Simulation–Extrapolation Gaussian Processes for Input Noise Modeling

Botond Bócsi, Hunor Jakab, and Lehel Csató
Faculty of Mathematics and Informatics, Babeş-Bolyai University, Kogalniceanu 1, 400084 Cluj-Napoca, Romania
{BBOTI,JAKABH,CSATOL}@CS.UBBCLUJ.RO

***Abstract*—Input noise is common in situations when data either is coming from unreliable sensors or previous outputs are used as current inputs. Nevertheless, most regression algorithms do not model input noise, inducing thus bias in the regression. We present a method that corrects this bias by *repeated* regression estimations. In *simulation extrapolation* we perturb the inputs with additional input noise and by observing the *effect* of this addition on the result, we *estimate* what would the prediction be without the input noise. We extend the examination to a non-parametric probabilistic regression, inference using Gaussian processes. We conducted experiments on both synthetic data and in robotics, *i.e.*, learning the transition dynamics of a dynamical system; showing significant improvements in the accuracy of the prediction.**

I. Introduction

Automatically extracting information from collected data is a common practice and much research is focused on improving the information extraction algorithms. For regression this information is a function: we search for the best mapping of inputs to values in an output space. For inference to be performed, first a candidate function class is chosen. Secondly, we have to model the data observation process, *i.e.*, we have to consider possible perturbations to the observed values, both at the input and at the output levels, *i.e.* noise. According to Jaynes and Bretthorst [8], inference from observed data without noise modeling is *ill-posed*, thus, we highlight two cases when noise is required: (1) we have no knowledge about the "true" data generation mechanism, meaning that any model will only partially reconstruct the data, the un-modeled part being added to the *output noise*; (2) inputs are inaccurate, coming from unreliable sensors or from a prediction at a different level, where noise is inevitable, thus, we have *input noise*. Conventional modeling only assumes output noise and input noise is neglected; mainly because it leads to analytical intractability or – for linear models and additive noises – it cannot be separated from output noise.

In this paper, we analyze regression in the presence of the input noise and we choose as regressors the non-parametric Gaussian processes (GPs). Assuming noisy inputs, we use the *simulation extrapolation* method [3, 2] to improve the approximation given by the GPs (referred to as SIMEXGP). Exploiting the possibility of probabilistic inference, the SIMEXGP method inherits the main advantages of both GP and the simulation extrapolation methods.

Owing to their non-parametric nature[1] GPs – or kernel methods in general – are popular due to their "automatic" adaptation to varying data set sizes and varying problem complexities: the same algorithm is likely to produce meaningful results in a wider variety of settings. The added benefit of GPs [14] comes from their probabilistic nature: they not only provide point-wise estimates but also "confidence" in the form of posterior variances – for more details see Section II.

The contribution of the paper is threefold: (1) we extend simulation extrapolation to arbitrary function approximation, *e.g.*, non-parametric function approximation; (2) we apply simulation extrapolation to probabilistic function approximation models, such as GPs. The resulting approximation is also a probability distribution; (3) we show that, with simplifying assumptions, the method reduces to the Hessian–corrected input noise modeling [1].

The paper is organized as follows: Section II introduces the input noise GP framework with the related work in Section II-A. Section III presents our method in details. Experiments with artificial data and from a robotic setup are shown in Section IV; conclusions and future research is discussed in Section V.

II. Model Definition and Related Work

In what follows we assume a data–set $\boldsymbol{D} = \{(w_i, y_i)\}_{i=1}^N$ where both the inputs w_i and the labels y_i are corrupted with noise:

$$y_i = \tilde{f}(x_i) + \eta_i, \qquad w_i = x_i + \epsilon_i, \tag{1}$$

where $y_i \in \mathbb{R}$ is the observed noisy label – or output –, $\tilde{f}(x_i) \in \mathbb{R}$ is the true label given by an unknown function at an input location $x_i \in \mathbb{R}^d$ – d is the dimension of the input space. The locations are also unobserved, we have their noisy version only: $w_i \in \mathbb{R}^d$. We assume in what follows $\eta_i \sim \mathcal{N}(0, \Phi)$ and $\epsilon_i \sim \mathcal{N}(0, \Sigma)$ are the output and the input noise respectively.

The goal is to find a function f that best *predicts* the labels at previously unseen – test – inputs w^*.[2] In the rest of the paper we use bold to denote vectors, *e.g.* $\boldsymbol{w} = [w_i]_{i=1}^N \in \mathbb{R}^{N\times d}$ and use bold capitals to denote matrices, *e.g.* $\boldsymbol{I}_N \in \mathbb{R}^{N\times N}$ is the identity matrix of size N.

[1]The word non-parametric means that the *number of parameters* is not fixed: this number varies with the size of the data-set.

[2]We use w^* as *test* input in *prediction*. Whether or not test inputs are noisy is application dependent and – for notational simplicity – we considered noisy inputs.

For regression we use GPs that are non-parametric Bayesian methods [14]. Given the training set, the prediction of a GP for a test input w^* is a Gaussian–distributed random variable with mean μ_* and variance σ_*^2 where

$$\begin{aligned}\mu(w^*) &= \boldsymbol{k}_*^\top (\boldsymbol{K} + \boldsymbol{I}_N \Phi)^{-1} \boldsymbol{y} \\ \sigma^2(w^*) &= k(w^*, w^*) - \boldsymbol{k}_*^\top (\boldsymbol{K} + \boldsymbol{I}_N \Phi)^{-1} \boldsymbol{k}_*, \end{aligned} \quad (2)$$

with $\boldsymbol{k}_* \in \mathbb{R}^{N\times 1}$ and $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ have elements $\boldsymbol{k}_*^i = k(w^*, w_i)$ and $\boldsymbol{K}^{ij} = k(w_i, w_j)$ respectively; Φ is the variance of the label noise and $\boldsymbol{I}_N$ is the identity matrix of size N. The main entity in GP approximation is the positive definite *kernel function* $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ – a generalization of the positive definite covariance matrix [16, 14]. Kernels define our prior knowledge about the regression in general, *e.g.*, smoothness or periodicity. Observe that once the kernel is chosen, we compute the GP solution based on Equation (2). To increase flexibility, most kernels have tunable parameters $\boldsymbol{\theta}$, called hyper-parameters. The hyper-parameters can be tuned efficiently by maximizing the log-likelihood – or evidence – of the training data with respect to $\boldsymbol{\theta}$:

$$\begin{aligned}\log p(\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{\theta}) &= -\frac{1}{2}\boldsymbol{y}^\top (\boldsymbol{K} + \boldsymbol{I}\Phi)^{-1}\boldsymbol{y} - \\ & -\frac{1}{2}\log|\boldsymbol{K} + \boldsymbol{I}\Phi| - \frac{N}{2}\log 2\pi.\end{aligned}$$

Second order gradient ascent search methods lead to good hyper-parameter values [11, 14], this procedure is used in our experiments.

A. Related Work

A first approach to tackling input noise within GP modeling is to use Taylor approximation of the GP posterior. Based on the second order expansion, Girard and Murray-Smith [6] approximated posterior moments and for linear and squared exponential kernels they provided analytical expressions; this method has been extended to other kernel functions by Dallaire et al. [4]. Using a known input noise Girard and Murray-Smith [5], proposed different approximations such as approximate moments, exact moments with linear and squared exponential kernels, and they also used a Monte Carlo integration of the noise. A first order Taylor approximation to the objective function around the noisy inputs led to a correction in predicted value proportional to the gradient of the GP, suggested by McHutchon and Rasmussen [12]. Their method involves solving a non-linear differential equation, thus, only an iterative approximation is proposed in their work.

A second approach is sampling-based: in Goldberg et al. [7] a second GP was used to model the input noise process together with a Monte Carlo integration for the posterior. Due to the limitations of Monte Carlo integration, this approach is difficult to be used in high-dimensional setup. Similar in setup, Kersting et al. [9] assumed varying – heteroscedastic – input noise and used a pair of GPs: one for the output values and a second for input noise variance. To decouple the two processes and to learn the hyper-parameters of the two GPs, an EM–style algorithm was proposed. Lázaro-Gredilla and Titsias [10], in a similar heteroscedastic setup, used variational approximations for the same scope. Last we mention the work of Quiñonero-Candela and Roweis [13] that suggestsed a precise estimation of the input locations: the most probable input locations were being computed also with an EM–style algorithm. This algorithm was prone to overfitting.

III. Simulation Extrapolation

Simulation extrapolation (SimEx) was introduced by Cook and Stefanski [3] in statistics for parameter estimation of regression models with input noise (more precisely Berkson errors [2]). Although the description was only for parametric models, here we present simulation extrapolation to general regression models so that it can accommodate GPs. In Section III-C we extend the algorithm to probabilistic models, leading to a full probabilistic simulation extrapolation.

In what follows we provide an overview of the method. Since SimEx is independent of the output noise, *in this section* we omit it and focus only to the noise in the input. Based on our model from Equation (1), the observed label can be written as $y = f(w) = f(x+\epsilon_x)$, In SimEx we introduce a parameter λ and define the mapping φ_λ taking λ to a function evaluated in w as follows:[3]

$$\begin{aligned}\varphi_\lambda(w) &= \int f(w + \lambda^{1/2}\epsilon_\lambda) dP(\epsilon_\lambda) \\ &= \int f(x + \epsilon_x + \lambda^{1/2}\epsilon_\lambda) dP(\epsilon_\lambda).\end{aligned} \quad (3)$$

where ϵ_x and ϵ_λ are two i.i.d. noise processes, therefore $\varphi_\lambda(w)$ is a regression function obtained from data with "noisier" inputs, and since we know ϵ_λ, we can integrate it out. In what follows we *replace the pair of random variables* with an equivalent random variable $\epsilon' = \epsilon_x + \lambda^{1/2}\epsilon_\lambda$ since the two are i.i.d. and zero-mean, the variance of the new variable is $var(\epsilon') = (1+\lambda)\, var(\epsilon_x)$, *i.e.*

$$\varphi_\lambda(w) \sim f(x + \sqrt{1+\lambda}\ \epsilon_x).$$

We see that for $\lambda = 0$, we have the original function $\varphi_0(w) = f(w)$. In *simulation extrapolation* the intuition is that if we could compute $\varphi_{-1}(x)$, this function would provide the "uncorrupted" labels at x. It is obvious that this intuition is incorrect – you cannot use negative values for λ – but the key observation is that we can evaluate φ_λ for *any positive* λ. Then, we use extrapolation to obtain the value at $\lambda = -1$.

Informally, we can state that in SimEx we first add additional noise to the inputs – with controlled variance – and observe the effect of the addition on the regression function. In a subsequent step – having φ_λ for a couple of positive λ-s – we extrapolate point-wise to $\varphi_{-1}(x)$. For example, we might use four functions to obtain the SimEx approximation, also shown in Figure 1:

$$\varphi_3, \varphi_2, \varphi_1, \varphi_0 \quad \to \quad \varphi_{-1}. \quad (4)$$

For the extrapolation we need a model for the extrapolating φ_λ as a function of λ. In the original, parametric, SimEx

[3]The mapping $\varphi_\lambda(w)$ is a function of both λ and w.

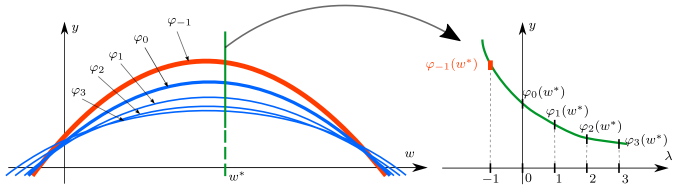

Fig. 1. Illustration of simulation extrapolation. (left) φ_λ for $\lambda = \{0, 1, 2, 3\}$ drawn with thin blue lines; φ_0 is slightly thicker; φ_{-1} in red, the point-wise extrapolation. (right) the individual extrapolation $\varphi_\lambda(w^*)$ – as a function of λ – for a test point w^*.

algorithm, the extrapolation is done in the parameter space [2]. Since we aim for a non-parametric solution, we extrapolate for every test point w^* and compute $\varphi_\lambda(w^*)$ for a set of λ's, followed by the computation of $\varphi_{-1}(w^*)$. The extrapolating function $\varphi(\lambda, w^*)$ is important and next we present different model choices. We choose φ_λ such that it is amenable for probabilistic approximations, such as GPs.

A. *Linear Extrapolation and Hessian–Corrected Models*

A first choice for φ is a function linear in λ, *i.e.*, $\varphi_\lambda^{(L)}(w) = \gamma_1(w) + \lambda\gamma_2(w)$. In this case we show that SIMEX is equivalent to Hessian–corrected input noise model [1] obtained from a second order Taylor expansion of f. Using the definition of $\varphi_\lambda(w)$ from Equation (3) and expanding $f(\cdot)$ to the second order around the test input w^* we get:

$$\begin{aligned}
\varphi_\lambda(w^*) &= \int f(w^* + \lambda^{1/2}\epsilon_\lambda) dP(\epsilon_\lambda) \\
&\approx \int \Big(f(w^*) + \lambda^{1/2} \epsilon_\lambda^\top J_f(w^*) + \\
&\quad + \frac{1}{2}\lambda\epsilon_\lambda \, H_f(w^*) \, \epsilon_\lambda^\top \Big) dP(\epsilon_\lambda) \\
&\approx f(w^*) + \frac{\lambda}{2} \operatorname{tr}(\Sigma H_f(w^*)),
\end{aligned}$$

where $J_f(w^*)$ and $H_f(w^*)$ are the Jacobian and the Hessian of f at w^*; Σ is the input noise covariance; tr is the trace operator, and we used that ϵ_λ has zero mean. Therefore, using the Taylor expansion, we have indeed that φ_λ is linear with $\gamma_1(w) = f(w)$ and $\gamma_2(w) = \operatorname{tr}(\Sigma H_f(w))$, leading to:

$$\varphi_{-1}^{(L)}(w^*) = f(w^*) - \frac{1}{2} \operatorname{tr}(\Sigma H_f(w^*)).$$

which is the expression of the Hessian–corrected noise modeling proposed by Bócsi and Csató [1]. We think this is an interesting connection between an approximation-based approach and a second one that uses an averaging with added noise.

B. *Polynomial Extrapolation*

In their setup, Carroll et al. [2] argued that linear functions are too restrictive for the extrapolation modeling and proposed two alternatives, the rational to the first order (φ_λ^R) and the quadratic (φ_λ^Q) functions, *i.e.*,

$$\begin{aligned}
\varphi_\lambda^{(R)}(w) &= \gamma_1(w) + \frac{\gamma_2(w)}{\gamma_3(w) + \lambda} \\
\varphi_\lambda^{(Q)}(w) &= \gamma_1(w) + \gamma_2(w)\lambda + \gamma_3(w)\lambda^2,
\end{aligned}$$

where the functions $\gamma_1(w)$, $\gamma_2(w)$, and $\gamma_3(w)$ – similarly to the linear case – are the parameters to the extrapolation. The discussion in [2] and experimental evaluations point out two issues: (1) the parameter estimation of φ_λ^R becomes unstable when $\gamma_2(w)$ is close to zero, and (2) higher then second order polynomials result in inaccurate extrapolation. Thus, we further limit ourselves to quadratic extrapolation only.

The parameters $\{\gamma_1(w), \gamma_2(w), \gamma_3(w)\}$ can be estimated similarly to the previous section, *i.e.*, get three values of φ_λ^Q and obtain the analytical expressions of the parameters. Another possibility is to get more than three values of φ_λ^Q and obtain the parameters by minimizing the least square error of the extrapolation model. The second approach proved to be a more robust solution, and we have chosen this one in our experiments.

C. *Probabilistic Extrapolation*

We assume next that the regression function $f(\cdot)$ – from Equation (3) – is a GP and the extrapolation function φ_λ is also a GP (φ_λ^{GP}) – here as a function of λ. Applying this assumption to the SIMEX framework from Equation (3), an integration of a GP posterior is required, and it leads to

$$\begin{aligned}
\varphi_\lambda^{GP}(w^*) &\sim \int \mathcal{N}(\mu_\lambda^\epsilon(w^*), \sigma_\lambda^\epsilon(w^*)) \, dP(\epsilon) \qquad (5) \\
\mu_\lambda^\epsilon(w^*) &= \boldsymbol{k}_\lambda^\top (\boldsymbol{K}_\lambda + \boldsymbol{I}\Phi)^{-1} \boldsymbol{y} \\
\sigma_\lambda^\epsilon(w^*) &= k_\lambda(w^*, w^*) - \boldsymbol{k}_\lambda^\top \, (\boldsymbol{K}_\lambda + \boldsymbol{I}\Phi)^{-1} \boldsymbol{k}_\lambda,
\end{aligned}$$

where the vector $\boldsymbol{k}_\lambda \in \mathbb{R}^{N\times 1}$ is $\boldsymbol{k}_\lambda^i = k(w^*, w_i + \lambda\epsilon_i)$, and the matrix $\boldsymbol{K}_\lambda \in \mathbb{R}^{N\times N}$ is $\boldsymbol{K}_\lambda^{ij} = k(w_i + \lambda\epsilon_i, w_j + \lambda\epsilon_j)$, with k being a positive definite kernel. The integration from Equation (5) is analytically intractable. However, the *point-wise* average of Gaussians has an analytical solution, *i.e.*, the GP is marginalized relative to input.[4]

Using the point-wise averaging, we compute φ_{-1}^{GP} using a second GP inference for every test point w^*, this time along the λ axis: we collect all lambda values into a vector $\boldsymbol{\lambda} \in \mathbb{R}^L$, with L the length of the vector. Then, we use the values of the Gaussian means and variances as data to this GP from Equation (5) and the *probabilistic* prediction is $\varphi_{-1}^{GP}(w^*) \sim \mathcal{N}(\mu_{-1}(w^*), \sigma_{-1}^2(w^*))$ with

$$\begin{aligned}
\mu_{-1}(w^*) &= \boldsymbol{h}^\top (\boldsymbol{H} + \boldsymbol{\sigma^2})^{-1} \boldsymbol{t} \\
\sigma_{-1}^2(w^*) &= h(-1, -1) - \boldsymbol{h}^\top \, (\boldsymbol{H} + \boldsymbol{\sigma^2})^{-1} \boldsymbol{h},
\end{aligned}$$

where the vector $\boldsymbol{h} \in \mathbb{R}^{L\times 1}$ is $\boldsymbol{h}^i = h(-1, \boldsymbol{\lambda}_i)$, the matrix $\boldsymbol{H} \in \mathbb{R}^{L\times L}$ is $\boldsymbol{H}^{ij} = h(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j)$ and h is a positive definite

[4]Since the operations are point-wise, the consistency of the – Gaussian – process might be violated.

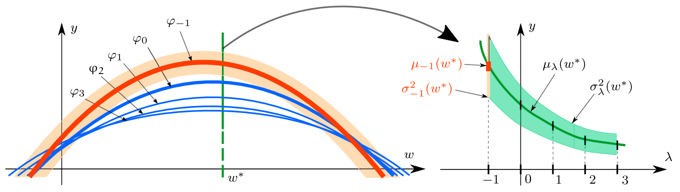

Fig. 2. Illustration of probabilistic simulation extrapolation. (left) φ_λ for different λ values drawn with thin blue lines – variances omitted for clarity; φ_{-1} in thick red is the point-wise extrapolated posterior mean and the posterior variance. (right) the individual extrapolation $\varphi_\lambda(w^*)$ – as a function of λ – for a test point w^*.

kernel function. This second kernel h – different from k – is used for the SIMEX extrapolation only, while k is used in the function space.

The data for this GP inference are the vector of labels $\boldsymbol{t} \in \mathbb{R}^{L\times 1}$ as $\boldsymbol{t}_\lambda = \mu_\lambda(w^*)$ from Equation (5). The variances corresponding the these values are put into a diagonal matrix $\boldsymbol{\sigma^2} \in \mathbb{R}^{L\times L}$ with $\boldsymbol{\sigma^2}_{\lambda\lambda} = {\sigma_\lambda}^2(w^*)$ playing the role of the observation – label – noise at this level. For an illustration of probabilistic simulation extrapolation see Figure 2.

We conclude with two observations: (1) $\boldsymbol{h}$ and $\boldsymbol{H}$ are *constants*, as they do not depend on the data, but only on $\boldsymbol{\lambda}$ and the extrapolation kernel h; (2) $\varphi^{GP}_{-1}(w)$ is not a GP, but is is still a Gaussian for every test point.

IV. Experimental Evaluation

This section contains the validation of SIMEXGP on both synthetic data and in a robotic setup for learning the state-transition dynamics of non-linear systems.

Regarding the implementation a crucial question is how to perform the integration from Equation (5). Since the integration is intractable, we approximated it with Monte Carlo integration. Experiments show that a relatively small number of evaluations (we used 30–40 samples in our experiments) are enough to be averaged out even in high dimensional domains. Another implementation issue is the set of $\boldsymbol{\lambda}$ values from Equation (4). If not mentioned otherwise, we set $\boldsymbol{\lambda} = [0, {}^1\!/\!_2, 1, {}^3\!/\!_2, 2, {}^5\!/\!_2, 3]$. In all the experiments we assumed that the variance of the input noise is known a-priori.

Another implementational concern is computational efficiency: SIMEXGP requires significantly more computations than a standard GP, mainly caused by the Monte Carlo integration and the extrapolation GPs. The positive side is that most computation can be done parallel that is highly supported by modern computer architectures. Experiments show that with proper implementation SIMEXGP is slower than other methods with *only* one order of magnitude, therefore we consider it is worth experimenting with it. We next present an illustrative example, that gives a visual intuition about the improvement induced by our method.

A. Illustrative Example

This experiment aims to give an intuition about how SIMEXGP works and shows the improvement induced by the method. We generated 1000 training points from the interval $[-12, 12]$ and added a Gaussian input noise with variance $\Sigma = 0.8$. The training labels were generated using the sinc(x/π) function. A Gaussian input noise with variance $\Phi = 0.1$ was also added.

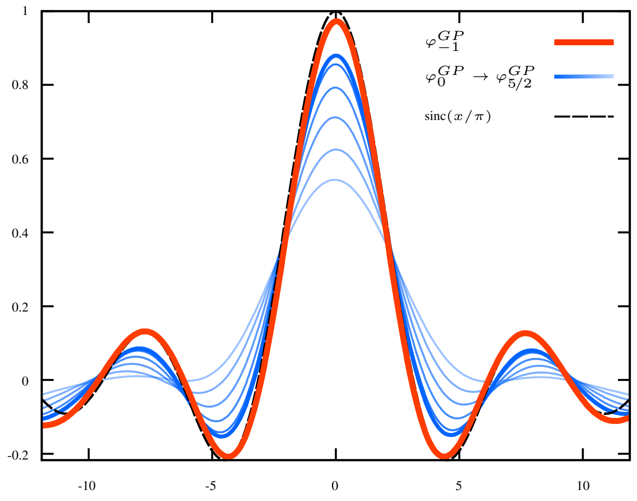

Fig. 3. Results for the sinc(x/π) function with input noise variance $\Sigma = 0.8$. The black dashed curve shows the true sinc$(\cdot)$ function. The blue curves $\varphi^{GP}_0 \to \varphi^{GP}_{5/2}$ show the posterior GP means when additional noise was added to the inputs. Note that φ^{GP}_0 corresponds to the GP when only the original input noise was present. The red curve φ^{GP}_{-1} shows the extrapolated GP.

Results are shown on Figure 3. The posterior variance is not presented since its value was very small caused by the large number of training points. This phenomena is an inherited property of GPs. Figure 3 contains the GPs with the artificially added noise, *i.e.*, $\varphi^{GP}_0 \to \varphi^{GP}_{5/2}$, where φ^{GP}_0 is the GP when only the original input noise was present. The red curve φ^{GP}_{-1} is the extrapolation of the GP means. The final solution is very close to the sinc$(\cdot)$ function that we wanted to approximate – black dashed curve.

B. Synthetic Data

We tested our model on a variety of one dimensional functions. Results are shown on Figure 4 with the functions shown above the respective charts[5]. Figure 4 contains averages of at least 10 runs.

For each case we generated 800 independently drawn data points from an interval where the respective function is *interesting* – the intervals are shown in Figure 4. For the training sets we added Gaussian input noise with variances $\Sigma = 0.2$, $\Sigma = 0.8$, and $\Sigma = 1.5$ – every column of Figure 4 corresponds to one of these three values. An additive Gaussian noise with variance $\Phi = 0.1$ was also added to the labels.

As a measure of performance we used the mean square error (MSE) of the learned function on the same interval where the training data was generated. We performed comparison between the following state-of-the-art GP input noise

[5]Missing columns means that the given method failed to give meaningful predictions in the respective setup.

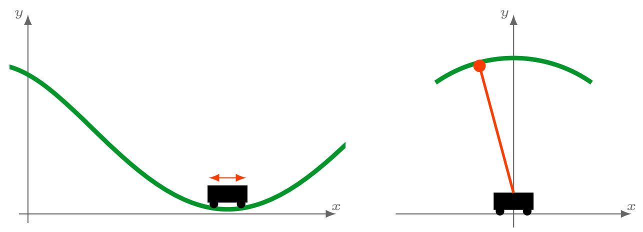

Fig. 5. Illustrations of the (a) car on the hill and (b) inverted pendulum.

modeling methods: (1) Standard GP (GP)[14]; (2) Simulation extrapolation with quadratic extrapolation function (SGP-Q); (3) Simulation extrapolation with GP extrapolation (SGP-GP); (4) Noisy input GP (NIGP) [12]; (5) Most likely heteroscedastic GP (MLHGP) [9]; (6) Approximate GP (GPA) [4]; (7) Hessian corrected GP (GP+H) [1]; (8) Variational heteroscedastic GP (VHGPR) [10]; We used the MATLAB implementation of the methods provided by the authors[6].

Most of the aforementioned methods do not report significant improvement in the mean squared error of posterior mean. They rather focus on better posterior variance estimations. We also highlight that we used homoscedastic – constant – noise. Making comparison with method designed for heteroscedastic noise, such as, VHGPR and MLHGP, may be *unfair*. They are included to obtain a more complete comparison. Another source of the difference in performance may be that most of the methods assume unknown input noise variance and its value is estimated as a parameter. We assume that this variance is known a-priori.

For all GPs we used a squared exponential kernel with hyper-parameters obtained using evidence maximization. For the second GP of VHGPR a linear kernel was used, while for the second GP of MLHGP we used the squared exponential kernel.

Results from Figure 4 show that the variance of the input noise has a significant effect on the improvement induced by our model. At low variance levels ($\Sigma = 0.2$) the improvement was insignificant, sometimes even leading to slight degradation of performance. At higher noise levels SIMEXGP outperformed nearly all of the methods, competing with GP+H. At the highest noise level ($\Sigma = 1.5$), an improvement of $30-60\%$ in the mean square error was obtained.

C. Learning Dynamics

To test the performance of our methods on higher dimensional data, we chose the problem of learning the short-term transition dynamics of nonlinear dynamical systems. In this setting, we model the transition function of the dynamical system as a GP, where inputs are state-action pairs and the outputs are the successor states. The state vectors are multi-dimensional, therefore, for each output dimension d, we train a separate GP, implying that there is no correlation between the output dimensions.

[6]Simulation extrapolation with linear extrapolation function is left out since it is essentially the same as GP+H.

This problem fits well to the problem of function approximation with noisy inputs, especially in the robotic control domain, where information about the robots current state is obtained from inaccurate sensors. For testing we used two well-known toy problems from reinforcement learning: the *car on hill* (a.k.a park on the hill) and the *cart-pole balancing*.

1) Experimental setup:: To assess the performance of the algorithms, we averaged the results of 30 experiments performed on data generated from the dynamical systems. In each experiment we generated 450 training data-points and 500 test-points, sampled from trajectories[7]. The start states were chosen randomly from the neighborhood of a fixed start-state and within the state-boundaries of the system. For each sampled state-action vector, one of the elements of the successor state was used for each separate GP as a label.

To generate the system inputs, we used a fixed Gaussian policy with a linear controller of the following form: $\pi(a|s) \sim \mathcal{N}(\text{ctrl}(s), \mathbf{\Pi})$, where $\text{ctrl}(s) = \sum_{i=1}^{d} \alpha_i s_i$ is the controller, α_i being the linear coefficients and $\mathbf{\Pi} = \text{diag}(\sigma)$ is the d dimensional diagonal covariance matrix. The parameters s and a denote the state and the action of the system. As a measure of performance we used the average of the prediction accuracy over the output dimensions. The prediction accuracy for a given dimension was calculated as the normalized mean squared error (NMSE) of the corresponding GP. In what follows, we describe each of the studied dynamical systems and present the test results where our method is compared to other state of the art noisy input regression methods mentioned in Section IV-B and the standard GP regression.

2) Car on hill control problem: A frequently used toy-problem in reinforcement learning is the Car on hill control problem [15]. In this setting a car is placed in the middle of a valley, the goal being to apply a sequence of horizontal forces such that it climbs out of the valley – see Figure 5.(a) for illustration. The maximum applicable force is such that the exit from the valley is only possible by a series of forward backward movements during which the car gains momentum.

This problem imposes a two dimensional continuous state space $s = [x \quad \dot{x}]$ composed of the position of the car x and its velocity $\dot{x}$. The applied action is one dimensional and continuous. For the approximation of the state-transition dynamics we used two GPs, each with 3 dimensional inputs and 1 dimensional output. We added normally distributed input noise with a range of different variances $\sigma \in [0.2, 2.0]$.

On Figure 6.(a) the bars indicate the normalized mean squared approximation errors averaged over the two output dimensions, for all five methods. At this level of input noise variance $\sigma = 1.8$ our method presents a significant improvement over the basic GP and the other noisy input methods. The same trend can be observed on Figure 6.(b) where the approximation errors are shown for a range of input variances, moreover the advantage of SIMEXGP increases with the variance of the input noise. It can be seen that SIMEXGP produced the best result consistently over all input noise variance values while the other methods did not have significant contribution.

[7] A trajectory is a set of successive states starting from an initial state and applying a fixed policy, until a terminal state has been reached.

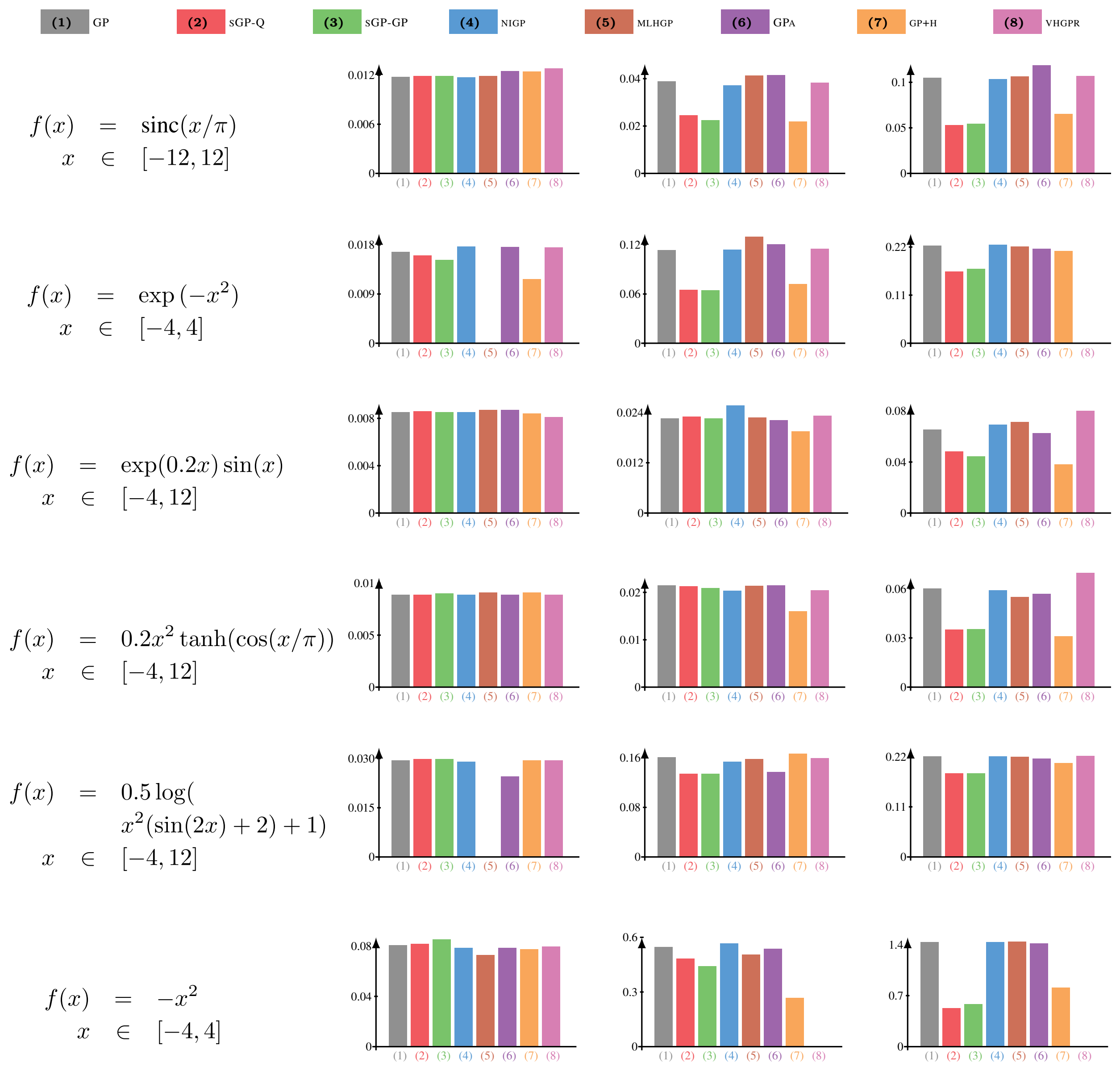

Fig. 4. Results for synthetic data (mean squared error). For each function the three columns correspond to different input noise variance levels: $\Sigma = 0.2$, $\Sigma = 0.8$, and $\Sigma = 1.5$.

3) Cart-pole balancing: We used the standard version of the cart-pole balancing problem, which was first introduced in [15]. A pole is attached with the help of a hinge to a cart that can move forward or backward in the plain. The goal of the problem is to control the horizontal force applied to the cart such that the attached pole stays upright – see Figure 5.(b) for illustration.

The underlying system has a four dimensional state space where a state variable $s = [x \quad \dot{x} \quad q \quad \dot{q}]$ has the following elements: x the position of the cart, $\dot{x}$ the velocity of the cart, q the angle of the pole, and $\dot{q}$ the angular velocity of the pole. The input of the system (the action) is one dimensional, continuous. For the approximation of the system dynamics we used four GPs where the input vectors were 5 dimensional – dim(states)+dim(actions) – and the output was one dimensional, representing one of the variables from the predicted next state. We added normally distributed input noise with a range of different variances $\sigma \in [0.2, 2.6]$.

In Figure 7 we plotted the evolution of the normalized mean squared error averaged over the four output dimensions, for different input noise variances. Although, the improvement induced by SIMEXGP is not as remarkable as in the previous experiment, the method is still consistently better over all input variance values.

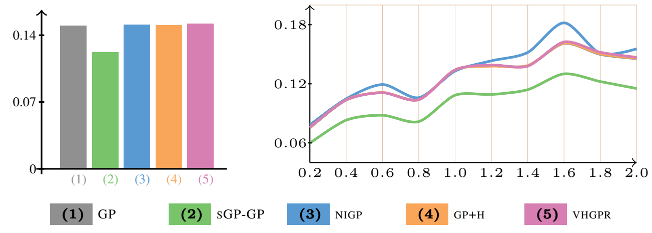

Fig. 6. Results of the car on the hill experiment. (a) Normalized mean squared error for $\sigma = 1.8$ and (b) normalized mean squared error for a range of input noise variances: $\sigma \in [0.2, 2.0]$.

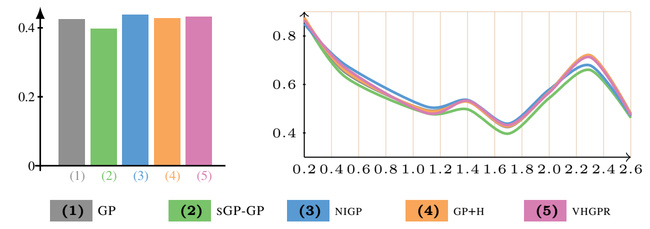

Fig. 7. Results of the pole balancing experiment. (a) Normalized mean squared error for $\sigma = 1.7$ and (b) normalized mean squared error for a range of input noise variances: $\sigma \in [0.2, 2.6]$.

V. Conclusions

We presented a method that corrects the bias of regression induced by input noise. SimEx has been extended to non-parametric function approximation and applied to GPs resulting in a probabilistic inference.

Our experiments show that significant improvement can be obtained using simexGP. Results on one dimensional functions suggest that the improvement is proportional to the variance of input noise.

The main limitation of the proposed method is the requirement of knowing the variance of the input noise a-priori. As future work we aim to alleviate this requirement (at least partially, *e.g.*, by having multiple measurements of the same experiment as suggested by Carroll et al. [2]). Another possible future direction is to make simexGP faster. Faster evaluation can be obtained by eliminating the Monte Carlo approximation of the integral from Equation (5) – either we find an analytical expression or use more efficient approximation methods.

The SimEx framework allows for changing the additive noise assumption to other input noise types in a straightforward way, *e.g.*, we change the addition of noise from Equation (3) into multiplication. This behavior is also an interesting direction for research.

We aim to apply simexGP in a *real-world* robotic setup, *e.g.*, learning kinematics, inverse kinematics, or reinforcement learning. Here, where we have to deal with data acquired from unreliable sensors that fits perfectly the prerequisites of simexGP.

The authors acknowledge the support of the Romanian Ministry of Education and Research via grant PN-II-RU-TE-2011-3-0278.

References

[1] Bócsi, B., Csató, L.: Hessian corrected input noise models. In: Proceedings of the International Conference on Artificial Neural Networks (ICANN). LNCS, vol. 8131, pp. 1–8. Springer, Sofia, Bulgaria (2013)
[2] Carroll, R.J., Ruppert, D., Stefanski, L.A., Crainiceanu, C.M.: Measurement Error in Nonlinear Models: A Modern Perspective, Second Edition. Chapman and Hall/CRC, 2 edn. (2006)
[3] Cook, J.R., Stefanski, L.A.: Simulation-extrapolation in parametric measurement error models. Journal of American Statistical Association pp. 1314–1328 (1994)
[4] Dallaire, P., Besse, C., Chaib-draa, B.: An approximate inference with Gaussian process to latent functions from uncertain data. Neurocomputing 74, 1945–1955 (2011)
[5] Girard, A., Murray-Smith, R.: Gaussian processes: Prediction at a noisy input and application to iterative multiple-step ahead forecasting of time-series. In: Murray-Smith, R., Shorten, R. (eds.) Switching and Learning in Feedback Systems. LNCS, vol. 3355, pp. 158–184. Springer (2003)
[6] Girard, A., Murray-Smith, R.: Learning a Gaussian process model with uncertain inputs. Tech. rep., University of Glasgow, Department of Computing Science (2003)
[7] Goldberg, P.W., Williams, C.K.I., Bishop, C.M.: Regression with input-dependent noise: A Gaussian process treatment. In: Advances in Neural Information Processing Systems (NIPS) (1997)
[8] Jaynes, E., Bretthorst, G.: Probability Theory: The Logic of Science. Cambridge University Press (2003)
[9] Kersting, K., Plagemann, C., Pfaff, P., Burgard, W.: Most likely heteroscedastic Gaussian process regression. In: Proceedings of the 24th international conference on Machine learning. pp. 393–400. ICML '07, ACM, New York, NY, USA (2007)
[10] Lázaro-Gredilla, M., Titsias, M.K.: Variational heteroscedastic Gaussian process regression. In: In 28th International Conference on Machine Learning (ICML). pp. 841–848. ACM (2011)
[11] MacKay, D.J.: Comparison of approximate methods for handling hyperparameters. Neural Computation 11, 1035–1068 (1999)
[12] McHutchon, A., Rasmussen, C.E.: Gaussian process training with input noise. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F.C.N., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 24. pp. 1341–1349 (2011)
[13] Quiñonero-Candela, J., Roweis, S.T.: Data imputation and robust training with Gaussian processes. Tech. rep., Technical University of Denmark (2003)
[14] Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)
[15] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
[16] Vapnik, V.N.: Statistical learning theory. John Wiley (1997)