To be published in: Neural Networks for Signal Processing 5, B. Wilson *et al.*, eds., IEEE Signal Processing Society, Piscataway, NJ, 1995.

Missing and Noisy Data in Nonlinear Time-Series Prediction

Volker Tresp and Reimar Hofmann
Siemens AG, Central Research
81730 Munich, Germany*

Abstract

We discuss the issue of missing and noisy data in nonlinear time-series prediction. We derive fundamental equations both for prediction and for training. Our discussion shows that if measurements are noisy or missing, treating the time series as a static input/output mapping problem (the usual time-delay neural network approach) is suboptimal. We describe approximations of the solutions which are based on stochastic simulations. A special case is K-step prediction in which a one-step predictor is iterated K times. Our solutions provide error bars for prediction with missing or noisy data and for K-step prediction. Using the K-step iterated logistic map as an example, we show that the proposed solutions are a considerable improvement over simple heuristic solutions. Using our formalism we derive algorithms for training recurrent networks, for control of stochastic systems and for reinforcement learning problems.

1 Introduction

Missing data in time-series prediction are a common problem in many applications. The goal is to obtain valid predictions even if some measurements become unavailable or are not recorded. Similarly, training data are often incomplete. In this paper we analyze this problem from a probabilistic point of view. In previous publications the problem of learning and prediction with missing and noisy features in (static) estimation problems was examined (see, for example [2, 3, 4]). The solutions for both prediction and learning consisted of integrals over the unknown variable weighted by the conditional probability density of the unknown variable given the known variables. The basic idea is the same for missing data in time-series prediction, but here, we can exploit the fact that the missing measurement itself is part of the time series. Similar issues arise if we can only obtain noisy measurements of the underlying true time series.

*Volker.Tresp@zfe.siemens.de, Reimar.Hofmann@zfe.siemens.de

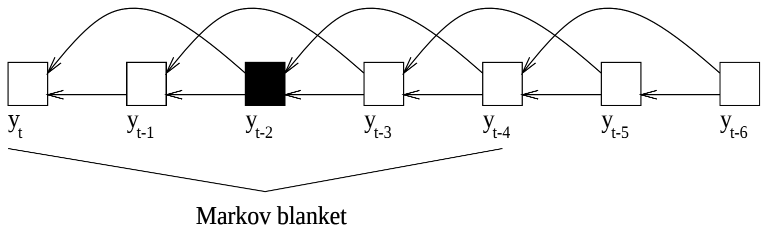

Figure 1: A time series unfolded in time. The arrows indicate that the next realization of the time series can be predicted from the two most recent values, $y_t = f(y_{t-1}, y_{t-2}) + \epsilon_t$. Here, y_{t-2} is assumed to be missing. The Markov blanket shows which variables are relevant for estimating y_{t-2}.

In this paper, we provide solutions for the problem of prediction and training with missing or noisy data in time-series predictions. As a special case, we consider K-step prediction in which a one-step predictor is iterated K times. We show how error bars can be derived for prediction with missing and noisy inputs and for K-step prediction. Finally, we point out that the learning algorithms derived in this paper can be used to train recurrent neural networks, for stochastic control and for reinforcement learning problems.

2 Prediction with Missing Inputs

2.1 One Missing Realization

We assume that the underlying probabilistic model of the time series can be described by

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-N}) + \epsilon_t \tag{1}$$

where $f()$ is either known or approximated sufficiently well by a function approximator such as a neural network. ϵ_t is assumed to be additive uncorrelated zero-mean noise with probability density $P_\epsilon(\epsilon)$ and typically represents unmodeled dynamics. The conditional probability density of the predicted instance of the time series is then

$$P(y_t|y_{t-1}, y_{t-2}, \ldots, y_{t-N}) = P_\epsilon(y_t - f(y_{t-1}, y_{t-2}, \ldots, y_{t-N})). \tag{2}$$

Often, Gaussian noise is assumed such that

$$P(y_t|y_{t-1}, y_{t-2}, \ldots, y_{t-N}) = G(y_t; f(y_{t-1}, \ldots, y_{t-N}), \sigma^2) \tag{3}$$

where $G(x; c, \sigma^2)$ is our notation for a normal density evaluated at x with center c and variance σ^2.

It is convenient to unfold the system in time which leads to the system shown in Figure 1. The realizations of the time series are now random variables in a probabilistic network. Our problem is to predict y_t using the available information. According to our assumptions, the joint probability density is

$$P(y_1, y_2, \ldots, y_t) = P(y_1, \ldots, y_N) \prod_{l=N+1}^{t} P(y_l | y_{l-1}, \ldots, y_{l-N}). \tag{4}$$

Let's now assume that y_{t-k} with $k \leq N$ is missing. Let $y^u = \{y_{t-k}\}$ and let $y^m = \{y_{t-1} \ldots y_{t-k-N}\} \setminus \{y_{t-k}\}$. We can calculate the expected value of the next realization of the time series as

$$E(y_t | M_{t-1}) = \int f(y_{t-1}, \ldots, y_{t-k}, \ldots, y_{t-N}) \; P(y^u | y^m) \; dy^u \tag{5}$$

where M_{t-1} stands for all measurements up to $t-1$. The last equation is the fundamental equation for prediction with missing data. Note, that the unknown y_{t-k} is not only dependent on realizations of the time series *previous* to $t-k$ but also on measurements *after* $t-k$. The reason is that the variables in $y^m \cup y_t$ form a minimal Markov blanket of y_{t-k} in the Bayesian net in Figure 1. A minimal Markov blanket in a Bayesian network consists of the direct parents, the direct successors of a variable and all direct parents of a variables direct successors. In our case, the direct successors are $y_t \ldots y_{t-k+1}$, the direct parents are $y_{t-k-1} \ldots y_{t-k-N}$ and the direct parents of a variables direct successor are $y_{t-1} \ldots y_{t-k-N+1}$. The theory of Bayesian and Markov networks now tells us that a variable is independent of all other variables in the network if the variables in the Markov blanket are known (see Figure 1). This discussion shows that simply approximating $y_{t-k} \approx f(y_{t-k-1}, y_{t-k-2}, \ldots, y_{t-k-N})$ is suboptimal. The required conditional density in Equation 5 is (recall that $y^u = y_{t-k}$)

$$P(y^u | y^m) \propto P(y_{t-1} | y_{t-2}, \ldots, y_{t-k}, \ldots, y_{t-1-N})$$

$$\times P(y_{t-2} | y_{t-3}, \ldots, y_{t-k}, \ldots, y_{t-2-N}) \ldots P(y_{t-k} | y_{t-k-1}, \ldots, y_{t-k-N}).$$

This expression can be evaluated easily using Equation 1 or in the Gaussian noise case Equation 3.

2.2 Several Missing Realizations

From the preceding discussion it should be clear that nothing changes if the missing realizations are separated by more than N known realizations. Then the Markov blankets of the missing variable are still completely known. If this is not the case we obtain Equation 5 where $y^u \subseteq \{y_{t-1}, y_{t-2}, \ldots, y_{t-N}\}$ denote all missing instances between $t-1$ and $t-N$ of the time series and where $y^m \subseteq \{y_{t-1}, y_{t-2}, \ldots, y_1\}$ denote the set of all measurement up to $t-1$. Also $P(y^u | y^m) \propto P(y_{t-1}, \ldots, y_2, y_1)$ where the right-hand side is obtained from Equation 4.

2.3 Training with Missing Realizations

We consider the case that $y_1, \ldots, y_t$ are possible realizations. Let $y^m \subseteq \{y_1, \ldots, y_t\}$ denote the set of all measurements and $y^u = \{y_1, \ldots, y_t\} \setminus y^m$ the set of all unknowns. Our model is assumed to be a neural network parameterized by a set of weights w

$$f(y_{t-1}, \ldots, y_{t-N}) \approx NN_w(y_{t-1}, \ldots, y_{t-N})$$

or any other kind of parameterized function approximator. The log-likelihood function of the time series is $L = \log \int P^M(y_t, y_{t-1}, \ldots, y_2, y_1)\, dy^u$. Here

$$P^M(y_t, y_{t-1}, \ldots, y_2, y_1) = P^M(y_N, \ldots, y_1) \prod_{l=N+1}^{t} P^M(y_l|y_{l-1}, \ldots, y_{l-N}). \tag{6}$$

is an approximation to the joint density and

$$P^M(y_t|y_{t-1}, y_{t-2}, \ldots, y_{t-N}) = P_\epsilon(y_t - NN_w(y_{t-1}, y_{t-2}, \ldots, y_{t-N})). \tag{7}$$

For backpropagation learning or other gradient based learning algorithms we need the gradient of the log-likelihood with respect to the weights which is[1]

$$\frac{\partial L}{\partial w} = \sum_{l=N+1}^{t} \int \frac{\partial \log P^M(y_l|y_{l-1}, \ldots, y_{l-N})}{\partial w} P^M(y^{u(l)}|y^m)\, dy^{u(l)}. \tag{8}$$

In case of Gaussian noise,

$$\frac{\partial L}{\partial w} \propto$$

$$\sum_{l=N+1}^{t} \int (y_l - NN_w(y_{l-1}, \ldots, y_{l-N})) \frac{\partial NN_w(y_{l-1}, \ldots, y_{l-N})}{\partial w} P^M(y^{u(l)}|y^m)\, dy^{u(l)}.$$

Here $y^{u(l)} = y^u \cap \{y_l, \ldots y_{l-N}\}$ are the missing realizations in the input of the network. The last equation shows that if all $y_l \ldots y_{l-N}$ are known, the integral "disappears".

3 Prediction and Training with Noisy Measurements

Let again $y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-N}) + \epsilon_t$ but now we assume that we have no access to y_t directly. Instead, we measure $z_t = y_t + \delta_t$ where δ_t is independent zero-mean noise. Let $z = \{z_1 \ldots z_{t-1}\}$ and $y = \{y_1 \ldots y_t\}$. The joint probability density is

$$P(y, z) = P(y_N, \ldots, y_1) \prod_{l=N+1}^{t} P(y_l|y_{l-1}, \ldots, y_{l-N}) \prod_{l=1}^{t} P(z_l|y_l).$$

[1] Assuming known initial conditions for $y_1, \ldots, y_N$. In this paper, we use repeatedly that if $f(x) > 0$, then $\frac{\partial f(x)}{\partial x} = \frac{\partial \log f(x)}{\partial x} f(x)$.

The expression for the expected value of the next instance of the time series (prediction) is

$$E(y_t|z) = \int f(y_{t-1},\ldots,y_{t-N})\; P(y_{t-1},\ldots,y_{t-N}|z)\; dy_{t-1}\ldots dy_{t-N}. \tag{9}$$

Similarly the gradient of the likelihood for training can be calculated. For the special case of Gaussian noise, with $z = \{z_1 \ldots z_t\}$

$$\frac{\partial L}{\partial w} \propto$$

$$\sum_{l=N+1}^{t} \int (y_l - NN_w(y_{l-1},\ldots,y_{l-N}))\; \frac{\partial NN_w(y_{l-1},\ldots,y_{l-N})}{\partial w} P^M(y_l,\ldots,y_{l-N}|z)\; dy_l \ldots dy_{t-N}.$$

4 Approximations

4.1 Approximations of the Solution

In general, if $f()$ is a nonlinear function the equations we obtained for prediction and for calculating the gradient cannot be solved analytically and must be approximated numerically. We will discuss a solution based on Monte Carlo sampling. Note that all solutions have the general form $\int h(u,m)P(u|m)du$ where u is the set of unknown variables and m is the set of known variables. An integral of this form can be solved by drawing random samples of the unknown variables following $P(u|m)$. Let $u^1,\ldots,u^S$ denote these samples. Then we can approximate

$$\int h(u,m)P(u|m)du \approx \frac{1}{S}\sum_{s=1}^{S} h(u^s,m).$$

The problem now reduces to sampling from $P(u|m)$. Let's first assume that only one variable is missing. Then the problem reduces to sampling from a one-variate distribution which can be done using *sampling-importance-resampling* or other sampling techniques [1].

If more than one realization is missing the situation becomes more complicated. The reason is that the unknown variables are in general dependent and we have to draw from the distribution of all unknowns. A general solution is Gibbs sampling. In Gibbs sampling we initialize the unknown variables either randomly or better with reasonable initial values. Then we select one of the unknown variables u_i and pick a sample from $P(u_i|m, u \setminus u_i)$ and set u_i to that value. Then we repeat the procedure for the next unknown variables and so on. Discard the first samples. Then samples are produced with the correct distribution. This of course means that we might have to sample all unknowns which ever occurred in the time series. In practice, one would restrict the sampling to some reasonable chosen time window. Note, that in the missing data case, if N consecutive values are known the coupling is broken and we do not need to consider missing values which lie further away. Also sampling is simple if only samples of *future* values are required as in K-step prediction (Section 5) and in control problems (Section 6.3). The reason is that we can sample forward in time. Note, that sampling does not work for deterministic systems. Finally, we want to point out the simplicity behind the complicated looking solutions. Both for prediction and training we draw samples of the unknown variables according to their probability density. In prediction we substitute those samples for the missing data and average the predictions. In training calculate the average of the error gradients using the substituted samples.

4.2 Maximum-Likelihood Substitution

Here, we do not sample but substitute the most likely values for u, that is

$$\int h(u,m)P(u|m)du \approx h(u^{ml}, m)$$

where $u^{ml} = \max_u \log P(u|m)$. For the prediction model in Equation 5 where y_{k-1} is missing and assuming Gaussians

$$y_{t-k}^{ml} = \min_{y_{t-k}} \sum_{l=t-k}^{t-1} (y_l - f(y_{l-1}, \ldots, y_{l-N}))^2$$

we simply find the substitution which minimizes the sum of the squared errors. In case of noisy measurements and Gaussian distributions

$$y^{ml} = \min_{y_1,\ldots,y_{t-1}} [-\log P(y_1,\ldots,y_N) + \frac{1}{2\sigma_\epsilon^2} \sum_{l=N+1}^{t-1} (y_l - f(y_{l-1},\ldots,y_{l-N}))^2 + \frac{1}{2\sigma_\delta^2} \sum_{l=1}^{t-1} (y_l - z_l)^2]$$

Where σ_ϵ^2 and σ_δ^2 are the variances of the two noise sources. Note, that this is a multidimensional optimization problem. In training, both the weights and the estimates of the missing or noisy realizations are adapted concurrently.

5 Experiments: K-step Prediction

We can use our preceding equations to predict K-steps into the future. In our framework, this is equivalent to the problem that $y_{t-1}, \ldots, y_{t-K+1}$ are missing and we want to predict y_t. In this case, Monte-Carlo sampling is very simple: generate a sample y_{t-K+1}^s of the first missing value using the distribution $P(y_{t-K+1}|y_{t-K}, \ldots, y_{t-K-N})$. Using that sample and the previous measurements, generate a sample of y_{t-K+2} following $P(y_{t-K+2}|y_{t-K+1}^s, \ldots, y_{t-K+1-N}))$ and so on until a sample of each unknown is produced. Repeat this procedure S times and approximate

$$E(y_t|M_{t-1}) \approx \frac{1}{S} \sum_{s=1}^{S} f(y_{t-1}^s, y_{t-1}^s, \ldots, y_{t-N}^s)$$

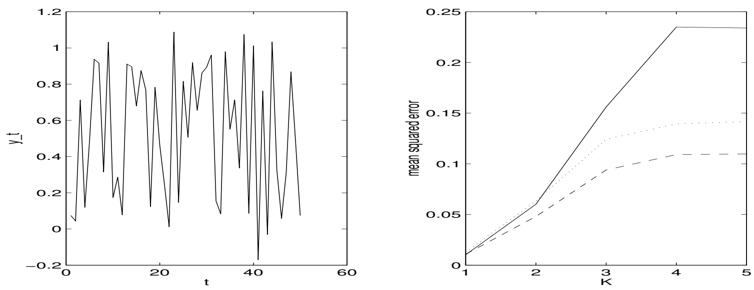

Figure 2: Left: Samples of the logistic map. Right: The mean squared error as a function of K in $K-$step prediction. The iterated solution (continuous) and the Monte-Carlo approximation with 3 (dotted) and 20 samples (dashed) are shown. Only for one-step prediction, the iterated model is optimal. Note, that by sampling we obtained an estimate of the prediction error of the iterated system (assuming a correct model).

where we have assumed that $K > N$. If $K \leq N$ substitute measured values for $k \geq K$. Note, that simply iterating the model K-times as it is usually done in $K-$step prediction is suboptimal in nonlinear time-series prediction if $K > 1$!

In our experiments, we wanted to find out to which degree our solutions are superior to simply iterating the time series in K-step prediction. We used the noisy logistic map $y_t = 4z_{t-1}(1 - z_{t-1}) + \epsilon_t$ where

$$z_t = \begin{cases} y_t & \text{if } 0 \leq y_t < 1 \\ y_t - 1 & \text{if } y_t \geq 1 \\ y_t + 1 & \text{if } y_t < 0 \end{cases}$$

where ϵ_t is uncorrelated Gaussian noise with a variance of $\sigma^2 = 0.01$. Figure 2 (left) shows the time series. Figure 2 (right) shows the mean squared error as a function of K. Shown are the iterated system (continuous line) and the solution following our sampling approach. As expected, for $K = 1$ the interated solution is optimal, but for $K > 1$, the Monte-Carlo approximation even with only few samples is far superior.

6 Extensions

6.1 Error Bars

Sampling provides much more information than just expected values. In all of the cases considered earlier — missing or noisy data, K-step prediction — we can also easily obtain error bars of the predicted value by calculating the variance (or the covariances) in the samples produced (Figure 2).

6.2 Recurrent Neural Networks for Stochastic Models

The approach can be applied to state space models of the form[2]

$y_t = f(y_{t-1}) + \epsilon_t \quad z_t = y_t + \delta_t$

where $y_t, \epsilon_t \in \Re^{D_y}$, $z_t, \delta_t \in \Re^{D_z}$. In the following discussion and for the rest of the paper we will assume independent Gaussian noise for all noise sources (here: for all components of ϵ_t and δ_t), although the results can be easily generalized to other distributions. We also assume that the components of ϵ_t have identical variances. Infinite variance in the components of δ_t indicates an unknown (hidden) variable, a variance of zero a certain measurement[3] and a finite variance a noisy measurement. $z = \{z_1, \dots z_{t-1}\}$ is the set of measured values and $y = \{y_1, \dots y_{t-1}\}$ are unknown. Let's assume that $f(y_{t-1}) \approx NN_w(y_{t-1})$ is approximated by a neural network. The gradient of the log-likelihood with respect to a weight w is

$$\frac{\partial L}{\partial w} \propto \sum_{l=2}^{t} \int \frac{\partial NN_w(y_{l-1})}{\partial w} \left(y_l - NN_w(y_{l-1})\right) P^M(y_l, y_{l-1}|z)\; dy_l dy_{t-1}. \tag{10}$$

Note, that $\frac{\partial NN_w(y_{l-1})}{\partial w}$ is a $D_w \times D_y$ dimensional matrix, where D_w are the number of weights in the network. In the Monte Carlo approximation, the main problem is the sampling from $P^M(y_l, y_{l-1}|z)$ which can be very time-consuming. We will discuss another learning algorithm for recurrent networks in the next section where the generation of samples is much easier.

6.3 Control of Stochastic Systems, Reinforcement Learning and Training of Recurrent Networks Revisited

Deterministic Control. Consider $y_t = f(y_{t-1}, u_{t-1}) + \epsilon_t$ where $u_t = NN_w(y_t)$ is a parameterized controller. The task is to minimize the expected cost up to a finite horizon T

$$E(cost) = \int \sum_{l=1}^{T} \gamma^{l-1} C(y_l) P(y_1, \dots, y_T) dy_1 \dots dy_T$$

where $\gamma \leq 1$ is a discount factor, $C(y_l)$ is the cost of being in state y_l and $P(y_1, \dots, y_T) = P(y_1) \prod_{l=2}^{T} p(y_l|y_{l-1})$. To optimize the controller, we need the gradient of the expected cost with respect to w. We obtain[4]

$$\frac{\partial E(cost)}{\partial w} \propto \sum_{l=2}^{T} \int \gamma^{l-1} C(y_l) \tag{11}$$

[2] Introducing the more general assumption that $z_t = g(y_t) + \delta_t$ poses no additional difficulties.

[3] For the sampling procedure to work, a small error should always be assumed, such that the noise variance is always greater than zero.

[4] Recall that we assume Gaussian noise distributions.

$$\times[\sum_{m=2}^{l} \frac{\partial f(y_{m-1}, u_{m-1}))}{\partial u_{m-1}} \frac{\partial NN_w(y_{m-1})}{\partial w}(y_m - f(y_{m-1}, u_{m-1}))] P(y_1, \ldots y_l)\, dy_1 \ldots dy_l$$

This solution can be approximated using stochastic sampling (see the following discussion). To avoid infinite control actions, it might be useful to introduce a cost which takes control actions into account or which adds a penalty for large weights in NN_w.

Stochastic Control. Now assume that the control action is stochastic $u_t = NN_w(y_t) + \delta_t$ and that we allow that the cost depends on the control action. Then,

$$\frac{\partial E(cost)}{\partial w} \propto \sum_{l=1}^{T} \int \gamma^{l-1} C(y_l, u_l)[\sum_{m=1}^{l} \frac{\partial NN_w(y_m)}{\partial w}(u_m - NN_w(y_m))] \tag{12}$$

$$\times P(y_1, \ldots, y_l, u_1, \ldots, u_{l-1})\, dy_1 \ldots dy_l du_1 \ldots du_{l-1}.$$

Note, that we do not need a model of the process $f()$ any more! This is a result of the fact that we execute *stochastic* control. The system "tries" different actions and adapts the controller to favor actions which lead to low costs. We simply simulate the system (or collect data on the real process) and execute control actions. In the course of training we might want to reduce the noise variance on the control to eventually converge to deterministic controls. Let's assume that we generated S time series of the process by starting at $l = 1$ and iterating until T generating samples u_l^s and y_l^s. For each experiment s, we iterate for $l = 2, \ldots, T$ ($a_0^s = e_0^s = 0$)

$$a_l^s = \gamma a_{l-1}^s + \gamma^{l-1} \frac{\partial NN_w(y_l^s)}{\partial w}(u_l^s - NN_w(y_l^s))$$

and

$$e_l^s = e_{l-1}^s + C(y_l^s, u_l^s) a_{l-1}^s.$$

Then

$$\partial E(cost)/\partial w \approx 1/S \sum_{s=1}^{S} e_T^s.$$

Recurrent Neural Networks. The previous equations also contain an algorithm for training recurrent neural networks. Assume $y_t = NN_w(y_{t-1}) + \epsilon_t$. Define $C(y_t) = b_t' \operatorname{diag}[(y_t^d - y_t)(y_t^d - y_t)']$, $\gamma = 1$. Here, y_t^d is a target at time t and b_t is a vector with $b_{ti} = 1$ if the i-th component of y_t is measured and zero otherwise (i. e. for the hidden variables). The operator diag forms a vector of the diagonal elements of a matrix. Then

$$a_l^s = a_{l-1}^s + \frac{\partial NN_w(y_{l-1}^s)}{\partial w}(y_l^s - NN_w(y_{l-1}^s))$$

and

$$e_l^s = e_{l-1}^s + C(y_t)a_l^s.$$

Finally,

$$\partial E(cost)/\partial w \approx \frac{1}{S}\sum_{s=1}^{S} e_T^s.$$

On-line Adaptation. Consider stochastic control again. We let $T \to \infty$. We now assume that at every time-step, we start a new experiment s. Then let $a_l = \sum_{s=1}^{l} a_l^s$ and

$$a_l = \gamma a_{l-1} + [\sum_{m=1}^{l} \gamma^{l-m}] \frac{\partial NN_w(y_l^s)}{\partial w}(u_l^s - NN_w(y_l^s))$$

$$\approx \gamma a_l + \frac{1}{1-\gamma}\frac{\partial NN_w(y_l^s)}{\partial w}(u_l^s - NN_w(y_l^s)).$$

The last approximation is valid for large l. Then:

$$e_l = \rho e_{l-1} + C(y_l^s)a_{l-1}$$

$(\rho = 1)$. In practice, a small gradient descent learning step might be executed at every time-step $\Delta w(l) \propto e_l$ with an additional decay term on e_l $(\rho < 1)$. Note that this is easily recognized as a variant of reinforcement learning. We can reduce the variance of the controller in the course of training and converge to a deterministic control law. The connection with reinforcement learning is even more obvious if we write for the right-hand side of Equation 12 (exchange the order of summations, $T \to \infty$)

$$\sum_{m=1}^{\infty} \int \frac{\partial NN_w(y_m)}{\partial w}(u_m - NN_w(y_m))\gamma^{m-1}(C(y_m, u_m) + \gamma V(y_{m+1}))$$

$$\times P(y_m, y_{m+1}, u_m)\ dy_m dy_{m+1} du_m.$$

The expression

$$V(y_{m+1}) = \int C(y_{m+1}, u_{m+1})du_{m+1} + \sum_{l=m+2}^{\infty} \gamma^{l-m-1} \int C(y_l, u_l)P(y_l, u_l)dy_l du_l$$

is the expected cost if the current state is y_{m+1} (not including the factor γ^m).

7 Conclusions

We have shown how the problem of missing snd noisy data can be approached in a principled way in time-series prediction. In addition, we derived equations for training recurrent neural networks, for stochastic control and for reinforcement learning problems. The proposed approximations are based on stochastic simulations which, in general, are computationally expensive. Sampling is particularly simple if we can sample only forward in time as in K-step prediction and in the control laws and learning rules discussed in Section 6.3.

References

[1] Bernardo, J. M., Smith, A. F. M. (1994) *Bayesian Theory.* Wiley & Sons.

[2] Buntine, W. L. and Weigend, A. S. (1991). Bayesian Back-Propagation. *Complex systems,* Vol. 5, pp. 605-643.

[3] Ghahramani, Z. and Jordan, M. I. (1994). Supervised Learning from Incomplete Data via an EM approach. In: Cowan, J. D. *et al.*, eds., *Advances in Neural Information Processing Systems 6,* Morgan Kaufman.

[4] Tresp, V., Ahmad, S. and Neuneier, R. (1994). Training Neural Networks with Deficient Data. In: Cowan, J. D. *et al.*, eds., *Advances in Neural Information Processing Systems 6,* Morgan Kaufman.