Drifting Gaussian Process Regression for Inverse Dynamics Learning

Franziska Meier[1,2] and Stefan Schaal[1,2]

***Abstract*— Computationally efficient online learning of non-stationary models remains a difficult challenge. A robust and reliable algorithm could have great impact on problems in learning control. Recent work on combining the worlds of computationally efficient and locally adaptive learning algorithms with robust learning frameworks such as Gaussian process regression has taken a step towards both robust and real-time capable learning systems. However, online learning of model parameters on streaming data – that is strongly correlated, such as data arriving along a trajectory – can still create serious issues for many learning systems. Here we investigate the idea of drifting Gaussian processes which explicitly exploit the fact that data is generated along trajectories. A drifting Gaussian process keeps a history of a constant number of recently observed data points and updates its hyper-parameters at each time step. Instead of optimizing the neighborhood size on which the GP is trained on, we propose to use several – in parallel – drifting GPs whose predictions are combined for query points. Initial evaluations on both synthetic data and an inverse dynamics learning task illustrate the potential of this approach.**

I. INTRODUCTION

Machine learning has the potential to significantly improve model-based control. For instance, a good dynamics model is a key component of compliant control and force control for complex robots, like humanoids. However, due to unknown and hard to model nonlinearities, analytical models of the dynamics for such systems are often only rough approximations. Indeed, for this setting, machine learning algorithms have been investigated to provide automatic model learning [1], [2], [3].

Because sensors of the robot can generate thousands of data points per second, a necessary focus of these approaches has been computational efficiency – in addition to producing accurate models. To this end local learning approaches were developed [4], [5], that are fast enough for real-time learning and can handle non-stationary data by learning local distance metrics. However, besides computational cost, another crucial design criterion has to be reliability and robustness. To this end, recent work has moved towards using Gaussian process regression methods [6], [2], [3].

Gaussian process regression (GPR) offers a powerful and robust hyper-parameter learning framework. On top of that, GPR enables us to associate uncertainty estimates with the models prediction, which can be useful in gauging the reliability of the prediction. The main challenge is to create computationally efficient approximations of the standard GP framework, while retaining its robust learning framework. Recent work has combined the worlds of fast local learning with the robust GPR framework for the task of learning dynamics models [6], [3]. An alternative to localizing GPs are sparse GP approximations. Indeed, online versions of sparse GPs [7], [2] have produced a viable alternative for real-time model learning problems. However, these sparse approaches typically learn one global distance metric, making it difficult to fit the non-stationary data encountered in robotics. Moreover, restricting the resources in a GP also restricts the function space that can be covered, such that with the need to cover a growing workspace, the accuracy of learning will naturally diminish.

[1]Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089, USA. [2]Max-Planck-Institute for Intelligent Systems, 72076 Tübingen, Germany. Email:`fmeier@usc.edu`

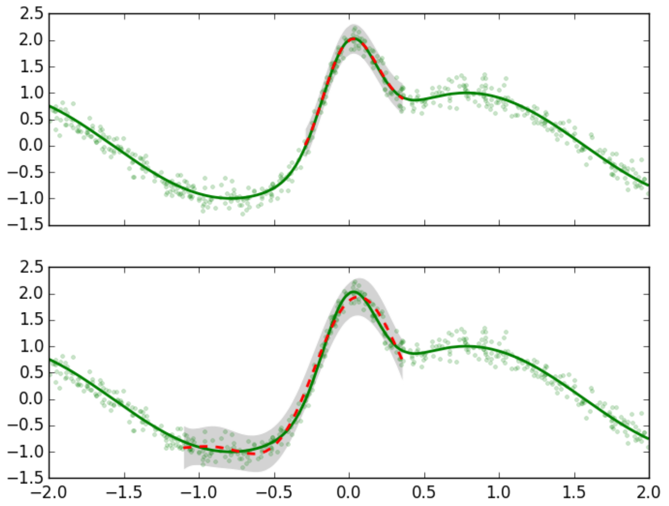

Fig. 1. Snapshot of 2 drifting Gaussian process regression models. The model on top is considering a shorter history of the trajectory than the model displayed on the bottom and thus was better able to adapt to the landscape.

Finally, a concern often not addressed is the online learning of (hyper)parameters on highly correlated data. Global models trained through online learning on correlated data would tend to unlearn previously optimized parameters to fit the currently observed data. This is true especially on tasks like learning kinematics or dynamics, where the generated data lies on a trajectory. Since it is not feasible to generate a representative dataset for offline training, algorithms need to be able to learn models from correlated streaming data. While stochastic learning of sparse GPs [8] exists, their success is dependent on randomly drawn mini batches. Thus it is questionable whether a global model learned online on correlated data is preferable over one locally drifting model.

To this end, we explicitly account for the fact that data comes in on a trajectory. Intuitively, at any given point in time one should be able to build a locally valid model from the last few observed data points. Here we explore the idea of a drifting GP. This allows us to be locally adaptive when experiencing curvature changes, but it comes with the advantages of GPs, namely robust parameter learning and uncertainty estimates. The key question is how many recently observed data points to utilize for training. Intuitively, one might think to use as many points as one can process in real time. However, Figure 1 illustrates that how many data points to use is also a question of the current function curvature. While we might be able to use a lot more of the recent observations, a model with *shorter attention span* might actually perform better because it can better adapt to the quickly changing curvature.

In the following, we review Gaussian process regression and sparse gaussian process regression in Section II, which we will utilize to achieve low computational complexity. Next, in Section III we introduce the notion of drifting Gaussian process models at varying neighborhood sizes. Finally, in Section IV an initial evaluation of our approach demonstrates the potential of this approach.

II. Background: Sparse Gaussian Process Regression

Gaussian process regression (GPR) [9] offers principled inference for hyper-parameters. However, the computational cost of GPR grows cubically with the number of data point.. Thus, in its standard form GPR is not practical in applications like inverse dynamics learning where thousands of data points are observed in just a few seconds of moving. Specifically, let N be the number of noisy observations $\boldsymbol{y}$, at input locations $\boldsymbol{X}$ with hidden (true) function values $\boldsymbol{f}$. In standard Gaussian process regression the likelihood function $p(\boldsymbol{y} \mid \boldsymbol{f})$ and the prior over hidden function values $p(\boldsymbol{f})$ are defined as

$$p(\boldsymbol{y} \mid \boldsymbol{f}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{f}, \beta^{-1}\mathbf{I}_N) \quad (1)$$

$$p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f} \mid 0, \mathbf{K}_{\text{NN}}) \quad (2)$$

where for simplicity we have assumed a zero-mean prior, and where $\mathbf{K}_{\text{NN}}$ is the kernel matrix evaluated between all input points $\boldsymbol{X}$. Optimization of the the hyper-parameters (noise precision β and parameters of the kernel) is performed by maximizing the log marginal likelihood

$$\log p(\boldsymbol{y}) = \iint p(\boldsymbol{y} \mid \boldsymbol{f})\, p(\boldsymbol{f})\, \mathrm{d}\boldsymbol{f} = \log \mathcal{N}(\boldsymbol{y} \mid 0, \mathbf{K}_{y,\text{NN}}) \quad (3)$$

Both, evaluation and optimization of the log marginal likelihood involves the inversion of $\mathbf{K}_{y,\text{NN}} = \beta^{-1}\mathbf{I}_N + \mathbf{K}_{\text{NN}}$ - thus causing the high computation cost of standard GPR.

As mentioned above, recent progress in sparsifying Gaussian processes [10], [11] has resulted in scalable implementations of GPR. These sparsification methods can broadly be classified into two approaches: Identification of M support points, where $M << N$, [12], [13] or sparsification of the spectrum of the GP [14], [15], which essentially transforms GPR to the parametric Bayesian regression domain. Both of these sparsification methods have been successfully applied in a variety domains.

A. *Variational Sparse Gaussian Process Regression*

Here, we are following the work of [12], which introduces a variational framework for optimization of M pseudo input locations. To create a sparse GP approximation [12] starts by introducing M additional hidden function values $\boldsymbol{u}$ at input locations $\mathbf{Z}$, to form the following hierarchical model

$$p(\boldsymbol{y} \mid \boldsymbol{f}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{f}, \beta^{-1}\mathbf{I}_N) \quad (4)$$

$$p(\boldsymbol{f} \mid \boldsymbol{u}) = \mathcal{N}(\boldsymbol{f} \mid \mathbf{K}_{\text{NM}}\mathbf{K}_{\text{MM}}^{-1}\boldsymbol{u}, \mathbf{K}_{\text{NN}} - \mathbf{K}_{\text{NM}}\mathbf{K}_{\text{MM}}^{-1}\mathbf{K}_{\text{MN}}) \quad (5)$$

$$p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid 0, \mathbf{K}_{\text{MM}}) \quad (6)$$

Marginalizing out $\boldsymbol{u}$ results in the standard GP regression setup from above, and thus marginalizing out both $\boldsymbol{u}$ and $\boldsymbol{f}$ results in the same expression for the log marginal likelihood as in (3),

$$\log p(\boldsymbol{y}) = \iint p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f} \mid \boldsymbol{u}) p(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}\, \mathrm{d}\boldsymbol{f}. \quad (7)$$

Thus with exact inference this model is equivalent to standard GPR and has the same high computational complexity. As a remedy, in [16] a variational approximation is sought that reduces the computational complexity. Specifically, [16] seeks to approximate the true posterior $p(\boldsymbol{f}, \boldsymbol{u} \mid \boldsymbol{y})$ with an approximate posterior $q(\boldsymbol{f}, \boldsymbol{u})$ by minimizing the KL-divergence $\mathrm{KL}(q(\boldsymbol{f}, \boldsymbol{u}) || p(\boldsymbol{f}, \boldsymbol{u} \mid \boldsymbol{y}))$. The key idea of using an approximate posterior $q(\boldsymbol{f}, \boldsymbol{u})$ is to capture the idea of $\boldsymbol{u}$ being a sufficient statistic for $\boldsymbol{f}$. Thus, optimizing pseudo-input locations $\mathbf{Z}$ *shifts* around $\mathbf{Z}, \boldsymbol{u}$ such that the sufficient statistics approximation is as exact as possible.

Marginalizing out $\boldsymbol{u}, \boldsymbol{f}$ under the approximate posterior induces a lower bound on the exact log marginal likelihood. This bound is given through

$$\mathcal{L}_{\text{sparse}}(\beta, \mathbf{Z}, \boldsymbol{\theta}) = \log \mathcal{N}(\boldsymbol{y} \mid 0, \beta^{-1}\mathbf{I}_N + \mathbf{K}_{\text{NM}}\mathbf{K}_{\text{MM}}^{-1}\mathbf{K}_{\text{MN}}) - \frac{\beta}{2} \operatorname{tr}\left[\mathbf{K}_{\text{NN}} - \mathbf{K}_{\text{NM}}\mathbf{K}_{\text{MM}}^{-1}\mathbf{K}_{\text{MN}}\right] \quad (8)$$

which depends on pseudo-input locations $\mathbf{Z}$, noise β and any kernel parameters $\boldsymbol{\theta}$. This lower bound on the log marginal likelihood becomes tight when $\mathbf{Z} = \boldsymbol{X}$. Note, how inversion of the $N \times N$ kernel matrix $\mathbf{K}_{\text{NN}}$ has been replaced by inverting the much smaller kernel matrix $\mathbf{K}_{\text{MM}}$.

III. Drifting Gaussian Process

We want to utilize sparse GP regression in the setting of continuous learning of, for instance, inverse dynamics. In this scenario data comes in on a trajectory. At time step t we want to use the last N data points to build a model to make a prediction for a new input $\boldsymbol{x}_{t+1}$. After observing the true y_{t+1} we want to update our model to include this new data point.

Thus let $\mathcal{M}^t$ denote the Gaussian process model at time step t, with parameter set $\{\beta^t, \theta^t, \mathbf{Z}^t\}$, where β^t is the noise precision, θ^t denote the kernel parameters and $\mathbf{Z}^t$ are the current pseudo input locations. This model $\mathcal{M}^t$ has been optimized using the last N observed data points $\mathcal{D}^t = \{\boldsymbol{x}_n, y_n\}_{n=t-N}^{t}$. We can thus use this model to make

Algorithm 1 Update Parallel Drifting GPs

```
Require: ${x_{t+1}, y_{t+1}}$, $D_k^t$, ${\mathcal{M}_k^t}_{k=1}^K$
    // update data set
 1: $\mathcal{D}_k^{t+1} \leftarrow \mathcal{D}_k^t - \{x_{t-N_k}, y_{t-N_k}\} + \{x_{t+1}, y_{t+1}\}$
    // initialize new model for time step $t+1$ with parameters
 2: $\mathcal{M}_k^{t+1} \leftarrow \mathcal{M}_k^t \forall k \in 1, \dots, K$
 3: $Z = 0$
 4: for $k = 1 \rightarrow K$ do
        // maximize log marginal likelihood (lml) of k$^{th}$ GP and return final lml
 5:     $\mathcal{L}_k = \mathcal{M}_k^{t+1} \rightarrow \text{optimize}(\mathcal{D}_{t+1})$
 6:     $Z = Z + \exp\{\mathcal{L}_k\}$
 7: end for
    // compute weight per GP, OPTION 1
 8: $w_k^{t+1} \leftarrow 1/Z \exp\{\mathcal{L}_k\}$
    // or via OPTION 2
 9: $k_{\max} = \arg\max \mathcal{L}_k$
10: $w_{k_{\max}}^{t+1} = 1.0, \forall k \neq k_{\max}, w_k^{t+1} = 0.0$
11: return ${\mathcal{M}_k^{t+1}, w_k^{t+1}}_{k=1}^K$
```

a prediction how much torque y^{t+1} is required to move to the next desired state. Once executed, the actual state and applied torque $\{x_{t+1}, y_{t+1}\}$ is used to update the model.

To avoid increasing computational complexity, we propose to drift the model along the trajectory instead of growing it. This allows us to keep the number of used data points N constant. The fact that data arrives on a trajectory can be used, to simply drop the *oldest* data point and replace it with the newest data point $\{x_{t+1}, y_{t+1}\}$. Specifically, a drifting GP model at time step $t+1$ is trained on $\mathcal{D}^{t+1}$, with $\mathcal{D}^{t+1} = \mathcal{D}^t - \{x_{t-N}, y_{t-N}\} + \{x_{t+1}, y_{t+1}\}$. Furthermore, instead of training models at each time step from scratch we can use the model from the previous time step to initialize the new one, and perform only a few optimization iterations per time step.

A. parallel GPs with varying neighborhood size

The key question now is how many past data points N to use to train the model at each time step with. It is likely that this number is not fixed and that it rather depends on the current curvature of the function to fit. Note, that in local methods this issue is being addressed by learning localized distance metrics, which define how large the neighborhood of active data points per local model is.

Here, instead of trying to learn the neighborhood size, we choose to run K drifting GP models $\mathcal{M}_k, \forall k = 1, \dots, K$ in parallel, each of which takes into account a different number N_k of data samples and thus covering neighborhoods with different sizes. At each time step all models $\mathcal{M}_k$ are updated with the new incoming data point. An overview of this algorithm is given in Algorithm 1. The question that remains is how to make predictions.

B. Prediction

Given a query point x^*, each of the K Gaussian Process models independently makes a prediction with mean μ_k^* and variance σ_k^*. A variety of options exist to form a final prediction. In this work we form the final prediction through a weighted combination of all GPs. Here we investigate two options:

a) Option 1, denoted as GP$_w$: Compute the lower bound on the log marginal likelihood value $\mathcal{L}(\mathcal{M}_k^t, \mathcal{D}_k^T)$ of each sparse GP, and compute the weights as

$$w_k = \frac{\mathcal{L}(\mathcal{M}_k^t, \mathcal{D}_k^t)}{\sum_{i=1}^K \mathcal{L}(\mathcal{M}_i^t, \mathcal{D}_i^t)} \tag{9}$$

b) Option 2, denoted as GP$_{max}$: At each time step we can choose the Gaussian process that has the highest log marginal likelihood value – more precisely the approximate lower bound $\mathcal{L}$, Equation (8) .Using this option, the weight of the model with the maximum lower bound is set to one, and all others are set to zeros:

$$k_{\max} = \arg\max \mathcal{L}_k \tag{10}$$

$$w_{k_{\max}}^{t+1} = 1.0, \quad \forall k \neq k_{\max}, w_k^{t+1} = 0.0 \tag{11}$$

Given these weights we can form a convex combination of all K predictions to compute the final predictive mean and variance as

$$\mu^* = \sum_{k=1}^K w_k \mu_k^* \qquad \text{and} \qquad \sigma^* = \sum_{k=1}^K w_k \sigma_k^* \tag{12}$$

The computation of the lower bound $\mathcal{L}(\mathcal{M}_k^t, \mathcal{D}_k^t)$ does not incur any additional cost, as this is the objective function that is being maximized with respect to the hyper-parameters. Thus, after each optimization step we automatically receive the current value of the lower bound, as illustrated in the pseudo algorithm 1.

IV. Experiments

We perform an initial evaluation of our approach on two datasets. First, we use a synthetic dataset with varying curvature to provide a detailed illustration of parallel drifting GPs. Then we use the Sarcos dataset, often used as a benchmark for inverse dynamics learning [9], for a quantitative analysis of the proposed method. In all experiments we use the squared exponential kernel $k(x_i, x_j) = \sigma_f^2 \exp\{-0.5 \sum_d \frac{1}{l_d^2}(x_{i,d} - x_{j,d})^2\}$ with automatic relevance determination. For both experiments, we perform 5 iterations of scaled conjugate gradients to update the hyperparameters given the current data sets $\mathcal{D}_k^t$, and given these models perform a one step look ahead prediction for input x_{t+1}. Errors reported are the (normalized) mean squared errors over these look ahead predictions.

A. Synthetic Data

For the synthetic data set we draw 1000 points from $f(x) = \sin(2x) + 2\exp(-16x^2)$ and add Gaussian noise with variance 0.1^2. We use $M = 5$ pseudo inputs for the approximate sparse Gaussian process model, which are optimized along the other

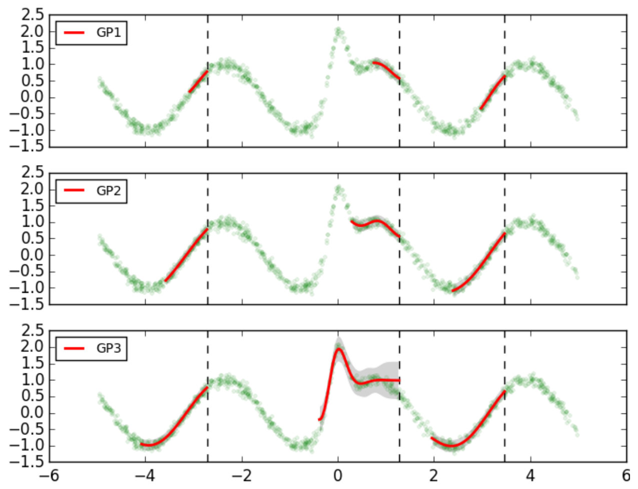

Fig. 2. Snapshots of all 3 Gaussian process regression models at 3 different time steps as they drift across the function. Black vertical dashed lines indicate the snapshot times. Noisy data points drawn from $f(x)$ shown in green.

TABLE I
PREDICTIVE PERFORMANCE ON SYNTHETIC DATA

MSE GP_1	MSE GP_2	MSE GP_3	MSE GP_w	MSE GP_{max}
0.01593	0.0154	0.0194	0.0150	0.0148

hyper-parameters at each time step. In this experiment we use $K = 3$ Gaussian process models that keep a history of the last $N_1 = 50, N_2 = 100, N_3 = 150$ data points seen to optimize their parameters. In Figure 2 we visualize a snapshot of the 3 drifting Gaussian process models GP_1, GP_2 and GP_3 at 3 different time steps. We can see that GP_3, which keeps the longest history (bottom), has a problem with the quickly changing curvature in the middle of $f(x)$. In Figure 3 we visualize the weighted one step look ahead predictions (top), computed using Option 2. We can see that the predictions are not affected by the suboptimal fit of GP_3 however. The bottom graph of Figure 3 displays a typical evolution of the lengthscale parameter l, showing how the lenghtscale decreases to account for the high curvature region of $f(x)$. We performed this experiment with 10 randomly seeded runs and report the average mean squared errors per Gaussian process model, and for both options of combining the predictions. GP_w and GP_{max} stand for predictions made via option 1) and option 2), respectively. On average, the weighted predictions of all 3 drifting GP outperforms the single GPs. Note however, that more importantly, in a real setting we wouldn't be aware which GP is performing how well. Thus, automatically combining the predictions of all K models is essential, and is here shown to perform at least as well as the best single GP.

B. Sarcos Data

We use the publicly available SARCOS data, which consists of rhythmic motions and contains $44,484$ training data points and $4,449$ test data points. The Sarcos arm has seven degrees of freedom, thus for the inverse dynamics learning tasks we have 21 input variables $\boldsymbol{x}$, representing joint positions, velocities and accelerations for the 7 joints. The task is to predict the 7 joint torques, each of which we treat as independent regression problems. We exclusively use the training set here, and simulate the setting of data incrementally arriving on a trajectory. We use $K = 2$ models, with GP_1 keeping a history of the last $N_1 = 1000$ and GP_2 of the last $N_2 = 2000$ data points. Each sparse GP uses $M = 10$ pseudo inputs that are being optimized via scaled conjugate gradients at each time step (along with the other hyper-parameters).

In Table II we present the normalized mean squared error made on the one step look ahead predictions, similar to the evaluation on the synthetic data set. It is noteworthy that these results are on par with recently reported result for local and global learning methods in [17]. Thus, drifting GPs can keep up with the current state of the art. Also, the weighted combinations for making the one step look ahead predictions, perform as on par or better as the best single drifting GP.

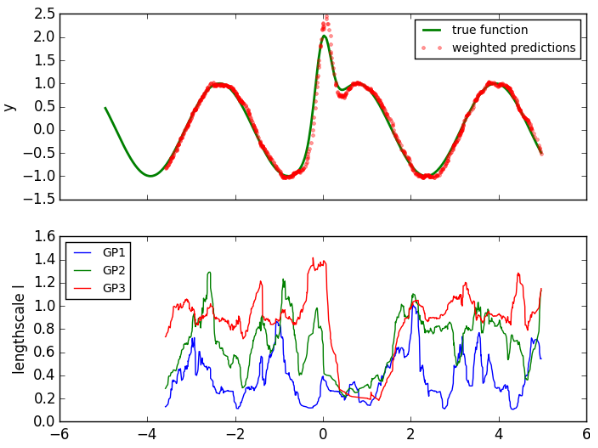

Fig. 3. Illustration of one instance of one step look ahead predictions using option 2 to combine predictions (top). Evolution of lengthscales for all 3 GP models (bottom).

TABLE II
PREDICTIVE PERFORMANCE ON SARCOS INVERSE DYNAMICS

Joint	nMSE GP_1	nMSE GP_2	nMSE GP_w	GP_{max}
j1	0.036	0.036	0.035	0.036
j2	0.048	0.041	0.042	0.042
j3	0.037	0.033	0.033	0.034
j4	0.013	0.013	0.012	0.011
j5	0.043	0.038	0.039	0.038
j6	0.060	0.057	0.055	0.056
j7	0.021	0.019	0.019	0.019

V. CONCLUSIONS

We have shown that a simple, drifting Gaussian process regression model can achieve good performance on tasks like inverse dynamics learning. We circumvent learning a neighborhood size - which typically involves gradient descent on non-convex objective functions - by drifting several GP models in parallel and investigate how to combine them to form predictions. Future work will investigate the use of such an approach on a real system.

References

[1] S. Vijayakumar, “Computational theory of incremental and active learning for optimal generalization,” Ph.D. dissertation, 1998.

[2] A. Gijsberts and G. Metta, “Real-time model learning using incremental sparse spectrum Gaussian process regression,” *Neural Networks*, vol. 41, pp. 59–69, 2013.

[3] F. Meier, P. Hennig, and S. Schaal, “Efficient Bayesian local model learning for control,” in *Proceedings of the IEEE International Conference on Intelligent Robotics Systems (IROS)*, 2014. [Online]. Available: http://www-clmc.usc.edu/publications/M/meier-IROS2014.pdf

[4] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally weighted learning for control,” *Artificial Intelligence Review*, no. 1-5, pp. 75–113, 1997. [Online]. Available: http://www-clmc.usc.edu/publications/A/atkeson-AIR-II-1997.pdf

[5] S. Schaal and C. G. Atkeson, “Constructive incremental learning from only local information,” *Neural Computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

[6] D. Nguyen-Tuong, J. R. Peters, and M. Seeger, “Local Gaussian process regression for real time online model learning,” in *Advances in Neural Information Processing Systems*, 2008, pp. 1193–1200.

[7] M. F. Huber, “Recursive Gaussian process: On-line regression and learning,” *Pattern Recognition Letters*, vol. 45, pp. 85–91, 2014.

[8] J. Hensman, N. Fusi, and N. D. Lawrence, “Gaussian Processes for Big Data,” *Proceedings of UAI*, pp. 282–290, 2013. [Online]. Available: http://auai.org/uai2013/prints/papers/244.pdf

[9] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[10] J. Quiñonero Candela and C. E. Rasmussen, “A unifying view of sparse approximate Gaussian process regression,” *JMLR*, vol. 6, pp. 1939–1959, 2005.

[11] K. Chalupka, C. K. Williams, and I. Murray, “A framework for evaluating approximation methods for Gaussian process regression,” *JMLR*, vol. 14, no. 1, pp. 333–350, 2013.

[12] M. K. Titsias, “Variational learning of inducing variables in sparse Gaussian processes,” in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 567–574.

[13] E. Snelson and Z. Ghahramani, “Sparse Gaussian processes using pseudo-inputs,” *Advances in neural information processing systems*, vol. 18, p. 1257, 2006.

[14] A. Rahimi and B. Recht, “Random features for large-scale kernel machines.” in *NIPS*, 2007.

[15] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, “Sparse spectrum Gaussian process regression,” *JMLR*, vol. 11, pp. 1865–1881, 2010.

[16] M. Titsias, “Variational Learning of Inducing Variables in Sparse Gaussian Processes,” in *AISTATS*, vol. 5, 2009, pp. 567–574. [Online]. Available: http://eprints.pascal-network.org/archive/00006353/

[17] F. Meier, P. Hennig, and S. Schaal, “Incremental Local Gaussian Regression,” in {*Advances in Neural Information Processing Systems*}, 2014, pp. 972–980.