Recurrent Neural Networks 2: LSTM, gates, and attention

Hakan Bilen

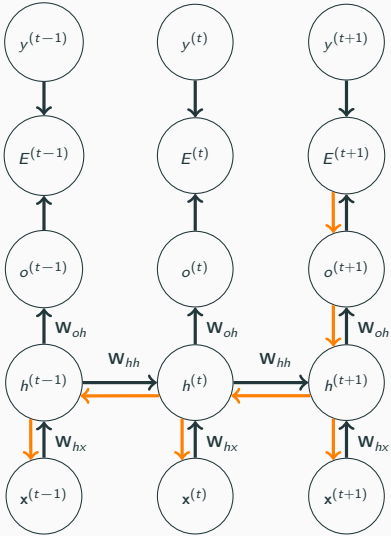
Machine Learning Practical — MLP Lecture 10

19 November 2019

Coursework 1 Overview Feedback

- **Almost everyone did a very good job, code passed the unit tests, reports were well written and structured**
- Most people:
  - carried out experiments well, clearly presented results with graphs and tables
  - gave a reasonable discussion of the results
  - used references to the literature
  - clearly described the EMNIST task and experiments, the activation functions
- However, some people
  - did not make clear that composition of linear transformations is linear too
  - used references incorrectly
  - did not present results in the clearest way (too many graphs)
- Many did not meaningfully discuss whether small differences in results are significant
- **The best reports discussed the "why" as well as the "what"...**

Simple RNN with recurrent hidden unit



- Recurrence through hidden units
- Hidden units can be thought as memory
- Effective network depth equals to the input sequence length
- BPTT involves backpropagating the error through the previous time steps (layers)

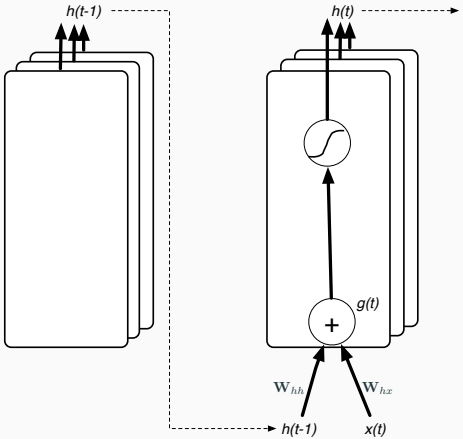
Vanishing and exploding gradients

- 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
  - Gradient clipping – prevent large noisy gradients
- 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

Long Short-Term Memory
LSTM

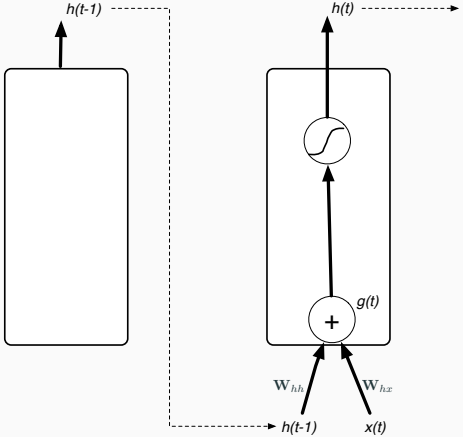
Simple recurrent network unit



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

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

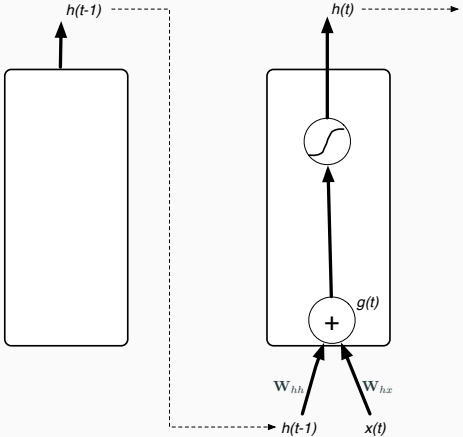
Simple recurrent network unit



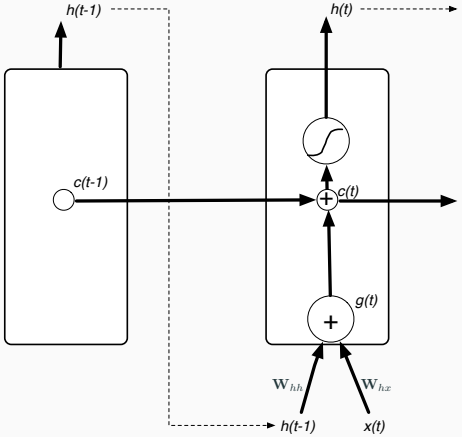
$$\mathbf{g}(t) = \mathbf{W}_{hx}\mathbf{x}(t) + \mathbf{W}_{hh}\mathbf{h}(t-1) + \mathbf{b}_h$$

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

LSTM

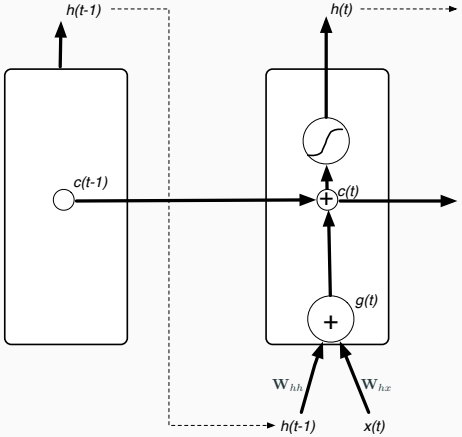


LSTM – Internal recurrent state



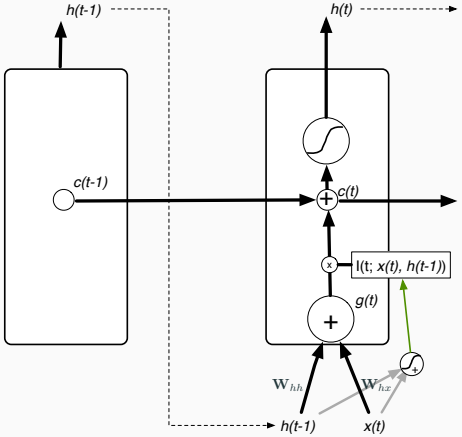
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and $g(t)$

LSTM – Internal recurrent state



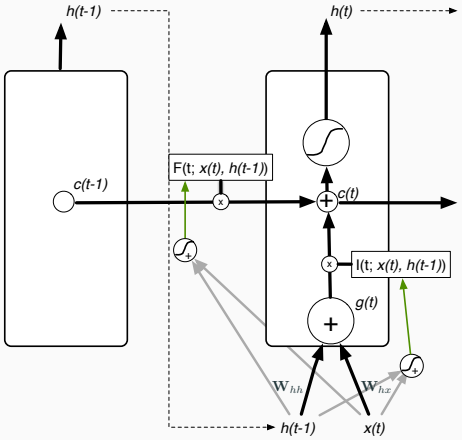
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and $g(t)$
- Gates - weights dependent on the current input and the previous state

LSTM – Input Gate



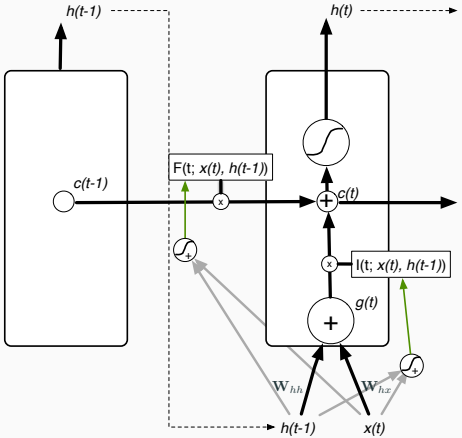
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and $g(t)$
- Gates - weights dependent on the current input and the previous state
- **Input gate**: controls how much input to the unit $g(t)$ is written to the internal state $c(t)$

LSTM – Forget Gate



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



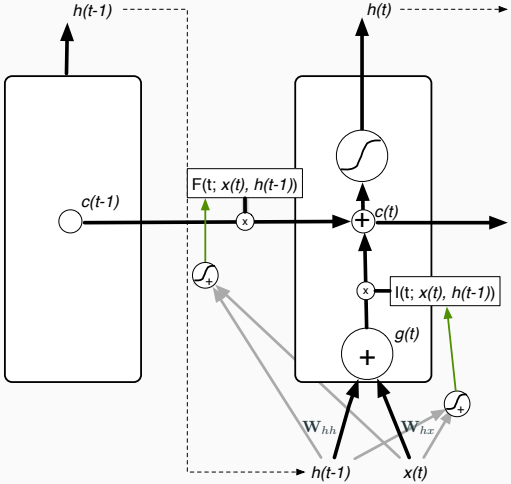
$$\mathbf{I}(t) = \sigma\left(\mathbf{W}_{ix}\mathbf{x}(t) + \mathbf{W}_{ih}\mathbf{h}(t-1) + \mathbf{b}_i\right)$$
$$\mathbf{F}(t) = \sigma\left(\mathbf{W}_{fx}\mathbf{x}(t) + \mathbf{W}_{fh}\mathbf{h}(t-1) + \mathbf{b}_f\right)$$

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

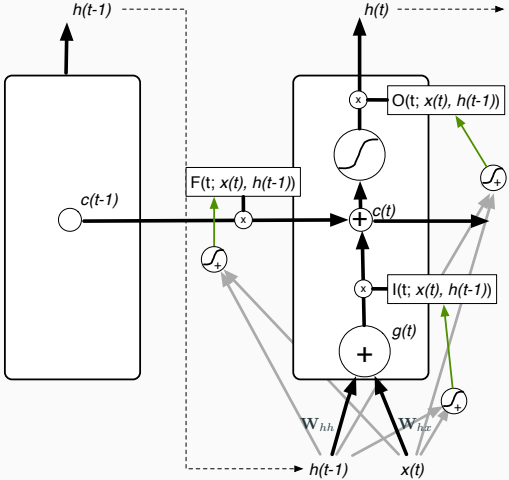
σ is the sigmoid function

$\odot$ is element-wise vector multiply

LSTM – Input and Forget Gates

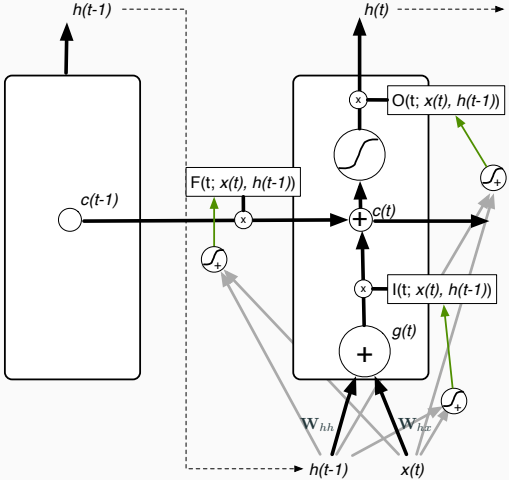


LSTM – Output Gate



- **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

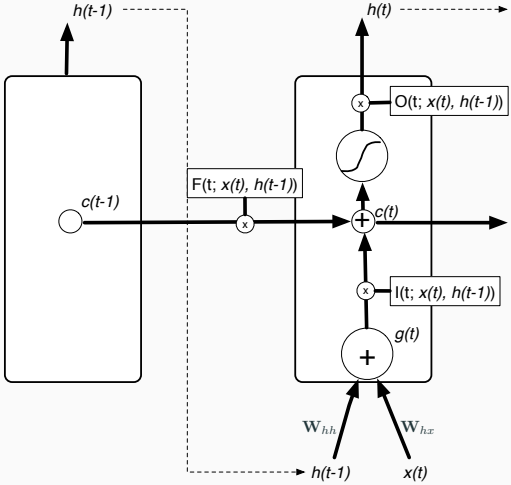
LSTM – Output Gate



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

$$\mathbf{h}(t) = \mathbf{O}(t) \odot \tanh\left(\mathbf{c}(t)\right)$$

LSTM equations



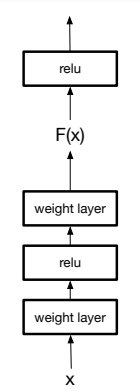
$$\mathbf{I}(t) = \sigma\left(\mathbf{W}_{ix}\mathbf{x}(t) + \mathbf{W}_{ih}\mathbf{h}(t-1) + \mathbf{b}_i\right)$$
$$\mathbf{F}(t) = \sigma\left(\mathbf{W}_{fx}\mathbf{x}(t) + \mathbf{W}_{fh}\mathbf{h}t-1) + \mathbf{b}_f\right)$$
$$\mathbf{O}(t) = \sigma\left(\mathbf{W}_{ox}\mathbf{x}(t) + \mathbf{W}_{oh}\mathbf{h}(t-1)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

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

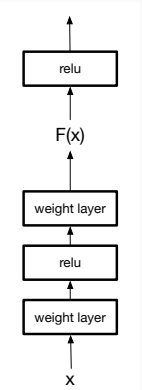
More gating units

Gating units in highway networks

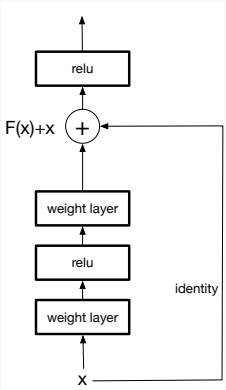


Deep network module

Gating units in highway networks

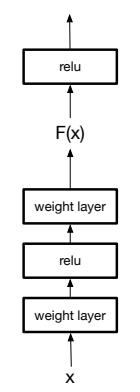


Deep network module

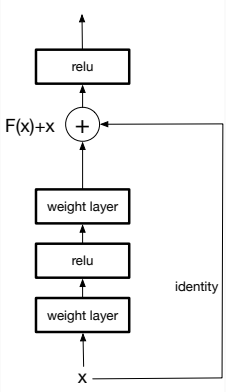


Resnet network module

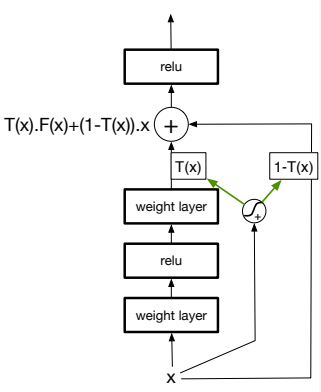
Gating units in highway networks



Deep network module



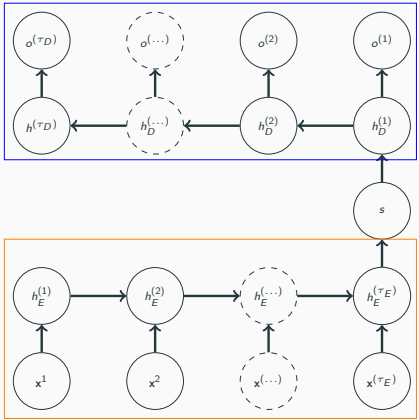
Resnet network module



Highway network module

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

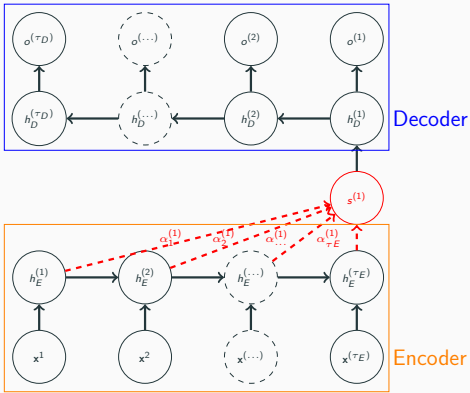
Sequence-to-sequence model ("seq2seq")



- Encoder and decoder use LSTM
- The encoder summarizes the whole input sequence into a single context vector $s = h_E^{\tau_E}$
- Performs poorly as the length of an input sentence increases

Sutskever et al, "Sequence to sequence learning with neural networks." NeurIPS2014.

Gating for attention in seq2seq



Each time the decoder generates a word, it "attends" to the most relevant information words in a source sentence

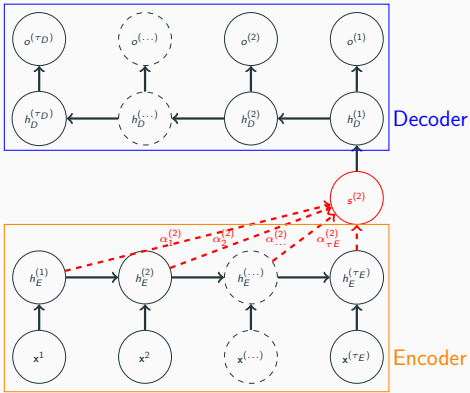
$$s^{(t)} = \sum_{k=1}^{\tau_E} \alpha_k^{(t)} h_E^{(k)}$$

$$\text{where } \alpha_i^{(t)} = \exp\left(e_i^{(t)}\right) / \sum_{k=1}^{\tau_E} \exp\left(e_k^{(t)}\right)$$

$$e_i^{(t)} = W_{ad} h_D^{(t-1)} + W_{ae} h_E^{(i)} + b_a.$$

Bahdanau et al. "Neural Machine Translation by Jointly Learning to Align and Translate", NeurIPS2016.

Gating for attention in seq2seq



Each time the decoder generates a word, it "attends" to the most relevant information words in a source sentence

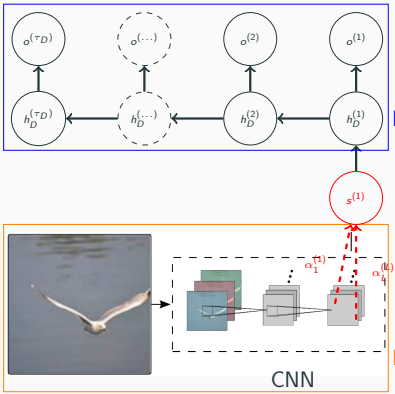
$$s^{(t)} = \sum_{k=1}^{\tau_E} \alpha_k^{(t)} h_E^{(k)}$$

where $\alpha_i^{(t)} = \exp\left(e_i^{(t)}\right) / \sum_{k=1}^{\tau_E} \exp\left(e_k^{(t)}\right)$

$$e_i^{(t)} = W_{ad} h_D^{(t-1)} + W_{ae} h_E^{(i)} + b_a.$$

Bahdanau et al. "Neural Machine Translation by Jointly Learning to Align and Translate", NeurIPS2016.

Gating for attention in vec2seq



- CNN as encoder and LSTM as decoder
- CNN extracts L features $\{\mathbf{x}^1, \ldots, \mathbf{x}^L\}$
- Each time the decoder generates a word, it "attends" to the most relevant information image feature

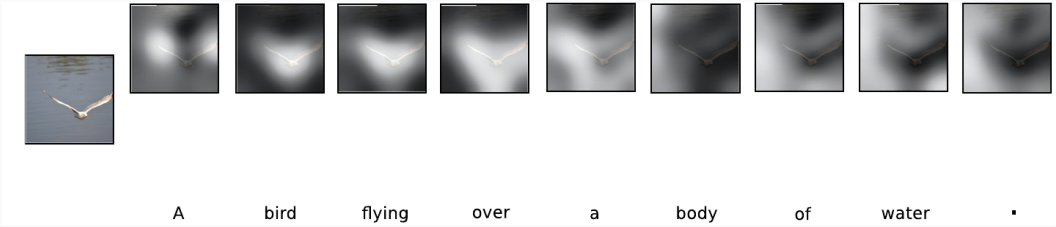
$$s^{(t)} = \sum_{k=1}^{L} \alpha_k^{(t)} x^{(k)}$$

where $\alpha_i^{(t)} = \exp\left(e_i^{(t)}\right) / \sum_{k=1}^{\tau_E} \exp\left(e_k^{(t)}\right)$

$$e_i^{(t)} = W_{ad} h_D^{(t-1)} + W_{ax} x^{(i)} + b_a$$

Xu et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML2015.

Gating for attention in vec2seq



Gating for attention in vec2seq

A bird flying over a body of water .

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

Summary

- Vanishing gradient problem
- LSTMs and gating
- Examples:
  - Highway network module
  - Seq2seq for machine translation
  - Vec2seq for image captioning
- Next week: Introduction to semester 2 projects