Learning in presence of input noise using the stochastic EM algorithm

Hichem Snoussi*, Abd-Krim Seghouane†, Ali Mohammad-Djafari* and Gilles Fleury†

*Laboratoire des Signaux et Systèmes (L2S),
Supélec, Plateau de Moulon, 91192 Gif-sur-Yvette Cedex, France
†Service des Mesures,
Supélec, Plateau de Moulon, 91192 Gif-sur-Yvette Cedex, France

Abstract. Most learning algorithms rely on the assumption that the input training data contains no noise or uncertainty. However, when collecting data under an identification experiment it may not be possible to avoid noise when measuring the input. The use of the errors-in-variable model to describe the data in this case is more appropriate. However, learning based on maximum likelihood estimation is far from straightforward because of the high number of unknown parameters. In this paper, to overcome the problems associated to the estimation with high number of unknown parameters, the nonlinear errors-in-variable estimation problem is treated under a Bayesian formulation. In order to compute the necessary maximum *a posteriori* estimate we use the restoration maximization algorithms where the true but unknown training inputs are treated as hidden variables. In order to accelerate the convergence of the algorithm a modified version of the stochastic EM algorithm is proposed. A simulation example on learning a nonlinear parametric function and an example on learning feedforward neural networks are presented to illustrate the effectiveness of the proposed learning method.

INTRODUCTION

Most learning algorithms take into account only the uncertainty in the output when treating the training data set. Therefore, they rely on the assumption that the input is known exactly. Generally, the description of the training data set is made by the additive-error regression model

$$y_t = f(x_t, \boldsymbol{\theta}) + \epsilon_t, \qquad t = 1, ..., T \tag{1}$$

where ϵ_t is sampled from a centered Gaussian law of variance σ_ϵ^2, $(x_t, y_t)_{t=1,....,T}$ are the experimental input-output training pairs, $f(., \boldsymbol{\theta})$ can represents a neural network approximator where $\boldsymbol{\theta}$ is the neural network parameter vector. In the case of feedforward neural networks with L neurons in the hidden layer

$$f(x, \boldsymbol{\theta}) = \alpha_0 + \sum_{l=1}^{L} \alpha_l \psi(\beta_l x + \gamma_l), \tag{2}$$

where $\psi(.)$ is generally the sigmoidal function although other non decreasing functions converging to 0 as $x \longrightarrow -\infty$ and to 1 as $x \longrightarrow +\infty$, can also be used [1]. The set of pa-

rameters that specify the neural network is stored in $\boldsymbol{\theta} = (\alpha_0, ..., \alpha_L, \beta_1, ..., \beta_L, \gamma_1, ..., \gamma_L)$, where $\alpha_l \in \Re$, $\beta_l \in \Re$ and $\gamma_l \in \Re$. In this paper, we shall restrict attention to single-input single-output examples, but the approach that we propose can easily be extended to multi-input multi-output cases.

The assumption of knowing exactly the training inputs is clearly an unsafe assumption because any data produced experimentally will have some degree of uncertainty (due to the measurement errors). In the case of neglected input uncertainty or high level input to noise ratio, the use of the previous model to describe the training data can be tolerated. In the inverse case, it is a necessary requirement for the learning algorithm to take into account this input noise or uncertainty otherwise a biased approximation will be produced [2]. In this case, an appropriate description of the training data set can made using the errors-in-variables model

$$\begin{cases} y_t = f(x_t^*, \boldsymbol{\theta}) + \epsilon_t, \quad t = 1, ..., T, \\ x_t = x_t^* + \eta_t, \end{cases} \tag{3}$$

where x_t^* represents the unobservable input which is related to the observed (measured) one via a simple addition with a noise η_t sampled from a centered Gaussian law of variance σ_η^2. The second equation of the model (3) describes the error in the input variables that are termed nuisance parameters [2] while the first equation describes the regression model.

The purpose of this paper is to use a Bayesian formulation of the errors-in-variables model to construct a learning algorithm which is able to cope with input uncertainty. For convergence requirements [3][4] it will be assumed for each $t = 1, ..., T$, that the training pairs (x_t, y_t) results in a mean over $j = 1, ..., r$ training pairs (x_{tj}, y_{tj}) obtained by repeating the experiment 't', 'j' times. In practice, the limiting result corresponding to $n = T \times r \longrightarrow +\infty$ with both T and r increasing can be used as an approximation when the error variance are small and the number of data points is large.
The problem that is addressed in this paper is the one of estimating parameters when the input training data are corrupted by noise. This suggest the availability of a correct model which can be obtained using a selection model procedure [5].

The paper is organized as follows: in the next section the maximum likelihood approach is developed and its difficulties are illustrated. Some approximate approaches are also briefly reviewed. In the aim of reducing the dimension of the problem a Bayesian approach is adopted in section 3, where the necessary *a posteriori* estimate is computed using a modified version of the EM algorithm while treating the unknown inputs as hidden variables. In order to illustrate the performance of the proposed learning method, two examples are given in section 4. Finally, in section 5 we provide a conclusion.

THE MAXIMUM LIKELIHOOD APPROACH

In the approach developed in this section both types of parameters are estimated simultaneously, no distinction is made between the vector of parameters of interest $\boldsymbol{\theta}$ and the vector of nuisance parameters (the unknown true inputs) $(x_1^*, ..., x_T^*) = \mathbf{x}^*$. We consider the composite $T + p$ (where $p = dim(\boldsymbol{\theta})$) dimensional parameter vector

$$(\mathbf{x}^*, \boldsymbol{\theta}) \in \Gamma \times \Theta \subset \Re^{T+p}.$$

The density of the training data pairs $(x_t, y_t)_{t=1,\ldots,T}$ generated by the model (3) is

$$\prod_{t=1}^{T} P_\epsilon(y_t - f(x_t^*, \boldsymbol{\theta})) P_\eta(x_t - x_t^*) \tag{4}$$

where P_ϵ and P_η represent the densities of the output ant the input noises which are Gaussian centered of variance σ_ϵ^2 and σ_η^2 respectively. The negative log-likelihood function is proportional to

$$L(\mathbf{x}^*, \boldsymbol{\theta}) = \sum_{t=1}^{T} \left(\frac{y_t - f(x_t^*, \boldsymbol{\theta})}{\sigma_\epsilon} \right)^2 + \left(\frac{x_t - x_t^*}{\sigma_\eta} \right)^2, \tag{5}$$

and the maximum likelihood estimators $\hat{\boldsymbol{\theta}}$ and $\hat{\mathbf{x}}^*$ are the value of $\boldsymbol{\theta} \in \Theta$ and $\mathbf{x}^* \in \Gamma$ that minimize $L(\mathbf{x}^*, \theta)$, therefore,

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \Theta} \min_{\mathbf{x}^* \in \Gamma} L(\mathbf{x}^*, \theta). \tag{6}$$

The maximum likelihood estimation causes a great deal of difficulty because of the large number of parameters which increases with the size of the training data set and goes to infinity as the size of the training data set approaches infinity. We are interested in eliminating the nuisance parameters in the estimating function so that it depends only on the vector of parameters of interest $\boldsymbol{\theta}$.

A local linear approximation

An iterative algorithm based on a local linear approximation was proposed for the optimization of the likelihood function (5) in [6]. This algorithm is based on the fact that the variables $\mathbf{x}^*$ and $\boldsymbol{\theta}$ can be separated. Therefore, the joint optimization on $(\mathbf{x}^*, \boldsymbol{\theta})$ can be made alternatively on $\boldsymbol{\theta}$ and $\mathbf{x}^*$. To this end, let $\bar{\mathbf{x}}^*$ denote a preliminary estimator of $\mathbf{x}^*$. For this, consider the measured vector $\mathbf{x}$. The corresponding preliminary estimate of $\boldsymbol{\theta}$ is therefore,

$$\bar{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \Theta} L(\bar{x}^*, \theta) = \arg \min_{\boldsymbol{\theta} \in \Theta} \sum_{t=1}^{T} \left(\frac{y_t - f(\bar{x}_t^*, \boldsymbol{\theta})}{\sigma_\epsilon} \right)^2. \tag{7}$$

The problem of the high optimization dimension on $\mathbf{x}^*$ was avoided thanks to the local linear approximation which transforms the optimization problem a system of linear equations. By expanding $f(x_t^*, \bar{\boldsymbol{\theta}})$ in a Taylor series about the point $\bar{x}^*$ and retaining only the linear term, we obtain

$$f(x_t^*, \bar{\boldsymbol{\theta}}) = f(\bar{x}_t^*, \bar{\boldsymbol{\theta}}) + f_x(\bar{x}_t^*, \bar{\boldsymbol{\theta}}) \Delta x_t^*, \tag{8}$$

where $\Delta x_t^* = x_t^* - \bar{x}_t^*$ and

$$f_x(\bar{x}_t^*, \bar{\boldsymbol{\theta}}) = \left.\frac{\partial f(x, \bar{\boldsymbol{\theta}})}{\partial x}\right|_{\bar{x}_t^*}.$$

Letting

- $a_t = y_t - f(\bar{x}_t^*, \bar{\boldsymbol{\theta}})$,
- $b_i = x_t^* - \bar{x}_t^*$ and
- $\Delta y_t = f(x_t^*, \bar{\boldsymbol{\theta}}) - f(\bar{x}_t^*, \bar{\boldsymbol{\theta}}) = f_x(\bar{x}_t^*, \bar{\boldsymbol{\theta}})\Delta x_t^*$,

the local approximation of $L(x^*, \bar{\boldsymbol{\theta}})$ is

$$\begin{aligned} L(x_t^*, \bar{\boldsymbol{\theta}}) &= \sum_{t=1}^{T} l(x_t^*, \bar{\boldsymbol{\theta}}) \\ &\simeq \sum_{t=1}^{T} (a_t - \Delta y_t, b_t - \Delta x_t^*)\Sigma^{-1}(a_t - \Delta y_t, b_t - \Delta x_t^*)^t \\ &= (a - F\Delta X)\Sigma_\varepsilon^{-1}(a - F\Delta X)^t + (b - \Delta X)\Sigma_\eta^{-1}(b - \Delta X)^t, \end{aligned} \tag{9}$$

where a and b are vectors of dimension T, F is a diagonal matrix of dimension $T \times T$ where the diagonal elements are the derivatives $f_x(\bar{x}_t^*, \bar{\boldsymbol{\theta}})$, $t = 1..., T$, $\Sigma_\varepsilon^{-1} = \sigma_\varepsilon^{-1} I_T$ and $\Sigma_\eta^{-1} = \sigma_\eta^{-1} I_T$.

The vector $\Delta\hat{x^*} = [\Delta\hat{x}_1^*, ..., \Delta\hat{x}_T^*]^t$ that minimizes this sum

$$\Delta\hat{x} = (F^t\Sigma_\varepsilon^{-1}F + \Sigma_\eta^{-1})^{-1}(F^t\Sigma_\varepsilon^{-1}a + \Sigma_\eta^{-1}b), \tag{10}$$

allows the update of the estimation of $\mathbf{x}^*$

$$\hat{x^*} = \bar{x^*} + \Delta\hat{x^*}, \tag{11}$$

and subsequently of $\boldsymbol{\theta}$ by the use of equation (7). The statistical properties of the estimator of $\boldsymbol{\theta}$ obtained by this algorithm has been derived in [6]. The major inconvenience of this algorithm is that the solution depends on the stability of the vector $\Delta\hat{x^*}$, which is not guaranteed when the vector x^* is of high dimension. Therefore methods which allow the elimination of the nuisance parameters from the likelihood function may be preferable.

A standard method of eliminating the nuisance parameters is to adopt a Bayesian approach [7]. This is made by multiplying the density (4) by the appropriate prior distribution to obtain the joint *a posteriori* distribution for the nuisance parameters and the vector of parameters of interest. Then the marginal distribution of it gives an estimating criterion independent of the nuisance parameters. The Bayesian estimators are particularly interesting because they are asymptotically effi cient and asymptotically equivalent to the maximum likelihood estimator (under regularity conditions) independent of the imposed prior [8]. Based on this, an estimator and the associated algorithm is proposed in the following section.

THE BAYESIAN APPROACH

Given the training data set $D = (x_t, y_t)_{t=1,....,T}$, the *a posteriori* distribution of the parameters vectors $\mathbf{x}^*$ and $\boldsymbol{\theta}$ is, according to the Bayesian rule,

$$p(\mathbf{x}^*, \boldsymbol{\theta}|\ D) \propto p(D|\ \mathbf{x}^*, \boldsymbol{\theta}) p(\mathbf{x}^*, \boldsymbol{\theta}) \tag{12}$$

where $p(D|\ \mathbf{x}^*, \boldsymbol{\theta})$ is the density of the training data set $D = (x_t, y_t)_{t=1,....,T}$ from which the likelihood function is constructed and $p(\mathbf{x}^*, \boldsymbol{\theta})$ is the *a priori* distribution of the unknown parameters. The choice of a non informative *a priori* distribution is not an easy task [9] and in the following we retain the flat prior which is not proper when unbounded but we suppose that the function f under hand guarantees the existence of the *a posteriori* distribution, i.e $\int p(D|\ \mathbf{x}^*, \boldsymbol{\theta}) p(\mathbf{x}^*, \boldsymbol{\theta}) d\mathbf{x}^* d\boldsymbol{\theta} < \infty$. We note that the *a posteriori* distribution carries nicely all our knowledge about our inferential problem and is suffi ciently flexible to incorporate any additional *a priori* information concerning the unknown parameters. As we have mentioned in the previous section, the joint estimation of the parameters of interest $\boldsymbol{\theta}$ and the nuisance parameters $\mathbf{x}^*$ is, if not intractable, computationally consuming. Moreover, the joint estimation of the unknown inputs $\mathbf{x}^*$ may introduce a bias to the resulting estimate of the parameter of interest $\boldsymbol{\theta}$ [10]. The Bayesian formulation of the problem makes the integration over the undesirable inputs possible which yields the marginal *a posteriori* distribution of the parameter $\boldsymbol{\theta}$:

$$\begin{aligned} p(\boldsymbol{\theta}|\ D) &= \int p(\mathbf{x}^*, \boldsymbol{\theta}|\ x_{1,..,T}, y_{1,..,T})\, d\mathbf{x}^* \\ &\propto p(\boldsymbol{\theta}) \int p(y_{1,..,T}\ |\ \boldsymbol{\theta}, \mathbf{x}^*) p(x_{1,..,T}|\ \mathbf{x}^*) d\mathbf{x}^* \end{aligned} \tag{13}$$

Now, our purpose is the estimation of the parameter $\boldsymbol{\theta}$ by maximizing the *a posteriori* distribution (8):

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \Theta} p(\boldsymbol{\theta}|\ D) \tag{14}$$

In most cases, the integration in (13) and maximization in (14) are not feasible and an explicit solution $\hat{\boldsymbol{\theta}}$ is unreachable. However, given the true inputs $\mathbf{x}^*$, the problem turns to be a classic supervised learning procedure. This suggests that we artifi cially complete the training data set $D = (x_t, y_t)_{t=1,....,T}$ into $(x_t, y_t, x_t^*)_{t=1,....,T}$ where we consider the unknown inputs $\mathbf{x}^*$ as hidden variables and consequently use restoration maximization algorithms like EM algorithm [17] which is an iterative algorithm consisting in two steps:

- E-step: Compute the functional:

$$\varrho(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k-1)}) = E\{\log p(\boldsymbol{\theta}, x^*|\ D)|\ D, \boldsymbol{\theta}^{(k-1)}\}$$

- M-step: Update the parameter $\boldsymbol{\theta}$ by maximizing the functional ϱ:

$$\boldsymbol{\theta}^{(k)} = \arg\max_{\boldsymbol{\theta} \in \Theta} \varrho(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k-1)})$$

The input and output noises are white leading to a point wise computation of expectations:

$$\varrho(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k-1)}) = -\frac{1}{2}\sum_{t=1}^{T} E\left[\left(\frac{y_t - f(x_t^*, \boldsymbol{\theta})}{\sigma_\epsilon}\right)^2\right] + \log p(\boldsymbol{\theta}) + K \tag{15}$$

where K is a constant independent of $\boldsymbol{\theta}$.

The existence of the nonlinear function f makes the E-step difficult which leads to the use of the SEM algorithm which is a stochastic version of the EM. The first step is replaced by a sampling from the *a posteriori* distribution of $\mathbf{x}^*$:

- S-step: generate $\tilde{\mathbf{x}}^*$ according to its posterior distribution:

$$\tilde{x}_t^* \sim p(x_t^* \mid x_t, y_t, \boldsymbol{\theta}^{(k-1)})$$

- M-step: the classic supervised learning of f knowing the inputs $\tilde{\mathbf{x}}^*$ and the outputs $\mathbf{y}$:

$$\boldsymbol{\theta}^{(k)} = \arg\min_{\boldsymbol{\theta}\in\Theta} \frac{1}{\sigma_\epsilon^2}\sum_{t=1}^{T}(y_t - f(\tilde{x}_t^*, \boldsymbol{\theta}))^2 - \log p(\boldsymbol{\theta})$$

The stochastic EM algorithm can be generalized [14] by drawing m samples $\tilde{x}_{tj}^*$, $j = 1, .., m$ at each time t and then, in the M-step,

$$\boldsymbol{\theta}_m^{(k)} = \arg\min_{\boldsymbol{\theta}\in\Theta} \frac{1}{\sigma_\epsilon^2}\sum_{t=1}^{T}\frac{1}{m}\sum_{j=1}^{m}(y_t - f(\tilde{x}_{tj}^*, \boldsymbol{\theta}))^2 - \log p(\boldsymbol{\theta}) \tag{16}$$

It appears clearly that when $m \longrightarrow \infty$ and assuming the ergodicity of the $(\tilde{x}_{tj}^*)$ chain [11], this algorithm has the same properties as the exact EM algorithm.

Sampling schemes for true inputs

The direct sampling of $\mathbf{x}^*$ is not an easy task because of the existence of the nonlinearity f. However, non-direct but exact sampling methods already exist such as the Accept-reject procedure or the Markov Chains Monte Carlo methods [16]. We briefly review these two methods to show their drawbacks when applied to our problem and propose a modified version which is efficient, fast and adapted to our general algorithm.

Accept-Reject method

The first step consists in sampling x_t^* from its *a posteriori* distribution $p(x_t^* \mid x_t, y_t, \boldsymbol{\theta})$ which is proportional to $q(x_t^*) = \exp\left(\frac{-1}{2\sigma_\epsilon^2}(y_t - f(x_t^*, \boldsymbol{\theta}))^2\right)\exp\left(\frac{-1}{2\sigma_\eta^2}(x_t - x_t^*)^2\right)$. In general, we are not able to sample from the distribution q. Instead, we can sample from

another distribution g which we call the instrumental distribution. Choosing the instrumental distribution $g(x_t^*) = N(x_t^*; x_t, \sigma_\eta^2)$, we can easily uniformly bound the ratio $q(x_t^*)/g(x_t^*) \leq M = 2\pi\,\sigma_\eta^2$, thus the sampling procedure is:

- 1. Sample $z \sim g(z)$
- 2. Sample $u \sim \mathcal{U}_{[0,1]}$
- 3. If $u \leq \frac{q(z)}{Mg(z)}$, then accept $x_t^* = z$, else reject z and return to 1

The drawback of using this method in our case is the fact that the acceptance probability $M^{-1} \int q(x_t^*) dx_t^*$ depends on the time t. Consequently, besides the randomness of the number of rejections, its law varies with time. The algorithm may be stuck in the first step (S-step) because of only one sample at time t_0 even if the other samples $z_{t \neq t_0}$ are all accepted !

Hasting-Metropolis method

This method consists of sampling from an instrumental distribution g (its choice is optimized to mimic the original distribution q) and then accept the sample or keep the previous one according to an acceptance probability ρ [12]. The algorithm is then,

At iteration k:

- S-step:
  a)- Sample $z \sim g(z)$.
  b)- Accept $x^{*(h)} = z$ with probability $\rho = Min\left(1, \frac{q(z)g(x^{*(h-1)})}{g(z)q(x^{*(h-1)})}\right)$
  else $x^{*(h)} = x^{*(h-1)}$
  c)- return to a) under convergence of the Markov chain: $h = h_{cv}$
  d)- When Markov chain converges put $\tilde{x}^* = x^{*(h_{cv})}$
- M-step:

$$\boldsymbol{\theta}^{(k)} = \arg\min_{\boldsymbol{\theta} \in \Theta} \frac{1}{\sigma_\epsilon^2} \sum_{t=1}^{T} (y_t - f(\tilde{x}_t^*, \boldsymbol{\theta}))^2 - \log p(\boldsymbol{\theta})$$

In the first step, the Markov chain $(x^*)^{(h)}$ has q as a stationary distribution. The convergence of the MCMC methods is a known problem studied in the literature [13] and an efficient tool for convergence diagnostic depends on the problem under hand and there is not a general procedure to decide if the Markov chain has converged or not. Moreover, even if we attain the convergence, then we have many samples $(x^*)^{(h)}$ and it will be more efficient in this case to apply the Monte Carlo EM algorithm and the SEM algorithm is useless in this case.

Modified Stochastic EM algorithm

In order to avoid the convergence problem at each step of the SEM algorithm, we implement only one step of Hasting-Metropolis procedure. In classical SEM algorithms,

the first step consists in computing a sample from the *a posteriori* of the hidden variable x^*. Such sample is obtained in the asymptotic regime of the Markov chain formed by Hasting-Metropolis procedure described above and so we need to repeat this procedure until convergence. We propose to perform only one iteration of Hasting Metropolis algorithm at each iteration based on the hidden variables $(\tilde{x}^*)^{(k-1)}$ sampled in the previous iteration. The algorithm is then,

At iteration k:

- S-step:
  a)- Sample $z \sim\ g(z)$.
  b)- Accept $x^{*(k)} = z$ with probability $\rho = Min\left(1, \frac{q(z)g(x^{*(k-1)})}{g(z)q(x^{*(k-1)})}\right)$
  else $x^{*(k)} = x^{*(k-1)}$
  c)- put $\tilde{x}^* = x^{*(k)}$
- M-step:

$$\boldsymbol{\theta}^{(k)} = \arg\min_{\boldsymbol{\theta}\in\Theta} \frac{1}{\sigma_\epsilon^2} \sum_{t=1}^{T} (y_t - f(\tilde{x}_t^*, \boldsymbol{\theta}))^2$$

Input noise variance estimation

Given the sampled true inputs x^*, the input noise variance can be estimated by maximizing its *a posteriori* distribution $p(\sigma_\eta \,|\, y_{1..T}, x_{1..T}, x^*_{1..T}$ which has the following expression:

$$\begin{aligned} p(\sigma_\eta \,|\, y_{1..T}, x_{1..T}, x^*_{1..T}) &\propto p(x_{1..T} \,|\, x^*_{1..T}, \sigma_\eta) p(\sigma_\eta) \\ &\propto \sigma_\eta^{-T} \exp\left[-\tfrac{1}{2\sigma_\eta^2} \textstyle\sum_{t=1}^{T} (x_t - x_t^*)^2\right] p(\sigma_\eta) \end{aligned}$$

Choosing a Jeffrey prior for σ_η leads to a degeneracy of the above function as the sampled inputs x_t^* will tend to x_t and then the variance goes to zero (see [15] for a detailed study of the degeneracy occurence). Therefore, an inverse Gamma prior is chosen for $p(\sigma_\eta)$

$$p(v = \sigma_\eta^{-2}) = \mathcal{G}(\alpha, \beta)$$

leading to an inverse Gamma *a posteriori*:

$$p(v) = \mathcal{G}(\alpha_{apost}, \beta_{apost})$$

$$\alpha_{apost} = \alpha + T/2$$

$$\beta_{apost} = \beta + \frac{\sum_{t=1}^{T} (x_t - x_t^*)^2}{2}$$

Then, the maximum is attained at $\frac{\alpha_{apost}-1}{\beta_{apost}}$. The α and β parameters are chosen so that the expectation $\alpha/\beta = E[p(\sigma_\eta^{-2})]$ is fixed to an *a priori* noise level and the variance α/β^2 expresses our uncertainty about this noise level.

Choice of the instrumental distribution

The choice of the instrumental distribution g is crucial to obtain fast, efficient and easy to implement algorithms. In the error-in-model special case, a simple and efficient choice is the Gaussian with mean x_t and covariance $\sigma_\eta^2 I$.

$$g(z_t) = N(z_t; x_t, \sigma_\eta^2 I) \tag{17}$$

Thus the acceptance probability ρ is simply:

$$\rho = Min\left(1, \exp\left[-\frac{1}{2\sigma_\epsilon^2}((y_t - f(z))^2 - (y_t - f(x_t^{*(k-1)}))^2)\right]\right) \tag{18}$$

Note that we don't need the knowledge of the function f but only its values $f(z)$ and $f(x_t^{*(k-1)})$ in z and $x_t^{*(k-1)}$. Therefore, the algorithm can be implemented with any learning architecture f providing we can get the values $f(z)$ and $f(x_t^{*(k-1)})$.

SIMULATION EXAMPLE

Example 1: Parametric learning

To illustrate the performance of the proposed algorithm, we consider the following parametric function:

$$y = g(x) = \theta_1 \exp\left[-\theta_2 x\right], \tag{19}$$

where we take the original values $\theta_1 = 2$ and $\theta_2 = 3$. We add white Gaussian noise to both the inputs x_i and outputs y_i:

$$\begin{cases} y_i = g(x_i^*, \boldsymbol{\theta}) + \epsilon_i, & i = 1, ..., 100, \\ x_i = x_i^* + \eta_i, \end{cases} \tag{20}$$

where the standard deviation for the output noise is $\sigma_\epsilon = 0.1$ and for the input noise is $\sigma_\eta = 0.05$. The learning of the parameters (θ_1, θ_2) with maximum likelihood from the noisy inputs (when we suppose that $y_i = g(x_i, \boldsymbol{\theta}) + \epsilon_i$) fails in recovering the original values of the parameters despite the low intensity of the input noise. This shows the sensitivity of the classic learning rule to the input noise. Applying the SEM algorithm to this data produced good results. Figure 1a shows the Markov chain of the first parameter θ_1, note the fluctuations around the original value. Figure 1b shows the convergence of the empirical expectation of θ_1. Figure 2a and Figure 2b show the same results for the parameter θ_2. We note a small biase in the empirical expectations due to the small number of data. In figures 3a and 3b, we plot the evolution of the estimated input variance σ_η and the corresponding empirical expectation of the Markov chain. We note

the convergence around the true value and the success of the algorithm when the input noise variance is unknown.

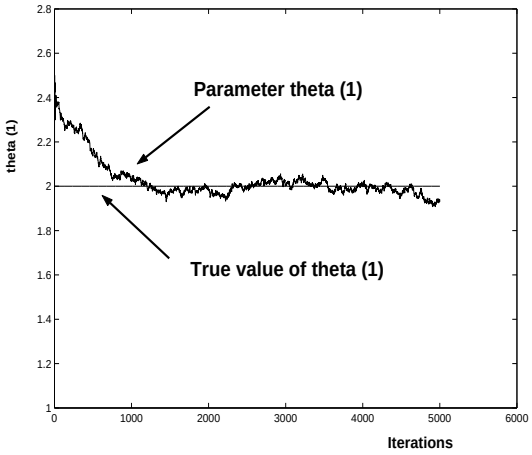

Figure $1.a$ Evolution of the parameter θ_1

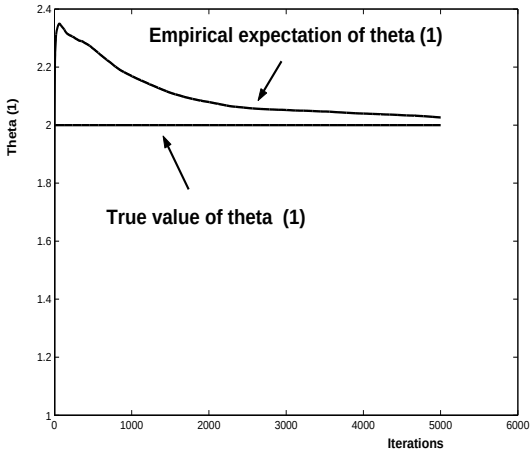

Figure $1.b$ Empirical expectation of θ_1

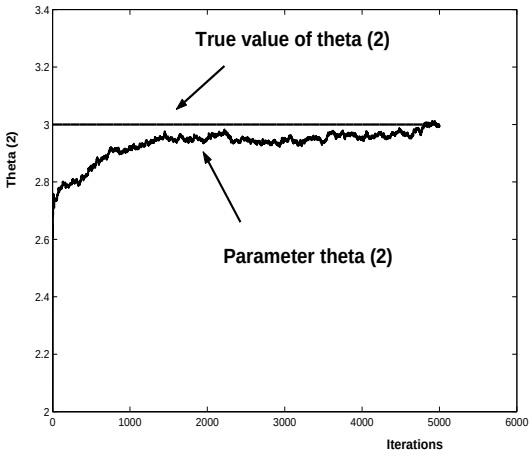

Figure $2.a$ Evolution of the parameter θ_2

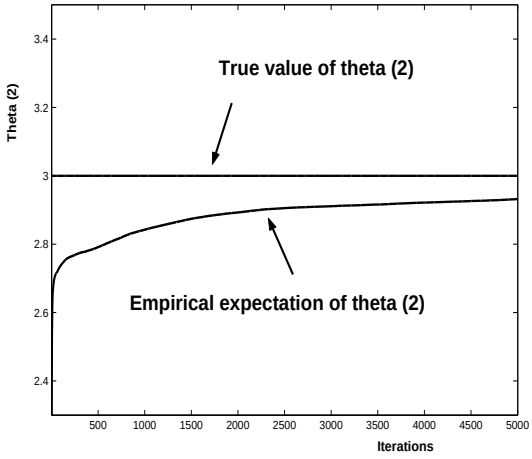

Figure $2.b$ Empirical expectation of θ_2

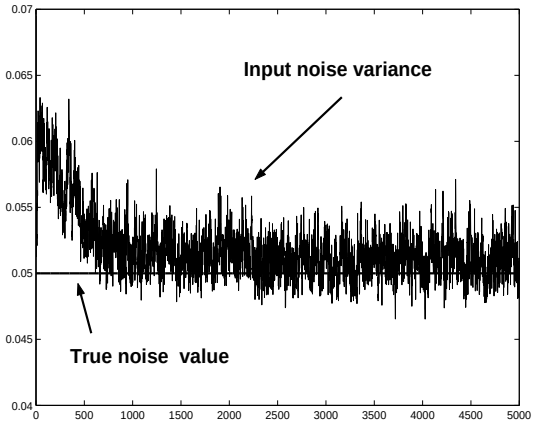

Figure 3.a Evolution of the input noise variance σ_η

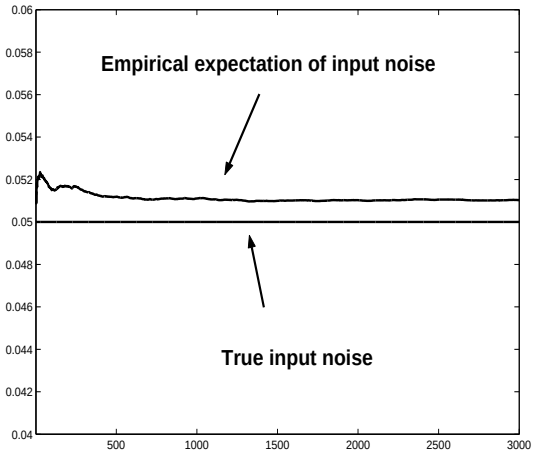

Figure 3.b Empirical expectation of the input variance σ_η

In classical regression modeling, we try to estimate the mapping $g : x \longrightarrow y = g(x)$ when the outputs y_i are noisy:

$$y_i = g(x_i, \boldsymbol{\theta}) + \epsilon_i \tag{21}$$

where the output noise ϵ_i is additive. However, when the function g is bijective in a certain input interval, one can try to estimate the inverse function $h = g^{-1}$. If we use the same data (x_i, y_i), equation (21) becomes:

$$x_i = h(y_i - \epsilon_i, \boldsymbol{\eta}) \tag{22}$$

where $\boldsymbol{\eta}$ are the parameters of the function h. We note that the noise is no longer additive but it is transformed under the nonlinear function h. However, we obtain an error in variables problem since the inputs y_i are noisy. Thus we can directly apply the proposed algorithm to estimate the inverse function $h = g^{-1}$. For the above example, $g(x) = \theta_1 \exp[-\theta_2 x]$, we have:

$$h(y) = g(y)^{-1} = -\frac{1}{\theta_2} \log \frac{y}{\theta_1} = \eta_1 \log(\eta_2 y)$$

with $\eta_1 = -1/3$ and $\eta_2 = 1/2$. Figures 4a, 4b, 5a and 5b illustrate the success of the proposed algorithm in estimating the parameters $\boldsymbol{\eta}$ of the inverse function h. Figures 6a and 6b show the evolution of the output noise estimation and the corresponding empirical

distribution.

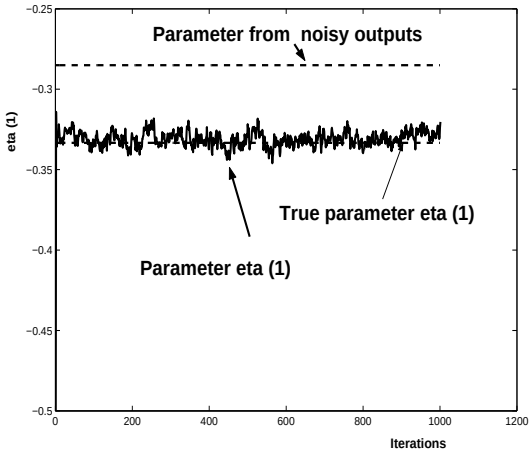

Figure $4.a$ Evolution of the parameter η_1

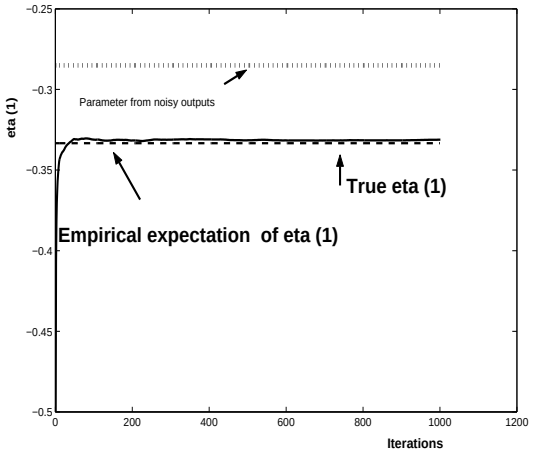

Figure $4.b$ Empirical expectation of η_1

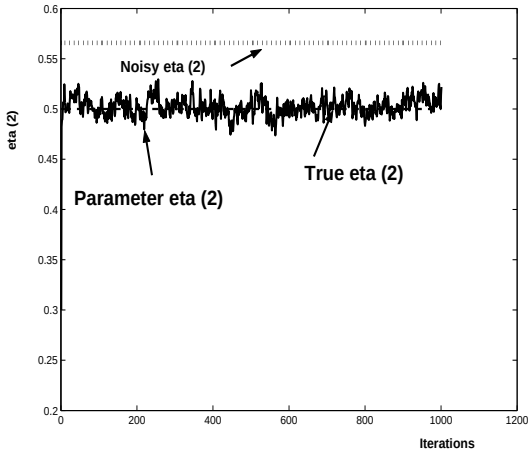

Figure $5.a$ Evolution of the parameter η_2

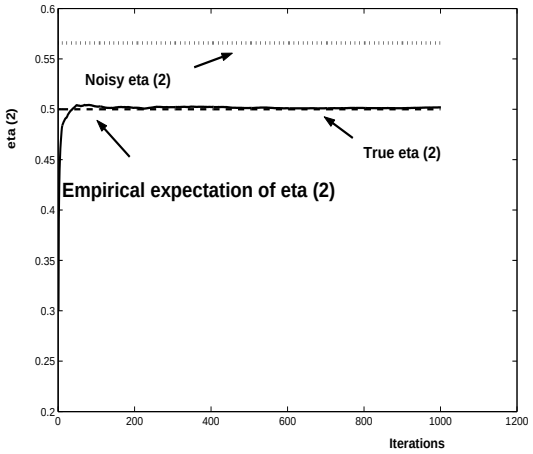

Figure $5.b$ Empirical expectation of η_2

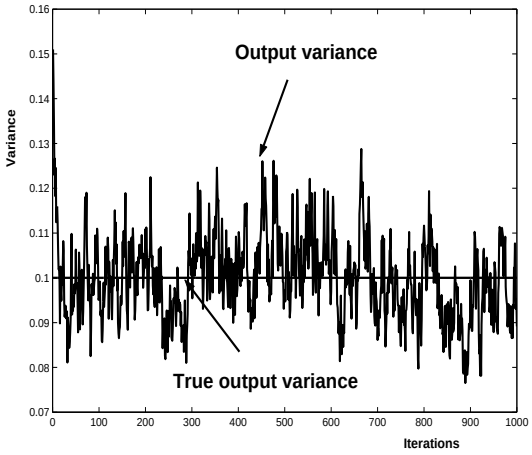

Figure 6.a Evolution of the output noise variance σ_ϵ

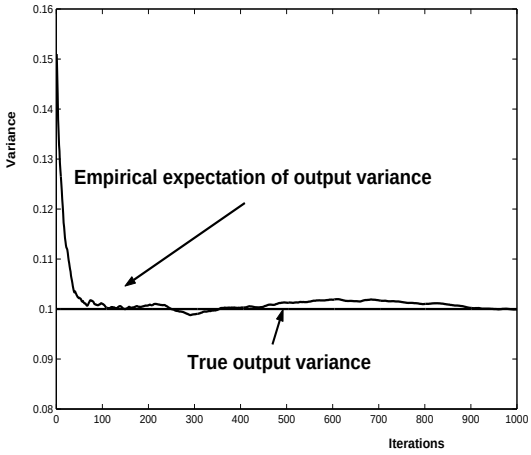

Figure 6.b Empirical expectation of the input variance σ_ϵ

Example 2: Feedforward Neural Network

As noted in the previous section, the algorithm can be applied in the nonparametric case. The form of the function g mapping the inputs x to the outputs y can be unknown or very complex. Therefore, we try in this section the learning with a feedforward neural network. To illustrate the performance of the proposed estimation algorithm, we tried to fi t the following function

$$y = g(x) = 3\sin(-1 + 7\sin^5(e^x)), \tag{23}$$

using a feedforward neural network f with 6 neurons in the hidden layer, based on data generated in such a way

$$\begin{cases} y_i = g(x_i^*, \boldsymbol{\theta}) + \epsilon_i, \quad i = 1, ..., 20, \\ x_i = x_i^* + \eta_i, \end{cases} \tag{24}$$

where $\eta_i \propto N(0, \sigma = 0.05)$ and $\epsilon_i \propto N(0, \sigma = 0.1)$ and the design points are uniformly distributed on $[0, 0.6]$. To make comparison, the feedforward neural network parameters vector is adjusted using both the classical Backpropagation algorithm and the proposed algorithm, based on the noisy inputs noisy outputs training data set. In Figure 7, we plot the estimated function produced by the feedforward neural network whose parameters vector has been adjusted using the Backpropagation algorithm based on the noisy inputs noisy outputs training data set. We can note the poor results due in large part to the error bias carried by the noisy training inputs $x_t, t = 1.., 20$. In Figure 8, we plot the estimated function produced by the feedforward neural network whose parameters vector has been adjusted using the proposed learning algorithm (the modifi ed version of the SEM algorithm) based on the noisy inputs / noisy outputs training data set. We can note the improvement produced in the estimation. Concerning the computational cost, each

step of the SEM algorithm consists of two operations: a very fast sampling operation due to the choice of a simple instrumental distribution and a classic network learning step. Consequently, if the stopping time of the algorithm is M, then the complexity is about $M \times (C_{sampling} + C_{learning})$, where $C_{sampling}$ is the complexity of sampling in the first step of the algorithm and $C_{learning}$ the complexity of learning. The diagnostic of convergence used here is the convergence of the empirical moments:

$$\begin{cases} \bar{\boldsymbol{\theta}}^m = \frac{1}{m}\sum_{k=1}^{m} \boldsymbol{\theta}^{(k)}, \\ \bar{x^*}^m = \frac{1}{m}\sum_{k=1}^{m} (x^*)^{(k)}. \end{cases}$$

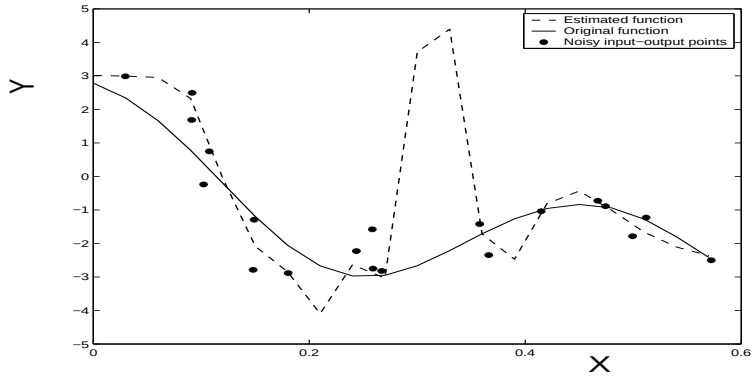

Figure 7. The target function (solid line) and the estimated function (dashed line) using the Backpropagation from the noisy input output training data set $(\bullet)$.

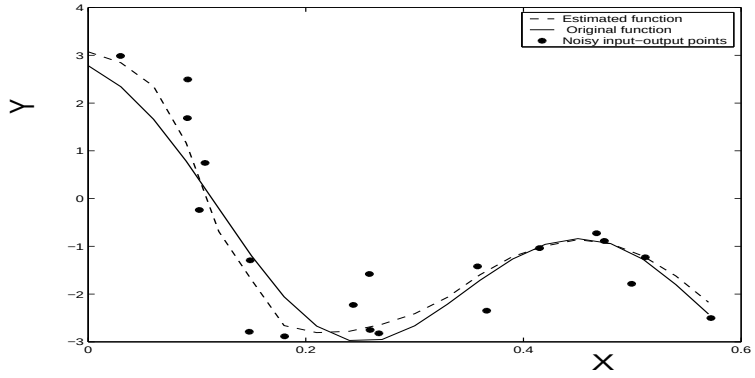

Figure 8. The target function $(\_)$ and the estimated function $(--)$ using the proposed algorithm from the noisy input output training data set $(\bullet)$.

CONCLUSION

The task of learning feedforward neural networks with noisy inputs / noisy outputs training data pairs has been studied. A direct approach using the Backpropagation algorithm which assumes noiseless training inputs will produce a biased estimate. The construction of learning algorithms based on cost functions obtained using the errors-in-variables model and the maximum likelihood approach is, if not intractable, computation consuming because of the high number of unknown parameters (which includes the parameters vector of interest and the true but unknown training inputs). In this paper, to overcome this problem, a Bayesian approach has been adopted treating the true but unknown training inputs as hidden variables. A modified version of the SEM algorithm has been proposed to compute the maximum *a posteriori* estimate of the parameters vector where the sampling is used to marginalize over the unknown training inputs. Two simulation examples have been presented to illustrate the effectiveness of the proposed learning method. A first example of parametric learning is considered where we study the success of the algorithm to learn the inverse of the bijective parametric function. The second example illustrates how the algorithm can be implemented with a nonparametric learning machine.

REFERENCES

1. A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, Vol. 39, pp. 930-945, 1993.
2. W. A. Fuller, *Measurement errors models*, New York, Wiley, 1987.
3. A. Kukush and S. Zwanzig, "On inconsistency of the least squares estimator in nonlinear errors-in-variables models," *Unpublished manuscrit*, 1996.
4. Y. Amemiya and W. A. Fuller, "Estimation for the nonlinear functional relationship," *The Annals of Statistics*, Vol. 16, pp. 147-160, 1988.
5. A. R. Gallant, *Nonlinear Statistical Models*, New York, Wiley, 1986.
6. A. K. Seghouane and Gilles Fleury, "An iterative algorithm for learning with input noise," *Internal Report*, No 2, Service des Mesures, SUPELEC, 2001.
7. J. D. Kalbfleisch and D. A. Sprott, "Application of likelihood methods to models involving large number of parameters," *Jouranl of the Royal statistical Society B*, Vol. 32, pp. 175-208, 1970.
8. A.W. Van der Vaart, *Asymptotic Statistics*, New York, Cambridge Unversity Press, 1998.
9. R.E. Kass and L. Wasserman, "Formal rules for selecting prior distributions: A review and annotated bibliography," *Technical report*, no. 583, Department of Statistics, Carnegie Mellon University, 1994.
10. R. J. A. Little and D. B. Rubin, "On jointly estimating parameters and missing data by maximizing the complete-data likelihood," *Journal of American Statistical Association*, vol. 37, pp. 218-220, 1983.
11. G. C. G. Wei and M. A. Tanner, "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms," *Journal of American Statistical Association*, vol. 85, pp. 699-704, 1990.
12. W.K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications", *Biometrika*, vol. 57, pp. 97-109, 1970.
13. S. Brooks and G. Roberts, "Assessing convergence of Markov chain Monte Carlo algorithms," *URL:http://www.stats.bris.ac.uk/MCMC/*, 1997
14. S. F. Nielsen, "The stochastic EM algorithm: Some large sample results," Manuscript, Department of Theoretical Statistics, University of Copenhagen. *URL:http://www.math.ku.dk/ feodor/publications.html*, 1997
15. H. Snoussi and A. Mohammad-Djafari, "Penalized maximum likelihood for multivariate Gaussian

mixture", in *Bayesian Inference and Maximum Entropy Methods*, R. L. Fry, Ed. MaxEnt Workshops, August 2002, pp. 36–46, Amer. Inst. Physics.
16. C. Robert, *Méthodes de Monte-Carlo par chaînes de Markov*, Economica, Paris, France, 1996.
17. A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *J. R. Statist. Soc. B*, vol. 39, pp. 1–38, 1977.