NLU course project - Lab4 - Part1

Emanuele Poiana (247176)

University of Trento

emanuele.poiana@studenti.unitn.it

1. Introduction

Project part one aimed to reduce perplexity(ppl) in an incremental setting. Tuning hyper-parameters played a key role in obtaining improvements over each step of the project. I started by understanding the role of each hyperparameter like patience, gradient clipping, embedding and hidden sizes, and learning rate across the baseline model(RNN).

First, I decide to set a particular hyperparameter configuration, to create a reference point for evaluating independently each new component (LSTM variants, dropout schemes, weight tying, etc.).

Then for each new modification, I tuned the new hyperparameters and tried to reduce even more the perplexity for the given architecture while analyzing the reasons behind an improve in performance.

2. Implementation details

1. Baseline setup: try different combinations of hyperparameters to find the best one
2. LSTM: switch the RNN class with LSTM (torch implementation)
3. Dropout: incrementally add dropout layer inside the model, and tune the new hyper-parameters
4. AdamW: switch the optimizer with AdamW(torch implementation)
5. Weight Tying: share the weights for the embedding and output layer modifing the model architecture
6. Variational dropout: implement it and add like the dropout and compare SGD and AdamW hyperparameters setting
7. NT-ASGD: modify the training loop to perform the switch to Average-SGD like the paper implementation

I set up a baseline RNN model using perplexity as measure for comparison. All the hyper-parameters search in both parts have been made by doing different test runs for sample the model response to some relevant intervals of values, this to not spend hundreds of runs just to find the "perfect" hyper-parameter but focus more on understanding why certain range of values works better than others.

Starting from Lab 4 RNN implementation and methods, I selected a base size (512) for both embedding and hidden dimensions that is the best performing from my test and also is a good compromise in terms of overfitting and underfitting for all future components that will be added. For the optimizer, SGD converged rapidly in the 0.1-0.9 range of learning rates but showed some instability. I tested gradient clipping up to 64, but did not notice changes in ppl probably given the dataset's short sequences, so I choose a low value(5). I decided to implement a custom patience mechanism where, instead of resetting the patience each time a better perplexity (from the evaluation set) is found, a counter is decrease(minimum 0 maximum the patience set), this prevented wasting epochs in unstable training conditions.

Change the RNN to LSTM was straightforward by replacing the RNN class with the LSTM one already implemented in torch. Then I Introduced two dropout layers incrementally in the model architecture, one before the output linear layer and one after the embedding layer, this permits to find the best hyper-parameters configurations.

Moving on AdamW implementation I use the torch implementation and tune with some runs both learning rate and weight decay. For weight tying I just replace the output linear layer weights in the with the embedding weights, I then set the embedding size equal to the output size in the layer definition which is required by weight tying. Implementing Variational Dropout [1] requires to create a new layer class and place it like the dropout layers of 1A. Finally, to implement NonMonoTonically triggered Average SGD(NT-ASGD) I replicate the paper [1] proposed version, which required to modify the training loop with a permanent switch to ASGD controlled by the evaluation ppl sampled every $logging\_int$ and not before n.

3. Results

All experiments ran up to 64 epochs with gradient clipping = 4 and patience = 5, hidden and output dimension size were 512. The **baseline** optimizer setting was with SGD (lr = 0.5, weight decay = 0), for the AdamW the best setting was lr = 0.005 and weight decay = 0.1 Embedding Dropout was set to 0.3 and the Output to 0.2(gives the same ppl to 0.3-0.3) the Variational Dropout values were the same for SGD while for AdamW emb 0.4, out 0.5. Table 1 compares all configurations. The best model combined AdamW, variational dropout, and weight tying, demonstrating the strongest generalization.

3.1. Discussion

LSTM: Switching from a vanilla RNN to an LSTM yielded lower perplexity and smoother optimization at the expense of extra compute resource and consequently time.
Dropout: Tipically keeping the output dropout rate low is preferred to preserve crucial sequence information on the opposite side, the embedding dropout rate can be higher.
AdamW: which accelerated convergence and achieved lower final perplexity despite more expensive updates.
WeightTying: Weight tying reduced parameter count without hurting performance
VariationalDropout: Variational dropout (repeated masks) supported even higher dropout rates for better generalization.
NT-ASGD: NT-ASGD doesn't have shown substantial differences with standard SGD in this settings, maybe 64 epochs where not enough to actually test NT-ASGD key features, which is more stable training.

4. References

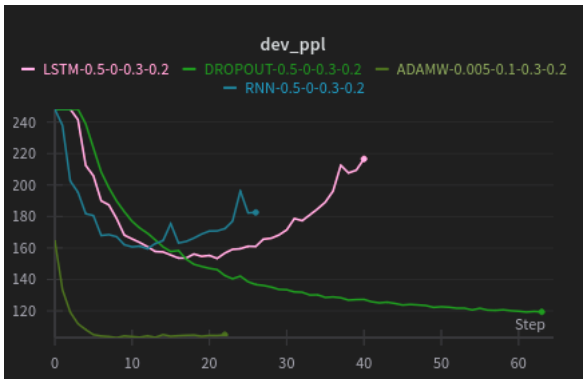
[1] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," 2017. [Online]. Available: https://arxiv.org/abs/1708.02182

Model	Perplexity (PPL)
RNN	158
LSTM	148
LSTM + Dropout (emb 0.3)	122
LSTM + Dropout (emb 0.3, out 0.3)	114
LSTM + Dropout + AdamW	**99**

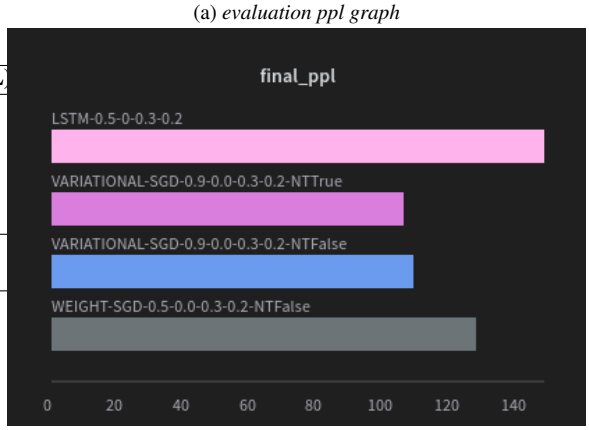
Table 1: *Perplexity comparison Part1A*

Model / Configuration	Perplexity (PPL)
RNN	158
LSTM	148
WeightTying	129
WT + VariationalDropout (emb 0.3, out 0.2)	**117**
WT + VD + NT-ASGD	**117**
WT + VD, SGD (lr=0.9)	107
WT + VD, AdamW	**88**

Table 2: *Perplexity comparison for the incremental model of Part1B, SGD with learning rate 0.5 and batch size 64. Following two extra models that are the best SGD and the best AdamW which reaches the lowest perplexity in part1*

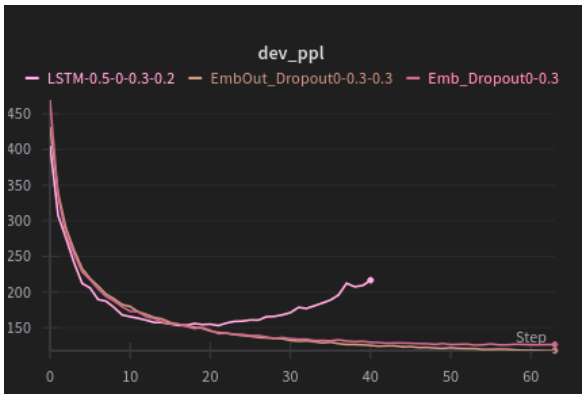


(a) *evaluation ppl graph*

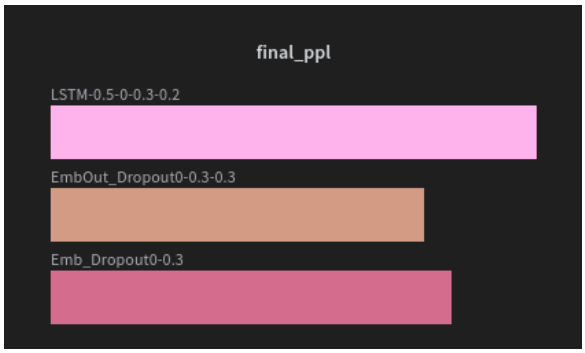


(b) *Test time perplexity comparison*

Figure 1: *Part 1A ppl training, AdamW converges faster and is more stable. Dropout introduces better results and more stable training too*

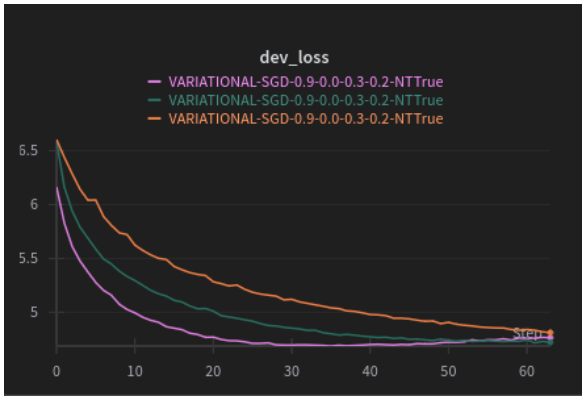


(a) *evaluation ppl graph*

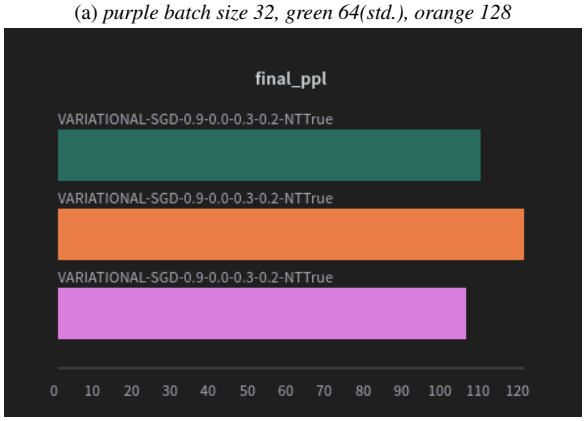


(b) *Test time PPL comparison*

Figure 2: *Part1A, impact of incrementally adding dropout layers*



(a) *purple batch size 32, green 64(std.), orange 128*



(b) *purple batch size 32, green 64(std.), orange(128)*

Figure 3: *Part1B, The effect of batch sizes on the NT-ASGD*