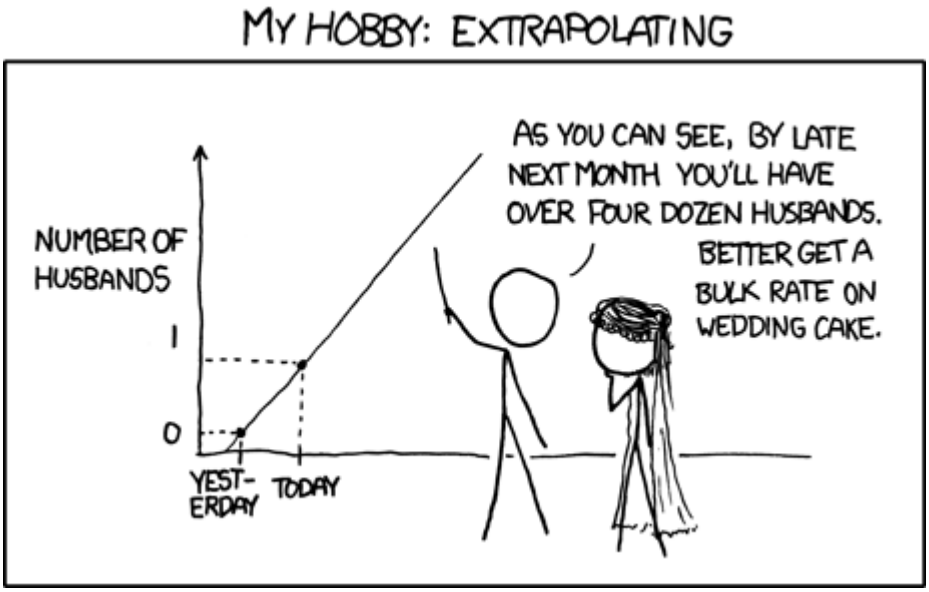
Forward and Backpropagation in GRUs — Derived | Deep Learning

Mihir Khandekar Oct 18, 2019 · 6 min read ★

An explanation of Gated Recurrent Units (GRUs) with the math behind how the loss backpropagates through time.

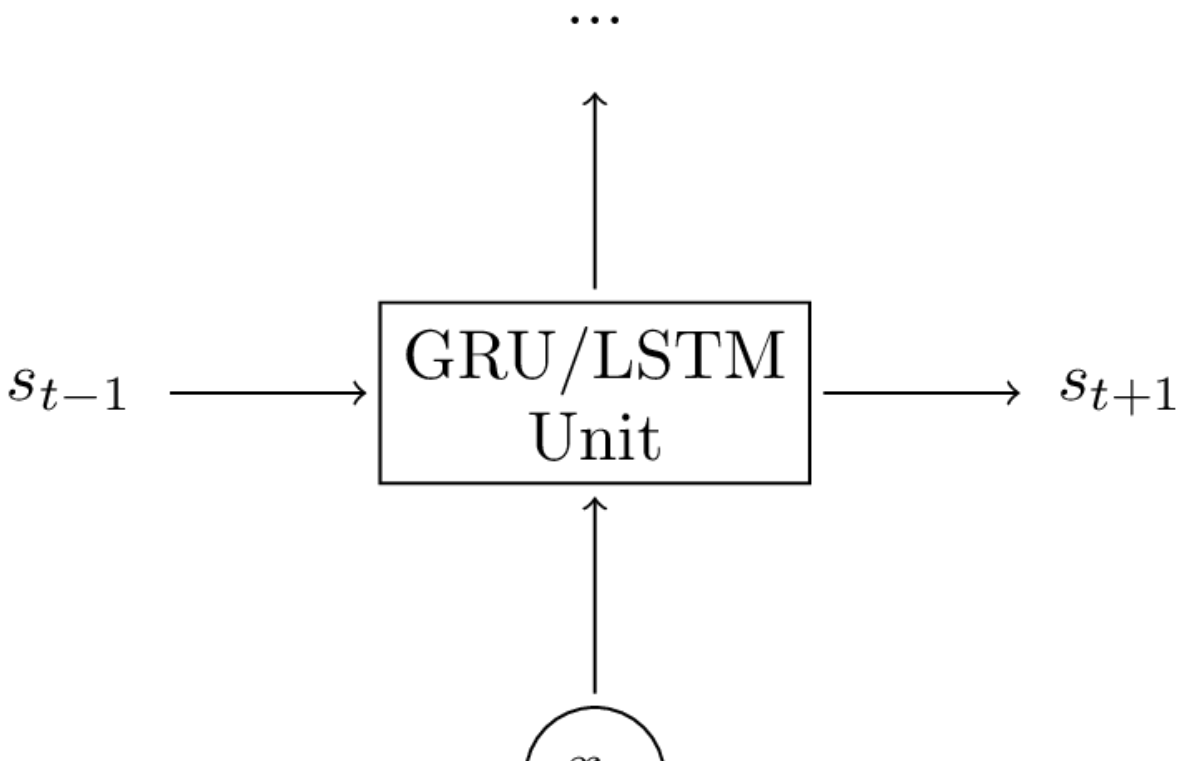


Source : xkcd

The **GRU Network, or Gated Recurrent Units** were initially proposed by Cho et. al, 2014 and is a very interesting type of recurrent neural network. It improved on the simple RNN by **reducing the vanishing gradient problem** by having reset, update gates. Additionally, the advantage of GRU over the newer LSTM (Long Short Term Memory) networks is that it is much simpler with fewer parameters to train. However, LSTMs are able to outperform GRUs with larger datasets and are able to remember information over longer periods. This is what makes the GRU a very interesting architecture to learn.

In this article, we first take a brief overview of GRU networks, following which we will do a detailed **mathematical derivation of the backpropagation** equations using a computation graph.

Architectures like GRUs, LSTMs are used mainly for predictions that need to be made by analyzing **data over time**. For example, for a weather prediction problem, it would not suffice to just see the current data, but we would have to train the model for data over a period of time so that it learns how previous weather conditions lead to future conditions. They are also used in **problems in Natural Language Processing** like Sentiment Analysis, Word Prediction, etc.

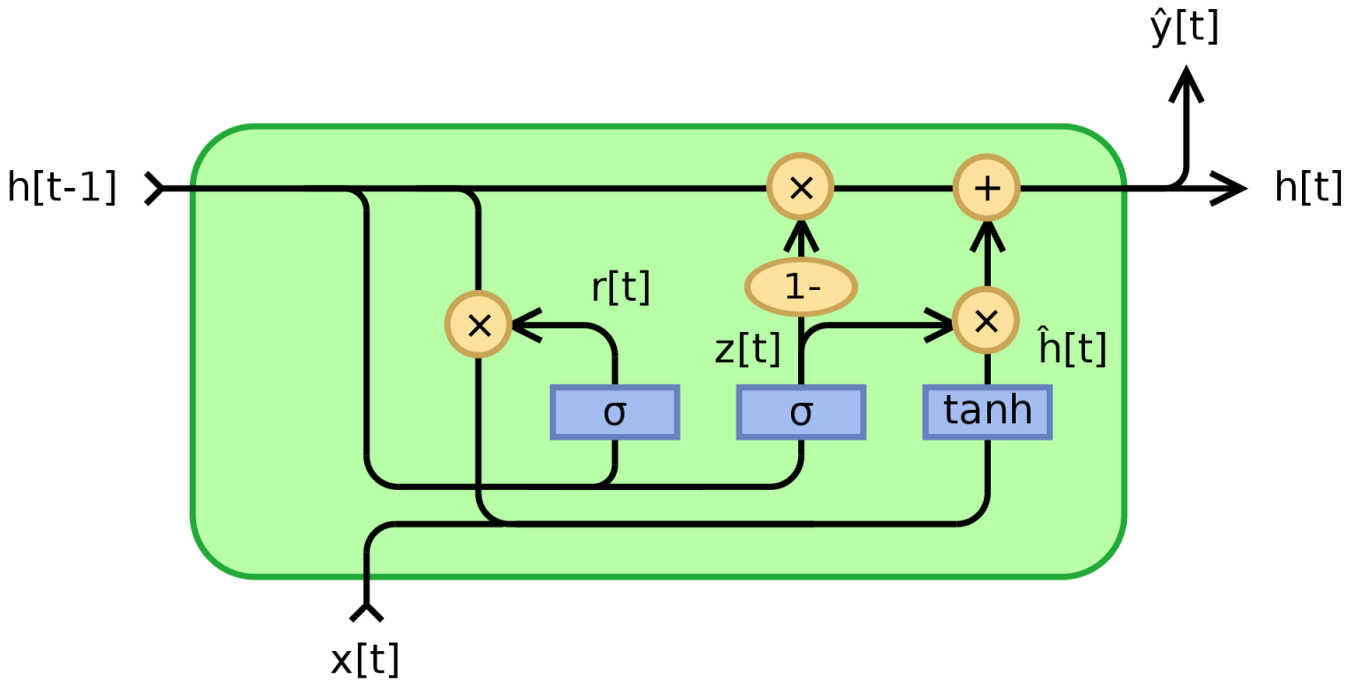


GRUs contain a state which is passed in layers across time, as well as an input given at each time. Source

For a brief overview of how these recurrent neural networks work, we can refer to a very helpful article by Christopher Olah. Additionally, a more detailed introduction to GRUs can be found on Simeon Kostadinov's article.

Overview of GRU

Gated Recurrent Units uses the update and reset gates to tackle the gradient vanishing problem faced in RNNs.



GRU Architecture. Source

In the above image, at each time t, we have the state **h** and the current time input **x.**
The **reset gate** learns which of the data from the input needs to be forgotten. For example, in the problem where we use time series weather data to predict the future weather, we might have some feature in the input like the population of the city, which the network might learn to be irrelevant to the weather prediction and "reset".
The **update gate** learns what data in the state to update with newer data from the input.

In the above example, there might be some data like the temperature, which the network might learn to update or modify the old one with the newer one in the input.

We now see how GRUs work mathematically.

GRU Equations

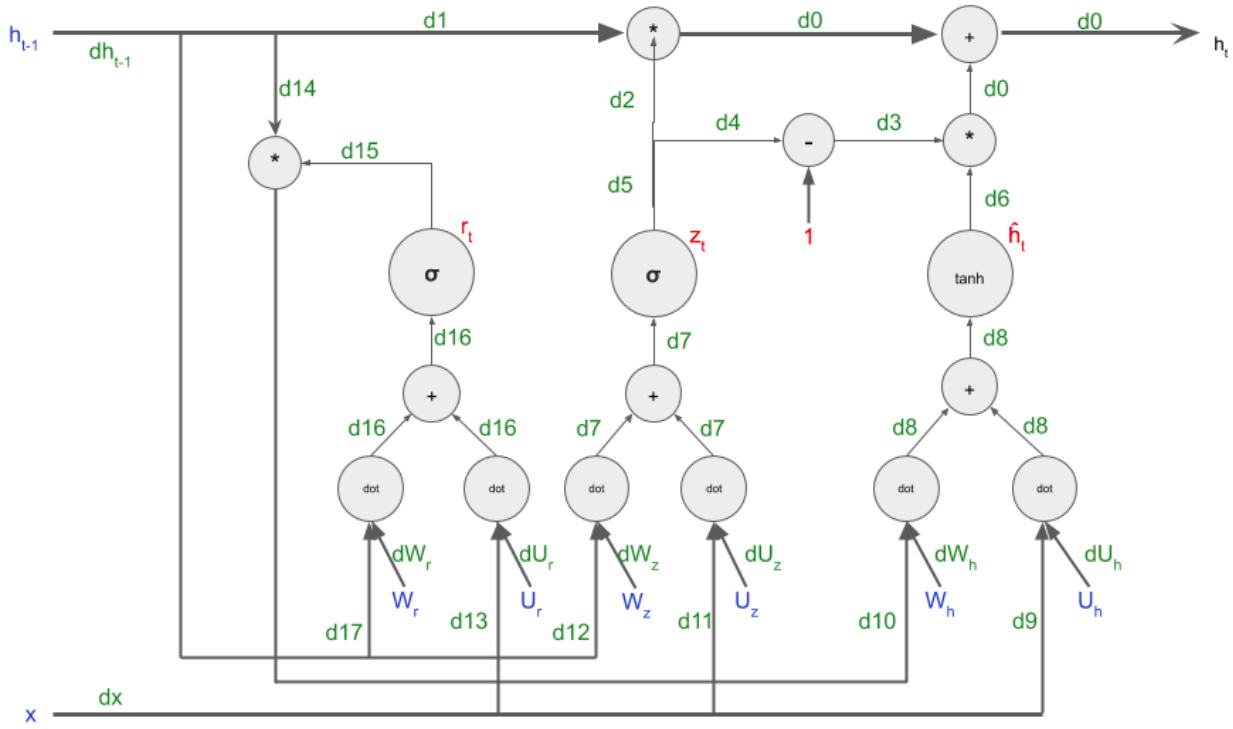
The GRU uses 4 main equations (Reset gate, Update gate, Current state and Layer Output)-

$$r_t = \sigma(W_r.h_{t-1} + U_r.x_t)$$

$$z_t = \sigma(W_z.h_{t-1} + U_z.x_t)$$

$$\tilde{h} = \tanh((r_t * h_{t-1}).W_h + U_h.x_t)$$

$$h_t = (1 - z_t) * \tilde{h}_t + z_t * h_{t-1}$$



GRU Computation Graph

The above image shows a computation graph for a single GRU layer. Here, the variables in blue are the inputs to the layer. We will find out the gradients to these layers in the backpropagation step. The variables in red are the intermediate variables that are calculated or used in the gates of the layer. The variables in green are the derivative of that step in the computation graph with respect to the loss. For example, if we have a gradient from the next layer coming as out_grad and we have the output from the layer, then d0 would refer to d(output)/d(0) or dWz would refer to d(output)/d(0).

We will now derive the equations for backpropagation. Here, d0 is the output gradient that the GRU layer receives from the next layer. Here, "*" refers to element-wise multiplication (Hadamard product), while "." refers to the dot product.

$$d0 = out\_grad$$

$$d1 = z_t * d0$$

$$d2 = h_{t-1} * d0$$

$$d3 = \tilde{h} * d0$$

$$d4 = -1 * d3$$

$$d5 = d2 + d4$$

$$d6 = (1 - z_t) * d0$$

$$d7 = d5 * (z_t * (1 - z_t))$$

$$d8 = d6 * (1 - \tilde{h}^2)$$

$$d9 = d8.U_h^T$$

$$d10 = d8.W_h^T$$

$$d11 = d7.U_z^T$$

$$d12 = d7.W_z^T$$

$$d14 = d10 * r_t$$

$$d15 = d10 * h_{t-1}$$

$$d16 = d15 * (r_t * (1 - r_t)))$$

$$d13 = d16.U_r^T$$

$$d17 = d16.W_r^T$$

And here, we have derived the computation graph. Now we look at the gradients to the inputs.

$$dx = d9 + d11 + d13$$

$$dh_{t-1} = d12 + d14 + d1 + d17$$

$$dU_r = x^T.(d16)$$

$$dU_z = x^T.(d7)$$

$$dU_h = x^T.(d8)$$

$$dW_r = h_{t-1}^T.(d16)$$

$$dW_z = h_{t-1}^T.(d7)$$

$$dW_{\quad} = (h_{\quad})^T.(d8)$$

$$dW_h = (h_{t-1} * r_t)^T . (d\delta)$$

That was quite simple. Though the equations of GRUs might seem intimidating, we can derive the backpropagation algorithms step-by-step easily.

These equations can be implemented in code to do the GRU backpropagation. Ensure that the dimensions remain consistent with the expected ones.

References:

[1] Christopher Olah, Understanding LSTM Networks (2015)

[2] Simeon Kostadinov, Understanding GRU Networks (2017), Towards Data Science

[3] Dimitri Fichou, GRU Units (2019)

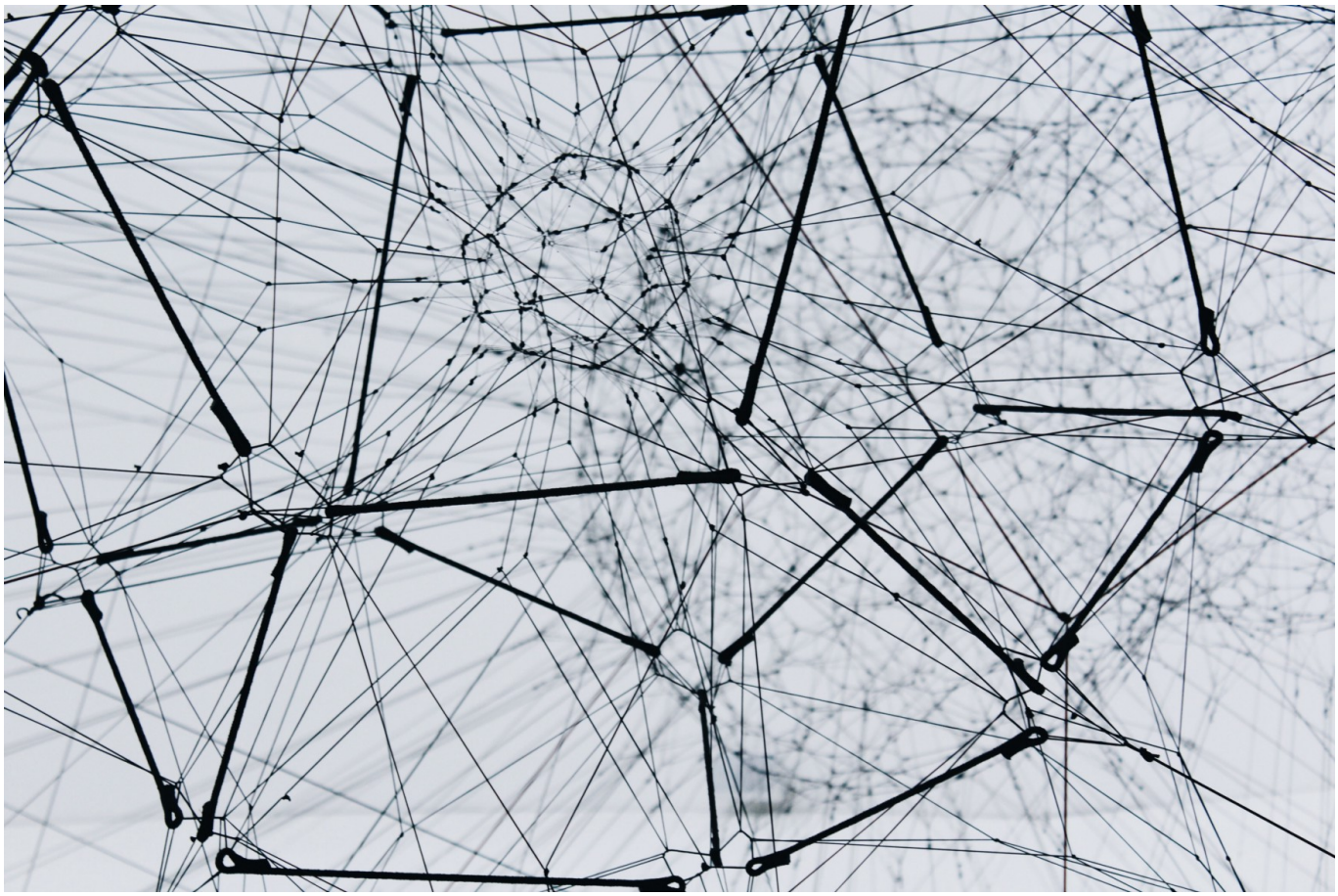


Photo by Alina Grubnyak on Unsplash

Sign up for The Daily Pick

By Towards Data Science

Hands-on real-world examples, research, tutorials, and cutting-edge techniques delivered Monday to Thursday. Make learning your daily ritual. Take a look

Your email

Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy practices.