EXPLORING ONE PASS LEARNING FOR DEEP NEURAL NETWORK TRAINING WITH AVERAGED STOCHASTIC GRADIENT DESCENT

Zhao You, Xiaorui Wang, Bo Xu

Interactive Digital Media Technology Research Center,
Institute of Automation, Chinese Academy of Sciences,
Beijing, P.R.China
{zhao.you, xiaorui.wang, xubo}@ia.ac.cn

ABSTRACT

Deep neural network acoustic models have shown large improvement in performance over Gaussian mixture models (G-MMs) in recent studies. Typically, deep neural networks are trained based on the cross-entropy criterion using stochastic gradient descent (SGD). However, plain SGD requires scanning the whole training set many passes before reaching the asymptotic region, making it difficult to scale to large dataset. It has been established that the second order SGD can potentially reach its asymptotic region in one pass through the training dataset. However, since it involves expensive computing for the inverse of Hessian matrix in the loss function, its application is limited. Averaged stochastic gradient descent (ASGD) is proved simple and effective for one pass on-line learning. This paper investigates the ASGD algorithm for deep neural network training. We tested ASGD on the Mandarin Chinese record speech recognition task using deep neural networks. Experimental results show that the performance of one pass ASGD is very close to that of multiple passes SGD.

Index Terms— deep neural network, speech recognition, averaged stochastic gradient descent, one pass learning

1. INTRODUCTION

In the past few years, deep neural networks (DNNs) were introduced to speech recognition tasks and gained great successes. Specially, the DNN-HMM acoustic models achieved significant recognition error reduction over discriminatively trained GMM-HMM models [1]. It is believed that the efficient and powerful modeling ability of deep networks is the critical factor for the remarkable accuracy gains [2, 3].

Stochastic gradient descent has become the most popular algorithm for training deep neural networks. For large scale learning, SGD operates on minibatches and generally converges to a stable region with acceptable performance. However, it takes about 10 to 20 passes (or epochs) through the training data to converge at satisfying parameters. The multiple passes procedure becomes difficult when one envisions very large training data. More recently, one pass large scale learning is considered as a solution of large scale learning because of its fast convergence property. Polyak and Juditsky [4] firstly proposed the ASGD and showed that ASGD could reach the asymptotic region by going through the training data in only one pass. The original algorithm for ASGD could only operates on very simple tasks. Therefore, Xu [5] further optimized this training algorithm and showed the superiority of ASGD. Besides, ASGD has already been applied to training the models for a number of classic machine learning schemes [6].

In this paper, we aim to explore the one pass learning algorithm (i.e., averaged stochastic gradient descent) for training DNNs. The first two experiments are conducted on a subset (about 100 hours) of the training dataset. It is observed that choosing appropriate learning rate schedule contributes to faster convergence and a better convergence value. Furthermore, we find that different sizes of minibatches result in different training time and character error rate. The last experiments are conducted on the whole 300-hours training dataset. We show that the results obtained by one pass ASGD are very close to the best results from the multiple passes SGD.

The rest of the article is organized as follows. The ASGD algorithm is described in Section 2. The experimental configuration is described in Section 3. We report the experimental results in Section 4 and conclude the paper in Section 5. In Section 6, we provide a discussion of relation to prior work.

2. ONE PASS LEARNING WITH AVERAGED STOCHASTIC GRADIENT DESCENT

2.1. Stochastic Gradient Descent

Let us first consider a supervised learning task. Let θ denote the deep network parameters, $L(\theta)$ denote the loss function,$\nabla L(\theta)$ denote gradient of the loss with respect to the

This work is partly supported by National Program on Key Basic Research Project (973 Program) under Grant 2013CB329302 and National Natural Science Foundation of China grant 61103152.

parameters θ. Typically, the loss $L(\theta)$ upon deep networks is minimized by gradient descent (GD) [7]. Concretely, the network parameters θ are updated as follows

$$\theta_{t+1} = \theta_t - \gamma_t \nabla L_N(\theta_t) \tag{1}$$

where $\nabla L_N(\theta) = \sum_{i=1}^{N} \nabla L_i(\theta)$ is the gradient on the training set, and N is the number of training samples. γ_t is learning rate. This method is based on the batch mode and the parameters updating at each epoch uses the gradient of the whole dataset. Consequently, this method usually shows poor performance of optimizing the highly non-linear deep model on large training dataset.

SGD is the on-line version of gradient descent. Instead of computing the gradient of the whole training dataset, SGD updates the parameters using gradient of n training samples randomly sampled from the training dataset,

$$\theta_{t+1} = \theta_t - \gamma_t \nabla L_n(\theta_t) \tag{2}$$

where $\nabla L_n(\theta) = \sum_{i=1}^{n} \nabla L_i(\theta)$. We refer to the n training samples as a minibatch. Generally, we use minibatch SGD to express this algorithm.

Compared with GD algorithm, SGD has shown great promise for non-linear model training. That's why SGD is the most popular algorithm for training DNNs. However, it takes too many passes of training dataset for SGD to reach the asymptotic region.

2.2. Averaged Stochastic Gradient Descent

For large scale learning tasks, dealing with optimization problems on billions of training samples is needed and it is desirable for the optimization algorithm to reach the asymptotic region by going through the training data in only one pass. In order to accelerate the convergence speed of SGD, Polyak and Juditsky [4] proposed the ASGD algorithm and performed the normal stochastic descent in only one pass. For the ASGD, the average of model parameters $\bar{\theta}_t = \frac{1}{t}\sum_{j=1}^{t} \theta_j$ from SGD is considered as the final model. They showed a nice result using the ASGD in just one pass of training data. To deal with large scale learning, Xu [5] improved this algorithm, by adding a running average $\bar{\theta}_{t+1} = (1-\eta_t)\bar{\theta}_t + \eta_t \theta_{t+1}$. The average procedure is shown concretely as follows

$$\theta_{t+1} = \theta_t - \gamma_t \nabla L(\theta_t) \tag{3}$$

$$\bar{\theta}_{t+1} = (1-\eta_t)\bar{\theta}_t + \eta_t \theta_{t+1} \tag{4}$$

where η_t is the rate of averaging. $\eta_t = 0.01$ for all the experiments described in this paper. The implementation of ASGD is shown in Algorithm 1.

In the beginning, the model θ and $\bar{\theta}$ are initialized with model generated from pretraining. Then the training dataset is divided into several minibatches. Like SGD, ASGD updates the parameters θ_t using gradient $\nabla L_i(\theta_t)$ which is the gradient of $i-th$ minibatch. In the initial period of training, we use the running average $\bar{\theta}_{t+1} = (1-\eta_t)\bar{\theta}_t + \eta_t \theta_{t+1}$. Once $\bar{\theta}_t$ is better than θ_t, we begin the model average $\bar{\theta}_{t+1} = \frac{1}{t+1}\sum_{j=1}^{t+1} \theta_j$. Unlike the second order optimization methods [8, 9], ASGD is extremely easy to implement, and we focus on this algorithm in this paper.

Algorithm 1 Averaged Stochastic Gradient Descent

```
1: initialize $\theta_0 = \theta_{init}$; $\bar{\theta}_0 = \theta_{init}$; $t = 0$
2: for $i$ in minibatches do
3:     $\nabla L_i(\theta_t) \leftarrow$ compute the gradient on model $\theta_t$
4:     $\theta_{t+1} = \theta_t - \gamma_t \nabla L(\theta_t)$
5:     if $L(\bar{\theta}_t) > L(\theta_t)$ then
6:         $\bar{\theta}_{t+1} = (1-\eta_t)\bar{\theta}_t + \eta_t \theta_{t+1}$
7:     else
8:         $\bar{\theta}_{t+1} = \frac{1}{t+1}\sum_{j=1}^{t+1} \theta_j$
9:     end if
10:    $t = t + 1$
11: end for
```

3. MANDARIN CHINESE RECORDED SPEECH RECOGNITION

3.1. Data

In order to evaluate the effectiveness of ASGD, we perform a series of experiments using 300 hours of conversational Mandarin Chinese recorded speech. The feature vectors used for this speech recognition system are 13-dimensional perceptual linear predictive (PLP) features appended with the first and second order derivatives. Frames are 25ms with a 10ms shift between successive frames. For DNN models, 11 consecutive frames are used as network input. Mean and variance normalization is performed on per utterance case. The speech recognition system is evaluated using two individual speech test sets, namely clean7k and noise360, which are collected through mobile microphone under clean and noise environments. We hold out about 10 hours as a development set for frame accuracy evaluation. The character error rate (CER) is used to measure the performance of Mandarin Chinese speech recognition.

3.2. Training tools

All the deep networks in this paper are trained using the modified Kaldi toolkit [10]. For fast training, we use the distributed training algorithm (asynchronous stochastic gradient descent). The implementation of asynchronous SGD refers to our prior work [11]. In this paper, we use a single sever with two dual-core GPU (NVIDIA Geforce GTX690) cards. Therefore, each deep network can be trained on the two dual-core cards in parallel.

3.3. Deep networks

Before training DNN, we first train a GMM-HMM system with maximum likelihood (ML) and boosted maximum mutual information (BMMI) criteria. The GMM system is served as the label generator at the frame level.

The deep networks used for our experiments have seven hidden layers each containing 2048 hidden units and an output layer with 19864 senones. Two steps are used for the deep networks training: generative pretraining with the restricted Boltzmann machines (RBMs) and discriminative training using the minibatch ASGD with distributed optimization (described in section 3.2).

4. EXPERIMENTS

4.1. Effect of learning rate scheduling

It is well known that learning rate scheduling is essential for obtaining satisfying parameters. Specially, it has been reported in [5] that it takes a large scale training samples for ASGD to obtain satisfying parameters if learning rate schedule is chosen improperly. Therefore, the goal of our first experiment is to explore an appropriate learning rate scheduling for ASGD. A typical choice of learning rate γ_t is to use the Adagrad [12] adaptive learning rate schedule. For this learning rate schedule, the learning rate $\gamma_t = \gamma_{ij}(t)$ of each parameter is defined as $\frac{\gamma(t)}{\sqrt{K+\sum_{i=\tau}^{t} \nabla L_{ij}(\theta_t)^2}}$, which depends on the history of the gradient with respect to that parameter. Following [13] we use a variant of Adagrad with a limited memory and set $K = 1$. Specially, we set $\tau = t$ to make the learning rate dependent only on the current gradient.

Next, we need to consider how to choose the global learning rate $\gamma(t)$. In this paper, we use four learning rate schedules to explore the effect of different global learning rate schedules.

- *ASGD-con*: the simplest method of setting the learning rate $\gamma(t)$ is to use a constant learning rate $\gamma(t) = \gamma$. We set $\gamma = 0.00025$ by observing the performance of different learning rates on a subset of the training dataset.
- *ASGD-xu*: another typical choice for the learning rate $\gamma(t)$ is to use a decaying learning rate $\gamma(t) = \gamma_0(1 + t/a)^{-c}$ (we set $\gamma_0 = 0.008$, $a = 30,000,000$). Following Xu [5] we set $c = 0.75$.
- *ASGD-exp*: the exponential decaying learning rate $\gamma(t) = \gamma_0 \times 10^{-t/\rho}$ (we set $\gamma_0 = 0.008$, $\rho = 18,000$) is also considered.
- *ASGD-per*: a further choice of the global learning rate is based on development set performance (frame accuracy). We decay the learning rate by a decay rate α (we set $\alpha = 0.9995$) when the development set frame accuracy does not improve.

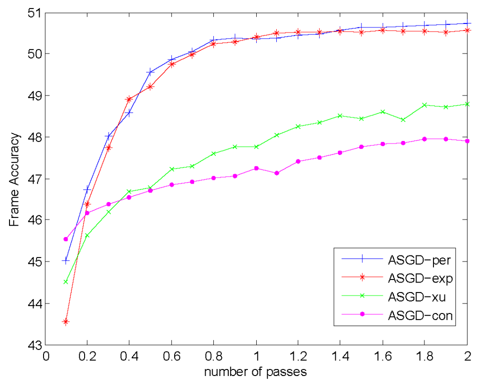

Fig. 1. Frames accuracy on the subset (about 100 hours) of training dataset against number of passes through the subset of training dataset for different learning rate schedule

Table 1. *Training runtimes in hours (h) and character error rate (%) for different sizes of minibatch. - denotes divergence.*

minibatch size	64	128	256	512	1024
training time	8.3	4.2	2.3	1.7	-
clean7k	9.25	9.27	9.30	9.45	-
noise360	26.28	26.47	26.80	27.23	-

Figure 1 shows a comparison of the four global learning rate schedules. We find that the frame accuracy of *ASGD-con* and *ASGD-xu* increase slowly due to a slow decrease of the learning rate. The performance of *ASGD-exp* is very close to that of *ASGD-per*. Especially, for these two schedules, the curve tends to flatten out after one pass. It is because that the learning rates become so small that the frame accuracy increases slowly. Moreover, we find that *ASGD-per* performs the highest frame accuracy. For the subsequent experiments, we use *ASGD-per* schedule.

4.2. Effect of minibatch size

The ASGD optimization is based on minibatch mode, thus the size T of the minibatch is a vital factor for the system performance and training efficiency. Table 1 shows that the size of minibatch influences both the training time and performance. Undersized minibatch ($T = 64$) will increase the training time due to the poor utilization of GPU computation units. Oversized minibatch ($T = 512$) will degrade the system performance. Moreover, training tends to diverge when the minibatch size increase to 1024. When the minibatch size is in the range 256 to 512, the training time decreases little, primarily due to the limited number of computing units in GPU. Thus, to make a balance between system performance and training efficiency, we set the minibatch size $T = 256$ for ASGD.

Table 2. *character error rate (%) on clean7k and noise360 for SGD and ASGD.*

Testset	clean7k	noise360
SGD multiple passes	8.11	22.00
ASGD one pass	8.49	22.87
ASGD two passes	8.35	22.43

4.3. One pass vs. multiple passes

As we know, it takes about 10 to 20 passes (epochs) to optimize the parameters for traditional SGD. Generally, we can get satisfying parameters using the multiple passes optimization. In this paper, the multiple passes optimization procedure is performed using the distributed asynchronous SGD algorithm. The learning rate decaying method is similar with *ASGD-per* schedule. Specially, we decay the learning rate using smaller decay rate ($\alpha = 0.5$) to adjust the learning rate. The iteration stops if the development set frame accuracy increment is smaller than a small constant λ (we set $\lambda = 0.05\%$). Note that we use different minibatch size during the multiple passes training iterations. For the first iteration, we use very small minibatch ($T = 64$) for training DNN with partial (about 10%) training dataset. For the next two iterations, large minibatch ($T = 256$) is used. For the subsequent iterations, we tend to use larger minibatch ($T = 1024$).

Comparing with multiple passes optimization, a natural question is whether we can get comparable results using one pass optimization. To answer this question, we use the entire 300 hours of training data and test two DNN acoustic models. One DNN is trained with ASGD optimization. The other DNN uses traditional SGD optimization. Figure 2 reports development set frame accuracy from the two DNNs. Table 2 further shows the corresponding CER on clean7k and noise360. We can observe that ASGD achieves near optimal results after one pass. Additionally, the results obtained by one pass ASGD are very close to the best results using the multiple passes SGD.

5. CONCLUSIONS

In this paper, one pass learning algorithm based on ASGD optimization for training DNN acoustic models is presented. We explore this algorithm for acoustic modeling on three different experiments. It is observed that careful choice of learning rate scheduling algorithm and minibatch size can result in fast convergence, less training time and good performance for ASGD. In particular we show that ASGD with one pass can be very close to the best results obtained from general SGD with multiple passes.

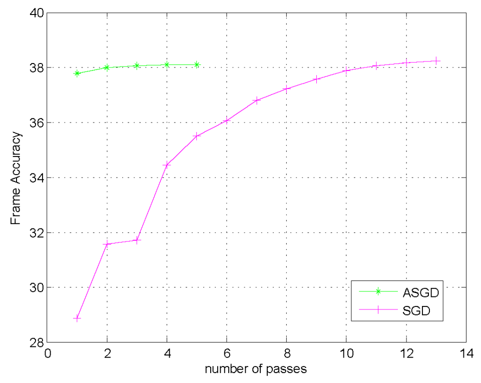

Fig. 2. Frames accuracy on development set against number of passes through training dataset for SGD and ASGD

6. RELATION TO PRIOR WORK

Most of previous works generally focus on the learning algorithms of SGD and the application on multiple passes optimization. To our knowledge, the work of this paper is the first time that ASGD with one pass optimization are applied for training DNNs. Our work is based on the efficient learning algorithm of ASGD proposed by [4, 5]. A similar work on ASGD with one pass optimization is described in [6], which uses this algorithm to solve the simple linear optimization problems. In our work AGSD are used as the solution of complex non-linear optimization problem.

7. ACKNOWLEDGEMENT

We would like to thank Dong Yu at Microsoft Corporation for valuable discussions and suggestions.

8. REFERENCES

[1] Abdel rahman Mohamed, George E. Dahl, and Geoffrey Hinton, "Acoustic Modeling Using Deep Belief Networks," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, pp. 14–22, 2012.

[2] Frank Seide, Gang Li, and Dong Yu, "Conversational speech transcription using context-dependent deep neural networks," in *INTERSPEECH*, 2011, pp. 437–440.

[3] Tara N. Sainath, Brian Kingsbury, Bhuvana Ramabhadran, Petr Fousek, Petr Novák, and Abdel rahman Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *ASRU*, 2011, pp. 30–35.

[4] Boris T Polyak and Anatoli B Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[5] Wei Xu, "Towards optimal one pass large scale learning with averaged stochastic gradient descent," *CoRR*, vol. abs/1107.2490, 2011.

[6] Léon Bottou, "Large-scale machine learning with s-tochastic gradient descent," in *Proceedings of COMP-STAT'2010*, pp. 177–186. Springer, 2010.

[7] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, "Learning internal representations by error propagation," Tech. Rep., DTIC Document, 1985.

[8] Brian Kingsbury, Tara N Sainath, and Hagen Soltau, "S-calable minimum bayes risk training of deep neural net-work acoustic models using distributed hessian-free op-timization.," in *INTERSPEECH*, 2012.

[9] O. Vinyals and D. Povey, "Krylov Subspace Descent for Deep Learning," in *AISTATS*, 2012.

[10] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Han-nemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *AS-RU*, 2011.

[11] Shanshan Zhang, Ce Zhang, Zhao You, Rong Zheng, and Bo Xu, "Asynchronous stochastic gradient descent for dnn training," ICASSP, 2013.

[12] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Re-search*, vol. 999999, pp. 2121–2159, 2011.

[13] Andrew Senior, Georg Heigold, M Ranzato, and K Yang, "An empirical study of learning rates in deep neural networks for speech recognition," in *IEEE Inter-national Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, vol. 1, pp. 6724–6728.