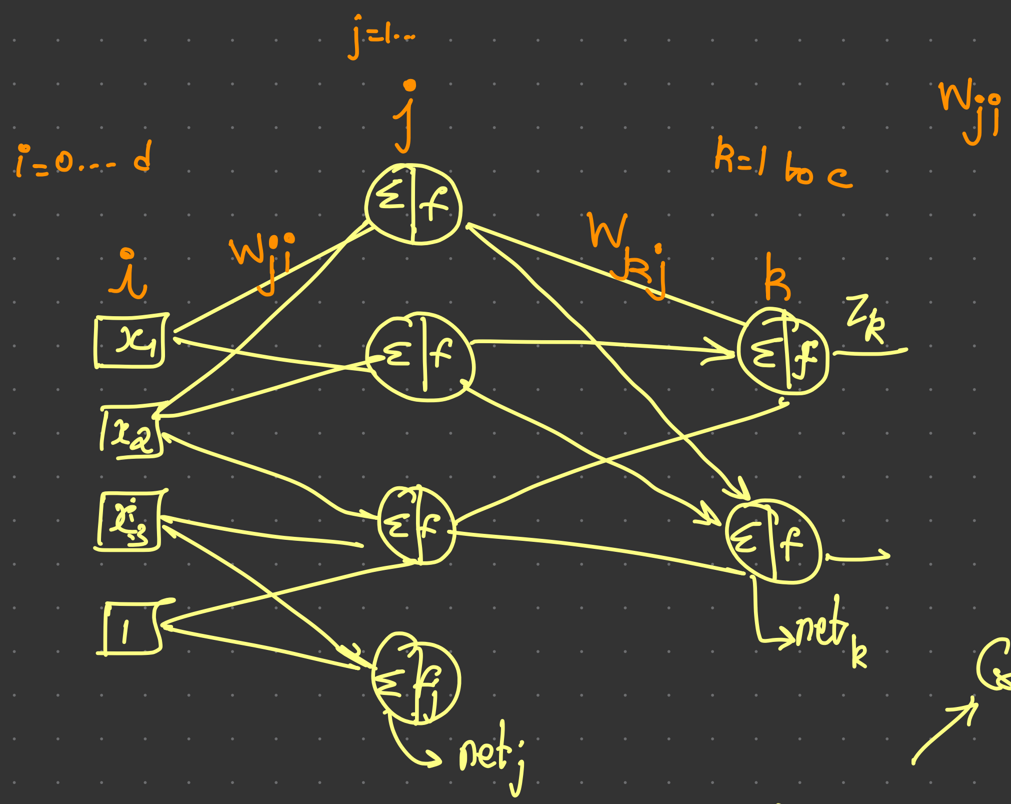
8th October, 2021



Credit Assignment Problem

Weights affect neuron just like a single perceptron

$\frac{\partial J}{\partial w_{ji}}$ → problem because in general there would be subsequent changes in w_{kj} & J

Back propagation Algorithms

$$J(w) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}\|t - z\|^2$$

↳ Vector form

$$\Delta w = -\eta \frac{\partial J}{\partial w} \quad \Big| \quad w = w - \eta \frac{\partial J}{\partial w}$$

$$\boxed{\Delta w_{pq} = -\eta \frac{\partial J}{\partial w_{pq}}}$$

→ Compute for both I/P & O/P layer

Updating ω_{kj}

$$\frac{\partial J}{\partial w_{kj}} = \underbrace{\overbrace{\frac{\partial J}{\partial net_k}}^{-\delta_k} \cdot \frac{\partial net_k}{\partial w_{kj}}}_{\text{Chain rule}} = -\delta_k \frac{\partial net_k}{\partial \omega_{kj}}$$

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} = (t_k - z_k) f'(net_k) \Rightarrow \underline{\delta_k = (t_k - z_k) f'(net_k)} \quad (1)$$

$$\frac{\partial J}{\partial \omega_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}}$$

$$\frac{\partial J}{\partial \omega_{kj}} = -\delta_k \cdot y_j \qquad \boxed{\Delta w_{kj} = \eta \delta_k y_j} \quad (2)$$

B) Updating w_{ji}

$$\frac{\partial J}{\partial \omega_{ji}} = \underline{\frac{\partial J}{\partial y_j}} \cdot \underbrace{\frac{\partial y_j}{\partial net_j}}_{f'(net_k)} \cdot \underbrace{\frac{\partial net_j}{\partial w_{ji}}}_{x_i} = \frac{\partial J}{\partial y_j} \cdot f'(net_k) \cdot x_i$$

$$\frac{\partial J}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2\right) = -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j}$$

$$= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j}$$

$$= -\sum_{k=1}^{c}(t_k - z_k) \cdot f'(net_k) \cdot w_{kj}$$

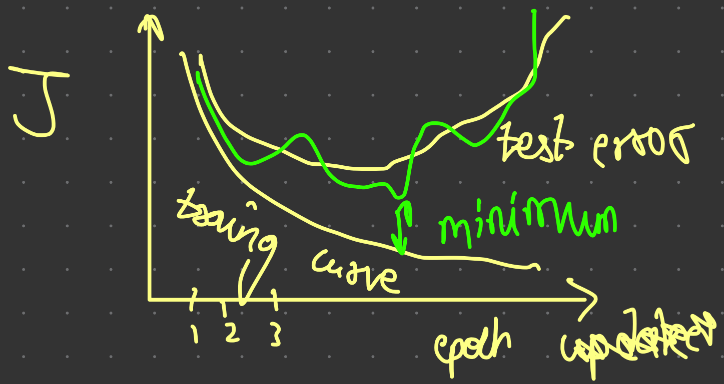
$$\boxed{\delta_j = f'(net_j) \sum_{k=1}^{c} w_{kj} \delta_k} \quad (3)$$

$$\frac{\partial J}{\partial w_{ji}} = -\delta_j x_i \quad \Big| \quad \boxed{\Delta w_{ji} = \eta\, \delta_j x_i} \text{ — (4)}$$

$$w^{t+1} = w^t + \Delta w$$

On an average increasing weights can increase | decrease

Learning curves



Validation set : Split part of training data

Iterate only till the minima is reached

Backprop Algorithms

a) Stocastic backprops

↳ Update after each sample

↳ Initialize $n_H, w, \theta, \eta, m = 0$

while ( ) {

$m = m + 1$

$x^m \rightarrow$ Randomly chosen

$w_{kj} \leftarrow w_{kj} + \eta \delta_k y_j$

$w_{ji} \leftarrow w_{ji} + \eta \delta_j x_i$

}

return (w)

Epoch's → Iterations

b) Batch Backprop

for (n=0 to N)

{ $x^n \leftarrow$ selecting one training sample

$\Delta w_{ji} = \Delta w_{ji} + \eta \delta_j x_i$

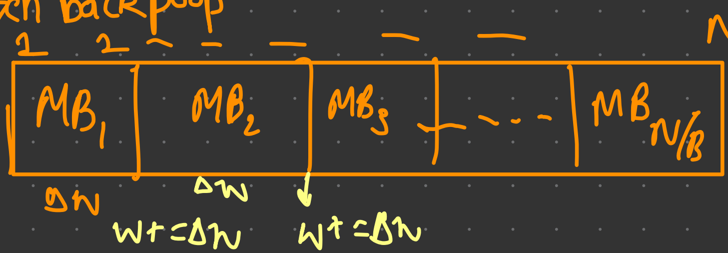
$\Delta w_{kj} = \Delta w_{kj} + \eta \delta_k y_j$

}

$w_{ji} += \Delta w_{ji}$ | $w_{kj} += \Delta w_{kj}$

→ Update only once per epoch

c) Minibatch backprop



11/10/2021

① Activation function

a) Continuity of f() & f'() & smooth as we need to get derivative

b) f() should saturate. [have some type of max & min]

- $f() \in [0,1]$

c) To make error function simple, so f() is monotonic & simple

d) f() should be linear around 0

→ Sigmoid satisfies all three properties

② Parameters of sigmoid :

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

To control unsaturated range, add scaling factor

$$f(net) = \frac{2a}{1+e^{-b \cdot net}} - a$$

a, b are parameters for appropriate range

Popular value: $\underbrace{a = 1.716,\ b = 2/3}$

$\Downarrow$

min/max $(f''(net))$ ± 2

$f(\ ) \rightarrow$ Antisymmetric

③ Input values:

If input values are too small, net is too small, sigmoid will become linear perceptrons

→ 0 mean, unit variance

④ Weight initialization

→ Should be in appropriate range

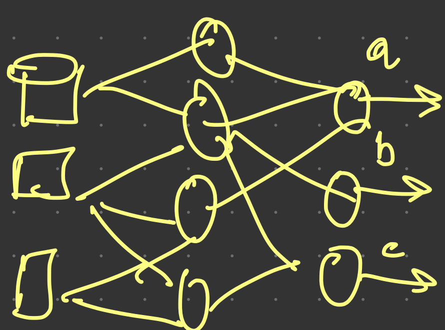
→ Should be random

→ Uniform weights from $w = U[-\hat{w}, \hat{w}]$

$\hat{w} = 1/\sqrt{d}$

$$\sum_{j=1}^{d} w_j y_j$$

⑤ Target Value:



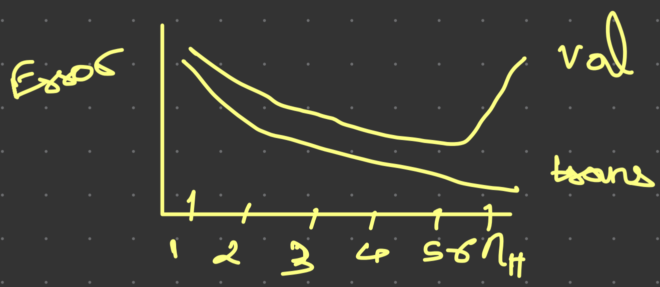
⑥ Training with noise

⑦ Manufac new data:

Adding new data → Data Aug

(3) # of hidden unit:

I/P layer	hidden layer	O/P layer
$d+1$	n_H	# of classi



(4) Learning rate

$$\eta_{opt} = \left(\frac{\delta^2 J}{\partial w^2}\right)^{-1} \qquad \eta_{max} = 2\eta_{opt}$$

In practice:

$\eta = 0.1$ as start

J diverges → reduce η

J is slow → increase η

η schedule → $\eta = 0.1$

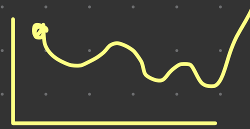
(5) Momentum

Multiple local minimas are possible

$$\Delta w_m \Rightarrow (1-\alpha)\Delta w_m + \alpha \Delta w_{m-1}$$

α

How much momentum we have



(11) Weight Decay [Revise]

$w \leftarrow w(1-\epsilon)$

Reduce everytime a little

(unwanted/ useless weights will all go to 0

Useful weights are those

More expressive powers

$$J_w = J(\,) + \underbrace{\frac{2\epsilon}{\eta} w^T w}_{\text{Regularizer}}$$

(12) Use of hints (Revise)

O
O } → Adding attributes using domain knowledge
O
O
O
O

(13) Loss fn / Criterion fn:

a) Square Error $J(w) = \sum (t_k - z_k)^2$

b) Minkowski $J(w) = \sum_k |(t_k - z_k|)^R$ → even

c) Cross entropy

$J(w) = \sum_{k=1}^{c} t_k \ln (t_k / z_k)$

(14) # of hidden layers

hidden layer > 1 ⇒ Deep network