Raiders of the Lost Architecture: Kernels for 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

When performing model-based 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 combine information between different architectures, we define a family of kernels for conditional parameter spaces.

1 Introduction

Recently, Bayesian optimization has been used to learn parameters for neural networks [1]. Because different neural net architectures have different numbers of parameters (or their parameters have different meanings), it is not clear how to combine information or extrapolate between the performance of different architectures. Historically, practitioners have simply built a separate model for each type of architecture [2]. However, if there is any relation between networks with different architectures, seperately modeling each architecture is wasteful.

Bayesian optimization methods rely on constructing a model of the function being optimized. We model this function using Gaussian process priors [3], where model assumptions are defined through the kernel. In this paper, we introduce a simple method for building a Gaussian process model over parameter spaces with varying numbers of parameters. We do this by defining an embedding of datapoints depending on which parameters are active, then perform standard regression in this new space.

2 A Kernel for Conditional Parameter Spaces

In this section, we construct a kernel between points having potentially differing numbers of dimensions.

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 regualization parameters for each layer, x_1 and x_2. If y represents the

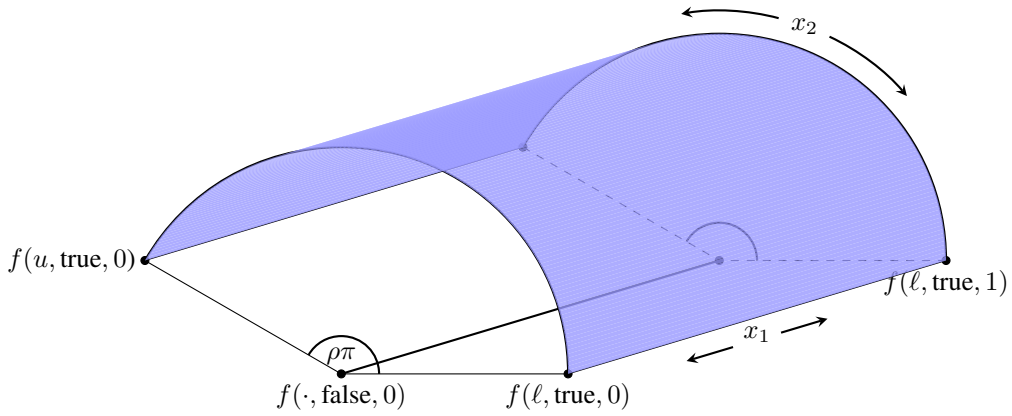

Figure 1: A demonstration of the embedding giving rise to the pseduo-metric in 2 dimensions. All points for which $\delta_i(x) = \text{false}$ are mapped onto a line varying only along x_1. Points for which $\delta_i(x) = \text{true}$ are mapped to the surface of a semicylinder, depending on both x_1 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.

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 this setting, we want a kernel on conditional spaces to have the following properties:

- If we are comparing two points with the same number of parameters, the value of any unused parameters shouldn't matter. $k(x_1, \text{false}, x_2, x_1', \text{false}, x_2') = k(x_1, \text{false}, x_2'', x_1', \text{false}, x_2''')\forall x_2, x_2', x_2'', x_2'''$
- The covariance between points using both parameters and points only using one should again only depend on their shared parameters. $k(x_1, \text{false}, x_2, x_1', \text{true}, x_2') = k(x_1, \text{false}, x_2'', x_1', \text{true}, x_2''')\forall x_2, x_2', x_2'', x_2'''$

2.2 Cylindrical Embedding

We can build a kernel with these properties by and embedding of points into a hiher-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 Euclidian 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.

3 Experiments

Regression 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 interating components of BO, optimizer performance might not correspond directly to the quality of the model. In this section, we isolate to what extent the conditional kernel is a better model than the alternatives.

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**

Models Separate Linear and Seperate GP build a separate model for each neural net architecture, as in [2]. 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.

4 Conclusion

References

[1] Jasper Snoek, Hugo Larochelle, and Ryan Prescott Adams. Practical bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems*, 2012.

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, Balázs Kégl, et al. Algorithms for hyper-parameter optimization. In *25th Annual Conference on Neural Information Processing Systems (NIPS 2011)*, 2011.

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