

Series Monday, Nov 26, 2018 (Deep Learning, Exercise series 9)

Problem 1 (LSTM):

Recall the structural of the LSTM cell (in figure a.) with the following explicit equations:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}), \quad (1)$$

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) \quad (2)$$

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) \quad (3)$$

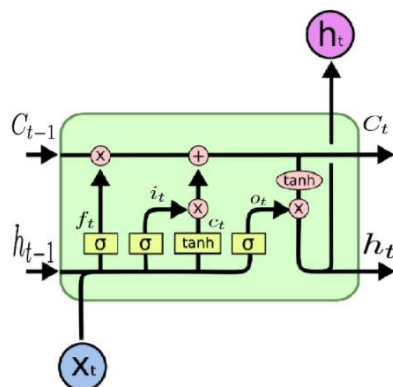
$$\tilde{c}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$

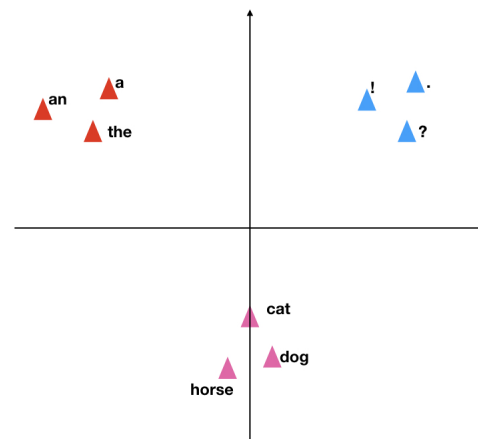
$$h_t = o_t \odot \tanh(c_t), \quad (6)$$

The input of the cell at time step t , denoted by x_t in figure a, is from word embeddings of figure b.

- Determine the direction of weights $W^{(f)} \in \mathbb{R}^2$ such that the LSTM unit forgets the past in a right time. Justify your choice for the weights.
- Determine the direction of weights $W^{(i)} \in \mathbb{R}^2$ such that the LSTM avoids unnecessary changes in the hidden state. Justify your choice for the weights.



a. LSTM cell



b. Embeddings

Problem 2 (Practical: RNN Sentiment Classification):

Model In this exercise, you have to implement a RNN that maps a sequence of inputs to a sequence of outputs while maintaining a state h_t . In our case the inputs will be words of a sentence and the state will be our sentence summary. At the end of the sentence, this summary should provide a fixed-size representation of the sentence that we can use to predict the sentiment. Just like the CNN max-pooling representation. Figure 1 shows the general RNN architecture.

Dataset We will use the *twitter-dataset* that has been used for the previous assignment (See exercise 8).

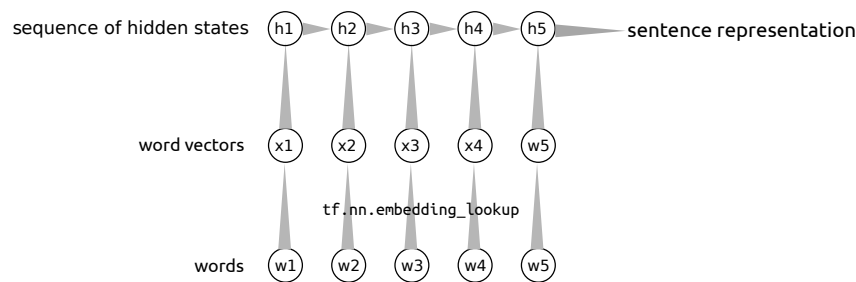


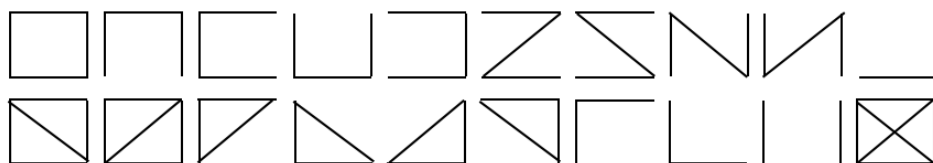
Figure 1: Abstract RNN architecture


Task We provide the training loop (*train.py*) and data helper script (*data-helper.py*), as well as a skeleton for the RNN model (*text-rnn.py*). You should complete the code and fill in the gaps in the RNN model.

1. Define the recurrent computation using the cell classes of tensorflow¹. Use a single RNN cell for a start (see: `tf.contrib.rnn.BasicRNNCell`). The number of hidden units is a hyper-parameter (use 64 as a start). Also your model should support multiple RNN cells (see: `tf.contrib.rnn.MultiRNNCell`). What is the effect in the accuracy when increasing the number of RNN cells?
2. Use `tf.nn.static_rnn` to define your RNN given the cell from above. The returned final state is your new sentence representation. This function expects a *python list* of tensors as input, so you might want to take a look at `tf.unstack`.
3. Replace the RNN cell with an LSTM cell² and compare the performance of both models using the same hyper-parameters.
4. Investigate the effect of the hyper-parameter values, e.g. how is the accuracy affected when the size of the hidden state is changed?

Problem 3 (Differentiable memory):

We want to retrieve the following 22 images from a differentiable memory. Suppose that each memory unit



can save one image. For example image  can be stored in one memory unit. Determine the minimum number of memory units with their content to resembles all the above images.

Problem 4 (Neural turning machine):

Consider the following two programming tasks: (i) copying a sequence of characters (ii) multiplication of two integers (iii) sorting a sequence of integers.

- a. For each above programming task, design a training set for NTM.
- b. Which program is easier to learn by a Neural Turning Machine (NTM)?
- c. Why do we need a **differentiable** memory for NTM?

¹https://www.tensorflow.org/api_guides/python/contrib.rnn

²See <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> or the original paper[1] for an explanation of the recurrent computation.

References

- [1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.