DIGITAL TWIN AI and Machine Learning: Deep Learning II: Recurrent Neural Networks

Prof. Andrew D. Bagdanov
andrew.bagdanov AT unifi.it

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Firenze

26 November 2020

Outline

Introduction

Overview

- Sometimes data doesn't cooperate with us.
- It's not always fixed-length (e.g. audio recordings) or continuous (e.g. text data).
- Often the problem isn't easily decomposable by slicing the data into fixed-length chunks for processing.
- Think about machine translation: to translate a single sentence, you often need the entire original sentence as context.
- We will now see some techniques for working with data of this type.
- Note: this lecture is very high-level because often working with variable-length data requires more preprocessing than we have time for.

Motivations

Motivation: not everything is fixed-length

- Not all problems can be converted into one with fixed-length inputs and outputs.
- Problems such as Speech Recognition or Time-series Prediction require a system to store and use context information.
- Simple case: Output 1 if the number of 1s is even, else 0:
  - 1000010101 –> 1
  - 100011 –> 0
  - Impossible to choose a fixed context window – *there can always be a new sample longer than anything seen*.

Motivation: how to encode discrete phenomena

- Let's say we want to feed a sentence to a network:
  the quick brown fox jumped over the lazy dog
- How on earth should we encode this?
- Words? Possible, but then we would need one-hot vectors of about 100,000 dimensions...
- Characters? Also possible, we only need about 26 dimensions, but they are discrete values.
- And in any case, still variable-length.

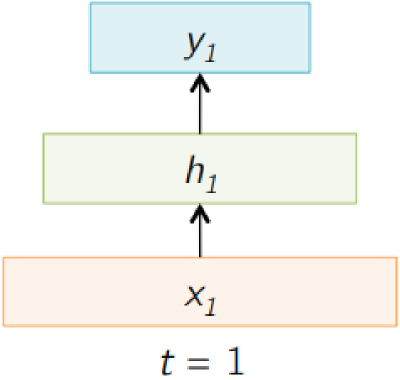
Motivation: Recurrent Models

- Recurrent Neural Networks (RNNs) take the previous output (or hidden states) as extra inputs.
- The composite input at time T has some historical information about at times t < T.
- RNNs are useful as their intermediate values (state) can store information about past inputs for a time that is not fixed *a priori*.
- In this lecture we will see a high-level overview of RNNs, and in particular of the Long Short-Term Memory (LSTM) model.

Basic Models

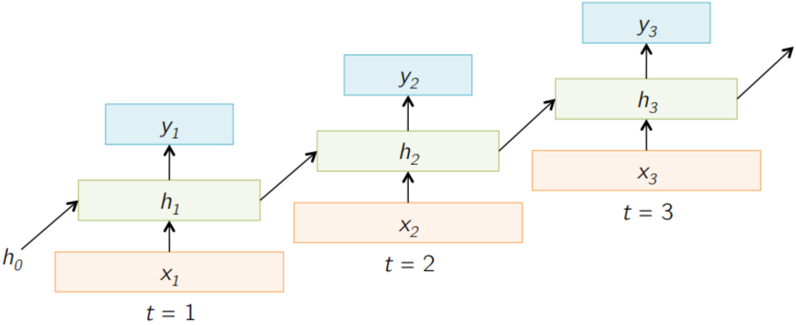
A basic (non-recurrent) network

- Start with a basic network:



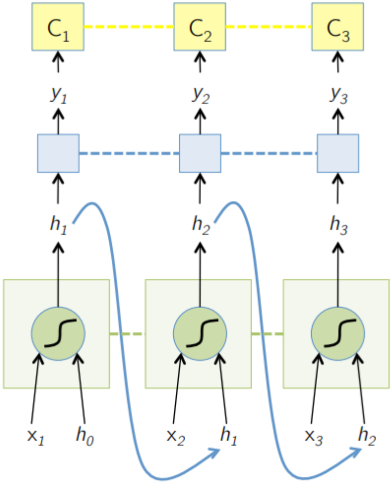
A sequence of networks

- Well, we can chain networks together:



A recurrent network

- Now share weights:



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

------ indicates shared weights

RNNs: The Main Point

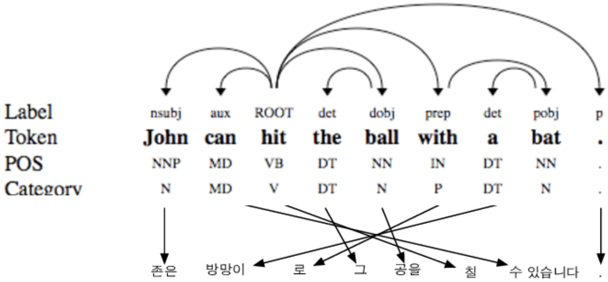
- Each step is just a copy of the same network – since weights are shared.
- At each step, the network produces an output and passes its hidden state on to the next step.

RNNs: training

- The main method used to train RNNs is known as Backpropagation Through Time (BPTT).
- The network is unfolded for the forward pass and treated as one big feed-forward network.
- This unfolded network takes the whole time series as input.
- Weight updates are computed for each copy in the unfolded network, then summed (or averaged) and and applied to the RNN weights

RNNs: a Big Problem

- Vanilla RNNs have serious problems capturing long distance dependencies between inputs.
- Take language translation as an example.



- Again: vanishing gradients are the real culprit.

Long Short-Term Memory (LSTM)

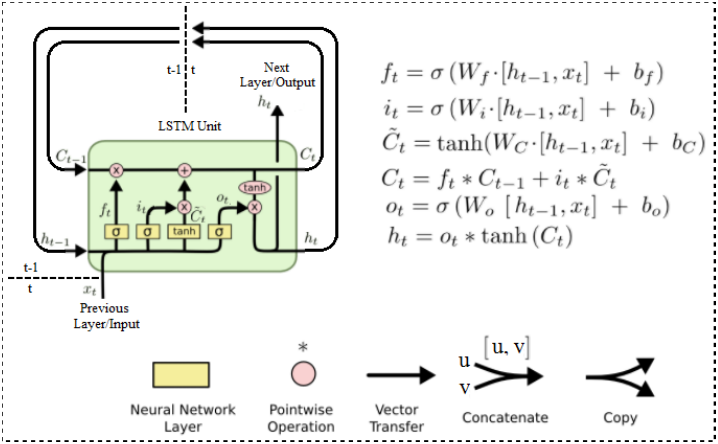
LSTMs: Overview

- The LSTM uses the idea of Constant Error Flow for RNNs to ensure gradients don't decay.
- The key component is a memory cell that acts like an accumulator over time.
- Instead of computing new state as a matrix product with the old state, it rather computes the difference between them.
- Expressiveness is the same, but gradients are better behaved.

Long Short-Term Memory, Hochreiter et al., 1997

LSTMs: The Diagram

- It's simpler than it looks...



Long Short-Term Memory, Hochreiter et al., 1997

LSTMs in Keras

LSTMs: the layer

- The simplest way to create an LSTM network us to just use the `tf.keras.layers.LSTM` layer type:

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import LSTM, Dense

model = Sequential()
model.add(LSTM(128, input_shape=(maxlen, len(characters))))
model.add(Dense(len(characters), activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

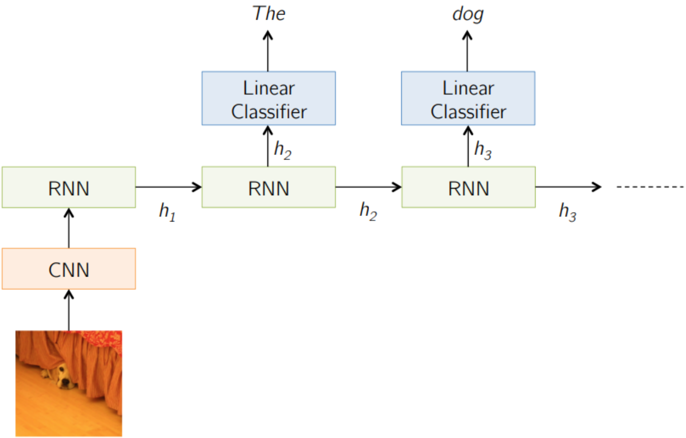
Reflections

LSTMs, LSTMs everywhere

- LSTMs (and other recurrent models) are being used everywhere these days.
- They are used for speech recognition, machine translation, as attention models for image and video understanding, etc.
- They are simple and flexible models – although they can be intimidating due to their theoretical complexity.
- The easiest way to think about them is as unrolled networks with shared weight.

LSTMs: Image Captioning

- A tantalizing application is Image Captioning:



Show and Tell: A Neural Image Caption Generator, CVPR 2015.

LSTMs: Image Captioning

- When it works, the results are amazing:

LSTMs: Laboratory

- The laboratory notebook for this afternoon:

  `http://bit.ly/DTwin-ML7`