WHY NATURAL GRADIENT?

S. Amari[†] *and S.C. Douglas*[‡]

[†]Brain-Style Information Systems Group, RIKEN Brain Science Institute, Wako-shi, Saitama 351-01 JAPAN

[‡]Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112 USA

ABSTRACT

Gradient adaptation is a useful technique for adjusting a set of parameters to minimize a cost function. While often easy to implement, the convergence speed of gradient adaptation can be slow when the slope of the cost function varies widely for small changes in the parameters. In this paper, we outline an alternative technique, termed natural gradient adaptation, that overcomes the poor convergence properties of gradient adaptation in many cases. The natural gradient is based on differential geometry and employs knowledge of the Riemannian structure of the parameter space to adjust the gradient search direction. Unlike Newton's method, natural gradient adaptation does not assume a locally-quadratic cost function. Moreover, for maximum likelihood estimation tasks, natural gradient adaptation is asymptotically Fisher-efficient. A simple example illustrates the desirable properties of natural gradient adaptation.

1. INTRODUCTION

Parameter estimation is the task of determining useful numerical values for certain constants within some problem structure. Perhaps the most popular parameter estimation method is *gradient descent*, an iterative optimization procedure that is often computationally-simple to implement. Its ability both to converge quickly to and to identify the best parameter values for a particular problem are limited in some cases. Even so, it has found wide use in numerous areas and is extremely popular in the adaptive filtering and neural network fields, as the well-known least-mean-square (LMS) and backpropagation algorithms are both gradient descent methods [1].

Although often not explicitly stated, gradient descent is most useful for cost functions (i) that have a single minimum and (ii) whose gradients are isotropic in magnitude with respect to any direction away from this minimum. In practice, however, the cost function being optimized is multimodal, and the gradient magnitudes are non-isotropic about any minimum. In such cases, the parameter estimates are only guaranteed to locally-minimize the cost function, and convergence to any local minimum can be slow.

This paper outlines a useful alternative to standard gradient adaptation. Termed *natural gradient adaptation*, the proposed iterative procedure modifies the standard gradient search direction according to the Riemannian structure of the parameter space [2, 3]. While not removing local cost function minima, natural gradient adaptation provides isotropic convergence properties about any local minimum independently of the model parametrization and of the dependencies within the signals being processed by the algorithm. Moreover, natural gradient adaptation overcomes many of the limitations of Newton's method, which assumes that the cost function being minimized is approximately locally-quadratic [4]. For this reason, it is appropriate for a large class of cost functions and for certain nonlinear models such as neural networks. Its two chief drawbacks are (i) the knowledge required to determine the Riemannian structure of the parameter space and (ii) the complexity of the resulting algorithm, although cases exist for which natural gradient adaptation has a simple form.

The organization of this paper is as follows. In the next section, we review the basic concepts behind standard gradient descent and indicate some of its performance limitations. The natural gradient is introduced in Section 3, and its general properties are discussed in Section 4. A simple example in Section 5 indicates some of the useful properties of natural gradient adaptation. Conclusions are drawn in Section 6.

2. STANDARD GRADIENT ADAPTATION

To describe standard gradient adaptation, define a vector of N adjustable parameters as

$$\mathbf{w}(k) = [w_1(k)\ w_2(k)\ \cdots\ w_N(k)]^T \quad (1)$$

where $w_i(k)$ is the ith parameter value at time k. For any $\{w_i(k)\}$, we define a twice-differentiable scalar *cost function* $\mathcal{J}(\mathbf{w})$ such that there exists at least one parameter vector $\mathbf{w}_{opt} = [w_{1,opt}\ \cdots\ w_{N,opt}]^T$ for which

i) the N-dimensional *gradient* of $\mathcal{J}(\mathbf{w})$, defined as

$$\frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}} = \left[\frac{\partial \mathcal{J}(\mathbf{w})}{\partial w_1}\ \frac{\partial \mathcal{J}(\mathbf{w})}{\partial w_2}\ \cdots\ \frac{\partial \mathcal{J}(\mathbf{w})}{\partial w_N}\right]^T \quad (2)$$

has all zero entries for $\mathbf{w} = \mathbf{w}_{opt}$, and

ii) the $(N \times N)$ *Hessian matrix* $\mathbf{F}(\mathbf{w})$ with entries $f_{ij}(\mathbf{w})$ given by

$$f_{ij}(\mathbf{w}) = \frac{\partial^2 \mathcal{J}(\mathbf{w})}{\partial w_i \partial w_j} \quad (3)$$

is positive-definite for $\mathbf{w} = \mathbf{w}_{opt}$.

When the above conditions hold, $\mathbf{w}_{opt}$ represents a local minimum of the cost function in parameter space.

The *steepest descent method* is an iterative procedure for locally-minimizing $\mathcal{J}(\mathbf{w})$ with respect to $\mathbf{w}$, defined as

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu(k)\frac{\partial \mathcal{J}(\mathbf{w}(k))}{\partial \mathbf{w}}, \quad (4)$$

0-7803-4428-6/98 $10.00 © 1998 IEEE

where $\mathbf{w}(0)$ is any initial parameter vector and $\mu(k)$, $k = \{0, 1, 2, \ldots\}$ is a positive-valued sequence of *step sizes*.

The steepest descent method is an iterative technique in which a fraction of the gradient of the cost function with respect to each parameter is subtracted from each parameter. This process is continued indefinitely or until the value of $\mathcal{J}(\mathbf{w}(k))$ reaches a suitably-small value, at which point $\mathbf{w}(k)$ is "close" to $\mathbf{w}_{opt}$. With proper selection of $\mu(k)$, the steepest descent method adjusts $\mathbf{w}(k)$ so that $\lim_{k\to\infty} \mathbf{w}(k) = \mathbf{w}_{opt}$ for values of $\mathbf{w}(0)$ that are suitably-close to $\mathbf{w}_{opt}$. Such an algorithm causes $\mathbf{w}(k)$ to *converge* to $\mathbf{w}_{opt}$. Depending on the form of $\mathcal{J}(\mathbf{w})$, convergence to $\mathbf{w}_{opt}$ occurs according to certain measures when particular step size sequences are used [5].

The transient behavior of any gradient descent method depends on the form of $\mathcal{J}(\mathbf{w})$. The convergence of $\mathbf{w}(k)$ to $\mathbf{w}_{opt}$ can be fast or slow, depending on the choice of $\mu(k)$ and on the gradient components $\partial\mathcal{J}(\mathbf{w}(k))/\partial w_i$. In general, it is difficult or impossible to choose $\mu(k)$ in such a way as to provide fast convergence from all possible initial coefficient values $\mathbf{w}(0)$. This difficulty stems from the fact that the gradient components $\partial\mathcal{J}(\mathbf{w}(k))/\partial w_i$ vary widely in magnitude in different directions from $\mathbf{w}_{opt}$ for typical parametrizations and cost functions. Without knowledge of where $\mathbf{w}(k)$ is with respect to $\mathbf{w}_{opt}$ in the space of possible parameters, one cannot choose $\mu(k)$ properly to obtain fast convergence of each $w_i(k)$ to $w_{i,opt}$.

3. NATURAL GRADIENT ADAPTATION

To understand natural gradient learning requires us to reconsider the fundamental notion of *distance*. In everyday life, we possess an intuitive concept of distance, as embodied in the familiar folk axiom "the shortest distance between two points is a straight line." In mathematical terms, the *Euclidean distance* between two points $\mathbf{v}$ and $\mathbf{v} + \delta\mathbf{v}$ in the N-dimensional space spanned by $\mathbf{v} = [v_1\ v_2\ \cdots\ v_N]^T$ is

$$d_E(\mathbf{v}, \mathbf{v}+\delta\mathbf{v}) = \sqrt{\sum_{i=1}^{N} \delta v_i^2} = \sqrt{\delta\mathbf{v}^T\delta\mathbf{v}} = ||\delta\mathbf{v}||_2, \quad (5)$$

where $\delta\mathbf{v} = [\delta v_1\ \delta v_2\ \cdots\ \delta v_N]^T$ and $||\mathbf{v}||_2$ denotes the L_2 or *Euclidean norm* of $\mathbf{v}$.

In reality, however, "straight-line" or Euclidean distance represents an approximation to a more-complex notion of distance. One such example can be found in most geographical maps, which depict the world as a flat two-dimensional plane instead of the curved surface that it is. The shortest physical route between two locations is represented as an arc on such a map, which actually is along the geodesic defined by the global curvature of the earth's surface. Surprisingly, non-Euclidean measures of distance are useful for predicting certain properties of the physical world. In his famous Theory of Relativity, Einstein postulated that mass distorts physical space, a fact that was verified in 1919 when photographs of the sun during a solar eclipse showed that the visual positions of stars were apparently altered by the sun's mass. The path taken by the stars' light is the shortest in gravitational space, although it is clearly "bent" in Euclidean terms. To predict such effects, one must employ the mathematics of curved space, a field known as *differential* or *Riemannian geometry*. It is Riemannian geometry upon which natural gradient adaptation is based [2].

In Riemannian geometry, distance is not measured according to the Euclidean norm in (5). Rather, for two vectors $\mathbf{w}$ and $\mathbf{w} + \delta\mathbf{w}$ where the elements of $\delta\mathbf{w}$ are of small magnitude, we define the distance metric $d_{\mathbf{W}}(\cdot,\cdot)$ at $\mathbf{w}$ as

$$d_{\mathbf{W}}(\mathbf{w}, \mathbf{w}+\delta\mathbf{w}) = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} \delta w_i \delta w_j g_{ij}(\mathbf{w})} \quad (6)$$

$$= \sqrt{\delta\mathbf{w}^T\mathbf{G}(\mathbf{w})\delta\mathbf{w}}, \quad (7)$$

where $\mathbf{G}(\mathbf{w})$, the *Riemannian metric tensor*, is an $(N \times N)$ positive-definite matrix whose (i,j)th entry is $g_{ij}(\mathbf{w})$. The Riemannian metric tensor characterizes the intrinsic curvature of a particular manifold in N-dimensional space. In the case of the Euclidean coordinate system, $\mathbf{G}(\mathbf{v}) = \mathbf{I}$ is the identity matrix, such that (6) reduces to (5).

When the nature of the manifold can be described in terms of a transformation of Euclidean orthogonal space with coordinate vector $\mathbf{v}$ to $\mathbf{w}$, then one can determine the form of $\mathbf{G}(\mathbf{w})$ through the relationship

$$d_E^2(\mathbf{v}, \mathbf{v}+\delta\mathbf{v}) = d_{\mathbf{W}}^2(\mathbf{w}, \mathbf{w}+\delta\mathbf{w}), \quad (8)$$

where $\delta\mathbf{v}$ is small and $\mathbf{w} + \delta\mathbf{w}$ is the transformed value of $\mathbf{v} + \delta\mathbf{v}$. An example illustrates this calculation.

Example: Define $\mathbf{v} = [x\ y]^T$ such that $\{x, y\}$ define a point in two-dimensional Euclidean space. We can represent this point using polar coordinates as

$$x = r\cos\theta, \qquad y = r\sin\theta, \quad (9)$$

where $\mathbf{w} = [r\ \theta]^T$ represents the same point in polar space.

The distance between $\mathbf{v}$ and $\mathbf{v}+\delta\mathbf{v}$ in Euclidean terms is

$$d_E(\mathbf{v}, \mathbf{v}+\delta\mathbf{v}) = \sqrt{\delta x^2 + \delta y^2}. \quad (10)$$

The Riemannian metric tensor $\mathbf{G}(\mathbf{w})$ for polar space is defined such that (8) holds. Noting (9), we have that

$$\mathbf{v} + \delta\mathbf{v} = \begin{bmatrix} (r+\delta r)\cos(\theta+\delta\theta) \\ (r+\delta r)\sin(\theta+\delta\theta) \end{bmatrix} \quad (11)$$

$$= \begin{bmatrix} r\cos\theta + \delta r\cos\theta - \delta\theta r\sin\theta \\ r\sin\theta + \delta r\sin\theta + \delta\theta r\cos\theta \end{bmatrix}, \quad (12)$$

where we have neglected terms of the form $\delta r\delta\theta^i$ and $\delta\theta^{i+1}$ for $i \geq 1$. Subtracting $\mathbf{v}$ from both sides of (12) gives

$$\delta\mathbf{v} = \begin{bmatrix} \delta r\cos\theta - \delta\theta r\sin\theta \\ \delta r\sin\theta + \delta\theta r\cos\theta \end{bmatrix}. \quad (13)$$

Therefore, (10) becomes

$$d_E^2(\mathbf{v}, \mathbf{v}+\delta\mathbf{v}) = \delta r^2 + r^2\delta\theta^2 = \delta\mathbf{w}^T\mathbf{G}(\mathbf{w})\delta\mathbf{w}, \quad (14)$$

where the Riemannian metric tensor for $\mathbf{w}$ is found to be

$$\mathbf{G}(\mathbf{w}) = \begin{bmatrix} 1 & 0 \\ 0 & r^2 \end{bmatrix}. \quad (15)$$

□.

Although the previous example indicates that $\mathbf{G}(\mathbf{w})$ depends on the value of $\mathbf{w}$ in general, it can be constant-valued in certain cases. The most-celebrated example is when $\mathbf{w}$ is obtained from $\mathbf{v}$ in standard Euclidean space via a linear transformation [3]. Morever, despite the form of the result in (15), $\mathbf{G}(\mathbf{w})$ is not diagonal in general.

In Euclidean coordinates, the gradient is defined in (2). It should come as no surprise, therefore, that cost functions that emulate the Euclidean distance measure in (5) are well-matched to gradient adaptation. In fact, if $\mathcal{J}(\mathbf{w}) =$

$c||\mathbf{w} - \mathbf{w}_{opt}||_2^2$, where c is an arbitrary constant, then $-\partial\mathcal{J}(\mathbf{w}(k))/\partial\mathbf{w} = -2c(\mathbf{w}(k) - \mathbf{w}_{opt})$, such that a step size value of $\mu(k) = 1/(2c)$ allows (4) to converge to $\mathbf{w}_{opt}$ in one step, irrespective of $\mathbf{w}(k)$. This result is obtained because the negative gradient of $c||\mathbf{w}-\mathbf{w}_{opt}||_2^2$ at any $\mathbf{w}$ points towards $\mathbf{w}_{opt}$.

In practice, the cost function to be minimized is not Euclidean, and the underlying space of parameters is not Euclidean but is curved and distorted, *i.e.* Riemannian. Thus, the negative of the standard gradient $-\partial\mathcal{J}(\mathbf{w})/\partial\mathbf{w}$ does not represent the steepest descent direction of the cost function $\mathcal{J}(\mathbf{w})$ in the parameter space, and thus the standard gradient direction is no longer appropriate. To obtain an algorithm with useful convergence properties, the standard gradient direction should be modified according to the local curvature of the parameter space. Such a technique is called *natural gradient adaptation* [2]. The parameter updates for natural gradient adaptation are given by

$$\mathbf{w}(k+1) \quad = \quad \mathbf{w}(k) - \mu(k)\mathbf{G}^{-1}(\mathbf{w}(k))\frac{\partial\mathcal{J}(\mathbf{w}(k))}{\partial\mathbf{w}}, \quad (16)$$

where $\mathbf{G}(\mathbf{w})$ is the Riemannian metric tensor for the parameter vector $\mathbf{w}$ as defined by the manifold of parameters. The form of $\mathcal{J}(\mathbf{w}(k))$ can be defined explicitly as in the Euclidean case, by measurements as in deterministic cost functions, or by instantaneous approximations as in stochastic gradient descent.

The Riemannian metric tensor is naturally defined from the characteristics of the parameter space in most practical applications. As a simple example, we can consider parameter spaces that are obtained from transformations of the orthogonal Euclidean space where an Euclidean-based cost function is most appropriate. In such situations, $\mathcal{J}(\mathbf{w}+\delta\mathbf{w})$ is identical to the squared Riemannian distance measure $d_{\mathbf{W}}^2(\mathbf{w}, \mathbf{w}+\delta\mathbf{w})$ in (6), and thus $\mathbf{G}(\mathbf{w})$ can be determined from the relationship in (8). Simulation examples employing these results are given in Section 5.

4. PROPERTIES OF NATURAL GRADIENT ADAPTATION

In this section, we summarize several properties about the natural gradient adaptive algorithm in (16). Details concerning these properties can be found in the references provided in a forthcoming paper [3].

1. Natural gradient adaptation differs from Newton's method in general. Newton's method employs the inverse of the Hessian $\mathbf{F}(\mathbf{w})$ to adjust the gradient search direction in (4). When $\mathcal{J}(\mathbf{w})$ is a quadratic function of $\mathbf{w}$, $\mathbf{F}(\mathbf{w})$ is equal to $\mathbf{G}(\mathbf{w})$ for the underlying parameter space, and thus Newton's method and natural gradient adaptation are identical. In more general contexts, the two techniques are different. In particular, $\mathbf{G}(\mathbf{w})$ is always positive definite by construction, whereas $\mathbf{F}(\mathbf{w})$ may not be for particular choices and values of $\mathcal{J}(\mathbf{w})$ and $\mathbf{w}$, respectively.

2. Natural gradient adaptation is asymptotically Fisher-efficient. The Riemannian metric tensor in the space of the parameters within a statistical model is the Fisher information matrix. Employing the natural gradient algorithm for the maximum-likelihood cost function in this situation with $\mu(k) = 1/k$, the asymptotic covariance matrix of the resulting parameter errors approaches the well-known Cramer-Rao lower bound for unbiased parameter estimates. In practical terms, this result means that natural gradient adaptation provides fast adaptation performance in such problems.

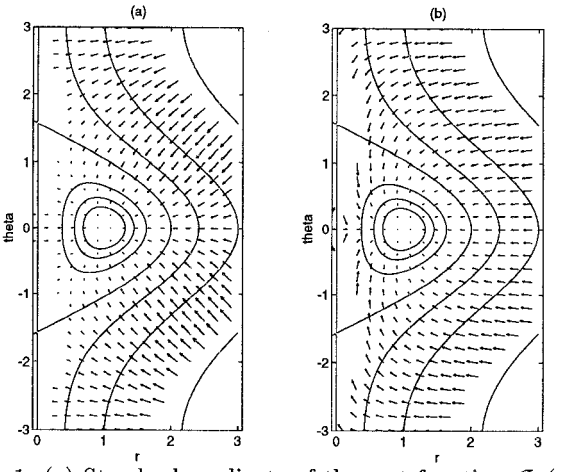

Fig. 1: (a) Standard gradients of the cost function $\mathcal{J}_P(\mathbf{w})$. (b) Natural gradients of the cost function $\mathcal{J}_P(\mathbf{w})$.

3. Natural gradient adaptation involves nonlinear parameter updates in general. Thus, characterizing the transient performance characteristics of natural gradient adaptation is a challenging task. Certain transient and steady-state properties of several natural gradient algorithms have already been studied, among them algorithms for blind source separation, spatial and temporal decorrelation, blind equalization, multichannel blind deconvolution, and multilayer perceptron training. The nonlinear form of the natural gradient algorithm implies, however, that sufficient bounds on the step size $\mu(k)$ to guarantee stability and convergence of the algorithm's parameters are often difficult to obtain.

4. Natural gradient adaptation requires extensive knowledge of the structure of the parameter estimation problem. Although the natural gradient is local in nature and only depends on the parameter values $\mathbf{w}(k)$, determining $\mathbf{G}(\mathbf{w})$ usually requires precise knowledge of the problem structure. However, the information needed to form $\mathbf{G}(\mathbf{w})$ varies from problem to problem, and there exist several practical cases where this information is easily obtained.

5. Natural gradient adaptation can be simpler to implement than standard gradient adaptation. There exist problems for which the update in (16) is simpler to compute than that in (4). Such problems generally involve the estimation of a linear system (*e.g.* a gain matrix or one or more transfer functions) that is related to the inverse of some unknown system model. In some cases, the matrix $\mathbf{G}(\mathbf{w})$ is a function only of the estimated parameters, and these parameters can combine with the components of $\partial\mathcal{J}(\mathbf{w})/\partial\mathbf{w}$ to simplify the algorithm's structure.

5. A SIMPLE EXAMPLE

Consider the cost function

$$\mathcal{J}_P(\mathbf{w}) \quad = \quad \frac{1}{2}\left[\{r\cos\theta - 1\}^2 + r^2\sin^2\theta\right], \quad (17)$$

where we define $\mathbf{w} = [r \;\; \theta]^T$. This cost function is identical to $\mathcal{J}_E(\mathbf{v}) = \{x-1\}^2 + y^2$, where the relationship between $\mathbf{v}$ and $\mathbf{w}$ is given by (9). Shown in Fig. 1(a) are the contours and gradients of $\mathcal{J}_P(\mathbf{w})$ for different $\mathbf{w}$ over the range $0 \leq$

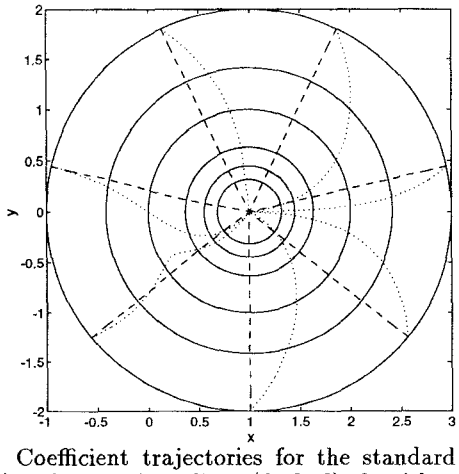

Fig. 2: Coefficient trajectories for the standard gradient (dotted) and natural gradient (dashed) algorithms for minimizing $\mathcal{J}_P(\mathbf{w})$, as plotted in Euclidean $\{x, y\}$-coordinates.

$r \leq 3$ and $|\theta| < \pi$, where

$$-\frac{\partial \mathcal{J}_P(\mathbf{w})}{\partial \mathbf{w}} = -\begin{bmatrix} r - \cos\theta \\ r\sin\theta \end{bmatrix}. \quad (18)$$

The magnitudes of the gradients vary widely across the error surface, and they do not point towards any $\mathbf{w}_{opt}^{(n)} = [(-1)^n \ \pi n]^T$ in general. In particular, the gradients are small along the locus of points $2|\theta| = \pi(1 - r)$ for $r > 0$ and $|\theta| < \pi$. When performing a steepest descent search of this cost function, the coefficient trajectories are far from straight paths, and it is challenging to choose $\mu(k)$ to quickly cause $\mathbf{w}(k)$ to converge to any $\mathbf{w}_{opt}^{(n)}$.

Because $\mathcal{J}_P(\mathbf{w})$ is equivalent to the squared Euclidean distance measure $\mathcal{J}_E(\mathbf{v})$, we have immediately from the polar-coordinate example in Section 4 that $\mathbf{G}(\mathbf{w})$ is given by (15), and the natural gradient algorithm in this case is

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu(k)\begin{bmatrix} r(k) - \cos\theta(k) \\ \dfrac{\sin\theta(k)}{r(k)} \end{bmatrix}. \quad (19)$$

Fig. 1(b) shows the natural gradient flows in this example. Note that the natural gradient flow lines are of a more-equal magnitude accros the error surface as compared to the standard gradient flows, and more importantly, the magnitudes of the natural gradients no longer greatly vary with angle away from $[1 \ 0]^T$ as do the standard gradient flows.

The useful behavior of the natural gradient method is shown via the coefficient trajectories of $\mathbf{w}(k)$ as transformed into Euclidean parameter space. Fig. 2 shows these trajectories for $\mu(k) = 0.05$, in which the behavior of natural gradient adaptation on $\mathcal{J}_P(\mathbf{w})$ is nearly-identical to that of standard gradient adaptation on $\mathcal{J}_E(\mathbf{v})$. The nearly-straight-path trajectories are in sharp contrast with the circuitous paths provided by standard gradient adaptation on $\mathcal{J}_P(\mathbf{w})$ as expressed in Euclidean parameter space.

We apply these concepts to sinusoidal estimation, where

$$y(k) = r_{opt}\cos(\omega k + \theta_{opt}) + \eta(k), \quad (20)$$

is a discrete-time sinusoid of known frequency corrupted by white Gaussian noise with variance σ_η^2. For the mean-squared error criterion, the natural gradient algorithm for

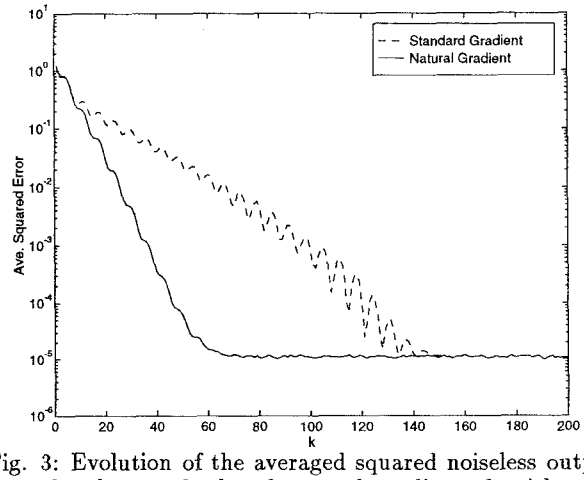

Fig. 3: Evolution of the averaged squared noiseless output error for the standard and natural gradient algorithms in the sinusoidal estimation task.

estimating $[r_{opt} \ \theta_{opt}]^T$ given a realization of $y(k)$ is [3]

$$r(k+1) = r(k) + \mu(k)e(k)\cos(\omega k + \theta(k)) \quad (21)$$

$$\theta(k+1) = \theta(k) - \mu(k)e(k)\frac{\sin(\omega k + \theta(k))}{r(k)} \quad (22)$$

$$e(k) = y(k) - r(k)\cos(\omega k + \theta(k)). \quad (23)$$

Fig. 3 shows the averaged squared noiseless error $\{e(k) - \eta(k)\}^2$ for the standard and natural gradient algorithms in this task, as obtained from an average of 1000 different simulation runs with $r_{opt} = 1$, $\omega = 0.5$, $\sigma_\eta^2 = 0.0001$, $\mu(k) = 0.2$, , and uniformly-distributed values for $r(0)$, $\theta(0)$, and θ_{opt}. The natural and standard gradient algorithm's squared noiseless output errors converge to the same steady-state level of 10^{-5} in about 70 and 140 iterations, respectively. The natural gradient algorithm provides inherently faster convergence to a low steady-state error as compared to the standard gradient algorithm in this situation.

6. CONCLUSIONS

This paper has outlined a novel optimization method known as natural gradient adaptation. The technique provides more-uniform convergence performance about a local minimum of a cost function as compared to that of standard gradient adaptation. The algorithm can also be simple to implement in some cases. The main disadvantage of natural gradient adaptation is the knowledge of the parameter space or cost function error surface that is required to determine the final form of the algorithm in any particular case. More details concerning the form and performance of natural gradient algorithms for perceptron training, instantaneous blind source separation, and blind equalization and deconvolution can be found in the forthcoming paper [3].

REFERENCES

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation* (New York: Macmillan, 1994).

[2] S. Amari, "Natural gradient works efficiently in learning," accepted for publication in *Neural Computation;* to appear.

[3] S.C. Douglas and S. Amari, "Natural gradient adaptation," *Proc. IEEE,* vol. 86, no. 6, June 1998; to appear.

[4] D.G. Luenberger, *Linear and Nonlinear Programming,* 2nd ed. (Reading, MA: Addison-Wesley, 1984).

[5] H.J. Kushner and D.S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems* (New York: Springer-Verlag, 1978).