729G78 Artificiell intelligens
Maskininlärning – Neurala nät

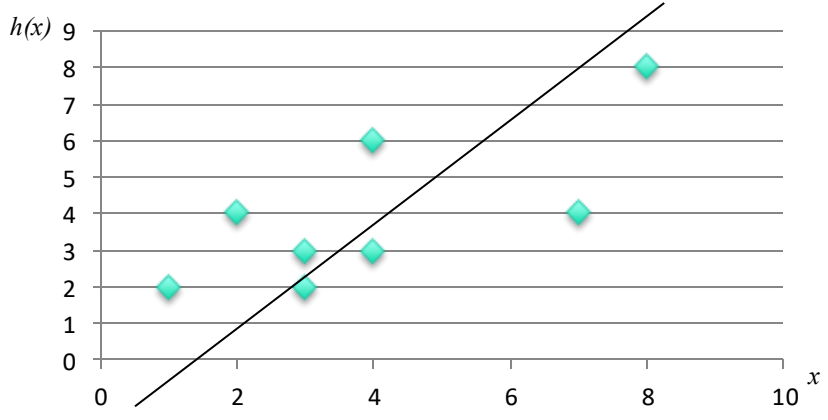
Robin Keskisärkkä

HCS/IDA

LINKÖPINGS UNIVERSITET

Linjär regression

- Hitta den linje som bäst beskriver en datamängd
  - Modellantagande: sambandet kan beskrivas med en rät linje
- $h(x) = w_1 x + w_0$ där w_1 anger lutningen och w_0 förskjutningen från origo
- Jämför avvikelsen mot linjen i varje punkt och minimera felet

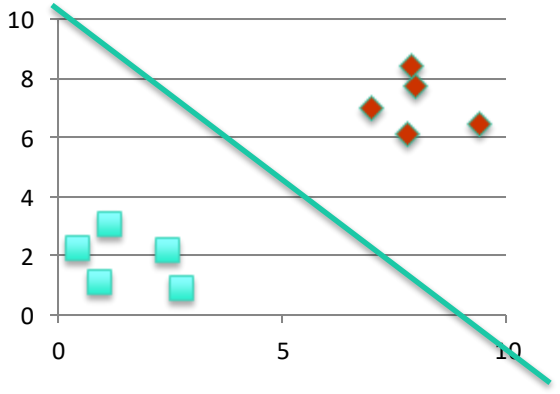


OBS: Räta linjens funktion: $y = kx + m$

Binär klassificering

- Avgöra vilken klass ett element tillhör:

  Dra en rät linje som på bästa sätt delar exemplen



- Kombinerar (multipel) linjär regression med en tröskelfunktion $f(x)$ så att vi får ex. ett ja/nej-svar istället för det beräknade värdet

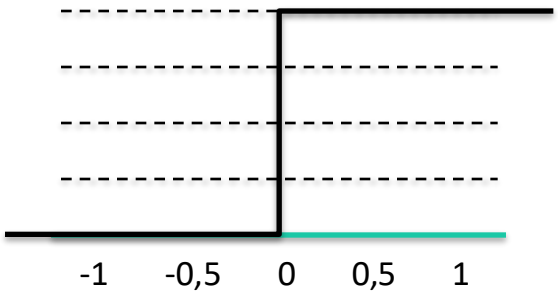
$$h(x) = f(w_1 x + w_0)$$

$$h(\boldsymbol{x}) = f(\boldsymbol{w}^T \boldsymbol{x})$$

Tröskelfunktion

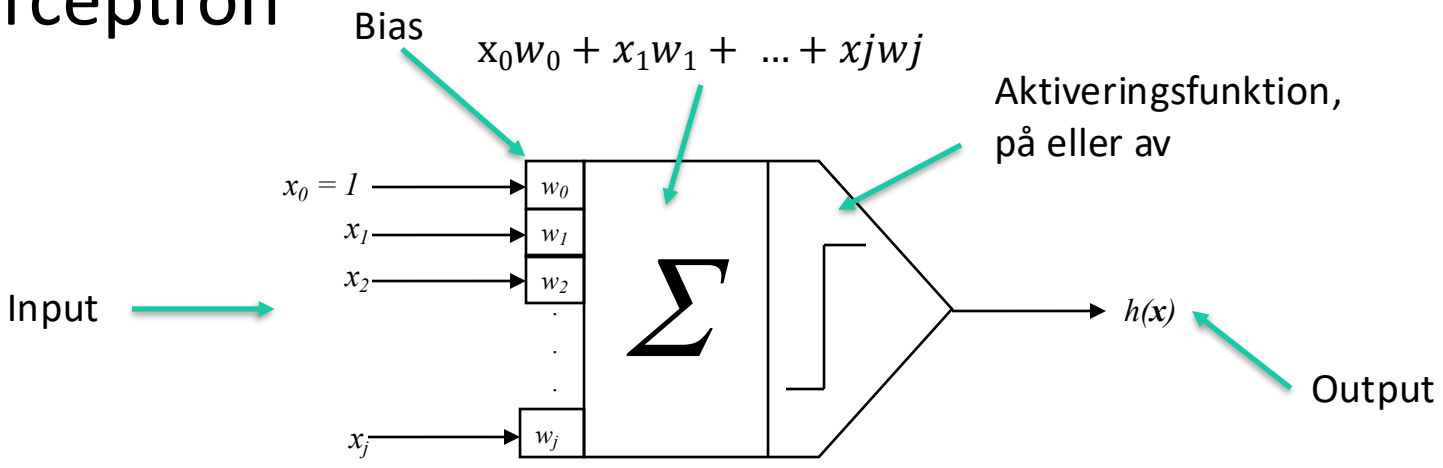
- T.ex. $f(x) = \begin{cases} 1 \; om \; x \geq 0 \\ 0 \; annars \end{cases}$

  dvs $f(x) = 1$ om x positiv



- Perceptroninlärning

Perceptron



$$h(\mathbf{x}) = f(\sum_{i=0}^{j} w_i x_i)$$

$$f(x) = \begin{cases} 1 \; om \; x \geq 0 \\ 0 \; annars \end{cases}$$

$$h(\mathbf{x}) = f(\boldsymbol{w}^T \boldsymbol{x})$$

Parametervektor (vikter) Särdragsvektor (input)

Repetition, parameteruppdatering

Uppdatera värdena på $w_0, w_1, \ldots, w_n$ så att L_2-felet minskar

Inlärningsparametern η (eta) och lutningen avgör hur snabbt vi uppdaterar w_1

$$w_0 = w_0 + \eta \frac{1}{N} \sum_{i=1}^{N} (t_i - h(x_i))$$

$$w_j = w_j + \eta \frac{1}{N} \sum_{i=1}^{N} x_{i,j} (t_i - h(x_i))$$

Matchande vikt och input

Partiella derivatan av felfunktionen. Det är lutningen på felfunktionen som avgör hur vi justerar vikterna.

En mer kompakt notation med vektorer:

$$\boldsymbol{w} = \boldsymbol{w} + \eta\big(\boldsymbol{t} - h(\boldsymbol{x})\big)\boldsymbol{x}$$

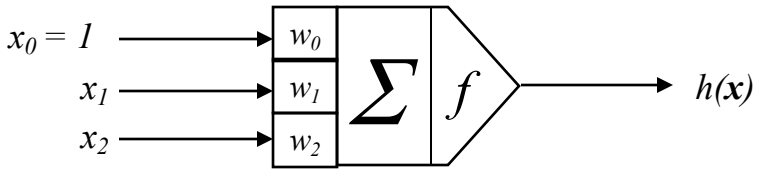
Perceptroninlärning

- Utdata $h(\mathbf{x}) = f(\boldsymbol{w}^T \boldsymbol{x})$
- För varje särdragsvektor, $\boldsymbol{x}$, och målvärde, $\boldsymbol{t}$, i träningsmängden med inlärningsparametern η
  1. Beräkna aktiveringen $h(\boldsymbol{x})$
  2. Uppdatera parametervektorn: $\boldsymbol{w} = \boldsymbol{w} + \eta\big(\boldsymbol{t} - h(\boldsymbol{x})\big)\boldsymbol{x}$
- Upprepa 1–2 tills felet är tillräckligt litet

Exempel

Träningsdata

x_1	x_2	t
1,0	1,0	1
9,4	6,4	-1
2,5	2,1	1
8,0	7,7	-1
0,5	2,2	1
7,9	8,4	-1
7,0	7,0	-1
2,8	0,8	1
1,2	3,0	1
7,8	6,1	-1



$$h(\mathbf{x}) = f(\boldsymbol{w}^T \boldsymbol{x})$$

$$f(x) = \begin{cases} 1 \; om \; x \geq 0 \\ -1 \; om \; x < 0 \end{cases}$$

Antag att vi startar med vikterna:

$w_0 =$ -0,6, $w_1 = 0{,}75$, $w_2 = 0{,}5$

$x_0 = 1$

Första input ger då:

$h(\boldsymbol{x}) = f(-0{,}6 + 1{,}0 \cdot 0{,}75 + 1{,}0 \cdot 0{,}5) = 1$

Felet blir $1 - 1 = 0$ så ingen uppdatering av vikterna sker

$$h(\boldsymbol{x}) = f(x_0 w_0 + x_1 w_1 + x_2 w_2) \qquad \eta = 0{,}2 \qquad \boldsymbol{w} = \boldsymbol{w} + \eta(\boldsymbol{t} - h(\boldsymbol{x}))\boldsymbol{x}$$

Exempel

T	x_1	x_2	w_0	w_1	w_2	t	h(x)	Fel
T_1	1,0	1,0	-0,6	0,75	0,5	1	1	0
T_2	9,4	6,4	-0,6	0,75	0,5	-1	1	-2
T_3								

$$w_0 = w_0 + \eta \frac{1}{N} \sum_{i=1}^{N} (t_i - h(x_i))$$

$$w_1 = w_1 + \eta \frac{1}{N} \sum_{i=1}^{N} x_{i1}(t_i - h(x_i))$$

$$w_2 = w_2 + \eta \frac{1}{N} \sum_{i=1}^{N} x_{i2}(t_i - h(x_i))$$

$$h(\boldsymbol{x}) = f(1 * (-0{,}6) + 9{,}4 * 0{,}75 + 6{,}4 * 0{,}5) = f(9{,}65) = 1$$

OBS: Vanligtvis uppdaterar man inte efter varje datapunkt utan vill utnyttja ett genomsnittsfel.

$$h(\boldsymbol{x}) = f(x_0 w_0 + x_1 w_1 + x_2 w_2) \qquad \eta = 0{,}2 \qquad \boldsymbol{w} = \boldsymbol{w} + \eta(\boldsymbol{t} - h(\boldsymbol{x}))\boldsymbol{x}$$

Exempel

T	x_1	x_2	w_0	w_1	w_2	t	h(x)	Fel
T_1	1,0	1,0	-0,6	0,75	0,5	1	1	0
T_2	9,4	6,4	-0,6	0,75	0,5	-1	1	-2
T_3			-1	-3,01	-2,06			

$$w_0 = w_0 + \eta * (-2) = -0{,}6 + 0{,}2 * (-2) = -1$$

$$w_1 = w_1 + \eta * x_1(-2) = 0{,}75 + 0{,}2 * 9{,}4(-2) = -3{,}01$$

$$w_2 = w_2 + \eta * x_2(-2) = 0{,}5 + 0{,}2 * 0{,}64(-2) = -2{,}06$$

$$w_0 = w_0 + \eta \frac{1}{N} \sum_{i=1}^{N} (t_i - h(x_i))$$

$$w_1 = w_1 + \eta \frac{1}{N} \sum_{i=1}^{N} x_{i1}(t_i - h(x_i))$$

$$w_2 = w_2 + \eta \frac{1}{N} \sum_{i=1}^{N} x_{i2}(t_i - h(x_i))$$

OBS: Vanligtvis uppdaterar man inte efter varje datapunkt utan vill utnyttja ett genomsnittsfel.

$$h(\boldsymbol{x}) = f(x_0 w_0 + x_1 w_1 + x_2 w_2) \qquad \eta = 0{,}2 \qquad \boldsymbol{w} = \boldsymbol{w} + \eta(\boldsymbol{t} - h(\boldsymbol{x}))\boldsymbol{x}$$

Exempel

T	x_1	x_2	w_0	w_1	w_2	t	h(x)	Fel
T_1	1,0	1,0	-0,6	0,75	0,5	1	1	0
T_2	9,4	6,4	-0,6	0,75	0,5	-1	1	-2
T_3	2,5	2,1	-1	-3,01	-2,06	1	-1	2
T_4	8	7,7	-0,6	-2,01	-1,22	-1	-1	0
T_5	0,5	2,2	-0,6	-2,01	-1,22	1	-1	2
T_6	7,9	8,4	-0,2	-1,81	-0,34	-1	-1	0
T_7	7	7	-0,2	-1,81	-0,34	-1	-1	0
T_8	2,8	0,8	-0,2	-1,81	-0,34	1	-1	2
T_9	1,2	3	0,2	-0,69	-0,02	1	-1	2
T_{10}	7,8	6,1	0,6	-0,21	1,18	-1	1	-2

$$w_0 = w_0 + \eta \frac{1}{N} \sum_{i=1}^{N} (t_i - h(x_i))$$

$$w_1 = w_1 + \eta \frac{1}{N} \sum_{i=1}^{N} x_i (t_i - h(x_i))$$

$$w_2 = w_2 + \eta \frac{1}{N} \sum_{i=1}^{N} x_i (t_i - h(x_i))$$

OBS: Vanligtvis uppdaterar man inte efter varje datapunkt utan vill utnyttja ett genomsnittsfel.

Resultat, exempel

Efter 500 iterationer med samma träningsmängd konvergerar vikterna

$w_0 = 10{,}9,\ w_1 = -1{,}3,\ w_2 = -1{,}1$

Vi får en beslutslinje om vi sätter $h(\mathbf{x})=0$, dvs

$0 = 10{,}9 - 1{,}3 \cdot x_1 - 1{,}1 \cdot x_2$

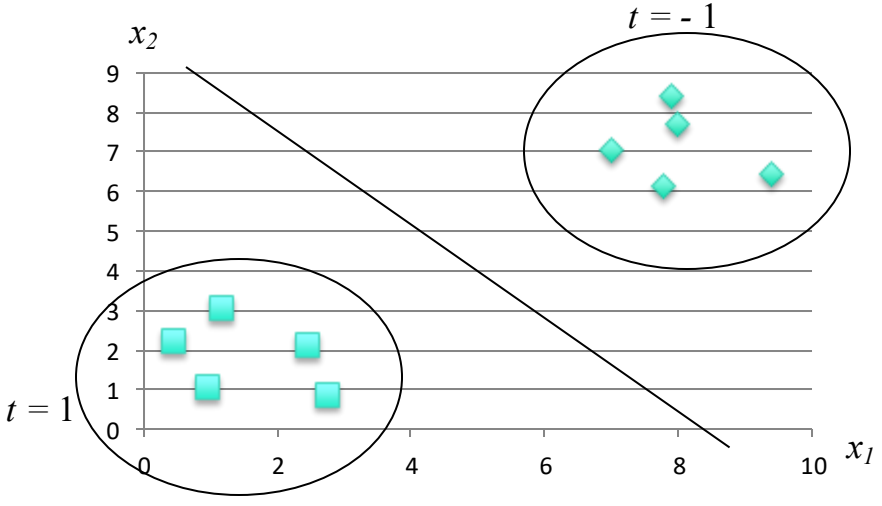
dvs

$x_1 = -0{,}85x_2 + 8{,}38$

Ex.

$x_2 = 9\ ger\ x_1 = 0{,}73$

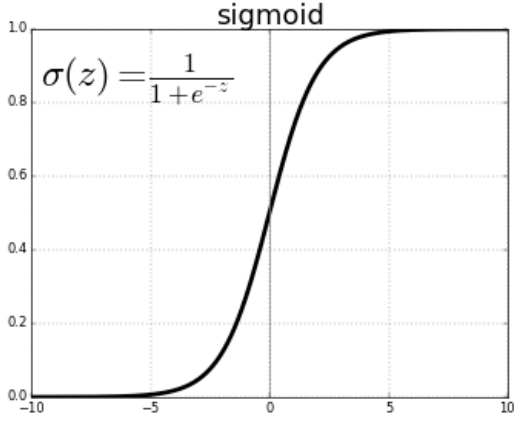
$x_2 = 0\ ger\ x_1 = 8{,}4$

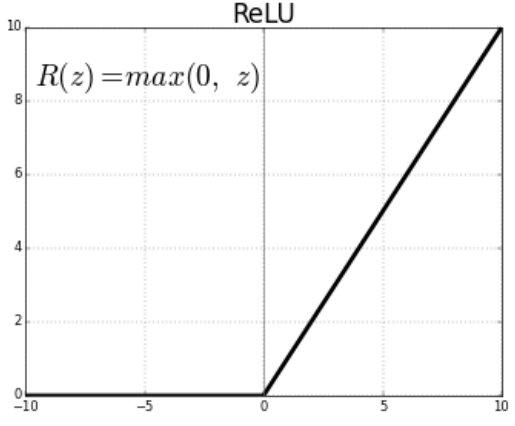


Andra aktiveringsfunktioner

Den vi använt ger oberäkneligt inlärningsbeteende och kan inte deriveras (finns ingen lutning!)

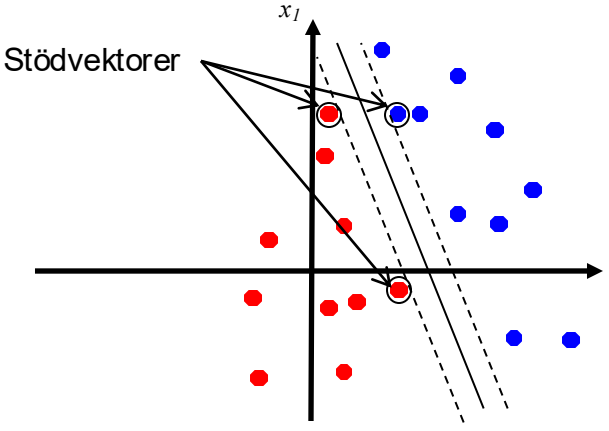
Alternativ?





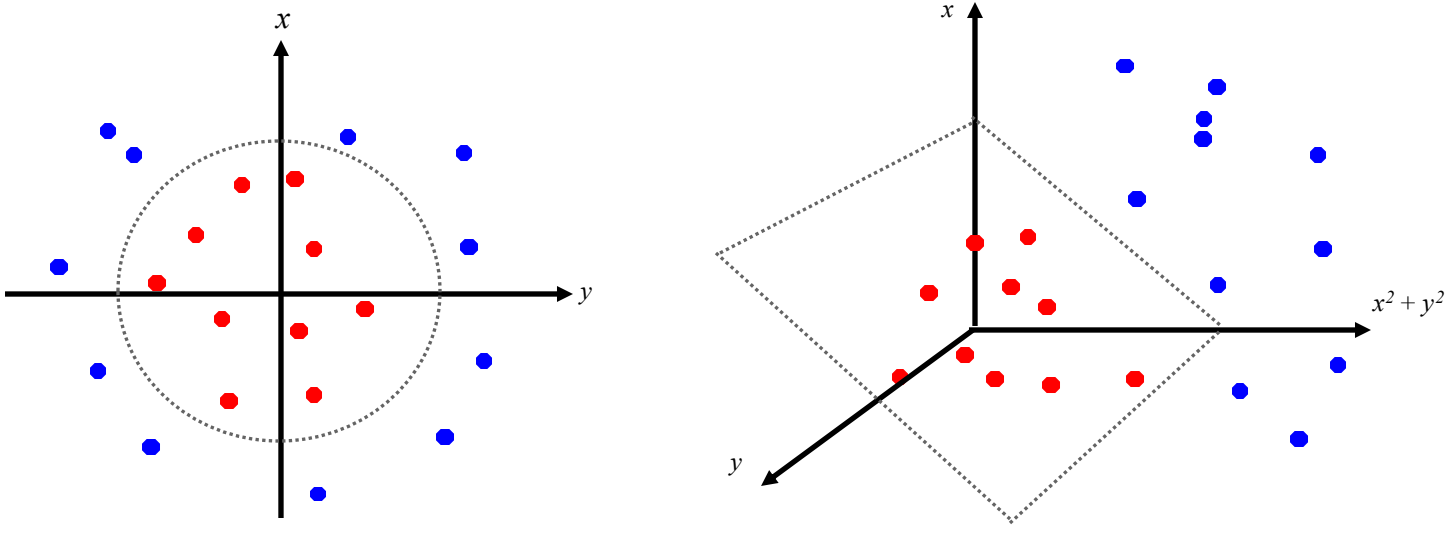
Support Vector Machines (SVM)

- Vanlig klassificeringsmetod med många färdiga bibliotek att använda
- Väljer att separera för att få bästa möjliga generalisering
  - Linjär regression väljer utifrån alla punkter medan SVM identifierar några punkter som viktigare än andra, stödvektorer
  - Väljer linjen som maximerar avståndet till alla dessa



SVM

- Ibland går det inte att dela linjärt men man kan ibland transformera till högre dimension där det är möjligt
- Ex. mappa om data med formen $[x, y]$ till $[x, y, x^2 + y^2]$



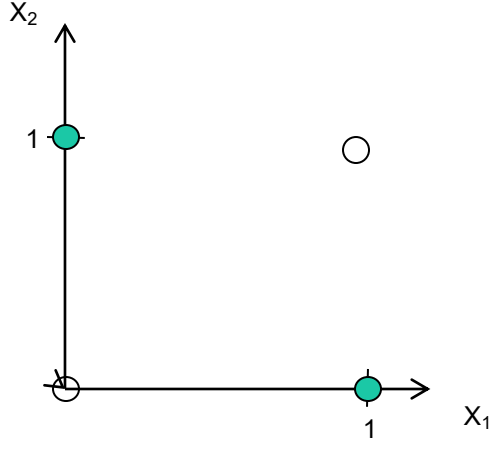
Kerneltricket

- Ex. Beräkna produkten av vektorer en högre dimension:
  - (a, b) till $(a, b, a^2 + b^2)$
    (c, d) till $(c, d, c^2 + d^2)$
    Kostsamt att beräkna för alla datapunkter!
  - Produkten utskrivet: $(a, b, a^2 + b^2) \cdot (c, d, c^2 + d^2) = ac + bd + (a^2 + b^2)(c^2 + d^2)$
- Med en kernelfunktion, $K(\boldsymbol{x}, \boldsymbol{y})$, kan vi göra beräkningen effektivt:
  - $K((a, b), (c, d)) = ac + bd + (a^2 + b^2)(c^2 + d^2)$
  - Kernelfunktionen kan räknas ut direkt, utan att transformera data först

Linjärt separerbara problem

- Booleska funktionerna OCH och ELLER linjärt separerbara
- Exlusivt ELLER (XOR)

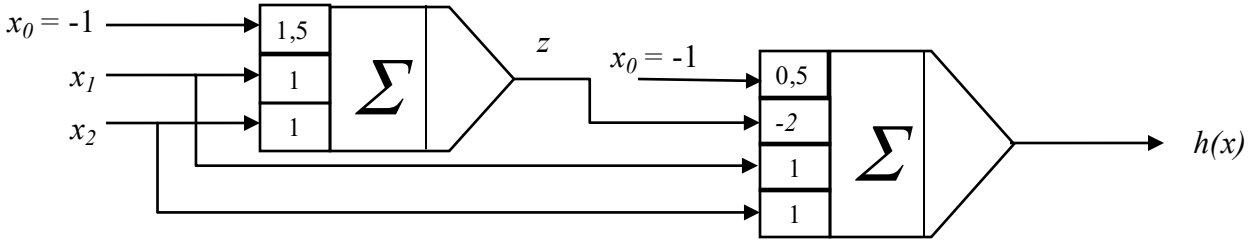
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



- Finns ingen linje som separerar
- Problem för perceptroner

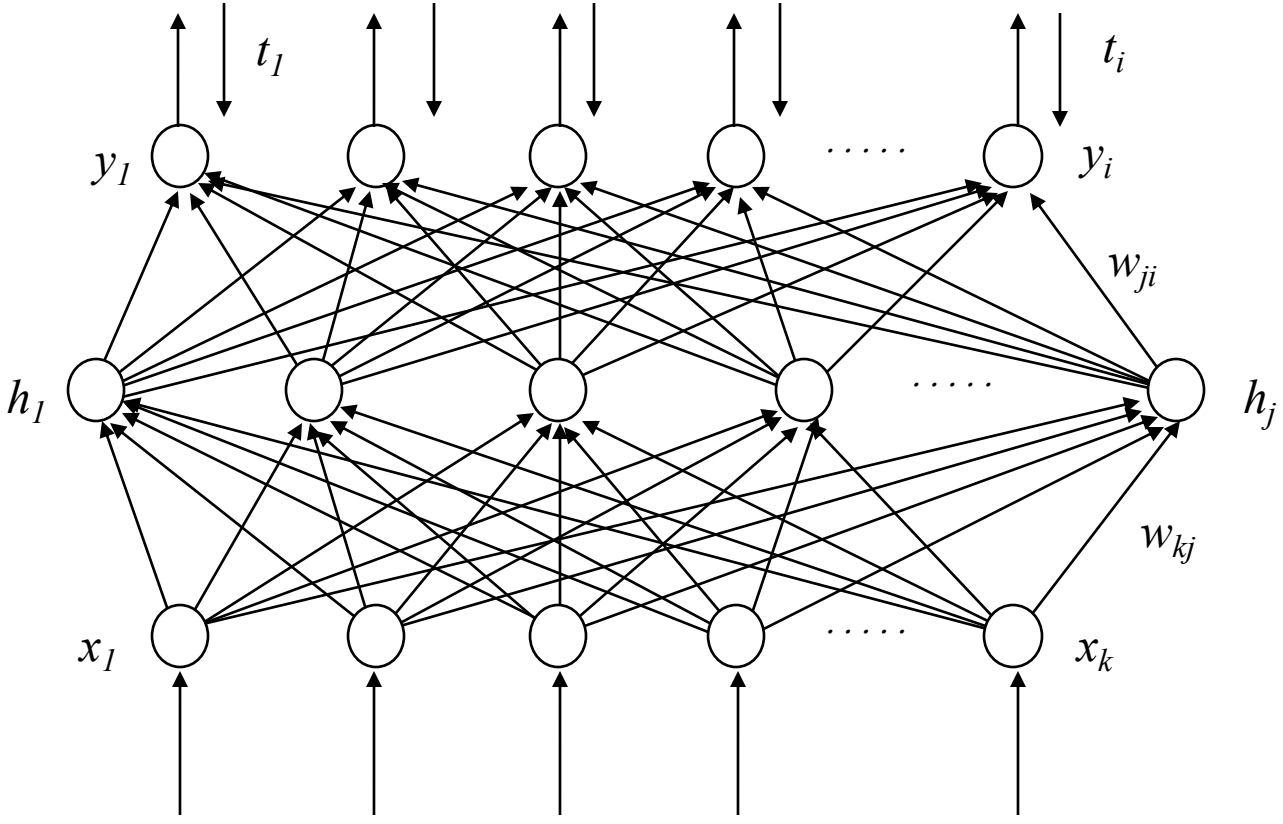
Neurala nät

XOR



x_1	x_2	t	$z = -1{,}5 + x_1 + x_2$	$h(x) = -0{,}5 - 2z + x_1 + x_2$
0	0	0	-1,5 + 0 + 0 = -1,5 → 0	-0,5 - 0 + 0 + 0 = -0,5 → 0
0	1	1	-1,5 + 0 + 1 = -0,5 → 0	-0,5 + 0 + 0 + 1 = 0,5 → 1
1	0	1	-1,5 + 1 + 0 = -0,5 → 0	-0,5 + 0 + 1 + 0 = 0,5 → 1
1	1	0	-1,5 + 1 + 1 = 0,5 → 1	-0,5 - 2 + 1 + 1 = -0,5 → 0

Nätverk



Neurala nät

- Inlärning
  - Problem att räkna ut felet för mellanlagret
  - Finns ingen metod som garanterar optimalitet
    - Gradient backpropagation vanlig metod
  - Undvika överinlärning
- Nätverkstopologin
  - Hur många dolda lager och hur många noder i varje lager
- Aktiveringsfunktion
  - Sigmoid, tanh, ReLU, ….
  - Olika funktioner i olika lager

Gradient backpropagation

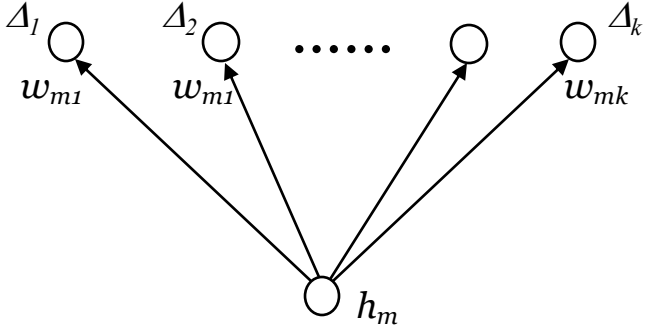
- Viktuppdatering för utdatalagret, $\boldsymbol{y}$
  - Samma som för perceptronen med utdata från dolda lagret, $\boldsymbol{h}$, som indata

$$\boldsymbol{w} = \boldsymbol{w} + \eta(\boldsymbol{t} - \boldsymbol{y})(1 - \boldsymbol{y})\boldsymbol{y}\boldsymbol{h}$$

- Felet från noder i dolda lagret antas bero av det fel som noden givit upphov till i lagret efter. Kalla det $\Delta = (\boldsymbol{t} - \boldsymbol{y})(1 - \boldsymbol{y})\boldsymbol{y}$

$$E \propto \boldsymbol{w}_m^T \Delta = \sum_{i=1}^{k} w_{mi}\, \Delta_i$$

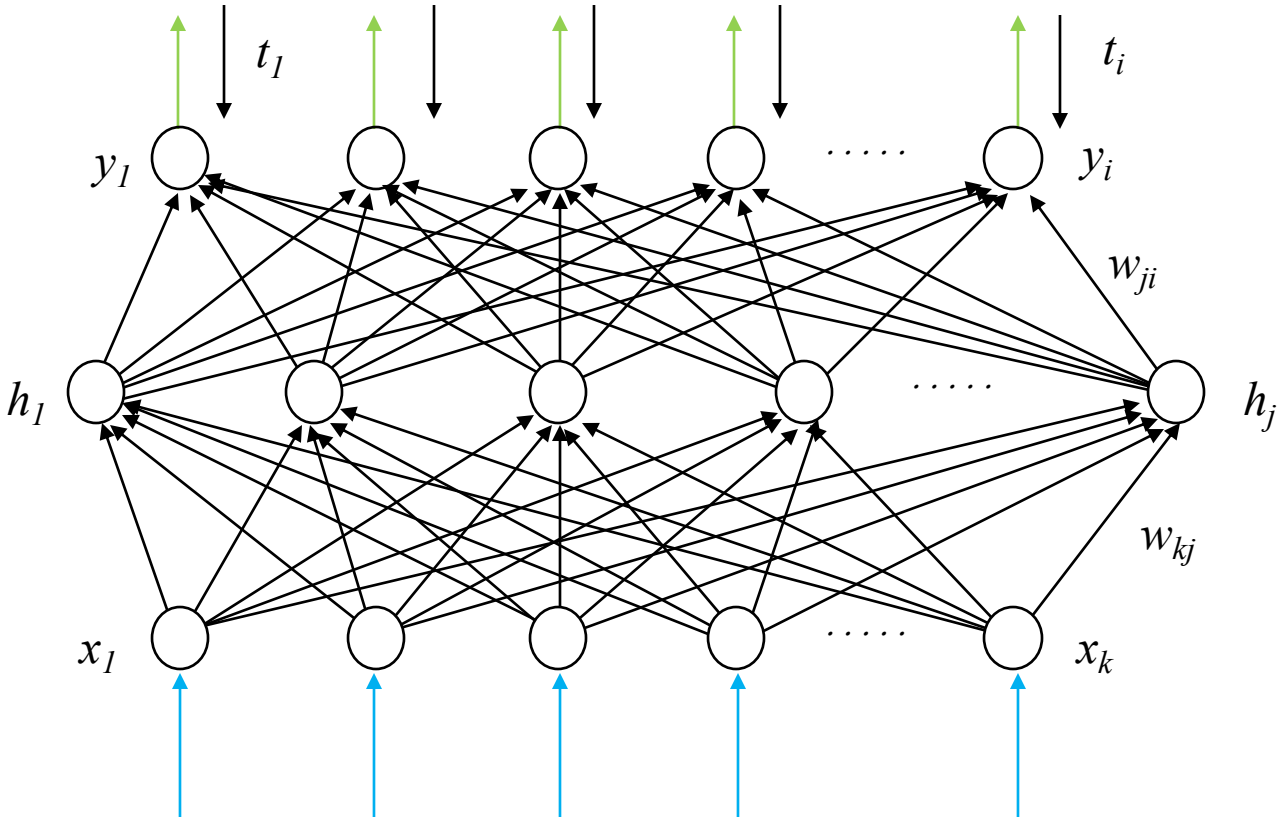
Proportionellt mot



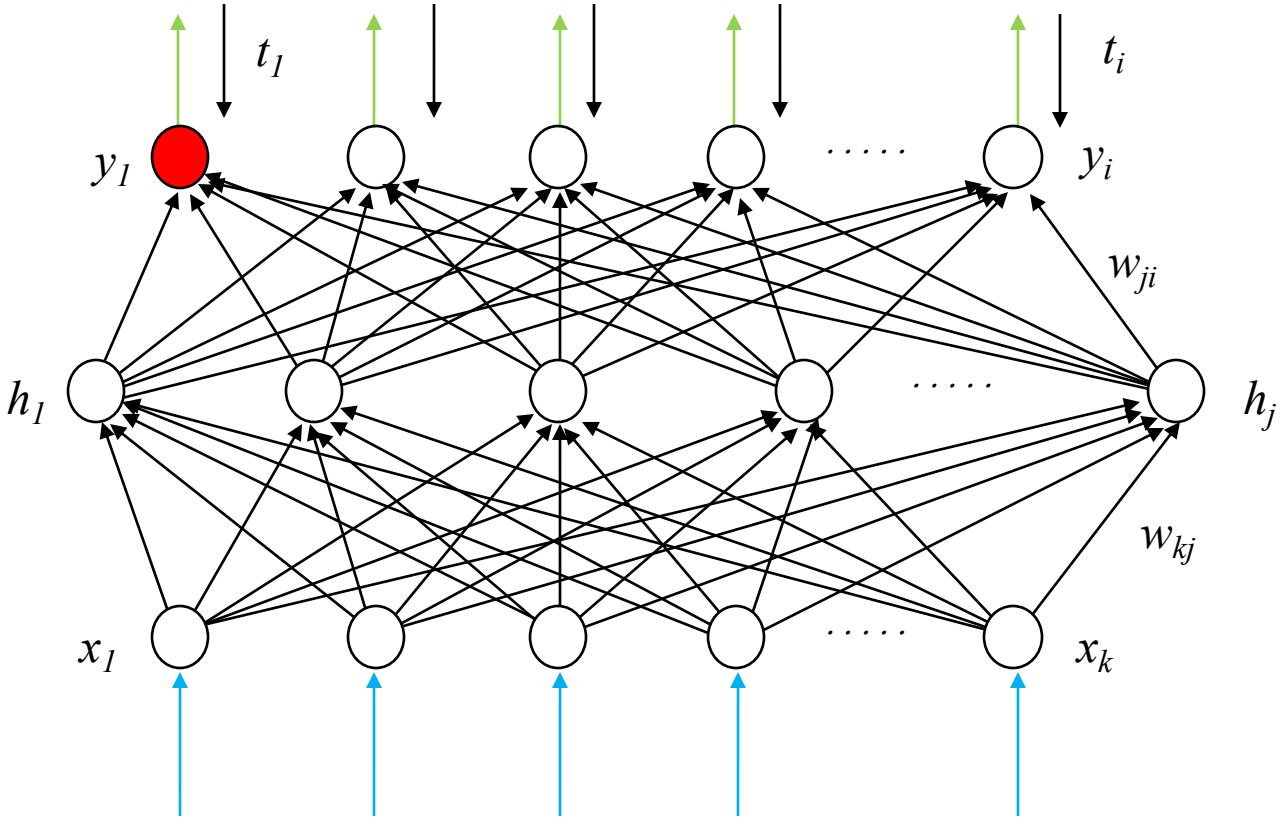
Gradient backpropagation

1. Initialisera vikterna slumpmässigt
2. Upprepa tills resultatet tillfredställande:
   - Applicera en indatavektor $\boldsymbol{x}$ (förväntat resultat $\boldsymbol{t}$)
   - Aktivera nätverket, ger utdata $\boldsymbol{y}$
   - Räkna ut felet för utdatalagret i förhållande till guldstandard: $\Delta_i = (t_i - yi)(1 - yi)y_i$
   - Uppdatera vikterna i utdatalagret: $w_{ji} = wji + \eta\Delta_i h_j$
   - Räkna ut felet för dolda lagret: $\Delta_j = \sum_i (w_{ij}\Delta_i)(1 - hj)h_j$
   - Uppdatera vikterna i dolda lagret: $w_{kj} = wkj + \eta\Delta_j x_k$

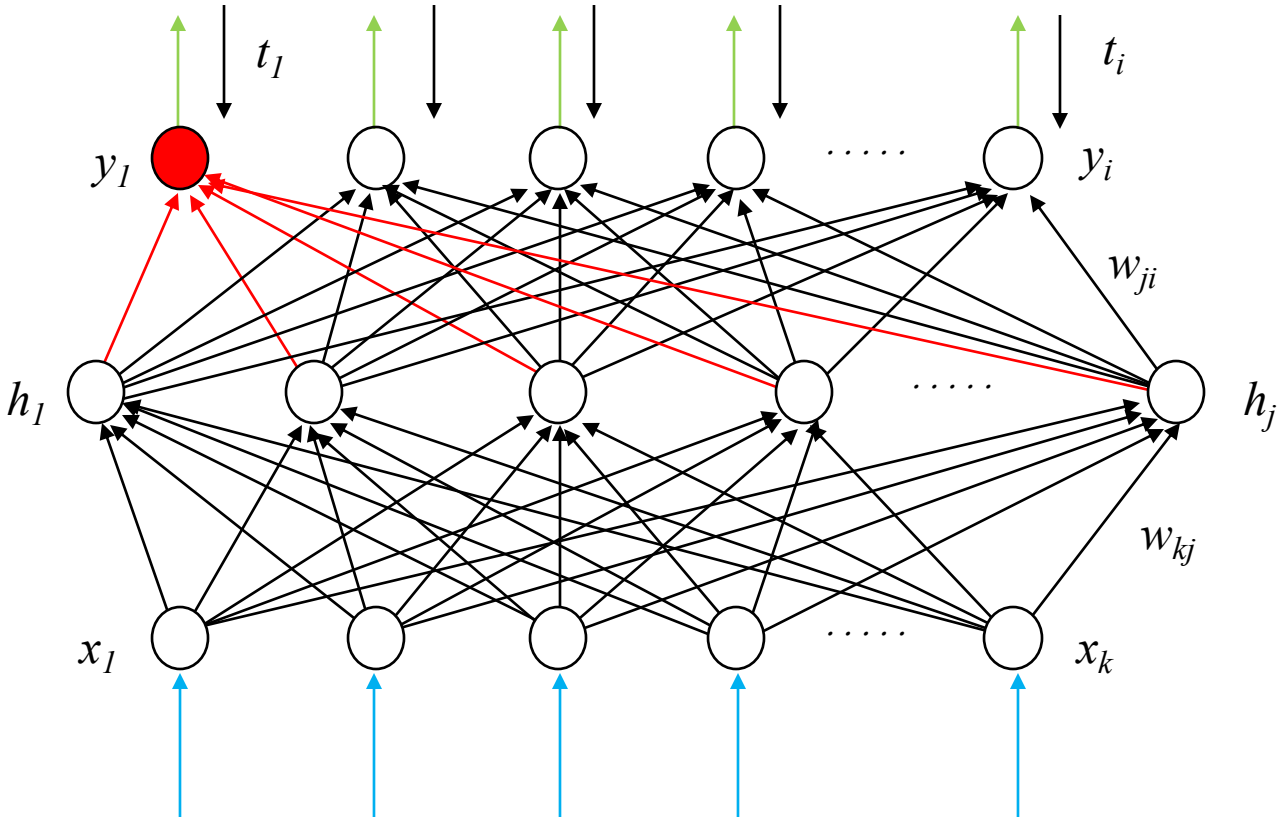
Två sista stegen upprepas om flera lager


t_1
t_i
y_1
y_i
w_{ji}
h_1
h_j
w_{kj}
x_1
x_k

Beräkna felet för y_i: $\Delta_i = (t_i - yi)(1 - yi)y_i$



Beräkna felet för y_i: $\Delta_i = (t_i - yi)(1 - yi)y_i$
Uppdatera vikterna mot y_i: $w_{ji} = wji + \eta\Delta_i h_j$

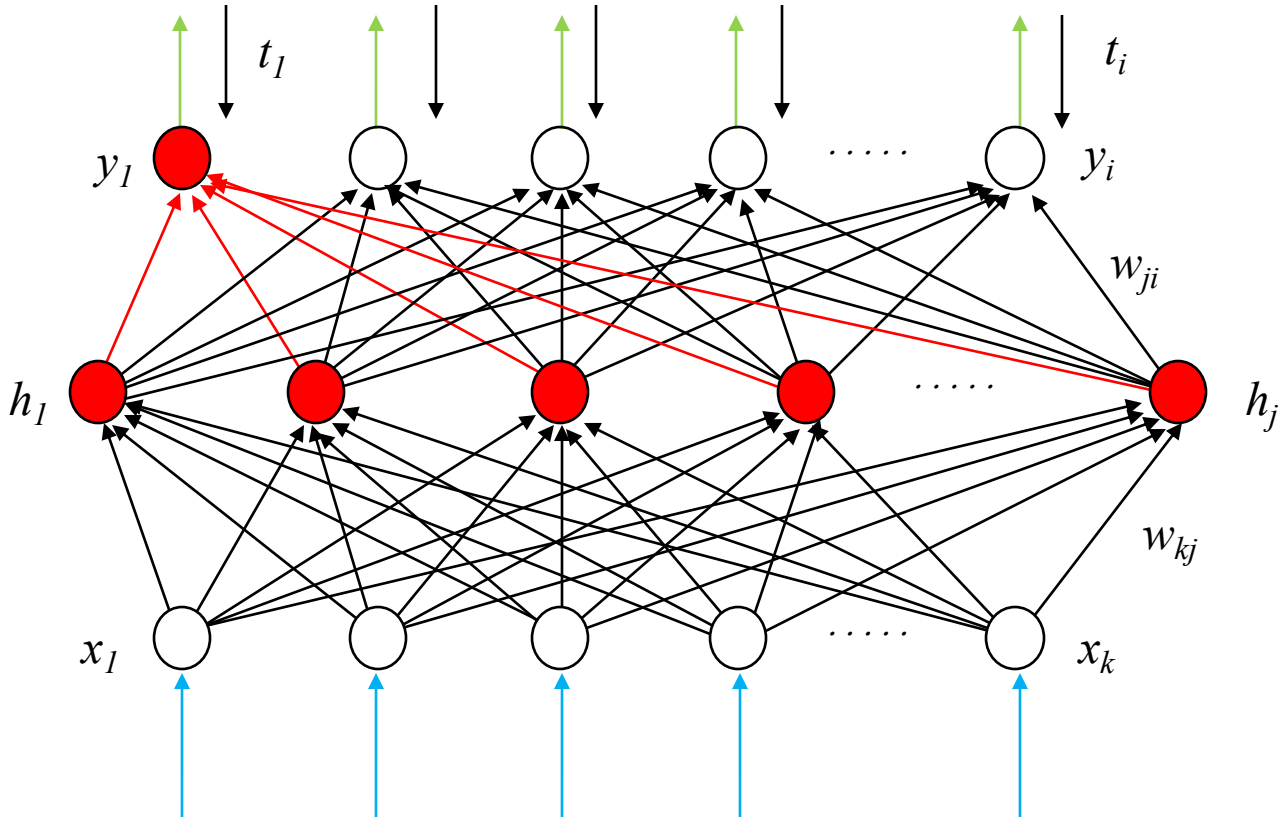


Beräkna felet för y_i: $\Delta_i = (t_i - yi)(1 - yi)y_i$

Uppdatera vikterna mot y_i: $w_{ji} = wji + \eta\Delta_i h_j$

Räkna ut felet för dolda lagret: $\Delta_j = \sum_i (w_{ij}\Delta_i)(1 - hj)h_j$

Felet är propertionerligt mot felet i utdatalagret

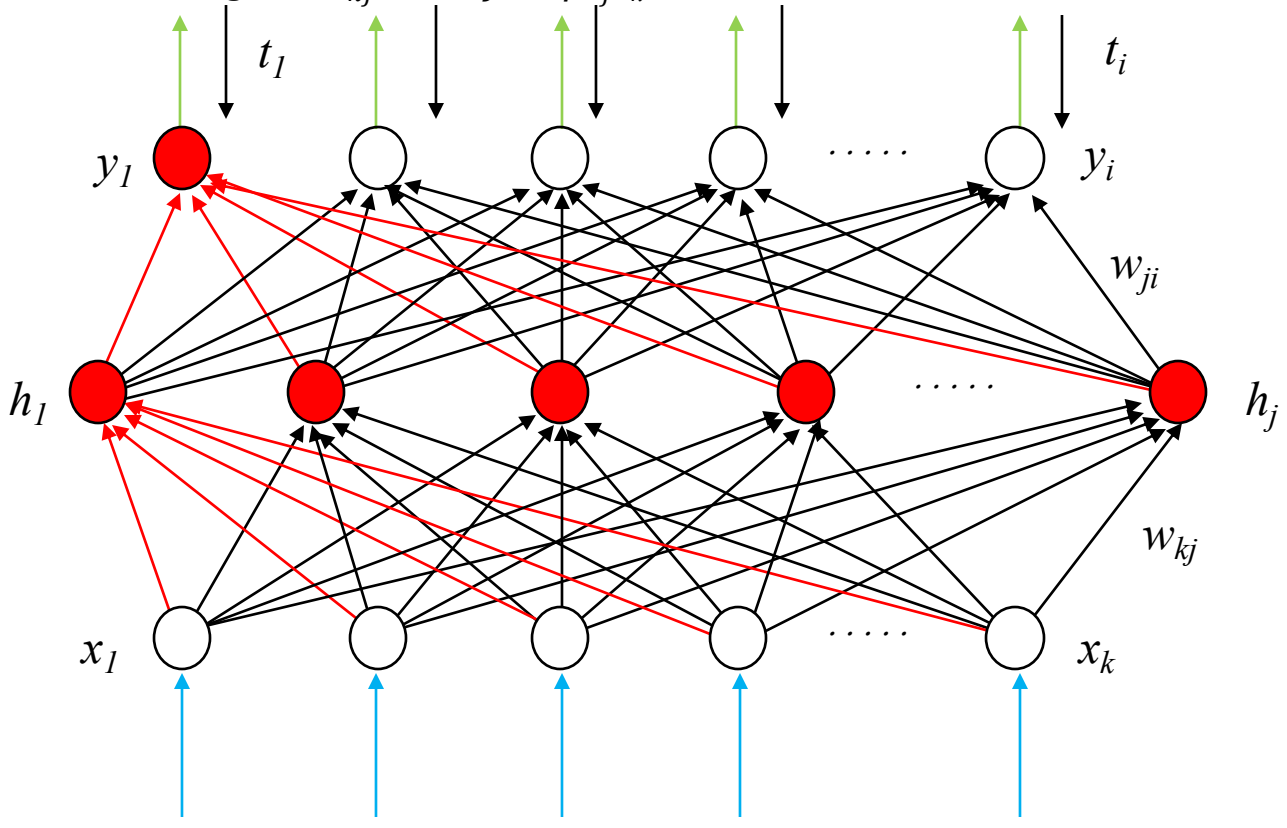


Beräkna felet för y_i: $\Delta_i = (t_i - yi)(1 - yi)y_i$

Uppdatera vikterna mot y_i: $w_{ji} = wji + \eta\Delta_i h_j$

Räkna ut felet för dolda lagret: $\Delta_j = \sum_i (w_{ij}\Delta_i)(1 - hj)h_j$

Uppdatera vikterna i dolda lagret: $w_{kj} = wkj + \eta\Delta_j x_k$



Indatalagret

- Bestämma hur indata kodas
  - 0, 1, -1, ...
  - Numeriska värden, skalade, logaritmerade
  - Binära vektorer (*one-hot encoding*, dvs. en aktiv och alla andra inaktiva)
    - Kategoridata
    - En indatabit för varje alternativ
    - Ex. färger (röd, grön, orange, blå)
      - Röd: [1, 0, 0, 0]

Utdatalagret

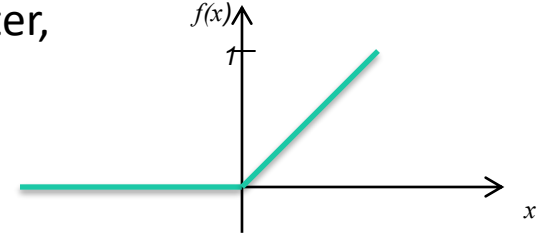
- Snarlikt indata, ofta *one-hot encoding*
  - Restaurangexemplet en utnod {Ja/Nej}
  - Väder med fyra värden {Regn, Sol, Snö, Moln} har fyra utnoder (bit)
- Inte binära utdata utan ”sannolikheter”
  - En utnod: ofta sigmoid-funktionen
  - Flera utdatanoder, d, ofta softmax (omvandla till sannolikheter)

$$softmax(\boldsymbol{in})_k = \frac{e^{in_k}}{\sum_{k'=1}^{d} e^{in_{k'}}}$$

$$ger\ utdata\ för\ nod\ k\ med\ indata\ \boldsymbol{in} = (in_1, in_2, \ldots, in_d)$$

Fler dolda lager (hidden layers)

- Varje lager kan ha en representation av indata
  - Hörn, kanter, ögon, ansikten etc. (men i verkligheten vet vi inte!)
- Oklart hur man väljer antal
  - För komplexa problem ofta bättre med djupa, smala nätverk än grunda breda
- Djupa nät
  - Gradienten blir väldigt liten, ger liten, eller ingen viktuppdatering
  - Andra aktiveringsfunktioner med starkare gradienter, vanligt med ReLU



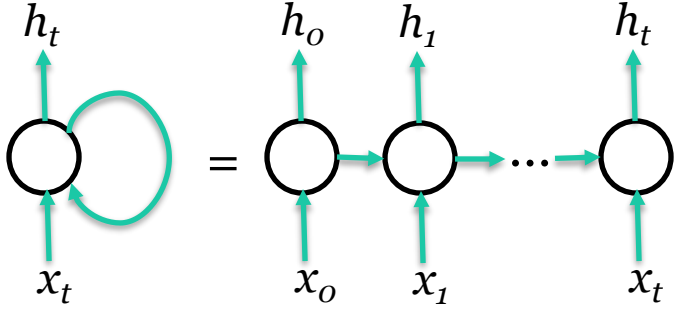
Träning av nätverk

Nätverket påverkas av:

- Antal lager
- Antal noder i dolda lagren
- Aktiveringsfunktioner
- Antal epoker
- Inlärningsparametern
- Felkvoten

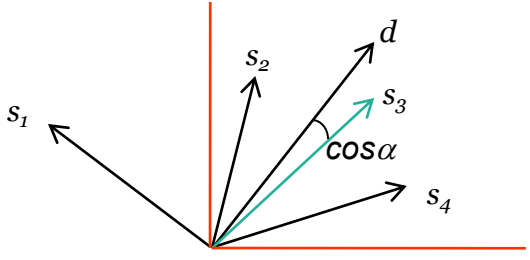
Förbättringar

- Regularisering
- Skapa nya datamängder genom att transformera befintlig
  - Ex. för bilddata rotera, skala, ändra kontrast etc.
- Dropouts
  - Slå ut slumpvisa delar av nätet under träningen. Kan förhindra att nätet blir överkänsligt för egenheter hos datamängden.
- Återkoppla näten
  - RNN, LSTM



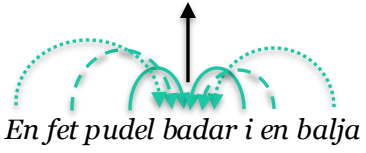
Vektorrumsmodeller

- Tekniker som reducerar den lingvistiska variationen och fångar semantiskt relaterade begrepp
- Ord representeras som vektorer (*word embeddings*)
- Meningar fås genom att addera ordvektorer, dokument genom att addera meningsvektorer
- Likhet mellan meningar och dokument mäts som närhet i vektorrummet



Distribuerad semantik

- Ett ords betydelse beror på hur det används i kontext
- Varje ord representeras som en vektor
  - Några hundra dimensioner
  - Träningskorpus, några miljoner ord
- Exempel, random indexing



Han bet hunden.

1. Tilldela en gles slumpvektor till varje ord

Han	[0 −1 0 1 0 −1 0 0 0 1]
bet	[0 0 0 0 0 1 1 −1 −1 0]
hunden	[−1 0 1 0 0 1 0 −1 0 0]

2. Varje gång ett ord förekommer i texten, addera kontextvektor i ett fönster runt ordet
   bet: [0 0 0 0 0 1 1 -1 -1 0] + [0 -1 0 1 0 -1 0 0 0 1] + [-1 0 1 0 0 1 0 -1 0 0] = [-1 -1 1 1 0 1 1 -2 -1 1]

- Träningskorpusar på miljontals ord, och mer

Mot GPT - BERT

- Kontextberoende vektorer
  - Mer än en ordvektor för varje ord
  - Delar av ord (subwords)
- Modell
  - Transformer
    - 340 miljoner parametrar (vikter)
  - Tränad på
    - Bokkorpus (800 miljoner ord)
    - Wikipedia (2,500 miljoner ord)
  - Meningar som indata
    - Ord maskerade under träningen
    - Gissa nästa mening

- Kan anpassas (fine-tuned)
  - T.ex. abstraktiv sammanfattning
- Kan anlyseras
  - Probing
- Google, 2018

KB tillgängliggör kraftfulla modeller för språkförståelse

4 februari 2020

Forskning, KB-labb

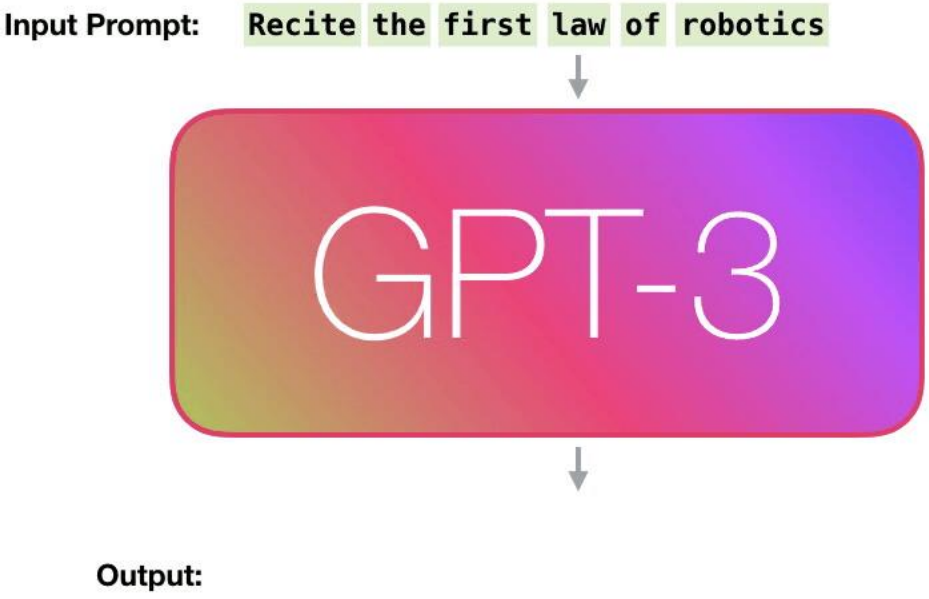
I dag publicerar KB tre svenska språkmodeller baserade på Googles "BERT" (Bidirectional Encoder Representations from Transformers). De första testerna visar att KB:s modeller överträffar Googles flerspråkiga modell.

GPT-3

- Distribuerad semantik
- Kontextberoende ordvektorer
- Transformer
  - 175 miljarder parametrar
- Ingen fine-tuning
- Gissar vilket som är det mest sannolika nästa ordet
  - Prompting
- OpenAI, 2020
  - DALL-E

Träningsdata

- 45 TB
  - Common Crawl (410 miljarder ord)
    - 8 års samlade webbdata
  - WebText2, text från Reddit webbsidor(19 miljarder)
  - Internet bokkorpus (67 miljarder)
  - Wikipedia (3 miljarder)
- Effektåtgång
  - Runt 1,000GWh



Alammar, J (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/how-gpt3-works-visualizations-animations/

Sammanfattning

- Agent som lär sig
  - Typer av återkoppling: övervakad, oövervakad, förstärkt inlärning
- Beslutsträdsinlärning
  - Val av attribut, informationsmått, entropi
- Linjär regression
  - Felfunktioner
  - Gradientsökning
- Grundläggande begrepp
  - Träningsdata, valideringsdata, testdata, träningsfel, generaliseringsfel
- Vektorer, matriser, skalärprodukt
- Multipel linjär regression
- Klassificering
  - Perceptroner
  - Inlärning av perceptroner
  - Gradientsökning
  - SVM
- Neurala nät
  - Nätverk av perceptroner
  - Gradient backpropagation