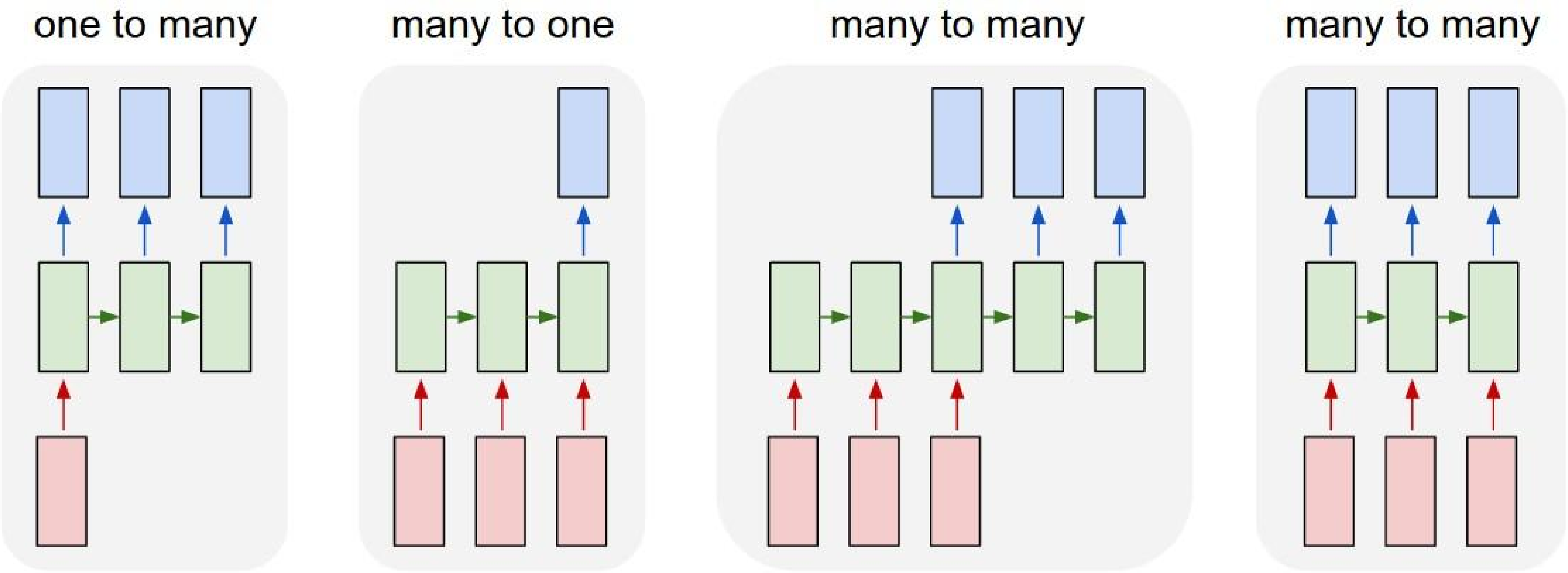


Recurrent Neural Networks

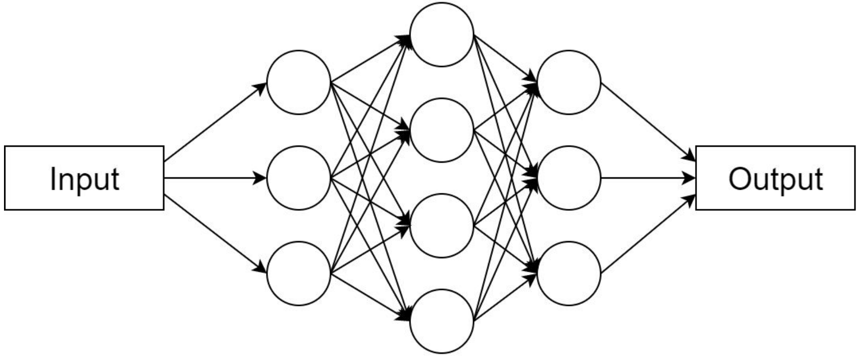
Deep Learning - Recitation (2/16/2018)

Table of Contents

What is an RNN

An RNN is a type of artificial neural network in where the weights form a directed cycle

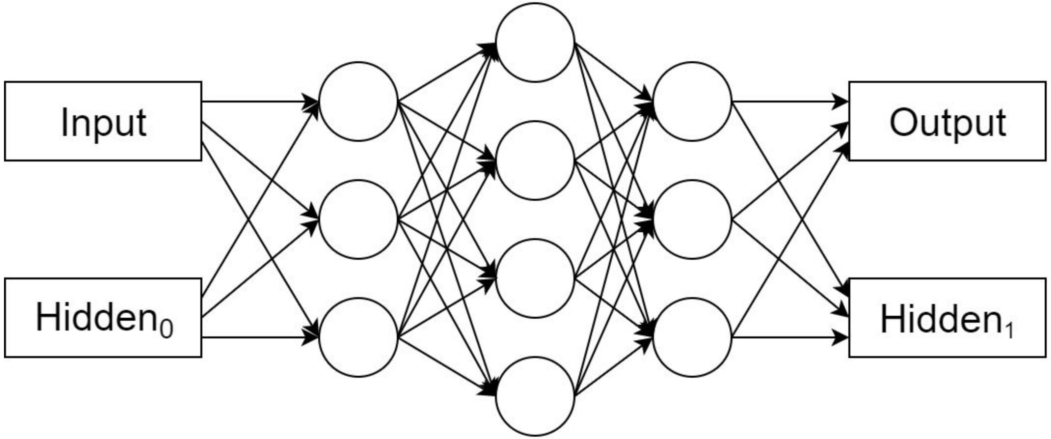
Let’s take a step back to a typical feedforward NN to explain what this means…



What is an RNN

An RNN is a type of artificial neural network in where the weights form a directed cycle

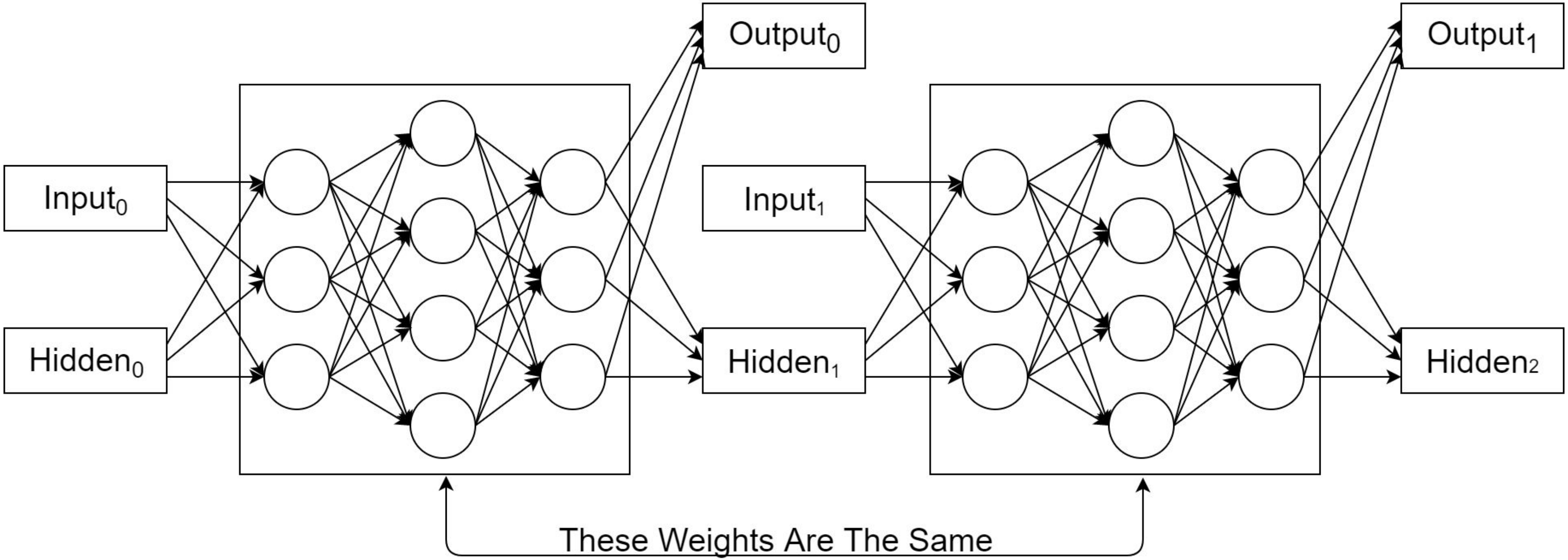
Let’s say this is 0



And this is 0

What is an RNN

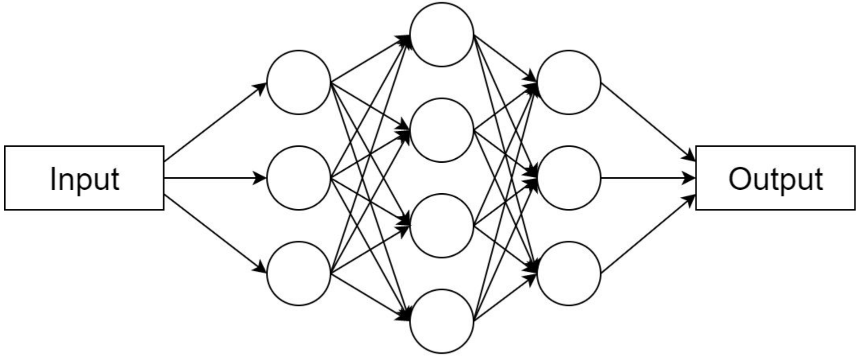
An RNN is a type of artificial neural network in where the weights form a directed cycle



Backpropagation Through Time

Note: This is handwavy

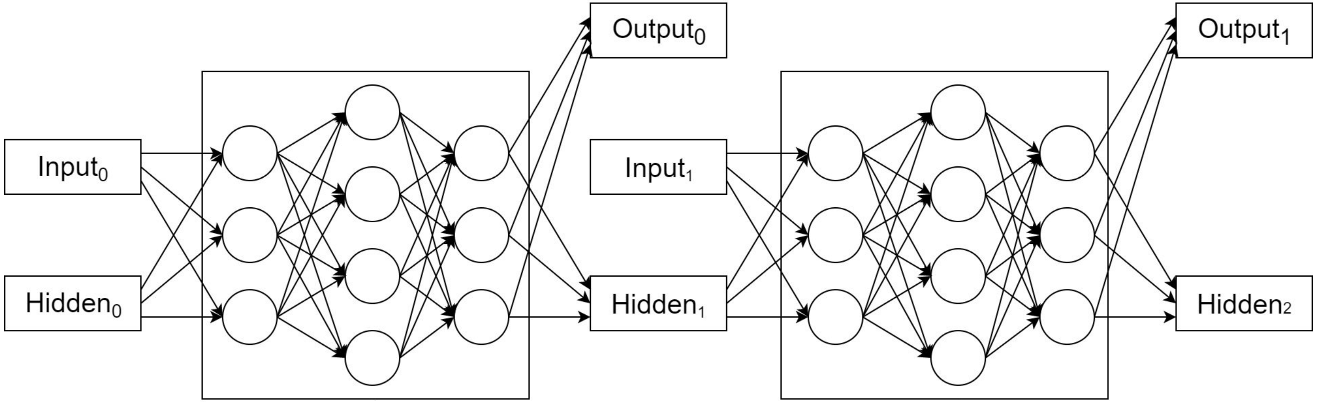
We can already back propagate
error through this...



Backpropagation Through Time

Note: This is handwavy

What does this look like?

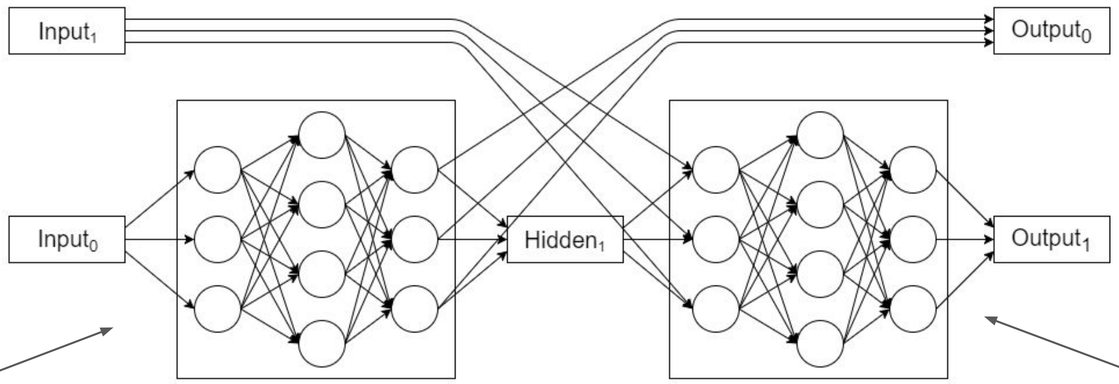


Backpropagation Through Time

Note: This is handwavy

What if we rearrange some things?

This just looks like a feedforward network with some strange connections...



Note that we dropped Hidden$_0$

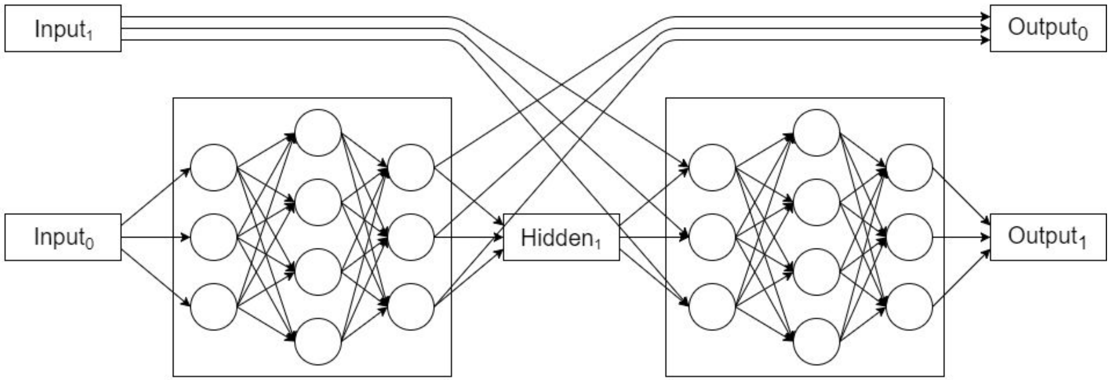
Note that we dropped Hidden$_2$

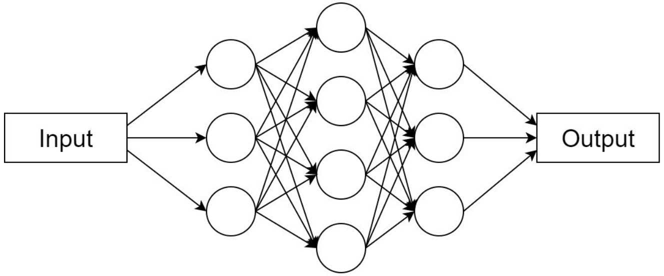
Backpropagation Through Time

Note: This is handwavy

These two can be trained in exactly the same way!

Regular Backprop!

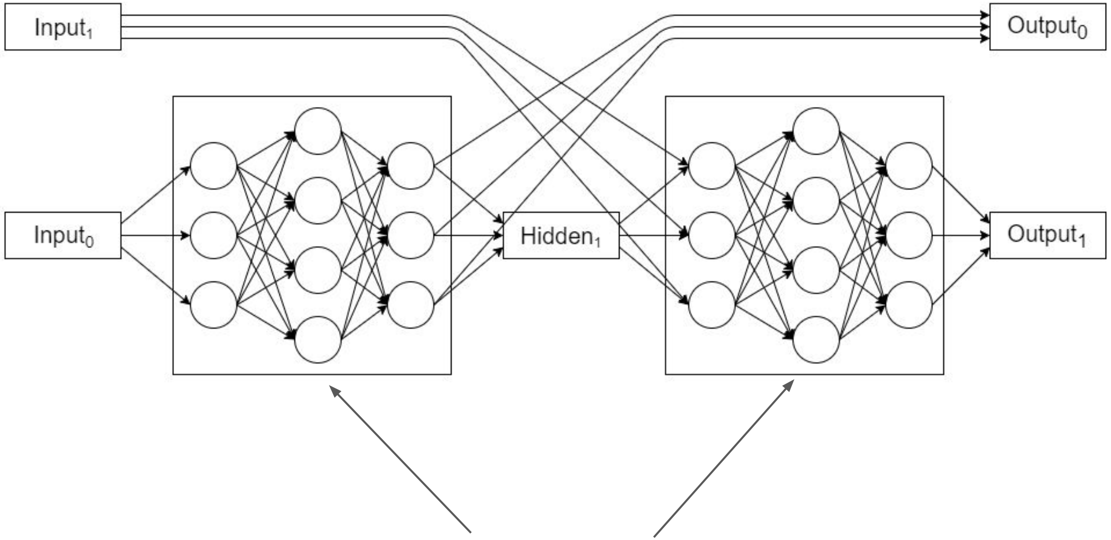




Backpropagation Through Time

Note: This is handwavy

“Unroll” your network

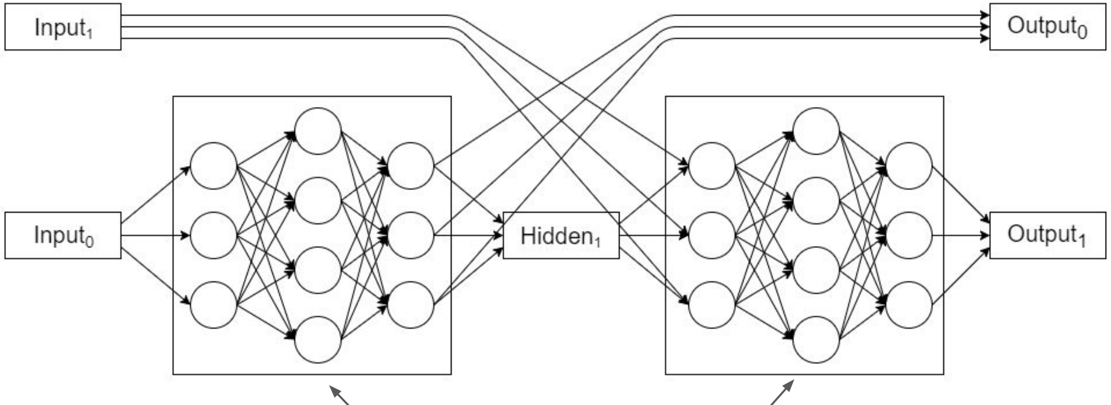


Calculate the gradients
for all of these weights

Backpropagation Through Time

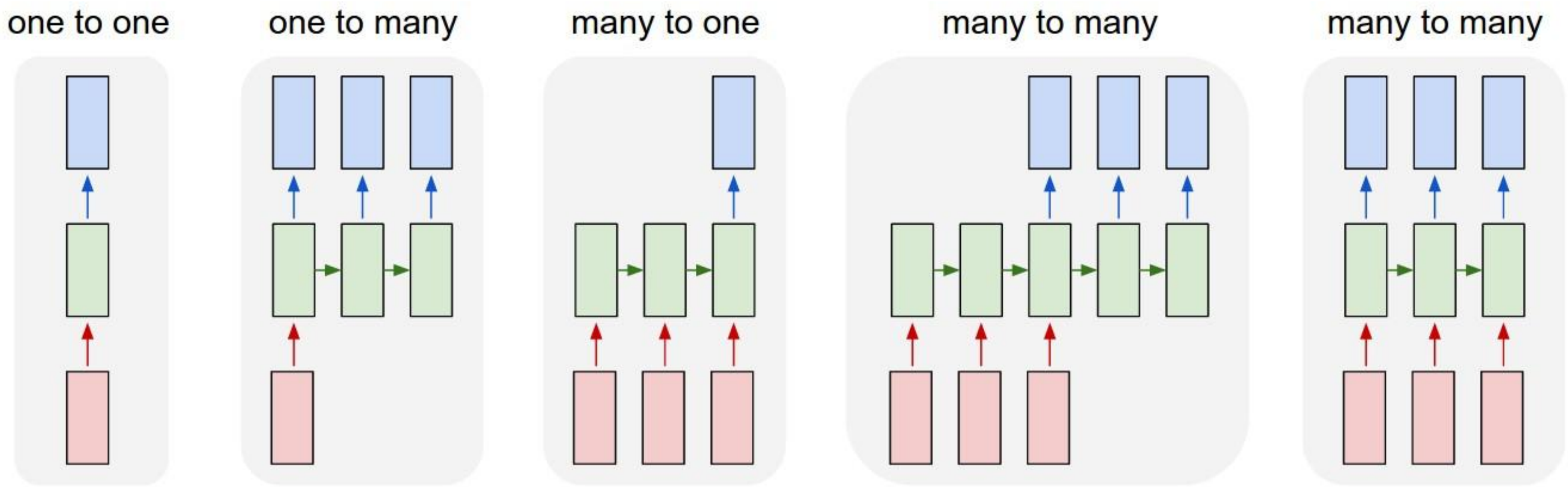
Note: This is handwavy

"Unroll" your network



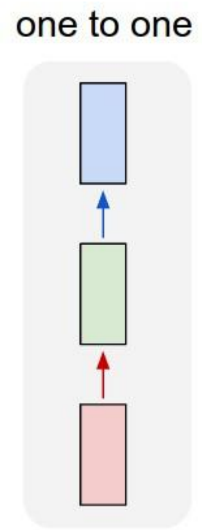
Because all these weights are tied update them at the same time… Just like tied weights in a CNN

RNN Paradigms



Different problems are more suited for different RNN paradigms

RNN Paradigms

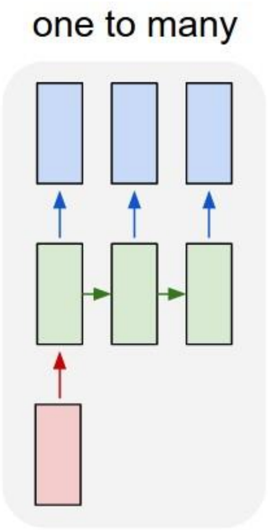


Given a single input predict a single output

This is just a simple feedforward neural network

Different problems are more suited for different RNN paradigms

RNN Paradigms



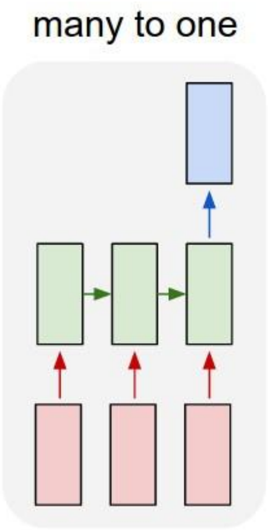
Given a single input predict a sequence of outputs

Ex. Image Captioning

Given an image describe the image textually

Different problems are more suited for different RNN paradigms

RNN Paradigms

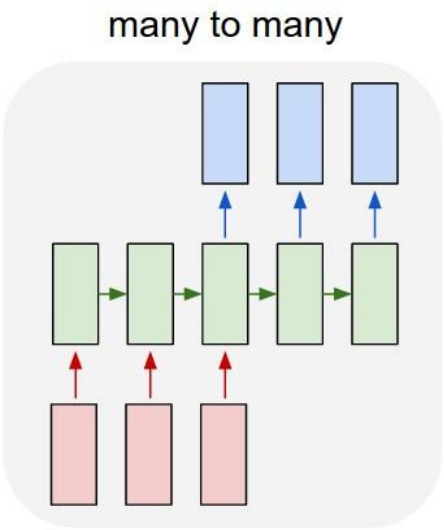


Given a single input predict a sequence of outputs

Ex. Sentiment Analysis
Given text predict positive or negative sentiment

Different problems are more suited for different RNN paradigms

RNN Paradigms



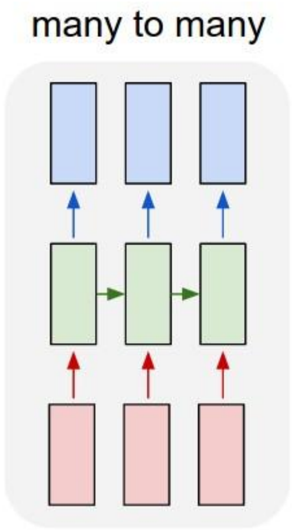
Given a sequence of inputs predict a sequence of outputs (of potentially different length)

Ex. Machine Translation

Given text in language A, translate it to language B

Different problems are more suited for different RNN paradigms

RNN Paradigms



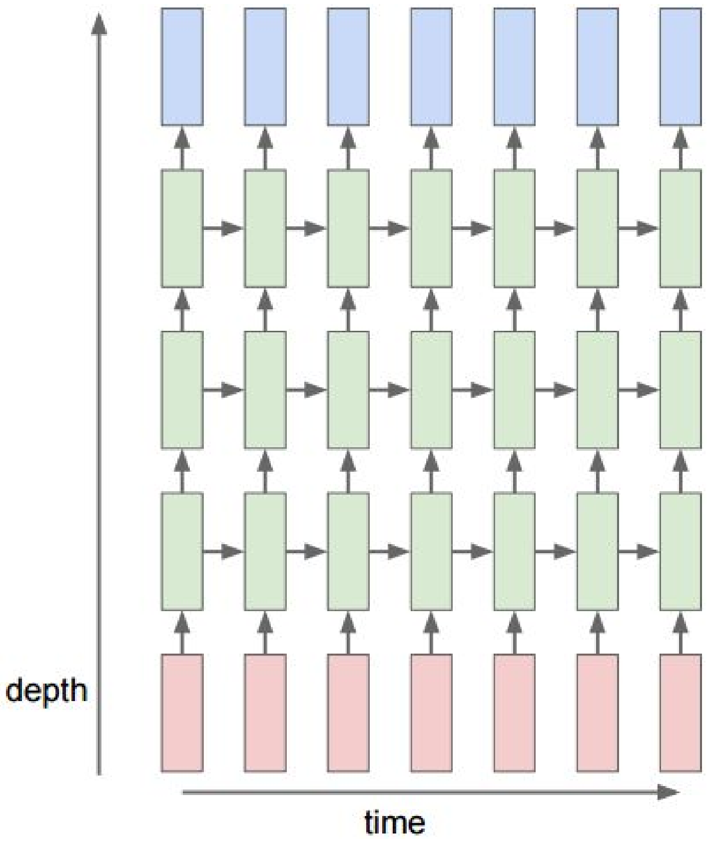
Given a sequence of inputs predict a sequence of outputs (of the same length)

Ex. Part of Speech Tagging

Given a sequence of words, label each word with its part of speech (Noun, Verb, etc)

Different problems are more suited for different RNN paradigms

How Layering RNNs Work



You just stack them like this… Nothing special

You can change the order or the direction

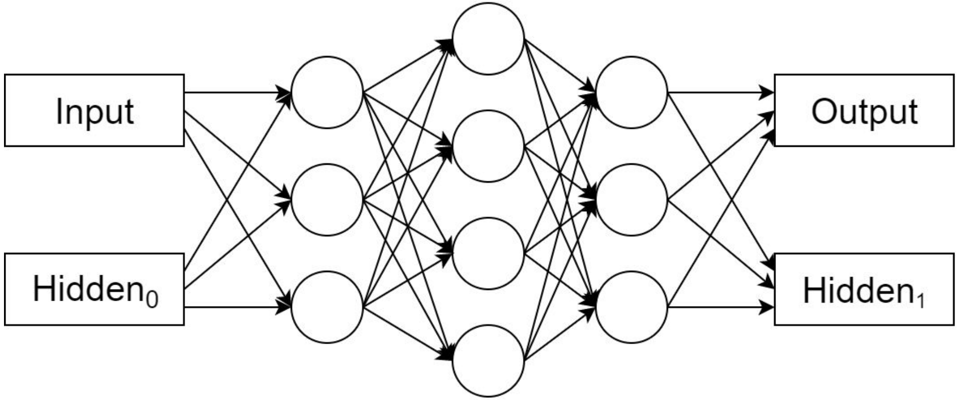
You can change the granularity as well (Hierarchical Networks)

The world is your oyster

Common Types of RNN Cells

Note, depending on the implementation “Output” and “Hidden_1” may be the same thing

Dense Cell



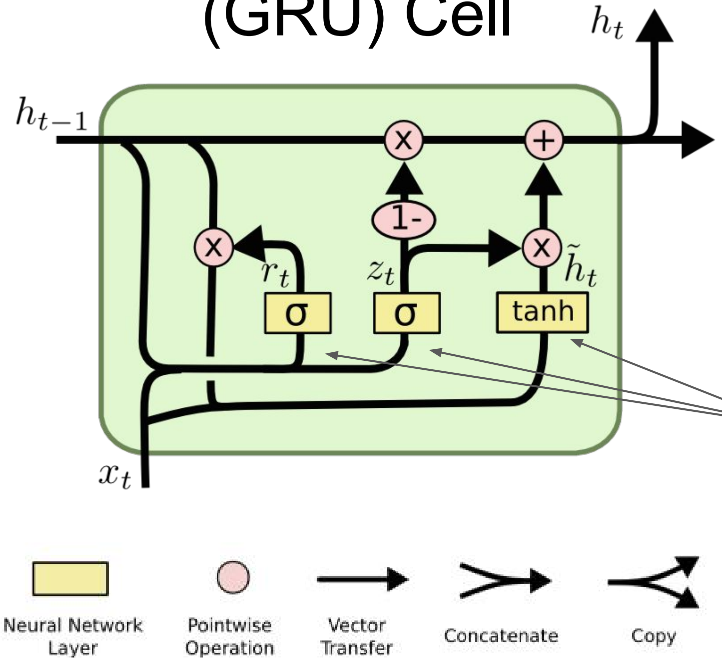
You aren’t limited to the above 3 layer structure, any feedforward style neural network architecture could work

To calculate the output, simply perform the traditional feedforward network calculation

Different cells are better for different problems

Common Types of RNN Cells

Gated Recurrent Unit (GRU) Cell



Where are the gates???

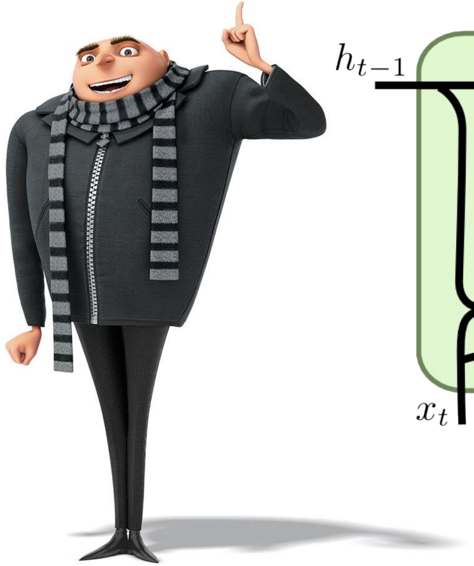
The gates are the neural nets

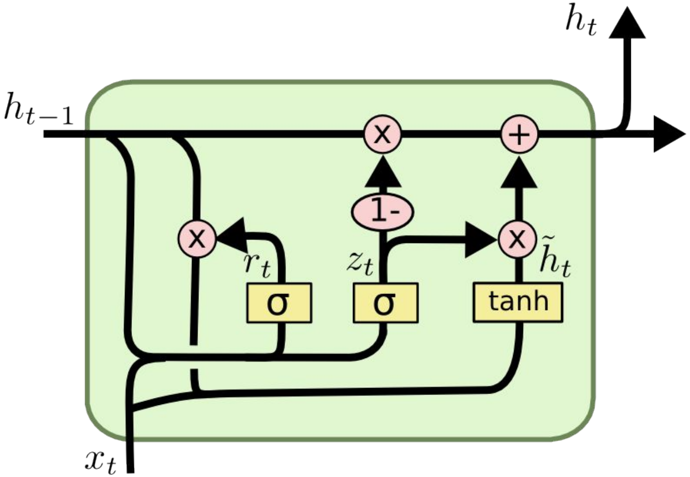
Different cells are better for different problems

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Common Types of RNN Cells

Gated Recurrent Unit (GRU) Cell Mathematics





$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

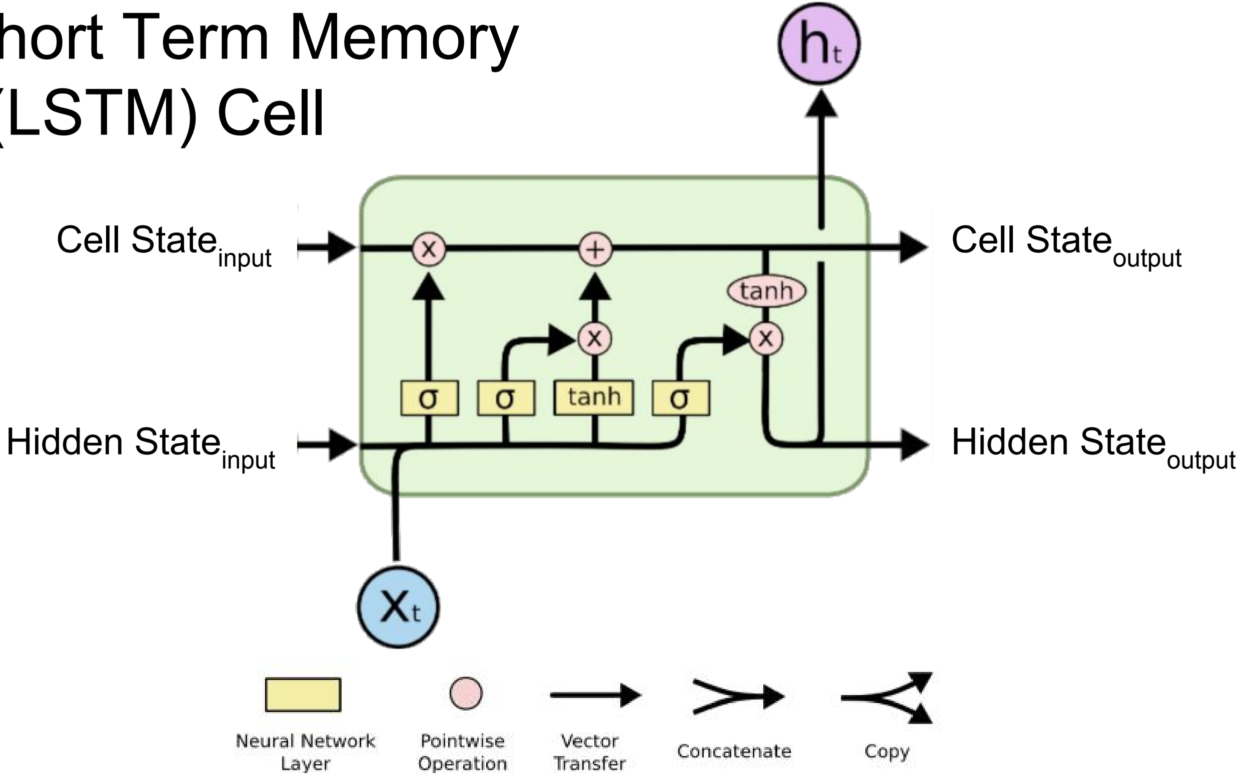
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Different cells are better for different problems

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Common Types of RNN Cells

Long Short Term Memory (LSTM) Cell

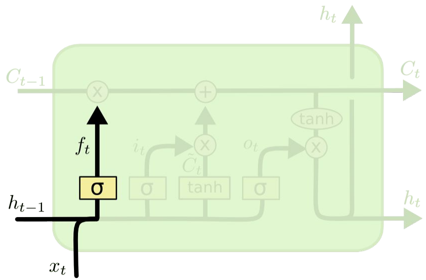


Different cells are better for different problems

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

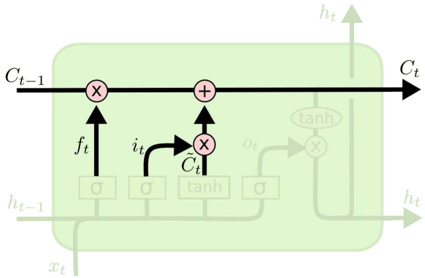
Common Types of RNN Cells

Long Short Term Memory (LSTM) Cell Mathematics



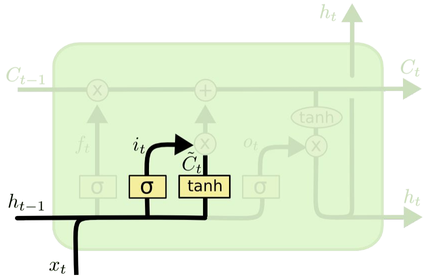
Forget Gate

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$



Cell State Output

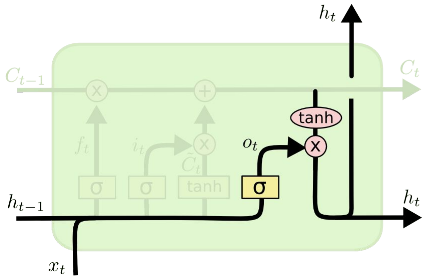
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



Input Gate

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



Output Gate

$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Different cells are better for different problems

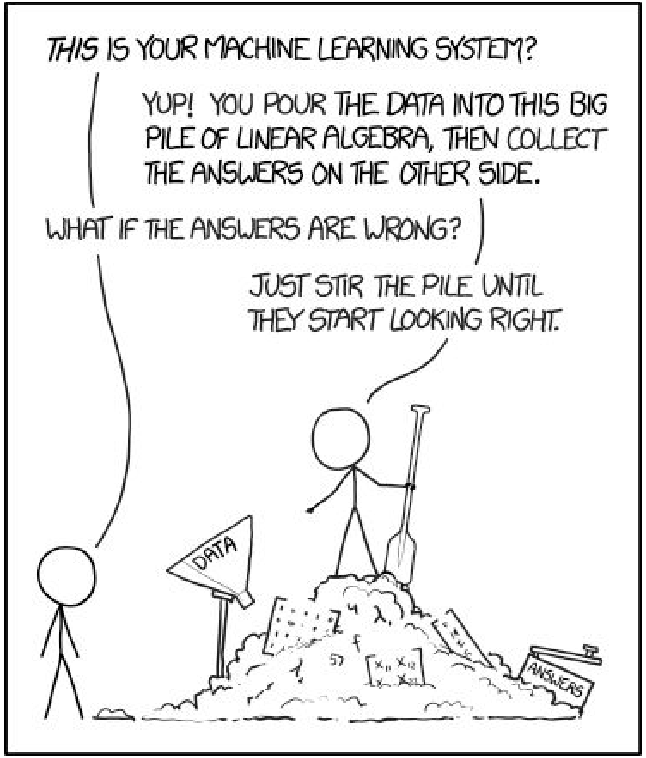
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Common Pitfalls of RNNs

- These models can overfit incredibly easily.
  - Start with an incredibly simple model, with small gates and few layers, then expand.

- Vanishing/Exploding Gradients
  - Depending on your data, BTT can cause your gradients to become incredibly small (vanish) or become incredibly large (explode)
    - Gated cells can mitigate this to an extent, but not entirely.
    - Be sure to regularize and keep an eye on your gradients to see how they are doing

Conclusion



Any Questions?