Automated model search using Bayesian optimization and genetic programming

Louis Schlessinger
Washington University in St. Louis
lschlessinger@wustl.edu

Gustavo Malkomes
SigOpt
gustavo@sigopt.com

Roman Garnett
Washington University in St. Louis
garnett@wustl.edu

Abstract

Selecting a kernel for a Gaussian process is both crucial and requires considerable expertise. A kernel can impose many differing modeling assumptions and, therefore, can impact predictive performance. This paper presents a method to evolve compositional kernels of Gaussian processes that explain a given fixed-size dataset well using genetic programming. Experiments compare different kernel search strategies on real-world data. We propose an algorithm that uses Bayesian optimization and genetic programming to traverse the infinite search space of probabilistic models. Preliminary results indicate that this surrogate-assisted evolutionary kernel construction algorithm can discover underlying structure on different latent functions.

1 Introduction

Machine learning has been a driving force behind the success of multiple — and now commonplace — applications, *e.g.*, spam filters, search engines, and face recognition. Before deploying a machine learning algorithm, practitioners have to spend many hours cleaning data, choosing models, and tuning hyperparameters. Seeking to reduce the human effort associated with these tasks and allowing machine learning to be accessible to a broader audience, many papers have proposed automated machine learning techniques [4, 1, 13]. Recent developments on hyperparameter optimization are great examples [5, 12, 2]. In this paper, we combine ideas from hyperparameter optimization to tackle model selection in an automated fashion.

Bayesian optimization (BO) is a popular meta-learning strategy for global optimization. Given a black-box function $f : \mathcal{X} \mapsto \mathbb{R}$ over some domain $\mathcal{X}$, we wish to optimize f by sequentially performing function evaluations to the function f. The main features of a Bayesian optimization procedure are: it maintains a probabilistic belief about the objective function f, and it uses an acquisition function $\alpha : \mathcal{X} \mapsto \mathbb{R}$ to guide the search to the optimum. While BO has been relatively well-studied for bounded Euclidean input spaces, few recent papers have investigated applications of Bayesian optimization for non-Euclidean domains, with particular attention given to neural architecture search.

Genetic programming is a paradigm for automatically creating computer programs using evolutionary algorithms [7]. In genetic programming, these possible solutions are of variable-length and typically represented as trees. To encode these trees, a terminal set T and a function set F, together forming the primitive set, must be defined. This meta-heuristic can be applied to domains other than computer programs as long as the parsed expression of primitives encodes a solution to the problem. Here, we

3rd Workshop on Meta-Learning at NeurIPS 2019, Vancouver, Canada

are interested in using genetic programming to construct populations of objects that are non-Euclidean, such as kernel-based nonparametric probabilistic models.

Kernel-based nonparametric methods are a powerful class of techniques for discovering patterns or structure in data. Finding the underlying structure in data is important for extrapolation tasks. A popular kernel-based nonparametric model is the Gaussian process (GP) model, which depends on defining a kernel (or covariance function) $k(x, x')$. This kernel represents a similarity function between two given inputs x and x' and determines the inductive bias of the model. The task of selecting an appropriate kernel is crucial for generalization but is nontrivial, in part because the space of possible kernels is infinite. Consequently, it has been called a "black art" [3] and is often either left for experts or an off-the-shelf option, such as the radial basis function kernel, is used.

In this work, we combine a genetic programming search strategy with Bayesian optimization to perform optimization over the non-Euclidean space of covariance functions. Our automated statistical model search builds upon the work of Malkomes et al. [9]. The evolutionary algorithm constructs a population of candidate models using a surrogate GP model on the model space. This procedure allows the population to evolve, finding refined candidate models in the non-Euclidean domain. Then, we use a standard Bayesian optimization acquisition function to determine where to evaluate the expensive objective function. Preliminary results show that our approach is promising and could be used to combine the flexibility of evolutionary algorithms with the sample-efficiency of Bayesian optimization.

2 Automating model selection

Suppose we have a supervised learning problem defined on an input space $\mathcal{X}$ and output space $\mathcal{Y}$. We are given a set of observations $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ represents the design matrix of covariates $\mathbf{x}_i \in \mathcal{X}$, and $y_i \in \mathcal{Y}$ is a target value. We wish to automate the model selection process by considering a collection of probabilistic models $\mathbb{M}$ that could have plausibly generated the observed data. Our goal is to select one model from $\mathbb{M}$ to explain $\mathcal{D}$. In the Bayesian framework, a key quantity for model comparison is the *model evidence* (i.e., *marginal likelihood*), which is the probability of generating the observed data given a model $\mathcal{M}$:

$$p(\mathbf{y} \mid \mathbf{X}, \mathcal{M}) = \int_{\Theta_{\mathcal{M}}} p(\mathbf{y} \mid \mathbf{X}, \theta, \mathcal{M})\, p(\theta \mid \mathcal{M})\, \mathrm{d}\theta. \tag{1}$$

The evidence integrates over the hyperparameters $\theta \in \Theta_{\mathcal{M}}$ to account for all plausible explanations of the data offered by the model under a predefined prior on $p(\theta \mid \mathcal{M})$. Unfortunately, this quantity is often intractable for models such as Gaussian processes, and even an approximation to this quantity can be expensive to compute.

2.1 Gaussian process regression

A Gaussian process (GP) is a stochastic process such that for any finite collection of the random variables, they are jointly Gaussian. This powerful modeling tool can be used to describe distributions over functions. A GP is completely specified by its first two moments, the mean function $\mu : \mathcal{X} \mapsto \mathbb{R}$ and the covariance function (or kernel) $k : \mathcal{X}^2 \mapsto \mathbb{R}$. The kernel implies which structures and shapes are possible given the GP prior, which determines the model's capacity to interpolate and extrapolate. Here, we focus on GP models, but this work could be extended to any kernel-based probabilistic model. For more details on GPs, please see [11].

2.2 Model space

We need a space of models that is general and expressive. We adopt the context-free kernel grammar of Duvenaud et al. [3] due to its ability to create arbitrarily complex models. We start with a set of one-dimensional, so-called base kernels $\mathcal{B}$ that are added and multiplied to construct new kernels. Through this process, kernels over multiple dimensions are constructed. We use the same set of base kernels for reproducibility: squared exponential (SE), rational quadratic (RQ), linear (LIN), and periodic (PER). This results in a space of models $\mathbb{M}$ that is rich; it has different structural assumptions for modeling f as well as potentially different lower-dimensional subspaces when a subset of just a few features is selected.

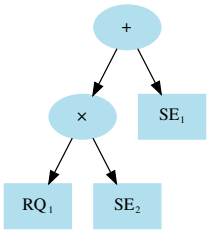

Figure 1: An example compositional kernel tree representing the expression $(RQ_1 \times SE_2) + SE_1$.

2.3 Bayesian optimization for model search

Given a model space $\mathbb{M}$, we aim to find a model $\mathcal{M} \in \mathbb{M}$ that maximizes the model evidence. Inspired by previous work [9], we view the evidence as a function $g : \mathbb{M} \mapsto \mathbb{R}$ to be optimized. They use Bayesian optimization to maximize this function to discover the optimal model

$$\mathcal{M}_{\text{OPT}} = \arg\max_{\mathcal{M} \in \mathbb{M}} g(\mathcal{M}; \mathcal{D}), \tag{2}$$

where $g(\mathcal{M}; \mathcal{D})$ is the log of the model evidence $\log p(\mathbf{y} \mid \mathbf{X}, \mathcal{M})$. They then placed a GP prior over g, $p(g) = \mathcal{GP}(g; \mu_g, K_g)$, where μ_g is a simple constant mean function and K_g is a special squared exponential "kernel kernel" that uses the Hellinger distance between models rather than ℓ_2-norm for computing the correlation (see Section 4.4 of the original paper). This approach was shown to be more sample-efficient than previously proposed approaches. As a computationally cheaper alternative to the Hellinger distance, here we use the *Frobenius distance*. We use this metric to define a new squared exponential kernel that uses the (averaged) squared Frobenius distance between the inputs. See Section A.2 for more details.

3 Automated model selection using Bayesian optimization and genetic programming

We next incorporate genetic programming for compositional kernel search into the sequential model-based optimization framework. The "outer" Bayesian optimization loop views g as a function to be optimized. The "inner" genetic programming loop returns a population of candidate models. Here, the fitness function is taken to be the acquisition function. We use the *expected improvement* [10].

3.1 Genetic programming for compositional kernel search

Compositional kernels can be represented as full binary trees; therefore, genetic programming can be used to search derivations of the compositional kernel grammar. Figure 1 shows an example kernel tree. Here, the function set $\mathcal{F} = \{+, \times\}$ and the terminal set $T = \mathcal{B}$. This setup is similar to that of [8]. This set of primitives allows for an evolution of compositional kernels because of the closure property of kernels under addition and multiplication.

A standard genetic programming algorithm is used, with minor modifications in the implementation. Two changes are adopted to account for the expensive cost of training a GP model. First, duplicate models are removed from the population after initialization and variation steps. We consider two models as duplicates if both of their covariance functions' have the same expanded form without considering hyperparameters. Second, the expanded expressions of evaluated models are saved to prevent repeated evaluations. Kernel trees are recombined using a standard subtree exchange operator and mutated using subtree replacement mutation with the *Ramped Half-n-Half* method [7] to generate random trees. Additional parameter settings can be found in Section A.1.

4 Experiments

Our approach is implemented using the Python Gaussian process library, GPy [6]. The code will be made publicly available at `https://github.com/lschlessinger1/boems`.

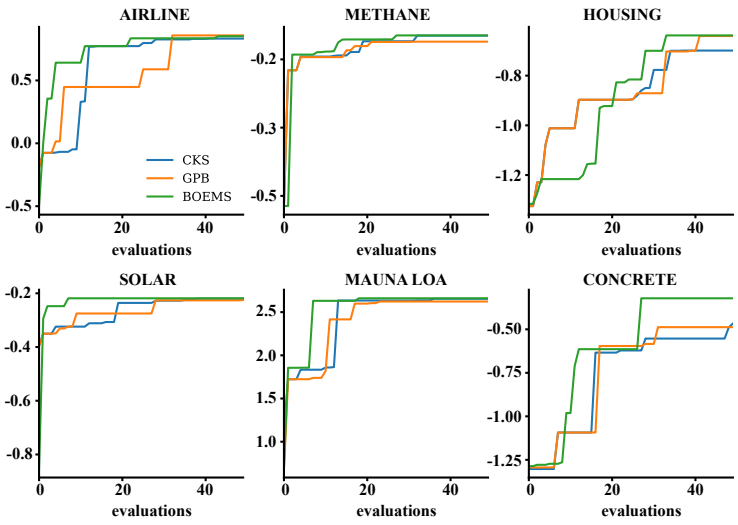

Figure 2: Plots of the best normalized model evidence (vertical axes) as a function of the number of model evidence evaluations.

We conducted experiments on several benchmark regression datasets[1] against alternative automated model selection methods. These publicly available real-world datasets were used in previous related work [9]. All datasets are standardized. For METHANE and CONCRETE, the first 500 observations are used. A baseline genetic programming model search strategy, referred to as GPB, is compared to the proposed method, here referred to as BOEMS. GPB has the same genetic programming parameter settings as our approach except that the population-level crossover rate is 0.6, the population-level mutation rate is 0.1, and the population size is 15. An adaptation of the greedy *compositional kernel search* [3] is also used, referred to here as CKS. BOEMS starts the search by evaluating SE first. BOEMS also uses an "inner" evaluation budget of 50 acquisition function evaluations for constructing the candidate model population.

All model search strategies use L-BFGS to optimize hyperparameters with 3 random restarts, each beginning from a sample of $p(\theta \mid \mathcal{M})$. The set of base kernels for one-dimensional problems is defined to be $\{SE_1, RQ_1, PER_1, LIN_1\}$ and $\{SE_i\} \cup \{RQ_i\}$ for multidimensional ones. The log evidence divided by the size of the dataset is reported $\log p(\mathbf{y}|\mathbf{X},\mathcal{M})/|\mathcal{D}|$.

Results in Figure 2 are shown for a budget of 50 model evidence evaluations on different regression problems. On the one-dimensional problems, (AIRLINE, METHANE, SOLAR, and, MAUNA LOA), our approach achieves higher approximate log evidence more quickly than GPB and CKS. On the multidimensional datasets (HOUSING and CONCRETE), our method outperforms CKS and is competitive with GPB. On CONCRETE, BOEMS outperforms GPB, but on HOUSING our method gets stuck early on in a bad region of model space, causing worse initial performance than GPB.

5 Conclusion

In this work, an automated method to learn covariance functions using evolutionary algorithms and Bayesian optimization is proposed. This algorithm searches for derivations of the kernel grammar using genetic programming to propose candidate models to query model evidence. It was empirically found that this approach was able to capture relevant structure for some one-dimensional time series and multidimensional regression problems. Future work aims to extend the experimental setting and apply this technique to other non-Euclidean domains.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under award numbers IIA–1355406 and IIS–1845434.

[1] `https://archive.ics.uci.edu/ml/datasets.php`

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3981–3989. Curran Associates, Inc., 2016.

[2] Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas. Bayesian optimization in alphago. *arXiv preprint arXiv:1812.06855*, 2018.

[3] David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pages 1166–1174, 2013.

[4] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc., 2015.

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

[6] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[7] John R Koza. Genetic programming: on the programming of computers by means of natural selection. 1992.

[8] Gabriel Kronberger and Michael Kommenda. Evolution of covariance functions for gaussian process regression using genetic programming. In *International Conference on Computer Aided Systems Theory*, pages 308–315. Springer, 2013.

[9] Gustavo Malkomes, Charles Schaff, and Roman Garnett. Bayesian optimization for automated model selection. In *Advances in Neural Information Processing Systems*, pages 2900–2908, 2016.

[10] J Močkus. On bayesian methods for seeking the extremum. In *IFIP Technical Conference on Optimization Techniques*, pages 400–404. Springer, 1974.

[11] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.

[13] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

A Appendix

A.1 Genetic programming parameters

Crossover Kernel trees are recombined using a standard subtree exchange operator, which selects one crossover point uniformly at random in each parent tree and swaps the subtrees to generate offspring. Hyperparameters from each tree are inherited in the swap.

Mutation Subtree replacement mutation with the *Ramped Half-n-Half* method [7], selecting both the *grow* and *full* method with equal probability to generate random trees having a maximum height of 2 is used here. For a given parent selected for mutation, a node is selected uniformly at random and replaced with the generated subtree.

We use fitness proportional selection to select parents and truncation selection to select offspring. A population size of 50 models is used here, initialized to all base kernels to start the search. The population-level crossover rate is 0.6, and the population-level mutation rate is 0.25. Thus, 85% of models each generation are selected for variation.

A.2 Frobenius squared exponential kernel kernel

The *Frobenius distance*, d_F, is defined as the sum of the element-wise squared differences.

We wish to compare two models $\mathcal{M}_\theta, \mathcal{M}'_{\theta'} \in \mathbb{M}$. The following distributions are defined, as in [9]

$$P = p(\mathbf{f} \mid \mathbf{X}, \mathcal{M}, \theta) = \mathcal{N}(\mathbf{f}; \mu_P, \Sigma_P) \quad Q = p(\mathbf{f} \mid \mathbf{X}, \mathcal{M}, \theta) = \mathcal{N}(\mathbf{f}; \mu_Q, \Sigma_Q). \tag{3}$$

The squared *Frobenius distance between* P and Q is

$$d_F^2\left(P, Q\right) = \left\|P - Q\right\|_F^2 = \operatorname{Tr}\left[\left(P - Q\right)\left(P - Q\right)^H\right], \tag{4}$$

where $(\cdot)^H$ denotes the conjugate transpose and $\left\|\cdot\right\|_F$ denotes the Frobenius norm.

To handle different hyperparameter settings of different model and their respective hyperpriors, we define the *expected squared Frobenius distance* of two models $\mathcal{M}, \mathcal{M}' \in \mathbb{M}$ is

$$\bar{d_F^2}(\mathcal{M}, \mathcal{M}'; \mathbf{X}) = \mathbb{E}\left[d_F^2\left(\mathcal{M}_\theta, \mathcal{M}'_{\theta'}; \mathbf{X}\right)\right] = \iint d_F^2\left(\mathcal{M}_\theta, \mathcal{M}'_{\theta'}; \mathbf{X}\right) p(\theta \mid \mathcal{M}) p(\theta' \mid \mathcal{M}') \,\mathrm{d}\theta \,\mathrm{d}\theta'. \tag{5}$$

We use this distance to create the *Frobenius squared exponential* covariance between models, which we define as

$$K_g(\mathcal{M}, \mathcal{M}'; \theta_g, \mathbf{X}) = \sigma^2 \exp\left(-\frac{1}{2}\frac{\bar{d_F^2}(\mathcal{M}, \mathcal{M}'; \mathbf{X})}{\ell^2}\right), \tag{6}$$

where $\theta_g = (\sigma, \ell)$ represents signal noise and length scale hyperparameters.