Recurrent Neural Networks

Advanced Topics in High-Performance Computing

Faisal Qureshi

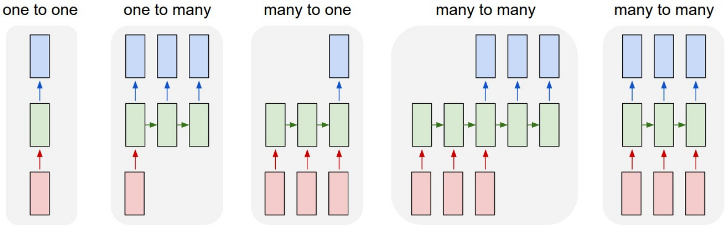
UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Copyright information

These slides draw heavily upon works of many individuals, notably among them are:

- Nando de Freitas
- Fei-Fei Li
- Andrej Karpathy
- Justin Johnson

Recurrent Neural Networks (RNN)

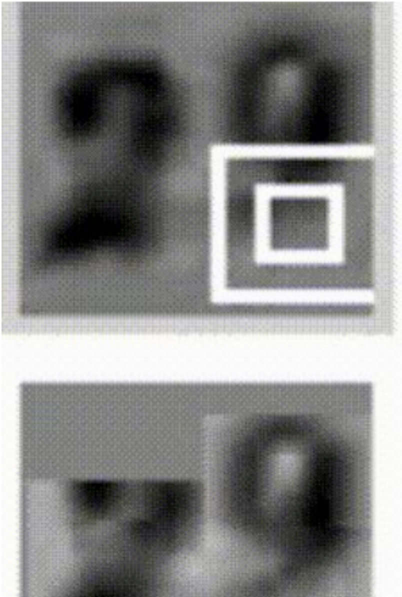


- one to one: image classification
- one to many: image captioning
- many to one: sentiment analysis
- many to many: machine translation
- many to many: video understanding

[From A. Karpathy Blog]

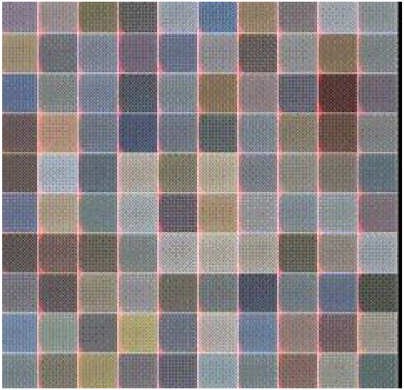
Sequential processing of fixed inputs

- Multiple object recognition with visual attention, Ba et al.



Sequential processing of fixed outputs

- DRAW: a recurrent neural network for image generation, Gregor et al.



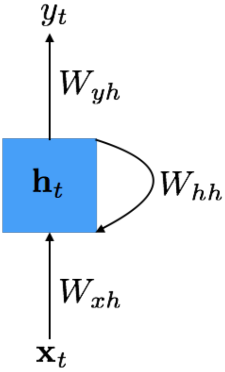
Recurrent Neural Network

- $\mathbf{h}_t = \phi_1 \left(\mathbf{h}_{t-1}, \mathbf{x}_t \right)$
- $\hat{y}_t = \phi_2 \left(\mathbf{h}_t \right)$

Where

- $\mathbf{x}_t$ = input at time t
- $\hat{y}_t$ = prediction at time t
- $\mathbf{h}_t$ = new state
- $\mathbf{h}_{t-1}$ = previous state
- ϕ_1 and ϕ_2 = functions with parameters Ws that we want to train

Subscript t indicates sequence index.

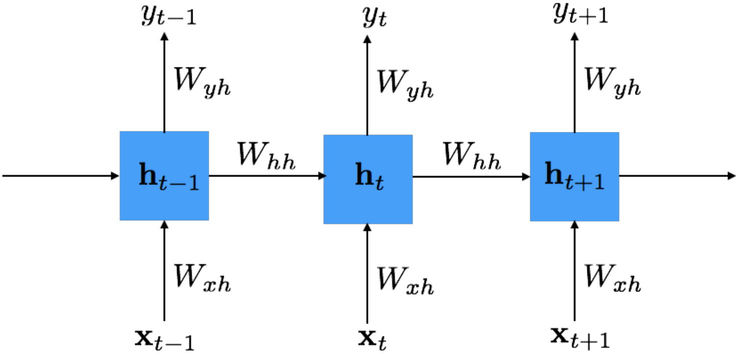


Example

$$\mathbf{h}_t = Tanh \left(W_{hh} \mathbf{h}_{t-1} + W_{xh} \mathbf{x}_t \right)$$
$$\hat{y}_t = softmax \left(W_{hy} \mathbf{h}_t \right)$$

Recurrent Neural Networks - Unrolling in Time

- Parameters W_{xh}, W_{hh} and W_{hy} are tied over time
- Cost: $E = \sum_t E_t$, where E_t depends upon y_t
- Training: minimize E to estimate W_{xh}, W_{hh} and W_{hy}



Recurrent Neural Network: Loss

When dealing with output sequences, we can define loss to be a function of the predicted output $\hat{y}_t$ and the expected value y_t over a range of times t

$$E(y, \hat{y}) = \sum_t E_t(y, \hat{y})$$

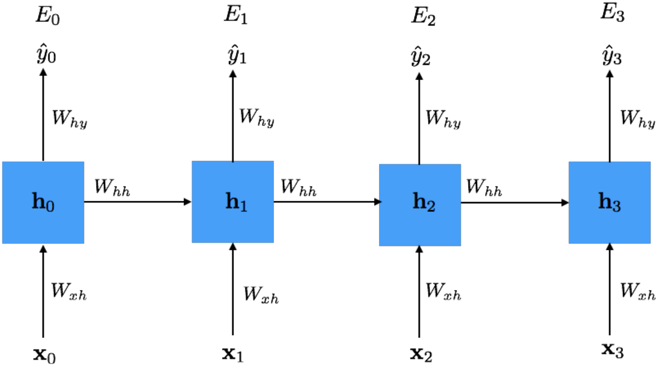
Example: using cross-entropy for k-class classification problem

$$E(y, \hat{y}) = -\sum_t y_t \log \hat{y}_t$$

Recurrent Neural Networks: Computing Gradients

We need to compute $\frac{\partial E}{\partial W_{xh}}$, $\frac{\partial E}{\partial W_{hh}}$ and $\frac{\partial E}{\partial W_{hy}}$ in order to train an RNN

Example



$$\frac{\partial E_3}{\partial W_{hh}} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial W_{hh}}$$

Recurrent Neural Networks: Vanishing and Exploding Gradients

We can compute the highlighted term in the following expression using *chain-rule*

$$\frac{\partial E_3}{\partial W_{hh}} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial W_{hh}}$$

Applying the chain-rule

$$\frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_k} = \prod_{j=k+1}^{3} \frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}$$

Or more generally

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{j=k+1}^{t} \frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}$$

Recurrent Neural Networks: Difficulties in Training

$$\frac{\partial E_t}{\partial W_{hh}} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial \mathbf{h}_t} \left(\prod_{j=k+1}^{t} \frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}} \right) \frac{\partial \mathbf{h}_k}{\partial W_{hh}}$$

$\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$ is a Jacobian matrix.

For longer sequences

- if $\left| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right| < 0$, the gradients vanish
  - Gradient contributions from "far away" steps become zero, and the state at those steps doesn't contribute to what you are learning.
  - Long short-term memory units are designed to address this issue
- if $\left| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right| > 0$, the gradients vanish
  - Clip gradients at a predefined threshold
- See also, On the difficulty of training recurrent neural networks, Pascanu et al.

Image Captioning

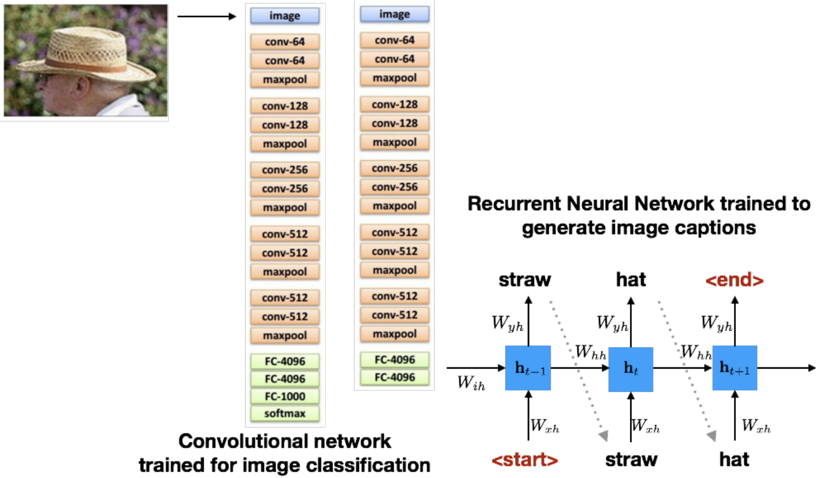


Image Captioning

For the image captioning example shown in the previous slide, $\mathbf{h}_t$ is defined as follows:

$$\mathbf{h}_t = Tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x} + W_{ih}\mathbf{v})$$
$$\hat{y}_t = softmax(W_{hy}\mathbf{h}_t)$$

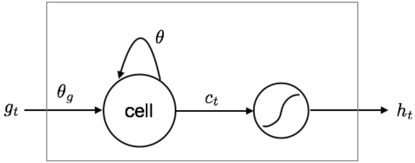
Image Captioning

- Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
- Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy - and Fei-Fei
- Show and Tell: A Neural Image Caption Generator, Vinyals et al.
- Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
- Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

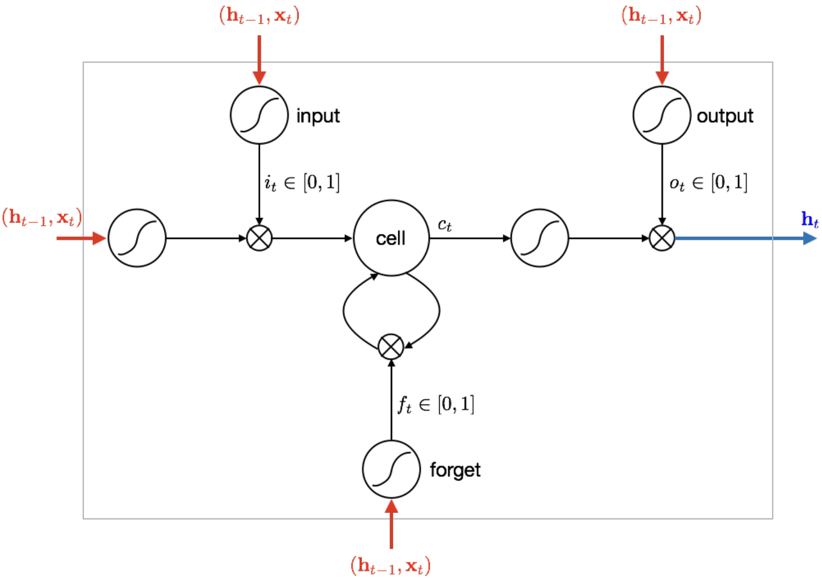
Dealing with Vanishing Gradients

Change of notation

$$c_t = \theta c_{t-1} + \theta_g g_t$$
$$h_t = tanh(c_t)$$



Long Short Term Memory (LSTM)



Long Short Term Memory (LSTM)

- Input gate: scales input to cell (*write operation*)
- Output gate: scales input from cell (*read operation*)
- Forget gate: scales old cell values (*forget operation*)

$$
\begin{aligned}
\mathbf{i}_t &= sigm(\theta_{xi}\mathbf{x}_t + \theta_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= sigm(\theta_{xf}\mathbf{x}_t + \theta_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= sigm(\theta_{xo}\mathbf{x}_t + \theta_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{g}_t &= tanh(\theta_{xg}\mathbf{x}_t + \theta_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \circ tanh(\mathbf{c}_t)
\end{aligned}
$$

$\circ$ represent element-wise multiplication

RNN vs. LSTM

Check out the video at https://imgur.com/gallery/vaNahKE

Summary

- RNN
  - Allow a lot of flexibility in architecture design
  - Very difficult to train in practice due to vanishing and exploding gradients
  - Control gradient explosion via clipping
  - Control vanishing gradients via LSTMs
- LSTM
  - Very power architecture for dealing with sequences (input/output)
  - Works rather well in practice