THEORY OF MONTE CARLO SAMPLING-BASED ALOPEX ALGORITHMS FOR NEURAL NETWORKS

Zhe Chen [1], *Simon Haykin* [1] *and Suzanna Becker* [2]

[1] Communications Research Lab, [2] Department of Psychology, McMaster University,
Hamilton, Ontario, Canada L8S 4K1
zhechen@soma.crl.mcmaster.ca, haykin@mcmaster.ca

ABSTRACT

We propose two novel Monte Carlo sampling-based Alopex algorithms for training neural networks. The proposed algorithms naturally combine the sequential Monte Carlo estimation and Alopex-like procedure for gradient-free optimization, and the learning proceeds within the recursive Bayesian estimation framework. Experimental results on various problems show encouraging convergence results.

1. INTRODUCTION

The idea of using Monte Carlo methods for optimization and machine learning is not new; genetic algorithms and simulated annealing are two representative examples. The Alopex (ALgorithm Of Pattern EXtraction) algorithm [6, 10-12] is a gradient-free, stochastic optimization method. Because it only requires the calculation of the objective function and not its derivatives, it has the advantage of low computational complexity, as well as being applicable within any arbitrary architecture. However, like simulated annealing, its major drawback is its slow convergence speed. Bia [1] therefore proposed a deterministic version of the Alopex algorithm called Alopex-B, which converges much more quickly. Recently, much effort has been devoted to applying Monte Carlo methods to various optimization problems, such as Gibbs sampling, Monte Carlo EM, dynamic weighting [14], Fisher scoring [2], and HySIR [4]. The motivation of our work is to formulate the optimization problem in a probabilistic sampling framework, using Alopex to compute the parameter updates. We thereby retain the computational advantages of a gradient-free method while taking advantage of the stochastic aspect of sampling methods. Usually, deterministic algorithms converge faster; however, stochastic algorithms are less likely to be stuck in local minima due to the employment of random perturbations. In particular, the recently-developed sequential Monte Carlo method (a.k.a. particle filter) is particularly attractive since it provides an efficient on-line Bayesian estimation framework via the sampling method. In this paper, we propose two novel Monte Carlo sampling-based Alopex algorithms by combining a quasi-deterministic Alopex-B algorithm with particle filtering. Our algorithms can be used for either on-line or off-line neural network learning as well as other gradient-free optimization problems.

The work was supported by the NSERC of Canada.

2. THE ALOPEX AND ALOPEX-B ALGORITHMS

The Alopex algorithm is a correlation-based *gradient-free* optimization technique. It was initially used in visual research [6, 10], and later applied to many other problems in training neural networks [11, 12, 1] and decision trees [8, 9]. It behaves asymptotically like a gradient-descent algorithm [8], though it does not require explicit calculation of the gradient. Suppose we are trying to optimize an objective (energy) function $E(\boldsymbol{\theta}) : \mathbb{R}^N \to \mathbb{R}$, given a model (e.g., neural network) parameterized by an unknown parameter vector $\boldsymbol{\theta}$. Throughout the paper let t denote the time index. The Alopex algorithm [12] is summarized as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \boldsymbol{\xi}_t, \tag{1}$$

where η $(0 < \eta < 1)$ is a step-size scalar, $\boldsymbol{\xi}_t$ is a random vector with its j-th entry determined element-wise by

$$\begin{aligned}
\xi_j(t) &= \operatorname{sgn}(u_j - p_j(t)), \quad u_j \sim \mathcal{U}(0,1), &(2)\\
p_j(t) &= \phi(c_j(t)/T(t)) = \frac{1}{1+\exp(-c_j(t)/T(t))}, &(3)\\
c_j(t) &= \Delta\theta_j(t)\Delta E(t), &(4)\\
\Delta\theta_j(t) &= \theta_j(t) - \theta_j(t-1), &(5)\\
\Delta E(t) &= E(t) - E(t-1), &(6)
\end{aligned}$$

where u is a uniform random variable; $\operatorname{sgn}(\cdot)$ is a signed function; $\phi(\cdot)$ is a logistic sigmoid function; and $T(t)$ is an annealing parameter that plays a similar role to "temperature" in simulated annealing. The key term is $c_j(t)$, which correlates changes in the objective function with parameter vector changes. The algorithm starts with a randomly initialized parameter $\boldsymbol{\theta}_0$ and stops when $E(t)$ is sufficiently small. The stochastic component $\boldsymbol{\xi}_t$ helps the algorithm escape from local minima.

In [1], Bia developed a new Alopex-type algorithm called Alopex-B. Unlike Alopex, Alopex-B does not employ any annealing scheme and uses fewer parameters, with reported simpler implementation and faster convergence. Consistent with the above notation, our proposed Alopex-B algorithm (slightly different from [1]) reads:

$$\begin{aligned}
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \eta\boldsymbol{\xi}_t + \gamma\Delta\boldsymbol{\theta}_t\Delta E(t), &(7)\\
p_j(t) &= \phi(C_j(t)), &(8)\\
C_j(t) &= \frac{\operatorname{sgn}(\Delta\theta_j(t))\Delta E(t)}{\sum_{k=2}^{t}\lambda(\lambda-1)^{t-k}|\Delta E(k-1)|}, &(9)
\end{aligned}$$

where λ is a forgetting parameter usually in the range $[0.35, 0.7]$, and γ is a step-size parameter in $[0.05, 0.1]$. In preliminary simulations we found that although Alopex-B converges more quickly, it gets trapped in local minima more frequently. This motivated our development of the Monte Carlo sampling-based Alopex algorithm.

3. SEQUENTIAL MONTE CARLO ESTIMATION

Let us formulate a generic parameter estimation problem in a state-space form:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\nu}_t \tag{10a}$$
$$\mathbf{y}_t = f(\boldsymbol{\theta}_t, \mathbf{x}_t) + \mathbf{v}_t \tag{10b}$$

where the nonlinear model, parameterized by $\boldsymbol{\theta}$, determines the mapping $f : X \mapsto Y$, given a number of inputs $\mathbf{x}_t$ and outputs $\mathbf{y}_t$. $\boldsymbol{\nu}_t$ and $\mathbf{v}_t$ are dynamical noise and measurement noise, respectively. In our paper, f can be a multilayer perceptron (MLP) network or a recurrent MLP (RMLP) network.

In the sequential Monte Carlo framework, $\boldsymbol{\theta}_t$ is estimated via particle filtering [5, 3]. Particle filters use a number of random samples called "particles", sampled directly from the state space, to represent the posterior, and update the posterior by involving the new observations; the "particle system" is properly *located, weighted*, and *propagated* recursively according to Bayes' rule. Among many variations, one of the most popular particle filters is the sampling-importance-resampling (SIR) filter. The basic principle of the SIR filter is using the importance sampling trick:

$$\int f(\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} = \int f(\boldsymbol{\theta})\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}q(\boldsymbol{\theta})d\boldsymbol{\theta}, \tag{11}$$

where q and p are proposal and target (posterior) densities, respectively. Given a number of i.i.d. samples $\boldsymbol{\theta}^{(i)} \sim q(\boldsymbol{\theta})$, we can estimate the mean of $f(\boldsymbol{\theta})$ by:

$$\mathbb{E}_p[f] \approx \frac{1}{N_p}\sum_{i=1}^{N_p} W(\boldsymbol{\theta}^{(i)})f(\boldsymbol{\theta}^{(i)}) \equiv \hat{f}, \tag{12}$$

where $W(\boldsymbol{\theta}^{(i)}) = p(\boldsymbol{\theta}^{(i)})/q(\boldsymbol{\theta}^{(i)})$ are called the *importance weights*. If the normalizing factor of $p(\boldsymbol{\theta})$ is not known, then $W(\boldsymbol{\theta}^{(i)}) \propto p(\boldsymbol{\theta}^{(i)})/q(\boldsymbol{\theta}^{(i)})$. To ensure that $\sum_{i=1}^{N_p} W(\boldsymbol{\theta}^{(i)}) = 1$, we can normalize the importance weights:

$$\hat{f} = \frac{\frac{1}{N_p}\sum_{i=1}^{N_p} W(\boldsymbol{\theta}^{(i)})f(\boldsymbol{\theta}^{(i)})}{\frac{1}{N_p}\sum_{j=1}^{N_p} W(\boldsymbol{\theta}^{(j)})} \equiv \sum_{i=1}^{N_p} \tilde{W}(\boldsymbol{\theta}^{(i)})f(\boldsymbol{\theta}^{(i)}),$$

where $\tilde{W}(\boldsymbol{\theta}^{(i)}) = \frac{W(\boldsymbol{\theta}^{(i)})}{\sum_{j=1}^{N_p} W(\boldsymbol{\theta}^{(j)})}$. By choosing a factorized proposal distribution, the importance weights can be updated recursively [5]:

$$W_t^{(i)} = W_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\boldsymbol{\theta}_t^{(i)}, \mathbf{x}_t)p(\boldsymbol{\theta}_t^{(i)}|\boldsymbol{\theta}_{t-1}^{(i)})}{q(\boldsymbol{\theta}_t^{(i)}|\boldsymbol{\theta}_{0:t-1}^{(i)}, \mathbf{y}_t)}. \tag{13}$$

When the proposal is taken as the prior, the importance weights turn out to be proportional to the likelihood. A well known intrinsic problem for the SIR filter is that as time increases, the distribution of the importance weights becomes more and more skewed; after some iterations, only very few particles have non-zero importance weights. This phenomenon is often called *weight degeneracy* or *sample impoverishment*. One empirical measure of sample efficiency is the variance of the importance weights [5]: $\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_p}(\tilde{W}_t^{(i)})^2}$. We also suggest another empirical efficiency measure, which is the Kullback-Leibler (KL) divergence between the proposal and target density, denoted by $D(q\|p)$. Given N_p samples drawn from proposal q, $D(q\|p)$ is approximated by:

$$\begin{aligned} D(q\|p) &= \mathbb{E}_q\Big[\log\frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\Big] \approx \frac{1}{N_p}\sum_{i=1}^{N_p}\log\frac{q(\boldsymbol{\theta}^{(i)})}{p(\boldsymbol{\theta}^{(i)})} \\ &= -\frac{1}{N_p}\sum_{i=1}^{N_p}\log(W(\boldsymbol{\theta}^{(i)})), \end{aligned} \tag{14}$$

where $\boldsymbol{\theta}^{(i)} \sim q(\boldsymbol{\theta})$. When $q = p$ and $W(\boldsymbol{\theta}^{(i)}) = 1$ for all i, $D(q\|p) = 0$. Since $D(q\|p) \geq 0$, $-\log(W(\boldsymbol{\theta}^{(i)}))$ should be non-negative. In practice, we instead calculate the logarithm of the normalized importance weights $\hat{N}_{\text{KL}} = -\frac{1}{N_p}\sum_{i=1}^{N_p}\log(\tilde{W}(\boldsymbol{\theta}^{(i)}))$, which achieves the minimum value $\hat{N}_{\text{KL}}^{\min} = \log(N_p)$ when all $\tilde{W}(\boldsymbol{\theta}^{(i)}) = 1/N_p$. Our previous studies have confirmed that $\hat{N}_{\text{KL}}$ is a good measure that is also consistent with $\hat{N}_{eff}$: when $\hat{N}_{\text{KL}}$ is small (large), $\hat{N}_{eff}$ is often large (small); and vice versa.

4. MONTE CARLO SAMPLING-BASED ALOPEX ALGORITHM

In neural network learning, in order to draw samples from the proposal $q(\boldsymbol{\theta}_t^{(i)}|\boldsymbol{\theta}_{0:t-1}^{(i)}, \mathbf{y}_t)$, we normally need to calculate the gradient of the cost function w.r.t. $\boldsymbol{\theta}$ (e.g., [4]). When the gradient (Jacobian matrix) is expensive to compute or the cost function is non-differentiable, we can avoid this by using an Alopex-like algorithm to update/propagate samples. The intuition is that by using a number of particles instead of only one to explore the hypothesis space, it is less likely to be stuck in local minima. Besides, by using an Alopex-like algorithm to produce samples, it not only avoids the tedious calculation of gradient, but also serves as a good proposal since it implicitly incorporates all of the information (namely, the proposal has a form $q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \mathbf{y}_t, \Delta\boldsymbol{\theta}_t, \Delta E(t))$). Another advantage is that the optimization is done within the recursive Bayesian estimation framework; the algorithm consists of both stochastic (sampling) and quasi-deterministic (Alopex-B) components, which makes it powerful.

In order to avoid the "blind" random walk behavior, we use a "relaxation" model [7] in place of (10a)

$$\boldsymbol{\theta}_{t+1}^{(i)} = \boldsymbol{\mu}_t + \alpha(\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\mu}_t) + \sqrt{1-\alpha^2}\sigma\boldsymbol{\nu}_t \tag{15}$$

where $\boldsymbol{\mu}_t = \sum_{i=1}^{N_p}\tilde{W}_t^{(i)}\boldsymbol{\theta}_t^{(i)}$; the noise vector $\boldsymbol{\nu}_t$ is standard Gaussian distributed: $\boldsymbol{\nu}_t \sim \mathcal{N}(0, \mathbf{I})$, σ is the standard deviation controlling the degree of the variation of $\boldsymbol{\theta}$, which often requires some prior knowledge of the problem. Relaxing parameter $\alpha \in [-1, 1]$ controls the degree of over-relaxation (or under-relaxation): when $\alpha = -1$, (15) reduces to an extreme over-relaxation $\boldsymbol{\theta}_{t+1}^{(i)} = 2\boldsymbol{\mu}_t - \boldsymbol{\theta}_t^{(i)}$; when $\alpha = 0$, (15) reduces to a random walk $\boldsymbol{\theta}_{t+1}^{(i)} = \boldsymbol{\mu}_t + \sigma\boldsymbol{\nu}_t$; when $\alpha = 1$, (15) reduces to a stationary point $\boldsymbol{\theta}_{t+1}^{(i)} = \boldsymbol{\theta}_t^{(i)}$. In summary, our first proposed sampling-based Alopex algorithm (termed Algorithm-1 hereafter) is as follows:

1. For $i = 1, \cdots, N_p$, initialize $\boldsymbol{\theta}_0^{(i)} \sim p(\boldsymbol{\theta}_0)$, $W_0^{(i)} = 1/N_p$.
2. Predict $\boldsymbol{\theta}_t^{(i)}$ from (15).

3. Update the samples $\boldsymbol{\theta}_t^{(i)}$ via the Alopex-B algorithm (7)-(9).
4. Evaluate importance weights $W_t^{(i)} = W_{t-1}^{(i)} p(\mathbf{y}_t|\boldsymbol{\theta}_t^{(i)})$ and $\tilde{W}_t^{(i)} = W_t^{(i)} / \sum_{j=1}^{N_p} W_t^{(j)}$.
5. Calculate $\hat{N}_{eff}$ and $\hat{N}_{\text{KL}}$, if $\hat{N}_{eff} > 0.8 N_p$ or $\hat{N}_{\text{KL}} > \kappa \log(N_p)$ ($\kappa = 3 \sim 5$), go to step 7; otherwise go to step 6.
6. (Systematic) Resampling [3, 5]: generate a new particle set $\{\boldsymbol{\theta}_t^{(j)}\}$ and reset the weights $\tilde{W}_t^{(j)} = 1/N_p$.
7. Repeat steps 2-5.[1]

Remarks: Note that when $N_p = 1$, Algorithm-1 reduces to a generalized form of Alopex-B, which consists of an additional randomness through (15). Further, there is no reason why we cannot use specific $\alpha^{(i)}$ for different $\boldsymbol{\theta}^{(i)}$; α might be also time-varying, but we didn't investigate these issues in the current paper. We fixed α for each specific problem in the experiments, but the optimal α varies from one problem to another.

It is worthwhile to compare Algorithm-1 with other Monte Carlo sampling-based optimization algorithms, e.g., Fisher-scoring [2] and HySIR [4] for training MLP networks. By avoiding the calculation of the Jacobian matrix, the complexity of our algorithm is much lower than these two algorithms. It is also simpler than another Monte Carlo sampling-based derivative-free estimation technique: the unscented particle filter (UPF), which is typically of complexity $\mathcal{O}(N_p N^3)$ [13]. Our algorithm is suitable for either on-line or off-line learning, and applicable to both sequential and non-sequential data.

We also propose another sampling-based Alopex algorithm (hereafter termed Algorithm-2) that is motivated by the hybrid Monte Carlo (HMC) method (e.g, see [7]). The idea of HMC is to augment the state space $\boldsymbol{\theta}$ with a momentum variable $\boldsymbol{\rho}$ for sampling. The energy-conserving Hamiltonian dynamics is defined as

$$H(\boldsymbol{\theta}, \boldsymbol{\rho}) = E(\boldsymbol{\theta}) + K(\boldsymbol{\rho}), \tag{16}$$

where $E(\boldsymbol{\theta})$ is the *potential energy* function, whereas $K(\boldsymbol{\rho}) = \boldsymbol{\rho}^T \boldsymbol{\rho}/2$ is the *kinetic energy*. The samples are drawn from the joint distribution

$$\begin{aligned} P_H(\boldsymbol{\theta}, \boldsymbol{\rho}) &= \frac{1}{Z} \exp(-H(\boldsymbol{\theta}, \boldsymbol{\rho})) \\ &= \frac{1}{Z} \exp(-E(\boldsymbol{\theta})) \exp(-K(\boldsymbol{\rho})), \end{aligned} \tag{17}$$

where Z is a normalizing constant. Note that $\exp(-E(\boldsymbol{\theta}))$ is essentially the likelihood up to a normalizing factor. The momentum dynamics can be approximated by the following difference equations

$$\boldsymbol{\rho}_t = \nabla \boldsymbol{\theta}_t \approx \Delta \boldsymbol{\theta}_t \tag{18a}$$

$$\nabla \boldsymbol{\rho}_t = -\frac{\partial E(\boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta}_t} \approx -\frac{\Delta E(\boldsymbol{\theta}_t)}{\Delta \boldsymbol{\theta}_t}, \tag{18b}$$

where, obviously, all of terms are intermediate results from the Alopex-like algorithm without additional computing overhead. By doing so, the posterior of $\boldsymbol{\theta}_{t+1}$ is then proportional to $p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t) p_H(\boldsymbol{\theta}_t, \boldsymbol{\rho}_t) = p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t) \exp(-\frac{1}{2}\boldsymbol{\rho}_t^T \boldsymbol{\rho}_t) p(\mathbf{y}_t|\boldsymbol{\theta}_t)$. Equivalently, while keeping the importance weights proportional to the likelihood, (15) is further changed to

$$\begin{aligned} \tilde{\boldsymbol{\theta}}_{t+1}^{(i)} &= \boldsymbol{\theta}_{t+1}^{(i)} + \beta \Delta \boldsymbol{\theta}_t^{(i)} \\ &= \boldsymbol{\mu}_t + \alpha(\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\mu}_t) + \beta \Delta \boldsymbol{\theta}_t^{(i)} + \sqrt{1-\alpha^2} \sigma \boldsymbol{\nu}_t \end{aligned} \tag{19}$$

where β is a momentum coefficient. Equation (19) implies that $p(\tilde{\boldsymbol{\theta}}_{t+1}|\boldsymbol{\theta}_t, \Delta\boldsymbol{\theta}_t) \propto p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t) \exp(-\Delta\boldsymbol{\theta}_t^T \Delta\boldsymbol{\theta}_t)$.

5. EXPERIMENTAL RESULTS

The algorithms and all of the experiments are coded in Matlab and implemented on a PC. We only compare our algorithms with the Alopex-B algorithm simply because we also found that it converges faster than the conventional Alopex algorithm. For Alopex-B algorithm, no exhaustive effort was made to find optimal parameters γ and λ; we used the same recommended values as [1] in our experiments. $\boldsymbol{\theta}_0$ are uniformly distributed within the region $[-1.5, 1.5]$. Once $\theta_j(0)$ is generated, an initial Gaussian prior $\mathcal{N}(\theta_j(0), 0.5)$ is used for generating the samples $\{\theta_j^{(i)}\}$. The error measure is the mean-squared error (MSE) $\frac{1}{2\ell}\sum_{t=1}^{\ell} \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|^2$ in the case of off-line learning for non-sequential data (with total ℓ observations). For sequential data, MSE corresponds to the time-averaged prediction error. Unless stated otherwise, we set $N_p = 10$. For sampling-based Alopex algorithms, we only monitor the minimum MSE among all $\{\boldsymbol{\theta}^{(i)}\}$; the one achieving minimum MSE is regarded as the *maximum a posteriori* (MAP) estimate. For non-sequential data, the typical parameter setup in the experiments is $\sigma \in [0.01, 0.02]$, $\eta = 0.01$, $\gamma \in [0.05, 0.1]$, $\beta = \gamma/10$, and $\lambda = 0.5$; for sequential data, $\sigma \in [0.1, 0.5]$, $\eta = 0.01$, $\gamma = 0.1$, $\beta = 0.01$, $\lambda = 0.1$. The relaxing parameter is often chosen in the region $\alpha \in [-0.7, 0.5]$ (for sequential data $\alpha < 0$).

For the purpose of comparison, the following five benchmark problems: `XOR`, `parity`, `encoder-decoder`, `encoder`, and `decoder`, taken from [12], [1], are used for MLP networks training; and the following three benchmark problems: `sequential XOR-`D, `delay-`D, and `sequential parity`, taken from [1], are used for different types of RMLP networks training. The network architectures for these problems and the convergence results are listed in Table 1. As seen, the sampling-based Alopex algorithms significantly improve the convergence behavior. Besides, we have also tested several other problems including pattern (two-spiral and digit) recognition, and logistic map prediction, financial data (option prices) tracking, robot-arm system identification. Due to space limitation, we refer the reader to [15] for the problem and experimental details.

6. SUMMARY

To summarize, we propose two Monte Carlo sampling-based Alopex algorithms for training neural networks. Although only the MLP and RMLP networks are tested in the paper, the proposed schemes are suitable for a large class of convex/non-convex optimization problems since they are *mode-free* and *architecture-independent*. The other characteristics of the proposed algorithms include: (i) The transfer and cost functions need not be differentiable; (ii) the algorithms contain two kinds of stochastic components (i.e., Monte Carlo samples and perturbation vector $\boldsymbol{\xi}$), both of which are helpful (but not sufficient) to escape local minima; (iii) they are easily implemented. These properties look attractive and make

[1]Sometimes it is suggested to insert a Markov chain Monte Carlo (MCMC) step between steps 6 and 7 to increase the diversity of the samples.

Table 1. Experimental Results. For non-sequential data, the numbers are the averaged and minimum epochs for achieving training error MSE $= 0.001$ (0.01 for digit recognition and two-spiral problems); if the desired MSE is not achieved, the corresponding values represent the averaged MSE within 100,000 epochs (500,000 for two-spiral problem). Testing errors are measured in terms of MSE or misclassification rate in the best cases.

Problem	network	ALOPEX-B			Algorithm-1			Algorithm-2		
	size	ave.	best	test	ave.	best	test	ave.	best	test
XOR	MLP2-2-1	7568	1333	—	1008	845	—	879	633	—
encoder-decoder	MLP4-2-4	4080	2657	—	2138	1749	—	1906	1734	—
parity	MLP4-4-1	65372	25519	—	5589	5240	—	5467	4546	—
encoder	MLP8-3-3	2594	2139	—	1402	1059	—	1374	926	—
decoder	MLP3-3-8	24220	10126	—	4951	4428	—	4848	4430	—
Delay-1	RMLP1-2-1	1210	1108	—	451	412	—	382	334	—
Delay-2	RMLP1-3-1	5031	4291	—	1922	1581	—	1309	1101	—
Delay-3	RMLP1-4-1	596	356	—	239	201	—	215	149	—
sequential XOR-0	RMLP1-2-1	6985	6235	—	3531	2946	—	3013	2511	—
sequential XOR-1	RMLP1-4-1	7032	6323	—	3155	2843	—	2983	2325	—
sequential XOR-2	RMLP1-6-1	11323	10832	—	7932	7059	—	7251	6501	—
sequential parity	RMLP1-3-1	13138	11098	—	7832	7031	—	7238	6235	—
logistic map	MLP1-8-1	14975	11302	0.0009	7057	6023	0.0009	5938	5015	0.0009
logistic map	RMLP1-5-1	30849	29872	0.0010	13248	9321	0.0009	10321	8312	0.0009
digit recognition	MLP35-5-4	15101	8607	—	2616	1451	—	1453	1327	—
two-spiral	MLP2-25-1	fail	—	—	0.0472	0.0433	12%	0.0391	0.0362	10%
two-spiral	net2-5-5-5-1	0.0485	0.0434	31%	—	—	—	0.0415	0.03	4%
robot-arm	MLP2-6-2	0.0718	0.0691	0.1210	0.0121	0.0097	0.0231	0.0110	0.0065	0.0158
option price	MLP2-6-2	0.1381	0.0983	—	0.0242	0.0193	—	0.0231	0.0175	—

them applicable to many scenarios. We believe our algorithms have a lot of potential in many adaptive systems (e.g., blind source separation [15], vision, reinforcement learning and control) as well as for parallel hardware implementation [11]: our algorithm can be viewed as a bank of Alopex's implemented in parallel.

Future work will consider incorporating tunnelling techniques and MCMC for global optimization. Convexity and regularization are also two important factors for optimization and learning. Further efforts are still needed for investigating large-scale optimization problems.

7. REFERENCES

[1] Bia, A. (2001). Alopex-B: A new, simple, but yet faster version of the Alopex training algorithm. *International Journal of Neural Systems, 11*(6), 497-507.

[2] Briegel, T. & Tresp, V. (1999). Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space models. In *Advances in Neural Information Processing Systems, 11*, MIT Press.

[3] Chen, Z. (2003). Bayesian filtering: From Kalman filters to particle filters, and beyond. Tech. Rep., available on line http://soma.crl.mcmaster.ca/~zhechen/download/ieee_bayesian.ps

[4] de Freitas, J. F. G., Niranjan, M., Gee, A. H., & Doucet, A. (2000). Sequential Monte Carlo methods to train neural network models. *Neural Computation, 12*(4), 955-993.

[5] Doucet, A., de Freitas, N. & Gordon, N. Eds. (2001). *Sequential Monte Carlo Methods in Practice*, Springer.

[6] Harth, E., Unnikrishnan, K. P., & Pandya, A. S. (1987). The inversion of sensory processing by feedback pathways: A model of visual cognitive functions. *Science, 237*, 184-187.

[7] MacKay, D. J. C. (1999). Introduction to Monte Carlo methods. In M. I. Jordan Ed. *Learning in Graphical Models*, MIT Press.

[8] Sastry, P. S., Magesh, M., & Unnikrishnan, K. P. (2002). Two timescale analysis of Alopex algorithm for optimization. *Neural Computation, 14*(11), 2729-2750.

[9] Shah S. & Sastry, P. S. (1999). New algorithms for learning and pruning oblique decision trees. *IEEE Transactions on Systems, Man and Cybernetics, C29*, 495-405.

[10] Tzanakou, E., Michalak, R., & Harth, E. (1979). The Alopex process: visual receptive fields by response feedback. *Biological Cybernetics, 35*, 161-174.

[11] Tzanakou E. (2000). *Supervised and Unsupervised Pattern Recognition: Feature Extraction and Computational Intelligence*, CRC Press.

[12] Unnikrishnan, K. P. & Venugopal, K. P. (1994). Alopex: A correlation-based learning algorithm for feedforward and recurrent neural networks. *Neural Comput., 6*, 469-490.

[13] van der Merwe, R., de Freitas, J. F. G., Doucet, A., & Wan, E. (2001). The unscented particle filter. In *Advances in Neural Information Processing Systems, 13*, MIT Press.

[14] Wong, W. H. & Liang, F. (1997). Dynamic importance weighting in Monte Carlo and optimization. *Proc. Natl. Acad. Sci. USA, 94*, 14220-14224.

[15] Haykin, S., Chen, Z., & Becker, S. (2003). Stochastic correlative learning algorithms. *IEEE Trans. Signal Processing* (to appear). See also Tech. Rep. *Correlation: novel basis for statistical learning algorithms*, Adaptive Systems Lab, McMaster University.