Regularizing and Optimizing LSTM Language Models

Christian Kolding and Lars Bryld

DTU Compute · Technical University of Denmark
Kgs. Lyngby, Denmark

Bioinformatics Centre · University of Copenhagen
Copenhagen N, Denmark

Language Model

Following *Gave et. al.* [1] we define a language model that is a probability distribution over a sequence of words, where V is the size of the vocabulary. Each word in the vocabulary V is represented by a one-hot encoding vector $x \in \mathbf{R}^{\nu} = \nu$, corresponding to its index in the vocabulary. By using the chain rule the probability of a sequence of words $x_1...x_T$ can be expressed as:

$$p(x_1...x_T) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}...x_1) \tag{1}$$

This conditional probability is at the center of our analyses, as we investigate the best approximation using the LSTM model with various regularizing techniques.

Model Specification

Recurrent Neural Network

Assuming that we have a vector $\mathbf{h}_t \in R^d$ encoding the history $x_1...x_t$ the conditional probability over a word w can be paramatized as:

$$p_{vocab}(w \mid x_t...x_1) \propto exp(h_t^T o_w) \tag{2}$$

The history vector h_t is computed using a RRN by recursively applying an equation of the form $h_t = \Phi(x_t, h_{t-1})$. The functional for of Φ is defined by the LSTM architecture.

LSTM Model

Within language modeling the LSTM is proven superior in finding and exploting long range context [2]. This feature stems from the LSTM model using purpose build memory cells which stores information ... . The LSTM model is defined and illustrated below:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{3}$$

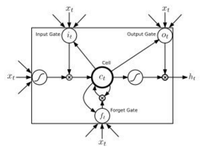
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{4}$$

$$c_t = f_t c_{t+1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{5}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{6}$$

$$h_t = o_t tanh(c_t) \tag{7}$$

where σ is a logistic sigmoid function, i_t is the input gate vector, f_t it the forget gate vector, c_t is the cell activation vector, o_t is the output gate vector and h_t is the hidden state vector. All vectors are of the same size as the hidden state vector h_t. Note that the weight matrices are diagonal meaning that element m in each gate vector only receives input from element m of the cell vector.



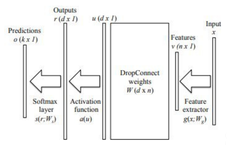
Single LSTM Memory Cell

Regularization and Optimization

Given the very flexible architecture regularisation is essential to avoid over fitting. At this stage we have implemented the below three regularization techniques to prevent over fitting.

Weight Drop

DropConnect is applied on the hidden-to-hidden weight matrix where a random subset of weights are set to zero [3].



Single DropConnect layer (M)

Weight Tying

The motivation for weight tying is to reduce the total number of parameters in the model. This method is theoretically motivated by *Inanelal.* and has the purpose of sharing the weights between the embedding and softmax layer. This prevents the model from having to learn a one-to-one correspondence between the input and output.

Randomized BBTT

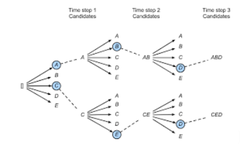
In order to use the data set more efficiently the batch size length for the forward and backward pass is randomly selected in two steps.

- The base sequence length is set to seq with probability p and $seq/2$ with probability $(1-p)$.
- The sequence length seq is selected according to $\mathcal{N}(seq, s)$, seq being base sequence length and s a standard diviation (insert value??)

Finally the learning rate must be rescaled according to the batch size sequence so that short sentences are not favored during training. We have trained our model using $p = 0.95$, $s = 5$, a min. batch size of 5 and a max batch size of 80.

Beam Search

Beam search is implemented in order to reduce memory requirements when generating text. When generating the search tree structure the beam search generated all successors of a given state sorting them in an increasing order. However only a given number of states are saved in memory, the beam sized b, corresponding to the best stated at each level. This should be compared to saving the entire tree structure at any state as in a greedy optimizer $b = 1$.



BeamSearch

Experiments

We train a 3-layer LSTM model, regularized as described, using hidden 800 units in each layer and training for 20 epochs. The word embedding size is set to 400 and we have chosen the optimizer to be ADAM and a gradient clipping with maximum norm of 0.25. The learning rate is set to 0.002. For text generating we use a beam search with a beam size of B = XX.

Training and Validation Result

- A [1] with a word tying LSTM [4].
- Drop Out [3]
- Random BBTT [1]
- Beam Search

Conclusions & Outlook

- A
- B
- C
- Currently implementing the rest of the synopsis.

References

[1] S. Merity and R. Socher N. S. Keskar. Regularizing and optimizing lstm language models. *arXiv1708.02782v1*, Aug 2007.

[2] G. Hinton A. Graves, A-r. Mohamed. Speech recognition with deep recurrent neural networks. *arXiv:1302.5778V1*, Mar 2013.

[3] Z. Ghahramani Y. Gal. A theoretically grounded application of dropout in recurrent neural networks. *University of Cambridge*, MM YYYY.

[4] R. Socher H. Inan, K. Khosravi. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv:1611.01462v3*, Mar 2007.