RECURRENT NEURAL NETWORK LANGUAGE MODEL TRAINING USING NATURAL GRADIENT

Jianwei Yu[⋆]*, Max. W. Y. Lam*[⋆]*, Xie Chen*[†]*, Shoukang Hu*[⋆]*, Songxiang Liu*[⋆]*, Xixin Wu*[⋆]*, Xunying Liu*[⋆]*, Helen Meng*[⋆]

[⋆]The Chinese University of Hong Kong, Hong Kong SAR, China
[†]Microsoft AI and Research, One Microsoft Way, Redmond, WA, USA
{jwyu, wylam, skhu, sxliu, wuxx, xyliu, hmmeng}@se.cuhk.edu.hk, xieche@microsoft.com

ABSTRACT

Recurrent neural network language models (RNNLMs) have become an increasing popular choice for state-of-the-art speech recognition systems. RNNLMs are normally trained by minimizing the cross entropy (CE) using the stochastic gradient descent (SGD) algorithm. However, the SGD method doesn't consider the correlation between parameters and therefore can lead to unstable and slow convergence in training. Second-order optimization methods provide a possible solution to this issue. However these methods are either computationally heavy or do not have competitive performance. In this paper, a novel optimization method – stochastic natural gradient based on minimum variance assumption (SNGM) is proposed for training RNNLMs. It allows the natural gradient method to operate at a comparable training efficiency to the SGD method. By modifying the gradient according to the local curvature of the KL-divergence between current and updated probabilistic distributions, the proposed SNGM approach is shown to outperform both the SGD and limited memory BFGS methods across three tasks: Penn Treebank, Switchboard conversational speech recognition and AMI meeting room transcription in terms of both perplexity and word error rate.

Index Terms— RNNLMs, Natural Gradient

1. INTRODUCTION

Language models (LMs) which estimate the probability of any given word sequence in a language are essential to many applications such as speech recognition. A variety of statistical language models have been proposed, such as n-gram [1] LMs and neural network LMs [2, 3, 4, 5]. In recent years, RNNLMs and long-short term of memory (LSTM) variants [6, 7, 8] have been shown to yeild state-of-the-art language modeling performance on a wide range of tasks and give significant improvements over n-gram LMs.

For the training of RNNLMs, cross entropy is normally used as the objective function and stochastic gradient descent (SGD) algorithm [9] used for optimization. In common with other gradient descent based techniques, the SGD method only considers about the first-order derivatives. No higher order gradient information [10, 11, 12, 13, 14] is used to consider the correlation between parameters and therefore can not fully capture the curvature of the objective function. This can lead to unstable and slow convergence in the training. Newton's method is one of the most renowned approaches to address this issue by using quadratic approximation of the objective function. However, for tasks with a large number of parameters such as language modeling, computing the Hessian matrix and its inverse required by Newton's method is problematic.

To address this issue, truncated Newton's methods based on Hessian-free [13] (HF) optimization have been proposed and successfully applied to speech recognition tasks [14]. Instead of directly computing the Hessian matrix and its inverse, the product between the inverse Hessian and the gradient is approximated using an iterative Conjugate Gradient (CG) algorithm. However, due to additional modified forwarded passes performed during CG search, the HF method is highly expensive [13, 15, 16] and can converge to a sub-optimal solution. Alternatively, the 2nd order update direction can also be approximated via a recursion over a limited number of past gradients based on the limited-memory Broyden Fletcher Goldfarb Shannon (L-BFGS) [11, 12, 17] algorithm. Using the KL-divergence between the current and updated probabilistic distributions over state-level sequence from an information theory perspective, a natural gradient [18] based DNN acoustic model training method was proposed in [16]. In the NG method, gradient descent is done on the space of densities $p_{\boldsymbol{\theta}}(\boldsymbol{w})$. The update direction is obtained using the product of inverse of the empirical Fisher Information (FI) matrix where the gradient estimates are derived by CG.

In this paper, we propose a novel optimization algorithm, namely stochastic natural gradient based on minimum variance assumption (SNGM) for training RNNLMs. Using this method, the product of the inverse FI matrix and the gradient can be approximated as the gradient multiplied with a dynamically adjusted scaling factor. Compared with the NG based method in [16], our method does not require expensive CG search and is as efficient as standard SGD training scheme. The proposed approach is shown to outperform both the SGD and L-BFGS based second-order optimization methods across three data sets: Penn Treebank, Switchboard conversational speech recognition and AMI meeting room transcription in terms of both perplexity and word error rate.

The rest of the paper is organized as follows. Section 2 and Section 3 briefly review the RNNLMs architecture and SGD based training algorithm. Section 4 presents the L-BFGS based second order optimization scheme for the RNNLMs. Section 5 introduces the SNGM method. Experimental results are presented in seciton 6. Section 7 is the conclusion and future work.

[0] This research was partially funded by Research Grants Council of Hong Kong General Research grant Fund No.14200218, and the Chinese University of Hong Kong (CUHK) grant No. 4055065

2. RECURRENT NEURAL NETWORK LANGUAGE MODELS

Language models (LMs) assign a probability to any given sentence of n words $\boldsymbol{W} = (\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_n})$. According to Bayesian rule, it can be decomposed as the product of the individual word probabilities given its word history:

$$P(\boldsymbol{W}) = P(\boldsymbol{w_1}, ..., \boldsymbol{w_n}) = \prod_{t=1}^{n} P(\boldsymbol{w_t}|\boldsymbol{w_{t-1}}, ..., \boldsymbol{w_1}). \quad (1)$$

In recurrent neural network language models (RNNLMs), the word probability can be written as:

$$P(\boldsymbol{w}_t|\boldsymbol{w}_{t-1}, ..., \boldsymbol{w_1}) \approx P(\boldsymbol{w}_t|\boldsymbol{w}_{t-1}, \boldsymbol{h_{t-2}}) = P(\boldsymbol{w}_t|\boldsymbol{h_{t-1}}), \quad (2)$$

where $h_{t-1} \in R^M$ is the hidden vector that represents the previous history $(\boldsymbol{w_{t-1}}, ..., \boldsymbol{w_1})$. The RNNLM can be generally divided into three parts: the projection layer, the recurrent layer and the output layer. The projection layer projects the one-hot word vector $\boldsymbol{w_t} \in R^N$ into a continuous space $\chi \subseteq R^M$ as $\boldsymbol{x_t}$, where N is vocabulary size and usually $M \ll N$. Followed by the projection layer, recurrent layer computes the hidden vector by recursively applying a gating function:

$$\boldsymbol{h_{t-1}} = g(\boldsymbol{x_{t-1}}, \boldsymbol{h_{t-2}}), \quad (3)$$

which is normally based on sigmoid activations in standard RNNLMs. In order to address vanishing gradient issue associated with conventional RNNLM training, long short-term memory (LSTM) [19] RNNLMs can be used. For both conventional RNNLMs and LSTM-RNNLMs, the output layer uses the hidden vector h_{t-1} to compute the word probabilities via a softmax activation:

$$P(\boldsymbol{w}_t|\boldsymbol{h_{t-1}}) = \frac{\exp((\mathbf{V}h_{t-1})^{\text{indx}(\boldsymbol{w_t})})}{\sum_{n=1}^{N}\exp((\mathbf{V}h_{t-1})^n)}, \quad (4)$$

where $\mathbf{V}$ is the the projection matrix of the output layer that project the hidden vector back to vocabulary space, $(\mathbf{V}h_{t-1})^n$ denotes the n-th elements of $(\mathbf{V}h_{t-1})$ and $\text{indx}(\boldsymbol{w_t})$ is the index of word $\boldsymbol{w_t}$.

3. RNNLM TRAINING USING SGD

Conventional RNNLM training minimizes the cross entropy (CE) of the training data. For a given sequence containing a total of N_w words, the CE objective function is given by

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{N_w}\sum_{t=1}^{N_w}\log P(\boldsymbol{w}_t|\boldsymbol{h_{t-1}}). \quad (5)$$

The stochastic gradient descent (SGD) algorithm is normally used in CE training. For the $(k+1)$-th randomly selected minibatch of N_w words within each training epoch, the gradient statistics accumulated over the minibatch are scaled by a tunable learning rate α before being used to update the model parameters

$$\boldsymbol{\theta_{k+1}} = \boldsymbol{\theta_k} - \alpha\nabla\mathcal{L}(\boldsymbol{\theta}). \quad (6)$$

The gradient of RNNLMs can be acquired using an extended form of the standard back-propagation algorithm, back-propagation through time (BPTT) [20]. The SGD method only considers the first-order derivatives. No higher order gradient information is used to consider the correlation between parameters. Therefore this method can not fully capture the curvature of the objective function. This can lead to unstable and slow convergence in the training.

4. RNNLMS TRAINING WITH SECOND ORDER OPTIMIZATION

In this section we briefly review the Newton's method and Quasi-Newton methods. Considering about the drawbacks of BFGS method, we apply the L-BFGS algorithm for RNNLM training.

4.1. Newton's Method

In contrast to SGD, Newton's method explicitly considers the second-order gradient information to capture the correlations between model parameters. By taking a quadratic approximation to the objective function, the objective function $\mathcal{L}(\boldsymbol{\theta})$ can be locally approximated using a second-order Taylor series expansion:

$$\mathcal{L}(\boldsymbol{\theta_k} + \Delta\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}_k) + \Delta\boldsymbol{\theta}^T\nabla\mathcal{L}(\boldsymbol{\theta}_k) + \frac{1}{2}\Delta\boldsymbol{\theta}^T\boldsymbol{H}\Delta\boldsymbol{\theta}, \quad (7)$$

where $\boldsymbol{H}$ is the Hessian matrix. Instead of directly minimizing $\mathcal{L}(\boldsymbol{\theta})$, Newton's method aims to minimize the above quadratic approximation of equation (7). Setting its gradient with respect to $\Delta\boldsymbol{\theta}$ to zero leads to the Newton update direction:

$$\Delta\boldsymbol{\theta} = -\boldsymbol{H}^{-1}\nabla\mathcal{L}(\boldsymbol{\theta_k}). \quad (8)$$

4.2. Quasi-Newton Methods

For tasks with a large number of model parameters, directly using Newton's method is problematic. The calculation of the Hessian matrix with $\boldsymbol{O}(N^2)$ parameters and its inversion of $\boldsymbol{O}(N^3)$ complexity are both computationally expensive. Furthermore, the Hessian matrix is not guaranteed to be positive-definite. In this case, the resulting Newton direction may not lead to the desired minimum, but rather towards an opposite direction or a saddle point.

To address these issues, two types of techniques that do not require an explicit direct calculation of Hessian and its inverse can be used. The first type is based on Hessian free optimization [13, 15] . Instead of directly computing the Hessian matrix, an iterative Conjugate Gradient (CG) algorithm is used to calculate the update direction. As the CG search requires additional modified forward passes [13, 15, 21] to be performed, the Hessian Free optimization algorithm is therefore expensive to use in practice [14, 16]. The second category of techniques are based on Quasi-Newton methods. These techniques approximate the inverse Hessian matrix by recursively analyzing the past gradient vectors. An early form of these methods was based on the Davidon Flecher Powell (DFP) algorithm, before being further developed into the widely used Broyden Fletcher Goldfarb Shannon (BFGS) algorithm.

4.3. Limited Memory BFGS Method

The standard BFGS algorithm requires a full matrix approximation to the inverse of the Hessian with $\boldsymbol{O}(N^2)$ parameters. This is computationally expensive for tasks with a large amount of model parameters. To address this issue, a low memory extension to the standard BFGS algorithm, limited memory BFGS (L-BFGS) method can be used. In contrast to the standard BFGS algorithm that approximates the inverse Hessian directly via a recursion over past gradients, the L-BFGS method approximates the matix-vector product between the inverse Hessian and gradient vector in equation (8). Only a few vectors representing a history of the past m updates of such matrix-vector product need to be stored. The gradient history size m can be set small as $m < 10$. In this paper, $m = 5$ is set to 5 throughout the experiments.

The L-BFGS algorithm requires an initial approximation of the Hessian before iteratively computes the product between Hessian inverse and the gradient vector. Following the work in [22], the initial Hessian approximate is set to be an identity matrix. The pseudo-code of the L-BFGS algorithm is shown in algorithm 1.

Algorithm 1 L-BFGS algorithm for RNNLMs

$\boldsymbol{g_k} \leftarrow \nabla\mathcal{L}(\boldsymbol{\theta_k})$, $\boldsymbol{q} \leftarrow \boldsymbol{g_k}$
for $i = k-1, k-2, ..., k-m$ **do**
  $\boldsymbol{s_i} \leftarrow \boldsymbol{\theta_{i+1}} - \boldsymbol{\theta_i}$, $\boldsymbol{y_i} \leftarrow \boldsymbol{g_{i+1}} - \boldsymbol{g_i}$
  $\rho_i \leftarrow \frac{1}{\boldsymbol{y_i^T s_i}}$, $\alpha_i \leftarrow \rho_i \boldsymbol{s_i^T g_k}$
  $\boldsymbol{q} \leftarrow \boldsymbol{q} - \boldsymbol{\alpha_i y_i}$
end for
$\boldsymbol{B_k^0} = I$, $\boldsymbol{z} \leftarrow \boldsymbol{B_k^0 q}$
for $i = k-m, k-m+1, ..., k-1$ **do**
  $\beta_i = \rho_i \boldsymbol{y_i^T z}$
  $\boldsymbol{z} = \boldsymbol{z} + \boldsymbol{s_i}(\alpha_i - \beta_i)$
end for
$\boldsymbol{H_k^{-1} g_k} \leftarrow \boldsymbol{z}$

5. STOCHASTIC NATURAL GRADIENT OPTIMIZATION

The natural gradient method was first proposed by Amari [18] as an effective method for training parametric density models. This method established a geometry on the Riemannian space of density functions inside which steepest descent is performed. Amari showed [23] that by inherently doing optimization on the Riemannian space of density models, the natural gradient method can be more effective than standard gradient descent. This method has attracted increasing interest in recent years for training DNNs [16, 24, 25, 26]. This work extends the application of NG method into RNNLMs.

5.1. Fisher Information Matrix for RNNLMs

Let F be the family of densities $p_{\boldsymbol{\theta}}(\boldsymbol{w_t}|\boldsymbol{h_{t-1}})$ that can be captured by different setting of $\boldsymbol{\theta}$. For simplicity in the rest of the paper we use $p_{\boldsymbol{\theta}}(\boldsymbol{w_t})$ to denote the $p_{\boldsymbol{\theta}}(\boldsymbol{w_t}|\boldsymbol{h_{t-1}})$. When one adds a small quantity $\Delta\boldsymbol{\theta}$, the changes of the probability distributions can be measured using the KL-divergence between the current and updated distributions. Using a Taylor expansion, within a convex neighbourhood of a given distribution $\boldsymbol{\theta}(\boldsymbol{w_t})$, the KL-divergence can be approximated as follows:

$$\begin{aligned}\mathrm{KL}(p_{\boldsymbol{\theta}}(\boldsymbol{w_t})||p_{\boldsymbol{\theta}+\boldsymbol{\Delta\theta}}(\boldsymbol{w_t})) \approx -\Delta\boldsymbol{\theta}^T \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{w_t})}[\nabla \log(p_{\boldsymbol{\theta}}(\boldsymbol{w_t})] \\ -\frac{1}{2}\Delta\boldsymbol{\theta}^T \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{w_t})}[\nabla^2 \log(p_{\boldsymbol{\theta}}(\boldsymbol{w_t})]\Delta\boldsymbol{\theta},\end{aligned} \tag{9}$$

where $\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{w_t})}[\nabla \log(p_{\boldsymbol{\theta}}(\boldsymbol{w_t})]$ in the first term of the quadratic approximation of the KL-divergence can be shown equal to zero because each candidate $p_{\boldsymbol{\theta}}(\boldsymbol{w_t})$ is a valid probability distribution. Thus the KL divergence can be locally approximated by

$$\begin{aligned}&\mathrm{KL}(p_{\boldsymbol{\theta}}(\boldsymbol{w_t})||p_{\boldsymbol{\theta}+\boldsymbol{\Delta\theta}}(\boldsymbol{w_t})) \approx \\ &-\frac{1}{2}\Delta\boldsymbol{\theta}^T \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{w_t})}[\nabla^2 \log(p_{\boldsymbol{\theta}}(\boldsymbol{w_t})]\Delta\boldsymbol{\theta}.\end{aligned} \tag{10}$$

The Fisher Information (FI) matrix is defined as the expected outer product of the likelihood score,

$$\boldsymbol{F} = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{w_t})}[\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{w_t}) \nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{w_t})^T]. \tag{11}$$

Under the condition where equation (10) holds, the FI matrix can be proved to be equal to the negated Hessian matrix w.r.t the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{w_t})$. Thus, we can now locally approximate the KL-divergence by

$$\mathrm{KL}(p_{\boldsymbol{\theta}}(\boldsymbol{w_t})||p_{\boldsymbol{\theta}+\boldsymbol{\Delta\theta}}(\boldsymbol{w_t})) \approx \frac{1}{2}\Delta\boldsymbol{\theta}^T \boldsymbol{F} \Delta\boldsymbol{\theta}. \tag{12}$$

For any given $\boldsymbol{\theta}$, Equation (12) allows us to define a notion of local 'distance' measure in the Riemannian parameter space of joint distributions $p_{\boldsymbol{\theta}}(\boldsymbol{w_t})$ using the FI matrix. Thus the FI matrix can be regarded as the Riemannian metric tensor of the space of the probability density functions. Amari proved that the steepest descent direction of $\mathcal{L}(\theta)$ in Riemannian space is [18]

$$\Delta\boldsymbol{\theta} = -\boldsymbol{F^{-1}} \nabla\mathcal{L}(\boldsymbol{\theta}). \tag{13}$$

5.2. Stochastic NG under minimum variance assumption

In this work, we randomly divide the training data set into several small minibatches in the same fashion as SGD. Each minibatch can be viewed as a local representation of the whole training set. We estimate the Fisher Information matrix by the data samples only in current minibatch and update the model parameters using the natural gradient method (SNG). Following the greedy strategy, this approach uses natural gradient to get the 'steepest' update direction instead of SGD direction within each minibatch.For tasks having a large amount of parameters, directly computing the update direction based on natural gradient method is expensive when explicitly computing and inverting the FI matrix. In [16] this issue is addressed by using a CG search scheme to iteratively approximate the product between the FI matrix inverse and the gradient. However, in practice, CG algorithm is time consuming and not always stable especially when the number of iterations is limited. In order to improve efficiency, a minimum variance assumption is made over the gradients obtained within each minibatch. The resulting algorithm, stochastic natural gradient under a minimum variance assumption (SNGM), allows the FI matrix to be approximate as

$$\boldsymbol{F} \approx \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{w_t})}[\nabla \log(p_{\boldsymbol{\theta}}(\boldsymbol{w_t})] \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{w_t})}[\nabla \log(p_{\boldsymbol{\theta}}(\boldsymbol{w_t})]^T \tag{14}$$

For RNNLMs with CE objective function, the expection of the gradients within a minibatch is exactly the update gradient $\nabla\mathcal{L}(\boldsymbol{\theta})$ provide by SGD. Therefore, the update direction can be written as

$$\Delta\boldsymbol{\theta} = -(\boldsymbol{g}\boldsymbol{g}^T)^{-1}\boldsymbol{g} \tag{15}$$

As $\boldsymbol{g}\boldsymbol{g}^T$ is semi-positive definite, it is easy to find out that the update direction is $\Delta\boldsymbol{\theta} = -\frac{\boldsymbol{g}}{\boldsymbol{g}^T\boldsymbol{g}}$. The inner product of vector is only $\boldsymbol{O}(N)$ complexity. Therefore, the SNGM method is as fast as SGD method. Compared with SGD, our method provides a scaling factor appied to the gradient w.r.t each minibatch to dynamically adjust the step size. We should carefully state that the minimum variance assumption may not always hold, but the practical experiment shows that SNGM method is effective. Table 1 shows the validation PPL of LSTM language models using the SNGM and SGD methods with different minibatch sizes varying from 5 to 200 on 3M words **swbd** task. The learning rate is set to be 0.5 in all these experiments. In contrast to the large performance degradation of the SGD LSTM LM, the performance of SGNM trained LSTM LMs were found to be very robust when using a wide range of minibatch sizes.

Table 1: Validation perplexities of SGD and SNGM trained LSTM LMs with varying minibatch sizes on 3M word Switchboard corpus

Batchsize	5	10	50	100	200
SGD	90.1	90.7	100.9	114.5	121.4
SNGM	87.7	88.9	88.1	88.2	89.0

6. EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of L-BFGS and SNGM algorithms on sigmoid RNN and LSTM LMs using the perplexity (PPL) measure and the word error rates (WERs) obtained in automatic speech recognition (ASR) tasks. The language model was built on the Python GPU computing library PyTorch [27], while the

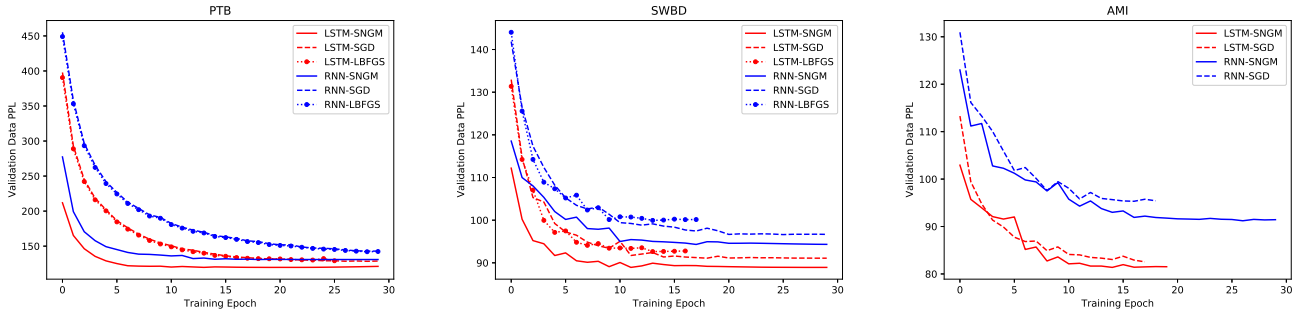

Fig. 1: Validation data PPL changes of the language models in the training procedure on Penn Treebank, Switchboard and AMI corpora using SGD, L-BFGS and SNGM methods.

optimization algorithm is developed by ourselves. In all our experiment, we used 200 hidden nodes in the sigmoid RNN and LSTM language models. The batch size and the learning rate are set to be 10 and 0.5 respectively for all these experiments. And we applied the same newbob scheduling strategy to all the optimization algorithm during training. The ASR systems were constructed using the HTK toolkit version 3.5 [28]. In state-of-the-art ASR systems, RNNLMs are often linearly interpolated within n-gram LMs to obtain better generalization ability [29, 7]:

$$P(\boldsymbol{w_t}|\boldsymbol{w_1^{t-1}}) = \lambda P_{\text{ngram}}(\boldsymbol{w_t}|\boldsymbol{w_1^{t-1}}) + (1-\lambda)P_{\text{RNN}}(\boldsymbol{w_t}|\boldsymbol{w_1^{t-1}}), \tag{16}$$

where λ is the weight assign to the back-off n-gram LM, and is kept fixed as 0.5 in all experiments in this paper.

6.1. Experiments on Penn Treebank Corpus

We first analyze the the performances of SGD, L-BFGS and SNGM algorithms using the Penn TreeBank (PTB) corpus [30], which consists of 10k vocabulary, 930k words for training, 74k words for development and 82k words for testing. The left picture in Figure 1 shows that the L-BFGS and SGD algorithm give very similar performances, while the SNGM method outperforms them on both sigmoid RNN and LSTM language model with lower validation data PPL and faster convergence.

6.2. Experiments on Conversational Telephone Speech

The Switchboard English system has 300 hour of conversational telephone speech from Switchboard I for acoustic modeling and 3.6M words of acoustic transcription with 30k words lexicon for language modeling. The acoustic model is a minimum phone error (MPE) trained stacked hybrid DNN-HMM acoustic model [17], which contains 6 sigmoid activation based hidden layers of 2000 nodes, except the 2nd last bottleneck (BN) layer contains 39 nodes used to produce BN features.The PPL on the validation data during training is shown in the middle part of Figure 1. The PPL and the WER results on Switchboard (swbd) and CallHome (callhm) test data can be found in Table 2. It can been seen that the SNGM method provides competitive performance on this task, while the L-BFGS is worse than SGD. Compared with previous work [17] on L-BFGS RNNLM, here we used a stronger SGD baseline.

6.3. Experiment on AMI Meeting Transcription Task

The previous experiments show that the SNGM method is more efficient than the other two methods. Because of computation limitation, we evaluate only the performance of SGD, and SNGM on the highly challenging meeting transcription task using the 59 hour Augmented Multi-party Interaction (AMI) corpus [31]. We used a mixture of text corpora with 26M words (AMI, Fisher 1/2 and SWBD) and 41k words in vocabulary to train the language model. For acoustic modeling, we trained a Tandem system and a Hybrid system separately and then combined them for better performance using joint decoding [32]. All systems were based on state-clustered decision-tree triphone models with 6000 nodes. The right part of Figure 1 shows the validation data PPL changes during training. The PPL and WER performances are shown in Table 3.The AMI experiment result indicates that the SNGM method outperforms SGD on PPL measure and sigmoid RNN language model based WERs.

Table 2: Validation data perplexities (PPL) and word error rates (WERs) on **swbd** and **callhm** test data

LMs	PPL		WER(%)			
			swbd		**callhm**	
	-	+ 4g	-	+ 4g	-	+ 4g
4-gram	80.65	-	12.1	-	23.9	-
RNN-SGD	96.28	75.00	11.7	11.5	24.1	23.4
RNN-LBFGS	99.66	76.38	11.9	11.6	24.2	23.6
RNN-SNGM	**93.96**	**73.93**	**11.6**	**11.3**	**23.8**	**23.3**
LSTM-SGD	90.74	71.95	11.4	11.3	**23.3**	23.1
LSTM-LBFGS	92.41	72.98	11.4	11.2	23.9	23.3
LSTM-SNGM	**88.93**	**70.56**	**11.3**	**11.1**	23.5	**23.1**

Table 3: Validation data perplexities (PPL) and word error rates (WERs) on AMI test data

LMs	PPL		WER(%)			
			dev		**eval**	
	-	+ 4g	-	+ 4g	-	+ 4g
4-gram	111.30	-	30.4	-	31.0	-
RNN-SGD	95.39	84.59	29.8	29.4	30.5	30.0
RNN-SNGM	**91.21**	**82.37**	**29.7**	**29.1**	**30.2**	**29.9**
LSTM-SGD	82.55	76.61	29.4	**29.0**	**29.6**	29.5
LSTM-SNGM	**81.38**	**75.34**	29.4	29.2	29.7	29.5

7. CONCLUSION

In this paper, an effective stochastic natural gradient method under minimum variant assumption (SNGM) is proposed for training RNNLMs. Experiment results on three datasets suggest the proposed technique is useful to improve the performance for RNNLMs. To the best of our knowledge, this is the first work using NG based optimization for RNNLMs. Future work will focus on the comparison of the SNGM method with CG based NG method on RNNLMs.

8. REFERENCES

[1] Ronald Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, 2000.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[3] Holger Schwenk, "Continuous space language models," *Computer Speech & Language*, vol. 21, no. 3, pp. 492–518, 2007.

[4] Junho Park, Xunying Liu, Mark JF Gales, and Phil C Woodland, "Improved neural network based language modelling and adaptation," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[5] Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon, "Structured output layer neural network language models for speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 1, pp. 197–206, 2013.

[6] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[7] Martin Sundermeyer, Hermann Ney, and Ralf Schlüter, "From feedforward to recurrent lstm neural networks for language modeling," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, 2015.

[8] Da Zheng, Zhehuai Chen, Yue Wu, and Kai Yu, "Directed automatic speech transcription error correction using bidirectional lstm," in *Chinese Spoken Language Processing (ISCSLP), 2016 10th International Symposium on*. IEEE, 2016, pp. 1–5.

[9] Léon Bottou, "Stochastic gradient learning in neural networks," *Proceedings of Neuro-Nımes*, vol. 91, no. 8, pp. 12, 1991.

[10] Roger Fletcher, *Practical methods of optimization*, John Wiley & Sons, 2013.

[11] Dong C Liu and Jorge Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[12] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[13] James Martens, "Deep learning via hessian-free optimization.," in *ICML*, 2010, vol. 27, pp. 735–742.

[14] Brian Kingsbury, Tara N Sainath, and Hagen Soltau, "Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[15] James Martens and Ilya Sutskever, "Training deep and recurrent networks with hessian-free optimization," in *Neural networks: Tricks of the trade*, pp. 479–535. Springer, 2012.

[16] Adnan Haider and Philip C Woodland, "Sequence training of dnn acoustic models with natural gradient," in *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*. IEEE, 2017, pp. 178–184.

[17] Xunying Liu, Shansong Liu, Jinze Sha, Jianwei Yu, Zhiyuan Xu, Xie Chen, and Helen Meng, "Limited-memory bfgs optimization of recurrent neural network language models for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6114–6118.

[18] Shun-ichi Amari, "Neural learning in structured parameter spaces-natural riemannian gradient," in *Advances in neural information processing systems*, 1997, pp. 127–133.

[19] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533, 1986.

[21] Barak A Pearlmutter, "Fast exact multiplication by the hessian," *Neural computation*, vol. 6, no. 1, pp. 147–160, 1994.

[22] Yingkai Li and Huidong Liu, "Implementation of stochastic quasi-newton's method in pytorch," *arXiv preprint arXiv:1805.02338*, 2018.

[23] Shun-Ichi Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[24] Razvan Pascanu and Yoshua Bengio, "Revisiting natural gradient for deep networks," *arXiv preprint arXiv:1301.3584*, 2013.

[25] James Martens, "New insights and perspectives on the natural gradient method," *arXiv preprint arXiv:1412.1193*, 2014.

[26] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al., "Natural neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2071–2079.

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017.

[28] Steve Young, G Evermann, M Gales, T Hain, D Kershaw, X Liu, G Moore, J Odell, D Ollason, D Povey, et al., "The htk book (for htk version 3.5). university of cambridge, 2015," .

[29] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[30] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[31] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al., "The ami meeting corpus: A pre-announcement," in *International workshop on machine learning for multimodal interaction*. Springer, 2005, pp. 28–39.

[32] Haipeng Wang, Anton Ragni, Mark John Gales, Katherine Mary Knill, Philip Charles Woodland, and Chao Zhang, "Joint decoding of tandem and hybrid systems for improved keyword spotting on low resource languages," 2015.