Long Short-Term Memory

M. Stanley Fujimoto

CS778 – Winter 2016

30 Jan 2016

Why

- List the alphabet forwards
  - List the alphabet backwards
- Tell me the lyrics to a song
  - Start the lyrics of the song in the middle of a verse
- Lots of information that you store in your brain is not random access
  - You learned them as a sequence
- How can we incorporate this into the machine learning algorithm?

“Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask.” Accessed January 31, 2016. https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.

Why – Use Cases

- Predict the next word in a sentence
  - The woman took out ________ purse
- Predict the next frame in a video
- **All these tasks are easier when you know what happened earlier in the sequence**

“Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask.” Accessed January 31, 2016. https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.

Why – Markov Models

- Traditional Markov model approaches are limited because their states must be drawn from a modestly sized discrete state space S
- Standard operations become infeasible when the set of possible hidden states grows large
- Markov models are computationally impractical for modeling long-range dependencies

Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.

Why – Neural Network

input -> hidden -> output

“Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask.” Accessed January 31, 2016. https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.

Why – Neural Network, Extended

(input + prev_input) -> hidden -> output

“Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask.” Accessed January 31, 2016. https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.

Why – Neural Network, Extended

```
(input + empty_input) -> hidden -> output
 (input + prev_input) -> hidden -> output
 (input + prev_input) -> hidden -> output
 (input + prev_input) -> hidden -> output
```

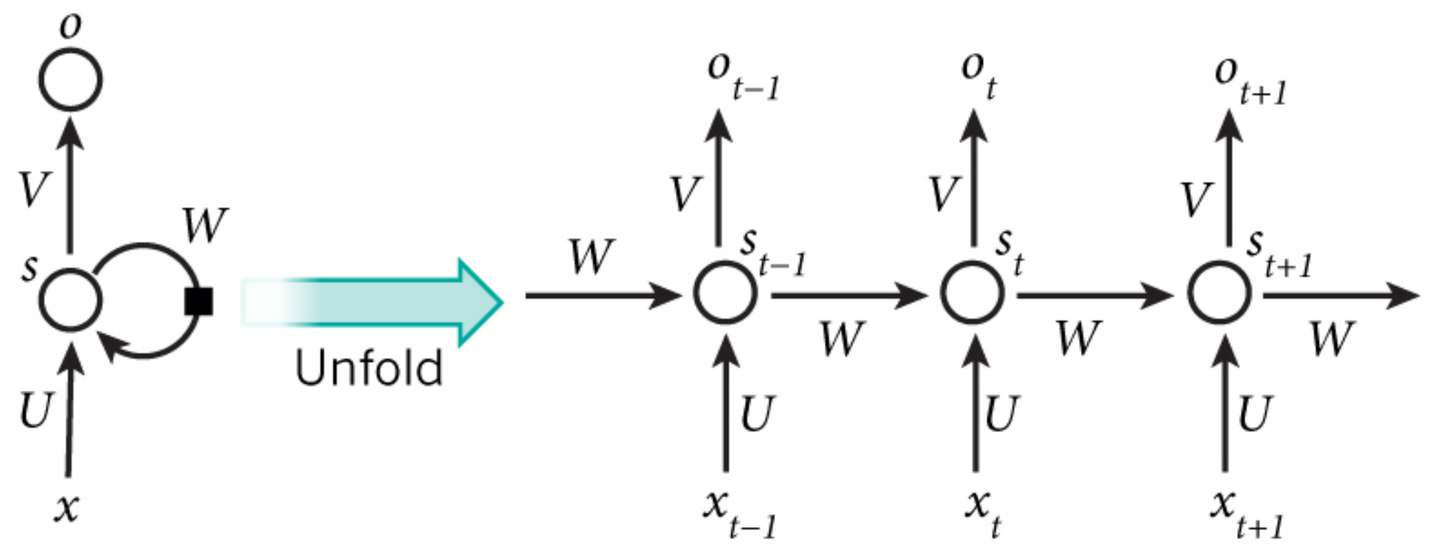
“Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask.” Accessed January 31, 2016. https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.

Why – Recurrent Neural Network

(input + prev_hidden) -> hidden -> output

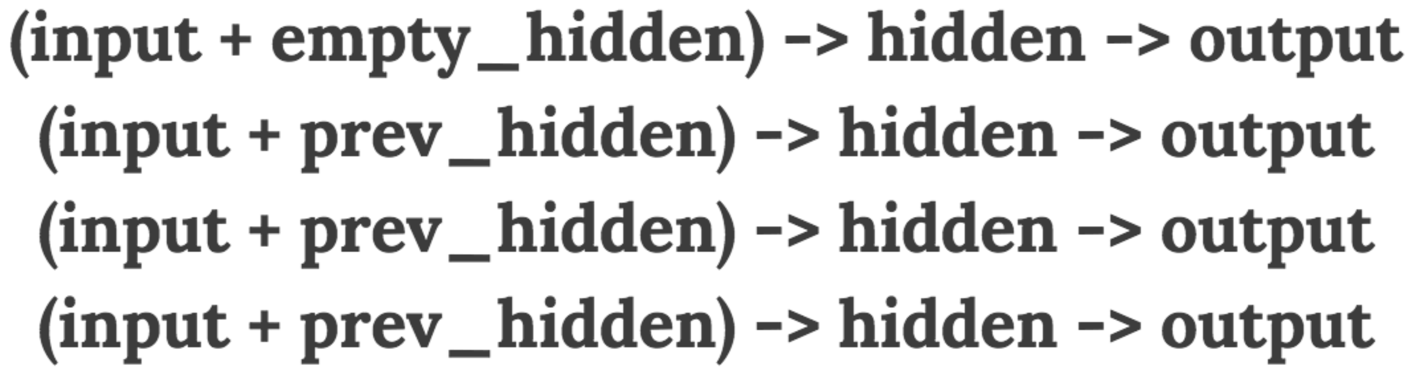
“Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask.” Accessed January 31, 2016.
https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.

Why – Recurrent Neural Network



"Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano." *WildML*, October 27, 2015. http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/.

Why – Recurrent Neural Network

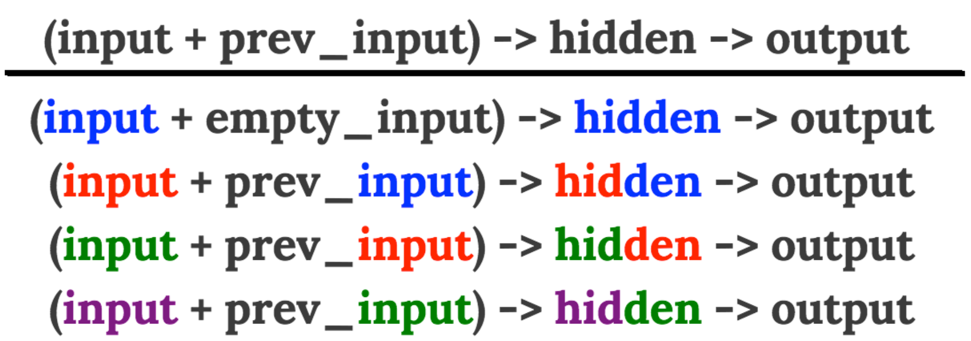


(input + empty_hidden) -> hidden -> output
(input + prev_hidden) -> hidden -> output
(input + prev_hidden) -> hidden -> output
(input + prev_hidden) -> hidden -> output

“Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask.” Accessed January 31, 2016. https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.

Why – Continued Influence

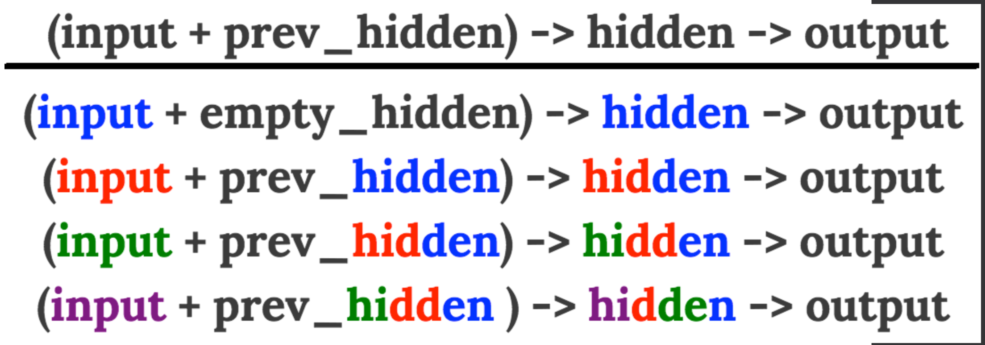
Neural Network, Extended



(input + prev_input) -> hidden -> output

(input + empty_input) -> hidden -> output
(input + prev_input) -> hidden -> output
(input + prev_input) -> hidden -> output
(input + prev_input) -> hidden -> output

Recurrent Neural Network

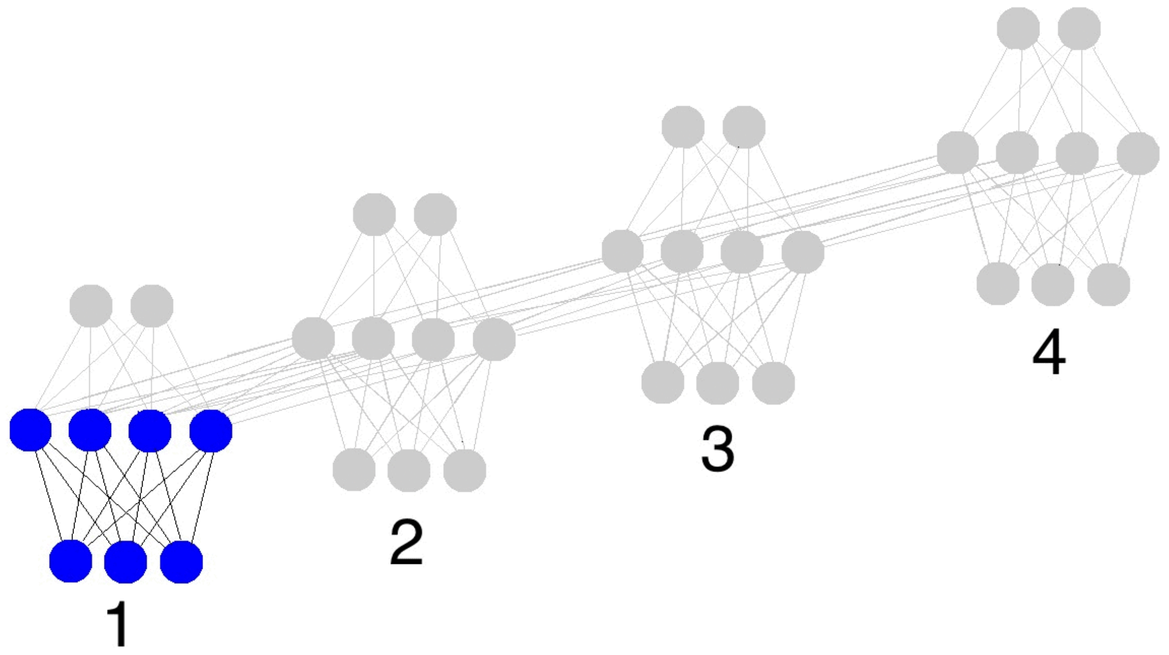


(input + prev_hidden) -> hidden -> output

(input + empty_hidden) -> hidden -> output
(input + prev_hidden) -> hidden -> output
(input + prev_hidden) -> hidden -> output
(input + prev_hidden) -> hidden -> output

“Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask.” Accessed January 31, 2016. https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.

Why – Recurrent Neural Network



"Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask." Accessed January 31, 2016. https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.

Why - LSTM

- Designed to overcome:
  - Long-term dependencies
  - Vanishing/exploding gradients

Why – Long-Term Dependencies

- We don't want to remember everything, just the important things for a long time

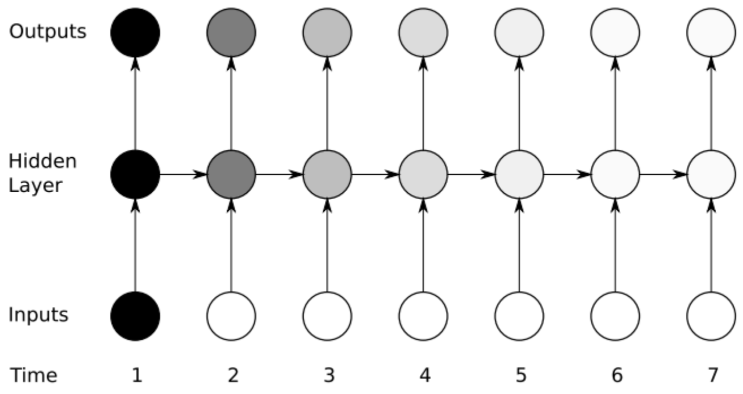


Figure 4.1: **The vanishing gradient problem for RNNs.** The shading of the nodes in the unfolded network indicates their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decays over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs.

Graves, Alex. *Supervised sequence labelling*. Springer Berlin Heidelberg, 2012.

Why – Long-Term Dependencies

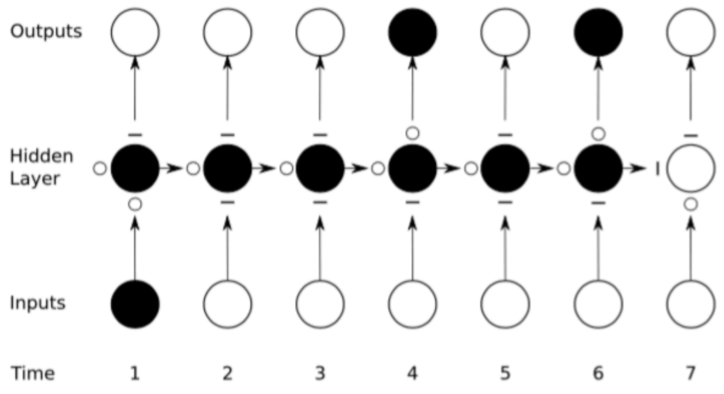
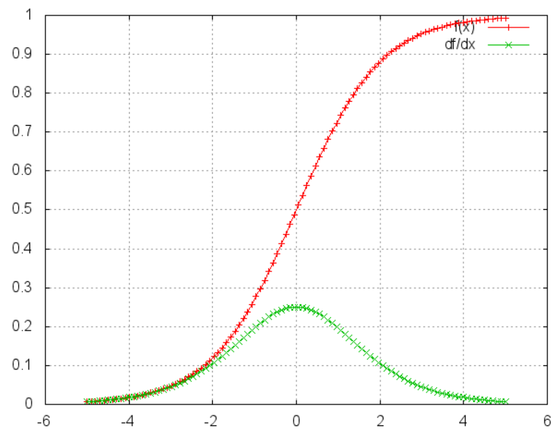


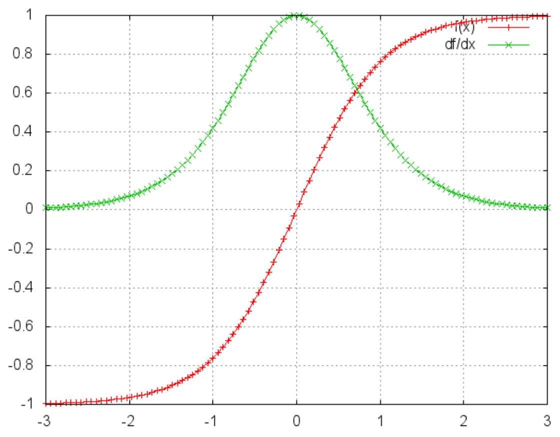
Figure 4.4: **Preservation of gradient information by LSTM.** As in Figure 4.1 the shading of the nodes indicates their sensitivity to the inputs at time one; in this case the black nodes are maximally sensitive and the white nodes are entirely insensitive. The state of the input, forget, and output gates are displayed below, to the left and above the hidden layer respectively. For simplicity, all gates are either entirely open ('O') or closed ('—'). The memory cell 'remembers' the first input as long as the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell.

Graves, Alex. *Supervised sequence labelling*. Springer Berlin Heidelberg, 2012.

Why – Vanishing/Exploding Gradients

- When weight or activation functions (their derivatives) are:
  - < 1 Vanishing Gradients
  - > 1 Exploding Gradients

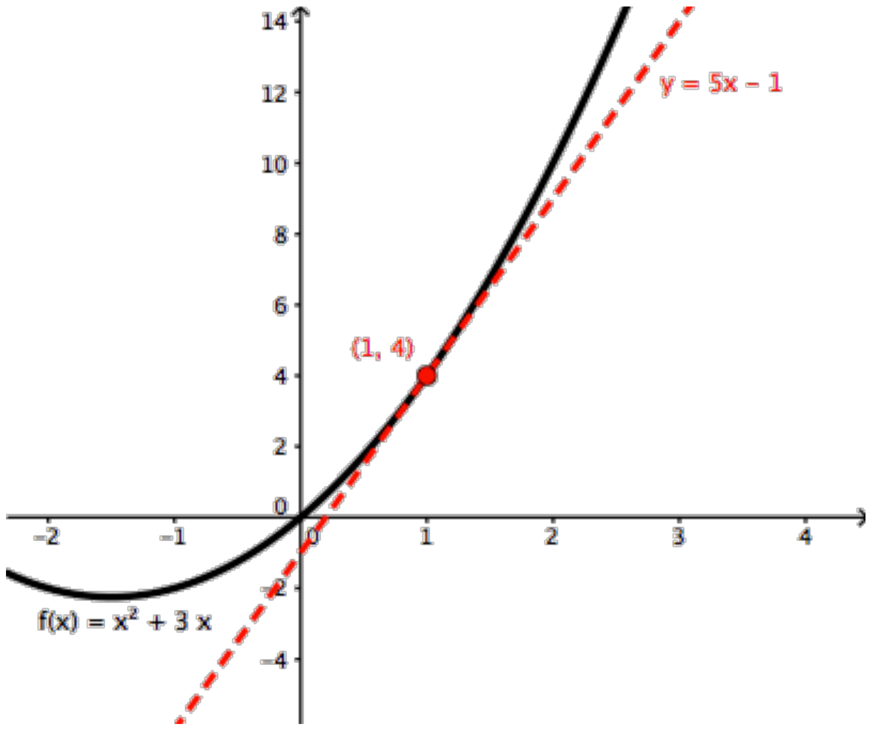




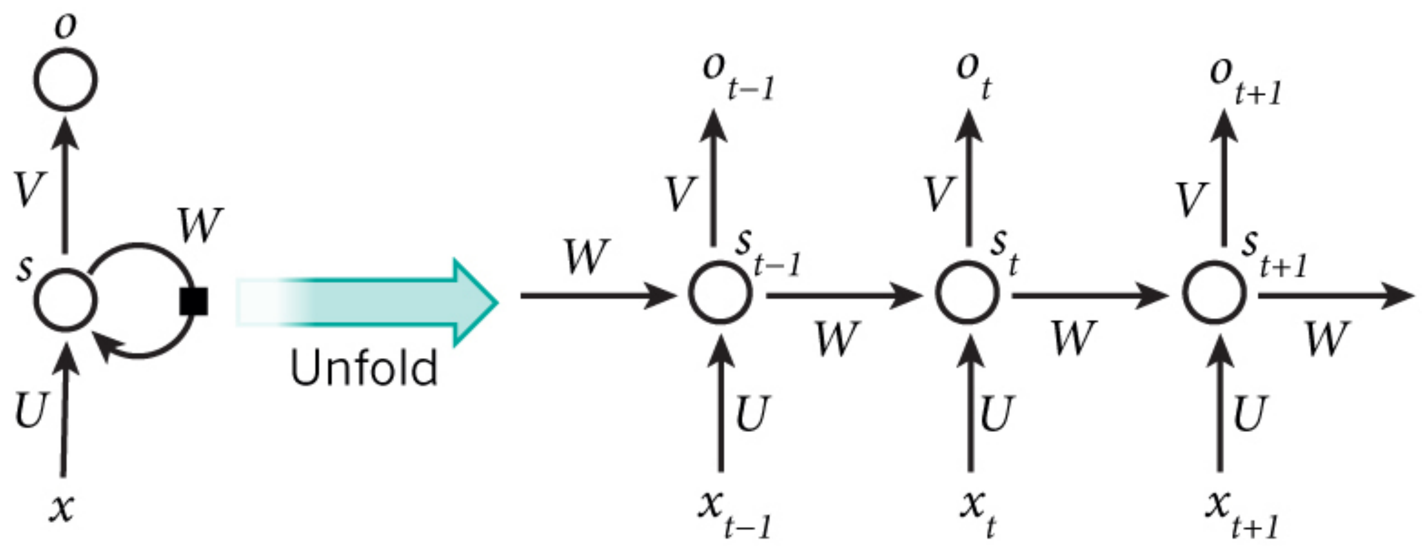
“(1) How Does LSTM Help Prevent the Vanishing (and Exploding) Gradient Problem in a Recurrent Neural Network? - Quora.” Accessed February 19, 2016. https://www.quora.com/How-does-LSTM-help-prevent-the-vanishing-and-exploding-gradient-problem-in-a-recurrent-neural-network.

“Transfer Functions - Nn.” Accessed February 19, 2016. http://nn.readthedocs.org/en/rtd/transfer/.

Why – Backpropagation

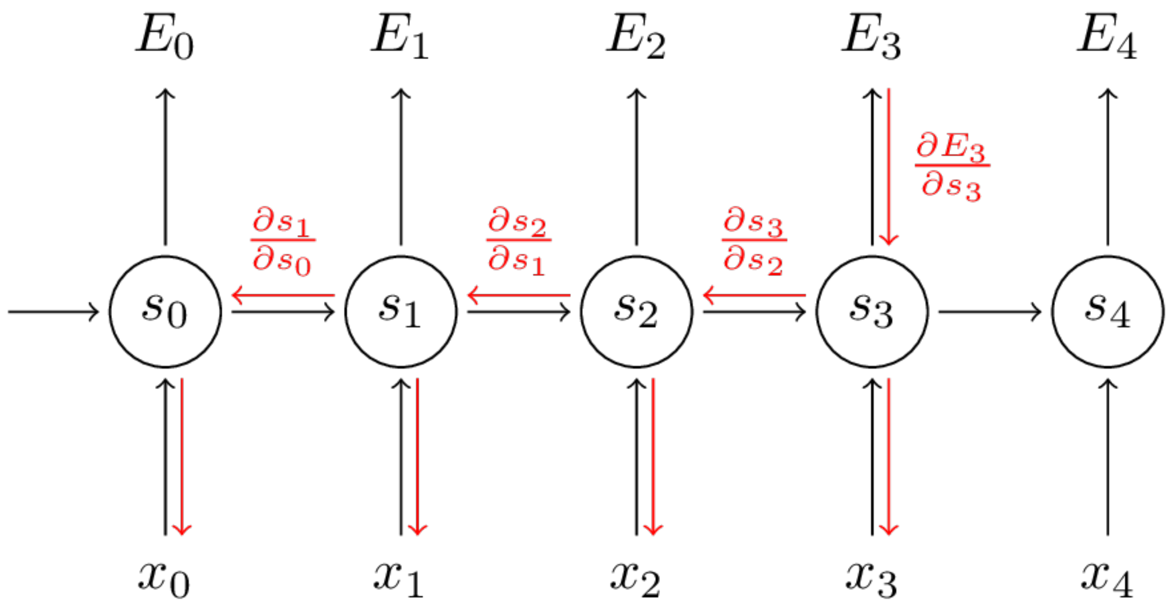


Why – Vanishing Gradients



"Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano." *WildML*, October 27, 2015. http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/.

Why – Vanishing Gradients



E_0
E_1
E_2
E_3
E_4
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4

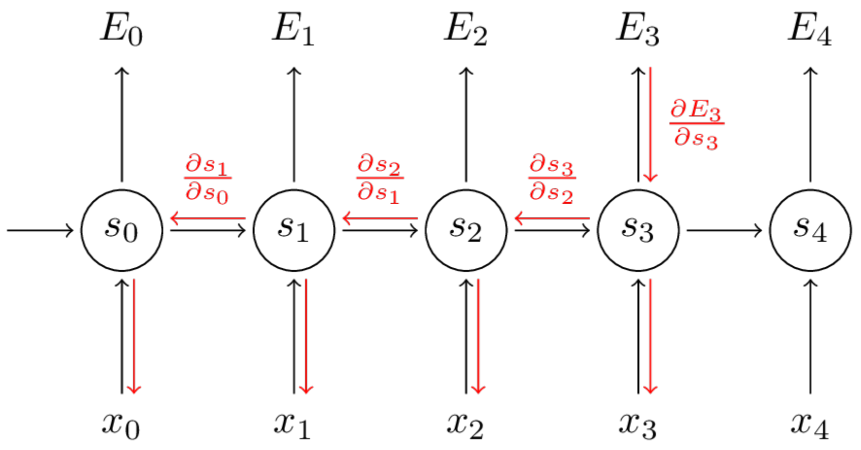
"Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients." *WildML*, October 8, 2015. http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/.

Why – Vanishing Gradients

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$

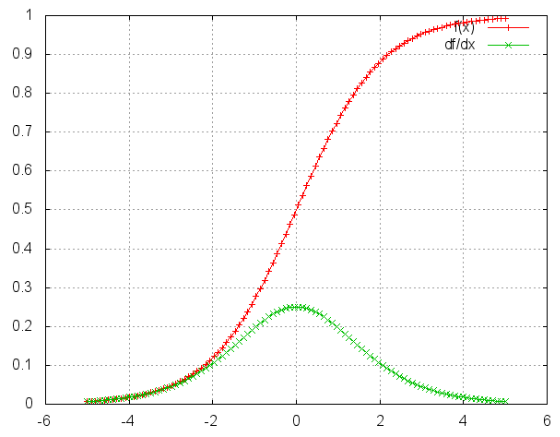
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\left(\prod_{j=k+1}^{3}\frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial s_k}{\partial W}$$

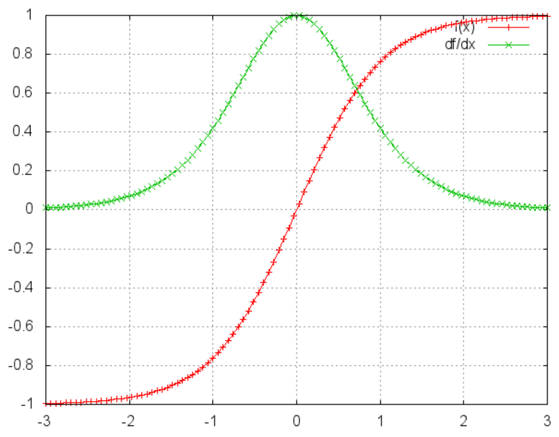


E_0
E_1
E_2
E_3
E_4
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4

"Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients." *WildML*, October 8, 2015. http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/.

How – Vanishing Gradients





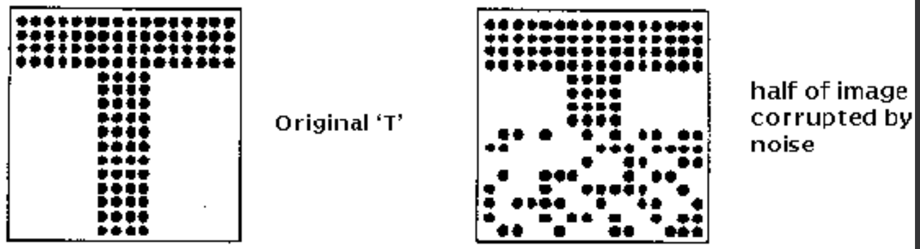
How – History

- Foundational research done in the 1980's
  - 1982 – Hopfield: Introduction of family of recurrent neural networks
  - 1986 – Jordan: RNN architecture
  - 1989 – Williams and Zipser: Truncated BackProp Through Time (TBPTT)
  - 1990 – Elman: simpler RNN architecture
  - 1997 – Hochreiter and Schmidhuber: LSTM networks
  - 1999 – Gers, Schmidhuber, Cummins: Forget gate
  - 2005 – Graves and Schmidhuber: Bidirectional LSTM
  - 2012 – Pascanu, Mikolov, Bengio: Gradient clipping
  - 2014 – Cho, Bahdanau, van Merrienboer, Bougares: Gated Recurrent Unit
  - 2014 – Sutsekver, et al.: Sequence-to-Sequence Learning with Neural Nets

Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.

How – 1982, Hopfield

- Recurrent neural networks with pattern recognition capabilities
- The net stores 1 or more patterns
- Recalls the pattern given partial input

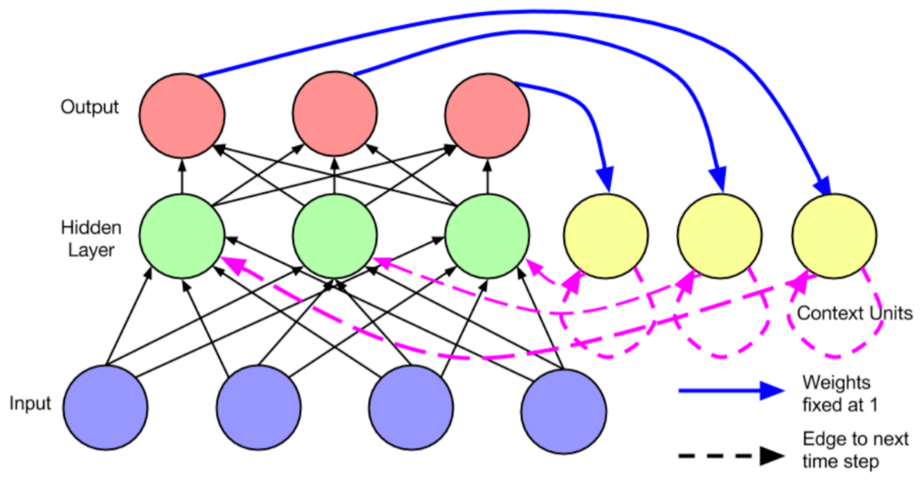


"A Hopfield Net Example." Accessed January 30, 2016. http://web.cs.ucla.edu/~rosen/161/notes/hopfield.html.

Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.

How – 1986, Jordan

- Feed-forward recurrent neural network for supervised learning
- Output layer into special units
- Special units are self-connected
  - Allow for information to be sent across multiple time steps without changing output during intermediate time steps



Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.

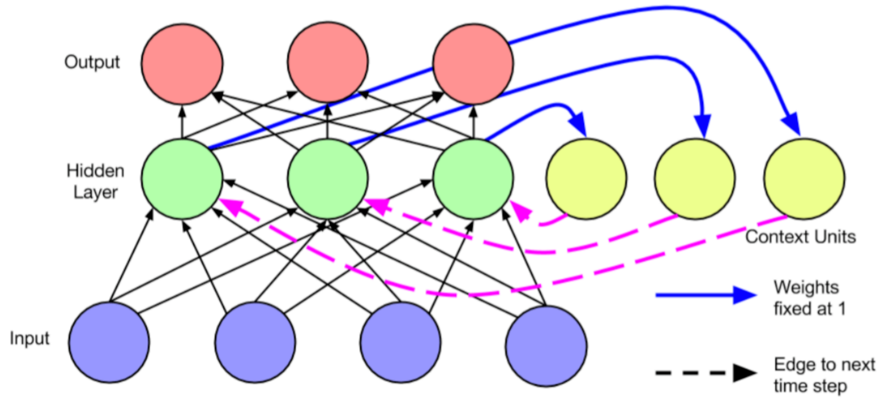
How – 1989, Williams and Zipser

- Truncated BackPropagation Through Time (TBPTT)
- Mitigates issues of exploding gradients
    - Comes at the cost of long-term dependencies due to vanishing gradient

Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.

How – 1990, Elman

- Simpler than Jordan network
- Hidden representation into special units
- Trained with backprop
- Fundamental to the development of LSTMs



Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.

How – 1997, Sepp & Jürgen

Hochreiter and Schmidhuber

- Designed to overcome the problem of vanishing/exploding gradients
  - Introduction of the memory cell
  - Each memory cell has a self-connected recurrent edge

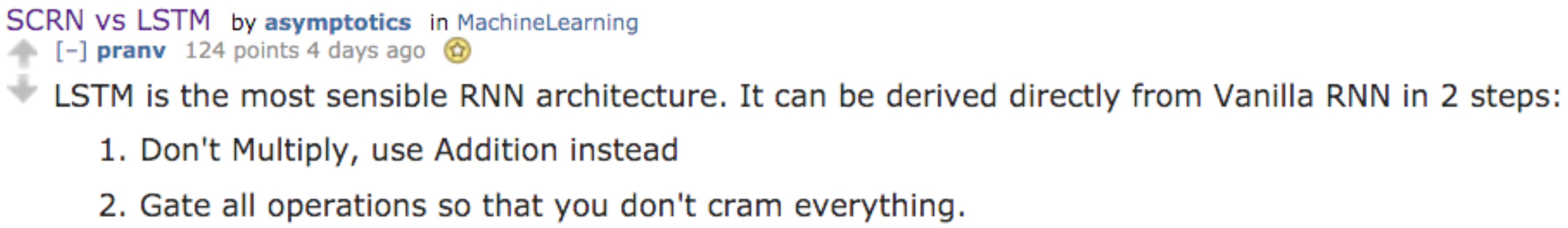
Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.

How – 1999, Gers, Schmidhuber, Cummins

- Added the Forget Gate to the LSTM structure, published in "Learning to Forget: Continual Prediction with LSTM"

Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM." *Neural Computation* 12, no. 10 (October 1, 2000): 2451–71. doi:10.1162/089976600300015015.

How – A Reddit Explanation


SCRN vs LSTM by asymptotics in MachineLearning

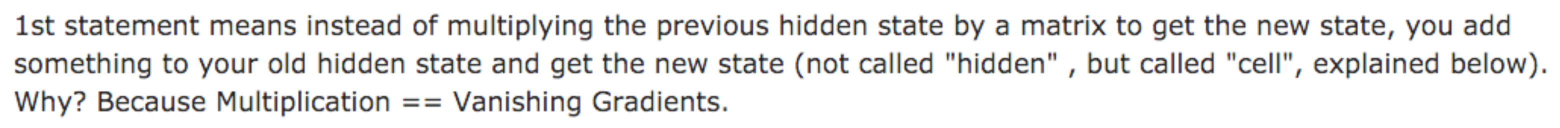
[–] pranv 124 points 4 days ago

LSTM is the most sensible RNN architecture. It can be derived directly from Vanilla RNN in 2 steps:

1. Don't Multiply, use Addition instead
2. Gate all operations so that you don't cram everything.

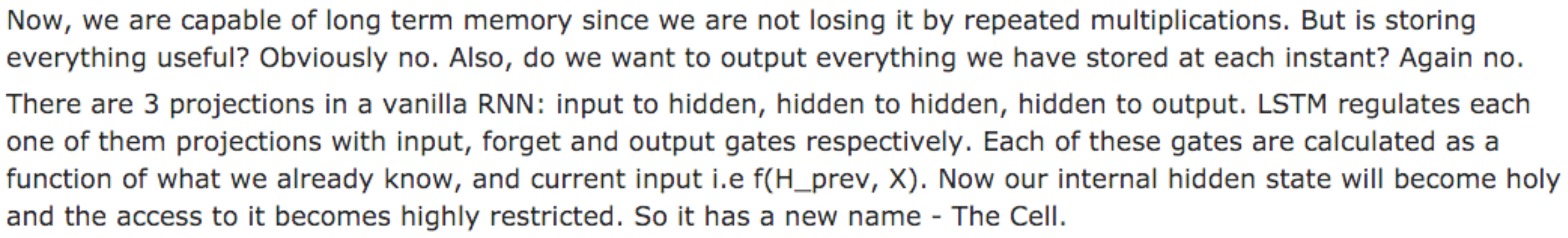
"SCRN vs LSTM • /r/MachineLearning." *Reddit*. Accessed February 10, 2016.
https://www.reddit.com/r/MachineLearning/comments/44bxdj/scrn_vs_lstm/.

Why – Don’t multiply


1st statement means instead of multiplying the previous hidden state by a matrix to get the new state, you add something to your old hidden state and get the new state (not called "hidden" , but called "cell", explained below). Why? Because Multiplication == Vanishing Gradients.

“SCRN vs LSTM • /r/MachineLearning.” *Reddit*. Accessed February 10, 2016. https://www.reddit.com/r/MachineLearning/comments/44bxdj/scrn_vs_lstm/.

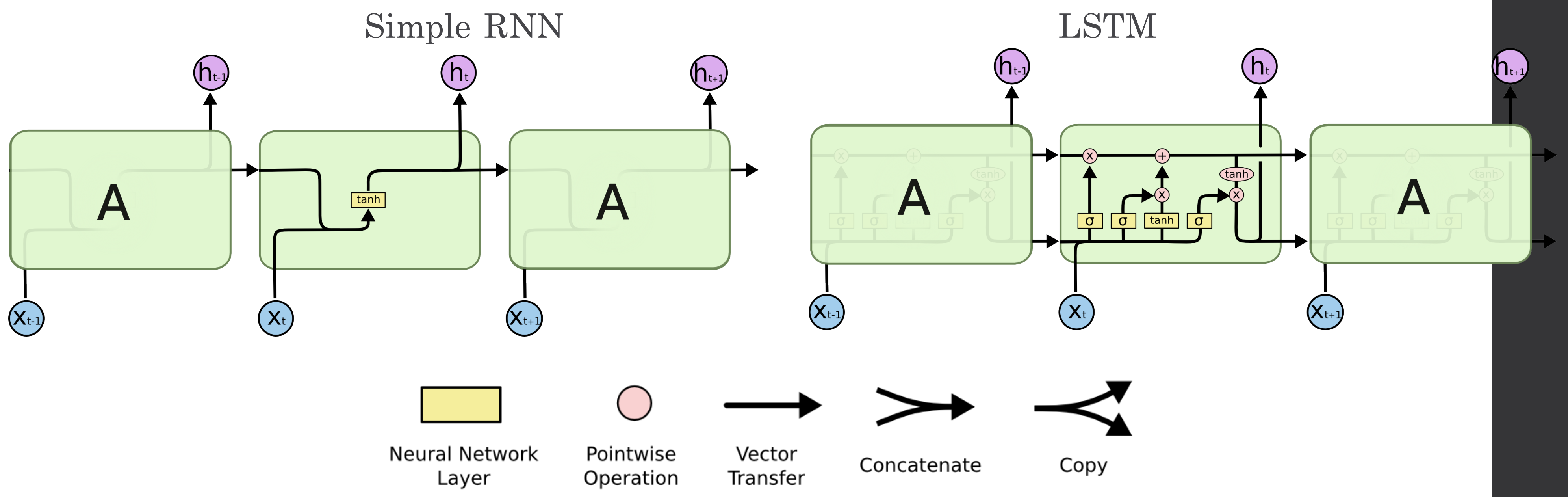
Why – Gate all operations


Now, we are capable of long term memory since we are not losing it by repeated multiplications. But is storing everything useful? Obviously no. Also, do we want to output everything we have stored at each instant? Again no.

There are 3 projections in a vanilla RNN: input to hidden, hidden to hidden, hidden to output. LSTM regulates each one of them projections with input, forget and output gates respectively. Each of these gates are calculated as a function of what we already know, and current input i.e f(H_prev, X). Now our internal hidden state will become holy and the access to it becomes highly restricted. So it has a new name - The Cell.

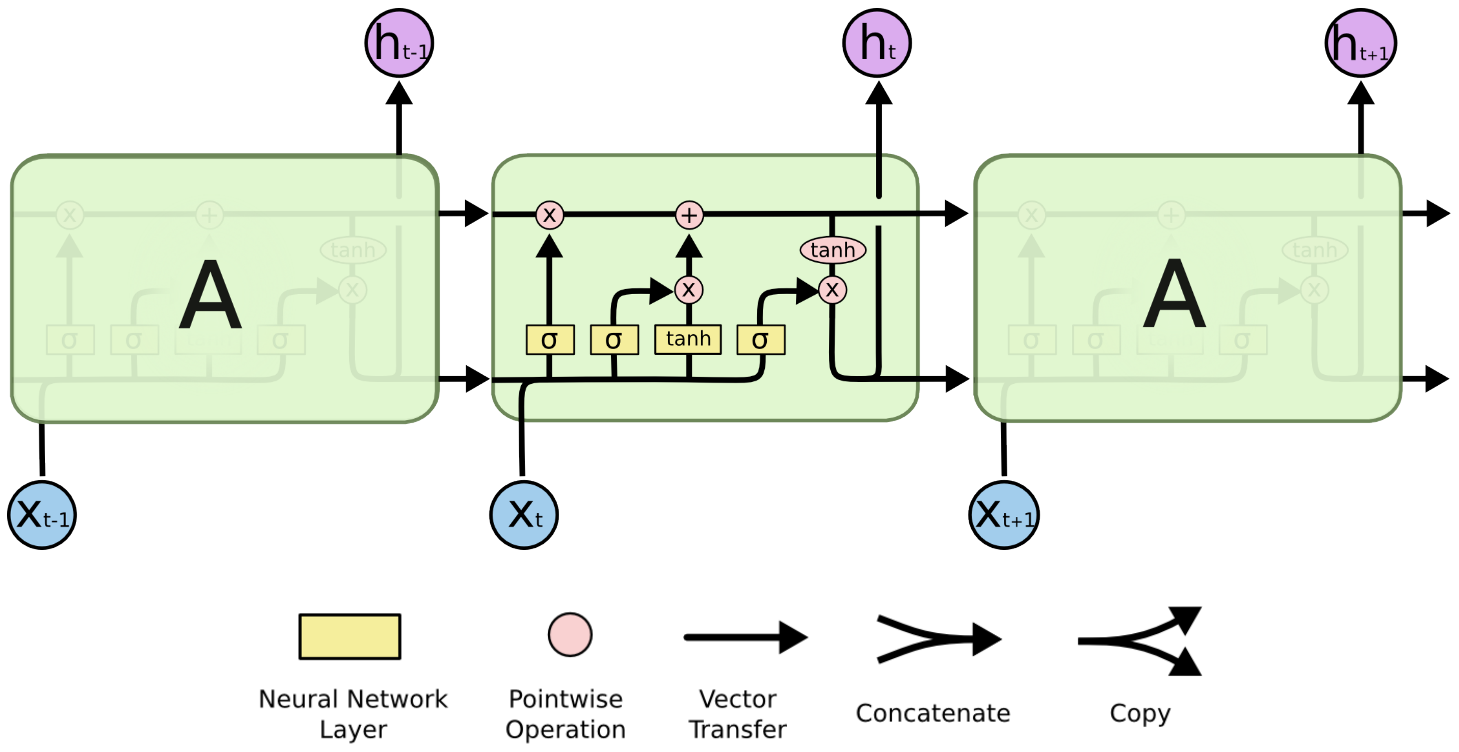
"SCRN vs LSTM • /r/MachineLearning." *Reddit*. Accessed February 10, 2016. https://www.reddit.com/r/MachineLearning/comments/44bxdj/scrn_vs_lstm/.

How

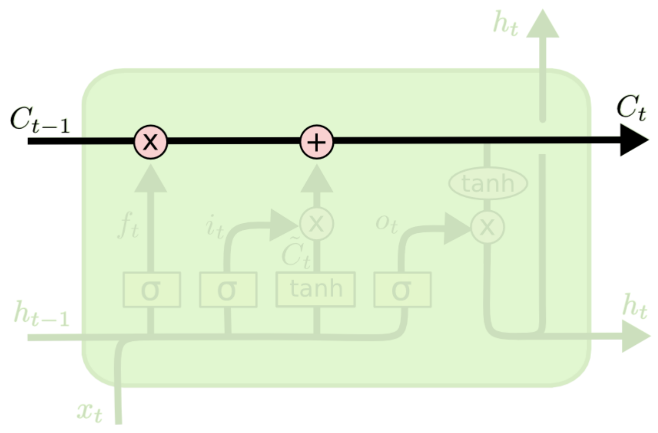


"Understanding LSTM Networks -- Colah's Blog." Accessed January 25, 2016. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

How – LSTM Structure

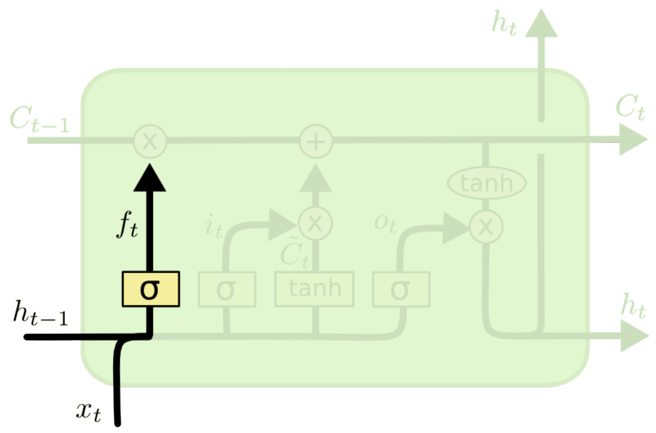


How – Step by Step: Cell State



"Understanding LSTM Networks -- Colah's Blog." Accessed January 25, 2016. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

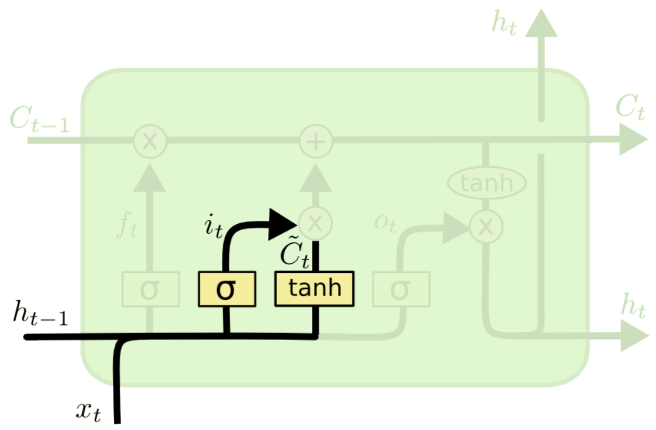
How – Step by Step: Forget Layer



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

“Understanding LSTM Networks -- Colah’s Blog.” Accessed January 25, 2016. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

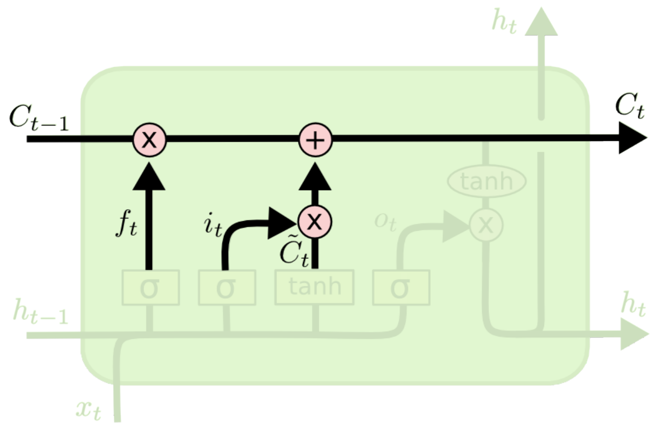
How – Step by Step: Input Gate Layer



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

“Understanding LSTM Networks -- Colah’s Blog.” Accessed January 25, 2016. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

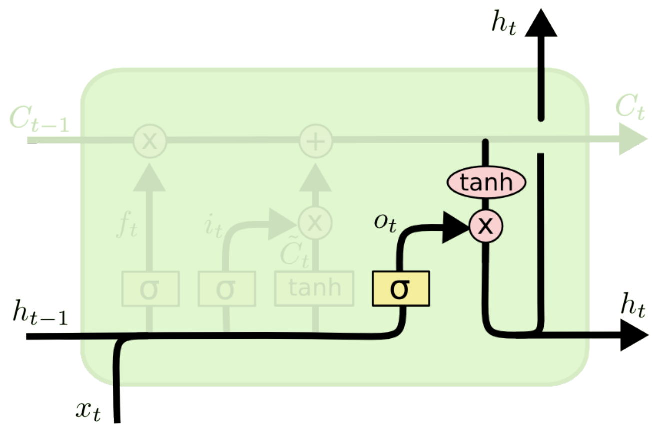
How – Step by Step: Cell State Update



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

"Understanding LSTM Networks -- Colah's Blog." Accessed January 25, 2016. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

How – Step by Step: Output Value

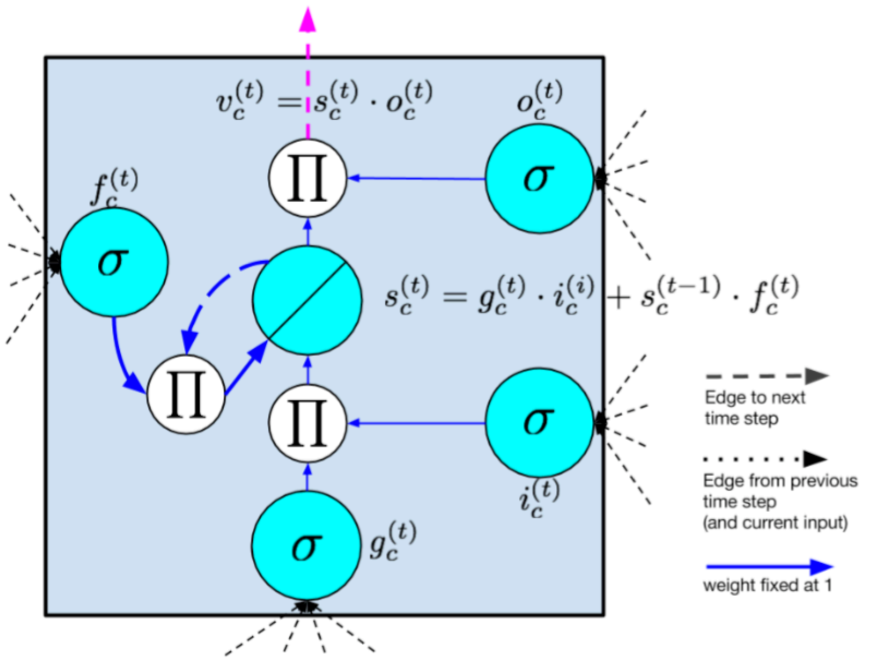


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

"Understanding LSTM Networks -- Colah's Blog." Accessed January 25, 2016. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

How - LSTM Memory Cell

- Input node: $g_c(t)$
- Input gate: $i_c(t)$
- Internal state: $s_c(t)$
- Forget gate: $f_c(t)$
- Output gate: $o_c(t)$
- Final output: $v_c(t)$


$v_c^{(t)} = s_c^{(t)} \cdot o_c^{(t)}$
$o_c^{(t)}$
$f_c^{(t)}$
$s_c^{(t)} = g_c^{(t)} \cdot i_c^{(i)} + s_c^{(t-1)} \cdot f_c^{(t)}$
$i_c^{(t)}$
$g_c^{(t)}$
σ
Π
Edge to next time step
Edge from previous time step (and current input)
weight fixed at 1

Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.

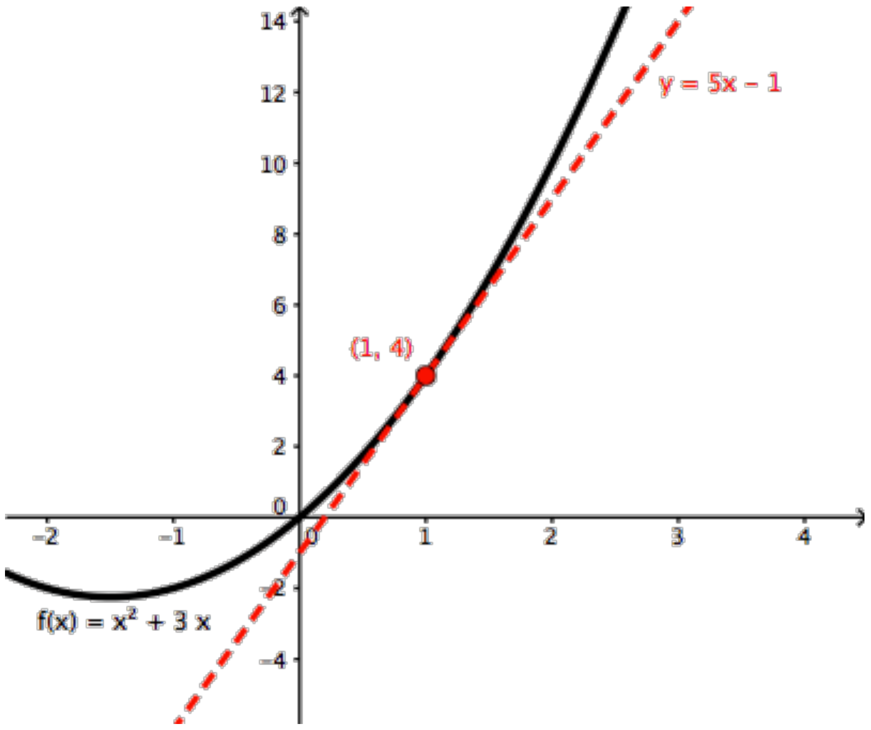
How – Backpropagation

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

$$\frac{\partial E_{total}}{\partial w_1} = (\sum_o \delta_o * w_{ho}) * out_{h1}(1 - out_{h1}) * i_1$$

Mazur. “A Step by Step Backpropagation Example.” *Matt Mazur*, March 17, 2015. http://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.

How – Backpropagation



How – RNN Backpropagation

- Normal BPTT (Backprop Through Time)

How – Clarifications

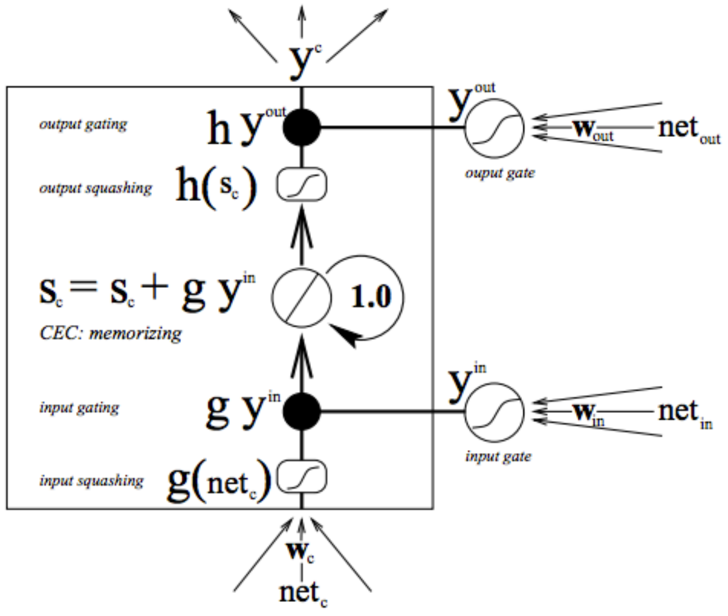


Figure 1: The standard LSTM cell has a linear unit with a recurrent self-connection with weight 1.0 (CEC). Input and output gate regulate read and write access to the cell whose state is denoted s_c. The function g squashes the cell's input; h squashes the cell's output (see text for details).

Gers, Felix A., J. urgen Schmidhuber, and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM." (1999).

How – Clarifications

- CEC
  - Constant error carousel
- Weights = 1
- Identity function as activation function

How – Clarifications

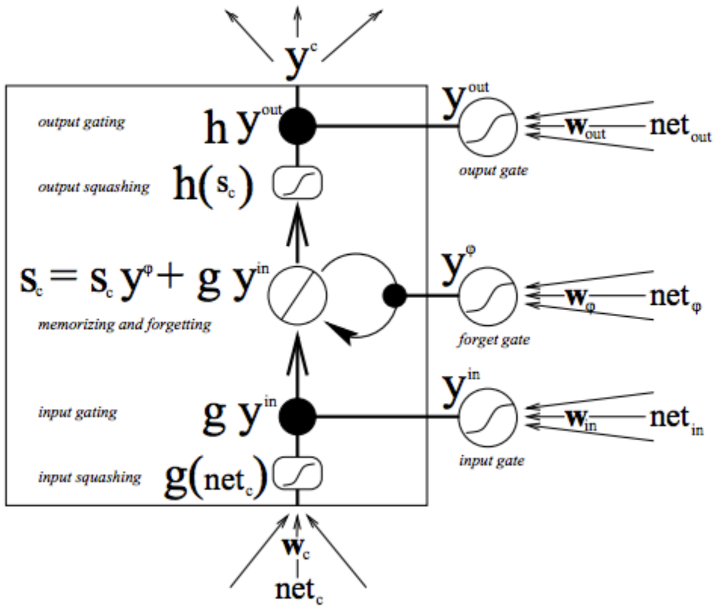
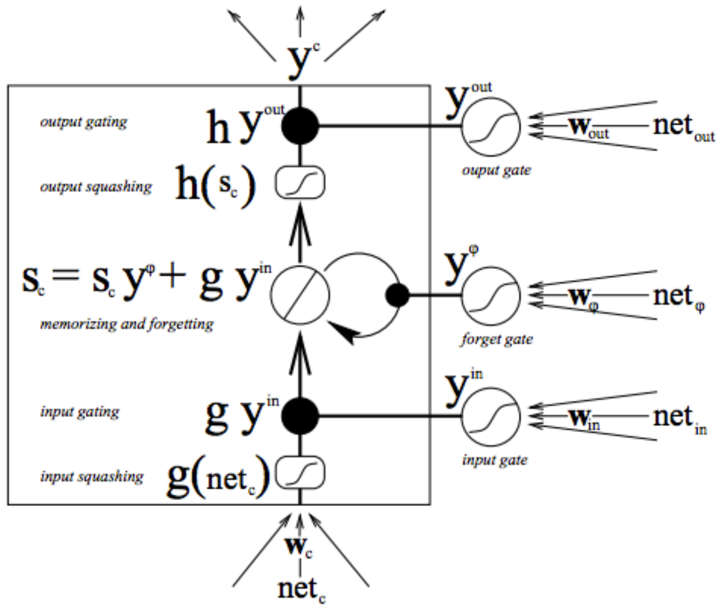


Figure 2: Memory block with only one cell for the extended LSTM. A multiplicative forget gate can reset the cell's inner state s_c.

Gers, Felix A., J. urgen Schmidhuber, and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM." (1999).

How – Clarifications

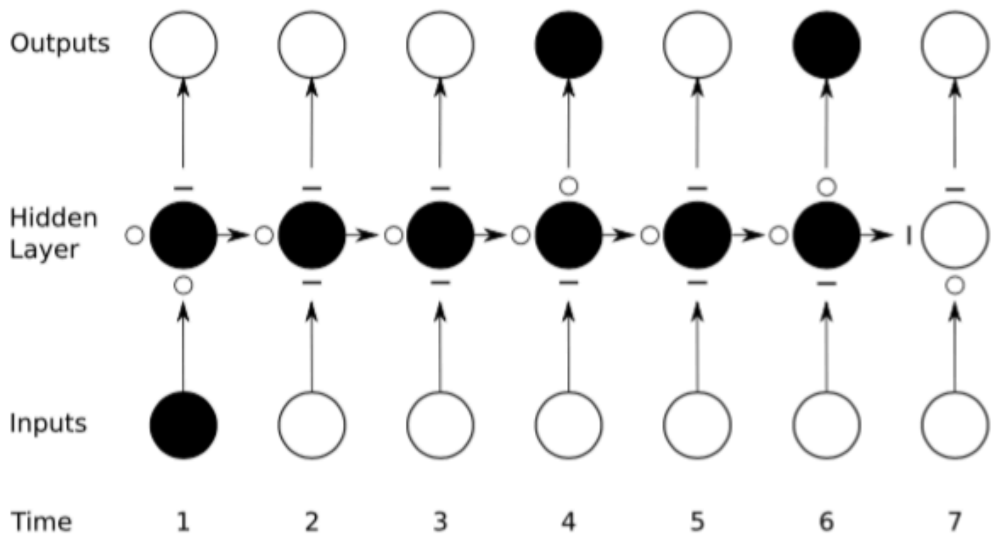


- Input Gate: Write
- Output Gate: Read
- Forget Gate: Reset

Gers, Felix A., J. urgen Schmidhuber, and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM." (1999).

“Deep Learning Lecture 12: Recurrent Neural Nets and LSTMs - YouTube.” Accessed February 27, 2016. https://www.youtube.com/watch?v=56TYLaQN4N8.

How – Clarifications



- Input Gate: Write
- Output Gate: Read
- Forget Gate: Reset

Graves, Alex. *Supervised sequence labelling*. Springer Berlin Heidelberg, 2012.

“Deep Learning Lecture 12: Recurrent Neural Nets and LSTMs - YouTube.” Accessed February 27, 2016. https://www.youtube.com/watch?v=56TYLaQN4N8.

How – Clarifications

- **Forget Gate**
- LSTMs (with no forget gate) have difficulty with continual input streams
  - A continual input stream is not segmented into subsequences (no starts, no ends)
- A forget (reset) gate allows the memory cell to reset itself
- Without the reset the cell state may grow indefinitely

Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. “Learning to Forget: Continual Prediction with LSTM.” *Neural Computation* 12, no. 10 (October 1, 2000): 2451–71. doi:10.1162/089976600300015015.

How – Clarifications

- **Vanishing/Exploding Gradients**
- In RNNs, caused by the repeated use of weight matrices

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta}$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1}))$$

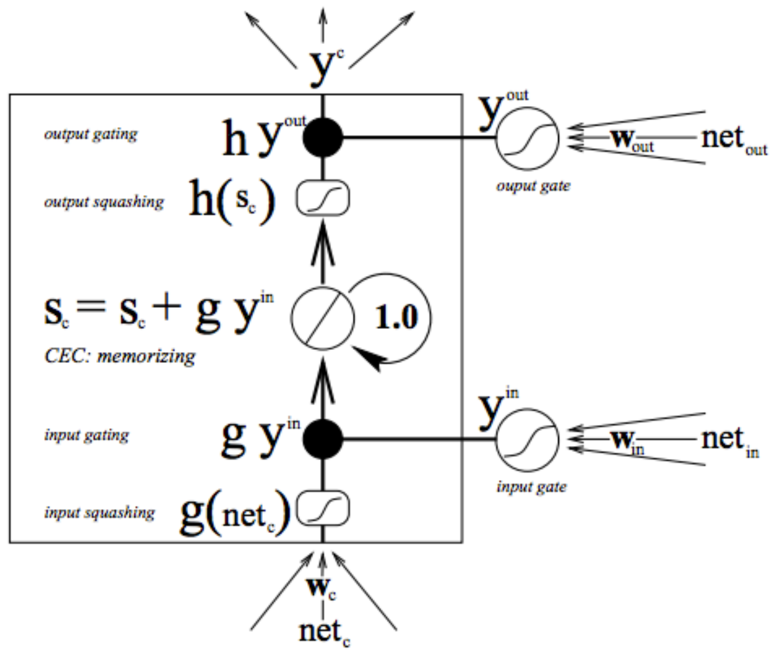
How – LSTM Backpropagation

- Same as BPTT but must take into account fancy activation functions
- http://arunmallya.github.io/writeups/nn/lstm/index.html#/

“LSTM.” Accessed February 20, 2016. http://arunmallya.github.io/writeups/nn/lstm/index.html#/.

How – Derive Weight Updates

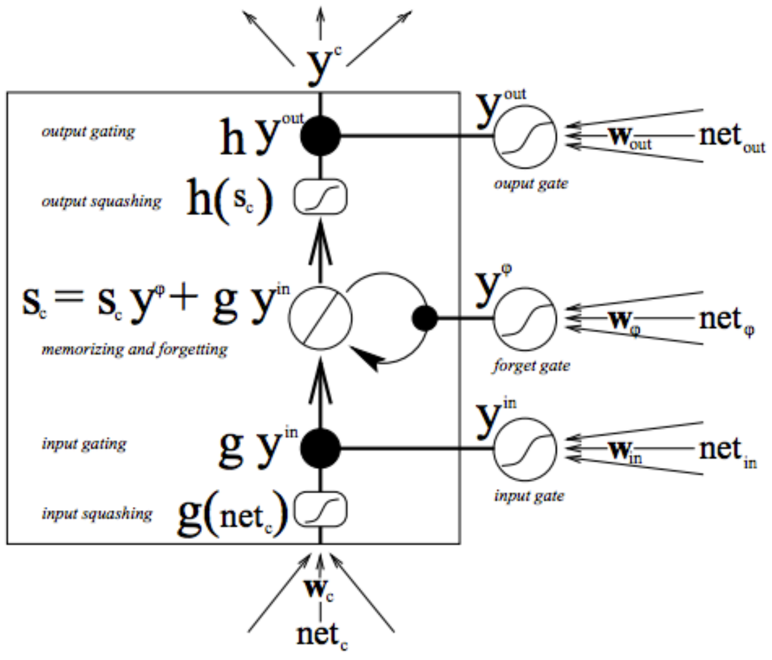
How – Solving Life’s Problems



- The cell state’s weights are the identity function
  - No repeated use of weights
- Protects against vanishing and exploding gradient

Gers, Felix A., J. urgen Schmidhuber, and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM." (1999).

How – Solving Life’s Problems

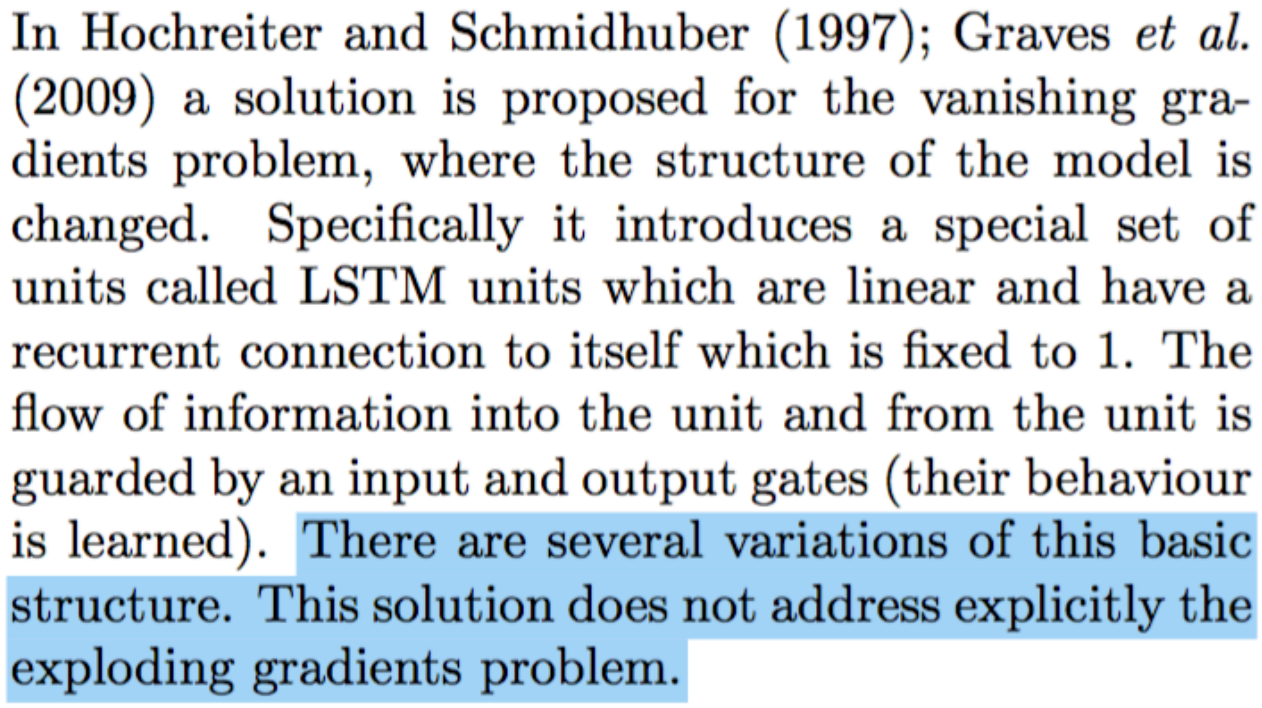


- Derivative is the forget gate
- Forgetting values and adding new values for long-term dependencies
- Does not explicitly address the exploding gradients problem

Gers, Felix A., J. urgen Schmidhuber, and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM." (1999).

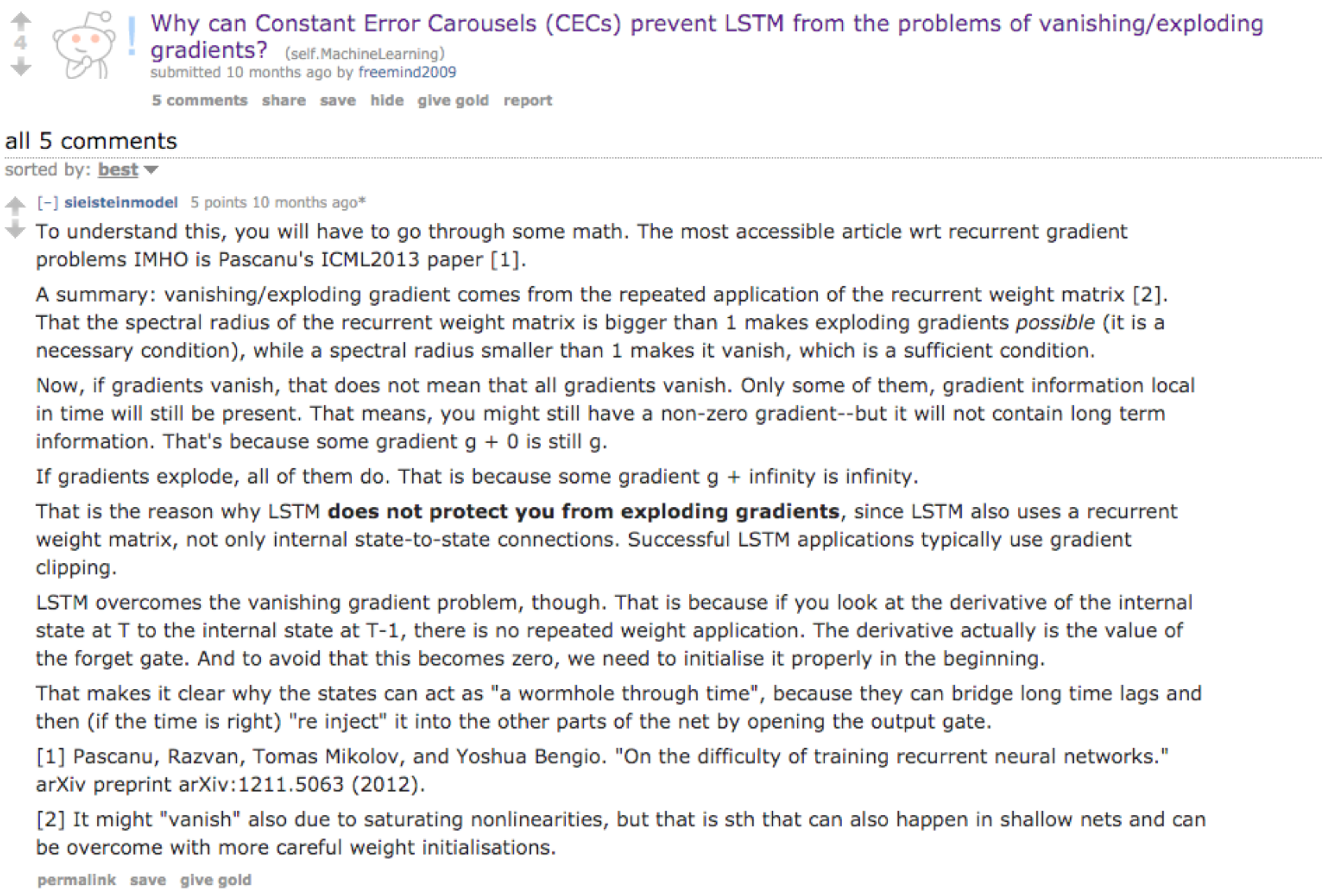
“Why Can Constant Error Carousels (CECs) Prevent LSTM from the Problems of Vanishing/exploding Gradients? • /r/MachineLearning.” *Reddit*. Accessed March 1, 2016. https://www.reddit.com/r/MachineLearning/comments/34piyi/why_can_constant_error_carousels_cecs_prevent/.

How – 2012, Pascanu et al.



In Hochreiter and Schmidhuber (1997); Graves *et al.* (2009) a solution is proposed for the vanishing gradients problem, where the structure of the model is changed. Specifically it introduces a special set of units called LSTM units which are linear and have a recurrent connection to itself which is fixed to 1. The flow of information into the unit and from the unit is guarded by an input and output gates (their behaviour is learned). There are several variations of this basic structure. This solution does not address explicitly the exploding gradients problem.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the Difficulty of Training Recurrent Neural Networks." *arXiv:1211.5063 [cs]*, November 21, 2012. http://arxiv.org/abs/1211.5063.



4

Why can Constant Error Carousels (CECs) prevent LSTM from the problems of vanishing/exploding gradients? (self.MachineLearning)

submitted 10 months ago by freemind2009

5 comments share save hide give gold report

all 5 comments

sorted by: best

[–] sielsteinmodel 5 points 10 months ago*

To understand this, you will have to go through some math. The most accessible article wrt recurrent gradient problems IMHO is Pascanu's ICML2013 paper [1].

A summary: vanishing/exploding gradient comes from the repeated application of the recurrent weight matrix [2]. That the spectral radius of the recurrent weight matrix is bigger than 1 makes exploding gradients *possible* (it is a necessary condition), while a spectral radius smaller than 1 makes it vanish, which is a sufficient condition.

Now, if gradients vanish, that does not mean that all gradients vanish. Only some of them, gradient information local in time will still be present. That means, you might still have a non-zero gradient--but it will not contain long term information. That's because some gradient g + 0 is still g.

If gradients explode, all of them do. That is because some gradient g + infinity is infinity.

That is the reason why LSTM **does not protect you from exploding gradients**, since LSTM also uses a recurrent weight matrix, not only internal state-to-state connections. Successful LSTM applications typically use gradient clipping.

LSTM overcomes the vanishing gradient problem, though. That is because if you look at the derivative of the internal state at T to the internal state at T-1, there is no repeated weight application. The derivative actually is the value of the forget gate. And to avoid that this becomes zero, we need to initialise it properly in the beginning.

That makes it clear why the states can act as "a wormhole through time", because they can bridge long time lags and then (if the time is right) "re inject" it into the other parts of the net by opening the output gate.

[1] Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." arXiv preprint arXiv:1211.5063 (2012).

[2] It might "vanish" also due to saturating nonlinearities, but that is sth that can also happen in shallow nets and can be overcome with more careful weight initialisations.

permalink save give gold

How – Exploding Gradients

$$\begin{aligned}
\frac{dL(t)}{ds_i(t)} &= \frac{dL(t)}{dh_i(t)}\frac{dh_i(t)}{ds_i(t)} + \frac{dL(t)}{dh_i(t+1)}\frac{dh_i(t+1)}{ds_i(t)} \\
&= \frac{dL(t)}{dh_i(t)}\frac{dh_i(t)}{ds_i(t)} + \frac{dL(t+1)}{dh_i(t+1)}\frac{dh_i(t+1)}{ds_i(t)} \\
&= \frac{dL(t)}{dh_i(t)}\frac{dh_i(t)}{ds_i(t)} + \frac{dL(t+1)}{ds_i(t)} \\
&= \frac{dL(t)}{dh_i(t)}\frac{dh_i(t)}{ds_i(t)} + [\texttt{top\_diff\_s}]_i .
\end{aligned}$$

“Simple LSTM.” Accessed March 1, 2016. http://nicodjimenez.github.io/2014/08/08/lstm.html.

How – Example

Inputs

x_t	0.1	0.9
h_t-1	0.6	0.4
c_t-1	0.5	0.4
T	0	1
learning rate	0.1	

Network Weights

		0->0	**1->0**	**0->1**	**1->1**
W	**c**	[0.8	0.6]	[0.1	0.5]
	i	[0.9	0.2]	[0.8	0.3]
	f	[0.1	0.3]	[0.7	0.8]
	o	[0.1	0.8]	[0.2	0.1]

		0->0	**1->0**	**0->1**	**1->1**
U	**c**	[0.2	0.4]	[0.7	0.4]
	i	[0.1	0.2]	[0.3	0.9]
	f	[0.7	0.5]	[0.2	0.5]
	o	[0.8	0.3]	[0.9	0.7]

How – Forward Pass: Equations

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$$

$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$$

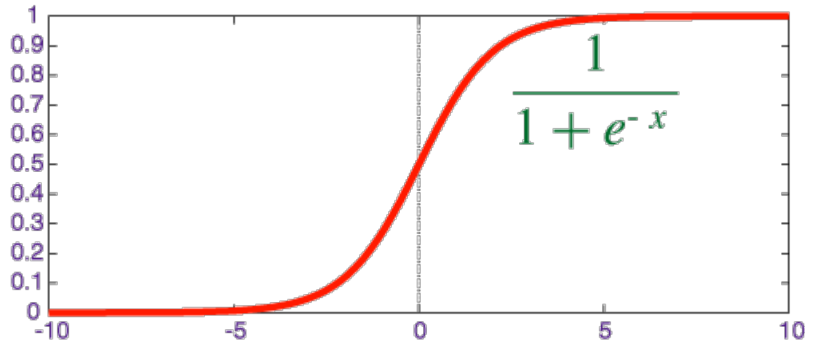
$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$$

$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$$

$$c^t = i^t \odot a^t + f^t \odot c^{t-1}$$

$$c^{t-1} \rightarrow c^t$$

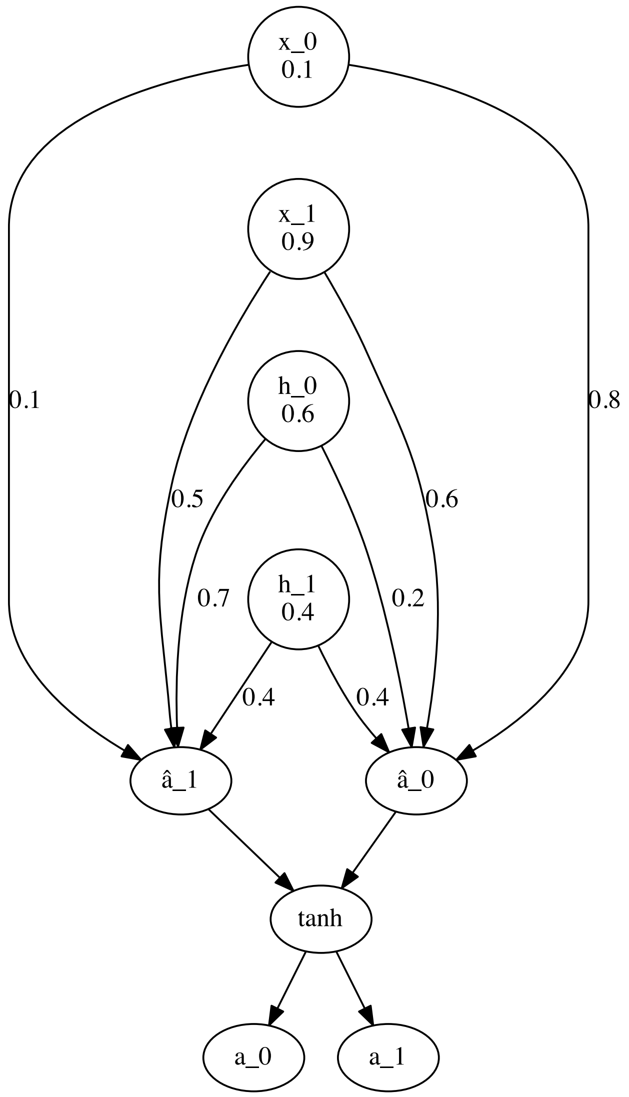
$$h^t = o^t \odot \tanh(c^t)$$



“LSTM.” Accessed February 20, 2016. http://arunmallya.github.io/writeups/nn/lstm/index.html#/.

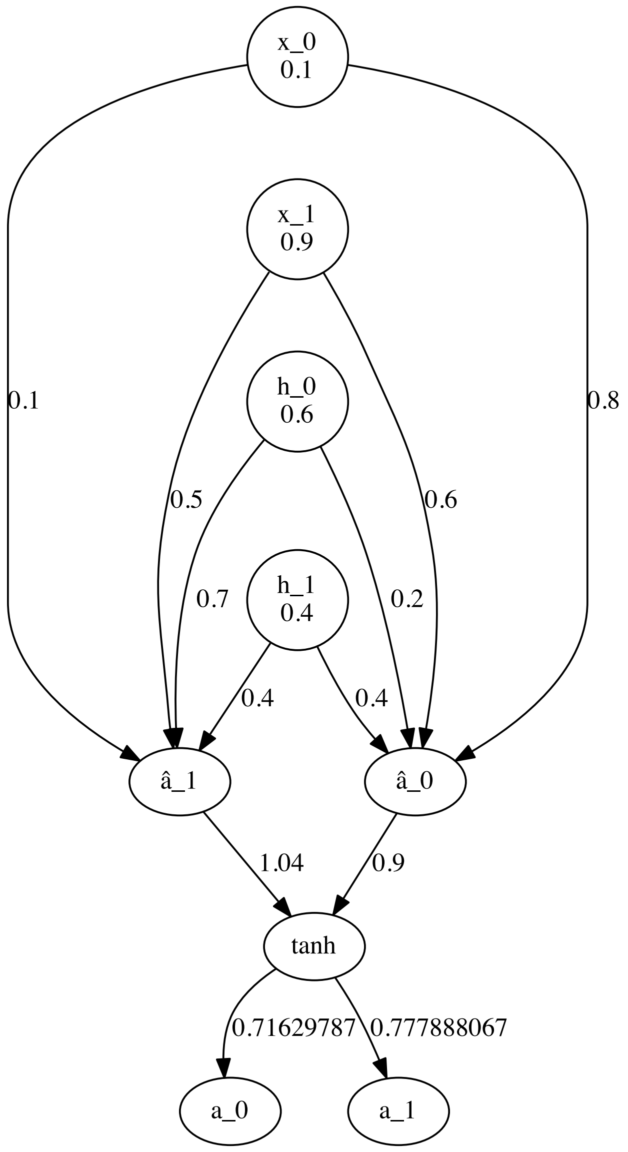
How – Forward Pass

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$$
$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$$
$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$$
$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$$



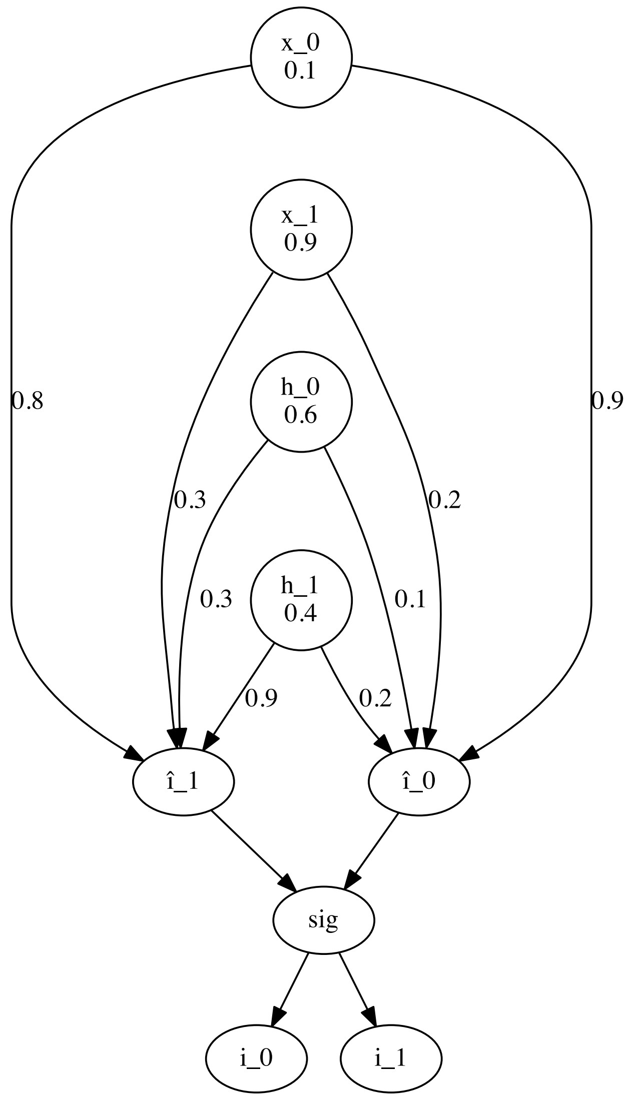
How – Forward Pass

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$$
$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$$
$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$$
$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$$



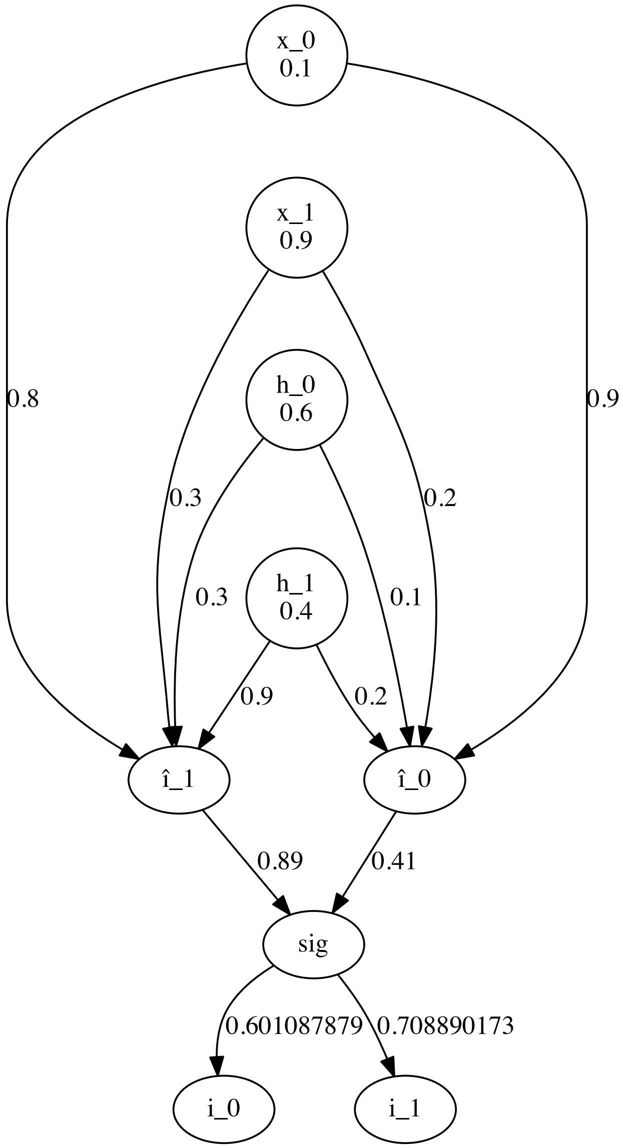
How – Forward Pass

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$$
$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$$
$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$$
$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$$



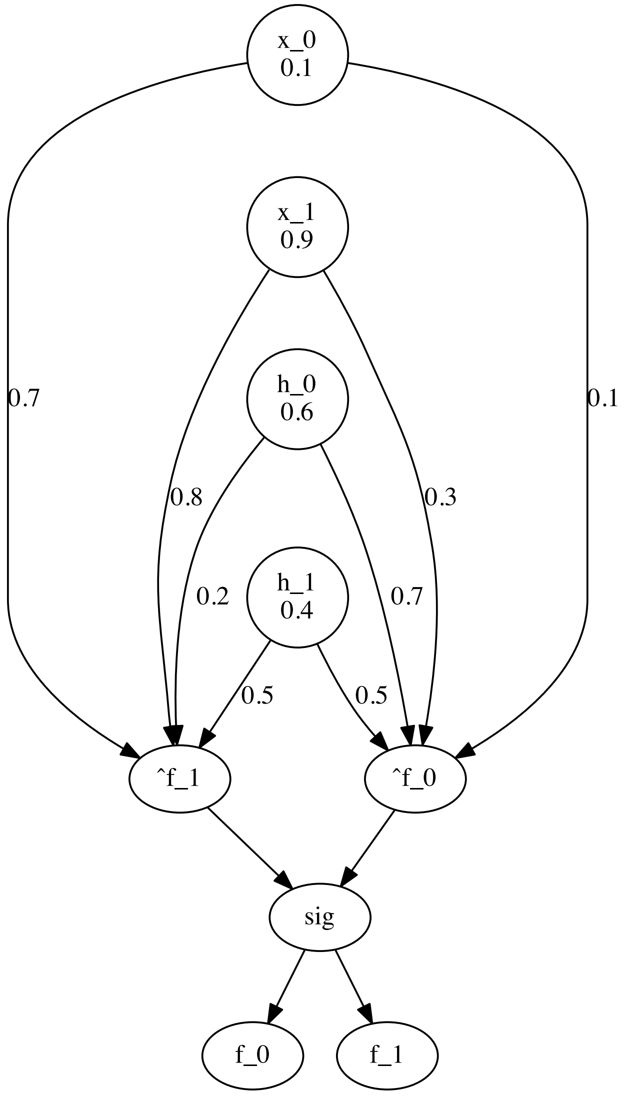
How – Forward Pass

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$$
$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$$
$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$$
$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$$



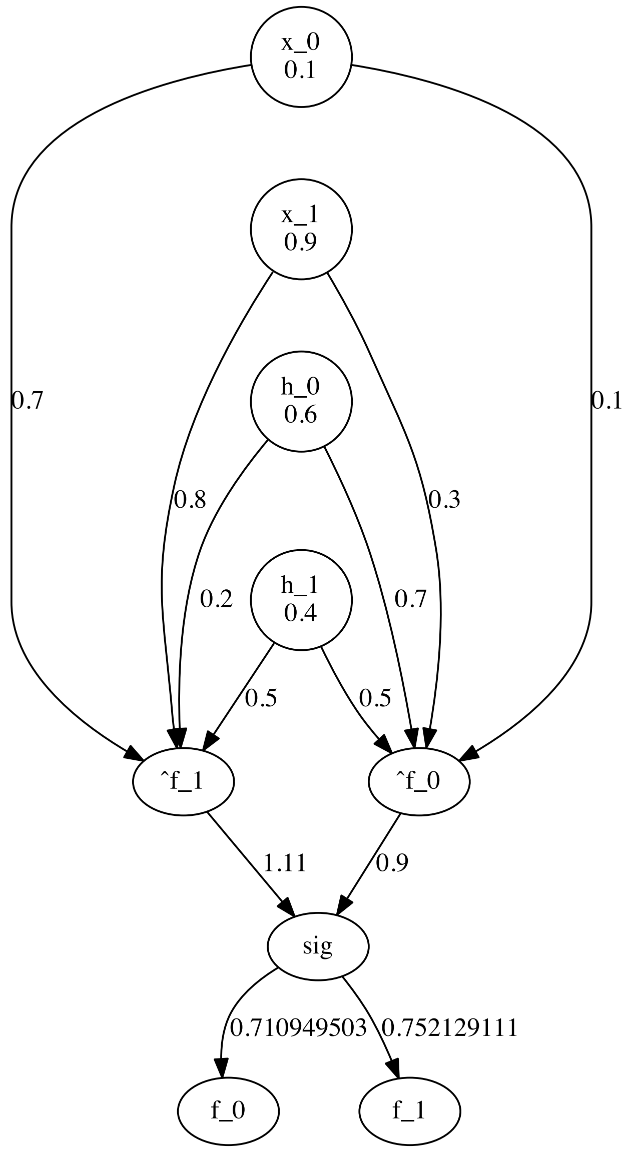
How – Forward Pass

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$$
$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$$
$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$$
$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$$



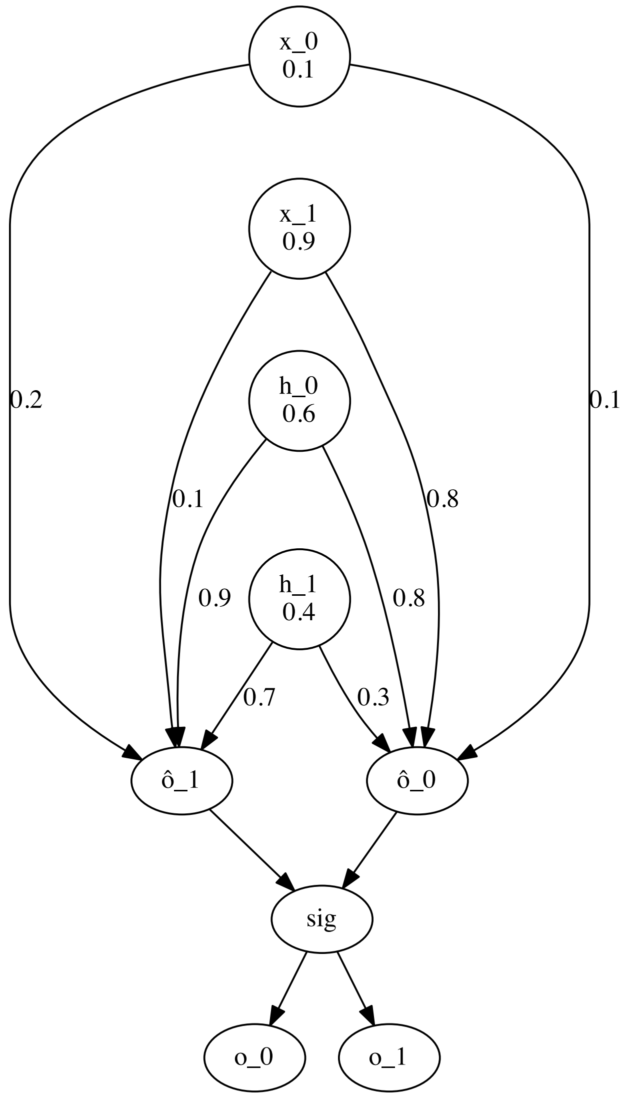
How – Forward Pass

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$$
$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$$
$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$$
$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$$



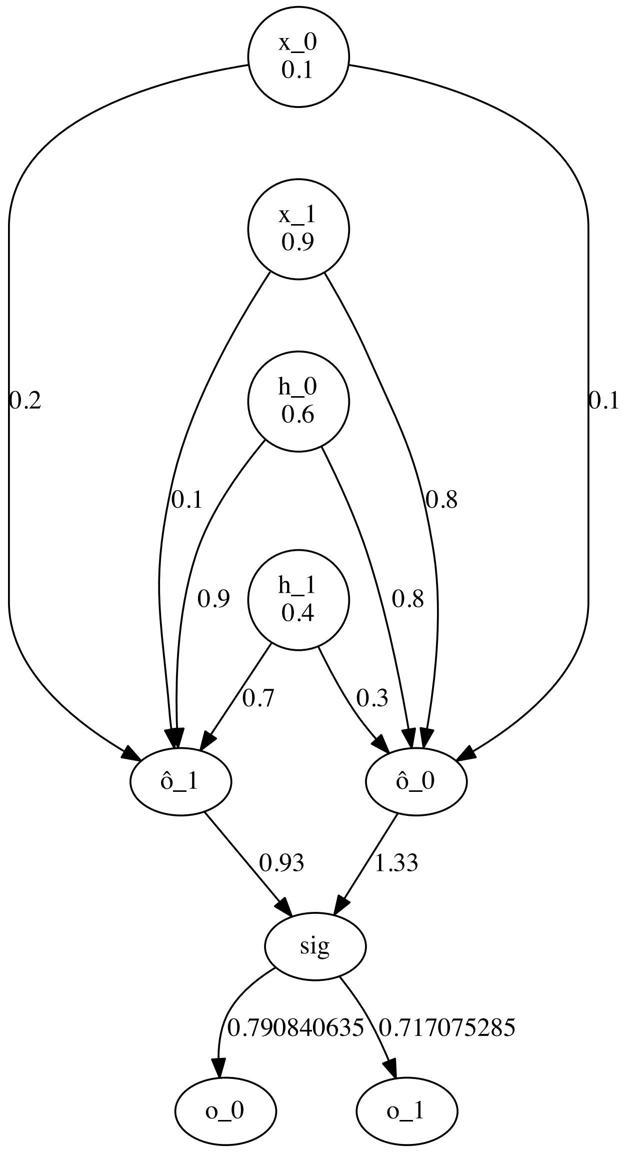
How – Forward Pass

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$$
$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$$
$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$$
$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$$



How – Forward Pass

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$$
$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$$
$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$$
$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$$



How – Forward Pass

$$c^t = i^t \odot a^t + f^t \odot c^{t-1}$$

$$h^t = o^t \odot \tanh(c^t)$$

How – Forward Pass

$$c^t = i^t \odot a^t + f^t \odot c^{t-1}$$

$$h^t = o^t \odot \tanh(c^t)$$

	0	1
c_t	0.786032719	0.85228885
h_t	0.518914596	0.496404923

How – Forward Pass

	0	1
â	0.9	1.04
î	0.41	0.89
ˆf	0.9	1.11
ô	1.33	0.93

		0	**1**
a	**tanh(a_hat)**	0.71629787	0.777888067
i	**sig(i_hat)**	0.601087879	0.708890173
f	**sig(f_hat)**	0.710949503	0.752129111
o	**sig(o_hat)**	0.790840635	0.717075285

	0	**1**
c_t	0.786032719	0.85228885
h_t	0.518914596	0.496404923

How – Backward Pass: Equations

$$\frac{\partial E}{\partial c_i^t} = \frac{\partial E}{\partial h_i^t} \cdot \frac{\partial h_i^t}{\partial c_i^t}$$
$$= \delta h_i^t \cdot o_i^t \cdot (1 - \tanh^2(c_i^t))$$
$$\therefore \delta c^t = \delta h^t \odot o^t \odot (1 - \tanh^2(c^t))$$

$$\frac{\partial E}{\partial o_i^t} = \frac{\partial E}{\partial h_i^t} \cdot \frac{\partial h_i^t}{\partial o_i^t}$$
$$= \delta h_i^t \cdot \tanh(c_i^t)$$
$$\therefore \delta o^t = \delta h^t \odot \tanh(c^t)$$

How – Backward Pass

∂h_t	**∂E/∂h_t**
∂o_t	**∂E/∂o_t**
∂c_t	**∂E/∂c_t**

$$\begin{aligned}\frac{\partial E}{\partial c_i^t} &= \frac{\partial E}{\partial h_i^t} \cdot \frac{\partial h_i^t}{\partial c_i^t} \\ &= \delta h_i^t \cdot o_i^t \cdot (1 - \tanh^2(c_i^t)) \\ \therefore \delta c^t &= \delta h^t \odot o^t \odot (1 - \tanh^2(c^t))\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial o_i^t} &= \frac{\partial E}{\partial h_i^t} \cdot \frac{\partial h_i^t}{\partial o_i^t} \\ &= \delta h_i^t \cdot \tanh(c_i^t) \\ \therefore \delta o^t &= \delta h^t \odot \tanh(c^t)\end{aligned}$$

How – Backward Pass

∂h_t	**∂E/∂h_t**	0.518914596	-0.503595077
∂o_t	**∂E/∂o_t**	0.340488773	-0.348620404
∂c_t	**∂E/∂c_t**	0.233694154	-0.188058699

How – Backward Pass: Equations

$$\begin{aligned} \frac{\partial E}{\partial i_i^t} &= \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial i_i^t} \\ &= \delta c_i^t \cdot a_i^t \\ \therefore \delta i^t &= \delta c^t \odot a^t \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial f_i^t} &= \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial f_i^t} \\ &= \delta c_i^t \cdot c_i^{t-1} \\ \therefore \delta f^t &= \delta c^t \odot c^{t-1} \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial a_i^t} &= \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial a_i^t} \\ &= \delta c_i^t \cdot i_i^t \\ \therefore \delta a^t &= \delta c^t \odot i^t \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial c_i^{t-1}} &= \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}} \\ &= \delta c_i^t \cdot f_i^t \\ \therefore \delta c^{t-1} &= \delta c^t \odot f^t \end{aligned}$$

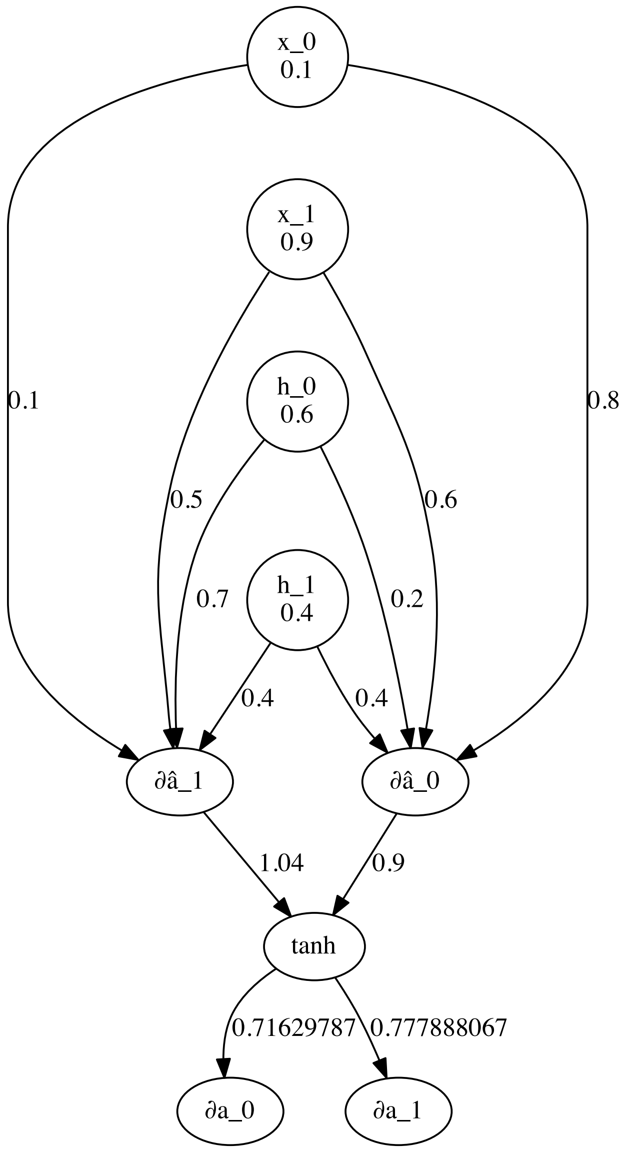
How – Backward Pass

$$\frac{\partial E}{\partial i_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial i_i^t}$$
$$= \delta c_i^t \cdot a_i^t$$
$$\therefore \delta i^t = \delta c^t \odot a^t$$

$$\frac{\partial E}{\partial f_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial f_i^t}$$
$$= \delta c_i^t \cdot c_i^{t-1}$$
$$\therefore \delta f^t = \delta c^t \odot c^{t-1}$$

$$\frac{\partial E}{\partial a_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial a_i^t}$$
$$= \delta c_i^t \cdot i_i^t$$
$$\therefore \delta a^t = \delta c^t \odot i^t$$

$$\frac{\partial E}{\partial c_i^{t-1}} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}}$$
$$= \delta c_i^t \cdot f_i^t$$
$$\therefore \delta c^{t-1} = \delta c^t \odot f^t$$



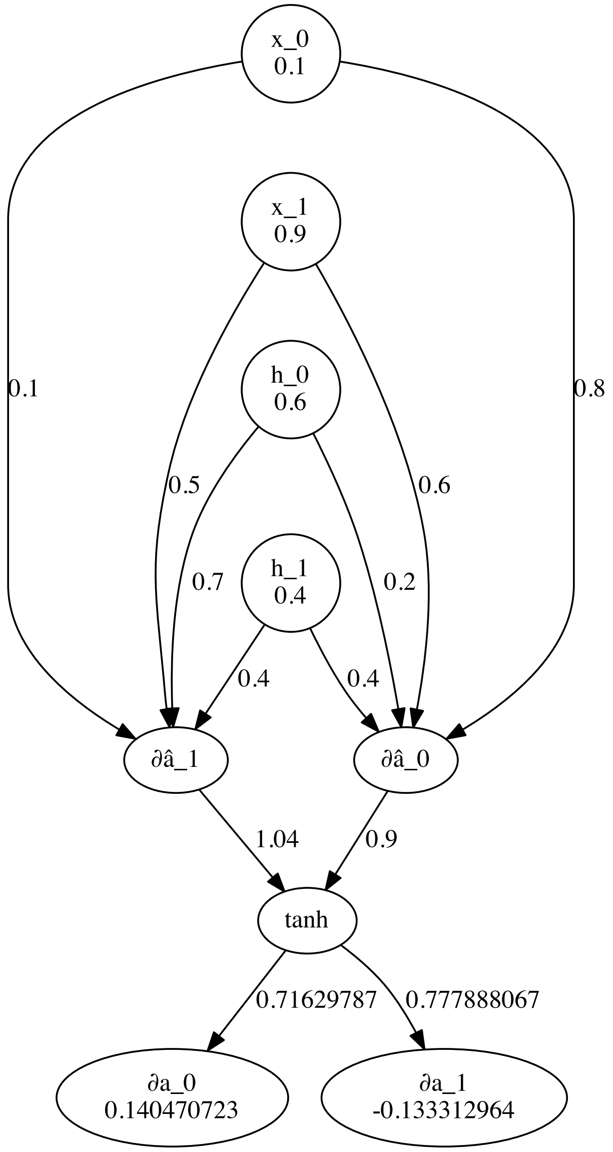
How – Backward Pass

$$\frac{\partial E}{\partial i_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial i_i^t} = \delta c_i^t \cdot a_i^t \quad \therefore \delta i^t = \delta c^t \odot a^t$$

$$\frac{\partial E}{\partial f_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial f_i^t} = \delta c_i^t \cdot c_i^{t-1} \quad \therefore \delta f^t = \delta c^t \odot c^{t-1}$$

$$\frac{\partial E}{\partial a_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial a_i^t} = \delta c_i^t \cdot i_i^t \quad \therefore \delta a^t = \delta c^t \odot i^t$$

$$\frac{\partial E}{\partial c_i^{t-1}} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}} = \delta c_i^t \cdot f_i^t \quad \therefore \delta c^{t-1} = \delta c^t \odot f^t$$



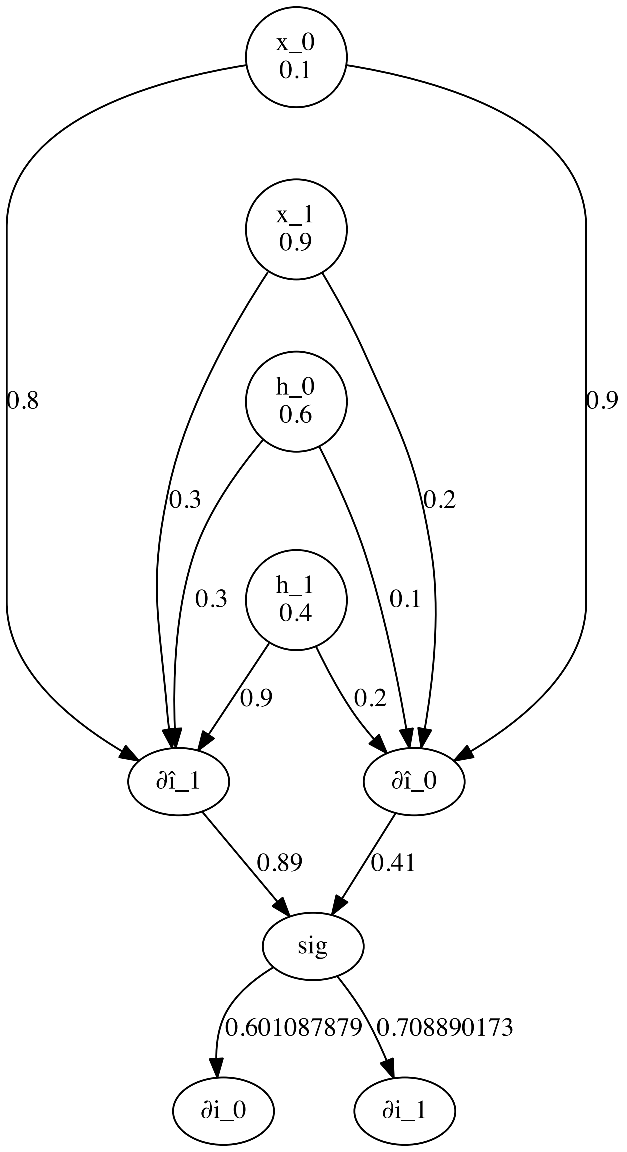
How – Backward Pass

$$\frac{\partial E}{\partial i_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial i_i^t} = \delta c_i^t \cdot a_i^t \qquad \therefore \delta i^t = \delta c^t \odot a^t$$

$$\frac{\partial E}{\partial f_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial f_i^t} = \delta c_i^t \cdot c_i^{t-1} \qquad \therefore \delta f^t = \delta c^t \odot c^{t-1}$$

$$\frac{\partial E}{\partial a_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial a_i^t} = \delta c_i^t \cdot i_i^t \qquad \therefore \delta a^t = \delta c^t \odot i^t$$

$$\frac{\partial E}{\partial c_i^{t-1}} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}} = \delta c_i^t \cdot f_i^t \qquad \therefore \delta c^{t-1} = \delta c^t \odot f^t$$



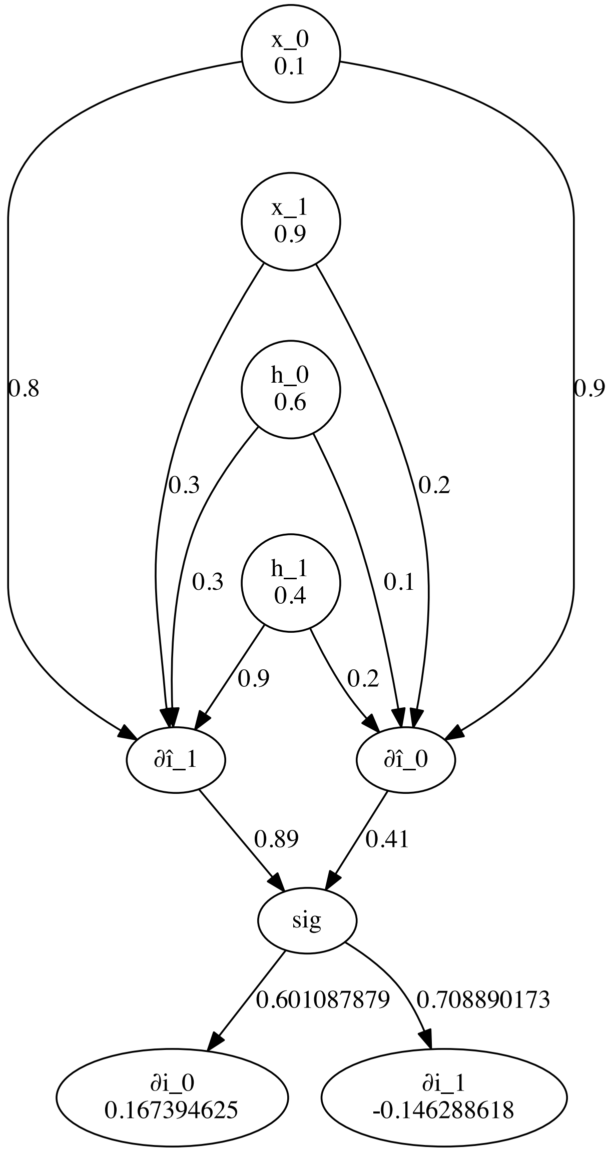
How – Backward Pass

$$\frac{\partial E}{\partial i_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial i_i^t}$$
$$= \delta c_i^t \cdot a_i^t$$
$$\therefore \delta i^t = \delta c^t \odot a^t$$

$$\frac{\partial E}{\partial f_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial f_i^t}$$
$$= \delta c_i^t \cdot c_i^{t-1}$$
$$\therefore \delta f^t = \delta c^t \odot c^{t-1}$$

$$\frac{\partial E}{\partial a_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial a_i^t}$$
$$= \delta c_i^t \cdot i_i^t$$
$$\therefore \delta a^t = \delta c^t \odot i^t$$

$$\frac{\partial E}{\partial c_i^{t-1}} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}}$$
$$= \delta c_i^t \cdot f_i^t$$
$$\therefore \delta c^{t-1} = \delta c^t \odot f^t$$



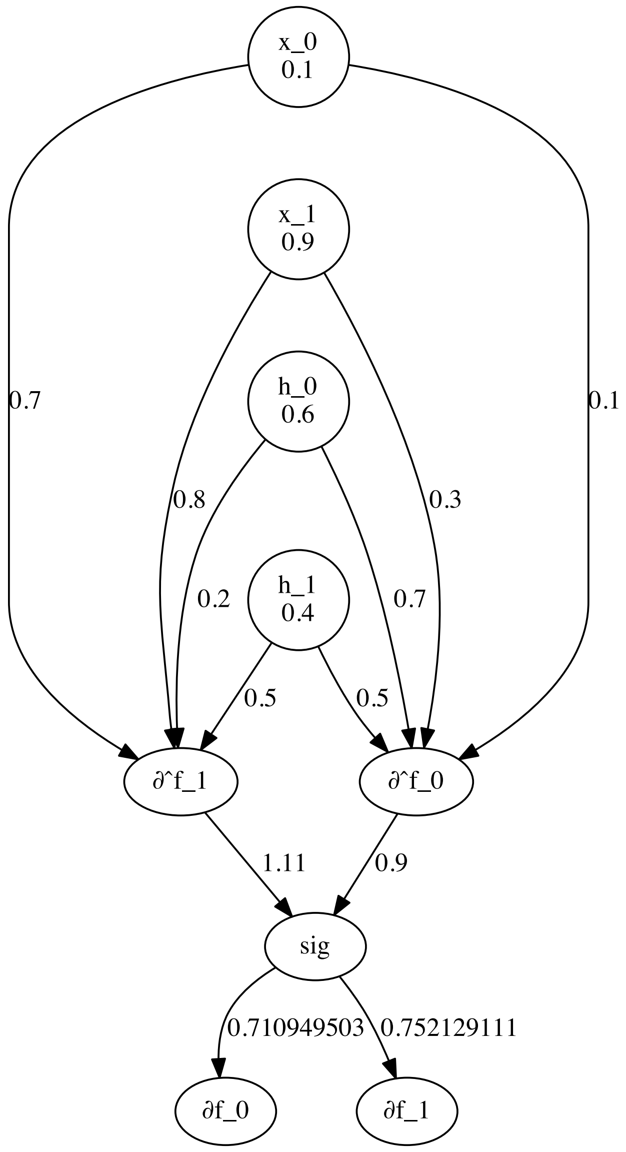
How – Backward Pass

$$\frac{\partial E}{\partial i_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial i_i^t}$$
$$= \delta c_i^t \cdot a_i^t$$
$$\therefore \delta i^t = \delta c^t \odot a^t$$

$$\frac{\partial E}{\partial f_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial f_i^t}$$
$$= \delta c_i^t \cdot c_i^{t-1}$$
$$\therefore \delta f^t = \delta c^t \odot c^{t-1}$$

$$\frac{\partial E}{\partial a_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial a_i^t}$$
$$= \delta c_i^t \cdot i_i^t$$
$$\therefore \delta a^t = \delta c^t \odot i^t$$

$$\frac{\partial E}{\partial c_i^{t-1}} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}}$$
$$= \delta c_i^t \cdot f_i^t$$
$$\therefore \delta c^{t-1} = \delta c^t \odot f^t$$



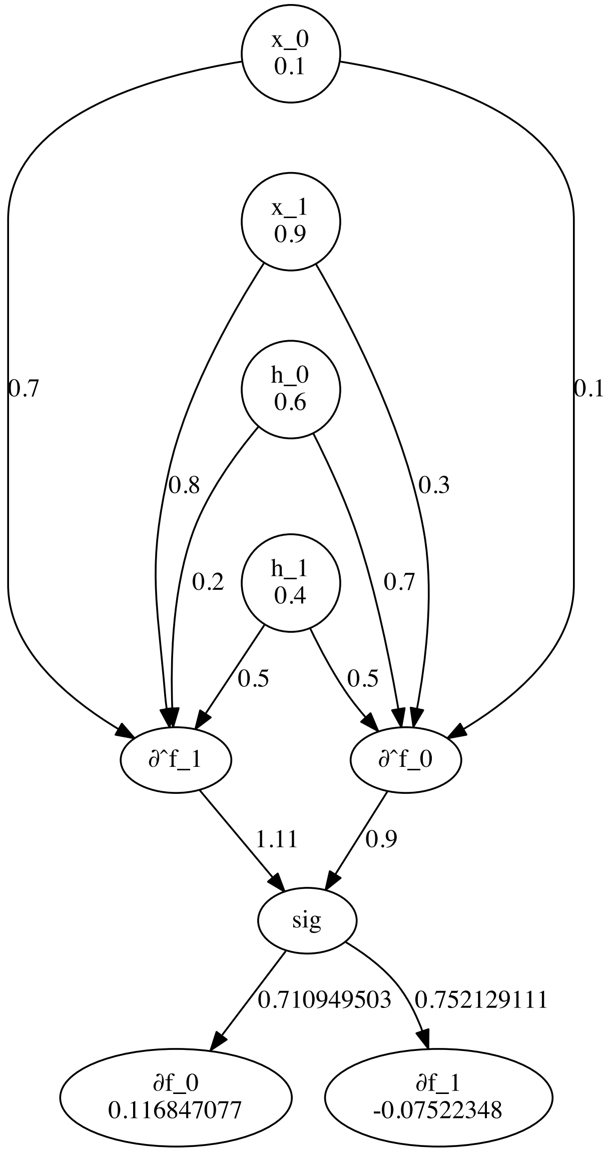
How – Backward Pass

$$\frac{\partial E}{\partial i_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial i_i^t}$$
$$= \delta c_i^t \cdot a_i^t$$
$$\therefore \delta i^t = \delta c^t \odot a^t$$

$$\frac{\partial E}{\partial f_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial f_i^t}$$
$$= \delta c_i^t \cdot c_i^{t-1}$$
$$\therefore \delta f^t = \delta c^t \odot c^{t-1}$$

$$\frac{\partial E}{\partial a_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial a_i^t}$$
$$= \delta c_i^t \cdot i_i^t$$
$$\therefore \delta a^t = \delta c^t \odot i^t$$

$$\frac{\partial E}{\partial c_i^{t-1}} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}}$$
$$= \delta c_i^t \cdot f_i^t$$
$$\therefore \delta c^{t-1} = \delta c^t \odot f^t$$



How – Calculate ∂c_t-1

How – Backward Pass

∂a_t	**∂E/∂a_t**	0.140470723	-0.133312964
∂i_t	**∂E/∂i_t**	0.167394625	-0.146288618
∂f_t	**∂E/∂f_t**	0.116847077	-0.07522348
∂c_t-1	**∂E/∂c_t-1**	0.166144743	-0.141444422

How – Backward Pass: Equations

$$\delta \hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$

$$\delta \hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$

$$\delta \hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$

$$\delta \hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\delta z^t = \left[\delta \hat{a}^t, \delta \hat{i}^t, \delta \hat{f}^t, \delta \hat{o}^t\right]^T$$

How – Backward Pass

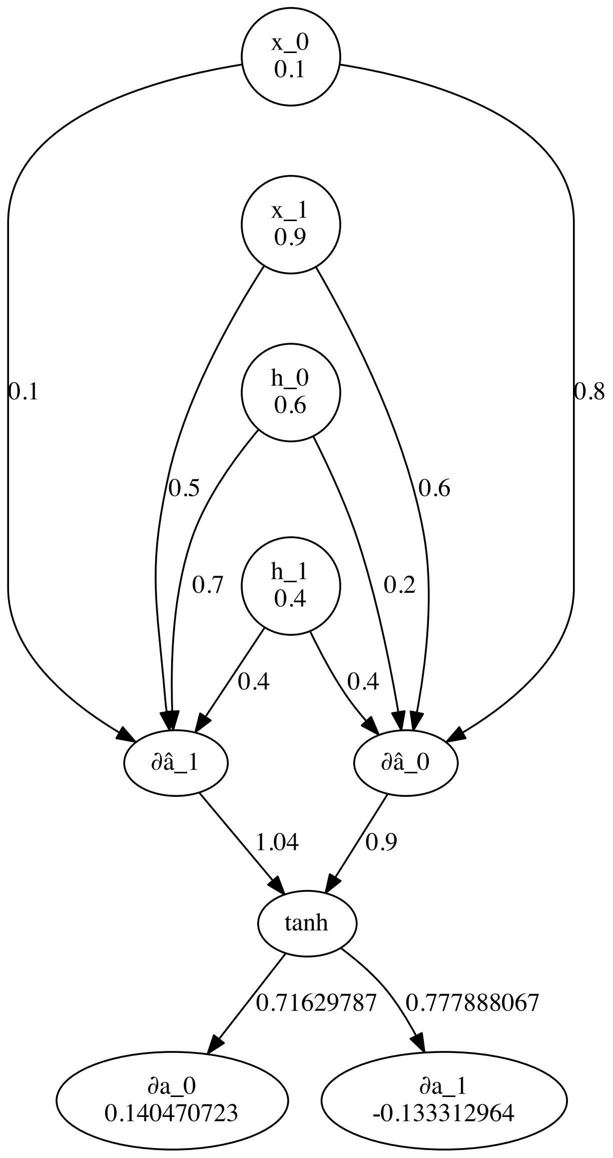
$$\delta\hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$

$$\delta\hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$

$$\delta\hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$

$$\delta\hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\delta z^t = \left[\delta\hat{a}^t, \delta\hat{i}^t, \delta\hat{f}^t, \delta\hat{o}^t\right]^T$$



How – Backward Pass

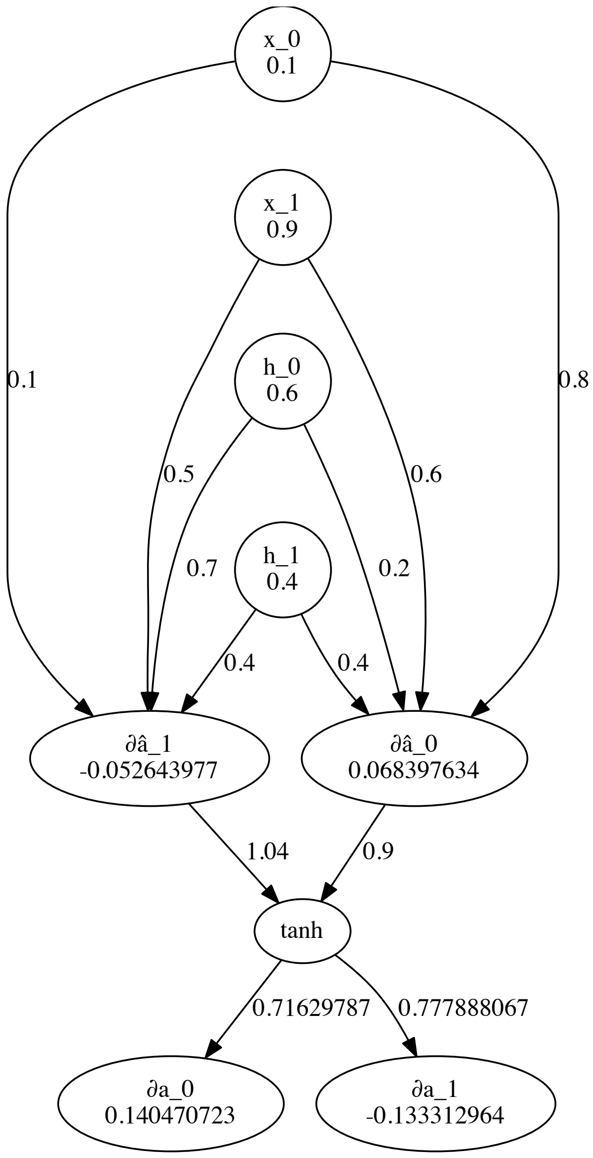
$$\delta\hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$

$$\delta\hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$

$$\delta\hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$

$$\delta\hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\delta z^t = \left[\delta\hat{a}^t, \delta\hat{i}^t, \delta\hat{f}^t, \delta\hat{o}^t\right]^T$$



How – Backward Pass

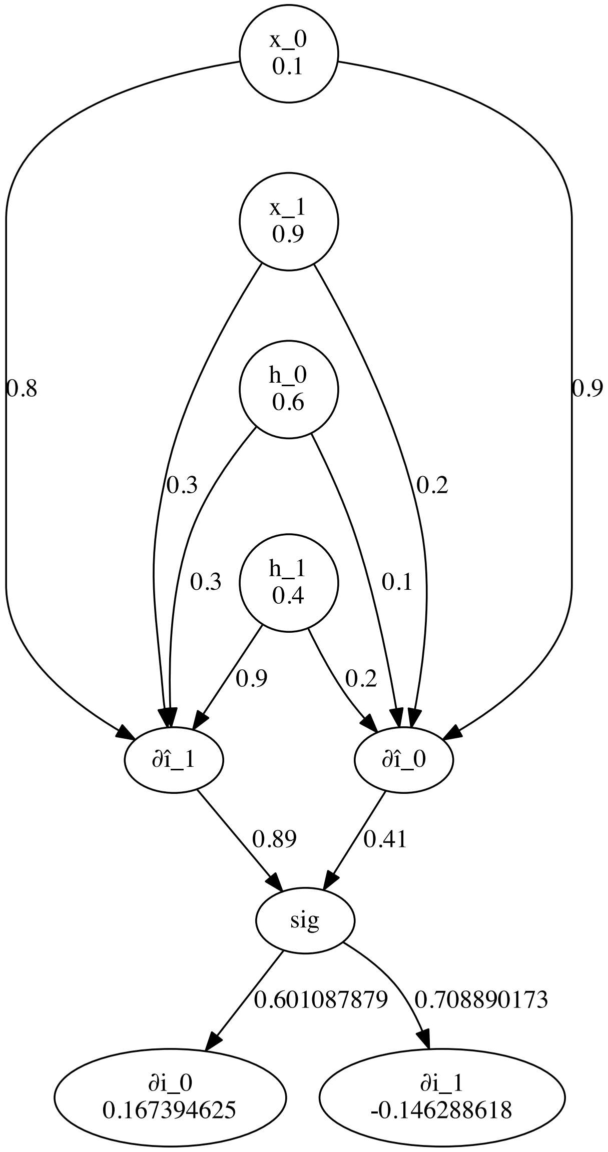
$$\delta\hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$

$$\delta\hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$

$$\delta\hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$

$$\delta\hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\delta z^t = \left[\delta\hat{a}^t, \delta\hat{i}^t, \delta\hat{f}^t, \delta\hat{o}^t\right]^T$$



How – Backward Pass

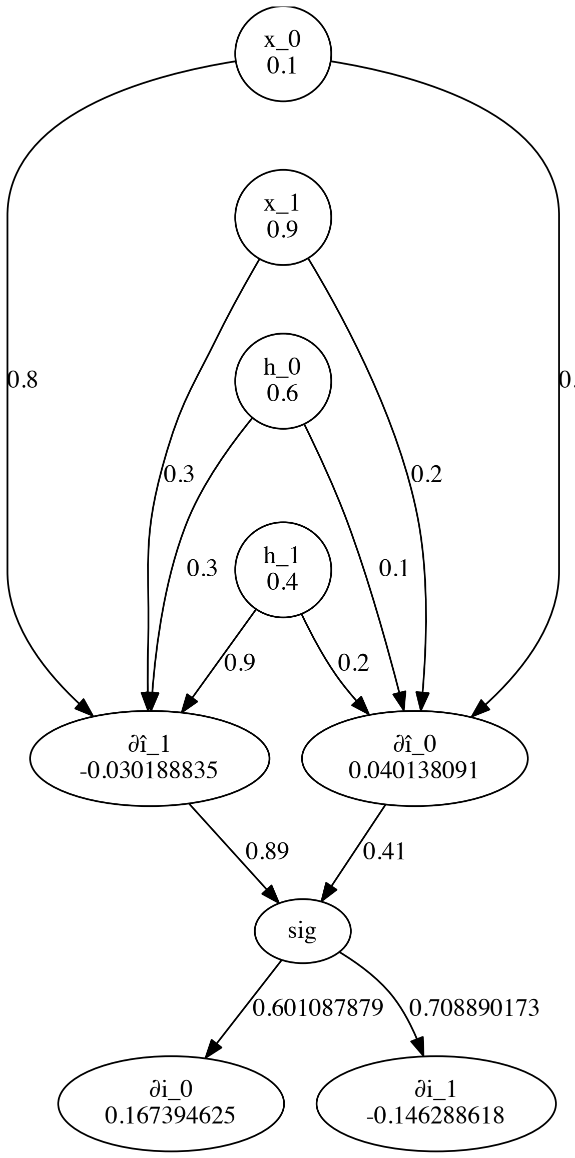
$$\delta\hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$

$$\delta\hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$

$$\delta\hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$

$$\delta\hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\delta z^t = \left[\delta\hat{a}^t, \delta\hat{i}^t, \delta\hat{f}^t, \delta\hat{o}^t\right]^T$$



How – Backward Pass

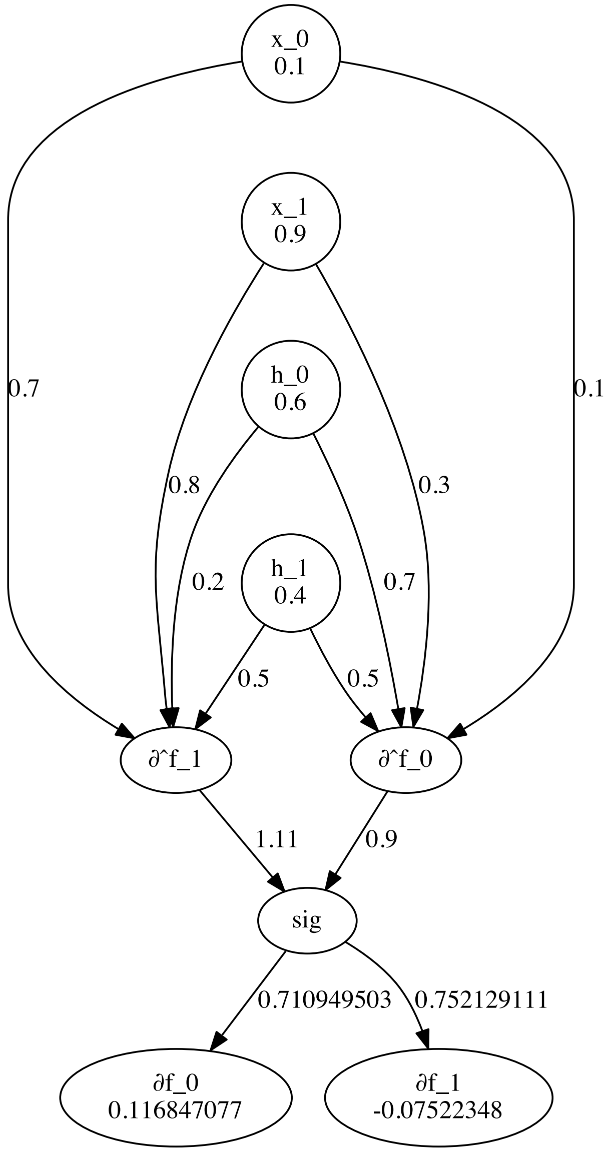
$$\delta \hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$

$$\delta \hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$

$$\delta \hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$

$$\delta \hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\delta z^t = \left[\delta \hat{a}^t, \delta \hat{i}^t, \delta \hat{f}^t, \delta \hat{o}^t\right]^T$$



How – Backward Pass

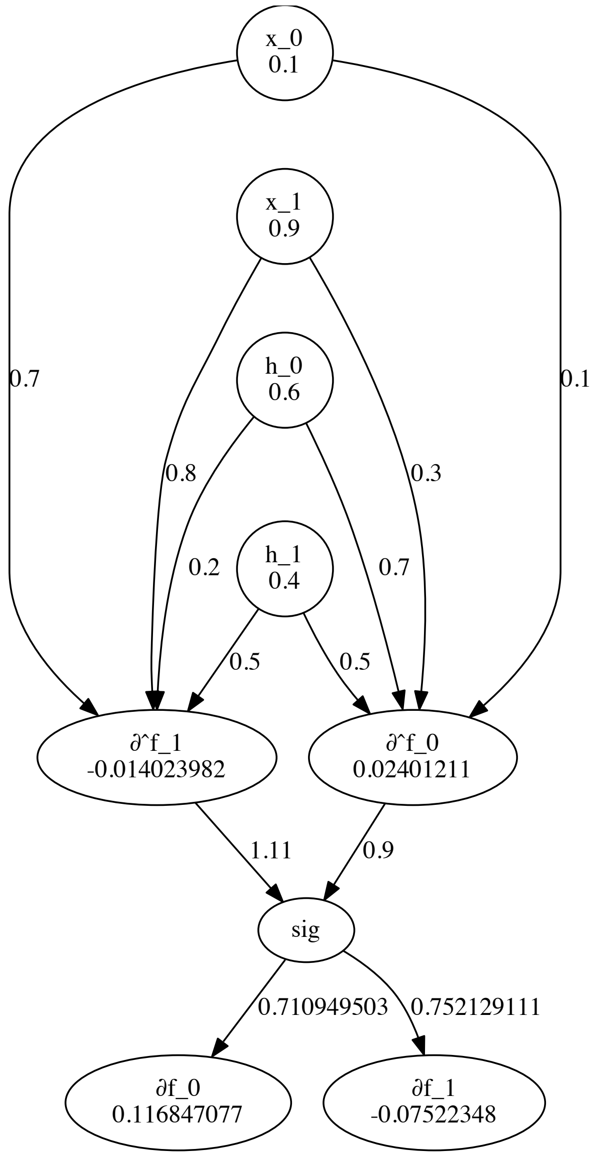
$$\delta \hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$

$$\delta \hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$

$$\delta \hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$

$$\delta \hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\delta z^t = \left[\delta \hat{a}^t, \delta \hat{i}^t, \delta \hat{f}^t, \delta \hat{o}^t\right]^T$$



How – Backward Pass

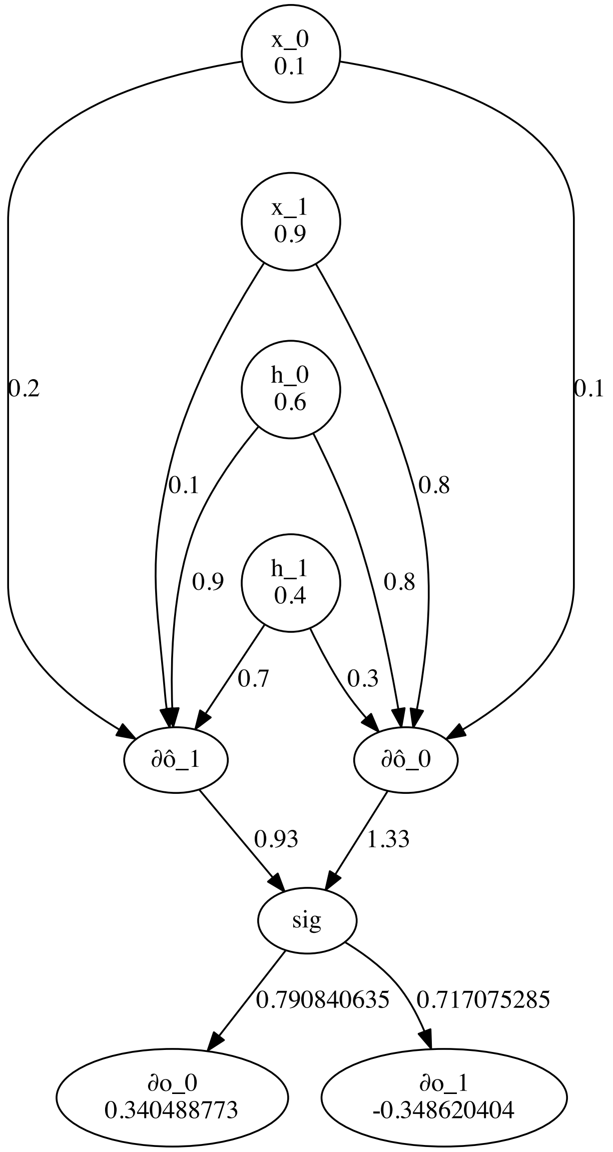
$$\delta\hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$

$$\delta\hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$

$$\delta\hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$

$$\delta\hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\delta z^t = \left[\delta\hat{a}^t, \delta\hat{i}^t, \delta\hat{f}^t, \delta\hat{o}^t\right]^T$$



How – Backward Pass

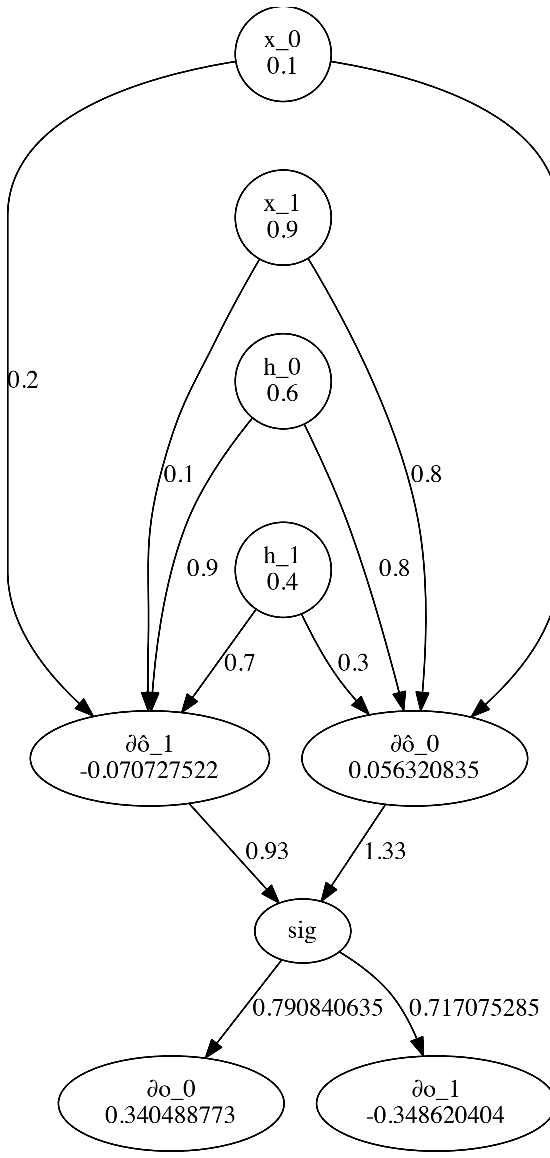
$$\delta\hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$

$$\delta\hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$

$$\delta\hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$

$$\delta\hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\delta z^t = \left[\delta\hat{a}^t, \delta\hat{i}^t, \delta\hat{f}^t, \delta\hat{o}^t\right]^T$$



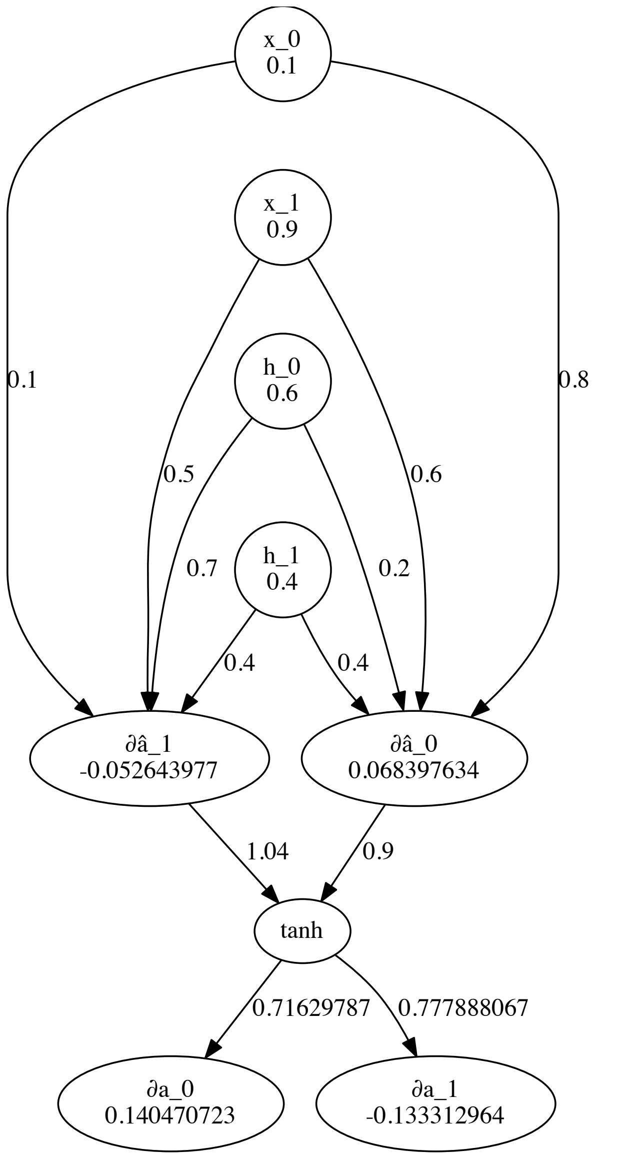
How – Backward Pass

∂z_t	**∂â_t**	0.068397634	-0.052643977
	∂î_t	0.040138091	-0.030188835
	∂ˆf_t	0.02401211	-0.014023982
	∂ô_t	0.056320835	-0.070727522

How – Update Weights

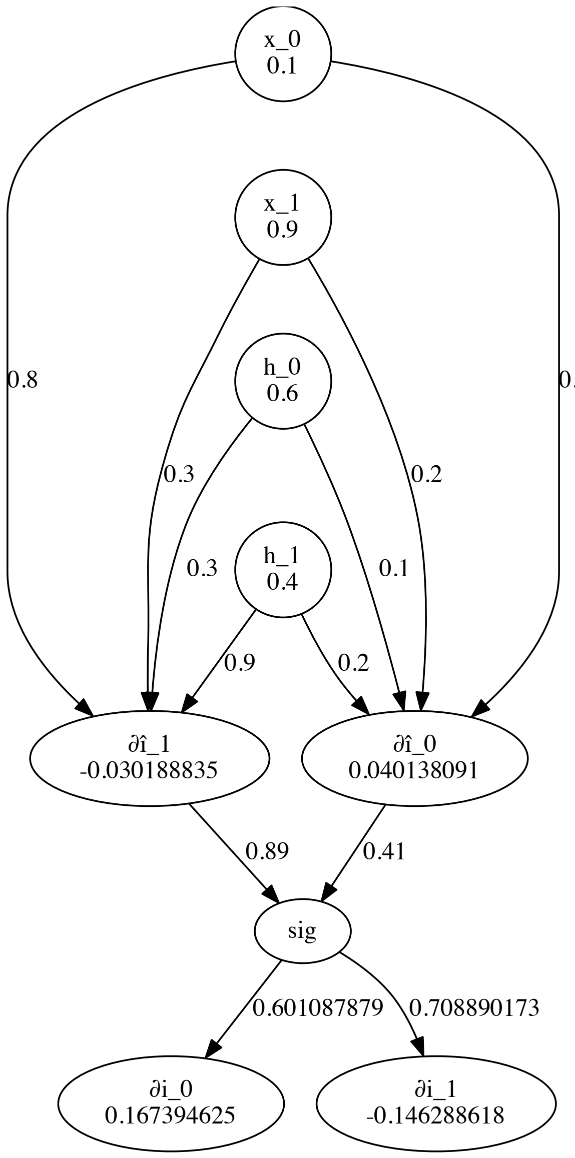
How – Backward Pass

Update Weights



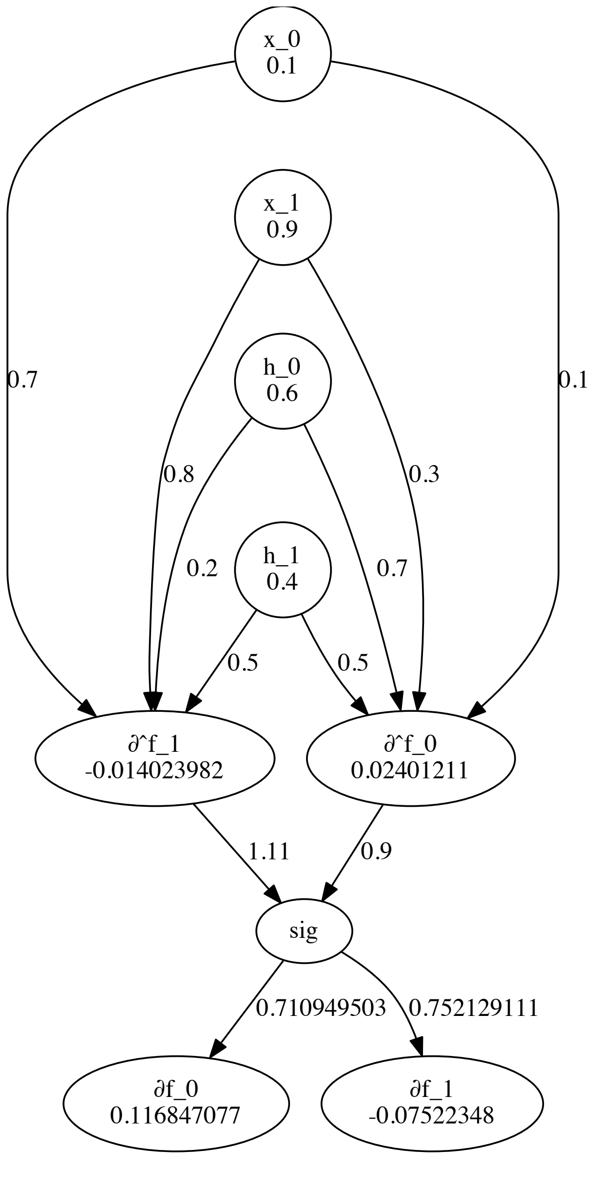
How – Backward Pass

Update Weights



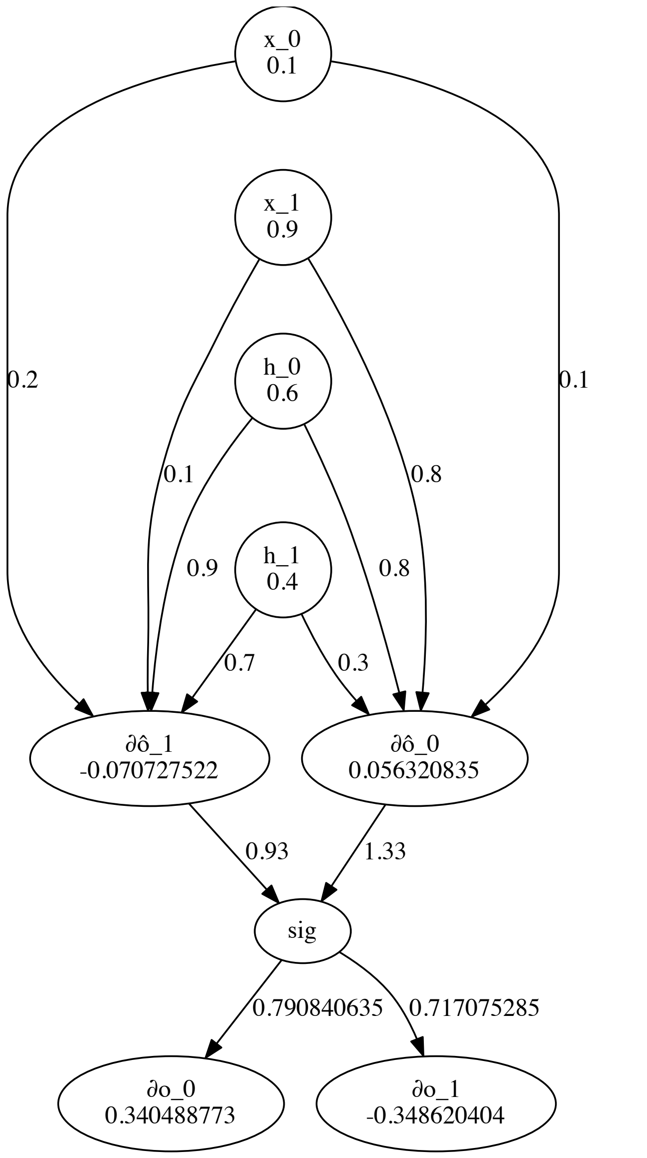
How – Backward Pass

Update Weights



How – Backward Pass

Update Weights



How – Backward Pass

	deltas				
		0	**1**	**0**	**1**
W	**c**	0.006839763	0.061557871	-0.005264398	-0.047379579
	i	0.004013809	0.036124282	-0.003018884	-0.027169952
	f	0.002401211	0.021610899	-0.001402398	-0.012621584
	o	0.005632084	0.050688752	-0.007072752	-0.06365477
U	**c**	0.04103858	0.061557871	-0.031586386	-0.021057591
	i	0.00403134	0.036124282	-0.018113301	-0.012075534
	f	0.001683447	0.021610899	-0.008414389	-0.005609593
	o	0.005614446	0.050688752	-0.042436513	-0.028291009

How – Old Weights

		0->0	1->0	0->1	1->1
W	**c**	[0.8	0.6]	[0.1	0.5]
	i	[0.9	0.2]	[0.8	0.3]
	f	[0.1	0.3]	[0.7	0.8]
	o	[0.1	0.8]	[0.2	0.1]

		0->0	1->0	0->1	1->1
U	**c**	[0.2	0.4]	[0.7	0.4]
	i	[0.1	0.2]	[0.3	0.9]
	f	[0.7	0.5]	[0.2	0.5]
	o	[0.8	0.3]	[0.9	0.7]

How – Backward Pass

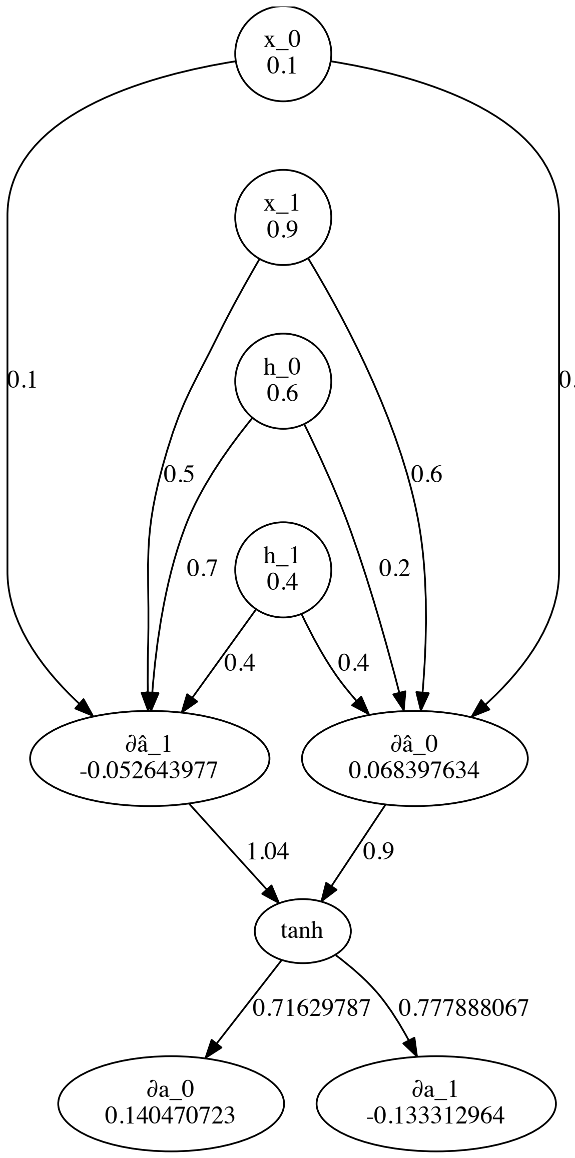
	new weights				
		0	1	0	1
W	c	0.799316024	0.593844213	0.10052644	0.504737958
	i	0.899598619	0.196387572	0.800301888	0.302716995
	f	0.099759879	0.29783891	0.70014024	0.801262158
	o	0.099436792	0.794931125	0.200707275	0.106365477
U	c	0.195896142	0.393844213	0.703158639	0.402105759
	i	0.099596866	0.196387572	0.30181133	0.901207553
	f	0.699831655	0.49783891	0.200841439	0.500560959
	o	0.799438555	0.294931125	0.904243651	0.702829101

How – Previous Timestep

- Need to calculate dE/dx_t and dE/dh_t-1
- Need to calculate dE/c_t+1

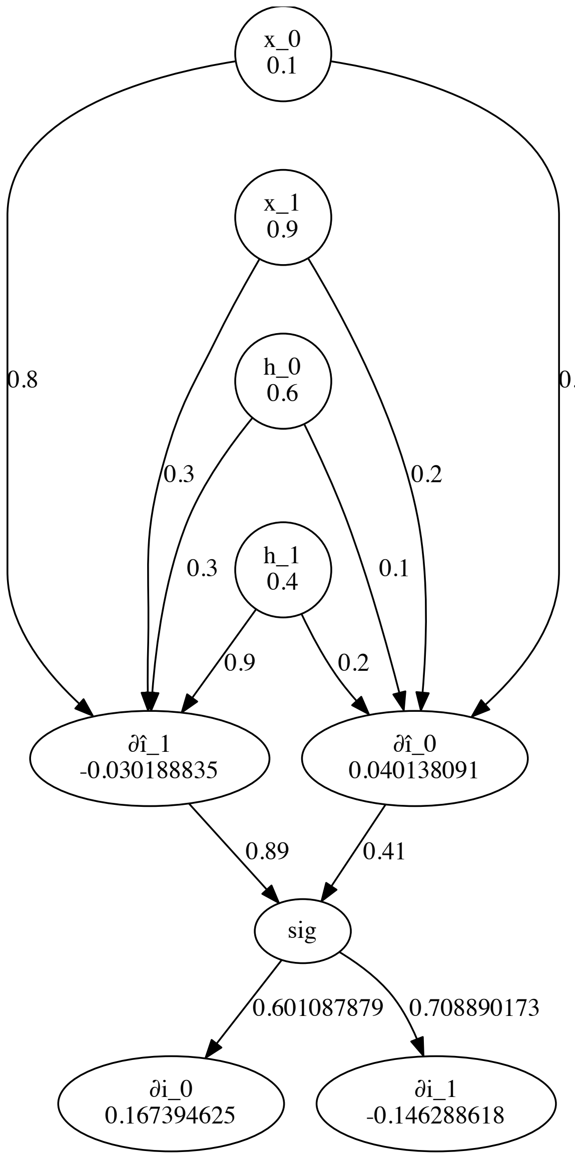
How – Backward Pass

Find error to pass back



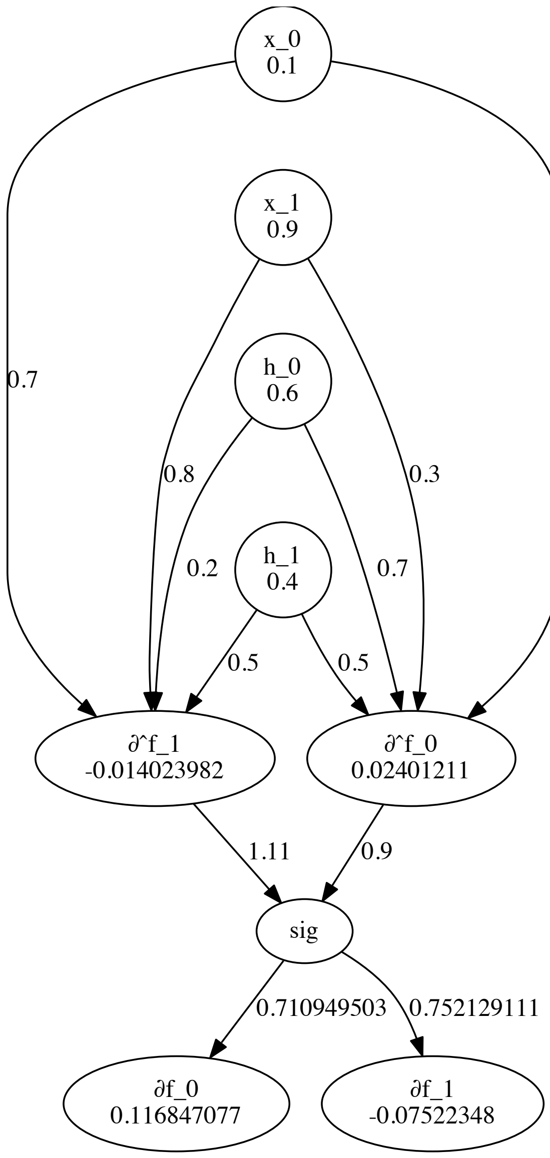
How – Backward Pass

Find error to pass back



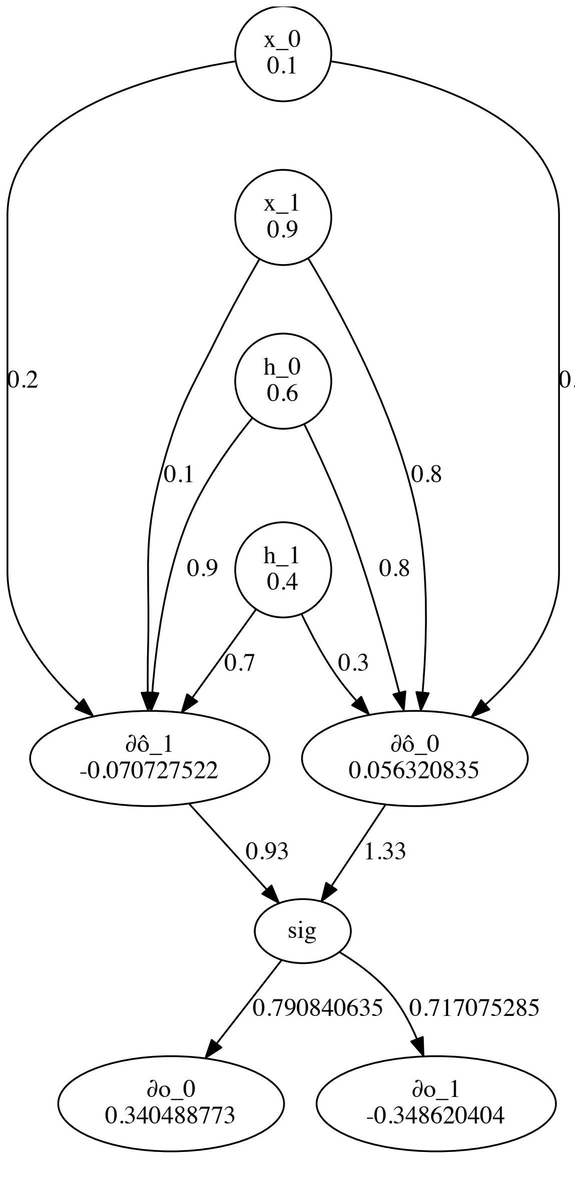
How – Backward Pass

Find error to pass back



How – Backward Pass

Find error to pass back



How – Backward Pass

∂x_t	0.045497926	0.047655923
∂h_t-1	-0.032808519	-0.040459821

How – Backward Pass: Equations

$$\begin{aligned}\frac{\partial E}{\partial c_i^{t-1}} &= \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}} \\ &= \delta c_i^t \cdot f_i^t \\ \therefore\ \delta c^{t-1} &= \delta c^t \odot f^t\end{aligned}$$

How – Backward Pass: Equations

$$\begin{aligned}\frac{\partial E}{\partial c_i^{t-1}} &= \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}} \\ &= \delta c_i^t \cdot f_i^t \\ \therefore \delta c^{t-1} &= \delta c^t \odot f^t\end{aligned}$$

∂c_t-1	0.166144743	-0.141444422

What – Lessons Learned

- Computational Graphs in Theano
- Computing partial derivatives/gradients in Theano is awesome
- Numerical stability ☹

What - Datasets

- Text generation (char-to-char, word-to-word)
  - Book of Mormon
    - English
    - Portuguese
    - Spanish

What – Book of Mormon, c2c

- Seed: and it came to pass
- and it came to pass that the Lord of the vineyard said unto them: behold, I am a descendant of the death of the seed of Joseph, have I prepared for the same being was a men of God, or of our prophecies which I have spoken, which is the season of his words, and I will bring the Father in the name of Jesus.
- Capitals added
- Translation

What – O Livro de Mórmon, c2c

- Seed: e o senhor disse
- e o Senhor dissera a meus irmãos, os lamanitas e os anlicitas nas eras de meu povo que se dirigiram para a terra de meu pai, meus irmãos, que haviam sido mortos para a terra de Néfi, de modo que ele nos deu os que se lembram de seu coração e conservante de sua morte;
- Capitals added
- Translation

What – El Libro de Mormón, c2c

- Seed: y el senhor dice
- y el senhor dice todas las cosas que están escritas de las palabras que habló el ángel del señor de los ejércitos, de modo que no pudieron ser en ellas; y también le dije a su pueblo: esto fue para llevar a causa de la iniquidad de ammoríah por el espacio de muchos días
- Translate

What – Book of Mormon, w2w

- and thus it did come to pass that the people began to wax strong in wickedness and abominations ; and he begat sons and daughters ; and he begat coriantum , and he anointed coriantum to reign in his stead .
- and it came to pass that the lamanites did come down against us to battle .
- and we did gather in the city of gid , when they did come unto the city of night , the city desolation .
- and it was one day that there was no contention among all the people , but insomuch that there was no wars , and rumors of wars ; and every kind that they might read the ways which he desired to murder the people .
- and he did teach them the words which had been spoken by the mouth of all the prophets ; and all the people who were in the land were called by the people of nephi .
- and thus did the people of nephi , throughout all the land of zarahemla , and there was no contention by the nephites .
- and thus they had become weak , because of their wickedness and their abominations . and it came to pass that the people did wax more strong in the land .

What - Translation

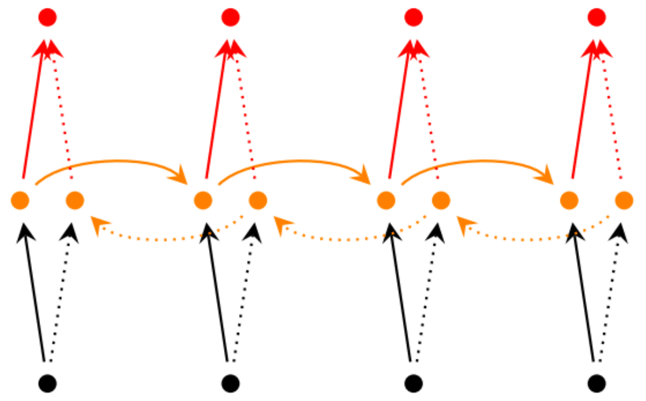
What – Gated Recurrent Units

- Differences from LSTMs:
  - GRU has 2 gates while LSTM has 3 gates
  - GRU's internal memory is completely exposed as output
    - No output gate

"Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano." *WildML*, October 27, 2015. http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/.

What – 2005, Graves and Schmidhuber

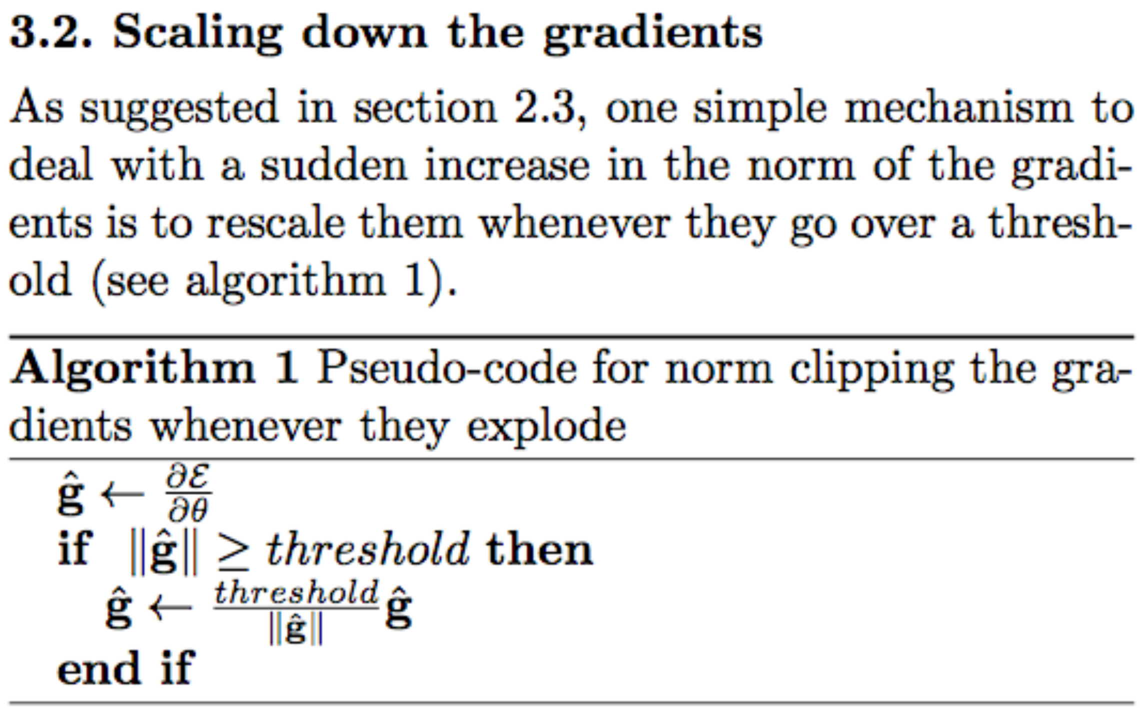
- **Bidirectional LSTM networks**
- Introduced architecture for phoneme classification
- Not useful for online applications



Graves, Alex, and Jürgen Schmidhuber. "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures." *Neural Networks* 18, no. 5–6 (July 2005): 602–10. doi:10.1016/j.neunet.2005.06.042.

"Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs." *WildML*, September 17, 2015. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

What – 2012, Pascanu et al.



3.2. Scaling down the gradients

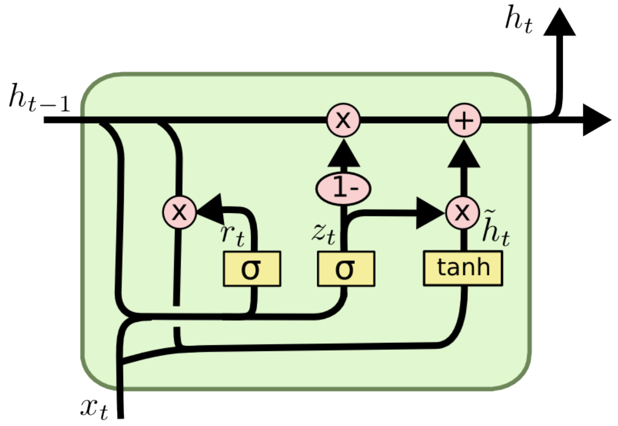
As suggested in section 2.3, one simple mechanism to deal with a sudden increase in the norm of the gradients is to rescale them whenever they go over a threshold (see algorithm 1).

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

```
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ then
    $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if
```

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the Difficulty of Training Recurrent Neural Networks." *arXiv:1211.5063 [cs]*, November 21, 2012. http://arxiv.org/abs/1211.5063.

What – 2014, Cho, et al.



Gated Recurrent Unit

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

“Understanding LSTM Networks -- Colah’s Blog.” Accessed January 25, 2016. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

What – Gated Recurrent Unit

- Reset to normal RNN by setting:
  - Reset gate to all 1s
  - Output gate to all 0s
- GRU only has 2 gates
  - Reset – how to combined previous hidden state and current input
  - Update – how much of the internal memory to keep

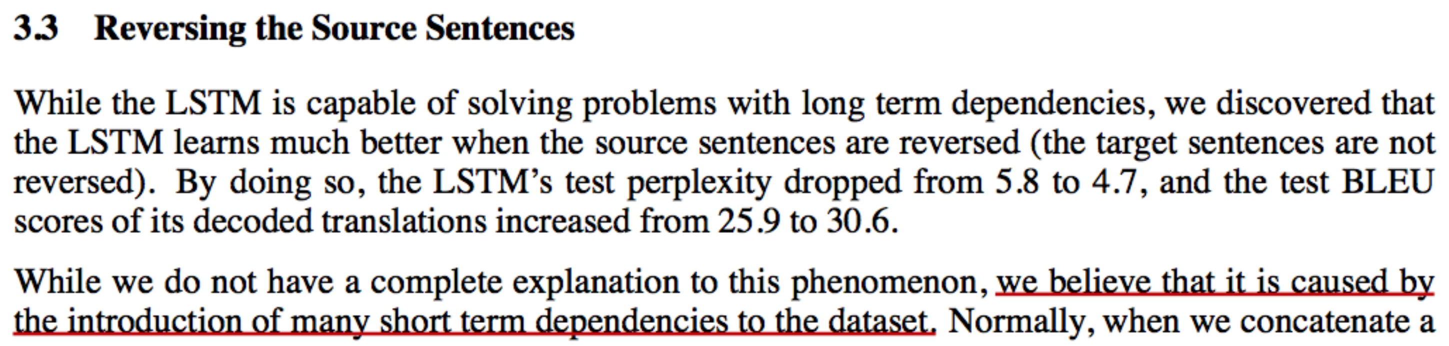
"Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano." *WildML*, October 27, 2015. http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/.

What – LSTM vs. GRU

- Unclear which is better
- GRUs have fewer parameters
  - May train faster and require less data for generalization
- LSTMs are very expressive
  - May require much more data

"Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano." *WildML*, October 27, 2015. http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/.

What – 2014, Sutskever, et al.



3.3 Reversing the Source Sentences

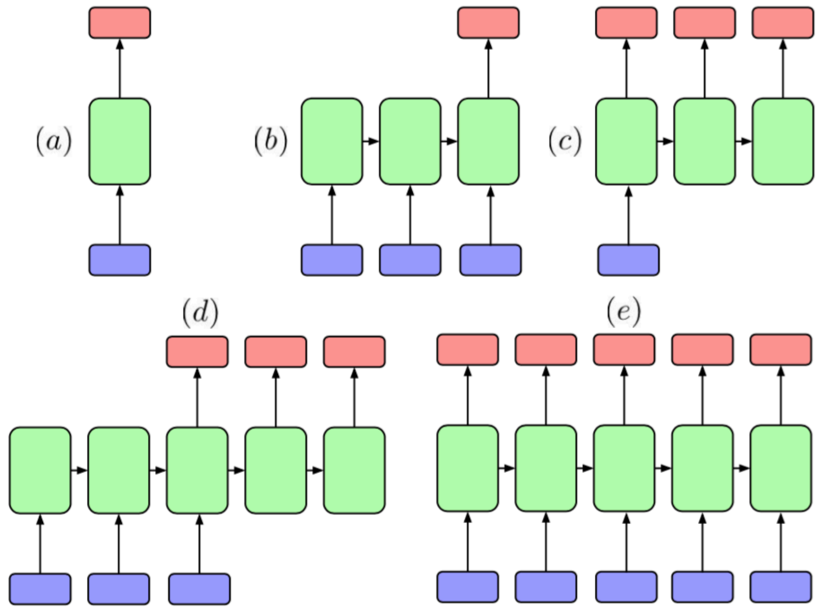
While the LSTM is capable of solving problems with long term dependencies, we discovered that the LSTM learns much better when the source sentences are reversed (the target sentences are not reversed). By doing so, the LSTM's test perplexity dropped from 5.8 to 4.7, and the test BLEU scores of its decoded translations increased from 25.9 to 30.6.

While we do not have a complete explanation to this phenomenon, we believe that it is caused by the introduction of many short term dependencies to the dataset. Normally, when we concatenate a

“NIPS: Oral Session 4 - Ilya Sutskever - Microsoft Research.” Accessed March 1, 2016. http://research.microsoft.com/apps/video/?id=239083.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks.” *arXiv:1409.3215 [cs]*, September 10, 2014. http://arxiv.org/abs/1409.3215.

What – Input/Output Architectures



tire hidden state of the neural network. (a) This is the conventional independent case, as assumed by standard feedforward networks. (b) Text and video classification are tasks in which a sequence is mapped to one fixed length vector. (c) Image captioning presents the converse case, where the input image is a single non-sequential data point. (d) This architecture has been used for natural language translation, a sequence-to-sequence task in which the two sequences may have varying and different lengths. (e) This architecture has been used to learn a generative model for text, predicting at each step the following character.

Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning." *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.

Bibliography

- “(1) How Does LSTM Help Prevent the Vanishing (and Exploding) Gradient Problem in a Recurrent Neural Network? - Quora.” Accessed February 19, 2016. https://www.quora.com/How-does-LSTM-help-prevent-the-vanishing-and-exploding-gradient-problem-in-a-recurrent-neural-network.
- “Anyone Can Learn To Code an LSTM-RNN in Python (Part 1: RNN) - I Am Trask.” Accessed January 31, 2016. https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/.
- “Deep Learning Lecture 12: Recurrent Neural Nets and LSTMs - YouTube.” Accessed February 27, 2016. https://www.youtube.com/watch?v=56TYLaQN4N8.
- Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. “Learning to Forget: Continual Prediction with LSTM.” *Neural Computation* 12, no. 10 (October 1, 2000): 2451–71. doi:10.1162/089976600300015015.
- Graves, Alex, and Jürgen Schmidhuber. “Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures.” *Neural Networks* 18, no. 5–6 (July 2005): 602–10. doi:10.1016/j.neunet.2005.06.042.
- Hochreiter, S, and J Schmidhuber. “Long Short-Term Memory.” *Neural Computation* 9, no. 8 (November 1997): 1735–80. doi:10.1162/neco.1997.9.8.1735
- Lipton, Zachary C., John Berkowitz, and Charles Elkan. “A Critical Review of Recurrent Neural Networks for Sequence Learning.” *arXiv:1506.00019 [cs]*, May 29, 2015. http://arxiv.org/abs/1506.00019.
- Mazur. “A Step by Step Backpropagation Example.” *Matt Mazur*, March 17, 2015. http://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.
- Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. “On the Difficulty of Training Recurrent Neural Networks.” *arXiv:1211.5063 [cs]*, November 21, 2012. http://arxiv.org/abs/1211.5063.
- “Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs.” *WildML*, September 17, 2015. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.
- “Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients.” *WildML*, October 8, 2015. http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/.
- “Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano.” *WildML*, October 27, 2015. http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/.
- “SCRN vs LSTM • /r/MachineLearning.” *Reddit*. Accessed February 10, 2016.
- https://www.reddit.com/r/MachineLearning/comments/44bxdj/scrn_vs_lstm/.
- “Simple LSTM.” Accessed March 1, 2016. http://nicodjimenez.github.io/2014/08/08/lstm.html.
- “The Unreasonable Effectiveness of Recurrent Neural Networks.” Accessed November 24, 2015. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.
- “Understanding LSTM Networks -- Colah’s Blog.” Accessed January 25, 2016. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
- “Vanishing Gradient Problem.” *Wikipedia, the Free Encyclopedia*, December 29, 2015. https://en.wikipedia.org/w/index.php?title=Vanishing_gradient_problem&oldid=697291222.
- “Why Can Constant Error Carousels (CECs) Prevent LSTM from the Problems of Vanishing/exploding Gradients? • /r/MachineLearning.” *Reddit*. Accessed March 1, 2016. https://www.reddit.com/r/MachineLearning/comments/34piyi/why_can_constant_error_carousels_cecs_prevent/.
- Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks.” *arXiv:1409.3215 [cs]*, September 10, 2014. http://arxiv.org/abs/1409.3215.
- “NIPS: Oral Session 4 - Ilya Sutskever - Microsoft Research.” Accessed March 1, 2016. http://research.microsoft.com/apps/video/?id=239083.
- Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks.” *arXiv:1409.3215 [cs]*, September 10, 2014. http://arxiv.org/abs/1409.3215.