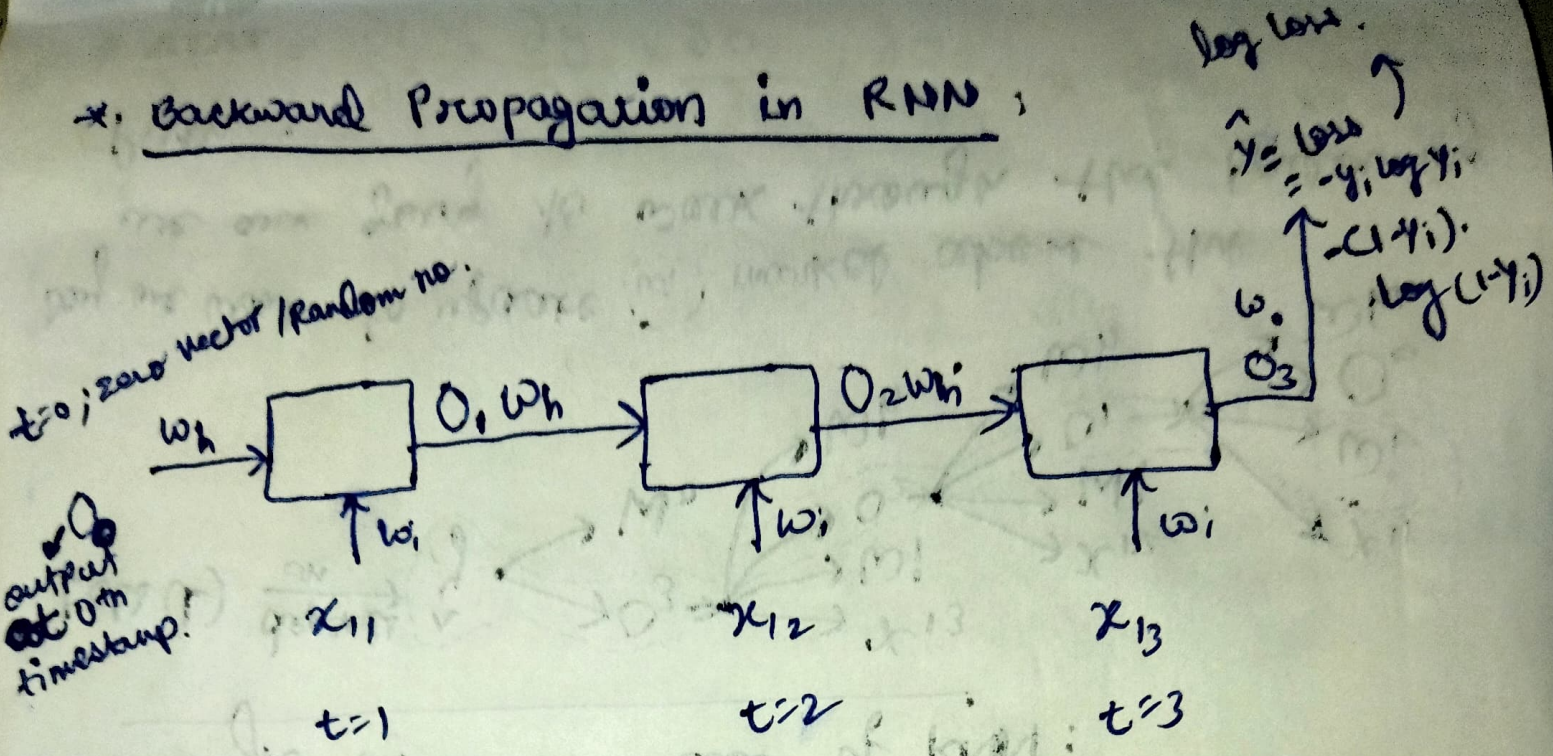
*. Backward Propagation in RNN :

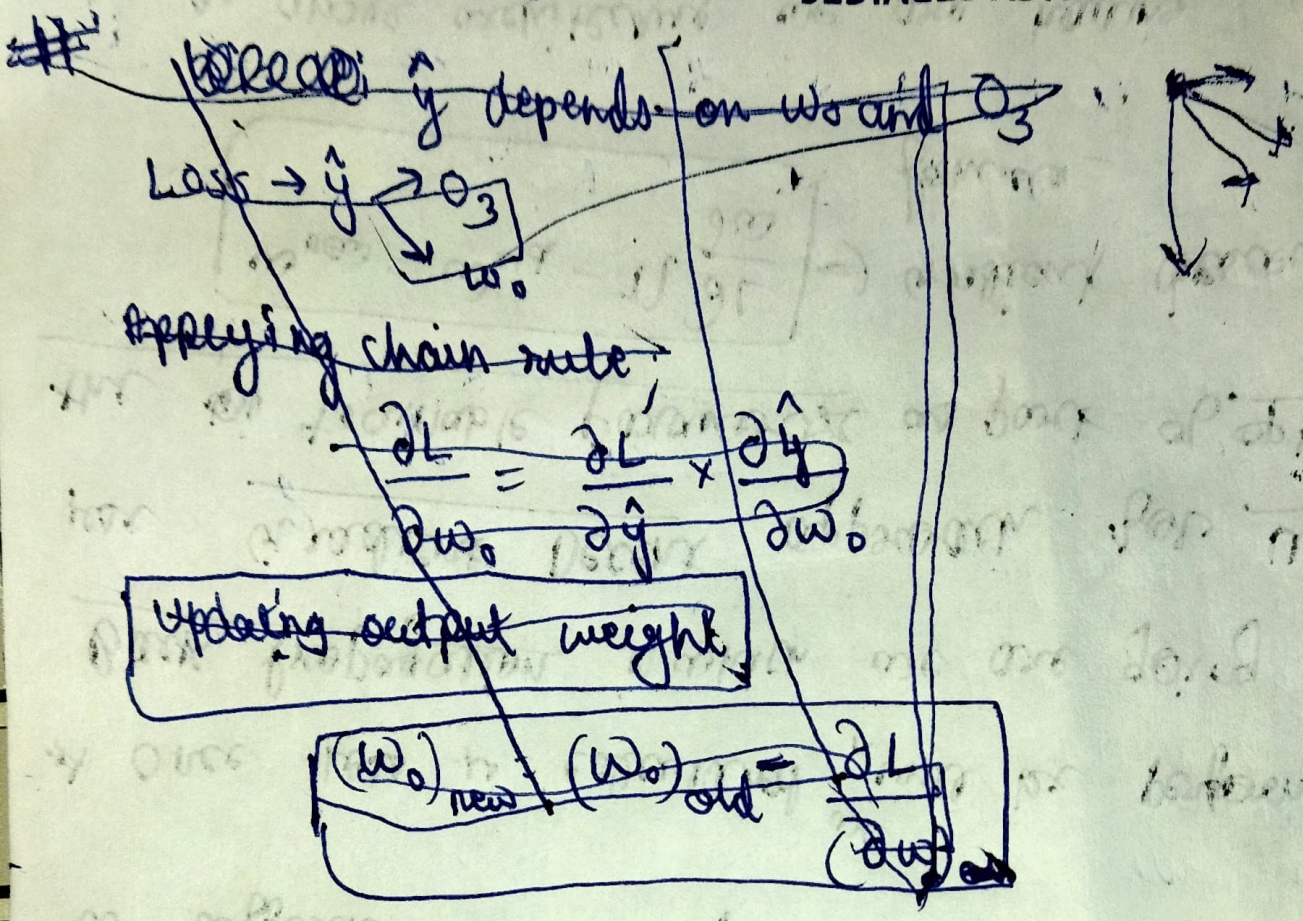


* Above each Box represents a common hidden neuron at different time stamp

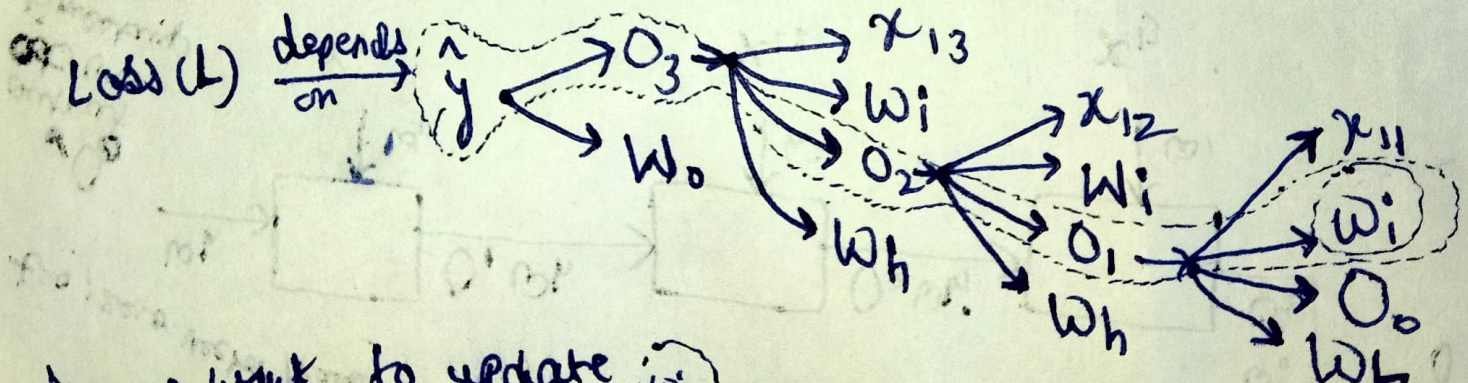
* Once loss is calculated we will be performing Back Propagation in which we are going to use Gradient Decent approach for updating the trainable parameters as part of optimisation.

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$$ ⇒ Gradient descent formula.

* In above architecture we are having 3 diff! weights
  - w_i = weight passed during I/P
  - w_h = weight " " in hidden layer neuron.
  - w_o = " " " O/P layer.



Drawing dependency map for Back Propagation Through time in case of RNN:



Say we want to update W_i marked above the we are going to walk through the following path.

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_3} \times \frac{\partial o_3}{\partial o_2} \times \frac{\partial o_2}{\partial o_1} \times \frac{\partial o_1}{\partial w_i}$$ → weight updation for 3 time stamp

or

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial w_i} + \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_i}$$

$$+ \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_i}$$ * Recall RNN lectures BPT

Problem with RNN:

- Exploding gradient (When weight value very high)
- Vanishing gradient (When weight value very low).

For Eg: India is a great country. There are many languages. You will find out and diversity is also very high. Here most of the people speak hindi?

>>> If we are going to process such huge sentence using simple RNN then it might happen that it losses the context with such a huge time stamp requirement. For Eg: In above sentence

If we are going to ask RNN that where most of the people speak hindi, then it might not give an answer as India due to huge sentence length.

** Reason why RNN losses content in case of huge sentence is it is prone to vanishing gradient & exploding gradient (since internally it uses gradient descent approach for optimisation of trainable params.

** Vanishing gradient occurs due to long term dependencies (long sentences).

↓

If to much sentences the weight updation formula mapping dependencies in chain (as discussed earlier for 3 ts) will be very huge & computational expensive. which leads to loss of context.

** Recall Vanishing and exploading problems from ANN lectures.

* LSTM (Long Short Term Memory)

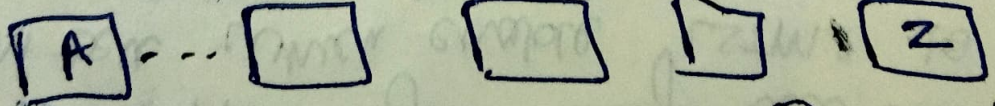
→ RNN is the Grajani suffering from short term memory. There is no mechanism in place within RNN that can basically enable it hold long term dependency of hold the context in case of large sentences.

→ So, LSTM is basically an update to the RNN cell that can eliminate the stated problem ~~of short term~~ with long term dependency.

→ upto what time stamp can RNN hold the context?

Ans: There is no such number as it may vary with sentences passed. This is more sort of experimental in nature.

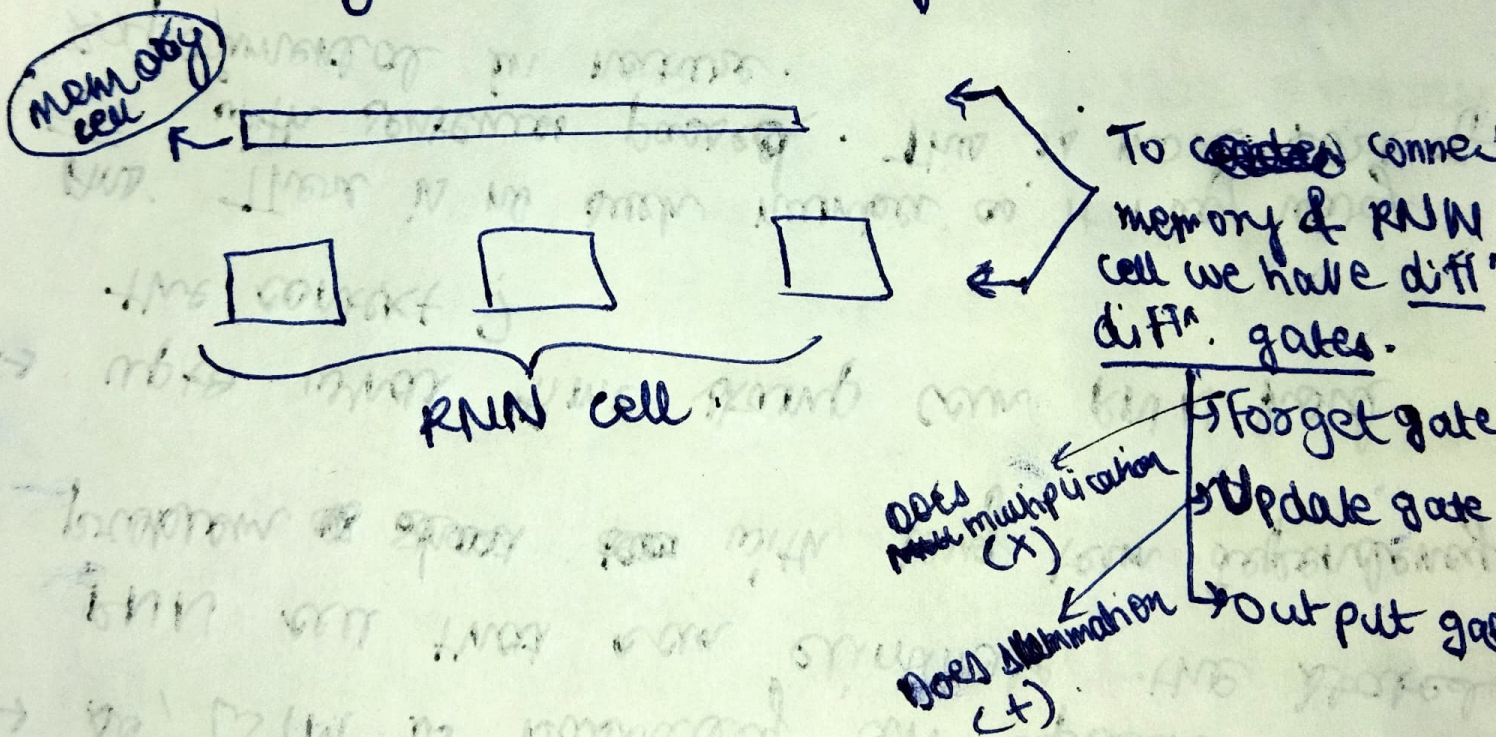
→



If (Z) cell wishes to know the state ~~of~~ context held by cell (A) which is very far so in RNN we are not provided with any mechanism -

→ LSTM is an improvement over RNN which is capable to hold both long and short term dependencies.

→ we are appending memory cell on top of RNN cell which enable LSTM to hold the long term dependency.



Using these gates we are going to update the latest information to the memory cell that will in turn help in maintaining / holding the long term dependencies.

→ Google LSTM interview question.

→ How size of hidden state calculated?

AND: No. of neurons we wish to take inside

Hidden layer = Size of Hidden State

→ See the practical in .ipynb file