Recurrent Neural Networks

Dit-Yan Yeung

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

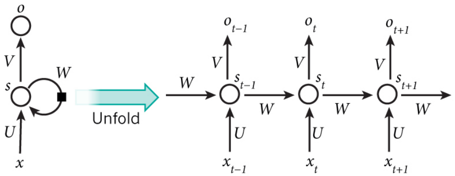
COMP 4211: Machine Learning (Fall 2022)

Introduction

- Recurrent neural network (RNN) models are extensions of feedforward neural networks for handling sequential data typically involving variable-length input or output sequences.
- There is weight sharing in an RNN such that the weights are shared across different instances of the units corresponding to different time steps.
- Weight sharing is important for processing variable-length sequences so that the absolute time step at which an event occurs does not matter but the context (relative to some other events) in which an event occurs is more important and relevant.

Recurrent Units

- A recurrent unit is a unit with a recurrent connection from its output back to itself.
- By unrolling the network through time, we can get an equivalent unfolded representation:



- Multiple recurrent units can form a layer which can express its computation in vector form as:

$$\mathbf{s}_t = g(\mathbf{U}\,\mathbf{x}_t + \mathbf{W}\,\mathbf{s}_{t-1} + \mathbf{b}),$$

where $g(\cdot)$ denotes the activation function such as ReLU.

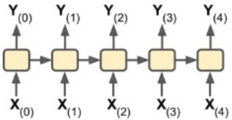
Memory Cells

- A memory cell (or simply cell) is a part of a neural network that preserves some state across time steps.
- Since $\mathbf{s}_t$ depends directly on the current input $\mathbf{x}_t$ and indirectly on all the previous inputs $\mathbf{x}_{1:t-1}$, a recurrent unit is a memory cell though of a very simple type.
- Later in this topic, we will consider some more powerful memory cells which overcome some of the limitations of simple memory cells for RNN models.

Sequence-to-Sequence Learning

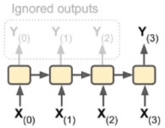
Conventional RNN architecture:

- The input and output sequences must align in time steps.
- The input and output sequences are of the same length.



Sequence-to-vector network (encoder):

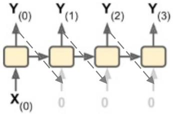
- All outputs except the last one are ignored.
- The input sequence is encoded by the last output.



Sequence-to-Sequence Learning (2)

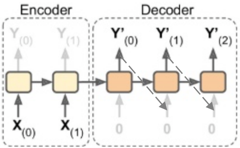
Vector-to-sequence network (decoder):

- There is only a single input at the first time step.
- The first input is decoded to give the output sequence.



Encoder-decoder RNN architecture:

- It is composed of an encoder followed by a decoder.
- The input and output sequences do not have to be of the length.

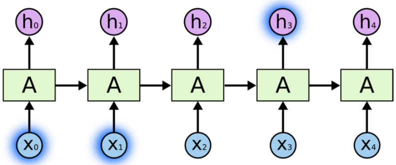


Backpropagation Through Time

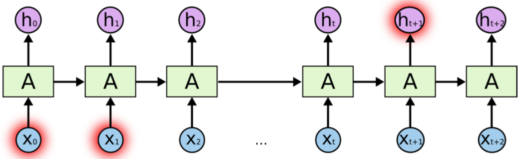
- Backpropagation through time (BPTT) is the strategy of training an RNN by unrolling it through time and then applying the BP algorithm.
- Like deep feedforward neural networks, an RNN may stack multiple layers of recurrent units (or cells) to give a deep RNN.
- The loss function is defined using all the outputs, not just the last output.
- The same network parameters are shared across all time steps.

Dependencies between Events

Short-term dependencies:



Long-term dependencies:



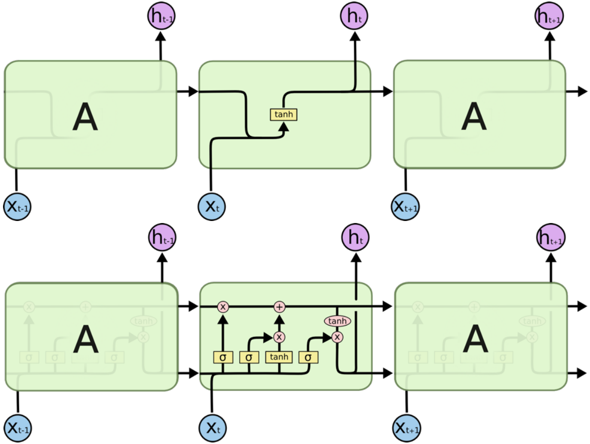
Learning Long Sequences

- Training an RNN on long sequences is challenging because the unrolled RNN is a very deep network which may suffer from the vanishing/exploding gradient problem.
- The vanishing gradient problem can be addressed by the techniques discussed before for deep feedforward neural networks.
- The exploding gradient problem can be addressed by gradient clipping which clips the gradients (e.g., normalizing the gradient vector by its L_2 norm when the L_2 norm exceeds a certain threshold) during BP learning so that they never exceed some threshold.
- Also, the memory of the first inputs gradually fades away for long sequences.
- Truncated BPTT unrolls the RNN only over a limited number of time steps.
- A better approach is to replace the simple recurrent units by more powerful memory cells which can handle long-term dependencies.

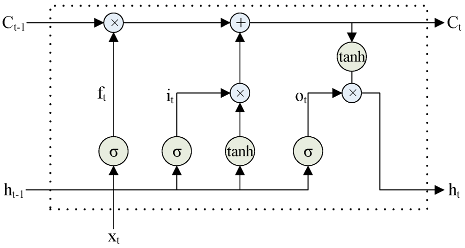
Long Short-Term Memory

- Long short-term memory (LSTM) cells can be used in place of the ordinary recurrent units in RNNs.
- Advantages of LSTM cells:
  - Faster convergence in training
  - More capable of detecting long-term dependencies in the sequences.
- The state $\mathbf{s}_t$ in an ordinary cell is now split into two vectors:
  - Long-term state: $\mathbf{c}_t$
  - Short-term state: $\mathbf{h}_t$
- The interplay between long-term state and short-term state is achieved through three gates in an LSTM cell:
  - Forget gate
  - Input gate
  - Output gate

RNNs with Ordinary Units vs. LSTM cells

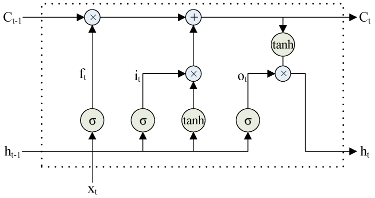


Three Gates in an LSTM Cell



- The long-term state $\mathbf{c}_{t-1}$ first goes through the forget gate which allows some old memories to be dropped.
- It then passes through the input gate which allows some new memories to be added to give the updated long-term state $\mathbf{c}_t$.
- The updated long-term state $\mathbf{c}_t$, after being transformed by tanh, flows through the output gate which controls the generation of the short-term state $\mathbf{h}_t$.

Computation in an LSTM Cell



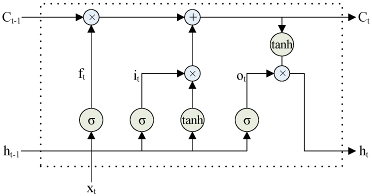
- Computation like in a basic cell:

$$\mathbf{g}_t = \tanh(\mathbf{W}_{gx}\,\mathbf{x}_t + \mathbf{W}_{gh}\,\mathbf{h}_{t-1} + \mathbf{b}_g).$$

- Three gate controllers:

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\,\mathbf{x}_t + \mathbf{W}_{fh}\,\mathbf{h}_{t-1} + \mathbf{b}_f)$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\,\mathbf{x}_t + \mathbf{W}_{ih}\,\mathbf{h}_{t-1} + \mathbf{b}_i)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\,\mathbf{x}_t + \mathbf{W}_{oh}\,\mathbf{h}_{t-1} + \mathbf{b}_o).$$

Computation in an LSTM Cell (2)



- Update of long-term state:

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t.$$

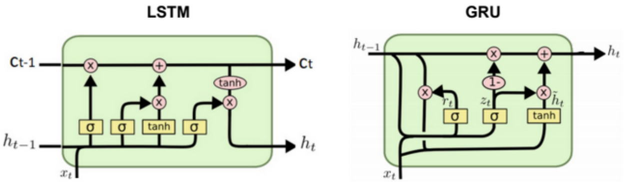
- Output of short-term state:

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t).$$

Gated Recurrent Unit

- The gated recurrent unit (GRU) is a simplified version of the LSTM cell with some simplifications:
  - The two state vectors are merged into $\mathbf{h}_t$.
  - One gate controller $\mathbf{z}_t$ controls both the forget gate and the input gate.
  - There is no output gate, but there is a new gate controller that controls which part of $\mathbf{h}_{t-1}$ will be shown to the main layer $\mathbf{g}_t$.

LSTM vs. GRU



- Computation in a GRU cell:

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_{zx}\,\mathbf{x}_t + \mathbf{W}_{zh}\,\mathbf{h}_{t-1} + \mathbf{b}_z) \\
\mathbf{r}_t &= \sigma(\mathbf{W}_{rx}\,\mathbf{x}_t + \mathbf{W}_{rh}\,\mathbf{h}_{t-1} + \mathbf{b}_r) \\
\mathbf{g}_t &= \tanh(\mathbf{W}_{gx}\,\mathbf{x}_t + \mathbf{W}_{gh}\,(\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{b}_g) \\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \circ \mathbf{h}_{t-1} + \mathbf{z}_t \circ \mathbf{g}_t.
\end{aligned}$$

To Learn More...

- Recurrent neural networks with attention mechanism
- Transformer networks