CS5480: Deep Learning
Homework 0

Ques 1

$$\text{Payout} = \begin{cases} +\text{Rs}100, & x=1 \\ -\text{Rs}25, & x \neq 1 \end{cases}$$

where (+) means payout to ELAN and (−) means payout to a visitor.

Expectation of gaining money (From ELAN's perspective) ⇒

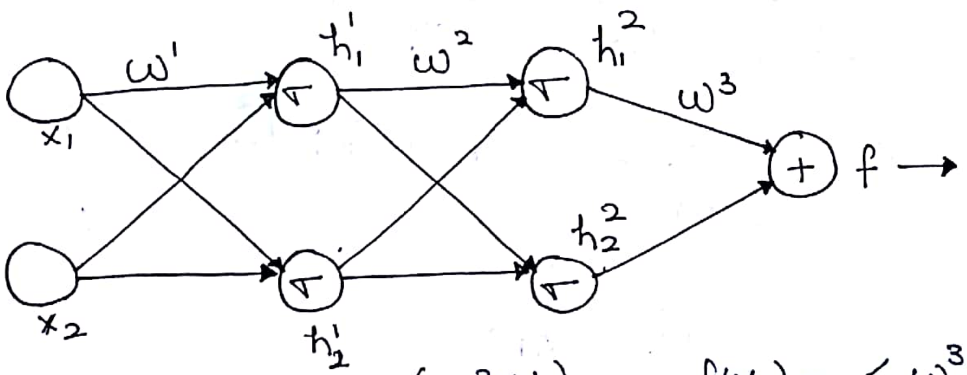
$$E(x) = \left(100 \times \frac{1}{6} - 25 \times \frac{5}{6}\right) \text{Rs}$$

$$E(x) = \frac{-25}{6} = -4.1667 \text{ Rs}$$

→ This means ELAN is expected to ~~would be~~ give money to visitors in most cases.

So it's not a good plan.

Ques 2



$$h^1 = \sigma(\omega^1 x), \quad h^2 = \sigma(\omega^2 h^1), \quad f(x) = \langle \omega^3, h^2 \rangle$$

→ h_k^2 denotes output of k^{th} node in layer 3

→ h_i^1 denotes output of i^{th} node in layer 2

→ x_j denote j^{th} input in layer 1

→ w_{ij}^1 denotes weight of j^{th} input in i^{th} node in layer 2.

Scanned by CamScanner

To Compute:

$$\frac{\partial f}{\partial w'_{ij}} = \frac{\partial f}{\partial h^2_K} \frac{\partial h^2_K}{\partial h'_i} \frac{\partial h'_i}{\partial w'_{ij}} \quad (\text{for all } K)$$

→ $\frac{\partial f}{\partial h^2_K} = w^3_K \Rightarrow \cancel{\begin{bmatrix} w^3_1 \\ w^3_2 \end{bmatrix}}$

→ $\frac{\partial h^2_K}{\partial h'_i} \Rightarrow h^2_K = \sigma(w^2_{K1} h'_1 + w^2_{K2} h'_2) = \sigma\left(\sum_{i=1}^{2} w^2_{Ki} h'_i\right)$

$$\frac{\partial h^2_K}{\partial h'_i} = (w^2_{Ki})\, h^2_K (1 - h^2_K)$$

→ $\frac{\partial h'_i}{\partial w'_{ij}} \Rightarrow h'_i = \sigma(w'_{i1} x_1 + w^2_{i2} x_2) = \sigma\left(\sum_{j=1}^{2} w^2_{ij} x_j\right)$

$$\frac{\partial h'_i}{\partial w'_{ij}} = h'_i (1 - h'_i) \cdot x_j$$

$$\frac{\partial f}{\partial w'_{ij}} = \frac{\partial f}{\partial h^2_K} \cdot \frac{\partial h^2_K}{\partial h'_i} \cdot \frac{\partial h'_i}{\partial w'_{ij}} \quad (\text{for all } K)$$

$$\Rightarrow \frac{\partial f}{\partial w'_{ij}} = w^3_K \, w^2_{Ki} \, h^2_K (1 - h^2_K)\, h'_i (1 - h'_i) \cdot x_j$$

(for all K)

$$\cancel{= w^3_1 w^3_{Ki}}$$

$$\cancel{= w^3_K} = w^3_1 \cdot w^2_{1,i}\, h^2_1 (1 - h^2_1)\, h'_i (1 - h'_i) \cdot x_j$$

$$+ w^3_2 \, w^2_{2,i}\, h^2_2 (1 - h^2_2)\, h'_i (1 - h'_i) \cdot x_j$$

$$\boxed{\frac{\partial f}{\partial w'_{ij}} = h'_i (1 - h'_i)\, x_j \left[w^3_1 \cdot w^2_{1,i} \cdot h^2_1 (1 - h^2_1) + w^3_2 \cdot w^2_{2,i}\, h^2_2 (1 - h^2_2) \right]}$$

Ques3

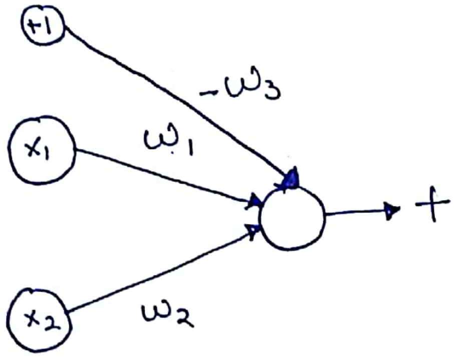
$$\Delta^{(2)}_{ij} := \Delta^{2}_{ij} + \delta^{(3)}_{i} * (a^{(2)})_j$$

As we can clearly see, while vectorising this equation we need to add two matrices of same size. And $\delta \times a$ should be computed such that $\delta_i \times a_j$ element is added to Δ^2_{ij}.

So we can simply rewrite the above equation as

$$\boxed{\Delta^2 := \Delta^2 + \delta^{(3)} * [a^{(2)}]^T}$$

Ques4



we can see that this perceptron will simply ~~give~~ learn a straight line $w_1x_1 + w_2x_2 - w_3 = 0 \Rightarrow w_1x_1 + w_2x_2 = w_3$

Now if $w_3 \neq 0$ we can always make another perceptron of type $\left(\frac{w_1}{w_3}\right)x_1 + \left(\frac{w_2}{w_3}\right)x_2 = 1$, where fixed bias would be 1.

But in our case $w_3 = 0$, so we cannot create another perceptron that will learn same function.

Another approach to look at this would be,

Our first Perceptron ~~neuron~~ will give ~~learns~~ $w_1x_1 + w_2x_2 = y$

Second Perceptron will give $w_1x_1 + w_2x_2 + 1 = y$

For these two outputs to be same

~~we can~~ $w_1 x_1 + w_2 x_2 = w_1' x_1 + w_2' x_2 + 1$

$\Rightarrow x_1(w_1 - w_1') + x_2(w_2 - w_2') = 1$

Here we are claiming that our perceptron learn some weights (v_1, v_2) s.t. $v_1 x_1 + v_2 x_2 = 1$ for every (x_1, x_2), which is clearly not possible

→ So, we cannot create another Perceptron with fixed bias 1 which will learn same function as a Perceptron with no bias.

Ques 5

$$f(x) = \log\left(\sum_{i=1}^{n} e^{x_i}\right)$$

(a) $$\frac{\partial f}{\partial x_i} = \frac{1}{\sum_{j=1}^{n} e^{x_j}} \times \frac{\partial\left(\sum_{j=1}^{n} e^{x_j}\right)}{\partial x_i}$$

$$= \frac{1}{\sum_{j=1}^{n} e^{x_j}} \cdot e^{x_i} = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

(b) Let's ~~call~~ assume $S = \sum_{j=1}^{n} e^{x_j}$

So, $$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \cdots \frac{\partial f}{\partial x_n}\right) \in \mathbb{R}^n$$

$$\Rightarrow \nabla f(x) = \left(\frac{e^{x_1}}{S}, \frac{e^{x_2}}{S}, \cdots \frac{e^{x_n}}{S}\right) \in \mathbb{R}^n$$

(c) Yes there exist a ~~chain~~ Matrix ~~cal~~ ~~out~~ Calculus equivalent of chain rule from Calculus.

Consider a matrix $g(x)$ which is acted upon by a function $f: \mathbb{R}^n \to \mathbb{R}^n$.

$h(x) = f(g(x))$. How to calculate $\frac{d f(x)}{dx}$

we can use ~~a~~ chain rule

~~a~~ $$\frac{d h(x)}{dx} = f'(g(x))\, g'(x). \quad [\cancel{f} g'(x)\, f'(g(x))]$$

The only condition imposed ~~a~~ in matrix calculus is that the order should remain same.

Ques 6

Let's first write Binary Cross entropy function which we will use as the error function in Neural Network.

Here y_i represent true labels and a_i represent predictions.

$$BCE(y) = -\sum_{i=1}^{n} y_i \log a_i + (1-y_i)\log(1-a_i)$$

In pure optimization we can just minimize above function.

So, if we use neural network to minimize the BCE loss we will get an optimal solution.

From Probability point of view

$$p(y|\pi) = \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i}$$

~~where π_i is~~ (Bernoulli for Binary classification)

For given inputs x_i and given output y_i

$$p(y|x,\theta) = \prod_{i=1}^{n} p_\theta(y|x_i)^{y_i}(1-p_\theta(y|x_i))^{1-y_i}$$

Since maximizing above function is hard and log is a monotone function, we can instead maximize log-likelihood

$$L(y|x,\theta) = \sum_{i=1}^{n} y_i \log p_\theta(y|x_i) + (1-y_i)\log(1-p_\theta(y|x_i))$$

Now as we can see maximising log likelihood and minimizing BCE gives same result.

For Multiclass classification one can use softmax cross-entropy as loss function and compare it with Multinoulli likelihood.

Ques 7

~~(i) Based on results obtained pred Perceptron and logistic regression are same. Both of them give probabilities of correct output.~~

(ii) Both the methods ~~are~~ give probabilities of the correct output.

(i) Solutions obtained by Logistic regression and Perceptron with sigmoid activation function are ~~same~~ more or less the same.
Perceptron follows online training and Logistic Regression follows offline training (closed form).

Ques 8

Ques 1 ~ 5 minutes

Ques 2 ~ 15 minutes

Ques 3 ~ 1 minute

Ques 4 ~ 10 minutes

Ques 5 ~ 10 minutes

Ques 6 ~ 20 minutes

Ques 7 ~ 5 minutes

Ques 8 ~ 2 minutes