Sparse Gaussian Process Regression using Progressively Growing Learning Representations

Christos N. Mavridis, George P. Kontoudis, and John S. Baras

***Abstract*— We present a new sparse Gaussian process regression model whose covariance function is parameterized by the locations of a progressively growing set of pseudo-inputs generated by an online deterministic annealing optimization algorithm. A series of entropy-regularized optimization problems is solved sequentially, introducing a bifurcation phenomenon, according to which, pseudo-inputs are gradually generated. This results in an active learning approach, which, in contrast to most existing works, can modify already selected pseudo-inputs and is trained using a recursive gradient-free stochastic approximation algorithm. Finally, the proposed algorithm is able to incorporate prior knowledge in the form of a probability density, according to which new observations are sampled. Experimental results showcase the efficacy and potential advantages of the proposed methodology.**

I. Introduction

Gaussian process (GP) regression models provide an efficient learning framework for non-parametric function approximation [1], [2]. They constitute non-parametric probabilistic learning models with the ability to (i) estimate uncertainty, (ii) cope with small datasets, and (iii) incorporate prior knowledge. As such, GP models can be used in many applications, including reinforcement learning [3], and dynamical model learning [4], [5]. However, their complexity can scale up to $O(N^3)$, where N is the number of the training data points. To reduce this computational cost, sparse GP approaches that make use of $M \ll N$ pseudo-inputs have been proposed [6]–[8].

The selection of the M pseudo-inputs is based on solving an optimization problem over the original dataset. The first approaches were based on maximum likelihood optimization solved using gradient ascent [6]. Such approaches showed good performance but were computationally expensive and lacked the properties of active learning, i.e., the ability to sequentially add more pseudo-inputs, as needed (see, e.g., Chapter 6 in [2]). Existing active learning approaches incorporate criteria from information theory that guide the sampling procedure and can yield powerful theoretical and practical results [9]–[11]. These approaches are based on solving a tractable approximation of an NP-hard optimization problem incorporating the mutual information between the existing pseudo-points and any possible new location for the next pseudo-point candidate. Online sparse GP algorithms have also been proposed to handle the sparse selection problem by observing input data one at a time [7], [8]. However, besides the complexity of these optimization problems, once a pseudo-point has been added to the model, it cannot be removed or replaced, and the location of the following pseudo-points heavily depends on the past selections.

In this work, we construct a sparse Gaussian process regression model based on a progressively growing set of pseudo-points that are computed using an online deterministic annealing learning algorithm [12]. A set of pseudo-inputs $\mu := \{\mu_i\}_{i=1}^{M}$, is used to represent the input space in an optimal way according to an entropy-regularized average distortion measure $F_T(\mu) := D(\mu) - TH(\mu)$. As we will show, solving a sequence of optimization problems $\min_\mu \ F_T(\mu)$ for decreasing values of T, induces a series of bifurcation phenomena (phase transitions) during which, the number of pseudo-inputs naturally increases. This behavior simulates an annealing process. and gives T the interpretation of a temperature parameter. Finally, the training rule of the above model is based on gradient-free stochastic approximation [12], [13], which results in fast online updates.

Adopting the above optimization framework, we parameterize the covariance function of the GP by the locations of the pseudo-inputs μ. By allowing the annealing process to achieve lower temperature levels T, we can gradually increase the number of pseudo-inputs μ and improve the quality of the GP fit, providing online performance-complexity control over the training process. This process resembles an active learning approach. In contrast to most existing approaches, however, the pseudo-inputs are not constrained to be a subset of the data, and previously selected pseudo-inputs can be modified at every stage. Moreover, the solution of the optimization problem depends on the underlying probability distribution of the input data, giving more weight to regions of the input space that are more represented. As will be shown, this probability distribution can also be constructed artificially to be used as a prior for accelerating the localization of appropriate pseudo-inputs $\{\mu_i\}$ for the sparse GP regression model. We empirically illustrate the efficacy and potential advantages of the proposed methodology compared to widely used sparse GP algorithms.

II. Gaussian Process Regression

A Gaussian process is a distribution over the space of functions where any subset of which has a Gaussian distribution [1]. It is completely defined by a mean function $m(x)$

Research partially supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111990027, by ONR grant N00014-17-1-2622, and by a grant from Northrop Grumman Corporation.

Christos N. Mavridis and John S. Baras are with the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland, College Park, USA. emails:{mavridis, baras}@umd.edu.

George P. Kontoudis is with the Maryland Robotics Center, Inst. of Systems Research and the Dept. of Aerospace Engineering, University of Maryland, College Park, MD, USA. email: kont@umd.edu.

and a covariance function $k(x, x^0)$, and can be viewed as probabilistic kernel machines if the covariance function is a semi-positive definite Mercer kernel, such that the covariance between points x_i and x_j is given by $k(x_i, x_j)$. Hence, it can provide not only a mean value prediction for a test sample, serving as a regression algorithm, but also quantify the uncertainty of the prediction at the test input measured in terms of standard deviation.

For regression, we assume the availability of N training inputs $X := \{x_i\}_{i=1}^N$, $x_i \in S \subset \mathsf{R}^d$, and corresponding outputs $y := \{y_i\}_{i=1}^N$, $y_i \in \mathsf{R}$, which are assumed to be instances drawn by the noisy process

$$y = f(x) + \epsilon,$$

where $\epsilon \sim \mathsf{N}(0, \sigma^2)$. The covariance function is then given by the $N \times N$ Gram matrix $K(X, X) = K$, which implies

$$p(y|X) = \mathsf{N}(0, K(X, X) + \sigma^2 \mathsf{I}).$$

The kernel function is chosen (or estimated from the data with respect to appropriate criteria) a priori and depends on a number of hyper-parameters θ, which are also called the hyperparameters of the Gaussian process, that need to be learned using the training dataset. We employ the squared exponential (SE) kernel given as

$$k(x_i, x_j) = c \exp \quad - \frac{\|x_i - x_j\|^2}{2\eta^2} \quad ,$$

with hyperparameters $\theta = (c, \eta)$. Hyperparameter training is typically done by maximizing the marginal log-likelihood

$$l(y|X, \theta) := -\frac{1}{2} \log |K + \sigma^2 \mathsf{I}| - \frac{1}{2} y^{\mathrm{T}} (K + \sigma^2 \mathsf{I}) \ {}^1 y,$$

where I is the identity matrix of same dimensions as K, and σ is the standard deviation of additive Gaussian noise. This learning step involves inverting a potentially large Gram matrix and can be performed offline. The prediction y for a test point x is computed by the conditional distribution on the test output given the training data and the test input. This is again a Gaussian distribution

$$p(y \ |X, Y, x \ ) = \mathsf{N}(y \ , \Sigma \ ), \tag{1}$$

where

$$\begin{aligned} y \ &= k \ {}^{\mathrm{T}} (K + \sigma^2 \mathsf{I}) \ {}^1 y, \\ \Sigma \ &= k(x \ , x \ ) - k \ {}^{\mathrm{T}} (K + \sigma^2 \mathsf{I}) \ {}^1 k \ , \end{aligned} \tag{2}$$

and $k \ = [k(x \ , x_1), \ldots, k(x \ , x_n)]$.

Gaussian process regression, as described above, however, is not useful for applications with large datasets. The time complexity for training is $O(N^3)$ for each iteration of the optimization (as it involves inversion of the Gram matrix $(K + \sigma \mathsf{I})$), while mean prediction requires $O(N)$ computations and variance prediction scales with $O(N^2)$. This also rules out the straightforward use of Gaussian processes in an incremental fashion.

To overcome this issue the prediction can be conditioned on just a subset of points (pseudo-points), which is typically learned by solving a large optimization problem over the entire dataset [6], [8]–[10]. In the following section we introduce the online deterministic annealing algorithm, that seeks an optimal representation of the input space to be used as pseudo-inputs for Gaussian process regression.

III. Pseudo-Input Generation with Online Deterministic Annealing

We represent the observated data by a random variable $X : \Omega \to S \subseteq \mathsf{R}^d$ defined in a probability space $(\Omega, \mathsf{F}, \mathsf{P})$. Given a similarity measure $d : S \to \mathrm{ri}(S)$ (where $\mathrm{ri}(S)$ represents the relative interior of S) the goal is to find a set of M pseudo-inputs $\mu := \{\mu_i\}_{i=1}^M$, $\mu_i \in \mathrm{ri}(S)$, on the input space such that an average distortion measure is minimized,

$$\min_{\mu} \ J(\mu) := \mathsf{E} \left[\min_i d(X, \mu_i) \right] . \tag{3}$$

This process is equivalent to finding the most suitable set of M local constant models, and results in a piecewise-constant approximation of the input space S. To construct a learning algorithm that progressively increases the number of pseudo-inputs M as needed, we will define a probability space over an infinite number of local models, and constraint their distribution using the maximum-entropy principle at different levels.

A. The Optimization Problem

First we need to adopt a probabilistic approach for (3), in which a quantizer $Q : S \to \mathrm{ri}(S)$ is defined as a discrete random variable with countably infinite domain $\mu := \{\mu_i\}$. Then we will constraint its distribution by formulating the multi-objective optimization

$$\min_{\mu} F(\mu) := D(\mu) - TH(\mu), \tag{4}$$

where

$$D(\mu) := \mathsf{E}\left[d\left(X, Q\right)\right] = \int p(x) \sum_i p(\mu_i|x) d_\phi(x, \mu_i) \ \mathrm{d}x$$

takes the place of $J(\mu)$ in (3), and

$$\begin{aligned} H(\mu) &:= \mathsf{E}\left[-\log P(X, Q)\right] \\ &= H(X) - \int p(x) \sum_i p(\mu_i|x) \log p(\mu_i|x) \ \mathrm{d}x \end{aligned} \tag{5}$$

is the Shannon entropy. This is now a problem of finding the locations $\{\mu_i\}$ and the corresponding probabilities $\{p(\mu_i|x)\} := \{p(Q = \mu_i|X = x)\}$. The Lagrange multiplier $T \in [0, \infty)$ controls the trade-off between D and H. As T is varied, we essentially transition from one solution of the multi-objective optimization (a Pareto point when the objectives are convex) to another, and, as we will show in Section III-B, reducing the values of T defines a direction that resembles an annealing process [12], [14].

We minimize F by successively minimizing it first respect to the association probabilities $\{p(\mu_i|x)\}$, and then with respect to the codevector locations μ. The solution of the optimization problem

$$F \ (\mu) := \min_{\{p(\mu_i|x)\}} F(\mu), \qquad \text{s.t.} \sum_i p(\mu_i|x) = 1 \tag{6}$$

is given by the Gibbs distributions

$$p\ (\mu_i|x) = \frac{\exp\ -\frac{d(x,\mu_i)}{T}}{\sum_j \exp\ -\frac{d(x,\mu_j)}{T}},\ \forall x \in S. \tag{7}$$

In order to minimize $F\ (\mu)$ with respect to the codevector locations μ we set the gradients to zero

$$\begin{aligned} &\frac{d}{d\mu}F\ (\mu) = 0 \implies \frac{d}{d\mu}\left(D(\mu) - TH(\mu)\right) = 0 \\ &\implies \sum_i \int p(x)p\ (\mu_i|x)\frac{d}{d\mu_i}d(x,\mu_i)\ \mathrm{d}x = 0 \end{aligned} \tag{8}$$

where we have used (7) and direct differentiation.

Next, we will show that (8) has a closed-form solution if the dissimilarity measure d belongs to the family of Bregman divergences—information-theoretic dissimilarity measures play an important role in learning applications and include the widely used Euclidean distance and Kullback-Leibler divergence [15].

Definition 1 (Bregman Divergence): Let $\phi : S \to \mathbb{R}$, be a strictly convex function defined on a vector space $S \subseteq \mathbb{R}^d$ such that ϕ is twice F-differentiable on S. The Bregman divergence $d_\phi : H \times S \to [0, \infty)$ is defined as:

$$d_\phi\left(x, \mu\right) = \phi\left(x\right) - \phi\left(\mu\right) - \frac{\partial \phi}{\partial \mu}\left(\mu\right)\left(x - \mu\right),$$

where $x, \mu \in S$, and the continuous linear map $\frac{\partial \phi}{\partial \mu}\left(\mu\right) : S \to \mathbb{R}$ is the Fréchet derivative of ϕ at μ.

Given Definition 1, we can prove the following.

Theorem 1: The optimization problem

$$\min_{\mu} F\ (\mu) \tag{9}$$

where $F\ (\mu)$ is defined in (6) is solved by the codevector locations μ given by

$$\mu_i = \mathbb{E}\left[X|\mu_i\right] = \frac{\int xp(x)p\ (\mu_i|x)\ \mathrm{d}x}{p\ (\mu_i)} \tag{10}$$

if $d := d_\phi$ is a Bregman divergence.

Proof: From Definition 1, we get,

$$\frac{\partial d_\phi}{\partial \mu}(x, \mu) = -\ \left\langle \nabla^2\phi(\mu), (x - \mu) \right\rangle\ , \tag{11}$$

where $x, \mu \in S$, and $\nabla^2\phi(\mu)$ represents the Hessian matrix of ϕ at μ. Then, (8) becomes

$$\int (x - \mu_i)p(x)p\ (\mu_i|x)\ \mathrm{d}x = 0 \tag{12}$$

which is equivalent to (10), as $\int p(x)p\ (\mu_i|x)\ \mathrm{d}x = p\ (\mu_i)$. ∎

B. Bifurcation and The Number of Pseudo-Inputs

In Section III-A we describe how to solve the optimization problem for a given value of the parameter T. To define an annealing approach, we are going to solve a sequence of optimization problems with decreasing values of T.

At very high temperature ($T \to \infty$), (7) yields uniform association probabilities $p(\mu_i|x) = p(\mu_j|x),\ \forall i, j, \forall x$, and as a result of (10), all pseudo-inputs are located at the same point $\mu_i = \mathbb{E}\left[X\right],\ \forall i$ which means that there is one unique "effective pseudo-input" given by $\mathbb{E}\left[X\right]$.

As T is lowered below a critical value, a bifurcation phenomenon occurs, when the number of "effective pseudo-inputs" increases, which describes an annealing process [12], [14]. Mathematically, it occurs when the existing solution μ given by (10) is no longer the minimum of the free energy F , as the temperature T crosses a critical value. Following principles from variational calculus, we can track bifurcation by the condition:

$$\frac{d^2}{d\epsilon^2}F\ (\{\mu + \epsilon\psi\})\Big|_{\epsilon=0} \geq 0 \tag{13}$$

for all choices of finite perturbations $\{\psi\}$. Using (13) and direct differentiation, we show that bifurcation depends on the temperature coefficient T (and the choice of the Bregman divergence, through the function ϕ) and occurs when

$$\frac{1}{T} = \frac{\partial^2\phi(y_n)}{\partial y_n^2}\bar{\nu} \tag{14}$$

where $\bar{\nu}$ is the largest eigenvalue of $C_{xjy_n} := \mathbb{E}\ (x - y_n)(x - y_n)^{\mathrm{T}}|y_n$. Moreover, since there is always a lower critical temperature value to be found, it follows that the number of "effective pseudo-inputs" always remains bounded between two critical temperature values.

In other words, the number of pseudo-inputs increases countably many times as the value of T decreases, and an algorithmic implementation needs only as many pseudo-inputs as the number of "effective pseudo-inputs". As shown in Alg. 1, we can detect the bifurcation points by introducing perturbing pairs of pseudo-inputs at each temperature level T. In this way, the pseudo-inputs μ are doubled by inserting a perturbation of each μ_i in the set of effective pseudo-inputs. The newly inserted pseudo-inputs will merge with their pair if a critical temperature has not been reached and separate otherwise. For more details about the implementation of the algorithm the reader is referred to [12].

C. Training Rule and Complexity

In the following Lemma we formulate a stochastic approximation algorithm that recursively estimates the solution to the optimization problem (10).

Lemma 1 ([12]): The online training rule

$$\begin{cases} \rho_i(n+1) &= \rho_i(n) + \beta(n)\left[\hat{p}(\mu_i|x_n) - \rho_i(n)\right] \\ \sigma_i(n+1) &= \sigma_i(n) + \beta(n)\left[x_n\hat{p}(\mu_i|x_n) - \sigma_i(n)\right] \end{cases} \tag{15}$$

where $\sum_n \beta(n) = \infty$, $\sum_n \beta^2(n) < \infty$, and the quantities $\hat{p}(\mu_i|x_n)$ and $\mu_i(n)$ are recursively updated as follows:

$$\mu_i(n) = \frac{\sigma_i(n)}{\rho_i(n)}, \quad \hat{p}(\mu_i|x_n) = \frac{\rho_i(n)\exp\ -\frac{d(x_n,\mu_i(n))}{T}}{\sum_i \rho_i(n)\exp\ -\frac{d(x_n,\mu_i(n))}{T}} \tag{16}$$

converges almost surely to a solution of (10).

Note that the recursive algorithm (15), (16) is also gradient-free, and converges to a finite set $\{\mu_i\}_{i=1}^{M}$ of locations that can be used as pseudo-inputs by a GP regression model. The pseudocode for the online determinsitic annealing algorithm is presented in Alg. 1 and the source code is available in [16]. A detailed discussion on the implementation of Alg. 1 and the effect of its parameters can be found in [12], [17].

Algorithm 1 Online Deterministic Annealing (ODA)

Select parameters and initial configuration $\{\mu_i\}$
while $M < M_{\max}$ **and** $T > T_{\min}$ **do**
  Perturb $\mu^i \leftarrow \{\mu_i + \delta, \mu_i - \delta\}$, $\forall i$
  Set $n \leftarrow 0$
  repeat
    Observe state x
    for $i = 1, \ldots, M$ **do**
      Update:

$$p(\mu_i|x) \leftarrow \frac{\rho(\mu_i) \exp\left(-\frac{d\ (x,\mu_i)}{T}\right)}{\sum_i \rho(\mu_i) \exp\left(-\frac{d\ (x,\mu_i)}{T}\right)}$$

$$\rho(\mu_i) \leftarrow \rho(\mu_i) + \beta_n \left[p(\mu_i|x) - \rho(\mu_i)\right]$$

$$\sigma(\mu_i) \leftarrow \sigma(\mu_i) + \beta_n \left[xp(\mu_i|x) - \sigma(\mu_i)\right]$$

$$\mu_i \leftarrow \frac{\sigma(\mu_i)}{\rho(\mu_i)}$$

    $n \leftarrow n + 1$
    end for
  until Convergence
  Keep effective codevectors
  Lower temperature $T \leftarrow \gamma T$
end while

The complexity of the recursive approach (15), (16) for a fixed temperature coefficient T_i (or λ_i) is $O(N_{c_i}(2M_i)^2 d)$, where N_{c_i} is the number of stochastic approximation iterations needed for the convergence of (15) and corresponds to the number of data samples observed, M_i is the number of codevectors of the model at temperature T_i, and d is the dimension of the input vectors, i.e., $X \in S \subseteq \mathsf{R}^d$. Therefore, assuming a training dataset of N samples and a temperature schedule $\{T_1 = T_{\max}, T_2, \ldots, T_{N_T} = T_{\min}\}$, the worst case complexity of the annealing approach becomes $O(N_c(2\bar{M})^2 d)$, where $N_c = \max_i \{N_{c_i}\}$ is an upper bound on the number of data samples observed until convergence at each temperature level, and $N_T \leq \bar{K} < N_T K_{\max}$, with the actual value of $\bar{K}$ depending on the bifurcations occurred. Note that typically $N_c \ll N$ and $\bar{K} \ll N_T K_{\max}$. For more details see [18] and the references therein.

IV. Sparse Gaussian Process Regression with Online Deterministic Annealing

The use of GP regression models based on the pseudo-inputs generated by Alg. 1 can be used for a smooth (in terms of continuity of the derivatives) function approximation. This also provides a way to mitigate the computational bottleneck of Gaussian processes, while conserving its properties, including the quantification of the uncertainty of the model [1], [11]. As a result, this approach can be appealing in many applications, including communication, control, multi-agent systems, and reinforcement learning [17], [19]–[21].

Following the definition of the GP regression model in Section II, the training inputs are now given by $\hat{X} := \mu = \{\mu_i\}_{i=1}^{M_T}$, where $M_T \ll N$ depends on the temperature level T of Alg. 1. The corresponding outputs $\hat{y} := \{y_i\}_{i=1}^{M_T}$, are given by $\hat{y}_i = f(\mu_i) + \epsilon$, and the covariance function is given by the $M_T \times M_T$ Gram matrix $K(\hat{X}, \hat{X}) = \hat{K}$, which implies that $p(y|\hat{X}) = \mathsf{N}(0, \hat{K} + \sigma^2 \mathsf{I})$, such that the prediction y for a test point x is computed by

$$p(y\ |\hat{X}, \hat{y}, x\ ) = \mathsf{N}(y\ , \Sigma\ ), \tag{17}$$

where $y\ = k\ ^{\mathrm{T}}(\hat{K} + \sigma^2 \mathsf{I})\ ^{1}\hat{y}$, and $\Sigma\ = k(x\ , x\ ) - k\ ^{\mathrm{T}}(\hat{K} + \sigma^2 \mathsf{I})\ ^{1}k$. Hyperparameter training now takes $O(M_T^2 N)$, mean prediction $O(M_T)$, and variance computation $O(M_T^2)$, where $M_T \ll N$.

A. Incorporating Priors

The locations of the pseudo-inputs $\mu = \{\mu_i\}_{i=1}^{M_T}$ depend on the underlying distribution which controls the frequency of observations we get by each region of the input space. In many applications, this is dictated by the observation mechanism, e.g., in probability density estimation, or, in control applications, where the data inputs come as a result of a sequence of actions. In standard function approximation, however, such a density does not naturally exist. In this case, we can construct an artificial probability density function that gives higher emphasis on regions of the space from which we expect larger improvement in the regression accuracy. Numerous heuristic objectives have been used in the literature to quantify potential regression improvement at a point of the input space:

The differential entropy $H(y\ ) := \frac{1}{2}\log(2\pi e \sigma^2(x\ ))$. This is a function of the predicted variance $\sigma^2(x\ )$ at point x and is used to greedily give more weight to underrepresented regions [10].

The Expected Improvement (EI). This objective function originates from Bayesian optimization and gives emphasis in the regions of the input space that result in the largest improvement [22]. In the context of function approximation, given the current prediction $\hat{f}(x)$ and the original function $f(x)$, EI can be defined by the expectation $\mathsf{E}[e(X)|\hat{x}, \hat{y}]$, where $e(X) = \|f(x) - \hat{f}(x)\|$ is the regression error. EI can be computationally hard to compute, since it takes into account the error $e(X)$ across the entire input space.

The magnitude of the gradient of the approximation function $\hat{f}$, i.e., $\|\nabla \hat{f}(x)\|$. This is not a commonly used objective function for searching the next pseudo-input in the GP regression literature. However, unlike the Expected Improvement (EI), it does not require the computation of the error $e(X)$ across the input space. In contrast, it uses the fact that the approximation function $\hat{f}(x)$ is differentiable as long as the kernel $\hat{K}$

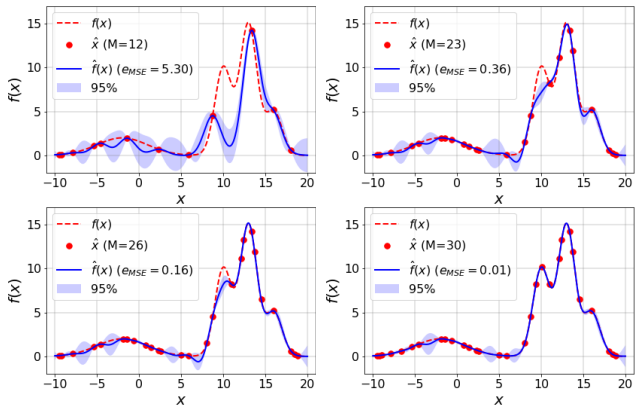

Fig. 1: Sparse GP regression model evolution with the differential entropy criterion.

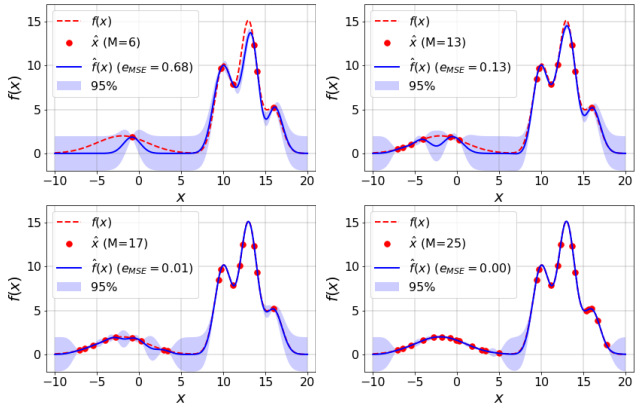

Fig. 2: Sparse GP regression model evolution with the Expected Improvement (EI) criterion.

is differentiable, with

$$\nabla \mathsf{E}\left[y \mid X, \hat{y}, \hat{x}\right] = \nabla k_*^{\mathrm{T}} (\hat{K} + \sigma^2 \mathsf{I})^{-1} \hat{y}.$$

This objective function gives more emphasis to the regions of the input space where the current estimate of the function $\hat{f}(x)$ changes rapidly, a similar principle to signal compression.

These heuristics can be used at every temperature level T to construct an artificial probability density for the observations of Alg. 1. As a result, the set of pseudo-inputs progressively grows in a way that resembles active GP learning approaches, while allowing for the modification of previously selected pseudo-inputs.

V. Experimental Results

We illustrate the properties and evaluate the performance of the proposed algorithm in 1D function approximation[1].

In Fig. 1 and 2 we showcase the evolution of the active learning process when increasingly more samples are added to the set of pseudo-inputs according to the differential entropy and the expected improvement objectives, respectively. These serve as a standard in the sparse GP regression literature. In Fig. 1, every point reduces the uncertainty of the model, but there is no mechanism to change the location of a pseudo-input after it has been added to memory. This often results in using an increased number of pseudo-inputs, with many of them being redundant (in retrospect). On the other hand, in Fig. 2, the EI objective is fast to find the regions of the input space that will more drastically improve the prediction of the GP model. Also, past pseudo-inputs are not able to be modified. In addition, the efficiency of the EI objective comes at the cost of the estimation of $\mathsf{E}\left[\|f(x) - \hat{f}(x)\| \mid \hat{x}, \hat{y}\right]$ across the function domain.

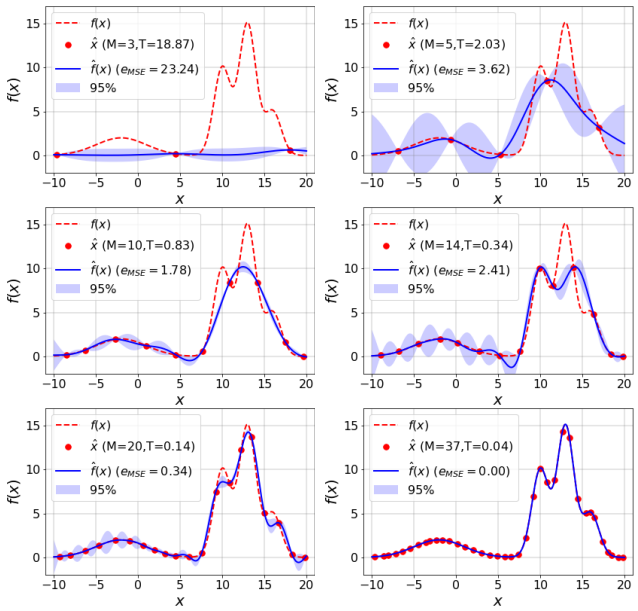

Fig. 3: Sparse GP regression model evolution with Alg. 1 and no prior.

In Fig. 3, we showcase the evolution of the learning process when a progressively growing set of pseudo-inputs is being generated by Alg. 1. In all applications of Alg. 1, the Euclidean distance is used as the proximity measure. As the temperature coefficient gradually decreases the number of pseudo-inputs progressively increases. In this case, no prior information has been used. To this end, the pseudo-inputs simulate a maximum entropy distribution, i.e., samples that are uniformly spaced in the input space. This behavior resembles the active learning algorithm in Fig. 1, since, as a result of the maximum entropy principle, the pseudo inputs are located such that the uncertainty of the model is being reduced. The main difference is that the locations of the pseudo-inputs are adjusted at every stage to preserve the maximum entropy.

In Fig. 4 and 5, Alg. 1 is used in conjunction with appropriately defined priors. In Fig. 4, $\|\nabla \hat{f}(x)\|$ is used to construct a probability density at every temperature level T_i, This is a heuristic that aims to find the areas that the function is fluctuating which results in a good approximation with quick convergence. This method does not guarantee the reduction of the MSE (mean-squared error) with the smallest number of pseudo-inputs. However, as a result of

[1]Code and Reproducibility: The source code is publicly available online at `https://github.com/MavridisChristos/ODASparseGaussians`.