Sigmoid & Softmax

Notations.

Target $\rightarrow$ T. (Ground truth, Labels)

Obtained $\rightarrow$ Y (model.forward( ))

Model's output.

10 digit binary numbers.

$$\_\ \_\ \_\ \_\ \_\ \_\ \_\ \_\ \_\ \_ = 2^{10}.$$

1/0 1/0 . . . . .

$\downarrow\ \downarrow\ \downarrow\ \ldots\ \downarrow$

2 2 2 2

#1's > #0's ?

1010110101 $\longrightarrow$ 1

1111110000 $\longrightarrow$ 1

$\underbrace{0000000011}$ $\longrightarrow$ 0

$$\text{I/p} \longrightarrow \boxed{\text{Model}} \longrightarrow \text{O/p.}$$

(1×10) , (10×1)

0
0
0
0 } 10 neurons.
:
0

output → 0/1 (Sigmoid). Activation.

Pytorch :— (Pipeline)

- Data → web/ Simulated/ Collect.
- Model
- Ip → Model → prediction.
- Visualization.

Dataloader

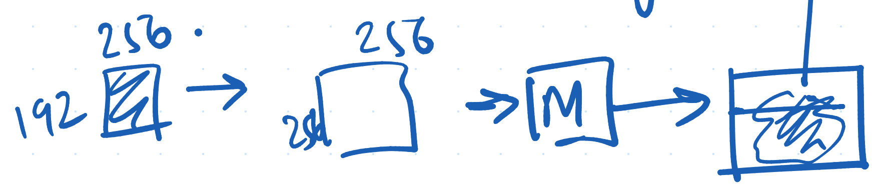
Raw Data → Preprocessing Aug / etc. → Model input

✓ train () → function.

✓ validation () → function.

✓ test () → function.

torch. no-grad(). torch.nograd()



→ prob. map.

Real values. ← [0 → 1]

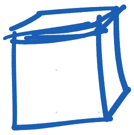
①

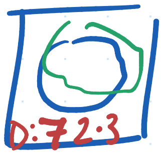
256 → total size.

save.

round(output).

$\left\{\begin{array}{l} 0.5\downarrow \to 0 \\ 0.5\uparrow \to 1 \end{array}\right\}$ threshold.

 (cv2.imwrite(——))



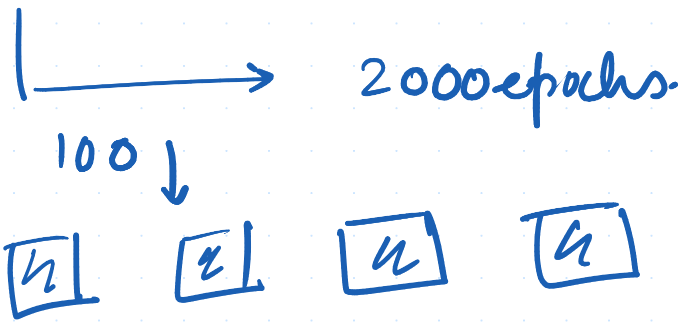
Dice → 70 %

Test function

↓

Similar to validation function with some added functionality. (Save). (test)

(validation X)



Logs.etc.

✓ Batch-wise → model → (cuda)

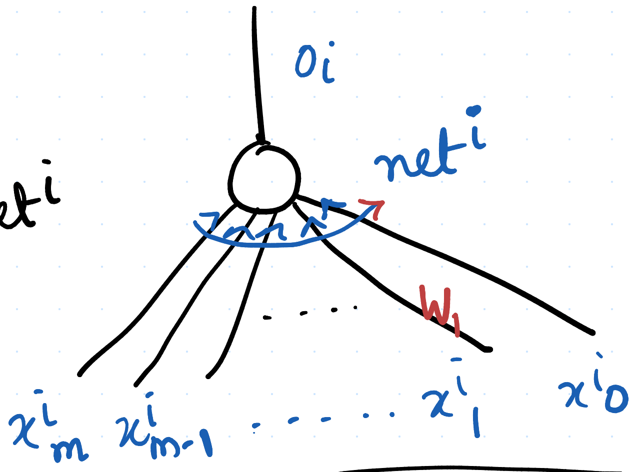
(Pipeline).

$\boxed{\text{Assignment} \rightarrow \text{(extend)}}$ ✓.

Target → T, Obtained → Y.

Sigmoid.

$$o^i = \frac{1}{1+e^{-x}} \quad = net^i$$



$$\boxed{o^i = \frac{1}{1+e^{-net^i}}}$$ ✓

X^i_j → ith sample
→ jth scalar.

$$\boxed{net_i = W \cdot X^i = \sum_{j=0}^{m} w_j x^i_j}$$

1 > 0 ?

$$X = \begin{array}{c} \\ 0th \\ 1st \\ 2nd \\ \vdots \\ N^{th} \end{array} \left\{ \begin{array}{c} \begin{array}{cccccccccc} 0 & 1 & 2 & \cdots & & & & & & 9 \end{array} \\ [1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1] \\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0] \\ [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \\ \vdots \\ [1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \end{array} \right\} \rightarrow Y = \left\{ \begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \end{array} \right\}$$

10 digit
1024 ip/op pairs.

$$X^0_7 = 1$$
$$X^1_7 = 0$$

$$(N+1) - \left\{ \begin{array}{c} [x^0_0, x^0_1, \ldots x^0_m] \rightarrow [y^0] \\ \vdots \\ [x^N_0, x^N_1, \ldots x^N_m] \rightarrow [y^N] \end{array} \right\}$$

Vectors → capital letters.
scalars → small letters.

$x^i \rightarrow$ ith input vector.

$o_i \rightarrow$ output scalar.

$net_i = W x^i$ (n-input/output observations).

$W =$ weight vector.

W & $x^i \rightarrow$ has (m-components).

$W = \langle w_m, w_{m-1}, \ldots, w_2, w_1, w_0 \rangle$

$x^i = \langle x^i_m, x^i_{m-1}, \ldots, x^i_2, x^i_0 \rangle$

Derivative of Sigmoid.

$o^i = \frac{1}{1+e^{-net_i}}$ $\rightarrow$ for ith input.

$\ln o^i = -\ln(1+e^{-net_i})$

$$\frac{1}{o^i}\frac{\partial o^i}{\partial net^i} = -\frac{1}{(1+e^{-net_i})} \cdot (-e^{-net_i})$$

$$= \frac{e^{-net_i}}{1+e^{-net_i}} = (1-o^i)$$

$$\boxed{\frac{\partial o^i}{\partial net^i} = o^i(1-o^i)} \rightarrow x(1-x)$$

-10 in pytorch.

$-\infty \rightarrow 0$
$+\infty \rightarrow 1$ } for sigmoid.

$+10 \rightarrow$ in pytorch.

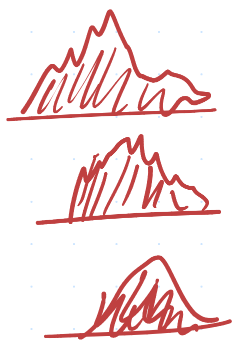
x = torch.tensor([[ ]])
t = torch.sigmoid(x)
print(t).

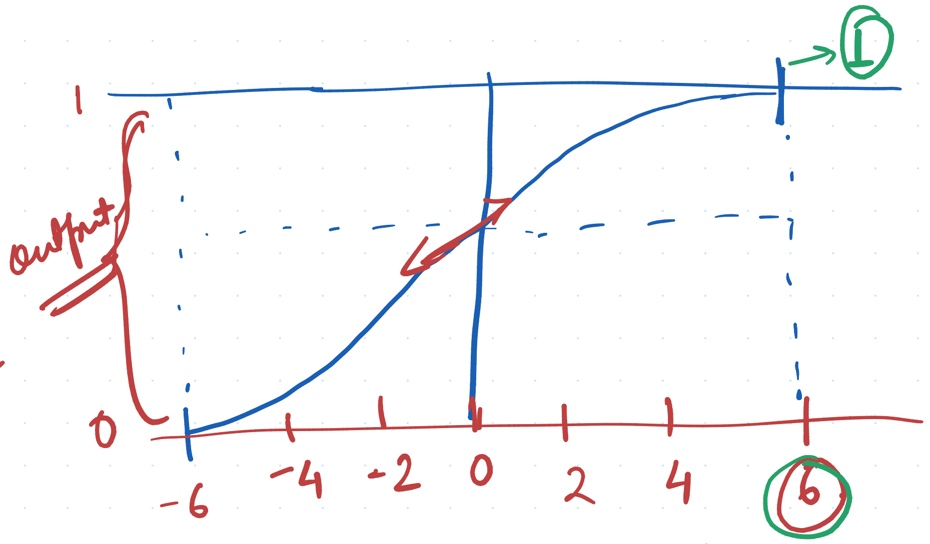
$x(1-x)$.

$X = [-10, -9, -8, \ldots\ldots, 0, 1, 2, \ldots, 9, 10]$

↓ (torch.sigmoid) ↓

$[0, 1e^{-20}, \ldots\ldots, \frac{1}{2}, \ldots\ldots 0.99, 1]$





Derivative of sigmoid is maximum at where?

input.

Maximum value of derivative.

(tune)

Normalize →stats.

Regularize.

overfit(x)

$\infty \rightarrow 1$

$\underbrace{6 \rightarrow 100}_{1}$

001.
100

$(-6, 6)$

$-100 \rightarrow 100$

$-100 \rightarrow -6$

$$\frac{\partial}{\partial x}\left(x(1-x)\right) \Rightarrow \frac{\partial}{\partial x}(x-x^2) \Rightarrow 1-2x$$

$$\left(\frac{x}{\|x\|}\right)$$

$$\boxed{\frac{x-\mu}{(std)}}$$

$$\downarrow$$

$$\frac{\partial}{\partial x}\left(x(1-x)\right)=0 \qquad 1-2x=0$$

$$\leftarrow \boxed{x=\frac{1}{2}}$$

Maximum value of derivative of sigmoid.

$$\frac{1}{2}\left(1-\frac{1}{2}\right)=\frac{1}{2}\left(\frac{1}{2}\right)=0.25.$$

$$o^i=\frac{1}{1+e^{-net^i}}$$

$$net^i=\infty$$

$\Sigma w x^i=\infty$

$$o^i=\frac{1}{1+\frac{1}{e^{net^i}}}=\frac{1}{1+\frac{1}{e^{\infty}}}=\frac{1}{1+0}=1$$

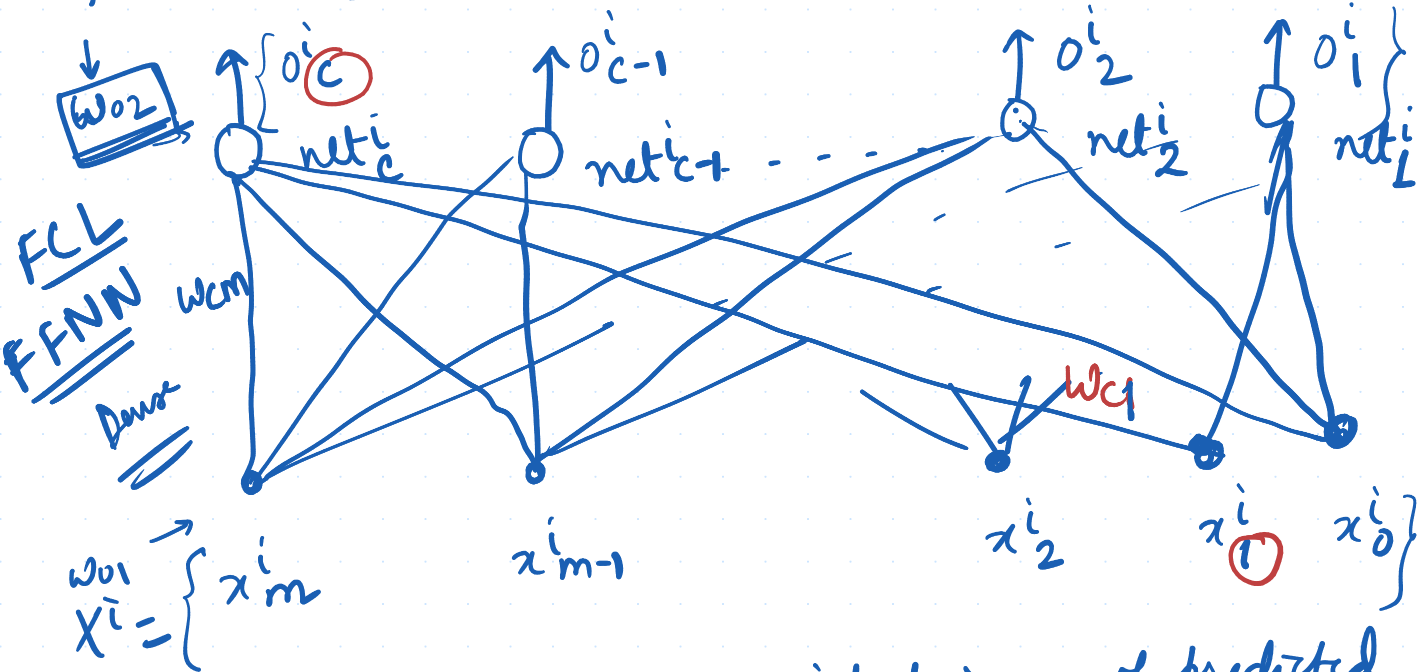
$$net_i=-\infty$$

$$o^i=\frac{1}{1+\frac{1}{e^{net_i}}}$$

$$=\frac{1}{1+e^{\infty}}=\frac{1}{\infty}=0$$

$y = mx + \boxed{c}$

Softmax



$m > c$ # of neurons in input > no. of predicted classes.

Output for class c, $c : 1$ to C.

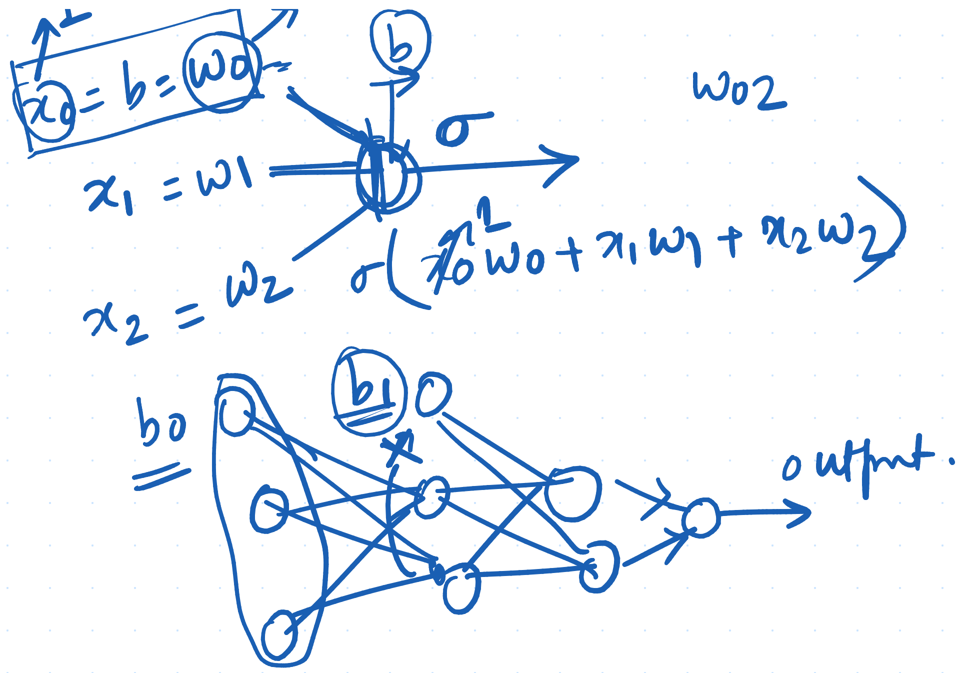
W_{c1}

$W_{(c-1)L}$ }.

$$\boxed{o^i_c = S(NET^i)_c = \frac{e^{net^i_c}}{\sum_{k=1}^{C} e^{net^i_k}}}$$

Softmax.

Rough



Notations.

$i = 0, 1, 2, \cdots N$ $\quad \{x^{(i)}\}_0^N$

(N+1) ip/op pairs. , i-runs over training data.

$j = 0, \ldots, m$, $(m+1) \rightarrow$ components.

also. size of weight vector

input vector $\rightarrow$ size.

output $o \rightarrow y$ $\quad$ # (c+1) class.

Rough

$$\mathcal{D} = \left\{ \begin{array}{l} [x_0^0, x_1^0, \ldots\ldots, x_m^0] \rightarrow [o_0^0, o_1^0, \ldots, o_c^0] \\ [x_0^1, x_1^1, \ldots\ldots, x_m^1] \rightarrow [o_0^1, o_1^1, \ldots\, o_c^1] \\ \quad\quad \vdots \quad\quad\quad\quad\quad\quad \vdots \quad\quad \vdots \\ [x_0^N, x_1^N, \ldots\ldots, x_m^N] \rightarrow [o_0^N, o_1^N, \ldots\, o_c^N] \end{array} \right.$$

$x \rightarrow (m+1)$

N (

$(N+1) \rightarrow$ i/p output pair.

$k = 0, \dots, C$, $C+1$ class.

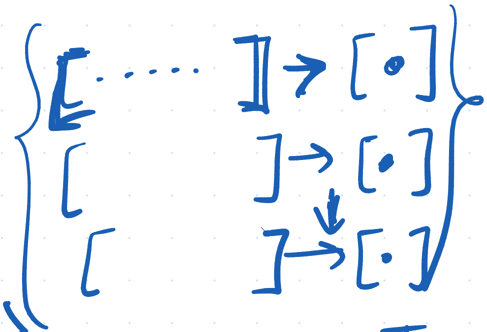
$(C+1) \rightarrow$ components in o/p vector.

$O^i, NET^i \rightarrow$ are the vectors for the i^{th} input.

$W_k \rightarrow$ weight vector for the c^{th} output neuron. $\underbrace{c = 1, 2, \dots C}$.

(Actual output) / GT.

Target vector $(T) \rightarrow$



T

$T^i = \langle t^i_c, t^i_{c-1}, \dots, t^i_2, t^i_1 \rangle$

$i \rightarrow$ for ith input.

One of these c-component is 1, rest are 0.

$$\begin{array}{ccc} 1 & 2 & 3 \\ \downarrow & \downarrow & \downarrow \end{array}$$

$\langle 0, 1, 0 \rangle \rightarrow 2$

$\langle 0, 0, 1 \rangle \rightarrow 3$

One hot vector ?

(1, 2, 3) output.

Softmax → (0 or 1) (class.)

$c \to 0, 1, 5$

classification. (2) $\boxed{\text{Fuzzy logic \& NN}}$

[0 − 1, 1 − 2, 2 − 3, ...]

$\downarrow$ 1, $\downarrow$ 2, $\downarrow$ 3

$\{(0-1](1-2], \ldots (N-1, N]\}^N$.

Regression X.

$\hat{x} \to \boxed{\text{Model}} \to \hat{y}$

Regression task. → continuous Real value.

$\hat{y} \to (0 - 100)$ $\langle 1, 0, 0 \rangle$

Softmax =

$$\begin{Bmatrix} -1 \\ 1 \\ 0 \end{Bmatrix} \xrightarrow{\text{softmax}} \begin{Bmatrix} 0.02 \\ 0.5 \\ 0.48 \end{Bmatrix} \text{argmax} \xrightarrow{\Downarrow} \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}.$$

$$\left\{ \frac{e^{-1}}{e^{-1}+e^{1}+e^{0}}, \frac{e^{1}}{e^{-1}+e^{1}+e^{0}}, \frac{e^{0}}{e^{-1}+e^{1}+e^{0}} \right\}$$

$\downarrow$

0 − 1   0 − 1   0 → 1

1 (Probability distribution)

(Andrew Ng) → CS ← (Daphne Koller).

Coursera
deeplearning.ai.

$\{(0,1], (1,2], \ldots (N-1, N]\}$.

↓ 1   ↓ 2   ↑   ↓ N

$N=5$.

Dog	Cat	fish	tree	human
↓	↓	↓	↓	↓
$\boxed{1}$	$\boxed{2}$	3	4	$\boxed{5}$

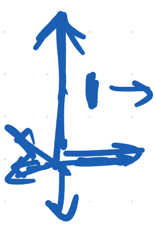
$\begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix}$ $\begin{pmatrix}0\\1\\0\\0\\0\end{pmatrix}$ $\begin{pmatrix}0\\0\\0\\0\\1\end{pmatrix}$

class 1 is more closer to class 2 than class 5.

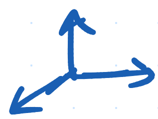
Inherent ordering → (X)

(1,0,0,0)

(Kilometers + Kg)



(0,1,0,0)



Softmax. $\left\{ o_c^i = \dfrac{e^{net_c^i}}{\sum_{k=1}^{c} e^{net_k^i}} \right.$, i^{th} input pattern.

$$\boxed{\ln o_c^i = net_c^i - \ln\left(\sum_{k=1}^{c} e^{net_k^i}\right)}$$

Derivative w.r.t. net_c^i :-

$$\frac{1}{o_c^i} \cdot \frac{\partial o_c^i}{\partial net_c^i} = 1 - \frac{e^{net_c^i}}{\sum_{k=1}^{c} e^{net_k^i}}$$

$\langle e^{net_1^i} \to 0, e^{net_2^i} \to 0, \ldots, \boxed{e^{net_c^i}}, e^{net_C^i} \to 0 \rangle$

Case 1 →

when class c for O & NET are the same.

$$= 1 - o_c^i$$

$$\boxed{\frac{\partial o_c^i}{\partial net_c^i} = o_c^i(1 - o_c^i).}$$

Case → 2 when class c' and net_c^i are different from c of o.

$$\boxed{\ln o^i_c = \underline{net^i_c} - \ln\left(\sum_{k=1}^{c} e^{net^i_k}\right)}$$

Derivative w.r.t. $net^i_{c'}$

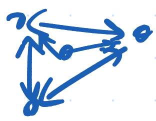
$$\frac{1}{o^i_c}\,\frac{\partial o^i_c}{\partial net^i_{c'}} = 0 - \frac{1 \times e^{net^i_{c'}}}{\sum_{k=1}^{c} e^{net^i_k}} = -o^i_{c'}.$$

$$\boxed{\frac{\partial o^i_c}{\partial net^i_{c'}} = -o^i_c\, o^i_{c'}}$$

when the classes are <u>unequal</u>.

Maximum Likelihood & Cross Entropy Loss.

Q) How did cross-entropy loss come to existence? (why this particular form?)



class value of input → R.V. (Random variable).

We are interested in modelling probability

$P(o^i | x^i)$, here $o^i \rightarrow$ output ~~vector~~ vector
$x^i \rightarrow$ input vector.

$o^i_c \rightarrow$ component of $o^i \rightarrow$ probability of x^i belonging to the class c $(c = 1, 2, \cdots C)$

c-components are redundant.

prob. of class $c \Rightarrow \underline{1 - \sum \text{prob.}(\text{class} \neq c)}$.

$$\langle 0.1, 0.2, 0.3, 0.4 \rangle = 1$$

$$\downarrow$$

$$\langle \underbrace{0.1, 0.2, 0.3}_{0.6}, \boxed{x} \rangle = \underbrace{1 - 0.6}_{\boxed{0.4}}$$

Hence in case of 2-class, only one sigmoid neuron is required.

$o^i \rightarrow$ value between 0 & 1 $\rightarrow$ intrepreted as probability

$$\sigma(\boxed{z}) \rightarrow [0 \rightarrow 1]$$

$o^i \rightarrow$ probability of class being equal to 1.

$P(\text{class} = 1 \text{ for ith input}) = o^i = \dfrac{1}{1+e^{-net_i}}$

Training data instance is labelled as 1 or 0.

Target value $\rightarrow$ 1/0 $\rightarrow t^i$ for ith input.

Likelihood of observation for (two classes)

For N no. of i/p - o/p pairs.

$$\mathcal{L} = \prod_{i=1}^{N} \underbrace{(o^i)^{t^i} (1-o^i)^{(1-t^i)}}$$

$, t^i = 1 \text{ or } 0.$

$\downarrow$

target

$o \rightarrow$ output.

$\downarrow$

Bernoulli Distribution.

(2 $\rightarrow$ Bernoulli)

Log-likelihood $\xrightarrow{\text{maximize.}}$ CE.

KL $\rightarrow$ CE.

$$\mathcal{L} = \prod_{i=1}^{N} (o^i)^{t^i} (1-o^i)^{(1-t^i)} \quad , t^i = 1 \text{ or } 0.$$

$o^i \rightarrow$ output.

$$LL = \log \left(\prod_{i=1}^{N} (o^i)^{t^i} (1-o^i)^{(1-t^i)} \right)$$

Log-likelihood $\rightarrow$ maximize

$$LL = \left[\sum_{i=1}^{N} \{ t^i \log(o^i) + (1-t^i) \log(1-o^i) \} \right]$$

Binary cross entropy loss. $\rightarrow$ Negative of log-likelihood
↓ minimize.

$$\Rightarrow -LL = -\sum_{i=1}^{N} \{ t^i \log o^i + (1-t^i) \log(1-o^i) \}$$

$$\text{BCE loss} = -\frac{1}{N} \sum_{i=1}^{N} \{ y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i) \}$$

L1, L2 loss why not?

D1 $X \longleftrightarrow Y$ D2.

$\left(\frac{1}{77}, \frac{1}{700000}\right) \rightarrow (0\cdots, 0.0001)$

$\longleftrightarrow$

(L1, L2)

(Binary cross entropy)?

c1 → (1,0,0)

c2 → (0,1,0)

$\frac{p_k}{q_k}$ → 0.001 → 0.02 → 2

$\underbrace{0.02 \rightarrow 2}_{100\%}$

X →

day 1

P(chance of acci) = 0.00001

P(chance of acci) = 0.001

100%

$\log\left(\frac{p_k}{q_k}\right)$

$(p_k - q_k)$

$\boxed{\log\left(\frac{p_k}{q_k}\right)}$

$p_k \log\left(\frac{p_k}{q_k}\right)$ → Probability / Distribution

(distance)

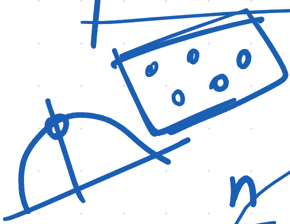
Distribution Difference

$$D_{KL}(p\|q) = \sum_{k=1}^{n} p_k \log\left(\frac{p_k}{q_k}\right)$$

$$\boxed{D_{KL}(p\|q) = \underline{H(p,q)} - H(p)}$$

$H(p,q) \rightarrow$ Cross entropy

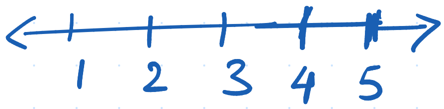
$H(p) \rightarrow$ Constropy



$$\sum_{k=1}^{n} p_k \log p_k \qquad - \sum_{k=1}^{n} p_k \log q_k .$$

GT label $\rightarrow p$,

$q \rightarrow$ model's output.

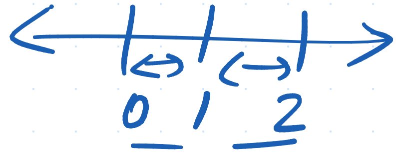


Entropy $\uparrow$ Information Gain $\downarrow$

$-LL \rightarrow$ negative of log likelihood is called Cross Entropy, Regarded as loss or Error.

$E \rightarrow$ Notation for Error.

Minimizing cross entropy brings o^i close to t^i, Hence we established, equivalence b/w maximization of likelihood $\Longleftrightarrow$



Softmax

$o^i_c \rightarrow$ value b/w 0 and 1, interpreted as probability, Multi-class situation.

$\rightarrow$ value is the prob of the class being 'c' for the ith input.

$$P(\text{class of } i\text{th input} = c) = o^i_c \, .$$

Likelihood L of observations in case of softmax :-

For N no. of i/p - o/p pairs:-

$$\mathcal{L} = \prod_{i=1}^{N} \prod_{k=1}^{C} (o_k^i)^{t_k^i} \; , \; t_k^i = 1/0.$$

For a pattern i, only one of t_k^i's is 1, rest are 0 :-

$$\boxed{\mathcal{L}\mathcal{L} = \sum_{i=1}^{N} \sum_{k=1}^{C} t_k^i \log o_k^i .}$$

$$\boxed{-\mathcal{L}\mathcal{L} = -\sum_{i=1}^{N} \sum_{k=1}^{C} t_k^i \log o_k^i}$$

Categorical Cross Entropy

For softmax also maximize the likelihood = minimize the cross-entropy.

$-\mathcal{L}\mathcal{L}$ = the cross entropy, Loss or Error.

Backpropagation & Gradient Descent.

Gradient Descent → Greedy Algorithm.

Always moves in the direction of reducing errors.

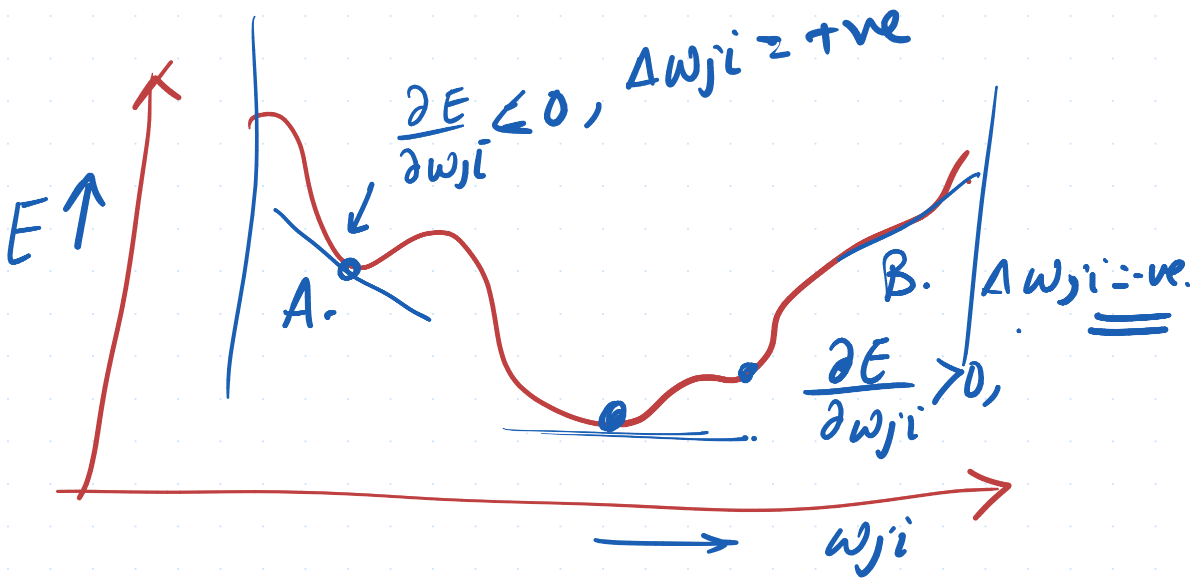
$$\boxed{\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}}$$

→ change in weight.

η = learning rate.

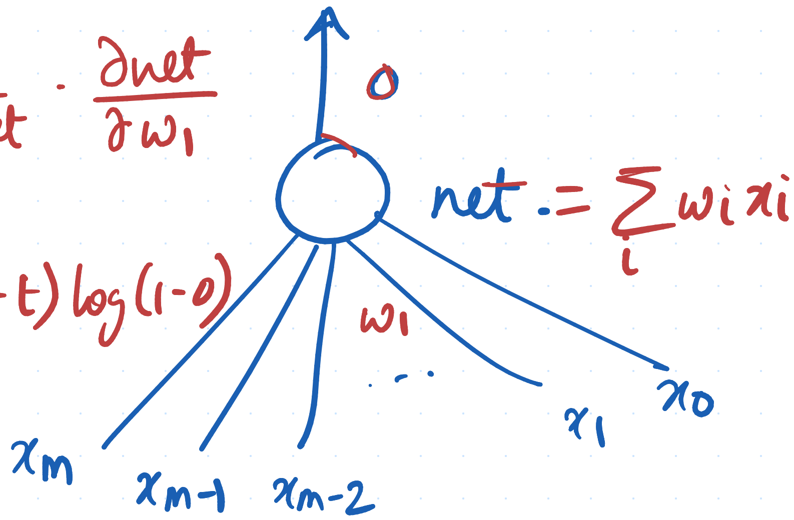
E = Loss function.

w_{ji} = weight of connection from the i^{th} neuron to the j^{th} neuron.



Single Sigmoid Neuron & Cross Entropy Loss, derived for single data point, hence, dropping right suffix i.

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial net} \cdot \frac{\partial net}{\partial w_1}$$



$$E = -t \log o - (1-t)\log(1-o)$$

$$\frac{\partial E}{\partial o} = -\frac{t}{o} - \frac{(1-t)}{(1-o)}(-1)$$

$$= \frac{-t(1-o) + (1-t)o}{o(1-o)} = \frac{-t + \cancel{to} + o - \cancel{to}}{o(1-o)}$$

$$\boxed{\frac{\partial E}{\partial o} = -\frac{t-o}{o(1-o)}}$$

Sigmoid Act.

$$o = \frac{1}{1+e^{-net}}$$

$$\boxed{\frac{\partial o}{\partial net} = o(1-o)}$$

$$\boxed{\frac{\partial net}{\partial w_1} = x_1}$$

$$\frac{\partial E}{\partial w_1} = -\frac{(t-o)}{\cancel{o(1-o)}} \times \cancel{o(1-o)} \times x_1$$

$$\frac{\partial E}{\partial w_1} = -(t-o)\,x_1$$

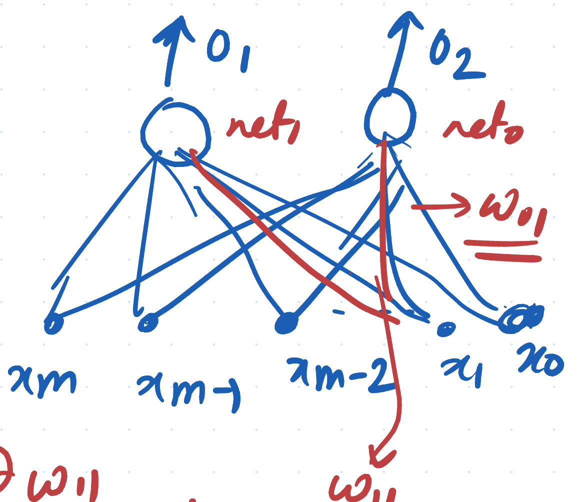
$$\boxed{\Delta w_1 = -\eta \frac{\partial E}{\partial w_1} = \underline{\underline{\eta\,(t-o)\,x_1}}}$$

Multiple Neurons. → in output layer

Softmax & Cross Entropy loss → Illustrated with 2 neurons.

$O = \langle o_1, o_0 \rangle$

$NET = \langle net_1, net_0 \rangle$



$$\frac{\partial w_{11}}{\partial w_{11}} = 1$$

∂w_{01} — 0

$$o_1 = \frac{e^{net_1}}{e^{net_1} + e^{net_0}} \qquad o_0 = \frac{e^{net_0}}{e^{net_1} + e^{net_0}}$$

$$\frac{\partial O}{\partial NET} = \begin{bmatrix} \frac{\partial o_0}{\partial net_0} & \frac{\partial o_1}{\partial net_0} \\ \frac{\partial o_0}{\partial net_1} & \frac{\partial o_1}{\partial net_1} \end{bmatrix} = \begin{bmatrix} o_0(1-o_0) & -o_0 o_1 \\ -o_1 o_0 & o_1(1-o_1) \end{bmatrix}$$

$$\boxed{\begin{array}{l} \text{Jacobian} \rightarrow \text{First order partial derivative} \\ \text{Hessian} \rightarrow \text{2nd order partial derivative.} \\ \text{m-tensor} \rightarrow \text{Higher order partial derivative.} \end{array}}$$

e^x. $\quad x = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$ $\rightarrow$ 3-tensor

4-tensor

$$E = -t_1 \log o_1 - (1-t_1) \log (1-o_1)$$

$$\frac{\partial E}{\partial w_{11}} = -\frac{t_1}{o_1} \frac{\partial o_1}{\partial w_{11}} - \frac{t_0}{o_0} \cdot \frac{\partial o_0}{\partial w_{11}}$$

$$\frac{\partial o_1}{\partial w_{11}} = \frac{\partial o_1}{\partial net_1} \cdot \frac{\partial net_1}{\partial w_{11}} + \frac{\partial o_1}{\partial net_0} \cdot \boxed{\frac{\partial net_0}{\partial w_{11}}}$$

$$= o_1(1-o_1)x_1 + -\cancel{o_1 o_0} \cdot 0 \rightarrow 0$$

$$= o_1(1-o_1)x_1$$

$$\frac{\partial o_0}{\partial w_{11}} = \frac{\partial o_0}{\partial net_1} \cdot \frac{\partial net_1}{\partial w_{11}} + \frac{\partial o_0}{\partial net_0} \cdot \frac{\partial net_0}{\partial w_{11}}$$

$$= -o_0 o_1 x_1 + 0$$

$$\frac{\partial w_{01}}{\partial w_{11}} = 0.$$

$$\frac{\partial E}{\partial w_{11}} = -\frac{t_1}{o_1} \cdot \frac{\partial o_1}{\partial w_{11}} - \frac{t_0}{o_0} \cdot \frac{\partial o_0}{\partial w_{11}}$$

$$= -\frac{t_1}{\cancel{o_1}} \cdot \cancel{o_1}(1-o_1)x_1 - \frac{t_0}{\cancel{o_0}}(-\cancel{o_0}o_1)x_1$$

$$= -t_1(1-o_1)x_1 + t_0 o_1 x_1$$

$$= x_1(-t_1+o_1) = -(t_1-o_1)x_1$$

$$\boxed{\Delta w_{11} = -\eta \frac{\partial E}{\partial w_{11}} = \eta (t_1 - o_1) x_1}$$