Gør det selv...et programeksempel

Jan Larsen og Jacob Nordfalk

Nedenstående program er skrevet i Turbo Pascal, og er nok et af de simpleste programmer, man kan lave som bruger backpropagation. Netværket er en implementering af en logisk operation der tæller antallet af "tændte" inputneuroner efter følgende sandhedstabel:

input	ønsket output
000	000
001 010 100	001
011 101 110	010
111	100

Input og output er skrevet direkte i programmet, men man kan naturligvis også hente dem ind fra en fil, bruge en funktion til at generere dem eller hvad man nu kan finde på.
Det er let at ændre de forskellige værdier af input og output, men hvis man ændrer antallet af dem skal man lige huske at ændre N1 og N3 tilsvarende. N2 er antallet af neuroner i det skjulte lag. Hvis man ændrer på N2, er der to ting man skal tænke på: Hvis antallet af skjulte neuroner er for lille kan nettet ikke lære alle eksemplerne fordi man smider for meget information væk, når man går fra input laget til det skjulte. Hvis man bruger for mange skjulte neuroner, tager det til gengæld alt for lang tid at træne nettet.
Programmet er delt op i to procedurer der udfører Feedforward og Backpropagation, to funktioner der beregner f og f_maerke, og et hovedprogram. Desuden er der en masse parameterdeklarationer, hvor bl.a. input og output står. Hovedprogrammet initialiserer først vægte og tærskler, og kalder så skiftevis Feedforward og Backpropagation, mens det løbende udskriver antallet af gennemløb, antallet af fejl i hvert eksempel, og den samlede fejl

Jan Larsen stud.scient (fysik), intereserer sig for tiden for simuleret udglødning.
Jacob Nordfalk stud.scient(fysik), interesserer sig for inklusioner og UNIX.

```
uses Crt;
const
  N1 = 3;  { antal input-"neuroner" }
  N2 = 3;  { antal neuroner i det skjulte lag. Prøv at ændre her. }
  N3 = 3;  { antal neuroner i output-laget }
  E  = 7;  { antal eksempler. Prøv at ændre til nogle andre eksempler }
  input: array[1..E,1..N1] of integer =
  ((0,0,1), (0,1,0), (1,0,0), (0,1,1), (1,0,1), (1,1,0), (1,1,1));
  forv_out: array[1..E,1..N3] of integer =
  ((0,0,1), (0,0,1), (0,0,1), (0,1,0), (0,1,0), (0,1,0), (1,0,0));
  eta = 1; { hvor kraftigt skal der reageres på fejl. Prøv at ændre her. }
var
  { data i netværket }
  W12: array[1..N1,1..N2] of real;      { vægte mellem input og skjult lag }
  th2: array[1..N2] of real;        { threshold på neuronerne i skjult lag }
  W23: array[1..N2,1..N3] of real;     { vægte mellem skjult og output lag }
  th3: array[1..N3] of real;      { threshold på neuronerne i output-laget }
  { output fra neuronerne }
  O2 : array[1..N2] of real; { output fra skjult lag = input til outputlag }
  out : array[1..N3] of real;                          { output fra netværket }
  { midlertidige variable til træning af netværket }
  h2  :array[1..N2] of real;                     { samlet input til neuron }
  d2  :real;
  h3 : array[1..N3] of real;                     { samlet input til neuron }
  d3 : array[1..N3] of real;
  i,j,k : integer;   { i er index lag 1, j for lag 2, og k index for lag 3 }
  ex    : integer;                                 { index for eksemplerne }
  iter  : integer;   { antal gange eksemplerne er præsenteret for netværket}
  sum,tmp,error,fejl : real;

function f(x:real):real;{Kvasefunktionen, der afgører om en neuron "fyrer" }
  begin f := 1/(1+exp(x));end;

function f_maerke(x:real):real;
begin f_maerke:= -exp(x)/sqr(exp(x)+1); end;

procedure Feedforward;
begin
  { feedforward fra input[ex,i] til O2[j] }
  for j := 1 to N2 do         { loop over neuronerne i det skjulte lag }
  begin
    sum := 0;
    for i := 1 to N1 do sum := sum + input[ex,i] * W12[i,j];
    sum := sum - th2[j];
    h2[j] := sum;                 { h2 bruges senere i Backpropagation }
    O2[j] := f(sum);
  end;
  { feedforward fra O2[j] til out[k] }
  for k := 1 to N3 do           { loop over neuronerne i output-laget }
  begin
    sum := 0;
    for j := 1 to N2 do sum := sum + O2[j] * W23[j,k];
    sum := sum - th3[k];
    h3[k] := sum;                 { h3 bruges senere i Backpropagation }
    out[k] := f(sum);             { her beregnes output fra nettet }
  end;
end;
```

Fortsættes side 28.

"...et programeksempel". Fortsat fra side 19.

```
procedure Backpropagation;
begin
  for k := 1 to N3 do   { fejlberegning og backpropagation i outputlag }
  begin
    tmp := out[k] - forv_out[ex,k]; { afv. mellem faktisk og ønsket output }
    error := error + sqr(tmp);
    d3[k] := eta * tmp * f_maerke( h3[k] );
    th3[k] := th3[k] + d3[k];
    for j := 1 to N2 do W23[j,k] := W23[j,k] - d3[k] * O2[j];
  end;
  for j := 1 to N2 do                    { backpropagation i skjult lag }
  begin
    d2 := 0;
    for k := 1 to N3 do d2 := d2 + d3[k]* W23[j,k];
    d2 := d2 * f_maerke( h2[j] );
    th2[j] := th2[j] + d2;
    for i := 1 to N1 do W12[i,j] := W12[i,j] - d2 * input[ex,i] ;
  end;
end;
{ ---------------------------- HOVEDPROGRAM ---------------------------- }
begin
  iter  := 0;
  randomize;           { vægtene skal initialiseres med tilfældige værdier }
  for i := 1 to N1 do for j := 1 to N2 do W12[i,j]:= (random(200)-99.5)/100;
  for j := 1 to N2 do                     th2[j]  := (random(200)-99.5)/100;
  for j := 1 to N2 do for k := 1 to N3 do W23[j,k]:= (random(200)-99.5)/100;
  for k := 1 to N3 do                     th3[k]  := (random(200)-99.5)/100;
  repeat
    error := 0;                                                  { nulstil }
    write('gennemløb nr:',iter:6,'  -  fejl:');
    for ex := 1 to E do                                { loop over eksempler }
    begin
      Feedforward;
      fejl := 0;                       { tæl op hvor mange fejl nettet laver }
      for k := 1 to N3 do
      if ((out[k] < 0.5) AND (forv_out[ex,k] > 0.5)) OR
         ((out[k] > 0.5) AND (forv_out[ex,k] < 0.5)) then fejl:= fejl+ 1;
      write(fejl:2);
      Backpropagation;
    end; { loop over eksempler }
    error := error/N3/E;                    { fejl per neuron per eksempel }
    writeln('  - samlet fejl:',error:6:5);
    Inc( iter );
  until keypressed;
  writeln;                                { udskrift af vægte og thresholds }
  writeln('mellem <<-- input');
  writeln('Thres.':8,' |   Vægte:');
  for j := 1 to N2 do
  begin
    write(th2[j]:8:3,' |');
    for i := 1 to N1 do write(W12[i,j]:8:3);
    writeln;
  end;
  writeln;
  writeln('output <<-- mellem');
  writeln('Thres.':8,' |   Vægte:');
  for k := 1 to N3 do
  begin
    write(th3[k]:8:3,' |');
    for j := 1 to N2 do write(W23[j,k]:8:3);
    writeln;
  end;
end.
```