Calcul neuronal și evolutiv.

Lab 2: Probleme de aproximare și predicție.
Implementarea în Scilab a rețelelor neuronale feedforward cu un nivel ascuns și a rețelelor RBF

---

1. Probleme de aproximare și predicție

Scop: determinarea unui model care descrie dependența intre niște date de intrare (predictori) și niște date de ieșire (rezultate).

Exemple:

- Analiza datelor experimentale:
  - Datele de intrare sunt rezultate ale măsurătorilor efectuate asupra uneia/unor mărimi (de exemplu caracteristici ale unui calculator/ masini/ case)
  - Rezultatele sunt valori estimate pentru altă/alte mărimi despre care se consideră că sunt corelate cu primele (de exemplu prețul calculatorului/ masinii/ casei)
- Predicție în serii temporale:
  - Datele de intrare sunt valori anterioare ale unei mărimi care variază în timp (de exemplu temperatura, numărul de accesări ale unei resurse web, cursul de schimb valutar etc.)
  - Rezultatul este o estimare a valorii următoare din serie

2. Utilizarea rețelelor neuronale de tip feedforward pentru aproximare și predicție

Etape:

a) Construirea setului de antrenare (depinde de problema de rezolvat)
b) Stabilirea arhitecturii rețelei (număr de nivele, număr de unități pe fiecare nivel, funcții de activare)
c) Stabilirea parametrilor algoritmului de antrenare: rata de învățare, acuratețe (nivel toleranță la eroare), număr maxim de epoci de antrenare).
d) Antrenare propriu-zisă
e) Testarea rețelei

Implementarea rețelelor feedforward antrenate cu BackPropagation – funcții SciLab (www.scilab.org) pentru crearea și antrenarea unei rețele cu un nivel ascuns

Crearea rețelei (rețea feedforward cu un nivel ascuns):

```
function network=NNcreate(N, K, M, activationFunction1, activationFunction2)
  network=tlist(["Retea feedforward", "N", "K", "M", "X0", "Y0", "X1", "Y1", "X2", "Y2", "W1", "W2",
                 "f1", "f2"], N, K, M, zeros(N+1,1), zeros(N+1,1), zeros(K,1), zeros(K+1,1), zeros(M,1),
                  zeros(M,1), zeros(K,N+1),zeros(M,K+1), activationFunction1,activationFunction2);
endfunction
```

Funcții de activare (utilizate în rețelele neuronale antrenate cu algoritmul BackPropagation)

```
// functie de activare:  logistica
function output=logistic(x)
```

```
  output=1/(1+exp(-x));
endfunction
```

```
// functie de activare:  liniara
function output=linear(x)
  output=x;
endfunction
```

Obs. Funcția tanh este predefinită în SciLab

Calculul semnalului de ieșire (etapa Forward din algoritmul BackPropagation):

```
function network=forward(network, X)
 network.X0=[-1;X]; network.Y0=network.X0;
 network.X1=network.W1*network.Y0;
 network.Y1=[-1; feval(network.X1,network.f1)];
 network.X2=network.W2*network.Y1;
 network.Y2=feval(network.X2,network.f2);
endfunction
```

Calculul erorii medii pătratice:

```
function err=error_computation(network, tset)
err=0;
L=tset.L;
for i=1:L
    network=forward(network,tset.X(:,i));
    delta=(tset.d(i)-network.Y2).^2;
    err=err+sum(delta);
end
err=err/L;
endfunction
```

Ajustarea ponderilor (algoritmul BackPropagation):

```
function [network,ap,aE]=BP(network, tset, pmax, Emax, eta)
L=tset.L;
// initializarea ponderilor cu valori aleatoare in [-1,1]
network.W1=2*rand(network.W1)-ones(network.W1);
network.W2=2*rand(network.W2)-ones(network.W2);
E=error_computation(network,tset); // calcul eroare
p=0;   // initializare contor epoca de antrenare
ap=[]; aE=[];
while p<pmax & E>Emax
  for l=1:L
    network=forward(network,tset.X(:,l));
    delta2=tset.d(l)-network.Y2;   // calcul eroare la nivel de iesire
    if (network.f2==logistic) delta2=delta2.*(network.Y2.*(ones(network.Y2)-network.Y2)); end;
    if (network.f2==tanh) delta2=delta2.*(ones(network.Y2)-network.Y2.^2); end;
    delta=network.W2'*delta2; // propagare eroare
    if (network.f1==logistic) delta=delta.*(network.Y1.*(ones(network.Y1)-network.Y1)); end;
    if (network.f1==tanh) delta=delta.*(ones(network.Y1)-network.Y1.^2); end;
    delta1=delta(2:length(delta)); // ignorarea componentei fictive
    network.W1=network.W1+eta*(delta1*network.Y0');
```

```
    network.W2=network.W2+eta*(delta2*network.Y1');
  end;
  E=error_computation(network,tset);
  disp(E,"Eroare=",p,"p=");
  ap=[ap p]; aE=[aE E];
  p=p+1;  // incrementarea contorului de epoca
 end
endfunction
```

Aplicație 1. Reprezentarea funcției XOR (fișier XOR_BP.sci)

Construirea setului de antrenare:

```
function tset=trainingSet()
  tset=tlist(["Set antrenare","L","X","d"],4,[-1 -1 1 1;-1 1 -1 1],[-1 1 1 -1]);
endfunction
```

Aproximarea funcției XOR utilizând o rețea neuronală cu două unități ascunse și funcții de activare de tip tanh

```
function rna=approximation_XOR(hiddenUnits, eta, epochs)
rn=NNcreate(2,hiddenUnits,1,tanh,tanh); // crearea retelei
tset=trainingSet();                  // construirea setului de antrenare
rna=BP(rn,tset,epochs,0.0001,eta);      // antrenarea retelei
// testarea retelei
for i=1:tset.L;
   rna=forward(rna,tset.X(:,i));
   disp(tset.X(:,i),"input=");       // afisare valori de intrare
   disp(rna.Y2,"output=");           // afisare valoare de iesire
end
endfunction
```

Apelul funcției de aproximare:

```
approximation_XOR(10,0.1,1000);
```

Exerciții:

1.Analizați influența numărului de unități ascunse și a ratei de învățare asupra performanței rețelei (Indicație: valori de testat pentru hiddenUnits: 1,2,5,20; valori de testat pentru eta: 0.01,0.1,0.5)

2. Analizați influența funcțiilor de activare asupra abilității rețelei de a aproxima XOR: (logistic, logistic), (tanh,logistic), (logistic, tanh). (Indicație: în cazul în care se utilizeaza funcția logistică la nivelul de ieșire în setul de antrenare răspunsurile corecte trebuie să fie [0,1,1,0] în loc de [-1,1,1,-1]

Aplicație 2. Aproximarea unei funcții neliniare (f:[a,b]->R) pornind de la un set de puncte corespunzătoare graficului său. (fișier regresieBP.sci)

Preluarea datelor de intrare (puncte aflate în apropierea graficului funcției de aproximat) – punctele se selectează folosind butonul drept al mouse-ului; finalizarea selectării punctelor se realizează folosind butonul din stânga al mouse-ului):

```
// construirea setului de antrenare
function tset=trainingSet()
  tset=tlist(["Training set","L","X","d"],0,zeros(1,100),zeros(1,100));
  clf;
  plot2d(0,0,0,rect=[0,0,10,10]); // definirea domeniului si a codomeniului
  xgrid(3);
  x=locate(-1,1);
  tset.L=size(x,2);
  tset.X(1:tset.L) = x(1,:);
  tset.d(1:tset.L) = x(2,:);
endfunction
```

Aproximarea funcției pornind de la punctele selectate:

```
function tset=regression(hiddenUnits, eta, epochs)
rn=NNcreate(1,hiddenUnits,1,logistic,linear); // crearea retelei
disp("dupa creare retea");
tset=trainingSet();          // construirea setului de antrenare
inf=min(tset.X); sup=max(tset.X);
rna=BP(rn,tset,epochs,0.0001,eta);  // antrenarea retelei
// testarea retelei
x=inf:(sup-inf)/100.:sup;
y=[];
for i=1:size(x,2);
   rna=forward(rna,x(1,i));
   y=[y rna.Y2];
end
plot(tset.X,tset.d,'b*',x,y,'r-');
endfunction
```

Apelul funcției de regresie:

```
regression(10,0.01,10000)
```

Exerciții:

1.Analizați influența numărului de unități ascunse și a ratei de învățare asupra performanței rețelei (Indicație: valori de testat pentru hiddenUnits: 1,2,5,20; valori de testat pentru eta: 0.01,0.1,0.5)

2. Analizați influența funcției de activare de la nivelul ascuns (tanh în loc de logistic)

Aplicație 3. Aproximarea unei serii temporale (fișier predictieBP.sci)

Considerăm o succesiune de valori (serie temporală): $x_1,x_2,\ldots,x_n$ care pot fi interpretate ca valori înregistrate la momente succesive de timp. Scopul este să se estimeze valoarea corespunzătoare momentului (n+1). Ideea principală este de a presupune că valoarea x_i depinde de N valori anterioare: x_{i-1}, $x_{i-2},\ldots,x_{i-N}$. Pe baza acestei ipoteze se poate proiecta o rețea neuronală care să învețe modelul dependenței dintre o valoare din serie și N valori anterioare. Rețeaua va avea N unități de intrare, un anumit număr de unități ascunse și o unitate de ieșire. Setul de antrenare va avea L=n-N perechi de forma (data intrare, răspuns corect) adică va fi: $\{((x_1,x_2,\ldots,x_N),x_{N+1}),((x_2,x_3,\ldots,x_{N+1}),x_{N+2}),\ldots,((x_{n-N},x_{n-N+1},\ldots,x_{n-1}),x_n)\}$.

Presupunem că într-un fișier text se află valori ale cursului valutar înregistrate într-o anumită perioadă. Pe fiecare linie din fișier se află o valoare, iar prima linie corespunde celei mai recente valori înregistrate.

Construirea setului de antrenare:

```
// construirea setului de antrenare
function tset=trainingSet(inputUnits)
  tset=tlist(["Training set","L","X","d"],0,zeros(1,100),zeros(1,100));
  date1=csvRead("cursEuroZilnic.csv")';
  date2(1:length(date1))=date1(length(date1):-1:1);
  tset.L=size(date2,2)-inputUnits;
  tset.X=zeros(inputUnits,tset.L);
  tset.d=zeros(1,tset.L);
  for i=1:tset.L
    tset.X(:,i)=date2(1,i:i+inputUnits-1)';
    tset.d(i)=date2(1,i+inputUnits);
  end
endfunction
```

Crearea, antrenarea și vizualizarea modelului descoperit de rețea:

```
function tset=prediction(inputUnits, hiddenUnits, eta, epochs)
rn=NNcreate(inputUnits,hiddenUnits,1,logistic,linear); // crearea retelei
tset=trainingSet(inputUnits);          // construirea setului de antrenare
rna=BP(rn,tset,epochs,0.0001,eta);  // antrenarea retelei
y=[];
for i=1:tset.L;
   rna=forward(rna,tset.X(:,i));
   y=[y rna.Y2];
end
plot(tset.d,'b-',y,'r-');
endfunction
```

Implementarea rețelelor RBF – funcții SciLab pentru crearea și antrenarea unei rețele cu un nivel ascuns

Crearea unei rețele RBF

```
function network=RBFcreate(N, K, M, sigma)
  network=tlist(["RBF NN", "N", "K", "M", "X0", "Y0", "X1", "Y1", "X2", "Y2", "C", "W", "f", "sigma"],
                N, K, M, zeros(N,1), zeros(N,1), zeros(K,1), zeros(K+1,1), zeros(M,1), zeros(M,1),
                zeros(K,N), zeros(M,K+1), gaussian,sigma);
endfunction
```

Funcție de activare:

```
function output=gaussian(x)
  output=exp(-x^2/2);
endfunction
```

Funcție de agregare:

```
function d=dist(x, y)
  d=sqrt((x-y)'*(x-y));
endfunction
```

Calcul semnal de ieșire:

```
function network=forward(network, X)
 network.X0=X; network.Y0=network.X0;
 for k=1:network.K
   network.X1(k)=dist(network.C(k)',network.Y0);
 end
 network.Y1=[-1; feval(network.X1./network.sigma,network.f)];
 network.X2=network.W*network.Y1;
 network.Y2=network.X2;
endfunction
```

Calcul funcție de eroare:

```
function err=error_computation(network, tset)
err=0;
[N,L]=size(tset.X);
for i=1:L
    network=forward(network,tset.X(:,i));
    delta=(tset.d(i)-network.Y2).^2;
    err=err+sum(delta);
end
err=err/L
endfunction
```

Algoritm de antrenare (centrii coincid cu datele din setul de antrenare)

```
function [network, ap, aE]=train(network, tset, pmax, Emax, eta)
L=tset.L;
network.C=tset.X';
network.W=2*rand(network.W)-ones(network.W);
E=error_computation(network,tset);
p=0;
ap=[];aE=[];
while p<pmax & E>Emax
  E=error_computation(network,tset);
  for i=1:L
     network=forward(network,tset.X(:,i));
     y=network.W*network.Y1;
     delta=tset.d(i)-y;
     network.W=network.W+eta*(delta*network.Y1')
  end;
  E=error_computation(network,tset);
  if (modulo(p,10)==0) then
      disp(p,"Iteration:");
      disp(E,"error=");
      ap=[ap p]; aE=[aE E];
  end
  p=p+1;
 end
 disp(E,"error=");
endfunction
```

Exemplu: regresie neliniară (fisier RBFnetwork.sci)

Construire set de antrenare:

```
function tset=trainingSet()
   tset=tlist(["Training set","L","X","d"],0,zeros(1,100),zeros(1,100));
   tset.X=[0:1:10*%pi];
   tset.L=length(tset.X);
   tset.d=sin(tset.X)+0.2*rand(tset.X)-0.1*ones(tset.X);
endfunction
```

Creare rețea, antrenare, vizualizare funcție aproximată, vizualizare eroare:

```
function tset=regressionRBF(eta, epochs, sigma)
tset=trainingSet();          // construirea setului de antrenare
rbf=RBFcreate(1,length(tset.X),1,sigma); // crearea retelei
disp(tset,"tset=");
[rbfa,ap,aE]=train(rbf,tset,epochs,0.0001,eta); // antrenarea retelei
// testarea retelei
test_plot(tset,rbfa);
pause;    // pauza in executie;  se continua tastand <resume>
clf;
plot(ap,aE);
endfunction
```

Temă:

Să se modifice algoritmul de antrenare al unei rețele RBF astfel încât numărul de centri și vectorii prototip să fie stabiliți în mod dinamic (algoritm de antrenare incrementală – curs 2-3 slide 62).