NLU course project lab4

Mattia Rigon (mat. 217182)

University of Trento

mattia.rigon@studenti.unitn.it

1. Introduction

In this project several techniques have been tested in order to improve the performance of a Neural Language Model. In the first part it has been measured the performance difference between a LSTM model and a RNN one and several others regularization techniques and experiments with the hyperparameters have been examined as well.

In the second part, more advanced regularization techniques were tested as suggested in the paper by Merity et al. [1]: weight tying, Variational dropout(Gal et. al. [2]) and Non-monotonically triggered AvSGD.

2. Implementation details

Firstly, in the first part, the RNN was replaced with the LSTM architecture since it had been proved to be more powerful, especially with long sequences. Secondly, two dropout layers were added: one after the embedding layer and the other just before the last output linear one. The first part was finally improved by switching the SGD optimizer with the AdamW one. All the three techniques show an incremental improvement in the perplexity reached by our neural language model; for this reason all the three techniques have been kept.

For the second part, all the improvements were applied starting from the model design that was implemented in the first point of the first part, the LSTM baseline. All the three methods implemented in this part are suggested in the paper by Merity et al. [1] which investigates strategies for regularizing and optimizing LSTM- based models.

The first method is 'weight tying' (Inan et. al. [3]) which consists of sharing the weight between the embedding and the SoftMax layers, in this way providing our model with a smaller dimension and a better performance. The implementation was quite trivial; the only thing to pay attention to is that the dimensions of the two layers must be the same.

Then the variational dropout was implemented since, with the standard dropout, a new binary mask is computed each time the dropout function is called. The idea proposed by Gal et. al. [2] is to keep the same mask for each batch of data. Since PyTorch does not implement it, the implementation requires the creation of an 'ad hoc' class where, for all the data in the batch, the same binary mask is applied.

Finally, the use of another optimizer in order to reach better results during the training was implemented. The idea pro- posed by Merity et. al. [1] is to improve the optimization of SGD when it ends in a flat region where it is harder to reach the minimum of the function. The authors suggest to use AvSGD only when SGD stops working well. In order to solve this problem, the Non Monotonically Triggered Avearage SDG (NT-AvSGD) was adopted. In this way, the switch between the two optimizers occurs only when the SGD converges to a steady-state distribution. The trigger event was implemented in a slightly different way than it is in the https://github.com/salesforce/awd-lstm-lm. It was used with the same patience algorithm that is used for early stopping; in more detail, when the patience reached negative values and the optimizer was SGD, it triggered the switch to ASGD. This change only represents a different implementation of the same idea suggested by the authors and it gives the same results.

Further, during the training process, as it is suggested by Merity et al. [1], a learning rate that is adaptive to the length of the sequence was implemented. Moreover, a learning rate that decreases over time was used, exactly as it is implemented by the authors. In more detail, starting from 10, it is multiplied by 0.75 every 5 epochs until it reaches the lower bound of 1.5. These values have been established as optimal during the training experiments.

The implementation suggested in https://github.com/salesforce/awd-lstm-lm has been used as reference for this project.

3. Results

In the first part, it has been proved that all the regularization techniques that were tested improved the results, providing a lower perplexity to the language model.

Table 1: *Performance comparison of different regularization techniques on LSTM models. This result has been reached by using a hidden size and embedding size of 400, a learning rate of 5 for the first two with SGD and* 10^{-3} *for AdamW.*

Regularization technique	Final PPL
LSTM	153.12
LSTM + Dropout layers	145.11
LSTM + Dropout layers + AdamW	108.36

In the second part, the higher complexity of the suggested techniques reached better results.

Table 2: *Performance comparison of different regularization techniques on LSTM models. This result has been reached by using a hidden size and embedding size of 650, a learning rate of 10 which decreases over epochs as discussed above, dropout=0.6, and non-monotonically trigger window set to 5.*

LSTM + weight tying	109.06
LSTM + weight tying + variational dropout	92.79
LSTM + weight tying + variational dropout + NT-AvSGD	80.85

4. References

[1] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," 2017.

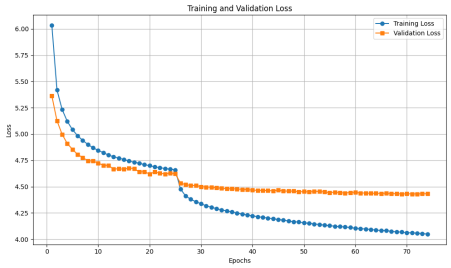

Figure 1: *Plot of the losses in training process, showing the improvement that AvSGD gives.*

[2] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," 2016.

[3] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," 2017.