A Sparse Nonlinear Bayesian Online Kernel Regression

Matthieu Geist[1,2], Olivier Pietquin[1] and Gabriel Fricout[2]
[1]SUPELEC, IMS Research Group, Metz, France
[2]ArcelorMittal Research, MCE Department, Maizières-lès-Metz, France

Abstract

In a large number of applications, engineers have to estimate values of an unknown function given some observed samples. This task is referred to as function approximation or as generalization. One way to solve the problem is to regress a family of parameterized functions so as to make it fit at best the observed samples. Yet, usually batch methods are used and parameterization is habitually linear. Moreover, very few approaches try to quantify uncertainty reduction occurring when acquiring more samples (thus more information), which can be quite useful depending on the application. In this paper we propose a sparse nonlinear bayesian online kernel regression. Sparsity is achieved in a preprocessing step by using a dictionary method. The nonlinear bayesian kernel regression can therefore be considered as achieved online by a Sigma Point Kalman Filter. First experiments on a cardinal sine regression show that our approach is promising.

1 Introduction

In a large number of applications, engineers have to estimate values of an unknown function given some observed samples. For example, in order to obtain a map of a wireless network coverage in a building, one solution would be to simulate the wave propagation in the building according to Maxwell's equations, which would be intractable in practice. An other solution is to measure the electromagnetic field magnitude in some specific locations, and to interpolate between theses observations in order to build a field map covering the whole building. This task is referred to as function approximation or as generalization. One way to solve the problem is to regress a family of parameterized functions so as to make it fit at best the observed samples. Lots of existing regression methods can be found in the literature for a wide range of function families. Artificial Neural Networks (ANN) [1] or Kernel Machines [6, 9] are popular methods. Yet, usually batch methods are used (gradient descent for ANN or Support Vector Regression for Kernel Machines); that is all the observed samples have to be known before regression is done. A new observed sample requires running again the regression algorithm using all the samples.

Online regression describes a set of methods able to incrementally improve the regression results as new samples are observed by recursively updating previously computed parameters. There exists online regression algorithm using ANN or Kernel Machines, yet the uncertainty reduction occurring when acquiring more samples (thus more information) is usually not quantified, as well as with the batch methods.

Bayesian methods are such recursive techniques able to quantify uncertainty about the computed parameters. They have already been applied to ANN [5, 8] and to some extent to Kernel machines [2, 10]. In this paper we propose a method based on the Bayesian filtering framework [3] for recursively regressing a non-linear function from noisy samples. In this framework a hidden state (here the regression parameter vector) is recursively estimated from observations (here the samples), while maintaining a probability distribution over parameters (uncertainty estimation).

The next section exposes more formally the basics of this framework. A practical solution is described in sections 3, 4 and 5. Then first results are given, showing the method is quite efficient and illustrating the uncertainty quantification.

2 Problem Statement

We address the problem of nonlinear function approximation. The aim here is to approximate a nonlinear function $f(x)$, $x \in \mathcal{X}$, where $\mathcal{X}$ is a compact set of $\mathbb{R}^n$, from noisy samples

$$(y_k = f(x_k) + v_k, x_k)_k \tag{1}$$

where k is the time index and v_k is the observation random noise, by a function $\hat{f}(x;\theta)$ parameterized by the vector θ.

Classical approaches are kernel-based regression meth-

978-0-7695-3369-8/08 $25.00 © 2008 IEEE
DOI 10.1109/ADVCOMP.2008.7

ods and the function of interest is approximated by

$$\hat{f}(x;\theta) = \sum_{i=1}^{p} \alpha_i K(x, x_i), \tag{2}$$

with $\theta = (\alpha_i)_{i=1}^{p}$ and where K is a kernel, that is a continuous, symmetric and positive semi-definite function. These methods rely on the Mercer theorem [9] which states that each Kernel is a dot product in a so-called feature space. More precisely, for any Kernel K, it exists a mapping $\varphi : \mathcal{X} \to \mathcal{F}$ such that

$$\forall x, y \in \mathcal{X},\, K(x,y) = \langle \varphi(x), \varphi(y) \rangle \tag{3}$$

For example, the Gaussian kernel

$$K_\sigma(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \tag{4}$$

is associated with a mapping from $\mathcal{X}$ to an infinite dimensional space. Thus, any linear regression algorithm which only uses dot products can be cast by this *Kernel Trick* into a nonlinear one by implicitly mapping the original space $\mathcal{X}$ to an higher dimensional one. The proposed approach can be summarized as follows.

In most approaches, the kernel has to be chosen beforehand. The regression is done on the weights α_i, but not on the kernel parameters (mean and deviation for Gaussian kernels), which casts this problem into a nonlinear regression one. Moreover, these methods are usually deterministic, and thus do not give uncertainty information about the quality of the parameter estimation. But this can be quite useful, depending on the application. Following [11] we use a state-space representation of this parameter estimation problem:

$$\begin{aligned} \theta_{k+1} &= \theta_k + n_k \\ y_k &= \hat{f}(x_k;\theta_k) + v_k \end{aligned} \tag{5}$$

where n_k is an artificial process noise. We aim to use bayesian filtering to obtain sequentially the posterior conditional probability distribution $p(\theta_k | y_{1:k})$.

Here we will focus on Gaussian kernels, but this approach can be straightforwardly extended to other kernels. As a preprocessing step, we put a prior on our kernel width (*id est* σ) and use a dictionary method [4] to compute an approximate basis of $\varphi(\mathcal{X})$. This gives a good sparse and observation noise independent initialisation for our kernel centers and number. We then use a Sigma Point Kalman Filter (SPKF) [8] to sequentially estimate the parameters, and our uncertainty on generalization.

3 Dictionary

A first problem is to choose the number p of kernel functions and the prior kernel centers. A variety of methods can be contemplated, the simplest one being to choose equally spaced kernel fonctions. However the method described below rests on the mathematical signification of kernels and basic algebra, and is thus well motivated.

As said in section 2, the kernel trick corresponds to a dot product in a feature space $\mathcal{F}$, associated with a mapping φ. By observing that although $\mathcal{F}$ is a (very) higher dimensional space, $\varphi(\mathcal{X})$ can be a quite smaller embedding, the objective is to find a set of p points in $\mathcal{X}$ such that

$$\varphi(\mathcal{X}) \simeq \mathrm{Span}\left\{\varphi(\tilde{x}_1), \ldots, \varphi(\tilde{x}_p)\right\} \tag{6}$$

This procedure is iterative. Suppose that samples $x_1, x_2, \ldots$ are sequentially observed. At time k, a dictionary

$$\mathcal{D}_{k-1} = (\tilde{x}_j)_{j=1}^{m_{k-1}} \subset (x_j)_{j=1}^{k-1} \tag{7}$$

of m_{k-1} elements is available where by construction feature vectors $\varphi(\tilde{x}_j)$ are approximately linearly independent in $\mathcal{F}$. A sample x_k is then uniformly sampled from $\mathcal{X}$, and is added to the dictionary if $\varphi(x_k)$ is linearly independent on $\mathcal{D}_{k-1}$. To test this, weights $a = (a_1, \ldots, a_{m_{k-1}})^T$ have to be computed so as to verify

$$\delta_k = \min_a \left\| \sum_{j=1}^{m_{k-1}} a_j \varphi(\tilde{x}_j) - \varphi(x_k) \right\|^2 \tag{8}$$

Formally, if $\delta_k = 0$ then the feature vectors are linearly dependent, otherwise not. Practically an approximate dependence is allowed, and δ_k will be compared to a predefined threshold ν determining the quality of the approximation (and consequently the sparsity of the dictionary). Thus the feature vectors will be considered as approximately linearly dependent if $\delta_k \leq \nu$.

By using the kernel trick and the bilinearity of dot products, equation (8) can be rewritten as

$$\delta_k = \min_{a \in \mathbb{R}^{m_{k-1}}} \left\{ a^T \tilde{K}_{k-1} a - 2a^T \tilde{k}_{k-1}(x_t) + K(x_k, x_k) \right\} \tag{9}$$

where

$$\left(\tilde{K}_{k-1}\right)_{i,j} = K(\tilde{x}_i, \tilde{x}_j) \tag{10}$$

is a $m_{k-1} \times m_{k-1}$ matrix and

$$\left(\tilde{k}_{k-1}(x)\right)_i = K(x, \tilde{x}_i) \tag{11}$$

is a $m_{k-1} \times 1$ vector. If $\delta_k > \nu$, $x_k = \tilde{x}_{m_k}$ is added to the dictionary, otherwise not. Equation (9) admits the analytical solution

$$a_k = \tilde{K}_{k-1}^{-1} \tilde{k}_{k-1}(x_k) \tag{12}$$

Moreover it exists a computationally efficient algorithm which uses the partitioned matrix inversion formula to construct this dictionary. The dictionary method is briefly sketched in Alg. 1, nevertheless see [4] for details.

Algorithm 1: Dictionary computation

inputs : a set of N samples randomly selected from $\mathcal{X}$, sparsification parameter ν

outputs: a dictionary $\mathcal{D}$

Initialization;
$\mathcal{D}_1 = \{\tilde{x}_1\}$;

Dictionary computation;
for $k = 1, 2, \dots N$ **do**
  Observe sample x_k;
  Compute approximate dependence:

$$\delta_k = \min_{a \in \mathbb{R}^{m_{k-1}}} \left\| \sum_{j=1}^{m_{k-1}} a_j \varphi(\tilde{x}_j) - \varphi(x_k) \right\|^2$$

  if $\delta_k > \nu$ **then**
    Add x_k to the dictionary: $\mathcal{D}_k = \mathcal{D}_{k-1} \cup \{x_k\}$
  else
    Let the dictionary unchanged: $\mathcal{D}_k = \mathcal{D}_{k-1}$

Thus, by choosing a prior on the kernel to be used, and by applying this algorithm to a set of points $(x_1, \dots, x_N)$ randomly sampled from $\mathcal{X}$, a sparse set of good candidates to the kernel regression problem is obtained. This method is theorically well founded, easy to implement, computationally efficient and it does not depend on kernels nor space's topology. Note that, despite the fact that this algorithm is naturally online, this dictionary cannot be built (straightforwardly) while estimating the parameters, since the parameters of the chosen kernels (such as mean and deviation for Gaussian kernels) will be parameterized as well. Observe that only bounds on $\mathcal{X}$ have to be known in order to compute this dictionary, and not the samples used for regression.

4 Sigma Point Kalman Filters

4.1 Bayesian filtering

The problem of Bayesian filtering can be expressed in its state-space formulation:

$$\begin{aligned} s_{k+1} &= f_k(s_k, n_k) \\ y_k &= g_k(s_k, v_k) \end{aligned} \tag{13}$$

The objective is to sequentially infer the hidden state s_k given the observations $y_{1:k}$. The state evolution is driven by the possibly nonlinear mapping f_k and the process noise n_k. The observation y_k is a function of the state s_k, corrupted by an observation noise v_k. If the mappings are linear and if the noises n_k and v_k are Gaussian, the optimal solution is given by the Kalman filter: quantities of interest are random variables, and inference (that is prediction of these quantities and correction of them given a new observation) is done online by propagating sufficient statistics (mean and variance) through linear transformations.

4.2 SPKF Approach

The Sigma Point framework [8] is a nonlinear extension of Kalman filtering (random noises are still Gaussian). The basic idea is that it is easier to approximate a probability distribution than an arbitrary nonlinear function. Recall that in the Kalman filter framework, basically linear transformations are applied to sufficient statistics. In the Extended Kalman Filter (EKF) approach, nonlinear mappings are linearized. In the Sigma Point approach, a set of so-called sigma points are deterministically computed using the estimates of mean and of covariance of the random variable of interest. These points are representative of the current distribution. Then, instead of computing a linear (or linearized) mapping of the distribution of interest, one calculates a nonlinear mapping of these sigma points, and use them to compute sufficient statistics of interest for prediction and correction equations, that is to approximate the following distributions:

$$p(S_k|Y_{1:k-1}) = \int_{\mathcal{S}} p(S_k|S_{k-1}) p(S_{k-1}|Y_{1:k-1}) dS_{k-1} \tag{14}$$

$$p(S_k|Y_{1:k}) = \frac{p(Y_k|S_k) p(S_k|Y_{1:k-1})}{\int_{\mathcal{S}} p(Y_k|S_k) p(S_k|Y_{1:k-1}) dS_k} \tag{15}$$

Note that SPKF and classical Kalman equations are very similar. The major change is how to compute sufficient statistics (directly for Kalman, through sigma points for SPKF). Alg. 2 sketches a SPKF update in the case of additive noise, based on the state-space formulation, and using the standard Kalman notations: $s_{k|k-1}$ denotes a prediction, $s_{k|k}$ an estimate (or correction), $P_{s,y}$ a covariance matrix, $\bar{n}_k$ a mean and k is the discrete time index. The reader can refer to [8] for details.

5 Proposed Algorithm

Recall that we want to sequentially approximate a nonlinear function, as samples (y_k, x_k) are available, with a set of kernels. We state this parameter estimation problem in a state-space problem:

$$\begin{aligned} \theta_{k+1} &= \theta_k + n_k \\ y_k &= \hat{f}(x_k; \theta_k) + v_k \end{aligned} \tag{16}$$

Algorithm 2: SPKF Update

inputs : $\bar{s}_{k-1|k-1}$, $P_{k-1|k-1}$
outputs: $\bar{s}_{k|k}$, $P_{k|k}$

Sigma-points computation;
Compute deterministically sigma-point set $S_{k-1|k-1}$ from $\bar{s}_{k-1|k-1}$ and $P_{k-1|k-1}$;

Prediction Step;
Compute $S_{k|k-1}$ from $f_k(S_{k-1|k-1}, \bar{n}_k)$ and process noise covariance;
Compute $\bar{s}_{k|k-1}$ and $P_{s_{k|k-1}}$ from $S_{k|k-1}$;

Correction Step;
Observe y_k;
$Y_{k|k-1} = g_k(S_{k|k-1}, \bar{v}_k)$;
Compute $\bar{y}_{k|k-1}$, $P_{y_{k|k-1}}$ and $P_{s_{k|k-1},y_{k|k-1}}$ from $S_{k|k-1}$, $Y_{k|k-1}$ and observation noise covariance;
$K_k = P_{s_{k|k-1},y_{k|k-1}} P^{-1}_{y_{k|k-1}}$;
$s_{k|k} = s_{k|k-1} + K_k(y_k - \bar{y}_{k|k-1})$;
$P_{s_{k|k}} = P_{s_{k|k-1}} - K_k P_{y_{k|k-1}} K_k^T$;

Note that here f does not depend on time, but our approach should be easily extended to nonstationary function approximation. Here the approximation is of the form

$$\hat{f}(x;\theta) = \sum_{i=1}^{p} \alpha_i K_{\sigma_i}(x, x_i) \tag{17}$$

$$\text{where } \theta = [(\alpha_i)_{i=1}^p,\ (x_i)_{i=1}^p,\ (\sigma_i)_{i=1}^p]^T \tag{18}$$

$$\text{and } K_{\sigma_i}(x, x_i) = \exp(-\frac{\|x - x_i\|^2}{2\sigma_i^2}) \tag{19}$$

The problem is to find the optimal number of kernels and a good initialisation for our parameters.

As a preprocessing step, we use the dictionary (see section 3). We first put a prior σ_0 on the Gaussian width. We then sample uniformly N random points from $\mathcal{X}$. Finally, we use these samples to compute the dictionary. We thus obtain a set of p points $\mathcal{D} = \{x_1, \ldots, x_p\}$ such that

$$\varphi_{\sigma_0}(\mathcal{X}) \simeq \text{Span}\left\{\varphi_{\sigma_0}(x_1), \ldots, \varphi_{\sigma_0}(x_p)\right\} \tag{20}$$

where φ_{σ_0} is the mapping corresponding to the kernel K_{σ_0}. Note that even if the sparsification procedure is offline, the algorithm (the regression part) is online. Moreover, we do not need any training sample for this preprocessing step, but only classical prior which is anyway needed for the bayesian filter (σ_0), one parameter ν and bounds for $\mathcal{X}$. These requirements are generally not restrictive.

We then use a SPKF to estimate our parameters. As for any bayesian approach we have to put a prior on our parameter distribution. We state that

$$\theta_0 \sim \mathcal{N}(\bar{\theta}_0, \Sigma_{\theta_0}) \tag{21}$$

where

$$\bar{\theta}_0 = [\alpha_0, \ldots, \alpha_0, \mathcal{D}, \sigma_0, \ldots, \sigma_0]^T \tag{22}$$

$$\Sigma_{\theta_0} = \text{diag}([\sigma^2_{\alpha_0}, \ldots, \sigma^2_{\alpha_0}, \sigma^2_{\mu_0}, \ldots, \sigma^2_{\mu_0}, \sigma^2_{\sigma_0}, \ldots, \sigma^2_{\sigma_0}]) \tag{23}$$

In these expressions, α_0 is the prior mean on kernel weights, $\mathcal{D}$ is the dictionary computed in the preprocessing step, σ_0 is the prior mean on kernel deviation, and $\sigma^2_{\alpha_0}$, $\sigma^2_{\mu_0}$, $\sigma^2_{\sigma_0}$ are respectively the prior variance on kernel weights, centers and deviations. All these parameters (except the dictionary) have to be set up beforehand. Note that $\bar{\theta}_0 \in \mathbb{R}^{3p}$ and $\Sigma_{\theta_0} \in \mathbb{R}^{3p \times 3p}$. We also have to put a prior on noises. We state that $n_0 \sim \mathcal{N}(0, R_{n_0})$ where $R_{n_0} = \sigma^2_{n_0} I_{3p}$, I_{3p} being the identity matrix, and that $v \sim \mathcal{N}(0, R_v)$, where $R_v = \sigma^2_v I_m$, m being the size of observation vector. Thus, We just have to apply a SPKF Update (which updates the sufficient statistics of our parameters) at each time step, as a new training sample (y_k, x_k) is available.

Note that we use a specific form of SPKF, the so-called Square-Root Central Difference Kalman Filter (SR-CDKF) parameter estimation form. This implementation uses the fact that for this parameter estimation problem, the evolution equation is linear and the noise is additive. This approach is computationally cheaper: the complexity per iteration is $O(|\theta|^2)$, where $|\theta|$ is the size of our parameter vector ($O(|\theta|^3)$ in the general case). See [8] for details.

A last issue is to choose the artificial process noise. Formally, since the target function is stationary, there is no process noise. However introducing an artificial process noise can strengthen convergence and robustness properties of the filter. Choosing this noise is still an open research problem. We follow [8] and use a Robbins-Monro stochastic approximation scheme for estimating innovation. That is we set the process noise covariance as

$$R_{n_k} = (1 - \alpha_{RM}) R_{n_{k-1}} + \alpha_{RM} K_k (y_k - \hat{f}(x_k; \bar{\theta}_{k-1}))(y_k - \hat{f}(x_k; \bar{\theta}_{k-1}))^T K_k^T \tag{24}$$

Here α_{RM} is a forgetting factor set by the user, and K_k is the Kalman gain obtained during the SR-CDKF update. The observation noise is chosen constant. We summarize our approach in Alg.3.

6 Preliminary results

In this section we present the results of our preliminary experiments, which aim at regressing a cardinal sine ($\text{sinc}(x) = \frac{\sin(x)}{x}$) on $\mathcal{X} = [-10, 10]$. It is an easy problem, but a common benchmark too.

6.1 Problem statement and settings

At each time step k we observe $x_k \sim \mathcal{U}_\mathcal{X}$ (x_k uniformly sampled from $\mathcal{X}$) and $y_k = \text{sinc}(x_k) + w_k$ where

Algorithm 3: Proposed algorithm

inputs : ν, N, α_0, σ_0, σ_{α_0}, σ_{μ_0}, σ_{σ_0}, σ_{n_0}, σ_v
outputs: $\bar{\theta}$, Σ_θ

Compute dictionary;
$\forall i \in \{1 \dots N\},\ x_i \sim \mathcal{U}_\mathcal{X}$;
Set $X = \{x_1, \dots, x_N\}$;
$\mathcal{D}$ =`Compute-Dictionary` (X,ν,σ_0) ;

Initialisation;
Initialise $\bar{\theta_0}$, Σ_{θ_0}, R_{n_0}, R_v;

for $k = 1, 2, \dots$ **do**
  Observe (y_k, x_k);
  SR-CDKF update;
  $[\bar{\theta}_k, \Sigma_{\theta_k}, K_k]$ =`SR-CDKF-Update` $(\bar{\theta}_{k-1},\Sigma_{\theta_{k-1}}, y_k,x_k,R_{n_{k-1}},R_v)$;
  Artificial process noise update;
  $R_{n_k} = (1 - \alpha_{RM})R_{n_{k-1}} + \alpha_{RM} K_k (y_k - \hat{f}(x_k, \bar{\theta}_{k-1}))(y_k - \hat{f}(x_k, \bar{\theta}_{k-1}))^T K_k^T$;

$w_k \sim \mathcal{N}(0, \sigma_w^2)$. Note that we distinguish the true observation covariance noise σ_w^2 and our prior σ_v^2. For those experiments we set $\sigma_w = 0.1$. We set the algorithm's parameters to $N = 100$, $\sigma_0 = 1.6$, $\nu = 0.1$, $\alpha_0 = 0$, $\sigma_v = 0.5$, $\alpha_{RM} = 0.7$, and all variances $(\sigma_{\alpha_0}^2, \sigma_{\mu_0}^2, \sigma_{\sigma_0}^2, \sigma_{n_0}^2)$ to 0.1. Note that this parameters were not finely tuned (only orders of magnitude are important).

6.2 Quality of regression

We measure the quality of regression with the RMSE (Root Mean Square Error), that is

$$\text{RMSE} = \int_{x \in \mathcal{X}} \Big(f(x) - \hat{f}(x;\theta) \Big)^2 \, dx \qquad (25)$$

computed over 200 equally-spaced points. Averaged over 100 runs of our algorithm, we obtain a RMSE of 0.0676 ± 0.0176 for 50 samples, of 0.0452 ± 0.0092 for 100 samples, and of 0.0397 ± 0.0065 for 200 samples, for an average of 9.6 kernels. This is illustrated on figure 1. We give a typical result in Fig. 2 (50 observed samples). The dotted line, solid line and the crosses represent respectively the cardinale sine, the regression and the observations.

The proposed algorithm compares favorably with state-of-the-art batch and online algorithms. Table 1 shows the performance of the proposed algorithm and of other methods (some being online and other batch, some handling uncertainty information and other not), for which results are reproduced from [10] (see this paper and references therein for details about these algorithms). For each method is given the RMSE as well as the number of kernel functions,

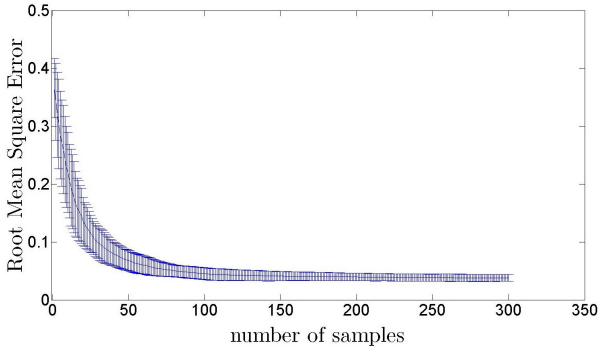

Figure 1. Mean and standard deviation of RMSE over 100 trainings.

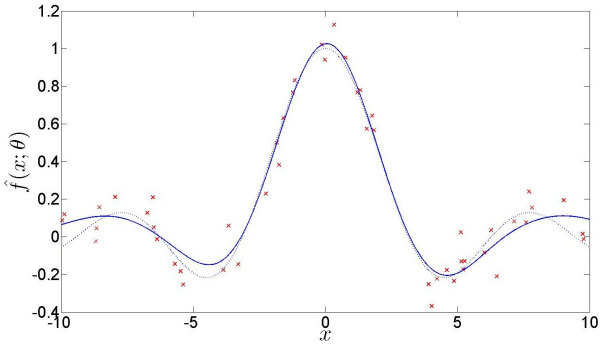

Figure 2. Typical regression. The cardinal sine is in dashed line, the regression in solid line and noisy observations are crosses.

and the associated variations, the SVM (Support Vector Machine) [9] being the baseline. We achieve the best RMSE with slightly more kernels than other approaches. However parameters can be tuned in order to address the trade-off between number of kernels and quality of approximation. Moreover, recall that our parameters were not finely tuned.

6.3 Uncertainty of generalization

Through the sigma point approach, we are also able to derive a confidence interval over $\mathcal{X}$. This allows to quantify the uncertainty of the regression at any point (and not a global upper bound as it is often computed in kernel-based regression methods). A typical confidence interval is illustrated on Fig.3. The dotted line, solid line and the crosses represent respectively the cardinale sine, the confidence interval and the observations. It can be particularly useful when the regression is used in a control framework, where this confidence approach can be used to take more cautious decisions (see [7] for example).

In Fig. 3, the samples used to feed the regressor are sam-

Method	test error	# kernels
Proposed algorithm	**0.0385 (-25.8%)**	**9.6 (-65.7%)**
Figueiredo	0.0455 (-12.3%)	7.0 (-75%)
SVM	**0.0519 (-0.0%)**	**28.0 (-0%)**
RVM	0.0494 (-4.8%)	6.9 (-75.3%)
VRVM	0.0494 (-4.8%)	7.4 (-73.5%)
MCMC	0.0468 (-9.83%)	6.5 (-76.8%)
Sequential RVM	0.0591 (+13.8%)	4.5 (-83.9%)

Table 1. Comparative results (batch and online results are taken from [10]).

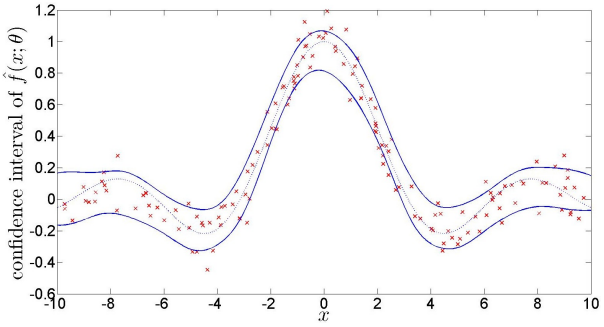

Figure 3. Confidence interval (uniform distribution of samples). The cardinal sine is in dashed line, the confidence interval in solid line and noisy observations are crosses.

pled uniformly, and thus the associated confidence interval has an approximately constant width. However, a regressor which handle uncertainty should do it locally. This is indeed the case for the proposed algorithm. In Fig. 4, the distribution of samples is Gaussian. It can be seen that the confidence interval is much larger where samples are less frequent (close to the bounds).

7 Future work

We have proposed a bayesian approach to online nonlinear kernel regression with a preprocessing sparsification procedure. Our method has proven to be effective on a simple cardinale sine regression problem. This simple example demonstrated that the proposed approach compares favorably with the state-of-the-art methods and it illustrated how the uncertainty of generalization is quantified. We aim to extend this framework to an online sparsification procedure, and to the problem where only a noisy known nonlinear mapping of the function of interest is available (here we directly observe the noisy function of interest). We also intend to work on the artificial process noise to derive more efficient exploration of the parameter space.

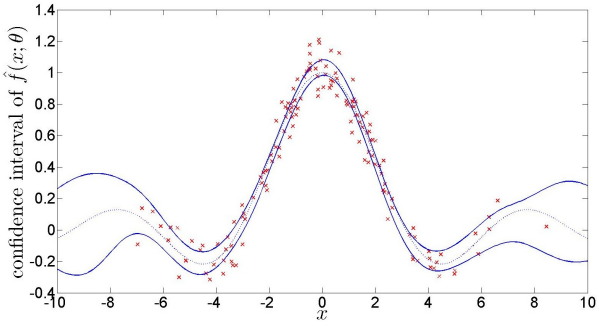

Figure 4. Confidence interval (normal distribution of samples). The cardinal sine is in dashed line, the confidence interval in solid line and noisy observations are crosses.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, NY, USA, 1995.

[2] C. M. Bishop and M. E. Tipping. Bayesian Regression and Classification. In *Advances in Learning Theory: Methods, Models and Applications*, volume 190, pages 267–285. OS Press, NATO Science Series III: Computer and Systems Sciences, 2003.

[3] Z. Chen. Bayesian Filtering : From Kalman Filters to Particle Filters, and Beyond. Technical report, Adaptive Systems Lab, McMaster University, 2003.

[4] Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, Hebrew University, April 2005.

[5] L. Feldkamp and G. Puskorius. A signal processing framework based on dynamic neural networks withapplication to problems in adaptation, filtering, and classification. In *Proceedings of the IEEE*, volume 86, pages 2259–2277, 1998.

[6] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[7] A. L. Strehl, L. Li, and M. L. Littman. Incremental model-based learners with formal learning-time guarantees. In *22nd Conference on Uncertainty in Artificial Intelligence*, pages 485–493, 2006.

[8] R. van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. PhD thesis, OGI School of Science & Engineering, Oregon Health & Science University, April 2004.

[9] V. N. Vapnik. *Statisical Learning Theory*. John Wiley & Sons, Inc., 1998.

[10] J. Vermaak, S. J. Godsill, and A. Doucet. Sequential Bayesian Kernel Regression. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[11] Wan, E. A. and Van Der Merwe, R. . The Unscented Kalman Filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, 2000.