Stochastic optimization
Stochastic gradient descent

Fabian Bastin (bastin@iro.umontreal.ca)
Université de Montréal – CIRRELT – IVADO – Fin-ML

Motivation

The recent years have seen a huge success of machine learning, and a renew interest in the stochastic gradient algorithm, and the development of various variants.

The stochastic gradient algorithm is a special case of the stochastic approximation method, which was first introduced in 1951, and can be seen as an alternative to the sample-path approach.

The first part of these slides is based on S. Bhatnagar, H.L. Prasad, L.A. Prashanth, “Stochastic Recursive Algorithms for Optimization Simultaneous Perturbation Methods”, Springer-Verlag, 2013.

Robbins–Monro algorithm

Introduced in 1951, initially as a root-finding problem.

It can be easily extended to unconstrained optimization using first-order condition as a necessary condition to the problem

$$\min_{x \in \mathbb{R}^d} f(x)$$

is

$$\nabla_x f(x) = 0.$$

We therefore search for a zero of the gradient of $f(x)$.

We can also restrain the feasible domain to $\Theta \subseteq \mathbb{R}^d$:

$$\min_{x \in \Theta} f(x) = E[Y(x, \xi)]$$

In this case, we have to assume that f reaches its minimum in the interior of Θ.

Problem in expectation

We consider here

$$f(x) = E[Y(x, \xi)]$$

where the support of ξ is Ξ.

The problem to solve is

$$\nabla_x E[Y(x, \xi)] = 0$$

We assume here that we can exchange the expectation and derivation operators, i.e.

$$\nabla_x E[Y(x, \xi)] = E[\nabla_x Y(x, \xi)]$$

Stochastic approximation

Also know as as the stochastic gradient (SG) method.

Choose a starting point x_1.
$k \leftarrow 1$
while Stopping criteria not satisfied **do**
  Draw ξ_k from $\boldsymbol{\xi}$.
  Select a step length α_k.
  Compute

$$x_{k+1} = x_k - \alpha_k \nabla_x Y(x_k, \xi_k).$$

  $k \leftarrow k + 1$
end while

α_k is also called the learning rate.

Assumptions

Assume a unique minimizer x^*, and

A.1 $f(x)$ is continuously differentiable and its gradient is Lipschitz continuous with Lipschitz constant $L > 0$, i.e. $\forall\, x, y \in \mathbb{R}^d$,

$$\|\nabla_x f(x) - \nabla_x f(y)\|_2 \leq L\|x - y\|_2$$

A.2 The iterates remain a.s. bounded, i.e.

$$\sup_k \|x_k\| < \infty \text{ almost surely.}$$

A.3 There exist scalars $M \geq 0$ and $M_V \geq 0$ s.t. $\forall\, k$,

$$Var[\nabla_x Y(x, \xi_k)] \leq M + M_V \|\nabla_x f(x_k)\|_2^2$$

A.4 The sequence α_k, $k = 1, 2, \ldots$, satisfies

$$\sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

Lipschitz continuity

A well-known consequence of the Lipschitz-continous gradients assumption is

$$f(x) \leq f(y) + \nabla f(y)^T (x - y) + \frac{L}{2} \|x - y\|_2^2,$$

$\forall\, x, y \in \mathbb{R}^d$.

Step lengths

Consider the sequence of step lengths, also called positive gains sequence $\{\alpha_k \,|\, k \geq 1\}$.

This sequence satisfies the previous assumption in particular with

- $\alpha_k = \alpha/k$, given $\alpha > 0$.
- $\alpha_k = \alpha/k^\beta$, $\forall\, k \geq 1$, given $\alpha > 0$ and $\beta \in (0.5, 1)$.
- $\alpha_k = \alpha(\ln k)/k$, $\forall\, k \geq 2$, given $\alpha_1 = \alpha > 0$.
- $\alpha_k = \alpha/(k \ln k)$, $\forall\, k \geq 2$, given $\alpha_1 = \alpha > 0$.

Properties

- Very cheap iteration: gradient w.r.t. just one observation. No function evaluation.
- Not a descent method as we can have $-\nabla_x Y(x,\xi_i)^T E[\nabla_x Y(x,\xi)] \geq 0$ with $\nabla_x f(x) \neq 0$.
- Descent in expectation: if $\nabla_x f(x) \neq 0$,

$$\begin{aligned} & E[-\nabla_x Y(x,\xi_i)^T E[\nabla_x Y(x,\xi)]] \\ & = -\nabla_x E[Y(x,\xi_i)]^T \nabla_x E[Y(x,\xi)]] \\ & = -\nabla_x f(x)^T \nabla_x f(x) < 0 \end{aligned}$$

Mini-batch method

Replace $\nabla_x Y(x, \xi_i)$ by

$$\frac{1}{n_k} \sum_{i=1}^{n_k} \nabla_x Y(x, \xi_i).$$

At each iteration, we take n_k new draws.

The cost per iteration is n_k times bigger, but

- it is a better estimate of the gradient
- the computation of the mini-batch can exploit parallelism

Batch method

Assume for now that the support Ξ is finite, of cardinality n. Then

$$E[Y(x,\xi)] = \frac{1}{n}\sum_{i=1}^{n} Y(x,\xi_i), \quad E[\nabla_x Y(x,\xi)] = \frac{1}{n}\sum_{i=1}^{n} \nabla_x Y(x,\xi_i)$$

Batch method:

$$x_{k+1} = x_k - \alpha_k \frac{1}{n}\sum_{i=1}^{n} \nabla_x Y(x,\xi_i).$$

In other words, we use all the observations to compute the true gradient.

Often, n is very large, and we prefer to work with $n_k << n$.

Stochastic approximation

We can generalize the expression of the stochastic approximation iteration using an estimator of the gradient of f at x_k:

$$x_{k+1} = x_k - \alpha_k \nabla \hat{f}(x_k).$$

As before, the gradient estimator can usually be taken as $\nabla Y(x_k, \xi_k)$, where $(\xi_k,\ k \geq 1)$ are i.i.d.

In that case, if $f(\cdot)$ is smooth, has a unique global minimizer x^*, and $\alpha_k = \alpha/k$ with $\alpha > 0$ sufficiently large, then under additional nonrestrictive conditions,

$$\sqrt{n}\,(x_n - x^*) \Rightarrow N(0, \Lambda),$$

as $n \to \infty$, for a certain $d \times d$ matrix Λ.

Convergence speed: stochastic boundedness

Source: `https://en.wikipedia.org/wiki/Big_O_in_probability_notation`

We would like to measure how fast we converge to the solution, knowning that we generate a sequence of realizations of random variables.

Stochastic boundedness

The notation

$$X_n = O_p(a_n),$$

means that the set of values X_n/a_n is stochastically bounded:

$\forall\, \epsilon > 0,\ \exists\, M > 0, N > 0$ such that $P[|X_n/a_n| > M] < \epsilon\ \forall\, n > N$.

$O_p(\cdot)$ vs $o_p(\cdot)$

Thus, $X_n = O_p(1)$ iff

$$\forall\, \epsilon > 0,\ \exists N_\epsilon, \delta_\epsilon \text{ such that } P[|X_n| \geq \delta_\epsilon] \leq \epsilon\ \forall\, n > N_\epsilon.$$

Convergence in probability

$X_n = o_p(1)$ iff

$$\forall\, \epsilon > 0, \delta > 0\ \exists N_{\epsilon,\delta} \text{ such that } P[|X_n| \geq \delta] \leq \epsilon\ \forall n > N_{\epsilon,\delta}.$$

Therefore

$$X_n = o_p(1) \Rightarrow X_n = O_p(1).$$

The reverse does not hold.

More generally, $X_n = o_p(a_n)$ iff $X_n/a_n = o_p(1)$, i.e.

$$\forall\, \epsilon > 0 \lim_{n\to\infty} P[|X_n/a_n| \geq \epsilon] = 0.$$

Complexity

Source: Kim, Pasupathy, and Henderson, “A Guide to Sample Average Approximation”, in “Handbook of Simulation Optimization”, edited by Michael C. Fu, Springer, 2015.

If the number of iterations of completed in c units of computer time, $n(c)$ grows roughly linearly in c (as would be the case if, e.g., sample gradients are computed in constant time).

A time-changed version of the CLT establishes that the resulting SA estimator has an error

$$x_{n(c)} - x^* = O_p(c^{-1/2}).$$

Equivalently, the computational effort required to obtain an error of order ϵ with SA is $O_p(\epsilon^{-2})$.

The performance of the recursion is highly dependent on the gain sequence $\{\alpha_n\}$.

Polyak–Ruppert averaging

Within the context of the SA iterative scheme, the fastest achievable convergence rate is $O_p(c^{-1/2})$.

This rate can be achieved under the "Polyak–Ruppert averaging".

- step-size sequence: $a_n = a/n^\gamma$ for some $\gamma \in (0, 1)$
- estimator of x^*:

$$\overline{x}_n = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Under mild conditions, the Polyak–Ruppert averaging scheme enjoys a CLT, although with a different covariance matrix Λ.

This happens irrespective of the value of the constant $a > 0$ (but the choice of a affects the small-sample performance). The Polyak–Ruppert averaging scheme also has other optimality properties related to the matrix Λ.

Order of Convergence

Denote the numerical procedure acting on the sample function $f_n(x)$ by the mapping $A(x) : \Theta \to \Theta$.

Let $A_k(x)$ represent the iterate obtained after k successive applications of the $A(\cdot)$ on the initial iterate x.

Assume that the function $f_n(x)$ attains its infimum $v_n^* := \inf\{f_n(x) : x \in \Theta\}$ and that $f_n(A_k(x)) \to v_n^*$ as $k \to \infty$ for all $x \in \Theta$. Also, to avoid trivialities, assume that $f_n(A_{k+1}(x))$ is different from v_n^* for all k.

Sublinear convergence

Denote

$$Q_t = \lim \sup_{k \to \infty} \frac{|f_n(A_{k+1}(x)) - v_n^*|}{|f_n(A_k(x)) - v_n^*|^t}.$$

Definition

$A(x) : \Theta \to \Theta$ is said to exhibit p^{th}-order sublinear convergence if $Q_1 \geq 1$, and

$\exists\, p, s > 0$ such that $p = sup\{r : f_n(A_k(x)) - v_n^* \leq s/k^r,\ \forall\, x \in \Theta\}$.

Linear convergence

Definition

The numerical procedure $A(x) : \Theta \to \Theta$ is said to exhibit linear convergence if $Q_1 \in (0, 1)$ for all $x \in \Theta$.

The definition of linear convergence implies that there exists a constant β satisfying $f_n(A(x)) - v_n^* \leq \beta(f_n(x) - v_n^*)$ for all $x \in \Theta$. The projected gradient method with Armijo steps when executed on certain smooth problems exhibits a linear convergence rate.

Superlinear convergence

Definition

The numerical procedure $A(x) : \Theta \to \Theta$ is said to exhibit superlinear convergence if $Q_1 = 0$ for all $x \in \Theta$. The convergence is said to be p^{th}-order superlinear if $Q_1 = 0$ and $\sup\{t : Q_t = 0\} = p < \infty$ for all $x \in \Theta$.

When $f_n(x)$ is strongly convex and twice Lipschitz continuously differentiable with observable derivatives, Newton method is second-order superlinear. For settings where the derivative is unobservable, there is a slight degradation in the convergence rate, but Newton method remains superlinear.

Convergence rate for the SAA method

Theorem

Assumptions:

- *A1* $E[Y^2(x,\xi)] < \infty$ *for all* $x \in \Theta$.
- *A2* *The function* $Y(x,\xi)$ *is Lipschitz w.p.1, with Lipschitz constant* $K(\xi)$ *having finite expectation.*
- *A3* *The function* $f_n(x)$ *attains its infimum on* Θ *for each n w.p.1.*

Let $c = n \times k$ *and* $n/c^{1/(2p+1)} \to a$ *as* $c \to \infty$, *with* $a \in (0,\infty)$. *Then, if the numerical procedure exhibits* p^{th}*-order sublinear convergence,*

$$c^{p/(2p+1)} \left(f_n(A^k(x)) - v^* \right) = O_p(1)$$

as $c \to \infty$.

Convergence rate for the SAA method

Main insight: the maximum achievable convergence rate with the SAA method, is $O_p(c^{-p/(2p+1)})$ when the numerical procedure in use exhibits p^{th}-order sublinear convergence.

It is also possible to show that the corresponding rates when using linearly convergent and p^{th}-order superlinearly convergent procedures are $O_p((c/\log c)^{-1/2})$ and $O_p((c/\log\log c)^{-1/2})$, respectively.

None of the families of numerical procedures considered are capable of attaining the canonical convergence rate $O_p(c^{-1/2})$.

The generic SG method

Source: Léon Bottou, Frank E. Curtis, and Jorge Nocedal, Optimization Methods for Large-Scale Machine Learning, SIAM Review 60(2), 2018, pp. 223–311, https://doi.org/10.1137/16M1080173

We generalize the stochastic gradient method with the update

$$x_{k+1} = x_k - \alpha_k g(x_k, \xi_k).$$

instead of

$$x_{k+1} = x_k - \alpha_k \nabla_x Y(x_k, \xi_k).$$

The generic SG method

The function $f : \mathbb{R}^d \to \mathbb{R}$ could be

$$f(x) = \begin{cases} R(x) = \mathbb{E}[f(x;\boldsymbol{\xi})] & \text{the expected risk,} \\ R_n(x) = \frac{1}{n}\sum_{\xi=1}^{n} f(x;\xi) & \text{the empirical risk.} \end{cases}$$

The stochastic vector could be

$$g(x;\xi_k) = \begin{cases} \nabla_x Y(x_k,\xi_k) & \text{(one realization)} \\ \frac{1}{n_k}\sum_{i=1}^{n_k} \nabla_x Y(x_k,\xi_k) & \text{(minibatch)} \\ B_k \frac{1}{n_k}\sum_{i=1}^{n_k} \nabla_x Y(x_k,\xi_k) & \text{(rescaled minibatch)} \end{cases}$$

Stochastic processes

While we assumme the draws ξ_i, $i = 1, 2, \ldots$ are i.i.d., it is possible to extend the results to the situation where $\{\xi_i, i = 1, 2, \ldots\}$ form an adapted stochastic process, where each ξ_i can depend on the previous ones.

Active learning

- In active learning, $g(x_k; \xi_k)$ produces a multinomial distribution on the training examples in a manner that depends on the current solution x_k.
- ξ_k is then transformed to draw from this distribution.

Active learning is not covered here, but again, the results can be extended to this situation.

Analysis

More details at:

- https://icml.cc/2016/tutorials/part-2.pdf
- https://icml.cc/2016/tutorials/part-3.pdf

We will refer to this material.