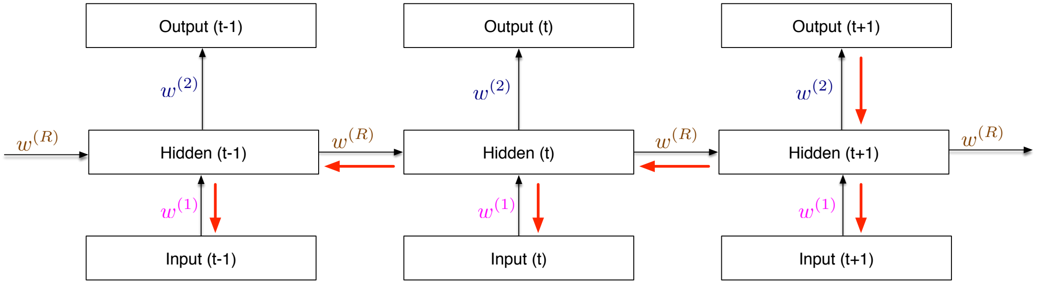
Recurrent Neural Networks 2: LSTM, gates, and current research

Hakan Bilen
Slide Credits: Steve Renals

Machine Learning Practical — MLP Lecture 10
13 November 2018 / 20 November 2018

Simple recurrent network



Vanishing and exploding gradients

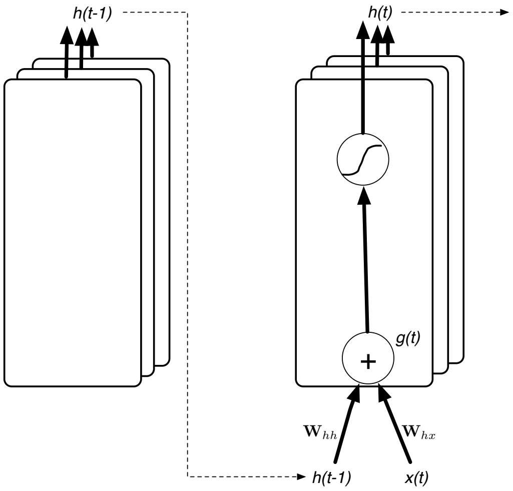
- BPTT involves taking the product of many gradients (as in a very deep network) – this can lead to vanishing (component gradients less than 1) or exploding (greater than 1) gradients
- This can prevent effective training
- Modified optimisation algorithms
    - RMSProp (and similar algorithms) – normalise the gradient for each weight by average of it 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

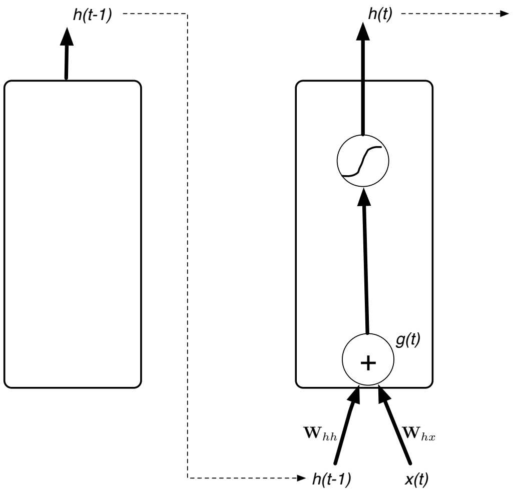
LSTM

Simple recurrent network unit



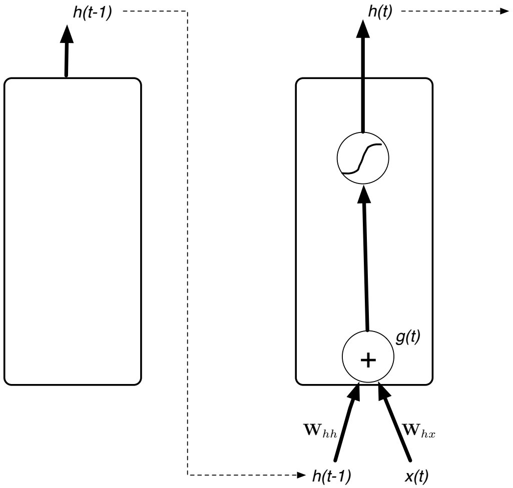
$$\boldsymbol{g}(t) = \boldsymbol{W}_{hx}\boldsymbol{x}(t) + \boldsymbol{W}_{hh}\boldsymbol{h}(t-1) + \boldsymbol{b}_h$$
$$\boldsymbol{h}(t) = \tanh\left(\boldsymbol{g}(t)\right)$$

Simple recurrent network unit

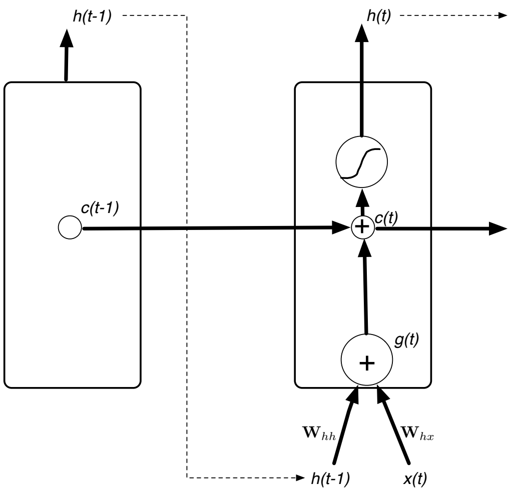


$$\boldsymbol{g}(t) = \boldsymbol{W}_{hx}\boldsymbol{x}(t) + \boldsymbol{W}_{hh}\boldsymbol{h}(t-1) + \boldsymbol{b}_h$$
$$\boldsymbol{h}(t) = \tanh\left(\boldsymbol{g}(t)\right)$$

LSTM

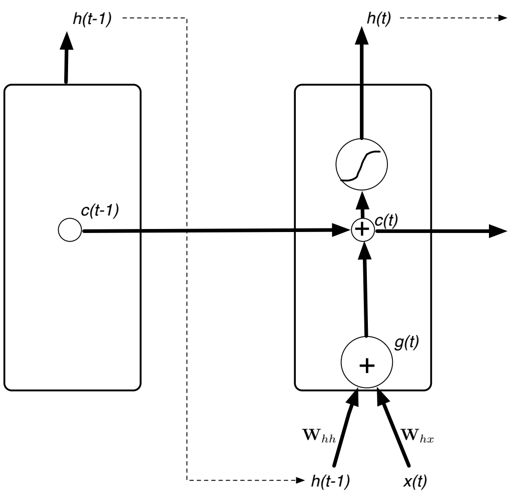


LSTM – Internal recurrent state



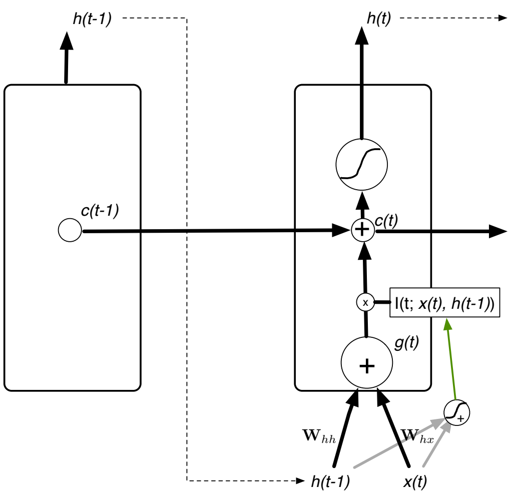
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$

LSTM – Internal recurrent state



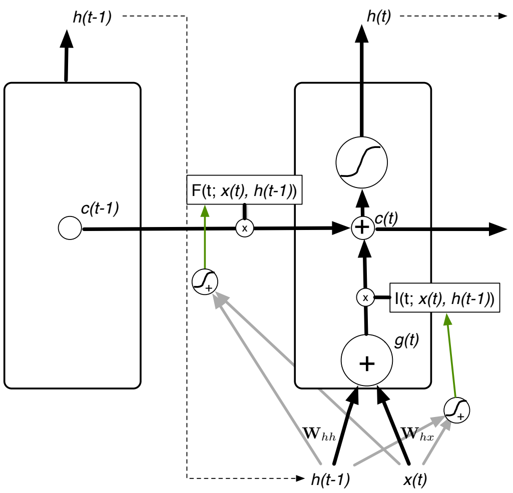
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state

LSTM – Input Gate



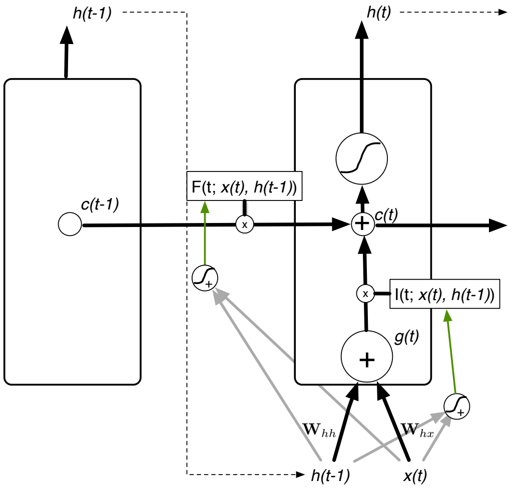
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state
- **Input gate**: controls how much input to the unit $g(t)$ is written to the internal state $c(t)$

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



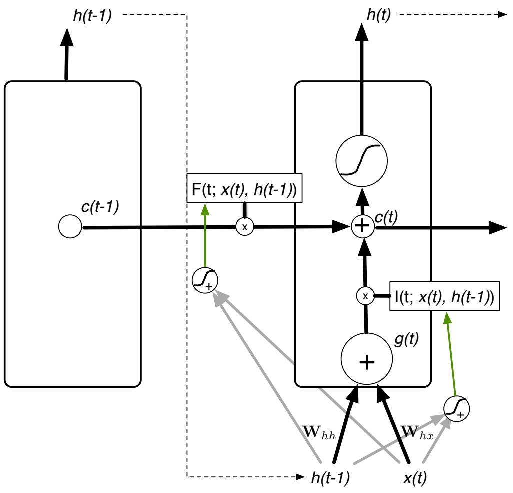
$$\boldsymbol{I}(t) = \sigma\left(\boldsymbol{W}_{ix}\boldsymbol{x}(t) + \boldsymbol{W}_{ih}\boldsymbol{h}(t-1) + \boldsymbol{b}_i\right)$$
$$\boldsymbol{F}(t) = \sigma\left(\boldsymbol{W}_{fx}\boldsymbol{x}(t) + \boldsymbol{W}_{fh}\boldsymbol{h}(t-1) + \boldsymbol{b}_f\right)$$

$$\boldsymbol{g}(t) = \boldsymbol{W}_{hx}\boldsymbol{x}(t) + \boldsymbol{W}_{hh}\boldsymbol{h}(t-1) + \boldsymbol{b}_h$$
$$\boldsymbol{c}(t) = \boldsymbol{F}(t) \circ \boldsymbol{c}(t-1) + \boldsymbol{I}(t) \circ \boldsymbol{g}(t)$$

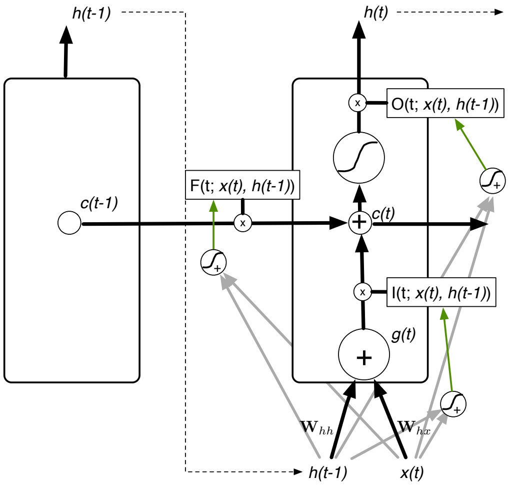
σ is the sigmoid function

$\circ$ is element-wise vector multiply

LSTM – Input and Forget Gates

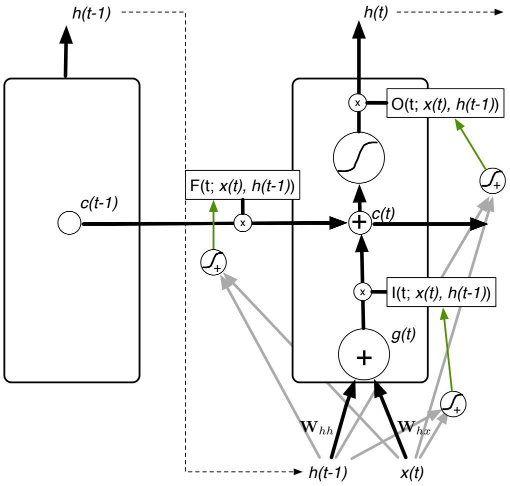


LSTM – Output Gate



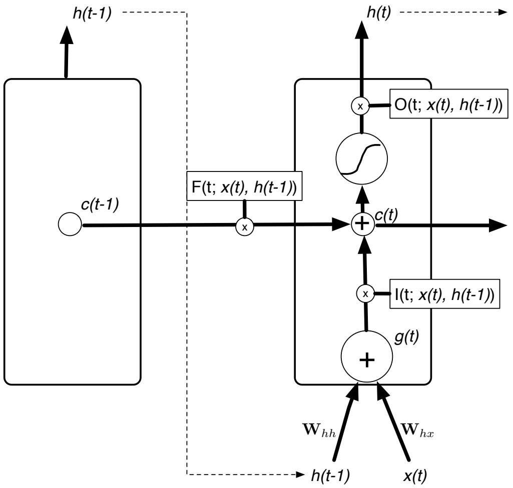
- **Output gate**: controls how much of each unit's activation is output by the hidden state – it allows the LSTM cell to keep information that is not relevant at the current time, but may be relevant later

LSTM – Output Gate



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

LSTM equations



$$
\begin{aligned}
\boldsymbol{I}(t) &= \sigma\left(\boldsymbol{W}_{ix}\boldsymbol{x}(t) + \boldsymbol{W}_{ih}\boldsymbol{h}(t-1) + \boldsymbol{b}_i\right)\\
\boldsymbol{F}(t) &= \sigma\left(\boldsymbol{W}_{fx}\boldsymbol{x}(t) + \boldsymbol{W}_{fh}\boldsymbol{h}t-1) + \boldsymbol{b}_f\right)\\
\boldsymbol{O}(t) &= \sigma\left(\boldsymbol{W}_{ox}\boldsymbol{x}(t) + \boldsymbol{W}_{oh}\boldsymbol{h}(t-1) + \boldsymbol{b}_o\right)\\
\boldsymbol{g}(t) &= \boldsymbol{W}_{hx}\boldsymbol{x}(t) + \boldsymbol{W}_{hh}\boldsymbol{h}(t-1) + \boldsymbol{b}_h\\
\boldsymbol{c}(t) &= \boldsymbol{F}(t)\circ\boldsymbol{c}(t-1) + \boldsymbol{I}(t)\circ\boldsymbol{g}(t)\\
\boldsymbol{h}(t) &= \boldsymbol{O}(t)\circ\tanh\left(\boldsymbol{c}(t)\right)
\end{aligned}
$$

An Empirical Exploration of Recurrent Network Architectures

Josefowicz et al (2015), `http://proceedings.mlr.press/v37/jozefowicz15.pdf`

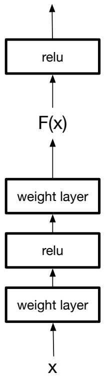
- Goal is to determine whether the LSTM architecture is optimal or whether much better architectures exist
- Conducted a thorough architecture search and evaluated over 10,000 different RNN architectures
- Vary activation functions/operations, insert/remove nodes
- "Though there were architectures that outperformed the LSTM on some problems, we were unable to find an architecture that consistently beat the LSTM and the GRU in all experimental conditions"

LSTM/RNN readings

- 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

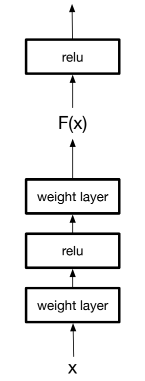
More gating units

Gating units in highway networks

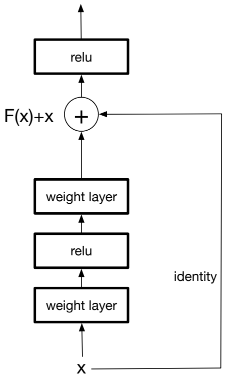


Deep network module

Gating units in highway networks

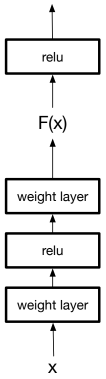


Deep network module

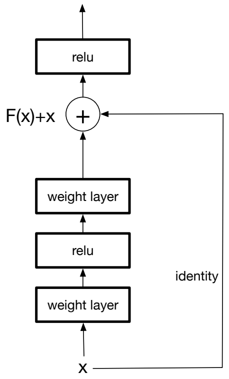


Resnet network module

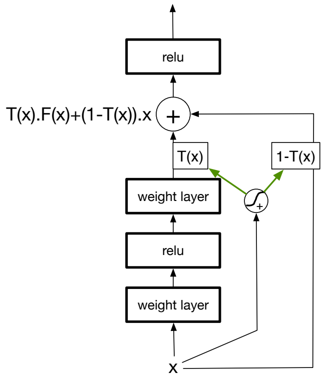
Gating units in highway networks



Deep network module



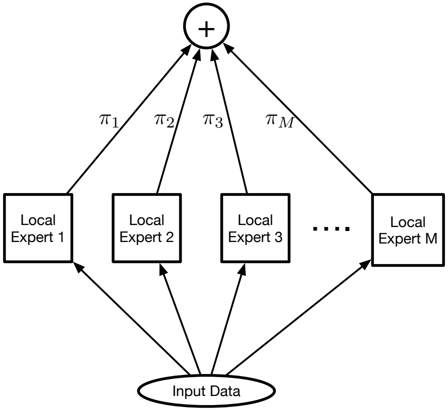
Resnet network module



Highway network module

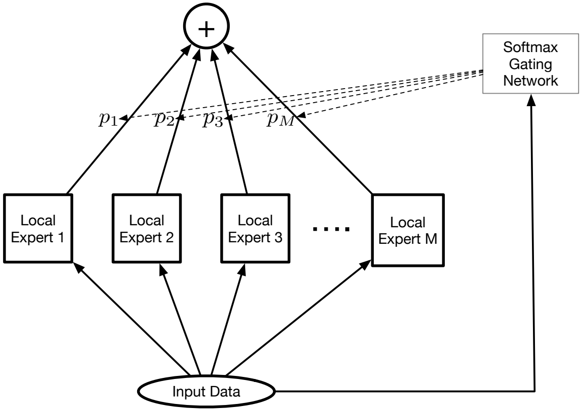
Srivastava et al, 2015, Training Very Deep Networks, NIPS-2015, https://arxiv.org/abs/1507.06228

Mixture of experts



Finite Mixture Model

Mixture of experts

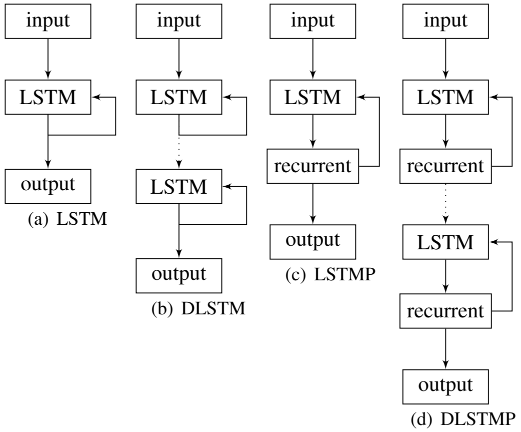


Mixture of Experts

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

Example applications using RNNs

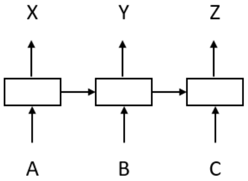
Example 1: speech recognition with stacked LSTMs



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

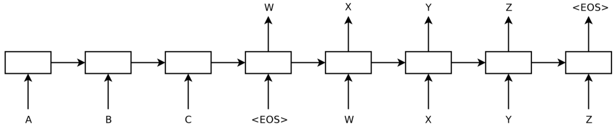
Question

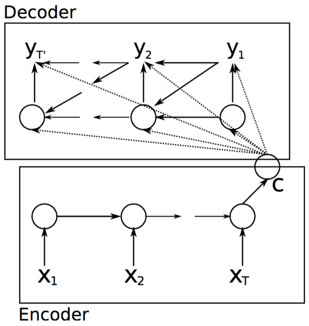
- We want to translate a sentence from one language to another one
- Is the architecture below suitable for this problem?



Example 2: recurrent encoder-decoder

Machine translation



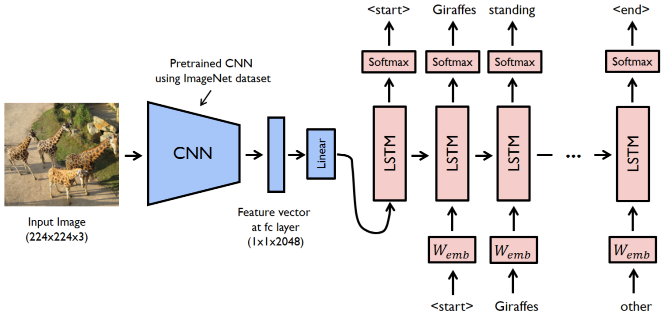


- I Sutskever et al (2014). "Sequence to Sequence Learning with Neural Networks", *NIPS*.
- K Cho et al (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", *EMNLP*.

Example 3: CNN-LSTM

Image captioning

https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning



- Goal is to convert an image to a natural language descriptor
- Input to LSTM is CNN features

Example 3: CNN-LSTM

Karpathy et al (2015). “Deep Visual-Semantic Alignments for Generating Image Descriptions”, CVPR.

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

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, ...)