CSLT TECHNICAL REPORT-20150032 [Thursday 7th January, 2016]

AdaMax Online Training for Speech Recognition

Xiangyu Zeng[1], Zhiyong Zhang[1] and Dong Wang[1,2*]

*Correspondence: wang-dong99@mails.tsinghua.edu.cn
[1]Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China
Full list of author information is available at the end of the article

Abstract

We introduce an online adaptive training approach based on AdaMax to improve speech recognition in time-variant conditions. AdaMax, a variant of Adam based on the infinity norm, is a first-order gradient-based optimization method. Due to its capability of adjusting the learning rate based on data characteristics, it is suited to learn time-variant process, e.g., speech data with dynamically changed noise conditions. We apply this property to train and adapt deep neural network (DNN) acoustic models for speech recognition. The experiments were conducted on Auroa4, where the time-variation is simulated by injecting different types of noise into the training data. The results show that AdaMax can adapt to a new noise condition more quickly than the conventional SGD algorithm.

Keywords: speech recognition; deep neural network; stochastic optimization

1 Introduction

Stochastic gradient-based optimization is of core practical importance in machine learning. Among various optimization algorithms, gradient descent (GD) is a particulary interesting due to its simplicity and efficiency. For large scale optimization problems, stochastic gradient descent (SGD) is often used instead of GD, attributed to its quick convergence. Many studies have demonstrated that SGD is very efficient and effective on a broad range of learning tasks, e.g., in training deep neural networks [1].

To ensure convergence, the learning rate of SGD needs to be reduced gradually to zero when the training iterates on the data and approach to a stationary point. It has been found that a simple geometric learning rate reduction is sufficient to obtain good performance, although many alternatives exist, for example the NewBob approach [2].

This learning rate reduction, however, encounters problems with online learning. For online learning, we mean the data will be fed to the training algorithm only once, so there will be no chance to iterate over the entire database. If the training data is time-invariant, e.g., the distribution of the training data keeps no change during the training, then it is still reasonable to reduce the learning rate according to the amount of data that has been processed. It can be proved that this online learning rate reduction approach converges to a stationary point.

If the training data is time-variant, the reduced learning rate will cause disasters, as there will be no hope to get out of the local minimum that have been learned based on past data. This means that new data can not be learned anymore, due to the infinitesimal learning rate. A possible solution is to use second-order information, so that the learning rate is not reduced gradually, but is determined by

the present Hessian. The second-order methods, however, are generally very expensive, which makes it infeasible to apply to large-scale learning tasks, for example DNN-based acoustic model training in speech recognition.

1.1 Adam for online training

Recently, an Adam approach was proposed to approximate second-order properties but keep the computation as the same order as SGD [3]. The name Adam comes from adaptive moment estimation, and the basic idea is to combine advantages of two popular methods: AdaGrad [4], which shows a good performance with sparse gradients, and RMSProp [5], which works well in on-line training and non-stationary data. More specifically, the method can compute individual adaptive learning rates for different parameters, by estimating the first- and second-order moments of the gradients. One advantage of Adam is that the stepsizes are approximately bounded by the stepsize hyperparameter, which makes the learning stable. Moreover, the effective stepsize is adjusted automatically, leading to automatic annealing at stationary points, and automatic resurgence with new data. This enables Adam amiable to online learning where the condition (hence the data distribution) changes dynamically: it can get out of local minima learned from old data and move to new minima that are suitable for new data. The Adam algorithm is shown as follows, reproduced from [3].

Table 1 Algorithm of Adam.

Algorithm 1: g_t^2 indicates the elementwise square $g_t \odot g_t$.
Good default settings for the tested machine learning problems are:
$\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^-8$.
All operations on vectors are element-wise.
With β_1^t and β_2^t, we denote β_1 and β_2 to the power t.
Require: α: Stepsize
Require: β_1, $\beta_2 \in$ [0, 1): Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1^{st} moment vector)
$v_0 \leftarrow 0$ (Initialize 2^{nd} moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged do
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1}$ + (1 - β_1) $\cdot$ g_t (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1}$ + (1 - β_2) $\cdot$ g_t^2 (Update biased second raw moment estimate)
$\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1}$ - $\alpha \cdot /(\sqrt{\hat{v}_t} + \epsilon)$(Update parameters)
end while
return θ_t (Resulting parameters)

An variant of Adam is to replace the second-order moment v_0 with the infinite-order moment. The algorithm is called AdaMax and shown as follows. More details can be found in [3].

2 Related work

Adam is related to RMSProp [5, 6] and AdaGrad [4]. The update formula of RMSProp with momentum is [6]:

Table 2 Algorithm of Adamax.

Algorithm 2: Good default settings for the tested machine learning problems are $\alpha = 0.002$, $\beta_1 = 0.9$, $\beta_2 = 0.999$.
With β_1^t , we denote β_1 to the power t.
Here, $(\alpha/(1-\beta_1^t))$ is the learning rate with the bias-correction term for the first moment.
All operations on vectors are element-wise.
Require: α: Stepsize
Require: β_1, $\beta_2 \in$ [0, 1): Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow$ 0 (Initialize 1^{st} moment vector)
$u_0 \leftarrow$ 0 (Initialize the exponentially weighted infinity norm)
$t \leftarrow$ 0 (Initialize timestep)
while θ_t not converged do
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1}$ + (1 - β_1) $\cdot$ g_t (Update biased first moment estimate)
$u_t \leftarrow$ max($\beta_2 \cdot u_{t-1}$,$\lvert g_t \rvert$) (Update exponentially weighted infinity norm)
$\theta_t \leftarrow \theta_{t-1}$ - $(\alpha/(1-\beta_1^t)) \cdot m_t/u_t$(Update parameters)
end while
return θ_t (Resulting parameters)

$$
\begin{aligned}
g_t &= \nabla_\theta f_t(\theta_{t-1}) && (1)\\
m_t &= \beta_1 m_{t-1} + (1-\beta_1) && (2)\\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 && (3)\\
\Delta_t &= \beta_3 \Delta_{t-1} - \alpha \frac{g_t}{\sqrt{v_t - m_t^2 + \eta}} && (4)\\
\theta_t &= \theta_{t-1} + \Delta_t. && (5)
\end{aligned}
$$

It can be seen that RMSProp and Adam share the same idea to adjust the learning rate by a running averaged gradient, so they are both suitable for online training. The difference is that they use different ways to rescale the learning rate. More importantly, RMSProp lacks the bias-correction, which leads to unstable training in particular with a small value β_2.

AdaGrad is another popular approach for learning rate adjustment. The update formulation of AdaGrad is

$$\theta_{t+1} = \theta_t - \alpha \cdot g_t / \sqrt{\sum_{i=1}^{t} g_t^2}.$$

This corresponds to a special Adam where $\beta_1 = 0$, infinitesimal $(1-\beta_2)$ and a replacement of α by an annealed version $\alpha_t = \alpha \cdot t^{-1/2}$. More specifically,

$$
\begin{aligned}
\theta_t - \alpha \cdot t^{-1/2} \cdot \hat{m}_t / \sqrt{\lim_{\beta_2 \to 1} \hat{v}_t} &= \theta_t - \alpha \cdot t^{-1/2} \cdot g_t / \sqrt{t^{-1} \cdot \sum_{i=1}^{t} g_t^2} && (6)\\
&= \theta_t - \frac{\alpha}{\sqrt{\sum_{i=1}^{t} g_t^2}} g_t. && (7)
\end{aligned}
$$

Note that this direct correspondence between Adam and AdaGrad does not hold when removing the bias-correction terms; A step further, without the bias correc-

tion, like in RMSProp, a β_2 infinitely close to 1 would lead to infinitely large bias, and infinitely large parameter updates.

Other stochastic optimization methods includes vSGD [7], AdaDelta [8] and the natural Newton method from Roux and Fitzgibbon [9]. All these methods set step-sizes by estimating the second-order statistics from first-order information. Sum-of-Functions Optimizer (SFO) [10] is a quasi-Newton method based on minibatches, but it requires large memory. Like natural gradient descent (NGD) [11], Adam employs a preconditioner that adapts to the geometry of the data, since v_t is an approximation to the diagonal elements of the Fisher information matrix [12]. However, Adam's preconditioner (like AdaGrad's) is more conservative in its adaption than vanilla NGD.

3 Online learning for ASR

We focus on online training with time-invariant data. In speech recognition, the acoustic environment may change significantly due to different background noise. Model adaptation is often used to deal with the changed acoustic conditions, for example MAP or MLLR. However, with DNN-based acoustic models, adaptation is not simple due to the compact structure. Particularly, the change on acoustic conditions may be unnoticed: the background noise may change gradually. This requires an online learning algorithm that can learn the DNN model sequentially with new available data.

The conventional SGD method does not meet this request well, since it uses an annealed learning rate to ensure convergence. Once the learning rate has been shrunk to a small value, it is hard to update the DNN model any more. One may think to enlarge the learning rate if new data is available, but this is not feasible because we often do not know whether the new data is 'the same as' or 'different from' the past data, and therefore do not know if we should enlarge or shrink the learning rate. We have to seek a way that can change the learning rate automatically, for example, the Adam algorithm.

The online training task that we focus involves two characteristics: (1) all the data are provided sequentially and then are thrown away; (2) the background noise (so the data distribution) is changed without notice. We simulate the time-variant condition by copying the clean data and injecting noise to simulate a particular acoustic condition. Note that we shuffle the data before the copy to ensure sufficient randomness.

4 Experiment

4.1 Databases

The experiments are conducted on the Aurora4 database. The training set involves 15 hours of clean speech (7138 utterances), and the test dataset involves four subsets, each in a particular noise condition (clean, Airport, Babble, Car). Each subsect contains XXX utterances. The signal-to-noise level ranges from xx db to xxdb. To produce noise-corrupted training data, we use the noise samples from the DEMAND noise corpus[1]. The random noise injection approach in [13] is adopted in this study.

[1]http://xxx

4.2 Experimental settings

We used the Kaldi toolkit to conduct the training and evaluation, and largely followed the WSJ s5 GPU recipe. Specifically, The first step was to establish a GMM baseline. The feature was 39-dimensional MFCCs, including 13 static components plus the first- and second-order derivatives. The acoustic model was based one context-dependent phones (tri-phones), clustered by decisions trees. After the clustering, the model consisted of 3390 probability density functions (PDF) and the number of Gaussian components was 39997. The GMM system was used to produce phoneme alignments for the training data and provide the prototypes for the DNN system, including the HMM model that describes the transition characteristics of phoneme models, and the decision tree that describes the sharing scheme of the tri-phones.

The DNN baseline system was then trained utilizing the phone alignments produced by the GMM system. The 40-dimensional Fbank feature was adopted and the cepstral mean normalization (CMN) was employed to eliminate the effect of channel noise. In order to use dynamic information of speech signals, the left and right 5 frames was spliced and concatenated with the current frame. A linear discriminant analysis (LDA) transform was used to reduce the feature dimension to 200. The LDA-transformed feature was used as the DNN input.

The DNN architecture involved 4 hidden layers and each layer consisted of 1200 units. The output layer was composed of 6674 units, equal to the total number of PDFs in the GMM system. The training criterion was set to cross entropy, and the stochastic gradient descendent (SGD) algorithm was employed to perform optimization, with the mini batch size set to 256 frames. This setting is quite close to the GPU recipe used in Kaldi. We used a NVIDIA G760 GPU unit to perform matrix manipulation.

4.3 Experimental results

4.3.1 Baseline

The first experiment employs various training algorithms to build the baseline system. The trained follows the conventional iterative training recipe, using the clean training data. The learning rate (or basic learning rate) is reduced following the Newbob style, as implemented in the Kaldi WSJ s5 recipe.

The results in terms of word error rate (WER) are shown in Table 3. In this table, only SGD is purely first-order, and all other algorithms use second-order information, in different forms. Nestrov and AdaGrad reduces learning rate exponentially, while AdaDelta and Adamax tune the learning rate according to data. It can be observed that both Adam and AdaMax deliver good performance, especially in noisy conditions. This means that Adam/AdaMax (and most learning rate adjustment methods) tends to learn more generalizable models. Interestingly, although only babble noise is used in noisy training, performance on other noisy conditions are generally improved, which is consistent to the results reported in [14].

4.3.2 Online learning

Due to the high performance of AdaMax, we focus on this approach and evaluate its capability in online learning, and compare it with the conventional SGD algorithm.

Table 3 Baseline performance on four test sets with different training algorithms.

	WER/%					
Test set	initial learning rate	Num of iter	Clean	Airport	Babble	Car
Baseline(SGD)	0.008	14	6.42	35.38	36.42	15.94
Nestrov	0.008	14	6.25	34.37	36.88	16.22
AdaGrad-P2	0.008	12	6.89	35.85	35.95	15.37
AdaGrad-max	0.008	12	6.42	30.81	31.34	13.63
AdaDelta	0.1	16	6.76	30.98	31.17	14.41
Adam	0.001	15	6.53	32.29	33.09	14.79
AdaMax	0.008	12	6.59	**27.30**	**28.24**	**13.98**

To enable SGD online training, the basic learning rate was fixed to 0.008 (otherwise the too small learning rate will prohibit learning new data with SGD).

In the simplest configuration, we copy some clean data, and then augment some copy of noisy data with babble noise injected. The mean and variance of the SNR are chosen to be xx and xx respectively for the noise injection.

We test two scenarios in this experiment: In test A, copy the clean data 9 times and then add one copy of babble noisy data; in test B, 5 copies of clean data are concatenated with 5 copies of babble noisy data. The results are shown in Table 4, where 'SGD-nc' means the model after n copies of clean data have been used in training, and 'SGD-nc-mb' means n copies of clean data and m copies of babble noisy data have been used.

Table 4 Performance of SGD and AdaMax online training.

Test set	Clean	Airport	babble	Car
SGD-9c	6.72	39.17	39.58	18.51
SGD-9c-1b	7.16	21.33	22.11	10.00
AdaMax-9c	7.04	30.18	31.02	14.74
AdaMax-9c-1b	7.56	18.43	19.35	9.58
SGD-5c	7.04	37.43	40.75	17.92
SGD-5c-5b	7.86	19.40	18.30	12.15
SGD-5c-5b-5c	6.78	24.03	24.70	13.61
AdaMax-5c	7.03	32.67	32.84	14.93
AdaMax-5c-5b	7.67	17.08	15.00	8.99
AdaMax-5c-5b-5c	7.41	27.76	31.19	13.42

Note that each copy, no matter clean or the noisy, contains the same number of utterances (7138 utterances and nearly 15h of speech signals). It can be seen that AdaMax generally produces much better performance than SGD, especially when the noise condition transits from one to another. For example, with the transition from '5c' to '5c-5b', AdaMax adapts the model quickly to the new condition and produces much lower WER than SGD.

An exception is that when the condition moves back to one that has been seen already, for example, '5c-5b-5c', AdaMax shows inferior performance than SGD, even in terms of generalizability. A possible reason is that the two clean data sets are totally the same, which is not realistic and may lead to some bias; unfortunately, this bias more benefits SGD. To verify this conjecture, we split the clean dataset into two parts: ca and cb, and re-run the experiments. The results are shown in Table 5.

It can be seen that in the condition '5c1-5b-5c2', AdaMax is still worse than SGD on clean data, but the generalizability on noisy conditions has been recovered. In the condition '5b1-5c-5b2', AdaMax shows clear better performance than SGD on noisy data. These results indicate that AdaMax can adjust the model quickly

Table 5 Performance of online training with SGD and AdaMax.

	WER/%			
Test set	Clean	Airport	Babble	Car
SGD-5c1	8.53	40.00	41.17	20.18
SGD-5c1-5b	8.02	21.08	18.47	12.55
SGD-5c1-5b-5c1	7.35	23.31	23.78	14.60
SGD-5c1-5b-5c2	6.93	24.33	26.16	14.64
AdaMax-5c1	8.04	33.82	31.97	17.73
AdaMax-5c1-5b	7.81	18.28	15.54	9.03
AdaMax-5c1-5b-5c1	7.54	23.86	24.09	13.42
AdaMax-5c1-5b-5c2	7.06	23.84	24.49	12.20
SGD-5b1	9.54	21.78	21.52	11.33
SGD-5b1-5c	7.20	31.65	33.28	16.17
SGD-5b1-5c-5b1	7.71	19.40	19.15	9.73
SGD-5b1-5c-5b2	7.83	21.52	21.88	10.74
AdaMax-5b1	9.20	19.63	18.77	11.04
AdaMax-5b1-5c	7.12	28.77	30.50	13.00
AdaMax-5b1-5c-5b1	8.23	18.07	16.95	10.09
AdaMax-5b1-5c-5b2	8.04	18.91	17.04	10.00

to accommodate a new condition, but tends to forget things quickly as well. In contrast, SGD does a slower job and remember more past things. This explains why fedding the same data leads to better performance with SGD: SGD re-uses the information that learned from the past clean data.

5 Conclusions

We report some empirical studies on online time-invariant DNN training with AdaMax. We focus on speech recognition in the scenario that the noise condition changes unnoticeably. The experimental results confirmed that AdaMax outperforms SGD in general, especially on time-invariant scenarios. It also suggests that AadaMax can quickly adapt to new conditions, but tends to forget things quickly as well.

Acknowledgement

This research was supported by the National Science Foundation of China (NSFC) under the project No. 61371136, and the MESTDC PhD Foundation Project No. 20130002120011. It was also supported by Sinovoice and Huilan Ltd.

Author details

[1]Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China. [2]Center for Speech and Language Technologies, Division of Technical Innovation and Development, Tsinghua National Laboratory for Information Science and Technology, ROOM 1-303, BLDG FIT, 100084 Beijing, China. [3]Department of Computer Science and Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China.

References

1. Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, and Jason Williams, "Recent advances in deep learning for speech research at microsoft," in *Proc. of ICASSP*, 2013.
2. "Quicknet", "http://www1.icsi.berkeley.edu/speech/qn.html. accessed: 2014-09-30.," .
3. Jimmy Lei Ba Diederik P. Kingma, "Adam: A method for stochastic optimization," in *Proc. of ICLR*, 2015.
4. John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.
5. T. Tieleman and G Hinton, "Rmsprop, coursera:neural networks for machine learning," in *Technical report*, 2012.
6. Alex Graves, "Generating sequences with recurrent neural networks," 2013.
7. Tom Schaul, Sixin Zhang, and Yann LeCun, "No more pesky learning rates," in *arXiv preprint*, 2012.
8. Matthew D Zeiler, "Adadelta: An adaptive learning rate method," in *arXiv preprint*, 2012.
9. Nicolas L Roux and Andrew W Fitzgibbon, "A fast natural newton method," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 623–630.
10. Jascha Sohl-Dickstein, Ben Poole, and Surya Ganguli, "Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 604–612.
11. Shun-Ichi Amari, "Natural gradient works efficiently in learning," in *Neural computation*, 1998, vol. 10, pp. 251–276.
12. Boris T Polyak and Anatoli B Juditsky, "Acceleration of stochastic approximation by averaging," in *IAM Journal on Control and Optimization*, 1992, vol. 30, pp. 838–855.
13. Martin Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," 2003.
14. Shi Yin, Chao Liu, Zhiyong Zhang, Yiye Lin, Dong Wang, Javier Tejedor, Thomas Fang Zheng, and Yiguo Li, "Noisy training for deep neural networks in speech recognition," in *EURASIP Journal on Audio, Speech, and Music Processing*, 2015.