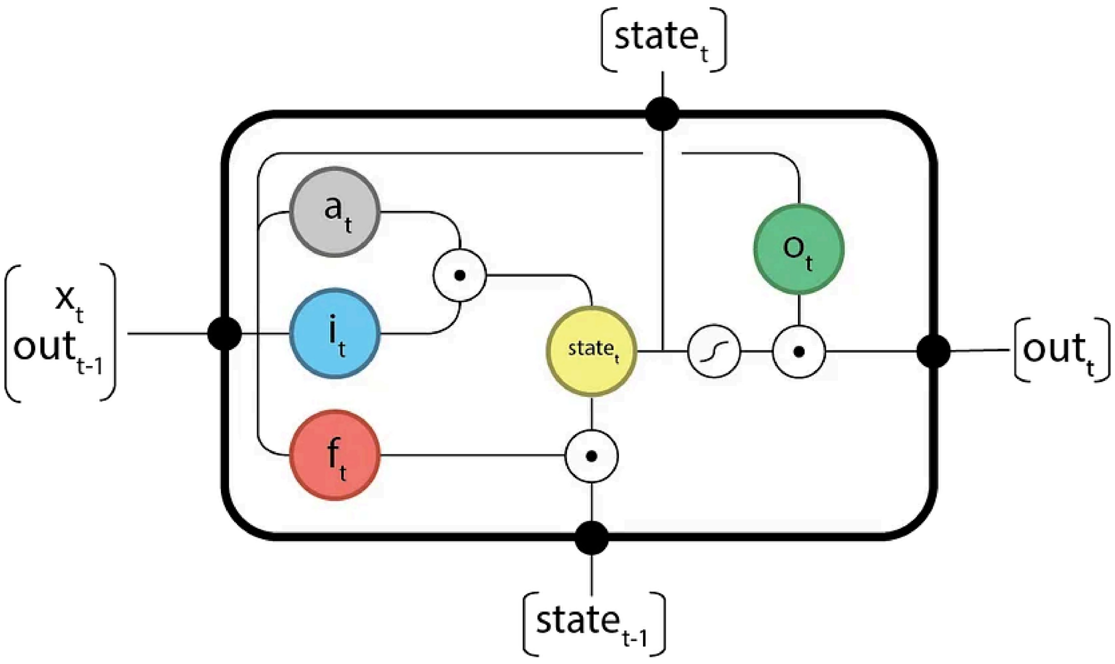
Backpropogating an LSTM: A Numerical Example

Aidan Gomez · Follow
5 min read · Apr 19, 2016

Lets do this…

We all know LSTM's are super powerful; So, we should know how they work and how to use them.



Syntactic notes

- Above ⊙ is the element-wise product or Hadamard product.
- Inner products will be represented as ·
- Outer products will be respresented as ⊗
- σ represents the sigmoid function

The forward components

The gates are defined as:

$$
\begin{aligned}
&\text{Input activation:}\\
&\quad a_t = \tanh(W_a \cdot x_t + U_a \cdot out_{t-1} + b_a)\\
&\text{Input gate:}\\
&\quad i_t = \sigma(W_i \cdot x_t + U_i \cdot out_{t-1} + b_i)\\
&\text{Forget gate:}\\
&\quad f_t = \sigma(W_f \cdot x_t + U_f \cdot out_{t-1} + b_f)\\
&\text{Output gate:}\\
&\quad o_t = \sigma(W_o \cdot x_t + U_o \cdot out_{t-1} + b_o)
\end{aligned}
$$

Which leads to:

Internal state:

$$state_t = a_t \odot i_t + f_t \odot state_{t-1}$$

Output:

$$out_t = \tanh(state_t) \odot o_t$$

Note for simplicity we define:

$$gates_t = \begin{bmatrix} a_t \\ i_t \\ f_t \\ o_t \end{bmatrix},\ W = \begin{bmatrix} W_a \\ W_i \\ W_f \\ W_o \end{bmatrix},\ U = \begin{bmatrix} U_a \\ U_i \\ U_f \\ U_o \end{bmatrix},\ b = \begin{bmatrix} b_a \\ b_i \\ b_f \\ b_o \end{bmatrix}$$

The backward components

Given:

- ΔT the output difference as computed by any subsequent layers (i.e. the rest of your network), and;
- Δout the output difference as computed by the next time-step LSTM (the equation for t-1 is below).

Find:

$$\begin{aligned}
\delta out_t &= \Delta_t + \Delta out_t \\
\delta state_t &= \delta out_t \odot o_t \odot (1 - \tanh^2(state_t)) + \delta state_{t+1} \odot f_{t+1} \\
\delta a_t &= \delta state_t \odot i_t \odot (1 - a_t^2) \\
\delta i_t &= \delta state_t \odot a_t \odot i_t \odot (1 - i_t) \\
\delta f_t &= \delta state_t \odot state_{t-1} \odot f_t \odot (1 - f_t) \\
\delta o_t &= \delta out_t \odot \tanh(state_t) \odot o_t \odot (1 - o_t) \\
\delta x_t &= W^T \cdot \delta gates_t \\
\Delta out_{t-1} &= U^T \cdot \delta gates_t
\end{aligned}$$

The final updates to the internal parameters is computed as:

$$\begin{aligned}
\delta W &= \sum_{t=0}^{T} \delta gates_t \otimes x_t \\
\delta U &= \sum_{t=0}^{T-1} \delta gates_{t+1} \otimes out_t \\
\delta b &= \sum_{t=0}^{T} \delta gates_{t+1}
\end{aligned}$$

Putting this all together we can begin…

The Example

Let us begin by defining out internal weights:

$$\begin{aligned}
W_a &= \begin{bmatrix} 0.45 \\ 0.25 \end{bmatrix}, U_a = \begin{bmatrix} 0.15 \end{bmatrix}, b_a = \begin{bmatrix} 0.2 \end{bmatrix} \\
W_i &= \begin{bmatrix} 0.95 \\ 0.8 \end{bmatrix}, U_i = \begin{bmatrix} 0.8 \end{bmatrix}, b_i = \begin{bmatrix} 0.65 \end{bmatrix} \\
W_f &= \begin{bmatrix} 0.7 \\ 0.45 \end{bmatrix}, U_f = \begin{bmatrix} 0.1 \end{bmatrix}, b_f = \begin{bmatrix} 0.15 \end{bmatrix} \\
W_o &= \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, U_o = \begin{bmatrix} 0.25 \end{bmatrix}, b_o = \begin{bmatrix} 0.1 \end{bmatrix}
\end{aligned}$$

And now input data:

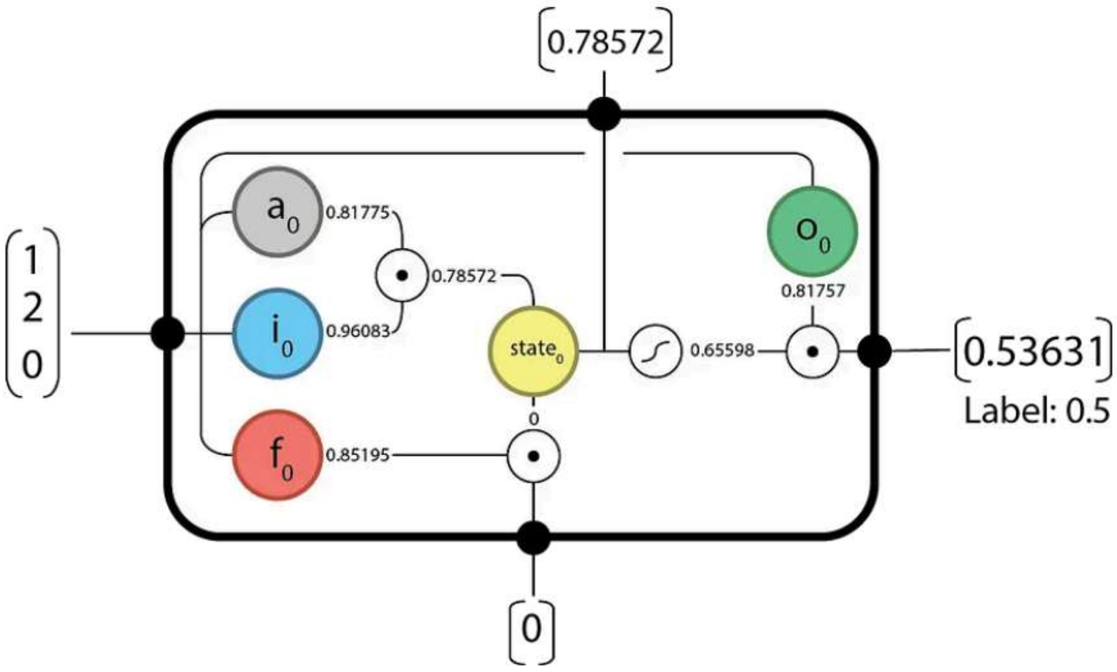
$$x_0 = \begin{bmatrix}1\\2\end{bmatrix} \text{ with label: } 0.5$$

$$x_1 = \begin{bmatrix}0.5\\3\end{bmatrix} \text{ with label: } 1.25$$

> ** Mohamed Challal pointed out to me that a label of 1.25 makes no sense since the outputs are a product of a tanh and sigmoid. Mohamed is completely correct!*

I'm using a sequence length of two here to demonstrate the unrolling over time of RNNs.

Forward @ t=0



$$a_0 = \tanh(W_a \cdot x_0 + U_a \cdot out_{-1} + b_a) = \tanh(\begin{bmatrix}0.45 & 0.25\end{bmatrix} \begin{bmatrix}1\\2\end{bmatrix} + \begin{bmatrix}0.15\end{bmatrix}\begin{bmatrix}0\end{bmatrix} + \begin{bmatrix}0.2\end{bmatrix}) = 0.81775$$

$$i_0 = \sigma(W_i \cdot x_0 + U_i \cdot out_{-1} + b_i) = \sigma(\begin{bmatrix}0.95 & 0.8\end{bmatrix} \begin{bmatrix}1\\2\end{bmatrix} + \begin{bmatrix}0.8\end{bmatrix}\begin{bmatrix}0\end{bmatrix} + \begin{bmatrix}0.65\end{bmatrix}) = 0.96083$$

$$f_0 = \sigma(W_f \cdot x_0 + U_f \cdot out_{-1} + b_f) = \sigma(\begin{bmatrix}0.7 & 0.45\end{bmatrix} \begin{bmatrix}1\\2\end{bmatrix} + \begin{bmatrix}0.1\end{bmatrix}\begin{bmatrix}0\end{bmatrix} + \begin{bmatrix}0.15\end{bmatrix}) = 0.85195$$

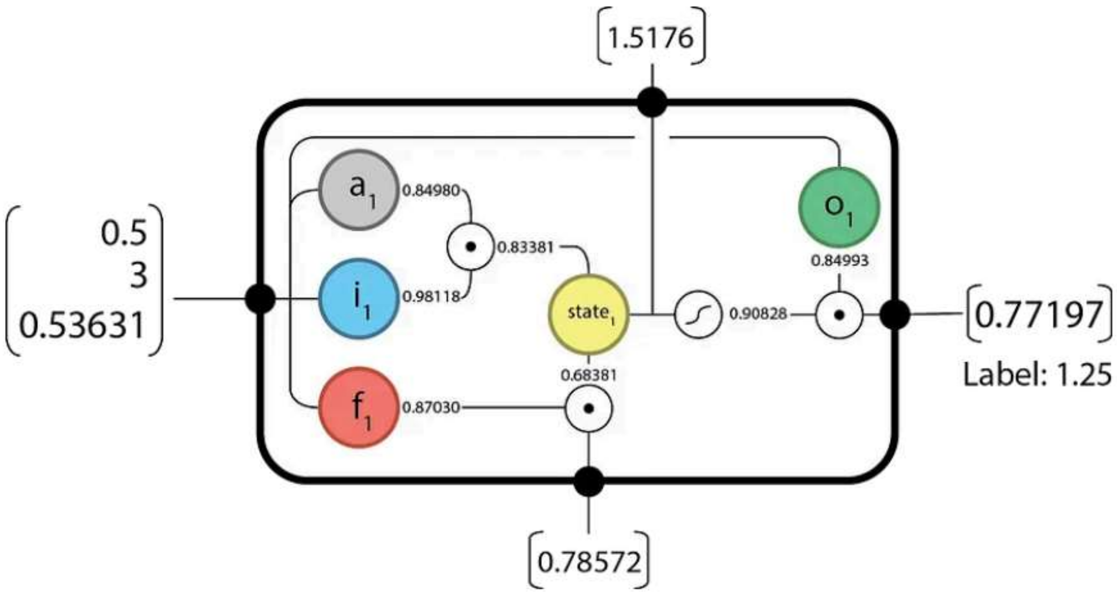
$$o_0 = \sigma(W_o \cdot x_0 + U_o \cdot out_{-1} + b_o) = \sigma(\begin{bmatrix}0.6 & 0.4\end{bmatrix} \begin{bmatrix}1\\2\end{bmatrix} + \begin{bmatrix}0.25\end{bmatrix}\begin{bmatrix}0\end{bmatrix} + \begin{bmatrix}0.1\end{bmatrix}) = 0.81757$$

$$state_0 = a_0 \odot i_0 + f_0 \odot state_{-1} = 0.81775 \times 0.96083 + 0.85195 \times 0 = 0.78572$$

$$out_0 = \tanh(state_0) \odot o_0 = \tanh(0.78572) \times 0.81757 = 0.53631$$

From here, we can pass forward our state and output and begin the next time-step.

Forward @ t=1



$$a_1 = \tanh(W_a \cdot x_1 + U_a \cdot out_0 + b_a) = \tanh(\begin{bmatrix}0.45 & 0.25\end{bmatrix} \begin{bmatrix}0.5 \\ 3\end{bmatrix} + \begin{bmatrix}0.15\end{bmatrix} \begin{bmatrix}0.53631\end{bmatrix} + \begin{bmatrix}0.2\end{bmatrix}) = 0.84980$$

$$i_1 = \sigma(W_i \cdot x_1 + U_i \cdot out_0 + b_i) = \sigma(\begin{bmatrix}0.95 & 0.8\end{bmatrix} \begin{bmatrix}0.5 \\ 3\end{bmatrix} + \begin{bmatrix}0.8\end{bmatrix} \begin{bmatrix}0.53631\end{bmatrix} + \begin{bmatrix}0.65\end{bmatrix}) = 0.98118$$

$$f_1 = \sigma(W_f \cdot x_1 + U_f \cdot out_0 + b_f) = \sigma(\begin{bmatrix}0.7 & 0.45\end{bmatrix} \begin{bmatrix}0.5 \\ 3\end{bmatrix} + \begin{bmatrix}0.1\end{bmatrix} \begin{bmatrix}0.53631\end{bmatrix} + \begin{bmatrix}0.15\end{bmatrix}) = 0.87030$$

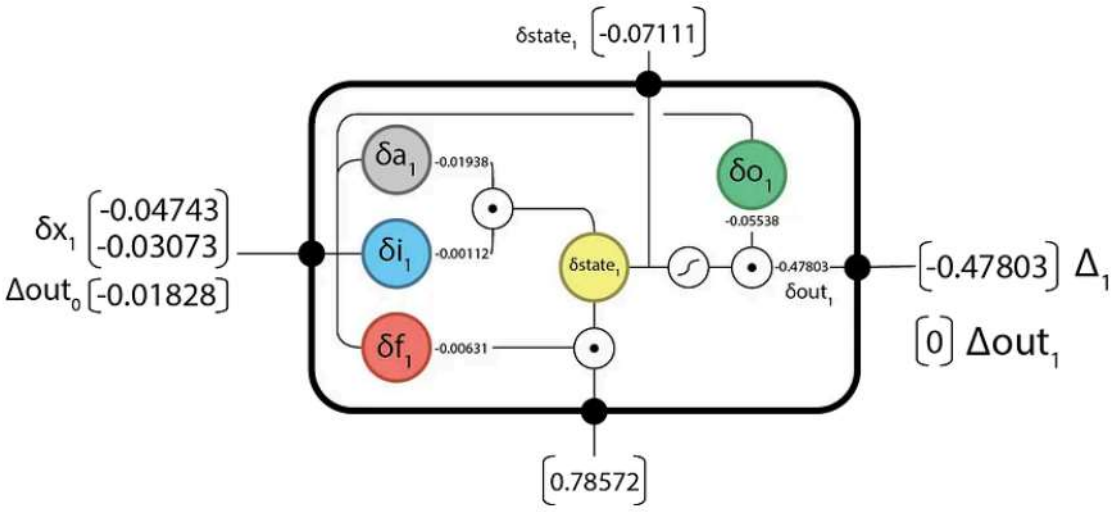
$$o_1 = \sigma(W_o \cdot x_1 + U_o \cdot out_0 + b_o) = \sigma(\begin{bmatrix}0.6 & 0.4\end{bmatrix} \begin{bmatrix}0.5 \\ 3\end{bmatrix} + \begin{bmatrix}0.25\end{bmatrix} \begin{bmatrix}0.53631\end{bmatrix} + \begin{bmatrix}0.1\end{bmatrix}) = 0.84993$$

$$state_1 = a_1 \odot i_1 + f_1 \odot state_0 = 0.84980 \times 0.98118 + 0.87030 \times 0.78572 = 1.5176$$

$$out_1 = \tanh(state_1) \odot o_1 = \tanh(1.5176) \times 0.84993 = 0.77197$$

And since we're done our sequence we have everything we need to begin backpropogating.

Backward @ t=1



First we'll need to compute the difference in output from the expected (label).

Note for this we'll be using L2 Loss:

$$E(x, \hat{x}) = \frac{(x - \hat{x})^2}{2}$$

The derivate w.r.t. x is:

$$\partial_x E(x, \hat{x}) = x - \hat{x}$$

So,

$$\Delta_1 = \partial_x E = 0.77197 - 1.25 = -0.47803$$

$$\Delta out_1 = 0 \text{ because there are no future time-steps.}$$

$$\delta out_1 = \Delta_1 + \Delta out_1 = -0.47803 + 0 = -0.47803$$

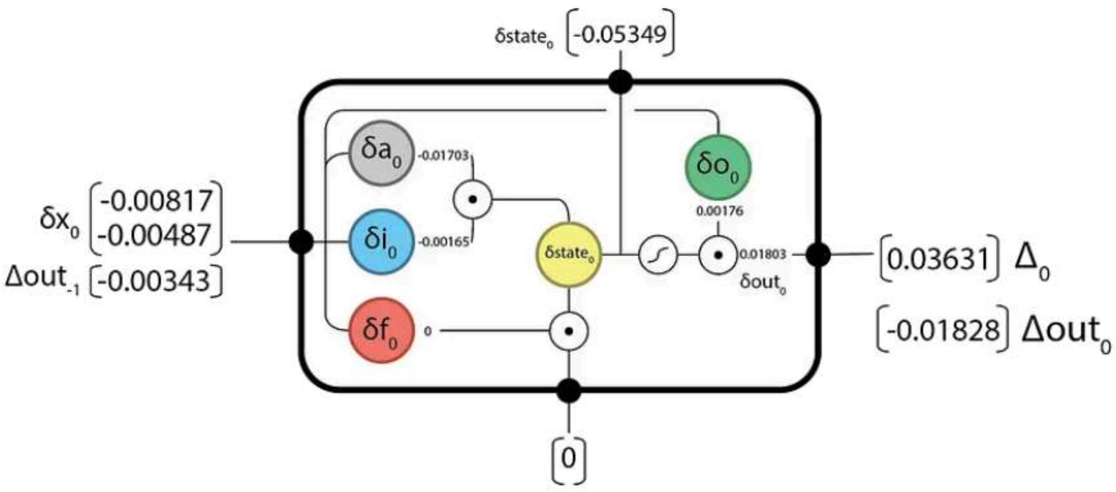
$$\delta state_1 = \delta out_1 \odot o_1 \odot (1 - \tanh^2(state_1)) + \delta state_2 \odot f_2 = -0.47803 \times 0.84993 \times (1 - \tanh^2(1.5176)) + 0 \times 0 = -0.07111$$

$$\delta a_1 = \delta state_1 \odot i_1 \odot (1 - a_1^2) = -0.07111 \times 0.98118 \times (1 - 0.84980^2) = -0.01938$$
$$\delta i_1 = \delta state_1 \odot a_1 \odot i_1 \odot (1 - i_1) = -0.07111 \times 0.84980 \times 0.98118 \times (1 - 0.98118) = -0.00112$$
$$\delta f_1 = \delta state_1 \odot state_0 \odot f_1 \odot (1 - f_1) = -0.07111 \times 0.78572 \times 0.87030 \times (1 - 0.87030) = -0.00631$$
$$\delta o_1 = \delta out_1 \odot \tanh(state_1) \odot o_1 \odot (1 - o_1) = -0.47803 \times \tanh(1.5176) \times 0.84993 \times (1 - 0.84993) = -0.05538$$

$$\delta x_1 = W^T \cdot \delta gates_1$$
$$= \begin{bmatrix} 0.45 & 0.95 & 0.70 & 0.60 \\ 0.25 & 0.80 & 0.45 & 0.40 \end{bmatrix} \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} = \begin{bmatrix} -0.04743 \\ -0.03073 \end{bmatrix}$$
$$\Delta out_0 = U^T \cdot \delta gates_1$$
$$= \begin{bmatrix} 0.15 & 0.80 & 0.10 & 0.25 \end{bmatrix} \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} = -0.01828$$

Now we can pass back our Δout and continue on computing…

Backward @ t=0



$$\Delta_0 = \partial_x E = 0.53631 - 0.5 = 0.03631$$

$$\Delta out_0 = -0.01828, \text{ passed back from T=1}$$

$$\delta out_0 = \Delta_0 + \Delta out_0 = 0.03631 + -0.01828 = 0.01803$$

$$\delta state_0 = \delta out_0 \odot o_0 \odot (1 - \tanh^2(state_0)) + \delta state_1 \odot f_1 = 0.01803 \times 0.81757 \times (1 - \tanh^2(0.78572)) + -0.07111 \times 0.87030 = -0.05349$$

$$\delta a_0 = \delta state_0 \odot i_0 \odot (1 - a_0^2) = -0.05349 \times 0.96083 \times (1 - 0.81775^2) = -0.01703$$

$$\delta i_0 = \delta state_0 \odot a_0 \odot i_0 \odot (1 - i_0) = -0.05349 \times 0.81775 \times 0.96083 \times (1 - 0.96083) = -0.00165$$

$$\delta f_0 = \delta state_0 \odot state_{-1} \odot f_0 \odot (1 - f_0) = -0.05349 \times 0 \times 0.85195 \times (1 - 0.85195) = 0$$

$$\delta o_0 = \delta out_0 \odot \tanh(state_0) \odot o_0 \odot (1 - o_0) = 0.01803 \times \tanh(0.78572) \times 0.81757 \times (1 - 0.81757) = 0.00176$$

$$\delta x_0 = W^T \cdot \delta gates_0$$

$$= \begin{bmatrix} 0.45 & 0.95 & 0.70 & 0.60 \\ 0.25 & 0.80 & 0.45 & 0.40 \end{bmatrix} \begin{bmatrix} -0.01703 \\ -0.00165 \\ 0 \\ 0.00176 \end{bmatrix} = \begin{bmatrix} -0.00817 \\ -0.00487 \end{bmatrix}$$

$$\Delta out_{-1} = U^T \cdot \delta gates_1$$

$$= \begin{bmatrix} 0.15 & 0.80 & 0.10 & 0.25 \end{bmatrix} \begin{bmatrix} -0.01703 \\ -0.00165 \\ 0 \\ 0.00176 \end{bmatrix} = -0.00343$$

And we're done the backward step!

Now we'll need to update our internal parameters according to whatever solving algorithm you've chosen. I'm going to use a simple Stochastic Gradient Descent (SGD) update with learning rate: λ=0.1λ0.1.

We'll need to compute how much our weights are going to change by:

$$\delta W = \sum_{t=0}^{T} \delta gates_t \otimes x_t$$

$$= \begin{bmatrix} -0.01703 \\ -0.00165 \\ 0 \\ 0.00176 \end{bmatrix} \begin{bmatrix} 1.0 & 2.0 \end{bmatrix} + \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} \begin{bmatrix} 0.5 & 3.0 \end{bmatrix} = \begin{bmatrix} -0.02672 & -0.0922 \\ -0.00221 & -0.00666 \\ -0.00316 & -0.01893 \\ -0.02593 & -0.16262 \end{bmatrix}$$

$$\delta U = \sum_{t=0}^{T-1} \delta gates_{t+1} \otimes out_t$$

$$= \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} \begin{bmatrix} 0.53631 \end{bmatrix} = \begin{bmatrix} -0.01039 \\ -0.00060 \\ -0.00338 \\ -0.02970 \end{bmatrix}$$

$$\delta b = \sum_{t=0}^{T} \delta gates_{t+1}$$

$$= \begin{bmatrix} -0.01703 \\ -0.00165 \\ 0 \\ 0.00176 \end{bmatrix} + \begin{bmatrix} -0.01938 \\ -0.00112 \\ -0.00631 \\ -0.05538 \end{bmatrix} = \begin{bmatrix} -0.03641 \\ -0.00277 \\ -0.00631 \\ -0.05362 \end{bmatrix}$$

And updating out parameters based on the SGD update function:

$$W^{new} = W^{old} - \lambda * \delta W^{old}$$

$$W_a = \begin{bmatrix} 0.45267 \\ 0.25922 \end{bmatrix}, U_a = \begin{bmatrix} 0.15104 \end{bmatrix}, b_a = \begin{bmatrix} 0.20364 \end{bmatrix}$$

$$W_i = \begin{bmatrix} 0.95022 \\ 0.80067 \end{bmatrix}, U_i = \begin{bmatrix} 0.80006 \end{bmatrix}, b_i = \begin{bmatrix} 0.65028 \end{bmatrix}$$

$$W_f = \begin{bmatrix} 0.70031 \\ 0.45189 \end{bmatrix}, U_f = \begin{bmatrix} 0.10034 \end{bmatrix}, b_f = \begin{bmatrix} 0.15063 \end{bmatrix}$$

$$W_o = \begin{bmatrix} 0.60259 \\ 0.41626 \end{bmatrix}, U_o = \begin{bmatrix} 0.25297 \end{bmatrix}, b_o = \begin{bmatrix} 0.10536 \end{bmatrix}$$

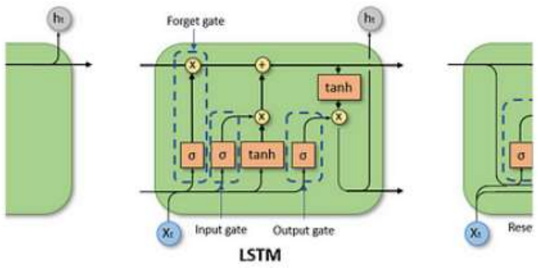
And that completes one iteration of solving an LSTM cell!

Errata and Frequently Asked Questions:

- **Q: in `d state_t` did you mean to use `tanh²(state_{t-1})`?**
  A: No.
- **Q: you compute `d x` but never use it. Why?**
  A: you would use it if there were LSTMs stacked beneath, or any trainable component leading into the LSTM. Since `x` is the input data in my example, we don't really care about that particular gradient.
- **Q: under Backwards @ t=0: you use `delta out_{-1} = U^T d gates_1`, but it should use `gates_0`.**
  A: Nice catch!

Of course, this whole process is sequential in nature and a small error will render all subsequent calculations useless, so if you catch something email me at hello@aidangomez.ca

Please feel free to share with the machine learning enthusiasts in your life!



Jonte Dancker in Towards Data Science

A Brief Introduction to Recurrent Neural Networks

An introduction to RNN, LSTM, and GRU and their implementation

Dec 26, 2022 · 624 · 7

Alexzap

How to Filter Out Market Noise in Algo-Trading Strategies: JPM Use...

Avoiding False Trading Signals in Stock Market Data: Zero-Phase Butterworth and 12...

Aug 1 · 55 · 3

Lists

Predictive Modeling w/ Python

20 stories · 1428 saves

Practical Guides to Machine Learning

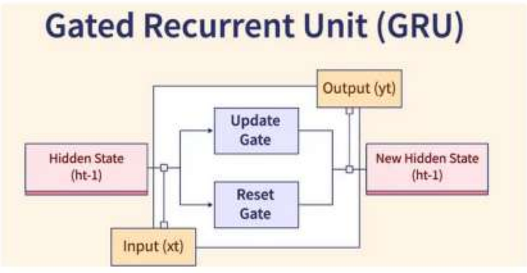
10 stories · 1725 saves

Natural Language Processing

1625 stories · 1191 saves

data science and AI

40 stories · 206 saves

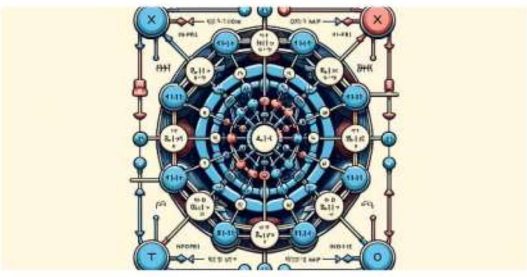


Harshed

Understanding Gated Recurrent Units (GRUs) in Deep Learning

An Informative Deep Dive into Gated Recurrent Units

Mar 25 · 3

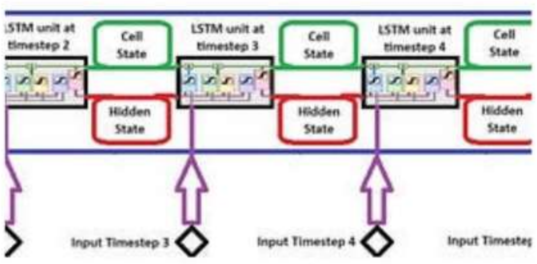


Cristian Leo in Towards Data Science

The Math Behind Recurrent Neural Networks

Dive into RNNs, the backbone of time series, understand their mathematics, implement...

Apr 27 · 733 · 3



Seyed Mousavi in GoPenAI

Temporal Data in LSTM Time Series Forecasting

If you noticed in our previous articles, when building LSTM models, we either specified...

Jun 8 · 23

Cristian Leo in Towards Data Science

The Math Behind LSTM

Dive into LSTMs, understand their mathematics, and implement them from...

May 1 · 621 · 2

See more recommendations