Stochastic Approximation Monte Carlo for MLP Learning

Faming Liang
Texas A&M University, USA

INTRODUCTION

Over the past several decades, multilayer perceptrons (MLPs) have achieved increased popularity among scientists, engineers, and other professionals as tools for knowledge representation. Unfortunately, there is no a universal architecture which is suitable for all problems. Even with the correct architecture, frustrating problems of connection weights training still remain due to the rugged nature of the energy landscape of MLPs. The energy function often refers to the sum-of-square error function for conventional MLPs and the negative log-posterior density function for Bayesian MLPs.

This article presents a Monte Carlo method that can be used for MLP learning. The main focus is on how to apply the method to train connection weights for MLPs. How to apply the method to choose the optimal architecture and to make predictions for future values will also be discussed, but within the Bayesian framework.

BACKGROUND

As known by many researchers, the energy landscape of an MLP is often rugged. The gradient-based training algorithms, such as back-propagation (Rumelhart et al., 1986), conjugate gradient, Newton's method, and the BFGS algorithm (Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970), tend to converge to a local minimum near the starting point, rendering the training data learned insufficiently. To reduce the chance of converging to local minima, a number of variants of these algorithms have been proposed based on the idea of perturbation (von Lehmen et al., 1988, Tang et al., 2003 and references therein). In practice, the effects of these perturbations are usually limited, which only delay the learning process converging to local minima a reasonable number of iterations (Ingman & Merlis, 1991).

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 & Abunawass (1993) show that for complex learning tasks, SA has a better chance to converge to a global minimum than have the gradient-based algorithms. Geman & Geman (1984) show that the global minimum can be reached by SA with probability 1 if the temperature decreases at a logarithmic rate of $O(1/\log t)$, where t denotes 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).

Other stochastic algorithms that have been used in MLP training include the genetic algorithm (Goldberg, 1989) and Markov chain Monte Carlo (MCMC). Although the genetic algorithm works well for some problems, see, e.g., van Rooij et al. (1996), there is no theory to support its convergence to global minima. MCMC algorithms are mainly used for Bayesian MLPs (MacKay, 1992a, Neal, 1996, Muller & Insua, 1998, de Freitas et al., 2000, Liang, 2003, 2005a,2005b), which will be discussed later.

MAIN FOCUS OF THE CHAPTER

This article presents how the stochastic approximation Monte Carlo (SAMC) (Liang et al., 2007) algorithm can be used for MLP learning, including training, prediction and architecture selection.

A Brief Review for the SAMC Algorithm

Suppose that we are working with the Boltzmann distribution,

$$p(x) = \frac{1}{Z} e^{-U(x)/\tau}, \quad x \in \Omega, \tag{1}$$

where Z is the normalizing constant, $U(x)$ is the energy function, τ is the temperature, and Ω is the sample space. Without loss of generality, we assume that Ω is compact. For MLPs, x denotes the vector of connection weights, and Ω can be restricted to a hyper-rectangle $[-B_\Omega, B_\Omega]^{\dim(\Omega)}$, where B_Ω is a large number such that Ω includes at least a global minimum of $U(x)$. Furthermore, we assume that the sample space can be partitioned according to the energy function into m disjoint subregions: $E_1 = \{x: U(x) \le u_1\}$, $E_2 = \{x: u_1 < U(x) \le u_2\}, \ldots, E_{m-1} = \{x: u_{m-2} < U(x) \le u_{m-1}\}$, and $E_m = \{x: U(x) > u_{m-1}\}$, where $u_1, \ldots, u_{m-1}$ are pre-specified real numbers. SAMC seeks to draw samples from each subregion with a pre-specified frequency. If this goal can be achieved, then the local-trap problem can be avoided successfully. Let x_{t+1} denote a sample simulated from the distribution

$$p_{\theta_t}(x) \propto \sum_{i=1}^{m} \frac{\Psi(x)}{e^{\theta_{ti}}} I(x \in E_i) \tag{2}$$

using the Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953, Hastings, 1970), where $\Psi(x) = e^{-U(x)/\tau}$ and $\theta_t = (\theta_{t1}, \ldots, \theta_{tm})$ is an m-vector in a space Θ. For simplicity, we assume that Θ is compact, e.g., $\Theta = [-B_\Theta, B_\Theta]^{\dim(0)}$ with B_Θ being a large number. Since adding to or subtracting from θ_t a constant will not change $p_{\theta_t}(x)$, θ_t can be kept in the compact set in simulations by adjusting with an additive constant. Let the proposal distribution, $q(x, y)$, of the MH moves satisfy the minorisation condition (Mengersen & Tweedie, 1996), i.e.,

$$\sup_{\theta \in \Omega} \sup_{x,y \in \Omega} \frac{p_\theta(y)}{q(x,y)} < \infty \tag{3}$$

Since Ω is compact, a sufficient design for the minorisation condition is to choose $q(x, y)$ as a global proposal distribution. A proposal distribution is said global if $q(x, y) > 0$ for all $x, y \in \Omega$. For MLPs, $q(x, y)$ can be chosen as a random walk Gaussian proposal, $y \sim N(x, \sigma^2 I)$, where I is an identity matrix and σ^2 is calibrated such that the MH moves have a desired acceptance rate. As discussed later, restricting the proposal distribution to be global ensures the convergence of the annealing SAMC algorithm to the global energy minima.

Let $\{\gamma_t\}$ be a positive non-decreasing sequence satisfying the conditions:

i. $\sum_{t=0}^{\infty} \gamma_t = \infty$,

ii. $\sum_{t=0}^{\infty} \gamma_t^\delta < \infty$

for some $\delta \in (1, 2)$. For example, one can set

$$\gamma_t = \left(\frac{t_0}{\max(t_0, t)} \right)^\eta \tag{4}$$

for some 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 in the presence of multiple local minima. Let $\pi = (\pi_1, \ldots, \pi_m)$ be an m-vector with $0 < \pi_i < 1$ and

$$\sum_{i=1}^{m} \pi_i = 1,$$

which defines a desired sampling frequency distribution on the subregions. With the above notations, an iteration of SAMC can be described as follows.

SAMC Algorithm

a. Generate $x_{t+1} \sim K_{\theta_t}(x_t, .)$ with a single MH step:
   1. Generate y according to the proposal distribution $q(x_t, y)$.
   2. Calculate the ratio

$$r = e^{\theta_{t,J(x_t)} - \theta_{t,J(y)}} \frac{\Psi(y)}{\Psi(x_t)} \frac{q(y, x_t)}{q(x_t, y)},$$

   where $J(x)$ denote the index of the subregion that the sample x belongs to.
   3. Accept the proposal with probability $\min(1, r)$. If it is accepted, set $x_{t+1} = y$; otherwise, set $x_{t+1} = x_t$.