WCCI 2010 IEEE World Congress on Computational Intelligence
July, 18-23, 2010 - CCIB, Barcelona, Spain IJCNN

Fixed point method of step-size estimation for on-line neural network training

Paweł Wawrzyński, *Member*, *IEEE*

Abstract— **This paper considers on-line training of feadforward neural networks. Training examples are only available sampled randomly from a given generator. What emerges in this setting is the problem of step-sizes, or learning rates, adaptation. A scheme of determining step-sizes is introduced here that satisfies the following requirements: (i) it does not need any auxiliary problem-dependent parameters, (ii) it does not assume any particular loss function that the training process is intended to minimize, (iii) it makes the learning process stable and efficient. An experimental study with the 2D Gabor function approximation is presented.**
Keywords: neural networks, on-line learning, step-size adaptation, reinforcement learning.

I. Introduction

A neural network with a given structure defines a class of functions parameterized by the vector of the network's synaptic weights. Network training identifies the best, in a given sense, function within this class. This property makes feedforward neural network a powerful tool to function modeling. Its primary application is nonlinear regression — a way to identify statistical relations among variables. It can also be applied by reinforcement learning techniques to identify and represent optimal input-output relations in controllers.

There are two basic settings in which neural networks are used, called usually the batch mode and the pattern mode. If all the training data can be collected in a database, they can be applied to define a quality index that maps the space of network weights to real numbers. Minimization of that quality index defines the batch mode of the network training. It can be performed by various techniques i.e., Levenberg–Marquardt algorithm [1] or the method based on linear regression presented in [2].

In some applications the training examples can only be sampled from a certain, possibly continuous distribution. The pattern mode of training may be applied then: Each sample is utilized to calculate a gradient estimator of the quality index. The estimator is multiplied by a step-size and applied to adjust the network weights. The problem of step-sizes determination is inherent in the pattern mode of training. It is solved by various methods, like *delta-delta* [3], *delta-bar-delta* [4], Extended Kalman Filter [5], stochastic meta-descent [6], or the method based on Lapunov stability [7]. These methods require providing several initial parameters and are based on certain assumptions regarding the momentary loss function being optimized.

Here, we are interested in the pattern mode of network training. In particular, we seek for fully autonomous schema of network training applicable by the algorithms of reinforcement learning with experience replay [8]. The schema must be applicable in the following setting:

1) The network is trained with the use of training samples randomly drawn from an infinite set.
2) The network is relatively large: with at least tens of inputs and outputs, and hundreds hidden neurons.
3) The objective of the network training is minimization of a certain complex quality index (having little to do with mean-quadratic error).

The first point excludes efficient batch techniques of network training, like Levenberg–Marquardt algorithm. The second point effectively excludes methods whose complexity is over linear in number of neural weights, like Extended Kalman Filter. The third one effectively excludes the method based on Lapunov stability [7]. An efficient method of step-size determination for reinforcement learning algorithms was investigated in a number of studies, including [9], [10], [11]. However, all the presented method lack autonomy as they require parameters, like a meta-step-size, that are problem dependent.

In this paper we introduce a method of pattern model neural network training that is suitable to the setting described above. It is also fully autonomous: it does not require any problem-dependent parameters nor it is based on any assumptions regarding the nature of the momentary loss function optimized by the network training.

The method presented here determines the step-size of the learning network by a comparison of two vectors in a network weights space: the sum of gradient estimators that actually govern the weights adjustment and the sum of gradient estimators measured in a fixed point in the weights space. We show here, that in order to make the training efficient, those vectors should be in a special relation.

The paper is organized as follows. In Sec. II the problem considered here is formulated. In the following section the proposed solution is outlined. Section IV analyzes a simplified model that serves to derive the solution to the discussed problem. The following section treats the issue of robustness. Section VI presents the algorithm and another one presents an experimental study: application of the presented algorithm to learn a noised 2D-Gabor function on the basis of a noised data. The last section concludes.

This work was supported in part by Grant N N514 237137 from the Ministry of Science and Higher Education in Poland.
Paweł Wawrzyński is with the Institute of Control and Computation Engineering, Warsaw University of Technology, Poland; email: p.wawrzynski@elka.pw.edu.pl.

II. Problem formulation

We consider the general issue of stochastic optimization defined as follows. There is a function, $J : \Re^m \mapsto \Re$. Its gradient, ∇J, is not available but a generator is available that samples random vectors, ξ, that are used to compute unbiased estimators of ∇J. Namely, a function g is given such that $Eg(\theta, \xi) = \nabla J(\theta)$. The term E denotes here the expected value of a function of the random vector ξ.

The above model describes the issue of feedforward network learning as follows. The weights of the network are denoted by θ; $\xi = [x^T, y^T]^T$, where x denotes the network input and y is the network desired output, J is the global quality index optimized through network learning, and $g(\theta, \xi)$ is the vector computed by means of error backpropagation.

A generic technique to approximate the minimum of J function is Robbins–Monroe procedure [12], called also the method of stochastic descent. It defines the sequence

$$\theta_{t+1} = \theta_t - \beta_t g(\theta_t, \xi_t), \quad t = 0, 1, 2, \dots, \tag{1}$$

where $\{\beta_t, t = 0, 1, 2, \dots\}$ is a vanishing sequence of positive parameters called step-sizes. It is known [12], that once certain regularity conditions are satisfied, the sequence $\{\theta_t\}$ converges to a local minimum of J.

Unfortunately there is no problem independent sequence of step-sizes that makes the convergence of Robbins–Monroe procedure fast and stable. In fact, the convergence may be arbitrarily slow for step-sizes that are too small for a given problem and may be unstable for arbitrarily many initial steps for relatively too large step-sizes.

We search for a way to determine the step-sizes on the basis of consecutive samples that excludes instability of the learning process and assures fast convergence. Also, we require that the complexity of the method is linear in θ_t dimension.

III. Idea

Suppose, we apply Robbins–Monroe procedure to find the minimum of a certain function. At a given instant, t, we begin the following experiment: (i) The step-sizes are fixed $\beta_{t+n} \equiv \beta$ for $n = 0, 1, 2, \dots$. (ii) Consecutive points θ_{t+n} are computed according to (1). (iii) Two sums are computed, namely

$$G_{t,n} = \sum_{i=0}^{n-1} g(\theta_{t+i}, \xi_{t+i}), \quad G^*_{t,n} = \sum_{i=0}^{n-1} g(\theta_t, \xi_{t+i}). \tag{2}$$

Notice that vector $G_{t,n}$ is proportional to the displacement of the point θ between instant t and $t+n$. On the other hand, vector $G^*_{t,n}$ is proportional to the displacement of the point θ that *would* have been if all the gradients estimators, g, had been calculated at θ_t.

To continue the experiment, let us fix n large enough to make the statistics $\frac{1}{n}G^*_{t,n}$ a good estimate of the gradient $\nabla J(\theta_t)$. A comparison of $G_{t,n}$ and $G^*_{t,n}$ may lead to one of the following conclusions:

1) If $G_{t,n}$ is only slightly different from $G^*_{t,n}$, it suggests that θ_{t+i} moves through flat slope of J; curvature of J plays no role in this movement. If the step-size was a little larger, the speed of walking down the slope would be larger too. The step-size may thus be increased. This case is illustrated on the left part of Fig. 1.
2) The value $\|G_{t,n} - G^*_{t,n}\|$ is relatively large in comparison to $\|G^*_{t,n}\|$, which means that the impact of the curvature of J on the movement of $\theta_{t+i}, 0 \le i < n$ is comparable to the impact of the noise in gradient estimates. β_t may be close to its optimal value. This case is illustrated in the middle of Fig. 1.
3) If $G_{t,n}$ is radically different from $G^*_{t,n}$, it may be caused by the instability of the learning process. In fact, the point θ_{t+n} is then likely to be further from the minimum of J than θ_t. This situation is depicted on the right part of Fig. 1.

The above discussion leads to the following schema of step-size determination: (i) The process (1) is divided into parts. (ii) Within each part the step-size is constant and the sums $G_{t,n}$, and $G^*_{t,n}$ are computed with Eqs. (2), where the part begins at time t and ends at time $t+n$. (iii) At the end of each part $\|G^*_{t,n} - G_{t,n}\|$ is compared with $\|G^*_{t,n}\|$; if $\|G^*_{t,n} - G_{t,n}\|$ is „relatively too large", the step-size is decreased, if the difference is „relatively too small", the step-size is increased.

IV. Model

Let us now assign a concrete meaning to the soft concepts of the previous section. In this order, we analyze a simple model. Namely, let $J : \Re \mapsto \Re$ be a scalar quadratic function

$$J(\theta) = \frac{1}{2}a\theta^2, \quad a > 0.$$

The gradient estimator of J is of the form

$$g(\theta_t, \xi_t) = \nabla J(\theta_t) + \xi_t = a\theta_t + \xi_t,$$

where ξ_t is a zero–mean random value with variance σ^2. We minimize J by means of Robbins–Monroe procedure (1) i.e.,

$$\theta_{t+1} = \theta_t - \beta_t(a\theta_t + \xi_t). \tag{3}$$

The question is: what is the right value of β_t? Let us first find the step-size that is optimal in the sense that it minimizes the expected one–step improvement of J-value. Namely, it minimizes

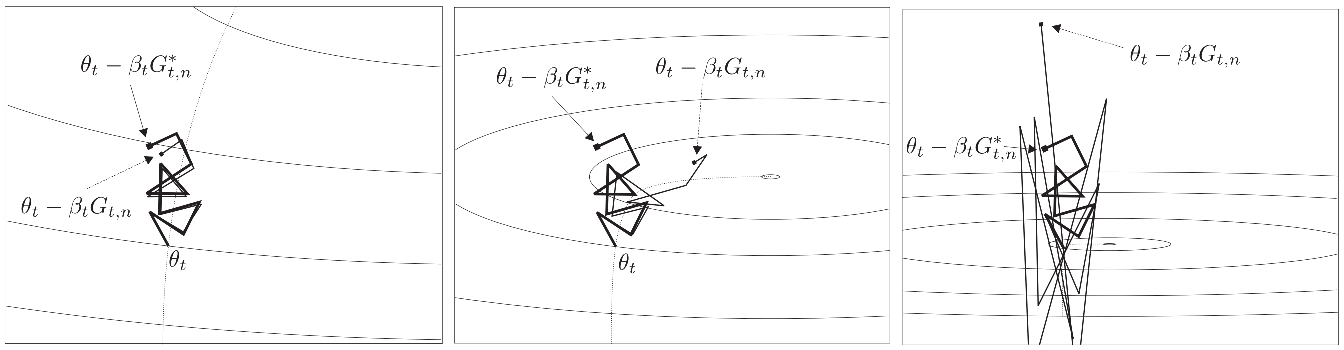
$$\begin{aligned} e_t(\beta) &= E\left(J(\theta_{t+1}) - J(\theta_t)|\theta_t\right) \qquad (4) \\ &= E_{\theta_t}\left(\frac{1}{2}a(\theta_t - \beta(a\theta_t + \xi_t))^2 - \frac{1}{2}a\theta_t^2\right) \\ &= \frac{1}{2}E_{\theta_t}\left(-2\beta(a\theta_t)(a\theta_t + \xi_t) + a\beta^2(a\theta_t + \xi_t)^2\right) \\ &= -\beta(a\theta_t)^2 + \beta^2\frac{1}{2}aE_{\theta_t}\left(a\theta_t + \xi\right)^2. \end{aligned}$$

The function e_t is quadratic with derivative

$$\frac{\partial}{\partial\beta}e_t(\beta) = -(a\theta_t)^2 + \beta a E_{\theta_t}(a\theta_t + \xi)^2$$

and has the minimum at the point

$$\beta^*_t = \frac{(a\theta_t)^2}{aE_{\theta_t}(a\theta_t + \xi)^2}. \tag{5}$$



$\theta_t - \beta_t G^*_{t,n}$
$\theta_t - \beta_t G_{t,n}$
θ_t
$\theta_t - \beta_t G^*_{t,n}$
$\theta_t - \beta_t G_{t,n}$
θ_t
$\theta_t - \beta_t G_{t,n}$
$\theta_t - \beta_t G^*_{t,n}$

Fig. 1. The sketches depict contour lines of three different functions J. However, $\nabla J(\theta_t)$, β, and $g(\theta_t, \xi_{t+i})$ for $i \geq 0$, are equal in all three cases. In n steps since t, Robbins–Monroe procedure drives the point in the J domain from θ_t to $\theta_{t+n} = \theta_t - \beta_t G_{t,n}$. *Left:* Small curvature of J in relation to β, causes small difference of $-\beta_t G_{t,n}$ and $-\beta_t G^*_{t,n}$. *Middle:* Moderate curvature of J causes moderate difference between $-\beta_t G_{t,n}$ and $-\beta_t G^*_{t,n}$. *Right:* Very large curvature of J relatively to β causes instability of the optimization process which is manifested by divergence of $-\beta_t G_{t,n}$ and $-\beta_t G^*_{t,n}$ that grows with n.

The derivative may be then rewritten as

$$\frac{\partial}{\partial \beta} e_t(\beta) = (a\theta_t)^2(-1 + \beta/\beta^*_t). \tag{6}$$

The objective of this section is to compare

$$E_{\theta_t}|G^*_{t,n}|^2 \quad \text{with} \quad E_{\theta_t}|G^*_{t,n} - G_{t,n}|^2$$

for n that makes the statistics $\frac{1}{n}G^*_{t,n}$ a reasonable estimate of the gradient $\nabla J(\theta_t)$. Let us choose

$$n = \left\lceil \frac{E_{\theta_t}\|g(\theta_t,\xi)\|^2}{\|\nabla J(\theta_t)\|^2} \right\rceil = \left\lceil \frac{E_{\theta_t}(a\theta_t + \xi)^2}{(a\theta_t)^2} \right\rceil. \tag{7}$$

Notice that variance of $G^*_{t,n}$ is equal to

$$\mathcal{V}_{\theta_t}\frac{1}{n}G^*_{t,n} = \frac{1}{n^2}n\mathcal{V}_{\theta_t}(a\theta_t + \xi) \leq \frac{1}{n}E_{\theta_t}(a\theta_t + \xi)^2 \leq (a\theta_t)^2 = |\nabla J(\theta_t)|^2.$$

Consequently, the standard deviation of $\frac{1}{n}G^*_{t,n}$ is no greater than the norm of estimated gradient. Notice the relation between n, (7), and β^*_t, (5), namely

$$a \cong (n\beta^*_t)^{-1}. \tag{8}$$

Let us fix $\beta_t = \beta_{t+1} = \cdots = \beta_{t+n-1} \equiv \beta$.

We are now ready to compare $E_{\theta_t}|G^*_{t,n}|^2$ with $E_{\theta_t}|G^*_{t,n} - G_{t,n}|^2$. First,

$$\begin{aligned} E_{\theta_t}|G^*_{t,n}|^2 &= E_{\theta_t}\left(\sum_{i=0}^{n-1}(a\theta_t + \xi_{t+i})\right)^2 \\ &= n^2a^2\theta_t^2 + n\sigma^2 \\ &\cong (\beta^*_t)^{-2}\theta_t^2 + n\sigma^2. \end{aligned} \tag{9}$$

The last equality is not a precise one only because of the rounding in (7). Then, because from (3) we have

$$\theta_{t+1} = \theta_t(1 - \beta a) - \beta\xi_t,$$

a simple induction implies that

$$\theta_{t+n} = \theta_t(1 - \beta a)^n - \beta\sum_{i=0}^{n-1}(1 - \beta a)^i\xi_{t+n-1-i}$$

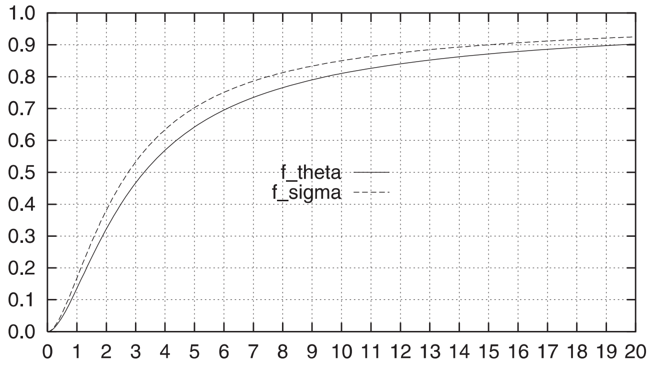


Fig. 2. Functions f_θ and f_σ.

Therefore,

$$\begin{aligned} E_{\theta_t}\big|G^*_{t,n} - G_{t,n}\big|^2 &= E_{\theta_t}|G^*_{t,n} - \beta^{-1}(\theta_t - \theta_{t+n})|^2 \\ &= E_{\theta_t}\Bigg(na\theta_t - \beta^{-1}\theta_t\big(1 - (1-\beta a)^n\big) \\ &\qquad + \sum_{i=0}^{n-1}\big(1 - (1-\beta a)^i\big)\xi_{t+n-1-i}\Bigg)^2 \\ &= \theta_t^2\left(\beta^{*-1}_t - \beta^{-1}\big(1 - (1-\beta a)^n\big)\right)^2 \\ &\qquad + \sigma^2\sum_{i=0}^{n-1}\big(1 - (1-\beta a)^i\big)^2 \\ &= \theta_t^2\left(\beta^{*-1}_t - \beta^{-1}\big(1 - (1-\beta a)^n\big)\right)^2 \\ &\qquad + \sigma^2\left(n - 2\frac{1 - (1-\beta a)^n}{\beta a} + \frac{1 - (1-\beta a)^{2n}}{1 - (1-\beta a)^2}\right). \end{aligned}$$

In the further analysis we notice that (8) implies

$$\beta a = \frac{\beta}{\beta^*_t n}$$

and it is a simple analytical fact that

$$\lim_{n\to\infty}\left(1 - \frac{\beta}{\beta^*_t n}\right)^n = \exp(-\beta/\beta^*_t).$$

Consequently, for $n \gg 1$ and $\beta \ll n\beta_t^*$ we have

$$\begin{aligned}
&E_{\theta_t}\left|G_{t,n}^* - G_{t,n}\right|^2 \\
&\quad\cong \theta_t^2\left(\beta_t^{*-1} - \beta^{-1}\left(1-\exp(-\beta/\beta_t^*)\right)\right)^2 \\
&\qquad + \sigma^2 n\Bigg(1 - \frac{2\beta_t^*}{\beta}(1-\exp(-\beta/\beta_t^*)) \\
&\qquad\qquad + \frac{\beta_t^*}{2\beta}(1-\exp(-2\beta/\beta_t^*))\left(1-\frac{\beta}{2n\beta_t^*}\right)^{-1}\Bigg) \\
&\quad\cong \frac{\theta_t^2}{\beta_t^{*2}}\Bigg(1 - \frac{2\beta_t^*}{\beta}\left(1-\exp(-\beta/\beta_t^*)\right) \\
&\qquad\qquad + \frac{\beta_t^{*2}}{\beta^2}\left(1-\exp(-\beta/\beta_t^*)\right)^2\Bigg) \\
&\qquad + \sigma^2 n\Bigg(1 - \frac{2\beta_t^*}{\beta}(1-\exp(-\beta/\beta_t^*)) \\
&\qquad\qquad + \frac{\beta_t^*}{2\beta}(1-\exp(-2\beta/\beta_t^*))\Bigg)
\end{aligned}$$

The above approximate equality may be rewritten as

$$E_{\theta_t}\left|G_{t,n}^* - G_{t,n}\right|^2 \cong \frac{\theta_t^2}{\beta_t^{*2}} f_\theta(\beta/\beta_t^*) + \sigma^2 n f_\sigma(\beta/\beta_t^*). \quad (10)$$

for accordingly defined functions f_θ and f_σ. Graphs of these functions are presented in Fig. 2. They have the following properties:

- $f_\theta(0) = f_\sigma(0) = 0$,
- both functions are increasing,
- $f_\theta(1) \cong 0.13533$,
- $f_\sigma(1) \cong 0.16809$.

Due to similarity of f_θ and f_σ, let us approximate them by their average denoted by f, namely

$$f(\beta/\beta_t^*) = 0.5 f_\theta(\beta/\beta_t^*) + 0.5 f_\sigma(\beta/\beta_t^*).$$

Comparison of $E_{\theta_t}|G_{t,n}^*|^2$, (9), and $E_{\theta_t}|G_{t,n}^* - G_{t,n}|^2$, (10), leads now to the equation

$$\begin{aligned}
E_{\theta_t}\left|G_{t,n}^* - G_{t,n}\right|^2 &\cong (\theta_t^2/\beta_t^{*2} + \sigma^2 n) f(\beta/\beta_t^*) \\
&= E_{\theta_t}\left|G_{t,n}^*\right|^2 f(\beta/\beta_t^*) \quad (11)
\end{aligned}$$

which has the following implications

- For the step size β close to its optimal value β_t^*, we have $\beta/\beta_t^* \cong 1$, and $E_{\theta_t}|G_{t,n}^* - G_{t,n}|^2 \cong 0.15 E_{\theta_t}|G_{t,n}^*|^2$,
- If $E_{\theta_t}|G_{t,n}^* - G_{t,n}|^2 \le 0.13 E_{\theta_t}|G_{t,n}^*|^2$, the step-size is smaller than optimal,
- If $E_{\theta_t}|G_{t,n}^* - G_{t,n}|^2 \ge 0.17 E_{\theta_t}|G_{t,n}^*|^2$, the step-size is greater than optimal.

1) Implications for optimization in one dimension: The above discussion suggests the following pattern of step-size adjustment: (i) θ_t is adjusted according to Eq. (1), (ii) the process is divided into periods of length n approximately satisfying Eq. (7), (iii) the step-size is constant within each period, (iv) If at the end of a period the difference $0.15|G_{t,n}^*|^2 - |G_{t,n}^* - G_{t,n}|^2$ happens to be larger than 0, the step-size should be increased; otherwise, it should be decreased.

To be more specific, we utilize each estimation period to estimate the derivative of the e_t function (6). To this end, let us analyze the statistics

$$d_{t,n} = |G_{t,n}^* - G_{t,n}|^2 - 0.15|G_{t,n}^*|^2. \quad (12)$$

From (11) we know that

$$\begin{aligned}
E_{\theta_t} d_{t,n} &\cong E_{\theta_t}|G_{t,n}^*|^2 (f(\beta/\beta_t^*) - f(1)) \\
&\cong (\theta_t^2/\beta_t^{*2} + \sigma^2 n) f(1)(-1 + f(1)^{-1} f(\beta/\beta_t^*)).
\end{aligned}$$

Applying the definitions of β_t^*, (5), and n, (7), we easily obtain

$$\begin{aligned}
\frac{\theta_t^2}{\beta_t^{*2}} &= n^2(a\theta_t)^2 \\
n\sigma^2 &= n(n-1)(a\theta_t)^2
\end{aligned}$$

which leads to

$$E_{\theta_t} d_{t,n} = 2n(n-1) f(1)\left(a\theta_t\right)^2\left(-1 + f(1)^{-1} f(\beta/\beta_t^*)\right). \quad (13)$$

Let us compare the above equation with the derivative $\partial e_t/\partial\beta$, (6). Because for $0 \le \beta < 4\beta_t^*$,

$$f(1)^{-1} f(\beta/\beta^*) \cong \beta/\beta_t^*,$$

the term

$$d_{t,n}\frac{1}{2n(n-1)f(1)} \quad (14)$$

is an approximately unbiased estimator of $(\partial/\partial\beta)e_t(\beta)$ for small β. For larger β, the estimators is biased, but it still indicates a proper sign of the derivative.

2) Optimization in many dimensions: All interesting neural networks have more than one weight. The above results must be thus extended to a multidimensional parameter θ representing neural weights.

In order to determine a single step-size approximately optimal for each dimension of θ, we define the function that the step-size is to minimize by

$$e_t^+(\beta) = \sum_{j=1}^m e_{t,j}(\beta) \quad (15)$$

where $e_{t,j}(\beta)$ is the quality index of β for j-th dimension of θ_t, defined in (4).

The idea of step-size adjustment is the following: (i) θ_t is adjusted according to Eq. (1), (ii) the process is divided into periods of length n satisfying Eq. (7), (iii) the step-size is constant within each period, (iv) within each period, the derivative $\partial e^+/\partial\beta$ is estimated, and (v) the step-size is modified to decrease e^+ i.e., incremented in the direction opposite to the gradient estimator of e^+.

The values θ_t, $g(\theta,\xi)$, $G_{t,n}^*$, and $G_{t,n}$ become now vectors in $\Re^m$. In order to estimate the derivative of the sum (15), we apply a sum of the estimators (14) of derivatives of the elements of (15). The estimator has the form

$$\begin{aligned}
D_{t,n} &= \frac{\sum_{j=1}^m |G_{t,n,j}^* - G_{t,n,j}|^2 - 0.15|G_{t,n}^*|^2}{2n(n-1)f(1)} \\
&= \frac{\|G_{t,n}^* - G_{t,n}\|^2 - 0.15\|G_{t,n}^*\|^2}{2n(n-1)f(1)}.
\end{aligned}$$

After each estimation period, the step-size is modified in the direction opposite to the above estimator.

V. Robustness

Most methods of step-size determination induce incremental modifications of this parameter. However, a step-size that is optimal for a certain weight vector θ_t, after some number of learning steps may become too large and introduce instability into the learning process. The incremental modifications need some time to adjust the step-size to new circumstances. This time may be too long to prevent divergence. The lack of robustness to instability is observed indeed in a number of studies, and that is probably the most significant reason why step-size determination schemes are still a subject of active research.

We propose here a simple way to assure stability of the learning process: If the process becomes instable, it is returned to the last point in which it was stable, with a decreased step-size. The question remains, how to detect the instability.

The process is divided into periods. Within each period, two sums of gradients are computed: $G^*_{t,n}$, (2), with the use of fixed weights, θ_t, and $G_{t,n}$ with the use of changing weights. If those sums are close to each other, it means that the average gradients measured in neighboring instants indicate the same coherent improvement direction; the process is stable. Instability means that there is no coherent improvement direction. It is manifested by discrepancy of $G^*_{t,n}$ and $G_{t,n}$. The algorithm presented below detects instability when the following condition is satisfied:

$$\|G_{t,n} - G^*_{t,n}\|^2 > \max_{1 \le i \le n} \|G^*_{t,i}\|^2.$$

VI. Algorithm

The algorithm resulting from the discussion above is presented in Tab. I. It divides the learning process into *estimation periods*, in which the step-size β remains fixed. The meaning of variables t, n, and ξ_{t+n} is the same as in the previous section. The variables θ^*, θ, G, G^* denote θ_t, θ_{t+n}, $G_{t,n}$, and $G^*_{t,n}$, respectively. In order to compute n according to (7) the term (L/M) estimates $E_{\theta_t}\|g(\theta_t, \xi)\|^2$ and the term (L'/M') estimates $\|\nabla J(\theta_t)\|^2$ on the basis of the following simple property. Let ξ and ξ' be independent. We thus have

$$E_{\theta_t} g(\theta_t, \xi)^T g(\theta_t, \xi') = E_{\theta_t} g(\theta_t, \xi)^T E_{\theta_t} g(\theta_t, \xi') = \|\nabla J(\theta_t)\|^2.$$

The variable h denotes $\max_{1 \le i \le n} \|G^*_{t,i}\|^2$.

In its Line 14. the algorithm check whether the learning process is stable. Its instability is manifested by large value of $\|G_{t,n} - G^*_{t,n}\|^2$ in comparison to $\max_{1 \le i \le n} \|G^*_{t,i}\|^2$. If instability is detected, the θ_t parameter comes back to its value recorded when there was no instability detected.

In Line 16. the algorithm checks if the present estimation period has come to its end. In this case a new value of the step-size is assigned (Line 19) and a new exploration period is begun.

TABLE I

Fixed point method of step-size estimation for neural network training. The coefficients suggested: $\alpha_1 = 0.9$, $\alpha_2 = 10^{-2}$, $t_{\min} = 100$, initial $\beta = 1$.

```
1:  Assign $t := 1$, $n := 0$, $L = L' = M = M' := 0$,
2:  Initialize $\theta = \theta^*$, and $\beta$. Assign $G = G^* := 0$, $h = \bar{h} := 0$.
3:  Get new $\xi_{t+n}$.
4:  $L := L + \|g(\theta^*, \xi_{t+n})\|^2$,
5:  $M := M + 1$,
6:  If $n > 0$
7:      $L' := \max\{L' + g(\theta^*, \xi_{t+n-1})^T g(\theta^*, \xi_{t+n}), 0\}$,
8:      $M' := M' + 1$.
9:  $G := G + g(\theta, \xi_{t+n})$,
10: $\theta := \theta - \beta g(\theta, \xi_{t+n})$,
11: $G^* := G^* + g(\theta^*, \xi_{t+n})$,
12: $h := \max\{h, \|G^*\|^2\}$.
13: $n := n + 1$, $newperiod := false$,
14: If $\|G - G^*\|^2 > h$,
15:     $\theta := \theta^*$, $\beta := \beta/2$, $newperiod := true$.
16: If $\neg newperiod \wedge t + n \ge t_{min} \wedge n \ge (L/M)/(L'/M')$,
17:     $\theta^* := \theta$,
18:     $\bar{h} := \max\{\alpha_1 \bar{h}, h/(1 - \alpha_1)\}$,
19:     $\beta := \beta \exp\left(\left(0.15\|G^*\|^2 - \|G - G^*\|^2\right)/\bar{h}\right)$,
20:     $\bar{M} = \max\{\alpha_1 M, \alpha_2 t\}$,
21:     $L := L\frac{\bar{M}}{M}$, $M := \bar{M}$,
22:     $L' := L'\frac{\bar{M}}{M'}$, $M' := \bar{M}$,
23:     $t := t + n$, $newperiod := true$.
24: If $newperiod$
25:     $G = G^* := 0$, $h := 0$, $n := 0$,
26: Goto Line 3.
```

In Line 18. the scaling factor for step-size adjustment is determined and in Line 20. the memory of $E_{\theta_t}\|g(\theta^*, \xi)\|^2$ and $\|\nabla J(\theta_t)\|^2$ estimators is shortened.

In Lines 18 and 20 coefficients α_1 and α_2 are applied to define „memory capacity” of estimators. The larger these values are, the larger capacity. In Line 16 coefficient $t_{\min}$ prevents from adjusting the step-size when the estimators are based on too little data. Also, the algorithm starts with a certain initial value of the step-size. In Tab. I values of all those four coefficients are suggested, but it seems that in practice the algorithm is highly insensitive to these parameters. It also does not appear that their optimal values are application dependent. What is application-dependent indeed is the length of an estimation period, n, that satisfies the last condition in Line 16. However, this value is estimated on the basis of the learning process observation.

VII. Experimental study

We test the proposed step-size adaptation scheme on the problem of 2D Gabor function approximation. The function is given by formula

$$g(x_1, x_2) = \frac{1}{2\pi(0.5)^2} \exp\left(\frac{-x_1^2 - x_2^2}{2(0.5)^2}\right) \cos\left(2\pi(x_1 + x_2)\right).$$

For this problem, training data are generated where input variables are sampled randomly with uniform distribution in the range $[0.5, 0.5]$. The desired output fed to a network is given by

$$y = g(x_1, x_2) + \eta,$$

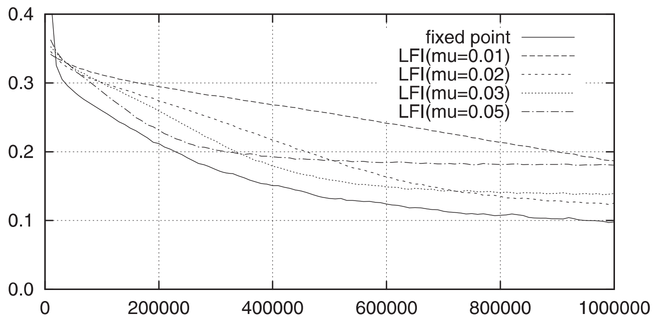


Fig. 3. 2D Gabor function approximation. Mean square error vs. step number the step-size adjusted with the use of the fixed point method and the method based on Lapunov stability [7] for various values of its parameter μ. Each curve averages 50 runs.

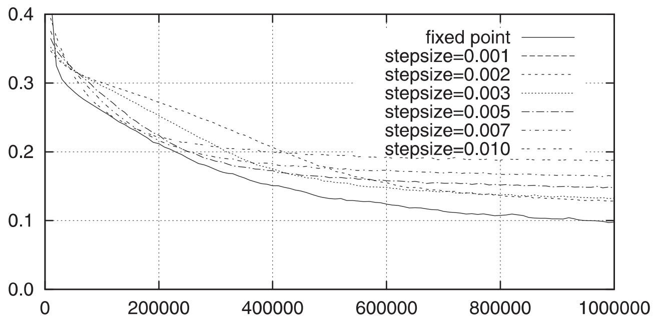


Fig. 4. 2D Gabor function approximation. Mean square error vs. step number the step-size adjusted with the use of the fixed point method and various values of constant step-size. Each curve averages 50 runs.

where η is a normal noise with unit variance. We approximate the same function as in [7], but train the network with noised data to make the problem resemble more the circumstances in which neural networks are applied in reinforcement learning. The network trained is a two-layer perceptron with 20 sigmoidal (arctan) hidden neurons and a linear output neuron. Initial weights of the hidden neurons are drawn randomly from the normal distribution $N(0, 1)$ and initial weights of the output neurons are equal to zero.

We compare time series of mean-square error for learning with the step-size adjusted by the method presented here and the method based on Lapunov stability [7], for various value of its parameter μ. The resulting learning curves are presented in Fig. 3. The fixed point method works better than the alternative for any value of its parameter μ: for μ smaller than 0.01, convergence of this method is slow, while for μ larger than 0.05, the ultimate approximation is weak. We also compare the fixed point method against constant step-sizes. Fig. 4 presents the results: For step-sizes smaller than 0.002 convergence is slow, while for step-sizes larger than 0.005, oscillations of the learning process prevent good approximation. Any constant step-size works worse than the one adapted by the fixed point method.

VIII. Conclusions and future work

In this paper a schema of step-size adaptation has been introduced that has several advantages:

- It is fully autonomous in the sense that it does not rely on any problem–dependent coefficients, such as the meta-step-size.
- The schema is not based on assumptions concerning the momentary loss function optimized through learning, as the algorithm presented in [7].
- The complexity of the computation leading to a step-size adjustment is linear in number of neural weights, unlike Extended Kalman Filter [5].
- The schema makes the learning stable and efficient.

Effectiveness of the proposed method is confirmed experimentally.

The empirical results encourage further development of the presented approach. Another step-size adaptation schema is left to be developed namely, the method that assigns a separate step-size to each weight of the network. Also, limit properties of the presented approach must be investigated as well its effectiveness should be compared experimentally with a wider range of alternative methods.

References

[1] M. T. Hagan and M. Menhaj, “Training feedforward networks with the marquardt algorithm,” *IEEE Trans. on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[2] E. Castillo, B. Guijarro-Berdinas, O. Fontenla-Romero, and A. Alonso-Betanzos, “A very fast learning method for neural networks based on sensitivity analysis,” *Journal of Machine Learning Research*, vol. 7, pp. 1159–1182, 2006.

[3] L. B. Almeida, T. Langlois, and J. D. Amaral, “On-line step size adaptation,” INESC/IST, Tech. Rep., 1997.

[4] R. A. Jacobs, “Increased rates of convergence through learning rate adaptation,” *Neural Networks*, vol. 1, no. 4, pp. 295–308, 1988.

[5] Y. Iiguni, H. Sakai, and H. Tokumaru, “A real time learning lagorithm for a multilayered neural network based on extended kalman filter,” *IEEE Trans. on Signal Processing*, vol. 45, no. 6, pp. 959–966, 1992.

[6] N. N. Schraudolph and X. Giannakopoulos, “Online independent component analysis with local learning rate adaptation,” in *Advances in NIPS*, vol. 12, 2000, pp. 789–795.

[7] L. Behera, S. Kumar, and A. Patnaik, “On adaptive learning rate that guarantees convergence in feedforward networks,” *IEEE Trans. on Neural Networks*, vol. 17, no. 5, pp. 1116–1125, 2006.

[8] P. Wawrzyński, “Real-time reinforcement learning by sequential actor-critics and experience replay,” *Neural Networks*, vol. 22, pp. 1484–1497, 2009.

[9] R. S. Sutton, “Adapting bias by gradient descent: An incremental version of delta-bar-delta,” in *Proc. of the 10th NCAI*, 1992, pp. 171–176.

[10] N. N. Schraudolph, J. Yu, and D. Aberdeen, “Fast online policy gradient learning with SMD gain vector adaptation,” in *Advances in NIPS*, vol. 18, 2006, pp. 1185–1192.

[11] I. Noda, “Recursive adaptation of stepsize parameter for non-stationary environments,” in *Principles of Practice in Multi-Agent Systems*, 2009, pp. 525–533.

[12] H. J. Kushner and G. Yin, *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, 1997.