Recurrent Neural Networks (cont.)

Nan Ye

School of Mathematics and Physics
The University of Queensland

Avoiding exploding/vanishing gradients

Truncated BPTT

- Idea: train on small chunks instead of using full sequences
- We still do a full forward propagation.
- We compute the losses for each chunk separately, and backprop is only run for one chunk of losses at a time.
- This prevents RNN from fully exploiting the history.

Good initialization

- Starting from large weights is bad — gradient can easily explode.
- Starting from 0 weights are bad — the hidden neurons will not be different from each other.
- Small random weights are often used in practice.

Gradient clipping

- If a gradient g has norm larger than a threshold v, then clip g to $gv/\|g\|$.
- Gradient clipping avoids exploding gradients, but does not solve the problem of vanishing gradients.

Design a better architecture

- LSTM or GRU has improved gradient flow due to their additive interactions.
- They can be effectively trained for very long sequences without encountering exploding/vanishing gradients.

Long Short-Term Memory (LSTM)

What's in a name? That which we call a rose
By any other name would smell as sweet;

Romeo and Juliet, William Shakespeare

- Hochreiter and Schmidhuber, 1997 introduced LSTM to avoid exploding and vanishing gradients.
- What's in the name
  - Long-term memory: the weights, persistent, change slowly
  - Short-term memory: the activations, volatile, can change quickly
  - LSTM has short-term memories that are long (i.e. histories further away still have an effect).

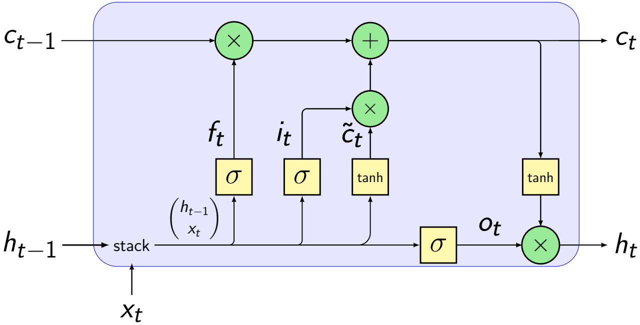
Hochreiter and Schmidhuber, Long short-term memory, 1997

Key ideas in LSTM

- The idea is to introduce a memory cell that can pass gradients without significantly reducing or increasing them.
- A modern version LSTM typically uses three gates to control information flow
  - The forget gate controls how much information stays in the cell.
  - The input gate controls how much information gets into the cell.
  - The output gate controls how much information can be read from the cell.

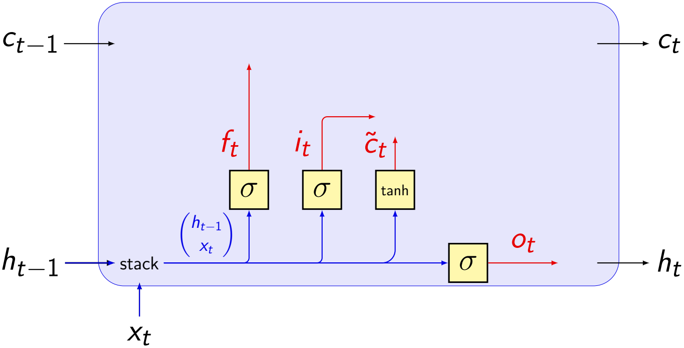
The LSTM architecture

- The modern LSTM architecture is shown below



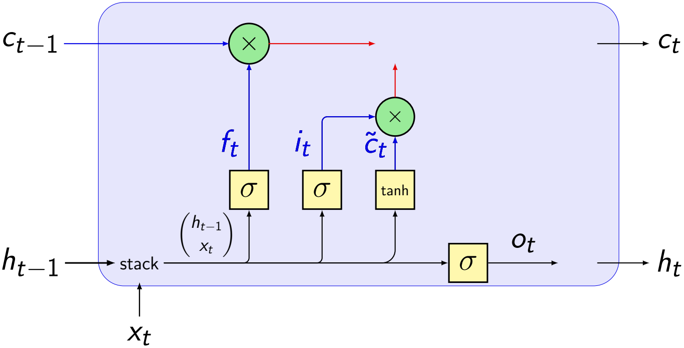
- This is very complex — we'll walk through it step by step.

- While an LSTM still has a hidden state h_t as standard RNNs, it additionally maintains a cell state c_t.
- c_t is internal, while h_t is exposed.



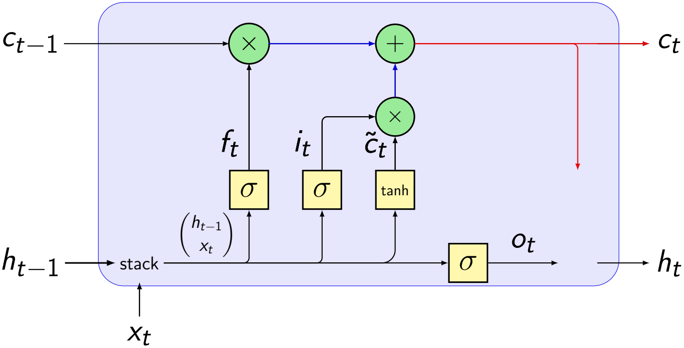
$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f), && \text{(forget gate: whether to erase cell)}\\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i), && \text{(input gate: whether to write to cell)}\\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), && \text{(output gate: how much to reveal cell),}\\
\tilde{c}_t &= \tanh(W_h \cdot [h_{t-1}, x_t] + b_h), && \text{(cell gate: how much to write to cell),}
\end{aligned}
$$

$[h_{t-1}, x_t]$ is the column vector with h_{t-1} stacked on x_t

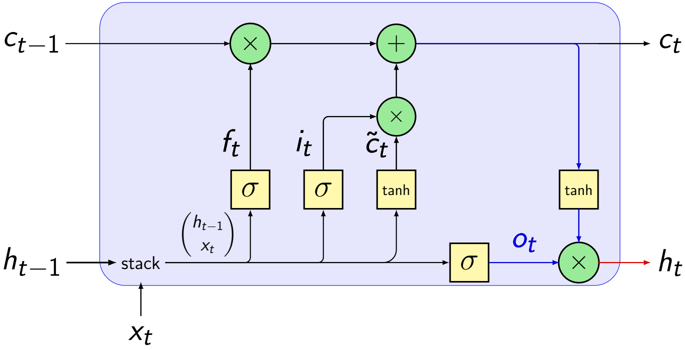


- $f_t \odot c_{t-1}$ is the amount of information passed to next time step.
- $i_t \odot \tilde{c}_t$ is the amount of new information to be stored.

  Product in the diagram is the Hadamard product $\odot$ (i.e., elementwise product).



- The new cell state is $c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t$.
- This combines what is remembered, and what is new (and should be remembered).



- The new hidden state is $h_t = o_t \odot \tanh c_t$.
- This is a partial revelation of the cell state.

Summary of the computation

Vanilla RNN

$$h_t = \tanh\left(W\begin{pmatrix}h_{t-1}\\ x_t\end{pmatrix} + b\right)$$

LSTM

$$\begin{pmatrix}f_t\\ i_t\\ \tilde{c}_t\\ o_t\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \tanh\\ \sigma\end{pmatrix}\left(W\begin{pmatrix}h_{t-1}\\ x_t\end{pmatrix} + b\right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t,$$
$$h_t = o_t \odot \tanh(c_t).$$

Why LSTM works?

- When we compute the gradient of h_t, we need to incorporate a component depending on the gradient of c_t.
- The gradient of c_t does not easily vanish
  - Gradient of c_t contains the Hadamard product of f_t and the gradient of c_{t-1}.
  - This keeps the gradient flow to be uninterrupted as long as f_t is sufficiently large (which should be the case in practice, because we will try to remember information long ago).
- LSTM can still suffer from exploding gradients.

Gated Recurrent Unit (GRU)

- GRU is a simplified variation of LSTM.
- GRU is more efficient: 3 nonlinearities instead of 4 in LSTM.
- Key changes
  - The cell state and hidden state are merged: only h_t, no more c_t.
  - There is no nonlinear output gate: h_t is exposed.
  - The roles of the forget and input gates are redistributed to an update gate and a reset gate.

Cho, Merrienboer, Gulcehre, Bougares, Schwenk, and Bengio, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014

- The update equations for GRU are given below

$$
\begin{aligned}
r_t &= \sigma(W_r[h_{t-1}, x_t] + b_r), && \text{(reset gate)} \\
\tilde{h}_t &= \tanh(W_c[r_t \odot h_{t-1}, x_t] + b_r), \\
z_t &= \sigma(W_z[h_{t-1}, x_t] + b_z), && \text{(update gate)} \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t.
\end{aligned}
$$

- z_t controls how much old information is kept – just the same as the forget gate in LSTM.
- z_t and r_t collectively controls how much new information entered. *There is a strong coupling between the amount of information retained and the amount of information entered.*

Which is Best?

- There are many more LSTM variants.
- Greff, Srivastava, Koutník, Steunebrink, and Schmidhuber, 2016 compared eight LSTM variants on speech recognition, handwriting recognition, and polyphonic music modelling.
- Highlight: none of the variants is significantly better than the standard LSTM.

Greff, Srivastava, Koutník, Steunebrink, and Schmidhuber, LSTM: A search space odyssey, 2016

Your Turn

Which of the following statement is correct? (Multiple choice)

(a) The long short-term memory in LSTM refers to the activation values of the hidden neurons.

(b) Both LSTM and GRU have mechanisms to control how much old information need to be retained and how much new information need to be stored.

(c) Updating the hidden states in GRU is about 25% faster than in LSTM (when input size and number of hidden neurons are the same).

Initial Hidden State

- We need to specify the initial hidden activations.
- One approach is to fix them to default values like 0's.
- Alternatively, we can treat them as learned parameters, and learn them in the same way as we learn the weights.

What You Need to Know...

- Techniques to alleviate exploding and vanishing gradient problem
  - Truncated BPTT, good initialization, gradient clipping, design a better architecture
- LSTM architecture
- GRU architecture
- Setting initial hidden state