Recurrent Neural Networks (RNN) Notes

Jashaina Thomas & Yiping Lu

April 14, 2020

Recurrent Neural Networks

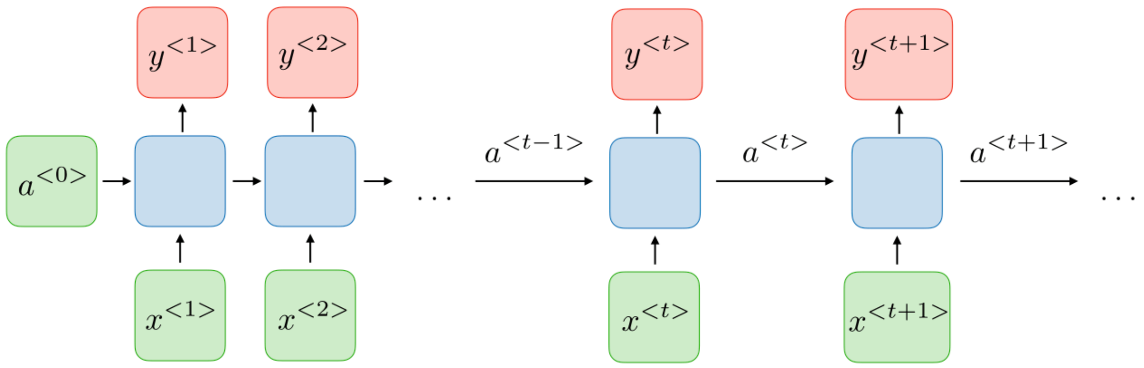
Recurrent neural networks, also known as RNNs, are a class of neural networks that allow previous outputs to be used as inputs while having hidden states. RNN models are mostly used in the fields of natural language processing and speech recognition. The pros and cons of a typical RNN architecture are summarized below:

Advantages
• Possibility of processing input of any length • Model size not increasing with size of input • Computation takes into account historical information • Weights are shared across time

Drawbacks
• Computation being slow • Difficulty of accessing information from a long time ago • Cannot consider any future input for the current state

0.1 Architecture Structure

Recurrent neural networks are typically structured as follows:



For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$
$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

0.2 Loss Function

In order to train our RNN, we first need a loss function.

Loss function — In the case of a recurrent neural network, the loss function L of all time steps is defined based on the loss at every time step as follows:

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L(\hat{y}^{<t>}, y^{<t>})$$

0.3 Backpropogation

In order to fully calculate the gradient of W_{aa}, we'll need to backpropagate.

Backpropagation is done at each point in time. At timestep T, the derivative of the loss L with respect to weight matrix W is expressed as follows:

$$\frac{\partial L^{(T)}}{\partial W} = \sum_{t=1}^{T} \left.\frac{\partial L^{(T)}}{\partial W}\right|_{(T)}$$

0.4 Activation Function

The most common activation functions used in RNN are listed below:

$$\text{Sigmoid: } g(z) = \frac{1}{1+e^{-z}}, \quad \text{Tanh: } g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}, \quad \text{RELU: } g(z) = max(0, z),$$

0.5 Word Representation

The two main ways of representing words are shown below:

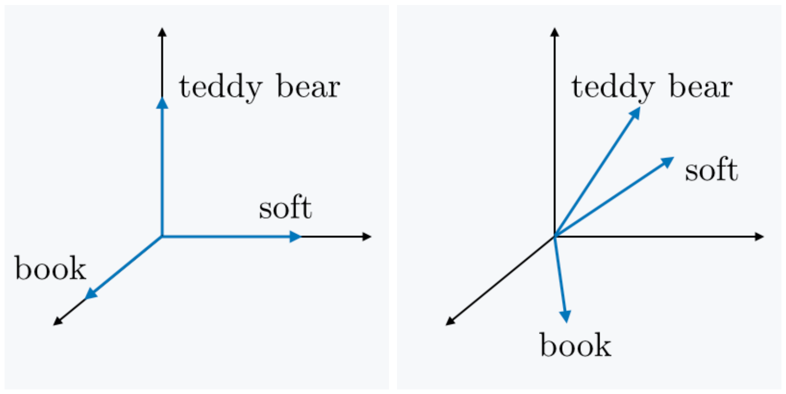


Figure 1: 1-hot representation (no similarity information) vs. Word embedding (words similarity)

Embedding matrix — For a given word w, the embedding matrix E is a matrix that maps its 1-hot representation o_w to its embedding e_w as follows:

$$e_w = E0_w$$

Where o_w is the 1-hot representation. Learning the embedding matrix E can be done using target/context likelihood models.

Word2vec — Word2vec is a framework aimed at learning word embeddings by estimating the likelihood that a given word is surrounded by other words. Popular models include skip-gram, negative sampling and CBOW.

RNN Summary

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. RNNs are a kind of neural network that specialize in processing sequences. They're often used in Natural Language Processing (NLP) tasks because of their effectiveness in handling text.

Recurrent Neural Network remembers the past and it's decisions are influenced by what it has learnt from the past. While RNNs learn similarly while training, in addition, they remember things learnt from prior input(s) while generating output(s). RNNs can take one or more input vectors and produce one or more output vectors and the output(s) are influenced not just by weights applied on inputs like a regular NN, but also by a "hidden" state vector representing the context based on prior input(s)/output(s). So, the same input could produce a different output depending on previous inputs in the series.

Some interesting models related to RNN include deep RNNs, bidirectional RNNS, Recursive Neural Networks, and Long Short Term Memory (LSTM).