Neurocomputing 74 (2011) 1945–1955

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An approximate inference with Gaussian process to latent functions from uncertain data

Patrick Dallaire *, Camille Besse, Brahim Chaib-draa

DAMAS Laboratory, Computer Science and Software Engineering Department, Laval University, Canada

ARTICLE INFO

Available online 19 February 2011

Keywords:
Supervised learning
Gaussian processes
Data uncertainty
Dynamical systems

ABSTRACT

Most formulations of supervised learning are often based on the assumption that only the outputs data are uncertain. However, this assumption might be too strong for some learning tasks. This paper investigates the use of Gaussian processes to infer latent functions from a set of uncertain input–output examples. By assuming Gaussian distributions with known variances over the inputs–outputs and using the expectation of the covariance function, it is possible to analytically compute the expected covariance matrix of the data to obtain a posterior distribution over functions. The method is evaluated on a synthetic problem and on a more realistic one, which consist in learning the dynamics of a cart–pole balancing task. The results indicate an improvement of the mean squared error and the likelihood of the posterior Gaussian process when the data uncertainty is significant.

1. Introduction

Supervised learning refers to the class of problems where a learner has to infer a function $f : X \rightarrow Y$ given a set of labelled examples. A typical underlying assumption concerning the training examples is that the inputs are known precisely and that only the output variables are affected by noise. However, this assumption can be unrealistic in many applications. There might be uncontrollable effects that corrupt the inputs in the data gathering process, which can affect the resulting quality of the estimated model if this fact is ignored in the learning formulation.

The problem of learning in the context of noisy input and output has been tackled in several ways. In computational mathematics and engineering, one generally uses a method known as *total least-squares* which consists in finding a minimal correction to apply on all data points such that the modified data satisfies a linear relation [1]. For their part, statisticians have investigated the *error-in-variables* models to deal with errors on the dependent variables as well as on the independent ones [2]. The general error-in-variables approach aims to create latent variables that correspond to the true observations which follow the underlying sought relation. There is a close link between the error-in-variables models and the total least-squares methods. The statistical model which corresponds to the basic total least-squares approach is the error-in-variables model with the restrictive condition that the measurement errors are zero mean, independent and identically distributed [3].

Researchers in machine learning have also addressed the problem of learning from uncertain data. Tresp et al. [4] proposed incorporating deficient data into the training of a neural network by integrating over the uncertain input using an estimated probability distribution for these inputs. Moreover, they showed that the expectation of the learned function will be biased if the inputs are altered with Gaussian noise along with increased error bars on predictions. Wright et al. [5] presented a framework for Bayesian neural networks where they infer the regression over the noiseless input by using Markov chain Monte Carlo sampling over the hidden variables. More recently, Ting et al. [6] proposed a Bayesian linear regression model including a precision parameter which enforces interdependency between the input and output noise models. Their algorithm then attempts to clean the uncertain data by using the expectation-maximization principle.

Kernel methods have also been used to learn from uncertain inputs. For instance, in classification tasks, Bi and Zhang [7] proposed a statistical model that extends Support Vector Classification methods in order to deal with uncertain inputs. For this purpose, they considered that each unobserved input is associated to exactly one component of a Gaussian mixture model. Then, by estimating the parameters of their Gaussian noise model, they were able to take input uncertainty into consideration.

The virtues of Gaussian processes to learn from uncertain inputs have also been investigated by some researchers. First, Girard and Murray-Smith [8] showed that the covariance of the data can be approximated using a new correlated process. This process uses a second order Taylor expansion to obtain a

* Corresponding author.
E-mail addresses: dallaire@damas.ift.ulaval.ca (P. Dallaire), besse@damas.ift.ulaval.ca (C. Besse), chaib@damas.ift.ulaval.ca (B. Chaib-draa).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.09.024

corrected covariance function accounting for input noise. Quiñonero-Candela and Roweis [9] stated that computing the marginal likelihood analytically is generally not possible. However, optimizing a lower bound of this quantity can lead to denoised inputs while learning the model. On the other hand, instead of approximating the covariance matrix, both previous authors mentioned that taking the expectation could lead to better estimation of the marginal likelihood [9,10].

In the same vein, this paper investigates an approach in which we not only make predictions for uncertain inputs with Gaussian processes, but also we learn from uncertain training sets. To do so, we marginalize out the inputs' uncertainty to obtain the expected covariance matrix and keep an analytical posterior distribution over functions. The main contribution of this paper is a formulation which allows learning from uncertain inputs and outputs by using Gaussian processes exactly as in the noise-free case. Results show that taking into account the uncertainty with this method improves the mean squared error and the likelihood of the sought function. As the noise decreases, the method tends towards its noise-free counter-part and therefore, cannot do worse than standard Gaussian process regression. To show that, the method is applied to the well-known task of balancing a pole over a cart, where the problem is to learn the four-dimensional (+ control) nonlinear dynamics of the system.

This paper is structured as follows. First, we formalize the problem of learning with Gaussian processes and the regression model. Then, in Section 3, we present insights to account for uncertain data. In Section 4, we present the experimental results on a difficult synthetic problem and on a more realistic problem. Section 5 discusses the results before Section 7 opens research avenues and concludes the paper.

2. Gaussian processes

Gaussian Processes (GPs) are stochastic processes which are used in machine learning to describe distributions directly into the space of functions. In supervised learning, we have to make the mandatory assumption that training examples are informative for further predictions. This information can be formalized is many ways. Some methods use correlation between the examples to express this information. When using Gaussian processes, it is assumed that the joint distribution of the data is a multivariate Gaussian. Consequently, the problem is to find a covariance function which explains the data properly. One interesting property of GPs is that they provide a probabilistic approach to the learning task and give uncertainty estimates while making predictions.

2.1. Gaussian processes for regression

As mentioned above, it is assumed that the joint distribution of a finite set of output observations given their inputs is multivariate Gaussian. Thus, a GP is fully specified by its mean function $\mu(\mathbf{x})$ and covariance function $C(\mathbf{x},\mathbf{x}')$. First, we assume that a set of training data $\mathcal{D}=\{\mathbf{x}_i,y_i\}_{i=1}^N$ is available where $\mathbf{x}_i\in\mathbb{R}^D$, y_i is a scalar observation such that

$$y_i = f(\mathbf{x}_i)+\varepsilon_i \tag{1}$$

and where ε_i is a white Gaussian noise with variance σ_ε^2. For convenience, we use the notation $X=[\mathbf{x}_1,\ \ldots,\ \mathbf{x}_N]$ for inputs and $\mathbf{y}=[y_1,\ \ldots,\ y_N]$ for outputs. According to the definition of a GP, the joint distribution of the training set is $\mathcal{N}(\mathbf{m},\mathbf{K}+\sigma_\varepsilon^2\mathbf{I})$, where the vector $\mathbf{m}$ and the matrix $\mathbf{K}$ are computed with the mean and covariance functions. The components of $\mathbf{m}$ are computed such that $m_i=\mu(\mathbf{x}_i)$. However, for the sake of notational simplicity, we make the weak assumption that the mean function is $\mu(\mathbf{x})=0$.[1] Then, the joint distribution of the training set becomes

$$\mathbf{y}|X\sim\mathcal{N}(\mathbf{0},\mathbf{K}+\sigma_\varepsilon^2\mathbf{I}) \tag{2}$$

where $\mathbf{K}$ is the covariance matrix whose entries K_{ij} are given by the covariance function $C(\mathbf{x}_i,\mathbf{x}_j)$. This equation is fundamental, since it corresponds to the *Gaussian process prior* assumption, which is used hereafter to make predictions.

There is a variety of covariance functions, each of them making different forms of functions more probable than others for a Gaussian process. Selecting the covariance function is an important aspect of the learning task. In particular, using Gaussian processes with certain covariance functions corresponds to using a neural network having an infinite number of sigmoidal or Gaussian hidden units [11]. This choice has to be made a priori and reflects the prior knowledge concerning the function to estimate directly in the space of functions, the so called Gaussian process prior. The multivariate Gaussian prior (2) defined over the training observations can be used to compute the posterior distribution over functions. Since this posterior distribution is obtained by computing the conditional distribution of a Gaussian, the resulting posterior remains a Gaussian process.

Accordingly, making predictions can be done by using the posterior Gaussian process' mean and its associated measure of uncertainty, given by the posterior covariance. For a single test input $\mathbf{x}_*$, the predictive distribution of the *noise-free* output f_* is obtained by first forming the joint distribution with (2)

$$\begin{bmatrix}\mathbf{y}\\ f_*\end{bmatrix}\sim\mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ 0\end{bmatrix},\begin{bmatrix}\mathbf{K}+\sigma_\varepsilon^2\mathbf{I} & \mathbf{k}_*\\ \mathbf{k}_*^\top & k_{**}\end{bmatrix}\right) \tag{3}$$

where $\mathbf{k}_*$ is the $N\times 1$ vector of cross-covariance $C(\mathbf{x}_*,\mathbf{x}_i)$ between the test input $\mathbf{x}_*$ and the training inputs X, and k_* is the prior variance given by $C(\mathbf{x}_*,\mathbf{x}_*)$. By computing the conditional distribution of (3) such that

$$f_*|\mathbf{x}_*,X,\mathbf{y}\sim\mathcal{N}(\mu(\mathbf{x}_*),\sigma^2(\mathbf{x}_*))$$

we obtain the Gaussian predictive distribution for any single input $\mathbf{x}_*$ with mean and variance given by

$$\mu(\mathbf{x}_*)=\mathbf{k}_*^\top(\mathbf{K}+\sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{y} \tag{4}$$

$$\sigma^2(\mathbf{x}_*)=k_{**}-\mathbf{k}_*^\top(\mathbf{K}+\sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{k}_* \tag{5}$$

It should be pointed out that Eqs. (4) and (5) produce an estimation of the latent function f and not its noisy version y. The interested reader is invited to refer to [12] for more details on different formulations of Gaussian processes.

2.2. Gaussian process prior

Although many covariance functions can be used to define the Gaussian process prior in (2), this paper we will mainly focus on the squared exponential (se) covariance function:

$$C_{se}(\mathbf{x}_i,\mathbf{x}_j)=\sigma_f^2\exp(-\tfrac{1}{2}(\mathbf{x}_i-\mathbf{x}_j)^\top W^{-1}(\mathbf{x}_i-\mathbf{x}_j)) \tag{6}$$

which is one of the most widely used kernel functions. In particular, using this covariance function is equivalent to a linear combination of an infinite number Gaussian basis functions [13]. Since the outputs might be corrupted by noise, we add the covariance function of an independent noise process:

$$C_n(\mathbf{x}_i,\mathbf{x}_j)=\sigma_\varepsilon^2\delta_{ij} \tag{7}$$

[1] Its common to find the bias of the training set and to subtract it from the data, making the zero-mean prior the ideal choice.

where δ_{ij} is the Kronecker delta. It results in a covariance function parameterized by the vector of hyperparameters

$$\theta = [W, \sigma_f^2, \sigma_\varepsilon^2], \tag{8}$$

where W is the diagonal matrix of characteristic length-scale, which accounts for a different covariance measure for each input dimension, σ_f^2 is the signal variance influencing the order of magnitude of the functions and σ_ε^2 represents the variance of the noise process.

Varying these hyperparameters affects the interpretation of the training data by changing the shape of functions of the GP prior expects. It might be difficult a priori to fix the hyperparameters of a covariance function and to expect these to fit the observed data correctly. A common way to estimate the hyperparameters is to maximize the log marginal likelihood of the observations [12]. In that case, it corresponds to take the logarithm of (2) and maximize this quantity with respect to hyperparameters θ:

$$\log p(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathbf{y}^\top(\mathbf{K}+\sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}+\sigma_\varepsilon^2\mathbf{I}| - \frac{N}{2}\log 2\pi \tag{9}$$

since the joint distribution of the observations is a multivariate Gaussian under a zero-mean prior. The optimization problem can be tackled using any gradient method to find an acceptable local maxima or the global maxima if possible.

3. Learning with uncertain data

In this paper, we consider the difficult problem of learning from uncertain examples. The ideal Bayesian solution to this task would be to make predictions by integrating over the uncertainty of the training set such that

$$p(f_*|\mathbf{x}_*) = \iint p(f_*|\mathbf{x}_*, X, \mathbf{y}) p(X) p(\mathbf{y})\, dX\, d\mathbf{y} \tag{10}$$

but these integrals are analytically intractable in general. Thus, the following sections present an approach to approximate this predictive distribution, which deals with Gaussian input and output distributions.

3.1. Learning with uncertain inputs

As we stated earlier in Introduction, the assumption that only the outputs are noisy is not enough for some learning tasks. Consider the case where the inputs are uncertain and where each input distribution is known. In this case, using Gaussian process regression as previously formulated will not handle the uncertainty explicitly. Nevertheless, one can apply the standard method naively by using only one representative per input, such as the mean, and ignoring other information about its distribution.

For Gaussian processes, accounting for uncertain examples implies computing uncertain covariance between these examples. In fact, the covariance of any pair of examples has a distribution that depends on the covariance function and the probability distribution of each input. Consequently, the covariance matrix has a very large distribution which makes the computation of the posterior over functions impossible in general. To deal with this issue, one generally relies on an approximation of the covariance matrix distribution. In this paper, we investigate a case where the expected covariance matrix is analytically computable.

Consider the case where inputs are a set of Gaussian distributions and where the true input value $\mathbf{x}_i$ is not observable, but we have access to the parameters of its distribution. Thus, the expected *covariance* between uncertain data is obtained by computing expectations with respect to $\mathbf{x}_i$ and $\mathbf{x}_j$

$$K_{ij} = \iint C(\mathbf{x}_i, \mathbf{x}_j) p(\mathbf{x}_i) p(\mathbf{x}_j)\, d\mathbf{x}_i\, d\mathbf{x}_j, \quad \text{if } i \neq j \tag{11}$$

where $p(\mathbf{x}_i) = \mathcal{N}(\mathbf{u}_i, \mathbf{\Sigma}_i)$ and $p(\mathbf{x}_j) = \mathcal{N}(\mathbf{u}_j, \mathbf{\Sigma}_j)$. Note that this equation is used to compute the elements of the covariance matrix $\mathbf{K}$ which are off diagonal. Since the diagonal corresponds to variances, Eq. (11) cannot be used, because the two variables are in fact the same random variable. Therefore, the case of expected *variance* requires the computation of a single expectation:

$$K_{ii} = \int C(\mathbf{x}_i, \mathbf{x}_i) p(\mathbf{x}_i)\, d\mathbf{x}_i \tag{12}$$

which is only useful for non-stationary covariance function. For stationary covariance functions, such as the squared exponential, this integral is simplified to σ_f^2 by definition.[2]

Before discussing the expected squared exponential, we first derive the simpler case of the expected covariance of the linear covariance function C_{lin}. Basically, this covariance function can be expressed as

$$C_{lin}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j + \sigma_b^2 \tag{13}$$

where $\sigma_b^2 = 0$ makes the covariance function homogeneous and non-zero value makes it inhomogeneous. In practice, homogeneous functions will force the estimated functions to pass by the origin and increasing σ_b^2 allows functions to be more off origin. To obtain the expected version $C_{\mathbb{E}lin}$ of this function, we have to compute Eqs. (11) and (12), which results in (see Appendix A for derivation)

$$C_{\mathbb{E}lin}((\mathbf{u}_i, \mathbf{\Sigma}_i), (\mathbf{u}_j, \mathbf{\Sigma}_j)) = \mathbf{u}_i^\top \mathbf{u}_j + \sigma_b^2 + \delta_{ij}\mathrm{Tr}(\mathbf{\Sigma}_i) \tag{14}$$

where Tr denotes the trace of a matrix. The effect of the trace term here is to increase the output variance of training examples according to their input variance. Thus, the contribution of highly uncertain examples is automatically decreased.

It is also possible to compute the expectation for polynomial covariance function. However, as the degree increases, so the complexity of the resulting expected covariance function. For instance, in the case of the quadratic covariance function:

$$C_{qua}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + \sigma_b^2)^2 \tag{15}$$

the computation of the expectations involves expectation over quadratic forms of Gaussian variables, which yields in the following expected quadratic covariance function (see Appendix B for derivation)

$$\begin{aligned} C_{\mathbb{E}qua}((\mathbf{u}_i, \mathbf{\Sigma}_i), (\mathbf{u}_j, \mathbf{\Sigma}_j)) &= \mathrm{Tr}([\mathbf{\Sigma}_i + \mathbf{u}_i\mathbf{u}_i^\top][\mathbf{\Sigma}_j + \mathbf{u}_j\mathbf{u}_j^\top])(1+\delta_{ij}) \\ &+ 2\sigma_b^2 \mathbf{u}_i^\top \mathbf{u}_j + \sigma_b^4 + \delta_{ij}[\mathrm{Tr}(\mathbf{\Sigma}_i + \mathbf{u}_i\mathbf{u}_i^\top)^2 + 2\sigma_b^2 \mathrm{Tr}(\mathbf{\Sigma}_i) - 2(\mathbf{u}_i^\top \mathbf{u}_i)^2] \end{aligned} \tag{16}$$

where the last line is used only to compute expected variances. Using linear or quadratic covariance function is an arguably strong hypothesis on the space of functions. Therefore, one could look for a more general function.

It has been shown by Girard [10] that, for normally distributed inputs and using the squared exponential covariance function, integrating over independent input distributions is analytically feasible since it involves integrations over product of Gaussians. However, to obtain the correct expected squared exponential covariance function, we had to incorporate the variance case to the function. The resulting expected covariance $C_{\mathbb{E}se}$ is computed

[2] For stationary covariance function $C(\mathbf{x},\mathbf{x}) = C(\mathbf{0},\mathbf{0})$ is constant.

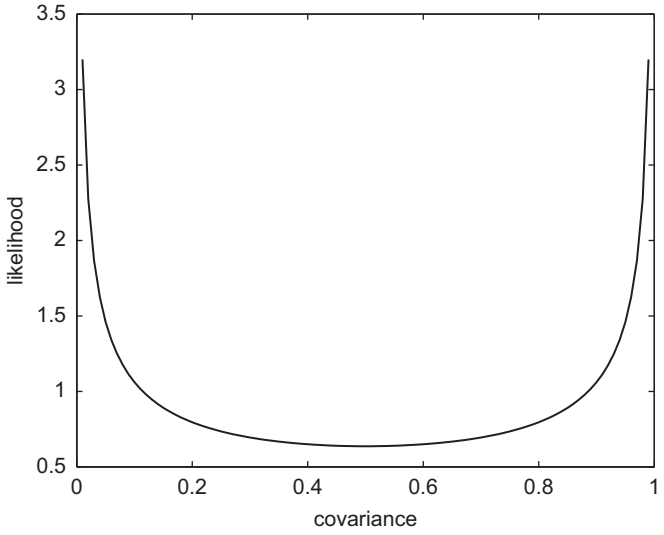

Fig. 1. Probability density function of the covariance between examples at $x_i \sim \mathcal{N}(0,1)$ and $x_j \sim \mathcal{N}(1,1)$ under a squared exponential with all parameters set to one. The expectation given by $C_{\mathbb{E}se}$ for these examples is 0.5.

exactly with (see Appendix C for derivation)

$$C_{\mathbb{E}se}((\mathbf{u}_i,\boldsymbol{\Sigma}_i),(\mathbf{u}_j,\boldsymbol{\Sigma}_j)) = \frac{\sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{u}_i-\mathbf{u}_j)^\top (W+\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)^{-1}(\mathbf{u}_i-\mathbf{u}_j)\right)}{|I+W^{-1}(\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)(1-\delta_{ij})|^{1/2}} \tag{17}$$

which is again a squared exponential form, where the covariance decreases as the uncertainty over the inputs increases, but farther examples now become correlated, thus making the resulting process smoother when examples are more uncertain. In Eq. (17), we introduced the inverse Kronecker delta $(1-\delta_{ij})$ in order to eliminate the reweighing effect of the denominator when computing a variance.

While computing the expected covariance matrix allows learning with Gaussian processes exactly as in the noise-free input case, this matrix can also be unlikely according to the full distribution over covariance matrices. For instance, Fig. 1 shows the case where the expected covariance between two training examples is exactly the worst choice. In that particular case, a better estimate would be to either say that the examples are covarying or not covarying at all. This setup is actually the worst possible, but it clearly indicates that the expected Gaussian process might not explain the training data properly in some cases.

A solution to this problem would be to clean the data by finding the maximum a posteriori or to compute a complete posterior Gaussian distribution on inputs (this avenue is discussed in Section 6), both modifying the covariance of the examples. However, among all possible density functions $p(C(\mathbf{x}_i,\mathbf{x}_j))$, only a few training examples suffer from this problem and the underlying function can still be inferred adequately for sufficiently large data sets.

3.2. Learning with uncertain outputs

So far, only the difficulty concerning uncertain inputs has been covered. Dealing with uncertain outputs having a Gaussian distribution is more manageable. Generally, the noise on the output is assumed stationary and has to be estimated from data. By observing each noisy output along with its variance, one can define a non-stationary noise process only known for the training examples. This noise process is introduced to reflect the outputs' uncertainty and decrease the influence of each example according to its respective variance. Formally, it entails a slight modification of the prior distribution $\mathbf{y}|X \sim \mathcal{N}(\mathbf{0},K+\sigma_\varepsilon^2)$, which consists in adding a diagonal covariance matrix containing the variances $\sigma_{y_i}^2$.

3.3. Learning the covariance function hyperparameters

Theoretically, learning with uncertain data is as difficult as in the noise-free case when using the covariance function (17), although it might require more data. The posterior distribution over functions is found by using Eqs. (4) and (5) with the expected covariance matrix. The hyperparameters of the covariance function can be learned with the log marginal likelihood of the expected process, but this likelihood is often riddled with many local maxima. Using standard conjugate gradient methods will often lead to a local maxima that might not explain the data properly. In the case of a squared exponential covariance function, the matrix W would tend to have large values on its diagonal, meaning that most dimensions are either very smooth or even irrelevant, and the value of σ_ε^2 would be overestimated to transpose the input error in the output dimensions.

A solution to this difficulty is to find a maximum a posteriori (MAP) estimation of the hyperparameters instead of a maximum likelihood. Specifying a prior on the hyperparameters will thus act as a regularization term preventing improper local maxima. As stated before, it is easy for Gaussian processes to explain everything as noise. To avoid such inappropriate explanation, larger values of noise and characteristic length-scales can be penalized. However, a bad local maxima can be an indication that there is no apparent structure in the data or no enough data to find correlations.

4. Experiments

The following experiments compare the performance of Gaussian process accounting for data uncertainty through the computation of the expected covariance matrix (uncertain-GP) and a Gaussian process which only uses the mean of the inputs (classic-GP). Both Gaussian processes are based on the squared exponential covariance function, where the classic-GP has only access to (6) and the uncertain-GP has access to its expected version (17). The purpose of these experiments is to determine whether using the expected covariance matrix leads to any improvement over the naive method that only uses the inputs' mean.

Indeed, the standard Gaussian process does not usually deal with uncertainty over the inputs. The expected behaviour of this method would be to explain Gaussian input uncertainty as extra Gaussian noise over the outputs. While this extra noise is in general non-Gaussian, it can still be approximated as Gaussian noise.

First, we evaluate the behaviour of each method on a one-dimensional synthetic problem and then compare their performances on a harder problem which consists in learning the nonlinear dynamics of a cart–pole system.

4.1. Synthetic problem: sincsig

To visualize easily the behaviour of both learning methods, we have chosen a one-dimensional task where the problem consists in learning a function composed by the unnormalized *sinc*

function and a biased *sigmoid* function such that

$$f(x)=\begin{cases}\sin(x)/x & \text{if } x\geq 0\\ 0.5/(1+\exp(-10x-5))+0.5 & \text{otherwise}\end{cases} \quad (18)$$

which we refer to as the *sincsig* function hereafter.

All evaluations have been done on randomly drawn training sets. Each data set is constructed by initially uniformly sampling N points in $[-10,10]$. These points are the noise-free inputs $\{\mathbf{x}_i\}_{i=1}^N$. To obtain the associated noisy outputs, each of them is sampled according to $y_i \sim \mathcal{N}(sincsig(x_i),\sigma_y^2)$. The set of uncertain inputs is constructed by first sampling the noise $\sigma_{x_i}^2$ to be applied on each input. Then, the sampled noise is used to corrupt the noise-free input x_i according to $u_i \sim \mathcal{N}(x_i,\sigma_{x_i}^2)$. It is easy to see that $x_i|u_i,\sigma_{x_i}^2 \sim \mathcal{N}(u_i,\sigma_{x_i}^2)$. Finally, the complete training set is defined as $\mathcal{D}=\{(u_i,\sigma_{x_i}^2),y_i\}_{i=1}^N$.

Fig. 2 shows a typical example of a training data set (crosses), with the true function to infer (solid line) and the resulting regression (thin line) for the uncertain-GP (top) and the classic-GP (bottom). Error bars around the thin lines indicate the confidence of the Gaussian process about its prediction and correspond to two standard deviations. Note here that whenever the solid line goes far from the error bars, the true underlying function becomes unlikely according to the prediction which can be considered as an inconsistent learning.

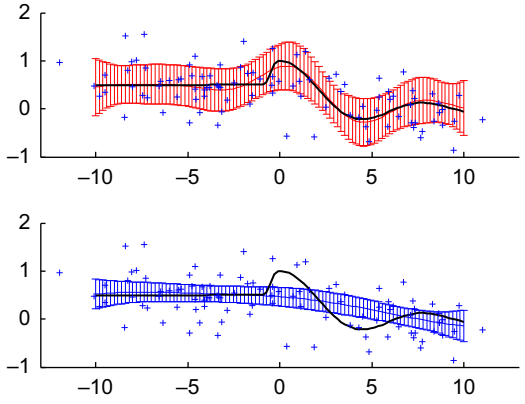

Fig. 2. Typical learning of the *sincsig* function with uncertain-GP (top) and classic-GP (bottom). Error bars represent two standard deviations. The crosses are the means of uncertain inputs and outputs.

The first experiment was designed to evaluate the learning performance on uncertain inputs only. Therefore, we conducted this one with a small output noise of standard deviation $\sigma_y=0.01$ with different size training sets. The input noise standard deviations σ_{x_i} were sampled uniformly in [0, 2.5], which causes the dataset to contain some very good examples on average. The standard deviations have been chosen high enough so that the noise over the inputs can be explained by adding an independent output noise during the optimization procedure and to measure the impact of highly uncertain examples on the posterior Gaussian process.

All performance evaluation of the classic-GP and the uncertain-GP have been done by training both with the same random data sets. For comparison purpose, we also trained a standard multilayer perceptron (MLP) having two hidden layers of five neurons. The weights of the network were found with the Levenberg–Marquardt algorithm [14]. Fig. 3(a) shows the averaged mean squared error over 25 randomly chosen training sets for various sizes. Results show that when very few data are available, both processes tend to explain the data with higher noise variance σ_ε^2 over the outputs. As expected, when the size of the data set increases, the classic-GP gives greater importance to the explanation that observed outputs are mostly noise. On the other hand, the uncertain-GP discriminates the most uncertain data and privileges the ones having less uncertainty in order to infer the function from the right data.

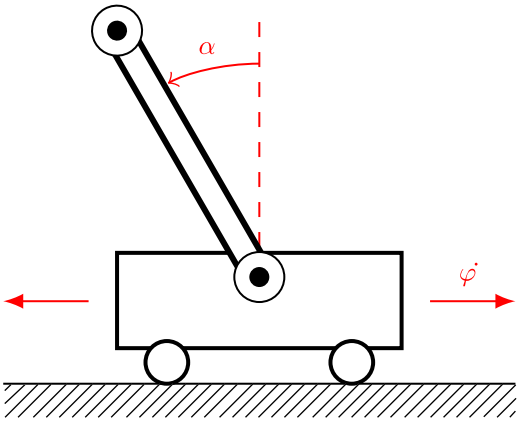

Fig. 4. The cart–pole balancing problem.

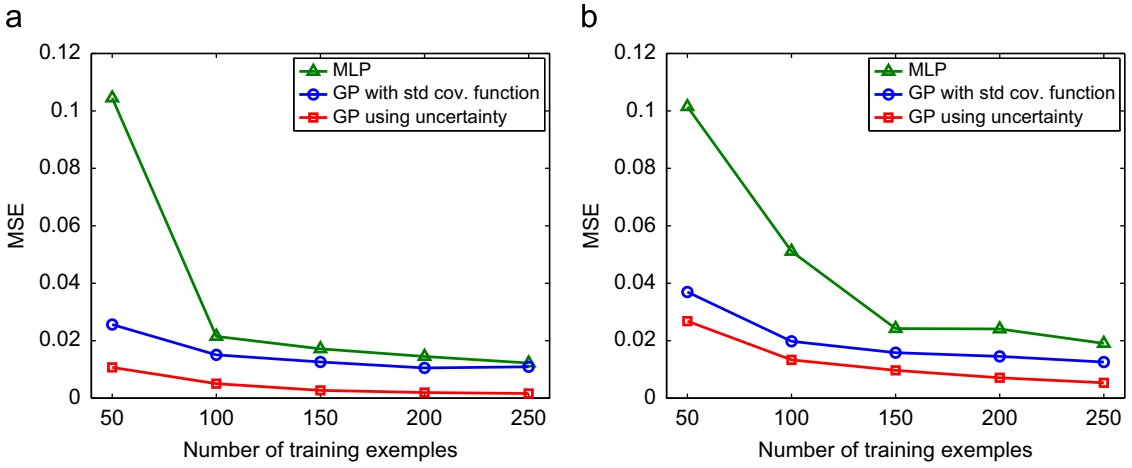

Fig. 3. Mean squared error results on the *sincsig* problem. Squared red line, uncertain-GP; circled blue line, classic-GP; triangled green line, MLP. (a) Output uncertainty is constant with $\sigma_y=0.1$ and (b) output uncertainty is random with $\sigma_{y_i} \sim \mathcal{U}(0.2,0.5)$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

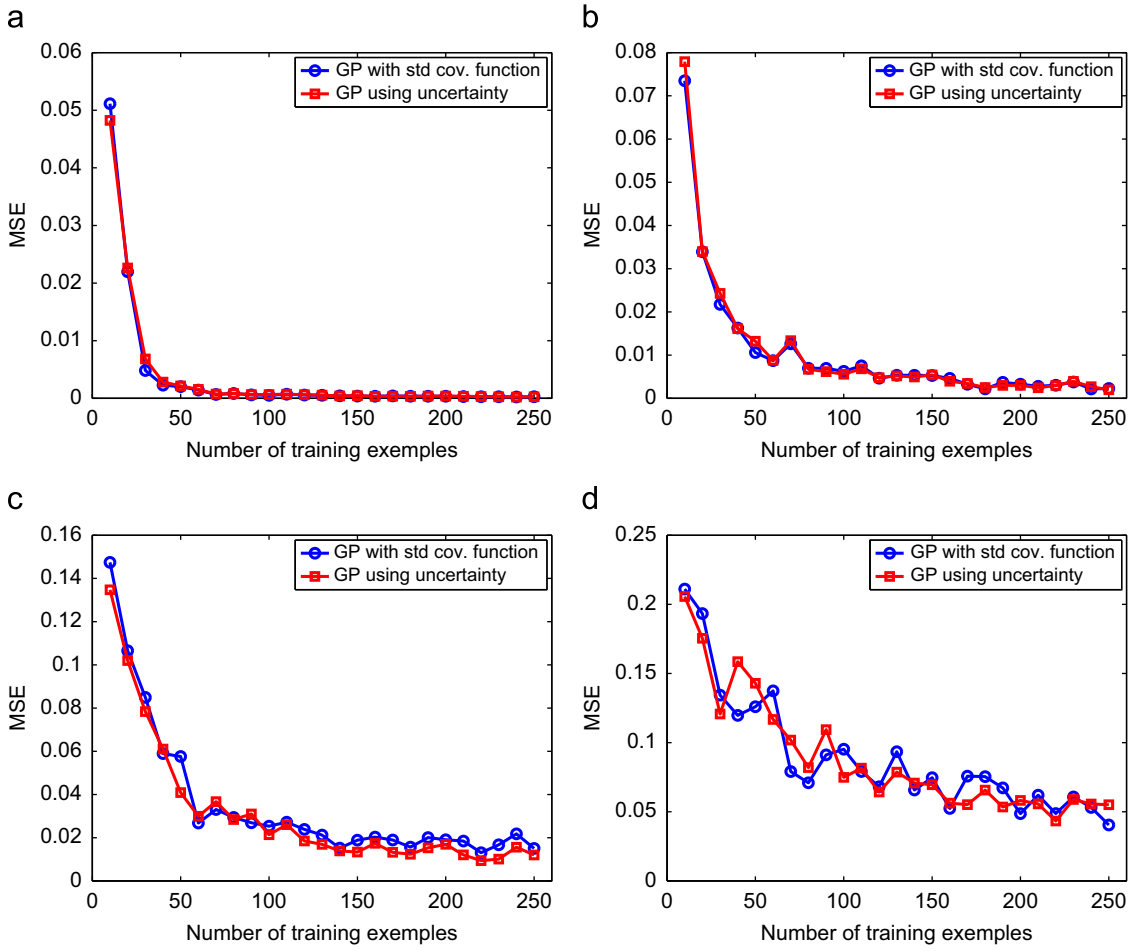

Fig. 5. Mean squared error results on *sincsig*. The input and output uncertainties are random with $\sigma_{x_i} \sim \mathcal{U}(0,v)$ and $\sigma_{y_i} \sim \mathcal{U}(0,v)$. The variable v is the upper bound on uncertainty. Squared red line, uncertain-GP; circled blue line, classic-GP. (a) With upper bound $v = 0.1$; (b) with upper bound $v = 0.4$; (c) with upper bound $v = 1$; (d) with upper bound $v = 3$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In the second experiment, we assumed that the Gaussian processes now know the noise's variance on the observations to evaluate the performance on (completely) uncertain datasets. Therefore, the noise hyperparameter σ_ε^2 is set to zero since the processes know exactly the noise matrix to add when computing the covariance matrix. For each output, the standard deviation σ_{y_i} is then uniformly sampled in [0,0.5] and the complete training set is defined as $\mathcal{D} = \{(u_i,\sigma_{x_i}^2),(z_i,\sigma_{y_i}^2)\}_{i=1}^N$. Fig. 3(b) shows the performance of the MLP, the classic-GP and the uncertain-GP on such training sets. Removing the independent noise process (7) from the prior has two effects: First, it prevents the classic-GP from explaining uncertain inputs by increasing the noise (hyperparameter) over the outputs, and it also forces the uncertain-GP to explain the data with the available information on the input variance.

The third experiment is set to validate the improvements brought by the expected covariance method. We thus let the optimization procedure choose itself the right kernel to use according to the uncertainty over the inputs. Consequently, we constructed two combinations of covariance functions. The first one (classic-GP) is based on the standard squared exponential covariance function (6) added to the known non-stationary noise process representing the outputs' uncertainty and an independent noise process given by

$$C(x_i,x_j) = C_{se}(x_i,x_j) + \delta_{ij}\sigma_{y_i}^2 + \delta_{ij}\sigma_\varepsilon^2 \tag{19}$$

where the first two terms are expected to explain the data and the last one to compensate the error due to inputs' noise.

The second one (uncertain-GP) is based on the expected squared exponential covariance function (17) added to the known non-stationary noise process representing the outputs' uncertainty and an independent noise process which is given by

$$C((u_i,\sigma_{x_i}^2),(u_j,\sigma_{x_j}^2)) = C_{\mathbb{E}se}((u_i,\sigma_{x_i}^2),(u_j,\sigma_{x_j}^2)) + C_{se}(u_i,u_j) + \delta_{ij}\sigma_{y_i}^2 + \delta_{ij}\sigma_\varepsilon^2 \tag{20}$$

where we expect the optimization process to balance the effect of each covariance function.

The experiments were conducted on various levels of noise and training set sizes. For these experiments, the uncertainty over the data is upper bounded by a standard deviation v. Like the previous experiments, noises are sampled independently from uniform distributions and are then applied to the data. In this case, $\sigma_{x_i} \sim \mathcal{U}(0,v)$ and $\sigma_{y_i} \sim \mathcal{U}(0,v)$. The performance of the classic-GP and the uncertain-GP has been measured with the mean squared error and the log-likelihood. The results are averaged over 10 trials for a same value of v and data set size. The likelihood is approximated with 100 equidistant noise-free points from the true function. The averaged mean squared error and averaged log-likelihood results specific to this third experiment are shown in Figs. 5 and 6 respectively.[3]

As stated in Section 3.3, it is possible to learn the hyperparameters given a training set. Since conjugate gradient methods

[3] Due to computational precision, likelihoods reached infinity making Fig. 6(a) incomplete.

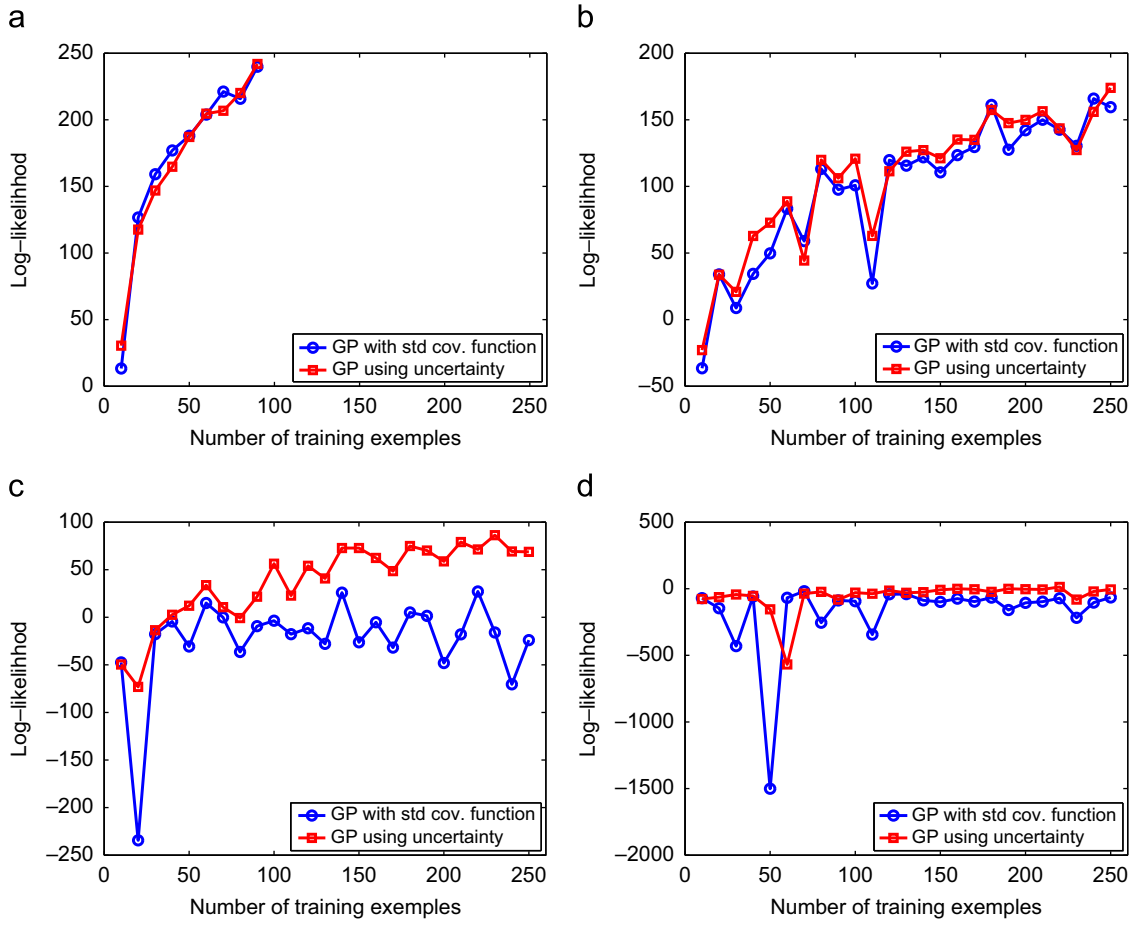

Fig. 6. Log-likelihood results on *sincsig*. The input and output uncertainties are random with $\sigma_{x_i} \sim \mathcal{U}(0,\nu)$ and $\sigma_{y_i} \sim \mathcal{U}(0,\nu)$. The variable ν is the upper bound on uncertainty. Squared red line, uncertain-GP; circled blue line, classic-GP. (a) With upper bound $\nu = 0.1$; (b) with upper bound $\nu = 0.4$; (c) with upper bound $\nu = 1$; (d) with upper bound $\nu = 3$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

performed poorly for the optimization of the log-likelihood in the uncertain-GP cases, we preferred stochastic optimization methods for this task. For these experiments, we did not use any prior on the hyperparameters to avoid helping the learning procedure by introducing additional information on the function.

4.2. The cart–pole problem

We now consider the harder problem of learning the cart–pole dynamics, which is also known as the inverted pendulum. Fig. 4 gives a picture of the system from which we try to learn the dynamics. The state is defined by the position (φ) of the cart, its velocity ($\dot{\varphi}$), the pole's angle (α) and its angular velocity ($\dot{\alpha}$). There is also a control input which is used to apply lateral forces on the cart. Following the equation in Florian [15] to govern the dynamics, we used Euler's method to update the system's state:

$$\ddot{\alpha} = \frac{g\sin\alpha + \cos\alpha((-F - m_p l\dot{\alpha}^2\sin\alpha)/(m_c+m_p))}{l(4/3 - (m_p\cos^2\alpha)/(m_c+m_p))}$$

$$\ddot{\varphi} = \frac{F + m_p l(\dot{\alpha}^2\sin\alpha - \ddot{\alpha}\sin\alpha)}{m_c+m_p}$$

where g is the gravity force, F is the force associated to the action, l is the half-length of the cart, m_p is the mass of the pole and m_c is the mass of the cart.

For this problem, the training sets were sampled exactly as in the *sincsig* case. State-action pairs were uniformly sampled on their respective domains. The outputs were obtained with the true dynamical system and then perturbed by noise with known random variance. Since the output variances are also known, the training set can be seen as Gaussian input distributions that map to Gaussian output distributions. Therefore, one might use a sequence of Gaussian belief states as its training set in order to learn a partially observable dynamical system. Following this idea, there is no reason for the output distributions to have a significantly smaller variance than the input distribution.

In this experiment, the input and output noises standard deviation were uniformly sampled in [0, 2.5] for each dimension. Every output dimension was treated independently by using a Gaussian Process prior on each of them. Fig. 7 shows the averaged mean square error over 25 randomly chosen training sets for different N values for each dimension.

5. Discussion

Learning from uncertain data is known to be hard. Under the Gaussian process assumption, a data set having a prior distribution has uncertain covariances and consequently a distribution over its covariance matrix. Considering the whole distribution on covariance matrices to compute a posterior Gaussian process is not feasible in general, even for the squared exponential covariance function with Gaussian input uncertainty. However, the former particular case allows computing the expectation of the covariance matrix distribution and makes the inference procedure no harder than with noise-free data. The experiments were conducted to determine whether using the expected covariance

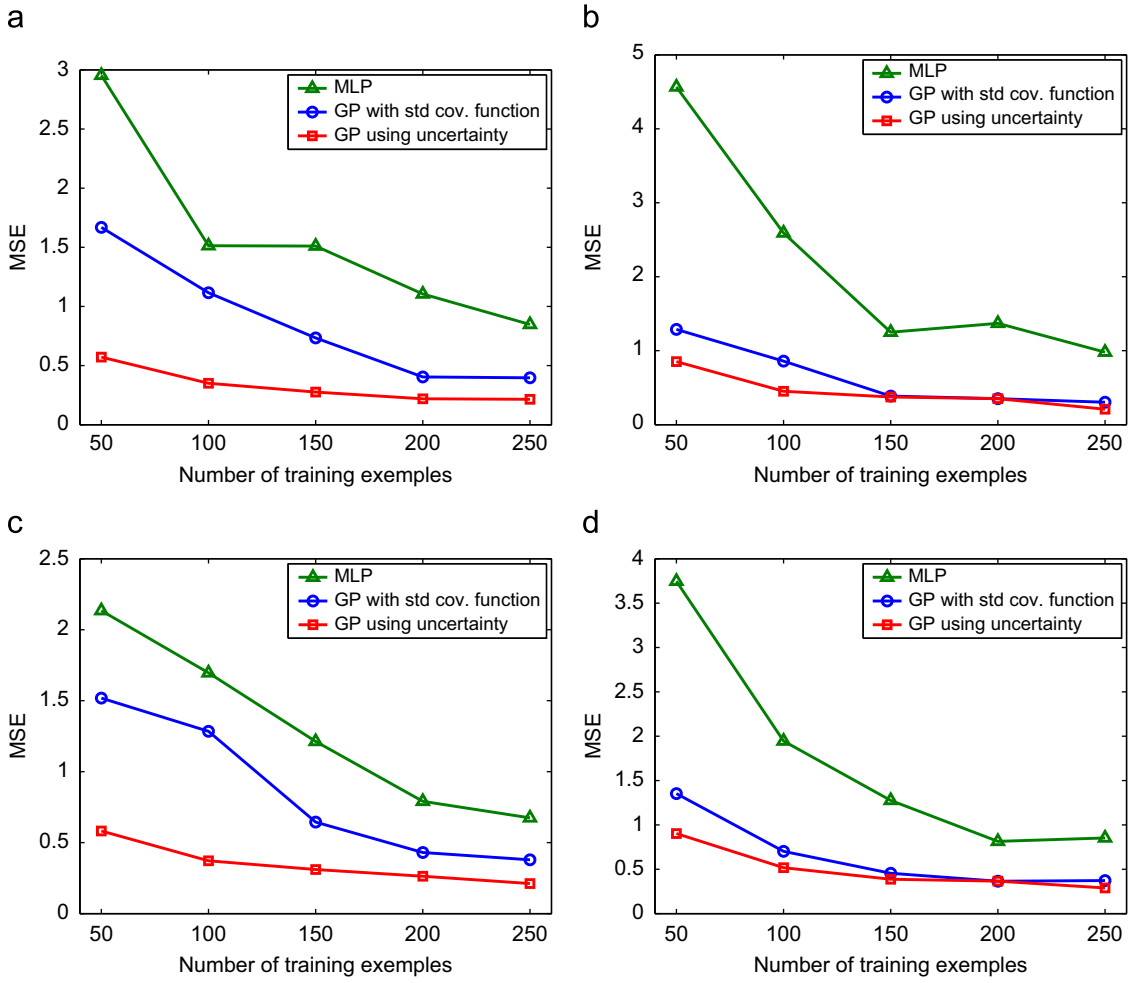

Fig. 7. Mean squared error results on the cart–pole problem. Squared red line, uncertain-GP; circled blue line, classic-GP; triangled green line, MLP. (a) Position of the cart; (b) velocity of the cart; (c) pole angle; (d) angular velocity of the pole. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

leads to performance improvement or not when dealing with input uncertainties.

The first results on the synthetic problem are presented in Fig. 3(a) and (b). It shows that using the expected covariance (uncertain-GP) leads to lower mean squared error than using the squared exponential with inputs mean (classic-GP). Fig. 2 shows the typical behaviour of each method. We observed that the classic-GP method often learns something closer to a convolution of the *sincsig* function rather than the function itself, which is the expected behaviour. On the other hand, the uncertain-GP method generally leads to better estimation of the function. Although it provides smaller mean squared errors, it also has the interesting property that the *sincsig* function completely lies inside its error bounds, meaning that most estimations are consistent with the true function and therefore, increasing the likelihood.

These experiments were done on data sets containing large input uncertainty up to 2.5 standard deviation. Since there are very few almost sure training examples to guide the inference, it makes the learning task very hard. In the case of such fuzzy training sets, using the expected covariance is a clear advantage over its standard counter-part.

A second set of experiments has been conducted where this time we measure the likelihood of the estimated function. To reinforce the relevance of using the expectation of the covariance matrix, the uncertain-GP method is now allowed to balance between the standard squared exponential covariance function and its expectation. The balancing procedure is part of the log marginal likelihood maximization. The results are shown in Figs. 5 and 6. Despite having a more expressive covariance function, the uncertain-GP method shows little improvement. We observed that the mean squared error is often caused by learning a biased version of the function and log-likelihood decreases as the bias increases. Fig. 8 shows the typical biased posterior learned by uncertain-GP for the *sinc* function and the convolution learned by the classic-GP. Although performance measures show similar curves for both methods, one can arguably prefer to obtain a biased function having the shape of the function rather than some estimations which capture less structure in the data.

In the context of learning the model of a partially observable Markov decision process [16], one has to identify first the dynamics, or state transition function, to be able to infer the state of the system. A reinforcement learning algorithm uses the inferred belief state to learn a value function by mapping the uncertain state to its expected reward. Learning the optimal policy can still be achieved if the learned dynamics are biased, since the reinforcement learning algorithm learns over this bias. Moreover, a far from truth model of the dynamics could lead the algorithm to learn a good policy if the estimated model's shape is close enough to the dynamics' true behaviour. Getting closer to this goal, experiments have been conducted on the identification of a dynamical system.

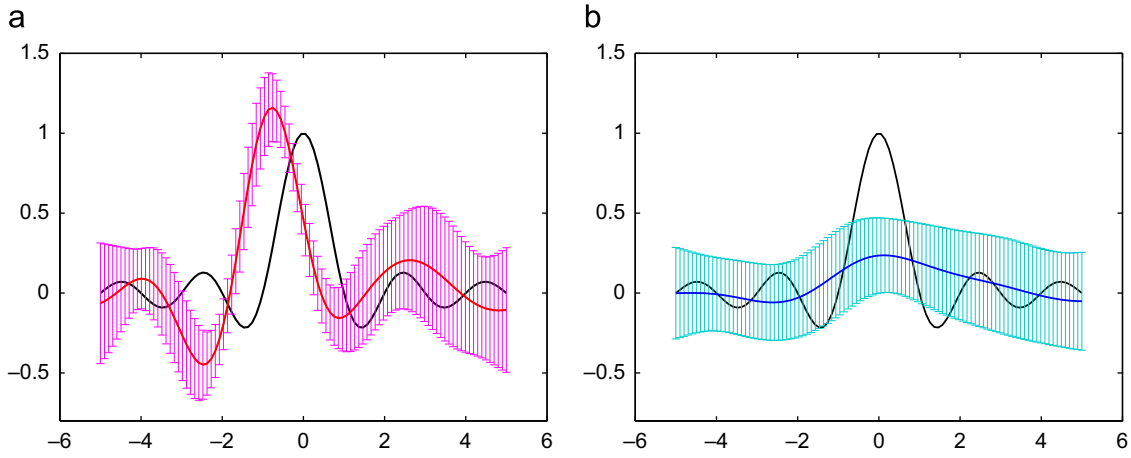

Fig. 8. Typical posterior distribution over functions when learning the *sinc* function with uncertain inputs and outputs. (a) Using the expected covariance matrix – uncertain-GP and (b) input noise is seen as extra output noise – classic-GP.

The cart–pole problem is well known in the reinforcement learning community. The last experiments aim to learn its dynamics from uncertain state transition examples. The results are exhibited in Fig. 7. The performances of the uncertain-GP and classic-GP methods are compared under the mean squared error measure. Again, using the expected covariance provided better estimates of the true function. However, much more data would have been required to correctly identify the dynamics, but the complexity of computing a posterior Gaussian process is $\mathcal{O}(n^3)$ and with hyperparameters optimization, it becomes the complexity of each optimization step. Moreover, computing the covariance matrix with the covariance function (17) takes more time, since it now involves determinant computations.

The optimization procedure is not only computationally cumbersome. The log marginal likelihood is riddled with local maxima in which vanilla gradient methods get driven quickly. The experiments were, therefore, carried out using stochastic optimization methods to move among the maxima and find the best one. An interesting avenue would be to look at natural gradient approaches which are well suited for this task [17].

6. Future work

Many previously proposed approaches for learning from uncertain data relied on the estimation of the hidden true inputs and outputs. Albeit not reported in this work, we conducted some experiments on computing a posterior Gaussian distribution over the training set. Since computing such posterior is not analytically tractable and might not be Gaussian, one has to look for approximations. By using Jensen's inequality, we obtained a lower bound approximation of the log marginal likelihood of the data and estimated the posterior distribution of the data with Gaussian distributions. The resulting quantity to maximize is

$$\log p(\mathbf{y}|X,\theta) - D_{KL}(q(X)\|p(X)) \tag{21}$$

where $p(X)$ is the prior distribution over the training set and $q(X)$ the posterior.

Eq. (21) corresponds to the common log marginal likelihood (9) with a penalty given by the Kullback–Leibler divergence of the estimated posterior over inputs with respect to its prior. The covariance matrix is computed with the expected squared exponential covariance function (17) with respect to the learned posterior over inputs. We obtained no significative results with this method, mostly because the optimization task is very hard, each input mean and variance becoming a variable to optimize. Further work is currently ongoing to address this issue.

7. Conclusion

To conclude, we investigated the use of the expected covariance matrix in Gaussian processes when learning from uncertain data. Results indicate that in many cases, depending on the amount of uncertainty, the proposed approach yields to do better inference. An interesting property is that even when the improvement on the mean squared error is negligible, the method will often improve the likelihood of the posterior over functions. We also stated that a bias may occur when using the expected covariance without denoising, but such biased models can still be useful for certain applications. Finally, the method finds its best performance when used on significantly uncertain data sets. In other cases, the approach provides performances similar to standard Gaussian process regression.

Appendix A. Expected linear covariance function

The expected linear covariance with respect to Gaussian input distributions $p(\mathbf{x}_i)=\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)$ and $p(\mathbf{x}_j)=\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j,\boldsymbol{\Sigma}_j)$ is given by:

$$K_{ij} = \iint C_{lin}(\mathbf{x}_i,\mathbf{x}_j)\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j,\boldsymbol{\Sigma}_j)\,d\mathbf{x}_i\,d\mathbf{x}_j = \mathbb{E}_i[\mathbb{E}_j[\mathbf{x}_i^\top\mathbf{x}_j+\sigma_b^2]]$$
$$= \mathbb{E}_i[\mathbf{x}_i^\top]\mathbb{E}_j[\mathbf{x}_j]+\sigma_b^2 = \mathbf{u}_i^\top\mathbf{u}_j+\sigma_b^2$$

Computing the expected variance is done with:

$$K_{ii} = \int C_{lin}(\mathbf{x}_i,\mathbf{x}_i)\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)\,d\mathbf{x}_i = \mathbb{E}_i[\mathbf{x}_i^\top\mathbf{x}_i+\sigma_b^2]$$
$$= \mathbb{E}_i[\mathbf{x}_i^\top\mathbf{x}_i]+\sigma_b^2 = \mathrm{Tr}(\boldsymbol{\Sigma}_i)+\mathbf{u}_i^\top\mathbf{u}_j+\sigma_b^2$$

By observing that the only difference between variance and covariance is the presence of a trace in the variance case, we can express the expected linear covariance function as

$$C_{\mathbb{E}lin}((\mathbf{u}_i,\boldsymbol{\Sigma}_i),(\mathbf{u}_j,\boldsymbol{\Sigma}_j)) = \mathbf{u}_i^\top\mathbf{u}_j+\sigma_b^2+\delta_{ij}\mathrm{Tr}(\boldsymbol{\Sigma}_i)$$

by the introduction of a Kronecker delta.

Appendix B. Expected quadratic covariance function

The expected quadratic covariance with respect to Gaussian input distributions $p(\mathbf{x}_i)=\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)$ and $p(\mathbf{x}_j)=\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j,\boldsymbol{\Sigma}_j)$ is

given by:

$$\begin{aligned}K_{ij} &= \iint C_{qua}(\mathbf{x}_i,\mathbf{x}_j)\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j,\boldsymbol{\Sigma}_j)\,d\mathbf{x}_i\,d\mathbf{x}_j\\ &= \mathbb{E}_i[\mathbb{E}_j[(\mathbf{x}_i^\top\mathbf{x}_j+\sigma_b^2)^2]] = \mathbb{E}_i[\mathbb{E}_j[\mathbf{x}_i^\top\mathbf{x}_j\mathbf{x}_i^\top\mathbf{x}_j+2\sigma_b^2\mathbf{x}_i^\top\mathbf{x}_j+\sigma_b^4]]\\ &= \mathbb{E}_i[\mathbb{E}_j[\mathbf{x}_i^\top\mathbf{x}_j\mathbf{x}_i^\top\mathbf{x}_j]]+2\sigma_b^2\mathbb{E}_i[\mathbb{E}_j[\mathbf{x}_i^\top\mathbf{x}_j]]+\sigma_b^4\\ &= \mathbb{E}_i[\mathbf{x}_i^\top\mathbb{E}_j[\mathbf{x}_j\mathbf{x}_j^\top]\mathbf{x}_i]+2\sigma_b^2\mathbf{u}_i^\top\mathbf{u}_j+\sigma_b^4\\ &= \mathbb{E}_i[\mathbf{x}_i^\top(\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top)\mathbf{x}_i]+2\sigma_b^2\mathbf{u}_i^\top\mathbf{u}_j+\sigma_b^4\\ &= \mathrm{Tr}\,([\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\boldsymbol{\Sigma}_i)+\mathbf{u}_i^\top[\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\mathbf{u}_i+2\sigma_b^2\mathbf{u}_i^\top\mathbf{u}_j+\sigma_b^4\end{aligned}$$

where we use some properties of the trace to obtain a more intuitive form for the first two terms:

$$\begin{aligned}&\mathrm{Tr}\,([\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\boldsymbol{\Sigma}_i)+\mathbf{u}_i^\top[\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\mathbf{u}_i\\ &= \mathrm{Tr}\,([\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\boldsymbol{\Sigma}_i)+\mathrm{Tr}\,(\mathbf{u}_i^\top[\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\mathbf{u}_i)\\ &= \mathrm{Tr}\,([\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\boldsymbol{\Sigma}_i)+\mathrm{Tr}\,([\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\mathbf{u}_i\mathbf{u}_i^\top)\\ &= \mathrm{Tr}\,([\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\boldsymbol{\Sigma}_i+[\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top]\mathbf{u}_i\mathbf{u}_i^\top)\\ &= \mathrm{Tr}\,([\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top][\boldsymbol{\Sigma}_i+\mathbf{u}_i\mathbf{u}_i^\top])\end{aligned}$$

By substituting back in the expected covariance equation, we can have

$$K_{ij} = \mathrm{Tr}\,([\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top][\boldsymbol{\Sigma}_i+\mathbf{u}_i\mathbf{u}_i^\top])+2\sigma_b^2\mathbf{u}_i^\top\mathbf{u}_j+\sigma_b^4$$

Computing the expected variance is done with:

$$\begin{aligned}K_{ii} &= \int C_{qua}(\mathbf{x}_i,\mathbf{x}_i)\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)\,d\mathbf{x}_i = \mathbb{E}_i[\mathbf{x}_i^\top\mathbf{x}_i\mathbf{x}_i^\top\mathbf{x}_i+2\sigma_b^2\mathbf{x}_i^\top\mathbf{x}_i+\sigma_b^4]\\ &= \mathbb{E}_i[\mathbf{x}_i^\top\mathbf{x}_i\mathbf{x}_i^\top\mathbf{x}_i]+2\sigma_b^2\mathbb{E}_i[\mathbf{x}_i^\top\mathbf{x}_i]+\sigma_b^4\\ &= \mathbb{E}_i[\mathbf{x}_i^\top\mathbf{x}_i\mathbf{x}_i^\top\mathbf{x}_i]+2\sigma_b^2\mathrm{Tr}\,(\boldsymbol{\Sigma}_i+\mathbf{u}_i\mathbf{u}_i^\top)+\sigma_b^4\\ &= 2\mathrm{Tr}\,(\boldsymbol{\Sigma}_i^2)+4\mathbf{u}_i^\top\boldsymbol{\Sigma}_i\mathbf{u}_i+\mathrm{Tr}\,([\boldsymbol{\Sigma}_i+\mathbf{u}_i\mathbf{u}_i^\top])^2\\ &\quad +2\sigma_b^2\mathrm{Tr}\,(\boldsymbol{\Sigma}_i+\mathbf{u}_i\mathbf{u}_i^\top)+\sigma_b^4\end{aligned}$$

By observing the differences between variance and covariance, we can express the expected quadratic covariance function as:

$$\begin{aligned}C_{\mathbb{E}qua}((\mathbf{u}_i,\boldsymbol{\Sigma}_i),(\mathbf{u}_j,\boldsymbol{\Sigma}_j)) &= \mathrm{Tr}\,([\boldsymbol{\Sigma}_i+\mathbf{u}_i\mathbf{u}_i^\top][\boldsymbol{\Sigma}_j+\mathbf{u}_j\mathbf{u}_j^\top])(1+\delta_{ij})\\ &\quad +2\sigma_b^2\mathbf{u}_i^\top\mathbf{u}_j+\sigma_b^4+\delta_{ij}[\mathrm{Tr}\,(\boldsymbol{\Sigma}_i+\mathbf{u}_i\mathbf{u}_i^\top)^2+2\sigma_b^2\mathrm{Tr}(\boldsymbol{\Sigma}_i)-2(\mathbf{u}_i^\top\mathbf{u}_i)^2]\end{aligned}$$

by the introduction of a Kronecker delta.

Appendix C. Expected squared exponential covariance

The expected squared exponential covariance with respect to Gaussian input distributions $p(\mathbf{x}_i)=\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)$ and $p(\mathbf{x}_j)=\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j,\boldsymbol{\Sigma}_j)$ is given by:

$$K_{ij} = \iint C_{se}(\mathbf{x}_i,\mathbf{x}_j)\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j,\boldsymbol{\Sigma}_j)\,d\mathbf{x}_i\,d\mathbf{x}_j$$

Using the Gaussian form of the squared exponential covariance function, $C_{se}(\mathbf{x}_i,\mathbf{x}_j)$ can be rewritten as the normalized Gaussian distribution $c\mathcal{N}_{\mathbf{x}_i}(\mathbf{x}_j,\mathbf{W})$, where $c=\sigma_f^2(2\pi)^{D/2}|\mathbf{W}|^{1/2}$ is its normalization constant:

$$K_{ij} = c\iint\mathcal{N}_{\mathbf{x}_i}(\mathbf{x}_j,\mathbf{W})\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j,\boldsymbol{\Sigma}_j)\,d\mathbf{x}_i\,d\mathbf{x}_j$$

Using the product of Gaussians' results in another Gaussian, which is no longer normalized:

$$K_{ij} = c\iint Z^{-1}\mathcal{N}_{\mathbf{x}_i}(\mathbf{a},\mathbf{B})\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j,\boldsymbol{\Sigma}_j)\,d\mathbf{x}_i\,d\mathbf{x}_j$$

where the resulting Gaussian with mean vector $\mathbf{a}$ and covariance matrix $\mathbf{B}$ will be eliminated by integration. It turns out that the factor has a Gaussian form such that $Z^{-1}=\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_i,\mathbf{W}+\boldsymbol{\Sigma}_i)$. Therefore, by integrating over $\mathbf{x}_i$, we obtain:

$$K_{ij} = c\int\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_i,\mathbf{W}+\boldsymbol{\Sigma}_i)\mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j,\boldsymbol{\Sigma}_j)\,d\mathbf{x}_j$$

For the second variable, we use the same reasoning. Thus, by applying the product of Gaussians again and this time integrating over $\mathbf{x}_j$, we obtain:

$$\begin{aligned}K_{ij} &= c\mathcal{N}_{\mathbf{u}_i}(\mathbf{u}_j,\mathbf{W}+\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j) = \sigma_f^2(2\pi)^{D/2}|\mathbf{W}|^{1/2}\mathcal{N}_{\mathbf{u}_i}(\mathbf{u}_j,\mathbf{W}+\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)\\ &= \frac{\sigma_f^2\exp((\mathbf{u}_i-\mathbf{u}_j)^\top(W+\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)^{-1}(\mathbf{u}_i-\mathbf{u}_j))}{|I+W^{-1}(\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)|^{1/2}}\end{aligned}$$

The expected squared exponential variance is easier to compute. First, we have:

$$K_{ii} = \int C_{se}(\mathbf{x}_i,\mathbf{x}_i)\mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i,\boldsymbol{\Sigma}_i)\,d\mathbf{x}_i$$

By the definition of a stationary covariance function, $C(\mathbf{x},\mathbf{x})$ is constant, which leads directly to

$$K_{ii} = C_{se}(\mathbf{x}_i,\mathbf{x}_i)$$

By the definition of the squared exponential covariance function, $C_{se}(\mathbf{x}_i,\mathbf{x}_i)=\sigma_f^2$.

Finally, combining the variance and covariance equations in a single function can be done by introducing an inverted Kronecker delta:

$$C_{\mathbb{E}se}((\mathbf{u}_i,\boldsymbol{\Sigma}_i),(\mathbf{u}_j,\boldsymbol{\Sigma}_j)) = \frac{\sigma_f^2\exp\left(-\frac{1}{2}(\mathbf{u}_i-\mathbf{u}_j)^\top(W+\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)^{-1}(\mathbf{u}_i-\mathbf{u}_j)\right)}{|I+W^{-1}(\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j)(1-\delta_{ij})|^{1/2}}$$

References

[1] H.G. Golub, F.C.V. Loan, An analysis of the total least squares problem, SIAM Journal on Numerical Analysis (17) (1980) 883–893.
[2] R.J. Carroll, D. Ruppert, L.A. Stefanski, Measurement Error in Nonlinear Models, Chapman and Hall/CRC, 1995.
[3] I. Markovsky, S.V. Huffel, Overview of total least-squares methods, Signal Processing 87 (10) (2007) 2283–2302.
[4] V. Tresp, S. Ahmad, R. Neuneier, Training neural networks with deficient data, in: Advances in Neural Information Processing Systems (NIPS'94), Morgan Kaufmann, 1994, pp. 128–135.
[5] W.A. Wright, G. Ramage, D. Cornford, I.T. Nabney, Neural network modelling with input uncertainty: theory and application, Journal of VLSI Signal Processing Systems 26 (1/2) (2000) 169–188.
[6] J.-A. Ting, A. D'Souza, S. Schaal, Bayesian regression with input noise for high dimensional data, in: Proceedings of the International Conference on Machine Learning (ICML'06), ACM, 2006, pp. 937–944.
[7] J. Bi, T. Zhang, Support vector classification with input data uncertainty, Advances in Neural Information Processing Systems (NIPS'04), vol. 17, MIT Press, 2004, pp. 161–168.
[8] A. Girard, R. Murray-Smith, Learning a Gaussian process model with uncertain inputs, Technical Report TR-2003-144, University of Glasgow, Glasgow, UK, 2003.
[9] J. Quiñonero-Candela, S.T. Roweis, Data imputation and robust training with Gaussian processes, 2003.
[10] A. Girard, Approximate methods for propagation of uncertainty with Gaussian process models, Ph.D. Thesis, 2004.
[11] C.K.I. Williams, Computation with Infinite Neural Networks, Neural Computation 10 (5) (1998) 1203–1216.
[12] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006.
[13] D.J.C. Mackay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003.
[14] J. Moré, The Levenberg–Marquardt algorithm: implementation and theory, in: G.A. Watson (Ed.), Numerical Analysis, Lecture Notes in Mathematics, vol. 630, Springer, Berlin, Heidelberg, 1978, pp. 105–116 (Chapter 10).
[15] R. Florian, Correct equations for the dynamics of the cart–pole system, Center for Cognitive and Neural Studies (Coneural), Romania, 2007.
[16] P. Dallaire, C. Besse, S. Ross, B. Chaib-draa, Bayesian reinforcement learning in continuous POMDPs with Gaussian processes, in: Proceedings of IEEE/RSJ

International Conference on Intelligent Robots and Systems (IROS'09), IEEE Press, 2009, pp. 2604–2609.
[17] N.L. Roux, P.-A. Manzagol, Y. Bengio, Topmoumoute online natural gradient algorithms, in: Advances in Neural Information Processing Systems (NIPS'07), MIT Press, 2008, pp. 849–856.

Brahim Chaib-draa received a Diploma in Computer Engineering from the École Supérieure d'Électricité (SUPELEC) de Paris, Paris, France, in 1978 and a Ph.D. degree in Computer Science from the Université du Hainaut-Cambrésis, Valenciennes, France, in 1990. In 1990, he joined the Department of Computer Science and Software Engineering at Laval University, Quebec, QC, Canada, where he is a Professor and Group Leader of the Decision for Agents and Multi-Agent Systems (DAMAS) Group. His research interests include agent and multiagent technologies, natural language for interaction, formal systems for agents and multiagent systems, distributed practical reasoning, and real-time and distributed systems. He is the author of several technical publications in these areas. He is on the Editorial Boards of IEEE Transactions on SMC, Computational Intelligence and The International Journal of Grids and Multiagent Systems. Dr. Chaib-draa is a member of ACM and AAAI and senior member of the IEEE Computer Society.

Camille Besse was born is 1981. He received the B.Sc. degree in Intelligent Systems in 2002 and his M.Sc. degree in Artificial Intelligence in 2005 from Paul Sabatier University, Toulouse, France. He is currently a Ph.D. student under the supervision of Prof. Brahim Chaib-draa and member of the DAMAS laboratory research group. His current research interests include partially observable Markov decision processes for single and multiple agents, for planning and reinforcement learning.

Patrick Dallaire was born is 1982. He received the B.Sc. degree in Computer Science in 2008 from Laval University, Canada. He is currently a M.Sc. student under the supervision of Prof. Brahim Chaib-draa and member of the DAMAS laboratory research group. His current research interests include Gaussian processes, partially observable Markov decision processes, Bayesian learning and reinforcement learning.