NLU Assignment 1

Edoardo Cecchinato (247434)

University of Trento

edoardo.cecchinato@studenti.unitn.it

1. Introduction

This assignment consists in the modification of a given RNN for language modeling, changing the architecture and adding features incrementally. The exercise is dived mainly in two parts. The first part requires the changing from RNN to LSTM, the adding of dropout layers, and the replacement of SGD optimizer with AdamW. The second one instead consists on applying more advanced changes to the LSTM model: weight tying, variational dropout, and non-monotonically triggered AvSGD. For this last task I used as reference the given paper[1] where the authors analyze these techniques.

2. Implementation details

For the first part I modified the given RNN module changing the RNN layers with an LSTM one, then incrementally added dropout in the module for the embedding and LSTM outputs, and finally I defined a new PyTorch AdamW optimizer in the *main* function.

In the second part I took inspiration from the paper that was given to us as documentation[1]. I created a new module named *LM_LSTM_REGULARIZED*, starting from the previous LSTM model, and I added the requested modifications. First I set the weights of the output layer equal to the weights of the embedding layer, in this way I could implement the weight tying functionality requested by the assignment. Then I created a class *VariationalDropout* that apply the same dropout mask for each time step inside a sequence, using the *torch.empty()* function and setting the elements to 0 or 1 based on a Bernoulli distribution with probability *1 - dropout*. As last modification I implemented the non-monotonically triggered AvSGD described by the algorithm in the paper. It can be found in the *train* function, the code checks if the performance aren't improved after a specified number of trials and in this case the optimizer is changed from SGD to ASGD. After the optimizer switching, the patience starts to be checked to exit the train procedure if the performance doesn't improve.

3. Results

The evaluation metric for these experiments is perplexity and the assignment requires a value of it lower than 250. This goal was achieved for each task and any increment modification to the current code led to improvement in the performance. The evaluation of the model was based on the function *eval_loop* where the respective perplexity value is returned as output. Following I'm going to show the results for the original RNN model, the LSTM one without dropout, with dropout, and with AdamW instead of SGD as optimizer, and finally I'll show the scores for the LSTM with weight tying, variational dropout, and non-monotonically triggered AvSGD as optimizer. You can see every result in table 1.

I'm also going to show the parameters used for each model during training in table 2. I'm inserting the *optimizers* and the *learning rates* used by each model. For the learning rates I tried increasing and decreasing but the results were starting to got worst so I chose these values. I'm not inserting other parameters like *clip* (5), *patience* (3), *emb_size* (300) and *hid_size* (300) because I kept them equal for each model since they weren't changing the performance too much.

Model	**Perplexity**
Original RNN	158.0949
LSTM	139.5922
LSTM + dropout	121.5704
LSTM + dropout + AdamW	118.7037
LSTM + weigth tying + Variational dropout + NT-AvSGD	188.9924

Table 1: *Evaluation results for different experiments.*

Model	**Optim**	**L.R.**
Original RNN	SGD	1.0
LSTM	SGD	5.0
LSTM + dropout	SGD	1.0
LSTM + dropout + AdamW	AdamW	0.0001
LSTM + weigth tying + Variational dropout + NT-AvSGD	NT-AvSGD	6,5

Table 2: *Parameters used for different models.*

4. References

[1] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," *CoRR*, vol. abs/1708.02182, 2017. [Online]. Available: http://arxiv.org/abs/1708.02182