Q3

x_0	x_1	x_2	t
1	0	0	1
1	1	1	1
1	1	0	0

let $w_0 = 1$ $\quad w_1 = -1$ $\quad w_2 = 1$

let $x_0 = (2x_1 - 1)^2$ problem so the points

This problem is similar to XOR we have taken x_0 as

are not linearly seperable, so

quadratic function.

$w_0 x_0 + w_1 x_1 + w_2 x_2$ for $(1\,0\,0)$ = 1

$w_0 x_0 + w_1 x_1 + w_2 x_2$ for $(1\,1\,1)$ = 1

$w_0 x_0 + w_1 x_1 + w_2 x_2$ for $(1\,1\,0)$ = 0

$w_0 x_0 + w_1 x_1$

If we make all three weights positive then in all cases we get positive output.

If we make all three negative then we get negative output in all cases.

If we make first two positive i.e w_0, w_1 and last one negative we incorrectly classify third case. $(1\,1\,0)$

If we make first two negative i.e w_0, w_1 and last one positive we incorrectly classify third case.

Q5

1. The filter size is $2 \times 2 \times 4$ for each filter.
There are in total 3 filters so no of channels = 12.

2.

0	1	0	1	0	1	0
0	0	1	0	1	0	0
0	1	0	1	0	1	0
0	0	1	0	1	0	0
0	1	0	1	0	1	0

output :-

0	0	0
0	0	0

3.

$\begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}$ (trivial solution)

$\begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$ (non trivial solution)

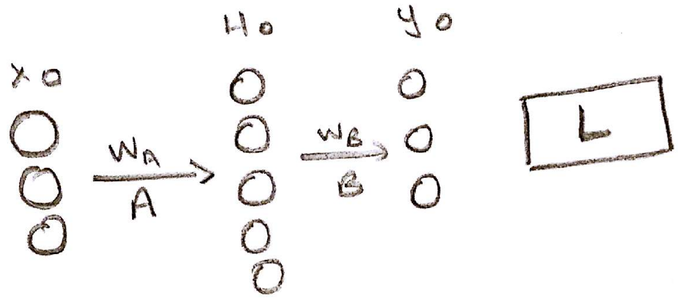
4.

$\begin{matrix} 1 & -1 \\ 0 & 0 \end{matrix}$

5. 4 filters were applied on layer 2 to get the feature maps in layer 3.

6. The first two feature maps in layer 3, I guess let us know that there might be a straight line at centre and a diagonal from left corner in image of layer 2.

Q 2



1) $J_x = J_B \times W_B^T \times J_a \times W_A^T$

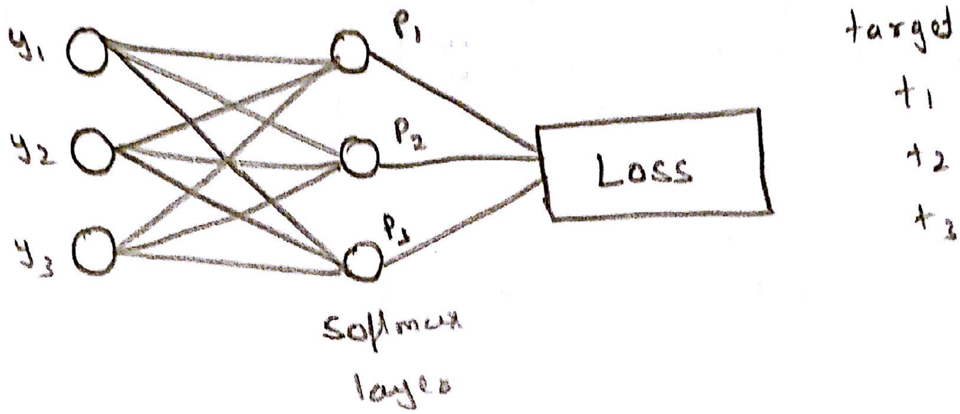
2) Dimension of J_x = [len (y_o) , len (x_o)]

3) $E_Y = \frac{\partial L}{\partial Y_o}$

$$\frac{\partial L}{\partial x_o} = E_y \times J_x^T$$

Scanned by CamScanner

option-2



Cross Entropy Loss $\quad L = -\sum_i t_i \log(P_i)$

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial P} \times \frac{\partial P}{\partial y}$$

$$\frac{\partial L}{\partial P} = -\sum_i t_i \times \frac{1}{P_i}$$

Now $\quad P_i = \dfrac{\exp(y_i)}{\sum_{k=1}^{n} \exp(y_k)}$

$$\frac{\partial P_i}{\partial y} = \frac{\partial \dfrac{\exp(y_i)}{\sum_{k=1}^{n} \exp(y_k)}}{\partial y}$$

Now if $i = j$ $\quad \dfrac{\partial \exp(y_i)}{\partial y(i)} = \exp(y_i)$

if $i \neq j$ $\quad \dfrac{\partial \exp(y_i)}{\partial y_{(j)}} = 0$

For $i = j$ $\quad \dfrac{\partial P}{\partial y} = P_i(1 - P_j)$

$i \neq j$ $\quad \dfrac{\partial P}{\partial y} = -P_i \cdot P_j$

$$\frac{\partial L}{\partial y_i} = \left(-\sum_k t_k \times \frac{1}{p_k} \right) \times \left(\underbrace{p_i(1-p_k)}_{\text{when } k=i} + \underbrace{(-p_i p_k)}_{k \neq i} \right)$$

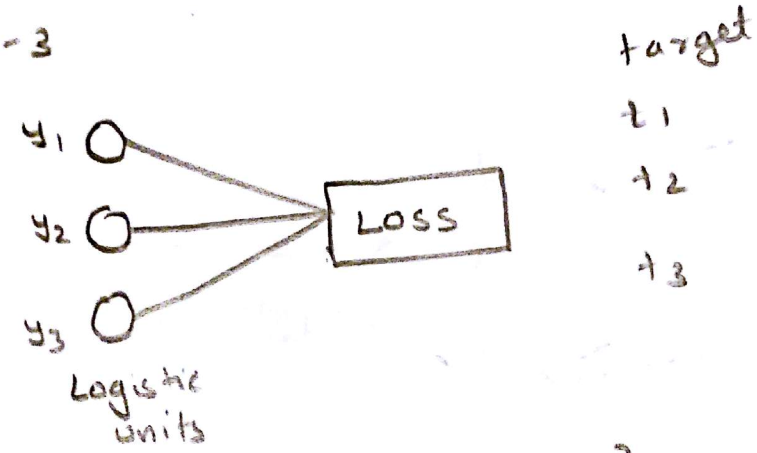
$$= -t_i(1-p_i) + \sum_{k \neq i} t_k \times p_i$$

$$= -t_i + p_i t_i + \sum_{k \neq i} t_k \times p_i$$

$$\frac{\partial L}{\partial y_i} = p_i - t_i \qquad \left(\sum_{k \neq i} t_k + t_i \right) = 1 \text{ because}$$

t is one not vector

Option - 3



$$\text{Loss} = \frac{1}{2} \sum_{i=1}^{n} (y_i - t_i)^2$$

$$y = \sigma(z)$$

$$y = \frac{1}{1 + e^{-z}}$$

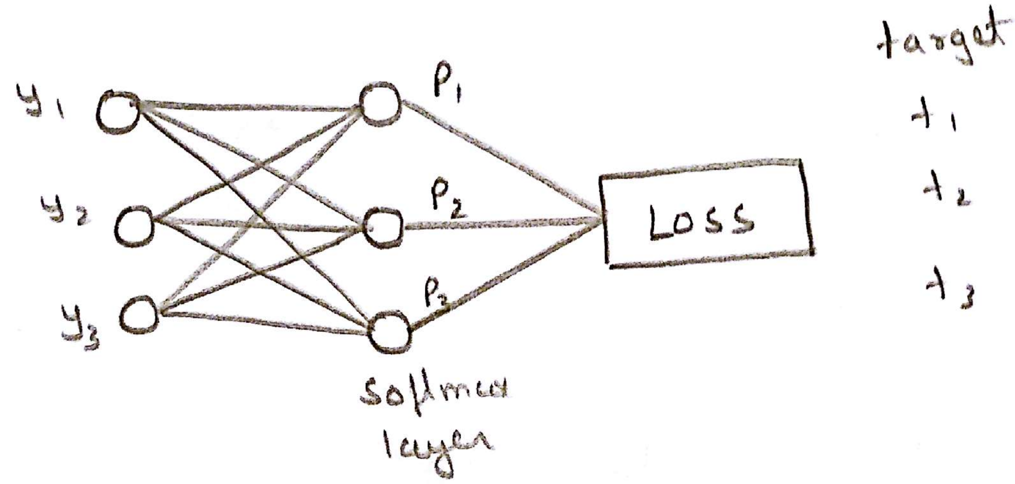
$$\frac{\partial \text{Loss}}{\partial z} = \frac{\partial \text{Loss}}{\partial y} \times \frac{\partial y}{\partial z}$$

$$\frac{\partial \text{Loss}}{\partial y} = \sum_{i=1}^{n} (y_i - t_i)$$

$$\frac{\partial y}{\partial z} = y(1 - y)$$

$$\frac{\partial \text{Loss}}{\partial z} = (y - t) \times y(1 - y)$$

option - 2



$$Loss = \frac{1}{2} \sum_{i=1}^{n} (p_i - t_i)^2$$

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial P} \times \frac{\partial P}{\partial y}$$

$$\frac{\partial L}{\partial P} = \sum_{i=1}^{n} (P_i - t_i)$$

$$P = \frac{\exp(y_i)}{\sum_{k=1}^{n} \exp(y_k)}$$

$$\frac{\partial P}{\partial y} = P_i (1 - P_j) \qquad i = j$$

$$= -P_i \times P_j \qquad i \neq j$$

$$\frac{\partial Loss}{\partial y} = \sum_{k} P_k - t_k \times (P_i (1 - P_k) - P_i P_k)$$

$$= (P_i - t_i) \times P_i (1 - P_i) + \sum_{k \neq i} (P_k - t_k) \times (-P_i P_k)$$

Identity, MSE

From the notes given in class it shows an example where our model confidently predict a negative lable with $z = -5$, which gives $y \approx 0.0067$, for a positive example ($t = 1$). Plugging these values we get $\frac{\partial L}{\partial z} = -0.0066$. This is pretty small value in comparision to our mistake. It shows that the more confident the wrong prediction the smaller the gradient is.

Softmax, Cross Entropy

Here we get the output $\frac{\partial L}{\partial y}$ similar to the linear regression. So if $y > t$ you made too positive a prediction, so you want to shift predition in negative direction. Conversely if $y < t$, you want to shift your prediction in positive direction.

Softmax, MSE

The problem with MSE in the classification task is that does not distinguish bad predictions from exteremly bad predictions. If $t = 1$ then predition of $y = 0.01$ has roughly same loss ~~th~~ as for prediction of $y = 0.00001$, even though the latter is more wrong.

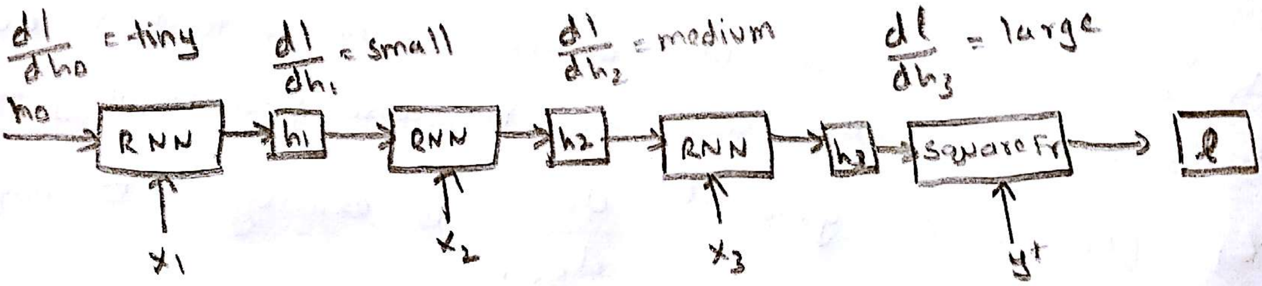
So I would suggest Jim to use softmax cross entropy loss.

Scanned by CamScanner

Q4

Here there is verb "go" and with that there is depedency "library". Then people borrow books from library" so yo have borrowed all of shakespeares work". Then you have verb "read" which is also dependent on "library" as it is place to read. Now at last we have "bench" which is dependend on verb "read".

The sentence starts with "I" and in the last sentence we see that dependency "I will read...".



RNN makes it possible to model long distance dependencies because they have ability to pass information between timesteps.

Ex: If some node h_{t-1} encode the information that "the subject of sentence is male", it is possible to pass this info to h_t which in turn can pass to h_{t+1}.

Suppose we have a RNN that predicts after several time steps and calculates loss that is expected to back propogate over all time steps. At each time step we run back propogation, the gradient gets smaller and smaller, and by the time we get back to begining time steps, we have a gradient that is so small that it has no significant effect on parameters to be updated.

The reason this happens because we have either $0 < \frac{dh}{dh_{t-1}} < 1$ or $\frac{dh}{dh_{t-1}} > 1$ and we want $\frac{dh}{dh_{t-1}} == 1$. We do not have $\frac{dh}{dh_{t-1}}$ exactly 1 so we face vanishing or exploding gradient problem.

One method to solve this problem, is the use of neural network architecture that is specifically designed to ensure that derivative of recurrent function is 1. This neural network architecture is LSTM.

The most fundamental idea behind LSTM is that in addition to standard hidden state h_t it also has memory cell c_t for which gradient $\frac{dc_t}{dc_{t-1}}$ is exactly 1. Because the gradient is 1 it do not suffer from vanishing or exploding gradient problem and can capture long term dependency efficiently.