Raiders of the Lost Architecture: Kernels for Bayesian Optimization in Conditional Parameter Spaces

Kevin Swersky
University of Toronto
kswesrky@cs.utoronto.edu

David Duvenaud
University of Cambridge
dkd23@cam.ac.uk

Jasper Snoek
Harvard University
jsnoek@seas.harvard.edu

Frank Hutter
Freiburg University
fh@informatik.uni-freiburg.de

Michael A. Osborne
University of Oxford
mosb@robots.ox.ac.uk

Abstract

In practical Bayesian optimization, we must often search over structures with differing numbers of parameters. For instance, we may wish to search over neural network architectures with an unkown number of layers. To relate performance data gathered for different architectures, we define a new kernel for conditional parameter spaces that explicitly includes information about which parameters are active in a given structure. We show that this kernel improves GP model quality and GP-based Bayesian optimization results over several simpler baseline kernels.

FH: I left notes throughout using this mechanism. These notes stand out ugly on purpose – so that they can't be overlooked easily. Once a note is dealt with, please remove it or comment it out in the source. For checking e.g. length, all notes can also be disabled at once by commenting out a single line towards the top of the source.

FH: changed title to include "Bayesian Optimization in" since its the BayesOpt workshop. I mildly prefer that but it's longer, so I'm also happy if someone wants to undo hte change.

MAO: I agree with title change, at least for this version of the paper.

1 Introduction

Bayesian optimization (see [**?**] for a detailed overview) is an efficient approach for solving blackbox optimization problems of the form $\arg\min_{x \in X} f(x)$, where f is expensive to evaluate. It employs a prior distribution $p(f)$ over functions that is updated as new information on f becomes available. The most common choice of prior distribution are Gaussian processes (GPs [1]), as they are powerful and flexible models for which the marginal and conditional distributions can be computed efficiently.[1] However, some problem domains remain challenging to model well with GPs, and the efficiency and effectiveness of Bayesian optimization suffers as a result. In this paper, we tackle the common problem of input dimensions that are only relevant if other inputs take certain values [**?**, **?**]. This is a general problem in algorithm configuration [**?**] that occurs in many machine learning contexts, such as, for example, in deep neural networks [**?**]; flexible computer vision architectures [**?**]; and the combined selection and hyperparameter optimization of machine learning algorithms [**?**]. We detail the case of deep neural networks below.

Bayesian optimization has recently been applied successfully to deep neural networks [**?**, **?**] to optimize high level model parameters and optimization parameters, which we will refer to collectively as *hyperparameters*. Deep neural networks represent the state-of-the-art on multiple machine learning benchmarks such as object recognition [**?**].

FH: TODO: as we're saying they are the state of the art in *multiple* benchmarks, we have to list at least two examples, better more.

They are multi-layered models by definition, and each layer is typically parameterized by a unique set of hyperparameters, such as regularization parameters and the layer capacity or number of hidden units. Thus adding additional layers introduces additional hyperparameters to be optimized. The result is a complex hierarchical conditional parameter space, which is difficult to search over. Historically, practitioners have simply built a separate model for each type of architecture [**?**] or assumed a fixed architecture [**?**]. However, if there is any relation between networks with different architectures, separately modeling each is wasteful.

While GPs with standard kernels fail to model the performance of architectures with such conditional parameters, the innovation of this paper is the introduction of a kernel that allows observed information to be shared across architectures when this is appropriate. We demonstrate empirically on a GP regression task and a Bayesian optimization task that this kernel models the conditional parameter space of a typical deep learning problem better than previous adhoc methods.

2 A Kernel for Conditional Parameter Spaces

In this section, we construct a kernel between points whose features may be irrelevant under known conditions. As an explicit example, we consider the case in which points may potentially have differing numbers of features: here no relevance can be assigned to the value of a feature in the first point which is missing in the second.

Formally, we aim to do inference about some function g with domain (input space) $\mathcal{X}$. $\mathcal{X} = \prod_{i=1}^{D} \mathcal{X}_i$ is a D-dimensional input space, where each individual dimension is bounded real, that is, $\mathcal{X}_i$ is $[l_i, u_i] \subset \mathbb{R}$ (with lower and upper bounds l_i and u_i, respectively). We define functions $\delta_i \colon \mathcal{X} \to \mathcal{B}$, for $i \in \{1, \ldots, D\}$, and where $\mathcal{B} = \{\text{true}, \text{false}\}$. $\delta_i(\underline{x})$ stipulates the relevance of the ith feature, x_i, to inference about $g(\underline{x})$.

2.1 The problem

To begin with, we can imagine trying to model the performance of neural networks with either one or two layers, with respect to the regularization parameters for each layer, x_1 and x_2. If y represents the performance of a one layer-net with regularization parameters x_1 and x_2, then the value x_2 doesn't matter, since there is no second layer to the network. In the below, we'll write an input triple as $(x_1, \delta_2(\underline{x}), x_2)$ and assume that $\delta_1(\underline{x}) = \text{true}$; that is, the regularization parameter for the first layer is always relevant.

In this setting, we want a kernel k to be dependent only on, firstly, whether parameters are relevant, and, secondly, the values of parameters relevant for both points. If we elaborate on the second consideration, assuming some x_1 and x_1':

[1]There are prominent exceptions to this rule, though. In particular, tree-based models, such as random forests, tend to be the better choice if there are many data points (and GPs thus become computationally inefficient), if the input dimensionality is high, if the noise is not normally distributed, or if there are non-stationarities [**?**, **?**, **?**].

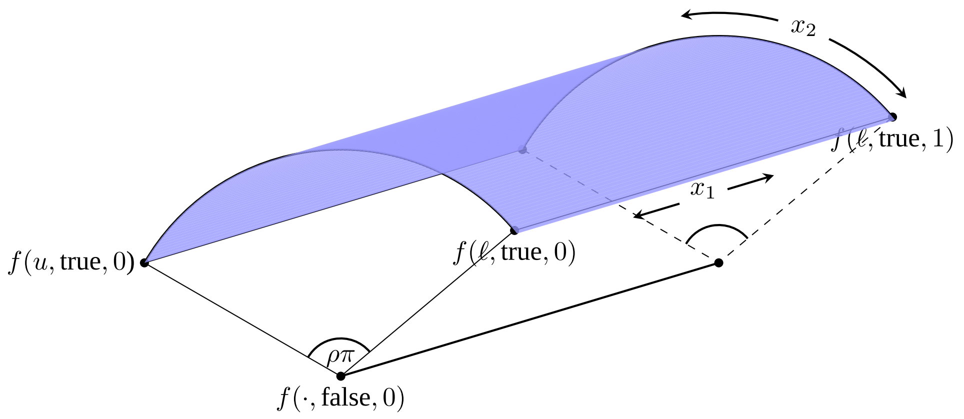
x_2
$f(\ell, \text{true}, 1)$
x_1
$f(u, \text{true}, 0)$
$f(\ell, \text{true}, 0)$
$\rho\pi$
$f(\cdot, \text{false}, 0)$

Figure 1: A demonstration of the embedding giving rise to the pseduo-metric in 2 dimensions. All points for which $\delta_2(x) = \text{false}$ are mapped onto a line varying only along x_1. Points for which $\delta_2(x) = \text{true}$ are mapped to the surface of a semicylinder, depending on both x_1 and x_2. This embedding gives a constant distance between pairs of points which have differing values of δ but the same values of x_1.

- If we are comparing two points for which the same parameters are relevant, the value of any unused parameters shouldn't matter,

$$k\big((x_1, \text{false}, x_2), (x_1', \text{false}, x_2')\big) = k\big((x_1, \text{false}, x_2''), (x_1', \text{false}, x_2''')\big),\ \forall x_2, x_2', x_2'', x_2''';\tag{1}$$

- The covariance between a point using both parameters and a point using only one should again only depend on their shared parameters,

$$k\big((x_1, \text{false}, x_2), (x_1', \text{true}, x_2')\big) = k\big((x_1, \text{false}, x_2''), (x_1', \text{true}, x_2''')\big),\ \forall x_2, x_2', x_2'', x_2'''.\tag{2}$$

Elaborating on the first consideration:

- For points that have identical values for all jointly relevant parameters, their covariance should depend only on the relative relevances of the remaining parameters,

$$k\big((x_1, \text{false}, x_2), (x_1, \text{false}, x_2')\big) = k_{\text{FF}},\ \forall x_2, x_2'\tag{3}$$

$$k\big((x_1, \text{false}, x_2), (x_1, \text{true}, x_2')\big) = k_{\text{FT}},\ \forall x_2, x_2'.\tag{4}$$

Here we want $k_{\text{FF}} > k_{\text{FT}}$, expressing the fact that points that have identical relevances $\delta_2(\underline{x})$ are more similar than points that differ in relevance $\delta_2(\underline{x})$.

2.2 Cylindrical Embedding

We can build a kernel with these properties by an embedding of points into a higher-dimensional space than they began, and performing regression in that space. Specifically, the embedding we use is:

$$f_i^{\text{r}}(\underline{x}) = \begin{cases} [0,0]^\intercal & \text{if } \delta_i(\underline{x}) = \text{ false} \\ \omega_i[\sin \pi\rho_i \frac{x_i}{u_i - l_i}, \cos \pi\rho_i \frac{x_i}{u_i - l_i}]^\intercal & \text{otherwise.} \end{cases} .$$

Figure 1 shows a visualization of the embedding of points x into a higher-dimensional space with relative distances that only depend on active parameters. This embedding gives rise to the Euclidean distances:

$$d_i^{\text{r}}(\underline{x}, \underline{x}') = \begin{cases} 0 & \text{if } \delta_i(\underline{x}) = \delta_i(\underline{x}') = \text{false} \\ \omega_i & \text{if } \delta_i(\underline{x}) \neq \delta_i(\underline{x}') \\ \omega_i \sqrt{2}\sqrt{1 - \cos(\pi\rho_i \frac{x_i - x_i'}{u_i - l_i})} & \text{if } \delta_i(\underline{x}) = \delta_i(\underline{x}') = \text{true.} \end{cases}$$

For kernels such as the exponentiated quadratic $k(x, x') = \sigma_f^2 \exp\left({}^{-(x-x')^2}/_{2\ell^2}\right)$, or the Matérn, which only depend on the Euclidian distance, the covariance has the desired properties.

Definition 1. *A function $\kappa\colon \mathbb{R}^+ \to \mathbb{R}$ is* a positive semi-definite covariance function over Euclidean space *if $K \in \mathbb{R}^{N \times N}$, defined by*

$$K_{m,n} = \kappa\big(d_E(\underline{y}_m, \underline{y}_n)\big), \quad \textit{for } \underline{y}_m, \underline{y}_n \in \mathbb{R}^P, \quad m, n = 1, \ldots, N,$$

is positive semi-definite for any $\underline{y}_1, \ldots, \underline{y}_N \in \mathbb{R}^P$.

A popular example of such a κ is the exponentiated quadratic, for which $\kappa(\delta) = \sigma^2 \exp(-\frac{1}{2}\frac{\delta^2}{\lambda^2})$; another popular choice is the rational quadratic, for which $\kappa(\delta) = \sigma^2(1 + \frac{1}{2\alpha}\frac{\delta^2}{\lambda^2})^{-\alpha}$.

3 Experiments

We now show that our new kernel yields better results than other alternatives. Bayesian optimization requires building a model of the function being optimized, and better models can be expected to lead to better outcomes. However, because of the many interacting components of BO, optimizer performance might not correspond directly to the quality of the model. Thus, we perform two types of experiments: first, we study model quality in isolation in a regression task; second, we demonstrate that the improved model indeed yields improved Bayesian optimization performance.

Data. Both types of experiments used similar data obtained from a blackbox function that models the performance of a deep neural networks with up to six layers. The function takes XXX inputs, out of which YYY are conditional: the parameters in layer k are only relevant if the network depth is at least k. Function evaluations were performed on a

type of GPU

and required ZZZ seconds on average.

FH: please fill in XXX, YYY, and ZZZ.

3.1 Model Quality Experiments

Models. Separate Linear and Seperate GP build a separate model for each neural net architecture, as in [**?**]. The hierarchical GP model combines all data together using the conditional kernel. In the case where all data comes from the same architecture, the hierarchical GP model makes slightly different assumptions than a standard GP, because it places the data on a semi-circle. To check whether this alternate assumption affects performance, we also include a Separate-Hierarchical GP model. We also compare against a simpler embedding method, or "Poor Man's Embedding". In this procedure, we combine data from all models into a single model, replacing any unused parameters with a heuristic constant, set to -1 in these experiments.

Data.

FH: very briefly describe the number of data points, where they come from, and how they were split to obtain error bars (I assume cross-validation).

Results.

FH: Describe Table 1.

3.2 Bayesian Optimization Experiments

Experimental Setup. For Bayesian optimization, we used the same process as in [**?**], including slice sampling and *expected* expected improvement, but not expected improvement per time spent.

Table 1: Normalized Mean Squared Error on Neural Network data

Method	NN	NN log	NN half	NN log half
Separate Linear	**0.968**	0.886	**1.039**	2.120
Separate GP	**0.925**	0.641	**0.860**	0.848
Poor Man's embedding Linear	**0.905**	0.763	0.996	0.851
Poor Man's embedding GP	**0.907**	0.518	**1.178**	**0.752**
Separate Hierarchical GP	**0.801**	0.627	**0.956**	0.950
Hierarchical GP	**0.801**	**0.441**	**0.993**	**0.674**

FH: mention anything that differed in the setup.

Results.

FH: Discuss results including the nice figure of error over time. Also briefly mention experiments on datasets that did *not* look awesome (we can't pick and choose!)

4 Conclusion

We introduced a kernel for conditional parameter spaces that facilitates modelling the performance of deep neural network architectures by enabling the sharing of information across architectures where useful. Empirical results show that this kernel improves GP model quality and GP-based Bayesian optimization results over several simpler baseline kernels.

FH: fleshed out very briefly - please feel free to expand, e.g., to add a highlight from the experiments.

References

[1] C.E. Rasmussen and CKI Williams. Gaussian Processes for Machine Learning. *The MIT Press, Cambridge, MA, USA*, 2006.