ADASECANT: ROBUST ADAPTIVE SECANT METHOD FOR STOCHASTIC GRADIENT

Caglar Gulcehre and Yoshua Bengio
Université de Montréal

ABSTRACT

Stochastic gradient algorithms have been the main focus of large-scale learning problems and they led to impressive successes in machine learning. The convergence of SGD depends on the careful choice of learning rate and the amount of the noise in stochastic estimates of the gradients. In this paper, we propose a new adaptive learning rate algorithm, which utilizes curvature information for automatically tuning the learning rates. The information about the element-wise curvature of the loss function is estimated from the local statistics of the stochastic first order gradients. We further propose a new variance reduction technique which automatically reduces the variance of noise in the local gradient estimates to speed up the convergence. In our preliminary experiments with deep neural networks, we obtained better performance compared to the popular stochastic gradient algorithms.

1 INTRODUCTION

In this paper we develop a stochastic gradient algorithm that reduces the burden of extensive hyper-parameter search for the optimization algorithm. The proposed algorithm exploits a low variance estimator of curvature of the cost function and uses it to obtain an automatically tuned adaptive learning rate for each parameter.

In the deep learning and numerical optimization, several papers suggest using a diagonal approximation of the Hessian (second derivative matrix of the cost function with respect to parameters), in order to estimate optimal learning rates for stochastic gradient descent over high dimensional parameter spaces (Becker and Le Cun, 1988; Schaul *et al.*, 2012; LeCun *et al.*, 1993). Among several proposals to estimate the diagonal of Hessian efficiently, one option is to estimate it via the Gauss-Newton matrix (LeCun *et al.*, 2012) or by using finite differences (Schaul and LeCun, 2013). The diagonal estimations of the Hessian may however be sensitive to noise due to the stochastic gradient. Schaul *et al.* (2012) suggested a reliable way to estimate the local curvature in the stochastic setting. A fundamental advantage of using the diagonal estimation of the Hessian for deep learning problems is that inverting the diagonal of the Hessian is a trivial and cheap operation. However note that the inverse of the diagonal Hessian is usually a bad approximation of the diagonal of inverse Hessian.

Schaul *et al.* (2012) kept track of the variance and average of the gradients in order to keep a reliable stochastic estimation of the diagonal Hessian matrix. In this paper, we follow a different approach: instead of using a diagonal estimate of Hessian, we propose to estimate curvature along the direction of the gradient and we apply a new variance reduction technique to compute this metric reliably. The root mean square statistics and the variance of gradients are reduced adaptively by a very simple transformation. We keep track of the estimation of curvature using a technique similar to that proposed by Schaul *et al.* (2012), which uses the variability of the expected loss. Standard adaptive learning rate algorithms only scale the gradients, but regular Newton-like second order methods, can perform more complicate transformations, e.g. rotating the gradient vector. Newton and quasi-newton methods can also be invariant to affine transformations in the parameter space. Our proposed **Adasecant** algorithm is basically a stochastic rank-1 quasi-Newton method. But in comparison with other adaptive learning algorithms, instead of just scaling the gradient of each parameter, Adasecant can also perform an affine transformation on them.

2 DIRECTIONAL SECANT APPROXIMATION

Directional Newton is a method proposed for solving equations with multiple variables (Levin and Ben-Israel, 2002). The advantage of directional Newton method proposed in Levin and Ben-Israel (2002), compared to Newton's method is that, it does not require a matrix inversion and still maintains a quadratic rate of convergence.

In this paper, we develop a second-order directional Newton method for nonlinear optimization. Step-size of update Δ^k for step k can be written as if it were a diagonal matrix:

$$\Delta^k = -\operatorname{diag}(\mathbf{d}^k)(\operatorname{diag}(\mathbf{H}\mathbf{d}^k))^{-1}\nabla_{\boldsymbol{\theta}}\mathrm{f}(\boldsymbol{\theta}^k) \tag{1}$$

where $\boldsymbol{\theta}^k$ is the parameter vector at update k, f is the objective function and $\mathbf{d}$ is a unit vector of direction that the optimization algorithm should follow.

A reformulation of Equation 1 for each diagonal element of the step-size would be:

$$\Delta_i^k = -\frac{\nabla_{\boldsymbol{\theta}_i}\mathrm{f}(\boldsymbol{\theta}^k)}{\sum_{j=1}^{N}\frac{\partial^2\mathrm{f}(\boldsymbol{\theta}^k)}{\partial\theta_i\partial\theta_j}d_j^k}d_i^k \tag{2}$$

One potential option might be to use the R-op to compute the Hessian-vector product (Schraudolph, 2002) for the update in Equation 1. Instead, writing $\mathbf{h}_i = \nabla_{\boldsymbol{\theta}}\frac{\partial\mathrm{f}(\boldsymbol{\theta}^k)}{\partial\theta_i}$ for the i^{th} row of the Hessian matrix $\mathbf{H}$ and using the *directional secant approximation*,

$$t_i^k = \frac{d_i^k}{\mathbf{h}_i^k\mathbf{d}^k}, \tag{3}$$

The per-parameter learning rate is t_i^k and $\mathbf{d}^k$ is the unit vector along the direction that we want to estimate the curvature at update k. We can rewrite the gradient scaling factor (learning rate) following An and Bai (2005):

$$\frac{d_i^k}{\mathbf{h}_i^k\mathbf{d}^k} \approx \frac{t_i^k d_i^k}{\nabla_{\boldsymbol{\theta}_i}\mathrm{f}(\boldsymbol{\theta}^k + \mathbf{t}^k\mathbf{d}^k) - \nabla_{\boldsymbol{\theta}_i}f(\boldsymbol{\theta}^k)} \tag{4}$$

where $\nabla_{\boldsymbol{\theta}_i}f(\boldsymbol{\theta}^k)$ is the i^{th} element of the gradient vector at update k.

To choose a good direction $\mathbf{d}^k$ in the stochastic setting, we used the block-normalized gradient vector for each weight matrix $\mathbf{W}_k^i$ and bias vector $\mathbf{b}_k^i$ where $\boldsymbol{\theta} = \left\{\mathbf{W}_k^i, \mathbf{b}_k^i\right\}_{i=1\cdots k}$ at each layer i and update k, i.e. $\mathbf{d}_{\mathbf{W}_k^i}^k = \frac{\nabla_{\mathbf{W}_k^i}\mathrm{f}(\boldsymbol{\theta})}{||\nabla_{\mathbf{W}_k^i}\mathrm{f}(\boldsymbol{\theta})||_2}$ and $\mathbf{d}_k = \left[\mathbf{d}_{\mathbf{W}_k^0}^k\mathbf{d}_{\mathbf{b}_k^0}^k\cdots\mathbf{d}_{\mathbf{b}_k^l}^k\right]$ for a neural network with l layers. It is easy see that, for each stochastic gradient update the angle between the stochastic gradient and block-normalized gradient will still be less than 90 degrees. Nevertheless block normalization of the gradients add adds an additional noise to the stochastic gradient estimates but empirically this doesn't effect the convergence significantly.

The update step is defined as $\Delta_i^k = t_i^k d_i^k$. The per-parameter learning rate t_i^k can be estimated via a finite difference approximation,

$$t_i^k = \frac{\Delta_i^k}{\nabla_{\boldsymbol{\theta}_i}\mathrm{f}(\boldsymbol{\theta}^k + \Delta^k) - \nabla_{\boldsymbol{\theta}_i}\mathrm{f}(\boldsymbol{\theta}^k)}, \tag{5}$$

since, in the vicinity of the quadratic local minima,

$$\nabla_{\boldsymbol{\theta}}\mathrm{f}(\boldsymbol{\theta}^k + \Delta^k) - \nabla_{\boldsymbol{\theta}}\mathrm{f}(\boldsymbol{\theta}^k) \approx \mathbf{H}^k\Delta^k \tag{6}$$

We can therefore recover $\mathbf{t}^k$ as

$$\mathbf{t}^k = \operatorname{diag}(\Delta^k)(\operatorname{diag}(\mathbf{H}^k\Delta^k))^{-1}. \tag{7}$$

The directional secant method, basically scales the gradient of each parameter with the curvature along the direction of the gradient vector and it is numerically stable.

3 Variance Reduction for Robust Stochastic Gradient Descent

Variance reduction techniques for stochastic gradient estimators have been well-studied in the machine learning literature. Both Wang *et al.* (2013) and Johnson and Zhang (2013) proposed new ways of dealing with this problem. In this paper, we propose a new variance reduction technique for stochastic gradient descent that just relies on the statistics related to the gradients. $\mathbf{g}_i$ refers to the i^{th} element of the gradient vector $\mathbf{g}$ with respect to the parameters $\boldsymbol{\theta}$ and $\mathrm{E}[\cdot]$, is for the expectation taken over minibatches and different trajectories of parameters.

We propose to apply the following transformation to reduce the variance of the stochastic gradients:

$$\tilde{g_i} = \frac{g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i]}{1 + \gamma_{\mathbf{i}}} \tag{8}$$

The variance of the transformed stochastic gradient is:

$$\mathrm{E}[||\tilde{g_i} - \mathrm{E}[g_i]||_2^2] = \mathrm{E}[||\frac{g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i]}{1 + \gamma_{\mathbf{i}}} - \mathrm{E}[g_i]||_2^2] \tag{9}$$

$$= \frac{1}{(1 + \gamma_{\mathbf{i}})^2} \mathrm{E}[||g_i - \mathrm{E}[g_i]||_2^2] \tag{10}$$

This transformation will therefore reduce the variance of each stochastic gradient g_i by $(1 + \gamma_{\mathbf{i}})^2$.

If the estimation of $\mathrm{E}[g_i]$ is unbiased than the bias of $\tilde{g_i}$ will be 0 as well:

$$\mathrm{E}[\tilde{g_i}] - \mathrm{E}[g_i] = \mathrm{E}[\frac{g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i]}{1 + \gamma_{\mathbf{i}}}] - \mathrm{E}[g_i] \tag{11}$$

$$= \mathrm{E}[\frac{g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i] - \mathrm{E}[g_i] - \gamma_{\mathbf{i}} \mathrm{E}[g_i]}{1 + \gamma_{\mathbf{i}}}] \tag{12}$$

$$= 0 \tag{13}$$

However, in practice, our estimator of $\mathrm{E}[g_i]$ based on past values of g_i will be biased because the parameters keep changing, making averages of old values stale. Nevertheless by changing γ, it is possible to control the bias-variance tradeoff. The optimal γ for minimizing the bias of $\mathrm{E}[g_i]$ can be estimated by minimizing the following objective function:

$$\arg\min_{\gamma_i} \mathrm{E}[||\tilde{g_i} - g_i'||_2^2] \tag{14}$$

where $\mathbf{g}'$ is the gradient we obtained in the next minibatch. We can find the optimal γ in closed-form by finding the γ that minimizes the objective function in Equation 14. We assumed that γ is strictly positive real number. γ_i is basically a correction term that reduces the variance and bias of the stochastic gradients.

$$\frac{\partial \mathrm{E}[||\tilde{g_i} - g_i'||_2^2]}{\partial \gamma_i} = 0 \tag{15}$$

We satisfy Equation 15 separately for each parameter, yielding a separate γ_i in γ for each gradient element g_i in $\mathbf{g}$:

$$\frac{\partial \mathrm{E}[(\frac{g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i]}{1 + \gamma_{\mathbf{i}}} - g_i')^2]}{\partial \gamma_i} = 0 \tag{16}$$

$$\mathrm{E}[(\frac{g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i]}{1 + \gamma_{\mathbf{i}}} - g_i') \frac{\partial (\frac{(g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i])}{1 + \gamma_{\mathbf{i}}} - g_i')}{\partial \gamma_i}] = 0 \tag{17}$$

$$\mathrm{E}[(\frac{g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i]}{1 + \gamma_{\mathbf{i}}} - g_i')(\frac{(g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i]) - (1 + \gamma_i)(\mathrm{E}[g_i])}{(1 + \gamma_{\mathbf{i}})^2})] = 0 \tag{18}$$

$$\mathrm{E}[(g_i + \gamma_{\mathbf{i}} \mathrm{E}[g_i] - (1 + \gamma_i) g_{i\prime})(g_i - \mathrm{E}[g_i])] = 0 \tag{19}$$

$$\mathrm{E}[(g_i - g^{i\prime} + \gamma_{\mathbf{i}} (\mathrm{E}[g_i] - g_i'))(g_i - \mathrm{E}[g_i])] = 0 \tag{20}$$

$$\mathrm{E}[(g_i - g_i')(g_i - \mathrm{E}[g_i])] = \gamma_i \mathrm{E}[(g_i - \mathrm{E}[g_i])(g_i' - \mathrm{E}[g_i]))] = 0 \tag{21}$$

$$\gamma_i = \frac{\mathrm{E}[(g_i - g_i')(g_i - \mathrm{E}[g_i])]}{\mathrm{E}[(g_i - \mathrm{E}[g_i])(g^{i\prime} - \mathrm{E}[g_i]))]} \tag{22}$$

As a result, in order to estimate γ for each dimension, we should keep track of $\frac{\mathrm{E}[(g_i - g_i')(g_i - \mathrm{E}[g_i])]}{\mathrm{E}[(g_i - \mathrm{E}[g_i])(g_i' - \mathrm{E}[g_i]))]}$, during training.

4 ADAPTIVE STEP-SIZE IN STOCHASTIC CASE

In the stochastic gradient case, the step-size for directional secant can be computed by using an expectation over the mini-batches:

$$\mathrm{E}_k[t_i] = \mathrm{E}_k[\frac{\Delta_i^k}{\nabla_{\boldsymbol{\theta}_i} \mathrm{f}(\boldsymbol{\theta}^k + \Delta^k) - \nabla_{\boldsymbol{\theta}_i} \mathrm{f}(\boldsymbol{\theta}^k)}] \tag{23}$$

Considering that multiple passes are done over the dataset, the step size is computed as the expected value of the directional secant over different minibatches and trajectories.

Computing the expectation in 23 was numerically very unstable in the stochastic setting. Thus we decided to use a numerically more stable approximation of it. If we do a second order Taylor approximation of Equation 23 around $(\sqrt{\mathrm{E}_k[(\alpha_i^k)^2]}, \sqrt{\mathrm{E}_k[(\Delta_i^k)^2]})$, with $\alpha_i^k = \nabla_{\boldsymbol{\theta}_i} \mathrm{f}(\boldsymbol{\theta}^k + \Delta^k) - \nabla_{\boldsymbol{\theta}_i} \mathrm{f}(\boldsymbol{\theta}^k)$, and assume $\sqrt{\mathrm{E}_k[(\alpha_i^k)^2]} \approx \mathrm{E}_k[\alpha_i^k]$ and $\sqrt{\mathrm{E}_k[(\Delta_i^k)^2]} \approx \mathrm{E}_k[\Delta_i^k]$ then we obtain:

$$\mathrm{E}_k[t_i] \approx \frac{\sqrt{\mathrm{E}_k[(\Delta_i^k)^2]}}{\sqrt{\mathrm{E}_k[(\alpha_i^k)^2]}} - \frac{\mathrm{Cov}(\alpha_i^k, \Delta_i^k)}{\mathrm{E}_k[(\alpha_i^k)^2]} \tag{24}$$

Equation 24 should always be non-negative. In our experiments, we used a simpler approximation in Equation 25, which in practice, worked at least as well as the formulations in Equations 24:

$$\mathrm{E}_k[t_i] \approx \frac{\sqrt{\mathrm{E}_k[(\Delta_i^k)^2]}}{\sqrt{\mathrm{E}_k[(\alpha_i^k)^2]}} - \frac{\mathrm{E}_k[\alpha_i^k \Delta_i^k]}{\mathrm{E}_k[(\alpha_i^k)^2]} \tag{25}$$

5 ALGORITHMIC DETAILS

5.1 APPROXIMATE VARIABILITY

To compute the moving averages as also adopted by Schaul *et al.* (2012), we used an algorithm to adaptively decide the time constant based on the step size being taken. As a result algorithm that we used will give larger weights to the updates that has large step-size and smaller weight to the updates that has smaller step-size.

By assuming that $\bar{\Delta}_i[k] \approx \mathrm{E}[\Delta_i]_k$, the moving average update rule for $\bar{\Delta}_i[k]$ can be written as,

$$\bar{\Delta}_i^2[k] = (1 - \tau_i^{-1}[k])\bar{\Delta}_i^2[k-1] + \tau_i^{-1}[k](t_i^k \tilde{\mathbf{g}}_i^k) \tag{26}$$

$$\bar{\Delta}_i[k] = \sqrt{\bar{\Delta}_i^2[k]} \tag{27}$$

This update rule effectively assigns different weight for each element in the gradient vector and for each different update. At each update, a scalar multiplication is performed with τ_i^{-1} and it is adapted during the training by using the following equation:

$$\tau_i[k] = (1 - \frac{\mathrm{E}[\Delta_i]_{k-1}^2}{\mathrm{E}[(\Delta_i)^2]_{k-1}})\tau_i[k-1] + 1 \tag{28}$$

5.2 OUTLIER GRADIENT DETECTION

We used an outlier detection algorithm which resets the time-constant when an outlier gradient is detected. Our algorithm is very similar to Schaul and LeCun (2013), but instead of incrementing $\tau_i[t+1]$ when an outlier is detected, the time-constant is reset by reinitializing it to 2.2. Noting that when $\tau_i[t+1] \approx 2$, this approximately assigns same amount of weight for the current and the average of previous observations. This outlier detection mechanism helped learning to become more stable, because when an outlier gradient is detected, empirically it tends to cause τ_i to saturate to a large value as well.

5.3 VARIANCE REDUCTION

The correction parameter γ is adapted based on the observed gradients with the Equation given in Section 3. This enables us to perform a fine-grained level of variance reduction for each parameter independently, by using the estimates of

$$\gamma_i^k = \frac{\mathrm{E}[(g_i - g_i')(g_i - \mathrm{E}[g_i])]_k}{\mathrm{E}[(g_i - \mathrm{E}[g_i])(g_i' - \mathrm{E}[g_i]))]_k}.$$

If we use the variance reduction technique covered in Section 3, asymptotically it is possible to recover the batch gradient descent. The noise in the stochastic gradient methods can have advantages both in terms of generalization and optimization. But the noise also introduces an exploration and exploitation trade-off. It is possible to fine-tune this trade-off by thresholding the values of γ. We upper bounded γ with a maximum threshold value ρ_i. Such that, thresholded γ_i will be estimated by $\gamma_i' = \text{minimum}(\rho_i, \gamma_i)$.

We block-wise normalized the gradients of each weight matrix and bias vectors in $\mathbf{g}$ to compute the $\tilde{\mathbf{g}}$, for example the gradient $\mathbf{g}_{\mathbf{W}^{(i)}}$ of weight matrix $\mathbf{W}^{(i)}$ is ensured to be $\frac{\mathbf{g}_{\mathbf{W}^{(i)}}}{||\mathbf{g}_{\mathbf{W}^{(i)}}||_2^2}$ as described in Section 2. The block-normalization of $\mathbf{g}$ makes the Adasecant to be scale-invariant, hence more robust to the scale of the inputs and the number of the layers of the network. We also empirically observed that, it was easier to train very deep neural networks with block normalized gradient descent.

6 IMPROVING CONVERGENCE

Classical convergence results for SGD are based on the conditions,

$$\sum_i (\eta^{(i)})^2 < \infty$$

and

$$\sum_i \eta^{(i)} = \infty,$$

such that the learning rate $\eta^{(i)}$ should decrease (Robbins and Monro, 1951). Due to the noise in the estimation of adaptive step-sizes for Adasecant, the convergence is still not strictly guaranteed. To ensure convergence, we developed a new variant of Adagrad (Duchi *et al.*, 2011). We thresholded Adagrad, such that each Adagrad scaling factor is lower bounded by 1. Assume a_i^k is the accumulated norm of all past gradients for i^{th} parameter at update k:

$$a_i^k = \sqrt{\sum_{j=0}^{k} (g_i^j)^2} \tag{29}$$

The a_i^k is thresholded from below by simply using

$$\rho_i^k = \text{maximum}(1, a_i^k), \tag{30}$$

to ensure that the algorithm will eventually converge:

$$\Delta_i^k = \frac{1}{\rho_i} \eta_i^k \tilde{\mathbf{g}}_i^k \tag{31}$$

In the initial stages of training, accumulated norm of the per-parameter gradients can be less than 1. If the accumulated per-parameter norm of a gradient is less than 1, Adagrad will augment the learning-rate determined by Adasecant for that update, i.e. $\frac{\eta_i^k}{\rho_i^k} > \eta_i^k$ where $\eta_i^k = \mathrm{E}_k[t_i^k]$ is the per-parameter learning rate determined by Adasecant. This behavior tends to create unstabilities during the training with Adasecant. Our modification of the Adagrad algorithm is to ensure that, it will reduce the learning rate determined by the Adasecant algorithm at each update, i.e. $\frac{\eta_i^k}{\rho_i^k} \leq \eta_i^k$ and the learning rate will be bounded. In the beginning of training, parameters of a neural network can get 0-valued gradients, e.g. in the existence of dropout and ReLU units. However this phenomena can cause the per-parameter learning rate scaled by Adagrad to be unbounded.

In Algorithm 1, we provide a simple pseudo-code of the Adasecant algorithm.

Algorithm 1: Adasecant: minibatch-Adasecant for adaptive learning rates with variance reduction

repeat

draw n samples, compute the gradients $\mathbf{g}^{(j)}$ where $\mathbf{g}^{(j)} \in \mathcal{R}^n$ for each minibatch j, $\mathbf{g}^{(j)}$ is computed as, $\frac{1}{n}\sum_{k=1}^{n} \nabla_{\boldsymbol{\theta}}^{(k)} \mathrm{f}(\boldsymbol{\theta})$

block-wise normalize gradients of each weight matrix and bias vector

for *parameter* $i \in \{1, \ldots, n\}$ **do**

compute the correction term by using, $\gamma_i^k = \frac{\mathrm{E}[(g_i - g_i')(g_i - \mathrm{E}[g_i])]_k}{\mathrm{E}[(g_i - \mathrm{E}[g_i])(g_i' - \mathrm{E}[g_i]))]_k}$

compute corrected gradients $\tilde{g}_i = \frac{g_i + \gamma_i \mathrm{E}[g_i]}{1 + \gamma_i}$

if $|g_i^{(j)} - \mathrm{E}[g_i]| > 2\sqrt{\mathrm{E}[(g_i)^2] - (\mathrm{E}[g_i])^2}$ or $\left|\alpha_i^{(j)} - \mathrm{E}[\alpha_i]\right| > 2\sqrt{\mathrm{E}[(\alpha_i)^2] - (\mathrm{E}[\alpha_i])^2}$ **then**

reset the memory size for outliers $\tau_i \leftarrow 2.2$

end

update moving averages according to Equation 26

estimate learning rate $\eta_i^{(j)} \leftarrow \frac{\sqrt{\mathrm{E}_k[(\Delta_i^{(k)})^2]}}{\sqrt{\mathrm{E}_k[(\alpha_i^k)^2]}} - \frac{\mathrm{E}_k[\alpha_i^k \Delta_i^k]}{\mathrm{E}_k[(\alpha_i^k)^2]}$

update memory size as in Equation 28

update parameter $\theta_i^j \leftarrow \theta_i^{j-1} - \eta_i^{(j)} \cdot \tilde{g}_i^{(j)}$

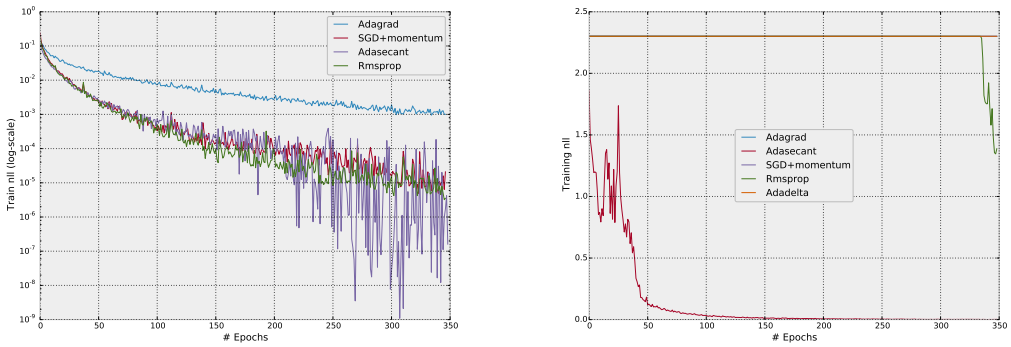
end

until *stopping criterion is met*;

7 EXPERIMENTS

We have performed our experiments on MNIST with Maxout Networks (Goodfellow *et al.*, 2013a).

We have compared our algorithm with other popular stochastic gradient learning algorithms have been tested as well, Adagrad, Rmsprop (Graves, 2013), Adadelta (Zeiler, 2012) and SGD+momentum (linearly decaying learning rate). Results are summarized in Figure 1 and show that Adasecant converges as fast or faster than other techniques, including the use of hand-tuned global learning rate and momentum for SGD, RMSprop, and Adagrad.



(a) 2 layer Maxout Network

(b) 16 layer Maxout Network

Figure 1: Comparison of different stochastic gradient algorithms on MNIST with Maxout Networks. Both a) and b) are trained with dropout and maximum column norm constraint regularization on the weights. Networks are initialized with weights sampled from a Gaussian distribution with 0 mean and standard deviation of 0.05. In both experiments, the proposed algorithm, Adasecant, seems to be converging faster and arrives to a better minima in training set. We trained both networks for 350 epochs over the training set.

8 CONCLUSION

We provided a new stochastic gradient algorithm with adaptive learning rates that is fairly insensitive to the tuning of the hyper-parameters and doesn't require tuning of learning rates. Furthermore, the variance reduction technique we provided for SGD improves the convergence in the settings where stochastic gradients have high variance. We provided a set of preliminary results with deep neural networks. According to our preliminary experiments, we were able to obtain better training performance compared to other well-tuned popular stochastic gradient algorithms. As a future work, we should do a more comprehensive analysis, which will help us to better understand the algorithm both analytically and empirically.

ACKNOWLEDGMENTS

We thank the developers of Theano (Bastien *et al.*, 2012) and Pylearn2 (Goodfellow *et al.*, 2013b) and the computational resources provided by Compute Canada and Calcul Québec. This work has been partially supported by NSERC, CIFAR, and Canada Research Chairs, Project TIN2013-41751, grant 2014-SGR-221. We would like to thank Tom Schaul for the valuable discussions. We also thank Kyunghyun Cho and Orhan Firat for proof-reading and giving feedbacks on the paper.

REFERENCES

An, H.-B. and Bai, Z.-Z. (2005). Directional secant method for nonlinear equations. *Journal of computational and applied mathematics*, **175**(2), 291–304.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements.

Becker, S. and Le Cun, Y. (1988). Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, pages 29–37. San Matteo, CA: Morgan Kaufmann.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, **12**, 2121–2159.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013a). Maxout networks. *arXiv preprint arXiv:1302.4389*.

Goodfellow, I. J., Warde-Farley, D., Lamblin, P., Dumoulin, V., Mirza, M., Pascanu, R., Bergstra, J., Bastien, F., and Bengio, Y. (2013b). Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323.

LeCun, Y., Simard, P. Y., and Pearlmutter, B. (1993). Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. *Advances in neural information processing systems*, **5**, 156–163.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Levin, Y. and Ben-Israel, A. (2002). Directional newton methods in n variables. *Mathematics of Computation*, **71**(237), 251–262.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Schaul, T. and LeCun, Y. (2013). Adaptive learning rates and parallelization for stochastic, sparse, non-smooth gradients. *arXiv preprint arXiv:1301.3764*.

Schaul, T., Zhang, S., and LeCun, Y. (2012). No more pesky learning rates. *arXiv preprint arXiv:1206.1106*.

Schraudolph, N. N. (2002). Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, **14**(7), 1723–1738.

Wang, C., Chen, X., Smola, A., and Xing, E. (2013). Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pages 181–189.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

A APPENDIX

A.1 EXPERIMENTAL DETAILS

In our experiments with Adasecant algorithm, adaptive momentum term γ_i^k was clipped at 1.8. In 2-layer Maxout network experiments for SGD-momentum experiments, we used the best hyper-parameters reported by Goodfellow *et al.* (2013a), for Rmsprop and Adagrad, we crossvalidated learning rate for 15 different learning rates sampled uniformly from the log-space. We crossvalidated 30 different pairs of momentum and learning rate for SGD+momentum, for Rmsprop and Adagrad, we crossvalidated 15 different learning rates sampled them from log-space uniformly for deep maxout experiments.