Recurrent Neural Networks (RNN)

Introduction

Recurrent Neural Networks (or RNNs) are neural networks in which information can propagate in both directions, including from deep layers to early layers. In this way, they are closer to the true functioning of the nervous system, which is not one-way.

These networks have recurrent connections in the sense that they store information in memory: they can consider a certain number of past states at a given instant t. For this reason, SNNs are particularly suitable for applications involving context, and more specifically for processing temporal sequences such as learning and signal generation, i.e. when data form a sequence and are not independent of each other.

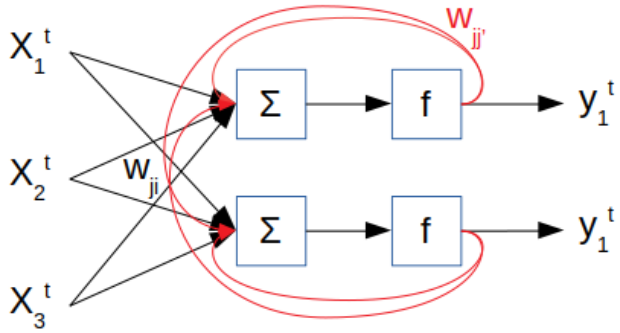
However, for applications involving long time gaps (typically the classification of video sequences), this "short-term memory" is not sufficient. Indeed, "classical" RNNs (single recurrent neural networks or Vanilla RNNs) are only able to memorize the so-called near past and start to "forget" after about 50 iterations. This two-way transfer of information makes their training much more complicated, and it is only recently that effective methods such as LSTM (Long Short-Term Memory) have been developed.

These networks with a large "short term memory" have revolutionised, among other things, speech recognition by machines (Speech Recognition) or text comprehension and generation (Natural Language Processing). From a theoretical point of view, NRNs have a much greater potential than traditional neural networks: research has shown that they are "Turing-complete", which means that they allow the simulation of any algorithm.

So if we want to translate the use of such a method in a real case, we can mention the handwritten text recognition : it will predict a letter by “remembering” the previous one (so we predict an event by looking at the probability in this exact context not in general).

Algorithmic translation

The following scheme shows in detail a recurrent layer, with the inputs noted xit and the outputs noted yjt. We can see here the wji representing the weights linking the ith input to the jth neuron of the recurrent layer, and the wjj’ representing the recurrence weights, linking the output of the j’th recurrent neuron to the input of the jth recurrent neuron.



We can then compare the output formula from a basic Neural Network to the Recurrent Neural Network with the following formulas:

Classic Neural Network:

$$y_j^t = f(\sum_i W_{ji} x_i^t)$$

Recurrent Neural Network:

$$y_j^t = f(\sum_i W_{ji} x_i^t + \sum_{j'} r_{ii'} y_i^{t-1})$$

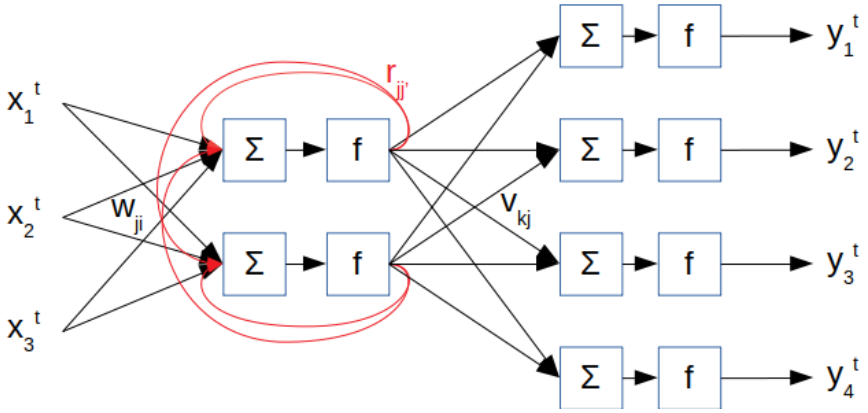
We can see that inside the function we also consider the sum of the previous states, hence the recurrence of the network.

Format

Most of the time the recurrent layers are combined with classical layers such as dense layers or convolution layers.

In the following scheme we can see the previously described scheme with the recurrent layer composed of two recurrent neurons followed by a layer of four dense neurons.

We can see here the wji representing the weights linking the ith input to the jth neuron of the recurrent layer, and the wjj' representing the recurrence weights, linking the output of the j'th recurrent neuron to the input of the jth recurrent neuron. The weights denoted by vkj link the jth recurrent neuron to the kth neuron of the output layer.

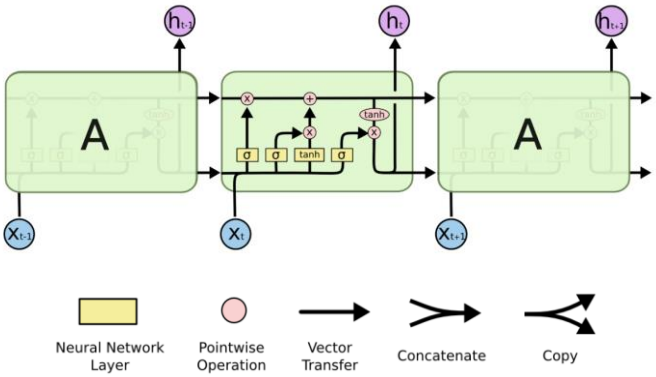


Variants

The Recurrent Neural Networks come in many variants. In fact, the one we introduced earlier is called "Vanilla RNN" and describes the basic idea used as a framework for many other variants.

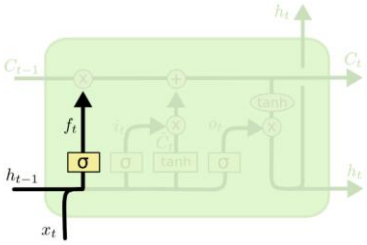
LSTM

One of the most known variants of RNN is the LSTM variant, which means Long Short Term Memory. This variant is particularly efficient to solve handwriting recognition or speech recognition problems.

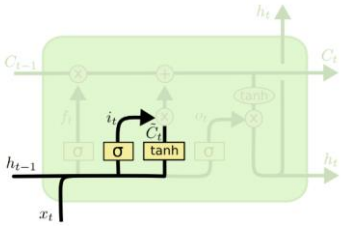


This is the representation of a LSTM network; we can see that it is much more complex than the classic RNN architecture.

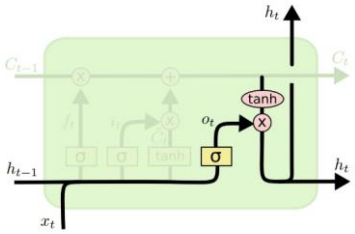
The top line crossing the iteration is called the cell state, the LSTM can remove or add information to the cell state, regulated by the three cell gates. The cell gates will modify the cell state with a sigmoid layer and a pointwise operation. The sigmoid layer will define whether the information can come through or not and if yes how much.



The first gate is called the “forget gate layer” it is mainly used to erase the memory of the network when we change the subject (ie. New sentence, the pronoun needs to be reset).



Then we need to define what to store in the cell state, the next layer need to define the new information we will add. We use another sigmoid layer, called the “input gate layer”, here if defines the value we will update. Then a tanh layer created a vector of candidates to add to the cell state, we then combine these two vectors to update the state.



The last step is to define what we will output from this iteration, for that, we filter the cell state that we built. we use a sigmoid layer to define which part of the cell state we keep and a tanh layer to push the values between -1 and 1. Then we finally multiply these two vectors, so we get the parts we decided to keep, we can send them to the next iteration.

We will now use a research paper to illustrate one of the use case of RNN, and more particularly LSTMs. Note that in the paper itself they didn't use LSTMs for the calculations but in the conclusion they precise that the LSTM method that they also trained performed better than the Multimodal RNN. We will use this paper to then explain how LSTMs outperforms on the same tasks than a classical RNN.

The research paper is "Deep Visual-Semantic Alignments for Generating Image Descriptions"

Motivation

Find a way to predict efficiently what happens in a picture, which means being able to make several predictions out of a single image. So, the goal is to be able to detect the different elements of the picture and then to define which one interacts with which. We then need to predict a sequence.


"man in black shirt is playing guitar."


"construction worker in orange safety vest is working on road."

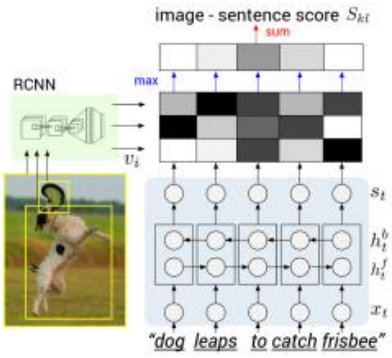

"two young girls are playing with lego toy."


"boy is doing backflip on wakeboard."

Therefore, RNN are very efficient and even more LSTM as they are deeper and more complex networks as we saw earlier in this presentation.

Method

Training phase



The method used in this paper is a novel combination of Convolutional Neural Networks on the images, and then a Recurrent Neural Network for the sentences.

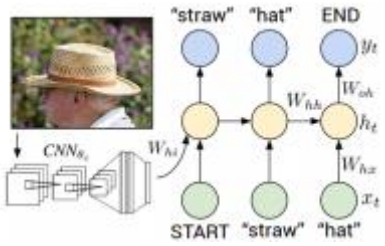
They firstly used a Region Convolutional Neural Network to learn the different components of an image and creates feature vectors to describe them.

On the other hand, they used the pictures annotations to embed the words in their context.

Then finally the pairwise similarities are computed with inner product and reduced to a sentence score. We combine the feature from the RCNN with the RNN to get a score.

Like that we can create a training set composed of bounding boxes and the word with the one the score was the highest.

Testing the Multimodal RNN (can be replaced by a LSTM)



In this phase we can train our RNN on the previous dataset. Now out model can train on this dataset, and further to make predictions thanks to what was learnt here. While in the paper itself they did not use the LSTM for their results shown in the graphs they affirmed that LSTMs consistently produced better results than the regular RNN.

Data

They used three different datasets, Flickr8K (8000 images), Flickr30K (31000 images) and MSCOCO (123000 images). Each image is also coming with a 5 sentences description attached to each picture to train the model in the first phase.

Benchmark

First let's evaluate the performance of the method to generate the Image/Sentence alignment

	Image Annotation				Image Search			
Model	**R@1**	**R@5**	**R@10**	**Med *r***	**R@1**	**R@5**	**R@10**	**Med *r***
Flickr30K								
SDT-RNN (Socher et al. [49])	9.6	29.8	41.1	16	8.9	29.8	41.1	16
Kiros et al. [25]	14.8	39.2	50.9	10	11.8	34.0	46.3	13
Mao et al. [38]	18.4	40.2	50.9	10	12.6	31.2	41.5	16
Donahue et al. [8]	17.5	40.3	50.8	9	-	-	-	-
DeFrag (Karpathy et al. [24])	14.2	37.7	51.3	10	10.2	30.8	44.2	14
Our implementation of DeFrag [24]	19.2	44.5	58.0	6.0	12.9	35.4	47.5	10.8
Our model: DepTree edges	20.0	46.6	59.4	5.4	15.0	36.5	48.2	10.4
Our model: BRNN	**22.2**	**48.2**	**61.4**	**4.8**	**15.2**	**37.7**	**50.5**	**9.2**
Vinyals et al. [54] (more powerful CNN)	23	-	63	5	17	-	57	8
MSCOCO								
Our model: 1K test images	38.4	69.9	80.5	1.0	27.4	60.2	74.8	3.0
Our model: 5K test images	16.5	39.2	52.0	9.0	10.7	29.6	42.2	14.0

We can see here that while using the 5K test images the model performs quite in the average compared to the other methods, it is much more efficient when the training set is bigger and then becomes the best performer in the state of the art.