Long Short-Term Memory (LSTM)

Ravi Kothari, Ph.D.
ravi.kothari@ashoka.edu.in

"Time moves in one direction, memory in another" – William Gibson

Examples

Examples

- Some tasks require memory i.e. require you to observe the history to be able to understand what is going on

Examples

- Some tasks require memory i.e. require you to observe the history to be able to understand what is going on
- Is the person on the way down or the way up?

Examples

- Some tasks require memory i.e. require you to observe the history to be able to understand what is going on
- Is the person on the way down or the way up?

- However, if you had observed the past you could easily say whether the person was on the way down or the way up

- Another example. Your understanding of a sentence does not just depend on the word you are reading but also on the past words (and in some cases words that are yet to be read)

- Another example. Your understanding of a sentence does not just depend on the word you are reading but also on the past words (and in some cases words that are yet to be read)

- Feed-forward neural networks cannot be used when the data originates from a dynamical system. For example, a CNN cannot produce the correct output on seeing the image of the person squatting. Why?

- Another example. Your understanding of a sentence does not just depend on the word you are reading but also on the past words (and in some cases words that are yet to be read)
- Feed-forward neural networks cannot be used when the data originates from a dynamical system. For example, a CNN cannot produce the correct output on seeing the image of the person squatting. Why?
- We thus turn our attention to neural networks that can *store* the past and that respond based on the present input as well as what occurred in the past

Recurrent Neural Networks

Recurrent Neural Networks

- To store the past, we need to introduce feedback connections in a neural network. The feedback can be implicit or explicit

Recurrent Neural Networks

- To store the past, we need to introduce feedback connections in a neural network. The feedback can be implicit or explicit
  - Tapped-delay line – store the past inputs explicitly

Recurrent Neural Networks

- To store the past, we need to introduce feedback connections in a neural network. The feedback can be implicit or explicit
  - Tapped-delay line – store the past inputs explicitly
  - Context networks – store the past state e.g. feed the next layers output to augment one of the previous layers

Recurrent Neural Networks

- To store the past, we need to introduce feedback connections in a neural network. The feedback can be implicit or explicit
  - Tapped-delay line – store the past inputs explicitly
  - Context networks – store the past state e.g. feed the next layers output to augment one of the previous layers
  - Fully recurrent networks allowing connections between arbitrary neurons

Recurrent Neural Networks

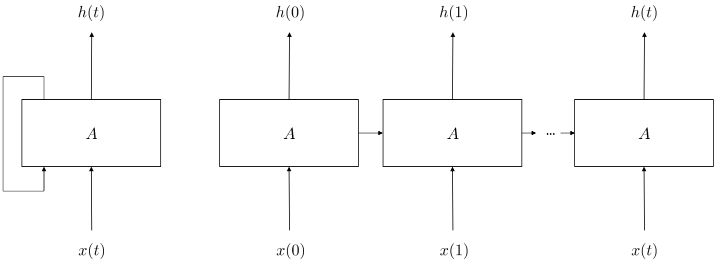
- To store the past, we need to introduce feedback connections in a neural network. The feedback can be implicit or explicit
  - Tapped-delay line – store the past inputs explicitly
  - Context networks – store the past state e.g. feed the next layers output to augment one of the previous layers
  - Fully recurrent networks allowing connections between arbitrary neurons
  - The idea is to concentrate information in time

- Back-propagation through time was developed to train neural networks with feedback connections

- Back-propagation through time was developed to train neural networks with feedback connections
  - There are issues of convergence with fully recurrent networks. Why?

- Back-propagation through time was developed to train neural networks with feedback connections
    - There are issues of convergence with fully recurrent networks. Why?
    - Most recurrent networks thus unfold time i.e. allow the feedback signal to persist for a limited time and then use standard back-propagation (gradient descent) to train the network

Unfolding Time



$$
\begin{aligned}
h(t) &= f\left[W_t x(t) + G_t h(t-1)\right] \\
&= f\left[W_t x(t) + G_t\left(f(W_{t-1}x(t-1) + G_{t-1}x(t-2))\right)\right] \\
\ldots &= \ldots
\end{aligned}
\tag{1}
$$

The Vanishing Gradient Problem

The Vanishing Gradient Problem

- We have,

$$
\begin{aligned}
h(t) &= \sigma\left[W_t x(t) + G_t h(t-1)\right] \\
&= \sigma\left[W_t x(t) + G_t\left(\sigma(W_{t-1} x(t-1) + G_{t-1} x(t-2))\right)\right] \\
\dots &= \dots
\end{aligned} \tag{2}
$$

where, σ is the sigmoid function, W and G denote the weights

The Vanishing Gradient Problem

- We have,

$$\begin{aligned} h(t) &= \sigma\left[W_t x(t) + G_t h(t-1)\right] \\ &= \sigma\left[W_t x(t) + G_t\left(\sigma(W_{t-1}x(t-1) + G_{t-1}x(t-2))\right)\right] \\ \dots &= \dots \end{aligned} \tag{2}$$

  where, σ is the sigmoid function, W and G denote the weights

- The gradient requires the multiplication of many many terms (increases with increasing history that we have to preserve)

The Vanishing Gradient Problem

- We have,

$$\begin{aligned} h(t) &= \sigma\left[W_t x(t) + G_t h(t-1)\right] \\ &= \sigma\left[W_t x(t) + G_t\left(\sigma(W_{t-1}x(t-1) + G_{t-1}x(t-2))\right)\right] \\ \ldots &= \ldots \end{aligned} \quad (2)$$

  where, σ is the sigmoid function, W and G denote the weights

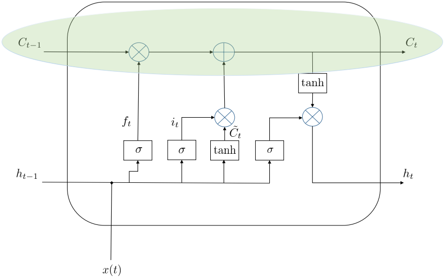
- The gradient requires the multiplication of many many terms (increases with increasing history that we have to preserve)

- The gradient vanishes after a few terms!

Elements of LSTM - Cell State

Elements of LSTM - Cell State

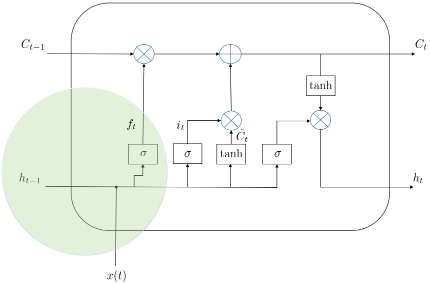
- LSTM's introduce the notion of a cell-state. Each cell has the ability to add to or remove from the cell state using gating mechanisms



Elements of LSTM - Forgetting

Elements of LSTM - Forgetting

- LSTM's introduce the notion of a cell-state. Each cell has the ability to add to or remove from the cell state using gating mechanisms



$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3}$$

Elements of LSTM – Updating the Cell State

Elements of LSTM – Updating the Cell State

- It has two parts,

Elements of LSTM – Updating the Cell State

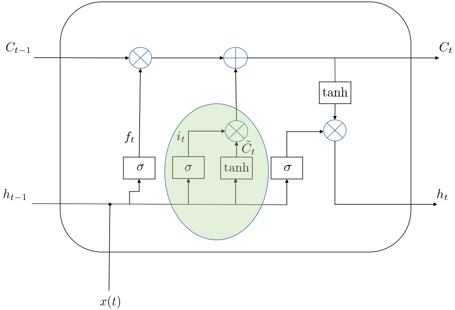
- It has two parts,
  - The input gate layer decides which values to update

Elements of LSTM – Updating the Cell State

- It has two parts,
  - The input gate layer decides which values to update
  - The tanh layer creates a vector of new candidate values $\tilde{C}_t$

Elements of LSTM – Updating the Cell State

- It has two parts,
  - The input gate layer decides which values to update
  - The tanh layer creates a vector of new candidate values $\tilde{C}_t$



$$
\begin{aligned}
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_{t-1}
\end{aligned} \tag{4}
$$

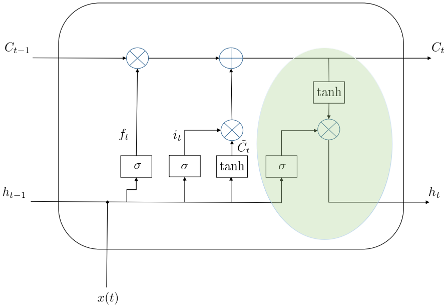
Elements of LSTM – Deciding the Output

Elements of LSTM – Deciding the Output

- The output is based on the cell state,

Elements of LSTM – Deciding the Output

- The output is based on the cell state,



$$
\begin{aligned}
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
\tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \\
h_t &= o_t * \tanh(C_t) \qquad (5)
\end{aligned}
$$

Demo

Hand-writing demo:
http://www.cs.toronto.edu/~graves/handwriting.html
Music composition: http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/
Image captioning:
http://karpathy.github.io/2015/05/21/rnn-effectiveness/