Active Learning for Efficiently Constructing Surrogate Models

Ross Alexander
Department of Aeronautics and Astronautics
Stanford University
Stanford, CA 94305
rbalexan@stanford.edu

***Abstract*—The design of experiments is the task of designing a one-shot sampling scheme over the input space that will efficiently capture variation in the output space. While a static, dense sampling scheme is often easy to design and execute, for many experiments, the cost of evaluating such a sample set can be expensive, time-consuming, or even impossible. To improve sample efficiency, we examine the sequential design of experiments, where the task consists of alternating between constructing a surrogate model using the current set of samples and identifying the next sample location utilizing information from the surrogate model. We compare three traditional sampling techniques against two active learning techniques on both Gaussian process (GP) and artificial neural network (NN) surrogate models across a variety of one-dimensional test functions. We find that GP surrogates outperform NN surrogates in sequential design of experiments tasks where the number of samples is limited and when prior information about the latent function can be incorporated into the GP.**

I. Introduction

The design of experiments (DOE) is a task that focuses on constructing a sampling scheme that, when sampled, explains the variation of an experiment's output space with respect to its input space. Typically, designers choose to sample according to a dense grid, since it is easy to implement by specifying the upper and lower bounds and resolution of the grid. However, in many technical experimental domains (e.g. hypersonic wind tunnel tests, biological cell assays, particle collider tests, etc.), experiments can be expensive, time-consuming, or impossible to evaluate. Thus, traditional design of experiments utilizing dense sampling schemes is often intractable. In order to make these experiments feasible, we must use a more sample-efficient method to generate our sample set.

Sequential design of experiments generalizes traditional design of experiments where we repeatedly construct a surrogate model using the current set of samples and then identify the next sample location using evaluations of the current surrogate model. The proposed sample location is sampled in the experiment or simulation and then added to the set of evaluated samples. This process repeats until the model is assumed accurate or until a maximum number of samples have been evaluated. This process is depicted in Figure 1.

While it is easy to define an approach for sequential design of experiments tasks, there are many practical challenges in achieving good performance. In particular, the experimental designer must choose a surrogate model that can sufficiently capture the variation in the latent (objective) function in unobserved regions of the input space. Moreover, surrogate models typically require prior knowledge about the behavior

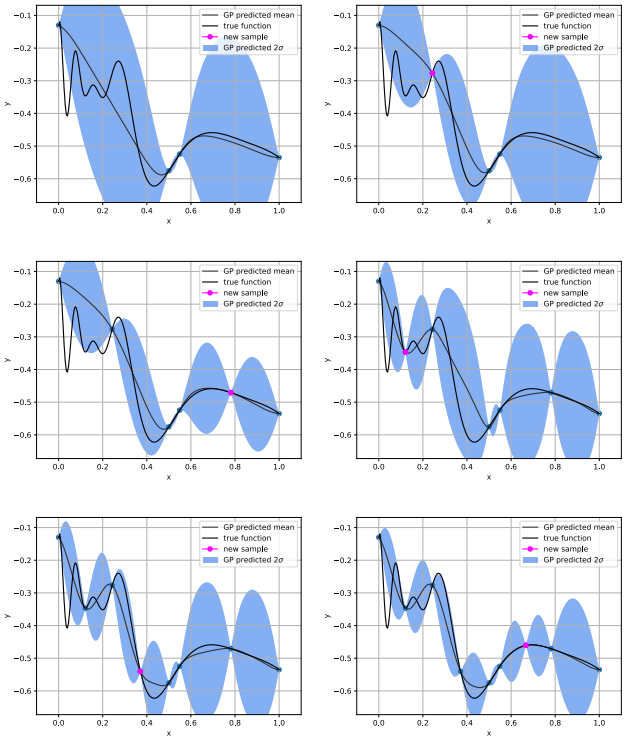

Fig. 1. Six iterations of variance-based active learning on the Hebbal function [1] using a Gaussian process surrogate model with a zero mean function and a rational quadratic covariance function.

of the latent function that may be unavailable. Even more challenging is defining the sequential sampling (active learning) approach so that the surrogate model quickly converges to the latent function. It is important to avoid over-sampling areas of the input space in which there is little variation and also to avoid under-sampling areas of the input space where there is significant nonlinearity.

In this work, we focus our attention on examining the choice of surrogate model, the incorporation of prior information in the surrogate model, and the choice of active learning approach. We examine Gaussian process (GP) surrogates and neural network (NN) surrogates due to the expressivity of these models. Additionally, we study the effects of a variety of covariance (kernel) functions for GP surrogates and the effects of two activation functions for NN surrogates. We initialize a sample set using a face-centered central composite design (CCD) along with a single random sample and then perform active learning

on a series of one-dimensional test functions. We explore five sampling approaches, including random sampling, Sobol sequence sampling, Halton sequence sampling, variance-based sampling, and local linear approximation (LOLA) sampling.

In Section II, we describe several surrogate models and active learning approaches and their related work. We then elaborate on how these are used in our experiments in Section III. We present the results of several active learning approaches across a variety of surrogate models and test functions in Section IV and discuss the key findings as they relate to the selection of surrogate models, incorporation of prior information, and choice of active learning approaches in Section V. In Section VI we describe some potential research directions for future work.

II. Background

The design of experiments is a natural part of every experimental domain and finds use across many disciplines. As a result, a wide variety of approaches to the design of experiments has emerged. The groups can be broadly separated by their choice of sample selection scheme and the choice of surrogate model.

A. Sample Selection Schemes

1) Static Sampling Techniques: Static sampling techniques are essentially one-shot sampling schemes. The most widely known is the full factorial sampling scheme, which gives a dense grid over the input space. Full factorial sampling requires a number of samples that is exponential with the number of dimensions, which can be prohibitive. Another approach called fractional factorial sampling seeks to reduce the sample complexity [2].

Due to the exponential sample complexity of dense factorial sampling plans, sparse sampling plans were developed with better sample complexities. Uniform projection sampling plans sparsely generate samples over a uniform grid such that the projections of the sample set are uniform along each dimension [3]. While uniform projection plans can be useful, they can sometimes fill the input space poorly (consider a diagonal in a two-dimensional grid). To improve on this, some low-discrepancy (space-filling) quasi-random sequences, such as the Sobol sequence and the Halton sequence serve as useful alternatives [4], [5]. Sample sequences from second-order (two-dimensional) Sobol and Halton sequences are shown in Figure 2.

2) Adaptive Sampling Techniques: Also called active learning techniques, adaptive sampling techniques identify the next sample based on a metric computed over the surrogate model. For Gaussian process surrogates with predicted mean $\hat{\mu}(\boldsymbol{x})$ and predicted variance $\hat{\sigma}^2(\boldsymbol{x})$, one convenient technique is variance-based active learning, where the next sample is identified as the sample location with the largest predicted variance.

$$\boldsymbol{x}^{(m+1)} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \hat{\sigma}^2(\boldsymbol{x}) \tag{1}$$

While the above approach is exclusive to Gaussian processes, many other techniques can be used for non-probabilistic models and probabilistic models alike. Another active learning is the

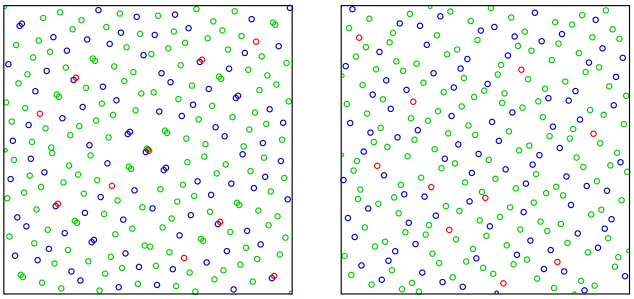

Fig. 2. Plots of the first 256 samples of the 2,3 Sobol sequence (left) and the 2,3 Halton sequence (right) [6], [7].

local linear approximation (LOLA) sampling [8]. We discuss only the one-dimensional case and invite the interested reader to review the higher-dimensional cases in the listed reference. In essence, LOLA identifies samples that would have the largest error between the midpoint of a linear approximation generated using neighboring samples and the surrogate model's prediction at that midpoint. Assuming a sorted sample set where $\bar{\boldsymbol{x}}^{(i,i+1)} = (\boldsymbol{x}^{(i)} + \boldsymbol{x}^{(i+1)})/2$, and $\bar{y}^{(i,i+1)} = (y^{(i)} + y^{(i+1)})/2$ the next sample is given by:

$$\boldsymbol{x}^{(m+1)} = \arg\max_{\bar{\boldsymbol{x}}^{(i,i+1)}} \left| \bar{y}^{(i,i+1)} - \hat{f}\left(\bar{\boldsymbol{x}}^{(i,i+1)}\right) \right| \quad i \in 1, ..., m-1 \tag{2}$$

Since the surrogate model is in error from a linear approximation, LOLA guides the identification of samples toward predicted nonlinear regions. There are some slight issues with aliasing when the predicted model is close to the linear approximation at the midpoint, which we observe and discuss later in Section IV.

Some other techniques for active learning involve using k-fold or leave-one-out cross-validation (KFCV, LOOCV) where samples are selected in the neighborhood of points in the fold with large cross-validation error [9] and other techniques that take a Lipschitzian approach [10].

B. Surrogate Models

There are many choices for surrogate models, though some important classes include linear functions, rational functions [11], radial basis functions (RBFs) [12], Gaussian processes (GPs) [13], [14], and artificial neural networks (NNs) [15]. Herein, we will discuss Gaussian process and neural network surrogates as these models have demonstrated success in modeling broad classes of functions. A detailed discussion of Gaussian processes in given in [16].

1) Gaussian Processes (GPs): We start with a dataset of m input-output pairs that are stored in matrices $\mathbf{X} \in \mathbb{R}^{m\times d}$ and $\mathbf{y} \in \mathbb{R}^m$ respectively. We then seek to develop an estimator for a latent function $f(\mathbf{x})$ that would generate the outputs $\mathbf{y}$ given the inputs $\mathbf{X}$. We model the data generation process as

$$y^{(i)} = f(\boldsymbol{x}^{(i)}) + \epsilon \qquad \text{where } \epsilon \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}) \tag{3}$$

and we model the latent function as

$$f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}')) \tag{4}$$

assuming a zero mean function and a specific covariance function k.

Using the properties of Gaussian distributions, the conditional distribution of $\hat{\boldsymbol{y}} \mid \boldsymbol{y}$ can be shown to be Gaussian with the following predicted mean and variance (using $\boldsymbol{K}$ as the kernel matrix between two sets of samples):

$$\hat{\mu}(\boldsymbol{x}) = m(\boldsymbol{x}) + \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}(\boldsymbol{y} - m(\boldsymbol{X})) \quad (5)$$

$$\hat{\sigma}^2(\boldsymbol{x}) = \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x}) \quad (6)$$

We can fit Gaussian process parameters (mean function and kernel function parameters) using maximum likelihood estimation (MLE) with gradient ascent.

While Gaussian processes tend to work well on small datasets and can give probabilistic estimates of the underlying function, they also require modeling assumptions about the mean and covariance of the underlying function. In the case of the design of experiments, it is worthwhile to consider an approach where we may not have prior information about the behavior of the latent function.

2) Neural Networks (NNs): Given a similar dataset of m input-output pairs that are stored in matrices $\mathbf{X} \in \mathbb{R}^{m \times d}$ and $\mathbf{y} \in \mathbb{R}^m$ respectively, we seek to develop an estimator for a latent function $f(\mathbf{x})$ that would generate the outputs $\mathbf{y}$ given the inputs $\mathbf{X}$. Our estimator will be a single-layer neural network with weights and biases of the ith layer as $W^{[i]}$ and $b^{[i]}$ and the activation function σ:

$$\hat{y} = W^{[2]}a^{[1]} + b^{[2]} \quad (7)$$

$$a^{[1]} = \sigma(W^{[1]}x + b^{[1]}) \quad (8)$$

We train the neural network by minimizing the mean squared error between the predicted outputs and the true outputs through gradient descent.

In comparison to GPs, neural networks benefit from a minimal need for prior knowledge about the latent function. However, neural networks typically require large datasets to learn suitable features.

III. Proposed Approach

We explore several active learning approaches on various test functions, surrogate models, initial sample schemes, and sequential active learning approaches.

A. Test Functions

We consider six different one-dimensional test functions (Table I) due to the simplicity of implementing active learning approaches in one dimension. Future work could include extending some of the active learning methods to two or more dimensions.

The first function is the HEBBAL function, which displays heteroscedastic behavior that would be difficult for a GP to learn. The HEBBAL function is a modified version of the Xiong function that was studied by Hebbal et al. [1]. The second and third problems (PROBLEM15 and PROBLEM20) are two homoscedastic optimization test functions with smooth behavior [17]. Finally, the last three functions are test function we chose to study due to the contrast they provide with the other functions. The SINC function is highly oscillatory, with a large peak near the origin, and an altered domain (SINCSHIFTED) could change the perceived covariance. The STEP function is purely pathological but worth studying due to the discontinuity.

B. Surrogate Models

We examine both Gaussian process (GP) and neural network (NN) surrogate models due to their expressivity. The GPs are constructed, fit, and evaluated using scikit-learn [18] while the NNs are constructed, trained, and evaluated using Keras [19].

1) Gaussian Process Hyperparameters: Using the dataset of input-output pairs, we fit a Gaussian process by minimizing the log-marginal-likelihood of the mean function and covariance function parameters given the dataset. For the minimization, we use the limited-memory Broyden-Fletcher-Goldfarb-Shanno with bound constraints optimizer (L-BFGS-B).

While GPs are non-parametric estimators, they require specifying a prior over the GP, given by the mean function $m(\boldsymbol{x})$ and the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$. The choice of mean function and covariance function is non-trivial and can greatly influence the resulting distribution over the latent function. Typically, prior knowledge about the behavior of the latent function over the design space is utilized in constructing the mean function and covariance function.

For convenience, we assume a zero mean function. In our experiments across all active learning methods, we examine a variety of standard, non-composed, isotropic kernel functions. In particular, we examine the constant, dot product, squared exponential (RBF), rational quadratic, and Matérn ($\nu = \{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}\}$) kernel functions, which are listed in Table II, where Γ is the gamma function and K_ν is the Bessel function of order ν.

2) Neural Network Architecture & Hyperparameters: In the language of machine learning, our problem can be framed as a supervised learning problem. Given our dataset, we train a neural network to minimize the mean-squared-error (MSE) of the input-output pairs. We use the adaptive moment estimation (ADAM) algorithm to minimize the loss.

The neural network and optimizer hyperparameters are listed in Table III. We note that the choice of a single-layer shallow neural network rather than a deep neural network is to achieve a fair comparison with the shallow GP. Despite this limitation, it was shown in the universal approximation theorem [20] that a single-layer feed-forward neural network with a finite number of hidden units can approximate continuous functions with up to an arbitrary degree of precision. We select 64 hidden units since it appears to give good function approximations. We also examine both the rectified linear unit (ReLU) [21] and scaled exponential linear unit (SELU) [22] activation functions, depicted in Figure 4 and given in the following equations, where α and λ are scaling parameters for the SELU activation function.

$$\text{ReLU}(x) = \max(0, x) \quad (9)$$

$$\text{SELU}(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geqslant 0 \end{cases} \quad (10)$$

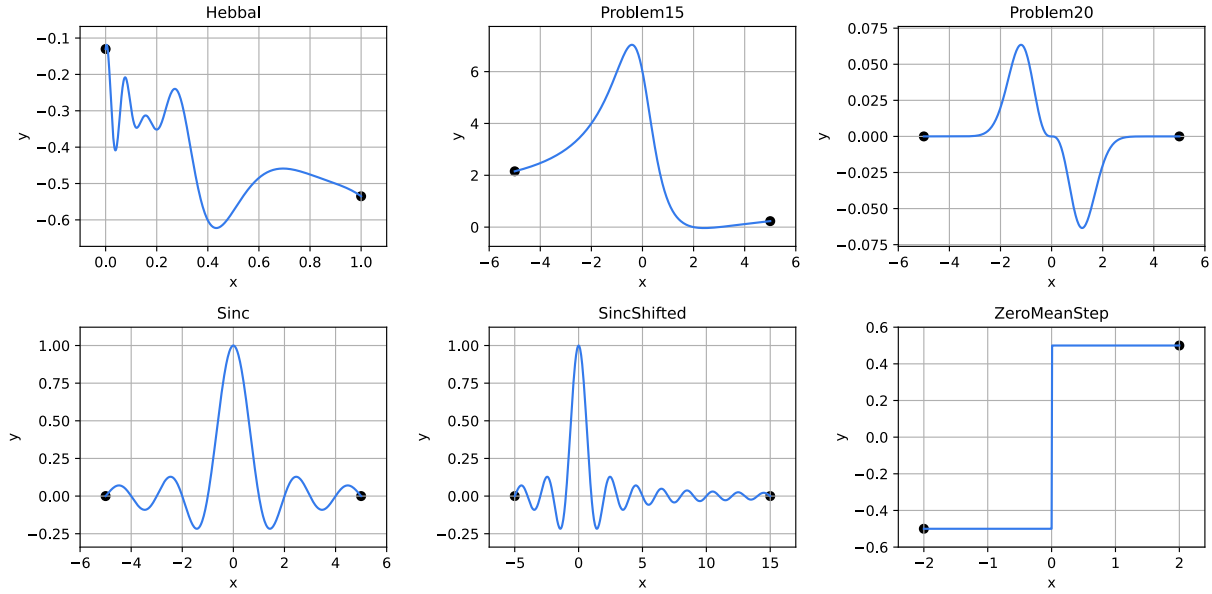

Fig. 3. The six test functions plotted over their domains (Table I).

TABLE I
TEST FUNCTIONS

Name	**Function**	**Domain**
HEBBAL	$f(x) = -0.5 \sin\left(40(x-0.85)^4\right) \cos(2(x-0.95)) + 0.5(x-0.9) + 1$	$x \in [0, 1]$
PROBLEM15	$f(x) = (x^2 - 5x + 6)/(x^2 + 1)$	$x \in [-5, 5]$
PROBLEM20	$f(x) = -(x - \sin(x)) \exp(-x^2)$	$x \in [-5, 5]$
SINC	$f(x) = \text{sinc}(x)$	$x \in [-5, 5]$
SINCSHIFTED	$f(x) = \text{sinc}(x)$	$x \in [-5, 15]$
ZEROMEANSTEP	$f(x) = (x > 0) - 0.5$	$x \in [-2, 2]$

TABLE II
GAUSSIAN PROCESS KERNELS

Kernel	**Equation**	**Parameters**
Constant	$k(\mathbf{x}, \mathbf{x}') = \sigma_0^2$	$\sigma_0^2 \in [0, \infty)$
Dot product	$k(\mathbf{x}, \mathbf{x}') = \sigma_0^2 + \mathbf{x}^T \mathbf{x}'$	$\sigma_0^2 \in [0, \infty)$
Squared exponential (RBF)	$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\|\mathbf{x}-\mathbf{x}'\|\|_2^2}{2\ell^2}\right)$	$\ell \in (0, \infty)$
Rational quadratic	$k(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\|\|\mathbf{x}-\mathbf{x}'\|\|_2^2}{2\alpha\ell^2}\right)^{-\alpha}$	$\ell, \alpha \in (0, \infty)$
Matérn	$k(\mathbf{x}, \mathbf{x}') = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{l}\|\|\mathbf{x}-\mathbf{x}'\|\|_2^2\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{l}\|\|\mathbf{x}-\mathbf{x}'\|\|_2^2\right)$	$\nu \in (0, \infty)$

TABLE III
NEURAL NETWORK AND OPTIMIZER HYPERPARAMETERS

Hyperparameter	Value
Number of layers	1
Number of hidden units	64
Hidden layer activation fns., $\sigma(z)$	ReLU & SELU
First moment decay parameter, β_1	0.9
Second moment decay parameter, β_2	0.999
Stability constant, ϵ	1E-7
Initial learning rate, α_0	1E-2
Learning rate reduction factor	0.5
Learning rate reduction criterion	500 iters. no impr.
Minimum learning rate	1E-5
Early stopping criterion	1500 iters. no impr. $\geqslant$ 1E-6
Maximum number of epochs	10000

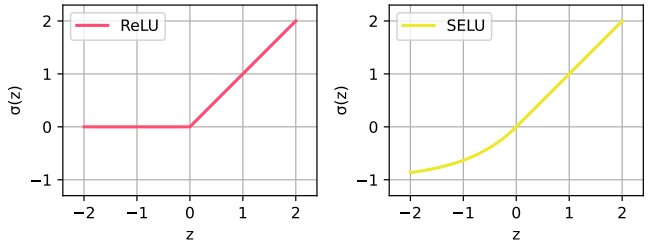

Fig. 4. The rectified linear unit (ReLU) and scaled exponential linear unit (SELU) activation functions (SELU: $\alpha = 1$, $\lambda = 1$). While ReLU is separably linear with straightforward gradients for positive and negative inputs, SELU is nonlinear and has a more complex gradient for negative inputs.

Since our neural network has only a single hidden layer, we simply want to investigate the difference between a piecewise linear activation function and a nonlinear activation function. Providing pure nonlinearity (rather than piecewise linearity) will enable the single-layer neural network to generate a much smoother approximation that varies between training examples when the dataset is small. This variation can be exploited to improve learning and is discussed further in the results. Using both the rectified linear unit (ReLU) and scaled exponential linear unit (SELU) activation functions, we train for a maximum of 10,000 epochs with an initial learning rate of 0.01, which is reduced by a factor of 0.5 if there are no improvements for 500 epochs. Additionally, we stop early if there is no improvement greater than 1E-6 for 1500 epochs.

C. Initial Sampling Scheme

Typically, using a random initialization would make sense, however, this tends to lead to excessive exploration near the boundaries of the design space. To mitigate this, we make use of the face-centered central composite design (CCD, or CCF) sampling scheme, which is essentially a full factorial grid over the design space, but with $m = 3$ samples along each dimension [23]. Once we have obtained these samples, we also add a single random sample to seed the learning of the metamodel. While adding a single random sample is not necessary, it helps us quantify the robustness of active learning approaches.

D. Sequential Sampling Methods

As mentioned in Section II, there are many sequential sampling approaches that we can use. We will test five different sampling approaches including random sampling, Sobol sequence sampling, Halton sequence sampling, variance-based active learning, and local linear approximation (LOLA) active sampling. Neural networks will not be tested on variance-based active learning due to the lack of a variance estimator.

IV. EXPERIMENTS & RESULTS

We run 50 different random initializations for each Gaussian process surrogate and 5 different random initializations for each neural network surrogate. From the initial sample set (CCD + random) with 4 total samples, we sequentially added 50 samples using the specified sequential sampling approach. We computed the approximate integrated squared error (ISE) by using 1001 equally spaced points along the interval to estimate the integral error between the predicted function and the true function.

Since there are a variety of results, many of which are similar, we choose the SINC function to discuss here. As depicted in Figure 5, as we add samples using Gaussian process variance-based active learning, the surrogate model ISE decreases. We observe various levels of performance for each GP kernel function, with the squared exponential (RBF) and rational quadratic (RQ) kernels performing best and quickly converging to low ISE. Other kernels that do not describe the behavior of the function can potentially still make progress, but convergence is slower and is not guaranteed. This reinforces our view that active learning using a GP surrogate without prior knowledge of the mean function or covariance function can bring its own particular set of challenges.

In Figure 6, we compare the GP and NN surrogates on their performance on the SINC function using a random sequence and using LOLA active learning. It is easy to see that the GP nearly

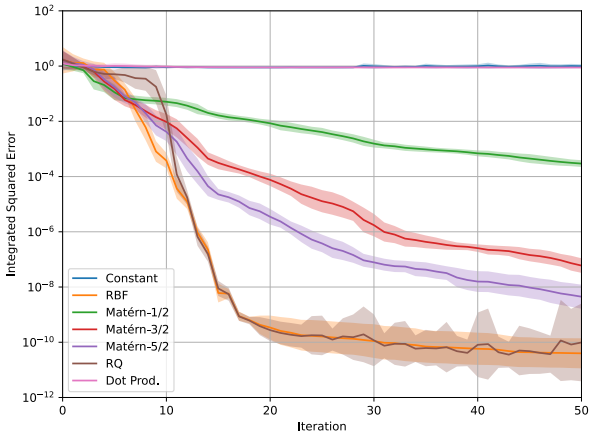

Fig. 5. Integrated squared error of the SINC function for various Gaussian process kernels with $k = 50$ additional samples sequentially selected using variance-based active learning. Mean behavior is computed over $n = 50$ random initializations.

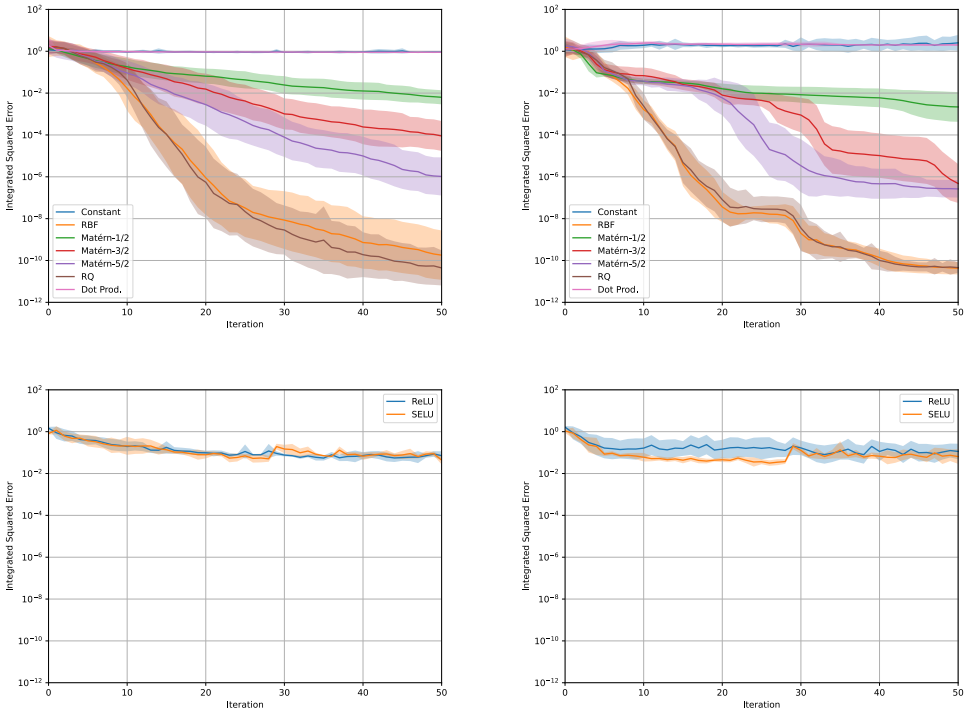

Fig. 6. Integrated squared error of the SINC function for various Gaussian process kernels and activation function with $k = 50$ additional samples sequentially selected using a random sequence (*left*) and LOLA active learning (*right*). Mean behavior is computed over $n = 50$ random initializations for GPs and $n = 5$ random initializations for NNs.

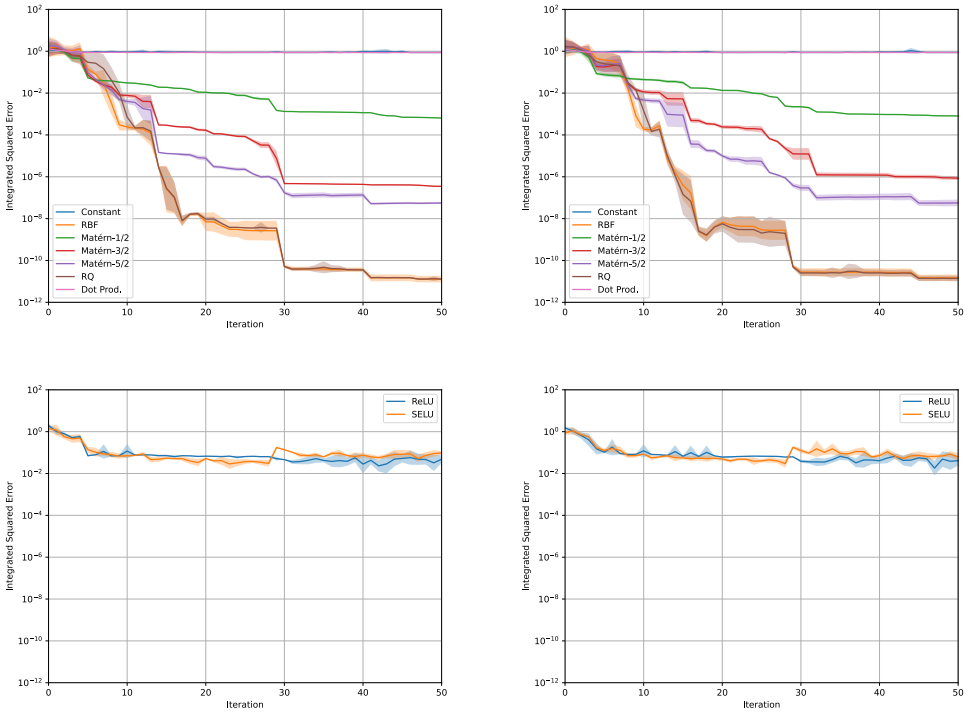

Fig. 7. Integrated squared error of the SINC function for various Gaussian process kernels and activation function with $k = 50$ additional samples sequentially selected using a Sobol sequence (*left*) and a Halton sequence (*right*). Mean behavior is computed over $n = 50$ random initializations for GPs and $n = 5$ random initializations for NNs.

always does better than the NN and that the difference between the ReLU and SELU activation functions is not very significant except for a slight offset in LOLA active learning. We attribute some of the poor performance in the neural network to the non-probabilistic approach which leads to unusual variations in the output space compared to a probabilistic approach. Moreover, the learning rate annealing and early stopping callbacks may have limited our ability to converge to a more optimal neural network. We also observe some interesting and consistent plateaus that can likely be attributed to a stall until obtaining enough samples to learn the proper kernel parameters.

For the ISE for the Sobol and Halton sampling sequences (Figure 7), we see similar behavior, where the final ISE for the NN surrogates using Sobol sequence and Halton sequence sampling is comparable to the final ISE for the NN surrogates using random sequence sampling and LOLA active learning. The sequences give quite consistent performance aside from the random initializations.

V. Conclusion

We have shown that GPs are typically more robust surrogate models than NNs for one-dimensional sequential design of experiments problems. While GPs can achieve much lower integrated squared error (ISE) than NNs, this is largely dependent on the choice of the kernel function, which may not be able to be selected *a priori*. If the mean and kernel function can be reasonably estimated beforehand, a GP surrogate with variance-based active learning appears promising for improving sample efficiency. For the NN surrogates, we see little difference in the choice of activation function, except in the LOLA active learning method. Without further study, NN surrogates do not look promising, which can likely be attributed to the small datasets and non-probabilistic approach.

Overall, we took on an ambitious project and generated a lot of data and there are many more conclusions to be drawn from the data we have at hand. Through this project, I learned a lot about the implementation of Gaussian processes and neural networks and how challenging it can be to get them to work.

VI. Future Work

In the future, it would be interesting to implement the local linear approximation (LOLA) and k-fold cross-validation (KFCV) active learning methods in multiple dimensions to begin to look at the scalability of GPs and NNs. Another important direction for future work is also the extension of the experiments to deep Gaussian processes (DGPs) and deep neural networks (DNNs) [24]. The expressivity and principled composability of DGPs and DNNs may lead to significant improvements over their shallow counterparts in cases when there are sufficiently large datasets. We foresee vast improvement particularly in high-dimensional spaces, where large datasets will be necessary to efficiently construct a surrogate model that can capture the underlying variation in the latent function. An interesting application of ongoing research could be to implement and examine the performance of the neural network Gaussian process (NNGP), which uses a neural network kernel function to model the prediction process of a Bayesian neural network (BNN).

References

[1] A. Hebbal, L. Brevault, M. Balesdent, E.-G. Talbi, and N. Melab, "Bayesian optimization using deep Gaussian processes," *arXiv preprint arXiv:1905.03350*, 2019.

[2] E. P. George, J. S. Hunter, W. G. Hunter, R. Bins, K. Kirlin IV, and D. Carroll, *Statistics for experimenters: design, innovation, and discovery*, 2nd ed. Wiley, 2005.

[3] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*, 2nd ed. MIT Press, 2019.

[4] I. M. Sobol, "On the distribution of points in a cube and the approximate evaluation of integrals," *USSR Computational Mathematics and Mathematical Physics*, 1967.

[5] J. H. Halton, "Algorithm 247: Radical-inverse quasi-random point sequence," *Communications of the ACM*, 1964.

[6] "Sobol sequence." [Online]. Available: https://en.wikipedia.org/wiki/Sobol_sequence

[7] "Halton sequence." [Online]. Available: https://en.wikipedia.org/wiki/Halton_sequence

[8] K. Crombecq, D. Gorissen, D. Deschrijver, and T. Dhaene, "A Novel Hybrid Sequential Design Strategy for Global Surrogate Modeling of Computer Experiments," *SIAM J. Scientific Computing*, vol. 33, pp. 1948–1974, 2011.

[9] A. L. Kaminsky, Y. Wang, K. Pant, W. N. Hashii, and A. Atachbarian, "Adaptive sampling techniques for surrogate modeling to create high-dimension aerodynamic loading response surfaces," in *2018 Applied Aerodynamics Conference*, 2018.

[10] A. Lovison and E. Rigoni, "Adaptive sampling with a Lipschitz criterion for accurate metamodeling," *Communications in Applied and Industrial Mathematics*, 2010.

[11] W. Hendrickx and T. Dhaene, "Sequential design and rational metamodelling," in *Proceedings - Winter Simulation Conference*, 2005.

[12] T. J. Mackman and C. B. Allen, "Investigation of an adaptive sampling method for data interpolation using radial basis functions," *International Journal for Numerical Methods in Engineering*, 2010.

[13] T. W. Simpson, D. K. J. Lin, and W. Chen, "Sampling Strategies for Computer Experiments: Design and Analysis," *International Journal of Reliability and Applications*, 2001.

[14] W. C. Van Beers, "Kriging metamodeling in discrete-event simulation: An overview," in *Proceedings - Winter Simulation Conference*, 2005.

[15] J. Eason and S. Cremaschi, "Adaptive sequential sampling for surrogate model generation with artificial neural networks," *Computers and Chemical Engineering*, 2014.

[16] C. K. I. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT Press, 2006, vol. 2, no. 3.

[17] "1-D Test Functions." [Online]. Available: http://infinity77.net/global_optimization/test_functions_1d.html

[18] "scikit-learn: machine learning in Python." [Online]. Available: https://scikit-learn.org/stable/#

[19] "Keras: the Python deep learning API." [Online]. Available: https://keras.io

[20] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, 1989.

[21] V. Nair and G. E. Hinton, "Rectified linear units improve Restricted Boltzmann machines," in *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010.

[22] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, 2017.

[23] "Central Composite Designs (CCD)." [Online]. Available: https://www.itl.nist.gov/div898/handbook/pri/section3/pri3361.htm

[24] A. C. Damianou and N. D. Lawrence, "Deep Gaussian processes," in *Journal of Machine Learning Research*, 2013.