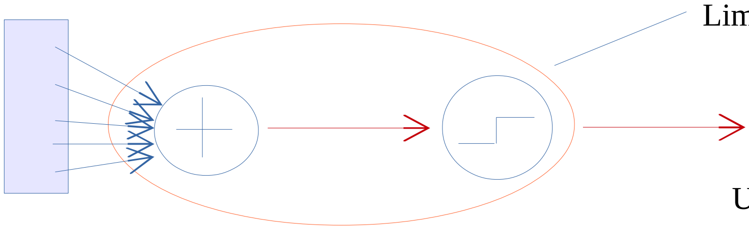
Mon 13 October

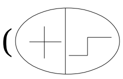
LTU

SVM

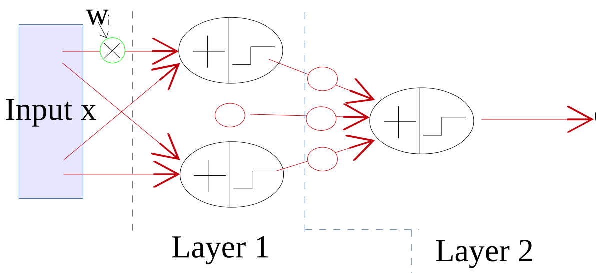
Perceptron



Limited Computational Power

Unit ()

Multi-Layer Perceptron

D1: 

Feed-forward data-flow diagram

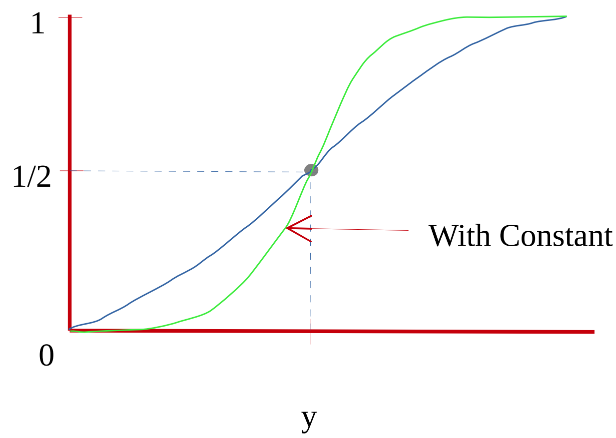
$$\boxed{y_i = \text{sign}(\Sigma_i w_{ij} x_i - \theta)}$$

sign = a function retuning +1 or -1 depending on wether it's input is positive or negative (abbreviated to s)

$$\boxed{s(y) = 1/(1+e^{-y})}$$

$$\boxed{E(w) = \Sigma_p 1/2 \; \| \hat{y}^{(?)} - y^{(p)} \|^2}$$

$\hat{y}(x;w)$



$\nabla_w E = (\partial E / \partial w_i)_i$

∇_E exists

How can we calculate this efficiently?

Mon 13 October

Aside:

$$f(x) = \lim_{h\to 0} (f(x+h) - f(x))/h$$

$$=(f(x+h) - f(x-\cancel{h}))/\cancel{2}h \Big|_{h = 0.001} + 0(h^\wedge\cancel{2})$$

Bad formula, quick and dirty derivative calculation

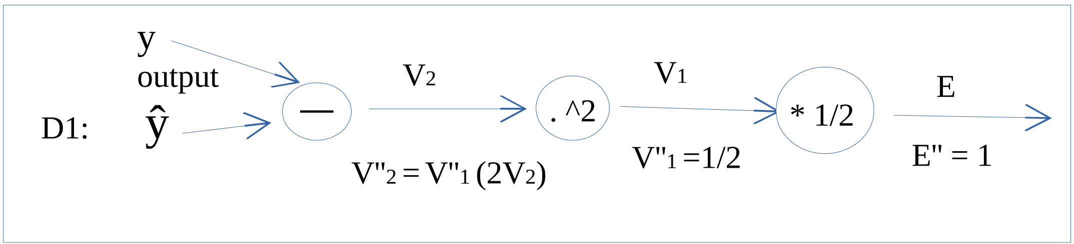
dim w = n

Overhead to calculate ∇E : (time to calculate E) * (dim w)

Backwards propogation of errors:
(Reverse mode accumulation automatic differentiation)

(time to calculate E) * ~~(dim w)~~

3 (safe with a 3)

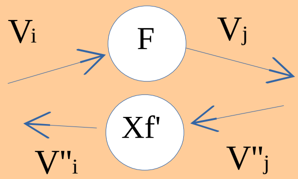


V

$\boxed{V'' = (\partial E) / V}$

$W''_i = (\partial E)/(\partial w_i)$

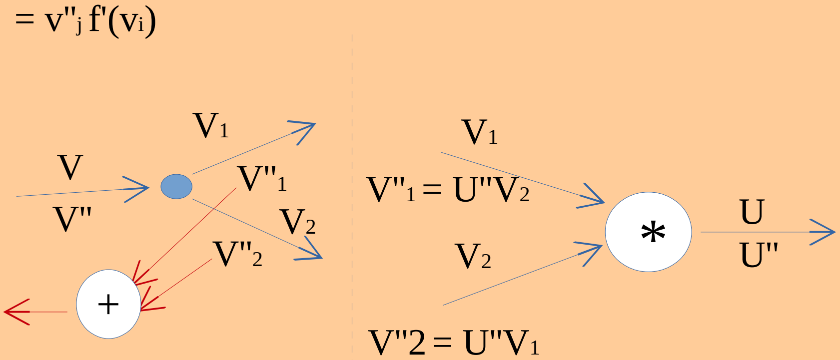
Trick: (Fan out)



$V_j = f(v_i)$

$V''_i = (\partial E) /(\partial V_i) = (\partial E) /(\partial V_j) * (\partial V_j) /(\partial V_i)$

$= v''_j f'(v_i)$



$V_1 = V_2{}^\wedge 2$

$V''_1 = (\partial E) /(\partial V_1)$

$V''_2 = (\partial E) /(\partial V_2)$

$= (\partial E) /(\partial V_1) * (\partial V_1) /(\partial V_2)$

$= (V''_1)(2v_2)$