Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

A NEW ALGORITHM FOR STOCHASTIC OPTIMIZATION

Sigrún Andradóttir

Department of Industrial Engineering
University of Wisconsin-Madison
1513 University Avenue
Madison, Wisconsin 53706

ABSTRACT

Classical stochastic optimization algorithms often diverge because of boundedness problems. We present a stochastic optimization algorithm that is guaranteed to converge even when the iterates are not assumed a priori to be bounded. The performance of this algorithm will be compared with the performance of classical stochastic optimization algorithms.

1. INTRODUCTION

Stochastic optimization involves the optimization of functions whose values are not known analytically and therefore have to be estimated or measured. This field originated in the early 1950's but is presently experiencing a resurgence of interest due to the recent development of efficient gradient estimation techniques for simulation.

The goal of stochastic optimization is to obtain a sequence of iterates $\{\theta_n\}$ that converges almost surely to a local minimizer of a function, possibly subject to some constraints. The functions involved are typically evaluated using simulation. They are not assumed to have any particular structure and the distribution of the estimates of the function values is unknown.

In this paper, we will briefly discuss classical stochastic optimization algorithms, before introducing a new algorithm for stochastic optimization. We will then compare the performance of these algorithms.

2. CLASSICAL STOCHASTIC OPTIMIZATION ALGORITHMS

Unconstrained stochastic optimization is the problem of finding a local minimizer for a predetermined function f, whose values can not be evaluated analytically, but have to be estimated or measured. Classical stochastic optimization algorithms obtain a sequence $\{\theta_n\}$ of estimates of the solution as described below:

Algorithm 1

```
Step 0: Choose $\theta_1 \in \Re^d$.
Step 1: Given $\theta_n$, generate an estimate $Y_n$ of $\nabla f(\theta_n)$.
Step 2: Compute
            $\theta_{n+1} = \theta_n - a_n Y_n$
Step 3: Go to step 1.
```

Algorithm 1 is a steepest descent algorithm. If θ_n is our current estimate of the solution, then θ_{n+1} is obtained by moving in the direction $-Y_n$, where Y_n is an estimate of $\nabla f(\theta_n)$. This is the direction along which we expect the objective function to decrease the fastest. The length of the step that the algorithm takes along this direction is $a_n\|Y_n\|$, where $\{a_n\}$ is a predetermined sequence of positive constants. We assume that $a_n \to 0$ and $\sum_{n=1}^{\infty} a_n = \infty$. These assumptions are needed to guaranty the convergence of algorithm 1. If $\sum_{n=1}^{\infty} a_n < \infty$ and the sequence Y_n is bounded, then the iterates $\{\theta_n\}$ will all lie in a compact sphere with center θ_1. This means that the algorithm can only converge if θ_1 is close enough to the solution of the problem. The assumption that $a_n \to 0$ is needed to dampen out the effect of the errors in the gradient evaluations to obtain almost sure convergence to the solution of the problem as the number of iterations goes to infinity. Typically, a_n is chosen as a/n, where a is a positive constant.

The Robbins-Monro algorithm [Robbins and Monro 1951] and the Kiefer-Wolfowitz algorithm [Kiefer and Wolfowitz 1952] are the two most commonly used algorithms for unconstrained stochastic optimization. They are both special cases of algorithm 1. They differ in how they estimate the gradient of the objective function. The Robbins-Monro algorithm estimates the gradient directly, whereas the Kiefer-Wolfowitz algorithm uses finite differences to estimate the gradient. These algorithms are far from ideal. It is well known that they converge extremely slowly when the objective function is very flat and we now show that they do not necessarily converge when the objective function is steep.

Typically, one expects an optimization algorithm to perform at least as well when it is applied to minimizing the function $f(\theta) = \frac{1}{4}\theta^4$, as when it is applied to minimizing the function $g(\theta) = \frac{1}{2}\theta^2$. This is because one feels that the minimizer is more stable in the former case. Algorithm 1 does not have this property. Let $a_n = a/n$ where $a > 0$ and assume that there are no errors in the evaluations of f' and g'. It is easy to show that algorithm 1 converges to the optimal solution when it is being used to minimize the function g:

$$\theta_{n+1} = \theta_n - \frac{a}{n}\theta_n = \frac{n-a}{n}\theta_n$$

for all n, so the sequence $\{\theta_{[a]+n}\}$ goes to zero monotonically. However, algorithm 1 does not necessarily converge when applied to minimizing the function f:

Lemma 1 *If $\theta_1 > \sqrt{3/a}$ and $\theta_{n+1} = \theta_n - \frac{a}{n}\theta_n^3$ for all $n \geq 1$ then*

$$|\theta_n| \geq |\theta_1| n! \tag{1}$$

for all $n \geq 1$.

Proof: Expression (1) clearly holds when $n = 1$. When $n = 2$, we have

$$|\theta_2| = |\theta_1|\left|1 - a\theta_1^2\right| = |\theta_1|\left[a\theta_1^2 - 1\right] \geq 2|\theta_1|$$

so expression (1) holds. Now assume $n \geq 2$. By induction, it suffices to show that expression (1) holds for $n+1$ whenever it holds for n. We have

$$\begin{aligned} |\theta_{n+1}| &= |\theta_n|\left|1 - \frac{a}{n}\theta_n^2\right| = |\theta_n|\left[\frac{a}{n}\theta_n^2 - 1\right] \\ &\geq \frac{|\theta_n|}{n}\left[3(n!)^2 - n\right] \geq \frac{|\theta_n|}{n}\left[3n^2(n-1) - n\right] \\ &= |\theta_n|\left[3n^2 - 3n - 1\right]. \end{aligned}$$

To show that expression (1) holds for $n+1$, it suffices to show that $3n^2 - 3n - 1 \geq n + 1$, or $P(n) := 3n^2 - 4n - 2 \geq 0$, whenever $n \geq 2$. But $P(x) \geq 0$ for $x \geq \left(2 + \sqrt{10}\right)/3 \simeq 1.72$, and the proof is complete. □

The above lemma shows that algorithm 1 does not necessarily converge when applied to minimizing the function $f(\theta) = \frac{1}{4}\theta^4$. The problem is that the function f is quite steep, so the length of the step taken in an iteration $(a_n|f'(\theta_n)|)$ can be very large. The algorithm generates a sequence of estimates of the solution that goes to infinity in norm, fluctuating around the solution.

3. A NEW STOCHASTIC OPTIMIZATION ALGORITHM

In the previous section, we discussed the problems associated with the classical algorithms for stochastic optimization: they converge extremely slowly when the objective function is flat and they often diverge when the objective function is steep. The problem in both cases is that the length of the step that these algorithms take in each iteration is directly proportional to the norm of the gradient evaluation. We present a new stochastic optimization algorithm that does not have this property. It can be used with both unbiased and finite difference estimates of the gradient of the objective function. It is based on the following deterministic mathematical programming algorithm:

$$\theta_{n+1} = \theta_n - a_n \frac{\nabla f(\theta_n)}{\|\nabla f(\theta_n)\|} \tag{2}$$

where $\{a_n\}$ is a sequence of positive constants such that $a_n \to 0$ and $\sum_{n=1}^{\infty} a_n = \infty$. This algorithm was proposed for deterministic optimization by Poljak [1967], based on an algorithm proposed by Shor [1964]. We now propose a new stochastic optimization algorithm, based on sequence (2).

Let $\epsilon > 0$ and consider the following algorithm:

Algorithm 2

Step 0: *Choose $\theta_1 \in \Re^d$.*

Step 1: *Given θ_n, generate two independent estimates Y_n^1 and Y_n^2 of $\nabla f(\theta_n)$.*

Step 2: *Compute*

$$Y_n = \frac{Y_n^1}{\max\{\epsilon, \|Y_n^2\|\}} + \frac{Y_n^2}{\max\{\epsilon, \|Y_n^1\|\}}$$

and

$$\theta_{n+1} = \theta_n - a_n Y_n$$

Step 3: *Go to step 1.*

It can be shown (see [Andradóttir 1990]) that this new algorithm converges under much more general assumptions than the Robbins-Monro and Kiefer-Wolfowitz algorithms. The assumptions are more general in that we do not assume a priori that the sequence of iterates generated by the algorithm is bounded, nor do we assume that the objective function or any of its derivatives are bounded. As shown in lemma 1, such assumptions are necessary to prove the convergence of the Robbins-Monro and the Kiefer-Wolfowitz algorithms. When unbiased gradient estimates are used and $a_n = a/n$, where $a > 0$, this algorithm converges to the solution at the rate $n^{-1/2}$, which is also the convergence rate of the Robbins-Monro algorithm (if it converges!). This is the best convergence rate achievable in simulation.

The performance of algorithm 2 is very dependent on the parameter ϵ. When the parameter ϵ is chosen, one should keep the following facts in mind: The smaller ϵ is, the more advantage one is taking of the normalization $Y_n^i / \max\{\epsilon, \|Y_n^j\|\}$, $(i,j) \in \{(1,2),(2,1)\}$. This makes the performance of the algorithm independent of whether it is being used to optimize flat or steep functions, and can therefore speed up the convergence of the algorithm significantly on problems where algorithm 1 converges very slowly. On the other hand, the smaller the parameter ϵ is chosen, the larger steps algorithm 2 is allowed to take, so the sequence θ_n will have a larger asymptotic variance. Also, it is worth observing that the parameter ϵ can be chosen so that algorithm 2 behaves very much like algorithm 1 on the problems where algorithm 1 converges, but it will also converge on problems where algorithm 1 diverges. Indeed, when ϵ is large, then typically $\max\{\epsilon, \|Y_n^i\|\} = \epsilon$, for $i = 1,2$, so $Y_n = \frac{1}{\epsilon}(Y_n^1 + Y_n^2)$ for most n.

4. EMPIRICAL WORK

In this section, we will compare the empirical behavior of algorithms 1 and 2. We have already discussed the fact that algorithm 2 is more robust than algorithm 1 in that it converges under more general conditions. We now show that it sometimes converges significantly faster than the Robbins-Monro and the Kiefer-Wolfowitz algorithms in practise. This is particularly true on the class of problems where the objective function is very flat and the gradient evaluations aren't very noisy. Notice that algorithm 2 uses two estimates of the gradient of the objective function in each iteration, whereas algorithm 1 uses only one estimate. This means that for algorithm 2 to be faster than algorithm 1, on the average it has to converge in at most half the number of iterations of algorithm 1. Clearly this is always the case on the class of problems where algorithm 1 can diverge, so we only need to consider problems where both algorithms are guaranteed to converge.

Consider the problem of finding the minimum of the function $f(\theta) = \frac{1}{2}\ln(1+\theta^2)$. The solution of this problem is $\theta^* = 0$. The derivative of f is given by $f'(\theta) = \theta/(1+\theta^2)$. We want to compare the performance of the Robbins-Monro algorithm and of algorithm 2, when applied to solving this problem with noisy estimates of the gradient: to each evaluation of the gradient, we have added a sample of a $U[-0.01\sqrt{3}, 0.01\sqrt{3}]$-distributed random variable. This random variable has a standard deviation of 0.01. (Very similar results are obtained when the errors are normally distributed with mean 0 and standard deviation 0.01.) To study this problem we conducted the experiment described below. Both algorithms were started at $\theta_1 = 100$ and run until f' had been evaluated 2000 times: we did 2000 iterations of the Robbins-Monro algorithm and 1000 iterations of algorithm 2. This process was repeated 1000 times and figure 1 shows the average performance of the two algorithms. The x-axis shows the number of gradient evaluations (n) and the y-axis shows θ_n for the Robbins-Monro algorithm and $\theta_{[n/2]}$ for algorithm 2. To be able to compare the performance of the two algorithms more meaningfully, we used common random numbers to estimate the errors in the gradient evaluations. This means that for $1 \le m \le 1000$ and $1 \le n \le 2000$, the error in the nth gradient evaluation in the mth replication is the same for both algorithms. Observe that $f'(\theta_1) = 100/10001 \simeq 0.01$, so the errors are of the same order of magnitude as $f'(\theta_1)$. For the purposes of this experiment, we chose $a_n = 1/n$ and $\epsilon = 10^{-3}$.

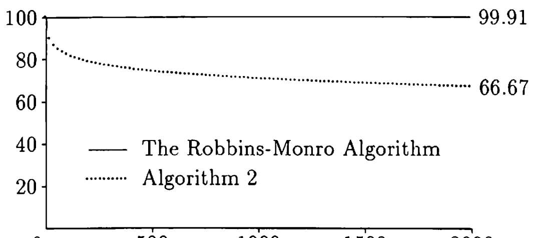

Figure 1. A comparison between the performance of the Robbins-Monro algorithm and algorithm 2.

The Robbins-Monro algorithm converges extremely slowly on this problem. After 2000 gradient evaluations, the average estimate of the solution is 99.91. Algorithm 2 converges somewhat faster on this problem: after 2000 gradient evaluations, its average estimate of the solution is 66.67. Notice however that when algorithm 2 is used, θ_n decreases very fast in the first few iterations, but the convergence becomes slow as time goes on. This is because the length of the step taken in iteration n is proportional to $a_n = 1/n$, so we are forcing the algorithm to take increasingly smaller steps. This is something that we want to do when we are close to the optimal solution to dampen out the effect of the errors in the evaluations of f', but when we are far from the solution, we want the algorithm to take large steps and move quickly towards the solution. This is the idea behind Kesten's acceleration [Kesten 1958]. The algorithms that we have considered update θ_n according to the following equation:

$$\theta_{n+1} = \theta_n - a_n Y_n \qquad (3)$$

where $-Y_n$ is the search direction and $a_n \|Y_n\|$ is the length of the step that the algorithm takes in the nth iteration. The sequence a_n is such that $a_n \to 0$ and $\sum_{n=1}^{\infty} a_n = \infty$. Typically $a_n = a/n$ where $a > 0$. Instead of updating θ_n according to equation (3), we want to update it according to

$$\theta_{n+1} = \theta_n - b_n Y_n$$

where b_n satisfies the same conditions as a_n, but b_n decreases only when we have reason to believe that θ_n is close to the solution. Clearly if Y_{n-1} and Y_{n-2} are of different sign, this is the case. Kesten therefore suggests choosing the sequence $\{b_n\}$ as described below:

$$b_n = a_{t_n}$$

where $t_1 = 1$, $t_2 = 2$ and for $n \geq 3$,

$$t_n = 2 + \sum_{k=3}^{n} I_{\{Y_{k-1} Y_{k-2} \leq 0\}}$$

where I_A is an indicator random variable. Figure 2 shows how the Robbins-Monro algorithm and algorithm 2 perform on the problem described above, when Kesten's acceleration is used.

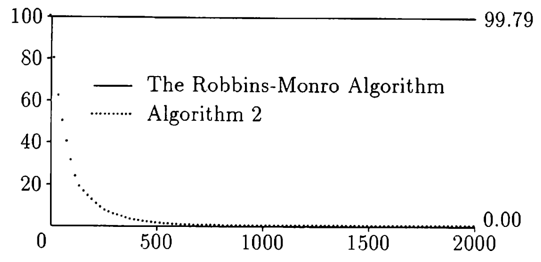

Figure 2. A comparison between the performance of the Robbins-Monro algorithm and algorithm 2, using Kesten's acceleration.

Observe that the performance of the Robbins-Monro algorithm improves only slightly when Kesten's acceleration is used. After 2000 gradient evaluations, the average estimate of the solution is 99.79. The performance of algorithm 2 is however greatly improved: after 500 gradient evaluations, the average estimate is 1.23 (the 90% confidence interval is 1.23 ± 0.26), after 1000 gradient evaluations, the average estimate is 0.05 (the 90% confidence interval is 0.05 ± 0.04) and after 2000 gradient evaluations, the average estimate is -0.26×10^{-3} (the 90% confidence interval is $-0.26 \times 10^{-3} \pm 0.83 \times 10^{-3}$).

This example shows that algorithm 2 sometimes converges much faster than the Robbins-Monro algorithm. This is particularly true when Kesten's acceleration is used. Kesten's acceleration does not seem to significantly increase the speed of convergence of the Robbins-Monro algorithm on this problem, whereas it increases the speed of convergence of algorithm 2 quite dramatically. This is partially because the normalization $Y_n^i / \max\{\epsilon, \|Y_n^j\|\}$ makes algorithm 2 take larger steps than the Robbins-Monro algorithm when we are far away from the solution, and partially because the search direction of the Robbins-Monro algorithm tends to change sign more often than the search direction of algorithm 2.

5. CONCLUSIONS

Classical stochastic optimization algorithms have severe problems associated with them: they converge extremely slowly on problems where the objective function is very flat, and they often diverge when the objective function is steep. We have developed a new stochastic optimization algorithm that is more robust than the older algorithms in that it is guaranteed to converge on a larger class of problems. At the same time, we have observed that it converges faster on a significant class of problems. As the parameters of the new algorithm can be chosen so that the new algorithm behaves very much like the older algorithms, while having the additional feature that it converges on a larger class of problems, we believe that this new algorithm should always be used in preference to the older algorithms. Additional work needs however to be done to guide the selection of values for the parameters of the new algorithm on specific problems.

ACKNOWLEDGEMENTS

This research forms a part of my dissertation [Andradóttir 1990], performed at the Department of Operations Research, Stanford University, under the supervision of Professor Peter W. Glynn. It was supported by the U.S. Army Research Office under contract DAAL-03-88-K-0063.

REFERENCES

Andradóttir, S. (1990), "Stochastic Optimization with Applications to Discrete Event Systems," Ph.D. Dissertation, Department of Operations Research, Stanford University, Stanford, CA.

Kesten, H. (1958), "Accelerated Stochastic Approximation," *Annals of Mathematical Statistics 29*, 1, 41 – 59.

Kiefer, J. and Wolfowitz, J. (1952), "Stochastic Estimation of the Maximum of a Regression Function," *Annals of Mathematical Statistics 23*, 3, 462 – 466.

Poljak, B.T. (1967), "A General Method of Solving Extremum Problems," *Soviet Mathematics Doklady 8*, 3, 593 – 597.

Robbins, H. and Monro, S. (1951), "A Stochastic Approximation Method," *Annals of Mathematical Statistics 22*, 3, 400 – 407.

Shor, N.Z. (1964), "On the Structure of Algorithms for the Numerical Solution of Optimal Planning and Design Problems," Dissertation, Cybernetics Institute, Academy of Sciences U.S.S.R. (In Russian.)