WIKIPEDIA

Stochastic gradient descent

Stochastic gradient descent (often abbreviated **SGD**) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). Especially in high-dimensional optimization problems this reduces the very high computational burden, achieving faster iterations in trade for a lower convergence rate.[1]

While the basic idea behind stochastic approximation can be traced back to the Robbins–Monro algorithm of the 1950s, stochastic gradient descent has become an important optimization method in machine learning.[2]

Contents

Background

Both statistical estimation and machine learning consider the problem of minimizing an objective function that has the form of a sum:

$$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w),$$

where the parameter w that minimizes $Q(w)$ is to be estimated. Each summand function Q_i is typically associated with the i-th observation in the data set (used for training).

In classical statistics, sum-minimization problems arise in least squares and in maximum-likelihood estimation (for independent observations). The general class of estimators that arise as minimizers of sums are called M-estimators. However, in statistics, it has been long recognized that requiring even local minimization is too restrictive for some problems of maximum-likelihood estimation.[3] Therefore, contemporary statistical theorists often consider stationary points of the likelihood function (or zeros of its derivative, the score function, and other estimating equations).

The sum-minimization problem also arises for empirical risk minimization. In this case, $Q_i(w)$ is the value of the loss function at i-th example, and $Q(w)$ is the empirical risk.

When used to minimize the above function, a standard (or "batch") gradient descent method would perform the following iterations:

$$w := w - \eta \nabla Q(w) = w - \frac{\eta}{n} \sum_{i=1}^{n} \nabla Q_i(w),$$

where η is a step size (sometimes called the *learning rate* in machine learning).

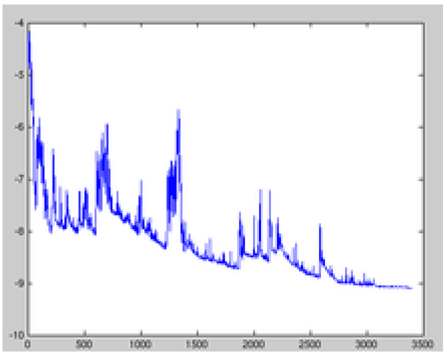
In many cases, the summand functions have a simple form that enables inexpensive evaluations of the sum-function and the sum gradient. For example, in statistics, one-parameter exponential families allow economical function-evaluations and gradient-evaluations.

However, in other cases, evaluating the sum-gradient may require expensive evaluations of the gradients from all summand functions. When the training set is enormous and no simple formulas exist, evaluating the sums of gradients becomes very expensive, because evaluating the gradient requires evaluating all the summand functions' gradients. To economize on the computational cost at every iteration, stochastic gradient descent samples a subset of summand functions at every step. This is very effective in the case of large-scale machine learning problems.[4]

Iterative method

In stochastic (or "on-line") gradient descent, the true gradient of $Q(w)$ is approximated by a gradient at a single sample:

$$w := w - \eta \nabla Q_i(w).$$



Fluctuations in the total objective function as gradient steps with respect to mini-batches are taken.

As the algorithm sweeps through the training set, it performs the above update for each training sample. Several passes can be made over the training set until the algorithm converges. If this is done, the data can be shuffled for each pass to prevent cycles. Typical implementations may use an adaptive learning rate so that the algorithm converges.[5]

In pseudocode, stochastic gradient descent can be presented as :

```
- Choose an initial vector of parameters $w$ and learning rate $\eta$.
- Repeat until an approximate minimum is obtained:
    Randomly shuffle samples in the training set.
    For $i = 1, 2, \ldots, n$, do:
        $w := w - \eta \nabla Q_i(w).$
```

A compromise between computing the true gradient and the gradient at a single sample is to compute the gradient against more than one training sample (called a "mini-batch") at each step. This can perform significantly better than "true" stochastic gradient descent described, because the code can make use of vectorization libraries rather than computing each step separately as was first shown in [6] where it was called "the bunch-mode back-propagation algorithm". It may also result in smoother convergence, as the gradient computed at each step is averaged over more training sample.

The convergence of stochastic gradient descent has been analyzed using the theories of convex minimization and of stochastic approximation. Briefly, when the learning rates η decrease with an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to a global minimum when the objective function is convex or pseudoconvex, and otherwise converges almost surely to a local minimum.[7][8] This is in fact a consequence of the Robbins–Siegmund theorem.[9]

Example

Let's suppose we want to fit a straight line $\hat{y} = w_1 + w_2 x$ to a training set with observations $(x_1, x_2, \ldots, x_n)$ and corresponding estimated responses $(\hat{y_1}, \hat{y_2}, \ldots, \hat{y_n})$ using least squares. The objective function to be minimized is:

$$Q(w) = \sum_{i=1}^{n} Q_i(w) = \sum_{i=1}^{n} (\hat{y_i} - y_i)^2 = \sum_{i=1}^{n} (w_1 + w_2 x_i - y_i)^2.$$

The last line in the above pseudocode for this specific problem will become:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} := \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial}{\partial w_1}(w_1 + w_2 x_i - y_i)^2 \\ \frac{\partial}{\partial w_2}(w_1 + w_2 x_i - y_i)^2 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \eta \begin{bmatrix} 2(w_1 + w_2 x_i - y_i) \\ 2x_i(w_1 + w_2 x_i - y_i) \end{bmatrix}.$$

Note that in each iteration (also called update), the gradient is only evaluated at a single point x_i instead of at the set of all samples.

The key difference compared to standard (Batch) Gradient Descent is that only one piece of data from the dataset is used to calculate the step, and the piece of data is picked randomly at each step.

Notable applications

Stochastic gradient descent is a popular algorithm for training a wide range of models in machine learning, including (linear) support vector machines, logistic regression (see, e.g., Vowpal Wabbit) and graphical models.[10] When combined with the backpropagation algorithm, it is the *de facto* standard algorithm for training artificial neural networks.[11] Its use has been also reported in the Geophysics community, specifically to applications of Full Waveform Inversion (FWI).[12]

Stochastic gradient descent competes with the L-BFGS algorithm, which is also widely used. Stochastic gradient descent has been used since at least 1960 for training linear regression models, originally under the name ADALINE.[13]

Another stochastic gradient descent algorithm is the least mean squares (LMS) adaptive filter.

Extensions and variants

Many improvements on the basic stochastic gradient descent algorithm have been proposed and used. In particular, in machine learning, the need to set a learning rate (step size) has been recognized as problematic. Setting this parameter too high can cause the algorithm to diverge; setting it too low makes it slow to converge.[14] A conceptually simple extension of stochastic gradient descent makes the learning rate a decreasing function η_t of the iteration number t, giving a *learning rate schedule*, so that the first iterations cause large changes in the parameters, while the later ones do only fine-tuning. Such schedules have been known since the work of MacQueen on k-means clustering.[15] Practical guidance on choosing the step size in several variants of SGD is given by Spall.[16]

Implicit updates (ISGD)

As mentioned earlier, classical stochastic gradient descent is generally sensitive to learning rate η. Fast convergence requires large learning rates but this may induce numerical instability. The problem can be largely solved[17] by considering *implicit updates* whereby the stochastic gradient is evaluated at the next iterate rather than the current one:

$$w^{\text{new}} := w^{\text{old}} - \eta \nabla Q_i(w^{\text{new}}).$$

This equation is implicit since w^{new} appears on both sides of the equation. It is a stochastic form of the proximal gradient method since the update can also be written as:

$$w^{\text{new}} := \arg\min_w \{Q_i(w) + \frac{1}{2\eta}||w - w^{\text{old}}||^2\}.$$

As an example, consider least squares with features $x_1, \ldots, x_n \in \mathbb{R}^p$ and observations $y_1, \ldots, y_n \in \mathbb{R}$. We wish to solve:

$$\min_w \sum_{j=1}^{n} (y_j - x_j' w)^2,$$

where $x_j' w = x_{j1} w_1 + x_{j,2} w_2 + \ldots + x_{j,p} w_p$ indicates the inner product. Note that x could have "1" as the first element to include an intercept. Classical stochastic gradient descent proceeds as follows:

$$w^{\text{new}} = w^{\text{old}} + \eta (y_i - x_i' w^{\text{old}}) x_i$$

where i is uniformly sampled between 1 and n. Although theoretical convergence of this procedure happens under relatively mild assumptions, in practice the procedure can be quite unstable. In particular, when η is misspecified so that $I - \eta x_i x_i'$ has large absolute eigenvalues with high probability, the procedure may diverge numerically within a few iterations. In contrast, *implicit stochastic gradient descent* (shortened as ISGD) can be solved in closed-form as:

$$w^{\text{new}} = w^{\text{old}} + \frac{\eta}{1+\eta||x_i||^2}(y_i - x_i' w^{\text{old}})x_i.$$

This procedure will remain numerically stable virtually for all η as the learning rate is now normalized. Such comparison between classical and implicit stochastic gradient descent in the least squares problem is very similar to the comparison between least mean squares (LMS) and normalized least mean squares filter (NLMS).

Even though a closed-form solution for ISGD is only possible in least squares, the procedure can be efficiently implemented in a wide range of models. Specifically, suppose that $Q_i(w)$ depends on w only through a linear combination with features x_i, so that we can write $\nabla_w Q_i(w) = -q(x_i' w)x_i$, where $q() \in \mathbb{R}$ may depend on x_i, y_i as well but not on w except through $x_i' w$. Least squares obeys this rule, and so does logistic regression, and most generalized linear models. For instance, in least squares, $q(x_i' w) = y_i - x_i' w$, and in logistic regression $q(x_i' w) = y_i - S(x_i' w)$, where $S(u) = e^u/(1+e^u)$ is the logistic function. In Poisson regression, $q(x_i' w) = y_i - e^{x_i' w}$, and so on.

In such settings, ISGD is simply implemented as follows. Let $f(\xi) = \eta q(x_i' w^{old} + \xi||x_i||^2)$, where ξ is scalar. Then, ISGD is equivalent to:

$$w^{\text{new}} = w^{\text{old}} + \xi^* x_i, \text{ where } \xi^* = f(\xi^*).$$

The scaling factor $\xi^* \in \mathbb{R}$ can be found through the bisection method since in most regular models, such as the aforementioned generalized linear models, function $q()$ is decreasing, and thus the search bounds for ξ^* are $[\min(0, f(0)), \max(0, f(0))]$.

Momentum

Further proposals include the *momentum method*, which appeared in Rumelhart, Hinton and Williams' paper on backpropagation learning.[18] Stochastic gradient descent with momentum remembers the update Δw at each iteration, and determines the next update as a linear combination of the gradient and the previous update:[19][20]

$$\Delta w := \alpha \Delta w - \eta \nabla Q_i(w)$$
$$w := w + \Delta w$$

that leads to:

$$w := w - \eta \nabla Q_i(w) + \alpha \Delta w$$

where the parameter w which minimizes $Q(w)$ is to be estimated, η is a step size (sometimes called the *learning rate* in machine learning) and α is an exponential decay factor between 0 and 1 that determines the relative contribution of the current gradient and earlier gradients to the weight change.

The name momentum stems from an analogy to momentum in physics: the weight vector w, thought of as a particle traveling through parameter space,[18] incurs acceleration from the gradient of the loss ("force"). Unlike in classical stochastic gradient descent, it tends to keep traveling in the same direction, preventing oscillations. Momentum has been used successfully by computer scientists in the training of artificial neural networks for several decades.[21] The *momentum method* is closely related to underdamped Langevin dynamics, and may be combined with Simulated Annealing. [22]

Averaging

Averaged stochastic gradient descent, invented independently by Ruppert and Polyak in the late 1980s, is ordinary stochastic gradient descent that records an average of its parameter vector over time. That is, the update is the same as for ordinary stochastic gradient descent, but the algorithm also keeps track of[23]

$$\bar{w} = \frac{1}{t}\sum_{i=0}^{t-1} w_i.$$

When optimization is done, this averaged parameter vector takes the place of w.

AdaGrad

AdaGrad (for adaptive gradient algorithm) is a modified stochastic gradient descent algorithm with per-parameter learning rate, first published in 2011.[24] Informally, this increases the learning rate for sparser parameters and decreases the learning rate for ones that are less sparse. This strategy often improves convergence performance over standard stochastic gradient descent in settings where data is sparse and sparse parameters are more informative. Examples of such applications include natural language processing and image recognition.[24] It still has a base learning rate η, but this is multiplied with the elements of a vector $\{G_{j,j}\}$ which is the diagonal of the outer product matrix

$$G = \sum_{\tau=1}^{t} g_\tau g_\tau^\mathsf{T}$$

where $g_\tau = \nabla Q_i(w)$, the gradient, at iteration τ. The diagonal is given by

$$G_{j,j} = \sum_{\tau=1}^{t} g_{\tau,j}^2.$$

This vector is updated after every iteration. The formula for an update is now

$$w := w - \eta\, \mathrm{diag}(G)^{-\frac{1}{2}} \odot g$$[a]

or, written as per-parameter updates,

$$w_j := w_j - \frac{\eta}{\sqrt{G_{j,j}}} g_j.$$

Each $\{G_{(i,i)}\}$ gives rise to a scaling factor for the learning rate that applies to a single parameter w_i. Since the denominator in this factor, $\sqrt{G_i} = \sqrt{\sum_{\tau=1}^{t} g_\tau^2}$ is the ℓ_2 norm of previous derivatives, extreme parameter updates get dampened, while parameters that get few or small updates receive higher learning rates.[21]

While designed for convex problems, AdaGrad has been successfully applied to non-convex optimization.[25]

RMSProp

RMSProp (for Root Mean Square Propagation) is also a method in which the learning rate is adapted for each of the parameters. The idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight.[26] So, first the running average is calculated in terms of means square,

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2$$

where, γ is the forgetting factor.

And the parameters are updated as,

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w)$$

RMSProp has shown good adaptation of learning rate in different applications. RMSProp can be seen as a generalization of Rprop and is capable to work with mini-batches as well opposed to only full-batches.[27]

Adam

Adam[28] (short for Adaptive Moment Estimation) is an update to the *RMSProp* optimizer. In this optimization algorithm, running averages of both the gradients and the second moments of the gradients are used. Given parameters $w^{(t)}$ and a loss function $L^{(t)}$, where t indexes the current training iteration (indexed at 0), Adam's parameter update is given by:

$$m_w^{(t+1)} \leftarrow \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w L^{(t)}$$
$$v_w^{(t+1)} \leftarrow \beta_2 v_w^{(t)} + (1-\beta_2)(\nabla_w L^{(t)})^2$$

$$\hat{m}_w = \frac{m_w^{(t+1)}}{1-\beta_1^t}$$
$$\hat{v}_w = \frac{v_w^{(t+1)}}{1-\beta_2^t}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}$$

where ϵ is a small scalar (e.g. 10^{-8}) used to prevent division by 0, and β_1 (e.g. 0.9) and β_2 (e.g. 0.999) are the forgetting factors for gradients and second moments of gradients, respectively. Squaring and square-rooting is done element-wise.

Backtracking line search

Backtracking line search is another variant of gradient descent. All of the below are sourced from the mentioned link. It is based on a condition known as the Armijo–Goldstein condition. Both methods allow learning rates to change at each iteration; however, the manner of the change is different. Backtracking line search uses function evaluations to check Armijo's condition, and in principle the loop in the algorithm for determining the learning rates can be long and unknown in advance. Adaptive SGD does not need a loop in determining learning rates. On the other hand, adaptive SGD does not guarantee the "descent property" – which Backtracking line search enjoys – which is that $f(x_{n+1}) \leq f(x_n)$ for all n. If the gradient of the cost function is globally Lipschitz continuous, with Lipschitz constant L, and learning rate is chosen of the order 1/L, then the standard version of SGD is a special case of backtracking line search.

Second-order methods

A stochastic analogue of the standard (deterministic) Newton–Raphson algorithm (a "second-order" method) provides an asymptotically optimal or near-optimal form of iterative optimization in the setting of stochastic approximation. A method that uses direct measurements of the Hessian matrices of the summands in the empirical risk function was developed by Byrd, Hansen, Nocedal, and Singer.[29] However, directly determining the required Hessian matrices for optimization may not be possible in practice. Practical and theoretically sound methods for second-order versions of SGD that do not require direct Hessian information are given by Spall and others.[30][31][32] (A less efficient method based on finite differences, instead of simultaneous perturbations, is given by Ruppert.[33]) These methods not requiring direct Hessian information are based on either values of the summands in the above empirical risk function or values of the gradients of the summands (i.e., the SGD inputs). In particular, second-order optimality is asymptotically achievable without direct calculation of the Hessian matrices of the summands in the empirical risk function.

Notes

a. $\odot$ is the element-wise product.

See also

- Backtracking line search
- Coordinate descent – changes one coordinate at a time, rather than one example
- Linear classifier
- Online machine learning
- Stochastic hill climbing
- Stochastic variance reduction

References

1. Bottou, Léon; Bousquet, Olivier (2012). "The Tradeoffs of Large Scale Learning" (https://www.google.com/books/edition/_/JPQx7s2L1A8C?hl=en&gbpv=1&pg=PA351). In Sra, Suvrit; Nowozin, Sebastian; Wright, Stephen J. (eds.). *Optimization for Machine Learning*. Cambridge: MIT Press. pp. 351–368. ISBN 978-0-262-01646-9.
2. Bottou, Léon (1998). "Online Algorithms and Stochastic Approximations". *Online Learning and Neural Networks* (https://archive.org/details/onlinelearningin0000unse). Cambridge University Press. ISBN 978-0-521-65263-6.

3. Ferguson, Thomas S. (1982). "An inconsistent maximum likelihood estimate". *Journal of the American Statistical Association*. **77** (380): 831–834. doi:10.1080/01621459.1982.10477894 (https://doi.org/10.1080%2F01621459.1982.10477894). JSTOR 2287314 (https://www.jstor.org/stable/2287314).
4. Bottou, Léon; Bousquet, Olivier (2008). *The Tradeoffs of Large Scale Learning* (http://leon.bottou.org/papers/bottou-bousquet-2008). Advances in Neural Information Processing Systems. Vol. 20. pp. 161–168.
5. Murphy, Kevin (2021). *Probabilistic Machine Learning: An Introduction* (https://probml.github.io/pml-book/book1.html). *Probabilistic Machine Learning: An Introduction*. MIT Press. Retrieved 10 April 2021.
6. Bilmes, Jeff; Asanovic, Krste; Chin, Chee-Whye; Demmel, James (April 1997). "Using PHiPAC to speed error back-propagation learning" (https://ieeexplore.ieee.org/document/604861). *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. ICASSP. Munich, Germany: IEEE. pp. 4153-4156 vol.5. doi:10.1109/ICASSP.1997.604861 (https://doi.org/10.1109%2FICASSP.1997.604861).
7. Bottou, Léon (1998). "Online Algorithms and Stochastic Approximations". *Online Learning and Neural Networks* (https://archive.org/details/onlinelearningin0000unse). Cambridge University Press. ISBN 978-0-521-65263-6.
8. Kiwiel, Krzysztof C. (2001). "Convergence and efficiency of subgradient methods for quasiconvex minimization". *Mathematical Programming, Series A*. Vol. 90, no. 1. Berlin, Heidelberg: Springer. pp. 1–25. doi:10.1007/PL00011414 (https://doi.org/10.1007%2FPL00011414). ISSN 0025-5610 (https://www.worldcat.org/issn/0025-5610). MR 1819784 (https://www.ams.org/mathscinet-getitem?mr=1819784).
9. Robbins, Herbert; Siegmund, David O. (1971). "A convergence theorem for non negative almost supermartingales and some applications". In Rustagi, Jagdish S. (ed.). *Optimizing Methods in Statistics*. Academic Press. ISBN 0-12-604550-X.
10. Jenny Rose Finkel, Alex Kleeman, Christopher D. Manning (2008). Efficient, Feature-based, Conditional Random Field Parsing (http://www.aclweb.org/anthology/P08-1109). Proc. Annual Meeting of the ACL.
11. LeCun, Yann A., et al. "Efficient backprop." Neural networks: Tricks of the trade. Springer Berlin Heidelberg, 2012. 9-48 (http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf)
12. Jerome R. Krebs, John E. Anderson, David Hinkley, Ramesh Neelamani, Sunwoong Lee, Anatoly Baumstein, and Martin-Daniel Lacasse, (2009), "Fast full-wavefield seismic inversion using encoded sources," GEOPHYSICS 74: WCC177-WCC188. (https://library.seg.org/doi/abs/10.1190/1.3230502)
13. Avi Pfeffer. "CS181 Lecture 5 — Perceptrons" (http://www.seas.harvard.edu/courses/cs181/files/lecture05-notes.pdf) (PDF). Harvard University.
14. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). *Deep Learning* (https://www.deeplearningbook.org). MIT Press. p. 291. ISBN 978-0262035613.
15. Cited by Darken, Christian; Moody, John (1990). *Fast adaptive k-means clustering: some empirical results*. Int'l Joint Conf. on Neural Networks (IJCNN). IEEE. doi:10.1109/IJCNN.1990.137720 (https://doi.org/10.1109%2FIJCNN.1990.137720).
16. Spall, J. C. (2003). *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Hoboken, NJ: Wiley. pp. Sections 4.4, 6.6, and 7.5. ISBN 0-471-33052-3.
17. Toulis, Panos; Airoldi, Edoardo (2017). "Asymptotic and finite-sample properties of estimators based on stochastic gradients". *Annals of Statistics*. **45** (4): 1694–1727. arXiv:1408.2923 (https://arxiv.org/abs/1408.2923). doi:10.1214/16-AOS1506 (https://doi.org/10.1214%2F16-AOS1506). S2CID 10279395 (https://api.semanticscholar.org/CorpusID:10279395).

18. Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". *Nature*. **323** (6088): 533–536. Bibcode:1986Natur.323..533R (https://ui.adsabs.harvard.edu/abs/1986Natur.323..533R). doi:10.1038/323533a0 (https://doi.org/10.1038%2F323533a0). S2CID 205001834 (https://api.semanticscholar.org/CorpusID:205001834).
19. Sutskever, Ilya; Martens, James; Dahl, George; Hinton, Geoffrey E. (June 2013). Sanjoy Dasgupta and David Mcallester (ed.). *On the importance of initialization and momentum in deep learning* (http://www.cs.utoronto.ca/~ilya/pubs/2013/1051_2.pdf) (PDF). In Proceedings of the 30th international conference on machine learning (ICML-13). Vol. 28. Atlanta, GA. pp. 1139–1147. Retrieved 14 January 2016.
20. Sutskever, Ilya (2013). *Training recurrent neural networks* (http://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf) (PDF) (Ph.D.). University of Toronto. p. 74.
21. Zeiler, Matthew D. (2012). "ADADELTA: An adaptive learning rate method". arXiv:1212.5701 (https://arxiv.org/abs/1212.5701) [cs.LG (https://arxiv.org/archive/cs.LG)].
22. Borysenko, Oleksandr; Byshkin, Maksym (2021). "CoolMomentum: A Method for Stochastic Optimization by Langevin Dynamics with Simulated Annealing" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8139967). *Scientific Reports*. **11** (1): 10705. arXiv:2005.14605 (https://arxiv.org/abs/2005.14605). Bibcode:2021NatSR..1110705B (https://ui.adsabs.harvard.edu/abs/2021NatSR..1110705B). doi:10.1038/s41598-021-90144-3 (https://doi.org/10.1038%2Fs41598-021-90144-3). PMC 8139967 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8139967). PMID 34021212 (https://pubmed.ncbi.nlm.nih.gov/34021212).
23. Polyak, Boris T.; Juditsky, Anatoli B. (1992). "Acceleration of stochastic approximation by averaging" (http://www.meyn.ece.ufl.edu/archive/spm_files/Courses/ECE555-2011/555media/poljud92.pdf) (PDF). *SIAM J. Control Optim*. **30** (4): 838–855. doi:10.1137/0330046 (https://doi.org/10.1137%2F0330046).
24. Duchi, John; Hazan, Elad; Singer, Yoram (2011). "Adaptive subgradient methods for online learning and stochastic optimization" (http://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf) (PDF). *JMLR*. **12**: 2121–2159.
25. Gupta, Maya R.; Bengio, Samy; Weston, Jason (2014). "Training highly multiclass classifiers" (http://jmlr.org/papers/volume15/gupta14a/gupta14a.pdf) (PDF). *JMLR*. **15** (1): 1461–1492.
26. Hinton, Geoffrey. "Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude" (http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf) (PDF). p. 26. Retrieved 19 March 2020.
27. Hinton, Geoffrey. "Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude" (http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf) (PDF). p. 29. Retrieved 19 March 2020.
28. Kingma, Diederik; Ba, Jimmy (2014). "Adam: A Method for Stochastic Optimization". arXiv:1412.6980 (https://arxiv.org/abs/1412.6980) [cs.LG (https://arxiv.org/archive/cs.LG)].
29. Byrd, R. H.; Hansen, S. L.; Nocedal, J.; Singer, Y. (2016). "A Stochastic Quasi-Newton method for Large-Scale Optimization". *SIAM Journal on Optimization*. **26** (2): 1008–1031. arXiv:1401.7020 (https://arxiv.org/abs/1401.7020). doi:10.1137/140954362 (https://doi.org/10.1137%2F140954362). S2CID 12396034 (https://api.semanticscholar.org/CorpusID:12396034).
30. Spall, J. C. (2000). "Adaptive Stochastic Approximation by the Simultaneous Perturbation Method". *IEEE Transactions on Automatic Control*. **45** (10): 1839−1853. doi:10.1109/TAC.2000.880982 (https://doi.org/10.1109%2FTAC.2000.880982).
31. Spall, J. C. (2009). "Feedback and Weighting Mechanisms for Improving Jacobian Estimates in the Adaptive Simultaneous Perturbation Algorithm". *IEEE Transactions on Automatic Control*. **54** (6): 1216–1229. doi:10.1109/TAC.2009.2019793 (https://doi.org/10.1109%2FTAC.2009.2019793).
32. Bhatnagar, S.; Prasad, H. L.; Prashanth, L. A. (2013). *Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods*. London: Springer. ISBN 978-1-4471-4284-3.

33. Ruppert, D. (1985). "A Newton-Raphson Version of the Multivariate Robbins-Monro Procedure" (https://doi.org/10.1214%2Faos%2F1176346589). *Annals of Statistics*. **13** (1): 236–245. doi:10.1214/aos/1176346589 (https://doi.org/10.1214%2Faos%2F1176346589).

Further reading

- Bottou, Léon (2004), "Stochastic Learning" (http://leon.bottou.org/papers/bottou-mlss-2004), *Advanced Lectures on Machine Learning*, LNAI, vol. 3176, Springer, pp. 146–168, ISBN 978-3-540-23122-6
- Buduma, Nikhil; Locascio, Nicholas (2017), "Beyond Gradient Descent" (https://www.google.com/books/edition/Fundamentals_of_Deep_Learning/80glDwAAQBAJ?hl=en&gbpv=1&pg=PA63), *Fundamentals of Deep Learning : Designing Next-Generation Machine Intelligence Algorithms*, O'Reilly
- LeCun, Yann A.; Bottou, Léon; Orr, Genevieve B.; Müller, Klaus-Robert (2012), "Efficient BackProp" (https://www.google.com/books/edition/Neural_Networks_Tricks_of_the_Trade/VCKqCAAAQBAJ?hl=en&gbpv=1&pg=PA9), *Neural Networks: Tricks of the Trade*, Springer, pp. 9–48, ISBN 978-3-642-35288-1
- Spall, James C. (2003), *Introduction to Stochastic Search and Optimization*, Wiley, ISBN 978-0-471-33052-3

External links

- Using stochastic gradient descent in C++, Boost, Ublas for linear regression (http://codingplayground.blogspot.it/2013/05/stocastic-gradient-descent.html)
- Machine Learning Algorithms (http://studyofai.com/machine-learning-algorithms/)
- "Gradient Descent, How Neural Networks Learn" (https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=2). *3Blue1Brown*. October 16, 2017. Archived (https://ghostarchive.org/varchive/youtube/20211222/IHZwWFHWa-w) from the original on 2021-12-22 – via YouTube.
- Goh (April 4, 2017). "Why Momentum Really Works" (https://distill.pub/2017/momentum/). *Distill*. **2** (4). doi:10.23915/distill.00006 (https://doi.org/10.23915%2Fdistill.00006). Interactive paper explaining momentum.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Stochastic_gradient_descent&oldid=1098148439"

This page was last edited on 14 July 2022, at 12:09 (UTC).