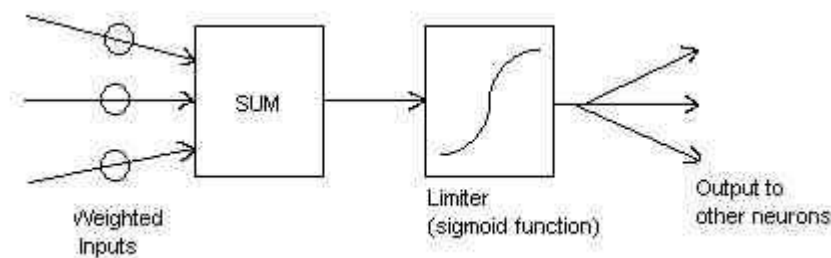
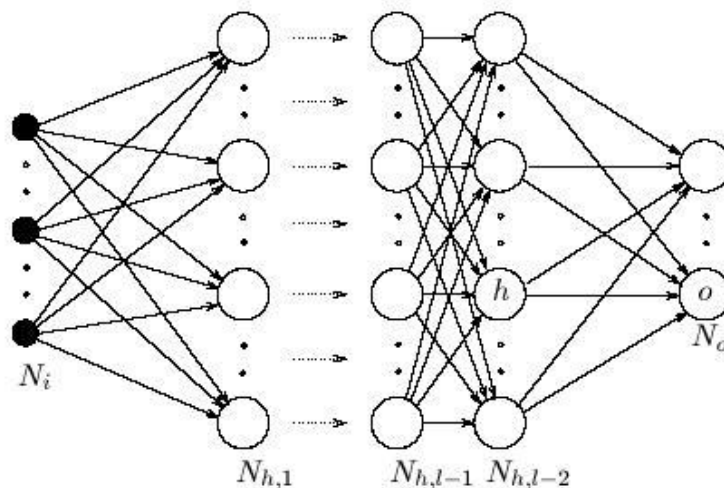


Neironu tīkli

Back propagation

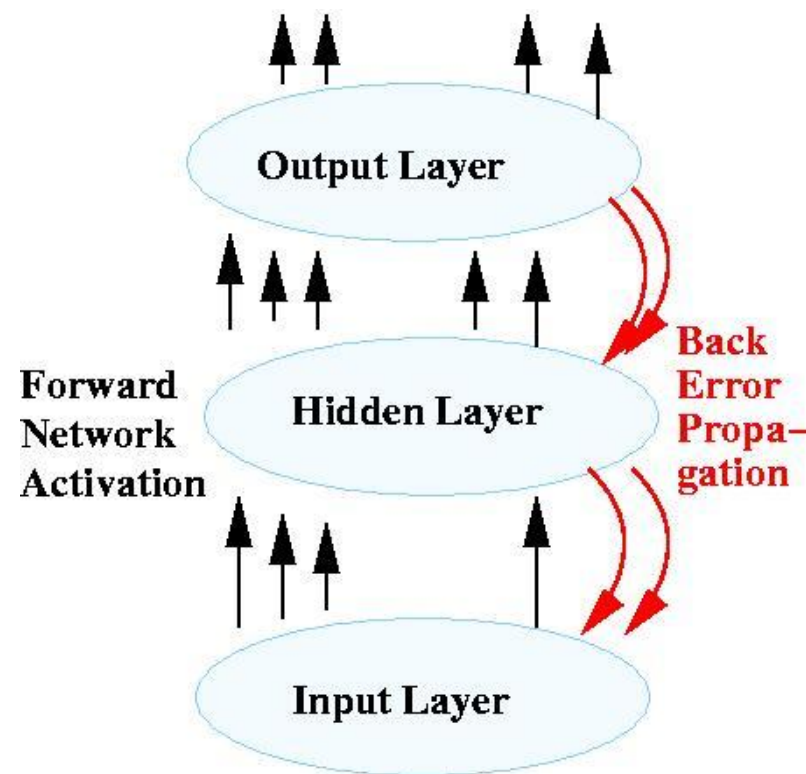
Ievads

- “Back propagation” algoritms tiek izmantots daudzslāņu neironu tīklos. Tas nozīmē, ka paslēpto slāņu jābūt vismaz 2.
- Tiek izmantota aktivācijas funkcija, lai noteikt katra neirona izejas vērtību.



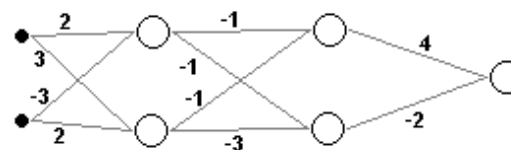
Kāda ir “back propagation” jēga?

- Vispirms ievada datus tīklā, un saskaita rezultātu.
- Ja rezultāts atšķiras no vēlamā tad tīkla katram neironam tiek izrēķināta kļūda, un neirona svāri tiek adaptēti.



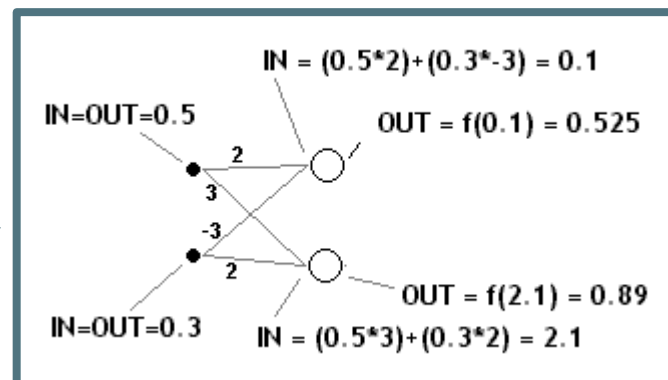
Piemērs

- Tīkla topoloģija 2-2-2-1
- Cipari virs saitēm – svari.
- Lai uz ieejam tiek padotas vērtības 0.5 un 0.3. Mūsu uzdevums, noteikt tīkla rezultātu, ņemot vērā, ka mēs izmantojam loģistisko aktivācijas funkciju.

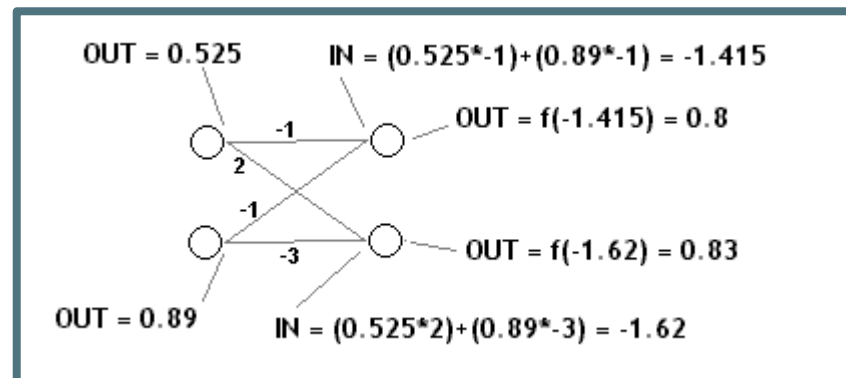


Piemērs

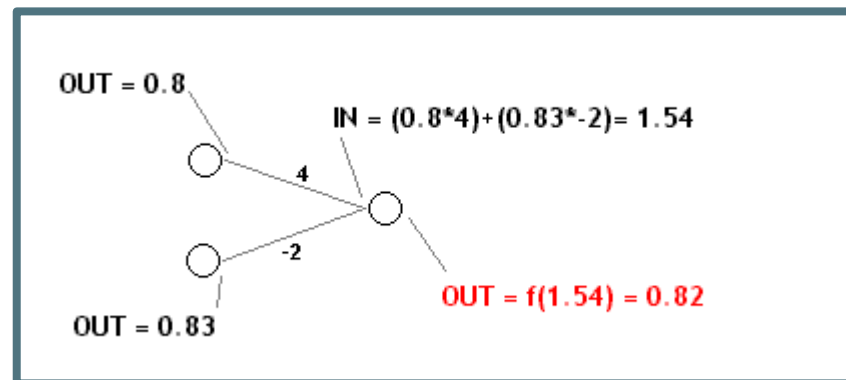
- No 1. slāņa uz 2. slāni
(IN – kombinētā ieeja,
OUT – neironu
aktivācijas rezultāts)



- No 2. līdz 3.



- No 3. slāņa uz izejas
slāni.

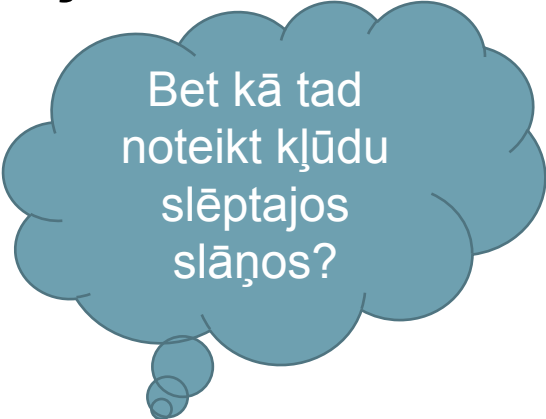


Piemērs

- Tagad pieņemsim, ka mēs gribējām, lai tīkls mums izdotu rezultātu 1. Tagad mums to jāapmāca (jāadaptē svarus), lai rezultātā tas mums izdotu vēlamu vērtību.
- Pirmkārt, mums jānosaka katra neirona kļūdu, lai pēc tam adaptēt svarus atbilstoši kļūdai.

$$\delta_n = (t_n - o_n) \cdot o_n (1 - o_n)$$

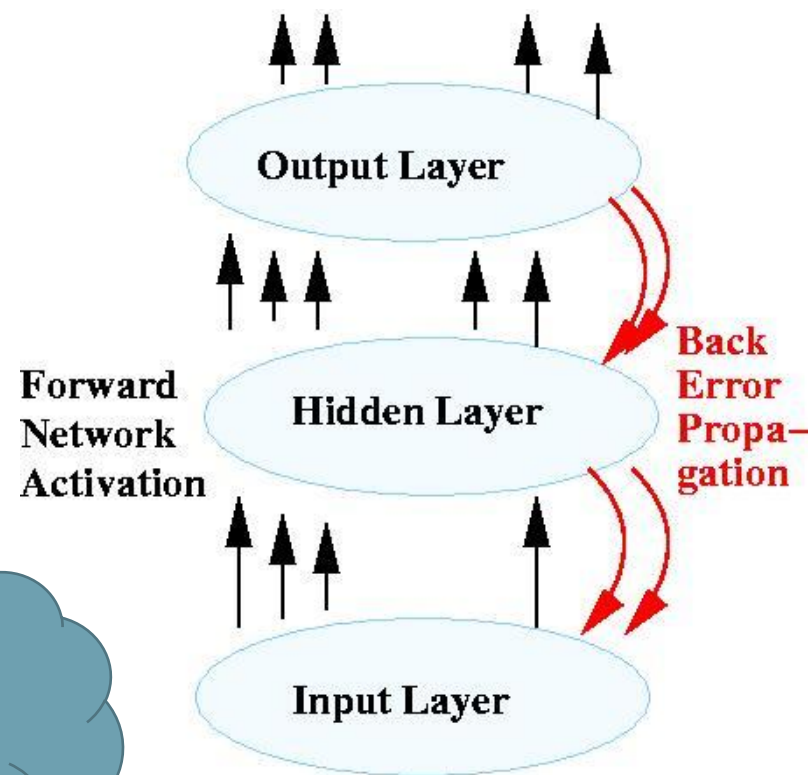
- t_n – vēlamais rezultāts
- o_n – tas, ko mēs dabūjām izejā



Bet kā tad
noteikt kļūdu
slēptajos
slāņos?

Piemērs

- Te arī stājas spēkā “back propagation”. Atpakaļvirziena kļūdu izplatīšanās. Mēs izmantojam kļūdas iepriekšējā augstākajā slānī, lai noteikt kļūdas zemākajā slānī.

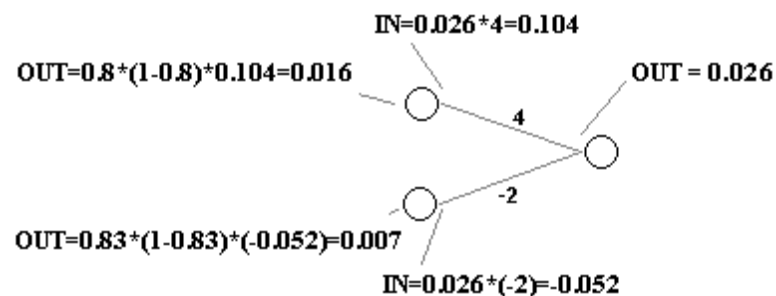


Piemērs

- Tātad, sākam no gala.
Tīkls izdod mums rezultātu 0.82, bet mums vajag 1.

$$\begin{aligned}\delta_n &= (t_n - o_n) \cdot o_n(1 - o_n) = \\ &= (1 - 0.82) \cdot 0.82 \cdot (1 - 0.82) = \\ &= 0.026\end{aligned}$$

- Izmantojam šo kļūdu tālāk, iepriekšējā slānī (neaizmirstam par svariem!)

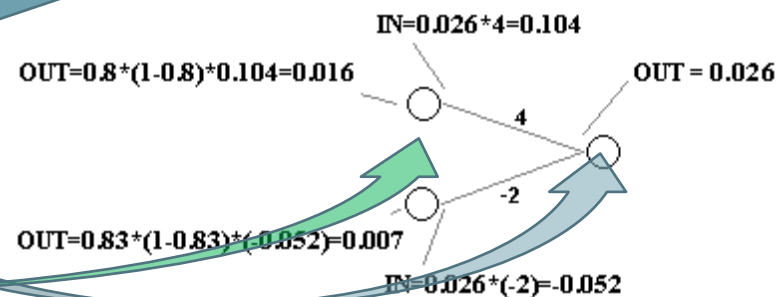


Piemērs

- IN – kļūda iet atpakaļ, iepriekšējā neironā
- OUT – kļūda neironā, tā ir vienāda ar

$$\delta_m = o_m(1 - o_m) \cdot \delta_n \cdot w_{m \rightarrow n}$$

Kļūda augstākajā slānī

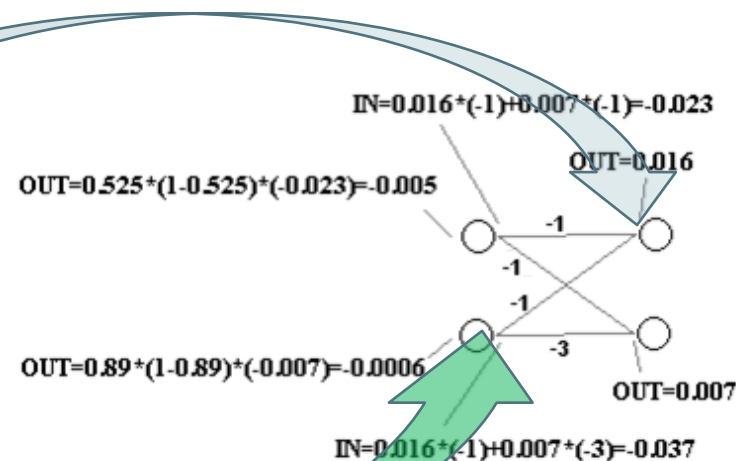


Piemērs

- IN – kļūda iet atpakaļ, iepriekšējā neuronā
- OUT – kļūda neuronā, tā ir vienāda ar

$$\delta_m = o_m(1 - o_m) \cdot \sum_{i=1}^N \delta_{n[i]} \cdot w_{m \rightarrow n[i]}$$

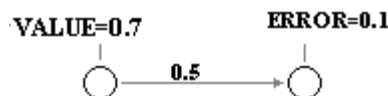
- N – neironu skaits slānī n



Piemērs

- Kad kļūdas katram neironam ir zināmas, varam ķerties pie svaru adaptācijas.
- Piemēram, ir neirons m ar vērtību 0.7, svars saitē 0.5, un kļūda neironā n ir 0.1. Tad jauna svara vērtība:

$$w_{m \rightarrow n[i]} = w_{m \rightarrow n[i]} + \eta \cdot \delta_n \cdot o_m$$



Algoritms

1. Nolasām ievaddatus
2. Ejam uz priekšu – skaitam rezultātu tīklam
3. Ja rezultāts neatbilst patiesībai (vai atšķiras no tās vairāk par uzdoto sliekšni) tad pāriet uz soli 4, pretējā gadījumā tīkls ir apmācīts (BEIGAS)
4. Noteikt kļūdas neironiem izejas slānī (rezultāta kļūda)
5. Noteikt kļūdas neironiem visos slēptos slānos (var arī ieejas slānī)
6. Adaptēt visus svarus
 - Atceramies, apmācība notiek līdz tam brīdim, kamēr VISI apmācošās izlases objekti tiek pareizi atpazīti.

Tīkla struktūra un inicializācija

TNeuron = record

value: single;

error: single;

outputWeights: array of single;

end;

TLayer = record

layer: array of TNeuron;

numberOfNeurons: integer;

end;

var NN: array [0..3] of TLayer;

procedure TForm1.NN_Initialize;

var i, j: integer;

begin

for i:=0 to High(NN) do

begin

if i<High(NN) then NN[i].numberOfNeurons:=60
else NN[i].numberOfNeurons:=1;

SetLength(NN[i].layer, NN[i].numberOfNeurons);

end;

for i:=0 to High(NN) do

for j:=0 to NN[i].numberOfNeurons-1 do with
NN[i].layer[j] do

begin

value:=0;

if i<High(NN) then setLength(outputWeights,
NN[i+1].numberOfNeurons);

end;

end;

Nejaušie svāri un ievaddati jaunā tīkla struktūrā

```
procedure TForm1.NN_Random_W();
var i,j,k :integer;
begin
  Randomize;
  for i:=0 to 2 do
    for j:=0 to 59 do
      for k:=0 to NN[i+1].numberOfNeurons-1 do
        begin
          NN[i].layer[j].outputWeights[k]:=1-2*Random;

          TStringGrid(FindComponent('sgw'+IntToStr(i))).Cells[k+1,j
+1]:=FloatToStr(NN[i].layer[j].outputWeights[k]);
        end;
      end;
    end;
  end;
```

```
procedure TForm1.NN_Input_X();
var sb,i,j:Integer;
    sbName: String;
begin
  for sb:=1 to nClass1+nClass2 do
    if (TSpeedButton(FindComponent('sb'+IntToStr(sb))).Down)
    then
      begin
        if sb<=nClass1 then currentClass:=0 else
          currentClass:=1;
        for i:=0 to 5 do
          for j:=0 to 9 do
            begin
              if
                TSpeedButton(FindComponent('sb'+IntToStr(sb))).Glyph.C
anvas.Pixels[i,j]=clBlack then NN[0].layer[i+6*j].value:=1
                else NN[0].layer[i+6*j].value:=0;
            end;
          end;
        end;
      end;
    for i:=0 to 59 do
      begin
        sg0.Cells[1,i+1]:=FloatToStr(NN[0].layer[i].value);
      end;
    end;
  End;
```

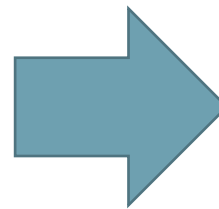
Programmēšana

- function
NN_Sumimator(which
Layer, whichNeuron:
integer):real;

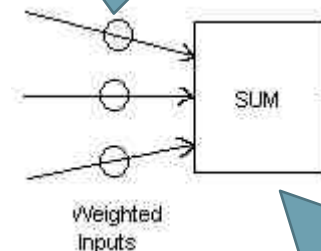
$$\text{sum}_n = \sum_{m=1}^M o_m w_{m \rightarrow n}$$

- function
NN_Activation_Functi
on(input: real):real;

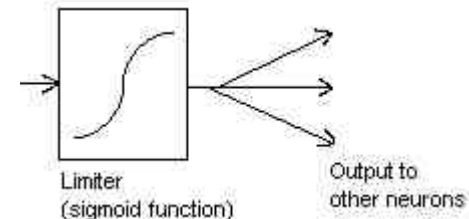
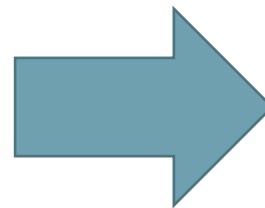
$$f(\text{sum}_n) = \frac{1}{1 + \exp(-\text{sum}_n)}$$



Mēs summējam
neironus*svarus slānī
whichLayer-1, jeb m

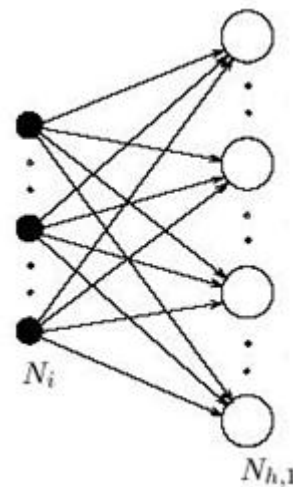


Kuram neironam skaitam summu
(whichNeuron)? Kurā slānī tas
atrodas (whichLayer jeb n)



Programmēšana

- procedure
NN_Calculate_Neuro
n_Layer(whichLayer:
integer);
 - Jāsaskaita visus
neironus slānī
 - Pielieto gan
summatora funkciju
NN_Summator, gan
aktivācijas funkciju
NN_Activation_Functi
on



Programmēšana

- Procedure
NN_Calculate_Errors();
 - Skaitam visas kļūdas neironiem neironu tīklā
 - Izejas slānim – viena formula (sk. 8. slaidu)
 - Pārējiem slāņiem - cita formula (sk. 10. slaidu)
- Procedure
NN_Adapt_Weights();
 - Skaitam jaunas svaru vērtības tīklā (sk. 11. slaidu)
- Procedure NN_Teach
 - Iteratīvi, atkārtot:
 - NN_Input_X;
 - for i:=1 to LayerCount do
NN_Calculate_Neuron_Layer(i)
 - NN_Calculate_Errors;
 - NN_Adapt_Weights
 - Atkārtoti kamēr tīkls nav pilnībā apmācīts, vai sasniegts noteikts iterāciju skaits.