Recurrent Neural Networks

CMPT 419/726
Mo Chen
SFU Computing Science
Mar. 16, 2020

Goodfellow, Bengio, and Courville: Deep Learning textbook
Ch. 10

Sequential Data with Neural Networks

- Sequential input / output
  - Many inputs, many outputs $x_{1:T} \rightarrow y_{1:S}$
    - e.g. object tracking, speech recognition with HMMs; on-line/batch processing
  - One input, many outputs $x \rightarrow y_{1:S}$
    - e.g. image captioning
  - Many inputs, one output $x_{1:T} \rightarrow y$
    - e.g. video classification

Outline

Outline

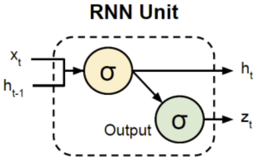
Recurrent Neural Networks

Long Short-Term Memory

Temporal Convolutional Networks

Examples

Hidden State

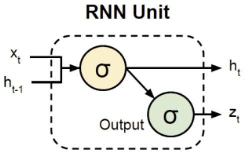


- Basic idea: maintain a state $\boldsymbol{h}_t$
- State at time t depends on input $\boldsymbol{x}_t$ and previous state $\boldsymbol{h}_{t-1}$
- It's a neural network, so relation is non-linear function of these inputs and some parameters $\boldsymbol{W}$:

$$h_t = f_x(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{W}) = \sigma(W_x x_t + W_h h_{t-1})$$

- Parameters $\boldsymbol{W}$ and function $f(\cdot)$ reused at all time steps

Outputs



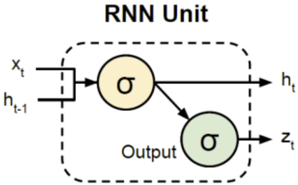
- Output z_t also depends on the hidden state:

$$z_t = f_z(\boldsymbol{h}_t; \boldsymbol{W}_z) = \sigma(W_z h_t)$$

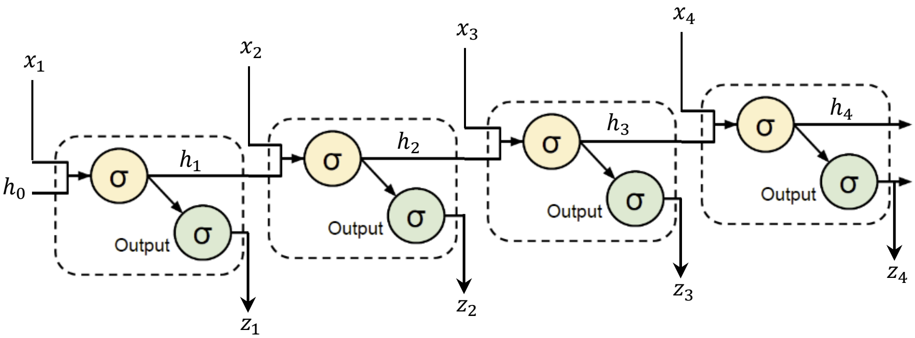
- Again, parameters/function reused across time

Recurrent neural network (RNN)

- Has feedback loops to capture temporal or sequential information
- Has the ability to learn tasks that require "memory" of events from many time steps ago
- Long short-term memory (LSTM): special type of RNN with advantages in numerical properties



Unfolding an RNN



$h_t = f_x(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{W})$

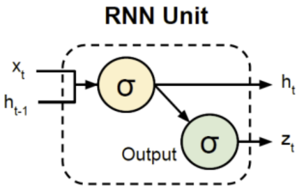
h_0 can be a vector of all zeros, or trained

$z_t = f_z(\boldsymbol{h}_t; \boldsymbol{W}_z)$

Vanishing Gradients in RNN

$$h_t = \sigma(W_x x_t + W_h h_{t-1}) \qquad z_n = f(\boldsymbol{h}_n; \boldsymbol{W}_z)$$

- Training requires $\frac{\partial E_n}{\partial W_x} = \sum_{i=1}^{n} \frac{\partial E_n}{\partial z_n} \frac{\partial z_n}{\partial h_n} \frac{\partial h_n}{\partial h_i} \frac{\partial h_i}{\partial W_x}$
  - Everything except $\frac{\partial h_n}{\partial h_i}$ involves "nearby" variables, so focus on $\frac{\partial h_n}{\partial h_i}$



$$\frac{\partial h_n}{\partial h_i} = \frac{\partial h_n}{\partial h_{n-1}} \frac{\partial h_{n-1}}{\partial h_{n-2}} \cdots \frac{\partial h_{i+1}}{\partial h_i} = \prod_{t=i}^{n-1} \frac{\partial h_{t+1}}{\partial h_t}$$

$$\frac{\partial h_{t+1}}{\partial h_t} = \frac{\partial}{\partial h_t} f_x(\boldsymbol{h}_t, \boldsymbol{x}_{t+1}; \boldsymbol{W}_x)$$

$$= \sigma'(W_x x_{t+1} + W_h h_t) W_h$$

$$\frac{\partial h_n}{\partial h_i} = \prod_{t=i}^{n-1} \sigma'(W_x x_{t+1} + W_h h_t) W_h$$

$$= W_h^{n-1} \prod_{t=i}^{n-1} \sigma'(W_x x_{t+1} + W_h h_t)$$

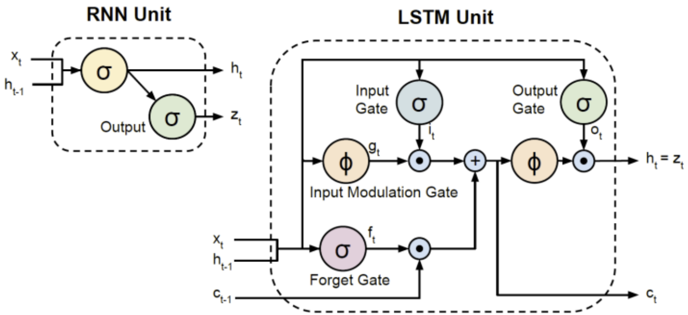
- Gradient blows up if largest eigenvalue of W_h is larger than 1
- Gradient vanishes otherwise

Gradients

- Basic RNN not very effective
- Need many time steps / complex model for challenging tasks
- Gradients in learning are a problem
  - Too large: can be handled with gradient clipping (truncate gradient magnitude)
  - Too small: can be handled with network structures / gating functions (LSTM, GRU)

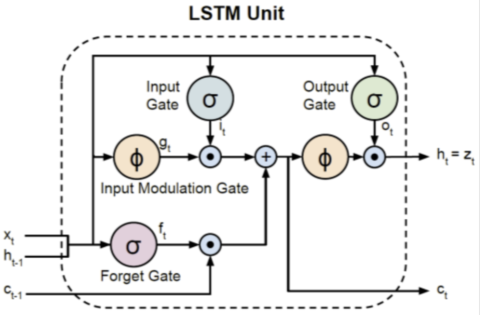
Outline

Long Short-Term Memory



- Hochreiter and Schmidhuber, Neural Computation 1997
  - (Figure from Donohue et al. CVPR 2015)
- Gating functions $g(\cdot), f(\cdot), o(\cdot)$ reduce vanishing gradients

Long Short-Term Memory



$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \tag{2}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{3}$$

$$g_t = \tanh(W_{sc}x_t + W_{hc}h_{t-1} + b_c) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{5}$$

$$h_t = o_t \odot \tanh c_t \tag{6}$$

see Graves, Liwicki, Fernandez, Bertolami, Bunke, and Schmidhuber, TPAMI 2009

Long Short-Term Memory

Let's consider $\frac{\partial c_t}{\partial c_{t-1}}$

- Full derivative of error function w.r.t. weights will be a product of these
- Note $c_t = c_t(f_t, c_{t-1}, i_t, g_t)$, where f_t, i_t, g_t also depend on c_{t-1}

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
g_t &= \tanh(W_{sc}x_t + W_{hc}h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh c_t
\end{aligned}$$

$$\begin{aligned}
\frac{\partial c_t}{\partial c_{t-1}} &= \frac{\partial c_t}{\partial f_t}\frac{\partial f_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial c_{t-1}} + \frac{\partial c_t}{\partial i_t}\frac{\partial i_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial g_t}\frac{\partial g_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial c_{t-1}} \\
&= c_{t-1}\sigma'(\cdots)W_{hf}o_{t-1}\tanh'(c_{t-1}) + f_t + g_t\sigma'(\cdots)W_{hi}\tanh'(c_{t-1}) + i_t\tanh'(\cdots)W_{hc}\tanh'(c_{t-1})
\end{aligned}$$

Discussion:

- When repeatedly multiplying four different terms added together, there is a smaller chance of vanishing compared to a single term
- f_t can be chosen to be larger or smaller, depending on whether the gradients should propagate backwards to before stage t
- f_t is learned, so neural network learns to control gradient propagation
- Vanishing gradients is *alleviated*, not solved
- Gradients may still explode as well

Outline

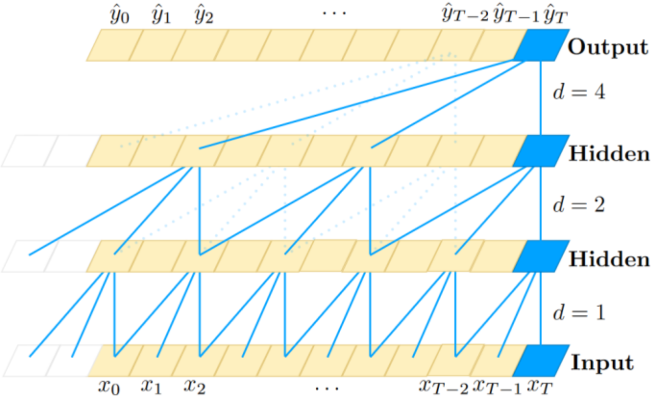
Recurrent Neural Networks

Long Short-Term Memory

Temporal Convolutional Networks

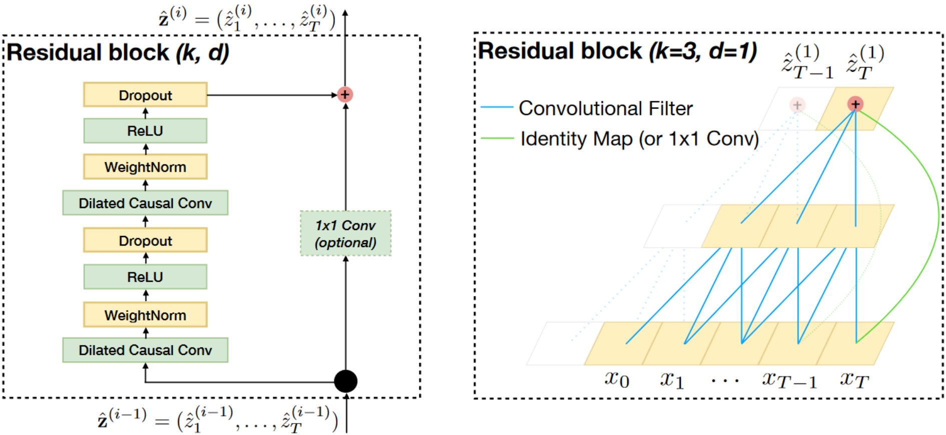
Examples

Convolutions to Aggregate over Time



- Control history by d (dilation, holes in the filter) and k (width of the filter)
- Causal convolution, only use elements from the past
- Bai, Kolter, Koltun arXiv 2018

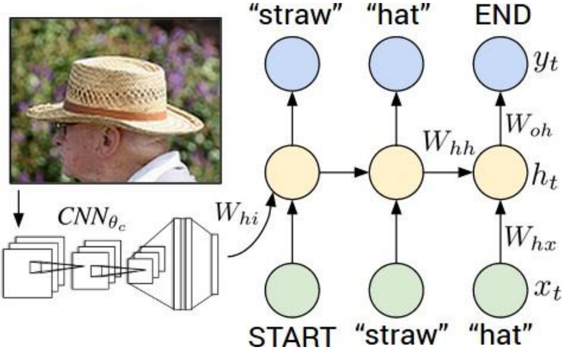
Residual (skip) Connections



- Include residual connections to allow long-range modeling and gradient flow

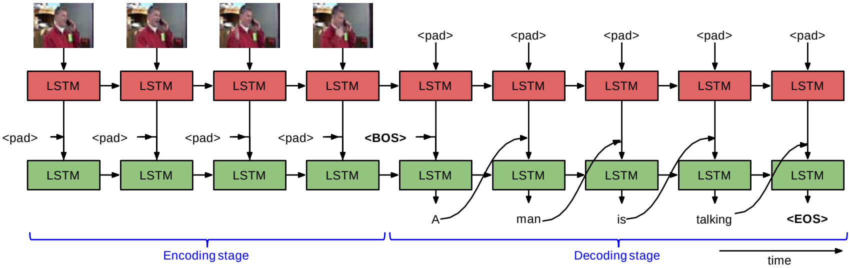
Outline

Example: Image Captioning



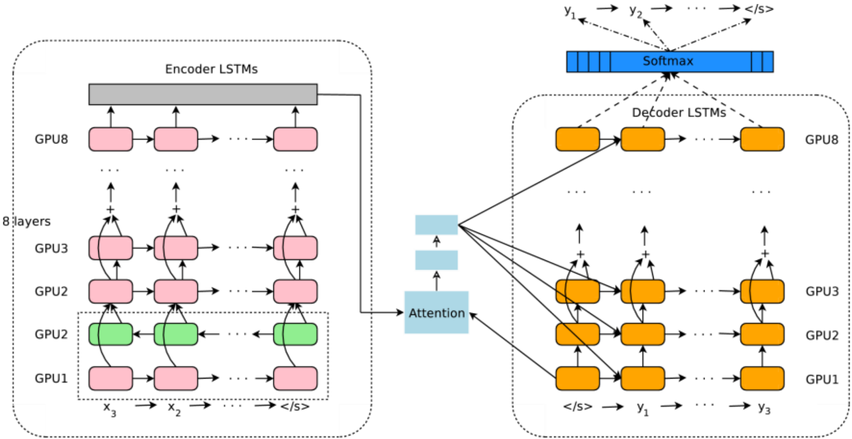
- Karpathy and Fei-Fei, CVPR 2015

Example: Video Description



- S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, K. Saenko, ICCV 2015

Example: Machine Translation



- Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*, arXiv 2016

Conclusion

- Readings:
  - http://www.deeplearningbook.org/contents/rnn.html
  - https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577
  - https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html
- Recurrent neural networks, can model sequential inputs/outputs
  - Input includes state (output) from previous time
  - Different structures:
    - RNN with multiple inputs/outputs
    - Gated recurrent unit (GRU)
    - Long short-term memory (LSTM)
  - Error gradients back-propagated across entire sequence