AUTOMATED KERNEL SEARCH USING EVOLUTIONARY ALGORITHMS

Louis Schlessinger
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
lschlessinger@wustl.edu

May 3, 2019

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 kernels of Gaussian processes that explains a given fixed-size dataset well using genetic programming. Experiments compare a different kernel search strategy on real-world data. An algorithm is proposed that uses genetic programming to traverse the infinite search space of probabilistic models. This evolutionary kernel construction algorithm can discover underlying structure on a variety of latent functions.

1 Introduction

Kernel-based nonparametric models are a powerful class of algorithms for discovering patterns or structure in data. Finding the underlying structure in data is important for extrapolation tasks. Many other methods require data to be explicitly transformed into a feature vector representation whereas kernel methods rely on computing inner products between data points in an implied high-dimensional feature space. A popular kernel-based non-parametric model is the Gaussian process (GP) model, which depends on defining a kernel (or covariance function) $k(\mathbf{x}, \mathbf{x}')$. This represents a similarity function between two given inputs $\mathbf{x}$ and $\mathbf{x}'$ and determines the inductive bias of the model. The task of selecting an appropriate kernel is crucial for generalization and nontrivial because the space of possible kernels is infinite. Consequently, it has been called a "black art" and is either left for experts or an off-the-shelf option, such as the radial basis function kernel, is used.

This work builds on recent work in automated statistical model construction [1, 2, 3]. A similar approach to that of [4], traversing the context-free grammar of [1] is taken here.

There has been much previous work on applying evolutionary algorithms to the task of kernel selection. [5] Uses genetic programming to evolve kernels for support vector machines (SVMs) and relevance vector machines. A critical shortcoming of this approach is that it aims to learn the covariance function itself rather than the expression describing covariance structure. [1] Totes that this captures structures that are not simply described parametrically. Additionally, evolving kernels for SVMs typically involves minimizing cross-validation error.

In this work, a kernel search algorithm is proposed for Gaussian process models. The aim of this paper is to investigate the effectiveness of using genetic programming to propose candidate models. These candidates are derivations of the context-free grammar defined by [1], in which it is noted that "a sum of kernels can be understood as an OR-like operation" and "multiplying kernels is an AND-like operation", hinting at an interpretation of the composite kernel expression as parse trees. Experiments on simple regression problems show promising results.

2 Gaussian Process Regression

A Gaussian process is a real-valued stochastic process such that for any finite collection of the random variables, they are jointly Gaussian [6]. This powerful modeling tool describes distributions over functions and is completely specified by their first two moments. The kernel implies which structures and shapes are likely given the GP prior, which determines the model's capacity to interpolate and extrapolate.

In this work, a similar framework to [3] is adopted We are faced with a supervised regression problem on some input domain $\mathcal{X}$ and output space $\mathcal{Y}$ and given some training set $\mathcal{D} = (X, \mathbf{y})$, where X is the design matrix of covariates and $\mathbf{y}$ is the target vector. The goal is to try to learn a latent function $f \colon \mathcal{X} \mapsto \mathbb{R}$ that generated observed values $\mathbf{y}$. Additive Gaussian observation noise is assumed. Then, a GP prior distribution is placed on f, $p(f \mid \theta) = \mathcal{GP}(f; \mu(x; \theta), k(x, x'; \theta))$, where $\mu = 0$, k is a covariance function, and θ is used to denote the vector of GP model hyperparameters. Given unseen test inputs X_* , we would like to predict unobserved targets $\mathbf{y}_*$.To do this, we form a predictive distribution

$$p(\mathbf{y}_* \mid X_*, \mathcal{D}) = \mathcal{N}(\mu_*, K_*) \tag{1}$$

$$\mu_* = k(X_*, X)(k(X, X) + \sigma^2 I)^{-1}\mathbf{y} \tag{2}$$

$$K_* = k(X_*, X_*) - k(X_*, X)(k(X, X) + \sigma^2 I)^{-1}k(X, X_*) \tag{3}$$

To measure the quality of a model fit, we can use the marginal likelihood, also known as the evidence [7]. The probability of observing the data under the GP prior

$$p(y \mid X, \theta) = \int p(y \mid f)p(f \mid X, \theta)\mathrm{d}f \tag{4}$$

balances data fit with model complexity. Therefore, the objective here is to find the θ and σ that maximize the log of the model evidence $\log p(y \mid X, \theta)$.

2.1 Model Space

The same kernel composition framework as defined in [1] is used because of its representational capacity and kernel grammar. They define a set of one-dimensional base kernels $\mathcal{B}$ that are added and multiplied to construct new kernels. The same base kernels are used here; squared exponential (SE), rational quadratic (RQ), linear (LIN), and periodic (PER). Genetic programming is used to search for derivations of this grammar.

3 Genetic Programming for Covariance Function Search

3.1 Evolutionary Algorithms

An evolutionary algorithm consists of a population of individuals guided by environmental pressure that is stochastically transformed each generation into a new population that hopefully has a higher fitness [8]. Given some objective function, a set of random candidate solutions are initialized and evaluated. Parent solutions are chosen as a function of the objective score to generate new candidate solutions using some combination of variation operators, such as crossover (or recombination) or mutation. The crossover operator maps two or more individuals to one or more individuals. The mutation operator maps one or more individuals to one new individual. After these operators are applied, the new candidates are evaluated, and as a function of the objective, members of the population are selected to seed the next iteration. This process is repeated until a termination condition is met, typically an evaluation budget or a quality threshold.

3.2 Genetic Programming

Genetic programming is a paradigm for automatically creating computer programs using evolutionary algorithms [9]. In genetic programming, these possible solutions are of variable-length and typically represented as trees. To encode these trees, it is necessary to specify a terminal set T and a function set F, together forming the primitive set. This metaheuristic can be applied to domains other than computer programs as long as the parsed expression of primitives encodes a solution to the problem. Variation operators greatly differ from those used in other evolutionary algorithms because of the tree representation.

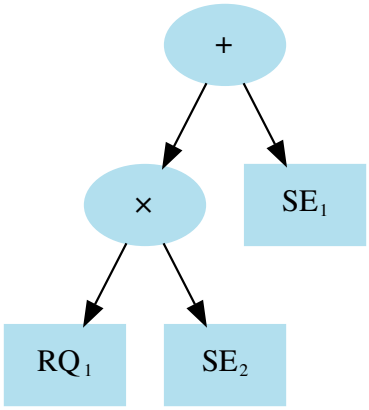

Figure 1: Example of a compositional kernel tree corresponding to $(RQ_1 \times SE_2) + SE_1$

Here, the fitness function is taken to be the model evidence. The aim is to search over derivations of the compositional kernel grammar to find the model that maximizes the model evidence. This setup is similar to that of [4]. However, in this work, linear genomes are not used.

4 Evolutionary Structure Discovery

In this section, the method for traversing model space, which is called evolutionary kernel construction (EKC) will be presented.

4.1 Kernel Representation

In order to define crossover and mutation operators, it is necessary to define the representation of a kernel. A parse tree is used to encode kernels. The function set $\mathcal{F} = \{+, \times\}$ and the terminal set $T = \mathcal{B}$. Similar to that of [4], this set of primitives allows for an evolution of compositional kernels because of the closure property of kernels under addition and multiplication.

This kernel encoding also gives us information about what kind of genotypic structure is expected. Because the kernel trees are *full* binary trees, (i.e. each node is either a leaf or has two children), the relationship between the total number of nodes, the number of internal nodes, and leaves is known. For a given kernel tree with n nodes, where $n \in \{1, 3, 5, ...\}$, there are $I = \frac{n-1}{2}$ internal nodes and $L = \frac{n+1}{2}$ leaf nodes.

4.2 Variation Operators

Both crossover and mutation are used to generate candidate models. The population-level crossover rate is 0.6 and the population-level mutation rate is 0.1. Thus, 70% of models each generation are selected for variation.

Crossover Kernel trees are recombined using a leaf-biased subtree exchange operator [9]. The standard subtree crossover operator selects one crossover point uniformly at random in each parent tree and swaps the subtrees to generate offspring. A leaf-biased operator biases the set of possible crossover points to have more internal nodes (operators) than leaf nodes (base kernels). Without this leaf bias, if a node in a nonempty full binary tree is naively selected uniformly at random, the probability of selecting a leaf is $\ell = \frac{n+1}{2n}$, which is quite high and would represent smaller steps in model space.

Swapping subtrees corresponds to an exchange of structural assumptions about the objective function. In this work, the leaf probability is set to $\ell = 0.1$ to make the swapping of larger kernel expressions more likely, implying larger steps in model space. Hyperparameters from each subtree are also inherited in each swap.

Mutation Subtree mutation with the *Ramped Half-n-Half* method [9], 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.

4.3 Evolutionary Kernel Learning Algorithm

A standard genetic programming algorithm is used, with some minor modifications. Two changes are adopted to account for the computational cost of optimizing a GP model. First, duplicate models are removed from the population after initialization and variation. Two models are considered duplicates if both of their covariance functions' have the same expanded form without considering hyperparameters. Second, all models that have been previously evaluated are tracked to prevent repeated evaluations.

The standard *Ramped Half-n-Half* initialization method [9], creating trees having a maximum height of 2 is done to start the search. Exponential ranking selection is used here with ranking base parameter $c = 0.7$ to select parent models to expand. This gives most of the probability mass to higher scoring models, while still allowing for a diverse set of kernels. Truncation selection is used to select offspring, keeping the models with the best model evidence at the end of each generation. A population size of 25 models is used here.

5 Experiments

For the implementation of this algorithm, the library GPy [10] is used. Code is publicly available at `https://github.com/lschlessinger1/MS-project`.

We split the experiments into two sections: real-world data and synthetic data. All datasets are standardized and an evaluation budget of 50 is used. 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 for EKS

$$\frac{\log p(y \mid X, \theta)}{\mid \mathcal{D} \mid}. \tag{5}$$

For BOMS the following log evidence is reported

$$\frac{\log p(y \mid X, \mathcal{M})}{\mid \mathcal{D} \mid}. \tag{6}$$

In EKS, all of the hyperparameters which were part of the previous parent model are initialized to their previous values. All hyperparameters are then optimized, randomly restarting the newly introduced parameters. For EKS, exact Gaussian inference is used to compute the marginal likelihood.

5.1 Empirical Evaluation

To evaluate the predictive performance of the proposed algorithm, experiments were conducted on several regression benchmark datasets.[1] These publicly available, real-world datasets were used in related previous work [3]. For methane and concrete, a subset of 500 observations was used. *Bayesian optimization for automated model selection* (BOMS) from [3] is compared to the proposed method. Both methods optimize hyperparameters using L-BFGS with random restarts. BOMS is run with the default set of parameters as described in the original work. For BOMS, the Laplace approximation to the marginal likelihood is used.

Figure 2 shows the kernel construction algorithm results in a wide range of regression problems. On the one-dimensional problems, (solar, mauna, methane, and, airline), EKS achieves higher approximate log evidence on airline and methane, but lower on solar and mauna loa. Interestingly, on the higher dimensional datasets (concrete and housing), EKS obtains a higher approximate log marginal likelihood than BOMS.

5.2 Synthetic Data Validation

The results for synthetic one-dimensional regression problems are shown in Table 1. A dataset is generated by sampling a path from a GP with the prior covariance equal to the true kernel. The signal noise variance σ_f^2 was set to 0.01. Multiple models are optimized with 10 random restarts using L-BFGS to compute the marginal likelihood. This follows a similar comparison to that of [4]. Unsurprisingly, the base kernels SE, RQ, and PER perform the worst. Interestingly, the evolved kernel in the simpler datasets generated by $SE + RQ$ and $SE + PER$ have higher marginal likelihood than the kernel that actually generated it. As expected, EKS is able to produce a better fit than the other models on the more complex dataset $LIN \times LIN \times PER$, but still worse than the true model.

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

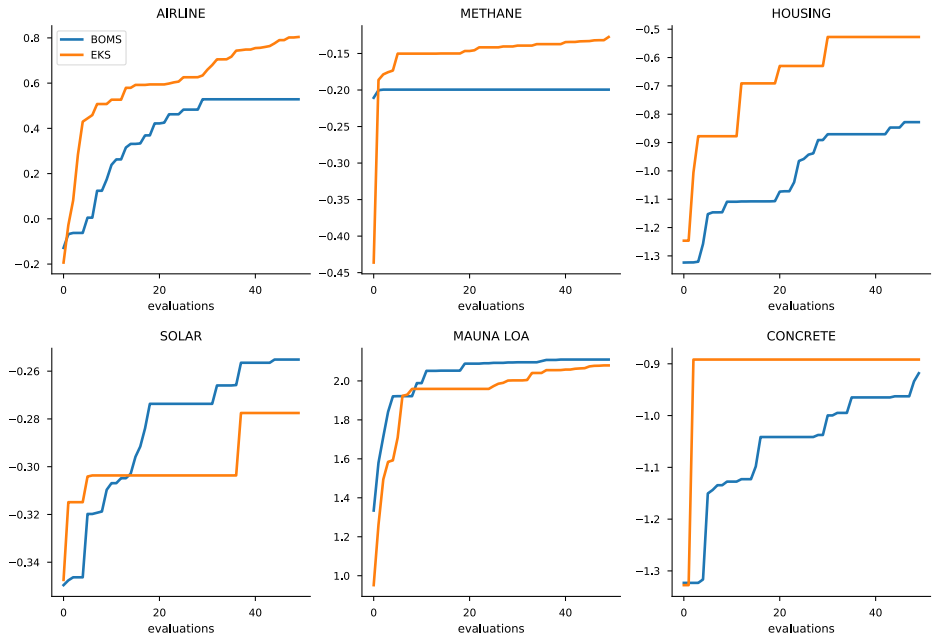

Figure 2: Plots of the best normalized model evidence as a function of the number of evaluations.

Table 1: Best normalized model evidence on three synthetic one-dimensional datasets sampled from the prior.

	True Kernel		
	$SE + RQ$	$SE + PER$	$LIN \times LIN \times PER$
SE	0.5083182	0.5008334	0.0055965
RQ	0.5160727	0.5008333	0.0055964
PER	0.5123873	0.5004492	0.0055889
Evolved	0.5403027	0.5041181	0.0303614
Actual	0.5240568	0.5008334	0.1899095

6 Conclusion

In this work, an automated method to learn covariance functions using evolutionary algorithms is introduced. This algorithm searches for derivations of the kernel grammar defined in [1] using principles from genetic programming. It was empirically found that this approach was able to capture relevant structure for some one-dimensional time series and multidimensional regression problems. Future research on the locality of the encoding proposed here and on the implicit candidate model distribution should be done.

Acknowledgments

I would like to acknowledge Gustavo Malkomes for helpful comments.

References

[1] David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. *arXiv preprint arXiv:1302.4922*, 2013.

[2] James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua Tenenbaum, and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *Twenty-eighth AAAI conference on artificial intelligence*, 2014.

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

[4] 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.

[5] Laura Diosan, Alexandrina Rogozan, and Jean Pierre Pecuchet. Evolving kernel functions for svms by genetic programming. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 19–24. IEEE, 2007.

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

[7] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[8] Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[9] John R Koza and John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

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