Mach Learn (2007) 68: 201–233
DOI 10.1007/s10994-007-5017-7

# Annealing stochastic approximation Monte Carlo algorithm for neural network training

**Faming Liang**

Received: 25 June 2005 / Revised: 17 June 2007 / Accepted: 22 June 2007 / Published online: 3 August 2007
Springer Science+Business Media, LLC 2007

**Abstract** We propose a general-purpose stochastic optimization algorithm, the so-called annealing stochastic approximation Monte Carlo (ASAMC) algorithm, for neural network training. ASAMC can be regarded as a space annealing version of the stochastic approximation Monte Carlo (SAMC) algorithm. Under mild conditions, we show that ASAMC can converge weakly at a rate of $\Omega(1/\sqrt{t})$ toward a neighboring set (in the space of energy) of the global minimizers. ASAMC is compared with simulated annealing, SAMC, and the BFGS algorithm for training MLPs on a number of examples. The numerical results indicate that ASAMC outperforms the other algorithms in both training and test errors. Like other stochastic algorithms, ASAMC requires longer training time than do the gradient-based algorithms. It provides, however, an efficient approach to train MLPs for which the energy landscape is rugged.

**Keywords** Back-propagation · Convergence rate · Markov chain Monte Carlo · Multiple layer perceptron · Simulated annealing · Stochastic approximation · Wang–Landau algorithm

## 1 Introduction

Over the past several decades, feed-forward neural networks, otherwise known as multiple-layer perceptrons (MLPs), have achieved increased popularity among scientists, engineers, and other professionals as tools for knowledge representation. Given a group of connection weights $\boldsymbol{x} = (\alpha, \beta, \gamma)$, the MLP approximator can be written as

$$\hat{f}(z_k|\boldsymbol{x}) = \varphi_o\left(\alpha_0 + \sum_{j=1}^{p} \gamma_j z_{kj} + \sum_{i=1}^{M} \alpha_i \varphi_h\left(\beta_{i0} + \sum_{j=1}^{p} \beta_{ij} z_{kj}\right)\right), \qquad (1)$$

Action Editor: Risto Miikkulainen.

F. Liang (✉)
Department of Statistics, Texas A&M University, College Station, TX 77843-3143, USA
e-mail: fliang@stat.tamu.edu

where $M$ is the number of hidden units, $p$ is the number of input units, $z_k = (z_{k1}, \ldots, z_{kp})$ is the $k$th input pattern, and $\alpha_i$, $\beta_{ij}$ and $\gamma_j$ are the weights on the connections from the $i$th hidden unit to the output unit, from the $j$th input unit to the $i$th hidden unit, and from the $j$th input unit to the output unit, respectively. The connections from input units to the output unit are also called the shortcut connections. Note that the shortcut connections may not exist in some MLPs. In (1), the bias unit is treated as a special input unit with a constant input, say, 1. The functions $\varphi_h(\cdot)$ and $\varphi_o(\cdot)$ are called the activation functions of the hidden units and the output unit, respectively. Popular choices of $\varphi_h(\cdot)$ include the sigmoid function and the hyperbolic tangent function. The former is defined as $\varphi_h(z) = 1/(1+e^{-z})$ and the latter $\varphi_h(z) = \tanh(z)$. The choice of $\varphi_o(\cdot)$ is problem dependent. For regression problems, $\varphi_o(\cdot)$ is usually set to the linear function $\varphi_o(z) = z$; and for classification problems, $\varphi_o(\cdot)$ is usually set to the sigmoid function. The problem of MLP training is to minimize the following objective function

$$U(\boldsymbol{x}) = \sum_{k=1}^{N} (\hat{f}(z_k|\alpha, \beta) - y_k)^2 + \lambda \left[ \sum_{i=0}^{M} \alpha_i^2 + \sum_{i=1}^{M} \sum_{j=0}^{p} \beta_{ij}^2 + \sum_{j=1}^{p} \gamma_j^2 \right], \tag{2}$$

by choosing appropriate connection weights, where $y_k$ denotes the target output corresponding to the input pattern $z_k$, the second term is the regularization term, and $\lambda$ is the regularization parameter. The regularization term is often chosen as the sum of squares of the connection weights, which stabilizes the generalization performance of the MLP. Henceforth, $U(\boldsymbol{x})$ will be called the energy function of the MLP in terms of physics.

As known by many researchers, the energy landscape of the MLP is often rugged. The gradient-based training algorithms, such as back-propagation (Rumelhart et al. 1986), conjugate gradient, Newton's method, the DFP algorithm (Davidon 1959; Fletcher and Powell 1963), the Levenberg–Marquardt algorithm (Levenberg 1944; Marquardt 1963), and the BFGS algorithm (Broyden 1970a; Broyden 1970b; Fletcher 1970; Goldfarb 1970; Shanno 1970), tend to converge to a local energy minimum near the starting point. Consequently, the information contained in the training data may not be learned sufficiently. To reduce the chance of converging to local energy minima, a number of variants of these algorithms have been proposed based on the idea of perturbation. For example, von Lehmen et al. (1988) and Abunawass and Owen (1993) proposed to add noise to the connection weights during the training process; Hanson (1991) proposed to replace the connection weights by random variables drawn from a distribution centered at their current values; and Tang et al. (2003) proposed to modify the activation function when the process was trapped in a local energy minimum. Although these algorithms work well for some examples, they are heuristic. It is hard, if not impossible, to establish their convergence to the global energy minima. In practice, the effects of these perturbations are usually limited, which only delay the learning process converging to local energy minima a reasonable number of iterations (Ingman and Merlis 1991; Wang and Principe 1999).

To avoid the local-trap problem, simulated annealing (SA; Kirkpatrick et al. 1983) has been employed by some authors to train neural networks. Amato et al. (1991) and Owen and Abunawass (1993) showed that for complex learning tasks, SA has a better chance to converge to a global energy minimum than the gradient-based algorithms. Let $T_1 > T_2 > \cdots > T_k > \cdots$ be a sequence of monotonically decreasing temperatures, where $T_1$ is reasonably large and $\lim_{k\to\infty} T_k = 0$. SA works in the following procedure.

(a) Initialize at the temperature level $T_1$ and an arbitrary configuration $\boldsymbol{x}_0$.

(b) At each level $T_k$, run $N_k$ iterations of an MCMC sampler, e.g., the Metropolis–Hastings (MH) algorithm (Metropolis et al. 1953; Hastings 1970), with the target distribution

$$p_{T_k}(\boldsymbol{x}) = \frac{1}{Z_{T_k}} \exp\{-U(\boldsymbol{x})/T_k\}, \tag{3}$$

where $Z_{T_k}$ is the normalizing constant. Pass the last sample to the next iteration as the starting configuration.

(c) Increase $k$ to $k+1$.

It has been shown by Geman and Geman (1984) that the global minimum of $U(\boldsymbol{x})$ can be reached by SA with probability 1 if temperature decreases sufficiently slowly such that $T_k > c/\log(\sum_{i=1}^{k} N_i)$ for some constant $c$. This translates to the logarithmic convergence rate $\Omega(1/\log(t))$ in numbers of iterations. In this paper, we use $t$ to denote the number of iterations. In practice, however, no one can afford to have such a slow cooling schedule. Most frequently, people use a linearly or geometrically decreasing cooling schedule, which can no longer guarantee the global energy minimum to be reached. Holley et al. (1989) showed that no cooling schedule faster than the logarithmic rate can be guaranteed to find the global energy minimum.

Other stochastic algorithms that have been used in neural network training include the genetic algorithm (Holland 1975; Goldberg 1989) and MCMC algorithms. Although the genetic algorithm works well for some problems (see van Rooij et al. 1996 for examples), there is no theory to support its convergence to global minima. The MCMC algorithms are mainly used for Bayesian neural networks (MacKay 1992a; Neal 1996; Müller and Insua 1998; de Freitas et al. 2000; Liang and Wong 2001; Liang 2003, 2005a), which are not the focus of this paper. The key difference between MCMC and SA is on their target distributions. MCMC attempts to draw samples from the posterior distribution of the MLP, while SA attempts to draw samples from a uniform distribution defined on the set $\{\boldsymbol{x} : U(\boldsymbol{x}) = u_{\min}\}$, where $u_{\min}$ denotes the global minimum value of $U(\boldsymbol{x})$. Since SA works by simulating from a sequence of distributions scaled with different temperatures, some authors, e.g., Jerrum and Sinclair (1997), also regarded it as MCMC with a varying temperature.

In this paper, we provide a new Monte Carlo optimization algorithm, the so-called annealing stochastic approximation Monte Carlo (ASAMC) algorithm, for MLP training. ASAMC can be regarded as a space annealing version of the stochastic approximation Monte Carlo (SAMC) algorithm (Liang et al. 2007). Let $E_* = \{\boldsymbol{x} : U(\boldsymbol{x}) < u_{\min} + \Delta\}$ denote a neighbor set (in the space of energy) of the global minimizers, where $\Delta$ is any positive number. Under mild conditions, we show that ASAMC can converge weakly toward the set $E_*$, and the convergence can be achieved at a rate of $\Omega(1/\sqrt{t})$. The latter is a new contribution in this paper. ASAMC is compared with SA, SAMC, and BFGS algorithms for training MLPs on a number of examples. Our numerical results indicate that ASAMC outperforms the other algorithms in both training and test errors. This makes the stochastic algorithms a viable alternative to the gradient-based algorithms for training MLPs for which the energy landscape is rugged.

The remainder of this paper is organized as follows. In Sect. 2, we describe the ASAMC algorithm. In Sect. 3, we illustrate ASAMC using two examples. In Sect. 4, we compare ASAMC, SAMC, SA and BFGS on two benchmark and three real-world examples. In Sect. 5, we discuss the paper briefly and point out some possible directions for future research. In Sect. 6, we conclude the paper.

## 2 Annealing stochastic approximation Monte Carlo algorithm

### 2.1 Stochastic approximation Monte Carlo algorithm

Before describing the ASAMC algorithm, we first give a brief description of SAMC. Suppose that we are working with the following Boltzmann distribution,

$$p(\boldsymbol{x}) = \frac{1}{Z}\exp\{-U(\boldsymbol{x})/\tau\}, \quad \boldsymbol{x} \in \mathcal{X}, \tag{4}$$

where $Z$ is the normalizing constant, $\tau$ is the temperature, and $\mathcal{X}$ is the sample space. Without loss of generality, we assume that $\mathcal{X}$ is compact. For MLPs, $\mathcal{X}$ can be set to the region $[-B_{\mathcal{X}}, B_{\mathcal{X}}]^{\dim(\boldsymbol{x})}$, where $B_{\mathcal{X}}$ is chosen such that the region includes at least a global minimum of $U(\boldsymbol{x})$; $\mathcal{X}$ can also be set to the region $\{\boldsymbol{x} : U(\boldsymbol{x}) \le u_{\max}\}$, where $u_{\max}$ is sufficiently large such that the region $\{\boldsymbol{x} : U(\boldsymbol{x}) > u_{\max}\}$ is not of interest at all. Furthermore, we assume that the sample space can be partitioned according to the energy function into $m$ disjoint subregions denoted by $E_1 = \{\boldsymbol{x} : U(\boldsymbol{x}) \le u_1\}$, $E_2 = \{\boldsymbol{x} : u_1 < U(\boldsymbol{x}) \le u_2\}$, ..., $E_{m-1} = \{\boldsymbol{x} : u_{m-2} < U(\boldsymbol{x}) \le u_{m-1}\}$, and $E_m = \{\boldsymbol{x} : U(\boldsymbol{x}) > u_{m-1}\}$, where $u_1, \dots, u_{m-1}$ are real numbers specified by the user. Let $\psi(\boldsymbol{x})$ be a non-negative function defined on the sample space with $0 < \int_{\mathcal{X}} \psi(\boldsymbol{x}) d\boldsymbol{x} < \infty$, and $g_i = \int_{E_i} \psi(\boldsymbol{x}) dx$. In practice, we often set $\psi(\boldsymbol{x}) = \exp\{-U(\boldsymbol{x})/\tau\}$.

SAMC seeks to draw samples from each of the subregions with a pre-specified frequency. If this goal can be achieved, then the local-trap problem can be avoided successfully. Let $\boldsymbol{x}_{t+1}$ denote a sample drawn from a MH kernel $K_{\theta_t}(\boldsymbol{x}_t, \cdot)$ with the proposal distribution[1] $q(\boldsymbol{x}_t, \cdot)$ and the stationary distribution[2]

$$p_{\theta_t}(\boldsymbol{x}) \propto \sum_{i=1}^{m} \frac{\psi(\boldsymbol{x})}{e^{\theta_{ti}}} I(\boldsymbol{x} \in E_i), \tag{5}$$

where $\theta_t = (\theta_{t1}, \dots, \theta_{tm})$ is an $m$-vector in a space $\Theta$. For simplicity, we assume that $\Theta$ is compact and set $\Theta = [-B_\Theta, B_\Theta]^m$ with $B_\Theta = 10^{20}$ in this article, although as a practical matter this is equivalent to setting $\Theta = \mathbb{R}^m$. Since adding to or subtracting from $\theta_t$ a constant will not change $p_{\theta_t}(\boldsymbol{x})$, $\theta_t$ can be kept in the compact set in simulations by adjusting with an additive constant. Let the proposal distribution satisfy the minorization condition, i.e.,

$$\omega^* = \sup_{\theta\in\Theta} \sup_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{X}} \frac{p_\theta(\boldsymbol{y})}{q(\boldsymbol{x}, \boldsymbol{y})} < \infty \tag{6}$$

which is a natural condition in study of MCMC theory (Mengersen and Tweedie 1996). In practice, this kind of proposals can be easily designed. Since $\mathcal{X}$ is compact, a sufficient

[1] The proposal distribution of a Markov chain refers to a distribution which defines how a new state is generated conditioned on the current one.

[2] The stationary distribution, also known as the invariant distribution, of a Markov chain refers to a probability measure $p(\boldsymbol{x})$ such that

$$p(B) = \int_{\mathcal{X}} K(\boldsymbol{x}, B) p(d\boldsymbol{x}), \quad \forall B \in \mathcal{B}(\mathcal{X}),$$

where $K(\boldsymbol{x}, B)$ denotes the transition kernel of the Markov chain and $\mathcal{B}(\mathcal{X})$ denotes the Borel set of the space $\mathcal{X}$. Refer to Karlin and Taylor (1998) for more details.

design for the minorization condition is to set $q(\boldsymbol{x}, \boldsymbol{y})$ to a global proposal distribution. A proposal distribution is called global if $q(\boldsymbol{x}, \boldsymbol{y}) > 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$. For MLPs, $q(\boldsymbol{x}, \boldsymbol{y})$ can be chosen as a random walk Gaussian proposal $\boldsymbol{y} \sim N(\boldsymbol{x}, \sigma^2)$ with $\sigma^2$ being calibrated to have a desired acceptance rate. As discussed later, restricting the proposal distribution to be global ensures the convergence of ASAMC to the global energy minima.

Let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)$ be an $m$-vector with $0 < \pi_i < 1$ and $\sum_{i=1}^m \pi_i = 1$, which defines a desired sampling frequency for the subregions. Henceforth, $\boldsymbol{\pi}$ will be called the desired sampling distribution. Define $H(\theta_t, \boldsymbol{x}_{t+1}) = (\boldsymbol{e}_{t+1} - \boldsymbol{\pi})$, where $\boldsymbol{e}_{t+1} = (e_{t+1,1}, \ldots, e_{t+1,m})$ and $e_{t+1,i} = 1$ if $\boldsymbol{x}_{t+1} \in E_i$ and 0 otherwise. Let $\{\gamma_t\}$ be a positive non-decreasing sequence satisfying the conditions,

$$\text{(i)} \quad \sum_{t=0}^{\infty} \gamma_t = \infty, \qquad \text{(ii)} \quad \sum_{t=0}^{\infty} \gamma_t^{\delta} < \infty, \tag{7}$$

for some $\delta \in (1, 2)$. In this paper, we set

$$\gamma_t = \left[\frac{t_0}{\max(t_0, t)}\right]^{\eta}, \quad t = 0, 1, 2, \ldots \tag{8}$$

for some specified values of $t_0 > 1$ and $\eta \in (\frac{1}{2}, 1]$. A large value of $t_0$ will allow the sampler to reach all subregions very quickly even for a large system. Let $J(\boldsymbol{x})$ denote the index of the subregion the sample $\boldsymbol{x}$ belongs to. With above notations, one iteration of SAMC can be described as follows. Refer to Appendix 1 for the pseudocode of the algorithm.

*SAMC algorithm:*

(i) Generate $\boldsymbol{x}_{t+1} \sim K_{\theta_t}(\boldsymbol{x}_t, \cdot)$ with a single Metropolis–Hastings simulation step:
  (i.1) Generate $\boldsymbol{y}$ according to the proposal distribution $q(\boldsymbol{x}_t, \boldsymbol{y})$.
  (i.2) Calculate the ratio

$$r = e^{(\theta_{tJ(\boldsymbol{x}_t)} - \theta_{tJ(\boldsymbol{y})})} \frac{\psi(\boldsymbol{y})}{\psi(\boldsymbol{x}_t)} \frac{q(\boldsymbol{y}, \boldsymbol{x}_t)}{q(\boldsymbol{x}_t, \boldsymbol{y})}.$$

  (i.3) Accept the proposal with probability $\min(1, r)$. If it is accepted, set $\boldsymbol{x}_{t+1} = \boldsymbol{y}$; otherwise, set $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t$.

(ii) Set $\theta^* = \theta_t + \gamma_t H(\theta_t, \boldsymbol{x}_{t+1})$, where $\gamma_t$ is called the gain factor.

(iii) If $\theta^* \in \Theta$, set $\theta_{t+1} = \theta^*$; otherwise, set $\theta_{t+1} = \theta^* + \boldsymbol{c}^*$, where $\boldsymbol{c}^* = (c^*, \ldots, c^*)$ and $c^*$ is chosen such that $\theta^* + \boldsymbol{c}^* \in \Theta$.

Recall that we have set $\Theta = [-B_\Theta, B_\Theta]^m$ with $B_\Theta$ being a huge number, so it is reasonable to assume that $\max_{i=1}^m \theta_i^* - \min_{i=1}^m \theta_i^* \ll B_\Theta$ holds at each iteration. Thus, in step (iii), we can set $c^* = B_\Theta/2 - \max_{i=1}^m \theta_i^*$ if $\max_{i=1}^m \theta_i^* > B_\Theta$ and $c^* = -B_\Theta/2 - \min_{i=1}^m \theta_i^*$ if $\min_{i=1}^m \theta_i^* < -B_\Theta$.

A remarkable feature of SAMC is its self-adjusting mechanism. If a proposal is rejected, the weight of the subregion that the current sample belongs to will be adjusted to a larger value, and thus the proposal of jumping out from the current subregion will be less likely rejected in the next iteration. This mechanism effectively prevents the system from getting trapped in local minima. This is very important for the energy functions with multiple local minima.

SAMC falls into the category of stochastic approximation algorithms (Benveniste et al. 1990; Andrieu et al. 2005). The convergence of this algorithm can be extended from a theorem presented in Liang et al. (2007) under a little different conditions. Refer to Theorem 8.2

for the details. Under mild conditions, we have

$$\theta_{ti} \to \begin{cases} C + \log(\int_{E_i} \psi(\boldsymbol{x})d\boldsymbol{x}) - \log(\pi_i + \aleph), & \text{if } E_i \neq \emptyset, \\ -\infty. & \text{if } E_i = \emptyset, \end{cases} \tag{9}$$

as $t \to \infty$, where $C$ is an arbitrary constant, $\aleph = \sum_{j\in\{i:E_i=\emptyset\}} \pi_j/(m - m_0)$, and $m_0 = \#\{i : E_i = \emptyset\}$ is the number of empty subregions. A subregion $E_i$ is called empty if $\int_{E_i} \psi(\boldsymbol{x})d\boldsymbol{x} = 0$. In SAMC, the sample space partition can be made blindly by simply specifying some parameter values as described in the pseudocode of the algorithm. This may lead to some empty subregions. The constant $C$ can be determined by imposing a constraint on $\theta_t$, say, $\sum_{i=1}^{m} e^{\theta_{ti}}$ is equal to a known number.

Let $\hat{\pi}_{ti} = P(\boldsymbol{x}_t \in E_i)$ be the probability of sampling from the subregion $E_i$ at iteration $t$. Equation (9) implies that as $t \to \infty$, $\hat{\pi}_{ti}$ will converge to $\pi_i + \aleph$ if $E_i \neq \emptyset$ and 0 otherwise. With an appropriate specification of $\boldsymbol{\pi}$ (refer to (12) for examples), sampling can be biased to the low energy subregions to increase the chance of sampling from $E_*$. Theorem 8.4 concerns the convergence rate of SAMC. Under mild conditions, $\hat{\boldsymbol{\pi}}_t$ can decrease as $1/\sqrt{t}$, where $t$ represents the number of iterations. In this sense, we claim that SAMC can converge at a rate of $\Omega(1/\sqrt{t})$.

The subject of stochastic approximation was founded by Robbins and Monro (1951). After five decades of continual development, it has developed into an important area in systems control and optimization. Many of the neural network training algorithms, such as the simultaneous perturbation stochastic approximation algorithm (Spall 1992; Wouwer et al. 1999), the Widrow–Hoff algorithm (also known as the "least mean square" algorithm) (Haykin 1999, pp. 128–135), the Alopex algorithm (Harth and Tzanakou 1974; Sastry et al. 2002) and self-organizing maps (Kohonen 1990; Mulier and Cherkassky 1995; Flanagan 1997), can be regarded as special instances of stochastic approximation. Refer to Bharath and Borkar (1999) for more discussions on this issue. Recently, stochastic approximation has been used with Markov chain Monte Carlo for solving maximum likelihood estimation problems (Gu and Kong 1998; Gelfand and Banerjee 1998; Delyon et al. 1999; Gu and Zhu 2001). The critical difference between SAMC and other stochastic approximation algorithms is at sample space partitioning. Sample space partitioning improves the performance of stochastic approximation in optimization. It forces each non-empty subregion to be visited with a fixed frequency, and thus increases the chance to locate the global energy minimizer.

A closely related algorithm to SAMC is the contour Monte Carlo (CMC) algorithm, which is also known as the Generalized Wang–Landau algorithm (Wang and Landau 2001; Liang 2005a, 2005b). In CMC, the simulation consists of a number of stages, and each stage associates with a fixed gain factor. If the gain factor and the number of iterations of each stage are chosen appropriately such that the condition (7) is satisfied, then CMC becomes a special instance of SAMC.

At last, we would like to mention that the SAMC algorithm described above is slightly different from the one presented in Liang et al. (2007). In the latter, the proposal distribution $q(\boldsymbol{x}, \boldsymbol{y})$ is required to satisfy the condition: for every $\boldsymbol{x} \in \mathcal{X}$, there exist $\epsilon_1 > 0$ and $\epsilon_2 > 0$ such that

$$|\boldsymbol{x} - \boldsymbol{y}| \leq \epsilon_1 \Longrightarrow q(\boldsymbol{x}, \boldsymbol{y}) \geq \epsilon_2. \tag{10}$$

To contrast with the global proposal distribution, the proposal distribution satisfying condition (10) is called the local proposal distribution. It is easy to see that if $\mathcal{X}$ contains several separated regions and a local proposal distribution is used, the MCMC simulation performed in step (i) of the SAMC algorithm will not leave the density $p_{\theta_t}(\boldsymbol{x})$ invariant. As discussed in Sect. 2.2, the sample space of ASAMC may contain several separated regions. To ensure the convergence of ASAMC, we restrict the proposal distribution to be global in simulations.

## 2.2 Annealing stochastic approximation Monte Carlo algorithm

In theory, SAMC is able to find the global energy minima if the run is long enough. However, due to the broadness of the sample space, the process may be slow even when sampling has been biased to low energy subregions. To accelerate the search process, we shrink the sample space over iterations. As argued below, this modification preserves the theoretical property of SAMC when a global proposal distribution is used.

Suppose that the subregions $E_1, \ldots, E_m$ have been arranged in ascending order by energy; that is, if $i < j$, then $U(\boldsymbol{x}) < U(\boldsymbol{y})$ for any $\boldsymbol{x} \in E_i$ and $\boldsymbol{y} \in E_j$. Let $\varpi(u)$ denote the index of the subregion that a sample $\boldsymbol{x}$ with energy $u$ belongs to. For example, if $\boldsymbol{x} \in E_j$, then $\varpi(U(\boldsymbol{x})) = j$. Let $\mathcal{X}^{(t)}$ denote the sample space at iteration $t$. Annealing SAMC, which will be abbreviated as ASAMC hereafter, starts with $\mathcal{X}^{(1)} = \bigcup_{i=1}^{m} E_i$, and then iteratively sets the sample space as

$$\mathcal{X}^{(t)} = \bigcup_{i=1}^{\varpi(U_{\min}^{(t)}+\Delta)} E_i, \tag{11}$$

where $U_{\min}^{(t)}$ is the minimum energy value obtained by iteration $t$, and $\Delta > 0$ is a user specified parameter. The sample space $\mathcal{X}^{(t)}$ shrinks iteration by iteration. In this sense, the modified algorithm is called annealing SAMC.

Since the proposal distribution is global, Theorems 8.2–8.4 hold for ASAMC on the limiting space $\mathcal{X}^{(\infty)} = \lim_{t\to\infty} \mathcal{X}^{(t)}$, although $\mathcal{X}^{(\infty)}$ may contain some separated regions. The existence of $\mathcal{X}^{(\infty)}$ is true due to the monotonicity of the sequence $\mathcal{X}^{(1)} \supseteq \mathcal{X}^{(2)} \supseteq \cdots \supseteq \mathcal{X}^{(t)} \supseteq \cdots$.

For an effective implementation of ASAMC, several issues need to be considered.

- Sample space partitioning. For training MLPs, the sample space is usually partitioned according to the energy function. The maximum energy difference in each subregion should be bounded by a reasonable number, say, $2\tau$, which ensures that the MH moves within the same subregion have a reasonable acceptance rate. Note that within the same subregion, sampling from (5) is reduced to sampling from $\psi(\boldsymbol{x})$.
- Choice of $\Delta$. The performance of ASAMC depends on the value of $\Delta$ to some extent. If $\Delta$ is too large, ASAMC may take a long time to locate the global minimum due to the broadness of the sample space. If $\Delta$ is too small, ASAMC may also take a long time to locate the global minimum. In this case, the sample space may contain only a few separated regions, and the most proposed transitions will be rejected. It is generally believed that allowing a sampler to jump to intermediate states of higher energy will increase the success probability of transiting between local minima. In our experience, a value of $\Delta$ between 5 and 10 works well for most MLP training problems. To compensate for the negative effect of sample space restriction, the proposal distribution used in ASAMC should be more spread than that used in SAMC.
- Choice of the desired sampling distribution $\boldsymbol{\pi}$. Since $\boldsymbol{\pi}$ controls the sampling frequency of each subregion, intuitively, one may choose it to bias sampling to the low energy subregions to increase the chance of finding the global minima. Our experience shows that

a fine tuning of $\boldsymbol{\pi}$ is not necessary, because in ASAMC sampling has been restricted to low energy subregions progressively. In practice, we can generally set

$$\pi_i \propto \frac{1}{i^\iota}, \quad i = 1, \ldots, m, \tag{12}$$

for $\iota \geq 0$. For ASAMC, we set $\iota = 0$ in this article. Whilst for SAMC, we tried different values of $\iota$ for different examples.

• Choice of $N$, $\eta$, and $t_0$, where $N$ denotes the total number of iterations. In practice, one often fixes the value of $\eta$ and varies the value of $t_0$ with problems. For example, $\eta$ was fixed to 0.6 in this paper. The more complex the problem is, the larger value of $t_0$ one should choose. A large value of $t_0$ will force the sampler to reach all subregions quickly, even in the presence of multiple local energy minima. The appropriateness of the choices of $t_0$, $\eta$ and $N$ can be diagnosed by examining the convergence of the run. In ASAMC, the desired sampling distribution has been set to uniform, so the convergence of the run can be diagnosed by examining the equality of the realized sampling frequencies on the subregions included in the limiting sample space $\mathcal{X}^{(\infty)}$. As suggested by Wang and Landau (2001), a run can be regarded as converged if the sampling frequency of each subregion is not less than 80% of the average sampling frequency; that is,

$$\epsilon_f = \min\left\{\frac{f_i}{\bar{f}} : i = 1, \ldots, \varpi(U_{\min}^{(\infty)} + \Delta), E_i \neq \emptyset\right\} \geq 80\%, \tag{13}$$

where $f_i$ denotes the realized sampling frequency of the subregion $E_i$, $\bar{f}$ is the average sampling frequency of the subregions included in $\mathcal{X}^{(\infty)}$, and $U_{\min}^{(\infty)}$ denotes the minimum energy value found in the run. If a run is diagnosed as unconverged, ASAMC should be re-run with a large value of $N$, a larger value of $t_0$, or a smaller value of $\eta$. For example, the following scheme was adopted in this paper to update their values: set $N \leftarrow 2N$ and $t_0 \leftarrow 1.5t_0$ and keep $\eta$ unchanged.

The diagnostic is only needed for the case that the global minimum value is unknown, e.g., training a MLP with a regularization term. For the case that the global minimum value is known, training can stop once a global minimizer was located, and the convergence diagnostic is not needed.

## 3 Two illustrative examples

### 3.1 The knapsack problem

To illustrate Theorem 8.2, we consider the following problem: given $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{R}^n$ and $\boldsymbol{b} = (b_0, b_1, \ldots, b_m) \in \mathbb{R}^m$, estimate the numbers $n_i = \#\{\boldsymbol{x} : b_{i-1} < \boldsymbol{a}'\boldsymbol{x} \leq b_i\}$ for $i = 1, \ldots, m$, where $\boldsymbol{x}$ is a (0,1)-vector, $\boldsymbol{a}'$ denotes the transpose of $\boldsymbol{a}$, and $b_0$ is a number less than 0. If the vector $\boldsymbol{a}$ gives the sizes of $n$ items to be packed into a knapsack of capacity $b_i$, then $\sum_{j=1}^{i} n_j$ is the total number of combinations of items that can fit into the knapsack. As known by many researchers, the knapsack problem is an NP-hard problem; that is, there is probably no algorithm which can solve it exactly in polynomial time.

In our example, $n = 10$, $\boldsymbol{a} = (0.6129, 0.1735, 0.5868, 0.2163, 0.3486, 0.1233, 0.6224, 0.8658, 0.8564, 0.1756)$, and $\boldsymbol{b} = (-1, 0, 1, 2, 3, 4, 5)$. The exact values of $(n_1, \ldots, n_m)$ are shown in Table 1, which are obtained by an exhaustive evaluation. SAMC was applied to this example. The sample space was partitioned into 7 subregions according to $\boldsymbol{b}$: $E_1 =$

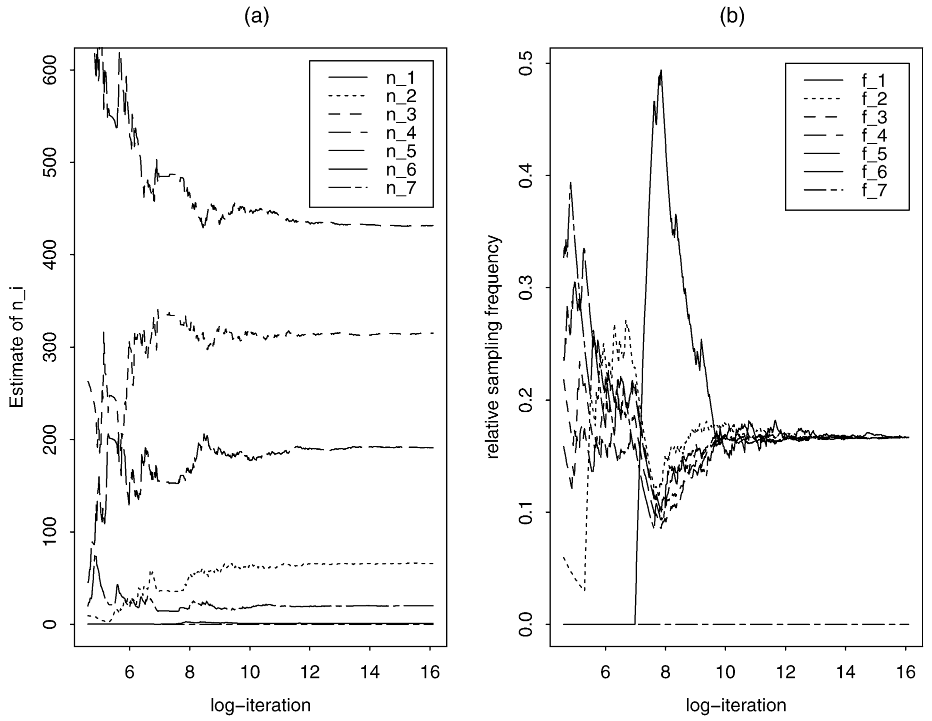

**Fig. 1** Computational results of ASAMC for the knapsack example. Plot (**a**) shows the evolution of $\hat{n}_{t,i}$ in a run; plot (**b**) shows the evolution of the relative sampling frequencies in a run. This figure illustrates the estimation and convergence of ASAMC

$\{\boldsymbol{x} : -1 < \boldsymbol{a}'\boldsymbol{x} \le 0\}, \ldots, E_6 = \{\boldsymbol{x} : 4 < \boldsymbol{a}'\boldsymbol{x} \le 5\}, E_7 = \{\boldsymbol{x} : \boldsymbol{a}'\boldsymbol{x} > 5\}$. We note that $E_7$ is empty, because the total size of the 10 items is less than 5. In simulations, we set $\psi(\boldsymbol{x}) = 1$, $t_0 = 10$, and $\iota = 0$. Let $\boldsymbol{x}_t = (x_{t,1}, \ldots, x_{t,n})$ denote the current solution. A new solution can be proposed as follows.

(a) Set $\boldsymbol{y} \leftarrow \boldsymbol{x}_t$, and select a number $k$ at random from the set $\{1, \ldots, 5\}$.
(b) Repeat the following two steps $k$ times:

- Select a number $j$ at random from the set $\{1, \ldots, n\}$.
- "Flip" $y_j$ by setting $\boldsymbol{y} = (y_1, \ldots, y_{j-1}, 1 - y_j, y_{j+1}, \ldots, y_n)$.

(c) Output $\boldsymbol{y}$ as the proposed solution.

It follows from Theorem 8.2 that as $t$ becomes large, $\theta_t \to C + (\log n_1, \ldots, \log n_7)$, where $C$ can be determined by imposing the constraint $\sum_{i=1}^{7} \exp\{\theta_{t,i} - C\} = 2^{10}$ on $\theta_t$. Given $\theta_t$ and $C$, $n_i$ can then be estimated by $\hat{n}_{t,i} = \exp\{\theta_{t,i} - C\}$. The algorithm was run 10 times independently. Each run consisted of $10^7$ iterations. Figure 1 summarizes graphically the results obtained in one run, where plot (a) shows the evolution of $\hat{n}_{t,i}$, and plot (b) shows the evolution of the relative sampling frequencies, $f_1, \ldots, f_7$, which correspond to the subregions $E_1, \ldots, E_7$, respectively. The approximate equality of the frequencies $f_1, \ldots, f_6$, for which the corresponding subregions are nonempty, indicates the convergence of the run. Figure 1 also shows the correspondence between the convergence of $\theta_t$ and the convergence of $f_i$'s, both occurring after about 20 000 ($\approx e^{10}$) iterations. Table 1 summarizes the results

**Table 1** Computational results of SAMC for the knapsack example. Notations: $\hat{n}_i$ denotes the estimate of $n_i$, $f_i$ denotes the realized relative sampling frequency of the subregion $E_i$, and SD denotes the standard error of the corresponding estimates. The values of $\hat{n}_i$ and $f_i$ are obtained by averaging over 10 runs

| Subregion | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ | $E_7$ |
|---|---|---|---|---|---|---|---|
| $n_i$ | 1 | 66 | 315 | 431 | 191 | 20 | 0 |
| $\hat{n}_i$ | 0.983 | 65.908 | 315.042 | 431.460 | 190.728 | 19.879 | 0 |
| SD | 0.006 | 0.055 | 0.118 | 0.102 | 0.070 | 0.034 | 0 |
| $f_i$ | 0.1665 | 0.1667 | 0.1667 | 0.1667 | 0.1667 | 0.1668 | 0 |
| SD | 0.0003 | 0.0001 | 0.0001 | 0.0001 | 0.0001 | 0.0001 | 0 |

obtained in 10 runs. From the figure and table, it can be seen that as the number of iterations becomes large, $\exp\{\theta_{t,i} - C\}$ converges to $n_i$, and $f_i$ converges to $\pi_i + \aleph$ if $E_i$ is non-empty. Since $E_7$ is empty, theoretically we should have $f_7 \equiv 0$ and $e^{\theta_{t,7}} \to 0$ as $t \to \infty$. Since our aim for this example is to estimate the cardinality function, the sample space can not be shrinked with iterations. In this case, ASAMC is reduced to SAMC.

## 3.2 A multiple local minima problem

To illustrate ASAMC, we consider minimizing the following function on $[-1.1, 1.1]^2$:

$$\begin{aligned} U(\boldsymbol{x}) = &-(x_1 \sin(20x_2) + x_2 \sin(20x_1))^2 \cosh(\sin(10x_1)x_1) \\ &- (x_1 \cos(10x_2) - x_2 \sin(10x_1))^2 \cosh(\cos(20x_2)x_2), \end{aligned}$$

whose global minimum is $-8.12465$ attained at $(x_1, x_2) = (-1.0445, -1.0084)$ and $(1.0445, -1.0084)$. This example is identical to Example 6.1 of Liang (2005a). Figure 2 shows that $U(\boldsymbol{x})$ has a multitude of local minima separated by high-energy barriers.

ASAMC was first applied to this example. The sample space was partitioned into 41 subregions with an equal energy bandwidth: $E_1 = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) \le -8.0\}$, $E_2 = \{(\boldsymbol{x}) \in \mathcal{X} : -8.0 < U(\boldsymbol{x}) \le -7.8\}$, …, and $E_{41} = \{(\boldsymbol{x}) \in \mathcal{X} : -0.2 < U(\boldsymbol{x}) \le 0\}$. In simulations, we set $\psi(\boldsymbol{x}) = \exp(-U(\boldsymbol{x})/0.1)$, $t_0 = 100$, $\Delta = 6$, $\iota = 0$, and the proposal distribution to be $N_2(\boldsymbol{x}, 0.3^2 I_2)$. In this paper, $I_d$ denotes an identity matrix of size $d$ by $d$. The algorithm was run for 50 000 iterations, and 500 samples were collected at equally spaced time intervals.

For comparison, SAMC and SA were also applied to this example. SAMC was run for 50 000 iterations with the same setting as ASAMC (the parameter $\Delta$ does not exist in SAMC), and 500 samples were collected at equally spaced time intervals. For SA, we tried the linear and geometric cooling schedules.

(a) Linear. The temperature decreases linearly, i.e.,

$$T_k = T_{k-1} - \varrho_l, \quad k = 1, \dots, K,$$

where $\varrho_l = (T_1 - T_K)/(K - 1)$.

(b) Geometric. The temperature decreases geometrically with a constant rate, i.e.,

$$T_k = \varrho_e T_{k-1}, \quad k = 1, \dots, K,$$

where $\varrho_e = \exp\{(\log T_K - \log T_1)/(T - 1)\}$.

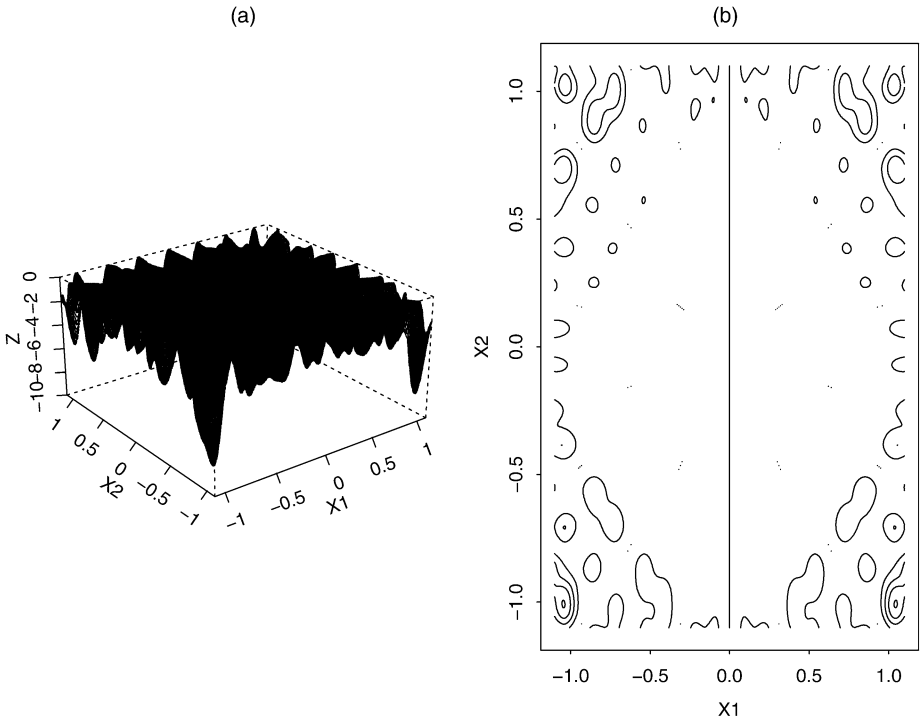

**Fig. 2** Grid (**a**) and contour (**b**) representations of the function $U(\boldsymbol{x})$ on $[-1.1, 1.1]^2$. This figure characterizes multiple local minima problems

In all simulations of this paper, we set the total number of temperature levels $K = 500$, and set the number of iterations performed at each temperature level to $N_k = N/K$, where $N$ is the total numbers of iterations of a run. For this example, we set $T_1 = 10$, $T_{500} = 0.01$ and $N = 50\,000$ for both cooling schedules. The resulting values of $\varrho_l$ and $\varrho_e$ are 0.02 and 0.986, respectively. The proposal distribution used at level $k$ is $N(\boldsymbol{x}_t, 0.1^2 T_k I_2)$. In each run, 500 samples were collected at equally spaced time intervals.

Figure 3 shows the evolving paths of the samples collected in the above runs. It is remarkable that ASAMC is ergodic as shown by Fig. 3(a). Even though the sample space has been restricted to four isolated regions (four corners) by the choice of $\Delta$, successful transitions between different regions can still be made due to the use of the global proposal distribution. This also explains why a widely spread proposal distribution is preferred in ASAMC. Comparing to the sample path of SAMC, we can see that in ASAMC, sampling is more focused on the low energy regions. Hence, ASAMC is potentially more efficient than SAMC in optimization.

Figures 3(c) and 3(d) show that at high temperatures, SA results in a random walk in the sample space; and that at low temperatures, SA tends to get trapped in a local minimum. Note that the linear cooling schedule contains more high temperature levels than the geometric cooling schedule. The sample paths of SA are significantly different from those of SAMC and ASAMC. The central part of the sample space (Fig. 2(b)) has a big area, which is about half of the total area of the sample space, but it is seldom visited by ASAMC and SAMC. However, this part is visited by SA frequently with both linear and geometric cooling schedules. The reason is that SA tends to have a random walk in the sample space at

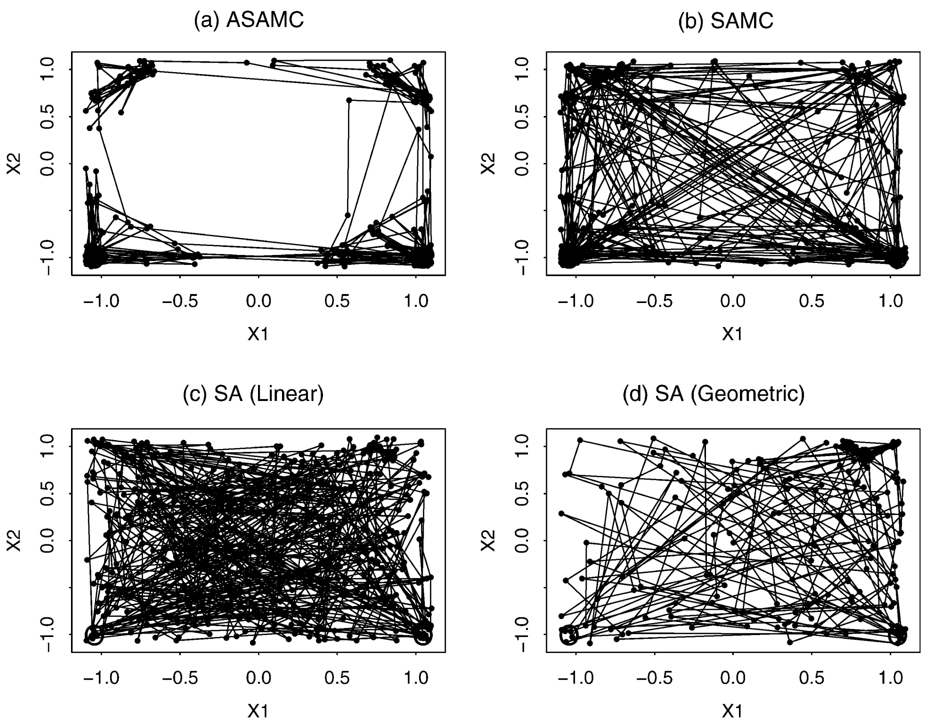

**Fig. 3** Sample paths of the ASAMC, SAMC and SA samples. The circles in the plots show the locations of the two global energy minima. **a** Sample path of ASAMC. **b** Sample path of SAMC. **c** Sample path of SA with the linear cooling schedule. **d** Sample path of SA with the geometric cooling schedule. This figure characterizes the performance of ASAMC for multiple local minima problems: it is capable of transiting between different local minimum regions

high temperatures, whereas ASAMC (so is SAMC) tends to have a random walk in the space of subregions, if each subregion is regarded as a single point. This implies that potentially ASAMC can overcome any barriers on the energy landscape and locate global energy minima quickly. Figure 3 shows that during the above runs SAMC and ASAMC have located the global energy minima many times, whilst SA has only located them a few times.

To compare efficiency of ASAMC, SAMC and SA in global optimization, we conducted the following experiment. Each algorithm was run 1000 times independently. Each run consisted of 20 000 iterations. ASAMC and SAMC were run under the same setting as used above except that the proposal distribution was changed to $N_2(\boldsymbol{x}, 0.1^2 I_2)$. This change would force them to move more locally and thus to have more chances to locate the global energy minima. The proposal distribution used in SA has already been fine enough, and was not changed in this experiment. The numerical results are summarized in Table 2. The comparison shows that both ASAMC and SAMC are superior to SA for this example. Note that in all runs of the three algorithms, the total numbers of iterations were the same, and they cost about the same CPU times because the CPU time cost by each iteration is dominated by the part used for energy evaluation. This is especially true for more complicated problems, e.g., MLP training considered in the rest of this paper.

**Table 2** Comparison of the SAMC, ASAMC, and SA algorithms for the multiple local minima example. Notations: let $z_i$ denote the minimum energy value obtained in the $i$th run for $i = 1, \dots, 1000$, "Mean" is the average of $z_i$, "SD" is the standard deviation of "Mean", "Minimum" $= \min_{i=1}^{1000} z_i$, "Maximum" $= \max_{i=1}^{1000} z_i$, and "Proportion" $= \#\{i : z_i \le -8.12\}$. The cooling schedules used in SA-1 and SA-2 are linear and geometric, respectively

| Algorithm | Mean | SD($\times 10^{-3}$) | Minimum | Maximum | Proportion |
|---|---|---|---|---|---|
| ASAMC | −8.12320 | 0.047 | −8.12465 | −8.11278 | 968 |
| SAMC | −8.12308 | 0.059 | −8.12465 | −8.10191 | 944 |
| SA-1 | −8.10326 | 1.264 | −8.12465 | −7.77922 | 572 |
| SA-2 | −8.08168 | 3.102 | −8.12466 | −7.33146 | 609 |

## 4 Numerical examples

### 4.1 Two benchmark examples

There are two fundamental tasks in neural network research, namely, modeling and estimation. To MLP, the former task is to find an appropriate objective function which can guide the learning of input patterns, and the latter task is to find an algorithm which can minimize the given objective function. This research falls into the latter category. In this section, we compare the efficiency of ASAMC, SAMC and SA in training MLPs through two benchmark examples, the $n$-parity and two-spiral problems. To calibrate them better, the regularization term in the energy function (2) is dropped such that the energy function has a global minimum value known as 0. The regularization term is usually included when we concern about generalization errors. To compare with the existing results of the two examples, the shortcut connections are not included in the MLPs, and both $\varphi_h(z)$ and $\varphi_o(z)$ are set to the sigmoid function.

#### *4.1.1 N-parity problem*

The training set of the $n$-parity problem (Rumelhart et al. 1986) consists of $2^n$ training pairs, where each input pattern consists of $n$ binary numbers (0 or 1), and the output is 1 if the input pattern consists of an odd number of 1s and 0 otherwise. This problem is very challenging for MLPs because the target output changes whenever a single bit in the input vector changes (Hanson 1991). Tesauro and Janssens (1988) reported that back-propagation solved the 8-parity problem using a one-hidden layer MLP with 16 hidden units, and Tang et al. (2003) reported the results from some gradient-based algorithms for 5, 6, and 7-parity problems with the same MLP structure.

For the 8-parity problem, we trained a smaller MLP with 11 hidden units, which consists of 111 connections. In ASAMC, the sample space was restricted to the region $\mathcal{X} = [-30, 30]^d$, and was partitioned into 320 subregions: $E_1 = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) \le 0.2\}$, $E_2 = \{\boldsymbol{x} \in \mathcal{X} : 0.2 < U(\boldsymbol{x}) \le 0.4\}, \dots$, and $E_{320} = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) > 63.8\}$. Even though $\mathcal{X}$ is much smaller than $\mathbb{R}^d$, it still contains multiple global minima for this example as shown by our numerical results. In simulations, we set $\psi(\boldsymbol{x}) = \exp\{-U(\boldsymbol{x})\}$, $t_0 = 2500$, $\iota = 0$, and $\Delta = 5$. The simulation starts with a random configuration generated from $N_d(0, 0.01^2)$, and stops when a configuration with energy $U(\boldsymbol{x}) < 0.2$ has been located or the maximum number of iterations $2 \times 10^6$ has been reached. At each iteration, one of the following two types of local moves is selected randomly. Let $\boldsymbol{x}_t$ denote the current configuration of the

**Table 3** Comparison of ASAMC, SAMC, SA and BFGS for the 8-parity problem. Notations: let $z_i$ denote the minimum energy value obtained in the $i$th run for $i = 1, \dots, 20$, "Mean" is the average of $z_i$, "SD" is the standard deviation of "mean", "minimum" $= \min_{i=1}^{20} z_i$, "maximum" $= \max_{i=1}^{20} z_i$, "proportion" $= \#\{i : z_i \le 0.21\}$, "Iteration" is the average number of iterations performed in each run, and "Time" is the average CPU time cost by each run. SA* refers to the runs of SA with the unrestricted sample space. SA-1 and SA*-1 employ the linear cooling schedule, and SA-2 and SA*-2 employ the geometric cooling schedule

| Algorithm | Mean | SD | Minimum | Maximum | Proportion | Iteration($\times 10^6$) | Time |
|---|---|---|---|---|---|---|---|
| ASAMC | 0.195 | 0.017 | 0.14 | 0.52 | 19 | 1.25 | 9.5 m |
| SAMC | 1.150 | 0.247 | 0.20 | 3.44 | 10 | 1.74 | 13.4 m |
| SA-1 | 25.272 | 2.841 | 10.11 | 60.023 | 0 | 2.00 | 13.7 m |
| SA-2 | 17.770 | 2.532 | 2.24 | 46.97 | 0 | 2.00 | 13.7 m |
| SA*-1 | 64.321 | 0.087 | 64.00 | 65.23 | 0 | 2.00 | 13.7 m |
| SA*-2 | 63.668 | 0.387 | 58.39 | 65.17 | 0 | 2.00 | 13.7 m |
| BFGS | 56.05 | 3.879 | 9.00 | 64.00 | 0 | – | 1 s |

MLP. In type I moves, a component of $\boldsymbol{x}_t$ is selected at random to undergo a modification by a Gaussian random variable $\epsilon \sim N(0, \sigma_t^2)$. In type II moves, a spherical proposal distribution is used: A direction is generated uniformly, and then the radius is drawn from $N(0, \sigma_t^2)$, where $\sigma_t$ is a stepwise function,

$$\sigma_t = \begin{cases} 0.5, & 0 < t \le 5 \times 10^5, \\ 1, & 5 \times 10^5 < t \le 10^6, \\ 2, & t > 10^6. \end{cases} \tag{14}$$

Here $\sigma_t$ increases as the simulation proceeds. This will improve the ergodicity of ASAMC, because the isolated regions included in $\mathcal{X}^{(t)}$ tend to be far and far separated. Under the above setting, ASAMC was run 20 times independently. On average, each run costs 9.5 m CPU time on a computer of 2.8 GHz (all simulations reported in this section were done on this machine). Table 3 summarizes the computational results produced by ASAMC and other algorithms described below.

SAMC and SA were also applied to this example. SAMC was run 20 times under the same setting as ASAMC except for $\iota = 5$. Other settings of $\iota$, say, $\iota = 0, 1, 2$ and 10, were also tried. The results are similar to or worse than those reported in Table 3. We note that for SA the sample space is not necessarily restricted to a compact region, but the restriction can often improve its performance, as the restriction makes the search focused on a small region and avoids blind search in the broad sample space. For each of the linear and geometric cooling schedules, SA was run 20 times with the highest temperature $T_1 = 10$ and the lowest temperature $T_{500} = 0.01$. Each run consisted of $N = 2 \times 10^6$ iterations. The proposal distribution is the same as that used by ASAMC except for $\sigma_k = \sqrt{T_k}$. For SA, we have also tried other cooling schedules, for example, linear and geometric schedules with a higher value of $T_1$ and a scaled reciprocal schedule (Szu 1986). The results are similar to or worse than those reported in Table 3.

For a thorough comparison, BFGS was also applied to this example. Like back-propagation, BFGS is gradient-based, but it usually converges faster than back-propagation. As commented by Hastie et al. (2001), back-propagation can be very slow, and for that reason it is usually not the algorithm of choice. Because of the difficulty in evaluation of the Hessian matrix, second-order techniques such as Newton's method are not attractive here.

The conjugate gradient and BFGS algorithms are often recommended for MLP training, as they avoid explicit evaluation of the Hessian matrix while still providing faster convergence. For the derivation of BFGS, refer to Bishop (1995) or a standard text such as Polak (1971) or Fletcher (1987). BFGS has been implemented in many software, such as S-PLUS and R, where a MLP can be trained by BFGS by calling the command $nnet(\cdot)$. The software R is publicly available at `http://cran.hostingzero.com/`. For this example, BFGS was also run 20 times independently, and it usually converged within 300 iterations in each run.

Table 3 shows that both ASAMC and SAMC have made dramatic improvements over SA and BFGS for this example. ASAMC produced perfect solutions in 19 out of 20 runs with an average of $1.25 \times 10^6$ iterations, whereas SA never produced a perfect solution even with more iterations in each run. Table 3 also indicates that ASAMC is better than SAMC in optimization. This is consistent with the results reported in Table 2.

### *4.1.2 Two-spiral problem*

Another famous benchmark example for MLP training is the two spiral problem (Lang and Witbrock 1989). The task requires the MLP to learn a mapping that distinguishes between points on two intertwined spirals shown in Fig. 4. This problem can be solved using MLPs with multiple hidden layers (with or without shortcut connections). Lang and Witbrock (1989) solved the problem using a 2-5-5-5-1 MLP with shortcut connections (138 trainable connection weights). Fahlman and Lebiere (1990) solved the problem using cascade-correlation networks with 12–19 hidden units, the smallest network having 114 connections. Wong and Liang (1997) solved the problem using a 2-14-4-1 MLP without shortcut connections. It is generally believed that this problem is very difficult to solve using the standard one-hidden-layer MLP, because it requires the MLP to learn a highly nonlinear separation of the input space. Baum and Lang (1991) reported that a solution could be found using a 2-50-1 back-propagation MLP, but only the MLP has been initialized with queries.

For this problem, we trained a 2-30-1 MLP, which consists of a total of 121 connections. In ASAMC, the sample space was restricted to the region $\mathcal{X} = [-50, 50]^d$, and was partitioned into 250 subregions: $E_1 = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) \le 0.2\}$, $E_2 = \{\boldsymbol{x} \in \mathcal{X} : 0.2 < U(\boldsymbol{x}) \le 0.4\}$, …, and $E_{250} = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) > 49.8\}$. In simulations, we set $\psi(\boldsymbol{x}) = \exp\{-U(\boldsymbol{x})\}$, $t_0 = 10\,000$, $\iota = 0$, and $\Delta = 5$, and employed the same proposal distribution as in the 8-parity example. ASAMC was run 20 times independently, and each run consisted of a maximum of $10^7$ iterations. The simulation stopped early if a solution with $U(\boldsymbol{x}) < 0.2$ was found. Figure 4(a) shows the classification map learned in one run, which indicates that a MLP with 30 hidden units is able to separate the two spirals successfully. Figure 4(b) shows the average classification map over the 20 runs by the ensemble averaging approach (Perrone 1993). Each solution in the ensemble was weighted equally. Ensemble averaging smoothes the classification boundary and improves the generalization performance of the MLP.

For comparison, SAMC, SA and BFGS were also applied to this example. SAMC was run 20 times under the same setting as ASAMC except for $\iota = 5$. Other choices of $\iota$, 0, 1, 2 and 10, have also been tried. The results are similar. BFGS was run 20 times independently, and it converged within 1000 iterations in each run. In the runs of SA, the sample space was restricted to $\mathcal{X} = [-50, 50]^d$ as in ASAMC, and the parameters held the same setting as used in the 8-parity example except that the total number of iterations was increased to $N = 10^7$.

Table 4 summarizes the results produced in the above runs. The comparison shows that ASAMC has made a dramatic improvement over SAMC, SA and BFGS for this example.

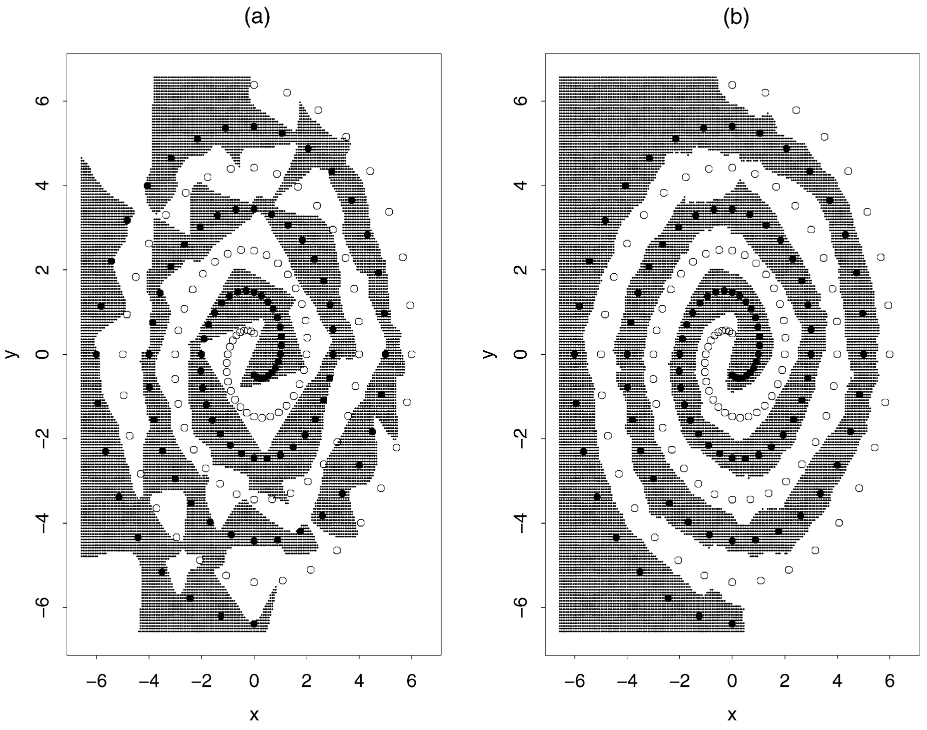

**Fig. 4** Classification maps learned by AESAMC with a MLP of 30 hidden units. The black and white points show the training data for two different spirals. **a** Classification map learned in one run. **b** Classification map averaged over 20 runs. This figure demonstrates the success of ASAMC in minimization of complex functions

**Table 4** Comparison of ASAMC, SAMC, SA and BFGS for the two-spiral example. The notations are the same as in Table 3

| Algorithm | Mean | SD | Minimum | Maximum | Proportion | Iteration($\times 10^6$) | Time |
|---|---|---|---|---|---|---|---|
| ASAMC | 0.620 | 0.191 | 0.187 | 3.23 | 15 | 7.07 | 94 m |
| SAMC | 2.727 | 0.208 | 1.092 | 4.089 | 0 | 10.0 | 132 m |
| SA-1 | 17.485 | 0.706 | 9.02 | 22.06 | 0 | 10.0 | 123 m |
| SA-2 | 6.433 | 0.450 | 3.03 | 11.02 | 0 | 10.0 | 123 m |
| BFGS | 15.50 | 0.899 | 10.00 | 24.00 | 0 | – | 3 s |

ASAMC found perfect solutions in 15 out of 20 runs with an average of $7.07 \times 10^6$ iterations, while SAMC, SA and BFGS failed to find perfect solutions in all runs.

We note that the MLP structure considered above is not minimal to this problem. ASAMC can find perfect solutions to this problem using a MLP with 27 hidden units (109 connections). Under the same setting as used above, ASAMC found perfect solutions using this small MLP in 5 out of 20 runs. The success rate can be increases by increasing the number of hidden units. In our simulations, ASAMC succeeded in 19 out of 20 runs when the number of hidden units is 35, and succeeded in 20 out of 20 runs when the number of hidden units is 40.

The $n$-parity and two-spiral problems represent, as explained above, different types of learning tasks, although both are very challenging. The former is to learn a mapping which is highly disconnected in the input space, and the latter is to learn a mapping which is highly nonlinear in the input space. The success of ASAMC in both problems suggests its superiority in minimization of complex functions.

## 4.2 Real-world examples

In this section, ASAMC was tested with three real-world examples for which all attributes are numeric and there are no missing values. Refer to Bishop (1995) for discussions on the treatments for missing values in the context of neural networks, whilst this is beyond the scope of this paper. For the real-world problems, we are often concerned with the generalization performance of MLP. Hence, the regularization term was included in the energy function (2) for all the three examples. The regularization parameter was set to 0.05. In applying ASAMC and SAMC to these examples, the sample space was restricted to the region $\mathcal{X} = [-50, 50]^d$, and the parameters were set as follows: $\psi(\boldsymbol{x}) = \exp\{-U(\boldsymbol{x})\}$, $t_0 = 1000$, and $\Delta = 5$. The proposal distribution employed in this section is the same as that used for the 8-parity example except that $\sigma_t$ was fixed to 1. Each algorithm was run 50 times independently, and each run consisted of $2.5 \times 10^5$ iterations. In all the three examples, both $\varphi_h(z)$ and $\varphi_o(z)$ were set to the sigmoid function, the input variables were normalized to have mean 0 and variance 1, and $\iota$ was set to 0 for ASAMC and 5 for SAMC. For SAMC, other choices of $\iota$, e.g., 0 and 2, were also tried. The results are similar.

As implied by Theorem 8.3, both ASAMC and SAMC will converge weakly toward the set $E_*$. Although it may be easy for them to reach $E_*$, it may not be easy for them to locate a global energy minimum exactly. So we suggest the best solutions sampled by SAMC and ASAMC be further subject to a post minimization procedure, say, conjugate gradient or a few hundred steps of MH moves performed at a low temperature. For the three examples, the MH moves were performed at temperature $t = 10^{-4}$.

SA was also applied to the three examples in this section. For each example, SA was run 50 times with the highest temperature $T_1 = 10$ and the lowest temperature $T_{500} = 10^{-4}$. Each run consisted of $N = 2.5 \times 10^5$ iterations. The sample space was restricted to $[-50, 50]^d$ as in ASAMC and SAMC. The proposal distribution was the same as that used by ASAMC except for $\sigma_k = \sqrt{T_k}$. Both the geometric and linear cooling schedules were tried. Since the results from the linear cooling schedule are inferior to those from the geometric one, only the results from the geometric cooling schedule are reported below.

### *4.2.1 BUPA liver disorders*

This dataset was from Richard S. Forsyth at BUPA Medical Research Ltd and can be downloaded from the machine learning repository at UCI (`http://www.ics.uci.edu/~mlearn/MLRepository.html`). Each sample in the dataset constitutes the record of a single male individual. Five attributes (mean corpuscular volume, alkaline phosphatase, alamine aminotransferase, aspartate aminotransferase, and $\gamma$-glutamyl transpeptidase) in each record are results from blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. The sixth feature is the number of drinks per day. There are a total of 345 samples in the dataset.

In our study, 200 samples were randomly chosen as the training samples and the rest as the test samples. The data was modeled by a MLP with 2 hidden units and shortcut connections. The total number of connections is 23. The MLP was trained by ASAMC,

**Table 5** Comparison of the ASAMC, SAMC, SA and BFGS algorithms for the BUPA liver disorders example. The training error is measured as the minimum energy value found in each run, and the test error is measured as the misclassification rate for the test samples

| Algorithms | Training error | | | Test error rate (%) | | | Time[7] |
|---|---|---|---|---|---|---|---|
| | mean[1] | min[2] | max[3] | mean[4] | min[5] | max[6] | |
| ASAMC | 32.838 (0.053) | 32.483 | 33.208 | 33.556 (0.251) | 31.724 | 35.862 | 17s |
| SAMC | 32.994 (0.050) | 32.483 | 34.270 | 34.634 (0.218) | 31.034 | 40.000 | 17s |
| SA | 33.239 (0.063) | 32.483 | 34.533 | 34.483 (0.278) | 29.655 | 40.690 | 14s |
| BFGS | 33.690 (0.112) | 32.696 | 36.200 | 34.828 (0.290) | 29.655 | 40.690 | 1s |

[1]Average of the training errors over 50 runs (the number in the parentheses represent the standard deviation of the average)

[2]Minimum training error over 50 runs

[3]Maximum training error over 50 runs

[4]Average misclassification rate over 50 runs (the number in the parentheses represent the standard deviation of the average)

[5]Minimum misclassification rate over 50 runs

[6]Maximum misclassification rate over 50 runs

[7]CPU time cost by a single run on a core Intel Xeon X5355 with speed 2.66 GHz

**Table 6** Comparison of the ASAMC, SAMC, SA and BFGS algorithms for the Pima Indians diabetes example. Refer to Table 5 for notations of the table

| Algorithms | Training error | | | Test error rate (%) | | | Time |
|---|---|---|---|---|---|---|---|
| | mean | min | max | mean | min | max | |
| ASAMC | 82.986 (0.035) | 82.784 | 83.494 | 19.60 (0.174) | 17.71 | 22.40 | 63 s |
| SAMC | 83.356 (0.054) | 82.785 | 83.817 | 21.28 (0.264) | 18.23 | 26.04 | 63 s |
| SA | 83.517 (0.066) | 82.785 | 84.782 | 21.13 (0.238) | 17.71 | 23.96 | 53 s |
| BFGS | 83.841 (0.103) | 82.786 | 87.480 | 21.77 (0.214) | 18.75 | 26.04 | 1 s |

SAMC, SA and BFGS. The numerical results are summarized in Table 5. In ASAMC and SAMC, the sample space were partitioned into 250 subregions: $E_1 = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) \leq 0.2\}$, $E_2 = \{\boldsymbol{x} \in \mathcal{X} : 0.2 < U(\boldsymbol{x}) \leq 0.4\}, \ldots$, and $E_{250} = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) > 49.8\}$. For this example, ASAMC outperforms the other algorithms in both training and test errors.

### *4.2.2 Pima Indians diabetes classification*

The data were collected by the National Institute of Diabetes and Digestive and Kidney Diseases and can be downloaded from the machine learning repository at UCI. The dataset contains 768 records of female Pima Indians, characterized by eight physiological descriptors. The task is to predict presence or absence of diabetes. An early study of this data was carried out by Smith et al. (1988) using a MLP. In their study, the MLP was trained using 576 samples, and an accuracy rate of 76% was achieved on the rest 192 samples.

We modeled the data by a MLP with 3 hidden units (without shortcut connections) and a total of 31 connections. The MLP was trained using ASAMC, SAMC, SA and BFGS on the first 576 samples. The results are summarized in Table 6. In ASAMC and SAMC,

**Table 7** Comparison of the ASAMC, SAMC, SA and BFGS algorithms for the email spam identification example. Refer to Table 5 for notations of the table

| Algorithms | Training error | | | Test error rate (%) | | | Time |
|---|---|---|---|---|---|---|---|
| | mean | min | max | mean | min | max | |
| ASAMC | 115.833 (0.501) | 111.198 | 123.455 | 6.464 (0.051) | 5.534 | 7.357 | 7.3 m |
| SAMC | 129.113 (0.665) | 122.241 | 147.121 | 6.629 (0.048) | 5.990 | 7.161 | 8.0 m |
| SA | 120.406 (0.609) | 112.632 | 129.590 | 6.652 (0.049) | 5.859 | 7.487 | 6.9 m |
| BFGS | 124.303 (1.056) | 114.713 | 141.684 | 6.665 (0.052) | 5.859 | 7.292 | 3.4 s |

the sample space were partitioned into 500 subregions with an equal energy bandwidth, $E_1 = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) \leq 0.2\}$, $E_2 = \{\boldsymbol{x} \in \mathcal{X} : 0.2 < U(\boldsymbol{x}) \leq 0.4\}$, $\ldots$, and $E_{500} = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) > 99.8\}$. The comparison shows that ASAMC outperforms the other algorithms in both training and test errors. ASAMC achieved an accuracy rate of about 81%, which is better than 76% obtained by Smith et al. (1988).

### 4.2.3 *Email spam identification*

The data were collected by Hewlett-Packard laboratories, Palo Alto, California in a study to screen email for "spam" and can be downloaded from the machine learning repository at UCI. The data consists of information from 4601 email messages. For all 4601 email messages, the true outcome (email type) email or spam is available, along with the relative frequencies of 57 of the most commonly occurring words and punctuation marks in the email message.

In this study, 3065 samples were randomly chosen as the training samples and the rest as the test samples. The data was modeled by a MLP with 2 hidden units and without shortcut connections, which consists of 119 connections. The MLP was trained by ASAMC, SAMC, SA and BFGS. The numerical results are summarized in Table 7. In ASAMC and SAMC, the sample space were partitioned into 1250 subregions with an equal energy bandwidth: $E_1 = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) \leq 0.2\}$, $E_2 = \{\boldsymbol{x} \in \mathcal{X} : 0.2 < U(\boldsymbol{x}) \leq 0.4\}$, $\ldots$, and $E_{250} = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) > 248.2\}$. For this example, ASAMC again outperforms the other training algorithms in both training and test errors.

The three real-world examples represent applications of ASAMC to classification MLPs with different sizes of training sets. If, for example, we use "samples $\times$ attributes" to measure the size of a training set (suppose that the numbers of samples and attributes have about the same ratio in each training set), then the training set sizes of the three examples are 1200, 4608 and 174,705, respectively. So the three examples represent problems with a small, moderate and large size of training set, respectively. The success of ASAMC in the three examples suggests that ASAMC can make reliable training and predictions for problems with various sizes of training sets.

## 4.3 Significance of ASAMC results

ASAMC has been compared with SAMC, SA and BFGS on two benchmark examples and three real-world examples. The significance of the ASAMC results has been assessed using the two-sample $t$-tests. The corresponding $p$-values are summarized in Table 8. The hypotheses corresponding to the first entry of Table 8 are $H_0$: *ASAMC has the same training error as SAMC for the 8-parity example* versus $H_1$: *ASAMC has a smaller training error*

**Table 8** The $p$-values of the two-sample $t$-tests (with unequal variances) for the training and test errors produced by ASAMC versus those produced by SAMC, SA and BFGS. For SA, only the results generated with the geometric cooling schedule and from the restricted sample space are considered

| Example | Training error | | | Test error | | |
|---|---|---|---|---|---|---|
| | SAMC | SA | BFGS | SAMC | SA | BFGS |
| 8-parity | $5.2\times10^{-4}$ | $6.4\times10^{-7}$ | $5.6\times10^{-12}$ | – | – | – |
| 2-spiral | $3.1\times10^{-9}$ | $3.1\times10^{-12}$ | $1.6\times10^{-13}$ | – | – | – |
| BUPA | $1.7\times10^{-2}$ | $2.2\times10^{-6}$ | $1.1\times10^{-9}$ | $8.1\times10^{-4}$ | $7.5\times10^{-3}$ | $6.4\times10^{-4}$ |
| Pima | $7.0\times10^{-8}$ | $3.0\times10^{-10}$ | $4.2\times10^{-11}$ | $4.3\times10^{-7}$ | $6.5\times10^{-7}$ | $3.0\times10^{-12}$ |
| Spam | 0.0 | $4.4\times10^{-8}$ | $2.2\times10^{-10}$ | $1.0\times10^{-2}$ | $4.6\times10^{-3}$ | $3.5\times10^{-3}$ |

*than SAMC for the 8-parity example*. The hypotheses corresponding to other entries are similar. Table 8 shows that for all MLP examples studied in this paper, ASAMC can outperform significantly (at a level of 0.01) SAMC, SA and BFGS in both training and test errors. Like other stochastic algorithms, ASAMC is slower than the gradient-based algorithms. However, as indicated by the running times reported above, its training speed is still acceptable to us.

## 5 Discussion and future work

The strength of ASAMC comes from two aspects, its self-adjusting mechanism and its progressive shrinkage for the sample space. The self-adjusting mechanism makes ASAMC capable of overcoming the local-trap problems encountered in MLP training, and the sample space shrinkage accelerates exploration for low energy regions. We note that for a given problem, the significance of ASAMC should only apply to the appropriately chosen networks. For the networks which overfit to the data excessively, ASAMC and other training algorithms may perform equally in both training and test errors.

In this paper, we considered only the MLPs that have a single layer of hidden units and use the sigmoid function as activation functions. Such MLPs are rather general. Cybenko (1989), Funahashi (1989), Hornik et al. (1989), and Barron (1993) have shown the so-called universal approximation theorem: a MLP with a single hidden layer can approximate any given training set uniformly by increasing the number of hidden units, assuming that the activation function of the hidden units is a nonconstant, bounded, and monotonically increasing continuous function. If the MLP is modified, e.g., to include multiple hidden layers, multiple output units, or to use other functions as activation functions, ASAMC should still work well given its superiority in function optimization. It is worth noting that the tangent activation function often gives rise to faster convergence of training algorithms than does the sigmoid function (Bishop 1995, p. 127).

ASAMC avoids the requirement for the gradient information of the energy function, so it can be applied to the optimization problems for which the gradient information is not available, e.g., decision tree learning problems (Sastry et al. 2002). This is similar to SA. Besides optimization, SAMC and ASAMC can also be applied to solve the problems of function approximation, as illustrated by the knapsack example. A neural learning example in this respect is evidence evaluation for Bayesian MLPs. As pointed out by MacKay (1992b), the Bayesian evidence can provide a useful guideline of architecture selection for

MLPs. Let $f(D|\boldsymbol{x})$ denote the likelihood function of a MLP, and let $l(\boldsymbol{x})$ denote the prior density imposed on $\boldsymbol{x}$. Define the function

$$\psi(\boldsymbol{x},k)=\begin{cases} f(D|\boldsymbol{x})l(\boldsymbol{x}), & k=1,\\ 1/V, & k=0,\end{cases} \tag{15}$$

on the product space $\mathcal{X}\times\{0,1\}$, where $V=|\mathcal{X}|$ denotes the hypervolume of the space $\mathcal{X}$. Recall that $\mathcal{X}$ has been restricted to a compact set in this paper. Partition the product space as follows: $E_0=\{(\boldsymbol{x},k):k=0,\boldsymbol{x}\in\mathcal{X}\}$, $E_1=\{(\boldsymbol{x},k):k=1,U(\boldsymbol{x})\le u_1,\boldsymbol{x}\in\mathcal{X}\}$, …, $E_m=\{(\boldsymbol{x},k):k=1,U(\boldsymbol{x})>u_{m-1},\boldsymbol{x}\in\mathcal{X}\}$. If SAMC is run with this partition, it then follows from equation (9) that the evidence of the MLP can be estimated by

$$\widehat{EV}=\frac{\sum_{i=1}^{m}(\pi_i+\aleph)\hat{g}_i}{(\pi_0+\aleph)\hat{g}_0}g_0, \tag{16}$$

where $\hat{g}_i=\lim_{t\to\infty}e^{\theta_{ti}}$ and $0<\pi_0<1$. The function $\psi(\boldsymbol{x},0)$ does not need to be uniform, it can be any non-negative function with $g_0=\int_{E_0}\psi(\boldsymbol{x},0)d\boldsymbol{x}$ being analytically available. Refer to Jerrum and Sinclair (1997) and Chawla et al. (2004) for more discussions on function approximation.

The theory and methodology related to function optimization can be further developed based on this work. In the present version of ASAMC, the space shrinkage parameter $\Delta$ is set to a constant. If we associate it with iterations by letting $\Delta_t$ be a monotonically decreasing function of $t$ with the limit 0, then ASAMC will converge in distribution toward the set of global energy minimizers. An interesting problem is to find the decreasing rate of $\Delta_t$ under which ASAMC has the fastest convergence to the global energy minimizers.

Evolutionary Monte Carlo (Liang and Wong 2001; Goswami and Liu 2005; Jasra et al. 2007) has been shown to be useful for alleviating the local-trap problem, due to the use of population information through crossover operations on pairs of candidate solutions. We believe that the efficiency of ASAMC can be further improved by extending it to multiple chains. Appropriate crossover operations can then be adopted to accelerate its convergence to the set of global minimizers.

## 6 Conclusion

In this paper we have proposed ASAMC as a new stochastic optimization algorithm, and shown under mild conditions that ASAMC can converge weakly at a rate of $\Omega(1/\sqrt{t})$ toward a neighboring set (in the space of energy) of the global minimizers. We have compared ASAMC with simulated annealing, SAMC, and the BFGS algorithm for training MLPs on two benchmark and three real-world examples. Our numerical results indicate that ASAMC can outperform the other algorithms in both training and test errors. Like other stochastic optimization algorithms, ASAMC requires longer training times than does the BFGS algorithm. It provides, however, an efficient approach to the problems of minimization of rugged functions.

**Acknowledgements** The author thanks Professor Risto Miikkulainen and two referees for their helpful comments which have led to significant improvement of this paper. The author's research is partially supported by grants from the National Science Foundation (DMS-0405748) and the National Cancer Institute (CA104620).

## Appendix 1

### 7.1 Pseudocode of ASAMC

Consider the problem of minimizing an energy function $U(\boldsymbol{x})$ over a compact space $\mathcal{X}$. Let $U_{low}$ be a known lower bound of $U(\boldsymbol{x})$ on $\mathcal{X}$, which can be smaller than the true global minimum value of $U(\boldsymbol{x})$. Let $m$ denote the number of subregions to be partitioned, let $h$ denote the energy bandwidth of each subregion, let $J(\boldsymbol{x})$ be the index of the subregion that $\boldsymbol{x}$ belongs to, i.e., $\boldsymbol{x} \in E_{J(\boldsymbol{x})}$, let $U_{\min}^{(t)}$ be the minimum energy value found by iteration $t$, and let $\boldsymbol{x}_{\min}^{(t)}$ be the corresponding minimizer, i.e., $U_{\min}^{(t)} = U(\boldsymbol{x}_{\min}^{(t)})$. In addition, we define the notations: $N$ is the total number of iterations of a run, $\Delta$ is the space annealing control parameter, $\tau$ and $t_0$ determine the gain factor sequence as in (8), and $\iota$ determines the desired sampling distribution as in (12).

```
(a) (Initialization)
    (a.1) Choose appropriate values for the parameters[3]: $B_\Theta$, $U_{low}$, $m$, $h$, $\Delta$, $N$, $\tau$, $t_0$ and $\iota$.
    (a.2) Normalize the desired sampling distribution $\boldsymbol{\pi}$ by setting $\pi_i = i^{-\iota}/\sum_{j=1}^{m} j^{-\iota}$ for
          $i = 1, \ldots, m$.
    (a.3) Initialize the working estimate $\theta_0 = (\theta_{01}, \ldots, \theta_{0m}) = (0, \ldots, 0)$, draw a random
          sample $\boldsymbol{x}_0$ from $\mathcal{X}$ ($\boldsymbol{x}_0$ can be an arbitrary point in $\mathcal{X}$), and set $U_{\min}^{(0)} = U(\boldsymbol{x}_0)$ and
          $\boldsymbol{x}_{\min}^{(0)} = \boldsymbol{x}_0$.
(b) (Iterations) For $t = 0$ to $N - 1$, do the following:
    (b.1) Set $\mathcal{X}^{(t)} \leftarrow \bigcup_{i=1}^{\varpi(U_{\min}^{(t)}+\Delta)} E_i$.
    (b.2) Draw a random sample $\boldsymbol{y}$ from the proposal distribution $q(\boldsymbol{x}, \boldsymbol{y})$.
    (b.3) Calculate the ratio

              $$r = e^{(\theta_{tJ(\boldsymbol{x}_t)} - \theta_{tJ(\boldsymbol{y})})} \frac{\psi(\boldsymbol{y})}{\psi(\boldsymbol{x}_t)} \frac{q(\boldsymbol{y}, \boldsymbol{x}_t)}{q(\boldsymbol{x}_t, \boldsymbol{y})},$$

          where $q(\boldsymbol{x}, \boldsymbol{y})$ is a global proposal distribution defined on $\mathcal{X}$.
    (b.4) If $r > 1$, set $\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{y}$
          else {

            * Draw a random number $u \sim$ Unif[0, 1].
            * If $u \le r$, set $\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{y}$; otherwise, set $\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}_t$.

          }
    (b.5) Set $\boldsymbol{e}_{t+1} \leftarrow (e_{t+1,1}, \ldots, e_{t+1,m})$, where $e_{t+1,i} = 1$ if $i = J(\boldsymbol{x}_{t+1})$ and 0 otherwise.
    (b.6) Set $\theta^* \leftarrow \theta_t + \gamma_t(\boldsymbol{e}_{t+1} - \boldsymbol{\pi})$.
    (b.7) If $\theta^* \in \Theta$, set $\theta_{t+1} \leftarrow \theta^*$;
          else {

            * If $\max_{i=1}^{m} \theta_i^* > B_\Theta$, set $c^* \leftarrow B_\Theta/2 - \max_{i=1}^{m} \theta_i^*$;
              else if $\min_{i=1}^{m} \theta_i^* < -B_\Theta$, set $c^* \leftarrow -B_\Theta/2 - \min_{i=1}^{m} \theta_i^*$.
            * Set $\theta_{t+1} \leftarrow \theta^* + c^*$.

          }
```

[3]By choosing the values of $U_{low}$, $m$ and $h$, the sample space is partitioned implicitly into $m$ subregions: $E_1 = \{\boldsymbol{x} : U_{low} < U(\boldsymbol{x}) \le U_{low} + h\}$, ..., $E_{m-1} = \{\boldsymbol{x} : U_{low} + (m-2)h < U(\boldsymbol{x}) \le U_{low} + (m-1)h\}$, $E_m = \{\boldsymbol{x} : U(\boldsymbol{x}) > U_{low} + (m-1)h\}$.

(b.8) If $U(\boldsymbol{x}_{t+1}) < U_{\min}^{(t)}$, set $U_{\min}^{(t+1)} \leftarrow U(\boldsymbol{x}_{t+1})$ and $\boldsymbol{x}_{\min}^{(t+1)} \leftarrow \boldsymbol{x}_{t+1}$; else set $U_{\min}^{(t+1)} \leftarrow U_{\min}^{(t)}$ and $\boldsymbol{x}_{\min}^{(t+1)} \leftarrow \boldsymbol{x}_{\min}^{(t)}$.

(c) (Output) Output $U_{\min}^{(t+1)}$ and $\boldsymbol{x}_{\min}^{(t+1)}$ as a solution to the minimization problem under consideration.

### 7.2 Pseudocode of SAMC

The pseudocode of SAMC can be simply obtained by making the following modifications on the pseudocode of ASAMC:

- Set $\mathcal{X}^{(t)} \equiv \mathcal{X}$ in step (b.1).
- Set $p'_{\theta_t}(\boldsymbol{x}) = p_{\theta_t}(\boldsymbol{x})$ in step (b.3).
- (Option) Employ a local proposal distribution in step (b.3) provided that $0 < \psi(\boldsymbol{x}) < \infty$ for all $\boldsymbol{x} \in \mathcal{X}$.

## Appendix 2

This appendix is organized as follows. In Sect. 8.1, we briefly review the published results on the convergence of a general stochastic approximation algorithm. In Sect. 8.2, we proved three theorems concerning the convergence and convergence rate of ASAMC.

### 8.1 Published results on the convergence of a general stochastic approximation algorithm

Suppose that we are interested in solving the following integration equation for the parameter vector $\theta$:

$$h(\theta) = \int_{\mathcal{X}} H(\theta, \boldsymbol{x}) p(d\boldsymbol{x}) = 0, \quad \theta \in \Theta.$$

The stochastic approximation algorithm with Markov chain Monte Carlo works iteratively as follows. Let $K(\boldsymbol{x}_t, \cdot)$ denote a probability transition kernel. For example, it can be the MH kernel of the form

$$K(\boldsymbol{x}_t, d\boldsymbol{y}) = s(\boldsymbol{x}_t, d\boldsymbol{y}) + I(\boldsymbol{x}_t \in d\boldsymbol{y}) \left(1 - \int_{\mathcal{X}} s(\boldsymbol{x}_t, \boldsymbol{z}) d\boldsymbol{z}\right),$$

where $s(\boldsymbol{x}_t, d\boldsymbol{y}) = q(\boldsymbol{x}_t, d\boldsymbol{y}) \min\{1, [p(\boldsymbol{y}) q(\boldsymbol{y}, \boldsymbol{x}_t)]/[p(\boldsymbol{x}) q(\boldsymbol{x}, \boldsymbol{y})]\}$, $q(\cdot,\cdot)$ is the proposal distribution, and $p(\cdot)$ is the target distribution.

Let $\Theta \subset \tilde{\Theta}$ be a compact subset of $\tilde{\Theta}$. Let $\{\gamma_t\}_{t=0}^{\infty}$ be a monotone, non-increasing sequence. Also define a function $\Phi : \mathcal{X} \times \tilde{\Theta} \to \mathcal{X} \times \Theta$, which reinitializes the non-homogeneous Markov chain $\{(\boldsymbol{x}_t, \theta_t)\}$. The function $\Phi$ can for instance generate a random or fixed point, or project $(\boldsymbol{x}_{t+1}, \theta_{t+1})$ onto $\mathcal{X} \times \Theta$. An iteration of the algorithm is given as follows.

*Stochastic approximation algorithm*

(i) Generate $\boldsymbol{y} \sim K_{\theta_t}(\boldsymbol{x}_t, \cdot)$.
(ii) Set $\theta^* = \theta_t + \gamma_{t+1} H(\theta_t, \boldsymbol{y})$.
(iii) If $\theta^* \in \Theta$, then set $(\boldsymbol{x}_{t+1}, \theta_{t+1}) = (\boldsymbol{y}, \theta^*)$; otherwise, set $(\boldsymbol{x}_{t+1}, \theta_{t+1}) = \Phi(\boldsymbol{y}, \theta^*)$.

The above algorithm is a simplified version of the stochastic approximation MCMC algorithm presented in Andrieu et al. (2005) and Erland (2003). To show the convergence of the algorithm, the following conditions are assumed.

*Lyapunov condition on $h$* Let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denote the Euclidean inner product and let $|\boldsymbol{x}|$ be the corresponding norm of $\boldsymbol{x}$.

(A$_1$) The function $h : \Theta \to \mathbb{R}^d$ is continuous, and there exists a continuously differentiable function $v : \Theta \to [0, \infty)$ such that

(i) For any integer $M > 0$, the level set $\mathcal{V}_M = \{\theta \in \Theta, v(\theta) \le M\} \subset \Theta$ is compact.

(ii) There exists $M_0 > 0$ such that

$$\mathcal{L} = \{\theta \in \Theta, \langle \nabla v(\theta), h(\theta) \rangle = 0\} \subset \text{int}(\mathcal{V}_{M_0}),$$

and $\langle \nabla v(\theta), h(\theta) \rangle < 0$ for any $\theta \in \Theta \setminus \mathcal{V}_{M_0}$, where $\text{int}(A)$ denotes the interior of set $A$.

(iii) For all $\theta \in \Theta$, $\langle \nabla v(\theta), h(\theta) \rangle \le 0$, and $\text{int}(v(\mathcal{L})) = \emptyset$.

*Drift and continuity conditions on the transition kernel $K_\theta$* Assume that a transition kernel $K$ is $\varphi$-irreducible, aperiodic and has a unique stationary distribution. A set $C \subset \mathcal{X}$ is said to be a small set if there exists a probability measure $\nu$ on $\mathcal{X}$, a positive integer $l$, and $\delta > 0$ such that

$$K_\theta^l(\boldsymbol{x}, A) \ge \delta \nu(A), \quad \forall \boldsymbol{x} \in C, \ \forall A \in \mathcal{B},$$

where $\mathcal{B}$ is the Borel set. Refer to Meyn and Tweedie (1993) for more discussions on small set. A function $V : \mathcal{X} \to [1, \infty)$ is said to be a drift function outside $C$ if there exist constants $\lambda < 1$ and $b$ such that

$$K_\theta V(\boldsymbol{x}) \le \lambda V(\boldsymbol{x}) + bI(\boldsymbol{x} \in C), \quad \forall \boldsymbol{x} \in \mathcal{X},$$

where $K_\theta V(\boldsymbol{x}) = \int_{\mathcal{X}} K_\theta(\boldsymbol{x}, \boldsymbol{y}) V(\boldsymbol{y}) d\boldsymbol{y}$. For $g : \mathcal{X} \to \mathbb{R}^d$, define the norm

$$\|g\|_V = \sup_{\boldsymbol{x} \in \mathcal{X}} \frac{|g(\boldsymbol{x})|}{V(\boldsymbol{x})},$$

and define the set $\mathcal{L}_V = \{g : \mathcal{X} \to \mathbb{R}^d, \|g\|_V < \infty\}$.

(A$_2$) If $K_\theta$ is irreducible and aperiodic for all $\theta \in \Theta$, there exist a function $V : \mathcal{X} \to [1, \infty)$ and constants $p \ge 2$ and $\beta \in [0, 1]$ such that for any compact subset $\Theta_0 \in \Theta$,

(i) there exist an integer $l$, constants $0 < \lambda < 1$, $b, \kappa, \delta > 0$ and a probability measure $\nu$ such that

- $\sup_{\theta \in \Theta_0} K_\theta^l V^p(\boldsymbol{x}) \le \lambda V^p(\boldsymbol{x}) + bI(\boldsymbol{x} \in C)$, $\forall \boldsymbol{x} \in \mathcal{X}$,
- $\sup_{\theta \in \Theta_0} K_\theta V^p(\boldsymbol{x}) \le \kappa V^p(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathcal{X}$,
- $\inf_{\theta \in \Theta_0} K_\theta^l(\boldsymbol{x}, A) \ge \delta \nu(A)$, $\forall \boldsymbol{x} \in C$, $\forall A \in \mathcal{X}$;

(ii) there exists a constant $c_1$ such that for all $\boldsymbol{x} \in \mathcal{X}$,

- $\sup_{\theta \in \Theta_0} |H(\theta, \boldsymbol{x})| \le c_1 V(\boldsymbol{x})$,
- $\sup_{(\theta, \theta') \in \Theta_0 \times \Theta_0} |H(\theta, \boldsymbol{x}) - H(\theta', \boldsymbol{x})| \le c_1 V(\boldsymbol{x}) |\theta - \theta'|^\beta$;

(iii) there exists a constant $c_2$ such that for all $(\theta, \theta') \in \Theta_0 \times \Theta_0$,

- $\|K_\theta g - K_{\theta'} g\|_V \le c_2 \|g\|_V |\theta - \theta'|^\beta$, $\forall g \in \mathcal{L}_V$,
- $\|K_\theta g - K_{\theta'} g\|_{V^p} \le c_2 \|g\|_{V^p} |\theta - \theta'|^\beta$, $\forall g \in \mathcal{L}_{V^p}$.

*Conditions on the step-size*

(A$_3$) The sequence $\{\gamma_t\}_{t=0}^{\infty}$ is non-increasing and positive and satisfies the conditions
  (i) $\sum_{t=1}^{\infty}\gamma_t = \infty$,
  (ii) $\sum_{t=1}^{\infty}\gamma_t^{\zeta} < \infty$,
  (iii) at each step, $|\gamma_t H(\theta_{t-1}, \boldsymbol{x}_t)| < C\gamma_t^{\eta}$, $t = 1, 2, \ldots$,
  for some $\zeta \in (1, p(1+\alpha)/(p+\alpha)]$, some $\eta \in [(\zeta - 1)/\alpha, (p-\zeta)/p]$, and a constant $C$. Here $\alpha \in (0, \beta]$, $\beta \in [0, 1]$ and $p \geq 2$ are defined in (A$_2$).

*A main convergence result* Let $\mathbb{P}_{\boldsymbol{x}_0,\theta_0}$ denote the probability measure of the Markov chain $\{(\boldsymbol{x}_t, \theta_t)\}$, started in $(\boldsymbol{x}_0, \theta_0)$, and implicitly defined by the sequences $\{\gamma_t\}$. Also define $D(\boldsymbol{x}, A) = \inf_{\boldsymbol{y}\in A} |\boldsymbol{x} - \boldsymbol{y}|$.

**Theorem 8.1** (Andrieu et al. 2005) *Assume the conditions* (A$_1$), (A$_2$) *and* (A$_3$) *hold, and* $\sup_{\boldsymbol{x}\in\mathcal{X}} V(\boldsymbol{x}) < \infty$. *Let the sequence* $\{\theta_n\}$ *be defined as in the stochastic approximation algorithm. Then for all* $(\boldsymbol{x}_0, \theta_0) \in \mathcal{X} \times \Theta$,

$$\lim_{t\to\infty} D(\theta_t, \mathcal{L}) = 0, \quad \textit{almost surely.}$$

## 8.2 Three theorems for the SAMC algorithm

Without loss of generality, we only show the convergence presented in (9) of the paper for the case that all subregions are non-empty or, equivalently, $\aleph = 0$. Extension to the case $\aleph \neq 0$ is trivial, because changing step (ii) of the SAMC algorithm to (ii)$'$ (given below) will not change the process of simulation.

(ii)$'$ Set $\theta' = \theta_t + \gamma_{t+1}[H(\theta_t, \boldsymbol{x}_{t+1}) - \aleph]$.

**Theorem 8.2** *Let* $E_1, \ldots, E_m$ *form a partition of a compact sample space* $\mathcal{X}$ *and* $\psi(\boldsymbol{x})$ *be a non-negative function defined on* $\mathcal{X}$ *with* $0 < \int_{E_i}\psi(\boldsymbol{x})d\boldsymbol{x} < \infty$ *for all* $E_i$*'s. Let* $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)$ *be an m-vector with* $0 < \pi_i < 1$ *and* $\sum_{i=1}^{m}\pi_i = 1$. *Let* $\Theta$ *be a compact set of m dimensions. Assume that there exists at least a constant C such that* $\breve{\theta} \in \Theta$, *where* $\breve{\theta} = (\breve{\theta}_1, \ldots, \breve{\theta}_m)$ *with* $\breve{\theta}_i = C + \log(\int_{E_i}\psi(\boldsymbol{x})d\boldsymbol{x}) - \log(\pi_i)$. *Let* $\theta_0 \in \Theta$ *be an initial estimate of* $\breve{\theta}$, *let* $\theta_t \in \Theta$ *be the estimate of* $\breve{\theta}$ *at iteration t, and let* $\{\gamma_t\}$ *be a non-increasing, positive sequence as specified in* (8). *Suppose that* $p_{\theta_t}(\boldsymbol{x})$ *is bounded away from* 0 *and* $\infty$ *on* $\mathcal{X}$, *and that the proposal distribution is global. As* $t \to \infty$, *we have*

$$P\left\{\lim_{t\to\infty}\theta_{ti} = \text{Const} + \log\left(\int_{E_i}\psi(\boldsymbol{x})d\boldsymbol{x}\right) - \log(\pi_i)\right\} = 1, \quad i = 1, \ldots, m, \tag{17}$$

*where* Const *denotes an arbitrary constant.*

*Remark* A similar theorem has been proved in Liang et al. (2007) for the case that a local proposal distribution is used in the MCMC sampling step of SAMC. Below this theorem is reproved under the condition (6), which allows the sample space to include several separated regions. Hence, this theorem is applicable to ASAMC.

*Proof* To prove this theorem, it suffices to verify that the conditions (A$_1$) to (A$_3$) (given in Sect. 8.2) hold for SAMC. Since the change of the proposal distribution only affects the verification for the condition A$_2$-(i), only the condition A$_2$-(i) is verified below. Other conditions can be verified as in Liang et al. (2007). For the MH kernel, we have

$$\begin{aligned}
K_\theta(\boldsymbol{x}, A) &= \int_A s_\theta(\boldsymbol{x}, \boldsymbol{y})d\boldsymbol{y} + I(\boldsymbol{x} \in A)\left(1 - \int_{\mathcal{X}} s_\theta(\boldsymbol{x}, \boldsymbol{z})d\boldsymbol{z}\right) \\
&\geq \int_A q(\boldsymbol{x}, \boldsymbol{y}) \min\left\{1, \frac{p_\theta(\boldsymbol{y})q(\boldsymbol{y}, \boldsymbol{x})}{p_\theta(\boldsymbol{x})q(\boldsymbol{x}, \boldsymbol{y})}\right\}d\boldsymbol{y} \\
&= \int_A \min\left\{q(\boldsymbol{x}, \boldsymbol{y}), \frac{p_\theta(\boldsymbol{y})q(\boldsymbol{y}, \boldsymbol{x})}{p_\theta(\boldsymbol{x})}\right\}d\boldsymbol{y} \\
&\geq \int_A \min\left\{q(\boldsymbol{x}, \boldsymbol{y}), \frac{p_\theta(\boldsymbol{y})}{\omega^*}\right\}d\boldsymbol{y} \quad \text{(by the minorization condition)} \\
&= \int_A \frac{p_\theta(\boldsymbol{y})}{\omega^*}d\boldsymbol{y} \quad \text{(by definition of } \omega^*\text{)} \\
&\geq \frac{\psi(A)}{\omega^* \int_{\mathcal{X}} \psi(\boldsymbol{x})d\boldsymbol{x}} \\
&= \frac{\psi^*(A)}{\omega^*},
\end{aligned}$$

where $\psi^*(\cdot)$ denotes a normalized measure of $\psi(\cdot)$. Suppose the constraint $\sum_{i=1}^m \int_{E_k} \psi(\boldsymbol{x})d\boldsymbol{x}/e^{\theta_k} = 1$ is imposed on $\theta$, $e^{\theta_k}$ is then bounded above by $\int_{\mathcal{X}} \psi(\boldsymbol{x})d\boldsymbol{x}$. Therefore, the condition

$$\inf_{\theta \in \Theta} K_\theta^l(\boldsymbol{x}, A) \geq \delta\nu(A), \quad \forall \boldsymbol{x} \in \mathcal{X},\ \forall A \in \mathcal{B} \tag{18}$$

is satisfied by choosing $\delta = \frac{1}{\omega^*}$, $l = 1$, and $\nu(\cdot) = \psi^*(\cdot)$. Equation (18) implies that $C = \mathcal{X}$ is a small set. The sample space $\mathcal{X}$ being a small set implies directly the following condition,

$$\sup_{\theta \in \Theta_0} K_\theta^l V^p(\boldsymbol{x}) \leq \lambda V^p(\boldsymbol{x}) + bI(\boldsymbol{x} \in C), \quad \forall \boldsymbol{x} \in \mathcal{X}, \tag{19}$$

by choosing $\Theta_0 = \Theta$, $V(\boldsymbol{x}) = 1$, $l = 1$, $0 < \lambda < 1$, $b = 1 - \lambda$, and $p \in [2, \infty)$. Since $V(\boldsymbol{x})$ has been specified as a constant function, the condition

$$K_\theta V^p(\boldsymbol{x}) \leq \kappa V^p(\boldsymbol{x}) \tag{20}$$

holds automatically by choosing $\kappa \geq 1$. Equations (18), (19) and (20) imply that ($A_2$-i) is satisfied. □

**Lemma 8.1** (Scheffé 1947; Billingsley 1986, p. 218) *Suppose that $F_n(A) = \int_A f_n(\boldsymbol{x})d\boldsymbol{x}$ and $F(A) = \int_A f(\boldsymbol{x})d\boldsymbol{x}$ for densities $f_n(\boldsymbol{x})$ and $f(\boldsymbol{x})$ defined on $\mathcal{X}$. If $f_n(\boldsymbol{x})$ converges to $f(\boldsymbol{x})$ almost surely, then $F_n(A) \longrightarrow F(A)$ as $n \to \infty$ uniformly for any $A \in \mathcal{B}(\mathcal{X})$, where $\mathcal{B}(\mathcal{X})$ denotes the Borel set of the space $\mathcal{X}$.*

**Theorem 8.3** *Let $F_{\theta_t}(\boldsymbol{x})$ denote the cumulative distribution function (CDF) corresponding to the density function $p_{\theta_t}(\boldsymbol{x})$ given in (5), and let $F(\boldsymbol{x})$ denote the CDF corresponding to* *the density function*

$$p_\theta(\boldsymbol{x}) = \sum_{i=1}^{m} \frac{\pi_i \psi(\boldsymbol{x})}{\int_{E_i} \psi(\boldsymbol{x}) d\boldsymbol{x}} I(\boldsymbol{x} \in E_i).$$

*Then we have*

$$\lim_{t\to\infty} F_{\theta_t}(\boldsymbol{x}) = F(\boldsymbol{x}).$$

*Proof* Theorem 8.2 implies that $p_{\theta_t}(\boldsymbol{x}) \to p_\theta(\boldsymbol{x})$ almost surely. Since $p_{\theta_t}(\boldsymbol{x})$ and $p_\theta(\boldsymbol{x})$ are both densities, it follows from Lemma 8.1 that $\lim_{t\to\infty} F_{\theta_t}(\boldsymbol{x}) = F(\boldsymbol{x})$. □

**Lemma 8.2** *Let* $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)$ *be an m-vector,* $\pi_i > 0$, $\sum_{i=1}^m \pi_i = 1$, *and there exists a constant* $\delta > 0$ *such that* $\min_{1\le i\le m} \pi_i - \frac{1}{m}\sum_{i=1}^m \pi_i^2 > \delta$. *Let* $\boldsymbol{p} = (p_1, \ldots, p_m)$ *be a random vector such that* $p_i > 0$ *and* $\sum_{i=1}^m p_i = 1$. *Let* $e_i = p_i - \pi_i$, $i = 1, \ldots, m$, *and suppose that each* $e_i$ *is distributed symmetrically about* 0. *Then there exists a constant* $\lambda$ *such that*

$$E\left\{\sum_{i=1}^m e_i^2 p_i - \left[\sum_{i=1}^m e_i p_i\right]^2\right\} \ge \lambda E\left\{\sum_{i=1}^m e_i^2\right\}. \tag{21}$$

*Proof* By Cauchy–Schwarz theorem, we have

$$\begin{aligned}
&\sum_{i=1}^m e_i^2 p_i - \left[\sum_{i=1}^m e_i p_i\right]^2 - \lambda\sum_{i=1}^m e_i^2 \ge \sum_{i=1}^m e_i^2 p_i - \sum_{i=1}^m e_i^2 \sum_{i=1}^m p_i^2 - \lambda \sum_{i=1}^m e_i^2\\
&= \sum_{i=1}^m \left[\pi_i + e_i - \frac{1}{m}\sum_{j=1}^m (\pi_j + e_j)^2 - \lambda\right] e_i^2\\
&= \sum_{i=1}^m \left[\pi_i - \frac{1}{m}\sum_{j=1}^m \pi_j^2 - \lambda\right] e_i^2 + \left[\sum_{i=1}^m e_i^3 - \frac{2}{m}\sum_{i=1}^m e_i^2 \sum_{i=1}^m \pi_i e_i - \frac{1}{m}\left(\sum_{i=1}^m e_i^2\right)^2\right].
\end{aligned} \tag{22}$$

Taking expectation on both sides of (22), we have

$$E\left\{\sum_{i=1}^m e_i^2 p_i - \left[\sum_{i=1}^m e_i p_i\right]^2 - \lambda \sum_{i=1}^m e_i^2\right\} \ge (\delta - \lambda)\sum_{i=1}^m \sigma_i^2 + o\left(\sum_{i=1}^m \sigma_i^2\right), \tag{23}$$

where $\sigma_i^2$ denotes the variance of $e_i$. It is obvious that $\sigma_i^2 < 1$ for all $i$. Hence, there exist a constant $0 < \lambda < \delta$ such that (21) holds. □

*Remark* In Lemma 8.2, the existence of $\delta$ is obvious when $\boldsymbol{\pi}$ is set to the uniform distribution with $\pi_1 = \cdots = \pi_m = 1/m$. The following theorem concerns the convergence rate of ASAMC. Let $\mathcal{F}_t = \sigma\{\theta_0, \boldsymbol{x}_0; \ldots; \theta_t, \boldsymbol{x}_t\}$ be the sigma-algebra formed by $\theta_0, \boldsymbol{x}_0; \ldots; \theta_t, \boldsymbol{x}_t$. Since $\lim_{t\to\infty} \gamma_t = 0$, the condition (24) is asymptotically satisfied by ASAMC when $t$ becomes large. The same condition has been assumed by Benveniste et al. (1990) (p. 244, Theorem 22) in studying the convergence rate of conventional stochastic approximation MCMC algorithms.

**Theorem 8.4** *Assume the following conditions hold*:

(B$_1$)

$$E\big[H(\theta_t,\boldsymbol{x}_{t+1})-h(\theta_t)|\mathcal{F}_t\big]=0, \tag{24}$$

*where* $h(\theta)=\int_{\mathcal{X}^n}H(\theta,\boldsymbol{x})p(d\boldsymbol{x})$ *for* $\theta\in\Theta$.

(B$_2$) *The difference* $S_i^{(t)}/S^{(t)}-\pi_i$ *is distributed symmetrically about* 0, *where* $S_i^{(t)}=\int_{E_i}\boldsymbol{\psi}(\boldsymbol{x})d\boldsymbol{x}/e^{\theta_{ti}}$ *and* $S^{(t)}=\sum_{i=1}^m S_i^{(t)}$.

*Let* $\boldsymbol{\pi}$ *be chosen such that there exists a constant* $\delta$ *satisfying the inequality* $\min_{1\le i\le m}\pi_i-\sum_{i=1}^m\pi_i^2/m>\delta>0$, *and let the gain factor be chosen as in* (8) *with* $T_0>\delta^{-1/\eta}$. *Define* $v(\theta_t)=\frac{1}{2}\sum_{k=1}^m(S_k^{(t)}/S^{(t)}-\pi_k)^2$, *where* $S_i^{(t)}=\int_{E_i}\psi(\boldsymbol{x})d\boldsymbol{x}/e^{\theta_{ti}}$ *and* $S^{(t)}=\sum_{k=1}^m S_k^{(t)}$. *Then*

$$E[v(\theta_t)]\le\lambda^*\gamma_t^\beta,$$

*for all* $0<\beta\le 1$ *and some constant* $\lambda^*>0$.

*Proof* Let $\nabla(\theta)$ be the vector of the first partial derivatives of $v(\theta)$ and let $\boldsymbol{A}(\theta)$ be the matrix of the second partial derivatives of $v(\theta)$, i.e.,

$$\nabla(\theta)=\left(\frac{\partial v}{\partial\theta_i}\right)\bigg|_\theta,\qquad \boldsymbol{A}(\theta)=\left(\frac{\partial^2 v}{\partial\theta_i\partial\theta_j}\right)\bigg|_\theta.$$

By Taylor's theorem, we have

$$\begin{aligned}v(\theta_{t+1})&=v\big(\theta_t+\gamma_{t+1}H(\theta_t,\boldsymbol{x}_{t+1})\big)\\&=v(\theta_t)+\gamma_{t+1}\langle\nabla v(\theta_t),H(\theta_t,\boldsymbol{x}_{t+1})\rangle\\&\quad+\frac{1}{2}\gamma_{t+1}^2\langle H(\theta_t,\boldsymbol{x}_{t+1}),\boldsymbol{A}(\theta_t+\xi\gamma_{t+1}H(\theta_t,\boldsymbol{x}_{t+1}))H(\theta_t,\boldsymbol{x}_{t+1})\rangle,\end{aligned}$$

where $\xi$ is a real number between 0 and 1. Consequently, we may take conditional expectations on both sides to obtain

$$\begin{aligned}E[v(\theta_{t+1})|\mathcal{F}_t]&=v(\theta_t)+\gamma_{t+1}\langle\nabla(\theta_t),h(\theta_t)\rangle\\&\quad+\frac{1}{2}\gamma_{t+1}^2E[\langle H(\theta_t,\boldsymbol{x}_{t+1}),\boldsymbol{A}(\theta_t+\xi\gamma_{t+1}H(\theta_t,\boldsymbol{x}_{t+1}))H(\theta_t,\boldsymbol{x}_{t+1})\rangle|\mathcal{F}_t].\end{aligned} \tag{25}$$

Let $\Xi_t=\frac{1}{2}E[\langle H(\theta_t,\boldsymbol{x}_{t+1}),\boldsymbol{A}(\theta_t+\xi\gamma_{t+1}H(\theta_t,\boldsymbol{x}_{t+1}))H(\theta_t,\boldsymbol{x}_{t+1})\rangle|\mathcal{F}_t]$. Below we show that $\Xi_t$ is bounded above by a constant.

Taking expectations on both sides of (25), we get

$$E[v(\theta_{t+1})]=E[v(\theta_t)]+\gamma_{t+1}E[\langle\nabla(\theta_t),h(\theta_t)\rangle]+\frac{1}{2}\gamma_{t+1}^2E(\Xi_t). \tag{26}$$

Since in ASAMC the MH kernel leaves the density $p_{\theta_t}(\boldsymbol{x})$ invariant, we have

$$\int_{\mathcal{X}^n}H_i(\theta_t,\boldsymbol{x})p_{\theta_t}(\boldsymbol{x})d\boldsymbol{x}=\frac{\int_{E_i}\boldsymbol{\psi}(\boldsymbol{x})d\boldsymbol{x}/e^{\theta_{ti}}}{\sum_{k=1}^m[\int_{E_k}\boldsymbol{\psi}(\boldsymbol{x})d\boldsymbol{x}/e^{\theta_{tk}}]}-\pi_i=\frac{S_i^{(t)}}{S^{(t)}}-\pi_i,\quad i=1,\dots,m, \tag{27}$$

where $H_i(\theta_t, \boldsymbol{x})$ denotes the $i$th element of the vector $H(\theta_t, \boldsymbol{x})$. By (24), we have

$$h(\theta_t) = \left( \frac{S_1^{(t)}}{S^{(t)}} - \pi_1, \ldots, \frac{S_m^{(t)}}{S^{(t)}} - \pi_m \right)'. \tag{28}$$

It follows from (28) and (31) that

$$\langle \nabla(\theta_t), h(\theta_t) \rangle = -\left\{ \sum_{i=1}^m \left( \frac{S_i^{(t)}}{S^{(t)}} - \pi_i \right)^2 \frac{S_i^{(t)}}{S^{(t)}} - \left( \sum_{i=1}^m \left( \frac{S_i^{(t)}}{S^{(t)}} - \pi_i \right) \frac{S_i^{(t)}}{S^{(t)}} \right)^2 \right\}.$$

Following from Lemma 8.2, there exists a constant $\lambda_*$ such that

$$E[\langle \nabla(\theta_t), h(\theta_t) \rangle] \le -\lambda_* E[v(\theta_t)].$$

This further implies that

$$E[v(\theta_{t+1})] \le (1 - \gamma_{t+1}\lambda_*) E[v(\theta_t)] + \gamma_{t+1}^2 \Xi, \tag{29}$$

where $\Xi$ denotes the upper bound of $\Xi_t$, i.e., $|\Xi_t| \le \Xi$ for all $t$.

It is easy to verify that there exist $\lambda_0$ and $t_0$ such that for $\lambda > \lambda_0$ and $t > t_0$ the sequence $u_t = \lambda \gamma_t^\beta$ satisfies

$$u_{t+1} \ge (1 - \gamma_{t+1}\lambda_*) u_t + \gamma_{t+1}^2 \Xi,$$

provided that $\gamma_t$ is chosen as in (8) with $T_0 > \delta^{-1/\tau}$. Choose $\lambda^* > \lambda_0$ and $t^* > t_0$ such that

$$E[v(\theta_{t^*})] \le \lambda^* \gamma_{t^*}^\beta.$$

It follows immediately by induction on $t$ that the sequence $u_t = \lambda^* \gamma_t^\beta$ satisfies $E[v(\theta_t)] \le u_t$ when $t > t^*$.

Now we show that $\Xi_t$ is bounded above by a constant for all $t$. To simplify notations, in the following we will drop the superscript $t$, denoting $S_i^{(t)}$ by $S_i$, denoting $S^{(t)}$ by $S$, and denoting $\theta_t = (\theta_{t1}, \ldots, \theta_{tm})$ by $\theta = (\theta_1', \ldots, \theta_m')$.

$$\begin{aligned} &\frac{\partial S}{\partial \theta_i'} = \frac{\partial S_i}{\partial \theta_i'} = -S_i, \qquad \frac{\partial S_i}{\partial \theta_j'} = \frac{\partial S_j}{\partial \theta_i'} = 0, \\ &\frac{\partial (\frac{S_i}{S})}{\partial \theta_i'} = -\frac{S_i}{S}\left(1 - \frac{S_i}{S}\right), \qquad \frac{\partial (\frac{S_i}{S})}{\partial \theta_j'} = \frac{\partial (\frac{S_j}{S})}{\partial \theta_i'} = \frac{S_i S_j}{S^2}, \end{aligned} \tag{30}$$

for $i, j = 1, \ldots, m$ and $i \neq j$, and

$$\begin{aligned} \frac{\partial v(\theta')}{\partial \theta_i'} &= \frac{1}{2} \sum_{k=1}^m \frac{\partial (\frac{S_k}{S} - \pi_k)^2}{\partial \theta_i'} = \sum_{j \neq i} \left( \frac{S_j}{S} - \pi_j \right) \frac{S_i S_j}{S^2} - \left( \frac{S_i}{S} - \pi_i \right) \frac{S_i}{S} \left(1 - \frac{S_i}{S}\right) \\ &= \sum_{j=1}^m \left( \frac{S_j}{S} - \pi_j \right) \frac{S_i S_j}{S^2} - \left( \frac{S_i}{S} - \pi_i \right) \frac{S_i}{S} = \mu_\eta \frac{S_i}{S} - \left( \frac{S_i}{S} - \pi_i \right) \frac{S_i}{S}, \end{aligned} \tag{31}$$

for $i = 1, \dots, m$, $\mu_\eta = \sum_{j=1}^{m} (\frac{S_j}{S} - \pi_j)\frac{S_j}{S}$. Further, we have

$$\begin{aligned}\frac{\partial \mu_\eta}{\partial \theta_i'} &= \sum_{k \neq i} \left[ 2\frac{S_i S_k^2}{S^3} - \pi_k \frac{S_i S_k}{S^2} \right] - 2\frac{S_i^2}{S^2}\left(1 - \frac{S_i}{S}\right) + \pi_i \frac{S_i}{S}\left(1 - \frac{S_i}{S}\right) \\ &= \frac{S_i}{S}\left[ \sum_{k=1}^{m} \frac{S_k^2}{S^2} + \mu_\eta - 2\frac{S_i}{S} + \pi_i \right], \end{aligned} \tag{32}$$

$$\begin{aligned}\frac{\partial^2 v(\theta)}{\partial (\theta_i')^2} &= \frac{\partial[\mu_\eta \frac{S_i}{S} - (\frac{S_i}{S} - \pi_i)\frac{S_i}{S}]}{\partial \theta_i'} \\ &= -\mu_\eta \frac{S_i}{S}\left(1 - \frac{S_i}{S}\right) + \frac{S_i}{S}\frac{\partial \mu_\eta}{\partial \theta_i'} + 2\frac{S_i^2}{S^2}\left(1 - \frac{S_i}{S}\right) - \pi_i \frac{S_i}{S}\left(1 - \frac{S_i}{S}\right) \\ &= \frac{S_i^2}{S^2}\left[ \sum_{k=1}^{m} \frac{S_k^2}{S^2} + 2\mu_\eta - 4\frac{S_i}{S} + 2\pi_i + 2 \right] - \frac{S_i}{S}(\mu_\eta + \pi_i), \end{aligned} \tag{33}$$

and

$$\begin{aligned}\frac{\partial^2 v(\theta)}{\partial \theta_i' \partial \theta_j'} &= \frac{\partial[\mu_\eta \frac{S_i}{S} - (\frac{S_i}{S} - \pi_i)\frac{S_i}{S}]}{\partial \theta_j'} = \frac{S_i S_j}{S^2}\mu_\eta + \frac{S_i}{S}\frac{\partial \mu_\eta}{\partial \theta_j'} - \left[ 2\frac{S_i^2 S_j}{S^3} - \pi_i \frac{S_i S_j}{S^2} \right] \\ &= \frac{S_i S_j}{S^2}\left[ \sum_{k=1}^{m} \frac{S_k^2}{S^2} + 2\mu_\eta - 2\frac{S_i}{S} - 2\frac{S_j}{S} + \pi_i + \pi_j \right]. \end{aligned} \tag{34}$$

Since $0 \le S_i/S \le 1$, $0 \le \pi_i \le 1$ and $|\mu_\eta| \le 1$, both $|\partial^2 v(\theta)/\partial(\theta_i')^2|$ and $|\partial^2 v(\theta)/\partial\theta_i'\partial\theta_j'|$ are bounded above by a constant. Each element of $H(\theta_t, \boldsymbol{x}_{t+1})$ lies between $-1$ and 1. Therefore, $\Xi_t$ is bounded above by a constant for all $t$. □

## References

Abunawass, A. M., & Owen, C. B. (1993). A statistical analysis of the effect of noise injection during neural network training. *SPIE Proceedings*, *1966*, 362–371.

Amato, S., Apolloni, B., Caporali, G., Madesani, U., & Zanaboni, A. (1991). Simulated annealing approach in back-propagation. *Neurocomputing*, *3*, 207–220.

Andrieu, C., Moulines, E., & Priouret, P. (2005). Stability of Stochastic Approximation Under Verifiable Conditions. *SIAM J. Control and Optimization*, *44*, 283–312.

Barron, A. (1993). Universal approximation bounds for superposition of a sigmoidal function. *IEEE Transactions on Information Theory*, *3*, 930–945.

Baum, E. B., & Lang, K. J. (1991). Constructing hidden units using examples and queries. In *Advances in neural information processing systems* (Vol. 3, pp. 904–910). San Mateo: Kaufmann.

Benveniste, A., Métivier, M., & Priouret, P. (1990). *Adaptive algorithms and stochastic approximations*. New York: Springer.

Bharath, B., & Borkar, V. S. (1999). Stochastic approximation algorithms: overview and recent trends. *Sadhana*, *24*, 425–452.

Billingsley, P. (1986). *Probability and measure* (2nd ed.). New York: Wiley.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Oxford University Press.

Broyden, C. G. (1970a). The convergence of a class of double rank minimization algorithms, part I. *Journal of the Institute of Mathematics and Applications*, *6*, 76–90.

Broyden, C. G. (1970b). The convergence of a class of double rank minimization algorithms, part II. *Journal of the Institute of Mathematics and Applications*, *6*, 222–231.

Chawla, D., Li, L., & Scott, S. (2004). On approximating weighted sums with exponentially many terms. *Journal of Computer and System Sciences*, *69*, 196–234.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *3*, 303–314.

Davidon, W. C. (1959). *Variable metric method for minimization*. AEC Res. and Dev. Report ANL-5990.

de Freitas, N., Niranjan, M., Gee, A. H., & Doucet, A. (2000). Sequential Monte Carlo methods to train neural network models. *Neural Computation*, *12*, 955–993.

Delyon, B., Lavielle, M., & Moulines, E. (1999). Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics*, *27*, 94–128.

Erland, S. (2003). *Adaptive Markov chain Monte Carlo review*. Technical Report, Department of Mathematical Science, Norwegian University of Science and Technology.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* (Vol. 2, pp. 524–532). San Mateo: Kaufmann.

Flanagan, J. A. (1997). Analyzing a self-organizing algorithm. *Neural Networks*, *10*, 875–883.

Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, *13*, 317–322.

Fletcher, R. (1987). *Practical methods of optimization* (2nd ed.). New York: Wiley.

Fletcher, R., & Powell, M. J. D. (1963). A rapidly convergent descent method for minimization. *The Computer Journal*, *6*, 163–168.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, *2*, 183–192.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721–741.

Gelfand, A. E., & Banerjee, S. (1998). *Computing marginal posterior modes using stochastic approximation*. Technical report, Department of Statistics, University of Connecticut.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, & machine learning*, Reading: Addison–Wesley.

Goldfarb, D. (1970). A family of variable metric methods derived by variational means. *Mathematics of Computation*, *24*, 23–26.

Goswami, G. R., & Liu, J. S. (2005) *On real parameter evolutionary Monte Carlo algorithm*. Technical Report, Harvard University.

Gu, M. G., & Kong, F. H. (1998). A stochastic approximation algorithm with Markov chain Monte Carlo method for incomplete data estimation problems. *Proceedings of the National Academy of Sciences USA*, *95*, 7270–7274.

Gu, M. G., & Zhu, H. T. (2001). Maximum likelihood estimation for spatial models by Markov chain Monte Carlo stochastic approximation. *Journal of the Royal Statistical Society, Series B*, *63*, 339–355.

Hanson, S. J. (1991). Behavioral diversity, search and stochastic connectionist systems. In M. Commons, S. Grossberg, & J. Staddon (Eds.), *Neural network models of conditioning and action* (pp. 295–345). Mahwah: Erlbaum.

Harth, E., & Tzanakou, E. (1974). Alopex: a stochastic method for determining visual receptive fields. *Vision Research*, *14*, 1475–1482.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York: Springer.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *57*, 97–109.

Haykin, S. (1999). *Neural networks: a comprehensive foundation* (2nd ed.). New York: Prentice Hall.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Holley, R. A., Kusuoka, S., & Stroock, D. (1989). Asymptotic of the spectral gap with applications to the theory of simulated annealing. *Journal of Functional Analysis*, *83*, 333–347.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359–366.

Ingman, D., & Merlis, Y. (1991). Local minimization escape using thermodynamic properties of neural networks. *Neural Networks*, *4*, 395–404.

Jerrum, M., & Sinclair, A. (1997). The Markov chain Monte Carlo method: an approach to approximate counting and integration. In D. S. Hochbaum (Ed.), *Approximation algorithms for NP-hard problems* (pp. 482–520). Boston: PWS Publishing Company.

Jasra, A., Stephens, D. A., & Holmes, C. C. (2007, to appear). On population-based simulation for static inference. *Statistics and Computing*.

Karlin, S., & Taylor, H. M. (1998). *An introduction to stochastic modeling*, (3rd ed.). Orlando: Academic Press.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the Institute of Electrical and Electronics Engineers*, *78*, 1464–1480.

Lang, K. J., & Witbrock, M. J. (1989). Learning to tell two spirals apart. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), *Proceedings of the 1988 connectionist models* (pp. 52–59). San Mateo: Kaufmann.

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathematics*, *2*, 164–168.

Liang, F. (2003). An effective Bayesian neural network classifier with a comparison study to support vector machine. *Neural Computation*, *15*, 1959–1989.

Liang, F. (2005a). Generalized Wang-Landau algorithm for Monte Carlo computation. *Journal of the American Statistical Association*, *100*, 1311–1327.

Liang, F. (2005b). Evidence evaluation for Bayesian neural networks using contour Monte Carlo. *Neural Computation*, *17*, 1385–1410.

Liang, F., & Wong, W. H. (2001). Real parameter evolutionary Monte Carlo with applications in Bayesian mixture models. *Journal of the American Statistical Association*, *96*, 653–666.

Liang, F., Liu, C., & Carroll, R. J. (2007). Stochastic approximation in Monte Carlo computation. *Journal of the American Statistical Association*, *102*, 305–320.

MacKay, D. J. C. (1992a). A practical Bayesian framework for backprop networks. *Neural Computation*, *4*, 448–472.

MacKay, D. J. C. (1992b). The evidence framework applied to classification problems. *Neural Computation*, *4*, 720–736.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society of Industrial and Applied Mathematics*, *11*, 431–441.

Mengersen, K. L., & Tweedie, R. L. (1996). Rates of convergence of the Hastings and Metropolis algorithms. *The Annals of Statistics*, *24*, 101–121.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, *21*, 1087–1091.

Meyn, S. P., & Tweedie, R. L. (1993). *Markov chains and stochastic stability*. London: Springer.

Mulier, F. M., & Cherkassky, V. S. (1995). Statistical analysis of self-organization. *Neural Networks*, *8*, 717–727.

Müller, P., & Insua, D. R. (1998). Issues in Bayesian analysis of neural network models. *Neural Computation*, *10*, 749–770.

Neal, R. M. (1996). *Bayesian learning for neural networks*. New York: Springer.

Owen, C. B., & Abunawass, A. M. (1993). Applications of simulated annealing to the back-propagation model improves convergence. *SPIE Proceedings*, *1966*, 269–276.

Perrone, M. P. (1993). *Improving regression estimation: averaging methods for variance reduction with extension, to general convex measure optimization*. PhD thesis, Brown University, Rhode Island.

Polak, E. (1971). *Computational methods in optimization: a unified approach*. New York: Academic Press.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, *22*, 400–407.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by back-propagating errors. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: explorations in the microstructure of cognition* (Vol. 1, pp. 318–362). Cambridge: MIT Press.

Sastry, P. S., Magesh, M., & Unnikrishnan, K. P. (2002). Two timescale analysis of the Alopex algorithm for optimization. *Neural Computation*, *14*, 2729–2750.

Scheffé, H. (1947). A useful convergence theorem for probability distributions. *Annals of Mathematical Statistics*, *18*, 434–438.

Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, *24*, 647–656.

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In R. A. Greenes (Ed.), *Proceedings pf the symposium on computer applications in medical care* (pp. 261–265). Los Alamitos: IEEE Computer Society Press.

Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, *37*, 332–341.

Szu, H. (1986). Fast simulated annealing. In *AIP conference proceedings: Vol. 151. Neural network for computing*, Snowbird, UT.

Tang, Z., Wang, X., Tamura, H., & Ishii, M. (2003). An algorithm of supervised learning for multilayer neural networks. *Neural Computation*, *15*, 1125–1142.

Tesauro, G., & Janssens, B. (1988). Scaling relations in back-propagation learning. *Complex System*, *2*, 39–44.

van Rooij, A. J. F., Jain, L. C., & Johnson, R. P. (1996). *Neural network training using genetic algorithms*. Singapore: World Scientific.

von Lehmen, A., Paek, E. G., Liao, P. F., Marrakchi, A., & Patel, J. S. (1988). Factors influencing learning by back-propagation. In *Proceedings of IEEE international conference on neural networks* (pp. 335–341). New York: IEEE Press.

Wang, F., & Landau, D. P. (2001). Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical Review Letters*, *86*, 2050–2053.

Wang, C., & Principe, J. C. (1999). Training neural networks with additive noise in the desired signal. *IEEE Transactions on Neural Networks*, *10*, 1511–1517.

Wong, W. H., & Liang, F. (1997). Dynamic weighting in Monte Carlo and optimization. *Proceedings of the National Academy of Sciences USA*, *94*, 14220–14224.

Wouwer, A. V., Renotte, C., & Remy, M. (1999) On the use of simultaneous perturbation stochastic approximation for neural network training. In *Proceedings of the American control conference* (pp. 388–392), San Diego, CA.