VARIANCE REGULARIZATION OF RNNLM FOR SPEECH RECOGNITION

Yongzhe Shi, Wei-Qiang Zhang, Meng Cai and Jia Liu

Tsinghua National Laboratory for Information Science and Technology,
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
{shiyz09, caimeng06}@gmail.com, {wqzhang, liuj}@tsinghua.edu.cn

ABSTRACT

Recurrent neural network language models (RNNLMs) have been proved superior to many other competitive language modeling techniques in terms of perplexity and word error rate. The remaining problem is the great computational complexity of RNNLMs in the output layer, resulting in long time for evaluation. Typically, a class-based RNNLM with the output layer factorized was proposed for speedup, which was still not fast enough for real-time systems. In this paper, a novel variance regularization algorithm is proposed for RNNLMs to address this problem. All the softmax-normalizing factors in the output layers are penalized to make them converge to one during the training phase, so that the output probability can be estimated efficiently via one dot-product of vectors in the output layer. The computational complexity of the output layer is reduced significantly from $O(|\mathbf{V}|H)$ to $O(H)$.We further use this model for rescoring in an advanced CD-HMM-DNN system. Experimental results show that our proposed variance regularization algorithm works quite well, and the word prediction of the model is about 300 times faster than that of RNNLM without any obvious deteriorations in word error rate.

Index Terms— Variance regularization, recurrent neural network language model, speech recognition

1. INTRODUCTION

Recurrent neural network language models (RNNLMs) have been proved to outperform many competitive language modeling techniques in terms of perplexity and word error rate on speech-to-text tasks [1, 2]. Like other neural network language models (NNLMs), the main drawback of RNNLMs is the long training and testing time. The heavy computational burden comes from the output layer that contains tens of thousands of units corresponding to the words in the vocabulary. The output needs to be normalized as probability in the output layer. Many speeding techniques are explored for NNLMs, including GPU-based parallelization, shortlist [3], structured output layer [4, 5, 6, 7], pre-computing [8] and other methods[9, 10]. Generally, most of these techniques can be easily extended to RNNLMs. Typically, a class-based output layer based on word clustering [11, 7] was proposed for speedup. Most of these techniques focus on speeding up the training phase, even though the testing phase is speeded up at the same time. Little attention is focused on the testing phase, while fast evaluation is more critical for recognition. In this work, speeding up the word prediction at the testing phase is investigated for RNNLMs.

This paper introduces a novel variance regularization algorithm for RNNLMs to address this problem. All the softmax-normalizing factors in the output sub-layers are penalized to make them converge to one during the training phase, so that the output probability can be estimated efficiently via one dot-product of vectors in the output layer. The computational complexity for evaluation is reduced significantly, without explicit softmax normalization. There are a large number of local minima in the parameter space of the RNNLM, the variance regularization in this paper means to make the RNNLM converge to a specific local minimum during the training phase.

The remainder of this paper is organized as follows: The class-based RNNLM is firstly reviewed in Section II. Section III presents our proposed variance regularization algorithm for RNNLMs. Experimental evaluation is given in Section IV and V. Section VI concludes this paper and gives our main findings.

2. REVIEW OF CLASS-BASED RNNLM

To speed up the training of the RNNLM, a class-based RNNLM via frequency factorization was proposed in 2011 [11], shown in Fig. 1. In this section, the class-based RNNLM via frequency factorization is firstly reviewed, and then the computational complexity is analyzed. Given a word sequence $\bar{s}$, let the word corresponding to step t be denoted as w_t. The identity of w_t can be denoted as $y_i \in \mathbf{V}$, where the subscript i of y_i is the word index in the vocabulary. The word w_t can be represented by 1-of-V coding vector $\boldsymbol{v}_t$, where all the elements are null except the i-th.

The states of hidden nodes compactly cluster the history and the current input.

$$\boldsymbol{h}_t = \text{sigmoid}(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{ih}\boldsymbol{v}_t), \tag{1}$$

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

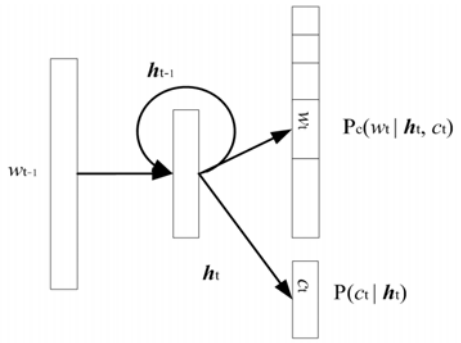

Fig. 1. Class-based RNNLM via frequency-based factorization

where $\boldsymbol{W}_{ih}$ maps each word to its real-valued representation, and $\boldsymbol{W}_{hh}$ denotes the dynamics of sequence in time.

To reduce the complexity, the output layer is divided into a class layer and many sub-layers as the output. Many methods can be used to construct these output layers, including frequency-binning factorization [11], Brown clustering [12], etc. Perhaps, the simplest method is the frequency-binning factorization technique, where words are assigned to classes proportionally. This method divides the cumulative probability of words in a corpus into K partitions to form K frequency-binnings which corresponds to K clusters. It means that there are $K+1$ sub-layers as the output, including the class layer.

Two transformation matrices $\boldsymbol{W}_{hc} \in \Re^{C\times H}$ and $\boldsymbol{W}_{ho} \in \Re^{|V|\times H}$ are defined as $\boldsymbol{W}_{hc} = [\vartheta_1, \vartheta_2, \ldots, \vartheta_C]^T$ and $\boldsymbol{W}_{ho} = [\theta_1, \theta_2, \ldots, \theta_{|V|}]^T$ in the output layer, respectively, where $\vartheta_i \in \Re^{H\times 1}$ or $\theta_j \in \Re^{H\times 1}$ corresponds to each output node. The class probability is computed as

$$\begin{aligned} &\mathrm{P}(c_t = k|\boldsymbol{h}_t) = \frac{\exp(s_t)}{z_{st}}, \\ \text{with} \quad &s_t = \boldsymbol{\vartheta}_k^T \boldsymbol{h}_t \quad \text{and} \quad z_{st} = \sum_{\forall i} \exp(\boldsymbol{\vartheta}_i^T \boldsymbol{h}_t), \end{aligned} \tag{2}$$

where $\exp(s_t)$ and z_{st} correspond to the unnormalized probability and the softmax-normalizing factor in the class layer, respectively. The word probability given the class is estimated similarly as

$$\begin{aligned} &\mathrm{P_c}(w_t = y_j|\boldsymbol{h}_t, c_t) = \frac{\exp(o_t)}{z_{ot}}, \\ \text{with} \quad &o_t = \boldsymbol{\theta}_j^T \boldsymbol{h}_t \quad \text{and} \quad z_{ot} = \sum_{\forall i \in \mathrm{C}(w_t)} \exp(\boldsymbol{\theta}_i^T \boldsymbol{h}_t), \end{aligned} \tag{3}$$

where $\mathrm{C}(\cdot)$ denotes all the nodes belonging to the same cluster.

The probability of the next word w_t is computed as

$$\mathrm{P}(w_t|\boldsymbol{h}_{t-1}, w_{t-1}) = \mathrm{P}(c_t|\boldsymbol{h}_t)\mathrm{P_c}(w_t|\boldsymbol{h}_t, c_t), \tag{4}$$

where c_t denotes the class corresponding to word w_t.The nodes in the same sub-layer, instead of all the nodes in the output layer, need to be normalized via $\mathrm{softmax}$ function.

The class-based RNNLM requires $H\times H + H\times C + H\times O_i$ multiplications for evaluation, where H, C and O_i denote the number of nodes in the hidden layer, the class layer and the i-th sub-layer, respectively. Empirically, O_i ranges from 1 to thousands, depending on the class that the word belongs to. The complexity is reduced for training and testing.

3. VARIANCE REGULARIZATION FOR RNNLM

The computational bottleneck comes from the softmax output layer, even though the output layer is factorized into many sub-layers. It takes long time to compute z_{st} and z_{ot} in the output layer for normalization.

Given the training text $\mathbf{T}$, the normalizing factors, z_{st} and z_{ot}, are introduced in the objective function and penalized during the training phase.

$$\begin{aligned} \tilde{J}(\Theta) = J(\Theta) &+ \frac{\eta}{2} \cdot \frac{1}{|\mathbf{T}|} \sum_{t=1}^{|\mathbf{T}|} (\log(z_{st}))^2 \\ &+ \frac{\lambda}{2} \cdot \frac{1}{|\mathbf{T}|} \sum_{t=1}^{|\mathbf{T}|} (\log(z_{ot}))^2, \end{aligned} \tag{5}$$

where Θ denotes the parameters of the RNNLM, η and λ are the penalties of the log-normalizing factors, and $J(\Theta)$ is the cross-entropy based objective function, presented as

$$J(\Theta) = -\frac{1}{|\mathbf{T}|} \sum_{t=1}^{|\mathbf{T}|} \log(\mathrm{P}(w_t|\boldsymbol{h}_t)). \tag{6}$$

Our goal is to make the RNNLM converge to a specific local minimum in the parameter space, where the normalizing factors in the sub-layers and the class layer are close to one as much as possible. The gradient of $\tilde{J}(\Theta)$ can be efficiently computed as

$$\begin{aligned} \frac{\partial \tilde{J}}{\partial \Theta} = \frac{\partial J}{\partial \Theta} &+ \frac{\eta}{|\mathbf{T}|} \sum_{t=1}^{|\mathbf{T}|} \frac{\partial z_{st}}{\partial \Theta} \frac{\log(z_{st})}{z_{st}} \\ &+ \frac{\lambda}{|\mathbf{T}|} \sum_{t=1}^{|\mathbf{T}|} \frac{\partial z_{ot}}{\partial \Theta} \frac{\log(z_{ot})}{z_{ot}}, \end{aligned} \tag{7}$$

where the partial derivatives of z_{st} are computed as

$$\begin{aligned} \frac{\partial z_{st}}{\partial \boldsymbol{\theta}_j} &= \exp(\boldsymbol{\theta}_j^T \boldsymbol{h}_t)\boldsymbol{h}_t \quad \forall j \in [1, C], \\ \frac{\partial z_{st}}{\partial \boldsymbol{h}_t} &= \exp(\boldsymbol{\theta}_j^T \boldsymbol{h}_t)\boldsymbol{\theta}_j, \end{aligned} \tag{8}$$

and the partial derivatives of z_{st} for the other parameters in Θ can be obtained via chain-rule. The partial derivatives of z_{ot} can be also computed similarly.

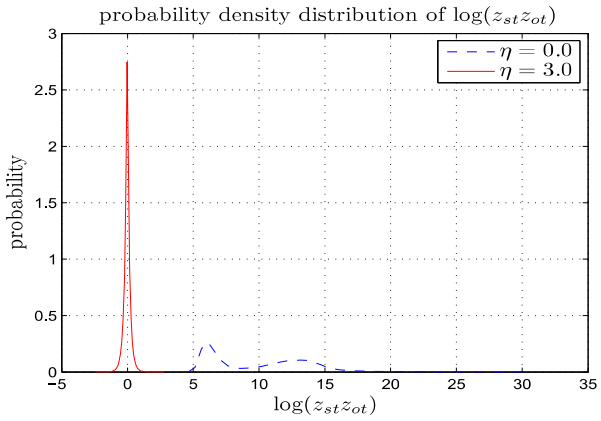

Fig. 2. Probability density distribution of $\log(z_{st}z_{ot})$ on the test set of the PTB corpus, where $\eta = 0.0$ means no variance regularization.

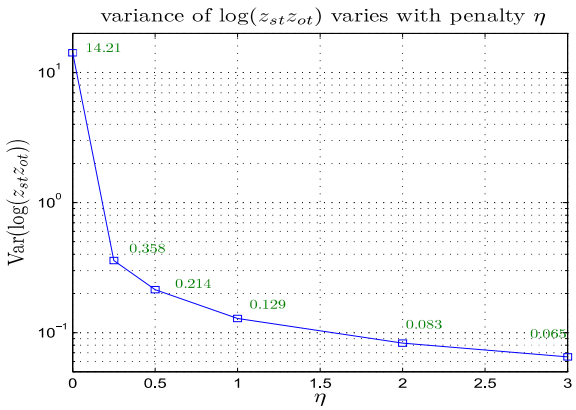

Fig. 3. Variance of $\log(z_{st}z_{ot})$ varies with penalty η on the test set of the PTB corpus, where $\eta = 0.0$ means no variance regularization.

Based on our proposed variance regularization algorithm, the log-probability of the next word can be simplified as

$$\begin{aligned} \log(\mathrm{P}(w_t = y_j|\boldsymbol{h}_{t-1}, w_{t-1})) &\approx (\vartheta_k + \theta_j)^T \boldsymbol{h}_t, \\ s.t. \quad \log(z_{st}z_{ot}) &\approx 0, \end{aligned} \tag{9}$$

where the subscript k of ϑ_k denotes the index of the class that the word w_t belongs to. The log-probability of the next word can be approximately estimated via one summation and one dot-product of vectors in the output layer, where the computational complexity is reduced significantly. Note that the accuracy of Eq. (9) depends on how close to one the normalizing factors are in the statistical sense.

Two open questions are considered for our proposed variance regularization algorithm. One is how close to zero $\log(z_{st}z_{ot})$ is in the statistical sense, and the other one is whether the model performance is degraded or not under our proposed constraint. The both questions will be answered based on experimental evaluations in the following section.

4. PERPLEXITY EVALUATION

One of the most widely used data sets for evaluating the performance of statistical language models is the Penn Treebank portion of the Wall Street Journal Corpus, denoted as PTB corpus. The PTB corpus is preprocessed by lowercasing words, removing punctuation and replacing numbers with the "N" symbol. Sections 00-20 are used as training sets (930K words), sections 21-22 as validation sets (74K words), and sections 23-24 as test sets (82K words). The vocabulary size is 10K. In this section, the PTB corpus is used to evaluate our proposed algorithm.

An RNNLM model with 200 hidden nodes is trained using rnnlm toolkit [13], where the 100 classes are used for speedup. Another model with the same setup is also trained with $\eta = \lambda = 3.0$ for variance regularization. If there are

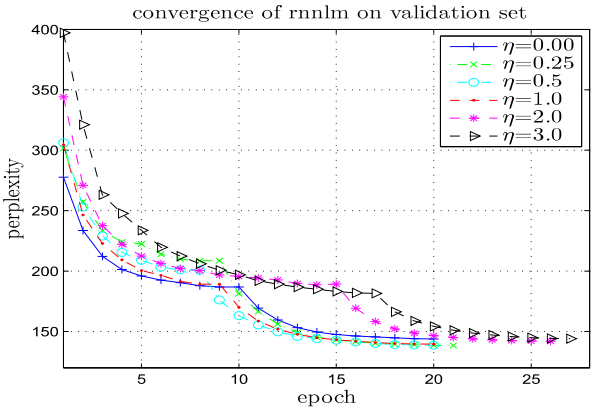

Fig. 4. Perplexity convergence of RNNLM on the validation set, where $\eta = 0.00$ means no variance regularization.

no specific instructions, η is set equally to λ for convenience. The two models are evaluated on the test set of the PTB corpus, where the normalizing factor $z_{st}z_{ot}$ in the logarithmic domain is computed at each time step. The distribution of the normalizing factor in the logarithmic domain is shown in Fig. 2 for comparisons. The normalizing factor of the RNNLM ($\eta = 0.0$) ranges from 5 to 20 in the logarithmic domain. On the contrary, the normalizing factor of the other model with $\eta = 3.0$ shrinks sharply.

Several RNNLM models with different η are also trained for comparisons. All the models are evaluated on the test set of the PTB corpus, and the variance of $\log(z_{st}z_{ot})$ is computed and shown in Fig. 3. It is straightforward to see that the variance of $\log(z_{st}z_{ot})$ decreases with the increase of η. The smaller the variance, the more accurate the Eq. (9).

Finally, the perplexities of these models on the validation set during the training phase are shown in Fig. 4, where the model with large η requires more epochs to converge com-

pletely. It is clear that our proposed variance regularization algorithm doesn't degrade the model performance.

5. SPEECH RECOGNITION EXPERIMENTS

The effectiveness of our proposed variance regularization algorithm is evaluated on the STT task with the 309-hour Switchboard-I training set [14]. The data for system development is the 1831-segment SWB part of the NIST 2000 Hub5 eval set (Hub5'00-SWB). The FSH half of the 6.3h Spring 2003 NIST rich transcription set (RT03S-FSH) acts as the evaluation set. A well-tuned CD-DNN-HMM system [15, 16] is used in the STT task. The input to the DNN contains 11 (5-1-5) frames of 39-dimensional PLP features, and the DNN uses the architecture of 429-2048×7-9308. A back-off trigram (KN3) was trained via Kneser-Ney smoothing on the 2000h Fisher transcripts, containing 23 million tokens, for decoding, where the vocabulary is limited to 53K words and unknown words are mapped into a special token <unk>. A back-off 5-gram (KN5) was also trained similarly as KN3 for rescoring. Note that no other unknown text is used to train LMs for interpolations, so that the following experimental results are easily repeatable. The pronouncing dictionary comes from CMU [17].

An RNNLM with 300 hidden nodes and 400 classes is trained on the transcripts. The truncated backpropagation through time algorithm (BPTT) is used for training the RNNLM with 10 time steps. The learning rate is initially set to 0.1 and halved when the perplexity decreases very slowly or increases. Another RNNLM with variance regularization η=λ=2.0 is also trained in the same setup for comparisons. For convenience, 100-best hypotheses are generated from the well-tuned STT system and rescored by KN5 and RNNLMs. The weight for interpolation, scale of LM scores, and word penalty are all tuned on the Hub5'00-SWB set and the performance of each LM is evaluated on RT03S-FSH set.

These hypotheses are first rescored with exact probability of RNNLM as our baseline, shown in Table 1, where the absolute reduction in WER is also shown for comparisons. The RNNLM reduces the WER by 1.8% and 1.7% on Hub5'00-SWB and RT03S-FSH sets, respectively. Larger WER reductions are obtained through the interpolation with KN5. Then, the same rescoring experiment is performed with UP-RNNLM-VR for comparisons, where UP-RNNLM-VR denotes the unnormalized probability of RNNLM with variance regularization. The similar WER reductions are obtained for UP-RNNLM-VR, shown in Table 1. Experimental results show that our proposed variance regularization doesn't degrade the performance of RNNLM.

The complexity is analyzed and shown in Table 2 for comparisons. The testing speed is measured by the number of words processed per second on a machine with an Intel(R) Xeon(R) 8-core CPU E5520@2.27GHz and 8G RAM, shown in Table 2. The word prediction of UP-RNNLM-C400-VR is about three times faster than that of RNNLM-C400 or 300 times as fast as that of RNNLM. The word prediction at the testing phase is speeded up significantly based on variance regularization. Experimental results show that our proposed variance regularization algorithm works quite well for fast word predictions.

Table 1. Word error rates (WERs) of Hub5'00-SWB and RT03S-FSH for 100-best rescoring using different RNNLMs and back-off 5-gram (KN5).

Model	WER % (absolute change)	
	Hub5'00-SWB	RT03S-FSH
One-best	17.3	20.2
KN5	17.1	19.5
RNNLM	15.5 (-1.8)	18.5 (-1.7)
+ KN5	**15.3** (-2.0)	**18.1** (-2.1)
UP-RNNLM-VR	15.5 (-1.8)	18.7 (-1.5)
+ KN5	**15.2** (-2.1)	**18.1** (-2.1)

Table 2. Complexity and speed comparisons of RNNLM, RNNLM-C400 and UP-RNNLM-VR at the recognition stage.

Model	Complexity	Speed $\times 10^3$ (Words / Sec.)
RNNLM	$O(H^2 + \|\mathbf{V}\|H)$	0.041
+Class Layer	$O(H^2 + (\|\mathbf{V}\|/C + C)H)$	4.21
UP-RNNLM-VR	$O(H^2 + H)$	11.95

6. CONCLUSION

A novel variance regularization algorithm is proposed for the class-based RNNLM. The normalizing factors in the sub-layers and the class layer are constrained to one during the training, so that the probability of the next word can be efficiently estimated with one summation and one dot-product of vectors in the output layer for evaluation. The computational complexity is reduced significantly. Additionally, the work introduced in this paper is so general-purpose that can be easily extended to other neural network or multi-task classifiers. Also, this method can be easily extended into feed-forward NNLM and a larger speedup factor can be expected since the $\boldsymbol{h}_t$ in hidden layer can efficiently computed via lookup in table. Finally, our proposed model has the potential to be incorporated into the first decoding pass of STT system, which we are currently investigating.

7. ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China under Grant No. 61370034, No. 61273268 and No. 61005019.

8. REFERENCES

[1] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Honza Cernocky, "Empirical evaluation and combination of advanced language modeling techniques," in *Proc. of InterSpeech*, 2011.

[2] Tomas Mikolov, *Statistical Language Models Based on Neural Netowrks*, Ph.D. thesis, Brno University of Technology (BUT), 2012, [Online] http://www.fit.vutbr.cz/ imikolov/rnnlm/thesis.pdf.

[3] Holger Schwenk and Jean-Luc Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *Proc. of ICASSP*, 2002, pp. 765–768.

[4] Frederic Morin and Yoshua Bengio, "Hierarchical probabilistic neural network language model," in *Proc. of AISTATS*, 2005, pp. 246–252.

[5] Andriy Mnih and Geoffrey Hinton, "A scalable hierarchical distributed language model," *Advances in Neural Information Processing Systems*, vol. 21, 2008.

[6] Hai Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and Francois Yvon, "Structured output layer neural network language model," in *Proc. of ICASSP*, 2011, pp. 5524–5527.

[7] Yongzhe Shi, Wei-Qiang Zhang, Jia Liu, and Michael T. Johnson, "RNN language model with word clustering and class-based output layer," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 22, 2013.

[8] F. Zamora-Martinez, M. J. Castro-Bleda, and S. Espana-Boquera, "Fast evaluation of connectionist language models," in *Proc. of IWANN '09*, 2009, pp. 33–40.

[9] Andriy Mnih and Yee Whye Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proc. of ICML*, 2012.

[10] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. of ICASSP*, 2013.

[11] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Honza Cernocky, and Sanjeev Khudanpur, "Extensions of recurrent neural network language model," in *Proc. of ICASSP*, 2011.

[12] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai, "Class-based n-gram models for natural language," *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, 1992.

[13] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Honza Cernocky, "RNNLM - recurrent neural network language modeling toolkit," in *Proc. of ASRU*, 2011, [Available] http://www.fit.vutbr.cz/ imikolov/rnnlm/.

[14] J. Godfrey and E. Holliman, "Switchboard-1 release 2," Linguistic Data Consortium, Philadelphia, 1997.

[15] Meng Cai, Yongzhe Shi, and Jia Liu, "Deep maxout neural networks for speech recognition," in *Proc. of ASRU*, 2013.

[16] Frank Seide, Gang Li, Xie Chen, and Dong Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. of ASRU*, 2011.

[17] "The CMU Pronouncing Dictionary Release 0.7a," 2007, [Available] http://www.speech.cs.cmu.edu/cgi-bin/cmudict.