CS 446/ECE 449: Machine Learning

Shenlong Wang

University of Illinois at Urbana-Champaign, 2024

Logistics: Mid-term Exam

- Exam: regular class time & location, Thur, Mar 7.
- Reviewing session: regular class time & location, Tue, Mar 5, led by Jane.
- You are allowed to bring notes on two letter-sized (or A4-sized) sheets of paper (you may use both sides).
- 70 minutes long, 5 parts, each part containing 2-5 short questions.
- Covering lectures 1-15. (Today, we have two lecture slides.)

Sequential modeling (RNNs/LSTMs/GRUs)

Goals of this lecture

- Getting to know Recurrent Neural Nets (RNNs)
- Getting to know Long short term memory (LSTM)
- Getting to know Gated recurrent unit (GRU)
- Getting to know autoregressive generative models (PixelRNN)

Reading Material

- Goodfellow et al.; Deep Learning; Chapter 10
- Papers cited on the slides

What is Sequential Data?

- Data arrives and is processed in order.
- $x = (x^{(0)}, x^{(1)}, ...x^{(t)}...)$.

Could you name a few applications that, in your opinion, utilize deep sequential modeling?

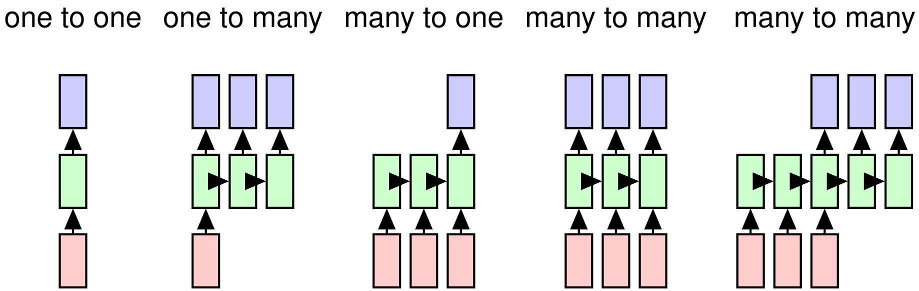
Applications:

- Weather forecasting
- Natural language processing
- Speech recognition
- Video processing
- Financial prediction

More flexibility regarding inputs and outputs:

- Sequences of inputs
- Sequences of outputs

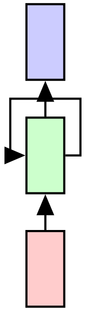
Length of sequences may vary



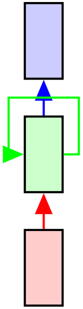
Recurrent Neural Nets (RNNs)

- input depends on previous output

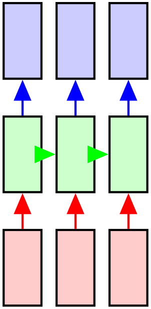
$$h^{(t)} = f(h^{(t-1)}, x^{(t)}, \boldsymbol{w})$$



Important concept: **Parameter sharing**



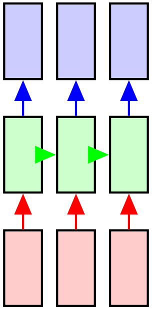
$\rightarrow$



$$\begin{aligned} h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\ y^{(t)} &= g(h^{(t)}) \end{aligned}$$

unfolded/unrolled network
performs identical operations
easier to understand

General structure for recurrence:



Mathematical description in general:

$$\begin{aligned} h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\ y^{(t)} &= g(h^{(t)}) \end{aligned}$$

Note that *f* and *g* are independent of time

What are *f* and *g*?

Any differentiable function can be used

Useful functions:

- Original recurrent nets
- LSTM nets
- GRU nets

Original recurrent nets (Elman network):

(Jordan network is slightly different)

Generally:

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}, \boldsymbol{w})$$

$$y^{(t)} = g(h^{(t)})$$

Specifically:

$$h^{(t)} = \sigma_h(W_{hx}x^{(t)} + W_{hh}h^{(t-1)} + w_{hb})$$

$$y^{(t)} = \sigma_y(W_{yh}h^{(t)} + w_{yb})$$

What is σ_h and σ_y?

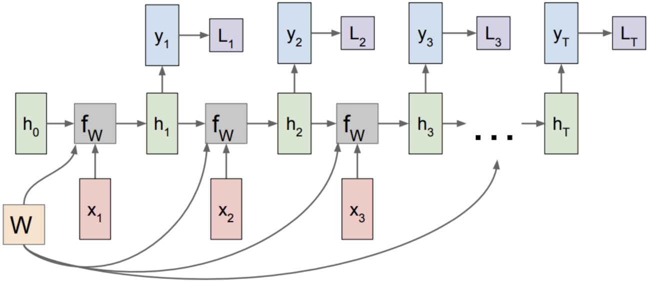
Activation functions: `tanh`, sigmoid

Affine transformations and point-wise non-linearity

What are the problems?

How to learn?

- Back-propagation, but not that simple...
- Same weights **w** influences every step $h^{(t)}$



- It's called back-propagation through time (BPTT)

How to learn?

Recall:

- $h^{(1)} = f(h^{(0)}, x^{(1)}, \boldsymbol{w})$
- $h^{(2)} = f(h^{(1)}, x^{(2)}, \boldsymbol{w})$
- $h^{(3)} = f(h^{(2)}, x^{(3)}, \boldsymbol{w})$
- $y^{(3)} = g(h^{(3)})$

We should be able to compute $\frac{\partial L^{(3)}}{\partial h^{(3)}} = \frac{\partial L^{(3)}}{\partial y^{(3)}} \frac{\partial y^{(3)}}{\partial h^{(3)}}$. Now let's consider $\frac{\partial L}{\partial \boldsymbol{w}}$:

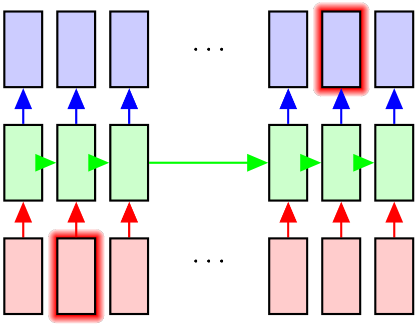
$$\frac{\partial L^{(3)}}{\partial \boldsymbol{w}} = \frac{\partial L^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial \boldsymbol{w}} + \frac{\partial L^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial \boldsymbol{w}} + \frac{\partial L^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial \boldsymbol{w}}$$

In other word, we have the following BPTT rule:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial L^{(t)}}{\partial h^{(t)}} \left(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(k)}}{\partial \boldsymbol{w}}$$

Problems with classical recurrent neural nets:

- Vanishing gradients
- Long term dependency



Long short term memory (LSTM)

- Particular functional relation
- Shown to better capture long-term dependencies
- Shown to address the vanishing gradient problem

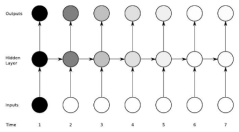


Figure 4.1: **Vanishing gradient problem for RNNs.** The shading of the nodes indicates the sensitivity over time of the network nodes to the input at time one (the darker the shade, the greater the sensitivity). The sensitivity decays exponentially over time as new inputs overwrite the activation of hidden unit and the network 'forgets' the first input.

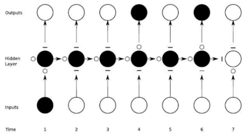


Figure 4.3: **Preservation of gradient information by LSTM.** As in Figure 4.1 the shading of the nodes indicates their sensitivity to the input unit at time one. The state of the input, forget, and output gate states are displayed below, to the left and above the hidden layer node, which corresponds to a single memory cell. For simplicity, the gates are either entirely open ('O') or closed ('—'). The memory cell 'remembers' the first input as long as the forget gate is open and the input gate is closed, and the sensitivity of the output layer can be switched on and off by the output gate without affecting the cell.

Long short term memory (LSTM)

- Particular functional relation
- Shown to better capture long-term dependencies
- Shown to address the vanishing gradient problem

Generally:

$$\begin{aligned} h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\ y^{(t)} &= g(h^{(t)}) \end{aligned}$$

Specifically: ($\circ$ denotes Hadamard product; σ is activation function)

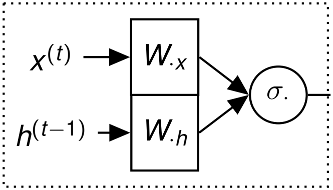
$$\begin{aligned} i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\ f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate} \\ o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate} \\ \tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell} \\ c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell} \\ h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)}) \end{aligned}$$

Equations:

$$\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate}\\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate}\\
o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate}\\
\tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell}\\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell}\\
h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)})
\end{aligned}$$

Intuition:

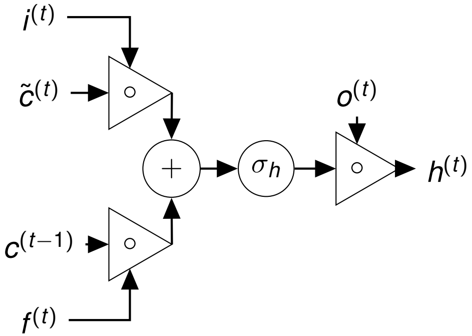
- $i^{(t)}$: Does $x^{(t)}$ matter?
- $f^{(t)}$: Should $c^{(t-1)}$ be forgotten?
- $o^{(t)}$: How much $c^{(t)}$ should be exposed?
- $\tilde{c}^{(t)}$: Compute new memory



Equations:

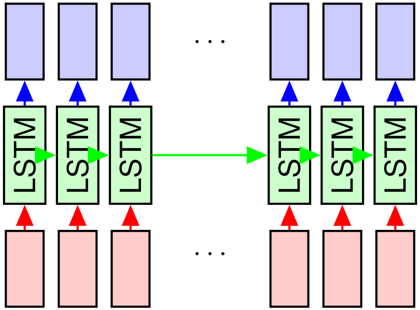
$$\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate}\\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate}\\
o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate}\\
\tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell}\\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell}\\
h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)})
\end{aligned}$$

Intuition:



Long short term memory (LSTM):

- Can be interpreted as a block in a neural net



Gated recurrent unit (GRU):

- Performance similar to LSTM
- Fewer parameters compared to LSTM (no output gate)

Equations: ($\circ$ denotes Hadamard product)

$$\begin{aligned}
z^{(t)} &= \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) && \text{Update gate}\\
r^{(t)} &= \sigma_r(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + w_{br}) && \text{Reset gate}\\
\tilde{h}^{(t)} &= \sigma_h(W_{hx}x^{(t)} + W_{rwh}(r^{(t)} \circ h^{(t-1)}) + w_{bh}) && \text{New memory cell}\\
h^{(t)} &= (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)} && \text{Hidden state}
\end{aligned}$$

Can again be interpreted as a block in the computation graph

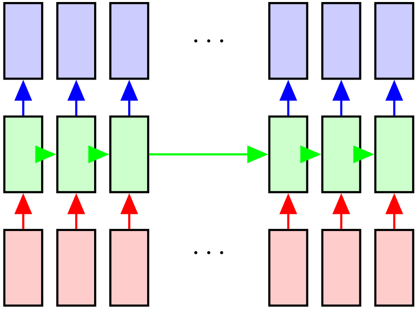
Equations: ($\circ$ denotes Hadamard product)

$$\begin{array}{rcll}
z^{(t)} & = & \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) & \text{Update gate} \\
r^{(t)} & = & \sigma_r(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + w_{br}) & \text{Reset gate} \\
\tilde{h}^{(t)} & = & \sigma_h(W_{hx}x^{(t)} + W_{rwh}(r^{(t)} \circ h^{(t-1)}) + w_{bh}) & \text{New memory cell} \\
h^{(t)} & = & (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)} & \text{Hidden state}
\end{array}$$

Intuition:

- $r^{(t)}$: Include $h^{(t-1)}$ in new memory?
- $z^{(t)}$: How much $h^{(t-1)}$ in next state?

Recurrent nets generally:



Other variants:

- Bi-directional LSTMs [Schuster&Paliwal (1997), Graves&Schmidhuber (2005)]
- Seq2Seq [Sutskever (2014), Talk: https://www.youtube.com/watch?v=-uyXE7dY5H0]
- Continuous time RNNs

Quiz:

- Describe the prediction process for an RNN?
- Describe the training process for RNNs?