International Journal of Bifurcation and Chaos, Vol. 16, No. 7 (2006) 2081–2091

APPROACHES TO ADAPTIVE STOCHASTIC SEARCH BASED ON THE NONEXTENSIVE q-DISTRIBUTION

GEORGE D. MAGOULAS
School of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London, WC1E 7HX, UK

ARISTOKLIS ANASTASIADIS
Birkbeck College, University of London, Malet Street, London, WC1E 7HX, UK

Received January 25, 2005; Revised April 4, 2005

This paper explores the use of the nonextensive q-distribution in the context of adaptive stochastic searching. The proposed approach consists of generating the "probability" of moving from one point of the search space to another through a probability distribution characterized by the q entropic index of the nonextensive entropy. The potential benefits of this technique are investigated by incorporating it in two different adaptive search algorithmic models to create new modifications of the diffusion method and the particle swarm optimizer. The performance of the modified search algorithms is evaluated in a number of nonlinear optimization and neural network training benchmark problems.

Keywords: Adaptive stochastic search; particle swarm; simulated annealing, diffusion; nonextensive statistics.

1. Introduction

Adaptive stochastic search methods are expected to lead to "optimal" or "near-optimal" configurations of a system or model as they manage to escape from sub-optimal (local) solutions. In that sense, they provide an automated "optimization" approach that handles the uncertainty arising from stochastic elements in either the environment (process noise/concept drift) or the objective function evaluation process (observation noise/parameters noise) and ultimately improve the performance of a simulated model or process. They include methods like *simulated annealing* [Corana *et al.*, 1987; Kirkpatrick *et al.*, 1983], *genetic and evolutionary algorithms* [Michalewicz, 1996], and *swarm intelligence* [Eberhart *et al.*, 1996; Kennedy & Eberhart, 1995]. Nevertheless, adaptive search methods often detect suboptimal solutions of the objective function in practical applications. This may happen either because finding the global solution is too time consuming or because the algorithm is "getting stuck" in local optima. In many *NP*-hard problems these suboptimal solutions are acceptable but there are applications where the optimal solution is not only desirable but also indispensable.

Several approaches have been proposed to alleviate this situation, particularly techniques of escaping local optima. *Restart algorithms*, on the one hand, generate a new starting point and commence the search all over again after converging to a local optimum. Their use is not restricted to a particular application area (e.g. [Magdon-Ismail & Atiya, 2000]), and they can be combined with almost any method [Muselli, 1997], from genetic algorithms [Ghannadian *et al.*, 1996] to the classic

"iterated hill-climber" [Michalewicz & Fogel, 2000]. Popular techniques to escape local optima within a single run of an algorithm, on the other hand, include accepting new candidate points/solutions in the search space depending on a probability distribution, like in the *Simulated Annealing* (SA) and the *stochastic hill-climbing*, or forcing the algorithm to explore further portions of the search space, like in the *tabu search* method.

This paper focuses on this second category of techniques for escaping local optima, and particularly on the use of probabilistic models in performing global exploration of the space. These models are usually inspired by the Boltzmann–Gibbs entropy $S_{BG} = -K \sum_i p_i \ln p_i$ that provides exponential laws for describing stationary states and basic time-dependent phenomena, where $\{p_i\}$ are the probabilities of the microscopic configurations, and $K > 0$. Attempts to explore the benefits of this approach have focused mainly on the use of Gaussian distributions, and applied to several cases, such as training complex neural architectures like the Boltzmann machine [Ackley *et al.*, 1985], training multilayer neural networks with noise that follows a gaussian distribution [Burton & Mpitsos, 1992; Rögnvaldsson, 1994], designing Boltzmann selection in genetic algorithms [Michalewicz, 1996; Prugel–Bennett & Shapiro, 1994], performing model reduction for control systems using evolutionary methods [Li *et al.*, 1997], equipping the Particle Swarm Optimizer (PSO) with gaussian mutation [Higashi & Iba, 2003] or updating the PSO velocities using gaussian distributions [Krohling *et al.*, 2004].

This paper builds on the theory of nonextensive statistical mechanics to equip adaptive search schemes with an alternative stochastic model. In the next section the new model is described and then used to design modifications of two well-known algorithms, namely Simulated Annealing and Particle Swarm. Section 3 presents experimental results from testing the modified algorithms in training multilayer neural networks with noise and optimizing nonlinear functions. Section 4 presents conclusions and directions for future work.

2. Adaptive Stochastic Search Based on the Nonextensive Entropic Index

The nonextensive entropy has been defined by Tsallis [1988] as:

$$S_q \equiv K \frac{1 - \sum_{i=1}^{W} p_i^q}{q-1} \quad (q \in \mathbb{R}), \tag{1}$$

where W is the total number of microscopic configurations, whose probabilities are $\{p_i\}$, K is a conventional positive constant, and q is *nonextensive entropic index*. When the entropic index $q = 1$, Eq. (1) recovers to Boltzmann–Gibbs entropy. The entropic index works like a biasing parameter: $q < 1$ privileges rare events (values of p close to 0 are benefited), while $q > 1$ privileges common events (values of p close to 1). The optimization of the entropic form described by Eq. (1) under appropriate constraints [Tsallis, 1988], yields for the canonical ensemble

$$p_i \propto [1 - (1-q)\beta E_i]^{\frac{1}{(1-q)}} \equiv e_q^{-\beta E_i}, \tag{2}$$

where β is a Lagrange parameter, $\{E_i\}$ is the energy spectrum, and the *q-exponential function* is defined as:

$$e_q^x \equiv [1 + (1-q)x]^{\frac{1}{(1-q)}} = \frac{1}{[1 - (q-1)x]^{\frac{1}{(q-1)}}} \tag{3}$$

Inspired from this approach, we introduce stochasticity in search by a nonextensive schedule of the form:

$$Q_{(T,k)} = e_q^{-T(\ln 2)\cdot k} = [1 - (1-q)T(\ln 2)\cdot k]^{\frac{1}{1-q}}, \tag{4}$$

where T is the temperature and k indicates iterations. In the sequel, we equip two adaptive stochastic search algorithms with the nonextensive schedule, as described below, in an attempt to explore the use of this approach in minimizing nonlinear functions.

2.1. *A hybrid diffusion algorithm with nonextensive schedule*

Simulated Annealing is a method based on Monte Carlo simulation, which solves difficult combinatorial optimization problems. The name comes from the analogy to the behavior of physical systems by melting a substance and lowering its temperature slowly until it reaches freezing point (physical annealing). Simulated annealing was first used for optimization by Kirkpatrick *et al.* [1983]. In this context, SA is a procedure that has the capability to move out of regions near local minima [Corana *et al.*, 1987; Szu, 1987]. SA is based on random evaluations of the objective function, in such a way that transitions out of a local minimum are possible.

Thus, candidate solutions are updated following the relation

$$x^{k+1} = x^k + \Delta x \tag{5}$$

where k is the iteration number, and Δx is random noise from a uniform distribution. The method applies the *Metropolis criterion*, i.e. it either accepts or rejects a candidate solution depending on the probability

$$p(x^k \to x^{k+1}) = \begin{cases} 1, & \text{if } \Delta f = f(x^{k+1}) \\ & \quad - f(x^k) < 0 \\ e^{-\frac{\Delta f}{T}}, & \text{otherwise} \end{cases} \tag{6}$$

Thus, if the sign of $\Delta f = f(x^{k+1}) - f(x^k)$ is *negative*, then the new point can be accepted with probability 1; otherwise, it depends on the probability value and the threshold value θ, i.e. $p(x^k \to x^{k+1}) = e^{-\Delta f/T} > \theta$.

The effectiveness of the method depends on the parameter T that controls the annealing reduction rate:

$$T^k = \frac{T^0}{1 + \ln k}. \tag{7}$$

When the temperature takes high values SA behaves like a random search, whilst at low values it works like a hill climbing procedure. Usually one starts with a high temperature value and gradually reduces: it is important to start at a temperature where most changes are accepted, i.e. probability is 1 in Eq. (6), then we can reduce the temperature in steps of a few percent while maintaining each temperature just long enough to reach an approximate solution. SA does not guarantee, of course, to find the global minimum, but if the function has many good near-optimal solutions, it should find one. First, SA reaches an area in the function domain space where a global minimizer should be present, following the gross behavior of the function, irrespectively of small local minima found on the way. It then develops finer details, finding a good, near-optimal local minimizer, if not the global minimum itself.

Despite the fact that the above approach has produced good results in several applications, there are cases where the method requires a high number of function evaluations, and the so-called *Metropolis* move of Eq. (6) is not strong enough to move the algorithm out of regions of local minina when these are considerably broad. For example, this challenging situation has been reported when minimizing nonlinear functions from the neural networks domain [Anastasiadis & Magoulas, 2004; Burton & Mpitsos, 1992; Weslstead, 1994].

Along this line, we derive a hybrid method, named Diffusion Algorithm with Nonextensive schedule (DAN):

$$\begin{aligned} x^{k+1} = x^k &- \tau^k \text{diag}\{\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k\} \nabla f(x^k) \\ &+ \mu \cdot \text{diag}\left\{ \frac{x_1^k Q_{1(T,k)}}{[1 + (x_1^k)^2]^2}, \ldots, \right. \\ &\left. \frac{x_i^k Q_{i(T,k)}}{[1 + (x_i^k)^2]^2}, \ldots, \frac{x_n^k Q_{n(T,k)}}{[1 + (x_n^k)^2]^2} \right\} \end{aligned} \tag{8}$$

where $\tau^k > 0$, η_i is the ith stepsize, and $\mu > 0$. The stepsizes are adapted according to the following schedule:

$$\text{if} \quad (\partial f_i(x^{k-1}) \cdot \partial f_i(x^k) > 0) \quad \text{then} \quad \eta_i^k = \min\{\eta_i^{k-1} \cdot \eta^+, \Delta_{\max}\}, \tag{9}$$

$$\text{if} \quad (\partial f_i(x^{k-1}) \cdot \partial f_i(x^k) < 0) \quad \text{then} \quad \eta_i^k = \max\{\eta_i^{k-1} \cdot \eta^-, \Delta_{\min}\}, \tag{10}$$

$$\text{if} \quad (\partial f_i(x^{k-1}) \cdot \partial f_i(x^k) < 0) \text{ and } (\eta_i^{k-1} < \rho Q_{i(T,k)}^2) \quad \text{then} \quad \eta_i^k = \max\{\eta_i^{k-1}\eta^- + 2c\rho Q_{i(T,k)}^2, \Delta_{\min}\} \tag{11}$$

$$\text{if} \quad (\partial f_i(x^{k-1}) \cdot \partial f_i(x^k) = 0) \quad \text{then} \quad \eta_i^k = \eta_i^{k-1}, \tag{12}$$

where $0 < \eta^- < 1 < \eta^+$, $\Delta_{\max}$ and $\Delta_{\min}$ are bounds that prevent the adaptive stepsizes from taking very large or very small values respectively, $0 < \rho < 1$ and $c \in (0, 1)$ is a random number.

Equations (8)–(12) describe a new scheme that expands further our previous work presented in [Anastasiadis & Magoulas, 2004]. This approach seems eminently suitable when the first-order derivatives are available or can be computed analytically like in neural networks training problems. The hybrid scheme of Eq. (8) does not use the notion of the acceptance probability, such as the Metropolis algorithm in the classic SA [Kirkpatrick *et al.*, 1983], or the generalized acceptance probability in the generalized SA [Tsallis & Stariolo, 1996]. Instead, it implements a form of Langevin noise that has been proved quite effective in neural systems learning, [Hoptroff & Hall, 1989; Rögnvaldsson, 1994; Treadgold & Gedeon, 1998], and in global optimization [Geman & Hwang, 1986; Gelfand & Mitter, 1991].

The nonextensive schedule is not applied proportionally to the size of each parameter; instead a form of regularization is used [Tikhonov & Arsenin,

1977]. This is realized by using bias terms which can decay small parameter values more rapidly than large values, and a regularization parameter μ that regulates the influence of the combined application of decay and nonextensive schedule (in our experiments a fixed value of $\mu = 0.01$ was used). The regularization parameter controls the "smoothness" of the regularized solution by modifying the search on the objective function landscape so that parameters with smaller values are favored at the beginning of the minimization process but as the procedure progresses the magnitude of the decay is reduced to favor the growth of parameters that have large values; this is considered beneficial for certain problems [Anastasiadis & Magoulas, 2004; Bishop, 1995].

Lastly, it is important to mention that the trade off between T and q influences the stochasticity of the hybrid algorithm, Eqs. (8)–(12). This is illustrated here using a simple problem, i.e. a single artificial neuron with two weights and a logistic activation function. The error landscape in Figs. 1 and 2 has a global minimum and two local minima. The parameter (weight) trajectories generated by the DAN algorithm are compared with the trajectories generated by a popular neural networks training

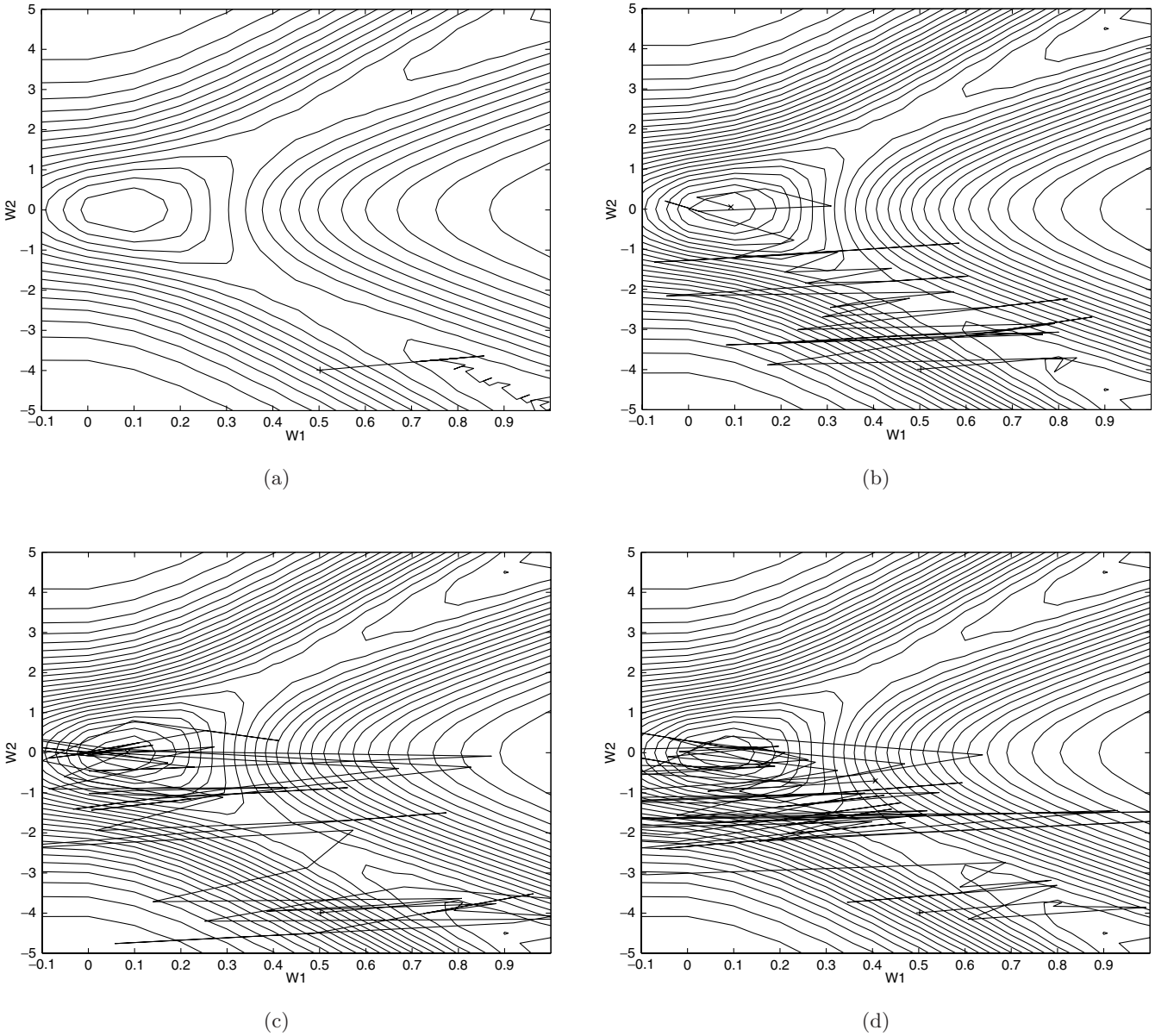
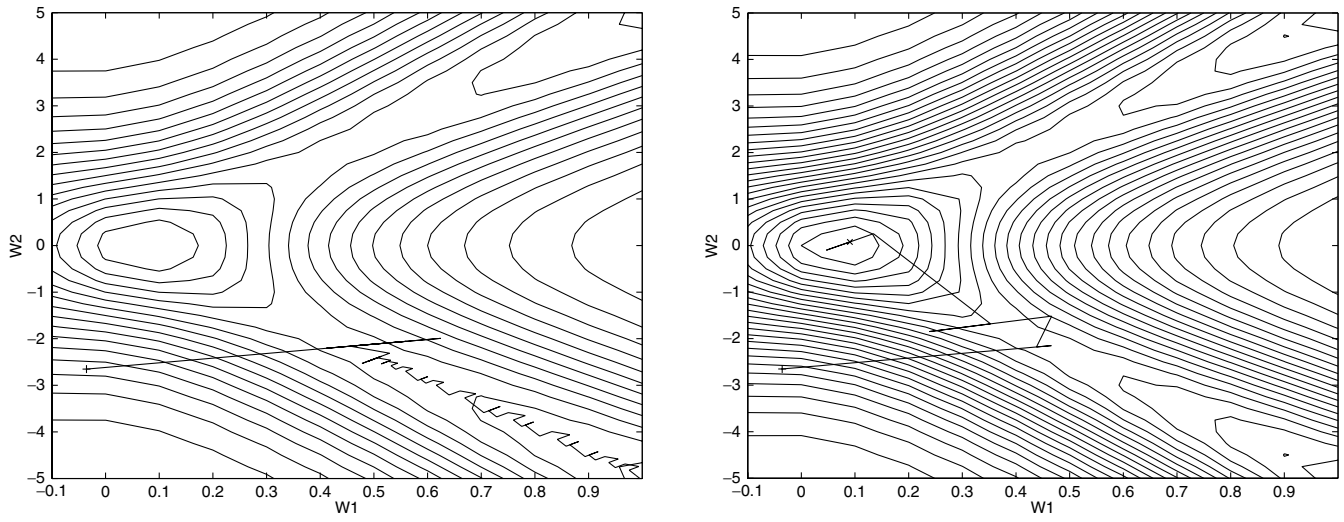
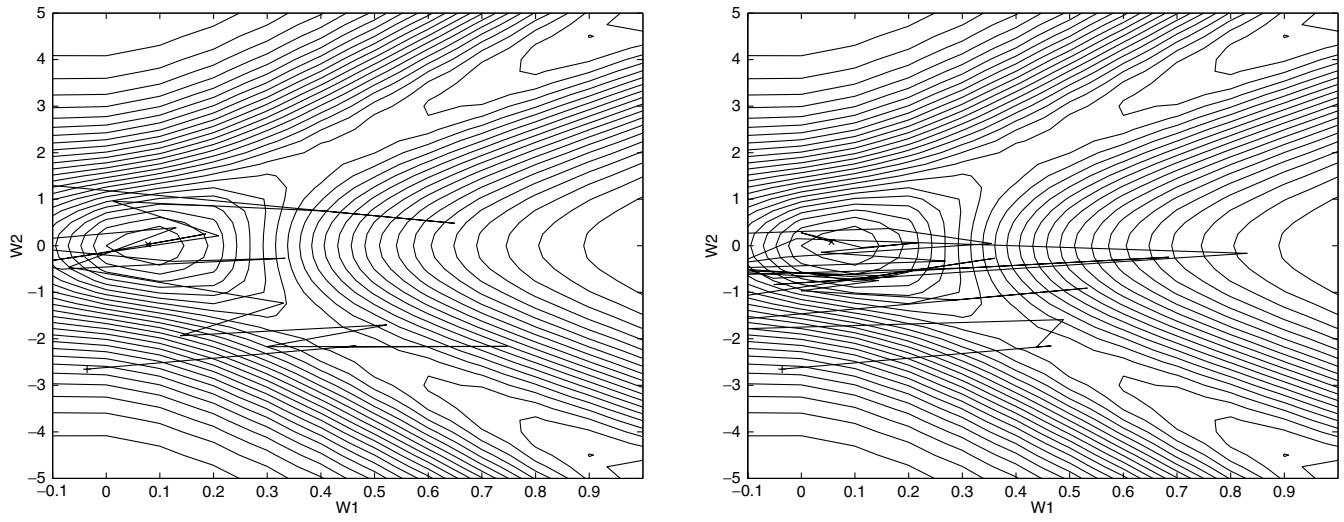


Fig. 1. Rprop and DAN weight trajectories for various T (see text for details).



(a) (b)



(c) (d)

Fig. 2. Rprop and DAN weight trajectories for various T (see text for details).

algorithm, called *Rprop* [Riedmiller & Braun, 1993], starting from the same initial conditions. Figures 1(a) and 2(a) show how Rprop converges to the local minimizer from the two different initial weights. We have applied the DAN with $q = 1.2$ for the following temperatures: $T = 0.01$ [Figs. 1(b) and 2(b)], $T = 0.001$ [Figs. 1(c) and 2(c)], $T = 0.0001$ [Figs. 1(d) and 2(d)]. The DAN algorithm escapes the region around the local minimum, and converges to the global minimizer located at the center of the contour plot in all cases. The value of T influences the shape of the DAN's trajectory. Small values generate more stochastic paths, while larger values lead to more deterministic behavior. Best results for this problem are achieved by setting $T = 0.01$.

In Sec. 3, we will test the performance of the DAN algorithm in high dimensional spaces using benchmark problems from the neural networks domain.

2.2. *Particle swarm optimizer with nonextensive schedule*

In the PSO, each companion, called *particle*, in the swarm (population) is assumed to "fly" over

the search space in order to find promising regions of the landscape. For example, in the minimization case, such regions possess lower functional values than other regions visited previously. In this context, each particle is treated as a point in an D-dimensional space which adjusts its own "flying" according to its flying experience as well as the flying experience of other particles (companions). PSO, to some extent, resembles Evolutionary Algorithms. However, in PSO, instead of using evolutionary operators, each individual (particle) updates its own position based on its own search experience and other individuals' (companions) experience and discoveries.

There are many variants of the PSO proposed so far, after Eberhart and Kennedy introduced this technique [Eberhart *et al.*, 1996; Kennedy & Eberhart, 1995]. Before deriving the method let us define the notation used in this subsection: the ith particle of the swarm is represented by the n-dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and the best particle in the swarm, i.e. the particle with the smallest function value, is denoted by the index g. The best previous position (the position giving the best function value) of the ith particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, and the position change, called *velocity*, of the ith particle is $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$.

Here we have used a version of the PSO that was derived by adding an *inertia weight* to the original PSO dynamics [Eberhart & Shi, 1998]. In this version, the particles are manipulated according to the relations

$$v_{id}^k = w_{id} v_{id}^{k-1} + c_1 r_{i1}^k (p_{id}^k - x_{id}^k) + c_2 r_{i2}^k (p_{gd}^k - x_{id}^k) \tag{13}$$

and

$$x_{id}^{k+1} = x_{id}^k + v_{id}^k \tag{14}$$

where $d = 1, 2, \ldots, D$; N is the swarm size, $i = 1, 2, \ldots, N$; w is the inertia weight; c_1, c_2 are positive constants, also called cognitive and social parameters respectively; r_{i1}^k and r_{i2}^k are two random numbers with values ranging within $[0, 1]$.

Equation (13) is used to calculate ith particle's new velocity by taking into consideration three terms: the particle's previous velocity, the distance between the particle's best previous and current positions, and, lastly, the distance between swarm's best experience (the position of the best particle in the swarm) and ith particle's current position. Then, following Eq. (14), the ith particle "flies" toward a new position x_{id}^{k+1}. Adding the velocity term to the current position, in order to generate the next position, resembles the mutation operation in evolutionary algorithms. Note that in PSO, however, the "mutation" operator is guided by particle's own "flying" experience and benefits by the swarm's "flying" experience.

In Eqs. (13) and (14) the performance of each particle is measured according to a predefined fitness function, which is problem-dependent. Also note that the role of the inertia weight, w, is considered very important in PSO convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. In this way, the parameter w regulates the trade-off between the global (wide-ranging) and local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e. fine-tuning the current search area. A suitable value for the inertia weight w usually provides balance between global and local exploration abilities and consequently a reduction on the number of iterations required to locate the optimum solution. A general rule of thumb suggests that it is better to initially set the inertia to a large value, in order to make better global exploration of the search space, and gradually decrease it to get more refined solutions, thus a time decreasing inertia weight value is used.

In Eq. (13), an alternative approach is to include a fixed *constriction* coefficient to control the magnitude of the velocity term. Although, this approach can improve the stability of the method [Clerc, 1999; Eberhart & Shi, 2000], several works have suggested that it is preferable to use the standard velocity term of Eq. (13) as the use of an inertia weight seems to provide a better swarm ability to locate a moving target [Eberhart & Shi, 2001; Hu & Eberhart, 2002].

In our approach particles have no neighborhood restrictions, the velocity is calculated using Eq. (13), and the location of a particle is updated by incorporating information from the nonextensive schedule of Eq. (4)

$$x_{id}^{k+1} = x_{id}^k + Q_{i(T,k)} \cdot v_{id}^k \tag{15}$$

where $Q_{i(T,k)}$ is defined by Eq. (4). By tuning the entropic index q and the temperature T, the term $Q_{i(T,k)}$ provides an alternative to using a fixed constriction coefficient [Clerc, 1999; Eberhart & Shi,

2000] to control the velocity term without compromising the diversity of the search. We call this modified PSO, PSO with Nonextensive schedule (PSON).

3. Application Examples

In this section we investigate the applicability of the proposed approach in two different cases in order to get an insight of its performance in different problems. The first case concerns using the DAN algorithm in neural networks training with noise, and the second one using the PSON algorithm in nonlinear optimization.

3.1. *Neural networks training with noise*

Although noise plays a influential role in the operation of real neurons, e.g. neural cells' responses to identical stimuli have been found to be stochastic in nature, the effect of noise on the training and operation of artificial neural networks has not been investigated in great depth. Notwithstanding this situation, several researchers have tried to study the influence of noise when training neural networks [Jim *et al.*, 1995; Murray & Edwards, 1993, 1994]. In [Reed *et al.*, 1995] and [Sietsma & Dow, 1991], noise was added to the training set in order to improve the performance of a network in classifying noisy patterns. In [Bishop, 1995] it was shown that the part of the error induced when training with noise corresponds to a class of Tikhonov regularisers. Also, in [An, 1996], the influence of input noise, weight noise, output noise and Langevin noise was studied from theoretical and empirical points of view.

In our experiments we have used well-studied problems from the UCI Repository of Machine Learning Databases of the University of California [Murphy & Aha, 1994], as well as problems studied extensively by other researchers in an attempt to reduce as much as possible biases introduced by the size of the weights space. In all cases we have used neural networks with classic logistic activations and followed literature guidelines when setting the learning parameters of the Rprop [Riedmiller & Braun, 1993] and SARprop [Treadgold & Gedeon, 1998]. The parameters of the DAN algorithm were set to the same values for all experiments in an attempt to test the robustness of the method in different types of problems: the entropic index $q = 1.2$; the temperature $T = 0.1$; $\eta^+ = 1.2$; $\eta^- = 0.5$; for all i, $\eta_i^0 = 0.1$; $\Delta_{\max} = 50$ is used in order to prevent the neural network weights from becoming too large; $\Delta_{\min} = 10^{-6}$.

Below, we report results from 150 independent trials for four UCI problems. These 150 random weight initializations are the same for the three learning algorithms, and the training and testing sets were created according to *Proben1* [Prechelt, 1994].

The first benchmark is known as the *Fisher's Iris* problem [Murphy & Aha, 1994; Prechelt, 1994]. The data set consists of 120 examples and the test set of 30 examples. Following [Treadgold & Gedeon, 1998], an 4–2–3 FNN (4 input–2 hidden–3 output nodes; 19 weights overall) was used, and the maximum number of iterations to find a "near-optimal" weight configuration (defined as a weight set w^* that results in an error function value $E(w^*) \leq 0.01$) was set to 2000.

The second benchmark is the *breast cancer diagnosis* problem which classifies a tumor as benign or malignant based on nine features [Murphy & Aha, 1994; Prechelt, 1994]. We have used an FNN with 9–4–2–2 nodes (a total of 56 weights) as suggested in [Prechelt, 1994].

The *diabetes1* benchmark is a real-world classification task which concerns deciding if a Pima Indian individual is diabetes positive or not [Murphy & Aha, 1994; Prechelt, 1994]. There are eight features representing personal data and results from a medical examination. The Proben1 collection suggests a 8–2–2–2 FNN (30 weights overall). The termination criterion was an $E \leq 0.14$ within 2000 iterations.

Lastly, the *thyroid1* problem, [Murphy & Aha, 1994; Prechelt, 1994; Schiffmann *et al.*, 1993] was tested. We have used a 21–4–3 nodes FNN, suggested by Treadgold and Gedeon [1998], to decide whether the patient's thyroid has over function, normal function, or under function. A data set with 3600 examples was used and the target is to find a weight set that produces $E \leq 0.0036$ within a maximum of 2000 iterations.

Table 1 gives the average performance of the three algorithms in the pattern recognition problems and the dimensionality of each problem (D). It shows the average performance in terms of the number of iterations (IT) for the simulations that successfully converged to the error target and the corresponding percentage of successful runs ($CONV.$); the table does not include the iterations

Table 1. Average performance in the Iris, Cancer, Diabetes and Thyroid problems.

Algorithm	Iris $(D = 19)$		Cancer $(D = 56)$		Diabetes $(D = 30)$		Thyroid $(D = 103)$	
	IT	*CONV.*	*IT*	*CONV.*	*IT*	*CONV.*	*IT*	*CONV.*
Rprop	1340	56%	296	93%	650	86%	812	84%
SARprop	980	96%	420	97%	440	97%	818	90%
DAN	1050	100%	119	100%	272	100%	361	100%

for the unsuccessful runs (i.e. runs that exceeded the 2000 iterations limit). As shown in Table 1 the new method outperforms the others in the number of iterations required to reach a suitable solution, and converges in all cases to a minimizer that satisfies the termination conditions (i.e. 100% success of convergence to the error target within 2000 iterations).

Another set of experiments has been conducted to empirically evaluate the performance of the new method in a well-studied class of boolean function approximation problems that exhibit strong local minima [Gori & Tesi, 1992]. This class includes the various parity-N problems, which are considered as classic benchmarks [Plagianakos *et al.*, 2001; Treadgold & Gedeon, 1998; Van der Smagt, 1994]. The error target was set to $E \leq 10^{-7}$ within 2000 iterations in all cases (this is considered low enough to guarantee convergence to a "global" solution). Following the recommendations of [Treadgold & Gedeon, 1998] we have used the architectures: 3–3–1 for parity–3, 4–6–1 for parity–4, 5–7–1 for parity–5. The results are presented in Table 2.

3.2. *Nonlinear optimization*

In this section, we evaluate the performance of the PSON algorithm and compare it with the standard PSO, the Fuzzy Particle Swarm Optimizer (FPSO) [Shi & Eberhart, 2001a, 2001b], and the Hybrid Particle Swarm Optimizer with mass extinction (HPSO) [Xie *et al.*, 2002]. We have used three well-studied functions, namely the Rastrigrin, Rosenbrock and Griewank. The generalized Rastrigrin function is described by the following equation:

$$f_1(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2). \quad (16)$$

The Rosenbrock function is:

$$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10), \quad (17)$$

while the generalized Griewank function is:

$$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod 10_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} + 1\right), \quad (18)$$

In our tests we have followed the guideline of [Shi & Eberhart, 2001a] and adopted asymmetric

Table 2. Average performance in the Parity-3, Parity-4 and Parity-5 problems.

Algorithm	Parity3 $(D = 16)$		Parity4 $(D = 37)$		Parity5 $(D = 49)$	
	IT	*CONV.*	*IT*	*CONV.*	*IT*	*CONV.*
Rprop	866	79%	1345	42%	330	67%
SARprop	848	94%	1186	64%	610	98%
DAN	399	100%	960	100%	190	100%

Table 3. Average fitness values for the Rastrigrin function.

Algorithm	$m = 20$ $D = 20$	$m = 20$ $D = 30$	$m = 40$ $D = 20$	$m = 40$ $D = 30$	$m = 80$ $D = 20$	$m = 80$ $D = 30$
PSO	22.89	47.29	16.35	38.52	13.43	29.31
FPSO	23.27	48.47	15.01	35.20	10.86	22.52
HPSO	14.99	35.59	9.01	21.13	5.12	13.02
PSON	13.04	34.10	9.21	18.64	6.015	12.58

Table 4. Average fitness values for the Rosenbrock function.

Algorithm	$m = 20$ $D = 20$	$m = 20$ $D = 30$	$m = 40$ $D = 20$	$m = 40$ $D = 30$	$m = 80$ $D = 20$	$m = 80$ $D = 30$
PSO	214.67	316.45	180.97	299.71	87.28	205.56
FPSO	108.29	183.8	63.88	175.01	46	124.42
HPSO	92.50	128	58.97	82.9	38.72	64.4
PSON	110.5	180.24	64.01	106.4	34	50.1

Table 5. Average fitness values for the Griewank function.

Algorithm	$m = 20$ $D = 20$	$m = 20$ $D = 30$	$m = 40$ $D = 20$	$m = 40$ $D = 30$	$m = 80$ $D = 20$	$m = 80$ $D = 30$
PSO	0.030	0.018	0.028	0.012	0.028	0.019
FPSO	0.027	0.022	0.031	0.012	0.026	0.026
HPSO	0.025	0.016	0.022	0.013	0.023	0.010
PSON	0.021	0.042	0.021	0.034	0.022	0.010

initialization method [Angeline, 1998] for the population initialization. The initialization ranges and the $V_{\max}$ and $X_{\max}$ values are the same as suggested in [Shi & Eberhart, 2001a]. We have used swarms with $m = \{20, 40, 80\}$ particles for each function using two different dimensions $D = \{20, 30\}$; the maximum number of iterations was set to 1500 and 2000 respectively; the temperature was set to $T = 0.01$ and the choice of q was dependent on the number of particles m in the swarm, i.e. $q = \{4, 6, 8\}$, without doing any fine-tuning. The detailed results, presented in Tables 3–5, show that the new methods provide improved performance compared with the other PSO variants.

4. Conclusions

Proper adaptation of a model or function parameters is a powerful tool to improve model's performance or locate feasible solutions in practice. Adaptive stochastic search methods are eminently suitable for this task. These methods provide global exploration of the search space which may be particularly useful when the environment changes, since information contained within the population of solutions may allow efficient discovery of better-adapted solutions.

In this paper we proposed an approach that generates perturbations of the candidate points produced by the adaptive stochastic algorithm based on the q-distribution of the nonextensive statistical mechanics. These techniques were used to equip two stochastic search algorithms, a hybrid diffusion algorithm and the particle swarm optimizer, with a nonextensive schedule. The modified algorithms were tested in nonlinear functions minimization with benchmarks from the neural networks and the optimization domains. The results of our small scale study showed that this new technique compares well with exisitng approaches, improving the performance of the two methods in many cases. The use of the nonextensive schedule demands of course further exploration along several practical and theoretical dimensions. For example, questions regarding the fine-tuning of the parameters of the schedule, the analysis of the convergence characteristics and properties of the algorithms that incorporate it, and the role of the q entropic index are important issues that need further consideration in the future.

Acknowledgments

George D. Magoulas and Aristoklis Anastasiadis gratefully acknowledge support by the Engineering and Physical Sciences Research Council, UK, (GR/R92554/01-02), and Birkbeck College (University of London), respectively.

References

Ackley, D., Hinton, G. & Sejnowski, T. [1985] "A learning algorithm for Boltzmann machines," *Cogn. Sci.* **9**, 147–169.

An, G. [1996] "The effect of adding noise during backpropagation training on a generalization performance," *Neural Comput.* **8**, 643–674.

Anastasiadis, A. & Magoulas, G. D. [2004] "Nonextensive statistical mechanics for hybrid learning of neural networks," *Physica A*: *Stat. Mech. Appl.* **344**, 372–382.

Angeline, P. J. [1998] "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *Evolutionary Programming*, Lecture Notes in Computer Science (Springer), pp. 601–610.

Bishop, C. M. [1995] "Training with noise is equivalent to Tikhonov regularization," *Neural Comput.* **7**, 108–116.

Burton, R. M. & Mpitsos, G. J. [1992] "Event dependent control of noise enhances learning in neural networks," *Neural Networks* **5**, 627–637.

Clerc, M. [1999] "The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization," in *Proc. Congress of Evolutionary Computation*, eds. Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X. & Zalzala, A., Vol. 3 (IEEE Press, NY), pp. 1951–1957.

Corana, A., Marchesi, M., Martini, C. & Ridella, S. [1987] "Minimizing multimodal functions of continuous variables with the simulated annealing algorithm," *ACM Trans. Math. Soft.* **13**, 262–280.

Eberhart, R. C., Simpson, P. K. & Dobbins, R. W. [1996] *Computational Intelligence PC Tools* (Academic Press Professional, Boston, MA).

Eberhart, R. C. & Shi, Y. H. [1998] "Evolving artificial neural networks," in *Proc. Int. Conf. Neural Networks and Brain* (Publishing House of Electronics Industry, Beijing, P. R. China), PL5–PL13.

Eberhart, R. C. & Shi, Y. [2000] "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. 2000 Congress Evolutionary Computation*, San Diego, CA, pp. 84–88.

Eberhart, R. C. & Shi, Y. [2001] "Tracking and optimizing dynamic systems with particle swarms," in *Proc. IEEE Congress on Evolutionary Computation*, Seoul, Korea, pp. 94–97.

Gelfand, S. B. & Mitter, S. K. [1991] "Recursive stochastic algorithms for global optimization in $\mathbb{R}$," *SIAM J. Cont. Optim.* **29**, 999–1018.

Geman, S. & Hwang, C. R. [1986] "Diffusions for global optimization," *SIAM J. Contr. Optim.* **24**, 1031–1043.

Ghannadian, F., Alford, C. & Shonkwiler, R. [1996] "Application of random restart to genetic algorithms," *Inform. Sci.* **95**, 81–102.

Gori, M. & Tesi, A. [1992] "On the problem of local minima in backpropagation," *IEEE Trans. Patt. Anal. Mach. Intell.* **14**, 76–85.

Higashi, N. & Iba, H. [2003] "Particle swarm optimization with Gaussian mutation," in *Proc. IEEE Swarm Intelligence Symp.*, Indianapolis, IN, pp. 72–79.

Hoptroff, R. & Hall, T. [1989] "Learning by diffusion for multilayer perceptron," *Electron. Lett.* **25**, 531–533.

Hu, X. & Eberhart, R. C. [2002] "Adaptive particle swarm optimization: Detection and response to dynamic systems," in *Proc. IEEE Congress on Evolutionary Computation*, Honolulu, Hawaii, USA, pp. 1666–1670.

Jim, K., Home, B. G. & Lee Giles, C. [1995] "Effects of noise on convergence and generalization in recurrent networks," in *Advances in Neural Information Processing Systems*, eds. Tesauro, G., Touretzky, D. & Leen, T., Vol. 7 (MIT Press, Cambridge, MA), pp. 649–656.

Kennedy, J. & Eberhart, R. C. [1995] "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks* (IEEE Press Piscataway, NJ), pp. 1942–1948.

Kirkpatrick, S., Gelatt Jr., C. D. & Vecchi, M. P. [1983] "Optimization by simulated annealing," *Science* **220**, 671–680.

Krohling, R. A., Hoffmann, F. & dos Santos Coelho, L. [2004] "Co-evolutionary particle swarm optimization for min-max problems using gaussian distribution," in *IEEE Proc. Congress Evolutionary Computation* Vol. 1, Portland, Oregon, pp. 959–964.

Li, Y., Chen Tan, K. & Gong, M. [1997] "Global structure evolution and local parameter learning for control system model reductions," in *Evolutionary Algorithms in Engineering Applications*, eds. Dasgupta, D. & Michalewicz, Z. (Springer, Berlin), pp. 345–360.

Magdon-Ismail, M. & Atiya, A. F. [2000] "The early restart algorithm," *Neural Comput.* **12** 1303–1312.

Michalewicz, Z. [1996] *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, NY).

Michalewicz, Z. & Fogel, D. B. [2000] *How to Solve It*: *Modern Heuristics* (Springer, Berlin).

Murphy, P. M. & Aha, D. W. [1994] *UCI Repository of Machine Learning Databases*, University of California, Department of Information and Computer Science, Irvine, CA.

Murray, A. F. & Edwards, P. J. [1993] "Synaptic weight noise during multilayer perceptron training: Fault tolerance and training improvements," *IEEE Trans. Neural Networks* **4**, 722–725.

Murray, A. F. & Edwards, P. J. [1994] "Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training," *IEEE Trans. Neural Networks* **5**, 792–802.

Muselli, M. [1997] "A theoretical approach to restart in global optimization," *J. Global Optim.* **10**, 1–16.

Plagianakos, V. P., Magoulas, G. D. & Vrahatis, M. N. [2001] "Learning in multilayer perceptrons

using global optimization strategies," *Nonlin. Anal.: Th. Meth. Appl.* **47**, 3431–3436.

Prechelt, L. [1994] "PROBEN1–A set of benchmarks and benchmarking rules for neural network training algorithms," Technical Report 21/94, Fakultät fr Informatik, Universität Karlsruhe.

Prugel–Bennett, A. & Shapiro, J. L. [1994] "An analysis of genetic algorithms using statistical mechanics," *Phys. Rev. Lett.* **72**, 1305–1309.

Reed, R., Marks, R. J. & Oh, S. [1995] "Similarities of error regularization, sigmoid gain scaling, target smoothing and training with jitter," *IEEE Trans. Neural Networks* **6**, 529–538.

Riedmiller, M. & Braun, H. [1993] "A direct adaptive method for faster backpropagation learning: The Rprop algorithm," in *Proc. Int. Conf. Neural Networks*, San Francisco, CA, pp. 586–591.

Rögnvaldsson, T. [1994] "On Langevin updating in multilayer perceptrons," *Neural Comput.* **6**, 916–926.

Schiffmann, W., Joost, M. & Werner, R. [1993] "Comparison of optimized backpropagation algorithms," in *Proc. European Symp. Artificial Neural Networks*, Brussels, pp. 97–104.

Shi, Y. & Eberhart, R. C. [2001a] "Fuzzy adaptive particle swarm optimization," in *Proc. IEEE Congress on Evolutionary Computation*, Vol. 1, Seoul, Korea, pp. 101–106.

Shi, Y. & Eberhart, R. C. [2001b] "Particle swarm optimization with fuzzy adaptive inertia weight," in *Proc. Workshop on Particle Swarm Optimization*, Purdue School of Engineering and Technology, IUPUI, Indianapolis, IN.

Sietsma, J. & Dow, R. J. [1991] "Creating artificial neural networks that generalize," *Neural Networks* **4**, 67–79.

Szu, H. [1987] "Nonconvex optimization by fast simulated annealing," *Proc. IEEE* **75**, 1538–1540.

Tikhonov, A. N. & Arsenin, V. Y. [1977] *Solutions of Ill-Posed Problems* (W. H. Winston, Washington, DC).

Treadgold, N. K. & Gedeon, T. D. [1998] "Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm," *IEEE Tr. Neural Networks* **9**, 662–668.

Tsallis, C. [1988] "Possible generalization of Boltzmann–Gibbs statistics," *J. Stat. Phys.* **52**, 479–487.

Tsallis, C. & Stariolo, D. A. [1996] "Generalized simulated annealing," *Physica A* **233**, 395–406.

Van der Smagt, P. P. [1994] "Minimization methods for training feedforward neural networks," *Neural Networks* **7**, 1–11.

Weslstead, S. T. [1994] *Neural Network and Fuzzy Logic Applications in C/C++* (Wiley).

Xie, X., Zhang, W. & Yang, Z. [2002] "Hybrid particle swarm optimizer with mass extinction," in *Proc. Int. Conf. Communication, Circuits and Systems*, Chengdu, China, pp. 1170–1173.