MO434 - Deep Learning
Recurrent Neural Networks

Alexandre Xavier Falcão

Institute of Computing - UNICAMP

afalcao@ic.unicamp.br

So far, we have seen that CNNs require fixed-size inputs. We will see now Recurrent Neural Networks (RNNs).

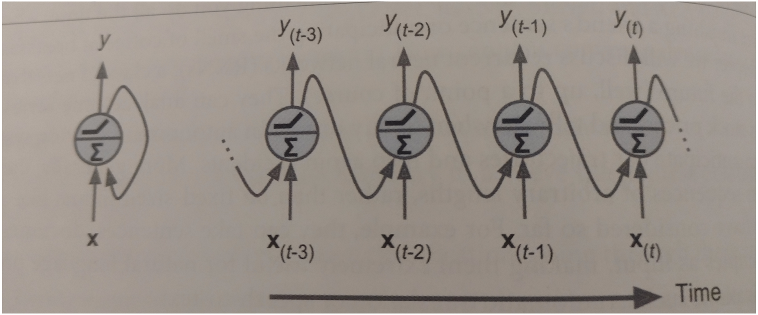
So far, we have seen that CNNs require fixed-size inputs. We will see now Recurrent Neural Networks (RNNs).

- RNNs can analyze sequences of arbitrary sizes – e.g., time series data, that can
  - anticipate car trajectories, avoiding accidents and
  - stock prices, telling you when to buy or sell, and text data,
  - predicting the next word of a sentence, translating sentences from one language to another, and classifying the sentiment about a movie review.
- We will also see extensions, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which address the limited short-term memory problem of RNNs.

Agenda

- RNN cells
- LSTM cells
- GRU cells
- Applications in Text Analysis

Let x be a feature vector that changes along time. For any instant t of time, a recurrent neuron (left) receives x and its output in $t-1$ as input and outputs an activation value y.

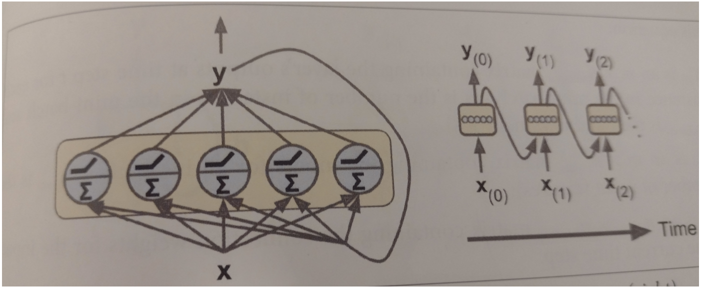


from Hands-On ML book.

We can unroll (unfold) this process over time as shown on the right. Activation is usually the hyperbolic tangent.

Recurrent Layer

Each neuron has two weight vectors, one for $x_{(t)}$ and the other for $y_{(t-1)}$. For a layer with multiple neurons, these vectors form two matrices, W_x and W_y.



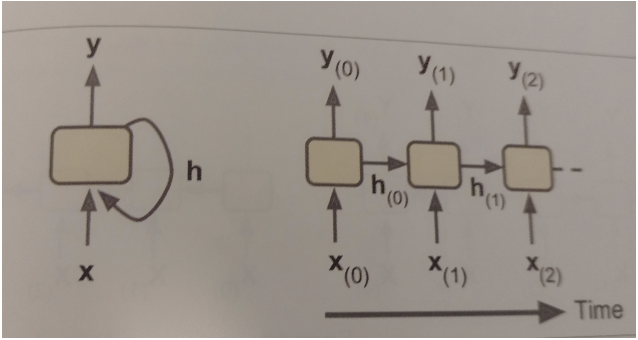
from Hands-On ML book.

For a bias vector b and activation ϕ, the output

$$y_{(t)} = \phi\left(W_x^t x_{(t)} + W_y^t y_{(t-1)} + b\right).$$

RNN Cell

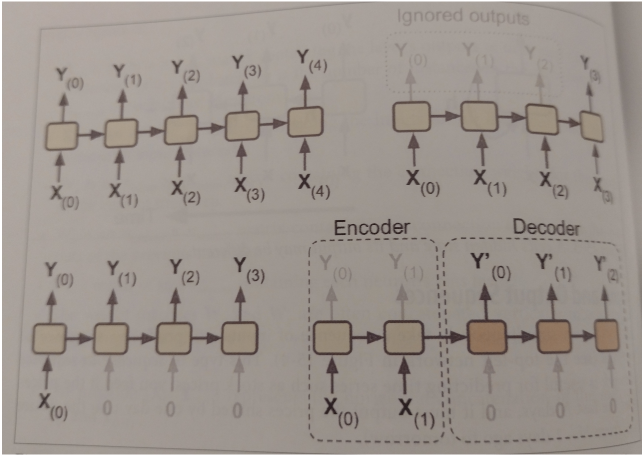
We call each recurrent neuron/layer a RNN memory cell.



from Hands-On ML book.

Its output y and hidden state h may also be different.

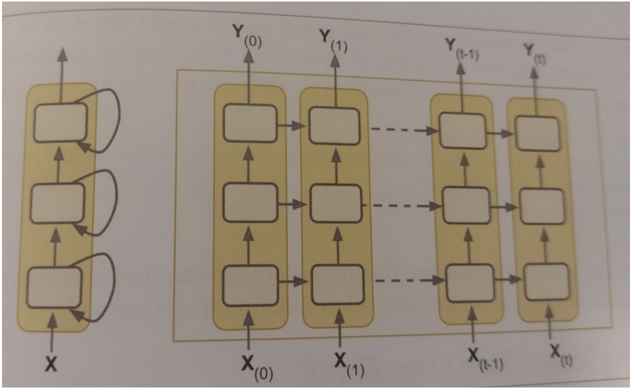
RNNs can convert one-to-many (image to caption), many-to-one (movie review to sentiment), and combine many-to-one with one-to-many to form an encoder-decoder (language translation).



from Hands-On ML book.

Deep RNN

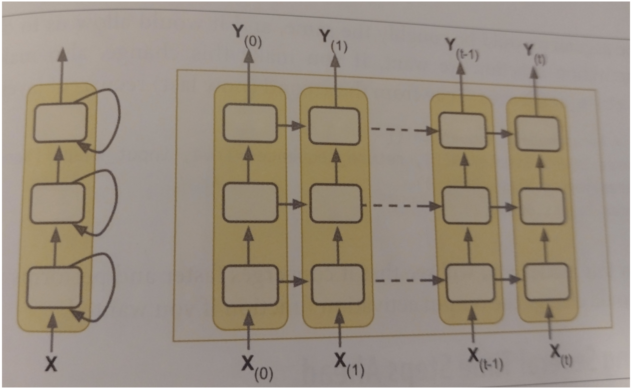
Multiple layers of cells can also be stacked to form a Deep RNN.



from Hands-On ML book.

Deep RNN

Multiple layers of cells can also be stacked to form a Deep RNN.



from Hands-On ML book.

However, due to the way data goes through an RNN, information is lost at every step, loosing trace of the first inputs after a few steps – short-memory problem.

Short-Memory Problem

$y_{(t)} = h_{(t)}$ depends on the output from all previous time steps.

$$y_{(t)} \quad = \quad \phi\left(W_x^t x_{(t)} + W_h^t h_{(t-1)} + b\right).$$

- If the weights in W_h^t are less than 1.0, $y_{(t)}$ will highly depend on $x_{(t)}$ – i.e., memory loss as the time increases and vanishing gradient.

Short-Memory Problem

$y_{(t)} = h_{(t)}$ depends on the output from all previous time steps.

$$y_{(t)} \quad = \quad \phi\left(W_x^t x_{(t)} + W_h^t h_{(t-1)} + b\right).$$

- If the weights in W_h^t are less than 1.0, $y_{(t)}$ will highly depend on $x_{(t)}$ – i.e., memory loss as the time increases and vanishing gradient.
- If the weights in W_h^t are greater than 1.0, $y_{(t)}$ will depend much less of $x_{(t)}$ – i.e., input loss as the time increases and exploding gradient.

Short-Memory Problem

$y_{(t)} = h_{(t)}$ depends on the output from all previous time steps.

$$y_{(t)} \quad = \quad \phi\left(W_x^t x_{(t)} + W_h^t h_{(t-1)} + b\right).$$

- If the weights in W_h^t are less than 1.0, $y_{(t)}$ will highly depend on $x_{(t)}$ – i.e., memory loss as the time increases and vanishing gradient.

- If the weights in W_h^t are greater than 1.0, $y_{(t)}$ will depend much less of $x_{(t)}$ – i.e., input loss as the time increases and exploding gradient.

Clearly, we have a problem when the prediction of a next word depends on far away inputs (long-term dependency). For instance, "I am from England. Bla bla bla ... I speak _______". The next word English depends on "speak" and "England" – a distant input.

LSTM cell

LSTM cells introduce a long-term memory state c and gate controllers to address the problem.

- A forgetting gate $f_{(t)}$, with logistic activation, which can set to zero (forget) part of the information in $c_{(t-1)}$ by the element-wise multiplication $c_{(t)} = c_{(t-1)} \otimes f_{(t)}$.

$$f_{(t)} = \psi\left(Wf_x^t x_{(t)} + Wf_h^t h_{(t-1)} + bf\right).$$

LSTM cell

LSTM cells introduce a long-term memory state c and gate controllers to address the problem.

- A forgetting gate $\mathsf{f}_{(t)}$, with logistic activation, which can set to zero (forget) part of the information in $\mathsf{c}_{(t-1)}$ by the element-wise multiplication $\mathsf{c}_{(t)} = \mathsf{c}_{(t-1)} \otimes \mathsf{f}_{(t)}$.

$$\mathsf{f}_{(t)} = \psi\left(\mathsf{Wf}_x^{\mathsf{t}}\mathsf{x}_{(t)} + \mathsf{Wf}_h^{\mathsf{t}}\mathsf{h}_{(t-1)} + \mathsf{bf}\right).$$

- An ignoring (input) gate $\mathsf{i}_{(t)}$, with logistic activation, which can set to zero (ignore by $\mathsf{g}_{(t)} \otimes \mathsf{i}_{(t)}$) part of the information in $\mathsf{g}_{(t)}$ – the output similar to an RNN cell with tanh activation.

$$\begin{aligned}
\mathsf{g}_{(t)} &= \phi\left(\mathsf{Wg}_x^{\mathsf{t}}\mathsf{x}_{(t)} + \mathsf{Wg}_h^{\mathsf{t}}\mathsf{h}_{(t-1)} + \mathsf{bg}\right), \\
\mathsf{i}_{(t)} &= \psi\left(\mathsf{Wi}_x^{\mathsf{t}}\mathsf{x}_{(t)} + \mathsf{Wi}_h^{\mathsf{t}}\mathsf{h}_{(t-1)} + \mathsf{bi}\right).
\end{aligned}$$

LSTM cell

Finally, the long-term state passed to the next time step is $\mathsf{c}_{(t)} = (\mathsf{c}_{(t-1)} \otimes \mathsf{f}_{(t)}) \oplus (\mathsf{g}_{(t)} \otimes \mathsf{i}_{(t)})$, where $\oplus$ is the element-wise addition.

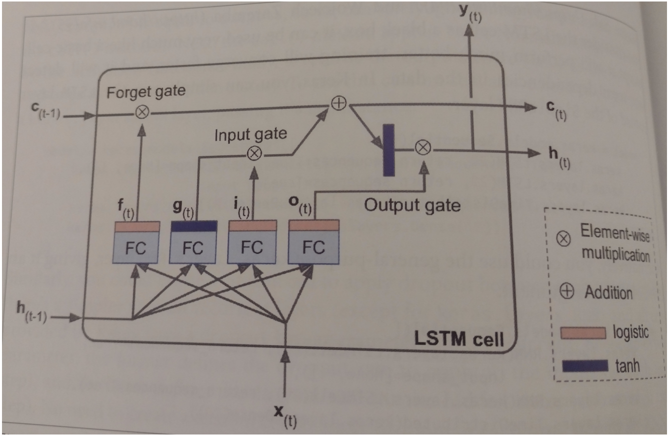
Finally, the long-term state passed to the next time step is $c_{(t)} = (c_{(t-1)} \otimes f_{(t)}) \oplus (g_{(t)} \otimes i_{(t)})$, where $\oplus$ is the element-wise addition.

In order to define the output $y_{(t)}$ and the hidden state $h_{(t)}$ passed to the next time step, a selection (output) gate selects by $h_{(t)} = o_{(t)} \otimes \phi\left(c_{(t)}\right)$ the tanh-activated parts of $c_{(t)}$, where

$$o_{(t)} = \psi\left(Wo_x^t x_{(t)} + Wo_h^t h_{(t-1)} + bo\right).$$

LSTM

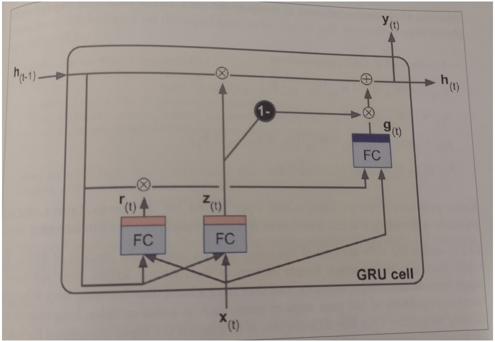
This figure illustrates all operations such that one logistic activation ϕ per neuron at the gates can keep parts of them open with 1's or closed with 0's.



from Hands-On ML book.

GRU

GRU simplifies LSTM and seems to perform just as well.



from Hands-On ML book.

- It uses a gate $r_{(t)}$ to select which parts of $h_{(t-1)}$ will be presented to the main layer.
- Another gate $z_{(t)}$ substitutes $f_{(t)}$ and $i_{(t)}$ – it forgets some parts of $h_{(t-1)}$ and the complementary parts of $g_{(t)}$ to output $h_{(t)} = (h_{(t-1)} \otimes z_{(t)}) \oplus (g_{(t)} \otimes (1 - z_{(t)}))$.

Applications in Text Analysis

Let's see a couple of applications in Text Analysis.

- Sentiment Analysis (SENTIMENT ANALYSIS).
- Image Captioning (IMAGE CAPTIONING).