"Unit-04"

DATE: __/__/____
PAGE: _____

Credit Assignment Problem: If any distributed system or any distributed network, then it is a common practice to give some credits or plain 3d or blame to each and every internal decisions so that it can be reflected on the overall outcome of the network.

⇒ Internal decisions are dependent on many actions. Among such, some actions are very prominent to get the outcome of internal decision. In this cases, the credit assignment is divided into two parts:

i) Temporal: If we have given credit or blame for outcomes of the internal decision, then this is called temporal credit assignment problem.

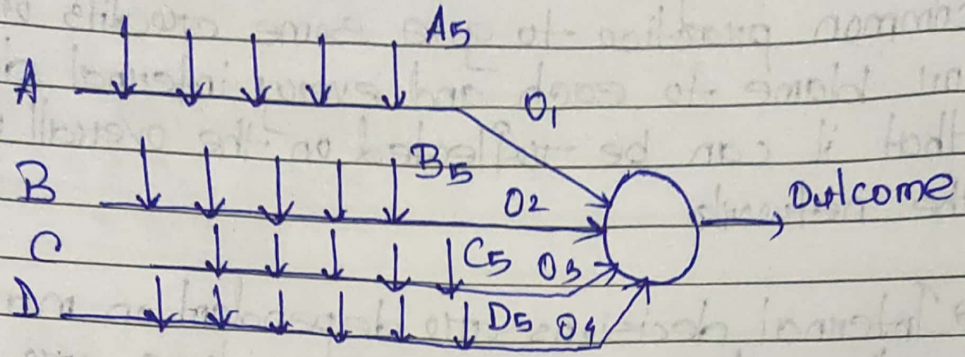
ii) Structural: If we have given credit or blame for outcomes of the actions of the internal decision, then this is called structural credit assignment.

⇒ temoporal = to get the selected outcomes of the internal decisions.

⇒ Structural = to get the selected outcome actions of the internal decision.

Good Write

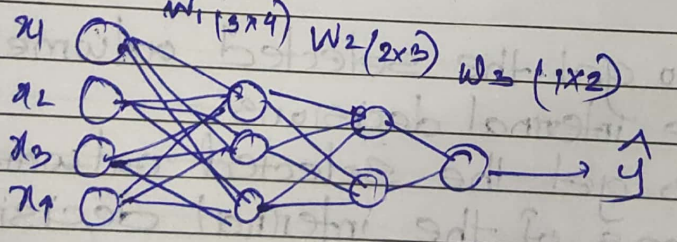
DATE: __/__/____
PAGE: 71



~~Here, internal decision means~~
Credit assignment problem, we need to give some credit or blame to ~~on~~ outcomes of internal decisions only. that means $O_1 - O_4$, we need to provide whether it is credit or blame
⇒ if credit is there, it will be included in the final outcome
⇒ if blame is there, it is not included in final outcome.

-o-

Delta Rule: (Perceptron learning)
⇒ derived from gradient descent.

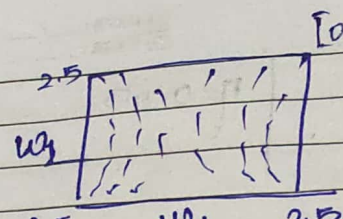


Whenever we design a neural network, there is a problem called "Symmetry Breaking".

When we are initially assigning all the weights equally, the model is updated equally and it is learning nothing. So, we assign weights randomly.

Good Write

DATE : __/__/____
PAGE : _____



It is forming a saddle shape across the error pattern. So, the error in one particular layer is mitigated with the help of adjusting the weights which are put randomly.

⇒ Our major goal is of Delta learning is to learn the weights so for that weight updation whenever we get the final output that is changed with the help of some rule.

Now,

error function, $J = \frac{1}{2} \sum_{d \in} (t_d - O_d)^2$

dependent on the weights and biases

t_d = train value
O_d = o/p function value or output value

Initially,

$$0 = w_0 + w_1 x_1 + w_1 x_1^2 + w_2 x_2 + w_2 x_2^2 + \dots w_n x_n + w_n x_n^2.$$

Here,

$$\frac{\partial J}{\partial w_i} = \left[\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \dots, \frac{\partial J}{\partial w_n}, \frac{\partial J}{\partial w_{n+1}}\right]$$

or, $$\frac{\partial J}{\partial w_i} = \frac{1}{2} \sum 2 (t_d - O_d) \cdot \frac{\partial J}{\partial w_i} (t_d - O_d)$$

or, $$\Delta \vec{w} = (t_d - O_d) \sum_{d \in D} \frac{2}{\partial w_i} (t_d - O_d)$$

$$= \sum_{d \in D} (t_d - O_d)/\partial w_i \left[t_d - \left(\vec{w} x_d + \vec{w} x_d^2\right)\right]$$

$$= \sum_{d \in D} (t_d - O_d) \left(-\vec{x}_{i,d} - \vec{x}_{i,d}^2\right)$$

$$= \sum_{d \in D} (t_d - O_d) \cdot - \left(\vec{x}_{i,d} - \vec{x}_{i,d}^2\right)$$

Good Write

$$= -\sum_{d \in D} (t_d - o_d)(\vec{x}_{i,d} + \vec{x}_{i,d}^2) \quad | \quad \eta = 0.05$$

$$= -\eta \nabla J[\vec{w}]$$

$$\therefore \Delta \vec{w}_i = -\eta \frac{\partial J}{\partial w_i}$$

Derivation of Back Propagation Algorithm :

⇒ Back Propagation : (training_exam, η, n_{in}, n_{out}, n_{hidden})

i) training example is a form (x, t) where, x is the vector of network input values, and t is the vector of target network values.

ii) η = learning rate (0.05)

iii) n_i = no. of network inputs

iv) n_{out} = no. of output units

v) n_{hidden} = no. of units in the hidden layer

- The input from unit i into unit j is denoted x_{ji}, the weight from unit i to unit j is denoted w_{ji}.

DATE: __/__/____
PAGE: _____

Algorithm:

step 01: Create a feed forward network with n_i inputs, n_{hidden} hidden units, and n_{out} output units.

step 02: Initialize all network weights to small random numbers.

step 03: Until the termination condition is met, Do, for each training set (x, t) in Do,

→ Propagate the input forward through the network.

→ Propagate the errors backward through the network.

• For each network k, calculate error term, $\delta_k \leftarrow o_k(1-o_k)(t_k - o_k)$

• For each network unit h, calculate it's error term,

$$\delta_h = o_h(1-o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

Step 04: Update each network weights.

$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ where, $\Delta w_{ji} = \eta \delta_j x_{ji}$

DATE: __/__/____
PAGE: _____

Derivation:

⇒ To derive the equation for updating weights in back propagation algorithm, we use Stochastic Gradient descent rule.

⇒ Stochastic gradient descent involves iterating through the training examples one at a time. For each training example (d), descending the gradient of the error → (Ed) with the respect to this single example

⇒ In other words, for each training example d every weight w_{ji} is updated by adding to it Δw_{ji}.

$$w_{ij} \leftarrow w_{ji} + \Delta w_{ji}$$

where,

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

where, E_d is the error on training example d that is half the square difference between the target output and the actual ~~pain~~ output over all outputs units in the network.

$$E_d(\vec{w}) \equiv \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

Here, outputs is the set of output units in the network, t_k is the target value of unit k for training example d, and o_k = output of unit k.

t_k = target value

o_k = output of unit k.

Good Write

DATE: __/__/____
PAGE: _____

Notation used:

x_{ji} = the i^{th} input to unit j.
w_{ji} = the weight associated with the i^{th} input to unit j
$net_j = \sum_i w_{ji} x_{ji}$ (the weighted sum of inputs for unit j
o_j = the output computed by unit j.
t_j = the target output for unit j.
σ = the sigmoid function.

outputs = the set of units in the final layer of the network

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

where,

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

1) Notice that weight w_{ji} can influence the rest of the network only through net_j.
∴ We can use the chain rule to write,

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$net_j = \sum_i w_{ji} x_{ji}$$

$$\frac{\partial net_j}{\partial w_{ji}} = x_{ji}$$

or, $$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \cdot x_{ji}$$

$$\therefore \Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} \cdot x_{ji}$$

To derive a convenient expression for $\frac{\partial E_d}{\partial net_j}$

We consider two cases in turn:

case 1: where unit j is an output unit for the network.
case 2: where unit j is an internal unit of the network.

Case 1: Training Rule for Output Unit Weights.

Just as w_{ji} can influence the rest of the network only through net_j, net_j can influence the network only through o_j.

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \times \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

or, $$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \times \frac{1}{2} (t_j - o_j)^2$$

$$= \frac{1}{2} \times 2 (t_j - o_j) \times \frac{\partial (t_j - o_j)}{\partial o_j}$$

$$= -(t_j - o_j).$$

$$\frac{\partial o_j}{\partial (net_j)} = \frac{\partial \sigma(net_j)}{\partial (net_j)}$$

$$= \sigma(net_j)(1 - \sigma(net_j))$$

$$= o_j (1 - o_j).$$

$$\frac{\partial \sigma}{\partial (x)} = \sigma(x)(1 - \sigma(x))$$

$$\boxed{\frac{\partial E_d}{\partial net_j} = -(t_j - o_j) o_j (1 - o_j)}$$

Now, $$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j) o_j (1 - o_j)$$

Good Write

$\therefore \Delta w_{ji} = \eta (t_j - o_j) o_j (1 - o_j) x_{ji}$

$\delta_j = (t_j - o_j) o_j (1 - o_j)$.

$\Delta w_{ji} = \eta \delta_j x_{ji}$

Case 2: Training Rule for Hidden Unit Weights

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream} \frac{\partial E_d}{\partial net_k} \cdot \frac{\partial net_k}{\partial net_j}$$

or, $$\frac{\partial E_d}{\partial net_j} = -\delta_k \frac{\partial net_k}{\partial net_j}$$

or, $$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream} -\delta_k \frac{\partial net_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j}$$

or, $$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream} -\delta_k w_{kj} \times \frac{\partial o_j}{\partial net_j}$$

or, $$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream} -\delta_k w_{kj} o_j (1 - o_j)$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$\Delta w_{ji} = \eta o_j (1 - o_j) \sum_{k \in D_1} \delta_k w_{kj} x_{ji}$$

$$\therefore \Delta w_{ji} = \eta \delta_j x_{ji}$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j) o_j (1 - o_j)$$

$\Leftarrow$ $\because$ $\delta_j = (t_j - o_j) o_j (1 - o_j)$.

$$\frac{\partial net_k}{\partial o_j} = \frac{\partial x_{kj} w_{kj}}{\partial o_j} = \frac{\partial o_j w_{kj}}{\partial o_j}$$

$$\frac{\partial o_j}{\partial (net_j)} = \frac{\partial \sigma(net_j)}{\partial (net_j)} = \sigma(net_j)(1 - \sigma(net_j))$$

$= o_j (1 - o_j)$

Good Write

DATE: __/__/____
PAGE: _____

Kolmogrov theorem:

⇒ Chapman-Kolmogrov equation was derived independently by both the British mathematician Sydney Chapman and Russian mathematic Andrey Kolmogrov.

Equation:

Suppose, $x_n, n = 0, 1, 2 \ldots$ is a homogenous Markov chain.

$$P_{ij}^{(m+n)} = \sum_K P_{iK}^{(m)} P_{Kj}^{(n)}$$

P_{ij} = conditional probability that the Markov chain gives from state i to j in $m+n$ steps.

P_{iK} = conditional probabilities of reaching an intermediary state K in m steps.

P_{Kj} = conditional probabilities from K reaching state j in n steps.

We know that, n-step transition probability:

$$P_{ij}^{(n)} = P(X_{n+1} = j \mid X_1 = i)$$

That is, $P(X_{m+n} = j \mid X_0 = i) = \sum P(X_m = K \mid X_0 = i)$

$$P(X_{m+n} = j \mid X_m = K)$$

Good Write

$$P_{ij}^{(m+n)} = P(X_{m+n}=j,\ X_m=k \mid X_0=i)$$

$$\text{or, } P_{ij}^{(m+n)} = \sum_k P(X_{m+n}=j,\ X_m=k \mid X_0=i)$$

$$= \sum_k \frac{P(X_{m+n}=j,\ X_m=k,\ X_0=i)}{P(X_0=i)} \qquad \left| P(A|B) = \frac{P(A \cap B)}{P(B)} \right.$$

$$= \sum_k \frac{P(X_{m+n}=j,\ X_m=k,\ X_0=i)}{P(X_m=k,\ X_0=i)} \cdot \frac{P(X_m=k,\ X_0=i)}{P(X_0=i)}$$

$$= \sum_k P(X_{m+n}=j \mid X_m=k,\ X_0=i) \cdot P(X_m=k \mid X_0=i)$$

$$= \sum_k P(X_{m+n}=j \mid X_m=k) \cdot P(X_m=k \mid X_0=i)$$

$$= \sum_k P_{kj}^{(m)} \cdot P_{ik}^{(m)}$$

In matrix notation,

$$\boxed{P^{(m+n)} = P^{(m)} \cdot P^{(n)}} \quad \text{or}$$

$$\boxed{P(m+n) = P(m)\,P(n)}$$