04 - Neural LM

Marina Segala (mat. 248447)

University of Trento

marina.segala@studenti.unitn.it

1. Introduction

In the last two exercises of the LM laboratory, in general the main goal was to obtain the value of the perplexity lower than 250, in order to improve the performance of the language model. As a main approach I have tried to change some hyper-parameters to adjust the final results: those changes were made considering both the state-of-the-art and the results obtained in the previous steps, paying attention to not having too much discrepancy. In fact, at each step of both exercises, I have tried multiple times the same phase: this can be considered as a try-and-error approach for finding the best hyper-parameters to use. [1, 2]

2. Implementation details

In the first task, as an initial step, I tried to understand which value of the learning rate was better to use and if it was a good idea to cut it in half when the patience reached a certain threshold. After some tries, I decided to use a lr = 5 and halved it when the patience decreased: however this operation is useful only when the lr is not too low, therefore I had an if-case that permits reducing the lr only if it is greater than 1e-2, also seeing that the SDG optimizer is used.

With the progress of the assignment, and the introduction of the two dropout layers, the lr was maintained the same but I decided to focus more on the probability that the dropout layers had to be used: for this reason I did not implement anymore the halving of the learning rate.

In the end, when AdamW optimizer replaced the SGD one, I did some tests maintaining different values for the probability of the dropout and combining a lr small enough. With the changing of the optimizer, the lr chosen was 1e-3 and there was no need anymore for reducing it manually (this function is already implemented in the AdamW optimizer).

Starting from what has been done in the previous task, in the second part of the assignment, I started to execute and improve the performance of the model from a starting initialization: the model used was an implementation of the LSTM, without any adjustments. Also based on what is described in the paper (Merity et al., 2017) [3], as first thing, I added the weight tying: for doing that, I had to set the size of hidden layer equal to the one of the embedded layer. Since I was using the SGD as optimizer, the lr was set to 5 - considering the results obtained in the previous task.

After that, to increase the performance and obtain a lower PPL, I implemented the Variational Dropout (instead of inserting the 'normal' one): it is more adaptable and it can be different also inside the same networks, thanks to the usage of the Bernoulli distribution. At this point another important factor to consider was the dimension of the size of the batch: until now performances were good anyway. Reducing its size from 128 to 64 is important especially for the creation of the train_loader: this permits the model to reach a higher level of generalization and the training phase is more stable.

As the ultimate step, the optimizer was changed and a non-monotonically Triggered AvSGD was applied, rather than the SGD in the entire process. Its implementation was built 'manually', following also the direction described in [3]: the model initially used the SGD optimizer, but at the first downscale of the performance, it is substituted by the ASGD one.
I decided to manually save each time the parameters of the best model, i.e. when the new value of the ppl calculated is better than the best_ppl_val, and for each epoch there is the updating of the weights and parameters of the model.

3. Results

In both exercises, the evaluation of the model was done by calculating the perplexity, always remaining under a (pre_defined) threshold equal to 250. To see how well the model was performing, I decided to plot data for each epoch, in order to see the trend of the perplexity, either for the training or validation set. Indeed in this way, I was able to understand if the model was overfitting or not, and in case of a positive answer on that, I tried different techniques for adjusting hyperparameters and obtaining better results.

I noticed that there is a problem during the iterations for calculating the training and evaluation loop: at a certain point, the values of the PPL (and so the ones of the losses) start to increase. I tried to adjust the problem and as a result, those peaks are followed by a decrease and an adjustment of the curves.

Improvements (phase I)	**PPL**
Insert LSTM	145.62
Adding Dropout	125.62
Use AdamW	121.63
Improvements (phase II)	**PPL**
Adding weight tying	131.18
Use variational Dropout	94.45
Apply non-monotonically AvSGD	92.18

Table 1: *Decreasing of PPL value starting from an initial state and incrementally adding improvements*

4. References

[1] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." vol. 2, no. 3, pp. 1045–1048, 2010.

[2] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," pp. 5528–5531, 2011.

[3] 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

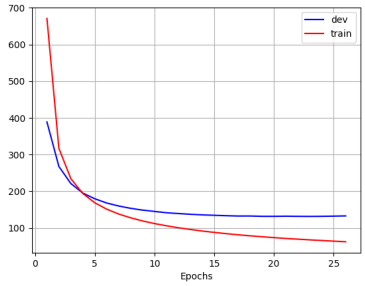

Figure 1: *Graph of the PPL values in phase I, after the insert of all improvements*

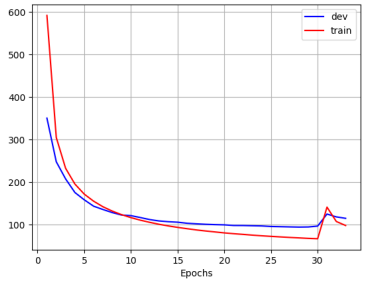

Figure 2: *Graph of the PPL values in phase II, after the insert of all improvements*