-.034604	-.096188	-.044298	-.525313	-.310723

-.370016	-.393133	-.138138	-.539364

W41[3][9]

.140956	.493397	.298279	.442398	.344335	.448327	.490095	.528143	.415802
.389096	.128437	.319559	.068898	.480995	.544747	.203035	.365163	.617109
.188896	.215877	.443239	.273538	.142815	.093187	.318855	.335443	.141382

W42[9[12]

.267290	.328196	.020240	.082827	.248957	.599769	.510610	.626729	.084579	.134868	.379066	.373762
.390828	.318160	.210501	.011796	.485971	.185209	.115059	.428405	.256888	.171863	.385465	.367867
.504853	.119326	.228373	.399004	.582640	.643231	.249900	.549949	.521570	.064274	.383147	.132622
.093090	.528054	.056071	.285327	.509625	.634217	.076445	.225339	.560153	.520885	.237474	.577920
.267561	.596067	.400388	.335700	.411031	.394675	.292944	.263568	.508520	.248256	.202539	.354611
.172231	.388423	.175663	.379149	.358416	.360620	.490992	.389415	.233831	.451962	.015639	.194181
.137432	.081753	.216962	.938751	.276874	.290927	.226641	.599513	.393940	.566123	.214371	.182388
.045152	.345821	.195016	.171372	.639038	.226098	.316595	.492031	.305007	.628448	.542450	.639009
.057965	.095972	.300366	.305399	.228863	.353614	.526888	.032283	.439551	.493885	.564525	.563096

W43[12]

-.461861	-.061123	.116633	-.416144	-.405206	-.133234	.097569	-.289616

.043005	-.413506	.033338	-.196900