Stochastic gradient descent

VO Van Nghia PHAM Tuan Kiet

Abstract

Stochastic gradient descent is an iterative method for optimizing an objective function 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 Gradient descent

An approximate and iterative technique for mathematical optimization is the gradient descent algorithm. It can be used to approach any differentiable function's minimum. For a better imagination about the algorithm, the process is someho the same as a dropping drip of water following the slope. During the process of the algorithm, the new position for eaxh iteration is determined by:

$$x_{n+1} = x_n + -\tau \nabla f(x)$$

The main problem of this algorithm is that sometimes the solution falls into a local minimum point or a saddle point instead of the desired global minimum. It is frequently used internally by data science and machine learning to optimize model parameters. For instance, gradient descent is used by neural networks to find weights and biases.

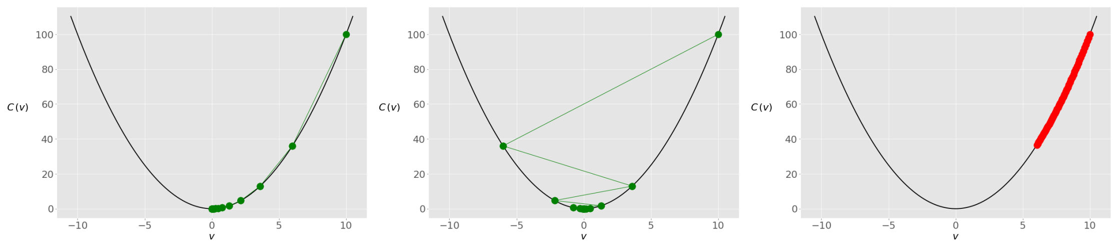

Figure 1: Impact of learning rate

Learning rate (τ) is a very important parameter that affects the perfomance of the algorithm. Right here is an example showing how this value influences a process of gradient descent. With a too small rate, the algorithm seems to be far away from the desired result even after a large alount of iteration. However, a high learning rate doesn't give a better performance as it cannot approach the local minimum.

2 Stochastic gradient descent

As you can see, a higher learning rate leads to faster convergence speed at the very first iterations but a lower convergence rate as the iteration rises. Stochastic gradient descent is an improved version of gradient descent which modified the learning rate throughout the process in order to reduce the computation burden as a trade of convergence rate due to the learning rate converges to 0.

For very large n (which could in theory even be infinite, in which case the sum needs to be replaced by an expectation or equivalenty an integral), computing ∇E is prohebitive. It is possible instead to use a stochastic gradient descent (SGD) scheme

$$w_{\ell+1} = w_\ell - \tau_\ell \nabla E_{i(\ell)}(w_\ell)$$

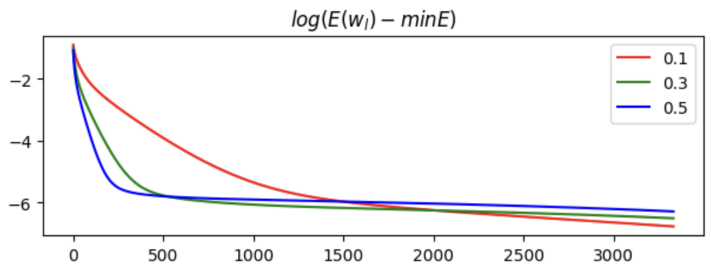

Figure 2: Impact of learning rate

where, for each iteration index ℓ, $i(\ell)$ is drawn uniformly at random in $\{1, \dots, n\}$

In this example, we are using a stochatics algorithm with

$$\tau_\ell \overset{\text{def}}{=} \frac{\tau_0}{1 + \ell/\ell_0}$$

where ℓ_0 indicates roughly the number of iterations serving as a "warmup" phase.

As compared to a normal Gradient Descent algorithm, the SGD algorithm out-performs for the first iterations with faster speed of convergence.

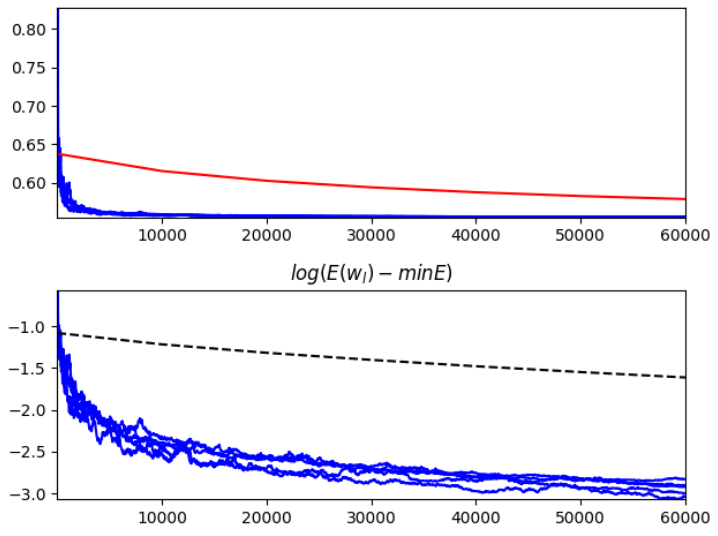

Figure 3: SGD performance

3 Improvement of Stochatics Gradient Descent Algorithm

As we all know that the normal SGD (Stochatics Gradient Descent) algorithm is slow due to its fast decay of the the learning rate (τ). To enhance its performance, there are alternatives that help solve the weakness of SGD.

3.1 SGD with Averaging

To improve somehow the convergence speed, it is possible to average the past iterate, i.e. run a "classical" SGD on auxiliary variables:

$$\tilde{w}_{\ell+1} = \tilde{w}_\ell - \tau_\ell \nabla E_{i(\ell)}(\tilde{w}_\ell)$$

and output as estimated weight vector the average

$$w_\ell \overset{\text{def}}{=} \frac{1}{\ell} \sum_{k=1}^{\ell} \tilde{w}_\ell.$$

This defines the Stochastic Gradient Descent with Averaging (SGA) algorithm. And in this case, the learning rate is define as:

$$\tau_\ell \overset{\text{def}}{=} \frac{\tau_0}{1 + \sqrt{\ell/\ell_0}}.$$

Typically, because the averaging stabilizes the iterates, the choice of (ℓ_0, τ_0) is less important than for SGD.

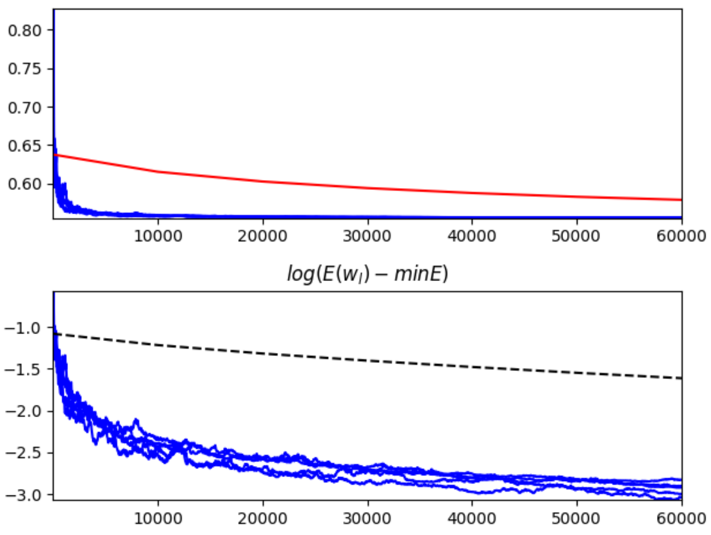

Figure 4: SGD vs SGDA

3.2 Stochatics Averaged Gradient Descent

SGD and SGA can be extended to infinite n and more general minimization of expectations. However, with a finite value of n, another improved algorithm performs better. That's is the Stochatics Averaged Gradient Descent(or SAGD).

The algorithm reads

$$h \leftarrow \nabla E_{i(\ell)}(\tilde{w}_\ell),$$

$$g \leftarrow g - G^{i(\ell)} + h,$$

$$G^{i(\ell)} \leftarrow h,$$

$$w_{\ell+1} = w_\ell - \tau g.$$

Note that in contrast to SGD and SGA, this method uses a fixed step size τ. Similarely to the the normal GD, in order to ensure convergence, the step size τ should be of the order of $1/L$ where L is the Lipschitz constant of E. A fixed step size means better rate of convergence in comparison to a 0-converge rate with a finite number of iterations.

As you can see in the plot, a fixed step leads to better convergence rate in spite of the low convergence speed at the very first iterations. However, with the adaptation of each iteration, the SAGD still out performs the normal Gradient Descent algorithm.

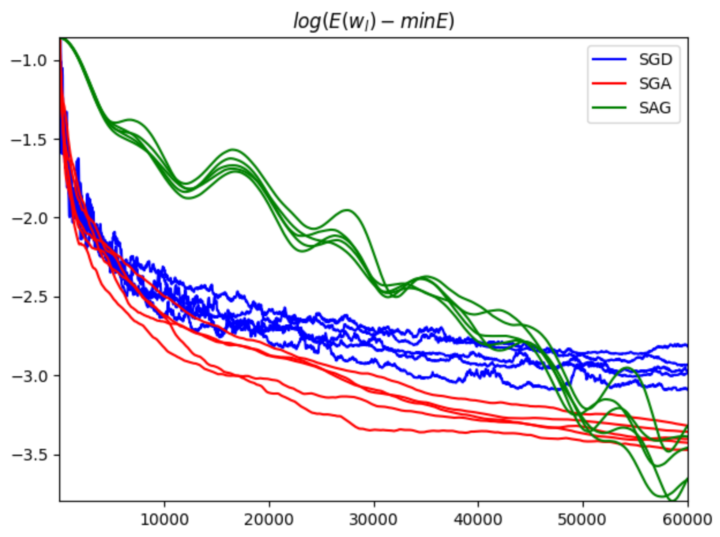

Figure 5: SAGD performance

4 Pros and Cons of SGD

In order to significantly speed up parameter updating, this approach randomly optimizes the loss function on a sample of training data rather than the loss function on the entire set of training data.

The SGD performs frequent updates with large variance, causing the objective function to vary significantly. That's why we often see the result of a SGD algorithm is covered with noise. The variability of SGD, on the other hand, allows it to reach new and potentially superior local minima. However, as SGD continues to overshoot, convergence to the exact minimum becomes more difficult without proper learning rate.