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 exchange for a lower convergence rate.[1]

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

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. There, $Q_i(w)$ 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:

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

The step size is denoted by η (sometimes called the *learning rate* in machine learning) and here "$:=$" denotes the update of a variable in the algorithm.

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.

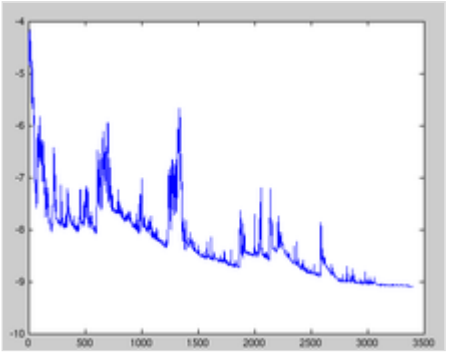
However, in other cases, evaluating the sum-gradient may require expensive evaluations of the gradients from all summand functions. When the training set is enormous and no simple formulas exist, evaluating the sums of gradients becomes very expensive, because evaluating the gradient requires evaluating 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:

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

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]



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

In pseudocode, stochastic gradient descent can be presented as :

- Choose an initial vector of parameters w and learning rate η.
- 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 samples.

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.[2][7] This is in fact a consequence of the Robbins–Siegmund theorem.[8]

Example

Suppose we want to fit a straight line $\hat{y} = w_1 + w_2 x$ to a training set with observations $((x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n))$ 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 or update step, the gradient is only evaluated at a single x_i. This is the key difference between stochastic gradient descent and batched gradient descent.

History

In 1951, Herbert Robbins and Sutton Monro introduced the earliest stochastic approximation methods, preceding stochastic gradient descent.[9] Building on this work one year later, Jack Kiefer and Jacob Wolfowitz published an optimization algorithm very close to stochastic gradient descent, using central differences as an approximation of the gradient.[10] Later in the 1950s, Frank Rosenblatt used SGD to optimize his perceptron model, demonstrating the first applicability of stochastic gradient descent to neural networks.[11]

Backpropagation was first described in 1986, with stochastic gradient descent being used to efficiently optimize parameters across neural networks with multiple hidden layers. Soon after, another improvement was developed: mini-batch gradient descent, where small batches of data are substituted for single samples. In 1997, the practical performance benefits from vectorization achievable with such small batches were first explored,[12] paving the way for efficient optimization in machine learning. As of 2023, this mini-batch approach remains the norm for training neural networks, balancing the benefits of stochastic gradient descent with gradient descent.[13]

By the 1980s, momentum had already been introduced, and was added to SGD optimization techniques in 1986.[14] However, these optimization techniques assumed constant hyperparameters, i.e. a fixed learning rate and momentum parameter. In the 2010s, adaptive approaches to applying SGD with a per-parameter learning rate were introduced with AdaGrad (for "Adaptive Gradient") in 2011[15] and RMSprop (for "Root Mean Square Propagation") in 2012.[16] In 2014, Adam (for "Adaptive Moment Estimation") was published, applying the adaptive approaches of RMSprop to momentum; many improvements and branches of Adam were then developed such as Adadelta, Adagrad, AdamW, and Adamax.[17][18]

Within machine learning, approaches to optimization in 2023 are dominated by Adam-derived optimizers. TensorFlow and PyTorch, by far the most popular machine learning libraries,[19] as of 2023 largely only include Adam-derived optimizers, as well as predecessors to Adam such as RMSprop and classic SGD. PyTorch also partially supports Limited-memory BFGS, a line-search method, but only for single-device setups without parameter groups.[18][20]

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.[21] When combined with the back propagation algorithm, it is the *de facto* standard algorithm for training artificial neural networks.[22] Its use has been also reported in the Geophysics community, specifically to applications of Full Waveform Inversion (FWI).[23]

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.[24]

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

Extensions and variants

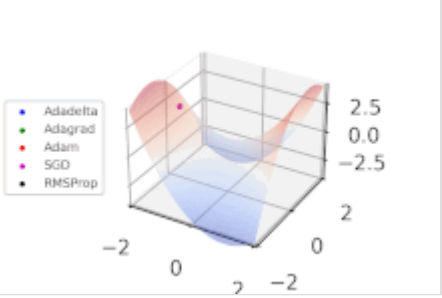
Many improvements on the basic stochastic gradient descent algorithm have been proposed and used. In particular, in machine learning, the need to set a learning rate (step size) has been recognized as problematic. Setting this parameter too high can cause the algorithm to diverge; setting it too low makes it slow to converge.[25] 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.[26] Practical guidance on choosing the step size in several variants of SGD is given by Spall.[27]

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[28] by considering *implicit updates* whereby the stochastic gradient is evaluated at the next iterate rather than the current one:

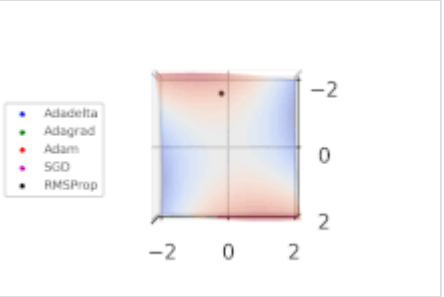
$$w^{\mathrm{new}} := w^{\mathrm{old}} - \eta \nabla Q_i(w^{\mathrm{new}}).$$

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:



A graph visualizing the behavior of a selected set of optimizers, using a 3D perspective projection of a loss function f(x, y)

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



A graph visualizing the behavior of a selected set of optimizers

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:

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

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:

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

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:

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

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* or the *heavy ball method*, which in ML context appeared in Rumelhart, Hinton and Williams' paper on backpropagation learning[29] and borrowed the idea from Soviet mathematician Boris Polyak's 1964 article on solving functional equations.[30] 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:[31][32]

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

$$w := w + \Delta w$$

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,[29] 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.[33] The *momentum method* is closely related to underdamped Langevin dynamics, and may be combined with simulated annealing.[34]

In mid-1980s the method was modified by Yurii Nesterov to use the gradient predicted at the next point, and the resulting so-called *Nesterov Accelerated Gradient* was sometimes used in ML in the 2010s.[35]

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[36]

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

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.[37] 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.[37]

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 essentially stores a historical sum of gradient squares by dimension and is updated after every iteration. The formula for an update is now[a]

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

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.[33]

While designed for convex problems, AdaGrad has been successfully applied to non-convex optimization.[38]

RMSProp

RMSProp (for Root Mean Square Propagation) is a method invented in 2012 by James Martens and Ilya Sutskever, at the time both PhD students in Geoffrey Hinton's group, in which the learning rate is, like in Adagrad, 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.[39] Unusually, it was not published in an article but merely described in a Coursera lecture.

So, first the running average is calculated in terms of means square,

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

where, γ is the forgetting factor. The concept of storing the historical gradient as sum of squares is borrowed from Adagrad, but "forgetting" is introduced to solve Adagrad's diminishing learning rates in non-convex problems by gradually decreasing the influence of old data.

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.[39]

Adam

Adam[40] (short for Adaptive Moment Estimation) is a 2014 update to the *RMSProp* optimizer combining it with the main feature of the *Momentum method*.[41] In this optimization algorithm, running averages with exponential forgetting 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:

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

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

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

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.

The initial proof establishing the convergence of Adam was incomplete, and subsequent analysis has revealed that Adam does not converge for all convex objectives.[42][43] Despite this, *Adam* continues to be used in practice due to its strong performance in practice.[44]

Variants

The popularity of *Adam* inspired many variants and enhancements. Some examples include:

- Nesterov-enhanced gradients: *NAdam*,[45] *FASFA*[46]
- varying interpretations of second-order information: *Powerpropagation*[47] and *AdaSqrt*.[48]
- Using infinity norm: *AdaMax*[40]
- *AMSGrad*,[49] which improves convergence over *Adam* by using maximum of past squared gradients instead of the exponential average.[50] *AdamX*[51] further improves convergence over *AMSGrad*.
- *AdamW*,[52] which improves the weight decay.

Sign-based stochastic gradient descent

Even though sign-based optimization goes back to the aforementioned *Rprop*, in 2018 researchers tried to simplify Adam by removing the magnitude of the stochastic gradient from being taken into account and only considering its sign.[53][54]

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.[55] 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.[56][57][58] (A less efficient method based on finite differences, instead of simultaneous perturbations, is given by Ruppert.[59]) Another approach to the approximation Hessian matrix is replacing it with the Fisher information matrix, which transforms usual gradient to natural.[60] 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. When the objective is a nonlinear least-squres loss

$$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w) = \frac{1}{n}\sum_{i=1}^{n}(m(w;x_i) - y_i)^2,$$

where $m(w;x_i)$ is the predictive model (e.g., a deep neural network) the objective's structure can be exploited to estimate 2nd order information using gradients only. The resulting methods are simple and often effective[61]

Approximations in continuous time

For small learning rate η stochastic gradient descent $(w_n)_{n\in\mathbb{N}_0}$ can be viewed as a discretization of the gradient flow ODE

$$\frac{d}{dt}W_t = -\nabla Q(W_t)$$

subject to additional stochastic noise. This approximation is only valid on a finite time-horizon in the following sense: assume that all the coefficients Q_i are sufficiently smooth. Let $T > 0$ and $g : \mathbb{R}^d \to \mathbb{R}$ be a sufficiently smooth test function. Then, there exists a constant $C > 0$ such that for all $\eta > 0$

$$\max_{k=0,\ldots,\lfloor T/\eta \rfloor} |\mathbb{E}[g(w_k)] - g(W_{k\eta})| \leq C\eta,$$

where $\mathbb{E}$ denotes taking the expectation with respect to the random choice of indices in the stochastic gradient descent scheme.

Since this approximation does not capture the random fluctuations around the mean behavior of stochastic gradient descent solutions to stochastic differential equations (SDEs) have been proposed as limiting objects.[62] More precisely, the solution to the SDE

$$dW_t = -\nabla \left(Q(W_t) + \tfrac{1}{4}\eta |\nabla Q(W_t)|^2 \right) dt + \sqrt{\eta}\Sigma(W_t)^{1/2} dB_t,$$

for

$$\Sigma(w) = \frac{1}{n^2} \left(\sum_{i=1}^{n} Q_i(w) - Q(w) \right) \left(\sum_{i=1}^{n} Q_i(w) - Q(w) \right)^T$$

where dB_t denotes the Ito-integral with respect to a Brownian motion is a more precise approximation in the sense that there exists a constant $C > 0$ such that

$$\max_{k=0,\ldots,\lfloor T/\eta \rfloor} |\mathbb{E}[g(w_k)] - \mathbb{E}[g(W_{k\eta})]| \leq C\eta^2.$$

However this SDE only approximates the one-point motion of stochastic gradient descent. For an approximation of the stochastic flow one has to consider SDEs with infinite-dimensional noise.[63]

See also

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

- Stochastic variance reduction

Notes

a. ⊙ denotes the element-wise product.

References

1. Bottou, Léon; Bousquet, Olivier (2012). "The Tradeoffs of Large Scale Learning" (https://books.google.com/books?id=JPQx7s2L1A8C&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). 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. Kiwiel, Krzysztof C. (2001). "Convergence and efficiency of subgradient methods for quasiconvex minimization". *Mathematical Programming, Series A*. **90** (1). Berlin, Heidelberg: Springer: 1–25. doi:10.1007/PL00011414 (https://doi.org/10.1007%2FPL00011414). ISSN 0025-5610 (https://search.worldcat.org/issn/0025-5610). MR 1819784 (https://mathscinet.ams.org/mathscinet-getitem?mr=1819784). S2CID 10043417 (https://api.semanticscholar.org/CorpusID:10043417).
8. 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.
9. Robbins, H.; Monro, S. (1951). "A Stochastic Approximation Method" (https://doi.org/10.1214%2Faoms%2F1177729586). *The Annals of Mathematical Statistics*. **22** (3): 400. doi:10.1214/aoms/1177729586 (https://doi.org/10.1214%2Faoms%2F1177729586).
10. Kiefer, J.; Wolfowitz, J. (1952). "Stochastic Estimation of the Maximum of a Regression Function" (https://doi.org/10.1214%2Faoms%2F1177729392). *The Annals of Mathematical Statistics*. **23** (3): 462–466. doi:10.1214/aoms/1177729392 (https://doi.org/10.1214%2Faoms%2F1177729392).
11. Rosenblatt, F. (1958). "The perceptron: A probabilistic model for information storage and organization in the brain". *Psychological Review*. **65** (6): 386–408. doi:10.1037/h0042519 (https://doi.org/10.1037%2Fh0042519). S2CID 12781225 (https://api.semanticscholar.org/CorpusID:12781225).

12. 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).
13. Peng, Xinyu; Li, Li; Wang, Fei-Yue (2020). "Accelerating Minibatch Stochastic Gradient Descent Using Typicality Sampling" (https://ieeexplore.ieee.org/document/8945166). *IEEE Transactions on Neural Networks and Learning Systems*. **31** (11): 4649–4659. arXiv:1903.04192 (https://arxiv.org/abs/1903.04192). doi:10.1109/TNNLS.2019.2957003 (https://doi.org/10.1109%2FTNNLS.2019.2957003). PMID 31899442 (https://pubmed.ncbi.nlm.nih.gov/31899442). S2CID 73728964 (https://api.semanticscholar.org/CorpusID:73728964). Retrieved 2023-10-02.
14. Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (October 1986). "Learning representations by back-propagating errors" (https://www.nature.com/articles/323533a0). *Nature*. **323** (6088): 533–536. doi:10.1038/323533a0 (https://doi.org/10.1038%2F323533a0). ISSN 1476-4687 (https://search.worldcat.org/issn/1476-4687). S2CID 205001834 (https://api.semanticscholar.org/CorpusID:205001834).
15. 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.
16. 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.
17. 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)].
18. "torch.optim — PyTorch 2.0 documentation" (https://pytorch.org/docs/stable/optim.html). *pytorch.org*. Retrieved 2023-10-02.
19. Nguyen, Giang; Dlugolinsky, Stefan; Bobák, Martin; Tran, Viet; García, Álvaro; Heredia, Ignacio; Malík, Peter; Hluchý, Ladislav (19 January 2019). "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey" (https://link.springer.com/content/pdf/10.1007/s10462-018-09679-z.pdf) (PDF). *Artificial Intelligence Review*. **52**: 77–124. doi:10.1007/s10462-018-09679-z (https://doi.org/10.1007%2Fs10462-018-09679-z). S2CID 254236976 (https://api.semanticscholar.org/CorpusID:254236976).
20. "Module: tf.keras.optimizers | TensorFlow v2.14.0" (https://www.tensorflow.org/api_docs/python/tf/keras/optimizers). *TensorFlow*. Retrieved 2023-10-02.
21. 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.
22. 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)
23. 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)
24. Avi Pfeffer. "CS181 Lecture 5 — Perceptrons" (http://www.seas.harvard.edu/courses/cs181/files/lecture05-notes.pdf) (PDF). Harvard University.
25. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). *Deep Learning* (https://www.deeplearningbook.org). MIT Press. p. 291. ISBN 978-0262035613.
26. 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).

27. 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.
28. 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).
29. 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).
30. "Gradient Descent and Momentum: The Heavy Ball Method" (https://boostedml.com/2020/07/gradient-descent-and-momentum-the-heavy-ball-method.html). 13 July 2020.
31. 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.
32. 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.
33. 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)].
34. 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).
35. "Papers with Code - Nesterov Accelerated Gradient Explained" (https://paperswithcode.com/method/nesterov-accelerated-gradient).
36. Polyak, Boris T.; Juditsky, Anatoli B. (1992). "Acceleration of stochastic approximation by averaging" (https://web.archive.org/web/20160112091615/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). S2CID 3548228 (https://api.semanticscholar.org/CorpusID:3548228). Archived from the original (http://www.meyn.ece.ufl.edu/archive/spm_files/Courses/ECE555-2011/555media/poljud92.pdf) (PDF) on 2016-01-12. Retrieved 2018-02-14.
37. 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.
38. 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.
39. 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.
40. 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)].

41. "4. Beyond Gradient Descent - Fundamentals of Deep Learning [Book]" (https://www.oreilly.com/library/view/fundamentals-of-deep/9781491925607/ch04.html).
42. Reddi, Sashank J.; Kale, Satyen; Kumar, Sanjiv (2018). *On the Convergence of Adam and Beyond* (https://openreview.net/forum?id=ryQu7f-RZ). 6th International Conference on Learning Representations (ICLR 2018). arXiv:1904.09237 (https://arxiv.org/abs/1904.09237).
43. Rubio, David Martínez (2017). *Convergence Analysis of an Adaptive Method of Gradient Descent* (https://damaru2.github.io/convergence_analysis_hypergradient_descent/dissertation_hypergradients.pdf) (PDF) (Master thesis). University of Oxford. Retrieved 5 January 2024.
44. Zhang, Yushun; Chen, Congliang; Shi, Naichen; Sun, Ruoyu; Luo, Zhi-Quan (2022). "Adam Can Converge Without Any Modification On Update Rules". *Advances in Neural Information Processing Systems 35*. Advances in Neural Information Processing Systems 35 (NeurIPS 2022). arXiv:2208.09632 (https://arxiv.org/abs/2208.09632).
45. Dozat, T. (2016). "Incorporating Nesterov Momentum into Adam". S2CID 70293087 (https://api.semanticscholar.org/CorpusID:70293087). {{cite journal}}: Cite journal requires |journal= (help)
46. Naveen, Philip (2022-08-09). "FASFA: A Novel Next-Generation Backpropagation Optimizer" (https://dx.doi.org/10.36227/techrxiv.20427852.v1). doi:10.36227/techrxiv.20427852.v1 (https://doi.org/10.36227%2Ftechrxiv.20427852.v1). Retrieved 2022-11-19. {{cite journal}}: Cite journal requires |journal= (help)
47. Whye, Schwarz, Jonathan Jayakumar, Siddhant M. Pascanu, Razvan Latham, Peter E. Teh, Yee (2021-10-01). *Powerpropagation: A sparsity inducing weight reparameterisation* (http://worldcat.org/oclc/1333722169). OCLC 1333722169 (https://search.worldcat.org/oclc/1333722169).
48. Hu, Yuzheng; Lin, Licong; Tang, Shange (2019-12-20). "Second-order Information in First-order Optimization Methods". arXiv:1912.09926 (https://arxiv.org/abs/1912.09926). {{cite journal}}: Cite journal requires |journal= (help)
49. Reddi, Sashank J.; Kale, Satyen; Kumar, Sanjiv (2018). "On the Convergence of Adam and Beyond". arXiv:1904.09237 (https://arxiv.org/abs/1904.09237). {{cite journal}}: Cite journal requires |journal= (help)
50. "An overview of gradient descent optimization algorithms" (https://www.ruder.io/optimizing-gradient-descent/#amsgrad). 19 January 2016.
51. Tran, Phuong Thi; Phong, Le Trieu (2019). "On the Convergence Proof of AMSGrad and a New Version" (https://ieeexplore.ieee.org/document/8713445/). *IEEE Access*. **7**: 61706–61716. arXiv:1904.03590 (https://arxiv.org/abs/1904.03590). doi:10.1109/ACCESS.2019.2916341 (https://doi.org/10.1109%2FACCESS.2019.2916341). ISSN 2169-3536 (https://search.worldcat.org/issn/2169-3536).
52. Loshchilov, Ilya; Hutter, Frank (4 January 2019). "Decoupled Weight Decay Regularization". arXiv:1711.05101 (https://arxiv.org/abs/1711.05101). {{cite journal}}: Cite journal requires |journal= (help)
53. Balles, Lukas; Hennig, Philipp (15 February 2018). "Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients" (https://openreview.net/forum?id=S1EwLkW0W).
54. "SignSGD: Compressed Optimisation for Non-Convex Problems" (https://proceedings.mlr.press/v80/bernstein18a.html). 3 July 2018. pp. 560–569.
55. 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).

56. 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).
57. 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). S2CID 3564529 (https://api.semanticscholar.org/CorpusID:3564529).
58. 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.
59. 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).
60. Amari, S. (1998). "Natural gradient works efficiently in learning". *Neural Computation*. **10** (2): 251–276. doi:10.1162/089976698300017746 (https://doi.org/10.1162%2F089976698300017746). S2CID 207585383 (https://api.semanticscholar.org/CorpusID:207585383).
61. Brust, J.J. (2021). "Nonlinear least squares for large-scale machine learning using stochastic Jacobian estimates". *Workshop: Beyond First Order Methods in Machine Learning*. ICML 2021. arXiv:2107.05598 (https://arxiv.org/abs/2107.05598).
62. Li, Qianxiao; Tai, Cheng; E, Weinan (2019). "Stochastic Modified Equations and Dynamics of Stochastic Gradient Algorithms I: Mathematical Foundations" (http://jmlr.org/papers/v20/17-526.html). *Journal of Machine Learning Research*. **20** (40): 1–47. ISSN 1533-7928 (https://search.worldcat.org/issn/1533-7928).
63. Gess, Benjamin; Kassing, Sebastian; Konarovskyi, Vitalii (14 February 2023). "Stochastic Modified Flows, Mean-Field Limits and Dynamics of Stochastic Gradient Descent". arXiv:2302.07125 (https://arxiv.org/abs/2302.07125) [math.PR (https://arxiv.org/archive/math.PR)].

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://books.google.com/books?id=80glDwAAQBAJ&pg=PA63), *Fundamentals of Deep Learning : Designing Next-Generation Machine Intelligence Algorithms*, O'Reilly, ISBN 9781491925584
- LeCun, Yann A.; Bottou, Léon; Orr, Genevieve B.; Müller, Klaus-Robert (2012), "Efficient BackProp" (https://books.google.com/books?id=VCKqCAAAQBAJ&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

- "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.
