"HENRI COANDA"
AIR FORCE ACADEMY
ROMANIA

GERMANY

"GENERAL M.R. STEFANIK"
ARMED FORCES ACADEMY
SLOVAK REPUBLIC

INTERNATIONAL CONFERENCE of SCIENTIFIC PAPER
AFASES 2011
Brasov, 26-28 May 2011

A VERSION OF ROBBINS-MONROE ALGORITHM

Neculai CRISMARU

"Transilvania" University, Brasov, Romania

Abstract: *In the present paper, we present a better version than the Robbins-Monroe algorithm, which uses the speed of convergence of Robbins-Monroe strings of real functions.*

Mathematics Subject Classification 2010*: 62L20.*

Key words: *Robins-Monroe algorithm, speed of convergence, stochastic approximation.*

1. INTRODUCTION

Let $f:R\to R$ be an unknown function, and x^* the unique unknown root of f(.)([1],[2]). We suppose that:

1) f(.) is not decreasing on R

2) f(.) can be observed at each real value x from R, but each observation, noted by y_x, is not very accurate, she is corrupted by observations errors (noises). So, we have

$$y_x=f(x)+\varepsilon_x \qquad (1)$$

where ε_x: $\Omega\to R$, is the random variable which means noises of observations, and it has the next property:

$$E[\varepsilon_x]=0, \forall\, x\in R \qquad (2)$$

So, for f(.) we have random variables families $\{\varepsilon_x\}_x$, $x\in R$([4], p.2,5). Let be known, a sequence of the real numbers, $(a_n)_{n\in N^*}$, and $x_0\in R$, called the initial point. It will be made a sequence of the random variables, called the Robbins-Monroe sequence of the approximations, which has the next forms ([4], p5):

$$x_{n+1}(.)=x_n(.)+(a_{n+1})\, y_{x_n}(.), \qquad x_0\in R,$$

$$(a_n)_n\subset[0,+\infty) \qquad (3)$$

and

$$y_{x_n}(.)=f(x_n)+\varepsilon_{n+1}, \quad E[\varepsilon_{n+1}]=0, \quad y_{x_n}, \varepsilon_{n+1}: \Omega\to R, \forall\, n\in N \qquad (4)$$

We have an almost sure convergence theorem of the Robbins-Monroe sequences from (3).

Theorem 1. ([5], p.214-215)

In addition, we suppose that we have the next conditions:

(a) $E[y_{x_n}|x_1, x_2, \ldots, x_n]=f(x_n)$ (a.s.), $\forall\, n\in N$

(b) $a_n>0, \forall\, n\in N^*, \sum_{n=1}^{+\infty} a_n=+\infty, \sum_{n=1}^{+\infty} a_n^{\,2}<+\infty$

(c) There exists a continuously twice differentiable function on R, named Lyapunov function, V: $R\to R$, satisfying the following conditions:

(c.1.) there exists V''(.) on R, and V''(.) is bounded on R (that means, it's second derivative is bounded on R)

(c.2.) there are two functions a, b:[0,+∞)$\to$R, so that:

(c.2.1.) a(.) and b(.) is continuously on [0,+∞)

(c.2.2.) a(.) and b(.) is no decreasing on [0,+∞)

(c.2.3.) a(x) $\geq$0, b(x) $\geq$0, $\forall$ x$\in$[0,+∞)

(c.2.4.) a(0)=b(0)=0 and $\lim_{x\to\infty} a(x)=+\infty$

(c.2.5.)$a(|x|)\leq V(x)\leq b(|x|), \forall\, x\in R$

(c.3.) $\forall\, 0<\alpha_1<\alpha_2$, exists the real number $\inf_{\alpha_1\leq|x-x^*|\leq\alpha_2}\{V'(x-x^*)f(x)\}>0$

(d) $E[\,|y_x^2\,|\,|\,x_1, x_2, \ldots, x_n\,]<\sigma^2$ (a.s.)

In this conditions, we have that $x_n(.)\xrightarrow{a.s.}x^*$, when n $\to+\infty$ (or, with other words we have $\lim_{x\to+\infty} x_n=x^*$(a.s)).

In real applications, it will be frequently taken $a_n=\frac{1}{n}$([3], p.4). The practical method to use the Robbins-Monroe algorithm is:

Step 1. First, we elect (in random way) a real value, which is noted by x_0.

Step 2. Now, we make an observation (with noise) of f(.) in x_0, noted by $\tilde{y}_{x_0}$, this means that we have the error-corrupted observations, and the observation errors are noted by $\tilde{\varepsilon}_{x_0}$, so we have $\tilde{y}_{x_0}=f(x_0)+\tilde{\varepsilon}_{x_0}$. This observation is a real value, and it represents a selection of the random variable $y_{x_0}(.)=f(x_0)+\varepsilon_{x_0}(.)$, where $\tilde{\varepsilon}_{x_0}$ is a selection of the random variable $\varepsilon_{x_0}(.)$. In this moment, the only value which is obtained from external observation on f (.) in x_0, is the real number $\tilde{y}_{x_0}$ (which is called the observation of f(.) in x_0, corrupted by the observation errors).

Step 3. With the real number $\tilde{y}_{x_0}$ from step 2, and with (3), we obtain the real number

$$\tilde{x}_1=x_0-\frac{1}{1+0}\,\tilde{y}_{x_0} \qquad (5)$$

Now, we repeat the step 2 and 3, and we change on x_0 with $\tilde{x}_1$, obtaining another number $\tilde{x}_2$, and so on. After we repeat steps 2 and 3 for n-times, you get a string of real numbers, noted with $(\tilde{x}_n)_n$. Each value $\tilde{x}_n$ is a selection (real value) of random variable $x_{n+1}(.)$ from (3), and is achieved by formula (3) such that:

$$\tilde{x}_{n+1}=\tilde{x}_n-\frac{1}{n+1}\,\tilde{y}_{x_n},\ \forall\, n\in N \qquad (6)$$

If f(.) satisfies the conditions of the theorem 1, then we have:

$$(\tilde{x}_n)\xrightarrow[n\to+\infty]{} x^*. \qquad (7)$$

The solid application of the steps before, can generate a number of issues related to the convergence of string $(\tilde{x}_n)_n$, such as:

1. The first category of issues that may arise in the practical way, it is related to the influence of the observation error of function f (.) in each point $\tilde{x}_n$.

2. The second category of issues, it is related to the speed of convergence of string $(\tilde{x}_n)_n$ obtained with the formula (3). This problem is very important when this algorithm is implemented on computers for process, where the obtaining time for a single x* root has great importance in the management efficiency of that process through the computer.

3. Another category of problems are those related to the choice of initial point x_0, so that string (6) to be convergent, and with greater speed of convergence.

4. Problems may also appear in transforming this algorithm - which is a sequential arrangement, in a parallel one.

5. A last category of problems that may appear related to the algorithm (3) of Robbins-Monroe, is concerned about the optimum way in which we choose $(a_n)_n$ string, to fulfill the conditions of Theorem 1 (or equivalent), and to provide an increased speed of convergence.

In this paper we will mainly deal with the second category of issues, and partly with the last ones. There are examples of functions, which provides us $\tilde{y}_{x_n}$ observation strings and which obtain converged (6) form strings, but which have little convergence speed. We make the observation that, checking the theorem 1 hypothesis in practical situations, can create

technical problems (depending on the complexity of the correspondence law of that function).

2. THE MAIN RESULTS

Now, we start this section with an example about the convergence speed of a particular function.

Example 1. We consider the function

$$f:[0,+\infty)\to R, f(x)=\sqrt{x}-2, \quad (8)$$

having as unique root $x^*=4$. It will be taken another helping function, noted by $f_1(.)$

$$f_1(x)=\begin{cases}-2, x\le 0\\ f(x), x\in(0,10],\\ \sqrt{10}-2, x\ge 10\end{cases} \quad f_1:R\to R, \quad (9)$$

$f_1(.)$ is bounded on R.

This function took this form because we make the assumption that it is known before (additional information) that the only root of function $f_1(.)$ is in the range [0, 10], and for that the new function to be bordered. To be noticed that the equations $f(x)=0$ and $f_1(x)=0$ have the same root, $x^*=4$. From now on (in this example) we will work only with function $f_1(.)$.

For the function $f_1(.)$ in this example, it can be created a Lyapunov function that to fulfill the conditions of Theorem 1 and that to look like the Robbins-Monroe string from the previous example, string attached on function $f_1(.)$ that satisfies the assumptions (a) and (d) of Theorem 1 with $a_n=1/n$, $n\ge 1$.

If, for the function $f_1(.)$ from example 1. it is assumed that we have observations about it, in any point $x\in R$, a certain error, known in advance, then if it is chosen the initial point x_0 selected too far away from the unique root of $f_1(.)$, then the algorithm Robbins-Monroe converges very slow to this root. For example, if you take as the initial point on the $x_0=12.0$, and if we assume that the error of observation of $f_1(.)$ in each point x of R has the order 10^{-2} (meaning that in all $x\in[0,+)$, we have $|y_x - f(x)| < 0.01$, where $y_x= f_1(x) + \varepsilon_x$, with error ε_x that has the property $|\varepsilon_x| < 0.01$), then are obtained the next terms of the Robbins-Monroe string attached of $f_1(.)$, x_0, and the maximum precision 0.01. These results were obtained with a C++ program and are presented following the model in [6], p. 23. In this program, the initial value of $x_0=12.0$, and the number of iteration, n= 10. The final results are the following (presented with 6 decimal) presented like in ([7], p.20) and [6], p.23):

Table 1.

x[0]=12.000000
x[1]=10.837722778
x[2]=10.256584167
x[4]=9.585001945
x[5]=9.366808891
x[6]=9.190338135
x[7]=9.043643951
x[8]=8.917922974
x[9]=8.808657646
x[10]=8.711833954

The average error is equal to the Em = -002267.

Here $x[n+1]=x[n]-\frac{1}{n+1}y_{xn}$, where

$y_{xn}=f(x[n])+\varepsilon_{n+1}$, n=0,2,…,8, and

$\varepsilon_{n+1}=z$, from the (n+1) iteration on C++ program. Also, the error is the average (simple arithmetic average) of all the errors recorded in the 9 observations on $f_1(.)$, i.e. $Em=(\varepsilon1+...\varepsilon9)/9=0.002267$.

The crop was done to the 7th decimal (and have been retained the first 6 decimals). We give below the centralized values table of $f_1(.)$ function, seen (with given perturbations data of ε_i, i = 1,2, ...,8) in the points x[i], values noted by $y_{x[i]}$, i = 1,2,..,8.

Table.1'.

(i) Number of Iterations (1)	($y_{x[i]}$) The observed value of $f_1(.)$ on x[i] (2)	(f(x[i])) The exact value of $f_1(.)$ on x[i] (3)	The observations errors ε_{i+1} (on x[i]) ($\varepsilon_{i+1}=y_{x[i]} - f(x[i])$) (4)
0	$y_{x[0]}$= 1.162278	f(x[0])= 1.162278	ε_1 = 0.000000
1	$y_{x[1]}$= 1.162278	f(x[1])= 1.162278	ε_2 = 0.000000
2	$y_{x[2]}$= 1.162278	f(x[2])= 1.162278	ε_3 = 0.000000
3	$y_{x[3]}$= 1.136622	f(x[3])= 1.141522	ε_4 = - 0.004900
4	$y_{x[4]}$= 1.090965	f(x[4])= 1.095965	ε_5 = - 0.005000
5	$y_{x[5]}$= 1.058824	f(x[5])= 1.060524	ε_6 = - 0.001700
6	$y_{x[6]}$= 1.026857	f(x[6])= 1.031557	ε_7 = - 0.004700
7	$y_{x[7]}$= 1.005765	f(x[7])= 1.007265	ε_8 = - 0.001500
8	$y_{x[8]}$= 0.983389	f(x[8])= 0.986289	ε_9 = - 0.002900
9	$y_{x[9]}$= 0.968238	f(x[9])= 0.967938	ε_{10} = 0.000300

If we operate 100 iteration (in same conditions above), the following results are obtained:

x[100]=6.934031010

The average error is equal to the Em= - 0.000495

If we do now (with the same function in the same conditions) 500 iteration, we obtain:

Table 2.

x[493]=6.066642284
x[494]=6.065702915
x[495]=6.064774990
x[496]=6.063833237
x[497]=6.062898636
x[498]=6.061974525
x[499]=6.061045647
x[500]=6.060128212
x[501]=6.059205055

The average error is equal to the Em= - 0.000041

If we take 1000 iterations, in the same conditions (x_0 = 12.0 and 6-digit accuracy), we have:

Table 3.

x[992]=5.765196800
x[993]=5.764791965
x[994]=5.764384747
x[995]=5.763985157

x[996]=5.763581753
x[997]=5.763183594
x[998]=5.762779713
x[999]=5.762382984
x[1000]=5.761984348

The average error is equal to the Em= - 000071

We notice that after 1000 iteration, we are still far away from the root x^*= 4. After 5000 iteration we obtain the following results:

x[5]=5.216153145 and the error in the observations of $f_1(.)$ is 0.002800 and f_1(x[4,999])=0.283902 and
y(x[4,999])=0.286702

The average error is equal to the Em= -0.000069

After 10000 iteration, we obtain:

x[10000]=5.033749104 and the error in the observations of $f_1(.)$ is - 0.000600

This demonstrates that, in this case, the weak convergence of the algorithm, is caused by the chosen value for x_0, too far from x*=4. It is clear that the classic Robbins-Monroe algorithm, applied to this modified functions provided by departure, x_0 = 12.0, it is weakly convergent (by unique root x*=$\sqrt{2}$).

Now we slightly change the Robbins-Monroe algorithm form, about which we will show that is faster than Robbins-Monroe string.

Let it be the $(x_n)_n$ random variables string, attached to the function f:R$\rightarrow$ R, from the first part, string given by the recurrence formula:

$x_{n+1}(.)=x_n(.)+(a_{n+1})\, y_{x_n}(.), x_0 \in R$, cu

$y_x=f(x)+\varepsilon_x$ (10)

which has a weak convergence to the root x*. Usually, the weak convergence of this series is because of the high speed low variation of $f_1(.)$ in a neighborhood V_{x^*} of x*, meaning that the fraction $\frac{|x_{n+1}(\omega)-x^*|}{|x_n(\omega)-x^*|}$ has a value nearly equal to 1 (or equivalently, the difference $|x_{n+1}-x^*|$ is approximately equal to x^*)([6], p. 21-22). We suppose that the function $f_1(.)$ has all the properties from part A, and meet the assumptions of the theorem 1. Then we have:

and f (x[9999])=0.243607
and y(x[9999])=0.243007

The average error is equal to the Em= -0.000091

As we can see, studying the numerical results above (including those in table 1) shall we consider that, starting from

x0 = 12.0, we get a weakly convergent Robbins-Monroe string.

Theorem 2. If the function $f_1(.)$ has the above conditions and assumptions of the theorem 1, then the function $g_1(x)=Kf(x)$, K>1, has the same assumptions of the theorem 1, and so, the Robbins-Monroe string attached on the function $g_1(.)$, is almost certainly convergent to x* root of his $g_1(.)$ ($g_1(.)$ and $f_1(.)$ have the same root, x*=4).

Proof: Simple checking:

Returning to example 1, we make a new Robbins-Monroe string for $f_1(.)$ of (9) given by $x_{n+1}(.)=x_n(.)-\frac{1}{2n+1}(3\, y_{x_n}(.))$, $x_0 \in R$, $\forall n \in N$. We numerically simulate this variant of the algorithm, for the function $f_1(.)$ f_1:R $\rightarrow$R, $f_1(x)=\begin{cases}\sqrt{x}-2.0, x\in[0,10]\\ \sqrt{10}-2.0, x\geq 10\\ -2, x\leq 0\end{cases}$, x_0=12.0 and

$g_1(x)=\begin{cases}-6, x\leq 0\\ 3*(\sqrt{x}-2), x\in(0,10]\\ 3*(\sqrt{10}-2), x>10\end{cases}$, g_1:R $\rightarrow$R.

For algorithm Robbins-Monroe, applied his $f_1(.)$, see some results in the part A, in tables 1, 1', 2, and 3. We run the same number of iterations (10, 100, 500, 1000, 5000, and 10,000) starting from the same initial point x1 = 12.0, and with the same precision display (as

number of decimals displayed) in the algorithm Robbins-Monroe for f1(.) and 3f1(.), implemented in the C++ program for the function f1 (.), and with another C++ program, for function g1(.), with increase of the errors by multiplication with 3 to function $g_1(.)$, we reach to the results shown in the table below:

Table 4.

Nr. de iteration (n)	$f_1(.)$	$3f_1(.)=g_1(.)$
10	x[10]=8.711833954	x[10]=8.140531540
100	x[100]=6.934031010	x[100]=6.013100624
500	x[500]=6.060128212	x[500]=5.158834457
1000	x[1000]=5.761984348	x[1000]=4.907297134
5000	x[5000]=5.216153145	x[5000]=4.508469105
10000	x[10000]=5.033749104	x[10000]=4.394921303

For the same function $g_1(.)$, but with the stagnation of the order of magnitude of the error of observation at the multiplication by 3, the simulation gives the following results:

Table 5.

n=10, x[10]=8.137619972
n=100, x[100]=6.011349201
n=500, x[500]=5.157924652
n=1000, x[1000]=4.906517982
n=5000, x[5000]=4.507922173
n=10000, x[10000]=4.394413471, if we compare them with those in table 4. we see that this variant of the Robbins-Monroe algorithm (with or without raising the error of observation from $f_1(.)$ to $g_1(.)=3f_1(.)$) it is much more efficient than the "classic".

REFERENCES:

1. Orman, G. V., *Handbook of Limit Theorems and Stochastic Approximation*, Brasov: "Transilvania" University Press (2003), p.131-138.
2. Tze Leung Lai, *Stochastic approximation, The Annals of Statistics*, 2005, Vol 31, No. 2, Institute of Mathematical Statistics (2003), p. 391-406.
3. Kushner, H. J., Yin, G. G., *Stochastic Approximation Algorithms and Applications*, Springer (2003), p. 2- 7.
4. Chen, H. F., *Stochastic Approximation and Its Applications*, Kluwer Academic Publishers (2002), p.1-21.
5. Morozan T., *Stabilitatea sistemelor cu parametrii distribuiti*, Bucharest: Editura Academiei Republicii Socialiste Romania (1969), p. 214-223.
6. Maruster S., *Metode numerice in rezolvarea ecuatiilor neliniare*, Bucharest: Editura Tehnica (1981), p. 21-22.
7. Simionescu I., Draga M., Moise V., *Metode numerice in tehnica-aplicatii in Fortran*, Bucharest: Editura Tehnica (1995).