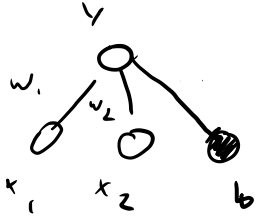
Klassisk maskininlärning

- regression, statistiska metoder

Neurala nätverk

50-talet: baserat på en idag något föråldrad syn på hjärnan.
Ändå så abstrakt att "det funkar ändå".

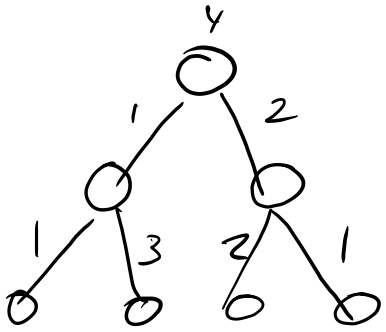
Perceptron



$y = w_1 x_1 + w_2 x_2 + b$

$y = w1 \cdot x + b$

"2D linje" – plan som spänns upp av w1

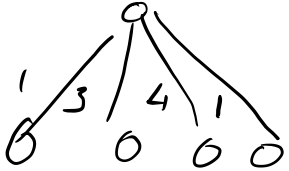


$$y = 1(1x_1 + 3x_2) + 2(2x_3 + 1x_4)$$

$$y = 1x_1 + 3x_2 + 4x_3 + 1x_4$$

$$L(a+b) = L(a) + L(b)$$

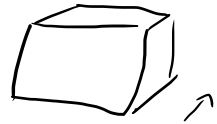
Linjärd, alltså inte komponerbara.
Vi behöl nåjon icke-linjäritet!



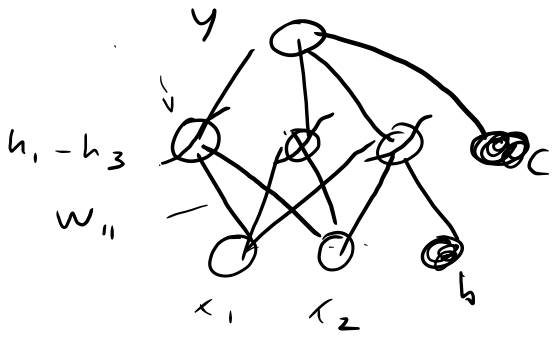
Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$

ReLU: $V(x)$ { x om $x>0$; 0 annars

MLP Multilayered perceptron



50-80 talet, teorin för hur man tränar djupa nätverke fanns inte. Dessutom beräkningsmässigt dyrt.

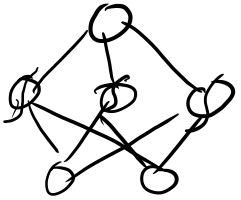


$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

$$\sigma \Rightarrow \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

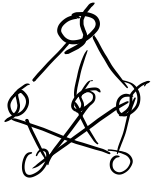
$$\begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} + c = y$$

$$f(x) = V\sigma(Wx + b) + c$$



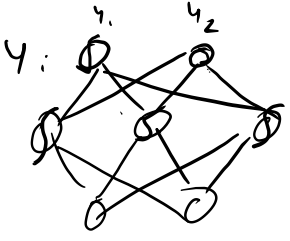
← regression

← feature expansion



sigmoid på $y \rightarrow [0, 1]$

tolka som sannolikhet → binär klassificering

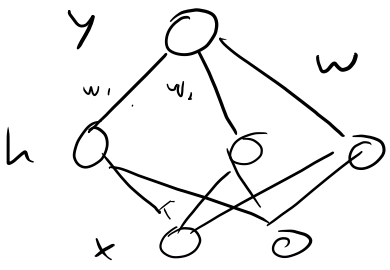


(S) ϕ

↑ softmax

$$y_i = \frac{e^{l_i}}{\sum_j e^{l_j}} \qquad \sum_{i=1}^{r} y_i^2 = 1$$

→ multinomiell



börja slumpmässigt vikter

$y - \hat{y}$ - justera varje vikt "lite grann" mot en bättre lösning.

Iterativt för all träningsdata:

$$W \leftarrow W - \eta \nabla C(W)$$

moduler:

$$f(x) = \frac{1}{1 - e^{-x}}$$

kedjeregeln över funktionella moduler

$$a(x) = -x$$

$$b(a(x)) = e^{-x}$$

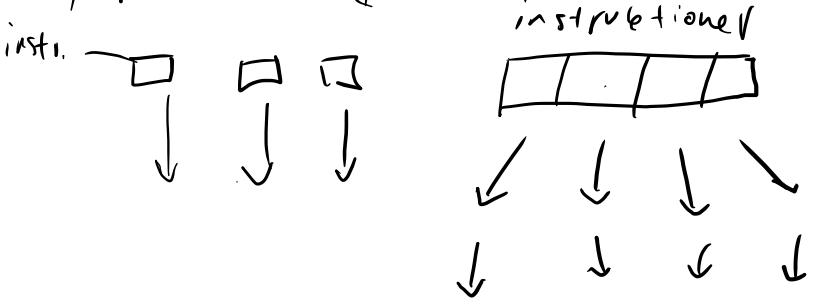
$$C(b(a(x))) = 1 - e^{-x}$$

$$d(C(b(a(x)))) = \frac{1}{1 - e^{-x}}$$

→ forward: symbolisk derivering

← backward: numeriskt

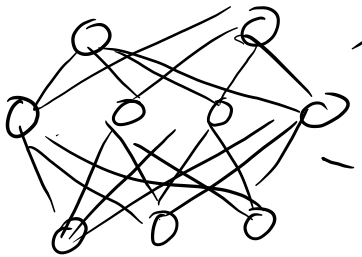
Parallellism (på två sätt)



kompilatorer för vektorinst. väldigt svåra att skriva.

Men NN fungerar som vektorprocessorer.

NN

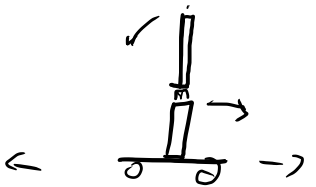


– regression

– feature expansion

lär sig features

βX



two-pole balancing

SVM

$K(x_i, x_j)$

expanderar featurerymden väldigt mycket, kollapsar dimensioner för att undvika overfit

βX

NLP

Klassiska tekniker (innan 2017) räcker inte.

GPT: generative pre-trained transformers.

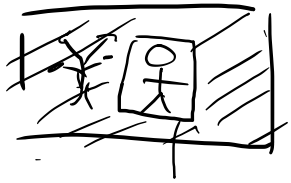
Transform: Encoder → dolda lager → Decoder

GPT 1

Embedding:

omvandlar ord till tal. Vi kallar dem då "tokens"

Attention



Du kanske inte är så dum ändå.

- skalär produkt mellan inbäddade ord.

problem med syftningar; höger eller vänster association.

imf ($\times$, $\div$) $1 + 2 \times 3 \div 4$

self-attention (masked self attention)

titta endast åt ett håll.

Du kanske inte är så dum ändå. $\rightarrow$ $[0,1, 0,3, \ldots]$

optimeringsvillkor för oövervakad inlärning
(pre-training)

$$\max \sum_{i}^{n} \log P(u_i \mid u_{i-k}, \ldots, u_{i-1} ; \theta)$$

Vi lär alltså systemet att lösa först.

Sedan tränar vi övervakat:

<sentence> | <answer>

På lär vi oss domänkunskap genom att förutsäga svaren.
(tex fakta om Mozart)

$A(e) = (Q, K, V)$

Multi-head, multi-cell self attention