2013 10th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)
Mexico City, Mexico. September 30-October 4, 2013

Numerical Study of Least Mean Square Method for Adjusting Curves

José Antonio Medina Hernández
Department of Mathematics and Physics
Universidad Autónoma de Aguascalientes
Aguascalientes, México
Phone +52 449 9107400
Email: jamedihe@correo.uaa.mx

Felipe Gómez Castañeda,
José Antonio Moreno Cadenas
Electrical Engineering Department
CINVESTAV-IPN
México D.F., México
Phone +52 55 57473800 Ext 6262
Email: fgomez@cinvestav.mx, jmoreno@cinvestav.mx

***Abstract*—The adjustment of parameters within a function for modeling a set of observations is a very frequent task in many applied areas of science. There are sophisticated techniques to reach this goal, such as regression, use of gradients, neural networks, neurofuzzy modeling, genetic algorithms, swarm optimization, etc. In this paper numerical simulations are done about the efficiency and capacity of the Least Mean Square (LMS) algorithm to find an optimal set of parameters for adjusting a function to a set of observed data. Although the LMS method has been very used for minimization of errors and extraction of noise in signal processing systems, its capacities for regression and approximation have been not very often explored. Using simple examples, conditions on which the learning parameters can be adjusted to model a set of training data are explored, using a iterative learning process where the approximation of the stochastic error is recalculated immediately after any parameter is actualized. A description of the speed for convergence, as a function of the learning rate, is shown for the cases under study.**

Keywords: Linear Regression, Mean Quadratic Error, Stochastic Gradient, Least Mean Square (LMS) Algorithm

I. INTRODUCTION

Fitting a curve to a set of data is a very frequent task at almost all areas of science. It is very typical to have a graph of points corresponding to observations of an experiment, and we would like to obtain a mathematical description of it using an analytical expression that shows how a response variable y depends on a vector of independent variables $\mathbf{x} \in \Re^m$. The basic procedure consists in the proposition of a functional form $y = F(\mathbf{x}, \mathbf{c})$ for modeling the dependient variable y, where $\mathbf{c} \in \Re^p$ is a vector of parameters. The election of the function F is done using graphic information, cuantitative criteria, empiric experience, etc. The task of the data analyst is to find the vector of parameters $\mathbf{c}$ for minimizing the error function

$$E = \sum_{i=1}^{n} (y_i - y_i^d)^2 \tag{1}$$

where $y_i = F(\mathbf{x}_i, \mathbf{c})$ is the estimated output for the $i-th$ input $\mathbf{x}_i$, meanwhile y_i^d is the desired output, also named target output. Among the techniques more used for doing this task are regression, neural networks, genetic algorithms, etc. [1]-[4]. In this paper the behavior of the stochastic gradient, also named Least Mean Square (LMS) [5]-[6] algorithm, is explored. In section II we review the stochastic gradient. In section III the learning rules for two studied examples are indicated. In section IV we show the numerical results, and the speed of convergence of the proposed iterative schema is described as a function of the learning rate. In section V we present discussion and conclusions.

II. THE STOCHASTIC GRADIENT

We suppose there is a set of observated pairs $(\mathbf{x}_i, y_i^d)$ at which the response y_i^d is determined by the vector of independent variables $\mathbf{x}_i$. Using some adequate criteria we choose a function F such that the calculated values $y_i = F(\mathbf{x}_i, \mathbf{c})$ can approximate the corresponding observed values y_i^d, as long as an adequate vector of parameters $\mathbf{c}$ is chosen. Our objective is the minimization of the error function in (1). It requires to find a vector $\mathbf{c}$, being critical point of E. The following necessary conditions must be satisfied:

$$\frac{\partial E}{\partial c_i} = 0 \quad \textit{for i=1,2,...,p} \tag{2}$$

Finding the solution vector $\mathbf{c}$ of this system may be very difficult, specially if E depends on the input and output data. A way for finding a minimum relative of $E(\mathbf{c})$ is using the gradient ∇E. A constant value α, usually small and frequently named learning rate, is used in the successive approximations

$$\mathbf{c}_{k+1} = \mathbf{c}_k - \alpha \nabla E(\mathbf{c}_k) \tag{3}$$

to find a minimum value of E, as a function of $\mathbf{c}$. A problem associated to this method is that if the function E has a complicated form, or if it depends on many input-ouput data, then the calculus of the gradient $\nabla E(\mathbf{c}_k)$ may be very difficult, so the method results impractical.

At the decade of 1960's, B. Widrow and M. E. Hoff indicated [5]-[6] that if F is a linear function on its parameters, the error could be approximated by

$$E_i \approx \frac{1}{2} \left(y_i - y_i^d\right)^2 \tag{4}$$

so that

$$\nabla E_i = \left(y_i - y_i^d\right) \frac{\partial y_i}{\partial \mathbf{c}} \tag{5}$$

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6

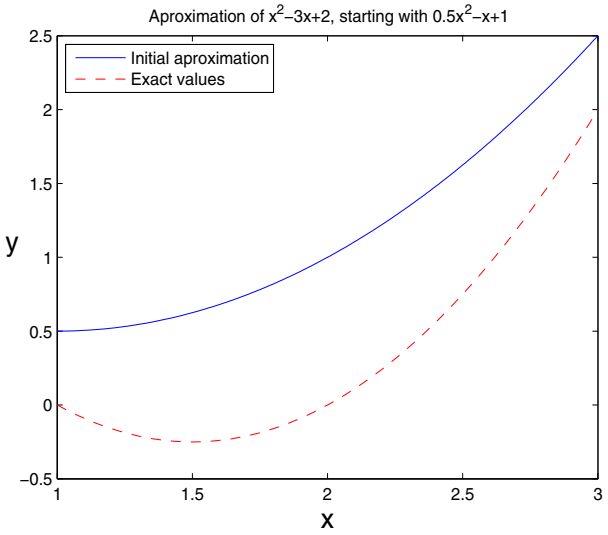

Fig. 1

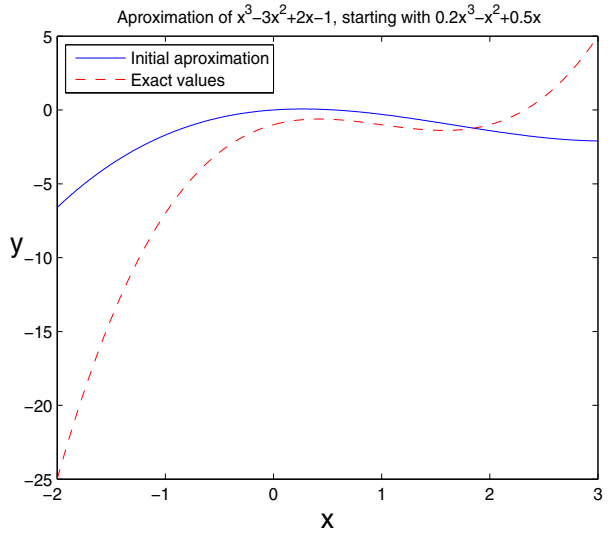

Fig. 2

where

$$\frac{\partial y_i}{\partial \mathbf{c}} = \left(\frac{\partial y_i}{\partial c_1}, ..., \frac{\partial y_i}{\partial c_p}\right). \tag{6}$$

This procedure is named stochastic gradient method or Least Mean Square (LMS) algorithm. Using Neural Networks terminology, the LMS algorithm is equivalent to training a neuron for producing an output similar to y_i^d when the input $\mathbf{x}_i$ is introduced to the neuron, for $i = 1, 2, ..., p$.

In the next section the LMS learning rules associated to two polynomial functions are detailed, which are linear respect to its parameters.

III. LEARNING RULES. EXAMPLES

In this section we will consider two polynomial functions and the capacity of convergence of the LMS algorithm for approximating these functions, starting with coarse approximations.

Example 1. We consider the function

$$f(x) = ax^2 + bx + c \tag{7}$$

and the values $a = 1$, $b = -3$, $c = 2$. We consider the values of this function in the interval [1,3], and we will use the sample points

$$x_i = x_0 + ih \tag{8}$$

taking

$$h = \frac{x_n - x_0}{n} \tag{9}$$

where $x_0 = 1$, $x_n = 3$ and $h = 0.01$. Hence, the set of target outputs is

$$y_i^d = x_i^2 - 3x_i + 2. \tag{10}$$

Next, we suposse that we don't know the values of the coefficients a, b and c associated to the target outputs, but we know only the y_i^d values. We will take as initial approximation to f a quadratic function as in (7), but with the values $a_0 = \frac{1}{2}$, $b_0 = -1$, $c_0 = 1$. Then we will have the set of initially observed outputs

$$y_i = \frac{1}{2}x_i^2 - x_i + 1. \tag{11}$$

The initially observed outputs and the target outputs are shown in Fig 1.

The learning rules for the LMS algorithm are given by

$$a_{new} = a_{old} - \alpha\frac{\partial E_i}{\partial a} \tag{12}$$

$$b_{new} = b_{old} - \alpha\frac{\partial E_i}{\partial b} \tag{13}$$

$$c_{new} = c_{old} - \alpha\frac{\partial E_i}{\partial c}. \tag{14}$$

Using the stochastic error

$$E = \frac{1}{2}(y - y^d)^2 \tag{15}$$

and the well conjectured fitting function

$$y = ax^2 + bx + c \tag{16}$$

we have the partial derivative

$$\frac{\partial E}{\partial a} = (y - y^d)\frac{\partial y}{\partial a} = (y - y^d)x^2. \tag{17}$$

We also have

$$\frac{\partial E}{\partial b} = (y - y^d)\frac{\partial y}{\partial b} = (y - y^d)x \tag{18}$$

$$\frac{\partial E}{\partial c} = (y - y^d)\frac{\partial y}{\partial c} = (y - y^d). \tag{19}$$

Then, at the point x_i we have

$$\frac{\partial E_i}{\partial a} = (y_i - y_i^d)(x_i^2) \tag{20}$$

$$\frac{\partial E_i}{\partial b} = (y_i - y_i^d)(x_i) \tag{21}$$

$$\frac{\partial E_i}{\partial c} = (y_i - y_i^d)(1) \tag{22}$$

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6

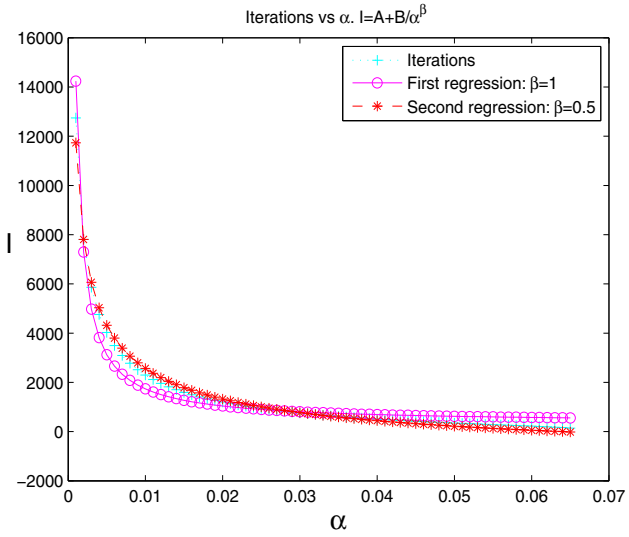

Fig. 3

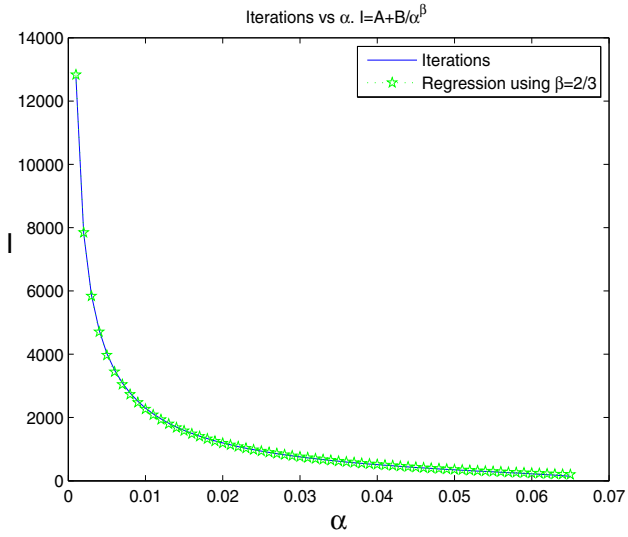

Fig. 4

so, the learning rules (12)-(14) are

$$a_{new} = a_{old} - \alpha(y_i - y_i^d)(x_i^2) \tag{23}$$

$$b_{new} = b_{old} - \alpha(y_i - y_i^d)(x_i) \tag{24}$$

$$c_{new} = c_{old} - \alpha(y_i - y_i^d). \tag{25}$$

In this paper we will propose and study a learning schema at which the above rules are applied in every point x_i, but the values y_i and E_i are recalculated immediately after any parameter is actualized. The pseudocode for this learning outline is:

while error>tolerance
  for i=1 to n
    $D_a y = x_i^2$
    $D_b y = x_i$
    $D_c y = 1$
    $y_i = f(a, b, c, x_i)$
    $Ea = (y_i - y_i^d) * D_a y$
    $a = a - \alpha * Ea$
    $y_i = f(a, b, c, x_i)$
    $Eb = (y_i - y_i^d) * D_b y$
    $b = b - \alpha * Eb$
    $y_i = f(a, b, c, x_i)$
    $Ec = (y_i - y_i^d) * D_c y$
    $c = c - \alpha * Ec$
  endfor
  error=norm($f(a, b, c, \mathbf{x}) - \mathbf{y}^d$)
endwhile.

Using this learning outline and a learning rate $\alpha = 0.01$, 2299 iterations are necessary to reach an error equal to a tolerance value 10^{-5}.

Example 2. Now, we consider the function

$$f(x) = ax^3 + bx^2 + cx + d \tag{26}$$

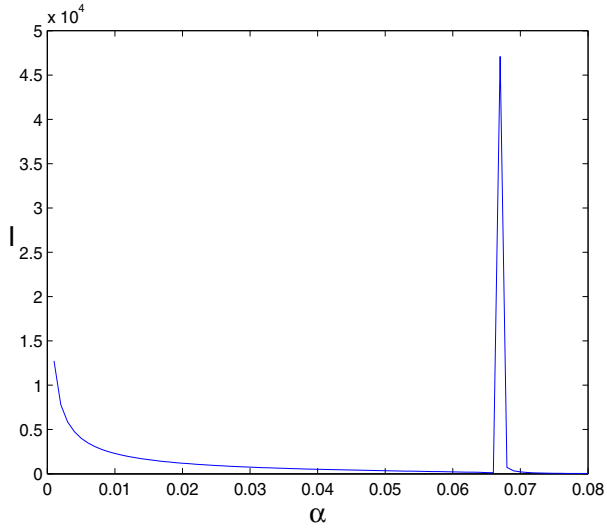

Fig. 5

taking $a = 1$, $b = -3$, $c = 2$, $d = -1$. We study this function in the interval [-2, 3], considering the sample points

$$x_i = x_0 + ih \tag{27}$$

where

$$h = \frac{x_n - x_0}{n} \tag{28}$$

and $x_0 = -2$, $x_n = 3$, $h = 0.01$.

The initially chosen parameters are $a = 0.2$, $b = -1$, $c = 0.5$, $d = 0$. The exact function and its initial approximation are showed in Fig. 2.

Following a similar procedure as the used in first example, it can be verified that also in this case we can obtain the correct values $a = 1$, $b = -3$, $c = 2$, $d = -1$ starting from their initial approximations $a = 0.2$, $b = -1$, $c = 0.5$, $d = 0$ (see next section).

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6

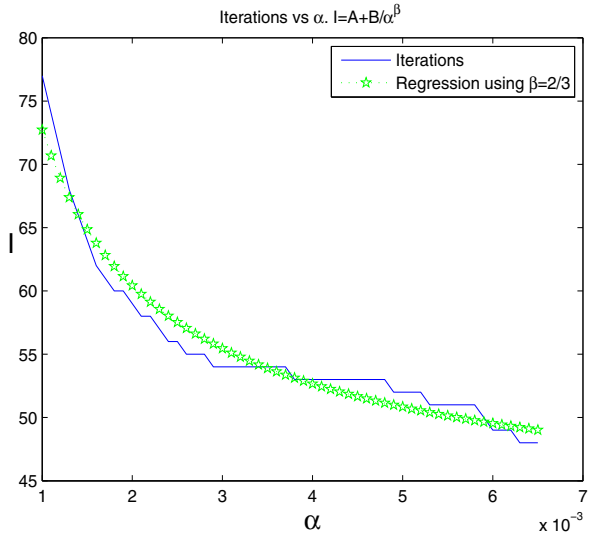

Fig. 6

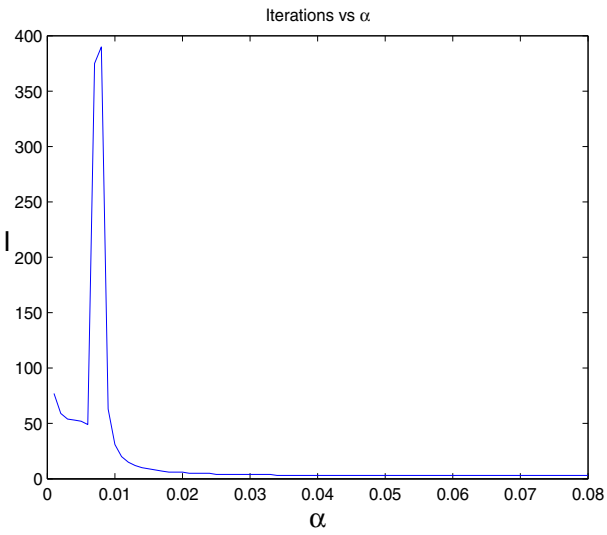

Fig. 8

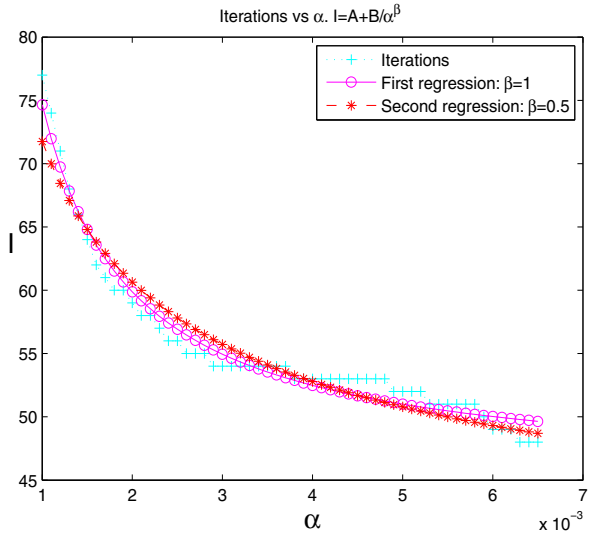

Fig. 7

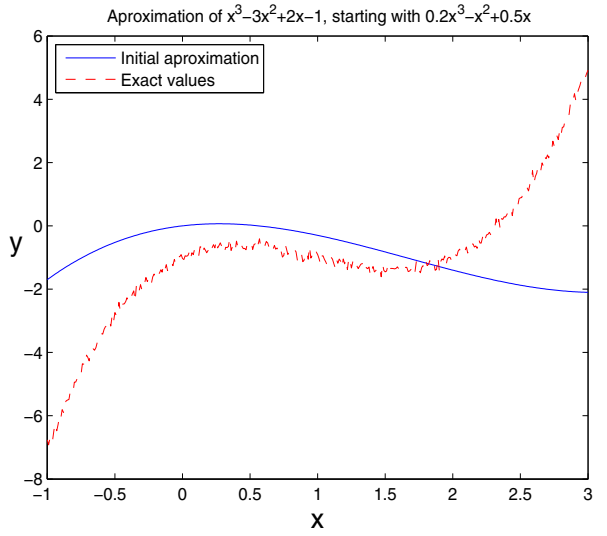

Fig. 9

IV. NUMERICAL RESULTS AND SPEED OF CONVERGENCE

The number of iterations necessary to reach the desired precision varies as a function of the learning parameter α. We will analyze this aspect for the two examples of the previous section.

1) We consider again the quadratic function of example 1. On Fig. 3 the number of iterations required to reach an error lower than 10^{-5} is shown as a function of α. Also, two hiperbolic interpolations of type

$$Iter = A + \frac{B}{\alpha^{\beta}} \tag{29}$$

are shown, using the values $\beta = 1$ and $\beta = 0.5$. If $\beta = 1$, then $A = 343.7333$ and $B = 13.8953$. Taking $\beta = 0.5$ we obtain $A = -1677.3$ and $B = 424.1138$.

It can be observed that the value $\beta = 0.5$ corresponds to a good approximation of the required number of iterations. A very precise interpolation is obtained using the value $\beta = \frac{2}{3}$ with associated values $A = -637.0503$ and $B = 134.6562$, as is shown at Fig. 4.

Fig. 5 shows a singular behavior of the required number of iterations when α takes the value 0.067. A lack of convergence occurs. Iterations corresponding to values $\alpha \geq 0.67$ are not significative, because the corresponding computed coefficients are not numeric values, and the proposed LMS iterative outline is divergent.

The iterations necessary to reach an error 10^{-5} are indicated on table I, using values of α smaller than 0.067.

2) Now, we consider the polynomial function of second example. On Fig. 6 the number of iterations required to reach an error 10^{-5} are shown as function of α. It can be observed that the required number of iterations is lower than the number of iterations used in first example, and the resulting graph is not diferentiable. Also, it is shown a graph of a hiperbolic regression, obtained using $\beta = \frac{2}{3}$. For comparative purposes,

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6

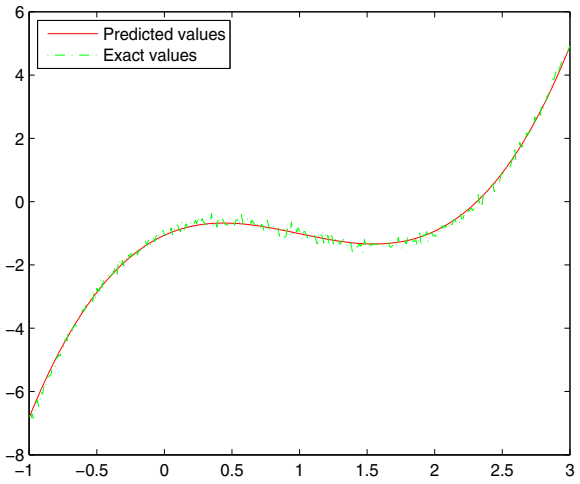

Fig. 10

TABLE I. ITERATIONS VS α AT FIRST EXAMPLE

α	Iterations
0.001	12741
0.002	7809
0.003	5863
0.004	4757
0.005	4024
0.006	3496
0.007	3093
0.008	2775
0.009	2515
0.010	2299
0.011	2115
0.012	1958
0.013	1820
0.014	1699
0.015	1592
0.016	1496
0.017	1410
0.018	1332
0.019	1261
0.020	1196
0.021	1136
0.022	1081
0.023	1030
0.024	982
0.025	938
0.026	897
0.027	858
0.028	822
0.029	788
0.030	756
0.035	618
0.040	510
0.045	422
0.050	348
0.055	283
0.060	221
0.065	144
0.066	125

hiperbolic regressions for $\beta = 1$ and $\beta = 0.5$ are shown at Fig. 7 . In this case, a lack of convergence also ocurrs when $\alpha \geq 0.0067$, as is shown at Fig. 8. On table II, the number of required iterations is shown as function of α.

An additional experiment can be done adding random ϵ_i values, normally distributed with mean 0 and standar desviation 0.01, to the function in (26). The target values y_i are now

TABLE II. ITERATIONS VS α AT SECOND EXAMPLE

α	Iterations
0.0010	77
0.0011	74
0.0012	71
0.0013	68
0.0014	66
0.0015	64
0.0016	62
0.0017	61
0.0018	60
0.0019	60
0.0020	59
0.0021	58
0.0022	58
0.0023	57
0.0024	56
0.0025	56
0.0026	55
0.0028	55
0.0029	54
0.0037	54
0.0038	53
0.0048	53
0.0049	52
0.0052	52
0.0053	51
0.0058	51
0.0059	50
0.0060	49
0.0062	49
0.0063	48
0.0065	48

TABLE III. REGRESSION COEFFICIENTS FOR SECOND EXAMPLE

α	iter	a	b	c	d
0.0010	35	0.9554	-2.8521	1.9482	-1.0638
0.0011	33	0.9568	-2.8594	1.9525	-1.0597
0.0012	31	0.9564	-2.8612	1.9555	-1.0580
0.0013	29	0.9540	-2.8575	1.9574	-1.0585
0.0014	30	0.9563	-2.8735	1.9706	-1.0501
0.0015	37	0.9628	-2.9065	1.9927	-1.0340

given by

$$y_i = x_i^3 - 3x_i^2 + 2x_i - 1 + \epsilon_i \tag{30}$$

where ϵ_i has distribution $N(0, 0.01)$. The target function and its initial approximation

$$y_i = 0.2x_i^3 - x_i^2 + 0.5x_i + \epsilon_i \tag{31}$$

are shown at Fig. 9.

Using a set of learning rates lower than 0.0016, the corresponding regression coefficients obtained using the proposed iterative outline for the LMS algorithm are shown on table III. The used tolerance value was 2.3529. Note that these coefficients are near to the expected values $a = 1$, $b = -3$, $c = 2$ and $d = -1$. On Fig. 10, the graph of the regression obtained using $\alpha = 0.0010$ is shown.

V. DISCUSSION AND CONCLUSIONS

If the function to fit is linear on the parameters, then the learning procedure proposed in section III is an iterative outline for the LMS algorithm. An interval of convergent values for α can be obtained using the eigenvalues of the matrix of variances and covariances [5]. But it requires a previous knowledge of the training pairs. An advantage of the stochastic gradient procedure over the ordinary regression or interpolation is that it is not necessary to have available all the training pairs in order to actualize the parameters

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6

of the model, so the learning is on-line. But this requires an estimation of the adequate learning rates for reaching a good fitting. If the function used for modeling the data set is not linear, it is enough to have an approximation of its partial derivatives respect to the parameters in order to use the learning rules. The numerical examples of section III and IV show that under adequate conditions, the proposed iterative outline for the method of stochastic gradient is able to find the optimum set of parameters for fitting the selected function to the observed data. The learning approach is adequate for both interpolation (Figs. 1, 2) and regression problems (Figs. 9, 10). The analysis of convergence for the proposed iterative learning outline shows that, as can be expected in a general situation, the number of iterations required to solve a problem diminishes as the learning rule α is increased. But there is an onset value such that if α is larger than it then the learning process is not convergent.

VI. ACKNOWLEDGMENT

Authors thank funding of the National Council for Science and Technology (CONACYT) for the development of this work.

REFERENCES

[1] J. Anderson, E. Rosenfeld *Neurocomputing: Foundations of Research*, Cambridge, MA:MIT Press, 1989.

[2] L. E. Scales, *Introduction to Nonlinear Optimization*, New York:Springer Verlag, 1985.

[3] A. Zhigljavsky, *Stochastic Global Optimization*, New York:Springer Verlag, 2008.

[4] D. Fouskakis, D. Draper "Stochastic Optimization: A Review," *International Statistical Review*, 70, 3, 315-349, 2002

[5] B. Widrow, E. Walach, *Adaptive Inverse Control*, Prentice-Hall, 1996.

[6] B. Widrow, R. Winter, "Neural Nets for adaptive filtering and adaptive pattern recognition," *IEEE Computer Magazine*, March 1988, pp. 25-39.

IEEE Catalog Number CFP13827-ART
ISBN 978-1-4799-1461-6