Neural Networks KU WS18

708.076 18W

Exercise Sheet 5

Recurrent Neural Networks [4P]

For this exercise, form groups of two students (see also last paragraph).

Use TensorFlow 1.5 in Python 3 for this task.

Train a recurrent neural network to predict the next character of a word from the embedded Reber grammar. The embedded Reber grammar was used as a task in the original LSTM paper[1]. It is shown in Fig. 1. The grammar is provided in the file `grammar.py`. The script includes a function that samples a string from the grammar, a function that converts a string into a one-hot-encoding, and a function that produces the target outputs of the network for a given string.

The input to the network is one symbol per time step, where symbols are encoded in a one-hot vector encoding (the grammar is composed of 7 symbols).

The task is to predict for a sequence of input symbols the possible next symbols within the grammar. Hence, the output of the network is a vector from $(0, 1)^7$ at every time step, each component giving the probability that the corresponding symbol is a possible next-symbol in the sequence. Note that, since several symbols can follow a given sequence, the output is not 7-class categorical. Instead, we treat each output as an independent Bernoulli varialbe (use logistic sigmoids as outputs).

Task details [4P]:

- Generate 5000 traing samples, 500 validation samples, and 500 test samples from the embeded Reber grammar.
- Train an LSTM with one hidden layer, consisting of 14 LSTM units. Use Stochastic gradient descent with a suitable batch size. Find a good learning rate.

[1] S Hochreiter and J Schmidhuber. Long short-term memory. Neural computation, 9(8), 1735-1780.

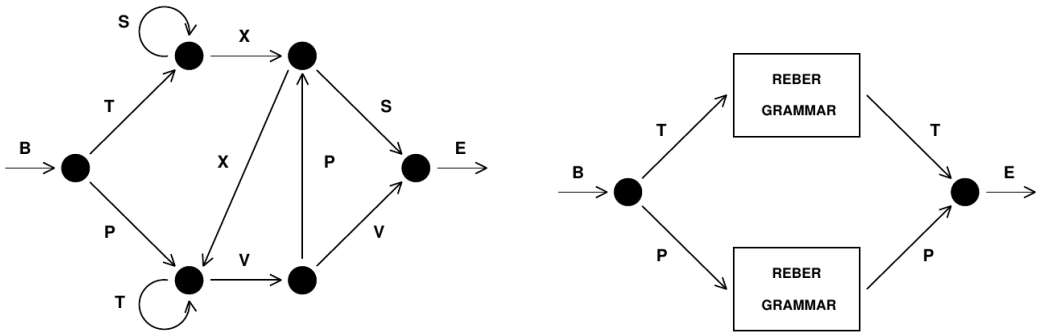

Figure 1: Transition diagram for the Reber grammar (left) and the embeded Reber grammar (right). When two transitions are possible, one tranisition is chosen with probability 0.5.

- Train the same LSTM, but now with ADAM. Find a good learning rate, using default values for the other parameters.
- Provide plots where the evolution of the training/validation error during training is shown for a number of learning rates. Do ten learning runs with the final parameter setting and report the mean and standard deviation of the convergence time (in epochs).

Remarks

- In order to compute a (linear) ouput layer for each time step, you can use the

  `tensorflow.contrib.rnn.OutputProjectionWrapper`

  function.
- Computation of the cross-entropy error for independent Bernoulli variables can be done with

  `tensorflow.nn.sigmoid_cross_entropy_with_logits`.

Submit the code as a single .py file and a report as PDF in the teach center. Do **not** provide one zip with code and PDF but submit them separately. The code should be executeable under the assumption that the files provided at the homepage are available (unzipped) at the working directory. Both students of a group have to submit their python code and report (it can be identical to the one of the group partner). In the report, indicate your group partner on top of the first page (write "Group partner: ⟨First name⟩, ⟨Last Name⟩, ⟨Matrikelnumber⟩").