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

Kevin Swersky
University of Toronto
kswersky@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 relevant in a given structure. We show that this kernel improves model quality and Bayesian optimization results over several simpler baseline kernels.

1 Introduction

Bayesian optimization (BO) is an efficient approach for solving blackbox optimization problems of the form $\arg\min_{x \in X} f(x)$ (see [1] for a detailed overview), 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 [2]), 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 [6, 7]. This is a general problem in algorithm configuration [6] that occurs in many machine learning contexts, such as, for example, in deep neural networks [8]; flexible computer vision architectures [9]; and the combined selection and hyperparameter optimization of machine learning algorithms [10]. We detail the case of deep neural networks below.

Bayesian optimization has recently been applied successfully to deep neural networks [11, 7] 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

[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 [3, 4, 5].

learning benchmarks such as object recognition [12], speech recognition [13], natural language processing [**?**] and 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 [14] or assumed a fixed architecture [11]. 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 The contribution 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 *ad-hoc* 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 (further details are available in [15]). 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 f 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, $\mathcal{X}_i = [l_i, u_i] \subset \mathbb{R}$ (with lower and upper bounds l_i and u_i, respectively). We define functions $\delta_i \colon \mathcal{X} \to \{\text{true}, \text{false}\}$, for $i \in \{1, \ldots, D\}$. $\delta_i(\underline{x})$ stipulates the relevance of the ith feature, x_i, to inference about $f(\underline{x})$.

GPs employ a positive-definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ to model the covariance between function values. Typical GP models cannot, however, model the covariance between function values whose inputs have different (possibly overlapping) sets of relevant variables. We address this issue next.

2.1 The problem

¡¡¡¡¡¡¡ HEAD 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. 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 on whether parameters are relevant and the values of relevant parameters for both points. For example, consider first-layer parameters x_1 and x_1'. ======= As an example, 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. 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. ¿¿¿¿¿¿¿ 9fe7849f5ddce186d8e75b87bea078da380dd798

In this setting, we want a kernel k to be dependent on which parameters are relevant, and the values of relevant parameters for both points. For example, suppose we have first-layer parameters x_1 and 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,

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

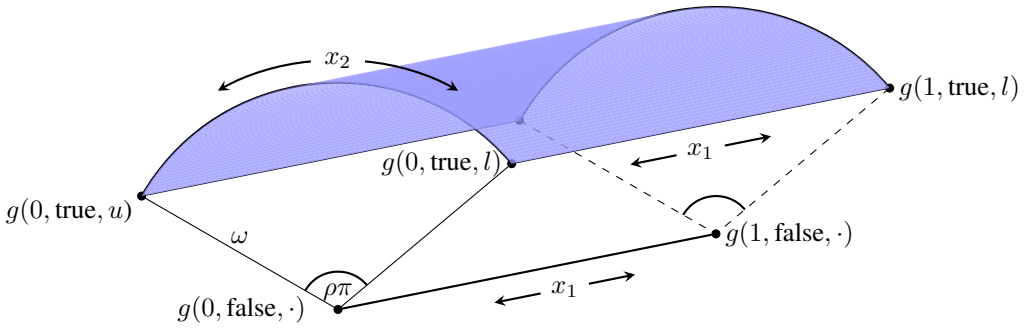

Figure 1: A demonstration of the embedding giving rise to the pseudo-metric in 2 dimensions. All points for which $\delta_2(x) = \text{false}$ are mapped onto a line varying only along x_1. Points for which $\delta_2(x) = \text{true}$ are mapped to the surface of a semicylinder, depending on both x_1 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.

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,

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

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

We usually additionally want $k_{\text{FF}} > k_{\text{FT}}$, expressing the fact that points that have identical relevance $\delta_2(\underline{x})$ 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 embedding our points into a Euclidean space. Specifically, the embedding we use is

$$g_i(\underline{x}) = \begin{cases} [0, 0]^\intercal & \text{if } \delta_i(\underline{x}) = \text{ false} \\ \omega_i[\sin \pi\rho_i \frac{x_i}{u_i - l_i}, \cos \pi\rho_i \frac{x_i}{u_i - l_i}]^\intercal & \text{otherwise.} \end{cases} \tag{5}$$

Where $\omega_i \in \mathbb{R}^+$ and $\rho_i \in [0, 1]$.

Figure 1 shows a visualization of the embedding of points $(x_1, \delta_2(\underline{x}), x_2)$ into $\mathbb{R}^3$. In this space, we have the Euclidean distance,

$$d_i(\underline{x}, \underline{x}') = ||g_i(\underline{x}) - g_i(\underline{x}')||_2 = \begin{cases} 0 & \text{if } \delta_i(\underline{x}) = \delta_i(\underline{x}') = \text{false} \\ \omega_i & \text{if } \delta_i(\underline{x}) \neq \delta_i(\underline{x}') \\ \omega_i\sqrt{2}\sqrt{1 - \cos(\pi\rho_i \frac{x_i - x_i'}{u_i - l_i})} & \text{if } \delta_i(\underline{x}) = \delta_i(\underline{x}') = \text{true.} \end{cases} \tag{6}$$

We can use this to define a covariance over our original space. In particular, we consider the class of covariances that are functions only of the Euclidean distance Δ between points. A popular example of such a covariance is the exponentiated quadratic, for which $\kappa(\Delta) = \sigma^2 \exp(-\frac{1}{2}\Delta^2)$; another popular choice is the rational quadratic, for which $\kappa(\Delta) = \sigma^2(1 + \frac{1}{2\alpha}\Delta^2)^{-\alpha}$. We can simply take (6) in the place of Δ, returning a valid covariance that satisfies all desiderata above.

Explicitly, note that, as desired, if i is irrelevant for both $\underline{x}$ and $\underline{x}'$, d_i specifies that $g(\underline{x})$ and $g(\underline{x}')$ should not differ owing to differences between x_i and x_i'. Secondly, if i is relevant for both $\underline{x}$ and $\underline{x}'$, the difference between $f(\underline{x})$ and $f(\underline{x}')$ due to x_i and x_i' increases monotonically with increasing $|x_i - x_i'|$. The parameter ρ_i controls whether differing in the relevance of i contributes more or less to the distance than differing in the value of x_i should i be relevant. If $\rho = 1/3$, and if i is irrelevant for exactly one of $\underline{x}$ and $\underline{x}'$, $f(\underline{x})$ and $f(\underline{x}')$ are as different as is possible due to dimension i; that is, $f(\underline{x})$ and $f(\underline{x}')$ are exactly as different in that case as if $x_i = l_i$ and $x_i' = u_i$. For $\rho > 1/3$, i being

relevant for both $\underline{x}$ and $\underline{x}'$ means that $f(\underline{x})$ and $f(\underline{x}')$ could potentially be more different than if i was relevant in only one of them. For $\rho < 1/3$, the converse is true. Hyperparameter ω_i defines a length scale for the ith feature.

We call the kernel defined above the *arc kernel*. The parameters of the kernel, ω and ρ, can be optimized using the GP marginal likelihood, or integrated out using Markov chain Monte Carlo.

3 Experiments

We now show that the arc kernel yields better results than other alternatives. Bayesian optimization requires building a surrogate model of the function being optimized, and better models can be expected to lead to more efficient optimization. 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 BO performance.

Data. We use two different datasets, both of which are common in the deep learning literature. The first is the canonical MNIST digits dataset [16] where the task is to classify handwritten digits. The second is the CIFAR-10 object recognition dataset. We pre-processed CIFAR-10 by extracting features according to the pipeline given in[]. We allow the neural networks for these problems to use up to 5 hidden layers. We optimize over learning rates, weight constraints, dropout rate [17], and the number of hidden units per layer leading to a total of up to 23 hyperparameters and 6 architectures. On MNIST, most effort is spent improving the error by a fraction of a percent, therefore we optimize this dataset using the log-classification error. For CIFAR-10, we just use the regular classification error as the objective. We use the Deepnet[2] package, and each function evaluation took approximately 1000 to 2000 seconds to run on NVIDIA GTX Titan GPUs.

3.1 Model Quality Experiments

Models. Our first experiments concern the quality of the regression models used to form the response surface for Bayesian optimization. We generated data by performing 10 independent runs of Bayesian optimization on MNIST and then treat this as a regression problem. We compare the GP with arc kernel (Arc GP) to several baselines: the first baseline is a simple linear regression model, the second is a GP using a Matérn $5/2$ kernel. Irrelevant dimensions are simply filled in randomly for each input. We also compare to the case where each architecture gets its own separate model, as in [14]. The results are averaged over 10-fold train/test splits. Kernel parameters were inferred using slice sampling [**?**].

Table 1: Normalized Mean Squared Error on MNIST Bayesian optimization data

Method	Original data	Log outputs
Linear	0.876 ± 0.043	0.834 ± 0.047
Separate Linear	0.812 ± 0.045	0.737 ± 0.049
GP	0.481 ± 0.031	0.401 ± 0.028
Separate GP	0.546 ± 0.038	0.446 ± 0.041
Arc GP	$\mathbf{0.421 \pm 0.033}$	$\mathbf{0.335 \pm 0.028}$
Separate Arc GP	0.535 ± 0.030	0.440 ± 0.031

Table 1 shows that a GP using the arc kernel performs favourably to a GP that ignores the relevance information of each point.

Results.

3.2 Bayesian Optimization Experiments

KS: Will add plots in the morning and finish this section.

[2] https://github.com/nitishsrivastava/deepnet

Experimental Setup. For Bayesian optimization, we used the same process as in [11], including slice sampling and *expected* expected improvement, but not expected improvement per time spent.

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] Eric Brochu, T Brochu, and Nando de Freitas. A Bayesian interactive optimization approach to procedural animation design. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2010.

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

[3] M.A. Taddy, R.B. Gramacy, and N.G. Polson. Dynamic trees for learning and design. *Journal of the American Statistical Association*, 106(493):109–123, 2011.

[4] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*, pages 507–523, 2011.

[5] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for Hyper-Parameter Optimization. In *Proc. of NIPS'11*, 2011.

[6] F. Hutter. *Automated Configuration of Algorithms for Solving Hard Computational Problems*. PhD thesis, University of British Columbia, Department of Computer Science, Vancouver, Canada, October 2009.

[7] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Bálázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS*. 2011.

[8] G. E. Hinton, S. Osindero, and Y Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[9] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proc. of ICML'12*, pages 115–123, 2013.

[10] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proc. of KDD'13*, pages 847–855, 2013.

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

[12] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 2012.

[13] Geoffrey E. Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97, 2012.

[14] 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.

[15] Frank Hutter and Michael A. Osborne. A kernel for hierarchical parameter spaces, 2013. arXiv:1310.5738.

[16] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[17] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[18] Radford M Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003.