Stochastic Gradient Descent: Going As Fast As Possible But Not Faster

Alice Schoenauer Sebag alice.schoenauersebag@ucsf.edu
Altschuler&Wu lab, Dpt of Pharm. Chem.
UCSF, San Francisco, CA 94158
Marc Schoenauer marc.schoenauer@inria.fr
INRIA-CNRS-UPSud-UPSay
TAU, U. Paris-Sud, 91405 Orsay
Michèle Sebag michele.sebag@lri.fr
CNRS-UPSud-INRIA-UPSay
TAU, U. Paris-Sud, 91405 Orsay

Abstract

When applied to training deep neural networks, stochastic gradient descent (SGD) often incurs steady progression phases, interrupted by catastrophic episodes in which loss and gradient norm explode. A possible mitigation of such events is to slow down the learning process.

This paper presents a novel approach, called SALERA, to control the SGD learning rate, that uses two statistical tests. The first one, aimed at fast learning, compares the momentum of the normalized gradient vectors to that of random unit vectors and accordingly gracefully increases or decreases the learning rate. The second one is a change point detection test, aimed at the detection of catastrophic learning episodes; upon its triggering the learning rate is instantly halved.

Experiments on standard benchmarks show that SALERA performs well in practice, and compares favorably to the state of the art.

Machine Learning (ML) algorithms require efficient optimization techniques, whether to solve convex problems (e.g., for SVMs), or non-convex ones (e.g., for Neural Networks). As the data size and the model dimensionality increase, mainstream convex optimization methods are adversely affected. Overall, Stochastic Gradient Descent (SGD) is increasingly adopted.

Within the SGD framework, one of the main issues is to know how to control the learning rate.The adequate speed depends both on the current state of the system (the weight vector) and the current mini-batch. Often, the eventual convergence of SGD is ensured by decaying the learning rate as $O(t)$ [23, 6] or $O(\sqrt{t})$ [29]. While learning rate decay effectively prevents catastrophic events (sudden rocketing of the training loss and gradient norm), many and diverse approaches have been designed to achieve quicker convergence through learning rate adaptation [1, 7, 24, 16, 26, 2] (more in Section 1).

This paper proposes a novel approach to adaptive SGD, called SALERA (*Safe Agnostic LEarning Rate Adaptation*), based on the conjecture that, if catastrophes are well taken care of, the learning process can speed up whenever successive gradient directions show more correlation than random.
The frequent advent of catastrophic episodes [11, Chapter 8], [3] raises the question of how to best mitigate their impact. Framing catastrophic episodes as random events,we adopt a purely curative strategy: detecting and instantly curing catastrophic episodes. Formally, a sequential cumulative sum change detection test, the Page-Hinkley (PH) test [20, 14] is adapted and used to monitor the learning curve reporting the minibatch losses. If a change in the learning curve is detected, the system undergoes an instant cure by halving the learning rate and backtracking to its former state.Once the risk of catastrophic episodes is well addressed, the learning rate can be adapted in a more agile manner: the ALERA (*Agnostic LEarning Rate Adaptation*) process increases (resp. decreases) the learning rate whenever the correlation among successive gradient directions is higher (resp. lower) than random, by comparing the actual gradient momentum and the agnostic momentum built from random unit vectors.

The contribution of the paper is twofold. First, it proposes an original and efficient way to control learning dynamics (section 2.1). Secondly, it opens a new approach for handling catastrophic events and salvaging a

OPTML 2017: 10th NIPS Workshop on *Optimization for Machine Learning* (NIPS 2017).

significant part of doomed-to-fail runs (section 2.2). The experimental validation thereof compares favorably with the state of the art on the MNIST and CIFAR-10 benchmarks (section 3).

1 Related work

SGD was revived in the last decade as an effective method for training deep neural networks with linear computational complexity in the size of the dataset [4, 13]. SGD faces two limitations, depending on the learning rate: too large, and the learning trajectory leads to catastrophic episodes; too small, and its convergence takes ages.

Dealing with catastrophic events in deep learning The exploding gradient problem, described in [11, Chapter 8] as the encounter of steep cliff structures in the derivative landscape during learning, is frequently met while training neural networks (and even more so when training recurrent neural networks [3]).
When it comes to dealing with such events, most published work focuses on creating the conditions so that they do not arise. Among the possibilities are the use of regularizations, e.g., L_1 or L_2 regularization [21]. Gradient clipping [21], is another possibility. The introduction of batch normalization [15] also helps diminishing the frequency of such events. Finally, proper initialization [10, 25] or unsupervised pre-training [8], i.e. initializing the optimization trajectory in a good region of the parameter space, also diminish the frequency of such events.

Learning rate adaptation The dynamic adjustment of the learning rate was acknowledged a key issue since the early SGD days (see [9] for a review).

Learning rate decay in $O(t)$ was indeed first proposed in [23]. More recently, ADAGRAD [7] uses the information of past gradients to precondition each update in a parameter-wise manner, dividing the learning rate by the sum of squared past gradients. Several approaches have been proposed to mitigate ADAGRADś learning rate decay to 0, including ADADELTA [28], RMSProp [26], and ADAM [16]. ADAM is based on estimating the first and second moments of the gradient w.r.t. each parameter, and using their ratio to update the parameters. SALERA also builds upon the use of the gradient second moment, with the difference that it is compared with a fixed agnostic counterpart.

More remotely related are the momentum approaches, both its classic [22] and Nesterov versions [19] as derived by [25], that rely on a relaxed sum of past gradients for indicating a more robust descent direction than the current gradient.

2 SALERA

SALERA involves two components: a learning rate adaptation scheme, which ensures that the learning system goes as fast as it can; and a catastrophic event manager, which is in charge of detecting undesirable behaviors and getting the system back on track.

2.1 Agnostic learning rate adaptation

Rationale The basic idea of the proposed learning rate update is to compare the current gradient descent to a random walk with uniformly chosen gradient directions. Indeed, the sum of successive normalized gradient vectors (hereafter cumulative path) has a larger norm than the sum of uniformly drawn unit vectors if and only if gradient directions are positively correlated. In such cases, the learning process has a global direction and the process can afford to speed up. On the opposite, if gradient directions are less correlated than that of random vectors, the cumulative path norm is smaller than its random equivalent and the learning rate should be decreased. The ALERA scheme takes inspiration from the famed CMA-ES [12] and NES [27] algorithms, today considered among the best-performing derivative-free continuous optimization algorithms.

The ALERA algorithm The partial adaptation of the CMA-ES scheme to the minimization of a loss function $\mathcal{L}$ on a d-dimensional parameter space is defined as follows. Let θ_t be the solution at time t, $g_t = \nabla_\theta \mathcal{L}(\theta_t)$ the gradient of the current loss, and η_t the current learning rate. SGD computes the solution at time $t+1$ by $\theta_{t+1} = \theta_t - \eta_t g_t$. Let $\|.\|$ denote the L_2 norm and $< ., . >$ the associated dot product.

Definition. *For $t > 0$, the exponential moving average of the normalized gradients with weight α, p_t, and its random equivalent r_t are defined as:*

$$p_0 = 0; \;\; \mathrm{r}_0 = 0 \tag{1}$$

$$p_t = \alpha \, g_t / \|g_t\| + (1 - \alpha) \, p_{t-1} \tag{2}$$

$$\mathrm{r}_t = \alpha \, u_t + (1 - \alpha) \, \mathrm{r}_{t-1} \tag{3}$$

where $(u_t)_{t>0}$ are independent random unit vectors in $\mathbb{R}^d$.

Proposition. *For $t > 0$, the expectation $\mu(t)$ and variance $\sigma^2(d,t)$ of $\|r_t\|^2$ as defined above are:*

$$\mu(t) = \mathbb{E}(\|\mathrm{r}_t\|^2) = \frac{\alpha}{2-\alpha}[1 - (1-\alpha)^{2t}] \tag{4}$$

$$\sigma^2(d,t) = Var(\|\mathrm{r}_t\|^2) = \frac{1}{d} \frac{2\alpha^2(1-\alpha)^2}{(2-\alpha)^2((1-\alpha)^2+1)}[1 - (1-\alpha)^t][1 - (1-\alpha)^{t-1}] \tag{5}$$

Proof: Appendix A.

Let μ and $\sigma^2(d)$ denote the limits of respectively $\mu(t)$ and $\sigma^2(d,t)$ as $t \to +\infty$. At time step t, the ALERA scheme updates the cumulative path p_t by comparing its norm with the distribution of the agnostic momentum defined from μ and $\sigma^2(d)$. The learning rate is increased or decreased depending on the normalized gap between the squared norm of p_t and r_t:

$$\eta_{t+1} = \eta_t \exp\left(C \frac{\|p_t\|^2 - \mu}{\sigma(d)}\right) \tag{6}$$

with C a hyper-parameter of the algorithm.

2.2 Catastrophic event management

As said, the ability to learn fast requires an emergency procedure, able both to detect an emergency and to recover from it. The PH detection test [20, 14] is chosen as it provides optimal guarantees about the trade-off between the detection delay upon a change (affecting the average or standard deviation of the signal) and the mean time between false alarms. For $t > 0$, it maintains the empirical mean $\bar{\ell}_t$ of the signal, and the cumulative deviation L_t from the empirical mean ($L_t = L_{t-1} + (\ell_t - \bar{\ell}_t)$). Finally, it records the empirical bounds of L_t ($L_{min} = min_t L_t$; $L_{max} = max_t L_t$). In case of a stationary signal, the expectation of L_t is 0 by construction; the PH change test is thus triggered when the gap between L_t and its empirical bounds is higher than a problem-dependent threshold Δ, which controls the alarm rate.
The PH test is implemented in the SALERA algorithm as follows. Δ is set to $\mathcal{L}(\theta, \text{first mini-batch})/\lambda$, with $\lambda = 10$ as it was found to be optimal on the datasets and architectures considered. Variables $\bar{\ell}_t$, L and L_{min} are maintained. Upon test triggering ($L_{min} - L > \Delta$), the learning rate is halved and the weight vector θ is reset to the last solution before then, and the PH test is reinitialized. See in Appendix B the full derivation of the algorithm.

3 Experiments

Datasets and metrics All experiments are performed on the **MNIST** [18] and **CIFAR-10** [17] datasets. These are classification problems, for which we report classification accuracy on the test set at 20 epochs (end of all runs), as well as their standard deviations on the 5 independent runs.

Algorithms **Adagrad**, **NAG**, and **Adam** are used as baselines. The agnostic adaptation rule and the change detection can be applied independently. In order to separate their effect, 3 original algorithms are studied here: **ALeRA** implements the agnostic learning rate adaptation without the change detection; **Ag-Adam** uses the same agnostic adaptation for the learning rate on top of the ADAM algorithm. Finally, the change detection mechanism is implemented with the agnostic adaptation, yielding **SALeRA** as described.

Network Models All experiments consider the following 4 network architectures, where the figure indicates the number of hidden layers: **M0**: a softmax regression model with cross-entropy loss, **M2**: 2 fully connected hidden layers with ReLU activation, on top of M0, **M2b** which is M2 with Batch Normalization layers [15] added in each hidden layer, and finally **M4b**, a LeNet5-inspired convolutional models [18]. The code of the algorithms and models, which was written using the Torch library [5], is available on GitHub.

Table 1: Best performances (test error over all tested parameter settings) at 20 epochs of NAG, ADAGRAD, ADAM, AG-ADAM, ALERA and SALERA (average and standard deviation on 5 runs). For each line, the best results are in bold, and results less than 1 std.dev. away are in italics.

			NAG	Adagrad	ADAM	AG-ADAM	ALERA	SALERA
MNIST	M0	20ep.	7.59 (.07)	7.51 (.09)	7.43 (.10)	**7.29 (.09)**	7.43 (.03)	7.44 (.04)
	M2	20ep.	*1.58 (.08)*	1.71 (.08)	*1.56 (.06)*	*1.57 (.04)*	**1.55 (.10)**	*1.59 (.09)*
	M2b	20ep.	*1.47 (.10)*	1.48 (.06)	1.57 (.94)	1.53 (.05)	**1.43 (.04)**	1.48 (.09)
	M4b	20ep.	.72 (.09)	.82 (.08)	.80 (.08)	.79 (.05)	**.63 (.05)**	*.64 (.07)*
CIFAR	M0	20ep.	59.73 (.19)	59.76 (.36)	59.81 (.24)	*59.34 (.24)*	**59.31 (.11)**	**59.31 (.25)**
	M2	20ep.	45.08 (.32)	43.59 (.51)	44.43 (.50)	43.25 (.40)	43.19 (.21)	**42.72 (.41)**
	M2b	20ep.	42.50 (.48)	43.79 (.25)	43.60 (.64)	42.72 (.33)	**42.12 (.15)**	42.50 (.29)
	M4b	20ep.	27.45 (.39)	29.15 (.67)	27.84 (.59)	**25.30 (.18)**	26.35 (.64)	25.93 (.64)

The experimental evidence (Table 1) shows that, on the one hand, AG-ADAM quite often slightly but statistically significantly improves on ADAM. On the other hand, in many cases, ALERA and SALERA yield similar results; indeed, whenever ALERA does not meet catastrophic episodes, ALERA and SALERA have the same behaviors. The advantage of catastrophe management is therefore to diminish the need for hyper-parameter tuning. In order to determine to which extent the best results in Table 1 depend on the hyper-parameter settings a sensitivity analysis was performed. The proposed hyper-parameters for ALERA and SALERA are found to be $\alpha = .01$ and $C = 3.10^{-6}$. Finally, if a failed run is one which attains more than 80% test error after 20 epochs, ALERA is observed to have 18.3% failed runs over the chosen parameter range. Our catastrophe management scheme makes it possible for SALERA to avoid approximately 40% of these failures, reaching a rate of failure of 11.7% on the same parameter range.

4 Discussion

The first proposed contribution relies on the comparison of the gradient momentum with a fixed reference. It is meant to estimate the overall correlation among the sequence of gradients, which can be thought of as a signal-to-noise ratio in the process generated from the current solution, the objective and the successive mini-batches. Depending on this ratio, the process can be accelerated or slowed down. The ALERA procedure, which implements this idea, proves to be significantly able to increase and decrease the learning rate. Furthermore, this process can be plugged on ADAM, with a performance improvement on average.

This however increases the risk of catastrophic episodes, with instant rocketing of the training loss and gradient norm. The proposed approach relies on the fact that, on the one hand catastrophic episodes can be rigorously detected, and that on the other hand, a neural net optimizer is almost doomed to face such episodes along the optimization process, which are mostly detrimental to learning. Rather than choosing to slow convergence to avoid them, the second contribution of the paper is an agnostic and principled way to detect and address such episodes, which relies on the Page-Hinkley change point detection test. As soon as an event is detected, learning rates are halved and the previous solution recovered.

A short-term perspective for further research is to apply the proposed approach to recurrent neural networks, and to consider more complex datasets. Another perspective is to replace the halving trick by approximating the line search, e.g. by exploiting the gaps between the actual momentum and the reference one, for several values of the momentum weight factor. A third perspective regards the adaptation of the PH detection threshold Δ during learning.The goal is to adapt Δ when the PH mechanism is re-initialized (line 9 of Algorithm 1) from the current loss values.

References

[1] S.-I. Amari. Natural gradient works efficiently in learning. *Neural Comput.*, 10(2):251–276, Feb. 1998. ISSN 0899-7667. doi: 10.1162/089976698300017746. URL http://dx.doi.org/10.1162/089976698300017746.

[2] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *NIPS 16*, pages 3981–3989, 2016.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, Mar. 1994. ISSN 1045-9227. doi: 10.1109/72.279181. URL http://dx.doi.org/10.1109/72.279181.

[4] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. NIPS Foundation (http://books.nips.cc), 2008. URL http://leon.bottou.org/papers/bottou-bousquet-2008.

[5] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[6] A. Defazio, F. R. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS14*, pages 1646–1654, 2014.

[7] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-24.html.

[8] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, Mar. 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1756025.

[9] A. P. George and W. B. Powell. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Mach. Learn.*, 65(1):167–198, Oct. 2006. ISSN 0885-6125. doi: 10.1007/s10994-006-8365-9. URL http://dx.doi.org/10.1007/s10994-006-8365-9.

[10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256, 2010. URL http://www.jmlr.org/proceedings/papers/v9/glorot10a.html.

[11] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[13] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. *CoRR*, abs/1509.01240, 2015. URL http://arxiv.org/abs/1509.01240.

[14] D. Hinkley. Inference about the change point from cumulative sum-tests. *Biometrika*, 58:509–523, 1970.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015. URL http://jmlr.org/proceedings/papers/v37/ioffe15.html.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://dblp.uni-trier.de/db/journals/corr/corr1412.html#KingmaB14.

[17] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[18] Y. Le Cun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[19] Y. Nesterov. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376, 1983.

[20] E. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[21] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318. JMLR.org, 2013. URL http://dl.acm.org/citation.cfm?id=3042817.3043083.

[22] B. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[23] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22: 400–407, 1951.

[24] T. Schaul, S. Zhang, and Y. Le Cun. No More Pesky Learning Rates. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, volume 28, pages 343–351, 2013.

[25] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, volume 28 of *ICML'13*, pages III–1139–III–1147. JMLR.org, 2013. URL http://dl.acm.org/citation.cfm?id=3042817.3043064.

[26] T. Tieleman and G. Hinton. Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. Technical report, Technical report, 2012.

[27] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *J. Mach. Learn. Res.*, 15(1):949–980, 2014. ISSN 1532-4435.

[28] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

[29] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 928–935. AAAI Press, 2003. ISBN 1-57735-189-4. URL http://dl.acm.org/citation.cfm?id=3041838.3041955.

Appendix A: Derivation of formula 4

Let us define $\mathrm{r}_0 = 0$, and for $t > 0$

$$\mathrm{r}_t = \alpha u_t + (1-\alpha)\mathrm{r}_{t-1}$$

with u_t a unit vector uniformly drawn in $\mathbb{R}^d$. We have by recurrence for $t > 0$

$$\mathrm{r}_t = \alpha \sum_{l=1}^{t} (1-\alpha)^{(t-l)} u_l$$

Thus

$$\|\mathrm{r}_t\|^2 = \alpha^2 \left(\sum_{l=1}^{t} (1-\alpha)^{2(t-l)} \|u_l\|^2 + \sum_{l \neq k} (1-\alpha)^{2t-l-k} < u_l, u_k > \right) \tag{7}$$

We are dealing with unit vectors which are uniformly drawn. Thus with δ denoting the Kronecker delta

$$\begin{aligned} \forall\, l > 0 \;\; & \|u_l\|^2 = 1 \\ \forall\, l, k > 0 \;\; & \mathbb{E}(< u_l, u_k >) = \delta_{lk} \end{aligned}$$

Taking the expectation in 7 we have

$$\mathbb{E}(\|\mathrm{r}_t\|^2) = \alpha^2 \sum_{l=1}^{t} (1-\alpha)^{2(t-l)} \tag{8}$$

that is,

$$\mathbb{E}(\|\mathrm{r}_t\|^2) = \alpha^2 \frac{1-(1-\alpha)^{2t}}{1-(1-\alpha)^2} \tag{9}$$

$$= \frac{\alpha}{2-\alpha} \left(1-(1-\alpha)^{2t}\right) \tag{10}$$

The derivation of formula 5 is similar.

Appendix B: algorithm of Safe Agnostic LEarning Rate Adaptation

Algorithm 1: S-ALERA: *Agnostic LEarning Rate Adaptation* and Page-Hinkley change detection

Input: Model with loss function $\mathcal{L}$
Parameters : Memory rate α, factor C //algorithm parameters
Initial learning rate η_0, mini-batch ratio ρ // run parameters
Initialize : $t \leftarrow 0; p \leftarrow 0; \eta \leftarrow \eta_0$; init$(\theta)$ // initialization
$L, L_{min}, \ell, \bar{\ell}, \leftarrow 0$; $\Delta \leftarrow \mathcal{L}(\theta, \text{first mini-batch})/10$ // initialize Page-Hinkley

while *stopping criterion not met* **do**
  $MB \leftarrow$ new mini-batch ; $t \leftarrow t + 1$ // perform forward pass
  $\ell = \rho\mathcal{L}(\theta, \text{MB}) + (1 - \rho)\ell$
  $\bar{\ell} \leftarrow (\ell + t\bar{\ell})/(t + 1)$ // empirical mean of batch losses
  $L \leftarrow L + (\ell - \bar{\ell})$ // cumulated deviations from mean
  $L_{min} \leftarrow min(L_{min}, L)$ // lower bound of deviations
  if $L - L_{min} > \Delta$ **then**
    $\theta \leftarrow \theta^{(b)}; \eta \leftarrow \eta/2$ // Page-Hinkley triggered: backtrack
    $L, L_{min}, \bar{\ell}, \ell \leftarrow 0; t \leftarrow 0$ // and re-initialize Page-Hinkley[1]
  else
    $\theta^{(b)} \leftarrow \theta$ // save for possible backtracks
    $g \leftarrow \nabla_\theta \mathcal{L}(\theta, \text{MB})$ // compute gradient with backward pass
    $p \leftarrow \alpha g/\|g\| + (1 - \alpha)p$ // exponential moving average of normalized gradients
    $\eta \leftarrow \eta \exp\left(C(\|p\|_2^2 - \mu)/\sigma(d)\right)$ // agnostic learning rate update
    $\theta \leftarrow \theta - \eta g$ // standard parameter update
  end
end