LETTER Communicated by Steven Nowlan and Gerard Dreyfus

Sequential Monte Carlo Methods to Train Neural Network Models

J. F. G. de Freitas
M. Niranjan
A. H. Gee
A. Doucet
Cambridge University Engineering Department, Cambridge CB2 1PZ, England, U.K.

We discuss a novel strategy for training neural networks using sequential Monte Carlo algorithms and propose a new hybrid gradient descent/sampling importance resampling algorithm (HySIR). In terms of computational time and accuracy, the hybrid SIR is a clear improvement over conventional sequential Monte Carlo techniques. The new algorithm may be viewed as a global optimization strategy that allows us to learn the probability distributions of the network weights and outputs in a sequential framework. It is well suited to applications involving on-line, nonlinear, and nongaussian signal processing. We show how the new algorithm outperforms extended Kalman filter training on several problems. In particular, we address the problem of pricing option contracts, traded in financial markets. In this context, we are able to estimate the one-step-ahead probability density functions of the options prices.

1 Introduction

Probabilistic modelling coupled with nonlinear function approximators is a powerful tool in solving many real-world problems. In the engineering community, there are many examples of problems involving large data sets that have been tackled with nonlinear function approximators such as the multilayer perceptron (MLP) or radial basis functions. Much of the emphasis here is on performance, in the form of accuracy of prediction on unseen data. The tricks required to estimate parameters of very large models and the handling of very large data sets become interesting challenges too. Tools such as cross validation to deal with overfitting and bootstrap to deal with model uncertainty have proved to be very effective. Neural networks applied to speech recognition (Robinson, 1994) and handwritten digit recognition (Le Cun et al., 1989) are examples. Such work has demonstrated that many interesting and hard inference tasks, involving thousands of free parameters, can indeed be solved at a desirable level of performance. On the other hand, in the statistics community, we see that the emphasis is on rigorous mathematical analysis of algorithms, careful model formulation, and model criticism. This work has led to a rich paradigm to handle various inference problems in which a good collection of powerful algorithms exist. Sampling techniques applied to neural networks, starting from the work of Radford Neal, is a classic example where both the above features are exploited (Neal, 1996). He shows that nonlinear function approximators in the form of MLPs can be trained, and their performance evaluated, in a Bayesian framework. This formulation leads to probability distributions that are difficult to handle analytically. Markov chain Monte Carlo (MCMC) sampling methods are used to make inferences in the Bayesian framework. While sampling methods tend to be much richer in exploring the probability distributions, approximate methods, such as gaussian approximation, have also attracted interest (Bishop, 1995; Mackay, 1992).

Many problems, such as time-series analysis, are characterized by data that arrive sequentially. In such problems one has the task of performing model estimation, model validation, and inference sequentially, on the arrival of each item of data. By the very nature of such real-time tasks, or if we suspect the source of the data to be time varying, we may wish to adopt a sequential strategy in which each item of data is used only once and then discarded. In this case, deciding what information we should be propagating on a sample-by-sample basis becomes an interesting topic. In the classic state-space model estimation by a Kalman filter setting, one defines a gaussian probability density over the parameters to be estimated. The mean and covariance matrix are propagated using recursive update equations each time new data are received. Use of the matrix inversion lemma allows an elegant update of the inverse covariance matrix without actually having to invert a matrix at each sample. This is a widely studied topic, optimal in the case of linear gaussian models. For nonlinear models, the common trick is Taylor series expansion around the operating point, leading to the extended Kalman filter (EKF). The derivation of the Kalman filter starting from a Bayesian setting is of much interest. This allows for tuning of the noise levels as well as running multiple models in parallel to evaluate model likelihoods. Jazwinski (1970) and Bar-Shalom and Li (1993) are classic textbooks dealing with these aspects. Our earlier work reviews these concepts and shows them applied in a neural network context (de Freitas, Niranjan, & Gee, 1997, 1998a).

Taylor series approximation leading to the EKF makes gross simplification of the probabilistic specification of the model. With nonlinear models, the probability distributions of interest tend to be multimodal. Gaussian approximation in such cases could miss the rich structure brought in by the nonlinear model. Sequential sampling techniques provide a class of algorithms to address this issue. A good application of these ideas is tracking in computer vision (Isard & Blake, 1996), where the Kalman filter approximation is shown to fail to capture the multimodalities. Sequential sampling algorithms, under the names of particle filters, sequential sampling-importance resampling (SIR), bootstrap filters, and condensation trackers have also been applied to a wide range of problems, including target tracking (Gordon, Salmond, & Smith, 1993), financial analysis (Müller, 1992; Pitt & Shephard, 1997), diagnostic measures of fit (Pitt & Shephard, 1997), sequential imputation in missing data problems (Kong, Liu, & Wong, 1994), blind deconvolution (Liu & Chen, 1995) and medical prognosis (Berzuini, Best, Gilks, & Larizza, 1997).

In this article we focus on neural networks, constraining ourselves to sequential training and predictions. We use sampling techniques in a state-space setting to show how an MLP may be trained in environments where data arrive one at a time. We consider sequential importance sampling and resampling algorithms and illustrate their performance on some simple problems. Further, we introduce a hybrid algorithm in which each sampling step is supplemented by a gradient-type update step. This could be a straightforward gradient descent on all the samples propagated or even an EKF type update of the samples. We do not deal here with the problem of choosing the network architecture.

We target this work primarily at the neural networks community. We believe that many signal processing applications addressed by this community will fall into this category. Part of the article is written in a tutorial form to introduce the concepts. We also believe the real-life example we include and the experience gained in applying sequential sampling techniques with a highly nonlinear function approximator will be of benefit to the statistics community. In particular, we improve the existing sequential sampling methods by incorporating gradient information. This results in a hybrid optimization scheme (HySIR), whereby each sampling trajectory of the sampling algorithm is updated with an EKF. The HySIR may therefore be interpreted as an efficient dynamic mixture of EKFs, whose accuracy improves as the number of filters increases. We show that the HySIR algorithm outperforms conventional sequential sampling methods in terms of computational speed and accuracy.

In section 2, we formulate the problem of training neural networks in terms of a state-space representation. A theoretical Bayesian solution is subsequently proposed in section 3. Owing to the practical difficulties that arise when computing the Bayesian solution, three approximate methods (numerical integration, gaussian approximation, and Monte Carlo simulation) are discussed in section 4. In sections 5 and 6, we derive a generic sequential importance sampling methodology and point out some of the limitations of conventional sampling methods. In section 7, we introduce the HySIR algorithm. Finally, in section 8, we test the algorithms on several problems, including the pricing of options on the Financial Times Stock Exchange 100-Share (FTSE-100) index.

2 State-Space Neural Network Modeling

As in our previous work (de Freitas et al., 1997), we start from a state-space representation to model the neural network's evolution in time. A transition equation describes the evolution of the network weights, and a measurements equation describes the nonlinear relation between the inputs and outputs of a particular physical process, as follows:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{d}_k \tag{2.1}$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{w}_k, \mathbf{x}_k) + \mathbf{v}_k, \tag{2.2}$$

where ($\mathbf{y}_k \in \Re^o$) denotes the output measurements, ($\mathbf{x}_k \in \Re^d$) the input measurements, and ($\mathbf{w}_k \in \Re^m$) the neural network weights. The measurements nonlinear mapping $\mathbf{g}(.)$ is approximated by an MLP. This neural model exhibits the capacity to approximate any continuous function, to an arbitrary precision, as long as it is not restricted in size (Cybenko, 1989). Yet our work may be extended easily to encompass recurrent networks, radial basis networks, and many other approximation techniques. The measurements are assumed to be corrupted by noise $\mathbf{v}_k$. In the sequential Monte Carlo framework, the probability distribution of the noise is specified by the user. In our examples we shall choose a zero mean gaussian distribution with covariance R. The measurement noise is assumed to be uncorrelated with the network weights and initial conditions.

We model the evolution of the network weights by assuming that they depend on the previous value $\mathbf{w}_k$ and a stochastic component $\mathbf{d}_k$. The process noise $\mathbf{d}_k$ may represent our uncertainty in how the parameters evolve, modeling errors or unknown inputs such as maneuvers in tracking applications (Bar-Shalom & Li, 1993). We assume the process noise to be a zero mean gaussian process with covariance Q; however, other distributions can also be adopted. This choice of distributions for the network weights requires further research. The process noise is also assumed to be uncorrelated with the network weights. In section 8, we propose that this random walk model can be improved by replacing it with the following gradient descent model,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha(\mathbf{y}_k - \hat{\mathbf{g}}(\mathbf{w}_k, \mathbf{x}_k))\frac{\partial \hat{\mathbf{g}}(\mathbf{w}_k, \mathbf{x}_k)}{\partial \mathbf{w}_k} + \mathbf{d}_k, \tag{2.3}$$

where α is a learning-rate parameter and $\hat{\mathbf{g}}(\mathbf{w}_k, \mathbf{x}_k)$ is the model prediction at time k. The gradient descent model allows for directed steps when searching for optimal locations in parameter space.

The noise terms are assumed to be uncorrelated with the network weights and the initial conditions. The evolution of the states (network weights) corresponds to a Markov process with initial probability $\mathrm{p}(\mathbf{w}_0)$ and transition probability $\mathrm{p}(\mathbf{w}_k|\mathbf{w}_{k-1})$. The observations are assumed to be conditionally independent given the states. These are standard assumptions in a large class of tracking and time series problems (Harvey, 1970).

Typically, the task is to estimate the network weights $\hat{\mathbf{w}}_k$ and the noise parameters R and Q given the measurements $Y_k = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$. To simplify the exposition, we do not treat the problem of estimating the noise covariances and the initial probabilities. (To understand how these variables may be estimated by hierarchical Bayesian models or expectation-maximization (EM) learning, see de Freitas et al., 1997, 1998a.) We shall, however, analyze the roles played by R and Q in sequential Monte Carlo simulation.

3 The Bayesian Solution

The posterior density $p(W_k|Y_k)$, where $Y_k = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ and $W_k = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k\}$, constitutes the complete solution to the sequential estimation problem. In many applications, such as tracking, it is of interest to estimate one of its marginals, the filtering density $p(\mathbf{w}_k|Y_k)$. By computing the filtering density recursively, we do not need to keep track of the complete history of the weights. Thus, from a storage point of view, the filtering density turns out to be more parsimonious than the full posterior density function. If we know the filtering density of the network weights, we can easily derive various estimates of the network weights, including centroids, modes, medians, and confidence intervals. In sections 5 and 6, we show how the filtering density may be approximated using sequential importance sampling techniques.

The filtering density is estimated recursively in two stages: prediction and update, as illustrated in Figure 1. In the prediction step, the filtering density $p(\mathbf{w}_{k-1}|Y_{k-1})$ is propagated into the future by the transition density $p(\mathbf{w}_k|\mathbf{w}_{k-1})$ as follows:

$$p(\mathbf{w}_k|Y_{k-1}) = \int p(\mathbf{w}_k|\mathbf{w}_{k-1})\, p(\mathbf{w}_{k-1}|Y_{k-1}) d\mathbf{w}_{k-1}. \tag{3.1}$$

The transition density is defined in terms of the probabilistic model governing the states' evolution (see equation 2.1) and the process noise statistics, that is:

$$\begin{aligned} p(\mathbf{w}_k|\mathbf{w}_{k-1}) &= \int p(\mathbf{w}_k|\mathbf{d}_{k-1}, \mathbf{w}_{k-1})\, p(\mathbf{d}_{k-1}|\mathbf{w}_{k-1}) d\mathbf{d}_{k-1} \\ &= \int \delta(\mathbf{w}_k - \mathbf{d}_{k-1} - \mathbf{w}_{k-1})\, p(\mathbf{d}_{k-1}) d\mathbf{d}_{k-1}, \end{aligned}$$

where the Dirac delta function $\delta(.)$ indicates that $\mathbf{w}_k$ can be computed via a purely deterministic relation when $\mathbf{w}_{k-1}$ and $\mathbf{d}_{k-1}$ are known. Note that $p(\mathbf{d}_{k-1}|\mathbf{w}_{k-1}) = p(\mathbf{d}_{k-1})$ because the process and measurement noise terms have been assumed to be independent of past and present values of the states.

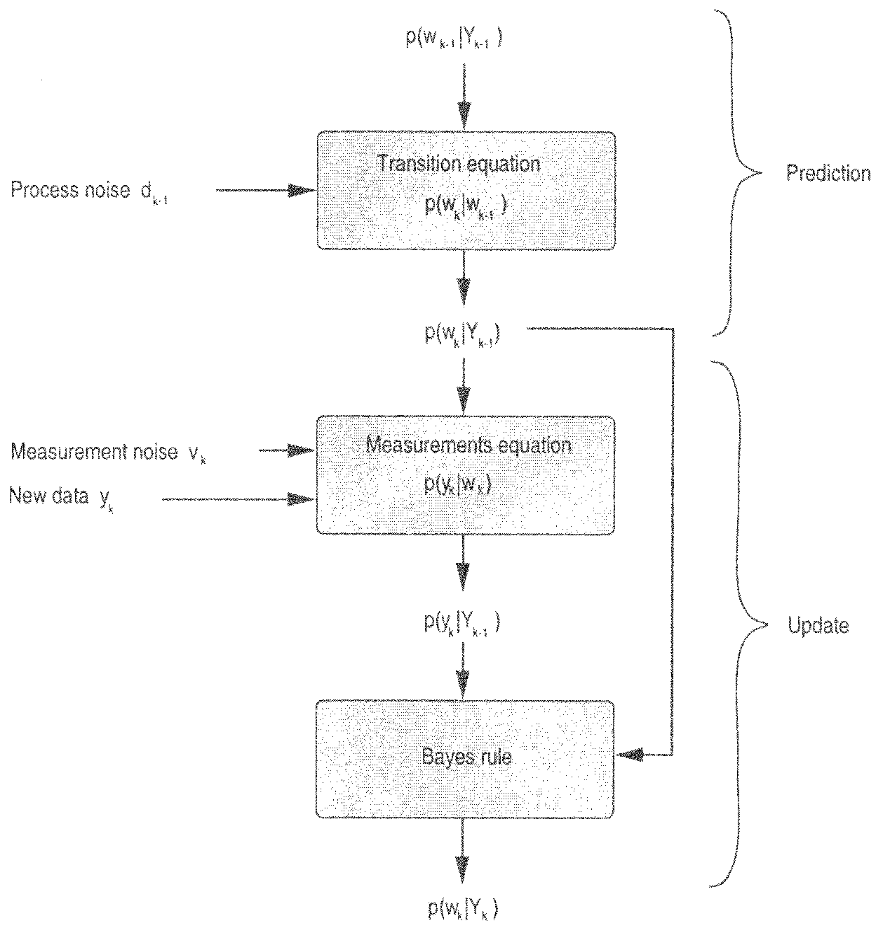

Figure 1: Prediction and update stages in the recursive computation of the filtering density.

The update stage involves the application of Bayes' rule when new data are observed:

$$\mathrm{p}(\mathbf{w}_k|Y_k) = \frac{\mathrm{p}(\mathbf{y}_k|\mathbf{w}_k)\,\mathrm{p}(\mathbf{w}_k|Y_{k-1})}{\mathrm{p}(\mathbf{y}_k|Y_{k-1})}. \tag{3.2}$$

The likelihood density function is defined in terms of the measurements model (see equation 2.2), as follows:

$$\mathrm{p}(\mathbf{y}_k|\mathbf{w}_k) = \int \delta(\mathbf{y}_k - g(\mathbf{w}_\mathrm{k}, \mathbf{x}_\mathrm{k}) - \mathbf{v}_\mathrm{k})\,\mathrm{p}(\mathbf{v}_\mathrm{k})d\mathbf{v}_\mathrm{k}.$$

The normalizing denominator of equation 3.2 is often referred to as the evidence density function. It plays a key role in optimization schemes that exploit gaussian approximation (de Freitas et al., 1997; Jazwinski, 1969; Mackay, 1992; Sibisi, 1989). It is given by:

$$p(\mathbf{y}_k|Y_{k-1}) = \int p(\mathbf{y}_k|\mathbf{w}_k)\, p(\mathbf{w}_k|Y_{k-1}) d\mathbf{w}_k.$$

In this formulation, the question of how to choose the prior $p(\mathbf{w}_0)$ arises immediately. This is a central issue in Bayesian inference. The problem is, however, much more general. It is related to the ill-posed nature of the inference and learning task. A finite set of data admits many possible explanations, unless some form of a priori knowledge is used to constrain the solution space. Assuming that the data conform to a specific degree of smoothness is certainly the most fundamental and widely applied type of a priori knowledge. This assumption is particularly suited to applications involving noisy data. Nonetheless, the Bayesian approach admits more general forms of a priori knowledge within a probabilistic formulation. It would be interesting to devise methods for mapping experts' knowledge, available in many formats, into the Bayesian probabilistic representation.

4 Practical Solutions

The prediction and update strategy given by equations 3.1 and 3.2 provides an optimal solution to the inference problem, but unfortunately, it involves multidimensional integrations, the source of most of the practical difficulties inherent in the Bayesian methodology. In most applications, analytical solutions are not possible. This is why we need to resort to alternative approaches, such as direct numerical integration, gaussian approximation, and Monte Carlo simulation methods.

4.1 Direct Numerical Integration. The direct numerical integration method relies on approximating the distribution of interest by a discrete distribution on a finite grid of points. The location of the grid is a nontrivial design issue. Once the distribution is computed at the grid points, an interpolation procedure is used to approximate it in the remainder of the space. Kitagawa (1987) used this method to replace the filtering integrals by finite sums over a large set of equally spaced grid points. He chose a piecewise linear interpolation strategy. Kramer and Sorenson (1988) adhered to the same methodology, but opted for a constant interpolating function. Pole and West (1990) have attempted to reduce the problem of choosing the grid's location by implementing a dynamic grid allocation method.

When the grid points are spaced "closely enough" and encompass the region of high probability, the method works well. However, the method is very difficult to implement in high-dimensional multivariate problems such as neural network modeling. Here, computing at every point in a dense multidimensional grid becomes prohibitively expensive (Gelman, Carlin, Stern, & Rubin, 1995; Gilks, Richardson, & Spiegelhalter, 1996).

4.2 Gaussian Approximation. Until recently, the popular approach to sequential estimation has been gaussian approximation (Bar-Shalom & Li, 1993). In the linear gaussian estimation problem, the Kalman filter provides an optimal recursive algorithm for propagating and updating the mean and covariance of the hidden states. In nonlinear scenarios, the EKF is a computationally efficient natural extension of the Kalman filter. The EKF is obtained by approximating the transition and measurements equations with Taylor expansion about the last predicted state. Typically, first-order expansions, which lead to a gaussian approximation of the posterior density function, are employed. The mean and covariance are propagated and updated by a simple set of equations, similar to the Kalman filter equations. As the number of terms in the Taylor expansion increases, the complexity of the algorithm also increases due to the computation of derivatives of increasing order. For example, a linear expansion requires the computation of the Jacobian, while a quadratic expansion involves computing the Hessian matrix.

Gaussian approximation results in a simple and elegant framework that is amenable to the design of inference and learning algorithms. Various authors have studied the problem of approximating the distribution of neural network weights with a gaussian function. One of the earliest implementations of EKF-trained MLPs is due to Singhal and Wu (1988). In their method, the network weights are grouped into a single vector **w** that is updated in accordance with the EKF equations. The entries of the Jacobian matrix (i.e., the derivative of the outputs with respect to the weights) are calculated by backpropagating the output observations through the network.

The algorithm Singhal and Wu proposed requires considerable computational effort. The complexity is of the order om^2 multiplications per estimation step, where m represents the number of weights and o the number of network outputs. Shah, Palmieri, and Datum (1992) and Puskorius and Feldkamp (1991; Puskorius, Feldkamp, & Davis, 1996) have proposed strategies for decoupling the global EKF estimation algorithm into local EKF estimation subproblems. For example, they suggest that the weights of each neuron could be updated independently. The assumption in the local updating strategies is that the weights are decoupled, and, consequently, the Kalman filter weights covariance is a block-diagonal matrix.

The EKF is an improvement over conventional neural network estimation techniques, such as on-line backpropagation, in that it makes use of second-order statistics (covariances). These statistics are essential for placing error bars on the predictions and combining separate networks into committees of networks. The backpropagation algorithm is, in fact, a degenerate form of the EKF algorithm (de Freitas et al., 1997; Ruck, Rogers, Kabrisky, Maybeck, & Oxley, 1992). Previously we have shown that gaussian approximation, within a hierarchical Bayesian framework, leads to interesting algorithms that exhibit adaptive memory and adaptive regularization coefficients. These algorithms enabled us to learn the dynamics and measurements noise statistics in slowly changing nonstationary environments (de Freitas et al., 1997; de Freitas, Niranjan, & Gee, 1998c). In stationary environments, where the data are available in batches, we have also shown (de Freitas et al., 1998a; de Freitas, Niranjan, & Gee, 1998b) that it is possible to learn these statistics by means of the Rauch-Tung-Striebel smoother (Rauch, Tung, & Striebel, 1965) and the EM algorithm (Dempster, Laird, & Rubin, 1977). There have been several attempts to implement the EM algorithm online (Collings, Krishnamurthy, & Moore, 1994; Elliott, 1994; Krishnamurthy & Moore, 1993). Unfortunately, these still rely on various heuristics and tend to be computationally demanding.

A natural progression on sequential gaussian approximation is to employ a mixture of gaussian densities (Kadirkamanathan & Kadirkamanathan, 1995; Li & Bar-Shalom, 1994; Sorenson & Alspach, 1971). These mixtures can be either static or dynamic. In static mixtures, the gaussian models assumed to be valid throughout the entire process are a subset of several hypothesized models. That is, we start with a few models and compute which of them describe the sequential process most accurately. The remaining models are discarded. In dynamic model selection, one particular model out of a set of r operating models is selected during each estimation step. Dynamic mixtures of gaussian models are far more general than static mixtures of models. However, in stationary environments, static mixtures are obviously more adequate. Dynamic mixtures are particularly suited to the problem of noise estimation in rapidly changing environments, such as tracking maneuvering targets. There each model corresponds to a different hypothesis of the value of the noise covariances (Bar-Shalom & Li, 1993).

In summary, gaussian approximation, because of its simplicity and computational efficiency, constitutes a good way of handling many problems where the density of interest has a significant and predominant mode. Many problems, however, do not fall into this category. Mixtures of gaussians provide a better solution when there are a few dominant modes. However, they introduce extra computational requirements and complications, such as estimating the number of mixture components.

4.3 Monte Carlo Simulation. Many signal processing problems, such as equalization of communication signals, financial time series, medical prognosis, target tracking, and geophysical data analysis, involve elements of nongaussianity, nonlinearity, and nonstationarity. Consequently, it is not possible to derive exact closed-form estimators based on the standard criteria of maximum likelihood, maximum a posteriori, or minimum mean-squared error. Analytical approximations to the true distribution of the data do not take into account all the salient statistical features of the processes under consideration. Monte Carlo simulation methods, which provide a complete description of the probability distribution of the data, improve the accuracy of the analysis.

The basic idea in Monte Carlo simulation is that a set of weighted samples, drawn from the posterior density function of the neural network weights, is used to map the integrations, involved in the inference process, to discrete sums. More precisely, we make use of the following Monte Carlo approximation:

$$\hat{\mathrm{p}}(W_k|Y_k) = \frac{1}{N}\sum_{i=1}^{N} \delta\left(W_k - W_k^{(i)}\right),$$

where $W_k^{(i)}$ represents the samples used to describe the posterior density and, as before, $\delta(.)$ denotes the Dirac delta function. We carry the suffix k, denoting time, to emphasize that the approximation is performed at the arrival of each data point. Consequently, any expectations of the form

$$\mathbf{E}[f_k(W_k)] = \int f_k(W_k)\,\mathrm{p}(W_k|Y_k)\mathrm{d}W_k$$

may be approximated by the following estimate:

$$\mathbf{E}[f_k(W_k)] \approx \frac{1}{N}\sum_{i=1}^{N} f_k(W_k^{(i)}),$$

where the samples $W_k^{(i)}$ are drawn from the posterior density function. Monte Carlo sampling techniques are an improvement over direct numerical approximation in that they automatically select samples in regions of high probability.

In recent years, many researchers in the statistical and signal processing communities have, almost simultaneously, proposed several variations of sequential Monte Carlo algorithms. These algorithms have been applied to a wide range of problems, including target tracking (Gordon et al., 1993; Isard & Blake, 1996), financial analysis (Müller, 1992; Pitt & Shephard, 1997), diagnostic measures of fit (Gelfand, Dey, & Chang, 1992; Pitt & Shephard, 1997), sequential imputation in missing data problems (Kong et al., 1994), blind deconvolution (Liu & Chen, 1995) and medical prognosis (Berzuini et al., 1997). Most of the current research has focused on statistical refinements of this class of algorithm (Carpenter, Clifford, & Fearnhead, 1997; Liu & Chen, 1998; Pitt & Shephard, 1997). The basic sequential Monte Carlo methods had been introduced in the automatic control field in the late sixties. For instance, Handschin and Mayne (1969) tackled the problem of nonlinear filtering with a sequential importance sampling approach. They combined analytical and numerical techniques, using the control variate method (Hammersley & Handscomb, 1968), to reduce the variance of the estimates. In the seventies, various researchers continued working on these ideas (Akashi & Kumamoto, 1977; Handschin, 1970; Zaritskii, Svetnik, &

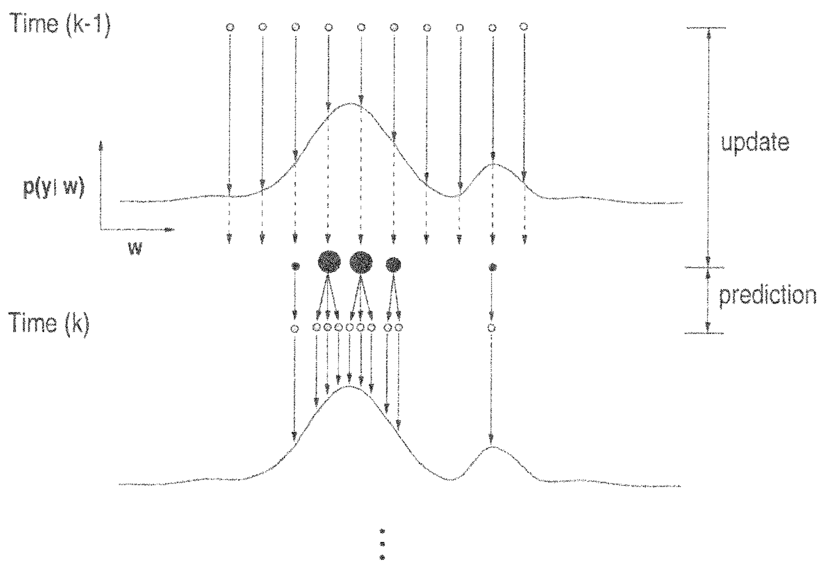

Figure 2: Update and prediction stages in the sequential sampling process. In the update stage, the likelihood of each sample is evaluated. The samples are then propagated with their respective likelihood. The size of the black circles indicates the likelihood of a particular sample. The samples with higher likelihood are allowed to have more "children." In the prediction stage, a process noise term is added to the samples so as to increase the variety of the sample set. The result is that the surviving samples provide a better weighted description of the likelihood function.

Shimelevich, 1975). Zaritskii et al. (1975) is particularly interesting. The authors treated the problems of continuous and discrete time filtering and introduced several novel ideas. With the exception of Doucet (1998), most authors have overlooked the Monte Carlo sampling ideas proposed in the seventies.

Figure 2 illustrates the operation of a generic sequential Monte Carlo method. It embraces the standard assumption that we can sample from the prior $p(\mathbf{w}_k|Y_{k-1})$ and evaluate the likelihood $p(\mathbf{y}_k|\mathbf{w}_k^{(i)})$ of each sample. Only the fittest samples, that is, the ones with the highest likelihood, survive after the update stage. These then proceed to be multiplied, according to their likelihood, in the prediction stage. The update and prediction stages are governed by equations 3.2 and 3.1, respectively. It is instructive to approach the problem from an optimization perspective. Figures 3 and 4 show the windowed global descent in the error function that is typically observed. The diagrams shed light on the roles played by the noise covariances R and Q. Q dictates by how much the cloud of samples is expanded in the

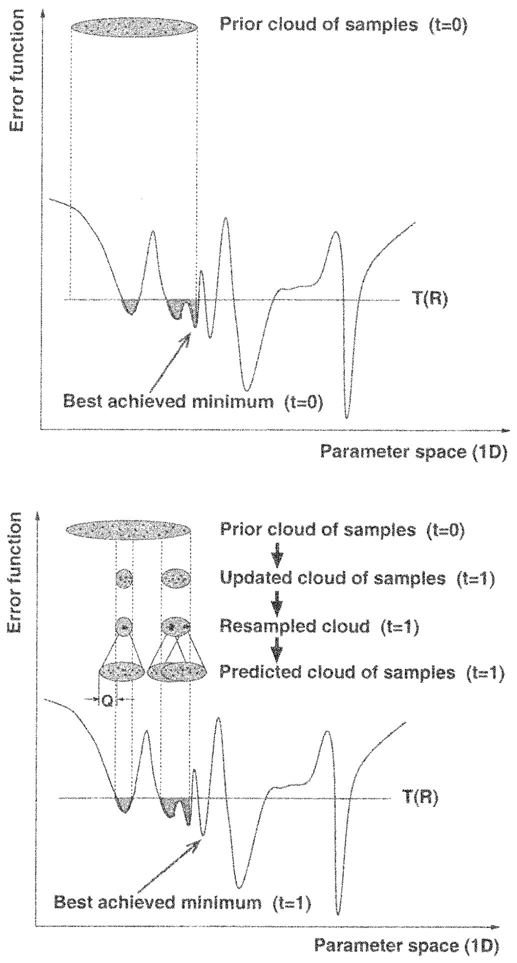
Error function
Prior cloud of samples (t=0)
T(R)
Best achieved minimum (t=0)
Parameter space (1D)
Error function
Prior cloud of samples (t=0)
Updated cloud of samples (t=1)
Resampled cloud (t=1)
Predicted cloud of samples (t=1)
Q
T(R)
Best achieved minimum (t=1)
Parameter space (1D)

Figure 3: First and second steps of a one-dimensional sequential Monte Carlo optimization problem. The size of the initial cloud of samples determines the search region in parameter space. The goal is to find the best possible minimum of the error function. It is clear that as the number of samples increases, the chances of reaching lower minima increase. In the second step, the updated clouds of samples are generated. The width of these clouds is obtained from the intersection of the width of the prior cloud of samples, the error function, and a threshold determined by the measurements noise covariance R. The updated clouds of samples are denser than the prior cloud of samples. Next, the samples are grouped in regions of higher likelihood in a resampling step. Finally, the clouds of samples are expanded by a factor determined by the process noise covariance Q. This expansion allows the clouds to reach regions of the parameter function where the error function is lower.

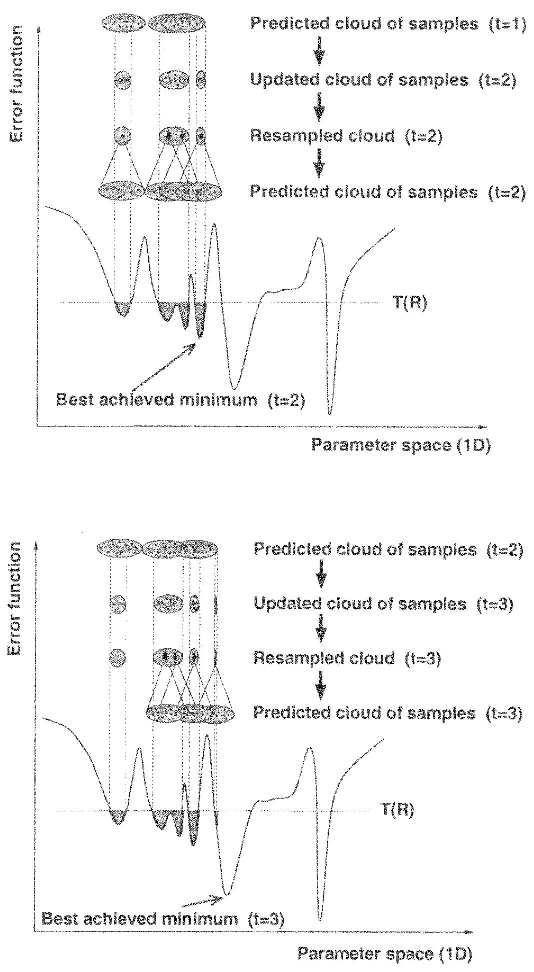
Predicted cloud of samples (t=1)
Updated cloud of samples (t=2)
Resampled cloud (t=2)
Predicted cloud of samples (t=2)
Error function
T(R)
Best achieved minimum (t=2)
Parameter space (1D)
Predicted cloud of samples (t=2)
Updated cloud of samples (t=3)
Resampled cloud (t=3)
Predicted cloud of samples (t=3)
Error function
T(R)
Best achieved minimum (t=3)
Parameter space (1D)

Figure 4: Third and fourth steps of a one-dimensional sequential Monte Carlo optimization problem. To reach the global minimum on the right, the number of samples has to be increased.

prediction stage. By increasing Q, the density of the cloud of samples is reduced. Consequently, the algorithm will take longer to converge. A very small Q, on the other hand, will not allow the algorithm to explore new regions of the parameter space. Ideally, one needs to implement an algorithm that adapts Q automatically. This is explored in section 7. R controls the resolution of the update stage, as shown in Figure 5. A small value of R will cause the likelihood to be too narrow. Consequently, only a few trajectories

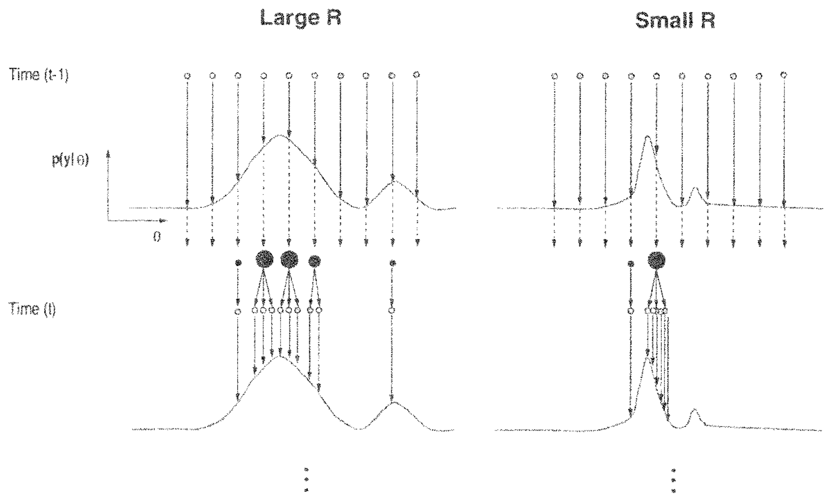

Figure 5: Role played by the measurements variance R. Small values of R result in a narrow likelihood, with only a few trajectories being able to propagate to the next time step. If R is too small, we might miss some of the important modes. On the other hand, if R is too large, we are not giving preference to any of the trajectories, and the algorithm might not converge.

will be able to propagate to the next time step. If R is too small, the optimization scheme might fail to detect some of the important modes of the likelihood function. By increasing R, we broaden the likelihood function, thereby increasing the number of surviving trajectories. If we choose R to be too large, all the trajectories become equally likely and the algorithm might not converge. As shown in Figures 3 and 4, R gives rise to a threshold $T(R)$ in the error function, which determines the width of the updated clouds of samples.

If we can sample from the prior and evaluate the likelihood up to proportionality, three sampling strategies can be adopted to sample from the posterior: acceptance-rejection sampling, Markov chain Monte Carlo (MCMC), and sampling-importance resampling (SIR).

After sampling from the prior, rejection sampling dictates that one should accept the samples with probability $p(\mathbf{y}_k|\mathbf{w}_k^{(i)})/\,p(\mathbf{y}_k|\mathbf{w}_{k-\text{Max}}^{(i)})$, where $\mathbf{w}_{k-\text{Max}}^{(i)} = \arg\max_{\mathbf{w}_k} p(\mathbf{y}_k|\mathbf{w}_k)$. Unfortunately, the rejection sampler requires a random number of iterations at each time step. This proves to be computationally expensive in high-dimensional spaces (Doucet, 1998; Müller, 1991; Pitt & Shephard, 1997).

The fundamental idea behind MCMC methods is to construct a Markov chain whose asymptotic distribution tends to the required posterior density (Gilks et al., 1996). Generally, it can take a large number of iterations before this happens. Moreover, it is difficult even to assess when the chain has converged. For most real-time sequential processing applications, MCMC methods can be too computationally demanding (Berzuini et al., 1997; Gordon & Whitby, 1995; Müller, 1992).

In our work, we favor the SIR sampling option. This approach is the core of many successful sequential Bayesian tools, such as the well-known bootstrap or particle filters (Avitzour, 1995; Beadle & Djuric, 1997; Berzuini et al., 1997; Doucet, 1998; Gordon et al., 1993; Kitagawa, 1996; Liu & Chen, 1998; Pitt & Shephard, 1997). In the following sections, we derive this algorithm from an importance sampling perspective.

5 Importance Sampling

We can approximate the posterior density function with a function on a finite discrete support. Consequently, it follows from the strong law of large numbers that as the number or samples N increases, expectations can be mapped into sums. Unfortunately, it is often impossible to sample directly from the posterior density function. However, we can circumvent this difficulty by sampling from a known, easy-to-sample, proposal density function $\pi(W_k|Y_k)$ and making use of the following substitution:

$$\begin{aligned}\mathbf{E}[f_k(W_k)] &= \int f_k(W_k)\frac{\mathrm{p}(W_k|Y_k)}{\pi(W_k|Y_k)}\pi(W_k|Y_k)\mathrm{d}W_k \\ &= \int f_k(W_k)\frac{\mathrm{p}(Y_k|W_k)\,\mathrm{p}(W_k)}{\mathrm{p}(Y_k)\pi(W_k|Y_k)}\pi(W_k|Y_k)\mathrm{d}W_k \\ &= \int f_k(W_k)\frac{q_k(W_k)}{\mathrm{p}(Y_k)}\pi(W_k|Y_k)\mathrm{d}W_k,\end{aligned}$$

where the variables $q_k(W_k)$ are known as the unnormalized importance ratios:

$$q_k = \frac{\mathrm{p}(Y_k|W_k)\,\mathrm{p}(W_k)}{\pi(W_k|Y_k)}. \tag{5.1}$$

We can get rid of the unknown normalizing density $\mathrm{p}(Y_k)$ as follows:

$$\begin{aligned}\mathbf{E}[f_k(W_k)] &= \frac{1}{\mathrm{p}(Y_k)}\int f_k(W_k)q_k(W_k)\pi(W_k|Y_k)\mathrm{d}W_k \\ &= \frac{\int f_k(W_k)q_k(W_k)\pi(W_k|Y_k)\mathrm{d}W_k}{\int \mathrm{p}(Y_k|W_k)\,\mathrm{p}(W_k)\frac{\pi(W_k|Y_k)}{\pi(W_k|Y_k)}\mathrm{d}W_k} \\ &= \frac{\int f_k(W_k)q_k(W_k)\pi(W_k|Y_k)\mathrm{d}W_k}{\int q_k(W_k)\pi(W_k|Y_k)\mathrm{d}W_k} \\ &= \frac{\mathbf{E}_\pi[q_k(W_k)f_k(W_k)]}{\mathbf{E}_\pi[q_k(W_k)]}.\end{aligned}$$

Hence, by drawing samples from the proposal function $\pi(.)$, we can approximate the expectations of interest by the following estimate:

$$\begin{aligned}\mathbf{E}[f_k(W_k)] &\approx \frac{1/N\sum_{i=1}^{N} f_k(W_k^{(i)})q_k(W_k^{(i)})}{1/N\sum_{i=1}^{N} q_k(W_k^{(i)})} \\ &= \sum_{i=1}^{N} f_k(W_k^{(i)})\tilde{q}_k(W_k^{(i)}),\end{aligned} \tag{5.2}$$

where the normalized importance ratios $\tilde{q}_k^{(i)}$ are given by

$$\tilde{q}_k^{(i)} = \frac{q_k^{(i)}}{\sum_{j=1}^{N} q_k^{(j)}}.$$

The estimate of equation 5.2 is valid if the $W_k^{(i)}$ correspond to a set of independently and identically distributed samples drawn from the proposal density function, the support of the proposal function includes the support of the posterior function, and the expectations of $f(.)$, q and $qf^2(.)$ exist and are finite (Geweke, 1989). Under these conditions and as N tends to infinity, the posterior density function can be approximated arbitrarily well by the estimate:

$$\hat{\mathrm{p}}(W_k|Y_k) = \sum_{i=1}^{N} \tilde{q}_k^{(i)}\delta(W_k - W_k^{(i)}).$$

6 Sequential Importance Sampling

In order to compute a sequential estimate of the posterior density function at time k without modifying the previously simulated states W_{k-1}, we may adopt the following proposal density:

$$\begin{aligned}\pi(W_k|Y_k) &= \pi(W_0|Y_0)\prod_{j=1}^{k}\pi(\mathbf{w}_j|W_{j-1}, Y_j) \\ &= \pi(W_{k-1}|Y_{k-1})\pi(\mathbf{w}_k|W_{k-1}, Y_k).\end{aligned} \tag{6.1}$$

At this stage, we need to recall that we assumed that the states correspond to a Markov process and that the observations are conditionally independent given the states—that is:

$$\mathrm{p}(W_k) = \mathrm{p}(\mathbf{w}_0)\prod_{j=1}^{k}\mathrm{p}(\mathbf{w}_j|\mathbf{w}_{j-1}) \tag{6.2}$$

and

$$\mathrm{p}(Y_k|W_k) = \prod_{j=1}^{k} \mathrm{p}(\mathbf{y}_j|\mathbf{w}_j). \tag{6.3}$$

By substituting equations 6.1–6.3 into equation 5.1, we can derive a recursive estimate for the importance weights as follows:

$$\begin{aligned} q_k &= \frac{\mathrm{p}(Y_k|W_k)\,\mathrm{p}(W_k)}{\pi(W_{k-1}|Y_{k-1})\pi(\mathbf{w}_k|W_{k-1},Y_k)} \\ &= q_{k-1}\frac{\mathrm{p}(Y_k|W_k)\,\mathrm{p}(W_k)}{\mathrm{p}(Y_{k-1}|W_{k-1})\,\mathrm{p}(W_{k-1})}\frac{1}{\pi(\mathbf{w}_k|W_{k-1},Y_k)} \\ &= q_{k-1}\frac{\mathrm{p}(\mathbf{y}_k|\mathbf{w}_k)\,\mathrm{p}(\mathbf{w}_k|\mathbf{w}_{k-1})}{\pi(\mathbf{w}_k|W_{k-1},Y_k)}. \end{aligned} \tag{6.4}$$

Equation 6.4 gives a mechanism to update the importance ratios sequentially. Since we can sample from the proposal function and evaluate the likelihood and transition probabilities (see equations 2.1 and 2.2), all we need to do is generate a prior set of samples and iteratively compute the importance ratios. This procedure allows us to obtain the following type of estimates:

$$\mathbf{E}[f_k(W_k)] \approx \sum_{i=1}^{N} \tilde{q}_k^{(i)} f_k(W_k^{(i)}). \tag{6.5}$$

In section 6.4 we give a more detailed description of a generic sequential importance sampling algorithm. We need to delay this presentation because the SIS algorithm discussed so far has a serious limitation that requires immediate attention: the variance of the ratio $(\mathrm{p}(W_k|Y_k)/\pi(W_k|Y_k))$ increases over time when the observations are regarded as random (Kong et al., 1994). To understand why the variance increase poses a problem, suppose that we want to sample from the posterior. In that case, we want the proposal density to be very close to the posterior density. When this happens, we obtain the following results for the mean and variance:

$$\mathbf{E}_\pi\left[\frac{\mathrm{p}(W_k|Y_k)}{\pi(W_k|Y_k)}\right] = 1$$

and

$$var_\pi\left[\frac{\mathrm{p}(W_k|Y_k)}{\pi(W_k|Y_k)}\right] = \mathbf{E}_\pi\left[\left(\frac{\mathrm{p}(W_k|Y_k)}{\pi(W_k|Y_k)} - 1\right)^2\right] = 0.$$

In other words, we expect the variance to be close to zero in order to obtain reasonable estimates. Therefore, a variance increase has a harmful effect on the accuracy of the simulations. In practice, the degeneracy caused by the variance increase can be observed by monitoring the importance ratios. Typically what we observe is that after a few iterations, one of the normalized importance ratios tends to 1, while the remaining ratios tend to zero.

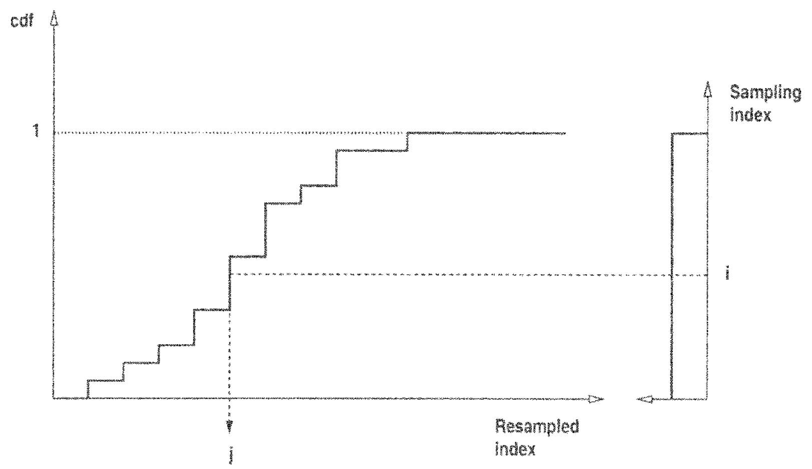

Figure 6: Resampling process, whereby a random measure $\{\mathbf{w}_k^{(i)}, \tilde{q}_k^{(i)}\}$ is mapped into an equally weighted random measure $\{\mathbf{w}_k^{(j)}, N^{-1}\}$. The index i is drawn from a uniform distribution.

6.1 Resampling. To avoid the degeneracy of the SIS simulation method, a resampling stage may be used to eliminate samples with low importance ratios and multiply samples with high importance ratios. It is possible to see an analogy to the steps in genetic algorithms. Many of the ideas on resampling have stemmed from the work of Efron (1982), Rubin (1988), and Smith and Gelfand (1992). Various authors have described efficient algorithms for accomplishing this task in $\mathcal{O}(N)$ operations (Doucet, 1998; Pitt & Shephard, 1997).

Resampling involves mapping the measure $\{\mathbf{w}_k^{(i)}, \tilde{q}_k^{(i)}\}$ into an equally weighted random measure $\{\mathbf{w}_k^{(j)}, N^{-1}\}$. This is accomplished by sampling uniformly from the discrete set $\{\mathbf{w}_k^{(i)}\}_{i=1}^N$ with probabilities $\{\tilde{q}_k^{(i)}\}_{i=1}^N$. A mathematical proof of this can be found in Gordon (1994, pp. 111–112). Figure 6 shows a way of sampling from this discrete set. After constructing the cumulative distribution of the discrete set, a uniformly drawn sampling index i is projected onto the distribution range and then onto the distribution domain. The intersection with the domain constitutes the new sample index j. That is, the vector $\mathbf{w}_k^{(j)}$ is accepted as the new sample. Clearly, the vectors with the larger sampling ratios will end up with more copies after the resampling process. In the subsequent prediction stage, random disturbances are added to the samples, thereby adding variety to the dominant samples, as was shown in Figure 2.

Liu and Chen (1995, 1998) have argued that when all the importance ratios are nearly equal, resampling only reduces the number of distinctive streams and introduces extra variation in the simulations. Using a result from Kong et al. (1994), they suggest that resampling should be performed only when the effective sample size N_{eff} is below a fixed heuristic threshold. The effective sample size is defined as

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{q}_k^{(i)}\right)^2}.$$

We have so far explained how to compute the importance ratios sequentially and how to improve the sample set by resampling. After discussing the choices of likelihood and proposal density functions, we shall be able to present a complete and detailed algorithm description.

6.2 Likelihood Function. The likelihood density function is defined in terms of the measurements equation. In our work, we adopted the following gaussian form:

$$\mathrm{p}(\mathbf{y}_k|\mathbf{w}_k) \propto \exp -\frac{1}{2}\left((\mathbf{y}_k - \hat{\mathbf{g}}(\mathbf{w}_k, \mathbf{x}_k))^T R^{-1} (\mathbf{y}_k - \hat{\mathbf{g}}(\mathbf{w}_k, \mathbf{x}_k))\right).$$

We also studied the effect of adding a weight decay term to the likelihood function to penalize for large network weights. This did not seem to improve the results in the experiments that we performed. In addition, it introduced an extra tuning parameter. If it is suspected that the data set may contain several outliers, likelihood functions with heavy tails should be employed.

6.3 Choosing the Proposal Function. The choice of proposal function is one of the most critical design issues in importance sampling algorithms. Doucet (1997) has shown that the proposal function:

$$\pi(\mathbf{w}_k|W_{k-1}, Y_k) = \mathrm{p}(\mathbf{w}_k|\mathbf{w}_{k-1}, \mathbf{y}_k) \tag{6.6}$$

minimizes the variance of the importance ratios conditional on W_{k-1} and Y_k. This choice of proposal function has also been advocated by other researchers (Kong et al., 1994; Liu & Chen, 1995; Zaritskii et al., 1975). Nonetheless, the density:

$$\pi(\mathbf{w}_k|W_{k-1}, Y_k) = \mathrm{p}(\mathbf{w}_k|\mathbf{w}_{k-1}) \tag{6.7}$$

is the most popular choice of proposal function (Avitzour, 1995; Beadle & Djuric, 1997; Gordon et al., 1993; Isard & Blake, 1996; Kitagawa, 1996). Although it results in higher Monte Carlo variation than the optimal proposal (see equation 6.6), as a result of its' not incorporating the most recent observations, it is usually easier to implement (Berzuini et al., 1997; Doucet, 1998; Liu & Chen, 1998).

In section 7, we describe an algorithm that allows us to update each sampling trajectory, with an extended Kalman filter, before the resampling stage. This Kalman update effectively constitutes a strategy for sampling from the prior of equation 6.6.

6.4 Algorithm Pseudocode. The essential structure of a sequential Monte Carlo algorithm to train neural networks with two layers, using the proposal of function of equation 6.7, involves the following steps:

```
1. INITIALISE NETWORK WEIGHTS ($k = 0$):
   For $i = 1, \dots, N$
      • Draw the weights $\mathbf{w}_0^{(i)}$ from the first layer prior
        $p_1(\mathbf{w}_0)$ and the second layer prior $p_2(\mathbf{w}_0)$.
      • Evaluate the importance ratios:
            $q_0^{(i)} = p(\mathbf{y}_0|\mathbf{w}_0^{(i)})$
      • Normalize the importance ratios:
            $\tilde{q}_0^{(i)} = \frac{q_0^{(i)}}{\sum_{j=1}^{N} q_0^{(j)}}$
2. For $k = 1, \dots, L$
     (a) SAMPLING STAGE:
         For $i = 1, \dots, N$
            • Predict via the dynamics equation:
                  $\hat{\mathbf{w}}_{k+1}^{(i)} = \mathbf{w}_k^{(i)} + \mathbf{d}_k^{(i)}$
              where $\mathbf{d}_k^{(i)}$ is a sample from $p(\mathbf{d}_k)$ ($\mathcal{N}(0, Q)$
              in our case).
            • Evaluate the importance ratios:
                  $q_{k+1}^{(i)} = q_k^{(i)} p(\mathbf{y}_{k+1}|\hat{\mathbf{w}}_{k+1}^{(i)})$
            • Normalize the importance ratios:
                  $\tilde{q}_{k+1}^{(i)} = \frac{q_{k+1}^{(i)}}{\sum_{j=1}^{N} q_{k+1}^{(j)}}$
     (b) RESAMPLING STAGE:
         For $i = 1, \dots, N$
         If $N_{eff} \geq$ Threshold:
            • $\mathbf{w}_{k+1}^{(i)} = \hat{\mathbf{w}}_{k+1}^{(i)}$
```

```
Else
    • Resample new index $j$ from the discrete
      set $\{\hat{\mathbf{w}}_{k+1}^{(i)}, \tilde{q}_{k+1}^{(i)}\}$
    • $\mathbf{w}_{k+1}^{(i)} = \hat{\mathbf{w}}_{k+1}^{(j)}$
    • $q_{k+1}^{(i)} = \frac{1}{N}$
```

An order N algorithm to perform the resampling stage is described in pseudocode by Carpenter et al. (1997). The generic sequential sampling algorithm is rather straightforward to implement using a gaussian likelihood function.[1] We make use of a different prior for each layer because the functional form of the neurons varies with layer (Müller & Rios Insua, 1998). In addition, we can select the prior hyperparameters for each layer in accordance with the magnitude of the input and output signals. Statistical estimates such as mean values and confidence intervals can be computed using equation 6.5.

6.5 Problems with Sequential SIR Simulation. The discreteness of the resampling stage (see Figure 6) implies that any particular sample with a high importance ratio will be duplicated many times. As a result, the cloud of samples may eventually collapse to a single sample. This degeneracy will limit the ability of the algorithm to search for lower minima in other regions of the error surface. In addition, the number of samples used to describe the posterior density function will become too small and inadequate. Gordon (1994) discusses various strategies to overcome this problem. He suggests that the sample set can be boosted by ensuring that the number of elements in the discrete distribution exceeds the posterior samples by a factor of at least 10. In addition, he proposes smoothing the discrete distribution with a kernel interpolator. Gordon et al. (1993) propose a roughening procedure, whereby an independent jitter, with zero mean and standard deviation σ, is added to each sample drawn in the resampling stage. The standard deviation of the jitter is given by:

$$\sigma = KIN^{-1/m}, \tag{6.8}$$

where I denotes the length of the interval between the maximum and minimum samples of each specific component, K is a tuning parameter, and, as before, m represents the number of weights. Large values of K blur the distribution, and very small values produce tight clusters of points around the original samples. We have found in our simulations that this technique works very well and that tuning K requires little effort.

[1] We have made the software for the implementation of the SIR algorithm available online at the following Web site: http://www.cs.berkeley.edu/~jfgf/.

Another serious problem arises when the likelihood function is much narrower than the prior. In this scenario, very few samples are accepted in the resampling stage. The problem is exacerbated when the regions of high likelihood are located in the tails of the prior density. This difficulty may be surmounted by the brute force method of increasing the sample size. Alternatively, one should select the prior variances and likelihood variance with great care, after an exploratory data analysis stage.

In our neural networks framework, we adopt different informative priors for each layer, whose variances depend on the magnitude of the input and output signals. Moreover, the covariance matrix of the network weights for each sampling trajectory is propagated and updated with an EKF. This leads to an improved annealed sampling algorithm, whereby the variance of each sampling trajectory decreases with time. That is, we start searching over a large region of the error surface, and as time progresses, we concentrate on the regions of lower error. This technique is described in the following section.

7 HySIR

The Monte Carlo conception of optimization relies solely on probing the error surface at several points, as shown in Figures 3 and 4. It fails to make use of all the information implicit in the error surface. For instance, the method could be enhanced by evaluating the gradient and other higher derivatives of the error surface. This is evident in Figure 7. By allowing the samples to follow the gradient after the first prediction stage, the algorithm can reach a better minimum.

In order to improve existing sequential Monte Carlo simulation algorithms, we propose a new hybrid gradient descent/SIR method. The main feature of the algorithm is that the samples are updated by a gradient descent step in the prediction stage, as follows:

$$\begin{aligned}\mathbf{w}_{k+1} &= \mathbf{w}_k - \frac{\alpha}{2}\frac{\partial}{\partial \mathbf{w}_k}(\mathbf{y}_k - \hat{\mathbf{g}}(\mathbf{w}_\mathrm{k}, \mathbf{x}_\mathrm{k}))^2 \\ &= \mathbf{w}_k + \alpha(\mathbf{y}_k - \hat{\mathbf{g}}(\mathbf{w}_k, \mathbf{x}_k))\frac{\partial \hat{\mathbf{g}}(\mathbf{w}_k, \mathbf{x}_k)}{\partial \mathbf{w}_k}. \qquad (7.1)\end{aligned}$$

The term $\partial\hat{\mathbf{g}}(\mathbf{w}_k, \mathbf{x}_k)/\partial\mathbf{w}_k$ is the Jacobian. It can be easily computed by back-propagating derivatives through the network as explained in appendix B of de Freitas et al. (1997). The gradient descent learning rate α is a tuning parameter that controls how fast we should descend. Too large a parameter would cause the trajectory in the error surface to overfluctuate, thereby never converging. A very small value, on the other hand, would not contribute toward speeding the SIR algorithm's convergence.

The plain gradient descent algorithm can get trapped at shallow local minima. One way of overcoming this problem is to define the error sur-

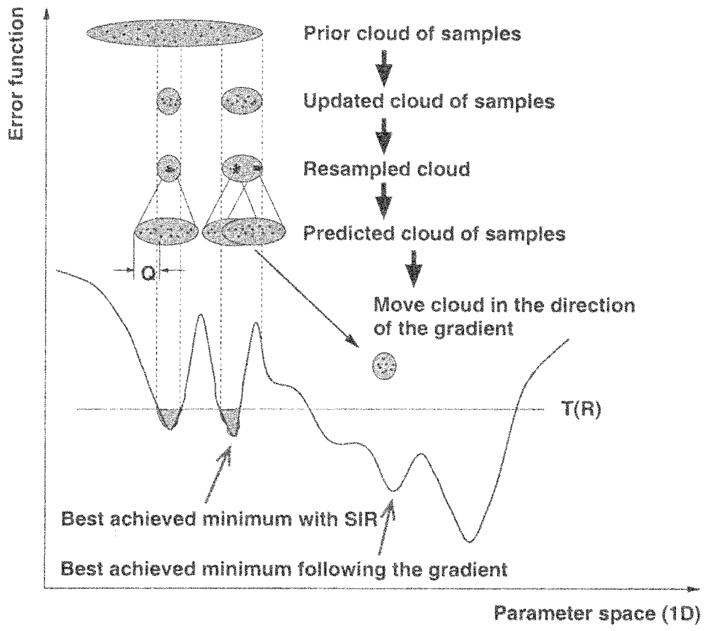
Error function
Prior cloud of samples
Updated cloud of samples
Resampled cloud
Predicted cloud of samples
Q
Move cloud in the direction of the gradient
T(R)
Best achieved minimum with SIR
Best achieved minimum following the gradient
Parameter space (1D)

Figure 7: HySIR algorithm. By following the gradient after the prediction stage, it is possible to reach better minima.

face in terms of a Hamiltonian that accounts for the approximation errors and the momentum of each trajectory. This is the basis of the hybrid Monte Carlo algorithm (Brass, Pendleton, Chen, & Robson, 1993; Duane, Kennedy, Pendleton, & Roweth, 1987). In this algorithm, each trajectory is updated by approximating the Hamiltonian differential equations by a leapfrog discretization scheme. The discretization, however, may introduce biases. In our work, we favor a stochastic gradient descent approach based on the EKF. This technique avoids shallow local minima by adaptively adding noise to the network weights. This is shown in Figure 8.

The EKF equations are given by:

$$\begin{aligned}
\mathbf{w}_{k+1|k} &= \mathbf{w}_k \\
P_{k+1|k} &= P_k^T + Q^* I_{mm} \\
K_{k+1} &= P_{k+1|k} G_{k+1} [R^* I_{oo} + G_{k+1}^T P_{k+1|k} G_{k+1}]^{-1} \\
\mathbf{w}_{k+1} &= \mathbf{w}_{k+1|K} + K_{k+1}(\mathbf{y}_{k+1} - \mathbf{g}(\mathbf{x}_k, \mathbf{w}_{k+1|k})) \\
P_{k+1} &= P_{k+1|k} - K_{k+1} G_{k+1}^T P_{k+1|k},
\end{aligned}$$

where K_{k+1} is known as the Kalman gain matrix, I_{mm} denotes the identity matrix of size $m \times m$, and R^* and Q^* are two tuning parameters, whose

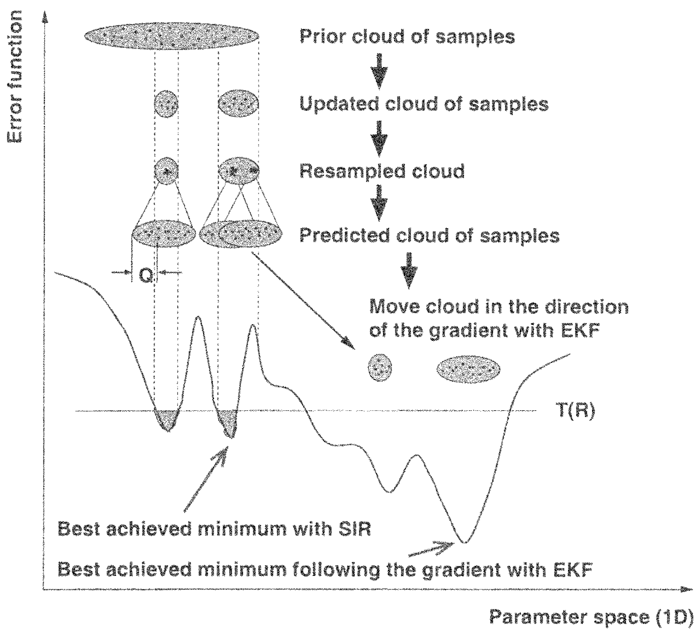
Error function
Prior cloud of samples
Updated cloud of samples
Resampled cloud
Predicted cloud of samples
Move cloud in the direction of the gradient with EKF
Q
T(R)
Best achieved minimum with SIR
Best achieved minimum following the gradient with EKF
Parameter space (1D)

Figure 8: HySIR with EKF updating. By adding noise to the network weights and adapting the weights covariance, it is possible to reach better minima with the EKF than with plain gradient descent techniques.

roles are explained in detail in de Freitas et al. (1997). Here, it suffices to say that they control the rate of convergence of the EKF algorithm and the generalization performance of the neural network. In the general multiple input, multiple output (MIMO) case, $\mathbf{g} \in \Re^o$ is a vector function and G represents the Jacobian matrix:

$$G = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{w}}\right|_{(\mathbf{w}=\hat{\mathbf{w}})} = \begin{bmatrix} \frac{\partial \mathbf{g}_1}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{g}_2}{\partial \mathbf{w}_1} & \cdots & \frac{\partial \mathbf{g}_o}{\partial \mathbf{w}_1} \\ \frac{\partial \mathbf{g}_1}{\partial \mathbf{w}_2} & & & \\ \vdots & & & \vdots \\ \frac{\partial \mathbf{g}_1}{\partial \mathbf{w}_m} & \cdots & & \frac{\partial \mathbf{g}_o}{\partial \mathbf{w}_m} \end{bmatrix}^T .$$

Since the EKF is a suboptimal estimator based on linearization of a nonlinear mapping, strictly speaking, P_k is an approximation to the covariance matrix. In mathematical terms:

$$P_k \approx \mathbf{E}[(\mathbf{w}_k - \hat{\mathbf{w}}_k)^T(\mathbf{w}_k - \hat{\mathbf{w}}_k)|Y_k].$$

The Kalman filter step, before the update stage, allows us to incorporate the latest measurements into the prior. As a result, we can sample from the prior $p(\mathbf{w}_k|\mathbf{w}_{k-1}, \mathbf{y}_k)$ before the update stage. Another advantage of this method is that the covariance of the weights changes over time. Because the EKF is a minimum variance estimator, the covariance of the weights decreases with time. Consequently, as the tracking improves, the variation of the network weights is reduced. This annealing process improves the efficiency of the search for global minima in parameter space and reduces the variance of the estimates. It should be pointed out that the weights need to be resampled in conjunction with their respective covariances. The pseudocode for the HySIR algorithm with EKF updating is as follows:[2]

```
1. INITIALIZE NETWORK WEIGHTS ($k = 0$):
2. For $k = 1, \ldots, L$
    (a) SAMPLING STAGE:
        For $i = 1, \ldots, N$
            • Predict via the dynamics equation:
                  $\hat{\mathbf{w}}_{k+1}^{(i)} = \mathbf{w}_k^{(i)} + \mathbf{d}_k^{(i)}$
              where $\mathbf{d}_k^{(i)}$ is a sample from $p(\mathbf{d}_k)$ ($\mathcal{N}(0, Q)$
              in our case).
            • Update samples with the EKF equations.
            • Evaluate the importance ratios:
                  $q_{k+1}^{(i)} = q_k^{(i)} \, p(\mathbf{y}_{k+1}|\hat{\mathbf{w}}_{k+1}^{(i)})$
            • Normalize the importance ratios:
                  $\tilde{q}_{k+1}^{(i)} = \frac{q_{k+1}^{(i)}}{\sum_{j=1}^{N} q_{k+1}^{(j)}}$
    (b) RESAMPLING STAGE:
        For $i = 1, \ldots, N$
        If $N_{eff} \geq$ Threshold:
            • $\mathbf{w}_{k+1}^{(i)} = \hat{\mathbf{w}}_{k+1}^{(i)}$
            • $P_{k+1}^{(i)} = \hat{P}_{k+1}^{(i)}$
```

[2] We have made the software for the implementation of the HySIR algorithm available online at the following web site: http://www.cs.berkeley.edu/~jfgf/.

```
Else
    • Resample new index $j$ from the discrete
      set $\{\hat{\mathbf{w}}_{k+1}^{(i)}, \tilde{q}_{k+1}^{(i)}\}$
    • $\mathbf{w}_{k+1}^{(i)} = \hat{\mathbf{w}}_{k+1}^{(j)}$ and $P_{k+1}^{(i)} = \hat{P}_{k+1}^{(j)}$
    • $q_{k+1}^{(i)} = \frac{1}{N}$
```

The HySIR with EKF updates may be interpreted as a dynamical mixture of EKFs. That is, the estimates depend on a few modes of the posterior density function. In addition to having to compute the normalized importance ratios ($\mathcal{O}(N(oS_1^2 + d^2S_1))$) operations, where S_1 is the number of hidden neurons, N the number of samples, d the input dimension, and o the number of outputs, and resampling ($\mathcal{O}(N)$ operations), the HySIR has to perform the EKF updates ($\mathcal{O}(Nom^2)$ operations, where m is the number of network weights). That is, the computational complexity of the HySIR increases approximately linearly with the number of EKFs in the mixture. For problems with dominant modes, we need to use only a few trajectories, and, consequently, the HySIR provides accurate and computationally efficient solutions.

The HySIR approach makes use of the likelihood function twice per time step. Consequently, if we make the noise distributions too narrow, the algorithm will give only a representation of a narrow region of the posterior density function. Typically, the algorithm will converge to a few modes of the posterior density function. This is illustrated in section 8.2.

8 Simulations

In this section, we discuss two experiments, using synthetic data and a real application involving the pricing financial options on the FTSE100 index. The first experiment provides a comparison of the various algorithms discussed so far. It addresses the trade-off between accuracy and computational efficiency. The second experiment shows how the HySIR algorithm can be used to estimate time-varying latent variables. It also serves the purpose of illustrating the effect of the process noise covariance Q on the simulations.

8.1 Experiment 1: Time-Varying Function Approximation. We generated input-output data using the following time-varying function:

$$y(x_1(k), x_2(k)) = 4\sin(x_1(k) - 2) + 2x_2(k)^2 + 5\cos(0.02k) + 5 + \nu,$$

where the inputs $x_1(k)$ and $x_2(k)$ were simulated from a gaussian distribution with zero mean and unit variance. The noise ν was drawn from a gaussian distribution with zero mean and variance equal to 0.1. Subsequently, we approximated the data with an MLP with five hidden sigmoidal neurons and a linear output neuron. The MLP was trained sequentially using the

Table 1: RMS Errors and Floating-Point Operations for the Algorithms Used in the Function Approximation Problem.

	EKF	**SIS**	**SIR**	**HySIR**
RMS error	6.51	3.87	3.27	1.17
Mega flops	4.8	5.6	5.8	48.6
Time (seconds)	1.2	27.4	28.9	24.9

Notes: The RMS errors include the convergence period. This explains why they are higher than the variance of the additive noise.

HySIR, SIR, SIS, and EKF algorithms. The difference between the SIR and the SIS algorithms is that SIR resamples every time, while SIS resamples only if the effective sample set is below a threshold. We set our threshold to $N/3$.

We set the number of samples (N) to 100 for the SIS and SIR methods and to 10 for the HySIR method. We assigned the values 100 and 1 to the initial variance of the weights and the diagonal entries of the weights covariance matrix. The diagonal entries of R and Q were given the values 0.5 and 2, respectively. Finally, the EKF noise hyperparameters R^* and Q^* were set to 2 and 0.01.

Table 1 shows the average one-step-ahead prediction errors obtained for 100 runs, of 200 time steps each, on a Silicon Graphics R10000 workstation. On average, there was a 50% occurrence of resampling steps using the SIS algorithm. It is clear from the results that avoiding the resampling stage does not yield a great reduction in computational time. This is because one needs to evaluate neural network mappings for each trajectory in the sampling stage. As a result, the sampling stage tends to be much more computationally expensive than the resampling stage.

The results also show that the HySIR performs much better than the conventional SIR and EKF algorithms. It is, however, computationally more expensive, as explained at the end of section 7.

It is interesting to note that the HySIR, despite requiring more floating-point operations, is faster than the SIR algorithm. This shows that the computational efficiency of sampling algorithms depends on the platform in which they are implemented. The design of efficient software and hardware platforms is an important avenue for further research.

Figure 9 shows two plots of the prediction errors for the EKF and HySIR. The left plot shows the best and worst results obtained with the EKF out of 100 trials, each trial involving a different initial weights vector. We then used the 100 initial weights vectors to initialize the HySIR algorithm with 100 sampling trajectories. As shown in the right plot, the global search nature of the HySIR algorithm allows it to converge much faster than the EKF algorithm. This is a clear example that dynamic mixtures of models,

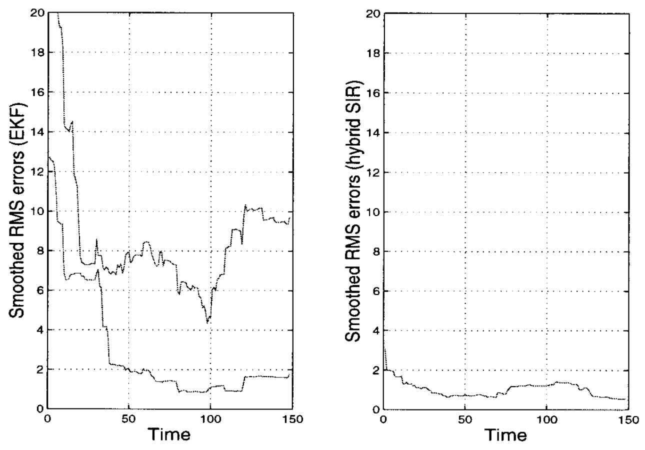

Figure 9: Convergence of the EKF and HySIR algorithms for experiment 1. The left plot shows the best and worst EKF convergence results out of 100 runs using different initial conditions. The initial conditions for each EKF were used to initialize the EKF filters within the HySIR algorithm. At each time step, the HySIR evaluates a weighted combination of several EKFs. By performing this type of dynamic model selection, it is able to search over large regions of parameter space and, hence, converge much faster than the standard EKF algorithm.

with model selection at each time step, perform better than static mixtures (choosing the best EKF out of the 100 trials).

8.2 Experiment 2: Latent States Tracking. To assess the ability of the hybrid algorithm to estimate time-varying hidden parameters, we generated input-output data from a logistic function followed by a linear scaling and a displacement, as shown in Figure 10. We applied two gaussian input sequences to the model. This simple model is equivalent to an MLP with one hidden neuron and an output linear neuron. We then trained a second model with the same structure using the input-output data generated by the first model.

In the training phase, of 200 time steps, we allowed the model weights to vary with time. During this phase, the HySIR algorithm was used to track the input-output training data and estimate the latent model weights. After the 200th time step, we fixed the values of the weights and generated another 200 input-output data test sets from the original model. The input test data were then fed to the trained model, using the weights values estimated at the 200th time step. Subsequently, the output prediction of the trained model was compared to the output data from the original model so as to assess the generalization performance of the training process.

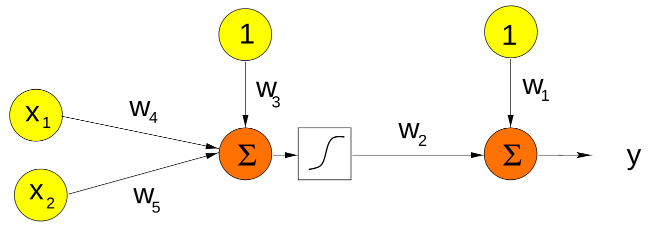

Figure 10: Logistic function with linear scaling and displacement used in experiment 2. The weights were chosen as follows: $\mathbf{w}_1(k) = 1 + k/100$, $\mathbf{w}_2(k) = \sin(0.06k) - 2$, $\mathbf{w}_3(k) = 0.1$, $\mathbf{w}_4(k) = 1$, $\mathbf{w}_5(k) = -0.5$.

We repeated the experiment using two different values of Q, as shown in Figures 11 and 12. With a very small value of Q (0.0001), the algorithm was able to estimate the time-varying latent weights. After 100 time steps, the algorithm became very confident, as indicated by the narrow error bars. By increasing Q to 0.01, we found that the algorithm becomes less confident. Yet it is able to explore wider regions of parameter space, as shown in the histogram of Figure 13.

The variance (Q) of the noise added in the prediction stage implies a trade-off between the algorithm's confidence and the size of the search region in parameter space. Too large a value of Q will allow the algorithm to explore a large region of the space, but will never allow the algorithm to converge to specific error minimum. A very small value, on the other hand, will cause all the sampling trajectories to converge to a very narrow region. The problem with the latter is that if the process generating the data changes with time, we want to keep the search region large enough so as to find the new minima quickly and efficiently. To address this problem, we have employed a very simple heuristic. Specifically, when we add gaussian noise in the prediction stage, we select a small set of the trajectories randomly and then add to them gaussian noise with a covariance much larger than Q. In theory, this should allow the algorithm to converge and yet allow it to keep exploring other regions of parameter space. From an evolutionary computing perspective, we can think of these few randomly selected trajectories as mutations. The results of applying this idea to our problem are shown in Figure 13. They suggest that our heuristic reduces the trade-off between the algorithms' confidence and the size of the search region. Yet we believe that further research on algorithms for adapting Q is necessary.

8.3 Example with Real Data. An interesting source of inference problems requiring nonlinear modeling in a time-varying environment is financial data analysis. Many applications ranging from time-series prediction to stock selection have been attempted using neural networks. Hutchinson,

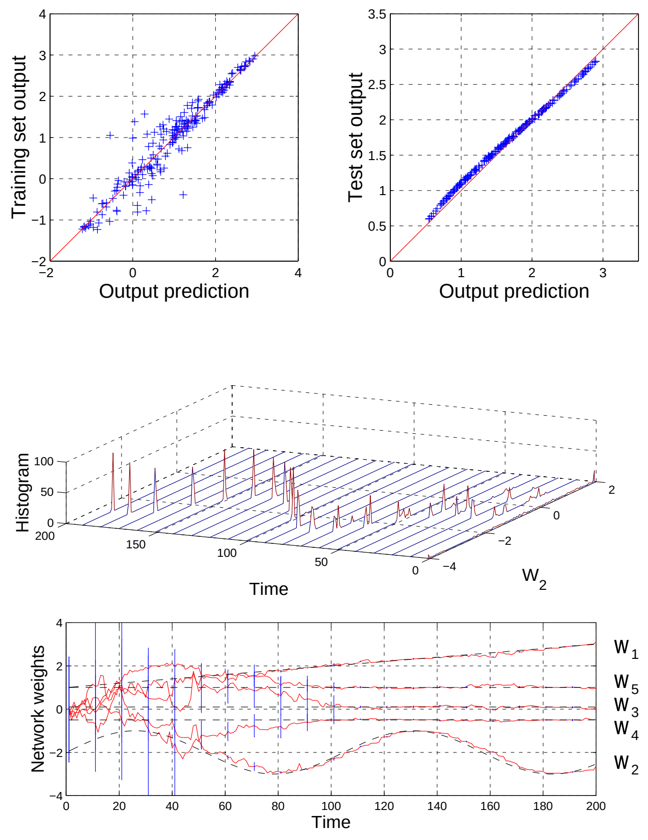

Figure 11: Weights tracking with $Q = 0.0001$. (Top left) One-step-ahead prediction errors in the training set. (Top right) Prediction errors on validation data, assuming that after the 200th time step, the weights stop varying with time. (Middle) Histogram of $\mathbf{w}_2$. (Bottom) Tracking of the weights with the posterior mean estimate computed by the HySIR and the one standard deviation error bars of the estimator. For this particular value of Q, the estimator becomes very confident after 100 iterations, and, consequently, the histogram for $\mathbf{w}_2$ converges to a narrow gaussian distribution.

Lo, and Poggio (1994) and Niranjan (1996) look at the problem of pricing options contracts in a data-driven manner. The relationship between the price of an options contract and the parameters relating to the option, such as the price of the underlying asset and the time to maturity, is known to be nonlinear. Hutchinson et al. show that a neural network can form a very good approximation to the price of the options contract as a function of variables relating to this. This relationship is also likely to change over the

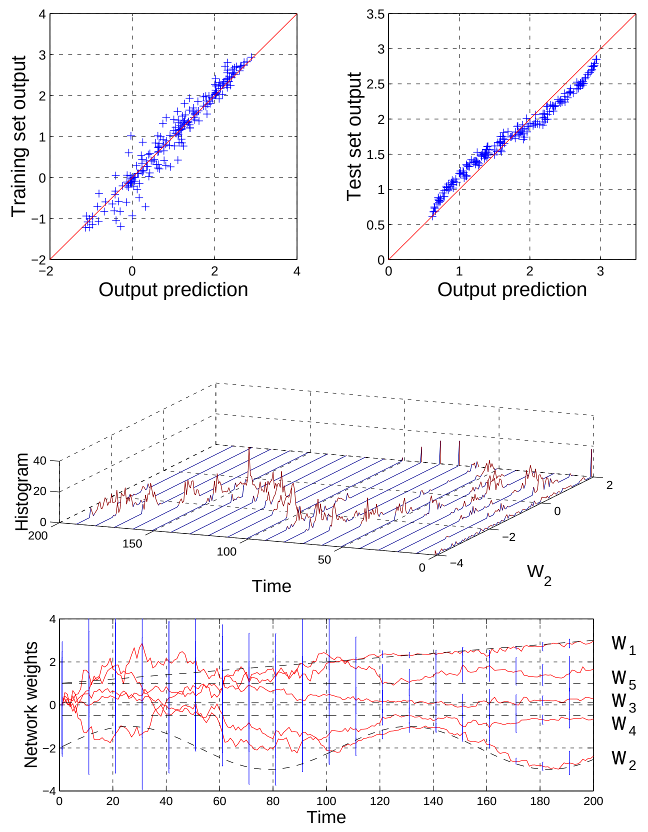

Figure 12: Weights tracking with $Q = 0.01$. (Top left) One-step-ahead prediction errors in the training set. (Top right) Prediction errors on validation data, assuming that after the 200th time step, the weights stop varying with time. (Middle) Histogram of $\mathbf{w}_2$. (Bottom) Tracking of the weights with the posterior mean estimate computed by the HySIR and the one standard deviation error bars of the estimator. For this value of Q, the search region is particularly large (wide histograms), but the algorithm lacks confidence (wide error bars).

lifetime of the option. Hence, one would expect that a sequential approach for tracking the price of the options contract might be a more natural way to handle the nonstationarity. Niranjan addresses this by means of an EKF. We extend on the above work, using the data in Niranjan (1996) consisting of daily FTSE100 options during the February 93–December 93 period. We compare the approximation and convergence capabilities of a number of algorithms discussed in earlier sections. The models considered are:

Trivial Uses the current value of the option as the next prediction.

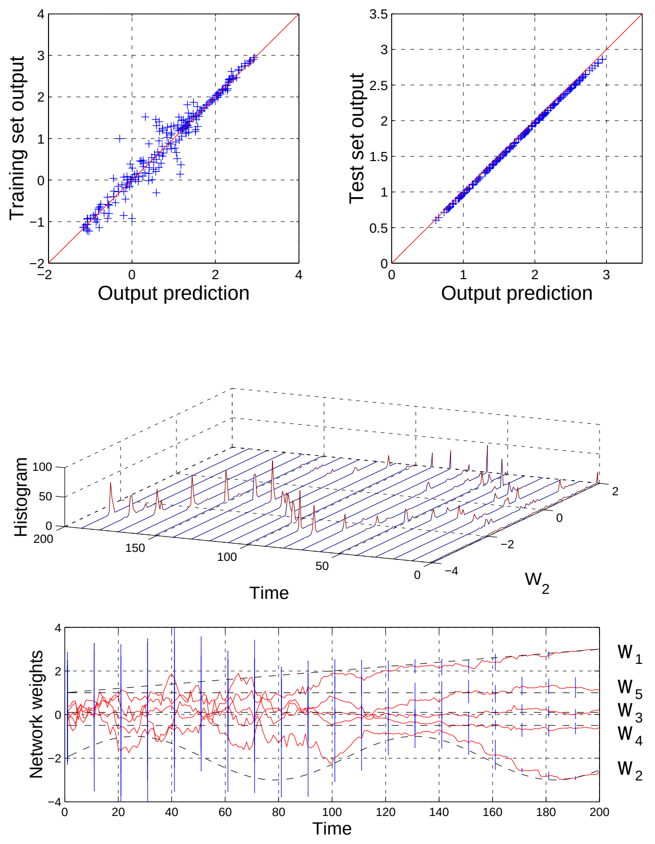

Figure 13: Weights tracking with $Q = 0.0001$ and two random mutations per time step with $Q = 0.1$. (Top left) One-step-ahead prediction errors in the training set. (Top right) Prediction errors on validation data, assuming that after the 200th time step, the weights stop varying with time. (Middle) Histogram of $\mathbf{w}_2$. (Bottom) Tracking of the weights with the posterior mean estimate computed by the HySIR and the one standard deviation error bars of the estimator. The histogram shows that the mutations expand the search region considerably.

RBF-EKF. Represents a regularized radial basis function network with four hidden neurons, which was originally proposed in Hutchinson et al. (1994). The output weights are estimated with a Kalman filter, while the means of the radial functions correspond to random subsets of the data and their covariance is set to the identity matrix, as in Niranjan (1996).

BS. Corresponds to a conventional Black-Scholes model with two outputs (normalised call and put prices) and two parameters (risk-free interest rate and volatility). The risk-free interest rate was set to 0.06, while the volatility was estimated over a moving window (of 50 time steps), as described in Hull (1997).

MLP-EKF. A multilayer perceptron with six sigmoidal hidden units and a linear output neuron, trained with the EKF algorithm (de Freitas et al., 1997). The noise covariances R and Q and the states covariance P were set to diagonal matrices with entries equal to 10^{-6}, 10^{-7}, and 10, respectively. The weights prior corresponded to a zero-mean gaussian density with covariance equal to 1.

MLP-EKFE. Represents an MLP, with six sigmoidal hidden units and a linear output neuron, trained with a hierarchical Bayesian model, whereby the weights are estimated with the EKF algorithm and the noise statistics are computed by maximizing the evidence density function (de Freitas et al., 1997). The states covariance P was given by a diagonal matrix with entries equal to 10. The weights prior corresponded to a zero mean gaussian density with covariance equal to 1.

MLP-SIR. Corresponds to an MLP, with six sigmoidal hidden units and a linear output neuron, trained with the SIR algorithm. One thousand samples were employed in each simulation. The noise covariances R and Q were set to diagonal matrices with entries equal to 10^{-4} and 10^{-5}, respectively. The prior samples for each layer were drawn from a zero-mean gaussian density with covariance equal to 1.

MLP-HySIR. Corresponds to an MLP, with six sigmoidal hidden units and a linear output neuron, trained with the Kalman HySIR algorithm. Each simulation made use of 10 samples. The noise covariances R and Q were set to diagonal matrices with entries equal to 1 and 10^{-6}, respectively. The prior samples for each layer were drawn from a zero-mean gaussian density with covariance equal to 1. The Kalman filter parameters R^*, Q^* and P were set to diagonal matrices with entries equal to 10^{-4}, 10^{-5}, and 1000, respectively.

Table 2 shows the mean squared error between the actual value of the options contracts and the one-step-ahead prediction from the various algorithms. Although the differences are small, one sees that sequential algorithms produce lower prediction errors and that the HySIR algorithm produces even lower errors than the Kalman filter. These errors are computed on data corresponding to the last 100 days, to allow for convergence of the algorithms. Note that in the on-line setting considered here, all of these errors are computed on unseen data.

Figure 14 shows the approximations of the MLP-HySIR algorithm on the call and put option prices at the same strike price (3325). The algorithm exhibits fast convergence and accurate tracking. In Figure 15, we plot the probability density function over the network's one-step-ahead prediction, computed through 100 samples propagated by the HySIR approach.

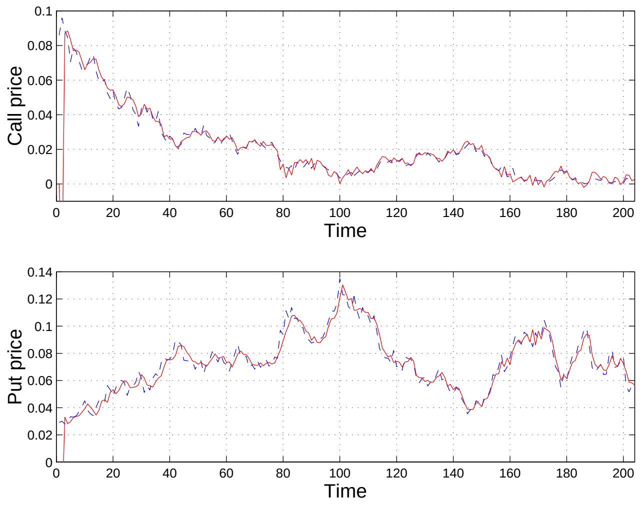

Figure 14: Tracking call and put option prices with the MLP-HSIR method. Estimated values [—] and actual values [- -].

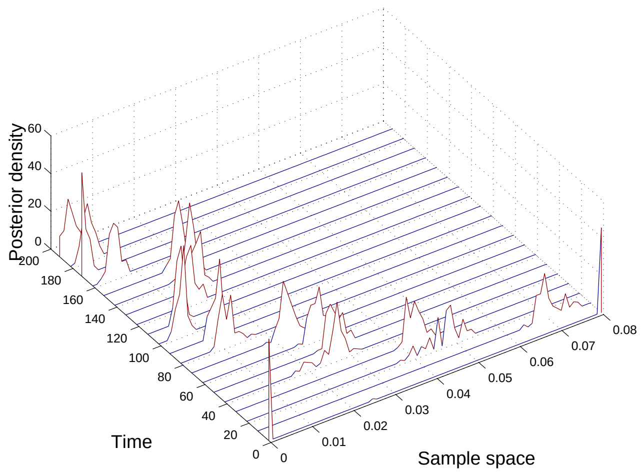

Figure 15: Probability density function of the neural networks' one-step-ahead prediction of the call price for the FTSE option with a strike price of 3325.

Table 2: One-Step-Ahead Prediction Errors on Call Options.

Strike Price	2925	3025	3125	3225	3325
Trivial	0.0783	0.0611	0.0524	0.0339	0.0205
RBF-EKF	0.0538	0.0445	0.0546	0.0360	0.0206
BS	0.0761	0.0598	0.0534	0.0377	0.0262
MLP-EKF	0.0414	0.0384	0.0427	0.0285	0.0145
MLP-EKFE	0.0404	0.0366	0.0394	0.0283	0.0150
MLP-SIR	0.0419	0.0405	0.0432	0.0314	0.0164
MLP-HySIR	0.0394	0.0362	0.0404	0.0273	0.0151

Typically the one-step-ahead predictions for a group of options on the same cash product, but with different strike prices or duration to maturity, can be used to determine whether one of the options is being mispriced. Knowing the probability distribution of the network outputs allows us to design more interesting pricing tools.

9 Conclusions

We have presented a sequential Monte Carlo approach for training neural networks in a Bayesian setting. This approach enables accurate characterization of the probability distributions, which tend to be very complicated for nonlinear models such as neural networks. Propagating a number of samples in a state-space dynamical systems formulation of the problem is the key idea in this framework.

Experimental illustrations presented here suggest that it is possible to apply sampling methods effectively to tackle difficult problems involving nonlinear and time-varying processes, typical of many sequential signal processing problems.

A particular contribution is the HySIR algorithm, which combines gradient information of the error function with probabilistic sampling information. Simulations show that this combination performs better than EKF and conventional SIR approaches. HySIR can be interpreted as a gaussian mixture filter, in that only a few sampling trajectories need to be employed. Yet as the number of trajectories increases, the computational requirements increase only linearly. Therefore, the method is also suitable as a sampling strategy for multimodal optimization.

Our experiment with financial data showed that the sequential sampling approach, in addition to matching the one-step-ahead square errors of the best available pricing methods, allowed us to obtain complete estimates of the probability density functions of the one-step-ahead predictions. Further avenues of research include the design of algorithms for adapting the noise covariances R and Q, studying the effect of different noise models for the network weights, and improving the computational efficiency of the algorithms.

10 Acknowledgments

We thank Neil Gordon (DERA, U.K.), Sue Johnson (Department of Engineering, Cambridge), and Michael Isard (Computer Vision Group, Oxford) for interesting and useful discussions on this work. We are also very grateful to the referees for their valuable comments. J. F. G. F. is financially supported by two University of the Witwatersrand Merit Scholarships, a Foundation for Research Development Scholarship (South Africa), an ORS award, and a Trinity College External Studentship (Cambridge).

References

Akashi, H., & Kumamoto, H. (1977). Random sampling approach to state estimation in switching environments. *Automatica, 13*, 429–434.

Avitzour, D. (1995). A stochastic simulation Bayesian approach to multitarget tracking. *IEE Proceedings on Radar, Sonar and Navigation, 142*(2), 41–44.

Bar-Shalom, Y., & Li, X. R. (1993). *Estimation and tracking: Principles, techniques and software*. Boston: Artech House.

Beadle, E. R., & Djuric, P. M. (1997). A fast weighted Bayesian bootstrap filter for nonlinear model state estimation. *IEEE Transactions on Aerospace and Electronic Systems, 33*(1), 338–343.

Berzuini, C., Best, N. G., Gilks, W. R., & Larizza, C. (1997). Dynamic conditional independence models and Markov chain Monte Carlo methods. *Journal of the American Statistical Association, 92*(440), 1403–1412.

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

Brass, A., Pendleton, B. J., Chen, Y., & Robson, B. (1993). Hybrid Monte Carlo simulations theory and initial comparison with molecular dynamics. *Biopolymers, 33*(8), 1307–1315.

Carpenter, J., Clifford, P., & Fearnhead, P. (1997). *An improved particle filter for nonlinear problems* (Tech. Rep.). Oxford: Department of Statistics, Oxford University. Available online at http://www.stats.ox.ac.uk/~clifford/index.htm.

Collings, I. B., Krishnamurthy, V., & Moore, J. B. (1994). On-line identification of hidden Markov models via recursive prediction error techniques. *IEEE Transactions on Signal Processing, 42*(12), 3535–3539.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems, 2*(4), 303–314.

de Freitas, J. F. G., Niranjan, M., & Gee, A. H. (1997). *Hierarchical Bayesian-Kalman models for regularisation and ARD in sequential learning* (Tech. Rep. No. CUED/F-INFENG/TR 307). Cambridge: Cambridge University. Available online at: http://svr-www.eng.cam.ac.uk/~jfgf.

de Freitas, J. F. G., Niranjan, M., & Gee, A. H. (1998a). *The EM algorithm and neural networks for nonlinear state space estimation* (Tech. Rep. CUED/F-INFENG/TR

313). Cambridge: Cambridge University. Available online at: http://svr-www.eng.cam.ac.uk/~jfgf.

de Freitas, J. F. G., Niranjan, M., & Gee, A. H. (1998b). Nonlinear state space learning with EM and neural networks. In *IEEE International Workshop on Neural Networks in Signal Processing*. Cambridge, England.

de Freitas, J. F. G., Niranjan, M., & Gee, A. H. (1998c). Regularisation in sequential learning algorithms. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in neural information processing systems, 10*. Cambridge, MA: MIT Press.

Depster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B, 39*, 1–38.

Doucet, A. (1997). *Monte Carlo methods for Bayesian estimation of hidden Markov models: Application to radiation signals*. Unpublished doctoral dissertation, University Paris-Sud, Orsay, France.

Doucet, A. (1998). *On sequential simulation-based methods for Bayesian filtering* (Tech. Rep. No. CUED/F-INFENG/TR 310). Cambridge: Cambridge University. Available online at: http://www.stats.bris.ac.uk:81/MCMC/pages/list.html.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B, 195*(2), 216–222.

Efron, B. (1982). *The bootstrap, jacknife and other resampling plans*. Philadelphia: SIAM.

Elliott, R. J. (1994). Exact adaptive filters for Markov chains observed in gaussian noise. *Automatica, 30*(9), 1399–1408.

Gelfand, A. E., Dey, D. K., & Chang, H. (1992). Model determination using predictive distributions with implementation via sampling-based methods. In J. M. Bernardo, J. O. Berger, A. P. Dawid, & A. F. M. Smith (Eds.), *Bayesian statistics 4* (pp. 147–167). Oxford: Oxford University Press.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). *Bayesian data analysis*. London: Chapman and Hall.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica, 24*, 1317–1399.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (Eds.). (1996). *Markov chain Monte Carlo in practice*. Soffolk: Chapman and Hall.

Gordon, N., & Whitby, A. (1995). A Bayesian approach to target tracking in the presence of a glint. In O. E. Drummond (Ed.), *Signal and data processing of small targets* (pp. 472–483). Bellingham, WA: SPIE Press.

Gordon, N. J. (1994). *Bayesian methods for tracking*. Unpublished doctoral dissertation, Imperial College, University of London.

Gordon, N. J., Salmond, D. J., & Smith, A. F. M. (1993). Novel appraoch to nonlinear/nonGaussian Bayesian state estimation. *IEE Proceedings-F, 140*(2), 107–113.

Hammersley, J. H., & Handscomb, D. C. (1968). *Monte Carlo methods*. London: Methuen.

Handschin, J. E. (1970). Monte Carlo techniques for prediction and filtering of non-linear stochastic processes. *Automatica, 6*, 555–563.

Handschin, J. E., & Mayne, D. Q. (1969). Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control, 9*(5), 547–559.

Harvey, A. C. (1970). *Forecasting, structural time series models, and the Kalman filter*. Cambridge: Cambridge University Press.

Hull, J. C. (1997). *Options, futures, and other derivative* (3rd ed.). Englewood Cliffs, NJ: Prentice Hall.

Hutchinson, J. M., Lo, A. W., & Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance, 49*(3), 851–889.

Isard, M., & Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision* (pp. 343–356). Cambridge, UK.

Jazwinski, A. H. (1969). Adaptive filtering. *Automatica, 5*, 475–485.

Jazwinski, A. H. (1970). *Stochastic processes and filtering theory*. Orlando, FL: Academic Press.

Kadirkamanathan, V., & Kadirkamanathan, M. (1995). Recursive estimation of dynamic modular RBF networks. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems, 8* (pp. 239–245). Cambridge, MA: MIT Press.

Kitagawa, G. (1987). Non-Gaussian state-space modeling of nonstationary time series. *Journal of the American Statistical Association, 82*(400), 1032–1063.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics, 5*, 1–25.

Kong, A., Liu, J. S., & Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association, 89*(425), 278–288.

Kramer, S. C., & Sorenson, H. W. (1988). Recursive Bayesian estimation using piece-wise constant approximations. *Automatica, 24*(6), 789–801.

Krishnamurthy, V., & Moore, J. B. (1993). On-line estimation of hidden Markov model parameters based on the Kullback-Leibler information measure. *IEEE Transactions on Signal Processing, 41*(8), 2557–2573.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., & Hubbard, W. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation, 1*, 541–551.

Li, X. R., & Bar-Shalom, Y. (1994). A recursive multiple model approach to noise identification. *IEEE Transactions on Aerospace and Electronic Systems, 30*(3), 671–684.

Liu, J. S., & Chen, R. (1995). Blind deconvolution via sequential imputations. *Journal of the American Statistical Association, 90*(430), 567–576.

Liu, J. S., & Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association, 93*, 1032–1044.

Mackay, D. J. C. (1992). Bayesian interpolation. *Neural Computation, 4*(3), 415–447.

Müller, P. (1991). Monte Carlo integration in general dynamic models. *Contemporary Mathematics, 115*, 145–163.

Müller, P. (1992). Posterior integration in dynamic models. *Computer Science and Statistics, 24*, 318–324.

Müller, P., & Rios Insua, D. (1998). Issues in Bayesian analysis of neural network models. *Neural Computation, 10*, 571–592.

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

Niranjan, M. (1996). Sequential tracking in pricing financial options using model based and neural network approaches. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems 9* (pp. 960–966). Cambridge, MA: MIT Press.

Pitt, M. K., & Shephard, N. (1997). *Filtering via simulation: Auxiliary particle filters* (Tech. Rep.). London: Department of Statistics, Imperial College of London. Available online at: http://www.nuff.ox.ac.uk/economics/papers.

Pole, A., & West, M. (1990). Efficient Bayesian learning on non-linear dynamic models. *Journal of Forecasting, 9*, 119–136.

Puskorious, G. V., & Feldkamp, L. A. (1991). Decoupled extended Kalman filter training of feedforward layered networks. *International Joint Conference on Neural Networks* (pp. 307–312). Seattle.

Puskorius, G. V., Feldkamp, L. A., & Davis, L. I. (1996). Dynamic neural network methods applied to on-vehicle idle speed control. *Proceedings of the IEEE, 84*(10), 1407–1420.

Rauch, H. E., Tung, F., & Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal, 3*(8), 1445–1450.

Robinson, T. (1994). The application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks, 5*(2), 298–305.

Rubin, D. B. (1988). Using the SIR algorithm to simulate posterior distributions. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, & A. F. M. Smith (Eds.), *Bayesian statistics, 3* (pp. 395–402). Oxford: Oxford University Press.

Ruck, D. W., Rogers, S. K., Kabrisky, M., Maybeck, P. S., & Oxley, M. E. (1992). Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 14*(6), 686–690.

Shah, S., Palmieri, F., & Datum, M. (1992). Optimal filtering algorithms for fast learning in feedforward neural networks. *Neural Networks, 5*, 779–787.

Sibisi, S. (1989). Regularization and inverse problems. In J. Skilling (Ed.), *Maximum entropy and Bayesian methods* (pp. 389–396). Norwell, MA: Kluwer.

Singhal, S., & Wu, L. (1988). Training multilayer perceptrons with the extended Kalman algorithm. In D. S. Touretzky (Ed.), *Advances in neural information processing systems, 1* (pp. 133–140). San Mateo, CA: Morgan Kaufmann.

Smith, A. F. M., & Gelfand, A. E. (1992). Bayesian statistics without tears: A sampling-resampling perspective. *American Statistician, 46*, 84–88.

Sorenson, H. W., & Alspach, D. L. (1971). Recursive Bayesian estimation using gaussian sums. *Automatica, 7*, 465–479.

Zaritskii, V. S., Svetnik, V. B., & Smihelevich, L. I. (1975). Monte-Carlo techniques in problems of optimal information processing. *Automatic Remote Control, 36*, 2015–2022.

Received July 17, 1998; accepted March 12, 1999.