NLU course projects

Alfredo Ceneri (249971)

University of Trento

alfredo.ceneri@studenti.unitn.it

1. Introduction

- *replacing RRN with LSTM* I started this lab by replacing the RNN module with an LSTM module and optimizing its hyperparameters. I optimized the following hyperparameters: embedding size, hidden size, layers, learning rate and batch size. I used a surrogate model technique to minimize the number of sample training runs, in total 50 training runs were sampled.
- *adding two regular dropout layers* In this section i add two regular dropout layers. One is applied after the embedding layer and one is applied after the lstm module.
- *using AdamW* I replace SGD with Adam and optimize the learning rate by a simple line search from a range of 3e-5, 1e-3.
- *Adding weight tying* In this section i add weight tying. I set the weights for the linear output equal to those of the embedding layer. Since we were specifically instructed not to add an additional linear layer to match the embedding output size and the last linear layer's input size, i arbitrarily chose to modify the embedding size to match the hidden size of the lstm module.
- *Adding variational dropout* Initially i added variational dropout after the embedding layer and before the last linear layer, which was trivial. I then adapted a module from the following repository: https://github.com/salesforce/awd-lstm-lm/ and modified it, making the dropout mask consistent across input to hidden connections and hidden to hidden connections. I then experimented with a few dropout configurations, even using the naive part 1 dropout (with surprising results).
- *Adding non monotonically triggered SGD* I implemented non monotonically triggered SGD, which can be seen as an early stopping - type of criterion. The two hyperparameters for NtAvgSGD are the logging interval and the n value, that determines how many initial steps we skip. I used the hyperparameters suggested by the paper, with logging interval equal to iterations in an epoch and n = 5.

2. Implementation details

3. Part 1

Replacing the RNN module with the LSTM module was straightforward and so was adding two regular dropout layers and using AdamW as an optimizer.

4. Regularization techniques

For weight tying only a few lines of code were necessary: i set the weights of the last linear layer equal to those of the embedding layer.

For variational dropout i had to consistently apply the same dropout mask on two tensors:

- on the layer-to-layer connections in the lstm
- on the lstm-to-lstm recurrent connections

In addition to that, i also implemented a variational dropout layer to apply after embeddings and before the last linear layer.

The original variational dropout required a finer control on the lstm module's weights. I used a wrapper class taken from the following repository: https://github.com/salesforce/awd-lstm-lm/ but then i still had to modify it to experiment with variational dropout. LSTM's have two types of weights that can be dropped out: input to hidden connections and hidden to hidden connections. Input to hidden connections connect different lstm layers and act for a given step in the sequence. Hidden to hidden connections represent the recurrent connections between lstm cells at different time steps in the sequence.

I tested three options for the way in which i apply masks. The first option was the intended variational dropout way: different masks per each connection that are consistent across time steps.

Then i tried fixing the mask across different input to hidden and hidden to hidden connections.

I also tried adding the embedding and last linear layer's variational dropout.

For NtAvgSGD, i record the model's dictionary state at each epoch with the corresponding perplexity on the dev set. Once the triggering condition is set, i then use the list of dictionary states to create the averaged weights and i apply them to the model. The hyperparemeters for NtAvgSGD are those suggested by the paper: logging interval is equal to the number of iterations in an epoch and the lower bound for the trigger is set to 5.

5. Results

The optimization of hyperparameters was carried out for the first model, with no additional modifications. The optimal parameters are: embedding size = 350, hidden size = 500, layers = 3, learning rate = 2.4, batch size = 8. These hyperarameters stay the same before weight tying is applied. Then, since the embedding size and the hidden size must be equal, i set the embedding size to 500.

I report the table with the results obtained for part one:

Table 1: *Assignment 1 Part 1*

Module parameters	test set ppl	parameters
best LSTM	113.83	best hyperparameters
naive dropout LSTM	108.79	dropout frequency = 0.1
Using adam	121.48	learning rate = 1e-3

These are the results i obtained for the second part:

Table 2: *Assignment 1 Part 2*

Module parameters	test set ppl	parameters
weight tying	112.64	
variational dropout 1	107.94	dropout frequency = 0.1
variational dropout 2	118.40	dropout frequency = 0.1
variational dropout 3	125.24	dropout frequency = 0.1
variational dropout 4	124.53	dropout frequency = 0.1
NtAvgSGD	113.52	L = 1 epoch, n = 5

Variational dropout 1: variational dropout with embedding and hidden states' dropout from part 1. Variational dropout 2: variational dropout with no embedding and no hidden states' dropout. Variational dropout 3: keep the same masks for input to hidden and hidden to hidden connections, no embedding and hidden states' dropout. Variational dropout 4: same masks for lstm connections across layers, with embedding and hidden states' dropout.

6. Images

Figure 1: *PPL per epoch for SGD trained lstm with different learning rates*

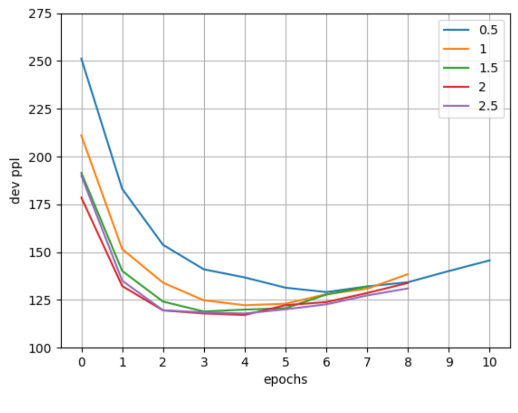


Figure 2: *PPL per epoch for Adam trained lstm with different learning rates*

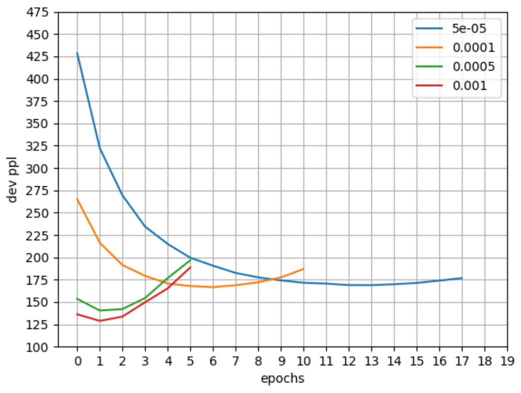


Figure 3: *best lstm compared to lstm with naive dropout*

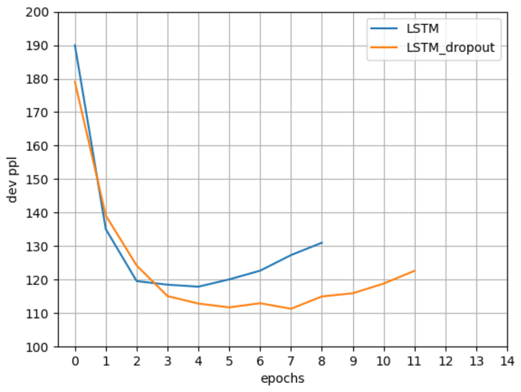
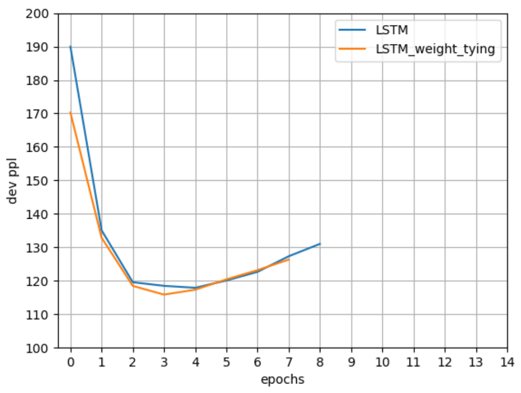


Figure 4: *best lstm compared to lstm with weight tying*


LSTM
LSTM_weight_tying
dev ppl
epochs