Bayesian Optimization of Fusion Plasma Model with Distribution Regression

Bob Junyi Zou JZ2819@NYU.EDU
Hongkang Yang HY1194@NYU.EDU

Abstract

To enable nuclear fusion and construct the Tokamak machine, we need to better understand and control the fusion plasma model SUR-BHM. The key is to stabilize the model's flux by manipulating the background density gradient κ. The main challenges to this optimization problem are that the flux, our objective function, is computationally expensive, and that the uncertainty in measurements of κ is not negligible. We model κ as a Gaussian random variable and propose a solution based on a novel combination of Bayesian optimization and distributional regression. In particular, we constructed stationary kernels on the space of probability distributions using the Wasserstein metric and maximum mean discrepancy, and applied non-stationary kernels based on neural networks to the Gaussian parameter space. Experiments indicate that our method is able to discover the global minima efficiently, and given the same computational resources, it consistently out-performs other popular non-gradient based methods such as evolutionary algorithm and grid search.

1. Introduction

Nuclear fusion has been the holy-grail of clean and efficient energy production, and understanding the fusion plasma dynamics inside the Tokamak machine is key to make fusion happen. One of the greatest challenges is to control the edge particle flux, and its time evolution is best modeled by the SUF-BHM model (Qi et al., 2019), which simulates the turbulence interactions governing the flux. The most important parameters of this fusion model is the background density gradient, κ. Our goal is to optimize κ to stabilize the flux.

One challenge is that due to constraints in measurement precision, model parameters cannot be measured exactly and often contain noise. Uncertainty in parameters then leads to uncertainty in the model's output, particle flux. In Section 3.1, we define an objective that captures the output flux's variability, given noisy model parameters.

Specifically, we model the input as a probability distribution of κ instead of a deterministic value. Regression from probability distribution to real-valued output is known as distribution regression. We will broadly cover both stationary and non-stationary kernel regressions. In particular, we will discuss stationary kernels based on Wasserstein metric and maximum mean discrepancy, and non-stationary kernels derived from neural networks.

Another challenge is that the SUF-BHM model is highly expensive to solve numerically, which greatly limits the number of queries to the solver function one can afford. Moreover, as this model is a nonlinear PDE, its value and gradient cannot be solved analytically, so one needs to resort to gradient-free methods. In response, we apply Bayesian optimization (BO), which reduces the amount of queries and is computationally economic. BO is based on Gaussian process (GP) and predicts the optimal places to evaluate the objective using a posterior-based acquisition function. Meanwhile, as a non-parametric model, GP can adapt to the chaotic behavior of the fusion model.

Our experiments on these kernels demonstrated that in general BO is much more efficient than other non-gradient based methods like evolutionary algorithm and grid search in terms of accuracy and running time. We conducted two sets of experiments, one in the 1D setting when the input variance or noise is fixed and the other in the 2D setting for the joint optimization of input mean and variance. In particular, the best performing methods are the Spectral Mixture kernels, which outperform than RBF and Matérn kernels and MMD distances in 1D case, and RBF, ARD, Deep Kernel Learning in 2D case.

2. Related work

Recently machine learning has been increasingly applied to physical models. For instance, Pathak et al. (2017) successfully used deep neural nets to predict the evolution of the Kuramoto-Sivashinsky equation, a PDE known for its chaotic behavior. Similarly, Breen et al. (2019) trained

Final Project Proposal for CSGA-3033-27 Bayesian Machine Learning

a neural network that solves the three-body problem with greater speed and accuracy than any classical algorithm.

Specifically for fusion plasma, previous studies have focused mainly on understanding and improving its theoretical models, but the growing attention to constructing a Tokamak machine is compelling the research community to recognize the importance of optimization as many design decisions cannott be made without knowing the optimal setting of various physical quantities. Majda has been a pioneer in fusion system model reduction and uncertainty quantifications and in fact, the model we used is derived from a more sophisticated two state model proposed by him.(Majda & Qi, 2018)(Qi & Majda, 2019) Our work can be seen as an improvement that combines more sophisticated and competent machine learning methods with fusion models.

One of our innovations is to combine distribution regression with BO, and we believe that this design is the first of its kind. The most relevant work is (Law et al., 2017), which offered a Bayesian treatment of distribution regression, whereas our results indicate that it is beneficial to explore a distributional treatment of Bayesian optimization.

As a quick introduction, the method of BO was proposed by Mockus et al. (1978). It has been applied successfully to the hyperparameter tuning of machine learning (Snoek et al., 2012) and architecture design of neural network (Kandasamy et al., 2018). Regarding distribution regression, it belongs to the increasingly popular field of learning with distributional data (Muandet et al., 2017). A thorough study of distribution regression based on RKHS is given by (Szabó et al., 2015), and it has been applied to causal inference by (Lopez-Paz et al., 2015).

3. Methodology

In this section, we define our objective function, the distribution kernels both stationary and non-stationary, and the acquisition function for BO.

3.1. Input and Output

In practice, the inputs κ cannot be measured with perfect precision and thus are often modeled as random variables. It is reasonable to use Gaussian random variable, as its mean corresponds to the desired parameter value and its variance corresponds to measurement noise. Thus, our input space consists of 1D Gaussian distributions $\mathcal{G}(\mathbb{R}^d)$, and we denote each element by $p = \mathcal{N}(m, \sigma^2)$.

Our output concerns the flux distribution of the fusion model. Given any parameter κ, let $f(\kappa) \in \mathbb{R}$ be the model's flux at statistical equilibrium . Then, given input p, the output is the flux distribution $f\#p$, or equivalently, the distribution of $f(\kappa)$ for $\kappa \sim p$.

Our objective function, the uncertainty of the fusion model, is characterized by the coefficient of variation, which is the ratio between the output flux distribution's mean and standard deviation:

$$F(p) = \frac{\sqrt{Var(f\#p)}}{\mathbb{E}[f\#p]}$$

In practice, it is difficult to limit the input variance of κ, so for the first part of our experiments, we fix the variance σ^2 to the best industry level precision, 0.02^2, and optimize the mean. For the second part of our experiment, we jointly optimize mean and variance. A penalty term, proportional to $-\sigma^2$, is added to F that models the difficulty of reducing noise. It encourage large variances and prevents the degenerate solution of a Dirac mass.

3.2. Stationary kernel for distributions

One way to define a kernel over probability distributions p is to apply a stationary kernel $k(\tau)$ such as RBF, which depends only on pairwise distance, and insert some distance d over probability distributions:

$$\tilde{k}(p_1, p_2) := k\big(d(p_1, p_2)\big)$$

Below we consider two distances with closed form formulas for Gaussians.

3.2.1. Distances on distributions

A natural choice of distance is the Wasserstein metric W_2, the L^2 optimal transport cost between probability measures (Villani, 2008):

$$W_2^2(p_1, p_2) = \inf_{\pi} \int ||x - y||^2 d\pi(x, y)$$

where π ranges over all probabilistic couplings, or transport plans, between p_1, p_2. For Gaussians, the distance can be computed by (Cuesta-Albertos et al., 1996)

$$W_2^2(p_1, p_2) = ||m_1 - m_2||^2 + Tr\big[\Sigma_1 + \Sigma_2 - 2(\Sigma_1^{\frac{1}{2}}\Sigma_2\Sigma_1^{\frac{1}{2}})^{\frac{1}{2}}\big]$$

It has been demonstrated in Arjovsky et al. (2017) that W_2 reliably captures our geometric intuition of distance whereas KL-divergence and total variation fail. In particular, in 1D, the formula reduces to the Euclidean distance in the parameter space.

Our second choice of distance is maximum mean discrepancy (MMD) (Muandet et al., 2017). Given a reproducing kernel Hilbert space (RKHS) H over $\mathbb{R}^d$, we can embed the space of probability distributions into H via the kernel mean embedding

$$p \mapsto \mu_p := \int k(x, \cdot) dp(x)$$

where k is a kernel function over $\mathbb{R}^d$ that characterizes H. Then, MMD is defined by

$$MMD^2(p_1, p_2) = ||\mu_{p_1} - \mu_{p_2}||_H^2$$

It is known that if k is a characteristic kernel such as Gaussian kernel, then the kernel mean embedding is injective (Sriperumbudur et al., 2010), so that MMD becomes a well-defined metric. Hence, we apply the Gaussian kernel $k(x, y) \propto \mathcal{N}(x - y|0, \Sigma_0)$. Then, Appendix A shows that MMD can be computed by

$$MMD^2(p_1, p_2) = \frac{1}{\sqrt{\det(\Sigma_0 + 2\Sigma_1)}} + \frac{1}{\sqrt{\det(\Sigma_0 + 2\Sigma_2)}} - \frac{2e^{-\frac{1}{2}\left[(m_1-m_2)^T(\Sigma_0+\Sigma_1+\Sigma_2)^{-1}(m_1-m_2)\right]}}{\sqrt{\det(\Sigma_0 + \Sigma_1 + \Sigma_2)}}$$

Note that this distance is bounded above.

3.2.2. Choice of kernels

In the following, the RBF kernel is generalized to

$$k(p_1, p_2) = \exp\big[-\frac{1}{2}d^2(p_1, p_2)\big]$$

and the Matén kernel is generalized similarly. The regularity of this form of kernel has been studied in Christmann & Steinwart (2010).

Another well-known stationary kernel is the spectral mixture kernel (Wilson & Adams, 2013), which is based on Fourier dual and can approximate any stationary kernel. Its formula in 1D is given by

$$k(\tau) = \sum_{p=1}^{P} w_p e^{-2\pi^2\tau^2 v_p} \cos(2\pi\tau\mu_p)$$

with weights w_p, length-scale $v_p^{-1/2}$ and period $1/\mu_p$. Since $k(\tau)$ is symmetric in the input τ, we can substitute τ by distance $d(p_1, p_2)$ to define a spectral mixture kernel on probability distributions.

Even though it is not known if this generalized kernel can approximate any stationary kernel on distributions, at least it includes the RBF kernel on distributions as a special case and should be more versatile.

3.3. Non-stationary kernel

Since a stationary kernel which depends only on pairwise distance might be too rigid to capture the complexity of a chaotic model, we consider several non-stationary kernels. Without the need to define distance, we consider the Gaussians by their parameters (m, Σ), and directly define the kernel on the parameter space.

One class of non-stationary kernels is obtained by concatenating stationary kernels with feature functions like neural networks. Notably, the deep kernel learning model (Wilson et al., 2016) has the form

$$k(x, y|w) = k\big(g(x|w) - g(y|w)\big)$$

where $g(\cdot|w)$ is a neural net and k is a stationary kernel. These feature functions often contain trainable parameters like network weights and can be optimized by maximizing the data's marginal likelihood.

Similarly, one can modify the Gibbs kernel (Williams & Rasmussen, 2006) into

$$k(x, y) = \Big(\frac{l(x|w)l(y|w)}{l^2(x|w) + l^2(y|w)}\Big) e^{-\frac{||x-y||^2}{l^2(x|w)+l^2(y|w)}}$$

where the length-scale $l(\cdot|w)$ is modeled by a neural net.

Another class of non-stationary kernels is derived from neural nets. The earliest example is the kernel derived by Williams (1997),

$$k(x, y) = \frac{2}{\pi}\sin^{-1}\Big(\frac{2\tilde{x}\Sigma\tilde{y}}{\sqrt{1 + 2\tilde{x}^T\Sigma\tilde{x}}\sqrt{1 + 2\tilde{y}^T\Sigma\tilde{y}}}\Big)$$

It is derived from an infinite-width two-layer net with random weights (with first layer weights drawn from $\mathcal{N}(0, \Sigma)$). One recent example is the neural tangent kernel (Jacot et al., 2018; Arora et al., 2019a), derived from fully-trained infinite-width nets, with strong performance that matches neural nets (Arora et al., 2019b). The case of two-layer net with ReLU activation yields the kernel

$$k(x, y) = (x^T y)\frac{\pi - \arccos\left(\frac{x^T y}{||x||\cdot||y||}\right)}{2\pi}$$

Recent success in modeling chaos with deep nets (Pathak et al., 2017) indicates that neural nets can capture the nonlinear interactions in the fusion model. Yet, training a neural net for interpolation requires a sizeable sample, which is infeasible due to the high computational cost of our objective. It is reasonable to expect that these kernels that involve neural network shares its expressivity while bein able to cope with small samples.

3.4. Acquisition Function

BO uses an acquisition function based on GP posterior to estimate the global maximizer. The *expected improvement* studied in Frazier (2018) is popular acquisition function

$$EI_n(x) = \mathbb{E}_{\mathcal{N}(y|m(x),\sigma^2(x))}[\max(y - y_n + \xi, 0)]$$

It estimates the expected gain of evaluating at x given the current maximum y_n, while constant ξ indicates the extent

Algorithm 1 Objective Function f

Input: mean m, standard deviation std of κ
Draw N samples $k_1, \ldots, k_N \sim \mathcal{N}(m, std^2)$
Compute flux $y_1, \ldots, y_N$ with $\kappa = k_1, \ldots, k_N$
Compute Monte-Carlo estimator of the flux mean $\bar{y}$
Compute empirical standard deviation σ_y
Return $\sigma_y/\bar{y} \times -100 + \delta_{0j}(std \times 250)$

of exploration of search space and it set to 0.1 in all the experiements as we want to encourage exploration. Its formula is given by (Jones et al., 1998),

$$EI_n(x) = z(x)\Phi\Big(\frac{z(x)}{\sigma(x)}\Big) + \sigma(x)\phi\Big(\frac{z(x)}{\sigma(x)}\Big)$$
$$z(x) = m(x) - y_n - \xi$$

where Φ, ϕ are the cdf and pdf of unit Gaussian.

In general, we find the acquisition's maximum via gradient descent. In 1D, we can also apply grid search, since the GP posterior is efficient to compute.

4. Experiments

The computation of our objective function $F(p)$ is displayed in Algorithm 1. The value of j is 1 when we fix variance and optimize mean and $j = 0$ when we optimize both variance and mean jointly.

4.1. Fixed Input Variance Optimization

First, we demonstrate that BO with distribution regression is capable of discovering the global minima and performs better than other non-gradient search algorithms.

To verify the global minima, we need to recover the entire objective function $F(p)$. Due to the computational expense of F, we restrict to the case when the input distribution of κ has fixed variance ($\sigma^2 = 0.02$) and minimize only the mean m. The objective $F(m)$ as a function of m is recovered by Matlab with brute force and displayed in Figure 1. Note that we multiplied the objective function by a factor of 10^3 to avoid numerical underflow in BO.
We performed BO with the following stationary kernels: Matérn Kernel, RBF Kernel and Spectral Mixture Kernel with Wasserstein distance, and RBF with MMD distance. We also tested the non-stationary kernel, deep kernel learning, introduced in Section 3.3. The implementation details, such as the training of the kernels and the neural network architecture of deep kernel learning, are listed in Appendix B. For comparison, we applied the evolutionary algorithm, grid search and random search.

Table 4.2 lists the minima of the objective discovered by these methods. All methods are tested with a fixed number of steps or queries ($N = 10$), so that a better result indicates that a method is able to discover the minima with greater accuracy and efficiency when given fixed computational resource.

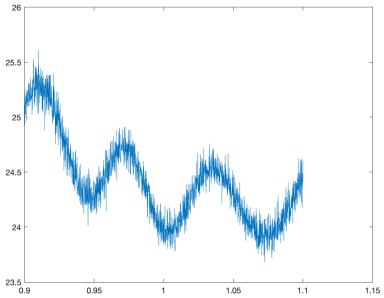

Figure 1. The objective function, evaluated at grid size of 1e-4

METHOD	KERNEL	DISTANCE	RESULT
BO	RBF	W_2	-23.9463
BO	MATÉRN	W_2	-23.8203
BO	SM	W_2	**-23.8021**
BO	RBF	MMD	-23.8743
BO	DEEP	N/A	-23.8911
EA	N/A	N/A	-24.0590
GS	N/A	N/A	-30.4767
RS	N/A	N/A	-32.3046

Table 1. Comparison of BO and other non-gradient based methods. "Deep" stands for deep kernel learning, EA stands for evolutionary algorithm, GS for grid search and RS stands for randomized search.

All runs are initialized with 3 evenly spaced data points and each BO iteration adds one data point at the place suggested by the acquisition. Note that we multiplied the objective F by -1 so that we can keep to a maximization scheme. BO is essentially deterministic because the query points are selected deterministically while we used large samples in Algorithm 1 to eliminate variability. We have repeated the experiments and obtained almost identical results, given the same initialization.

Snapshots of the GP posterior and acquisition function are shown in Fig 2 to 7. On the left plot, the stars are evaluations the objective function at the data points, the blue curve is the posterior mean and the blue shade is the 2-standard-deviation confidence interval. The plot on the right side shows the acquisition function.

4.2. Joint Optimization of Input Mean and Variance

Next, we plan to optimize over both mean and variance of the input distribution with a modified objective function that contains an additional penalty term that encourages large input variance. The initial data are 3 by 3 grid

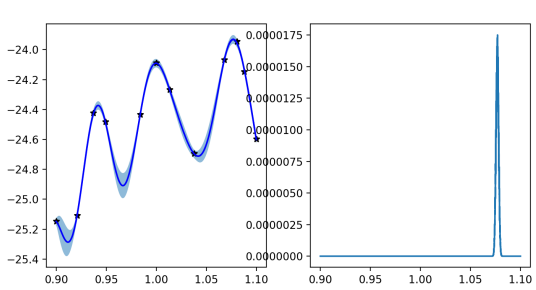

Figure 2. 10th BO Iteration with RBF Kernel.

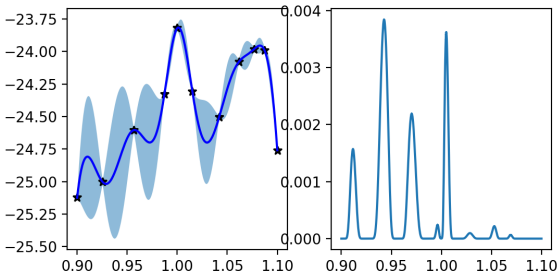

Figure 3. 9th BO Iteration with Matérn Kernel.

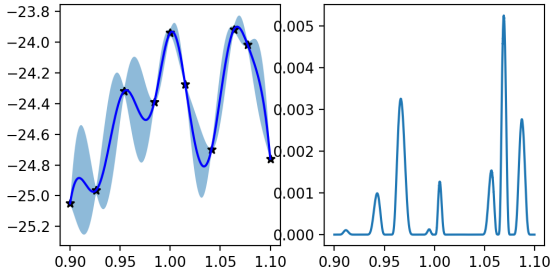

Figure 4. 8th BO Iteration with Matérn Kernel, 2nd run.

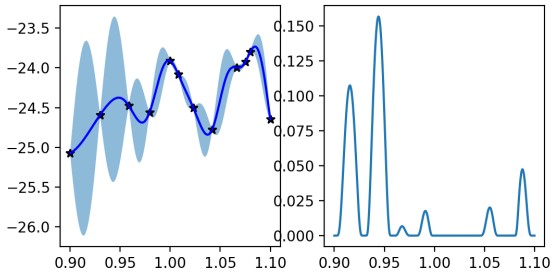

Figure 5. 10th BO Iteration with Spectral Kernel.

over the space $[0.9, 1.1] \times [0.02, 0.05]$, where the first dimension stands for the mean and second for the standard deviation of the input distribution. We used the same BO routine for the joint optimization case, but due to time constraint and curse of dimensionality, we decided to proceed with only the Wasserstein distance with a few interesting kernels. The results are tabulated in table 2. And the snapshots of the posterior mean, posterior confidence range and the acquisition function are given in figure 7 and figure 8

5. Discussion

There are several interesting phenomena in our experiments that contrast the behavior of different algorithms, distances and kernels. First, BO performs better than other non-gradient based algorithms like EA and GS, given equal computational resource. One probable cause is inductive bias. EA essentially assumes that the objective function is multimodal, so at each iteration, it only explores the neighborhood of the current parent population, which may or may not be close to global optima depending on initialization. Similarly, GS and RS has minimal inductive bias, so its exploration of the search space is uninformative. However, the posterior of GP is more concentrated around simple functions than complicated ones, and thus can better fit the true data generating processes, which are often as regular as the curve in Fig 1.

Second, even though the distances W_2 and MMD had similar performance (in Table 4.2), MMD tends to overestimate the posterior variance at unexplored points (e.g. compare Fig 2 and 6). One probable cause is the different asymptotics of the distances. The formula of W_2 indicates that W_2 behaves like the Euclidean distance, such that it grows linearly as we separate the distributions p_1 and p_2. However, MMD has the form $\sim 1 - \exp -(m_1 - m_2)^2$, such that as we separate the means m_1, m_2, it first grows rapidly and then slowly approximates its upper bound. This comparison indicates that Wasserstein distance might be a more reasonable choice for BO, unless the physical model admits some interpretation in terms of RKHS.

Third, different kernels have different confidence levels in their posterior estimates. Comparing Fig 2, 3, and 5, we see that RBF assigns very small confidence interval, followed by the medium intervals of Matérn, and then the large intervals of spectral mixture. The probable reason is that the spectral density of Matérn, which is t-distribution, has heavier tail than that of RBF, which is Gaussian, and thus contains more high frequency components and vary more rapidly. Similarly, spectral mixture kernel can freely adjust its frequency range and allow faster variation whenever needed. Larger confidence interval is particularly helpful if some of our current sample points are close to the true maximizer x_*, for then the acquisition $EI_n(x_*)$ is larger and BO is more likely to inspect x_* than some remote point. To overcome this shortcoming of RBF, one may increase the relaxation constant ξ in EI_n.

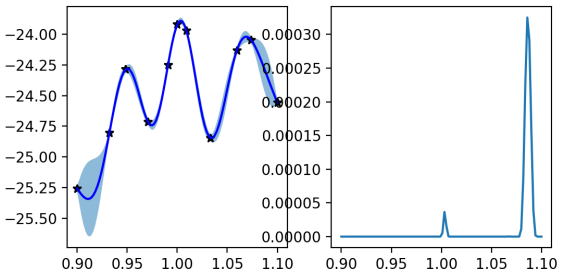

Figure 6. 9th BO Iteration with RBF Kernel and MMD distance.

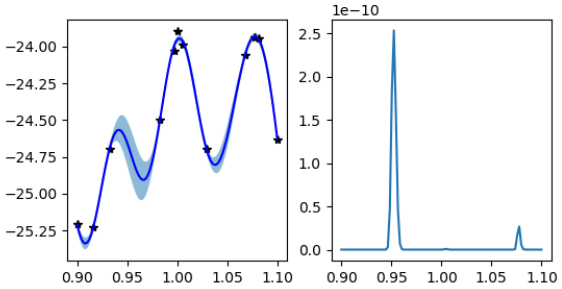

Figure 7. 10th BO Iteration with Deep Kernel Learning.

6. Conclusion and Future Work

Our current experiments have focused on the use of stationary kernels in distributional regression. In 1D we have verified using brute-force recovery that BO can successfully find the global minimum; and specifically, that among stationary kernels, the Spectral Mixture kernels perform the best for its ability to model small scale frequencies in the Fourier space and in 2D, we have also shown that Spectral Mixture performs better than ARD kernels and even non-stationary deep kernel learning. The next steps would be to test more non-stationary kernels, such as arccosine kernel, neural tangent kernel, and non-stationary spectral mixture kernels in BO and check if their versatility boosts their performance and outperforms stationary spectral mixture, and if not we may be inclined to conclude that the data generating process is stationary for our particular fusion model. The winning of either stationary or non-stationary kernels would have interesting implications to the fusion plasma and machine learning communities.

It would also be meaningful to explore how to fully utilize Wassertstein and MMD distances as compared to simple Euclidean distances. We believe that for the particular fusion model Wasserstein distance makes the most sense but we do not have a convincing theoretical justification for the observation. And it would be interesting to explore which regimes are most suitbale for W2 distances and which regimes are most suitable for MMD distance.

KERNEL	DISTANCE	RESULT
RBF	WASSERSTEIN	-18.7355
ARD	WASSERSTEIN	-18.7012
SM	WASSERSTEIN	-18.6405
DKL	WASSERSTEIN	**-18.5837**

Table 2. Comparison of various kernels for joint optimization scheme. "DKL" stands for deep kernel learning

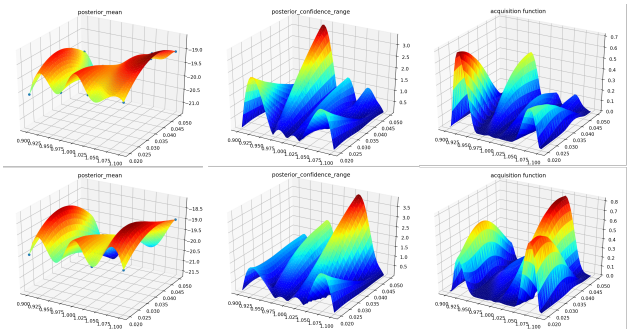

Figure 8. Joint Optimization: RBF Kernel and ARD Kernel

References

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

Arora, Sanjeev, Du, Simon S, Hu, Wei, Li, Zhiyuan, Salakhutdinov, Ruslan, and Wang, Ruosong. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019a.

Arora, Sanjeev, Du, Simon S, Li, Zhiyuan, Salakhutdinov, Ruslan, Wang, Ruosong, and Yu, Dingli. Harnessing the power of infinitely wide deep nets on small-data tasks. *arXiv preprint arXiv:1910.01663*, 2019b.

Breen, Philip G., Foley, Christopher N., Boekholt, Tjarda, and Zwart, Simon Portegies. Newton vs the machine: solving the chaotic three-body problem using deep neural networks, 2019.

Christmann, Andreas and Steinwart, Ingo. Universal kernels on non-standard input spaces. In *Advances in neural information processing systems*, pp. 406–414, 2010.

Cuesta-Albertos, JA, Matrán-Bea, C, and Tuero-Diaz, A. On lower bounds for the L^2 Wasserstein metric in a hilbert space. *Journal of Theoretical Probability*, 9(2): 263–283, 1996.

Frazier, Peter I. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

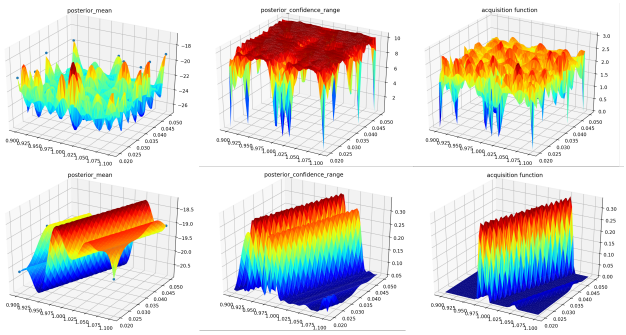

Figure 9. Joint Optimization: Spectral Mixture Kernel and Deep Kernel Learning

Jacot, Arthur, Gabriel, Franck, and Hongler, Clément. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Jones, Donald R, Schonlau, Matthias, and Welch, William J. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13 (4):455–492, 1998.

Kandasamy, Kirthevasan, Neiswanger, Willie, Schneider, Jeff, Poczos, Barnabas, and Xing, Eric P. Neural architecture search with Bayesian optimisation and optimal transport. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2016–2025. Curran Associates, Inc., 2018.

Law, Ho Chung Leon, Sutherland, Dougal J, Sejdinovic, Dino, and Flaxman, Seth. Bayesian approaches to distribution regression. *arXiv preprint arXiv:1705.04293*, 2017.

Lopez-Paz, David, Muandet, Krikamol, Schölkopf, Bernhard, and Tolstikhin, Iliya. Towards a learning theory of cause-effect inference. In *International Conference on Machine Learning*, pp. 1452–1461, 2015.

Majda, Andrew J and Qi, Di. Strategies for reduced-order models for predicting the statistical responses and uncertainty quantification in complex turbulent dynamical systems. *SIAM Review*, 60(3):491–549, 2018.

Mockus, Jonas, Tiesis, Vytautas, and Zilinskas, Antanas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

Muandet, Krikamol, Fukumizu, Kenji, Sriperumbudur, Bharath, Schölkopf, Bernhard, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and in Machine Learning*, 10(1-2):1–141, 2017.

Pathak, Jaideep, Lu, Zhixin, Hunt, Brian R., Girvan, Michelle, and Ott, Edward. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017. doi: 10.1063/1.5010300.

Qi, Di and Majda, Andrew J. Zonal jet creation from secondary instability of drift waves for plasma edge turbulence. *arXiv preprint arXiv:1901.08590*, 2019.

Qi, Di, Majda, Andrew J., and Cerfon, Antoine J. A flux-balanced fluid model for collisional plasma edge turbulence: Numerical simulations with different aspect ratios. *Physics of Plasmas*, 26(8):082303, 2019. doi: 10.1063/1.5083845.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Sriperumbudur, Bharath K, Gretton, Arthur, Fukumizu, Kenji, Schölkopf, Bernhard, and Lanckriet, Gert RG. Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 11 (Apr):1517–1561, 2010.

Szabó, Zoltán, Gretton, Arthur, Póczos, Barnabás, and Sriperumbudur, Bharath. Two-stage sampled learning theory on distributions. In *Artificial Intelligence and Statistics*, pp. 948–957, 2015.

Villani, Cédric. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Williams, Christopher KI. Computing with infinite networks. In *Advances in neural information processing systems*, pp. 295–301, 1997.

Williams, Christopher KI and Rasmussen, Carl Edward. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Wilson, Andrew and Adams, Ryan. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning*, pp. 1067–1075, 2013.

Wilson, Andrew Gordon, Hu, Zhiting, Salakhutdinov, Ruslan, and Xing, Eric P. Deep kernel learning. In *Artificial Intelligence and Statistics*, pp. 370–378, 2016.

Supplementary Material

A. Gaussian kernel MMD

It follows from the reproducing property of RKHS that

$$\begin{aligned}||\mu_p||_H^2 &= \langle \int k(x,\cdot)dp(x), \int k(y,\cdot)dp(y)\rangle_H \\ &= \int \langle x, \int k(y,\cdot)dp(y)\rangle_H dp(x) \\ &= \iint k(x,y)dp(x)dp(y)\end{aligned}$$

Then,

$$\begin{aligned}MMD^2(p_1,p_2) &= k_{11}+k_{22}-2k_{12} \\ k_{ij} &= \iint k(x,y)dp_i(x)dp_j(y)\end{aligned}$$

Note that the Gaussian kernel $k(x,y)$ can be seen as $\mathcal{N}(0|x-y,\Sigma_0)$. It follows that

$$\begin{aligned}k_{ij} =& \iint \mathcal{N}(0|x-y,\Sigma_0)\mathcal{N}(x|m_i,\Sigma_i)\mathcal{N}(y|m_j,\Sigma_j)dxdy \\ =& \mathcal{N}\big(0 \mid m_i-m_j,\ \Sigma_0+\Sigma_i+\Sigma_j\big)\end{aligned}$$

which completes the proof.

B. Implementation details

BO is implemented by GPyTorch, and all kernels are concatenated with a `ScaleKernel` to model a variable output scale.

For deep kernel learning, the base kernel is RBF with ScaleKernel while the neural network g is a fully-connected feedforward net with layer size $d \to 5 \to 1$, where $d = 1, 2$ is the input dimension. All activation functions are ReLU.

At each step, after we obtain a new query, we train the kernel's hyperparameters (outputscale, lengthscale and neural network parameters) using maximal marginal likelihood on the enlarged sample set. We used Adam optimizer with learning rate 0.01.