SI630 HW1 Report

Chongdan Pan

January 24, 2022

1 NumPy Logistic Regression

The log-likelihood doesn't coverage very quickly when we're only using one instance for each update.

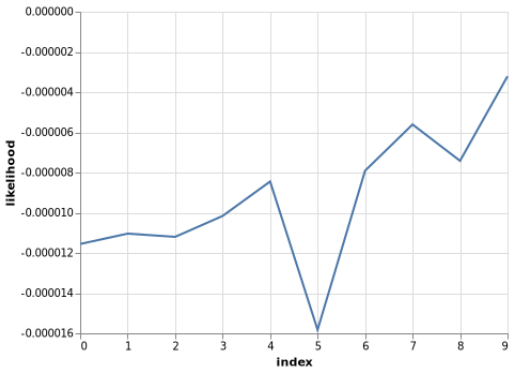

Figure 1: Log likelihood of NumPy model

The F1 score of my model is 0.823

2 Pytorch Logistic Regression

2.1 PyTorch model in 1 epoch

The F1 score of my model is 0.96

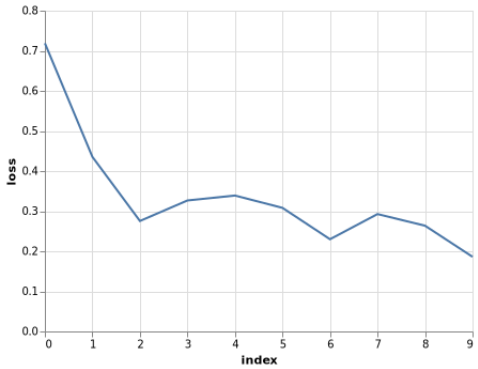

Figure 2: Loss of PyTorch model in 1 epoch

2.2 PyTorch model in 5 epochs

The loss and likelihood is not computed in same unit, but it looks like the loss is more smooth, it's probably due to the different learning rate.

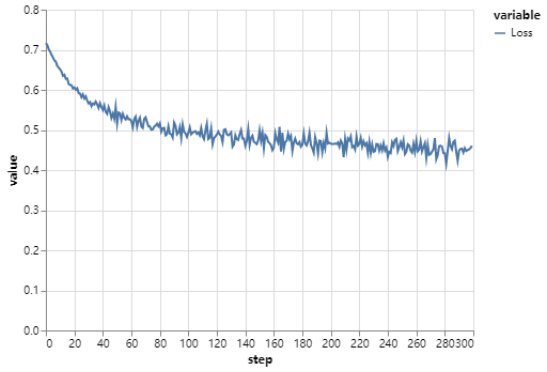

Figure 3: The loss for PyTorch model in 5 epochs

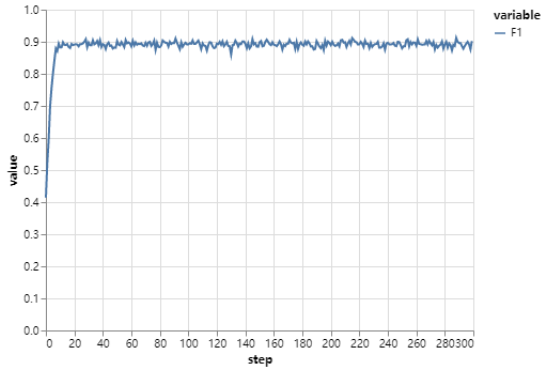

Figure 4: The F1 for PyTorch model in 5 epochs

It looks like the F1 score converges faster than the loss. F1 takes less than one epoch to converge, while loss takes about two epochs

2.3 PyTorch model with different regularization coefficient in 1 epoch

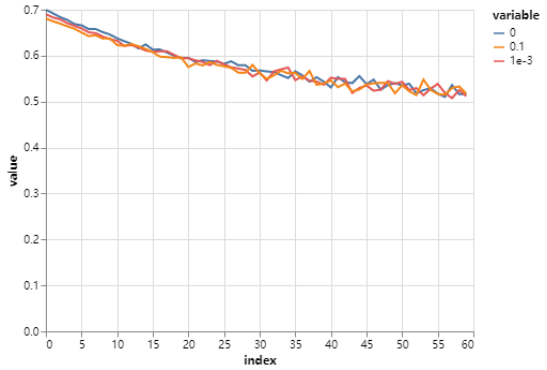

Figure 5: The loss for different regularization terms of PyTorch model in 1 epoch

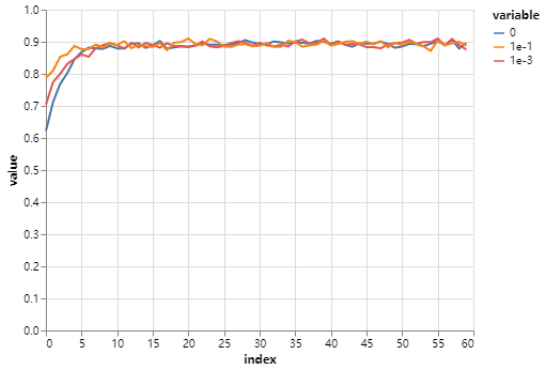

Figure 6: The F1 for different regularization terms of PyTorch model in 1 epoch

It turns out that setting the regularization term will lead to best performance at the beginning, but it won't make a hugh difference in the end

2.4 PyTorch model with different optimizers in 1 epoch

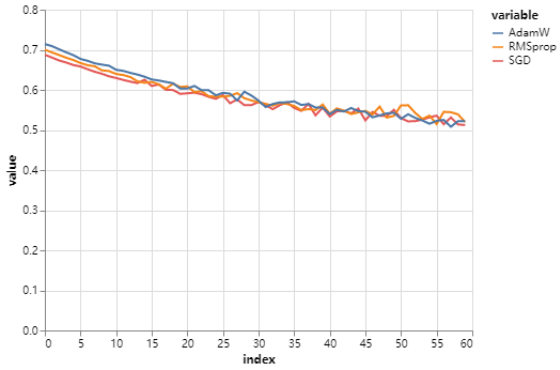

Figure 7: The loss for different optimizers of PyTorch model in 1 epoch

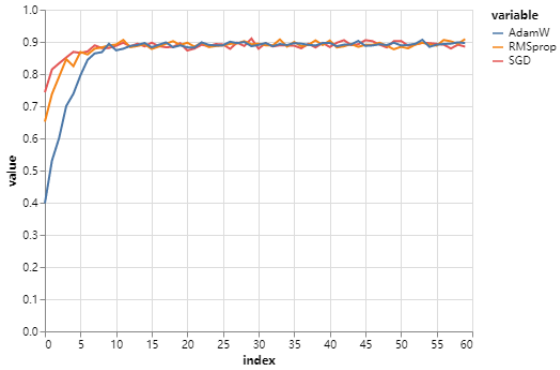

Figure 8: The F1 for different optimizers of PyTorch model in 1 epoch

All optimizers takes roughly same amount of time to converge a close value. Compared to SGD, it looks like that RMSprop will have higher instability while the curve for AdamW is smoother.

2.5 Different tokenizer

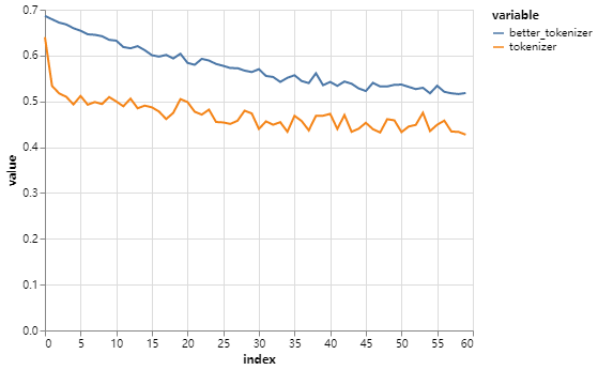

Figure 9: The loss for different tokenizers of PyTorch model in 1 epoch

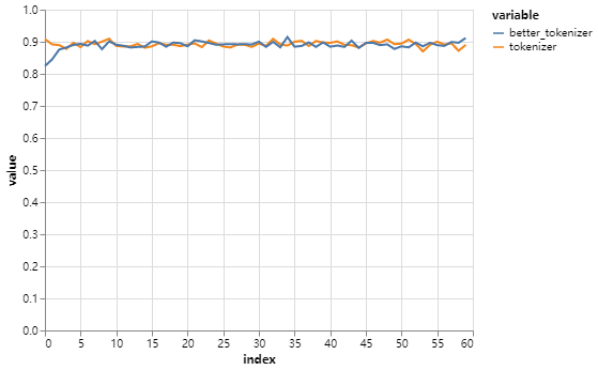

Figure 10: The F1 for different tokenizers of PyTorch model in 1 epoch

It turns out that better tokenizer's loss is higher than raw tokenizer, but its F1 score is a bit higher, too. I think the difference in loss may due to the size of vocabulary set and the shape of matrix.

2.6 Different Learning Rate

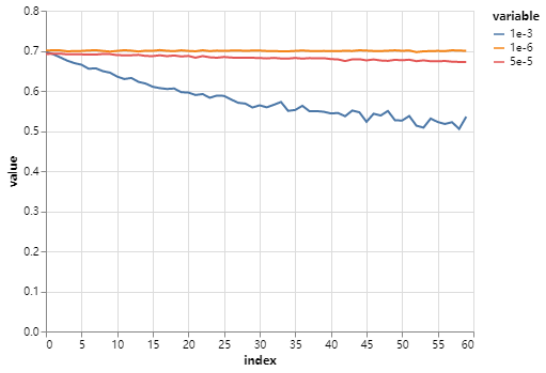

Figure 11: The Loss for different learning rate of PyTorch model in 1 epoch

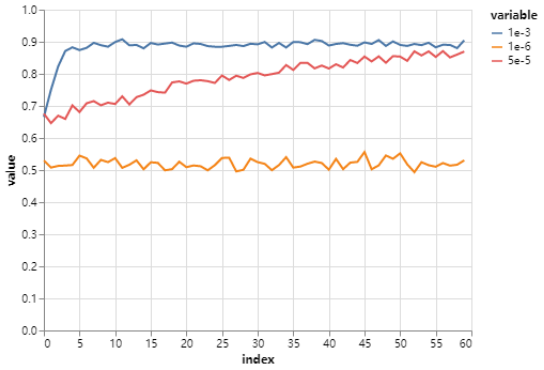

Figure 12: The F1 for different learning rate of PyTorch model in 1 epoch

Higher learning rate will help the loss and F1 converges to a specific value faster, but as shown in the Fig. 11, it will cause high volatility as well.

3 N-Grams Experiment

	token number	max count	mean count	present percent
unigram	15844	154362	604	0.67
bigram	207859	59999	38	0.38
trigram	215575	59999	24	0.28

Table 1: Minimum frequency equals to 1

	token number	max count	mean count	present percent
unigram	12121	154362	788	0.65
bigram	104325	59999	70	0.37
trigram	82951	59999	54	0.31

Table 2: Minimum frequency equals to 10

	token number	max count	mean count	present percent
unigram	4492	154362	2069	0.57
bigram	11812	59999	416	0.31
trigram	7052	59999	381	0.29

Table 3: Minimum frequency equals to 100

It turns out that we have much more tokens of bigrams rather than unigram and trigram, and there about 0.38, 0.67, 0.28 percent tokens from dev appearing in the train data.