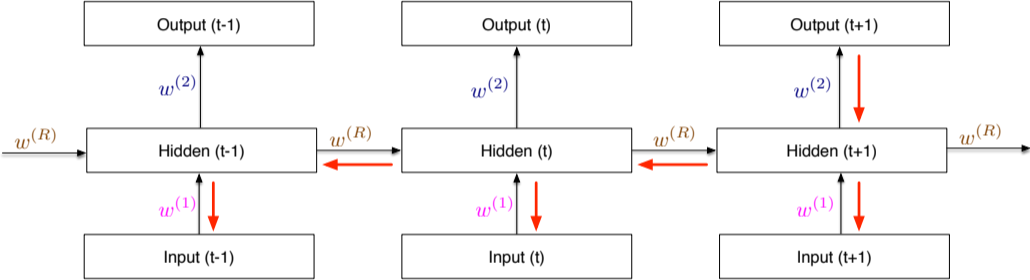


Recurrent Neural Networks 2: LSTM, gates, and current research

Steve Renals

Machine Learning Practical — MLP Lecture 10
22 November 2017 / 27 November 2017

Simple recurrent network

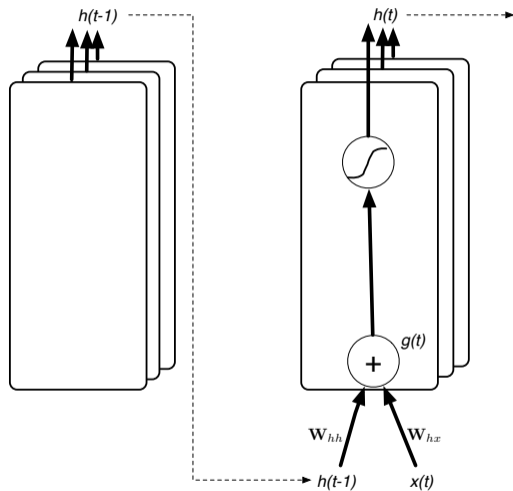


Vanishing and exploding gradients

- BPTT involves taking the product of many gradients (as in a very deep network)
 - this can lead to vanishing (component gradients less than 1) or exploding (greater than 1) gradients
- This can prevent effective training
- Modified optimisation algorithms
 - RMSProp (and similar algorithms) – normalise the gradient for each weight by average of its magnitude, with a learning rate for each weight
 - Hessian-free – an approximation to second-order approaches which use curvature information
- Modified hidden unit transfer functions:
 - Long short term memory (LSTM)
 - Linear self-recurrence for each hidden unit (long-term memory)
 - Gates - dynamic weights which are a function of their inputs

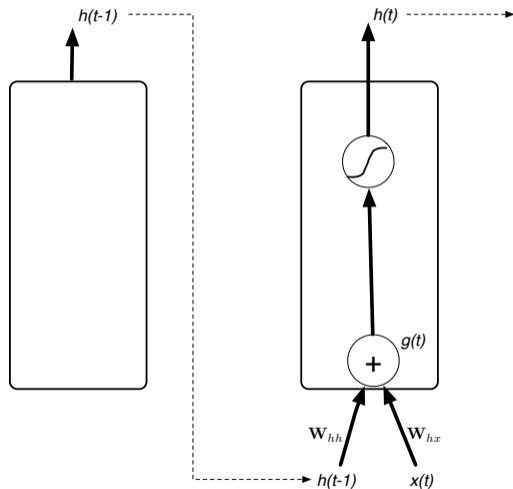
LSTM

Simple recurrent network unit



$$\mathbf{g}(t) = \mathbf{W}_{hx}\mathbf{x}(t) + \mathbf{W}_{hh}\mathbf{h}(t-1) + \mathbf{b}_h$$
$$\mathbf{h}(t) = \tanh(\mathbf{g}(t))$$

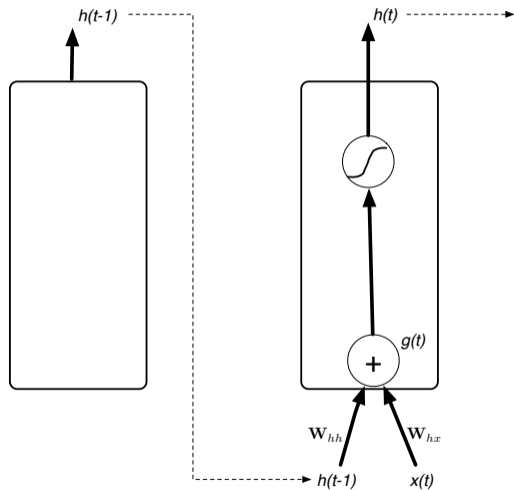
Simple recurrent network unit



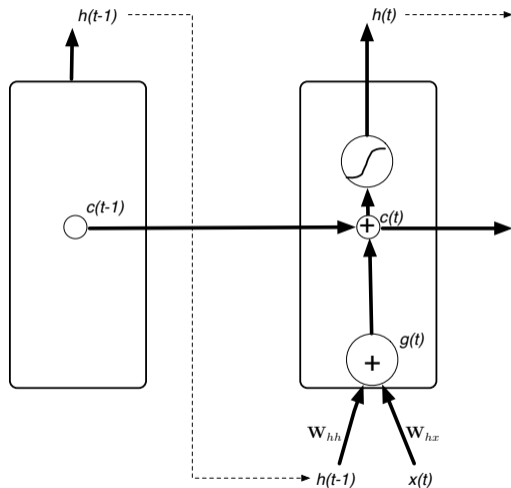
$$\mathbf{g}(t) = \mathbf{W}_{hx}\mathbf{x}(t) + \mathbf{W}_{hh}\mathbf{h}(t-1) + \mathbf{b}_h$$

$$\mathbf{h}(t) = \tanh(\mathbf{g}(t))$$

LSTM

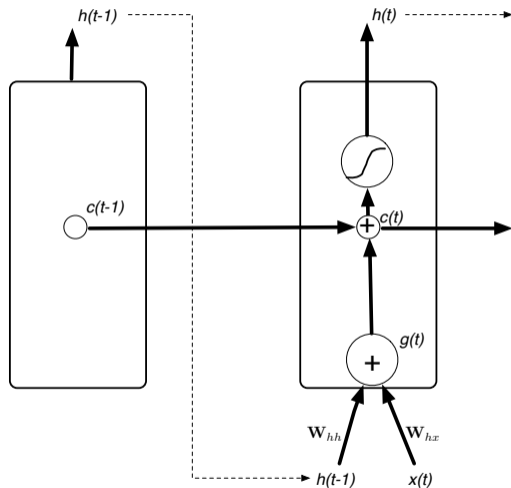


LSTM – Internal recurrent state



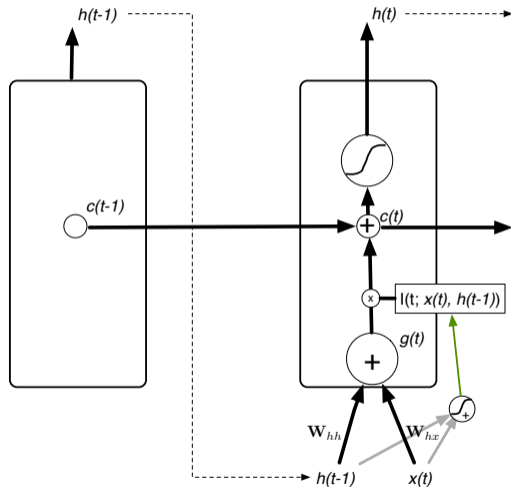
- **Internal recurrent state** (“cell”) $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$

LSTM – Internal recurrent state



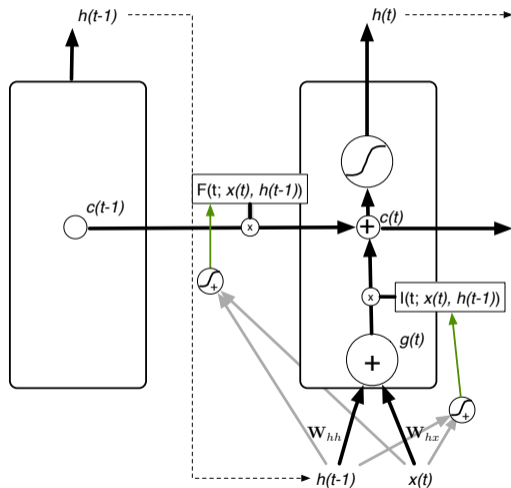
- **Internal recurrent state** (“cell”) $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state

LSTM – Input Gate



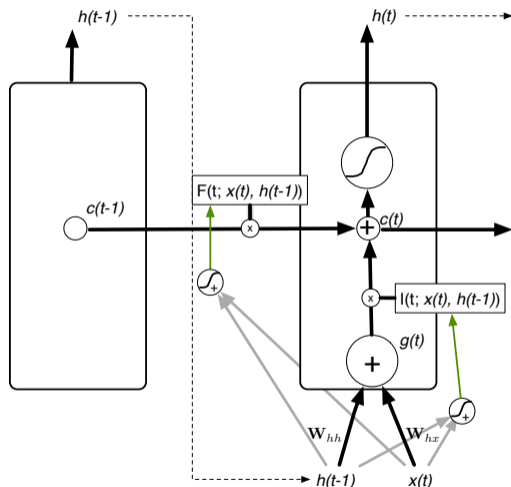
- **Internal recurrent state** (“cell”) $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state
- **Input gate**: controls how much input to the unit $g(t)$ is written to the internal state $c(t)$

LSTM – Forget Gate



- **Internal recurrent state** (“cell”) $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state
- **Input gate**: controls how much input to the unit $g(t)$ is written to the internal state $c(t)$
- **Forget gate**: controls how much of the previous internal state $c(t-1)$ is written to the internal state $c(t)$
 - Input and forget gates together allow the network to control what information is stored and overwritten at each step

LSTM – Input and Forget Gates



$$\mathbf{I}(t) = \sigma(\mathbf{W}_{ix}\mathbf{x}(t) + \mathbf{W}_{ih}\mathbf{h}(t-1) + \mathbf{b}_i)$$

$$\mathbf{F}(t) = \sigma(\mathbf{W}_{fx}\mathbf{x}(t) + \mathbf{W}_{fh}\mathbf{h}(t-1) + \mathbf{b}_f)$$

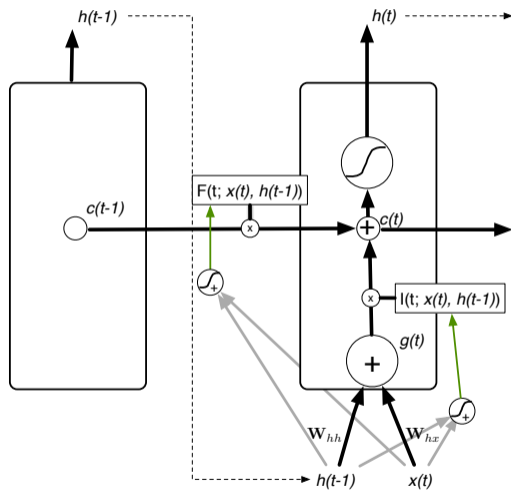
$$\mathbf{g}(t) = \mathbf{W}_{hx}\mathbf{x}(t) + \mathbf{W}_{hh}\mathbf{h}(t-1) + \mathbf{b}_h$$

$$\mathbf{c}(t) = \mathbf{F}(t) \circ \mathbf{c}(t-1) + \mathbf{I}(t) \circ \mathbf{g}(t)$$

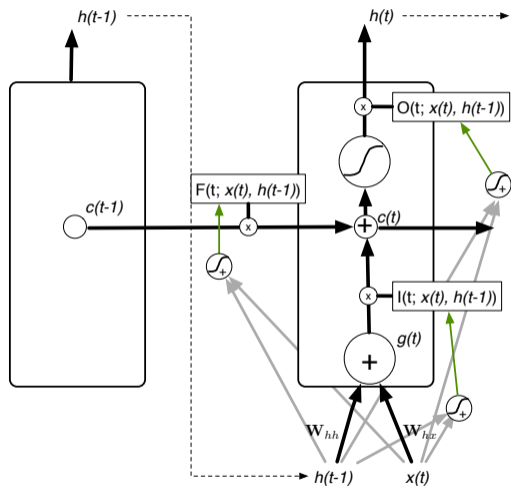
σ is the sigmoid function

\circ is element-wise vector multiply

LSTM – Input and Forget Gates

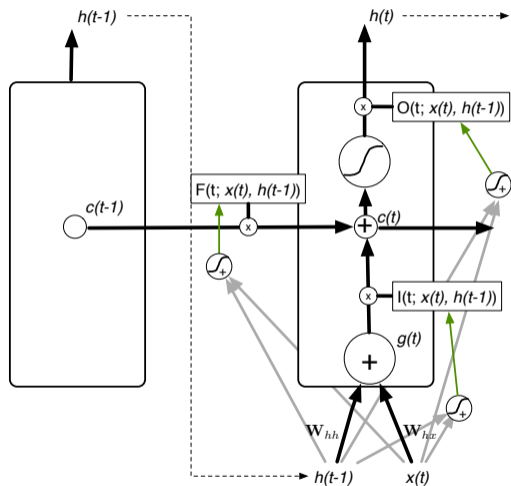


LSTM – Output Gate



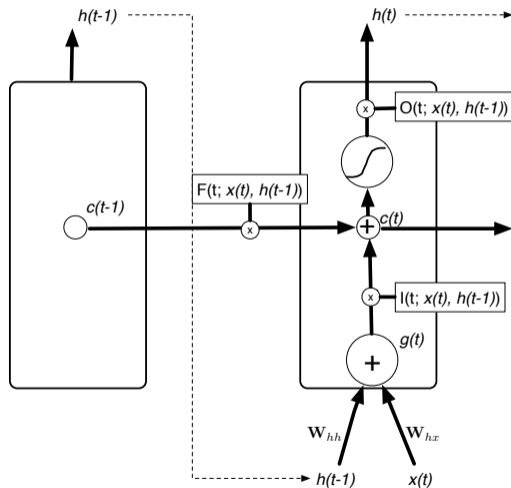
- **Output gate:** controls how much of each unit's activation is output by the hidden state – it allows the LSTM cell to keep information that is not relevant at the current time, but may be relevant later

LSTM – Output Gate



$$\mathbf{O}(t) = \sigma(\mathbf{W}_{ox}\mathbf{x}(t) + \mathbf{W}_{oh}\mathbf{h}(t-1) + \mathbf{b}_o)$$
$$\mathbf{h}(t) = \mathbf{O}(t) \circ \tanh(\mathbf{c}(t))$$

LSTM equations



$$\mathbf{I}(t) = \sigma(\mathbf{W}_{ix}\mathbf{x}(t) + \mathbf{W}_{ih}\mathbf{h}(t-1) + \mathbf{b}_i)$$

$$\mathbf{F}(t) = \sigma(\mathbf{W}_{fx}\mathbf{x}(t) + \mathbf{W}_{fh}\mathbf{h}(t-1) + \mathbf{b}_f)$$

$$\mathbf{O}(t) = \sigma(\mathbf{W}_{ox}\mathbf{x}(t) + \mathbf{W}_{oh}\mathbf{h}(t-1) + \mathbf{b}_o)$$

$$\mathbf{g}(t) = \mathbf{W}_{hx}\mathbf{x}(t) + \mathbf{W}_{hh}\mathbf{h}(t-1) + \mathbf{b}_h$$

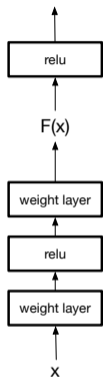
$$\mathbf{c}(t) = \mathbf{F}(t) \circ \mathbf{c}(t-1) + \mathbf{I}(t) \circ \mathbf{g}(t)$$

$$\mathbf{h}(t) = \mathbf{O}(t) \circ \tanh(\mathbf{c}(t))$$

- Goodfellow et al, chapter 10
- C Olah (2015), Understanding LSTMs,
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- A Karpathy et al (2015), Visualizing and Understanding Recurrent Networks,
<https://arxiv.org/abs/1506.02078>

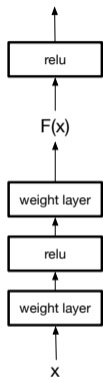
More gating units

Gating units in highway networks

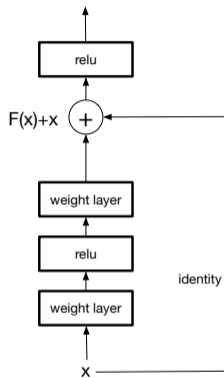


Deep network module

Gating units in highway networks

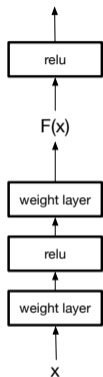


Deep network module

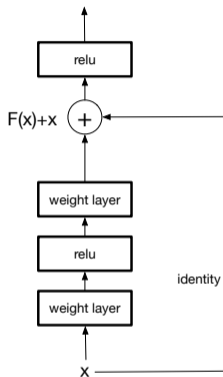


Resnet network module

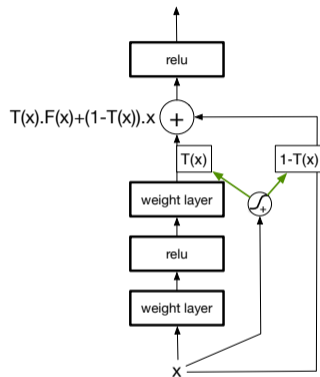
Gating units in highway networks



Deep network module



Resnet network module

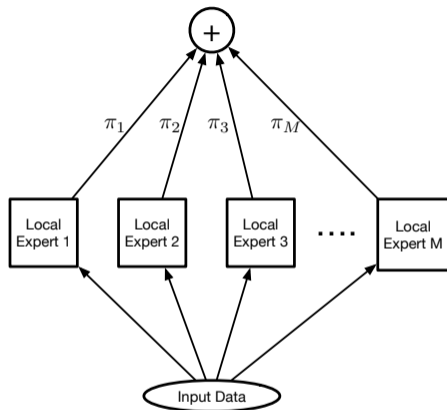


Highway network module

Srivastava et al, 2015, Training Very Deep Networks, NIPS-2015,

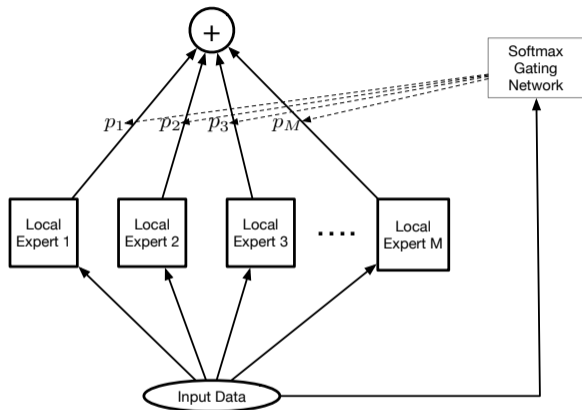
<https://arxiv.org/abs/1507.06228>

Mixture of experts



Finite Mixture Model

Mixture of experts

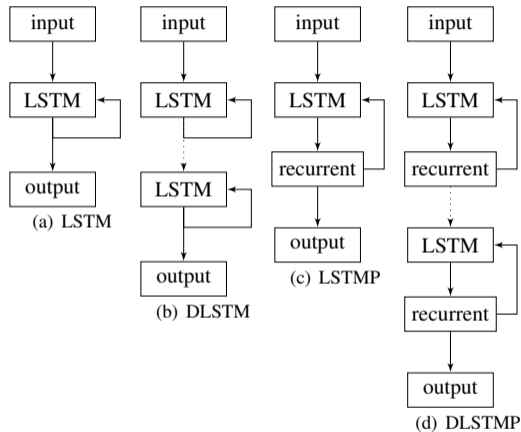


Mixture of Experts

Jacobs et al (1991), Adaptive Mixtures of Local Experts, <http://cognet.mit.edu/node/29931>

Example applications using RNNs

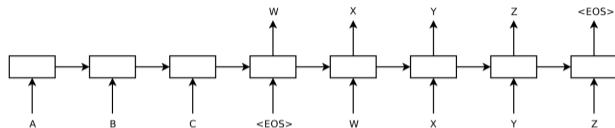
Example 1: speech recognition with stacked LSTMs



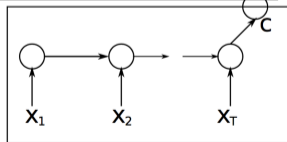
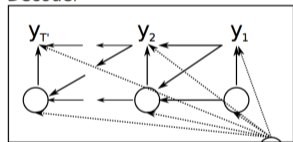
H Sak et al (2014). "Long Short-Term Memory based Recurrent Neural Network Architectures for Large Scale Acoustic Modelling", *Interspeech*.

Example 2: recurrent encoder-decoder

Machine translation



Decoder



Encoder

- I Sutskever et al (2014). "Sequence to Sequence Learning with Neural Networks", *NIPS*.
- K Cho et al (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", *EMNLP*.

Summary

- Vanishing gradient problem
- LSTMs and gating
- Applications: stacked LSTMs for speech recognition, encoder-decoder for machine translation
- More on recurrent networks next semester in NLU, ASR, MT,)