Eötvös Loránd University

Faculty of Sciences

Department of Mathematics

SPSA with Momentum

Supervisor: Balázs Csanád Csáji

Author: CHTIBA REDA
Msc in APPLIED MATHEMATICS

Budapest, 12/09/2021

Introduction

In many optimization problems, having access to direct gradient information is convenient and reliable, it enable us to use algorithms such as the gradient descent method. There are however, a large number of problems where the direct gradient measurements are not available. Due to this issue, there is a lot of interest in algorithms that do not depend on direct gradient measurements. These algorithms are called gradient-free algorithms, and in this report we will be talking about two of them : Finite Difference Stochastic Approximation algorithm (FDSA) and Simultaneous Perturbation Stochastic Approximation algorithm (SPSA). We shall also present an enhanced version called the Momentum Acceleration of the SPSA and we will compare this enhanced version with the FDSA.

Contents

1

FDSA

1.1 Notation

The FDSA algorithm is an alternative to the stochastic gradient algorithm, where we only have access to noisy measurements of the objective function. The key idea to this algorithm is to replace the stochastic gradient which we don't have by an estimate of it, denoted by $\hat{g}_k(\hat{\theta}_k) = \frac{\partial L}{\partial \theta}$. This estimate approximates the stochastic gradient using the Finite-Difference Method, and in our case we will use the two-sided FD,then the gradient estimate will have the following form :

$$\hat{g}_k(\hat{\theta}_k) = \begin{bmatrix} \frac{y_k(\hat{\theta}_k + c_k.\xi_1) - y_k(\hat{\theta}_k - c_k.\xi_1)}{2c_k} \\ \vdots \\ \frac{y_k(\hat{\theta}_k + c_k.\xi_p) - y_k(\hat{\theta}_k - c_k.\xi_p)}{2c_k} \end{bmatrix}$$

Such that : ξ_i is a column vector with p components, 1 in it's i-th row and o everywhere else and c_k is a gain coefficient .
we give the recursive procedure for the FDSA algorithm :

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k.\hat{g}_k(\hat{\theta}_k)$$

where a_k is a gain coefficient (i.e a positive scalar valued sequence which is used to control the magnitude of the steps of the algorithm).

1.2 Convergence

We will present two sets of conditions, each one having it's own interpretation (statistics and engineering), which will be used for the convergence theorem.

Statistical conditions:

-A_1: $a_k > 0, c_k > 0, a_k \to 0, c_k \to 0, \sum_{k=0}^{\infty} a_k = \infty$, $\sum_{k=0}^{\infty} a_k c_k < \infty$, and $\sum_{k=0}^{\infty} a_k^2/c_k^2 < \infty$.

-A_2: There is a unique minimum θ^* such that for every $\eta > 0, \inf_{\|\theta-\theta^*\|>\eta} \|g(\theta)\| > 0$ and $\inf_{\|\theta-\theta^*\|>\eta} [L(\theta) - L(\theta^*)] > 0$

-A_3:For all i and k, $E\left[\left(\varepsilon_k^{(i+)} - \varepsilon_k^{(i-)}\right) \mid \mathfrak{J}_k\right] = 0$ a.s. and $E\left[\left(\varepsilon_k^{(i\pm)}\right)^2 \mid \mathfrak{J}_k\right] \leq C$ a.s. for some $C > 0$ that is independent of k and θ.

-A_4: The Hessian matrix $H(\theta) = \partial^2 L/\partial\theta\partial\theta^T$ exists and is uniformly bounded in norm for all $\theta \in \mathbb{R}^p$

Where, $\varepsilon_k^{(i\pm)} = \varepsilon\left(\hat{\theta}_k \pm c_k\xi_i\right)$, $\mathcal{J}_k = \left\{\hat{\theta}_0, \hat{\theta}_1, \ldots, \hat{\theta}_k\right\}$

Engineering conditions:

-B_1: $a_k > 0, c_k > 0, a_k \to 0, c_k \to 0, \sum_{k=0}^{\infty} a_k = \infty$, and $\sum_{k=0}^{\infty} a_k^2/c_k^2 < \infty$.

-B_2:Let $g(\theta)$ be continuous on $\mathbb{R}^p$. With $Z(\tau) \in \mathbb{R}^p$ representing a time-varying function (τ denoting time), suppose that the differential equation given by $d\boldsymbol{Z}(\tau)/d\tau = -g(\boldsymbol{Z}(\tau))$ at point θ^* has the following two requirements: (i) For every $\eta > 0$, there exists a $\delta(\eta)$ such that $\|Z(\tau) - \theta^*\| \leq \eta$ for all $\tau > 0$ whenever $\|Z(0) - \theta^*\| \leq \delta(\eta)$, and (ii) there exists a δ_0 such that $Z(\tau) \to \theta^*$ as $\tau \to \infty$ whenever $\|Z(0) - \theta^*\| \leq \delta_0$.)

-B_3: $\sup_{k\geq 0} \left\|\hat{\theta}_k\right\| < \infty$ a.s. Further, $\hat{\theta}_k$ lies in a compact subset of the "domain of attraction" for the differential equation in B_2 infinitely often.

-B_4: Same as condition A_3

-B_5: The third derivatives $L'''_{iii}(\theta)$ are continuous and uniformly bounded for all $i = 1, 2, \ldots, p$ and $\theta \in \mathbb{R}^p$.

Theorem:

Let's assume that the conditions of either one of the two sets given above hold. let's also suppose that $\theta \in \mathcal{R}^p$ and $\theta^\star$ is the unique minimum of the loss function L, then for the FDSA ,

$$\hat{\theta}_k \to \theta^* \text{ a.s. as } k \to \infty$$

1.3 Asymptotic Normality

Theorem: For gain sequences choice : $a_k = \frac{a}{(k+1+A)^\alpha}$ and $c_k = \frac{c}{(k+1+)^\gamma}$, where a,c,$\alpha$,$\gamma$ are positive, A is non-negative. Let's assume that $\beta = \alpha - 2 \times \gamma > 0$, $3 \times \gamma - \frac{\alpha}{2} \geq 0$, and that the algorithm converges then we have the following result :

$$k^{\beta/2}\left(\hat{\theta}_k - \theta^*\right) \xrightarrow{\text{dist.}} \mathcal{N}\left(\mu_{\text{FD}}, \Sigma_{\text{FD}}\right)$$

Where , μ_{FD} is a mean vector that depends on the Hessian $H\left(\theta^*\right)$ and the third derivative $L'''\left(\theta^*\right), \Sigma_{\text{FD}}$ is a covariance matrix that depends on $H\left(\theta^*\right)$, and both of these two parameters depend on the coefficients a, α, c, and γ.

Remark: the previous results says that the rate at which $\hat{\theta_k}$ approaches $\theta^\star$ is proportional to $k^{-\beta/2}$ for large k.

Implications:

If condition A_1 holds then $\alpha > 1/2$ and $\gamma > 0$, which implies that $0.6 < \alpha \leq 1$, $0.1 < \gamma < 1/2$ and $\alpha - \gamma > 1/2$. This implies again that β is maximized at $\alpha = 1$ and $\gamma = 1/6$, then it follows that for large k, the rate of convergence is proportional to $k^{-\sqrt{3}}$.

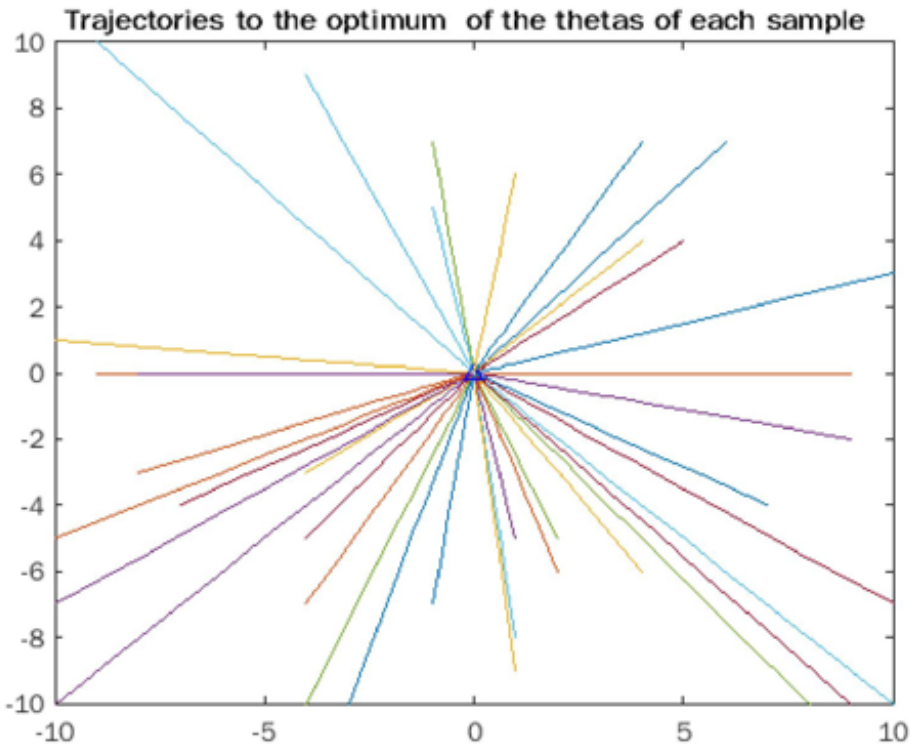
1.4 Numerical figures

In this subsection , we will present three plots from the implementation of the algorithm over 40 samples and each sample run trough 100 iterations. We shall consider, a quadratic test loss function mainly $L(\theta_1, \theta_2) = \theta_1^2 + \theta_2^2$, where $\theta = [\theta_1, \theta_2]^T$ and we note that $L(\theta) = 0$ at $\theta = [0, 0]^T$(the optimum where the loss function is minimized). We also assume that the function measurements are taken with i.i.d noise having distribution $\mathcal{N}(0, 1)$. We let, $\hat{\theta_0}$ (initial value of the parameter) to be generated randomly,we also choose the coefficient to be in the procedure as : A=10, c=0.05, a=0.5, $\alpha = 0.602$ and $\gamma = 0.101$.

-The first figure, represents the 40 (of each sample) trajectories of the $\hat{\theta_k}$ trough the 100 iterations(k=1:100), and it can be seen that they all reach the optimum value no matter at which point the procedure started

-The second figure, each curve corresponds to the plot of the distances of the $\hat{\theta_k}$ from the optimum trough all iterations , and there 40 curves, all of them seems to converge to zero, which means that as the number of iterations grows, the algorithm gives a

Figure 1.1: First figure



better approximation to the optimum.

-The third plot, is a curve where each point on it represents the standard deviation of the distances across all 40 samples , per iteration. We remark that as the number of iterations grows, the standard deviation of the distances converges to zero.

Figure 1.2: Second figure

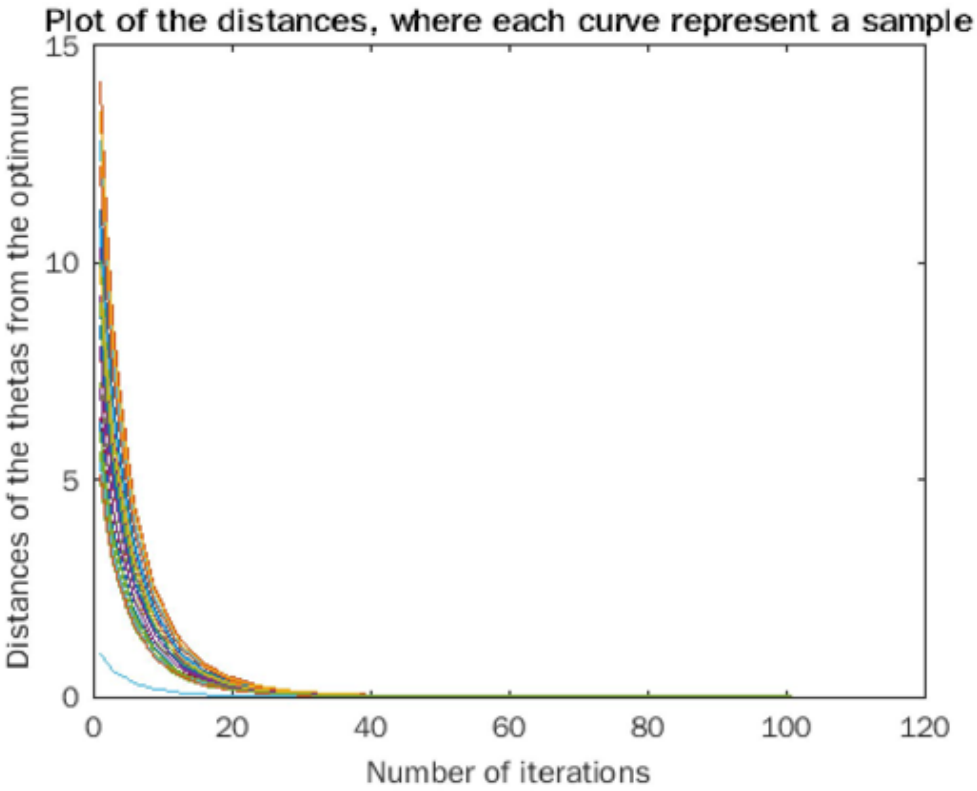
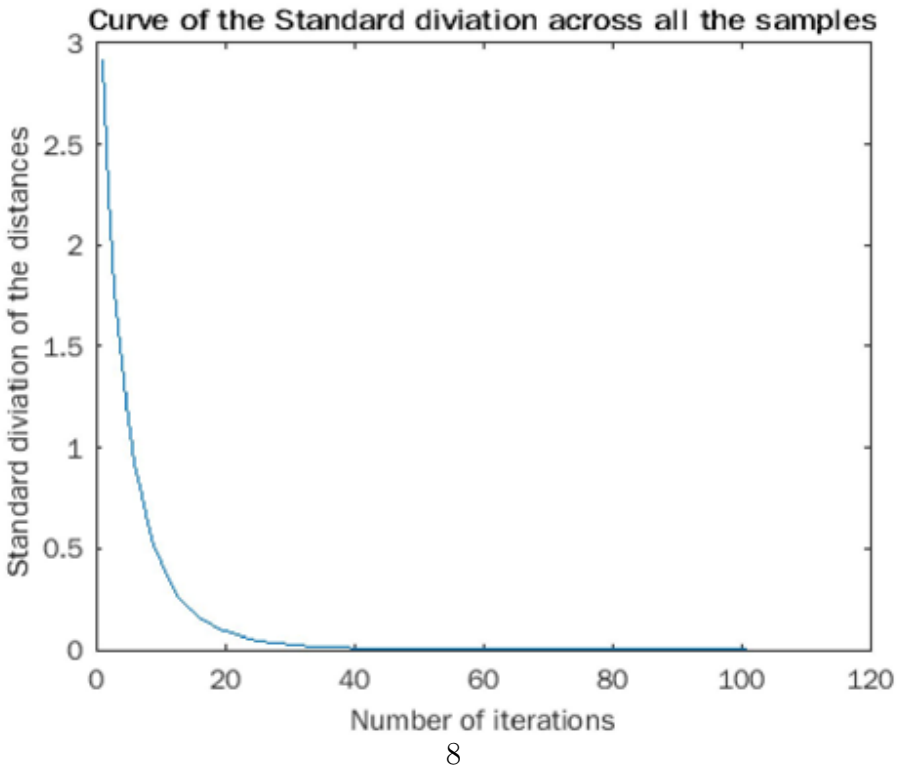


Figure 1.3: Third figure



2

SPSA

2.1 SPSA Algorithm

The SPSA algorithm, differs from the FDSA, where instead of using 2.p loss measurements in the former, we will this time use only a fixed number of loss measurements which is 2. We will rely on the method of Simultaneous Perturbation, to achieve such a feat, where instead of perturbing each coordinate of theta every time , we will perturb all the coordinate of theta one time, randomly, using a generated random vector (that has same dimension as the parameter) from a Bernoulli distribution.
Similarly to the FDSA, we will use consider the case where we only have acces to the noisy measurements. Which means that the algorithm is based on the $y(\theta) = L(\theta) + \varepsilon$. For this algorithm, we will use a different way to approximate the stochastic gradient, where we shall inject Monte-Carlo randomness in the search direction as the algorithm iterates towards a solution, additionally we will combine this random choice with a similar method to the Finite-Difference aproximation. The gradient estimate for the SPSA will have the following form :

$$\hat{g}_k(\hat{\theta}_k) = \begin{bmatrix} \dfrac{y_k(\hat{\theta}_k + c_k.\Delta_k) - y_k(\hat{\theta}_k - c_k.\Delta_k)}{2c_k.\Delta_{k_1}} \\ \vdots \\ \dfrac{y_k(\hat{\theta}_k + c_k k.\Delta_k) - y_k(\hat{\theta}_k - c_k k.\Delta_k)}{2c_k.\Delta_{k_p}} \end{bmatrix}$$

Where Δ_k is the random perturbation vector with p components and with zero mean and c_k is a positive scalar.

we give the recursive procedure for the SPSA algorithm :

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k.\hat{g}_k(\hat{\theta}_k)$$

where a_k is a gain coefficient.

2.2 Convergence

Conditions for convergence: Similarly to the previous chapter, we will introduce a set of conditions, which are required by the next theorem.

Engineering conditions:

-C_1: Same as condition B_1

-C_2: Same as condition B_2

-C_3: Same as condition B_3

-C_4:For all$k, E\left[\left(\varepsilon_k^{(+)} - \varepsilon_k^{(-)}\right) \mid \Im_k, \Delta_k\right] = 0$ and the ratio of measurement to perturbation is such that $E\left[\left(y\left(\hat{\theta}_k \pm c_k\Delta_k\right)/\Delta_{ki}\right)^2\right]$ is uniformly bounded (over k and i).

-C_5: the loss function L is three times continuously differentiable and bounded on $\mathcal{R}^p$

-C_6: The $\{\Delta_{ki}\}$ are independent for all k, i, identically distributed for all i at each k, symmetrically distributed about zero and uniformly bounded in magnitude for all k, i.

Theorem:

Let's assume that all the previous conditions holds, and we suppose again that $\theta^\star$ is the unique minimum. Then for the SPSA algortihm we have the following result :

$$\hat{\theta}_k \to \theta^* \text{ a.s. as } k \to \infty$$

2.3 Asymptotic Normality

Additional conditions for Asymptotic Normality

-C_7:The continuity and equicontinuity assumptions about $E\left[\left(\varepsilon_k^{(+)} - \varepsilon_k^{(-)}\right)^2 \mid \Im_k\right]$ in Spall (1992, Prop. 2) [2]

-C_8:$H(\theta^*)$ is positive definite where $H(\theta)$ is the Hessian matrix of $L(\theta)$. Further, let λ_i denote the i th eigenvalue of $aH(\theta^*)$ (the a here is the a in a_k). If $\alpha = 1$, then $\beta < 2\min_i(\lambda_i)$.

-C_9:$E\left(\Delta_{ki}^2\right) \to \rho, E\left(\Delta_{ki}^{-2}\right) \to \rho'$, and $E\left[\left(\varepsilon_k^{(+)} - \varepsilon_k^{(-)}\right)^2 \mid \Im_k\right] \to \rho''$ for strictly positive constants ρ, ρ', and ρ'' (a.s. in the latter case) as $k \to \infty$ (often, $E\left(\Delta_{ki}^2\right)$ and $E\left(\Delta_{ki}^{-2}\right)$ will be equal to ρ and ρ', respectively).

Theorem:
Suppose that $a_k = \dfrac{a}{(k+1+A)^\alpha}$ and $c_k = \dfrac{c}{(k+1+)^\gamma}$, where a,c,$\alpha$,$\gamma$ are positive, A is non-negative, $\beta = \alpha - 2 \times \gamma > 0$, $3 \times \gamma - \frac{\alpha}{2} \geq 0$, all the conditions from C_1 up to C_9 holds , then for the SPSA algorithm, we have the following result :

$$k^{\beta/2}\left(\hat{\theta}_k - \theta^*\right) \xrightarrow{\text{dist.}} \mathcal{N}\left(\mu_{\text{SP}}, \Sigma_{\text{SP}}\right) \text{ as } k \to \infty$$

where μ_{SP} and Σ_{SP} are respectively the mean vector and the covariance matrix

2.4 Numerical figures

In this section, similarly to the previous Implementation section of the FDSA we will also look at three figures, where each figure has the same explanation given in the case of the FDSA algorithm.

Figure 2.1: First figure

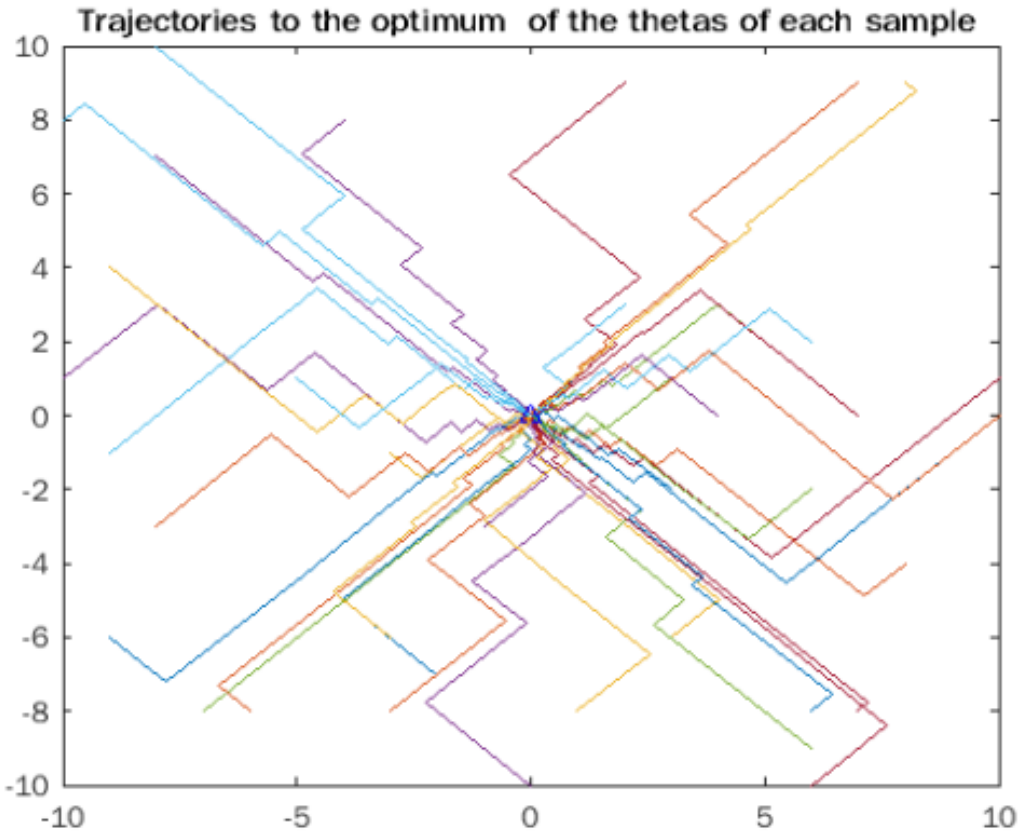


Figure 2.2: Second figure

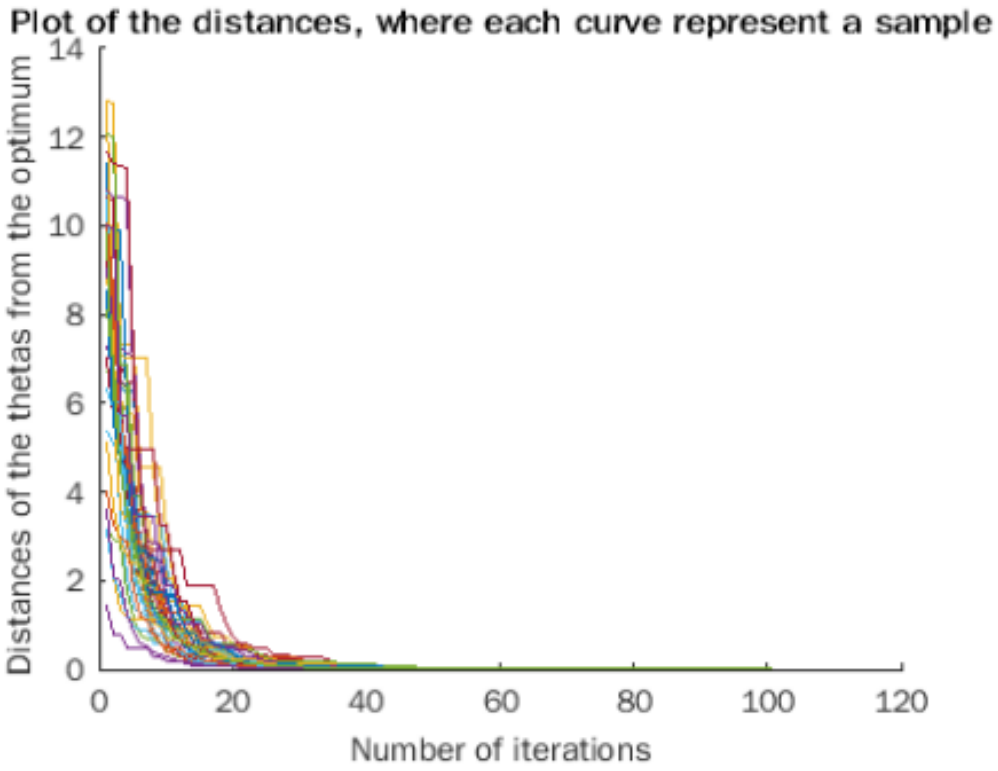
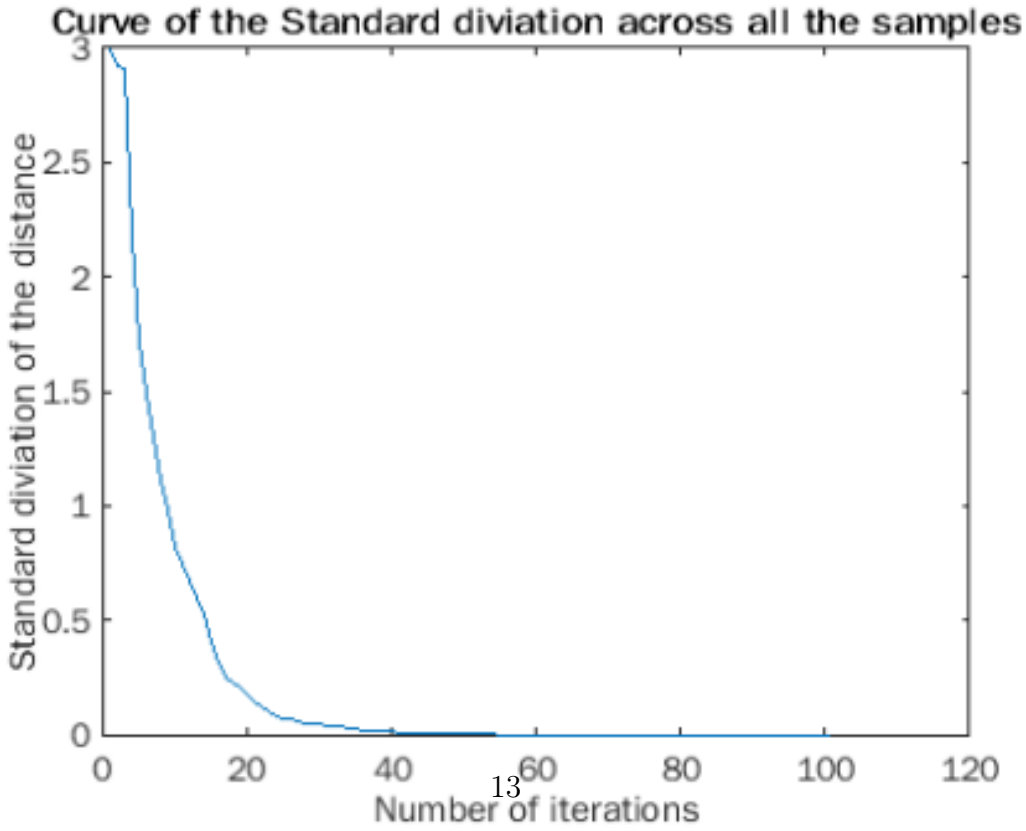


Figure 2.3: Third figure



3

SPSA with momentum

3.1 SPSA with momentum

There has already been done work on gradient descent with momentum done by Mathematician Boris Polyak, where the idea of momentum is used to accelerate the optimization process using gradient descent algorithm. Where the recursive procedure for gradient descent with momentum has the following form :

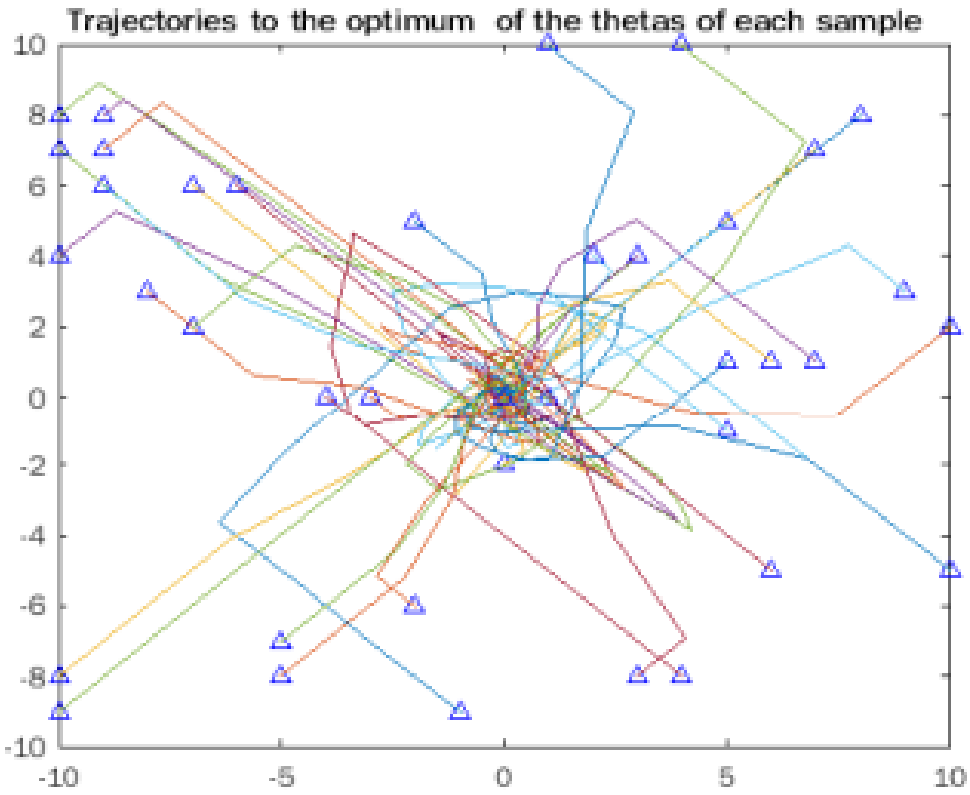
$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k.\hat{g}_k(\hat{\theta}_k) + b.(\hat{\theta}_k - \hat{\theta}_{k-1})$$

Where $\hat{g_k}$ is the gradient of an objective function, a_k a a gain coefficient, b is the momentum coefficient and $\hat{\theta}_k$ is the parameter estimate at the k-th iteration. The key here , is that this enhanced method, uses additional information of the history of the algorithm to make the algorithm more efficient.
We would like to extend this accelaration idea , to the SPSA algorithm, mainly in the recursive procedure of the SPSA algorithm we also add the momentum part :

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k.\hat{g_k}(\hat{\theta}_k) + b.(\hat{\theta}_k - \hat{\theta}_{k-1})$$

Where $\hat{g}_k$ this time represent the gradient approximation using the simultaneous perturbation method.
First, in the next section, we will try to show numerically that this method indeed work in the sense that the trajectory of the thetas in the algorithm will converge to an optimum, and that the distance between the thetas and the optimum will also converge to zero. Later in the next chapter, we will try to prove numerically that this enhanced version of the SPSA is more efficient than the normal SPSA Algorithm.

Figure 3.1: First figure



3.2 Numerical proof that the SPSA with momentum works

The next three figures, have the same explanation as the previous description given in the previous chapter. But, the difference this time, is that : Unlike the previous two algorithm, which have convergence theories behind them and it make sense that the trajectories should behave in the way shown in the figures, (i.e, trajectories of thetas converging to the optimum). This time, for this modified SPSA algorithm, as far as I searched , I unfortunately did not stumble upon any convergence theory material. The first figure, shows a sample of 40 experience, starting every thing from the beginning, each having different starting point and all converging to the optimum, this shows the potential to be a numerical proof. The inference of the remaining two figures, also suggest that this enhanced SPSA tend to have similar convergence as to the previous two algorithms.

Figure 3.2: Second figure

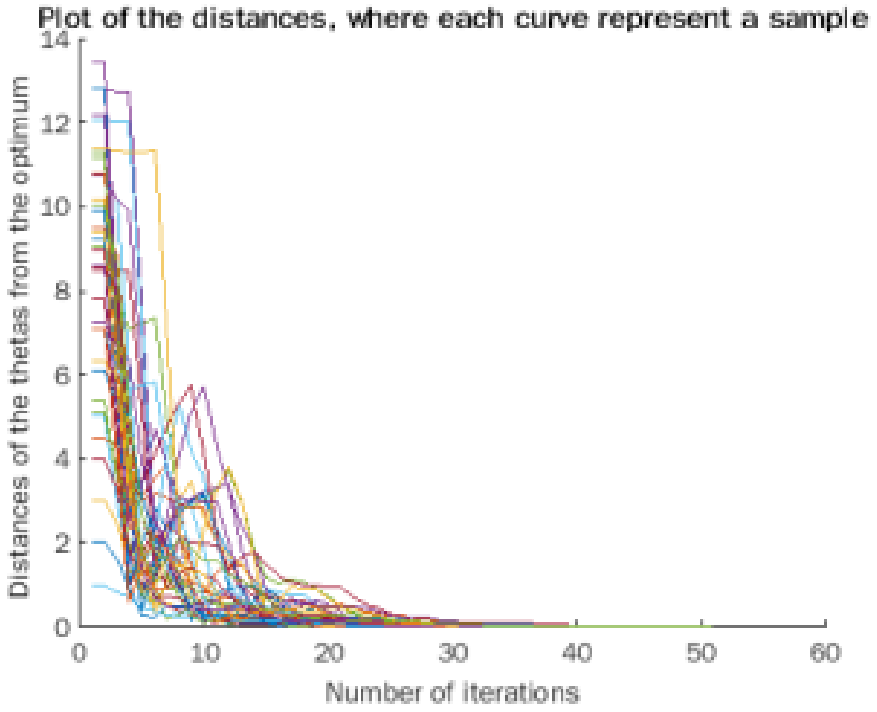
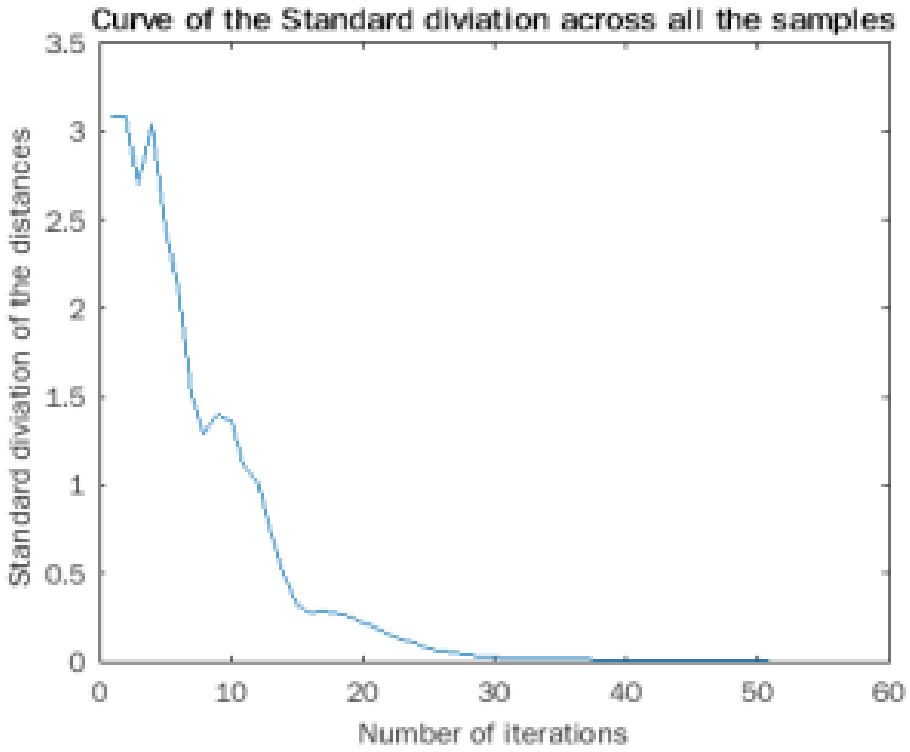


Figure 3.3: Third figure



4

SPSA with momentum and FDSA

4.1 Comparison between SPSA and FDSA

The first obvious difference between FDSA and SPSA, is that the SPSA algorithm requires only 2 loss measurements per iterations, in contrast to the FDSA algorithm that requires 2.p loss measurements per iterations. This difference has the potential to imply that the SPSA is more efficient than the FDSA, but that's only in the case if the SPSA won't require more iterations to give an optimum estimate as good as the optimum estimate from the FDSA algorithm, since it's obvious that the Finite Difference method has better approximation to the gradient than the Simultaneous Perturbation method. Mathematician Spall, discussed that using the asymptotic normality of both the FDSA and SPSA,assuming that $3.\gamma - \frac{\alpha}{2} > 0$ with $\alpha = 0.602$, $\gamma = 0.101$ and if both algorithm are using the same gain sequences a_k and c_k we get the following result .

Theorem The SPSA and FDSA recursions result in the same level of statistical accuracy for a given number of iterations, although SPSA uses only 2 loss measurements.

Remark: The previous theorem, can be reformulated to say that, as the number of loss measurements in both procedures gets large, this implies that the p-fold saving per iterations becames a p-fold saving in the over all optimization process.

4.2 Comparaison between SPSA with momentum and FDSA

Numerical proof that SPSA with momentum is better than FDSA

Let's consider the example where the parameter $\theta \in \mathcal{R}^2$, $\theta = [\theta_1, \theta_2]^T$ and the loss function $L(\theta_1, \theta_2) = \theta_1^2 + \theta_2^2$, and we note that $L(\theta) = 0$ at $\theta = [0, 0]^T$(the optimum where the loss function is minimized). We also assume that the function measurements are taken with i.i.d noise having distribution $\mathcal{N}(0, 1)$. We let, $\hat{\theta_0}$=$[0.1, -0.6]^T$ $\hat{\theta_1}$=$[0.1, -0.6]^T$be the initial value of the parameter at the first two steps of both the FDSA and SPSA with momentum algorithms(the reason we are taking two initials value is due to implementation of the SPSA with momentum algorithm), we also take in both procedures similar coefficients (the choice of the coefficient was chosen the best for the FDSA), mainly : A=10, c=0.05, a=0.3, $\alpha = 0.602$ and $\gamma = 0.101$. First result we can get, is that if we run both procedure for the same number of iterations (N=1000) we get the following optimum estimation for The FDSA and SPSA with momentum:
-FDSA: $\hat{\theta_N}$=$[0.08 * 10^{-13}, 0.14 * 10^{-13}]^T$
-SPSA with momentum: $\hat{\theta_N}$=$[0.11 * 10^{-17}, -0.05 * 10^{-17}]^T$

We can already see that it seems that the SPSA with momentum has way better approximation to the estimate than the FDSA, and we actually can get a similar approximation of the FDSA , while only using around 150 iterations for the SPSA with momentum.
Next, we will present a table that shows the difference of the average values of the normalized loss function, where we average over 50 experiments of both algorithms, both ruining for the first example 50 iterations, and second 1000iterations, starting from the same initial point,and having similar coefficients. (in the following table we abbreviate SPSA with momentum by SPSAM)

Table 3.1. Sample means of normalized loss $L_{\text{norm}} = L_{\text{norm}}\left(\hat{\boldsymbol{\theta}}_k\right)$ at terminal $\hat{\boldsymbol{\theta}}_k$ for FDSA and SPSA over 50 independent replications. Number of loss measurements $y(\theta)$ is such that FDSA and SPSAM take the same number of iterations in each comparison.

Number of $y(\theta)$ values [number of iterations]	Mean L_{norm} for FDSA and Elapsed time	Mean L_{norm} for SPSAM and Elapsed time
200-FDSA; 100-SPSAM [50 iterations]	3.73×10^{-4}	7.95×10^{-9}
4000-FDSA; 2000-SPSAM [1000 iterations]	4.07×10^{-18}	1.4×10^{-33}

Remark: Additionally to the previous table, we would like to add information on the elapsed times(of the running algorithm) of the four examples given above:
-Elapsed time for the run with 50 iterations : Thea Mean normalized loss for the FDSA took 0.3s, while the SPSAM took only 0.16
-Elapsed time for the run with 1000 iterations : Thea Mean normalized loss for the FDSA took 17.8s, while the SPSAM took only 10s

4.3 Conclusion:

The analysis given above, in the case of a quadratic function, indicates that not only does the SPSA have way better approximation than the FDSA when using the same number of iterations, but it also take way less time for the SPSAM to give results.

5

Notation

We will follow similar notation as in the reference book[1] :
θ: the parameter we are trying to estimate and in this paper, it is considered as a column vector with p components($\theta \in \mathcal{R}^p$
$L(\theta)$: the objective function depending on θ, which is also called the loss function
$\hat{\theta_k}$: the generic notation for the estimate θ at the k-th estimate
$y(\theta) = L(\theta) + \varepsilon(\theta)$: Noisy measurements of the loss function
$y_k(\hat{\theta_k}) = L(\hat{\theta_k}) + \epsilon_k(\hat{\theta_k})$: Noisy measurements of the loss function at the current estimate $\hat{\theta_k}$

Bibliography

[1]
James C. Spall, Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control. , New Jersey, WILEY-INTERSCIENCE, 2003.

[2]
Spall,J.C. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation - IEEE Transactions on Automatic Control, vol37, pp332-341