Regularizing and Optimizing LSTM Language Models

Team 2 Soyoung Yoon, Dongmin Seo, Jaeyoung Hwang

Background

About the model

NT-AvSGD
- Using AvSGD with well defined guidelines

Weight Drop
- DropConnect the hidden2hidden weight matrix
- Randomly sample a mask for each forward pass
- Efficient. Can use blackbox(ex: CuDNN) LSTM

Embedding and Variational dropout
-Embedding Dropout on the embedding matrix at a word level, thus reducing total parameter size.
-Variational Dropout to overcome the problem of disrupting RNN's ability to retain long term dependencies by appliying same dropout masks over multiple time steps.

Random BPTT length
- Prevents the model from seeing the exact same batches

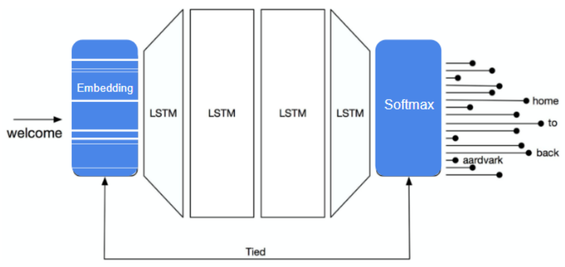
Weight tying
- Sharing weights between input-to-embedding layer and output-to-softmax layer.

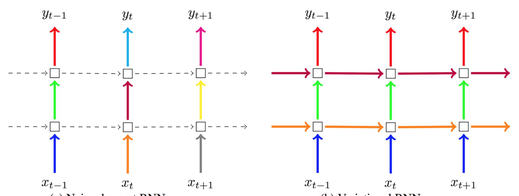
AR & TAR
-AR: Activity Regularization
-TAR: Temporal Activation Regularization
- L2 decay for each activation unit and difference in outputs of RNN

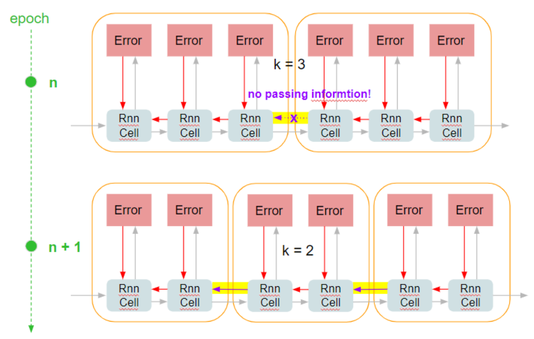
About the changes

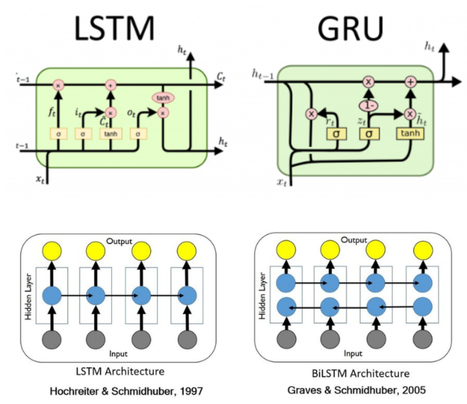
GRU:
- Fewer parameters than LSTM, as it lacks an output gate.

Bidirectional LSTM
- Connect two hidden layers of opposite directions to the same output.









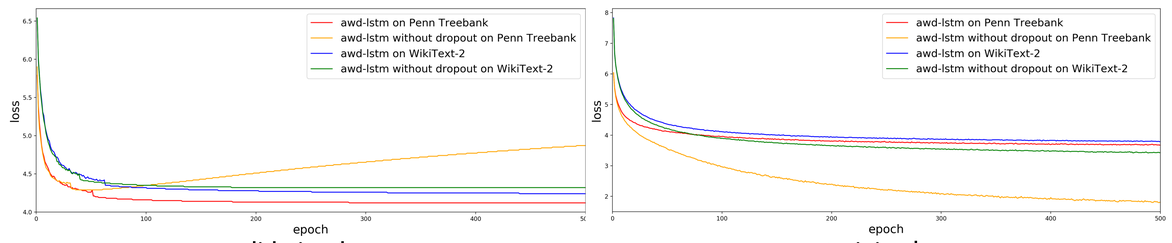
Result

Parameter Size & Training time (time per epoch)

	LSTM	GRU	GRU-5layer	biLSTM
PTB	24221600 48 sec.	19168700 41 sec.	35052500 76 sec.	18611600 58 sec.
WT2	33556078 91 sec.	28503178 76 sec.	-	27946078 96 sec.

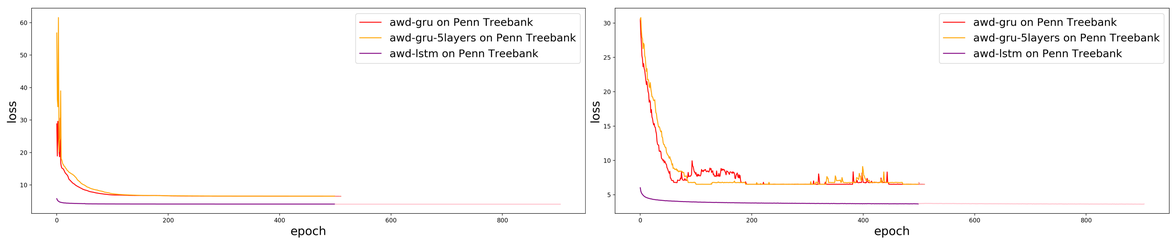
Model Performance

1. LSTM without dropout



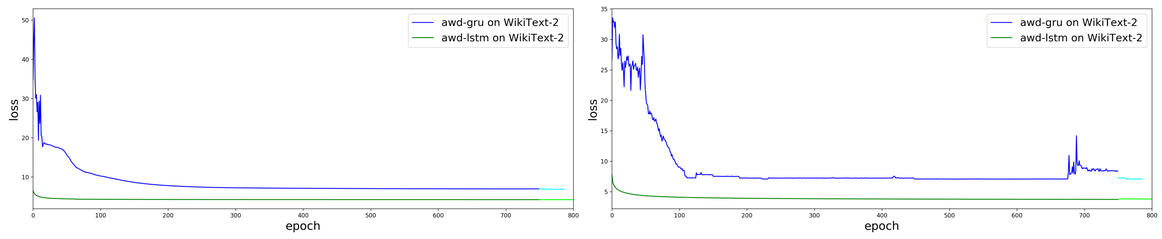
validation loss | training loss

2. GRU with layer 3 and 5 - on Penn Treebank



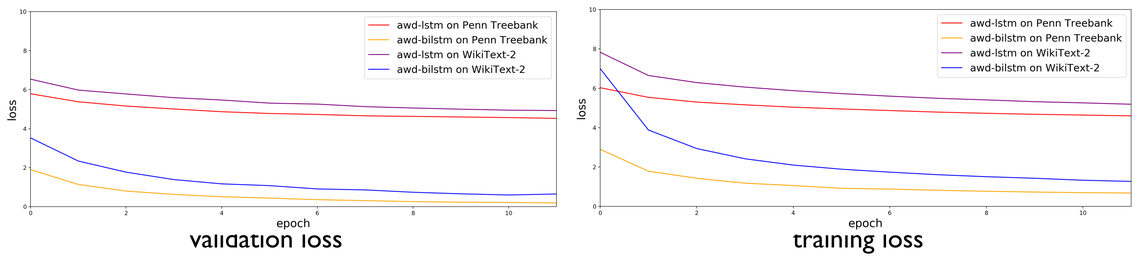
validation loss | training loss

3. GRU with layer 3 - WikiText-2



validation loss | training loss

4. biLSTM



validation loss | training loss

Contribution

Motivation

- Language models for real production has space limitations. (e.g. mobile)
- We need to provide a way to maintain the performance of these neural architectures while substantially reducing the parameter cost (by making it lightweight and faster to train)

Method - model change

1. LSTM without dropout
2. GRU with more layers (5)
3. GRU with same # of layers
4. Bidirectional LSTM with same number of parameters

Model Structure

Original: [
WeightDrop((module): LSTM(400, 1150)),
WeightDrop((module): LSTM(1150, 1150)),
WeightDrop((module): LSTM(1150, 400))
],

GRU(3~5 layers): [
WeightDrop((module): GRU(400, 1150)),
WeightDrop((module): GRU(1150, 1150)),
WeightDrop((module): GRU(1150, 1150)),
WeightDrop((module): GRU(1150, 1150)),
WeightDrop((module): GRU(1150, 400))
],

Bidirectional LSTM: [
WeightDrop((module): LSTM(400, 575, bidirectional=True)),
WeightDrop((module): LSTM(1150, 575, bidirectional=True)),
WeightDrop((module): LSTM(1150, 200, bidirectional=True))
]

Experiment

Experimental Setup

- LSTMs, biLSTMs, GRUs and embeddings were randomly initialized.
- Experiments were conducted on an NVIDIA GTX 1080 ti GPU.
- Datasets were Penn TreeBank and WikiText-2 and hyper parameters are same as written in Github.
- GRU, biLSTM and LSTM without any regularization techniques are used for comparing with 3 layers by default and additionally use 5 layer GRU
- biLSTM has same number of parameters as LSTM.

Experiment Detail

- We first train models with (main.py)
  weight tying,
  weight dropout for hidden to hidden matrixes,
  dropout for input embedding layer and RNN layers.
- Then, we fine-tune the trained model with (finetune.py)
  AR, TAR for decay
  apply random BPTT length
  use non-monotonic trigger for optimization.
- Implement bidirectional LSTMs by ourselves because it was not given on pytorch implementation.
- Conducted Word Level Penn Treebank(PTB) and WikiText-2(WT2) with various models.
- The explicit argument given for training data can be found on the github README. (https://github.com/salesforce/awd-lstm-lm)

PTB - LSTM: python main.py --batch_size 20 --data data/penn --dropouti 0.4 --dropouth 0.25 --seed 141 --epoch 500 --save PTB.pt --model LSTM
python finetune.py --batch_size 20 --data data/penn --dropouti 0.4 --dropouth 0.25 --seed 141 --epoch 500 --save PTB.pt --model LSTM
WT2-LSTM: python main.py --epochs 750 --data data/wikitext-2 --save WT2.pt --dropouth 0.2 --seed 1882
python finetune.py --epochs 750 --data data/wikitext-2 --save WT2.pt --dropouth 0.2 --seed 1882

Discussion

Overall Analysis

- LSTM performs better and gives better result than GRU over both datasets.
- Compared with LSTM, the training time (average time to run over one epoch) is faster for GRU with same number of layers, and slower for GRU with more layers.
- GRU has perturbation at initial training: few parameters of GRU get low impact of gradient vanishing, so parameters are variable than LSTM.
- Training loss increase after 400 epochs : our learning rate is too big to settle minima of loss function. We need to run the experiment with more diverse hyperparameters.
- LSTM with dropout performs better on validation dataset than LSTM without dropout.
- By changing from LSTM to bidirectional LSTM, the number of hidden layers become doubled.

New Findings

- GRU has fewer transferred information between layers than LSTM in the memory cell, so GRU performs worse than LSTMs on Language Modeling.
- Total parameter size of biLSTMs with same structure are actually smaller, but takes more time to train (per epoch).
- Bidirectional LSTMs show unbelievable performance on language modeling task.

Reason why Bidirectional Models show unbelievable performance

- Bidirectional LSTMs are said to have great performance over previous model structures, but having a perplexity of nearly 1 or less (compared to the proposed LSTM model which has single model perplexity of 60) seems like there is a problem.
- We are calculating the perplexity of the language modeling task. Since the target for language modeling is just the input sequence offset by one, the network "cheats" by using the other direction of the LSTM to know what word is coming next, thus it is nearly 100% confident of what the next word in the sequence will be.
- The solution to this problem could be by using tasks other than language modeling (e.g. generation tasks such as translation tasks), or do what the BERT people did, such as training a language model using masked sequence by giving a task to predict the masked word. Or if we apply bidirectional LSTM only on encoder in encoder-decoder structure, single-model perplexity can be used.