IMPROVING LANGUAGE MODEL PERFORMANCE WITH ADABOOST INSPIRED ENSEMBLES OF DEEP LSTMS

Nishant Subramani

Northwestern University

ABSTRACT

In this paper, we present two new ensemble methods each with two variants for Recurrent Neural Networks (RNNs) which have Long Short-Term Memory (LSTM) units. We propose two new methods: (1) AdaBoost Inspired Mini-Batch Sampling (ABIMBS) and (2) AdaBoost Inspired Sentence Sampling (ABISS) ensemble methods for the language modeling task. ABIMBS has a forward and backward variant and ABISS has a standard deviation and square root variant. We show that all four of these methods applied to both non-dropout and dropout LSTM architectures for language modeling have lower perplexity than the current state-of-the-art independently trained and uniformly averaged ensemble counterparts.

1. INTRODUCTION

The goal of language modeling is to build a model with an aim of accurately predicting the next word given some sequence of words; that is, a context. Statistical language modeling accomplishes this by probabilities for sequences of words. Language modeling has a plethora of applications. One popular application is speech recognition, specifically trying to estimate the most likely sequence of words given some acoustic evidence. Another popular application is machine translation; more precisely, using language modeling to improve software-aided translation tasks between languages.

This work focuses on improving the overall accuracy of state-of-the-art language models. The advent and usage of neural probabilistic language models is fairly new [1], and these models have consistently improved with the development and utilization of new neural network architectures. Although these models seem to perform relatively poorly in comparison to the state-of-the-art neural network models in other domains, language modeling has significantly more inherent unpredictability. Given various types of language modeling applications such as speech recognition and machine translation, improvements to accuracy are expected to have far-reaching impact in a variety of domains.

In this paper, the language modeling task is the only domain considered, the first task where recurrent neural networks (RNNs) have achieved significant success [2] [3] [4]. We develop two ensemble methodologies, inspired by Boosting, to be used in conjunction with long short-term memory networks (LSTMs) for this task.

2. PRIOR WORK

RNNs achieve state-of-the-art performance on language modeling. Ensembles of RNNs are even more effective, but these ensembles tend to be independently trained and uniformly averaged [5]. Language modeling requires large models due to the sheer number of tokens and relatively large size of embeddings, but since large RNNs tend to overfit, practical applications often use very small RNN models that do not fully describe the complexity of language modeling tasks. Recently with regularization through dropout [6], neural networks have improved performance in a variety of applications. Although dropout does not work well for RNNs, for LSTMs, dropout, when correctly used, greatly reduces overfitting in a variety of applications and provides state-of-the-art performance in language modeling [7]. Other researchers have also applied dropout to LSTMs and have achieved performance benefits on a variety of tasks [8] [9].

In general machine learning, boosting, a traditional meta-algorithm that combines multiple weak learners to form one strong learner, has had considerable impact improving performance on relatively simple models. Adaptive boosting, AdaBoost, has been widely used with decision trees and is referred to as the best-out-of-the-box classifier [10]. Boosting and more specifically AdaBoost have then been applied to neural networks and evaluated on general hand-written-digit recognition and image recognition tasks [11] [12]. Using multi-layer perceptrons (MLPs) and artificial neural networks (ANNs) respectively on these tasks, these works were able to improve generalizability and performance. Although boosting has seen promising results in these neural architectures, it has not been used in conjunction with LSTMs in language modeling. Due to the complex nature of language modeling, namely the fact that state-of-the-art models have high perplexity, these models can be thought of as weak learners, and

thus good candidates for boosting.

3. METHODS

In this section, we first introduce AdaBoost briefly. Next, we formalize our two main contributions for the language modeling task: (1) AdaBoost Inspired Mini-Batch Sampling (ABIMBS) and (2) AdaBoost Inspired Sentence Sampling (ABISS).

3.1. General AdaBoost

AdaBoost in a general context is given by the psuedocode below in Algorithm 1 [13]. In this setting, we are given m labeled training examples $\{(x_1, y_1), ..., (x_m, y_m)\}$ where x_i's are in some domain X and y_i's are binary variables. In each round $t = 1, ..., T$ a distribution D_t is computed over those m training examples. A weak learner is applied to find some weak hypothesis $h_t : X \mapsto \{-1, +1\}$, where the weak learner's objective is to find a weak hypothesis with low weighted error ϵ_t relative to the distribution D_t. Then the combined hypothesis is given by Equation 1.

$$F(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{1}$$

Algorithm 1 AdaBoost Pseudocode

1: Given: $\{(x_1, y_1), ..., (x_m, y_m)\}$ where $x_i \in X, y_i \in \{-1, +1\}$

2: Initialize: $D_1(i) = \frac{1}{m}$ for $i = 1, ..., m$.

3: For $t = 1, ..., T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \mapsto \{-1, +1\}$
- Aim: select h_t with low weighted error:

$$\epsilon_t = Pr_{i\ D_t}[h_t(x_i) \neq y_i]$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Update, for $i = 1, ..., m$:

$$D_{t+1}(i) = D_t(i) \frac{e^{(-\alpha_t y_i h_t(x_i))}}{Z_t}$$

- Z_t is a normalization factor that is chosen such that D_{t+1} will be a distribution

4: Output Final Hypothesis:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

3.2. AdaBoost in Language Modeling

In language modeling, however, each x_i is the previous word in a sequence of words for a specific y_i, the next word in the sequence. The goal of language modeling is to assign probabilities to the specific y_i's. Given this, the loss function we use in determining ϵ_t is given in Equation 2 where v is the vocabulary size. We initialize $D_1(i)$ in the same manner as above keeping the same α_t calculation. Another key difference is that we modify the way that we weight each example as shown in Equation 3. The factor $\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ is derived from the AdaBoost algorithm and then we normalize such that the vector W_{t+1} has a sum of 1. We also follow the same ensembling technique as AdaBoost given above in 4.

$$\epsilon_t = \frac{1 - \left[\frac{\sum_{i=1}^{m} \ln\left(\frac{1}{v}\right) - \sum_{i=1}^{m} \ln(P(y_i))}{\sum_{i=1}^{m} \ln\left(\frac{1}{v}\right)}\right]}{2} \tag{2}$$

$$W_{t+1}(i) = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \cdot W_t(i) \cdot \sum_{i=1}^{m} \ln(P(y_i)) \tag{3}$$

3.3. AdaBoost Inspired Mini-Batch Sampling (ABIMBS)

For the language modeling task, we use mini-batch gradient descent to learn the weights for the LSTMs we construct. We use sampling with replacement and adjust the sampling probabilities of each token i in each iteration t. The sampling probabilities for each token i in each iteration t is exactly $W_t(i)$. These sampling probabilities are calculated before the training step and remain fixed for each weak learner within each epoch and throughout all epochs.

There are two significant variants of the ABIMBS method: (1) Forward ABIMBS (FABIMBS) and (2) Backward ABIMBS (BABIMBS). These variants simply deviate when the number of rollout steps for the LSTM is greater than 1. FABIMBS samples starting locations based on case weight and rolls out the context following the word, while BABIMS samples ending locations based on case weight such that the last input-output pair in the rollout is the sampled case weight with the context preceding it. The formulation of ABIMBS weighting is given above in Equation 3. We apply this methodology to both the dropout and non-dropout LSTM architectures for this task where each LSTM in the ensemble is a weak learner.

3.4. AdaBoost Inspired Sentence Sampling (ABISS)

In ABISS, we *sentence sample* for the language modeling problem. This method constructs a new training set using the original training set by extending it by increasing the number of sentences by 50%. At the beginning, we construct a new training set with 1.5 times the number of sentences as the original training set. Instead of weighting examples like AdaBoost and ABIMBS, we weight sentences in a similar manner starting with uniform weighting. We use sampling with replacement and adjust the sampling probabilities of each sentence s in each iteration t to ascertain the new 50% sentence addition. The sampling probabilities for each sentence s with words w_i, ... , w_k in each iteration t are given by Equation 4 below.

In order to keep sentence weights relatively similar, two special variants of ABISS of note are developed: (1) SD ABISS and (2) Sqrt ABISS. In both of these methods, all case weights are modified. In SD ABISS, only the case weights that fall inside of 3 standard deviations of the mean of all of the case weights or in other words within 3 z-scores, contribute in the sentence weight calculation. In Sqrt ABISS, all of the case weights are modified element-wise by the square root operator until the largest case weight among all case weights is less than some constant l times the smallest case weight. For the sake of these experiments, we chose $l = 5$ arbitrarily. We then normalize the vector $W_{t+1}(s)$ such that it has a sum of 1. These sampling probabilities are calculated before the training step and remain fixed for each weak learner within each and throughout all epochs. We also apply this methodology to both the dropout and non-dropout LSTM architectures for the language modeling task.

$$W_{t+1}(s) = \frac{\sum_{i=1}^{k} W_{t+1}(i)}{k} \tag{4}$$

4. EXPERIMENTAL DESIGN

This work tests the hypothesis that the ABIMBS and ABISS strategies listed in the methods section applied to language modeling are likely to improve performance. The performance difference is measured and quantified using word-level perplexity, the standard metric for evaluating language models. Perplexity is defined as the predicted probability estimate assigned to a sequence of words w_1, ... , w_m by a language model raised to the power of $-1/m$ where m is the number of words in the sequence given in equation 5. In other words, perplexity is calculated as the per-word average of the probability that the test data set is generated by the language model. Thus, perplexity reduction by definition yields more precise and generalizable language models on average because the lower the perplexity of a language model, the better the language model is at modeling unseen data. We conduct word-level prediction experiments; namely, we try and identify the joint probability of a sequence of words via the formula given in equation 6.

$$Perplexity = \hat{P}(w_1, ..., w_m)^{\frac{-1}{m}} \tag{5}$$

$$P(w_1, ..., w_t) = \prod_{i=1}^{t} P(w_t | w_{t-1}, ..., w_1) \tag{6}$$

We then evaluate performance by measuring this word-level perplexity of our model defined previously on a sample of the Penn Tree Bank (PTB) dataset [14], the state-of-the-art dataset for language modeling. A perplexity reduction of more than 2% in comparison to the state-of-the-art model would be significant in these analyses. The PTB dataset consists of 929k training tokens, 73k validation tokens, 82k test tokens, and 10k words in its vocabulary. We use the same standard preprocessing that is common to the previous research in which all words not in the vocabulary are mapped to the <unk> token [15] [16]. Our sample consists of 42k training tokens, 3287 validation tokens, 3712 test tokens, but we keep the 10k words vocabulary for ease of generalization between PTB samples.

Since this work aims to compare our ensembling strategies, ABIMBS and ABISS, against independently trained ensembles, we will have two baseline models and eight experimental models. The two baseline models will be: (1) an independently trained ensemble of size k of non-dropout LSTMs and (2) an independently trained ensemble of size k of LSTMs using dropout. These two baseline models are the same types as the state-of-the-art language models that perform best on PTB and in general in this application [5] [7].

We will use two LSTM layers for each LSTM. We will use a small architecture, which has 150 units per layer with uniform parameter initializations in $[-0.1, 0.1]$ taking into consideration our currently available computational resources. We use gradient clipping, clipping the norm of the gradients, which are normalized by a minibatch size of 20, at 4. We train the LSTM with a learning rate of 1, and after 3 epochs, we decrease it by a factor of 0.4 after each epoch for a maximum of 10 epochs. We unroll the sequences for 20 steps such that each training example x_i will be a sequence of length 20 with a resulting y_i of sequence length 20 as well. In the dropout case, we will use 50% inclusion probability dropout for the hidden layers. These LSTM architectures were chosen to be proportionally slightly smaller to the small architecture used

5 LSTMs	Non-Dropout	Dropout
Baseline	454.72	465.96
FABIMBS	433.04	458.48
BABIMBS	437.81	455.48
SD ABISS	413.91	419.03
Sqrt ABISS	415.95	419.42

Table 1. Perplexity Scores for Ensemble Sizes of 5 using different methods

for recurrent neural network regularization [7].

Since the algorithms that need to be tested are additions to these models, comparison to models without the added boosting components (baseline models) is the most appropriate. This experimental setting isolates each of the additional boosting components which are being evaluated. Each of the four methods described in the methods section (FABIMBS, BABIMBS, SD ABISS, Sqrt ABISS) will be used to build an ensemble of size k of non-dropout LSTMs as well as an ensemble of size k of dropout LSTMs. This brings the total of experimental models to 8. Due to computational resource limits the models will be tested at only the ensemble size of 5.

5. EXPERIMENTAL RESULTS AND DISCUSSION

5.1. Experimental Results

In this section, the results of running word-level perplexity experiments on the small sample of the PTB dataset are provided. The results gathered so far from Table 1 show that for the small ensemble sizes of 5, all four methods work well to reduce perplexity. The two ABIMBS methods improve both non-dropout and dropout LSTMs lowering perplexity by about 20 and 9 respectively in comparison to the respective baseline models. The two ABISS methods had a much more significant and large impact improving both non-dropout and dropout LSTMs lowering perplexity by 40 and 45 respectively in comparison to the respective baseline models.

5.2. Discussion

As table 1 shows, all experimental groups performed better in comparison to the baseline. These results are significant due to a variety of factors. Both ABIMBS methods performed better due to the optimization of gradients based on harder examples. This follows directly from AdaBoost and improves our performance. In many cases, boosting methodology requires a large number of ensembles to achieve a strong effect, but in our experiments using just 5 ensembles, a sizable performance increase is seen. The ABISS methods performed vastly better than the baseline models. One reason for this may be the larger amount of data fed into the ensembles (50% more sentences). Although these models had a larger amount of data, those 50% more sentences came from the same training set that was fed into the baseline models, so comparison to the baseline models is still valid. Overall these results show that adding emphasis to hard examples improves (lowers) perplexity greatly. Although these preliminary results are promising, further studies using the full dataset and larger architectures are needed for a better understanding of how these methods work in comparison to the state-of-the-art baseline models.

6. LIMITATIONS AND FUTURE DIRECTIONS

Due to computational resource constraints, the dataset selected was a small subsample of a industry-standard dataset. This might affect the generalizability of our conclusions. Furthermore, prior work had been conducted on the full PTB dataset, rather than a subsample, so comparison was difficult. Due to the same computational constraints, both the complexity of the models and the number of ensembles could not be increased.

In the future, we plan to increase the size of subsample, the complexity of models, and the size of the ensembles. In their current form, ABIMBS and ABISS both rely on sampling with replacement. We plan to extend this by including a sampling without replacement scheme with the aim of reducing the amount of training examples needed in each ensemble without sacrificing performance. We plan to enhance ABISS's current implementation by adding different options for case weight normalization such as means of interquartile ranges. We also plan to vary the hyperparameters for both SD ABISS and Sqrt ABISS based on performance on the validation set.

7. CONCLUSIONS

Both AdaBoost Inspired Mini-Batch Sampling (ABIBMS) and AdaBoost Inspired Sentence Sampling (ABISS) provide alternate methodologies to train ensembles of LSTMs as language models. These methods rely on morphing the dataset such that different training examples are used unequally based on their importance during training. Based on the results gathered on size 5 ensembles, both versions

of ABIMBS (FABIMBS and BABIMBS) as well as both versions of ABISS (SD ABISS and Sqrt ABISS) provide significant improvements over the standard baseline models currently being employed. These results show that, similar to general machine learning algorithms and general machine learning tasks, state-of-the-art language models can be improved significantly using boosting-inspired approaches.

8. REFERENCES

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin, "A Neural Probabilistic Language Model," *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[2] T Mikolov, M Karafiat, L Burget, J Cernocky, and S Khudanpur, "Recurrent Neural Network based Language Model," *Interspeech*, , no. September, pp. 1045–1048, 2010.

[3] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černocký, "Strategies for training large scale neural network language models," *2011 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2011, Proceedings*, pp. 196–201, 2011.

[4] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio, "How to Construct Deep Recurrent Neural Networks," *arXiv preprint arXiv:1312.6026*, pp. 1–10, 2013.

[5] Tomas Mikolov, "Statistical Language Models Based on Neural Networks," *PhD thesis*, pp. 1–129, 2012.

[6] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 1929–1958, 2014.

[7] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals, "Recurrent Neural Network Regularization," *arXiv:1409.2329 [cs]*, , no. 2013, pp. 1–8, 2014.

[8] Marius Pachitariu and Maneesh Sahani, "Regularization and nonlinearities for neural language models: when are they needed?," *Arxiv*, pp. 1–9, 2013.

[9] Vu Pham, Theodore Bluche, Christopher Kermorvant, and Jerome Louradour, "Dropout Improves Recurrent Neural Networks for Handwriting Recognition," *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, vol. 2014-December, pp. 285–290, 2014.

[10] Y Freund and Re Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," *Computational learning theory*, vol. 55, pp. 119–139, 1995.

[11] H Schwenk and Y Bengio, "Boosting neural networks," *Neural computation*, vol. 12, pp. 1869–1887, 2000.

[12] Yoshua Bengio, Nicolas Le Roux, Pascal Vincent, Olivier Delalleau, and Patrice Marcotte, "Convex neural networks," *Advances in neural information processing systems*, vol. 18, pp. 123, 2006.

[13] Robert E Schapire and Yoav Freund, *Boosting: Foundations and algorithms*, MIT press, 2012.

[14] Mitchell P Marcus, Mary Ann Marcinkiewicz˜, and Beatrice Santorini, "Building a Large Annotated Corpus of English: The Penn Treebank," 1993.

[15] J. Mikolov, T., Deoras, A., Kombrink, S., Burget, L., Černocký, "Empirical Evaluation and Combination of Advanced Language Modeling Techniques ?," *Interspeech*, , no. August, pp. 605–608, 2011.

[16] Ahmad Emami and Frederick Jelinek, "Exact training of a neural syntactic language model," *Icassp-2004*, vol. 1, pp. 0–3, 2004.