NLU project exercise lab: 9

Marco Carraro (240642)

University of Trento

marco.carraro-1@studenti.unitn.it

1. Introduction

In Part 1 of the exercise, after collecting the necessary data, I calculated the value of perplexity for three different case histories:

- replaced the RNN model with the LM_LSTM class and modified the Learning Rate to fit the SGD optimizer;
- added two dropout layers to the model;
- replaced the SGD optimizer with an AdamW optimizer, with related modification to the Learning Rate value;

In Part 2 I added some other regularizations:

- Weight Tying;
- Variational Dropout (not working);
- Non Monotonically Triggered AvSGD;

2. Implementation details

In calculating the perplexity of the first two case histories, within Part 1 of the exercise, I kept an SGD optimizer with learning rate value equal to 0.5.
During the different attempts made with different parameters to identify those that offered better performance, I also identified values for hid_size and emb_size, equal to 250 and 400.
The code for evaluating perplexity was taken up and modified as needed from the laboratory lessons. For the addition of dropout levels, I inserted a control that would add, in case it was needed, two Dropout levels, called dropout_emb and dropout_out. Within the forward function, two controls were added that would apply Dropout regularization where needed. Dropout probability values were set to 0.1.
In the third case study in Part 1, by changing the optimizer from SGD to AdamW, I also changed the Learning Rate value to 0.0001, and obtained significant results.
In Part 2, Weight Tying was performed by setting the tie_weights variable to True, again using an AdamW optimizer, to share the weights between the encoder layer and the decoder layer, both embedding layers. I attempted Variational Dropout, for which I also compared with a classmate and tried using his implementation as well. However, I was not able to solve the problems related to it.
For the last part, I used PyTorch's ASGD optimizer, which implements the standard version of AvSGD, this being an effective method to regularize and optimize LSTM-based models.

3. Results

3.1. Part 1

Regularization	PPL results
RNN Replace and changing Learning Rate	231.688
Adding two Dropout layers	216.818
Replace SGD with AdamW optimizer	179.316

To evaluate the models, I used the functions "train_loop" and "eval_loop." The former runs the model training cycle, calculating the loss (loss) and updating the weights with an optimizer; "eval_loop", on the other hand, calculates the loss and perplexity (ppl), a measure of how much the model is confounded by the predictions.
To train the models, I used the train_loop function with a cross-entropy loss function.
In the evaluation of the best model, I used the mechanism named early stopping to avoid overfitting. This involved initializing a variable, called "patience", to 3 and decreasing it each time a perplexity result was obtained that was higher than that found previously. Once patience reaches 0 the cycle stops.
As can be seen from the data, an acceptable level of perplexity was achieved in all three cases, as the goal was to obtain values below the 250 threshold, the lower the results the better the performance. Changing the Learning Rate alone for the model used achieved a value just above 230, which, as can be seen, decreases significantly with the addition of the dropout layers, which probably decreases the possibility of overfitting. The value drops further sharply with the change of the optimizer from SGD to AdamW. With this change, and the related change to the Learning Rate value, a result below 180 is achieved.

3.2. Part 2

Regularization	PPL results
Weight Tying	177.619
Variational Dropout	None
Triggered AvSGD	203.401

Through the implementation of Weight Tying we achieve an even better result than the previous one while keeping the same optimizer. The perplexity value drops to about 177 in this first case study.
As specified in the introduction of this paper I found some difficulties in the implementation of Variational Dropout, having initially followed a different way to perform it that I found out to be incorrect later on. I did not find an adequate solution to this application, even after discussion with a classmate.
In the last case I modified the optimizer by switching to using an ASGD. Again the results were satisfactory with respect to the initial goal of our task, however lower (higher level of perplexity) than those obtained with the AdamW optimizer.