Recurrent Neural Networks

Data Science Immersive

FLATIRON SCHOOL

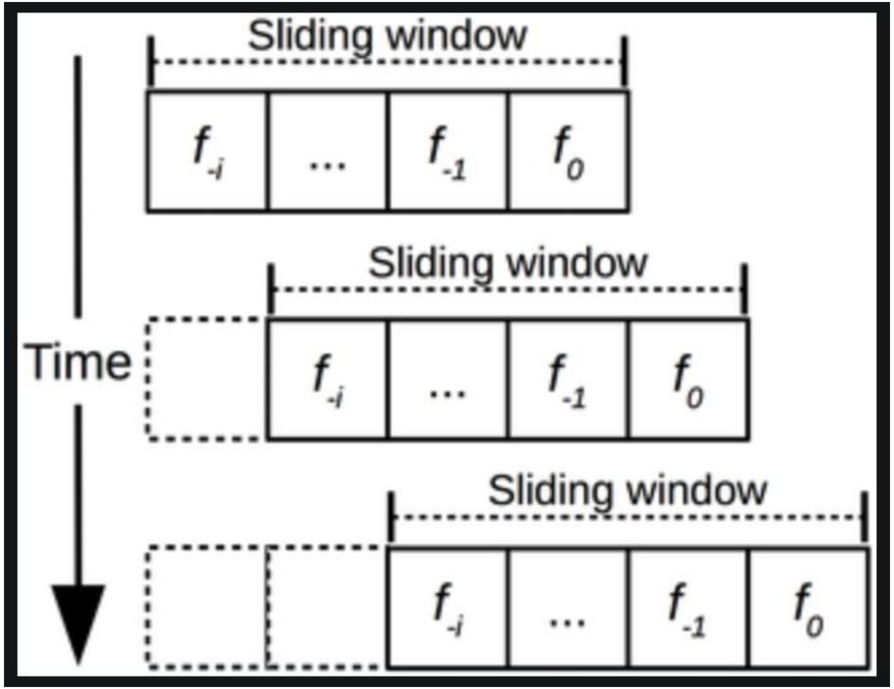
Sequence Learning

- Not just a collection of items, but an order too
  - A,B,C produces a different outcome from A,C,B

- Several considerations
  - How can each element be represented, either numerically or as a vector?
  - Sequence length - is it variable or constant?

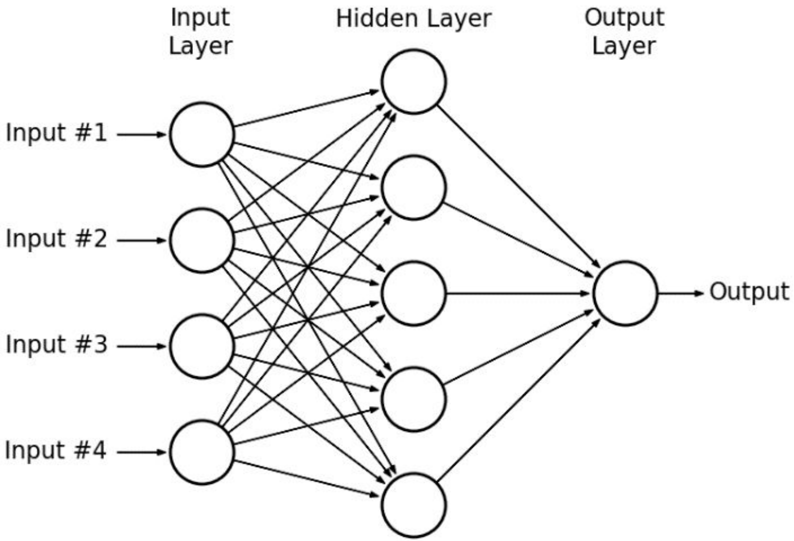
Sequence Learning: Examples

- Frequent Pattern Mining
- ARIMA (and variants)
- Markov Models
- Sliding Window Models



Why isn't MLP enough?

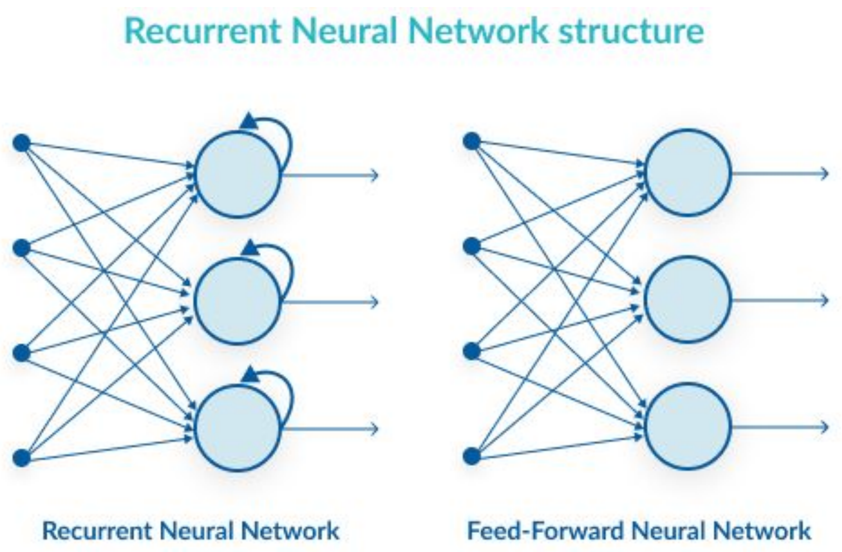
- Let's take a simple example - univariate time series forecasting. Why is a multilayer perceptron model not enough?



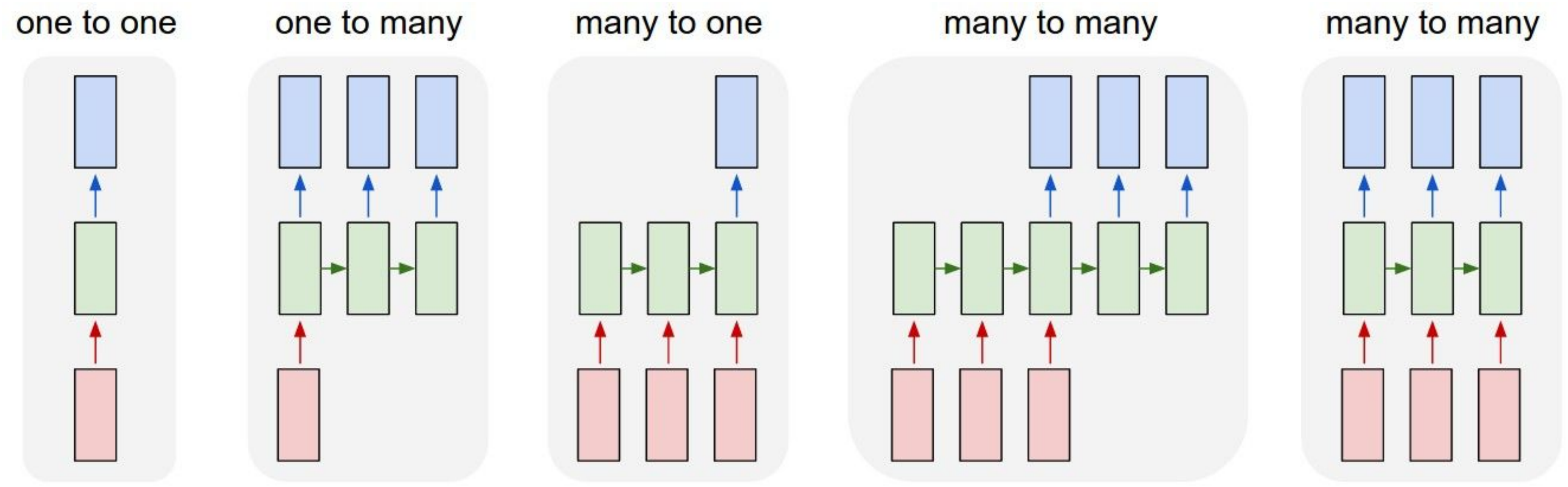
Recurrent Neural Networks

- Regular “forward feed” neural networks have an activation function that’s fed all the inputs, and then goes to the output.

- RNNs use a recurrent activation that’s also forward-propagated, and is fed from the output of the previous step in the sequence.


Recurrent Neural Network structure
Recurrent Neural Network
Feed-Forward Neural Network

RNNs - The Whole Sequence

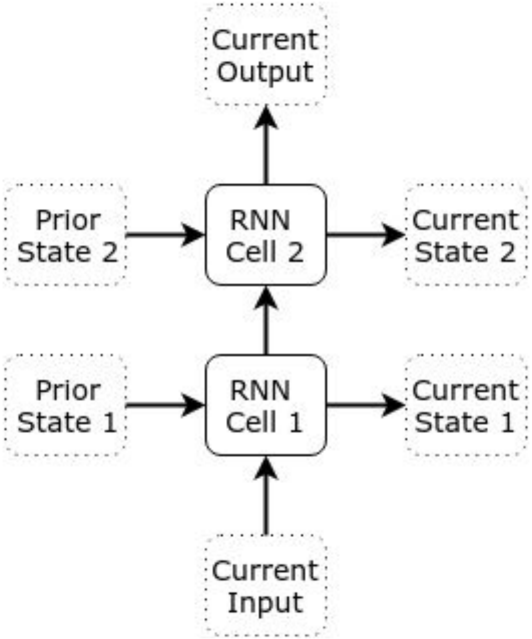


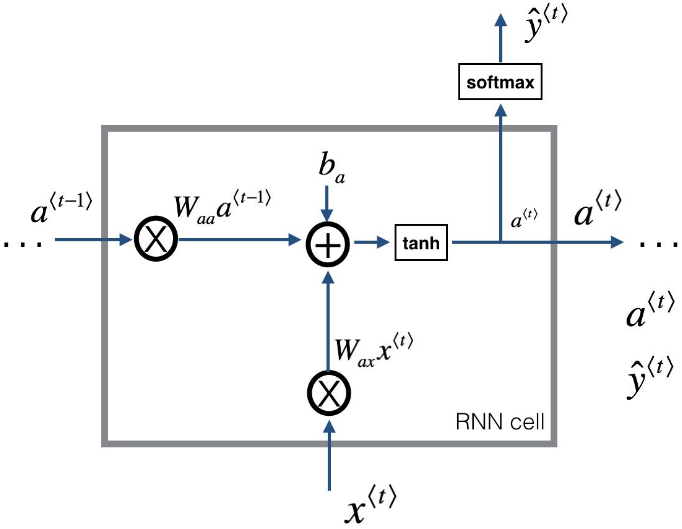
RNN Applications - Examples

- NLP
  - Predictive text
  - Sentiment analysis

- Recommendation systems
  - Collaborative filtering - (forget matrix factorization!)
  - Content filtering - this isn't a very common use of RNNs
- Multivariate time series forecasting
- Audio interpretation
  - Spectral analysis -> RNN -> Output
- Rudimentary image recognition
  - (CNNs are better)
- Recurrent convolutional neural networks

Forward Propagation In-Depth



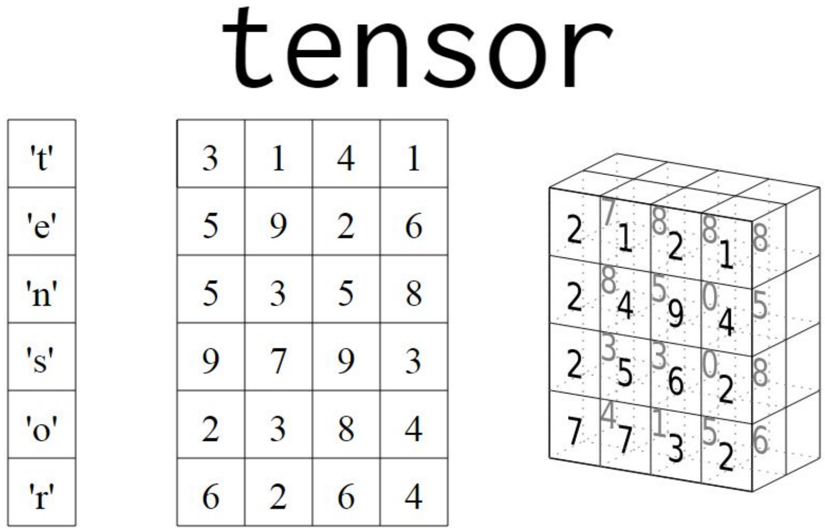


$$a^{\langle t\rangle} = \tanh(W_{ax}x^{\langle t\rangle} + W_{aa}a^{\langle t-1\rangle} + b_a)$$

$$\hat{y}^{\langle t\rangle} = \mathit{soft}\max(W_{ya}a^{\langle t\rangle} + b_y)$$

How To Do It

- Before we get into coding anything, let’s think about how this is gonna work input-wise
- 3 - dimensional tensor
  - How many sequences?
  - How many elements in each sequence?
  - What is each element?
- Whereas normally:
  - How many rows of variables?
  - How many variables in each row?



't'
'e'
'n'
's'
'o'
'r'

tensor of dimensions [6]
(vector of dimension 6)

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

tensor of dimensions [6,4]
(matrix 6 by 4)

tensor of dimensions [4,4,2]

Code

- Keras
- Either use .npy file/object or a multiindex pandas dataframe (we’re in 3 dimensions now)
- Keras masking layer
- Keras SimpleRNN

LSTM

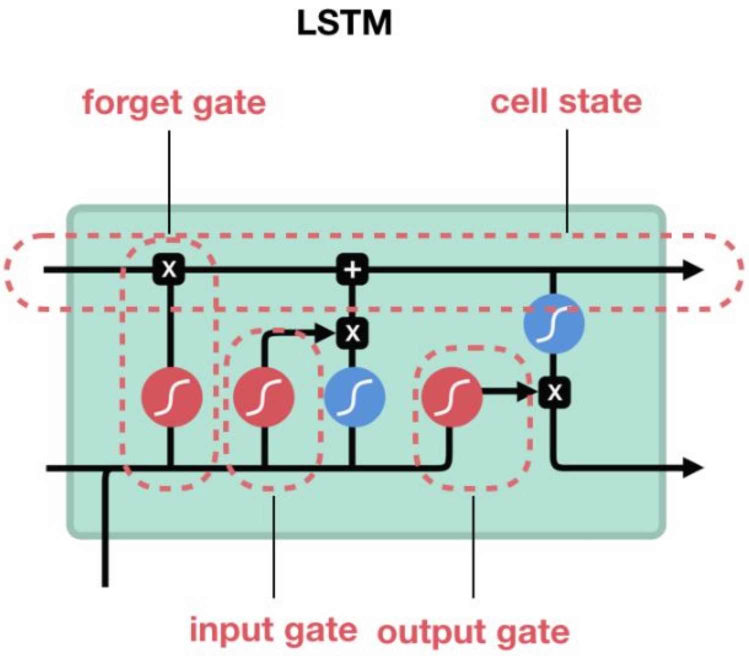
- Diminishing gradient problem
- Short term-memory: the memory across a sequence
- Predictive text example:

There was a castle in the mountains on a river across from the valley where the king's army was marching.

- King's -- Army
- Army -- Marching
- Castle -- King should be an obvious connection but is too separated by other words - the model has forgotten.

Logic Gates

- Forget gate
  - What information to keep or throw away
- Input gate
  - Updates cell state but not the output
- Output gate
  - What goes to cell state versus output
- Vectors are going into all these gates and have weights and biases associated with each, which are learned in backpropagation



Further Exploration

- Gated Recurrent Units
- Bidirectional RNNs

Resources

- Andrew Ng’s coursera course
- Live Coding RNNs: https://youtu.be/BSpXCRTOLJA
- Funny Russian guys: https://youtu.be/lycKqccytfU
- Illustrated guide to LSTM: https://youtu.be/8HyCNIVRbSU
- Famous RNN blog post: http://karpathy.github.io/2015/05/21/rnn-effectiveness/