NLU project exercise 2 lab 9

Luca Zanolo (20029226)

University of Trento

luca.zanolo@studenti.unitn.it

1. Introduction

In this lab I modified the model architecture of the first part of this laboratory introducing three of the techniques described in [1], in particular: weight tying, variational dropout and Non-monotonically Triggered AvSGD. If weight tying is enabled, the weights of the embeddings layer of the model and the output layer will be the same. With variational dropout the mask on dropout is fixed for each batch of examples and with the last techniques i defined a new train procedure that implements Algorithm 1 of paper [1].

2. Implementation Details

This section outlines the specific modifications made to the baseline 'LM_RNN' model, focusing on the implementation aspects.

Transition to LSTM: Replaced the RNN layer in the baseline model with LSTM layer.

Dropout Layers: Integrated two dropout layers into the model:

- An embedding dropout layer, applied post-embedding.
- An output dropout layer, applied to the LSTM outputs before the final linear layer.

Weight Tying: In the model, weight tying is implemented by sharing weights between the embedding and output layers. When the `weight_tying` flag is set to `True`, the weights of the `self.output` linear layer are directly linked to the `self.embedding` layer (`self.output.weight = self.embedding.weight`). This setup ensures that the same weight matrix is used for both embedding the input tokens and generating the output predictions.

Variational Dropout: The implementation of variational dropout in the `LangModel` class involves applying the same dropout mask across all time steps for each sequence in a batch. This is controlled by the `variational_dropout` flag. When enabled, the `generate_vd_mask` method is invoked to create a dropout mask (`self.vd_mask`). This mask is applied to the embeddings and the outputs of the recurrent layers. The mask remains constant for all time steps within a batch but changes across different batches.

NT-AvSGD (Non-monotonically Triggered Averaged Stochastic Gradient Descent): The `train_lm_nt_avsgd` function implements NT-AvSGD. During training, model weights are saved periodically. The decision to start averaging these weights is based on the model's performance on the validation set. If there is no improvement over a specified number of epochs (`non_monotonic_interval`), the averaging process begins. The `average_weights` function is used to calculate the average of the saved weights from the epoch where performance started to plateau. These averaged weights are then loaded back into the model.

3. Results

This section presents the Perplexity (PPL) results from the conducted experiments, each varying in model configuration with the use of Weight Tying, Variational Dropout, and NT-AvSGD.

3.1. Setup

The experiments were conducted using a baseline LSTM model with the following parameters: 3 layers, output and embedding dropouts of 0.10, hidden and embedded layer sizes of 400. Each model was trained for up to 100 epochs with a patience of 3 for early stopping. The Adam optimizer was used with a learning rate of 5e-3, except for NT-AvSGD experiments where the optimizer was SGD with a learning rate of 0.7. The dataset is organized in batch of 128, 64 and 128 elements respectively for training, validation and test.

3.2. Results Summary

Experiment ID	Model Configuration	PPL
Experiment 0	Baseline LSTM	220.414
Experiment 1	LSTM + Weight Tying	196.863
Experiment 2	LSTM + WT + VD	228.753
Experiment 3	LSTM + NT-AvSGD	186.292
Experiment 4	LSTM + NT-AvSGD + WT	214.404
Experiment 5	LSTM + NT-AvSGD + WT + VD	232.576

Table 1: *Perplexity Results for Each Experiment*

The results show a varied impact of the implemented techniques on the LSTM model. The lowest PPL was achieved with NT-AvSGD, indicating its effectiveness in this context. However, the combination of Weight Tying and Variational Dropout did not consistently improve performance.

While efforts were made to optimize parameters, the models may exhibit signs of overfitting. Future work could focus on refining these aspects to enhance model generalization and performance.

4. References

[1] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and Optimizing LSTM Language Models," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.