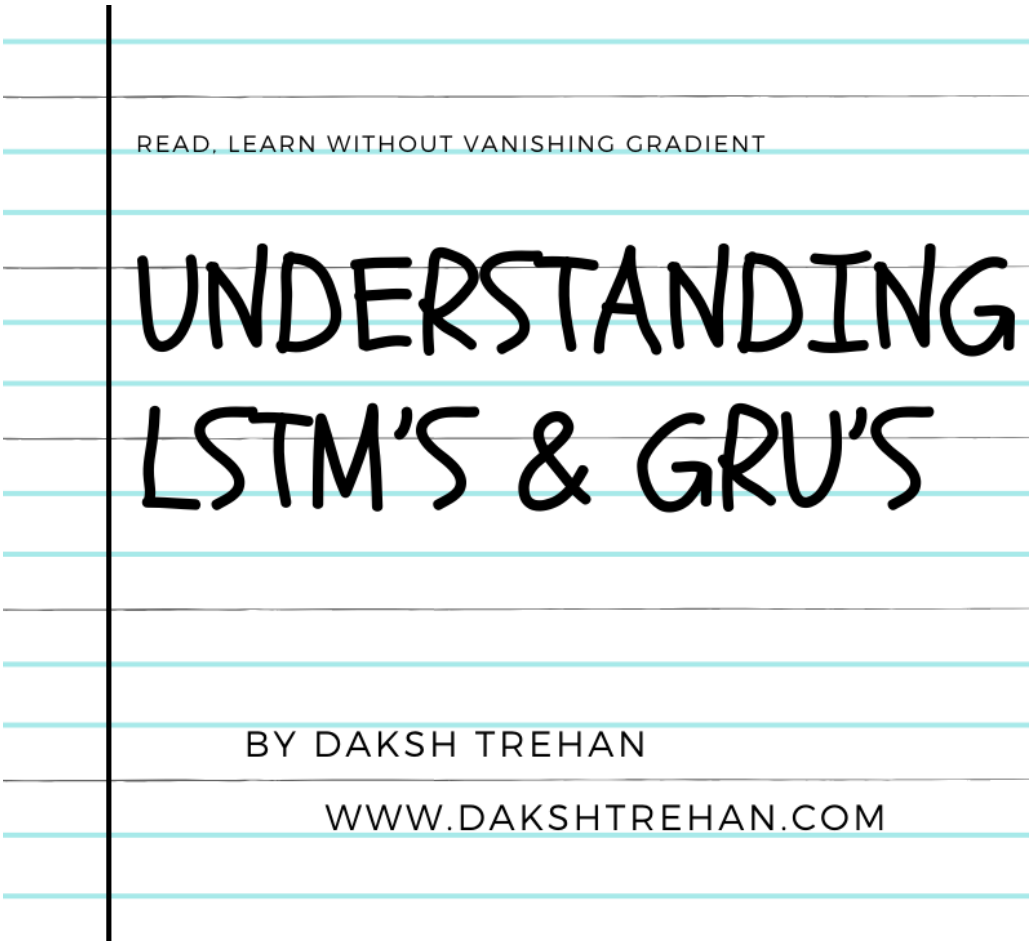
Deep Learning, Natural Language Processing

Understanding LSTM's and GRU's

A perfect guide to Long Short term Memory & Gated Recurrent Units.



In my last article, I have introduced Recurrent Neural Networks and the complications it carries. To combat the drawbacks we use LSTMs & GRUs.

Recurrent Neural Networks for Dummies
A perfect guide to Recurrent Neural Networks medium.com

The obstacle, Short-term Memory

Recurrent Neural Networks is confined to short-term memory. If a long sequence is fed to the network, they'll have a hard time remembering the information and might as well leave out important information from the beginning.

Besides, Recurrent Neural Networks faces Vanishing Gradient Problem when backpropagation comes into play. Due to the conflict, the updated gradients are much smaller leaving no change in our model and thus not contributing much in learning.

Weight Update Rule

“When we perform backpropagation, we calculate weights and biases for each node. But, if the improvements in the former layers are meager then the adjustment to the current layer will be much smaller. This causes gradients to dramatically diminish and thus leading to almost NULL changes in our model and due to that our model is no longer learning and no longer improving.”

Why LSTM’s and GRU’s?

Let us say, you’re looking at reviews for Schitt’s Creek online to determine if you could watch it or not. The basic approach will be to read the review and determine its sentiment.

★★★★★

Schitts Creek is Aaaashtonishing!
In today's TV world of uninspired shows, how on earth did this little Canadian sleeper take it to the full Monty?
I think their recipe was simple and timeless. Take a full cup of quirky yet interesting characters, "fold in" the shredded development of said characters with a riches, to rags to enrichment story line.
Then sprinkle some great situational comedic plot lines, being sure to stir it in carefully so as not to overmix and cause the formation of ridged stereotypes. Give it just a dash of drama to keep it real and sprinkle liberally with love, hope and care. Bake for exactly six seasons at the highest degree of incredible actor talents and remove when done.
When ready to serve, be sure to adorn with plenty of beautifully eccentric costumes, capricious vocabulary, and a purely touching moment or two.
And THAT folks is my take on the most enjoyment I've had with a TV show in a very long time - and that's no "pettyfogging".

139 people found this helpful.

When you look for the review, your subconscious mind will try to remember decisive keywords. You will try to remember more weighted words like “Aaaaastonishing”, “timeless”, “incredible”, “eccentric”, and “capricious” and will not focus on regular words like “think”, “exactly”, “most” etc.

The next time you’ll be asked to recall the review, you probably will be having a hard time, but, I bet you must remember the sentiment and few important and decisive words as mentioned above.

★★★★★

Schitts Creek is Aaaashtonishing!
In today's TV world of uninspired shows, how on earth did this little Canadian sleeper take it to the full Monty?
I think their recipe was simple and timeless. Take a full cup of quirky yet interesting characters, "fold in" the shredded development of said characters with a riches, to rags to enrichment story line.
Then sprinkle some great situational comedic plot lines, being sure to stir it in carefully so as not to overmix and cause the formation of ridged stereotypes. Give it just a dash of drama to keep it real and sprinkle liberally with love, hope and care. Bake for exactly six seasons at the highest degree of incredible actor talents and remove when done.
When ready to serve, be sure to adorn with plenty of beautifully eccentric costumes, capricious vocabulary, and a purely touching moment or two.
And THAT folks is my take on the most enjoyment I've had with a TV show in a very long time - and that's no "pettyfogging".

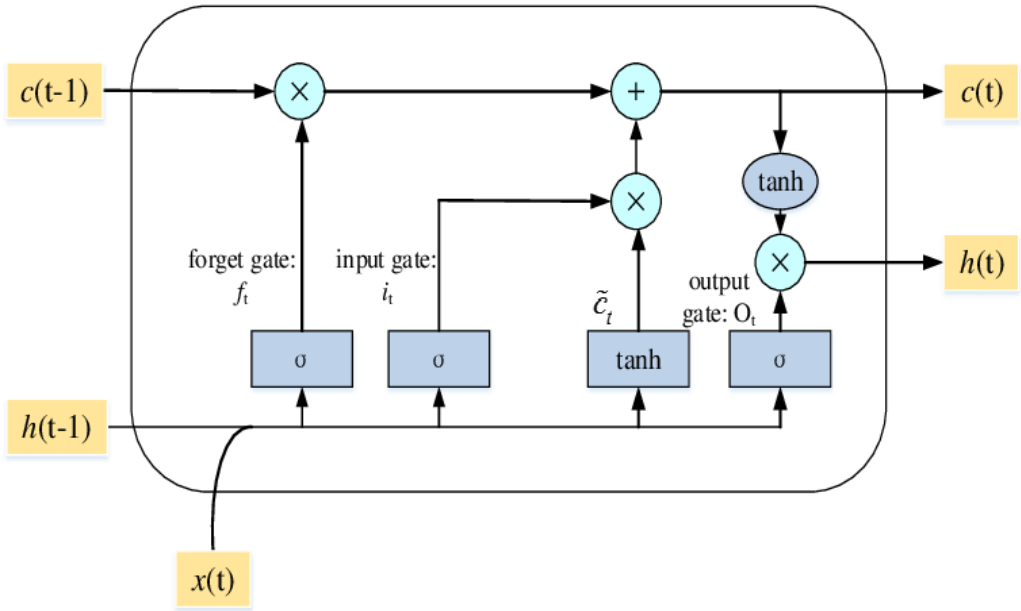
139 people found this helpful.

And that's what exactly LSTM and GRU are intended to operate.

> Learn and Remember only important information and forget every other stuff.

LSTM (Long short term memory)

LSTM's are a progressive form of vanilla RNN that were introduced to combat its shortcomings. To implement the above-mentioned intuition and administer the significant information due to finite-sized state vector in RNN we employ selectively read, write, and forget gates.



LSTM Cell, Source

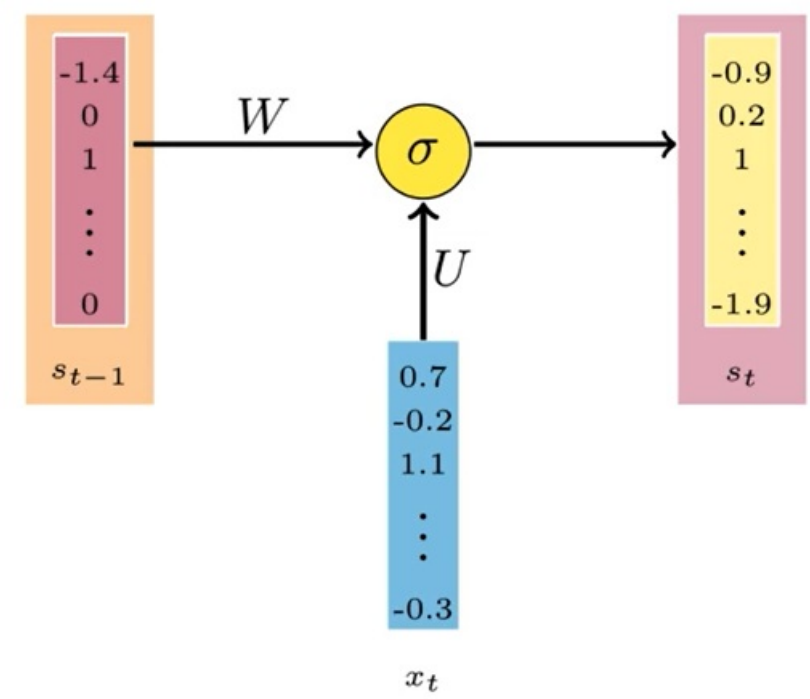
The abstract concept revolves around cell states and various gates. The cell state can transfer relative information to the sequence chain and is capable of carrying relevant information throughout the computation thus solving the problem of Short-term memory. As the process continues, the more relevant information is added and removed via gates. Gates are special types of neural networks that learn about relevant information during training.

Selective Write

Let us assume, the hidden state ($\mathbf{s_t}$), previous hidden state ($\mathbf{s_{t-1}}$), current input ($\mathbf{x_t}$), and bias ($\mathbf{b}$).

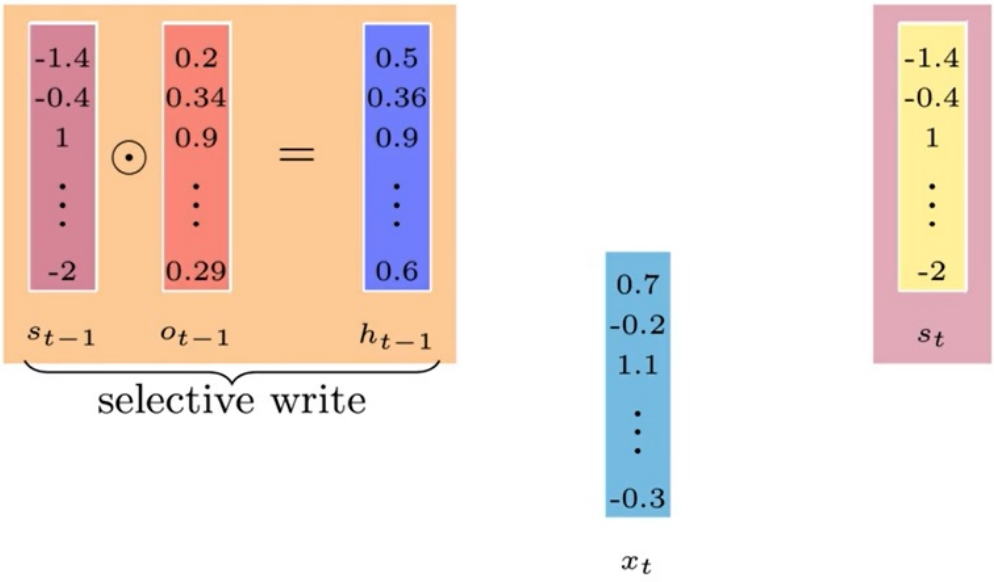
$$s_t = \sigma(Ux_t + W\mathbf{s_{t-1}} + b)$$

Now, we are accumulating all the outputs from previous state $\mathbf{s_{t-1}}$ and computing output for current state $\mathbf{s_t}$



Vanilla RNN

Using Selective Write, we are interested in only passing on the relevant information to the next state $\mathbf{s_t}$. To implement the strategy we could assign a value ranging from 0 to 1 to each input to determine how much information is to be passed on to the next hidden state.



Selective Write in LSTM

We can store the fraction of information to be passed on in a vector $\mathbf{h_{t-1}}$ **t**hat can be computed by multiplying previous state vector $\mathbf{s_{t-1}}$ and $\mathbf{o_{t-1}}$ that stores the value between 0 and 1 for each input.

The next issue we encounter is, how to get $\mathbf{o_{t-1}}$**?**

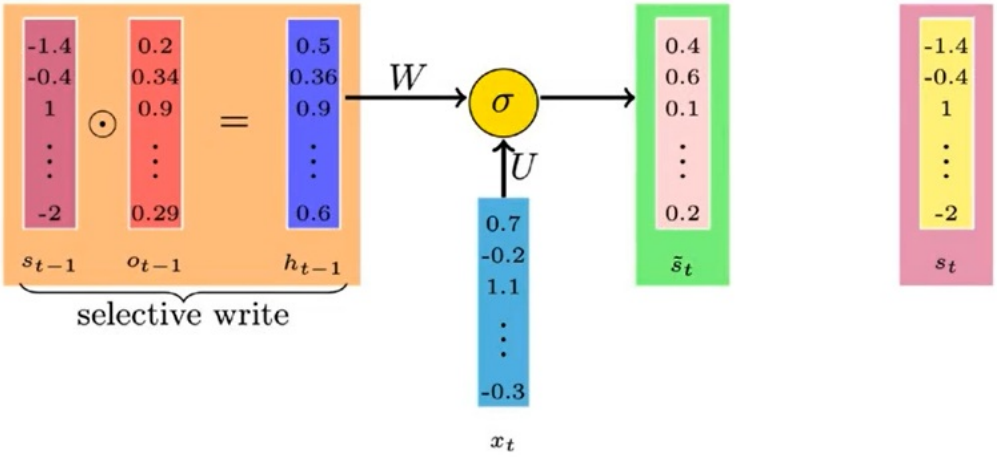
To compute $\mathbf{o_{t-1}}$ we must learn it and the only vectors that we have control on, are our parameters. So, to continue computation, we need to express $\mathbf{o_{t-1}}$ in form of parameters.

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

After learning **Uo, Wo** and **Bo** using Gradient Descent, we can expect a precise prediction using our output gate ($\mathbf{o_{t-1}}$) that is controlling how much information will be passed to the next gate.

Selective Read

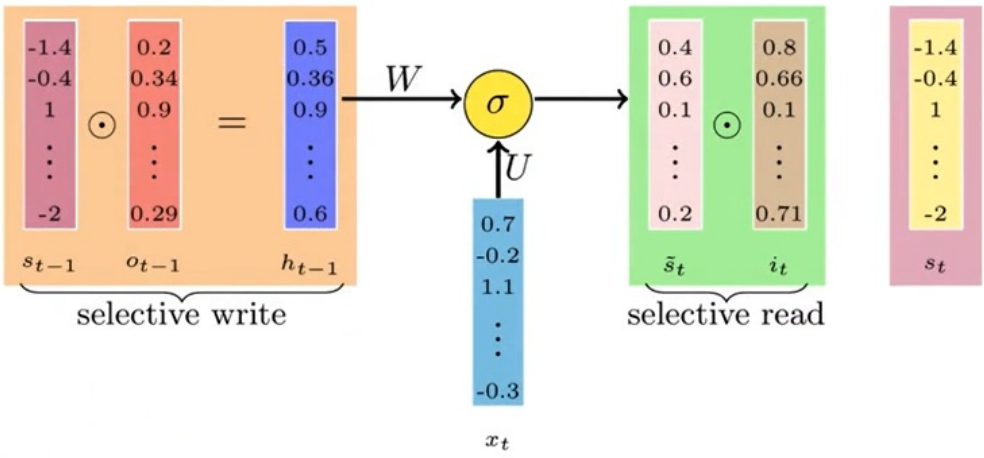
After passing the relevant information from previous gate, we introduce a new hidden state vector $\mathbf{\check{S}_t}$ (marked in green).



$\check{\mathbf{S}}_t$ captures all the information from the previous state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$.

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

But, our goal is to remove as much unimportant stuff as possible and to continue with our idea we will selectively read from the $\check{\mathbf{S}}_t$ to construct a new cell stage.



Selective Read

To store all important piece of content, we will again roll back to 0–1 strategy where we will assign a value between 0–1 to each input defining the proportion that we will like to read.

Vector $\mathbf{i}_t$ will store the proportional value for each input that will later be multiplied with $\check{\mathbf{S}}_t$ to control the information flowing through current input, which is called the **Input Gate.**

To compute $\mathbf{i}_t$ we must learn it and the only vectors that we have control on are our parameters. So, to continue computation, we need to express

$\mathbf{i_t}$ in form of parameters.

Input gate:

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

Selectively Read:

$$i_t \odot \tilde{s_t}$$

After learning **Ui, Wi** and **Bi** using Gradient Descent, we can expect a precise prediction using our input gate ($\mathbf{i_t}$) that is controlling how much information will be catered to our model.

Summing up the parameters that were learnt till now:

Previous state:

$$s_{t-1}$$

Output gate:

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

Selectively Write:

$$h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$$

Current (temporary) state:

$$\tilde{s_t} = \sigma(W h_{t-1} + U x_t + b)$$

Input gate:

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

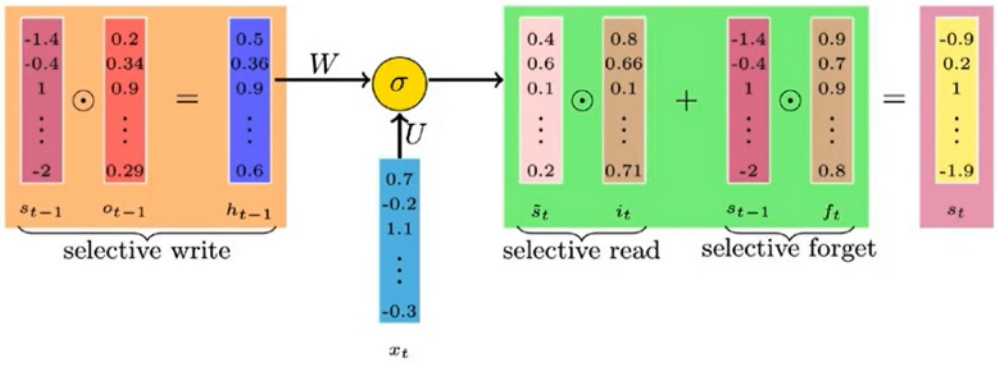
Selectively Read:

$$i_t \odot \tilde{s_t}$$

Selectively Forget

After selectively reading and writing the information, now we are aiming to forget all the irrelevant stuff that could help us to cut the clutter.

To discard all the squandered information from $\mathbf{s_{t-1}}$ we use Forget gate $\mathbf{f_t}$.



Selective Forget

Following the above-mentioned tradition, we will introduce forget gate **f_t** that will constitute a value ranging from 0 to 1 which will be used to determine the importance of each input.

To compute **f_t** we must learn it and the only vectors that we have control on are our parameters. So, to continue computation, we need to express **f_t** in form of provided parameters.

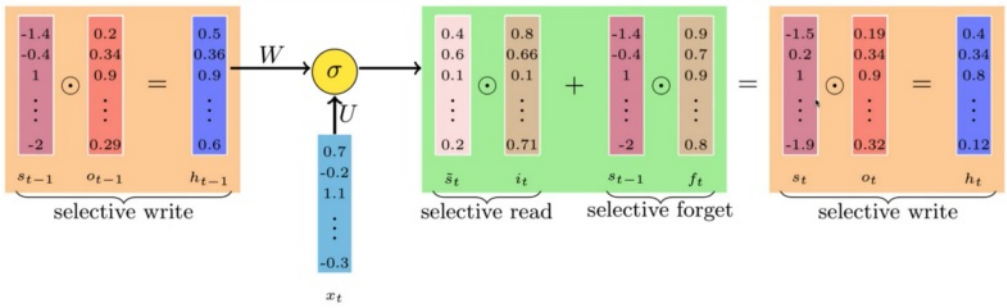
$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

After learning **Uf, Wf**, and **Bf** using Gradient Descent, we can expect a precise prediction using our forget gate (**f_t**) that is controlling how much information will be discarded.

Summing up information from forget gate and input gate will impart us about current hidden state information.

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

Final Model



LSTM model

The full set of equations looks like:

Gates:

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

States:

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

The parameters required in LSTM are way more than that required in vanilla RNN.

$$\begin{bmatrix} w & u & b \end{bmatrix} \qquad \begin{bmatrix} w_o & u_o & b_o \\ w_i & u_i & b_i \\ w_f & u_f & b_f \\ w & u & b \end{bmatrix}$$

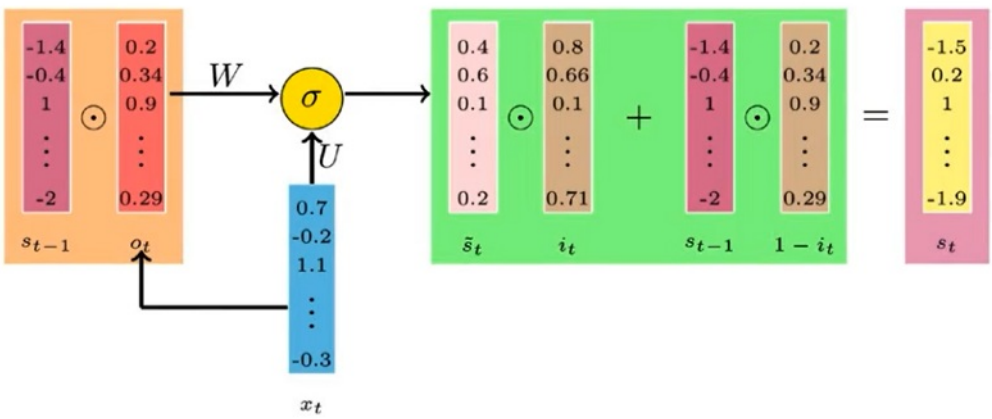
Parameters in RNN *Parameters in LSTM*

Due to the large variation of the number of gates and their arrangements, LSTM can have many types.

GRU(Gated Recurrent Units)

As mentioned earlier, LSTM can have many variations and GRU is one of them. Unlikely LSTM, GRU tries to implement fewer gates and thus helps to lower down the computational cost.

In Gated Recurrent Units, we have an output gate that controls the proportion of information that will be passed to the next hidden state, in addition, we have an input gate that controls information flow from current input and unlike RNN we don't use forget gates.



Gated Recurrent Units

To lower down the computational time we remove forget gate and to

discard the information we use compliment of input gate vector i.e. (1-i_t).

The equations implemented for GRU are:

Gates:

$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

States:

$$\tilde{s_t} = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$
$$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s_t}$$

Key Points

- LSTM & GRU are introduced to avoid short-term memory of RNN.
- LSTM forgets by using Forget Gates.
- LSTM remembers using Input Gates.
- LSTM keeps long-term memory using Cell State.
- GRUs are fast and computationally less expensive than LSTM.
- The gradients in LSTM can still vanish in case of forward propagation.
- LSTM doesn't solve the problem of exploding gradient, therefore we use gradient clipping.

Practical Use Cases

- Sentiment Analysis using RNN

dakshtrehan/IMDB-sentiment-analysis-NN
*Permalink Dismiss GitHub is home to over 50 million developers working together to host and review code, manage...*github.com

- AI music generation using LSTM

dakshtrehan/AI-Music-Generation
*Join me at www.dakshtrehan.com || www.linkedin.com/in/dakshtrehan The aim is to generate new music using LSTM and given...*github.com

---

Conclusion

Hopefully, this article will help you to understand about Long short-term memory(LSTM) and Gated Recurrent Units(GRU) in the best possible way and also assist you to its practical usage.

As always, thank you so much for reading, and please share this article if you found it useful!

Feel free to connect:

LinkedIn ~ https://www.linkedin.com/in/dakshtrehan/

Instagram ~ https://www.instagram.com/_daksh_trehan_/

Github ~ https://github.com/dakshtrehan

Follow for further Machine Learning/ Deep Learning blogs.

Medium ~ https://medium.com/@dakshtrehan

Want to learn more?

Detecting COVID-19 Using Deep Learning

The Inescapable AI Algorithm: TikTok

An insider's guide to Cartoonization using Machine Learning

Why are YOU responsible for George Floyd's Murder and Delhi Communal Riots?

Recurrent Neural Network for Dummies

Convolution Neural Network for Dummies

Diving Deep into Deep Learning

Why Choose Random Forest and Not Decision Trees

Clustering: What it is? When to use it?

Start off your ML Journey with k-Nearest Neighbors

Naive Bayes Explained

Activation Functions Explained

Parameter Optimization Explained

Gradient Descent Explained

Logistic Regression Explained

Linear Regression Explained

Determining Perfect Fit for your ML Model

Cheers!