18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

Online Sparse Gaussian Process Regression for Trajectory Modeling

Mattias Tiger
Department of Computer and Information Science
Linköping University, Sweden
Email: mattias.tiger@liu.se

Fredrik Heintz
Department of Computer and Information Science
Linköping University, Sweden
Email: fredrik.heintz@liu.se

***Abstract*—Trajectories are used in many target tracking and other fusion-related applications. In this paper we consider the problem of modeling trajectories as Gaussian processes and learning such models from sets of observed trajectories. We demonstrate that the traditional approach to Gaussian process regression is not suitable when modeling a set of trajectories. Instead we introduce an approach to Gaussian process trajectory regression based on an alternative way of combing two Gaussian process (GP) trajectory models and inverse GP regression. The benefit of our approach is that it works well online and efficiently supports sophisticated trajectory model manipulations such as merging and splitting of trajectory models. Splitting and merging is very useful in spatio-temporal activity modeling and learning where trajectory models are considered discrete objects. The presented method and accompanying approximation algorithm have time and memory complexities comparable to state of the art of regular full and approximative GP regression, while having a more flexible model suitable for modeling trajectories. The novelty of our approach is in the very flexible and accurate model, especially for trajectories, and the proposed approximative method based on solving the inverse problem of Gaussian process regression.**

I. Introduction

Gaussian processes [1] (GPs) is a flexible and powerful Bayesian non-parametric approach to modeling functions and performing inference on functions. They have been demonstrated to be practical and applicable to a wide variety of real-world statistical learning problems but also modeling, detecting and predicting spatio-temporal trajectories such as vehicles in crossings [2], marine vessel paths [3] and human body dynamics [4]. GPs are also good for handling noisy or missing data [2], [4], where large chunks of trajectories can be reliably reconstructed.

The usage of GPs for modeling purposes can be divided into two categories, (1) where an underlying function with exactly one function value for every input is sought from a set of data points or (2) where a distribution over functions is sought from a set of functions represented by sets of data points. In the first case we want to recover a function in an unknown or noisy environment where the variance of the GP distribution is the uncertainty of the predicted underlying function value. In the second case the variance of the predictive GP distribution should capture the paths of allowed or expected functions. Examples of the later are the modeling of allowed robot arm moment trajectories for task learning by demonstration [5] and modeling expected spatio-temporal car trajectories in crossings for traffic understanding and prediction [2]. This approach is useful for modeling trajectory clusters and for modeling motion paths. The second (2) modeling purpose is the focus of this paper.

Gaussian process regression is the inference of continuous values from a set of data with a Gaussian process prior. In this paper we demonstrate that the traditional approach to Gaussian process regression is not always the best option when GPs are used for modeling multiple trajectories. Instead we introduce an approach to Gaussian process trajectory regression based on an alternative way of combing two GP trajectory models and solving an inverse GP regression problem. Our method requires an existing Gaussian process regression algorithm for most applications, but is not limited to any specific algorithm since our method only needs to be able to evaluate the predictive mean and predictive variance of a GP distribution. This means that recent and future improvements of GP regression algorithms directly benefits our approach as well.

The benefit of our approach is that it allows for the usage of all available data, works well online and allows for sophisticated trajectory model manipulation. The flexibility of the methodology allows the merging and splitting of trajectory models, both parallel and sequential in the input space, accurately and in an efficient manner. The splitting and merging is very useful in spatio-temporal activity modeling and learning where trajectory models are considered discrete objects capturing for example a typical left turn or a slow down in a traffic monitoring application [6]. The presented method and accompanying approximate algorithm have time and memory complexities comparable to state of the art of regular full and approximative GP regression, while having a more flexible model for trajectory model manipulation.

The novelty of our approach is in the very flexible and accurate modeling, especially for trajectories, and the proposed approximative method based on solving the inverse problem of Gaussian process regression.

The structure of the paper is as follows. Section 2 provides the background to Gaussian processes and Gaussian process regression and section 3 contain related work. Our approach to trajectory modeling is presented in section 4 where we introduce the trajectory aggregation methods of combining and fusion and end the section with a computational complexity

analysis. Section 5 describes sparse inverse Gaussian process regression which is used in section 6 for our approach to sparse Gaussian process trajectory regression. This section describes trajectory learning in batch, online and contain a discussion of the complexities of the algorithms. Experiments are presented in section 7 and our conclusions in section 8.

II. Gaussian Process Regression

A Gaussian process is a distribution over functions,

$$f \sim \mathcal{GP}(m, k), \tag{1}$$

where the function f is distributed as a GP with mean function m and covariance function k. The mean function is often assumed to be zero, $m(x) = 0$, which means that a subtracted mean value of the data must be kept and handled outside of any Gaussian process regression in many applications. In this work we make use of the squared exponential kernel function (Eq. 2) which is commonly used in Gaussian Process regression.

$$k(x_1, x_2) \;=\; \theta_0^2 e^{-\frac{||x_1 - x_2||_2^2}{2\theta_1^2}}, \tag{2}$$

where θ_0^2 denotes the global variance of the mapping and θ_1^2 denotes the global smoothness parameter of the mapping. It provides a non-linear kernel space which allows a Gaussian process to model arbitrary non-linear functions.

In Gaussian process regression (GPR) [7] we want to estimate a mapping f,

$$y = f(x) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2), \tag{3}$$

from a set of data points $\{x_i, y_i, \sigma_{\epsilon_i}^2 \mid i = 1, \ldots, N\} = \{\mathbf{x}, \mathbf{y}, \epsilon_\sigma\}$, where $\mathbf{x}$, $\mathbf{y}$ and ϵ_σ are column vectors of the input, output and output noise respectively. The observation noise ϵ might be unknown but possible to estimate or even zero in the case of noise free observations. The noise is often assumed identically distributed for all data points. The parameters learned in GPR are the hyper parameters, in our case θ_0 and θ_1 (Eq. 2), and they are found by optimizing over the marginal likelihood function $P(\mathbf{y}|\mathbf{x})$ of the Gaussian process,

$$\log P(\mathbf{y}|\mathbf{x}) = -\frac{1}{2}\mathbf{y}^T\mathbf{V}^{-1}\mathbf{y} - \frac{1}{2}\log(\det(\mathbf{V})) - \frac{1}{2}N\log(2\pi), \tag{4}$$

where $\mathbf{V} = \mathbf{K}(\mathbf{x}, \mathbf{x}) + diag(\epsilon_\sigma)$ and $\mathbf{K}(\mathbf{a}, \mathbf{b})$ is the Gram matrix with entries $\mathbf{K}_{ij} = k(\mathbf{a}_i, \mathbf{b}_j)$ given by Eq. 2. The hyper parameters θ_0 and θ_1 that maximize the log likelihood are found by numerical optimization, but since the problem is non-convex only a local minimum is guaranteed and a good initial guess or the usage of random restarts is important.

For Gaussian process regression we assume a prior distribution over the function space. A prior distribution is shown in Fig. 1 (left) for the case of no previous training data (with the prior $\theta_0 = 1$ and $\theta_1 = 2$), in which the predictive distribution is flat with constant variance, and in Fig. 1 (middle) for the case that two data points have already been learned. A posterior distribution is calculated when some training data has been added to the process, which is illustrated in Fig. 1 (middle) for adding two data points and in Fig. 1 (right) for adding one additional data point. The third data point is noise free, while the two first data points was accompanies by some observation noise.

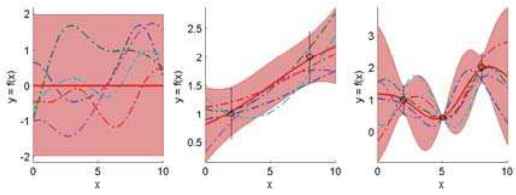

Fig. 1. Three GP probability density functions (pdf). Since GPs are generative models it is possible to generate samples from the distribution. Five function samples for each pdf are plotted in different colors. The pdf mean is showed as a thick red line and the 95% confidence is shown in light red around the mean. (Left) GP Prior. (Middle) GP Posterior with two data points. (Right) GP Posterior with three data points

The mean and confidence are shown in Fig. 1 are calculated by the predictive mean function (Eq. 5) and predictive variance function (Eq. 6) of the posterior GP distribution.

$$\mu(x^*) = m(\mathbf{x}) + k(x^*, \mathbf{x})\mathbf{V}^{-1}(\mathbf{y} - m(\mathbf{x})), \tag{5}$$

$$\sigma^2(x^*) = \mathbf{K}(x^*, x^*) - \mathbf{K}(x^*, \mathbf{x})\mathbf{V}^{-1}\mathbf{K}(x^*, \mathbf{x})^T. \tag{6}$$

The predictive mean and variance functions provide the possibility to represent the posterior distribution as a Gaussian distribution at any point x^*,

$$P(y^*|\mathbf{x}, \mathbf{y}, x^*) = \mathcal{N}(\mu(x^*), \sigma^2(x^*)). \tag{7}$$

Gaussian processes are continuous in the input x of the mapping (Eq. 3) and provide a value (the predictive mean) and an uncertainty (the predictive variance) everywhere in the domain of x.

The drawback with using GPs is their high computational costs. The optimization of the hyper parameters for training requires the inversion of a matrix ($\mathbf{V}$) as seen in Eq. 4 which has the complexity $O(N^3)$ in the number of data points N. The evaluation of the predictive mean (Eq. 5) and the predictive variance (Eq. 6) has the complexity $O(N)$ and $O(N^2)$ respectively per evaluated input x^* given that $\mathbf{V}^{-1}$ is pre-calculated. To reduce this computational cost a number of approximative inference approaches for a sparse posterior have been developed [8][9][10].

III. Related Work

There are several approximative methods for sparse GPs. Snelson and Ghahramani [8] introduce a set of pseudo-inputs as parameters in the covariance of their GPR model to get a sparse regression method with $O(NM^2)$ training time complexity and $O(M)$ respective $O(M^2)$ prediction time complexity per prediction, where M is the number of pseudo-inputs. Further approximations to this method combine local and global approaches which can perform even better under certain circumstances [11]. Local GP models [12] split the input domain into patches with a set of GPs modeling different

patches, thereby reducing the over all complexity of training and prediction to each patch's GP. This do however introduce problems with how the local models should overlap, and how to handle high densities of data in a single or several patches.

The sparse GP method of Snelson and Ghahramani requires all observed data points to be saved which means that the models grow linearly with the number of observations and prediction is computationally expensive for large data sets. The online sparse GP method of Ranganathan et al. [9] have an approximation that only keeps a window of data points with a "oldest-first" discarding strategy. Hensman et al. [10] get around the limitation of saving data points by using a set of global data points in a different way than Snelson and Ghahramani by using stochastic variational inference. In their approximative approach of online sparse Gaussian processes for Big Data the memory complexity is bounded in the number of induced points they use. As a consequence the time complexity for learning becomes $O(K^3)$ per data point and $O(K)$ respectively $O(K^2)$ for predicting mean and variance per prediction respectively, where K is the number of inducing variables. The memory complexity is $O(K)$. This means that the complexity of their approximation becomes independent of the number of data points, which allow much larger number of pseudo-inputs compared to the other approaches.

Modeling trajectory clusters is an important problem for trajectory clustering approaches to traffic behavior understanding [13]. Two approaches currently used are route envelopes consisting of a chain of connected nodes with breadth in their normal direction [14], chains of Gaussian distributions and Hidden Markov models [15]. The former method is susceptible to noise and is made more robust by defining the envelope probabilistically. Gaussian distributions have been widely used for this purpose. Gaussian processes is a generalization of a chain of Gaussian distributions.

Kim et al. [2] use GPs to model trajectories in a traffic monitoring application and address two important issues.

The first is how to model trajectories which may have different lengths. This is solved by normalizing all trajectories to run between time-point 0.0 and 1.0, given that they overlap fully spatially in the input domain.

The second is how to handle multiple GP trajectory models trained from different numbers of trajectories. The issue here is that the confidence band of a GP is narrower in regions with more samples, and it is therefore unfair to compare an input trajectory to the trajectory models because of uneven concentrations of data points within these models. Kim et al. balances the different trajectory models by a sampling strategy. They train GP models by using a representative set of observed trajectories known to be of the same class. These trajectories are normalized in a number of evenly distributed time segments and the data points used for training are selected by sampling the interpolated trajectories for a number of samples at each time segment. The number of samples at each time segment need to be the same for all learned trajectory models and they choose to sample three times per segment. This means that regardless of the number of representative trajectories of a class, only three of these will at any time segment be selected to sample from and to represent the class in this segment. This certainly provides good results with high probability, but the technique imposes limitations on future refinements of the model.

The approach we present allows us to make use of every continuous time point of all observed trajectories which we consider to be of the same class. This without having to worry about the issues related to varying data density between different time segments and varying number of observed trajectories used to learn the different classes' models. Our approach therefore allows future refinements without having to relearn the representative model from scratch and does in fact provide efficient online learning capabilities for continuous refinement of the models for each new observed trajectory.

IV. Trajectory Modeling

For a given trajectory class that we want to model, we assume that it is representable by a Gaussian process posterior distribution. An example of a trajectory class capturing a 2D turn is shown in Fig. 2. The Gaussian process posterior distribution's predictive-mean function captures the mean trajectory path of the class. The predictive-variance function of the GP posterior distribution captures the allowed variance of trajectory paths, the density within which any trajectory belonging to this class is expected to reside within. In the case of trajectories of higher dimension, for example 2D positions or a larger state space model, the trajectory class is modeled using a Gaussian process for each dimension for simplicity reasons. Treating the dimensions as independent introduces some limitations because we cannot capture the covariance between the outputs of each dimension, but we are still able to model common trajectories. We leave to future work to handle this for example using dependent GPs [16]. The two 1D GP models constituting the x- and y-dimensions of the full 2D trajectory class model in Fig. 2 are shown in Fig. 3.

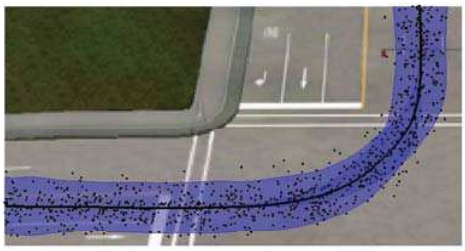

Fig. 2. A 2D GP model of a right turn class in a crossing. The black dots are the data points observed as part of 100 individual trajectories.

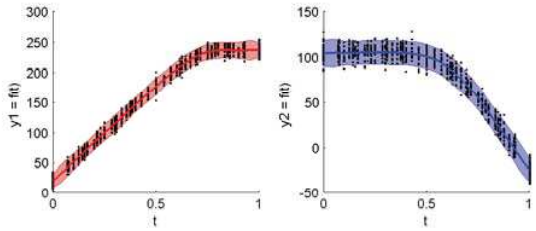

Fig. 3. The left figure shows the GP mapping normalized time $t \in [0\ \ 1]$ to the position in x-dimension ($y1 = f(t)$), while the right figure shows the GP mapping to the y-dimension ($y2 = f(t)$).

The first model to consider is the one which is learned from a single observed trajectory. It is reasonable to model a single observed trajectory using a Gaussian process with its inherent smoothing properties, given our assumption that the underlying model producing or explaining the observation is a Gaussian process. The smoothing of the trajectory compared to the observed points come from the ability of GPs to account for observation noise. We assume that trajectories are smooth, and that the posterior GP distribution of an observed trajectory is a good representative model of the un-observed trajectory of which the observations are samples of. This is illustrated in Fig. 4. Models of single trajectories are modeled using GPs and standard GPR is used.

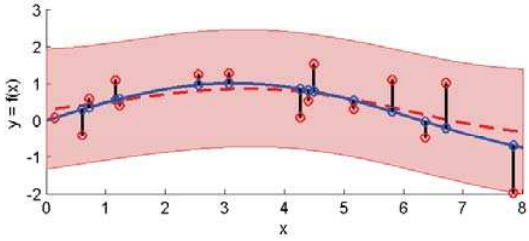

Fig. 4. The underlying function in blue is observed at 15 random x-values (blue diamonds) as $y = f(x) + \epsilon$ (red diamonds) with $\epsilon \sim \mathcal{N}(0, 0.5^2)$. The GP predictive distribution of the GP regression using the observed data points is shown with the mean as a red dashed line and the 95% confidence in light red.

Now that we are able to generate models of single trajectories, we would like to aggregate a set of observed trajectories modeled as GP posterior distributions into a new joint distribution of all the trajectory models together. The traditional approach would be to perform GP regression on the original set of data points from all the trajectories. However, this does not consider that each trajectory's data points are related and that the total set of data points are not independent, noisy samples from a single underlying process. Instead we therefore aggregate the trajectory models by treating each GP distribution as infinitely many point-wise Gaussian distributions seen as slices (Eq. 5-6). Using this approach we can both combine and fuse trajectory models. Combining is suitable for aggregating models of different trajectories while fusion is suitable for aggregating different models of the same trajectory. An example of the difference between regular GP regression and our approach to combining is shown in Fig. 5. The combining provides a Gaussian approximation of the distribution of a set of trajectories at each slice. It is a way to utilize the knowledge that the data points are divided into sets representing trajectories. Because of this, we argue that the representation is better in the case of modeling trajectories than merely GP regression over the total set of data points belonging to all trajectories in question.

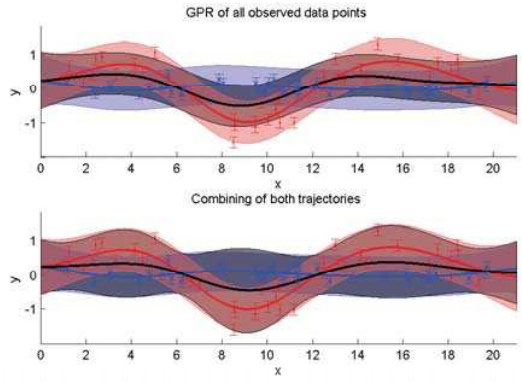

Fig. 5. Two trajectories are observed, one is blue and the other is red. The black/grey GP posterior in the top figure is the result of GPR of the data points from both trajectories while the bottom figure is the result of the combining of the two individual GP posteriors of each trajectory. The GP of all points do only barely capture the most likely trajectory paths (their predictive mean) of the two trajectories. The combined posterior captures the uncertainties in respective trajectory observation and accounts for this for both of them.

A. Combining trajectory models

Consider the case of each GP distribution representing a single trajectory. For a given slice for any given model we view the Gaussian distribution to be a distribution over a set of data points $\{z_k\}_{k=1}^{N}$ which we do not know except for their total mean and variance. Each Gaussian distribution of each model for a given slice contains an equal amount of data points since they all represent the same number of observed trajectories (namely one trajectory each). The data points of each Gaussian distribution is unique since the trajectories of each model are unique observations. Given the mean and variance of each Gaussian distribution $\{\mathcal{N}(\mu_j, \sigma_j^2)\}_{j=1}^{J}$ of a slice, we can calculate the total mean and variance of all the unknown data points z_k of all the Gaussian distributions. How this is done at any slice, indexed by x^*, is shown in Eq. 8-9 and these two equations constitutes the *combining formula*, which are derived in the appendix based on a re-written form of the population mean and population variance formula.

$$\mu(x^*) = \frac{\sum_{j=1}^{J} N_j \mu_j(x^*)}{\sum_{j=1}^{J} N_j}, \tag{8}$$

$$\sigma^2(x^*) = \frac{\sum_{j=1}^{J} N_j (\sigma_j^2(x^*) + \mu_j^2(x^*))}{\sum_{j=1}^{J} N_j} - \mu(x^*)^2, \tag{9}$$

where $\mu_j(x^*)$ and $\sigma_j^2(x^*)$ are the predictive mean and variance functions (Eq. 5-6) respectively. With the same amount of data points per set, these can be simplified by setting each $N_j = 1$, which is the case since the normalization by the sum $\sum_{j=1}^{J} N_j$ makes the weighting with $\{N_j\}_{j=1}^{J}$ relative and not absolute. However, if one of the trajectory models is based on two trajectories instead of one as the rest, then N_k of that set is twice as large compared to the other sets, i.e. $N_k = 2$. This provides a straight forward way of combining trajectory models learned with varying amount of training data (in the number of trajectories, independent of the number of data points in each trajectory). N_j in Eq. 8-9 is thereby the number of trajectories used to learn trajectory model j.

B. Fusion of trajectory models

One of the issues with local GP models is how the local models should overlap as well as how to perform coherent prediction over a chain of connected local GP models. In our setting this problem can be managed by applying the *fusion formula* [17] on the predictive mean and variance of the respective local GP posterior distributions. The resulting fused posterior distribution is a single continuous GP posterior-like distribution.

$$\sigma^2(x^*) = \frac{\sum_{j=1}^{J} N_j}{\sum_{j=1}^{J} N_j (\sigma_j^2(x^*))^{-1}}, \tag{10}$$

$$\mu(x^*) = \sigma^2(x^*)\left(\frac{\sum_{j=1}^{J} N_j \ (\sigma_j^2(x^*))^{-1} \ \mu_j(x^*)}{\sum_{j=1}^{J} N_j}\right) \tag{11}$$

Eq. 10-11 show a modified version of the fusion formula which also incorporates weights N_j that work in the same way as in the combining formula. However, when fusing two or more local GP models they will in most cases represent pieces of the same trajectory model and therefore model the same number of trajectories, i.e. $N_j = 1$ for all j. Fig. 6 shows combining and fusion of two Gaussian distributions, slices of the predictive distribution of a GP.

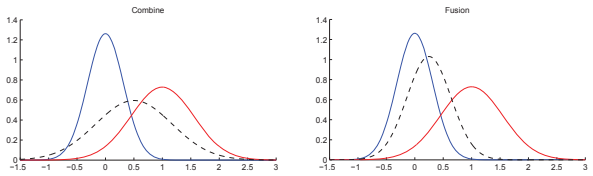

Fig. 6. Combining and Fusion of two Gaussian distributions with $N_1 = N_2 = 1$. The blue Gaussian distribution has mean 0 and variance 0.1. The red Gaussian distribution has mean 1 and variance 0.3.

Applying the fusion formula to the predictive distribution of multiple chained local GP models provides a straight forward way to get a predictive distribution, although it might be reasonable to reduce the influence of local models that are not next to each other for computational reasons.

It is also possible to merge local models by providing the fused predictive distribution to the sparse inverse GPR (SiGPR) algorithm described in the next section.

C. Computational Complexity

The point-wise mean and variance of the aggregated model are functions which are continuous in space and correspond to the predictive mean function and predictive variance function of a GP distribution. The aggregated distribution of the trajectory GP posterior distributions can be assumed to be a posterior GP distribution with an unknown mean and covariance function as well as unknown data points regressed over. What is known is its predictive mean and variance function.

If each trajectory is modeled by its own GP model then the space and time complexity of the regression of the aggregated model is amortized since we have a collection of *locally overlapping models*. The time complexity of GP regression for the aggregated model is $O(TK^3)$, where T is the number of trajectories and K is the (maximum) number of data points for any single trajectory. The time complexity of prediction has also improved and is $O(TK)$ for the mean and $O(TK^2)$ for the variance. The space complexity is the same as for regular GPR models.

The methods for trajectory model aggregation described in this section have a space and time complexity which grows linearly in the number of trajectories. Although this time complexity is a huge improvement compared to simultaneous Gaussian process regression of all data points, it can still be prohibitive to perform predictions for very large data sets with large numbers of trajectories or individual trajectories consisting of dense observations. Also the memory complexity can be prohibitive when storing all data points of all trajectories observed. This problem arises in Big Data and in online unsupervised applications with never ending flows of information. An often desired characteristic in such online unsupervised learning applications is to have a model that do not grow memory-wise with additional data.

Hensman et al. [10] achieve this with the approximation of using a set of representative inducing variables, smaller than the total number of data points, which are solved for by stochastic variational inference. The complexity for adding a new data point online becomes $O(K^3)$ and $O(MK^3)$ for a batch of M data points, where K is the number of inducing variables. This allows the selection of K to be independent of the total number of data points in the data set (N), and thereby allows K to be much larger than for SPGP by Snelson and Ghahramani [8] with time complexity $O(NK^2)$, where N needs to be the entire data set while K is a smaller set of pseudo-inputs (inducing variables). In the method by Hensman et al. only the inducing variables are needed for prediction and the complexity is reduced to $O(K)$ and $O(K^2)$ for the predictive mean and variance respectively.

We will in the following sections demonstrate how to achieve similar time and memory complexities as Hensman et al. but as an approximation to the trajectory model aggregation

by combining described above. We also use inducing data points but we do not minimizing the KL divergence of a lower bound of the log likelihood which they do. Our problem formulation is that we have an aggregated (e.g. combined) posterior distribution that we want to model by a single posterior distribution of an unknown GP, using a limited set of inducing data points. This is done by learning a parametric model in the form of a GP and input data which together produces a posterior distribution matching the wanted aggregated posterior. Since we want to use a smaller set of inducing variables than observed data points we compress a posterior GP distribution by finding a GP and input data that produces a approximately equivalent posterior distribution but using fewer inputs.

V. Sparse inverse Gaussian process regression

In this section we present how to solve the inverse problem to GP regression under certain limitations. More concretely, the problem we solve is that given a measurable posterior GP distribution with unknown data points and hyper parameters, recover a set of data points including their individual observation noise and hyper parameters. The data points we recover are called *support points* and we choose to let them be distributed with uniform distance between each other in the input domain (**x**) within a chosen range, see Fig. 7 for an example.

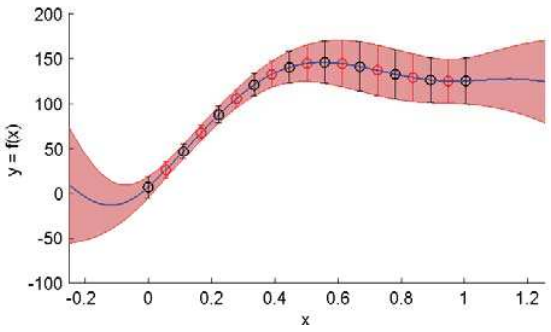

Fig. 7. The evaluation of the predictive mean function and the predictive variance function of a posterior GP distribution. The predictive mean is shown as a blue line and the predictive variance is shown as a light red 95% confidence around the predictive mean. The evaluation points that are distributed with uniform distance corresponding to **x** (the support points) is shown in black and the evaluation points between them shown in red, with the sampled mean shown as a circle and the sampled variance shown as the height of the 95% confidence of the error bars. **x** consists of 10 support points with a range of 0.0 to 1.0.

The uniform distance allows us to produce very good solutions by only evaluating the known predictive mean function and predictive variance function at and in the middle between support points. The reason for this is the following. We first notice that the predictive variance is independent of **y** and therefore is only dependent on the hyper parameters θ_0 and θ_1 of the kernel function (Eq. 2) together with the output noise $\sigma^2_{\epsilon_{1\dots K}}$. θ_0 is a global constant that scales the predictive variance and to a lesser extent scales the predictive mean. θ_1 is a global length scale that is proportional to the influence nearby input points have to a predicted point. It does not affect the variance at the input points **x** since there the kernel function is equal to θ_0. θ_1 does as a consequence affect the variance more the further away it is from $x \in \mathbf{x}$. With **x** being distributed with uniform distance we then know that the maximum influence of θ_1 on the predictive variance will be exactly in between each neighbor $\{x_i, x_j\} \in \mathbf{x}$. $\sigma^2_{\epsilon_j}$ is the observation noise for y_j at support point x_j and contributes to the predictive variance the most at x_j and less further away from x_j. The posterior distribution is approximated because we only use a limited number of inducing points and only require the predictive mean to be the same at the inducing input points **x** and in the middle between them. This works well in practice because the predictive mean is smooth due to the smooth prior on the GP due to the squared exponential kernel function.

According to Eq. 5-6 it is necessary to know **y**, ϵ_σ, θ_0 and θ_1, which are assumed to be unknown, in order to evaluate the predictive functions. However, we showed in the previous section how a combined assumed predictive mean and variance function can be evaluated without knowing hyper parameters or regressed data points of the presumed GP posterior distribution(s) equivalent to that of the combined GP posterior distribution. More precisely, what we need is to be able to evaluate a function representing the desired mean and a function representing the desired variance of the sought GP posterior distribution for some well chosen argument values. This is what the aggregated trajectory models from the previous section can provide.

The sparse inverse Gaussian process regression (SiGPR) algorithm takes as input a vector of GP input values $\mathbf{x} = \{x_k\}_{k=1}^K$ and two vectors of mean respective variance values, $\mu_{true}(x_j)$ and $\sigma^2_{true}(x_j)$ each with length M. The vector of input values **x** is assumed to have values that are distributed with uniform distance between each value. The mean and variance vectors contain the desired mean and variance values for the sought predictive GP distribution at each parameter value in **x** and in the middle of each neighboring values in **x**. Let this total vector of parameter values be called $\mathbf{x}_{total} = \{x_m\}_{m=1}^M$ where M is $2K-1$.

We want to find the parameters in ϕ,

$$\phi = \{\theta, \mathbf{y}, \epsilon_\sigma\} = \{\theta_0, \theta_1, y_1, \dots, y_K, \sigma^2_{\epsilon_1}, \dots, \sigma^2_{\epsilon_K}\}, \quad (12)$$

which minimize the least squares error function

$$\mathcal{E}(\phi) = \frac{1}{2M}\sum_{m=1}^{M} \mathcal{E}^2_{\sigma^2}(x_m, \phi) + \frac{1}{2K}\sum_{k=1}^{K} \mathcal{E}^2_{\mu}(x_k, \phi) \quad (13)$$

where

$$\mathcal{E}^2_{\sigma^2}(x, \phi) = (\sigma^2_{est}(x|\phi) - \sigma^2_{true}(x))^2, \quad (14)$$

$$\mathcal{E}^2_{\mu}(x, \phi) = (\mu_{est}(x|\phi) - \mu_{true}(x))^2. \quad (15)$$

Here $\mu_{est}(x|\phi)$ and $\sigma^2_{est}(x|\phi)$ are the predictive mean and predictive variance functions respectively of the GP posterior

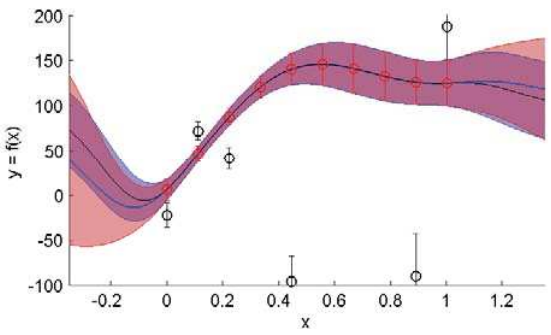

Fig. 8. The result of the SiGPR. The provided predictive mean is shown as a blue line and the predictive variance is shown as a light red 95% confidence interval around the predictive mean. The predictive mean of the estimated GP posterior is shown as a red line and the predictive variance is shown as a light blue 95% confidence interval around the predictive mean. Some of the support points are shown as black circles and their corresponding input value is shown as red circles placed on the posterior distribution. The range of the SiGPR is between 0.0 and 1.0 and the error in this range is $\mathbf{RMSE}_{\mu} = 0.1168$ and $\mathbf{RMSE}_{std} = 0.0163$.

distribution with support points $\{y_k, \sigma^2_{\epsilon_k}\}_{k=1}^{K}$ that we are solving for.

A solution is found using numerical optimization and the problem is non-convex for the same reasons as for optimization of the marginal likelihood function for regular GP training. The time complexity of the optimization hinges also on the inversion of the matrix $\mathbf{V}$ in the evaluation of the predictive mean and variance in Eq. 5-6 which potentially changes at every optimization step. The time complexity is for a naive implementation $O(M + M^2 + K^3) = O(K^3)$ where the evaluation of the predictive mean contributes $O(M)$, the evaluation of the predictive variance contributes $O(M^2)$ and the inversion of $\mathbf{V}$ contributes $O(K^3)$. The SiGPR of the GP posterior from Fig. 7 is shown in Fig. 8.

VI. Sparse Gaussian Process Trajectory Regression

Using the combining formula and the SiGPR it is now possible to do trajectory regression utilizing sparse Gaussian processes for both batch and online learning. The number of support points used in the SiGPR determines the trade-off between low computational time and accurate approximation. Since the support points are distributed with uniform distance it is possible to use the Nyquist-Shannon sampling theorem [18] to provide some guidance on the minimum number of support points that should be used for representing the predictive mean and variance functions of a posterior GP distribution for a desired input range. This well known theorem states that a function g can be fully reconstructed from samples if g contains no frequencies higher than f hz and the sample frequency is two times that frequency.

A. Batch Learning

For batch learning the data set is fully available and the learning is done once. Let $\{T_i\}_{i=1}^{N}$ be a set of trajectories each consisting of sets of observed data points $\{x_m, y_m, \sigma^2_{\epsilon_m}\}_{m=1}^{M}$. First each trajectory is modeled by GPR, with time complexity $O(NM^3)$. Then the combined posterior GP distribution is calculated using the *combining formula* and evaluated at the inducing points in preparation for the SiGPR, with a total time complexity of $O(N(M+M^2))$. Finally the sparse inverse GP regression is solved and the result is a GP model consisting of K support points and two hyper parameters which together produce the posterior GP distribution capturing the observed trajectories in the data set. Inverse SPGR has a time complexity of $O(K^3)$ and the total time complexity of batch learning is $O(NM^3 + N(M + M^2) + K^3) = O(NM^3 + K^3)$.

B. Online Learning

For online learning the data set becomes incrementally available over time. The GPR of a trajectory has time complexity $O(M^3)$ in the number of data points M of the trajectory. This needs to be done for each new observed trajectory as it becomes available.

When the first trajectory has been modeled using GPs then this constitutes the current model which is then refined with every new observed trajectory. This is done by using the *combining formula* of the current model and the GP model of the new trajectory and then performing the SiGPR which produces a new model which represents the previous model and the new trajectory. The complexity for the evaluation of the combining is $O(M^2 + K^2)$ and $O(K^3)$ for the SiGPR, where K is the number of support points. The total complexity for each new activity is thereby $O(M^3 + K^3)$ and for N trajectories it is $O(N(M^3 + K^3))$.

The online learning has the disadvantage of the risk of error accumulation with each SiGPR in comparison with the batch learning, since SiGPR is an approximative algorithm. We have therefore evaluated the error accumulation empirically in the experiment section.

C. Complexity Discussion

By using sparse GPR methods[8][11][9][10] the time complexity can be reduced to $O(NMP^2 + K^3)$ for batch learning and $O(N(MP^2 + K^3))$ for online learning, where P is the number of pseudo-inputs or inducing variables used. This is very valuable in cases where the trajectories are very dense in observed data points.

The time complexity for prediction is $O(K)$ for the predictive mean and $O(K^2)$ for the predictive variance. The memory complexity for batch learning is $O(NM)$ before learning and $O(K)$ afterwards, compressing NM data points into K inducing variables or support points. The memory complexity of the online learning algorithm on the other hand stays at $O(M+K)$ before learning and $O(K)$ after learning each new trajectory, in effect compressing $M + K$ data points into K support points.

The complexity of both batch and online trajectory learning is comparable to the complexity achieved with the stochastic variational inference approach [10] where they use inducing variables that fulfill a similar purpose as our support points.

The complexity in their work and for our algorithm is linear in the number of batches of data points (trajectories in our case) which means that their and our number of induction variables can be chosen to be large values independent of the number of data points in the total data set. The memory complexity is equally good between our methods, although our approach requires more support points in practice than they need inducing points since out support points are densely distributed with uniform distance. The memory complexity of our algorithm is $O(K)$ in the number of support points which means that it is independent of the number of trajectories and of the total number of data points. The memory complexity does not scale with higher dimensional trajectories, since a trajectory is 1D intrinsically. For the case of a generalization to random fields however our approach would scale $O(K^D)$ where D is the dimension. To handle higher dimensions we would be required to use inducing variables sparsely scattered in a random manner in the input domain, which is possible with the sparse GP approaches of Hensman et al. [10], Snelson & Ghahramani [8] and others. This is however outside of the scope of this paper since the focus is on trajectories.

VII. Experiments

We have compared the batch and online versions of the sparse Gaussian process trajectory regression algorithm on a publicly available simulated traffic intersection data set (CROSS) [15] used by others for evaluating trajectory pattern learning and clustering. The data set contains 19 different trajectory classes shown in Fig. 9. Each trajectory class contains about 90-100 trajectories with between 1028 and 2040 data points in total in each class.

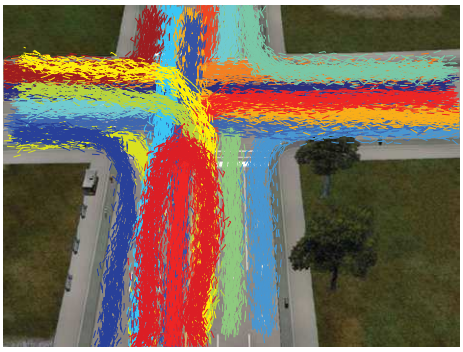

Fig. 9. The CROSS data set with 19 classes in different colors.

Each trajectory class is first modeled using batch learning. The resulting models are used as ground truth when testing the online learning and analyzing the error propagation. Fig. 10 shows the average and median root mean square error (RMSE) over all trajectory classes, both for the predictive mean and the predictive variance of respective x and y coordinate axis. The empirical evaluation using this data set shows a steady reduction of the model error for each new trajectory added and a convergence towards zero or some small constant.

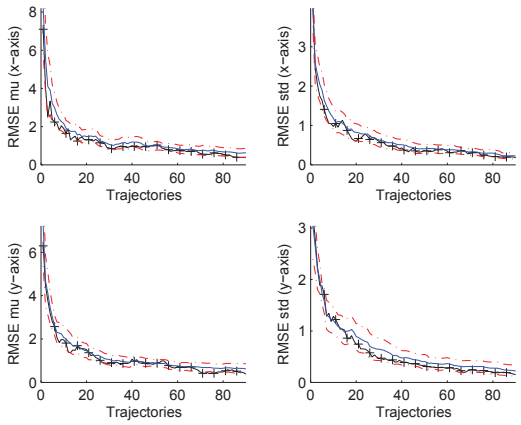

Fig. 10. Absolute RMSE in pixels between the online learned model and the ground truth combined model consisting of all observations. Average and median over all 19 trajectory classes. Mean value in blue, median value in black with plus symbols, 95% confidence in red.

Although the complexity for the online learning is slightly higher than for the batch learning, an interesting property of the online learning can be seen in Fig. 11. The computation time shows a decreasing trend as a function of the number of trajectories learned so far. This can be explained by the use of weights in the combining formula, where the relative influence of a new trajectory is decreasing as more and more trajectories are learned. So unless the new trajectory deviates significantly from the model the numerical optimization algorithm does not have to take many steps to account for the change.

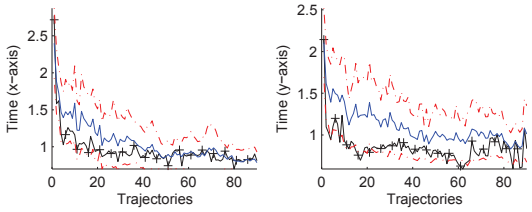

Fig. 11. Training time in seconds for each new trajectory, using 10 support points and optimizing in matlab using lsqnonlin without analytical Jacobian. Mean value in blue, median value in black with plus symbols, 95% confidence in red.

The main source of error in the online learning is observed to be close to the two edges in the input domain. This is because the SiGPR does not take into consideration the GP posterior distribution outside of the specified input range in which the support points are placed. It might be possible to reduce these errors by also taking the immediate outside of the input range into account, but that is left for future work.

Examples of the learned trajectory class models can be seen in Fig. 12, where the majority of models have sub-pixel RMSE accuracy compared to the batch learned models and the worst few have below 2 pixels RMSE compared to the

batch learning. The models capture the observed trajectories very accurately and it is only at the ends some errors can be noticed. For example the turning teal colored model in the top-right corner might be a little bit too much to the left at the top of the image.

All 19 models are shown in Fig. 13.

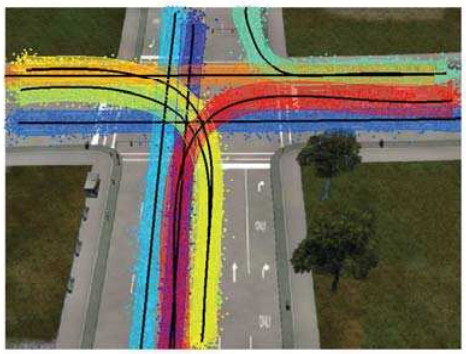

Fig. 12. Examples of the learned models in the CROSS data set. Models are shown for 8 trajectory classes.

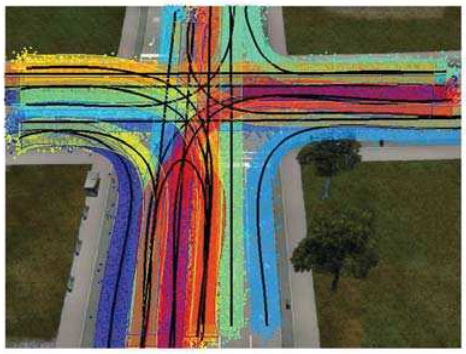

Fig. 13. All 19 trajectory classes and the learned GP models.

VIII. Conclusions

We have presented Gaussian process trajectory regression as an accurate and flexible approach for modeling and learning multiple trajectories using Gaussian processes. The approach is based on a better way of combining Gaussian process models of sets of trajectories than traditionally used and solving an inverse Gaussian process regression problem. The inverse GPR problem is to recover a set of data points including their individual observation noise and the GP hyper parameters, given a measurable posterior GP distribution with unknown data points and unknown hyper parameters. The presented method and accompanying approximation algorithm have the same time and memory complexities as state of the art of regular full and approximative GP regression. The approximative algorithm has been demonstrated to converge towards the ground truth by empirical evaluation. Besides being more appropriate for trajectories, it also supports online learning and sophisticated operations on trajectory models. The operations include splitting and merging of trajectories which is very useful in modeling and learning spatio-temporal activities where trajectory models are considered discrete objects [6]. We believe that the presented approach will be the starting point for many interesting extensions, especially in the direction of modeling and learning activities on multiple abstraction levels.

Acknowledgment

This work is partially supported by grants from the National Graduate School in Computer Science, Sweden (CUGS), the Swedish Foundation for Strategic Research (SSF) project CUAS, the Swedish Research Council (VR) Linnaeus Center CADICS, ELLIIT Excellence Center at Linkoping-Lund for Information Technology, and the Center for Industrial Information Technology CENIIT.

Appendix

Here we derive the *combining formula*. We consider the case where we have multiple Normal distributions which estimate the same global population using different observations, and we want to combine them into a single Normal distribution using all available observations. We further consider the population variance instead of the arithmetic variance since we assume that there is always a variance value associated with every sample (we do actually observe mean values and variance values together in the application) and therefor disregard the otherwise existing bias. The population mean and population variance is given by Eq. 16,

$$\mu = \frac{1}{N}\sum_{k=1}^{N} x_k, \quad \sigma^2 = \frac{1}{N}\sum_{k=1}^{N}(x_k - \mu)^2, \tag{16}$$

where N is the total number of data points x_k. By expanding the paranthesis of the population variance we get an expression relating the population variance, the squared population mean and the squared sum of data points (Eq. 17-19).

$$\sigma^2 = \frac{1}{N}\sum_{k=1}^{N}(x_k - \mu)^2 = \frac{1}{N}\sum_{k=1}^{N}(x_k^2 - 2x_k\mu + \mu^2) \tag{17}$$

$$= \frac{1}{N}\sum_{k=1}^{N} x_k^2 + -2\mu\frac{1}{N}\sum_{k=1}^{N} x_k + \frac{1}{N}\sum_{k=1}^{N}\mu^2 \tag{18}$$

$$= \frac{1}{N}\sum_{k=1}^{N} x_k^2 + -2\mu\mu + \mu^2 = \frac{1}{N}\sum_{k=1}^{N} x_k^2 - \mu^2 \tag{19}$$

This can be rewritten into an expression of the sum of squared data points on one side (Eq. 20-21).

$$\sigma^2 = \frac{1}{N}\sum_{k=1}^{N} x_k^2 - \mu^2 \Leftrightarrow \frac{1}{N}\sum_{k=1}^{N} x_k^2 = \sigma^2 + \mu^2 \tag{20}$$

$$\Leftrightarrow \sum_{k=1}^{N} x_k^2 = N(\sigma^2 + \mu^2) \tag{21}$$

We now consider the expressions for the population mean and population variance in Eq. 16-19 and divide respectively expression into J partitions of data points with N_j data points in each partition (Eq. 22-23).

$$\mu = \frac{\sum_{j=1}^{J}\sum_{k=1}^{N_j} x_{j,k}}{\sum_{j=1}^{J} N_j} = \frac{\sum_{j=1}^{J} N_j \mu_j}{\sum_{j=1}^{J} N_j} \tag{22}$$

$$\sigma^2 = \frac{\sum_{j=1}^{J}\sum_{k=1}^{N_j} x_{j,k}^2}{\sum_{j=1}^{J} N_j} - \mu_J^2 \tag{23}$$

$$= \frac{\sum_{j=1}^{J} N_j(\sigma_j^2 + \mu_j^2)}{\sum_{j=1}^{J} N_j} - \mu^2 \tag{24}$$

We can now substitute the sum of data points of each partition for the mean of the j:th partition times the number of data points in that partition, (Eq. 22). Likewise we can substitute the sum of squared data points of each partition for the expression we derived in Eq. 21, (Eq. 24). The derivation is complete.

REFERENCES

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[2] K. Kim, D. Lee, and I. Essa, "Gaussian process regression flow for analysis of motion trajectories," in *Proc. ICCV*, 2011.

[3] M. Smith, S. Reece, I. Rezek, I. Psorakis, and S. Roberts, "Maritime abnormality detection using gaussian processes," *Knowledge and Information Systems*, pp. 1–26, 2013.

[4] J. Wang, D. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 283–298, 2008.

[5] M. Schneider and W. Ertel, "Robot learning by demonstration with local gaussian process regression." in *Proc. IROS*, 2010.

[6] M. Tiger and F. Heintz, "Towards learning and classifying spatio-temporal activities in a stream processing framework," in *Proc. of the Starting AI Researcher Symposium (STAIRS)*, 2014.

[7] C. E. Rasmussen, "Gaussian processes for machine learning." MIT Press, 2006.

[8] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs." in *Proc. NIPS*, 2005.

[9] A. Ranganathan, M.-H. Yang, and J. Ho, "Online sparse gaussian process regression and its applications," *IEEE Transactions on Image Processing*, vol. 20, no. 2, pp. 391–404, Feb 2011.

[10] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," in *In Proc. UAI*, 2013.

[11] E. Snelson and Z. Ghahramani, "Local and global sparse gaussian process approximations." in *Proc. AISTATS*, 2007.

[12] D. Nguyen-Tuong and J. Peters, "Local gaussian process regression for real-time model-based robot control," in *In Proc. IROS*, Sept 2008, pp. 380–385.

[13] B. T. Morris and M. M. Trivedi, "Understanding vehicular traffic behavior from video: a survey of unsupervised approaches," *Journal of Electronic Imaging*, vol. 22, no. 4, pp. 041–113, 2013.

[14] N. Guillarme and X. Lerouvreur, "Unsupervised extraction of knowledge from S-AIS data for maritime situational awareness." in *Proc. FUSION*, 2013.

[15] B. Morris and M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2287–2301, Nov 2011.

[16] P. Boyle and M. Frean, "Dependent gaussian processes," in *In Advances in Neural Information Processing Systems 17*. MIT Press, 2005, pp. 217–224.

[17] F. Gustafsson, *Statistical Sensor Fusion*. Studentlitteratur, 2010.

[18] H. Nyquist, "Certain topics in telegraph transmission theory," *American Institute of Electrical Engineers, Transactions of the*, vol. 47, no. 2, pp. 617–644, April 1928.