Automated Kernel Fitting for Gaussian Process Models

Exploring Learned Gaussian Process Kernels for Pattern Discovery and Other Applications

Patrick Di Rita

Computer Science and Engineering
Washington University in St. Louis
December 17, 2020

Automated Kernel Fitting for Gaussian Process Models

Patrick Di Rita

December 17, 2020

Abstract

Over the course of any optimization algorithm, decisions are made under uncertainty at each sequential iteration. "Bayesian optimization" refers to a framework rooted in Bayesian inference which guides said uncertain decisions with respect to some unknown objective function, prior expectations regarding this function, and data acquired over the course of optimization. Bayesian optimization theory relies on a prior model of the objective function which is refined throughout optimization. The prior model is usually nonparametric in order to avoid unwarranted assumptions, and the most popular nonparametric objective model is the Gaussian process (GP). GPs extend the Gaussian distribution by replacing the distribution's mean and covariance parameters with mean and covariance functions. The choice of a GP's covariance function (also known as a kernel) prior to optimization is extremely important. While the mean function is most often chosen to be an arbitrary constant, there are a multitude of different covariance functions which react differently to conditioning on data. It is most often the job of an expert to select the proper kernel (or combination of kernels) in order to properly model complex behavior of the objective.

In this paper, we explore the work of Wilson et al. [1, 2] which aims to reduce the amount of human input required for accurate pattern discovery using Gaussian processes by introducing a a novel, trained kernel: the Spectral Mixture kernel [1]. By attempting to replicate a subset of Wilson's results, we were able to highlight numerous pitfalls in using the Spectral Mixture in practice, which stem from the difficulties of kernel initialization and optimization, and we have shown that the Spectral Mixture kernel regularly converges to undesirable results during the training process due to an abundance of local minima present in the latent space of the kernel hyperparameters. We then proceed to investigate an extension to the Spectral Mixture kernel through the process of Deep Kernel Learning [2], which combines the flexibility of kernel methods with the feature-extracting power of deep neural networks.

1 Introduction

1.1 Bayesian Optimization

Bayesian optimization techniques are most often used when the system under experimentation is not very well understood, which makes identifying a parametric prior distribution on the objective quite difficult. As a result, Bayesian optimization most often uses nonparametric objective function models, the most common of which is the Gaussian process (GP). GPs extend the Gaussian distribution by replacing the finite-dimensional parameters of the distribution (mean and covariance) with analogous functions. As a result, sampling a GP returns a function over the GP's domain, as opposed to a finite value or vector. As we condition the GP on available data throughout the optimization process, the sampled functions become closer to the objective function.

1.2 Gaussian Processes

Acquiring data about an unknown objective function can be quite computationally expensive, and the data is usually acquired in the presence of noise. As such, the amount of data available to make predictions about the objective function can be very limited. In the Bayesian optimization framework, the objective function is treated as a random variable by applying a model such as a Gaussian Process. The one conducting the experiment encodes their prior expectations about the objective function into the model's prior distribution by carefully selecting the GP's mean and covariance functions. The model is conditioned using Bayes' theorem as new data is acquired in order to derive a posterior belief, which includes the prior assumptions and all of the information gained regarding the objective.

Formally, a GP parameterized by mean function

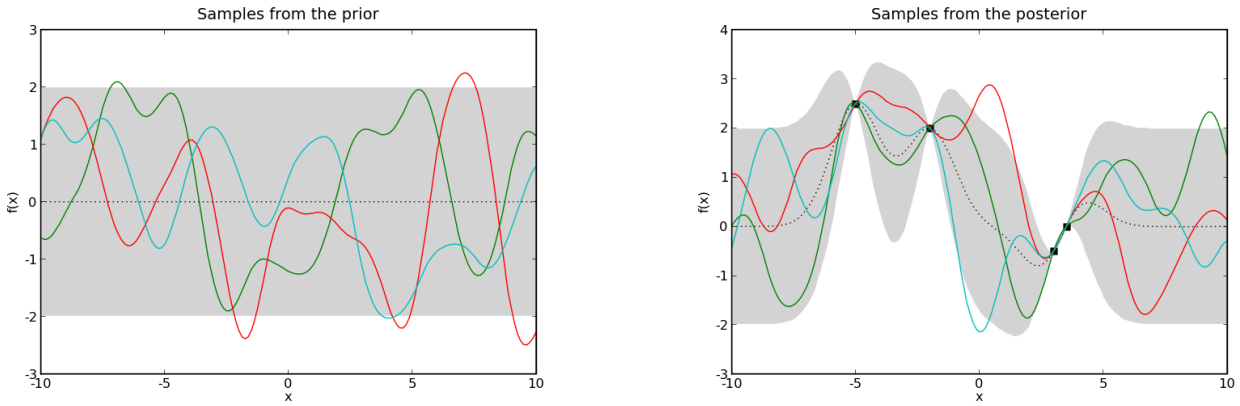

Figure 1: (Left) Samples from a Gaussian process prior before conditioning on any data. (Right) Samples from a Gaussian process posterior after conditioning on 4 datapoints. Key: Solid colored lines - functions sampled from GP; Dashed black line - Mean of the GP; Gray area - Confidence area of GP (2 standard deviations from mean); Black squares - Datapoints (true values of objective function) obtained through observation [3].

$\mu(x)$ and covariance function $k(x, x')$ defines a distribution over functions $f(x)$,

$$f(x) \sim GP(\mu(x), k(x, x')) \tag{1}$$

where x is an arbitrary input variable and the mean and kernel functions are defined as

$$\mu(x) = \mathbb{E}[f(x)] \tag{2}$$
$$k(x, x') = cov(f(x), f(x')) \tag{3}$$

The posterior predictive distribution is inferred by deriving the marginal log likelihood of the observed objective function values $\mathbf{y}$ given kernel hyperparameters and input locations $\mathbf{x}$. The marginal log likelihood of the data is defined as

$$\log p(\mathbf{y}|\mathbf{x}) = -\frac{1}{2}[(\mathbf{y} - \mathbf{m})^{\mathrm{T}}\alpha + 2\Sigma_i \log L_{ii} + n\log 2\pi] \tag{4}$$

Where $\mathbf{m}$ is the mean, α is a conditioning variable on the GP, and L is the Cholesky factorization of the covariance matrix. The marginal log likelihood (mll) is optimized as a function of the kernel hyperparameters using a basic optimizer such as Stochastic Gradient Descent to condition the GP on the observed data. Figure 1 compares a GP prior and its posterior after conditioning on a few datapoints.

1.3 Human Input

The most difficult part of using a GP to infer an unknown objective function is proper selection of the GP's kernel. While common practice is to select a constant or zero mean function, there is a multitude of options to choose from when selecting a kernel function. It is possible to create more complex kernels for specialized applications on a case-by-case basis by composing standard kernel functions. These compositions can change the structure of the GP's samples in unpredictable ways, and as a result, in-depth knowledge about kernel structure and the objective are required in order to properly model the objective. In other words, a significant amount of human input is required to properly encode prior assumptions about the objective into a Gaussian process model by properly selecting or constructing a prior kernel function.

The significance of proper kernel selection is illustrated in Figure 2. The samples from GPs with shared covariance functions are clearly very similar in overall structure, while the samples from the distribution with a different kernel differ substantially from the others.

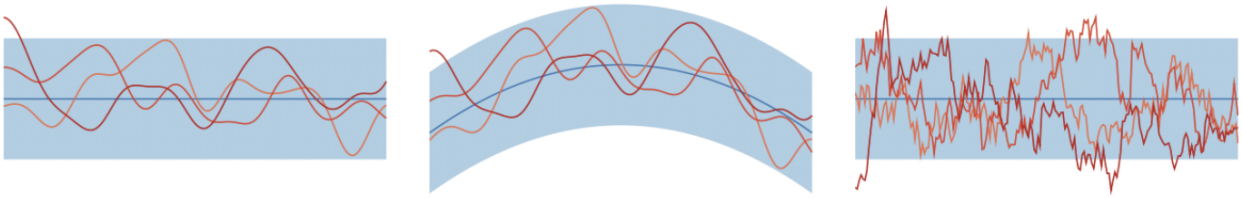

Figure 2: Prior distributions (a), (b), and (c) (left to right) with samples drawn from Gaussian processes. The GPs generating (a) and (b) share a covariance function but have different mean functions, while the GP generating (c) shares a mean function with (a) but has a different covariance function [4].

1.4 The Work of Andrew Gordon Wilson

In his 2013 thesis, Andrew Gordon Wilson et al. proposed the Spectral Mixture (SM) kernel, a simple closed-form kernel derived by modelling a spectral density (the Fourier transform of a kernel) with a Gaussian mixture [1]. Wilson showed that the proposed kernel was able to approximate a wide range of other covariance functions to an arbitrary degree, and demonstrated the power of the SM kernel by discovering patterns in many different types of data much more effectively than was achievable using standard kernels. The SM kernel's approximation characteristic greatly reduces the amount of human input required to model an unknown objective function and extrapolate the appropriate objective structure.

In 2015, Wilson extended the SM kernel by incorporating it into a deep neural network architecture in a process he called "Deep Kernel Learning". The result was a scalable deep kernel which "combines the structural properties of deep learning architectures with the flexibility of kernel methods," [2]. Wilson et al. showed that these closed-form deep kernels can be directly used in the place of standard kernels with little computational overhead, and GPs using these kernels outperform both regular GPs and deep neural networks in a wide range of applications.

2 The Spectral Mixture Kernel

A bit of background information is required before introducing the Spectral Mixture kernel, namely the definition of stationarity and Bochner's theorem. Stationarity is quite simple: any covariance function $K(x, x')$ that depends only on the distance between objective function input locations, $x - x'$, is called stationary and can be rewritten simply as $K(d)$ (where $d = x - x'$). This means that the covariance function has the same structure everywhere (from a statistical perspective), and the local behavior around a single point is enough to specify the global behavior.

Bochner's theorem is a theorem regarding stationary covariance functions. More specifically, it characterizes them in terms of their Fourier transforms:

Theorem 2.1 (Bochner's theorem)**.** *A continuous function $K : \mathbb{R}^d \to \mathbb{R}$ is positive semi definite if and only if*

$$K(\mathbf{x}) = \int e^{2\pi \mathbf{i} x^T \xi} dv$$

Where v is a finite, positive Borel measure on $\mathbb{R}^d$ [5].

In other words, the Fourier transform of any valid covariance function (covariance functions are valid if and only if their corresponding matrices are positive semi-definite) on $\mathbb{R}^d$ is proportional to a probability measure (and vice versa) with constant $K(\mathbf{0})$, and the measure v is known as the spectral measure of K. In the cases when there is a corresponding density function, this function is called the spectral density, κ. The spectral density and the covariance function form a Fourier pair as follows:

$$K(\mathbf{x}) = \int e^{2\pi \mathbf{i} x^T \xi} \kappa(\xi) d\xi \tag{5}$$

$$\kappa(\xi) = \int e^{2\pi \mathbf{i} x^T \xi} \mathbf{K}(\mathbf{x}) dx \tag{6}$$

This result allows us to approximate any stationary covariance function by approximating its spectral density.

The Spectral Mixture (SM) kernel builds off of this idea directly. By applying Bochner's theorem to the Squared Exponential (SE) kernel, $K_{SE}(x, x') = e^{-\frac{(x-x')^2}{2}}$, we get what is known as a Gaussian mixture spectral density. After performing an inverse Fourier transform on the resulting Fourier pair, we are left with the Spectral Mixture kernel [5, 4]:

$$K_{SM}(\mathbf{x}, \mathbf{x}'; \{w_i\}, \{\mu_i\}, \{\mathbf{\Sigma}_i\}) = \sum_i w_i e^{-2\pi^2(\mathbf{x}-\mathbf{x}')^T \Sigma_i (\mathbf{x}-\mathbf{x}')} \cos\left(2\pi(\mathbf{x}-\mathbf{x}')^T \mu_i\right) \quad (7)$$

Where hyperparameters $\{w_i\}, \{\mu_i\}, \{\mathbf{\Sigma}_i\}$ are vectors of the weights, means, and covariances of each component of the mixture, respectively. The kernel is initialized with Q components ($Q = 10$ is commonly used in practice) and initial values of $\{w_i\}, \{\mu_i\}, \{\mathbf{\Sigma}_i\}$ are set based on the initialization criteria. The kernel is then trained on available data as described above (by optimizing the marginal log likelihood as a function of the kernel hyperparameters). Due to its roots in Bochner's theorem, the SM kernel is theoretically able to approximate any desired stationary covariance function, so training the SM kernel is equivalent to discovering the combination of regular covariance functions (and their weights) that best fits the available data.

This ability to approximate any stationary kernel (or combination of kernels) without requiring human input or fine-tuning makes the SM kernel extremely powerful, in theory. The SM kernel can be used to discover patterns in data that would otherwise be impossible (or incredibly difficult) to extract using traditional, human-designed/selected kernels. This is not the fault of the traditional kernels, but rather is the result of some objective functions having structures that make it extremely hard for a human to make any valid prior assumptions given previously observed data.

2.1 Discovering the Sinc Pattern

One such objective function is

$$y(x) = \text{sinc}(x+10) + \text{sinc}(x) + \text{sinc}(x-10) \quad (8)$$

where $\text{sinc}(x) = \sin(\pi x)/(\pi x)$. If the only information available about $y(x)$ is the function values for $x \in [-15, -4.5) \cup (4.5, 15]$ (as seen in Figure 3) then it is extremely difficult for a human to extrapolate the pattern to the missing data, meaning it is even more difficult for a human to design a kernel that accurately captures the pattern's structure.

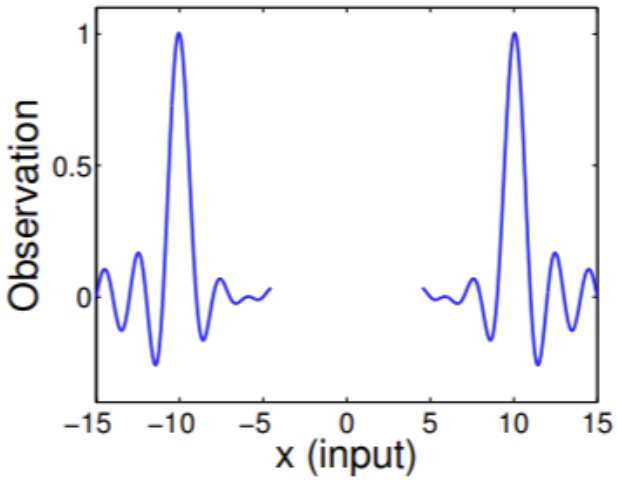

Figure 3: Graph of Equation 8 for $x \in [-15, -4.5) \cup (4.5, 15]$. Used as training data for the SM kernel [1].

In [1], the authors demonstrated the pattern discovering power of the SM kernel relative to other popular kernels (SE, Matern, rational quadratic, and periodic) by training each kernel on the data in Figure 3 and observing the mean of the resultant posterior predictive distributions. The results of this experiment are shown in Figure 4.

Although the other kernels are able to predict the pattern within a range of about 0.5 from the known training data, all perform extremely poorly at subsequent locations. In contrast, the SM kernel (with $Q = 10$ components) was able to almost exactly discover the missing pattern. The authors also noted that only 7 out of the 10 mixture components had non-negligible weights, meaning only 7 components actually contributed to the final kernel.

Wilson et al. conducted other, similar experiments on a few different types of real-world data and measured the mean squared error (MSE) and marginal log likelihood ($\mathcal{L}$) of each kernel for each experiment. Figure 5 contains the results of

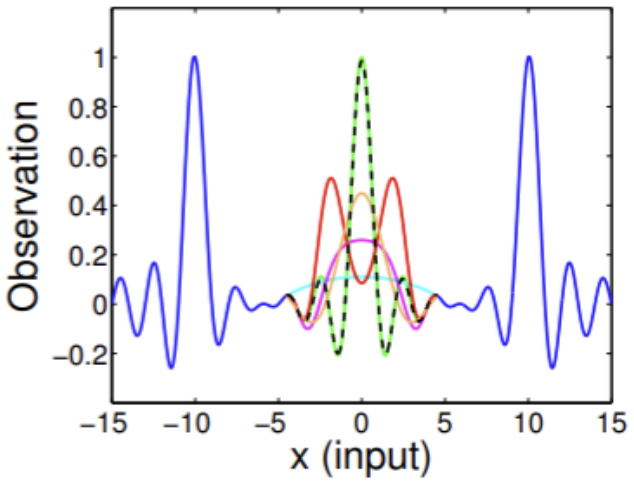

Figure 4: Graph of Equation 8 with means of the posterior predictive distributions using different kernels. Key: Blue - known data; Green - True value of missing portion of Equation 8; Dashed black - Prediction using SM kernel; Red - Prediction using SE kernel; Cyan - Prediction using Matern kernel; Magenta - Prediction using rational quadratic kernel; Orange - Prediction using periodic kernel [1].

the experiments.

	SM	SE	MA	RQ	PE
CO_2					
MSE	**9.5**	1200	1200	980	1200
$\mathcal{L}$	**170**	−320	−240	−100	−1800
NEG COV					
MSE	**62**	210	210	210	210
$\mathcal{L}$	**−25**	−70	−70	−70	−70
SINC					
MSE	**0.000045**	0.16	0.10	0.11	0.05
$\mathcal{L}$	**3900**	2000	1600	2000	600
AIRLINE					
MSE	**460**	43000	37000	4200	46000
$\mathcal{L}$	**−190**	−260	−240	−280	−370

Figure 5: Mean squared error (MSE) and marginal log likelihood ($\mathcal{L}$) of the Spectral Mixture (SM), Squared Exponential (SE), Matern (MA), Rational Quadratic (RQ), and Periodic (PE) kernels across the experiments conducted in [1] (lower MSE is better, higher $\mathcal{L}$ is better). The SM kernel performs better in both metrics for each experiment.

2.2 Implementation and Results

The results in [1] are very impressive, and the paper makes the SM kernel training process seem very straightforward. This perceived simplicity of the processled this author to attempt to replicate Wilson's results for the sinc pattern experiment. Interestingly, it appears that no other secondary sources have replicated the sinc experiment, so our implementation came entirely from the information available in the paper.

We implemented the entire experiment in Python using the GPyTorch library [6]. GPyTorch is a Gaussian process library implemented using PyTorch which allows for significant GPU acceleration and contains many useful functions, models, and state-of-the-art implementations of the latest algorithmic advances for scalability and flexibility [6]. It also just so happens that Andrew Gordon Wilson is one of the main developers for GPyTorch.

We modeled the objective function, training data, and testing data as PyTorch tensors to utilize GPU (CUDA) acceleration, and implemented the exact training and testing data from [1]. For our Gaussian process model, we defined a SpectralMixtureGPModel class (extending the GPyTorch ExactGP class) which has a zero mean function, a Spectral Mixture kernel with $Q = 7$ mixtures, and a Gaussian likelihood. To train the model, we fixed the noise parameter at 0.0001 (as we are dealing with exact observations from a synthetic function), and optimized the marginal log likelihood as a function of the SM kernel's weights, means, and scales (covariances) for 100 iterations using the Adam optimizer (with learning rate of 0.005). Additionally, we forced GPyTorch to directly compute the marginal log likelihood (instead of using Cholesky decomposition) in order to avoid any numerical instabilities. We then evaluated the trained Gaussian process model by predicting the testing data and plotting the prediction alongside the true objective function values.

To our surprise, the predictions generated by our model did not at all match the objective function, even though we achieved a marginal log likelihood that would suggest otherwise. At this point, we shifted our focus to further investigate the discrepancy between our results and those

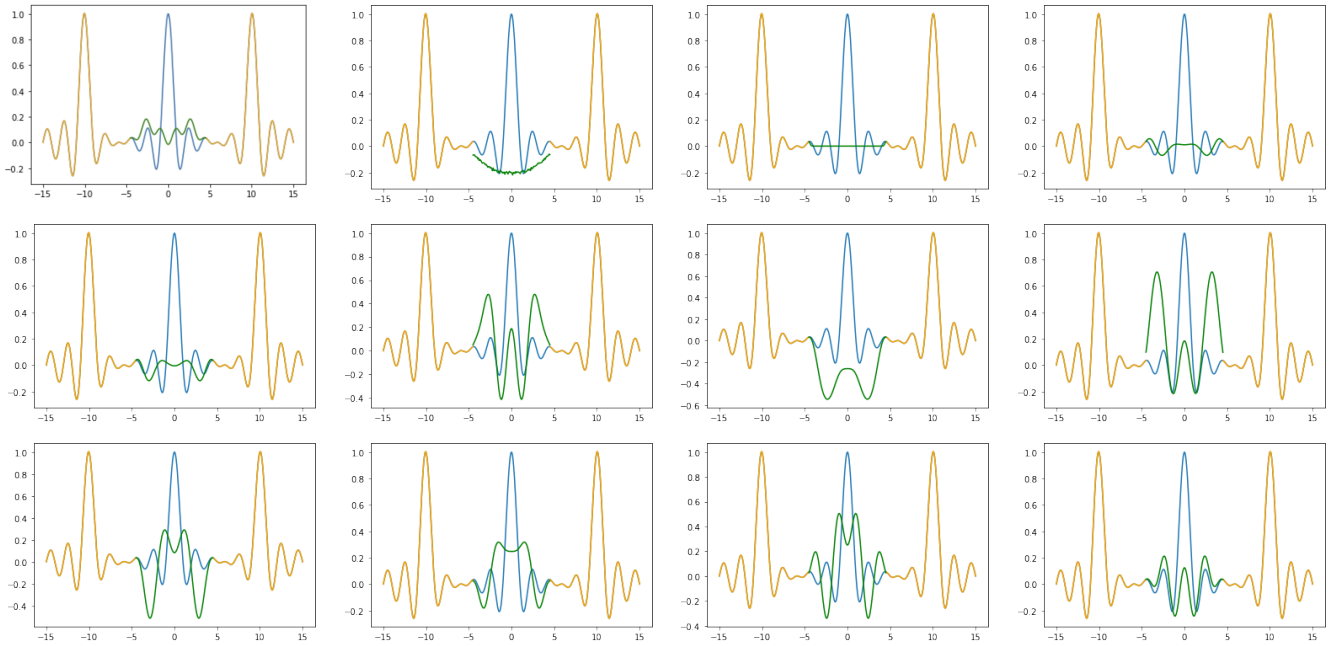

Figure 6: Erroneous model predictions generated during investigative phase by modifying optimizer, optimizer hyperparameters, SM kernel hyperparameter initialization, and trained hyperparameters. Training data, true target data, and predicted data are shown in orange, blue, and green, respectively.

in the original paper, and many hours were devoted to both trying to replicate the original results as well as understand why our model produced highly erroneous predictions. Once we had determined that our implementation of the Spectral Mixture GP was correct, it became clear that the issue stemmed from the initialization and hyperparameter optimization steps. We responded by trying many different optimizers (stochastic gradient descent (SGD), L-BFGS, Adagrad, etc.) with different optimizer hyperparameters (learning rate, weight decay, momentum, etc.), adjusting the set of kernel parameters we were training, adjusting the size of our training dataset, and many combinations of these modifications. Figure 6 contains a subset of the model outputs from this investigative phase. The optimizer converged to each of these erroneous kernels, which indicates a huge number of local minima present in the latent function. Interestingly, the kernel was sometimes observed converging to another known kernel such as the SM or Matern kernel.

We were eventually successful in predicting the sinc pattern, as shown in Figure 7. Unfortunately, we were only able to achieve this result under very carefully selected conditions. We had to explicitly

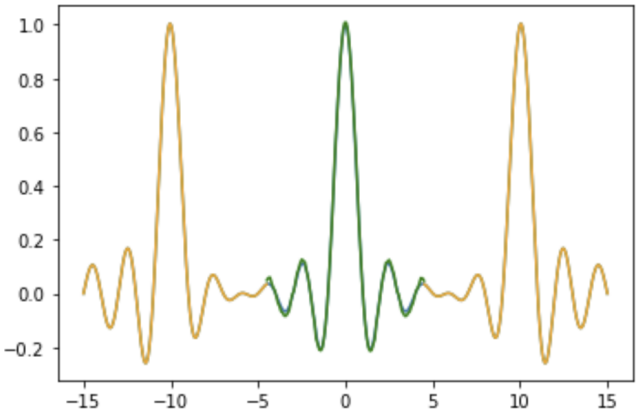

Figure 7: Successful prediction of the missing data from the sinc function achieved under unrealistic kernel initialization. Training data, true target data, and predicted data are shown in orange, blue, and green, respectively.

prohibit the optimizer from optimizing the noise parameter (fixed at 0.0001), and we initialized the means to values extremely close to the peaks of the spectral density plot of the trained SM kernel in the original literature. We were unable to replicate the results without both of these conditions.

We brought this issue to Roman Garnett, an expert in the field of Bayesian optimization and Gaussian processes, who attempted his own implementation in Matlab. Garnett obtained similar results, and once again could only replicate the paper's predictions by using the aforementioned initialization constraints [4].

2.3 Discussion

While it is theoretically possible to replicate the results of Wilson et al., it remains to be seen whether or not this can be done in the absence of a smart initialization. Although we had originally conjectured that convergence to the true pattern could be achieved through the use of learning rate scheduling and multistart optimization, further research has led us to conclude that it might be a much more complex and nuanced problem. Many of the erroneous predictions in Figure 6 were generated by GPs that converged to better marginal log likelihood values than the true prediction in Figure 7, which implies that a different loss function may be required. The prevailing literature on Gaussian processes and Bayesian optimization consistently points to marginal log likelihood as the best measure of Gaussian process fidelity, so research into other metrics could be very worthwhile.

Our results aside, the other experiments Wilson's team conducted in their original paper on the Spectral Mixture kernel continue to have merit. The SM kernel is a powerful and widely applicable covariance function, and the consistent inclusion of the kernel in Gaussian process libraries and popular literature highlights this fact. Since its inception in 2013, considerable work has been done to extend the usability of the SM kernel, and further research into its use cases and extensions is clearly warranted.

3 Extension: Deep Kernel Learning

One prominent extension of the Spectral Mixture kernel is deep kernel learning, a method developed by Wilson et al. in 2015. Scalable deep kernels "combine the structural properties of deep learning architectures with the non-parametric flexibility of kernel methods," [2]. Deep kernel learning entails transforming the inputs of a SM base kernel with a deep architecture. The result is a closed-form kernel that can be used in a standard Gaussian process.

More specifically, the SM kernel and its hyperparameters are transformed using a nonlinear mapping given by a deep architecture (e.g. a deep convolutional neural network). This can be interpreted as applying a GP with the SM kernel to the final hidden layer of the neural network, and since GPs correspond to infinite basis function representations, the result is a network whose final hidden layer effectively has an infinite number of units. The network weights and the kernel hyperparameters are trained together by optimizing the marginal log likelihood of the GP with the deep kernel, which provides "significant advantages over training a GP applied to the output layer of a trained deep neural network," [2].

Wilson et al. went on to show that Gaussian processes equipped with trained deep kernels outperformed both traditional GPs and deep neural networks in a wide range of tasks such as regression tasks from the popular UCI repository, orientation extraction from patches of facial images, magnitude recovery from handwritten digits, and step function recovery. GPs with deep kernels not only had the best performance in all of these tasks, but also did so without any extra computational overhead [2].

4 Conclusion

It is clear that the issue of human expertise in the context of Gaussian processes and Bayesian optimization still requires a large amount of further research. While advances such as the Spectral Mixture kernel do indeed hold a great deal of merit, our research has clearly shown that the kernel is not as powerful as was originally presented in Wilson's 2013 thesis. The authors clearly used the peaks of the spectral density of the sinc function as the initialization points for their Spectral Mixture kernel, a technique which is completely invalid in real-world cases where the objective function is entirely unknown. The unrealistic initialization aside, the Spectral Mixture kernel can clearly be utilized in creative ways in tandem with other deep learning tools and techniques, as was shown in Wilson's 2015 paper on Deep Kernel Learning. Advancements such as this must continually be investigated in order to increase the usability of Bayesian optimization techniques. It is clear that

we are not far-off from a fully-automated Bayesian optimization framework, and reaching this goal will lead to great discoveries across many disciplines.

5 Acknowledgements

We would like to give a special thank you to Roman Garnett and John Gibson for providing advice and insight throughout the course of this project. We would also like to thank Andrew Gordon Wilson and the rest of the GPyTorch developers for providing the tools necessary for continual advancement in the field of Bayesian inference and optimization.

References

[1] A. G. Wilson and R. P. Adams, "Gaussian process kernels for pattern discovery and extrapolation," 2013.

[2] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," 2015.

[3] J. Reid, "What are gaussian processes?," 2012. https://pythonhosted.org/infpy/gps.html.

[4] R. Garnett, "Cse544t - special topics in computer science theory: Bayesian optimization," 2020.

[5] S. Bochner, "Monotone funktionen, stielt- jessche integrale und harmonische analyse," 1933.

[6] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," in *Advances in Neural Information Processing Systems*, 2018.