Deep Learning
Module # 6 : Recurrent Neural Network [RNN]

Seetha Parameswaran
Asst Prof, BITS Pilani

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

The instructor is gratefully acknowledging the authors who made their course materials freely available online.

Issues of Feedforward Neural Networks

In feedforward and convolutional neural networks

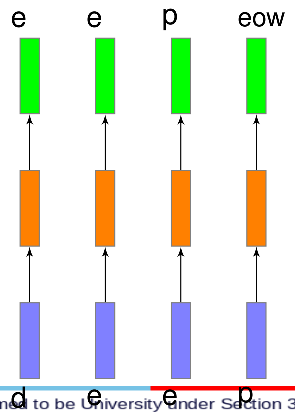
- The size of the input is always fixed.
- Each input to the network is independent of the previous or future inputs.
- The computations, outputs and decisions for two successive inputs / images are completely independent of each other.

Issues of Feedforward Neural Networks

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.



In This Segment

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.

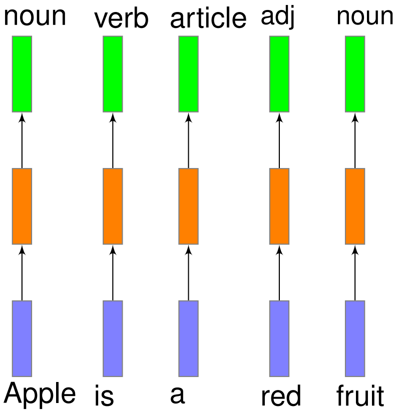
SEQUENCE MODEL

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.

Courtesy: Andrew Ng

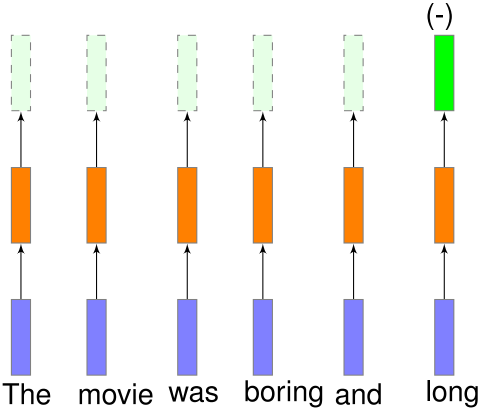
Part of Speech Tagging

- 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.



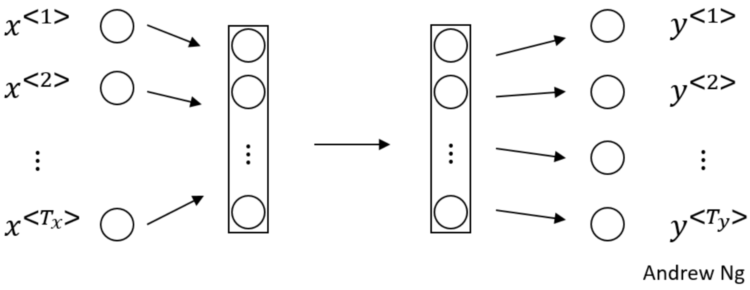
Sentiment Analysis

- 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 $+/-$.



In This Segment

Recurrent Neural Network (RNN)



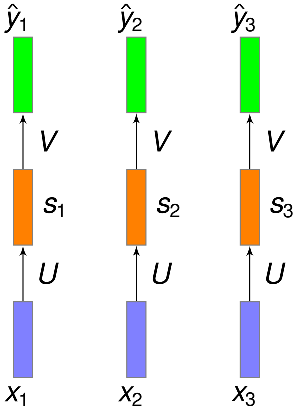
- Accounts for variable number of inputs.
- Accounts for dependencies between inputs.
- Accounts for variable number of outputs.
- Ensures that the same function executed at each time step.
- The features learned across the inputs at different time step has to be shared.

RNN I

- The function learned at each time step.

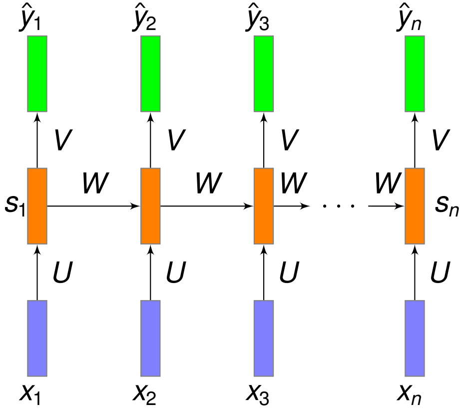
$$
\begin{aligned}
t &= \text{ time step} \\
x_t &= \text{ input at time step } t \\
s_t &= \sigma(Ux_t + b) \\
y_t &= g(Vs_t + c)
\end{aligned}
$$

- 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.



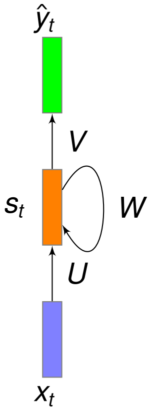
RNN II - Unrolled Network

- 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.

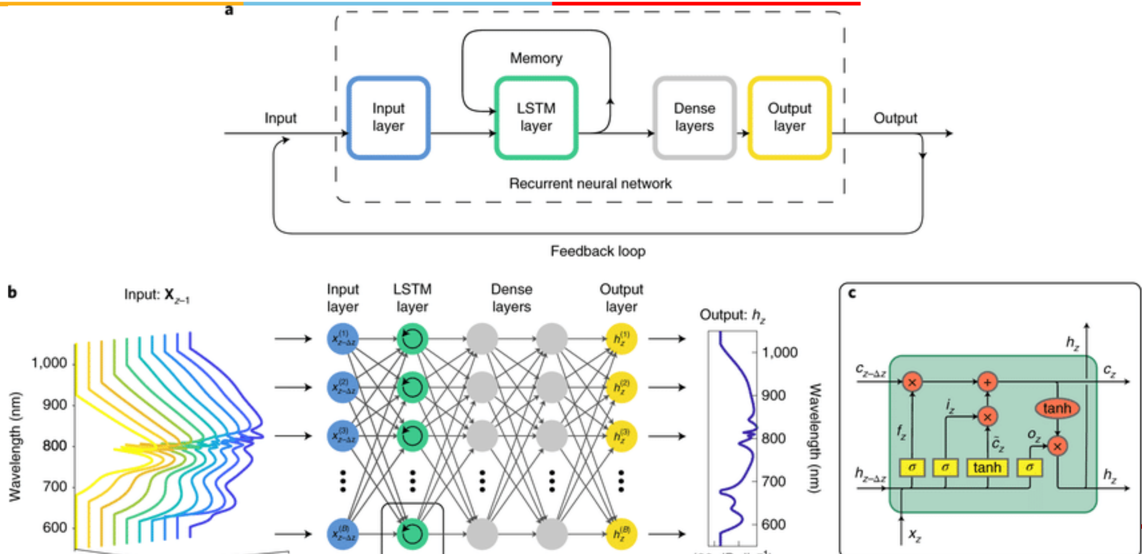


RNN III - Rolled Network

- Parameter sharing across time steps
- Self loop or recurrent connection

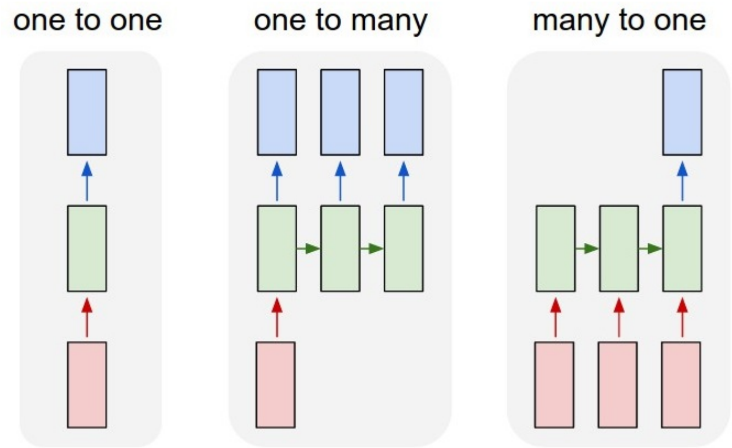


Schematic of RNN



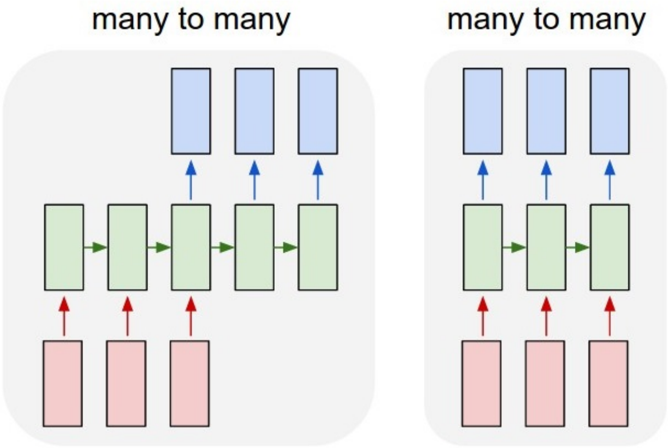
In This Segment

Types of RNN



Courtesy: Andrej Karpathy

Types of RNN



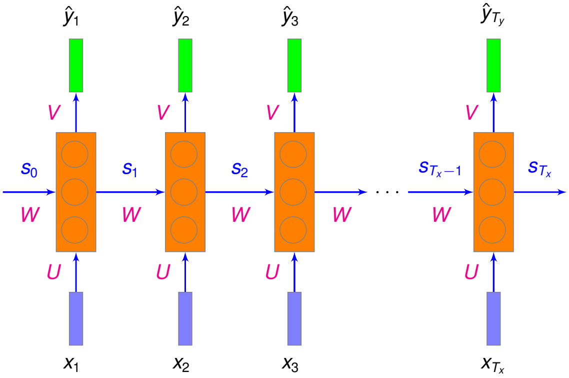
Courtesy: Andrej Karpathy

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

In This Segment

Forward Propagation in RNN



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

$$s_0 = 0$$

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

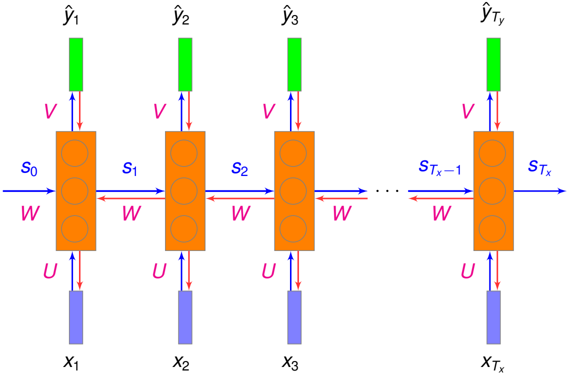
$$\hat{y}_t = g(Vs_t + c)$$

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



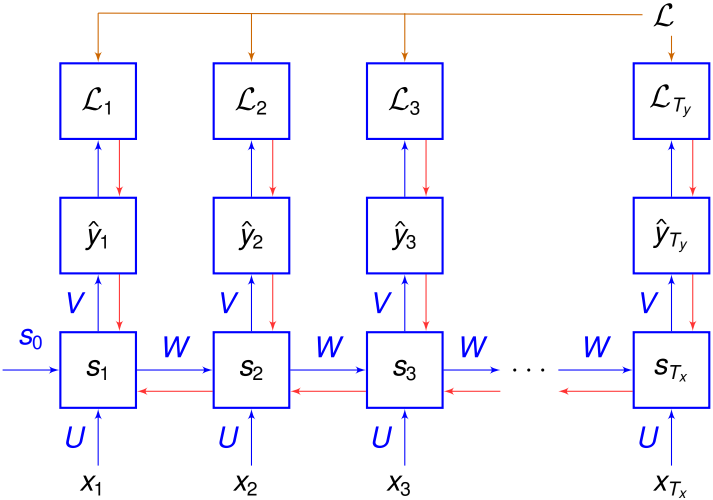
Loss function

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

Overall Loss

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

Back Propagation Through Time (BPTT) in RNN



In This Segment

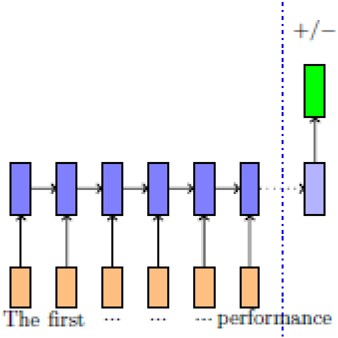
Issue of Maintaining States

- 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.

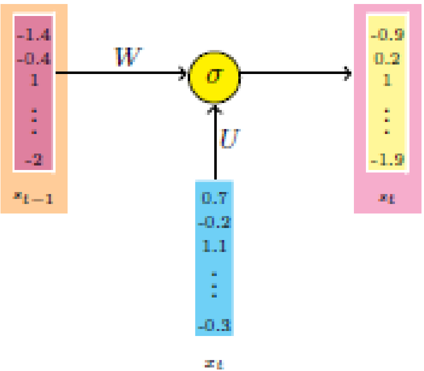
Sentiment Analysis



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.

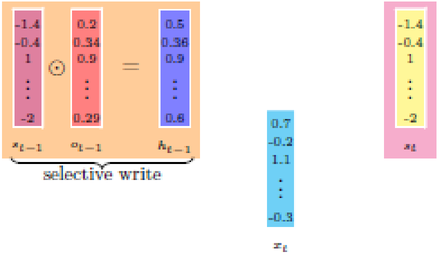
Selective Write



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

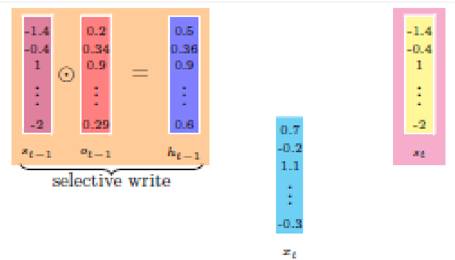
$$s_t = \sigma(Ws_{t-1} + Ux_t + b)$$

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

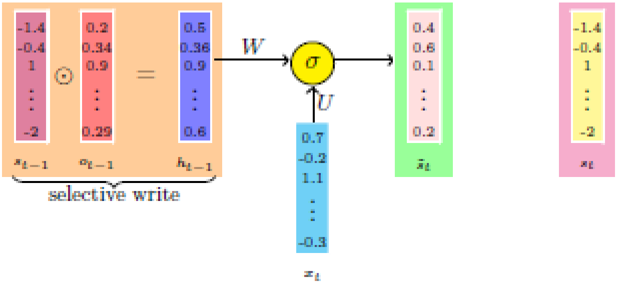


- Compute o_{t-1} and h_{t-1} as

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

- 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.

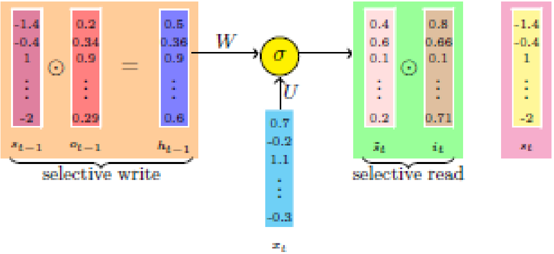
Compute State



- 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

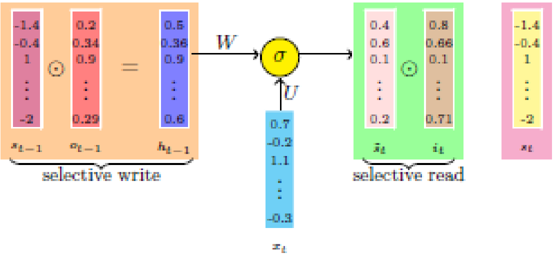


- $\tilde{s}_t$ captures all the information from the previous state h_{t-1} and the current input x_t.
- To do selective read, introduce another gate called the **input gate**.

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

$$\text{Selectively Read } = i_t \odot \tilde{s}_t$$

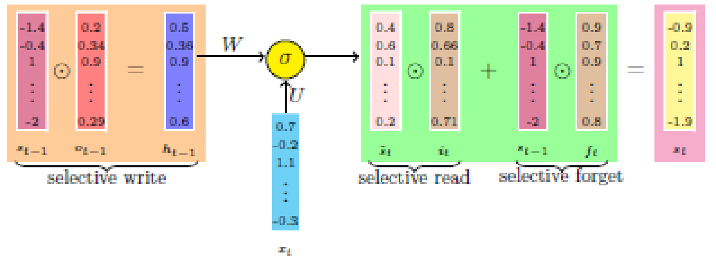
Selective Read



- $\tilde{s}_t$ 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**.

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

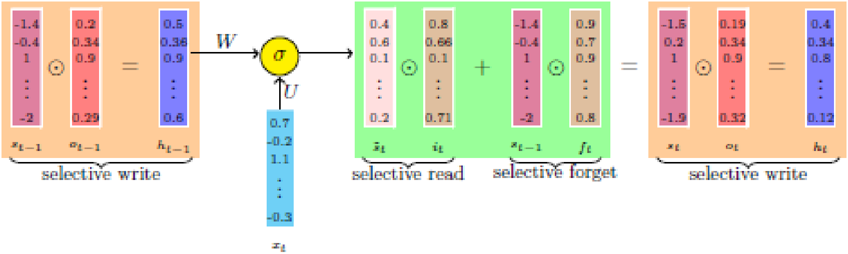
Selective Forget



- 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

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

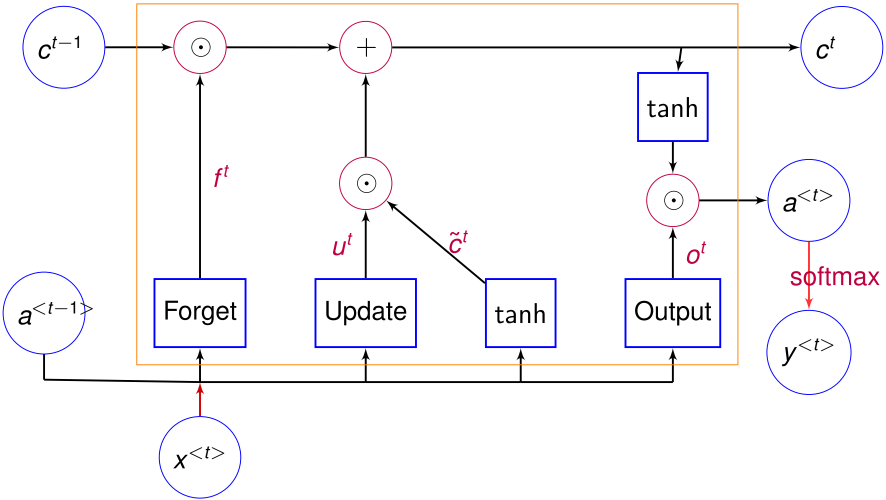
In This Segment

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.

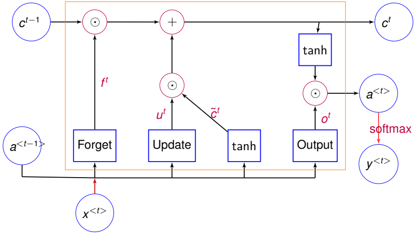
$$
\begin{aligned}
\tilde{c}^{<t>} &= \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \\
\Gamma_u &= \sigma(W_u[a^{<t-1>}, x^{<t>} + b_u]) \\
\Gamma_f &= \sigma(W_f[a^{<t-1>}, x^{<t>} + b_f]) \\
\Gamma_o &= \sigma(W_o[a^{<t-1>}, x^{<t>} + b_o]) \\
c^{<t>} &= \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \\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

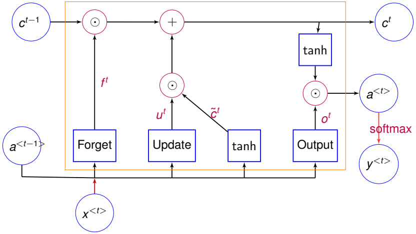
LSTM



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

LSTM


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


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

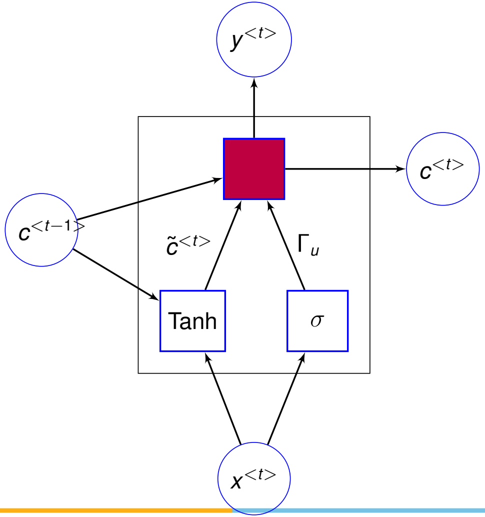
In This Segment

Gated Recurrent Unit (GRU)

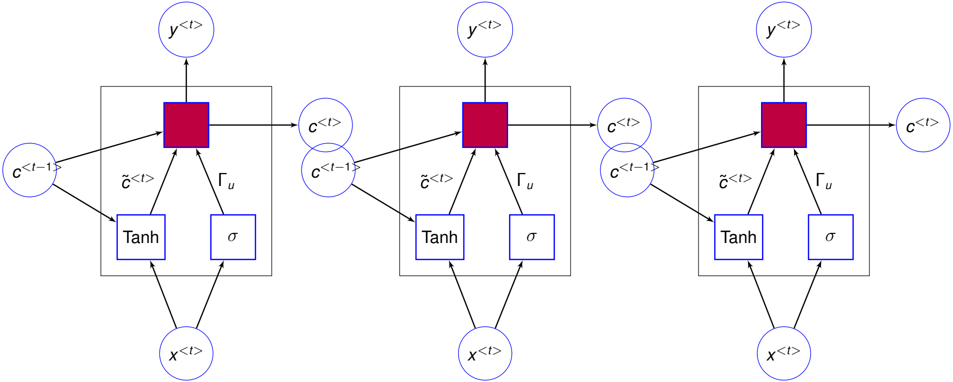
- Introduce a memory cell $c^{<t>} = a^{<t>}$.
- Candidate for replacing $c^{<t>}$ is given as $\tilde{c}^{<t>}$.
- The decision whether to update $c^{<t>}$ with $\tilde{c}^{<t>}$ is given by the **update gate** Γ_u. Γ_u takes the value of 0 or 1.

$$
\begin{aligned}
\tilde{c}^{<t>} &= \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c) \\
\Gamma_u &= \sigma(W_u[c^{<t-1>}, x^{<t>} + b_u]) \\
\Gamma_r &= \sigma(W_r[c^{<t-1>}, x^{<t>} + b_r]) \\
c^{<t>} &= \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>} \\
a^{<t>} &= c^{<t>}
\end{aligned}
$$

Gated Recurrent Unit (GRU)



Gated Recurrent Unit (GRU)

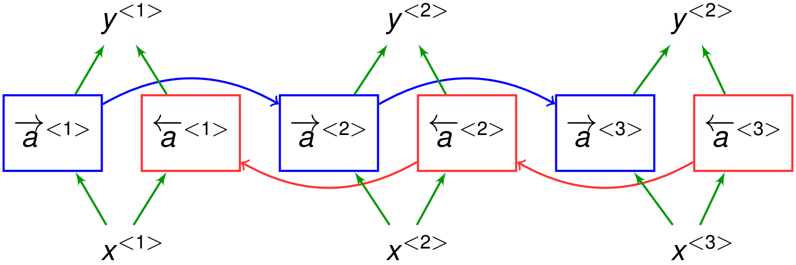


In This Segment

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. Dive into Deep Learning by Aston Zhang, Zack C. Lipton, Mu Li, Alex J. Smola
   https://d2l.ai/chapter_introduction/index.html
2. Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville
   https://www.deeplearningbook.org/

Thank You!