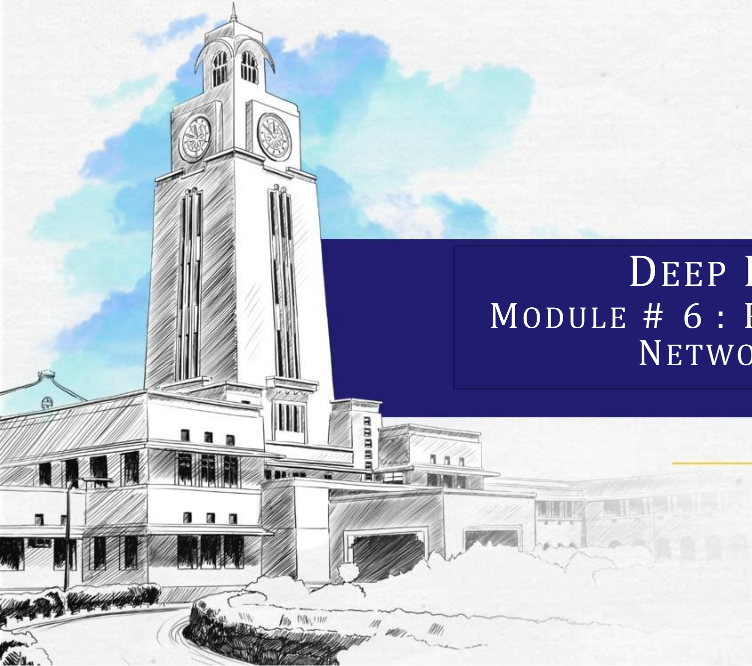


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Deep Learning

Module # 6 : Recurrent Neural Network [RNN]

In feedforward and convolutional neural networks

The size of the input is always fixed.

Each input to the network is independent of the previous or future inputs.

The computations, outputs and decisions for two successive inputs / images are completely independent of each other.

Sentiment Analysis ✓

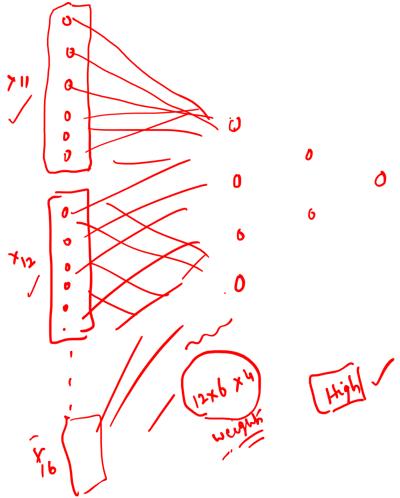
The Movie was boring and long — Bad

Movie is not good but songs are very good — Average.

ANN — classifier ✓

NB ✓

LR ✓



The Movie was boring and long

12 unique words

$t=1$ $x_{11} = [1\ 0\ 0\ 0\ 0\ 0\ 0]$

$t=2$ $x_{12} = [0\ 1\ 0\ 0\ 0\ 0\ 0]$

$t=3$ $x_{13} = [0\ 1\ 0\ 0\ 0\ 0]$

$t=4$ $x_{14} = [\ \ ]$

$x_{15} = [\ \ ]$

$x_{16} = [\ \ ]$

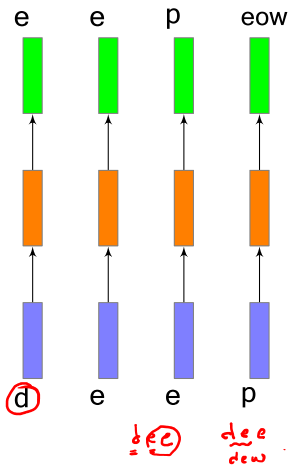
Long and was the

"sequence" $\xrightarrow{\text{borin}}$ ANN

This is not true in many applications.

- The size of the input is not always fixed.
- Successive inputs may not be independent of each other.
- Each network (blue - orange - green structure) is performing the same task –
  - input : character
  - output : character.

Example: Auto-completion.



Sequence Learning Problems

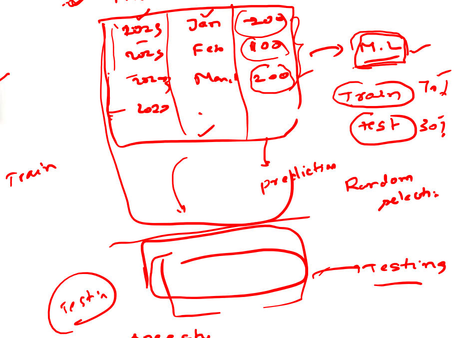
To model a sequence we need

- Process an input or sequence of inputs.
- The inputs may have be dependent.
- We may have to maintain the sequence order.
- Each input corresponds to one time step.
- Keep track of long term dependencies.
- Produce an output or sequence of outputs.
- Supervised Learning.
- Share parameters across the sequences.

Sequence:

sequential data

Time series data



- speech
- text
- audio, video

SEQUENCE MODEL

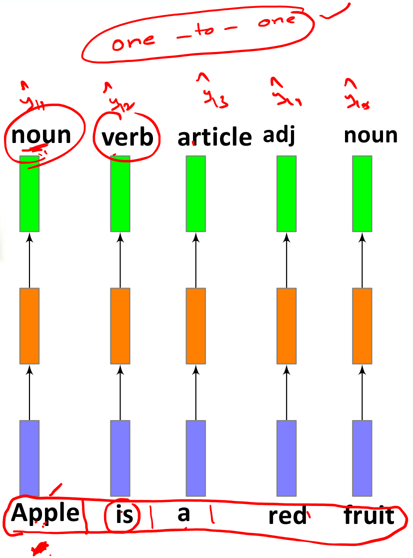
Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	∅	→	
Sentiment classification	"There is nothing to like in this movie."	→	
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

Alexa, Siri, Google translator

Courtesy: Andrew Ng

PART OF SPEECH TAGGING

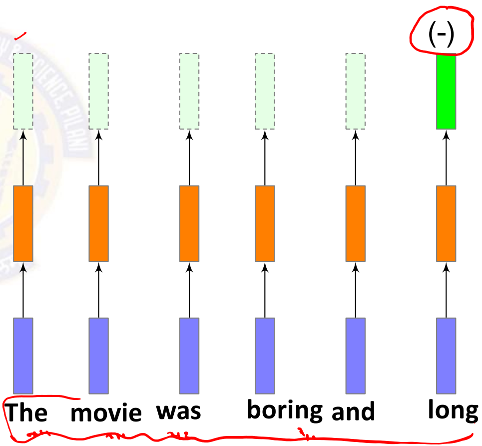
- Task is predicting the part of speech tag (noun, adverb, adjective, verb) ~~of~~ each word in a sentence.
- When we see an adjective we are almost sure, the next word should be a noun.
- The current output depends on the current input as well as the previous input.
- The size of the input is not fixed. Sentences have any number of words.
- An output is produced at end of each time step.
- Each network is performing the same task – input : word, output : tag.



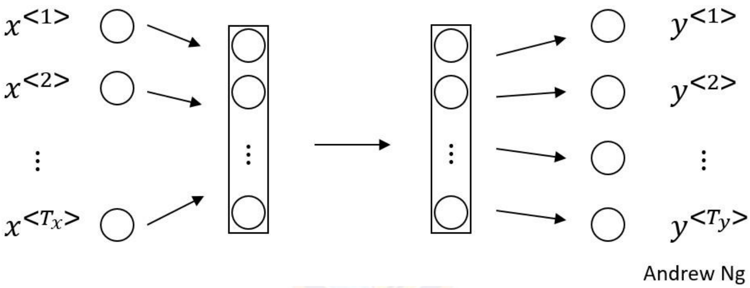
SENTIMENT ANALYSIS

- Task is predicting the sentiment of a whole sentence.
- Input is the entire sequence of inputs.
- An output is **not** produced at end of each time step.
- Each network is performing the same task –
  - input : word,
  - output : polarity +/–.

Many to one



RECURRENT NEURAL NETWORK (RNN)



- Accounts for variable number of inputs.
- Accounts for dependencies between inputs.
- Accounts for variable number of outputs.
- Ensures that the same function executed at each time step.
- The features learned across the inputs at different time step has to be shared.

x_{11} x_{12} x_{13} x_{14} x_{16} → ANN

Movie Won Good but no.

RNN

Ravi

Loss = Loss1 + Loss2 + Loss3

$t=1$

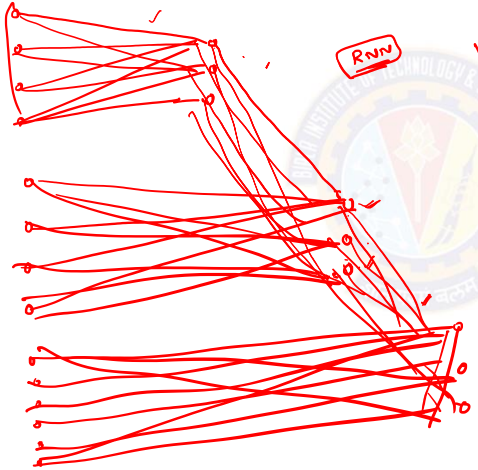
x_{11}

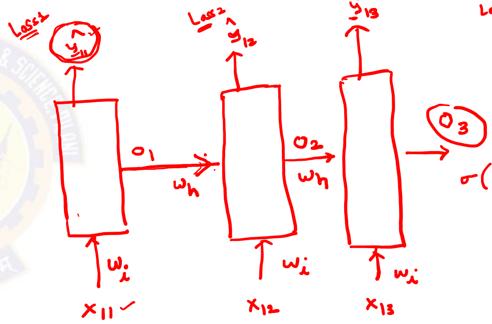
$t=2$

x_{12}

$t=3$

x_{13}





$\sigma(O_3) = \hat{y}$

$f(x_{11} w_i) = O_1$

$f(x_{12} w_i + O_1 w_h) = O_2$

$f(x_{13} w_i + O_2 w_h) = O_3$

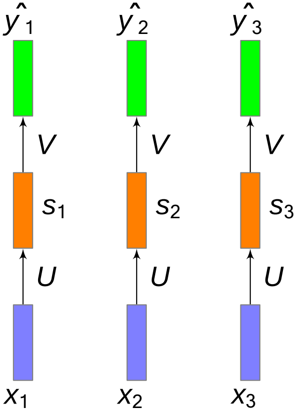
$\sigma(O_3)$ = output

RNN I

- The function learned at each time step.

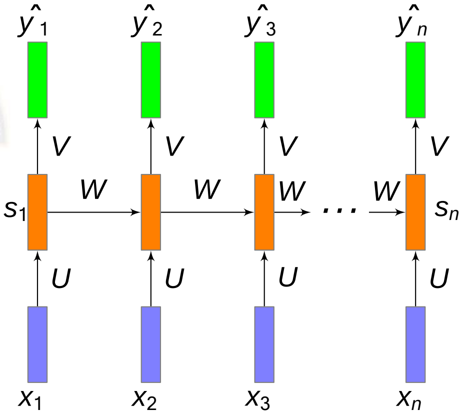
$$t = \text{time step}$$
$$x_t = \text{input at time step } t$$
$$s_t = \sigma(Ux_t + b)$$
$$y_t = g(Vs_t + c)$$

- Since the same function has to be executed at each time step we should share the same network i.e., same parameters at each time step.

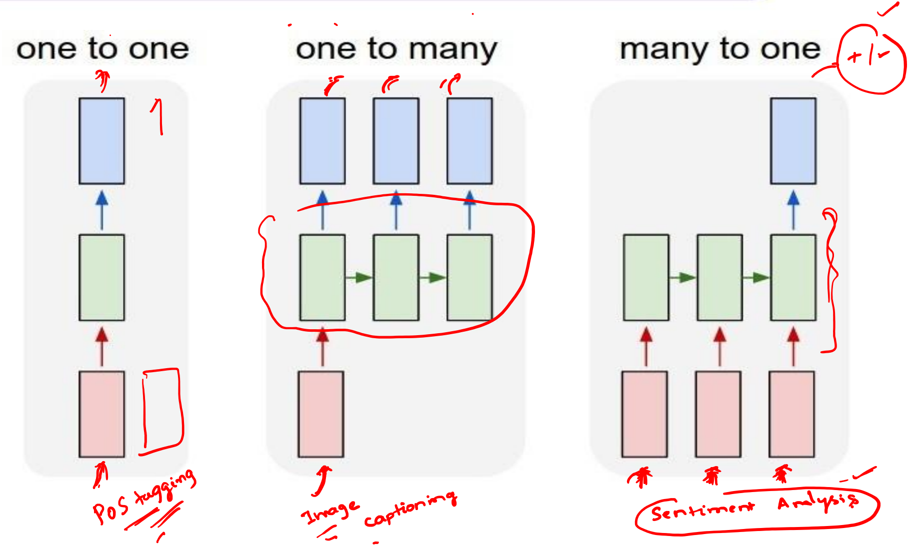


RNN II

- The parameter sharing ensures that
  - the network becomes invariant to the length of the input.
  - the number of time steps doesn't matter.
- Create multiple copies of the network and execute them at each timestep.
  - i.e. create a loop effect.
  - i.e. add recurrent connection in the network.

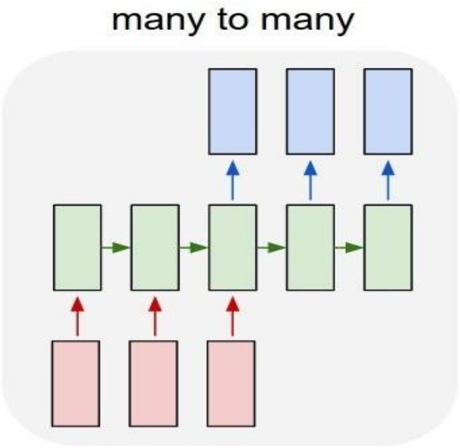


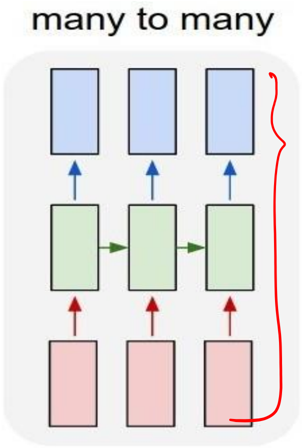
TYPES OF RNN



Courtesy: Andrej Karpathy

TYPES OF RNN





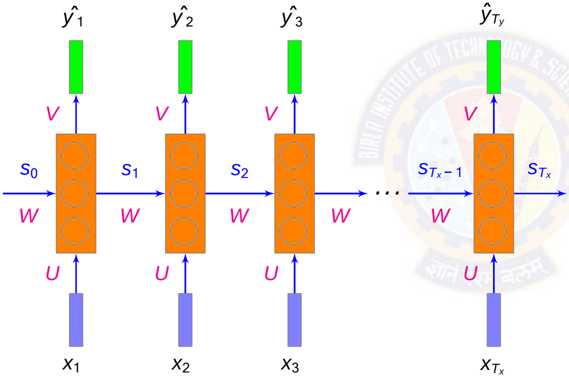
machine translation.

Courtesy: Andrej Karpathy

TYPES OF RNN AND APPLICATIONS

- **One to one –** Generic neural network, Image classification
- **One to many –** Music generation, Image Captioning
- **Many to one –** Movie review or Sentiment Analysis
- **Many to many –** Machine translation Synced
- **Many to many –** Video classification

Forward Propagation in RNN



- s_t is the **state** of the network at time step t.

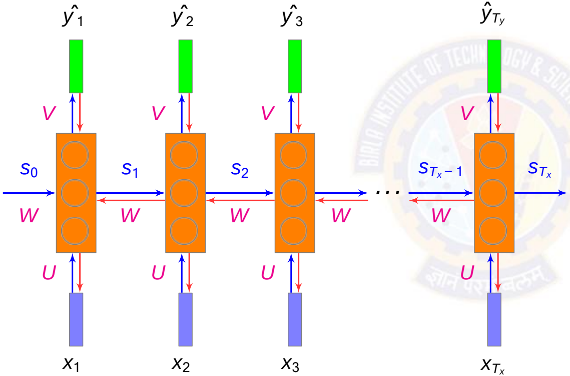
$$s_0 = 0$$
$$s_t = \sigma(Ux_t + Ws_{t-1} + b)$$
$$\hat{y}_t = g(Vs_t + c)$$

or

$$\hat{y}_t = f(x_t, s_{t-1}, W, U, V, b, c)$$

- The parameters W, U, V, b, c are shared across time steps.

BACK PROPAGATION IN RNN



Loss function

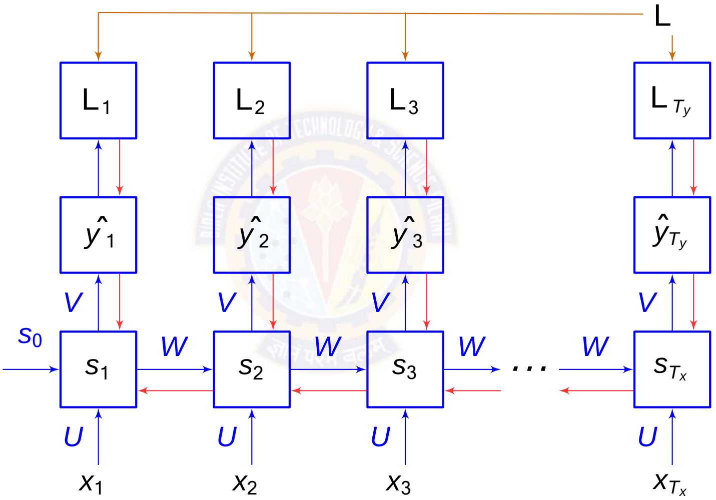
$$L_t(\hat{y}_t, y_t) = \prod_{t=1}^{T_y} P(\hat{y}_t \mid \hat{y}_{t-1}, \ldots, \hat{y}_1)$$

Overall Loss

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L_t(\hat{y}_t, y_t)$$

LSTM

BACK PROPAGATION IN RNN



Back-propagation through time.

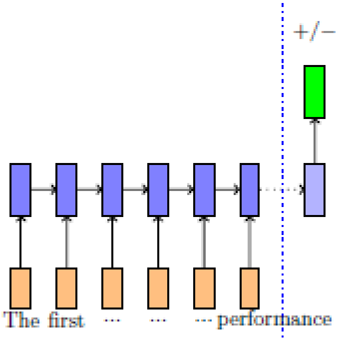
Issue of Maintaining States

- The old information gets morphed by the current input at each new time step.
- After t steps the information stored at time step $t - k$ (for some $k < t$) gets completely morphed so much that it would be impossible to extract the original information stored at time step $t - k$.
- It is very hard to assign the responsibility of the error caused at time step t to the events that occurred at time step $t - k$.
- Basically depends on the size of memory that is available.

Strategy to Maintain States

- Selectively write on the states.
- Selectively read the already written content.
- Selectively forget (erase) some content.

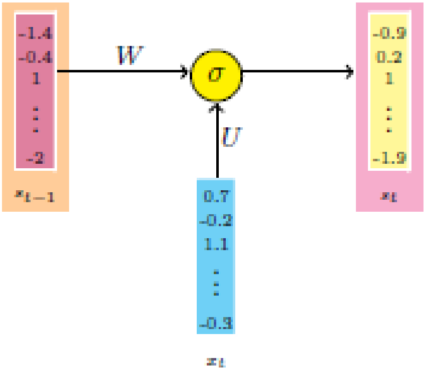
SENTIMENT ANALYSIS



Review: The first half of the movie was dry but the second half really picked up pace. The lead actor delivered an amazing performance

- RNN reads the document from left to right and after every word updates the state.
- By the time we reach the end of the document the information obtained from the first few words is completely lost.
- Ideally we want to
  - forget the information added by stop words (a, the, etc.).
  - selectively read the information added by previous sentiment bearing words (awesome, amazing, etc.)
  - selectively write new information from the current word to the state.

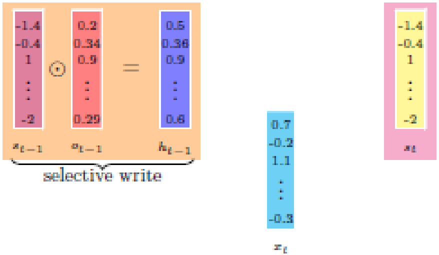
SELECTIVE WRITE



Recall that in RNNs we use s_{t-1} to compute s_t.

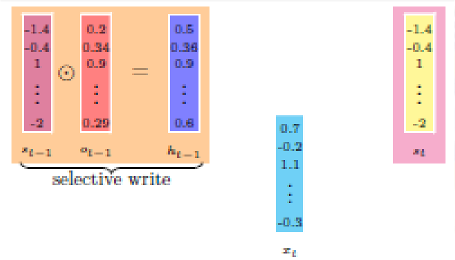
$$s_t = \sigma(Ws_{t-1} + Ux_t + b)$$

SELECTIVE WRITE



- Introduce a vector o_{t-1} which decides what fraction of each element of s_{t-1} should be passed to the next state.
- Each element of o_{t-1} gets multiplied with the corresponding element of s_{t-1}.
- Each element of o_{t-1} is restricted to be between 0 and 1.
- The RNN has to learn o_{t-1} along with the other parameters (W, U, V).

SELECTIVE WRITE

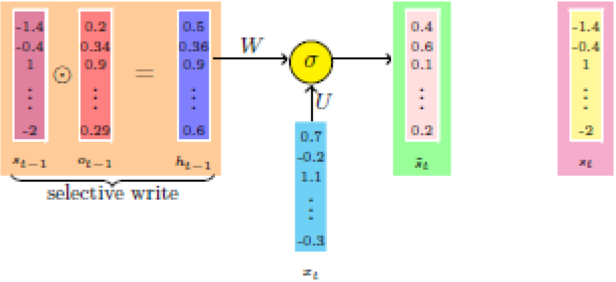


- Compute o_{t-1} and h_{t-1} as

  $$o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$$
  $$h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$$

- The parameters (W_o, U_o, b_o) are learned along with the existing parameters (W,U,V).
- The sigmoid function ensures that the values are between 0 and 1.
- o_t is called the **output gate** as it decides how much to pass (write) to the next time step.

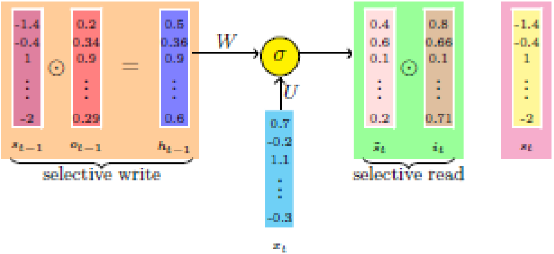
Compute State



- h_{t-1} and x_t are used to compute the new state at the next time step.

$$\tilde{s}_t = \sigma(Wh_{t-1} + Ux_t + b)$$

SELECTIVE READ

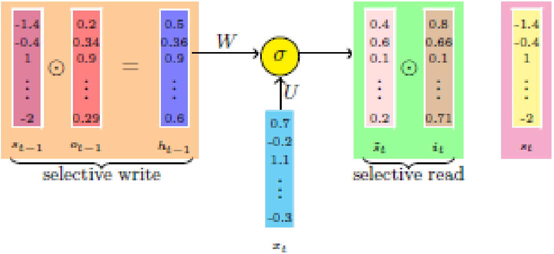


- s_t captures all the information from the previous state h_{t-1} and the current input x_t.
- To do selective read, introduce another gate called the **input gate**.

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

$$\text{Selectively Read} = i_t \odot s_t$$

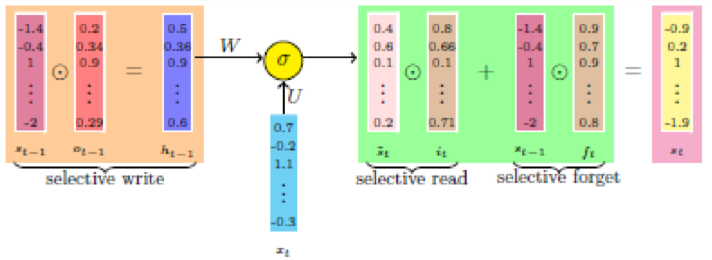
Selective Read



- $\tilde{s}_t$ captures all the information from the previous state h_{t-1} and the current input x_t.
- To do selective read, introduce another gate called the **input gate**.

$$s_t = s_{t-1} + i_t \odot \tilde{s}_t$$

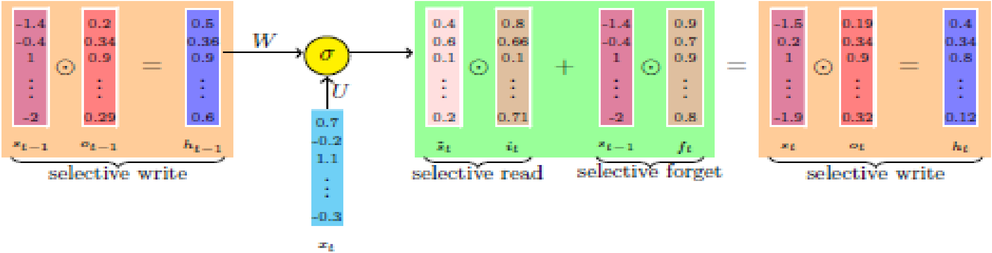
SELECTIVE FORGET



- To do selective forget, introduce another gate called the **forget gate**.

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$

Full LSTM



- 3 gates

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

- 3 states

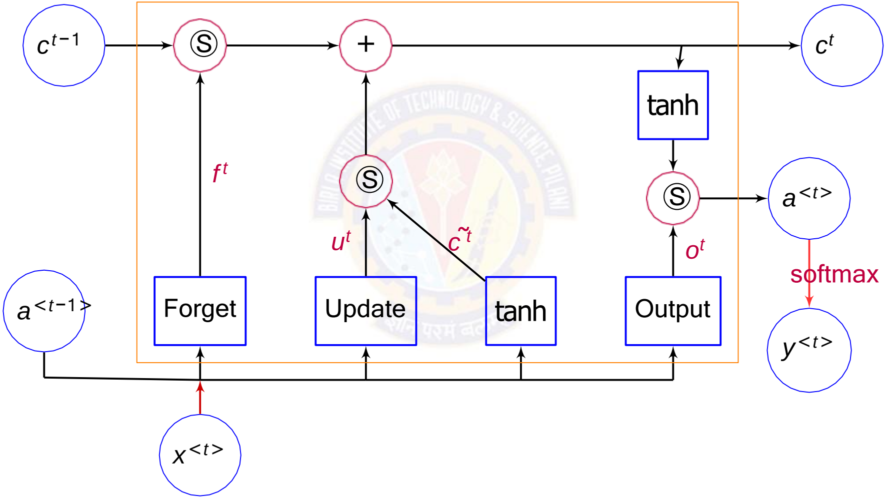
$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$
$$s_t = f_t \circledS s_{t-1} + i_t \circledS \tilde{s}_t$$
$$h_t = o_t \circledS \sigma(s_t)$$
$$\hat{y}_t = g(V s_t + c)$$

Long Short Term Memory Unit (LSTM)

- Another representation
- 3 gates are used – Update gate Γ_u, Forget gate Γ_f and Output gate Γ_o.

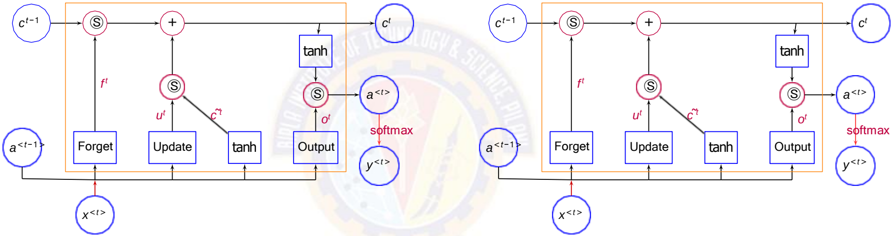
$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$
$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>} + b_u])$$
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>} + b_f])$$
$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>} + b_o])$$
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$
$$a^{<t>} = \Gamma_o * \tanh(c^{<t>})$$

LSTM



c^{t-1}
S
+
c^t
tanh
f^t
S
S
$a^{<t>}$
u^t
$\tilde{c}^t$
o^t
softmax
$a^{<t-1>}$
Forget
Update
tanh
Output
$y^{<t>}$
$x^{<t>}$

LSTM



c^{t-1}
Ⓢ
+
c^t
tanh
f^t
Ⓢ
$a^{<t>}$
u^t
$\tilde{c}^t$
o^t
softmax
$a^{<t-1>}$
Forget
Update
tanh
Output
$y^{<t>}$
$x^{<t>}$
c^{t-1}
Ⓢ
+
c^t
tanh
f^t
Ⓢ
$a^{<t>}$
u^t
$\tilde{c}^t$
o^t
softmax
$a^{<t-1>}$
Forget
Update
tanh
Output
$y^{<t>}$
$x^{<t>}$

Gated Recurrent Unit (GRU)

- Introduce a memory cell $c^{<t>} = a^{<t>}$
- Candidate for replacing $c^{<t>}$ is given $\tilde{c}^{<t>}$
- The decision whether to update $c^{<t>}$ with $\tilde{c}^{<t>}$ is given by the **update gate** Γ_u.
- Γ_u takes the value of 0 or 1.

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

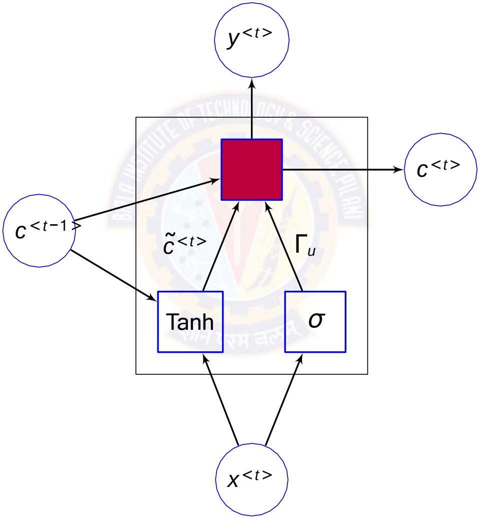
$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>} + b_u])$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>} + b_r])$$

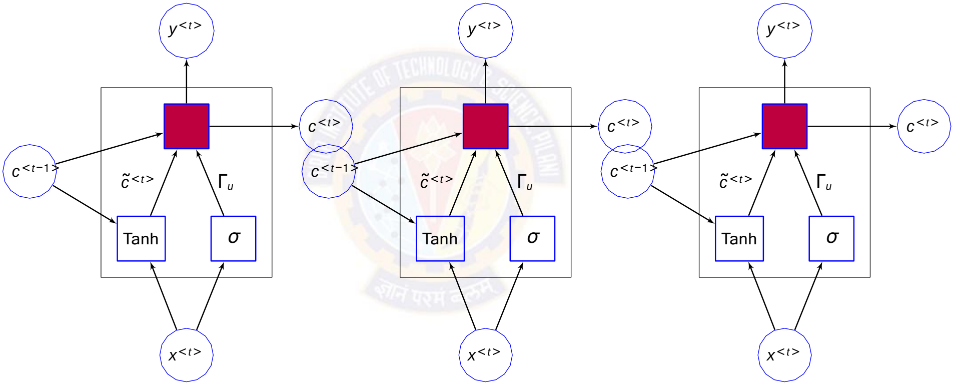
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

GATED RECURRENT UNIT (GRU)



Gated Recurrent Unit (GRU)



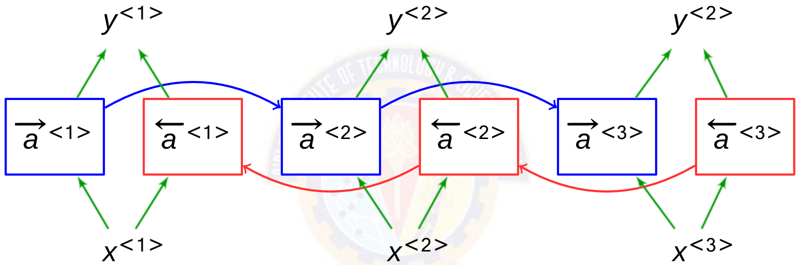
BIDIRECTIONAL RNN (BRNN)

- Forward and backward connections.
- The blocks can be RNN, GRU, LSTM.
- Mostly used in the NLP.
- Acyclic graph

Example: Name entity recognition He said "Teddy bear is soft."

He said "Teddy Roosevelt was a President."

BRNN ARCHITECTURE



$$\hat{y}^{<t>} = g(W_y[\overrightarrow{a}^{<t>} \overleftarrow{a}^{<t>} x^{<t>}] + b_y)$$

SUMMARY

- Use GRU, when dependency is short. Eg: Weather forecasting
- Use LSTM, when dependency is long. Eg: NLP Translation
- Use BRNN, dependency is in both direction. Eg: Stock prediction

References

1. Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville
   https://www.deeplearningbook.org/
2. Deep Learning with Python by Francois Chollet.
   https://livebook.manning.com/book/deep-learning-with-python/

Closed

Thank You!