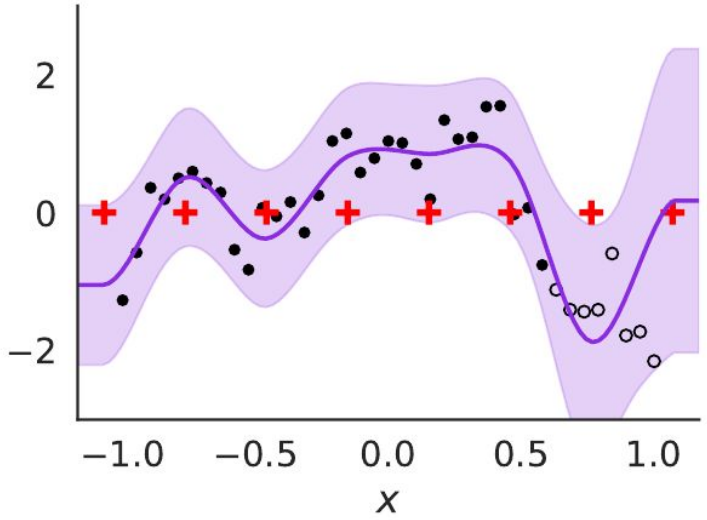
Kernel Interpolation for Continual Learning with Gaussian Processes

Samuel Stanton, Wesley Maddox,
Ian Delbridge, Andrew Gordon Wilson

November 16, 2020



Decision-Making and Continual Learning

Active Learning

Where should I direct data collection effort?

Contextual Bandits

How do I react to changes in a market?

Black-Box Optimization

How do I find promising candidates in a large search space?

Adaptive Control

How do I react to changes in my environment?

Decision Making and Continual Learning

- ML research is often evaluated on static benchmarks.
- Many motivating applications are online decision-making problems.
- Decisions must be timely, and made with incomplete information.
- When new information is available it should be leveraged immediately.

Decision-Making and Continual Learning

Continual Learning is characterized by:

- A vague/unknown range of possible observations.
- A datastream that is not assumed to be I.I.D.
- Transient observations (no training on old data).

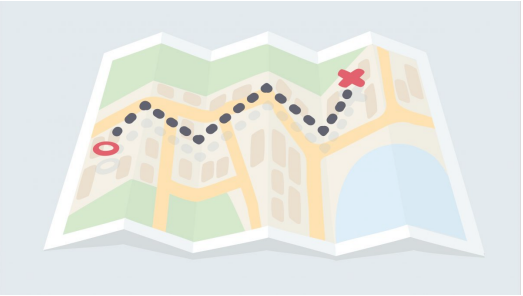
Outline

I. Intro to Gaussian Processes

II. GP Inference in the Streaming Setting

III. Kernel Interpolation for Continual Learning

IV. Empirical Results



I. Intro to Gaussian Processes

Gaussian Process Regression

Some Definitions

$$\mathbf{x} \in \mathbb{R}^d,\ y \in \mathbb{R} \qquad \mathcal{D} := \{(\mathbf{x}_i, y_i)\}_{i=1}^n$$

$$y = f(\mathbf{x}) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2)$$

$$f \sim \mathcal{GP}(0, k_\theta(\cdot,\cdot)) \qquad \mathbf{K}_{XX} := k_\theta(\mathbf{x}_{1:n}, \mathbf{x}_{1:n})$$

GP Inference

$$\mathcal{L}(\theta) = \log p(\mathbf{y}|\theta) := \log \int p(\mathbf{y}|f)p(f|\theta)df$$

$$= -\frac{1}{2}\left(\mathbf{y}^\top(\mathbf{K}_{XX} + \sigma^2 I_n)^{-1}\mathbf{y} + \log|\mathbf{K}_{XX} + \sigma^2 I_n|\right) + \text{const}$$

- Computing the marginal log-likelihood (MLL) is naively **O(n^3)**.
- Krylov subspace methods can get the cost down to **O(kn^2)**.
- Distributed computation can scale up to millions of points in the batch setting[1].

1. Wang et al (2020)

Querying a GP Posterior

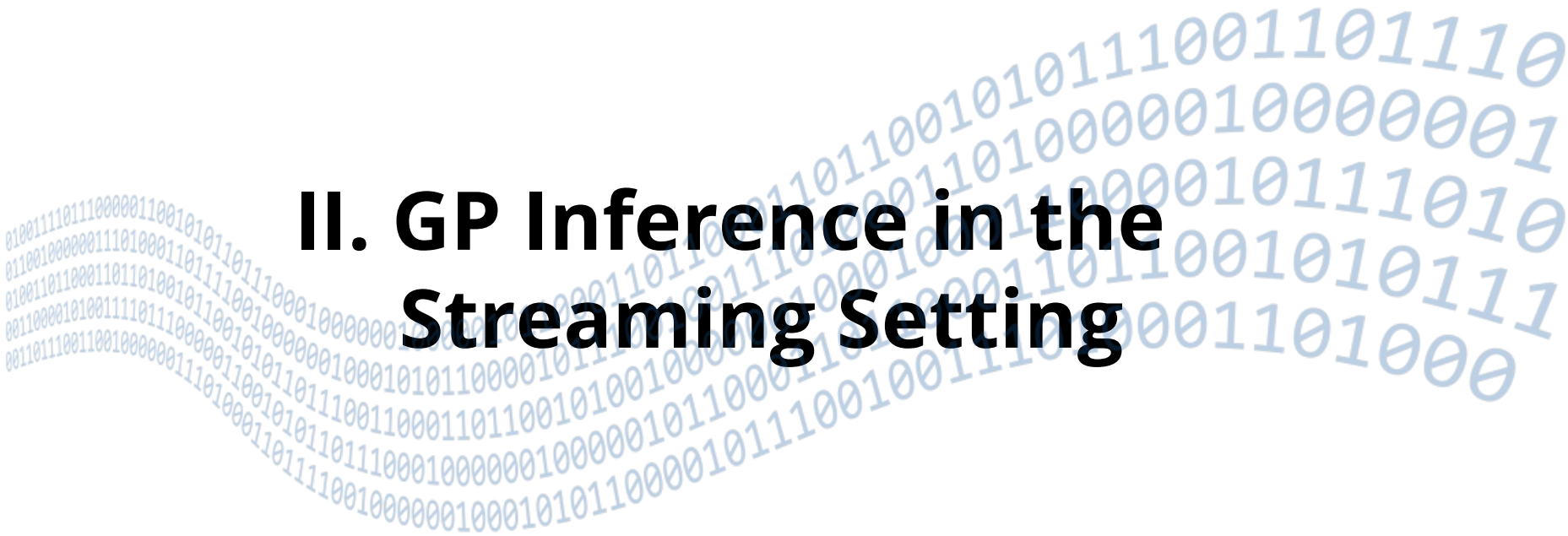
$$p(y|\mathbf{x}^*, \mathcal{D}, \theta) = \mathcal{N}(y; \mu(\mathbf{x}^*), \Sigma(\mathbf{x}^*))$$

$$\mu(\mathbf{x}^*) = K_{\mathbf{x}^* X}(K_{XX} + \sigma^2 I)^{-1}\mathbf{y}$$

$$\Sigma(\mathbf{x}^*) = K_{\mathbf{x}^*\mathbf{x}^*} - K_{\mathbf{x}^* X}(K_{XX} + \sigma^2 I)^{-1}K_{X\mathbf{x}^*}$$

- After an update to the GP hyperparameters a query costs **$O(n^3)$**.
- Again, Krylov subspace methods can get the cost down to **$O(kn^2)$**.

II. GP Inference in the Streaming Setting



Online Learning

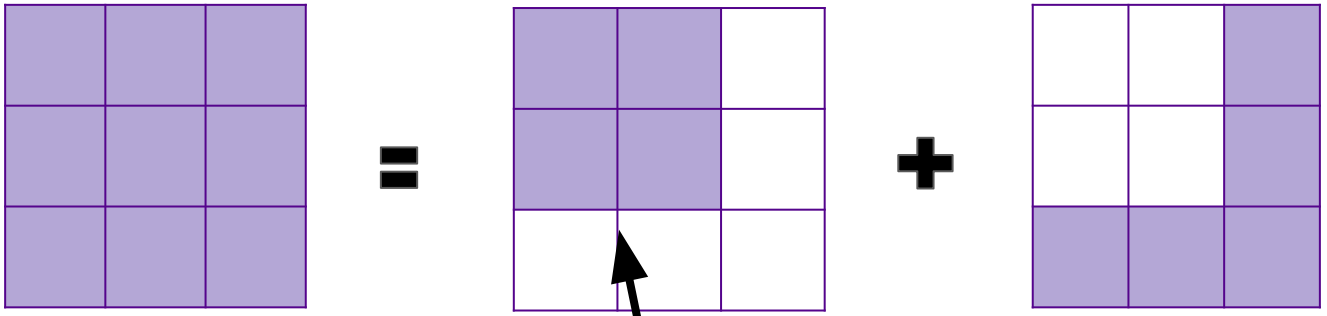
Problem

To learn and query as we do in the batch setting, we have to discard previous work.

Is there a way to reuse computation from one timestep to the next?

for $t = 1, 2, \ldots$ **do**
  receive $\mathbf{x}_t$
  predict $\hat{y}_t \sim \hat{p}(y|\mathbf{x}_{1:t}, \mathbf{y}_{1:t-1}, \theta)$
  receive $\mathbf{y}_t$
  update θ
end

Conditioning on New Observations



$$(\mathbf{K}_{X'X'} + \sigma^2 I) \qquad (\mathbf{K}_{XX} + \sigma^2 I) \qquad \begin{bmatrix} \mathbf{0} & k(X, \mathbf{x}') \\ k(\mathbf{x}', X) & k(\mathbf{x}', \mathbf{x}') + \sigma^2 \end{bmatrix}$$

Rank-one updates cost **O(n²)** naively, (e.g. Schur complement update, rank-one Cholesky update), at best **O(kn)** with low-rank approximate root decompositions.

Previous Work

- **Sparse GPs in the batch setting:** Silverman (1985), Snelson & Ghahramani (2006), Hensman et al (2013), Wilson et al (2015).
- **Early work on continual learning with sparse GPs:** Csato & Opper (2002), Girard et al (2002).
- **Recent work on continual learning with sparse GPs:** Cheng & Boots (2016), Bui et al (2017), Moreno-Muños et al (2019).

Subset of Regressors

Introduce inducing inputs **z** and inducing function values **u**,

$$\mathbf{z}_1, \ldots, \mathbf{z}_m \in \mathbb{R}^d \qquad \mathbf{u}_1, \ldots \mathbf{u}_m := f(\mathbf{z}_1), \ldots, f(\mathbf{z}_m)$$

Introduce an approximate kernel matrix,

$$\tilde{\mathbf{K}}_{XX} := \mathbf{K}_{XU}\mathbf{K}_{UU}^{-1}\mathbf{K}_{UX} \qquad \mathbf{K}_{UU} := k_\theta(\mathbf{z}_{1:m}, \mathbf{z}_{1:m})$$

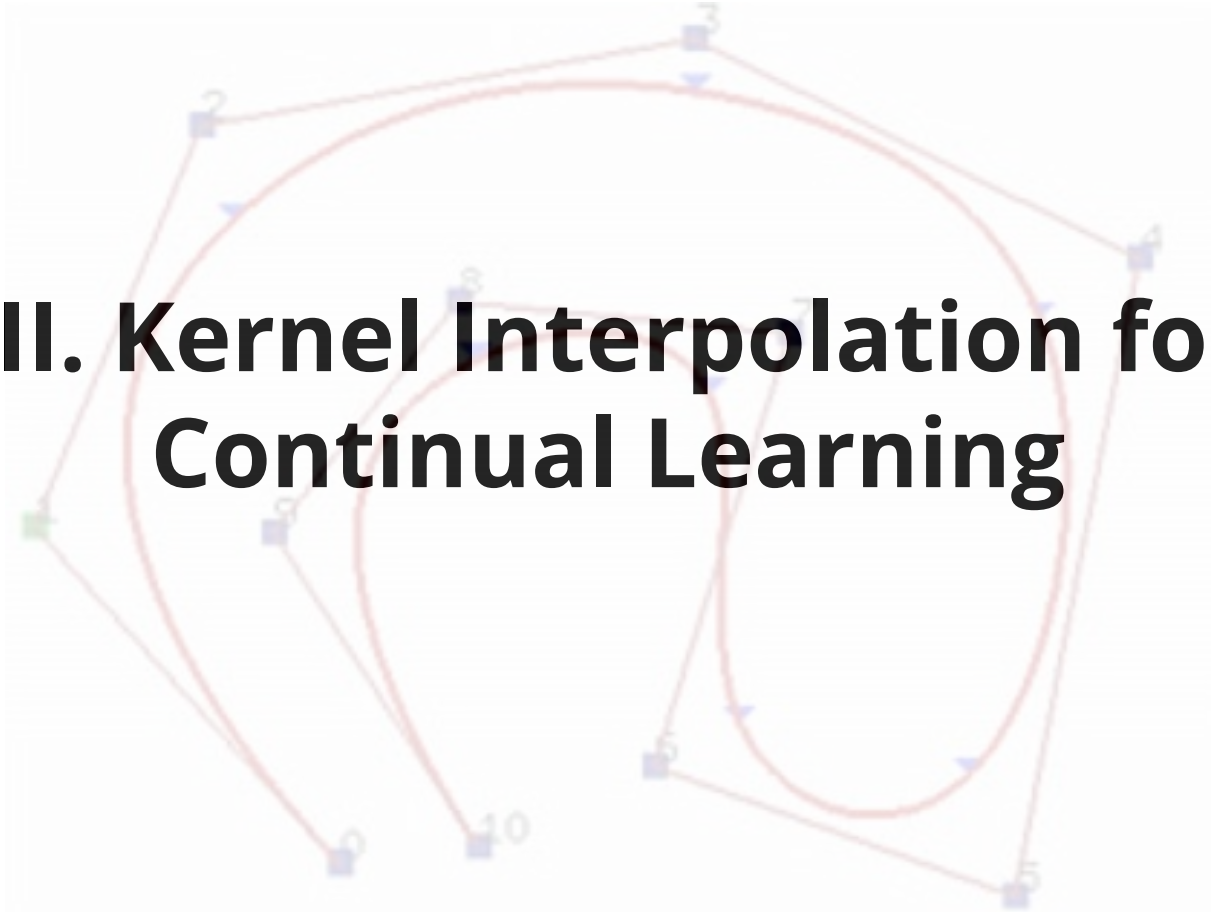
In general, **z** can be a subset of the data or pseudo-inputs optimized through the approximate MLL or the variational ELBO.

Subset of Regressors

$$\tilde{\mathbf{K}}_{XX} := \mathbf{K}_{XU}\mathbf{K}_{UU}^{-1}\mathbf{K}_{UX}$$

- Reduces complexity to **$O(m^2n + m^3)$** (Silverman 1985), can be combined with SVI (Hensman et al 2013).
- **K_{XU}** depends on the kernel hyperparameters.

III. Kernel Interpolation for Continual Learning



Structured Kernel Interpolation

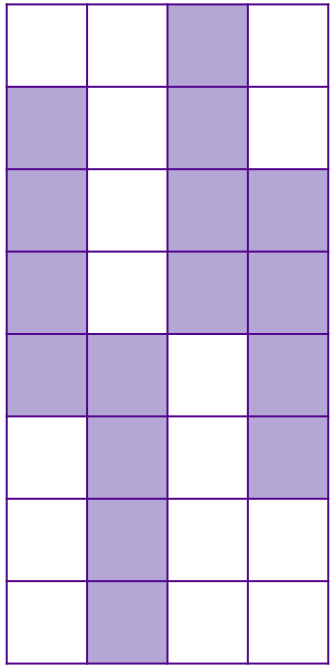
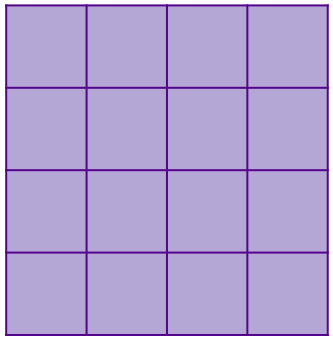
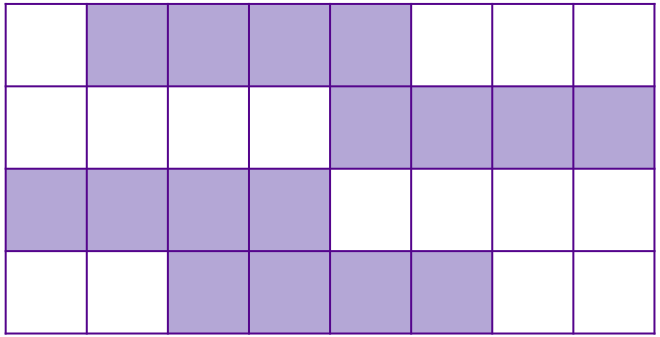
$$\mathbf{W} \in \mathbb{R}^{n \times m} \qquad \mathbf{K}_{XU} \approx \mathbf{W}\mathbf{K}_{UU}$$

$$\tilde{\mathbf{K}}_{XX} := \mathbf{K}_{XU}\mathbf{K}_{UU}^{-1}\mathbf{K}_{UX}$$

$$\Rightarrow \tilde{\mathbf{K}}_{XX} \approx \mathbf{W}\mathbf{K}_{UU}\mathbf{W}^{\top}$$

- Fixing **z** to a grid (inducing Toeplitz structure on $\mathbf{K}_{UU}$), and relying on local cubic interpolation (inducing sparsity on **W**) reduces complexity to **O(n + m log(m))** (Wilson & Nickisch 2015).
- Crucially, **W** does *not* depend on the kernel hyperparameters.

Structured Kernel Interpolation

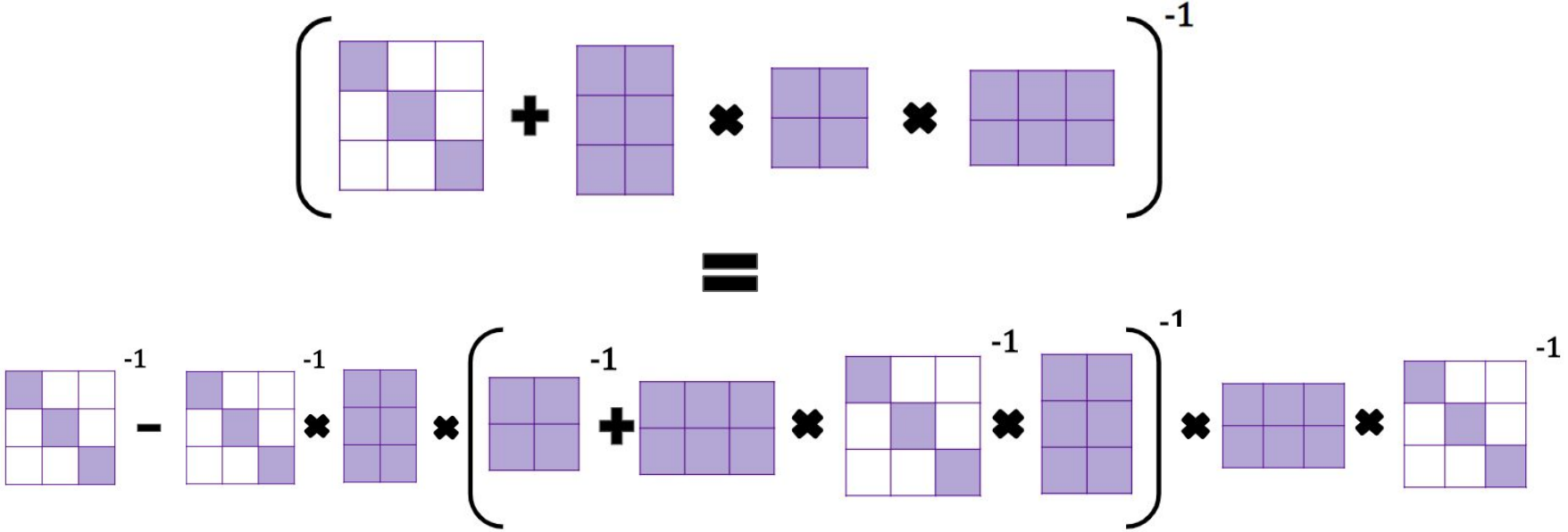
$\tilde{\mathbf{K}}_{XX} =$  $\times$  $\times$ 

Toeplitz

Sparse

$\mathbf{W}$ $\mathbf{K}_{UU}$ $\mathbf{W}^\top$

Woodbury Matrix Identity



$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1}+VA^{-1}U)^{-1}VA^{-1}$$

Woodbury Matrix Identity

Common sparse GP trick: turn a **nxn** system of equations into a **mxm** system.

$$(\mathbf{W}\mathbf{K}_{UU}\mathbf{W}^\top + \sigma^2 I)^{-1}$$

$$\sigma^{-2} I - \sigma^{-4}\mathbf{W}\underbrace{(\mathbf{K}_{UU}^{-1} + \sigma^{-2}\mathbf{W}^\top\mathbf{W})^{-1}}_{\mathbf{M}}\mathbf{W}^\top$$

Learning and Querying the GP

$$\log p(\mathbf{y}|\mathbf{X},\theta) = \frac{1}{2}\left(\sigma^{-2}\mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top\mathbf{W}\mathbf{M}\mathbf{W}^\top\mathbf{y}\right) - \log|\mathbf{K}_{UU}| - \log|\mathbf{M}| + (n-m)\log\sigma^2$$

cached m x m cached m x m m x m

$$\mu(\mathbf{x}^*) = \mathbf{w}_{\mathbf{x}^*}^\top\mathbf{M}\mathbf{W}^\top\mathbf{y}$$

m x m cached

$$\Sigma(\mathbf{x}^*) = \sigma^2\mathbf{w}_{\mathbf{x}^*}^\top\mathbf{M}\mathbf{w}_{\mathbf{x}^*}$$

m x m

Learning and Querying the GP

In the formulation on the last slide, *all* terms directly involving the data are linear with const. memory footprint and no dependence on the kernel.

$$\mathbf{y}_{t+1}^\top \mathbf{y}_{t+1} = \mathbf{y}_t^\top \mathbf{y}_t + y_{t+1}^2 \qquad \mathbf{W}_{t+1}^\top \mathbf{y}_{t+1} = \mathbf{W}_t^\top \mathbf{y}_t + \mathbf{w}_{\mathbf{x}_{t+1}} y_{t+1}$$

We are not summarizing the past observations (as in Bui et al 2017), we are performing *exact* inference with an approximate kernel.

A Closer Look

$$\mathbf{M} := (\mathbf{K}_{UU}^{-1} + \sigma^{-2}\mathbf{W}^\top\mathbf{W})^{-1}$$

But wait...

$$\mathbf{W}^\top\mathbf{W} \in \mathbb{R}^{m\times m}$$

$$\mathbf{W} \in \mathbb{R}^{n\times m}$$

So how does one update **M** in constant time?

A Closer Look

$$\mathbf{M} := (\mathbf{K}_{UU}^{-1} + \sigma^{-2}\mathbf{W}^\top\mathbf{W})^{-1}$$

Let's start with a root decomposition of $\mathbf{W}^\mathsf{T}\mathbf{W}$.

$$\mathbf{W}^\top\mathbf{W} = \mathbf{L}\mathbf{L}^\top$$

L is rank **k <= m**

$$\Rightarrow \mathbf{M} = \mathbf{L}^{-\top}(\mathbf{L}^{-1}\mathbf{K}_{UU}^{-1}\mathbf{L}^{-\top} + \sigma^{-2}\mathbf{I})^{-1}\mathbf{L}^{-1}$$

A Closer Look

Updating the root decomposition

$$\mathbf{W}_{t+1}^\top \mathbf{W}_{t+1} = \mathbf{W}_t^\top \mathbf{W}_t + \mathbf{w}_{\mathbf{x}_{t+1}} \mathbf{w}_{\mathbf{x}_{t+1}}^\top$$

$$\mathbf{W}_t^\top \mathbf{W}_t = \mathbf{L}_t \mathbf{L}_t^\top \qquad\qquad (\mathbf{W}_t^\top \mathbf{W}_t)^{-1} = \mathbf{R}_t \mathbf{R}_t^\top$$

$$\mathbf{L}_{t+1} = \mathbf{L}_t (\mathbf{I} + \mathbf{R}_t^\top \mathbf{w}_{\mathbf{x}_{t+1}} \mathbf{w}_{\mathbf{x}_{t+1}}^\top \mathbf{R}_t)^{1/2}$$

$$\mathbf{R}_{t+1} = \mathbf{R}_t^\top (\mathbf{I} + \mathbf{R}_t^\top \mathbf{w}_{\mathbf{x}_{t+1}} \mathbf{w}_{\mathbf{x}_{t+1}}^\top \mathbf{R}_t)^{-1/2}$$

Summary

Woodbury-**I**nverse **SKI**

Algorithm 1: WISKI

Input: $\theta, \mathbf{y}^\top\mathbf{y}, \mathbf{W}^\top\mathbf{y}, \mathbf{L}, \mathbf{R}$

for $t = 1, 2, \ldots$ **do**

- receive $\mathbf{x}_t$
- predict $\hat{y}_t \sim \hat{p}(y|\mathbf{x}_{1:t}, \mathbf{y}_{1:t-1}, \theta)$ — O(mk)
- receive y_t
- $\mathbf{y}^\top\mathbf{y} \leftarrow \mathbf{y}^\top\mathbf{y} + y_t^2$ — O(1)
- $\mathbf{W}^\top\mathbf{y} \leftarrow \mathbf{W}^\top\mathbf{y} + \mathbf{w}_{\mathbf{x}_t} y_t$ — O(m)
- $\mathbf{L} \leftarrow \mathbf{L}\left(\mathbf{I} + \mathbf{R}^\top\mathbf{w}_{\mathbf{x}_t}\mathbf{w}_{\mathbf{x}_t}^\top\mathbf{R}\right)^{1/2}$ — O(mk² + k)
- $\mathbf{R} \leftarrow \mathbf{R}^\top\left(\mathbf{I} + \mathbf{R}^\top\mathbf{w}_{\mathbf{x}_t}\mathbf{w}_{\mathbf{x}_t}^\top\mathbf{R}\right)^{-1/2}$ — O(mk² + k)
- $\theta \leftarrow \theta + \eta\nabla_\theta \log p(\mathbf{y}|\mathbf{x}, \theta)$ — O(m²k)

end

Computational Complexity

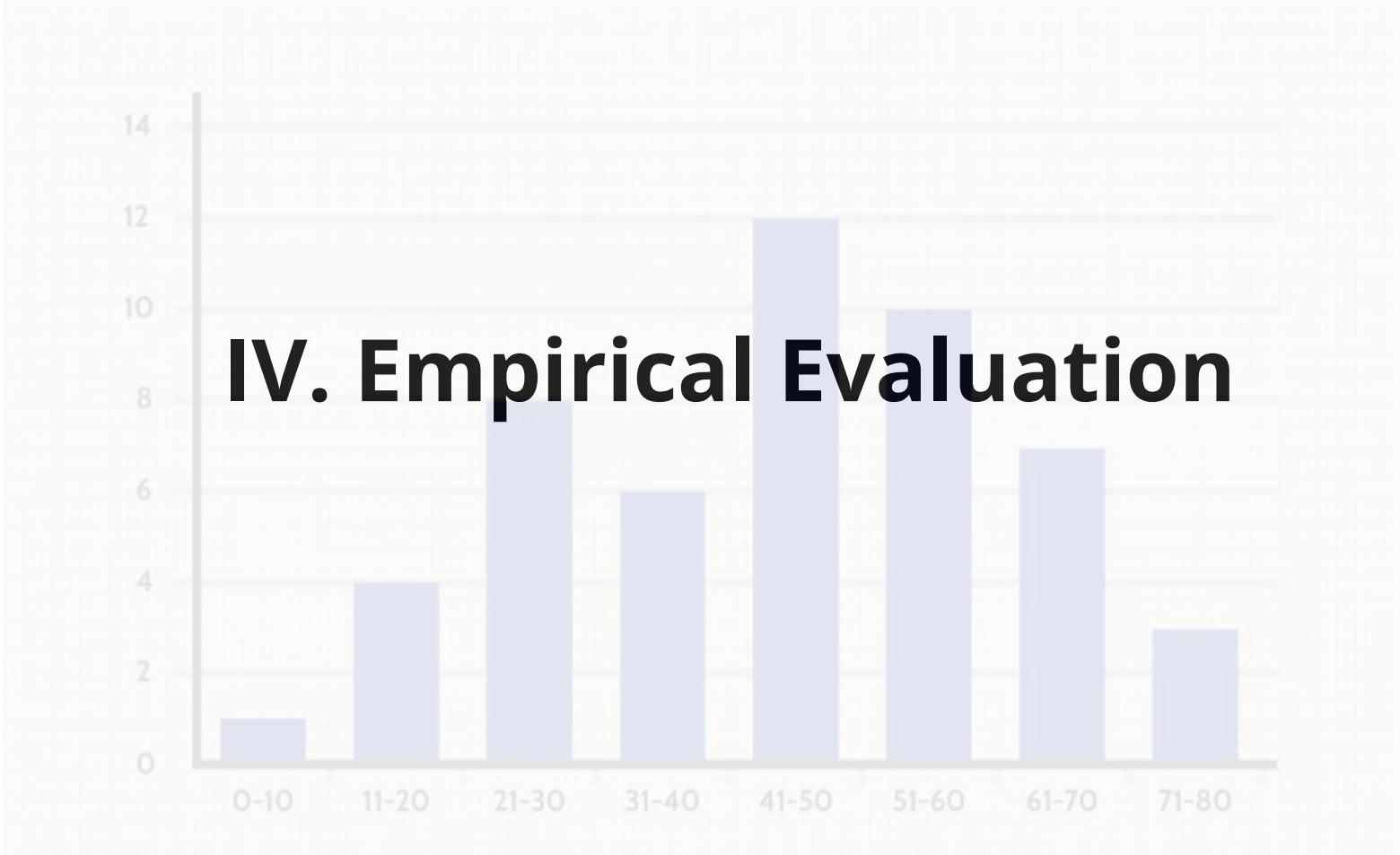
	Conventional GP	Deep Ensemble	WISKI GP
Parameter Inference	**$O(N^3)$**	**O(N)**	**O(1)**
Condition on New Data	**$O(N^2)$**	**O(N)**	**O(1)**
Query Test Point	**$O(N^2)$**	**O(1)**	**O(1)**
Data Storage	**O(N)**	**O(N)**	**O(1)**

Here we are focusing on the complexity in terms of **N**. Just as the computational cost of a deep ensemble depends on the number, width and depth of the components, so does the cost of a WISKI model depend on design choices like the number of inducing points.

Handling High-Dimensional Inputs

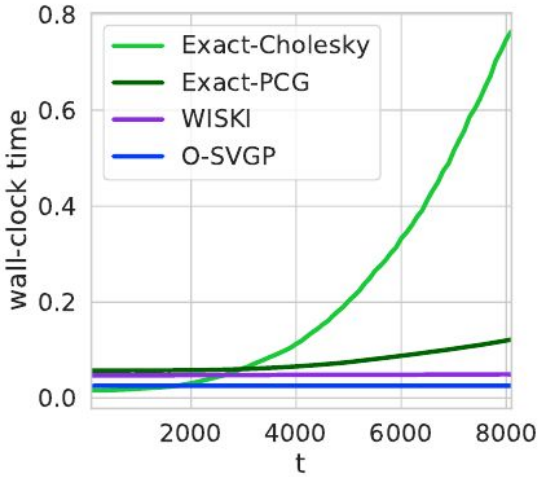
- SKI treats **W** as a sparse, deterministic matrix.
- For best results, you need large **m,** which is usually accommodated by placing **z** on a regular grid, incurring **$O(m^d)$** memory cost.
- For any problem with **d > 2**, we learned a linear projection.
- The projection is learned through the MLL, but one must assume that the projections of previous observations are fixed.

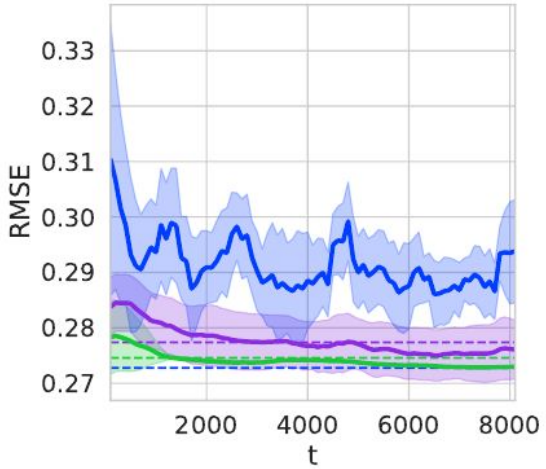
IV. Empirical Evaluation



Constant Runtime... Without VI!

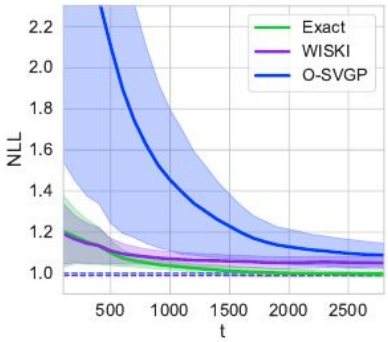
Previous work uses either a sliding window or some summary of previous data. WISKI is just as fast, and is analytically equivalent to the batch model.



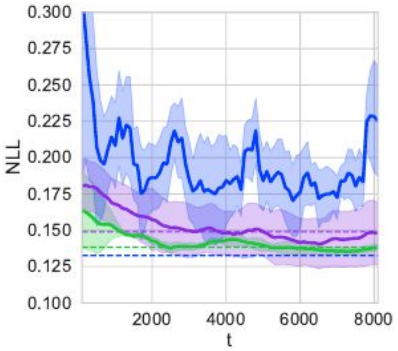


UCI Regression

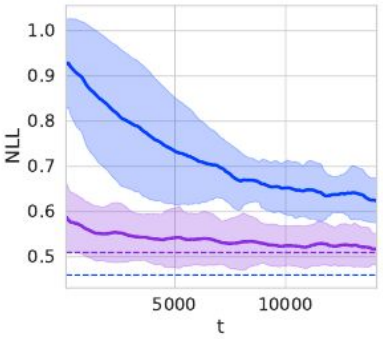
First some standard regression benchmarks,



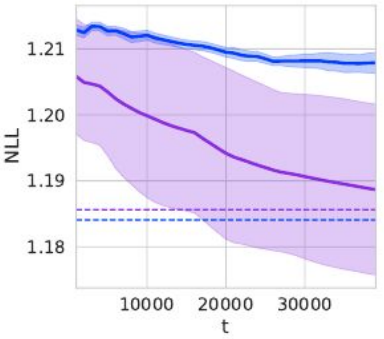
(a) Skillcraft (N=2855)



(b) Powerplant (N=8182)



(c) Elevators



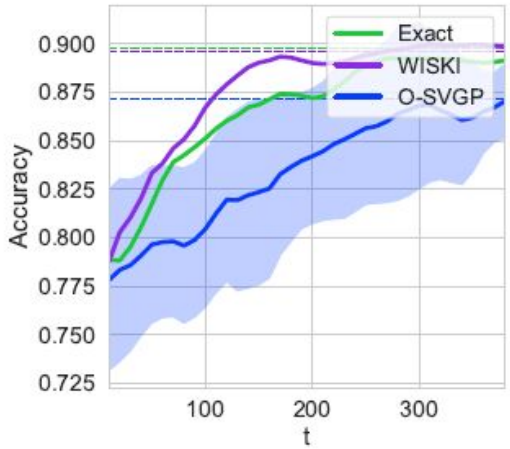
(d) Protein

Classification

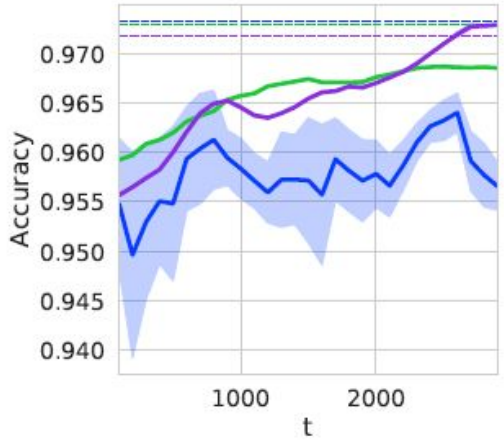
- So far we've assumed a Gaussian likelihood... what if you want to do classification?
- Gaussian Process Dirichlet classification turns classification into a heteroskedastic regression problem (Milios et al 2018).
- WISKI is easily extensible to heteroskedastic regression.

Classification

GP Dirichlet classification with exact and SKI kernels, compared to a standard variational GP classification model with a softmax likelihood.



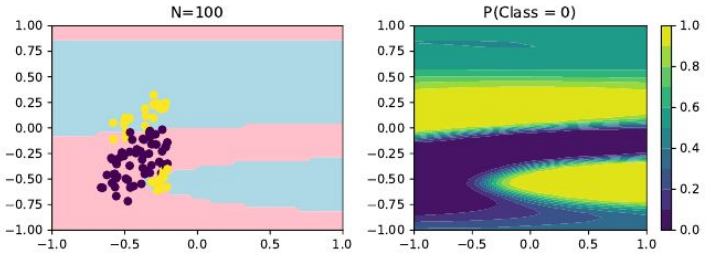
(a) Banana ($n = 400$)



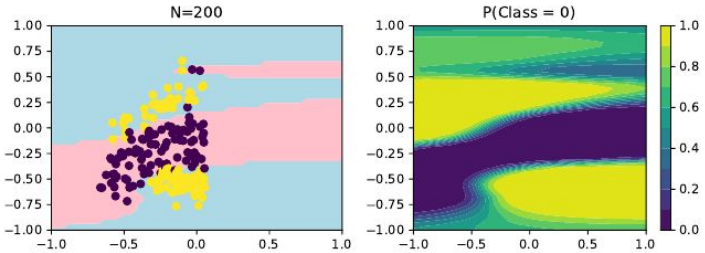
(b) SVM Guide 1 ($n = 3000$)

Classification

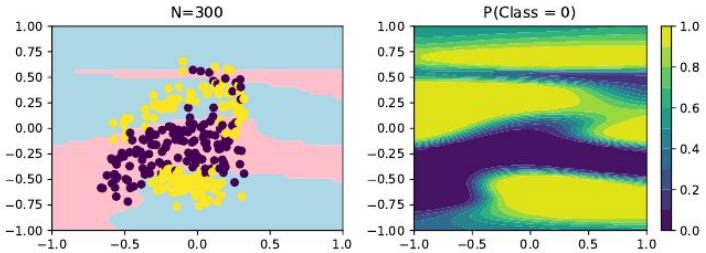
Visualizing a WISKI GP classifier on non-I.I.D. observations.



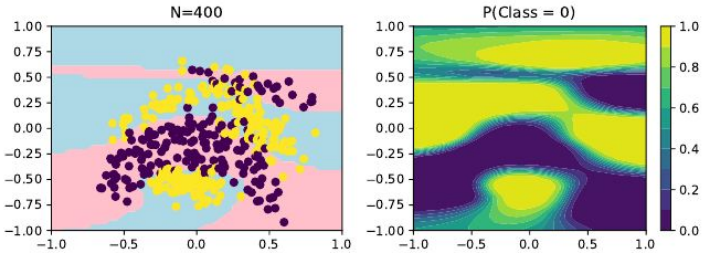
(a) Test Accuracy - 70%



(b) Test Accuracy - 70%



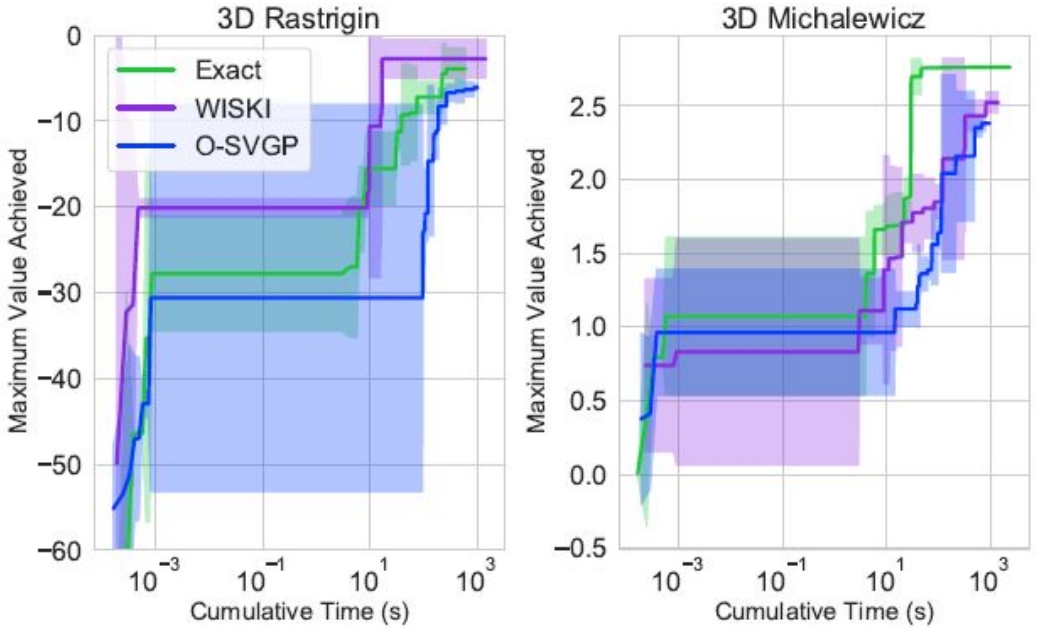
(c) Test Accuracy - 77%



(d) Test Accuracy - 88%

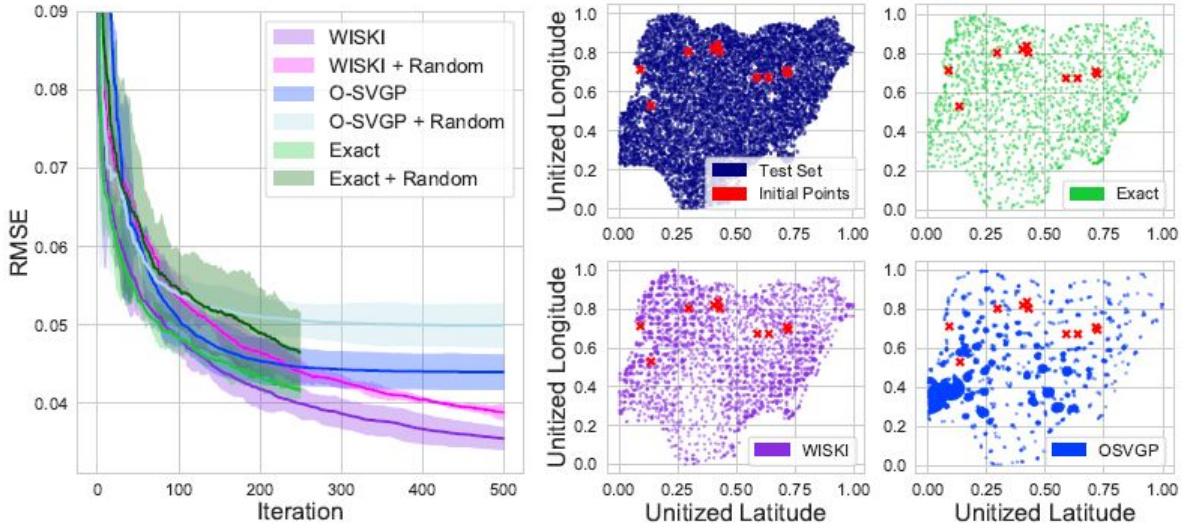
Bayesian Optimization

Results on synthetic objective functions. Even in the setting where an exact kernel is easily tractable, WISKI performs competitively.



Active Learning

Large-scale active learning with WISKI on a dataset from the Malaria Global Atlas. Eliminating the intractable scaling of online inference with exact kernels opens up new possibilities.



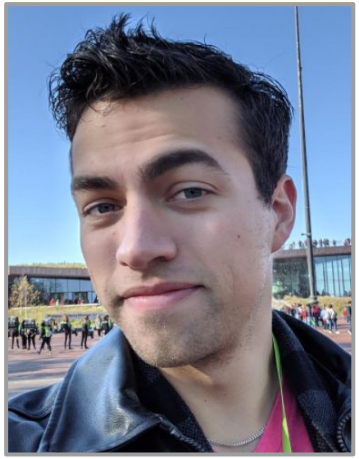
Discussion

- We haven't exploited the algebraic structure of $\mathbf{K}_{\mathbf{UU}}$ or the sparsity of $\mathbf{W}$. Is there induced structure in $\mathbf{W}^{\mathbf{T}}\mathbf{W}$ that we can use?

- Conventional applications of Bayesian optimization or active learning have tended to focus on the small-data regime. Are there impactful applications that have been heretofore intractable?

- Unsupervised learning may be a good way to initialize low-dimensional feature representations for a WISKI model.

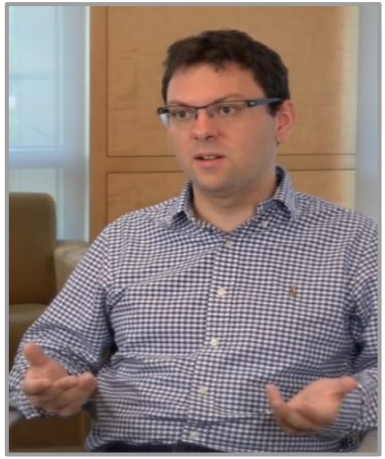
Collaborators



Samuel Stanton
NYU CDS

Wesley Maddox
NYU CDS

Ian Delbridge
Klaviyo



Andrew G. Wilson
NYU CDS/Courant

Thanks for listening!

 @samscub

 ss13641@nyu.edu

References

- Bui, T. D., Nguyen, C., & Turner, R. E. (2017). Streaming sparse Gaussian process approximations. In Advances in Neural Information Processing Systems (pp. 3299-3307).

- Cheng, C. A., & Boots, B. (2016). Incremental variational sparse Gaussian process regression. In Advances in Neural Information Processing Systems (pp. 4410-4418).

- Csató, L., & Opper, M. (2002). Sparse on-line Gaussian processes. Neural computation, 14(3), 641-668.

- Girard, A., Rasmussen, C., Candela, J. Q., & Murray-Smith, R. (2002). Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. Advances in neural information processing systems, 15, 545-552.

- Hensman, J., Fusi, N., & Lawrence, N. D. (2013). Gaussian processes for big data. arXiv preprint arXiv:1309.6835.

- Milios, D., Camoriano, R., Michiardi, P., Rosasco, L., & Filippone, M. (2018). Dirichlet-based Gaussian processes for large-scale calibrated classification. In Advances in Neural Information Processing Systems (pp. 6005-6015).

References

- Pleiss, G., Jankowiak, M., Eriksson, D., Damle, A., & Gardner, J. (2020). Fast matrix square roots with applications to Gaussian processes and Bayesian optimization. Advances in Neural Information Processing Systems, 33.

- Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. Journal of the Royal Statistical Society: Series B (Methodological), 47(1), 1-21.

- Snelson, E., & Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In Advances in neural information processing systems (pp. 1257-1264).

- Titsias, M. (2009, April). Variational learning of inducing variables in sparse Gaussian processes. In Artificial Intelligence and Statistics (pp. 567-574).

- Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., & Wilson, A. G. (2019). Exact Gaussian processes on a million data points. In Advances in Neural Information Processing Systems (pp. 14648-14659).

- Wilson, A. G., Dann, C., & Nickisch, H. (2015). Thoughts on massively scalable Gaussian processes. arXiv preprint arXiv:1511.01870.

References

- Wilson, A., & Nickisch, H. (2015, June). Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In International Conference on Machine Learning (pp. 1775-1784).