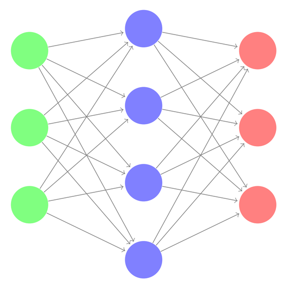
Raiders of the Lost Architecture: Kernels for Bayesian Optimization in Conditional Parameter Spaces

Kevin Swersky, David Duvenaud, Jasper Snoek, Frank Hutter, Michael Osborne

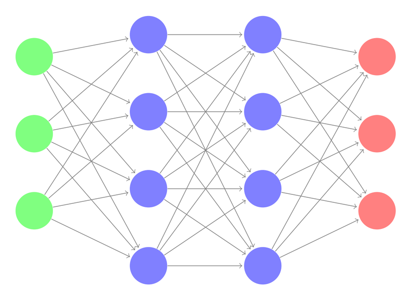
UNIVERSITY OF CAMBRIDGE

The Problem: Optimizing over Architectures

- Example: optimizing hyperparameters of a neural net.
- Can include architecture-**independent** hyperparameters: epochs, batch-size, number of layers, etc.
- Can also include architecture-**dependent** hyperparameters: learning rates, weight decays, dropout probabilities, etc.
- The number of architecture-dependent hyperparameters changes with different architectures.
- Need to optimize over a varying number of hyperparameters!
- This is difficult for Bayesian optimization with Gaussian processes because we need to define a kernel over vectors of different sizes.



One-layer MLP



Two-layer MLP

Comparing Architectures in Conditional Spaces

Formally, we aim to do inference about some function f with domain $\mathcal{X}$. $\mathcal{X} = \prod_{i=1}^{D} \mathcal{X}_i$ 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 $f(\underline{x})$.

Example

Imagine trying to model the performance of a neural network having either one or two hidden 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 which parameters are relevant, and the values of relevant parameters for both points. For example, consider 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,

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

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

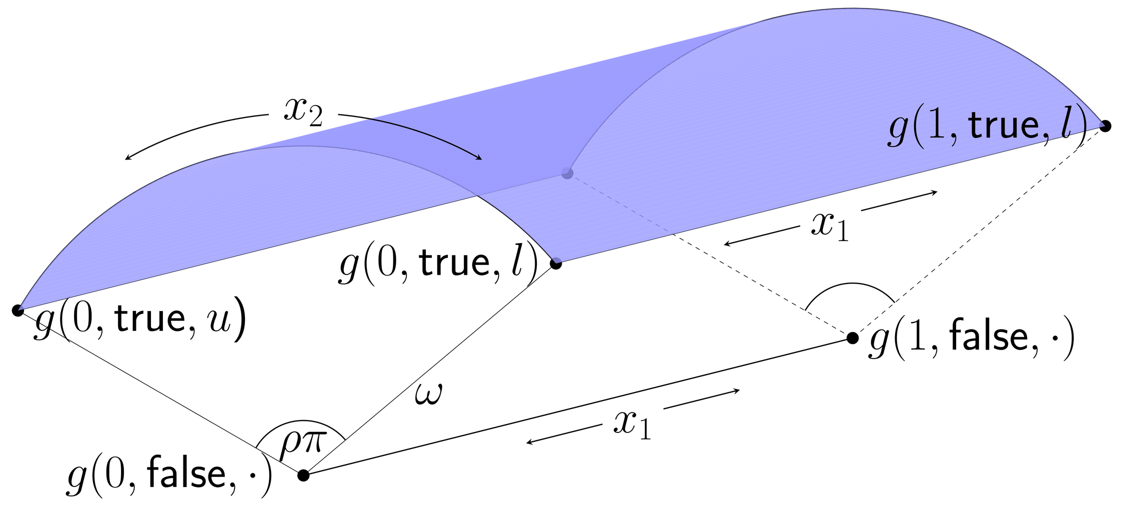
$$k\big((x_1, \text{false}, x_2), (x_1', \text{true}, x_2')\big) = k\big((x_1, \text{false}, x_2''), (x_1', \text{true}, x_2''')\big), \;\forall x_2, x_2', x_2'', x_2'''.$$

The Arc Kernel

We can build a kernel with these properties for each possibly irrelevant input dimension i by embedding our points into a Euclidean space. Specifically, we use the embedding

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

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



A demonstration of the embedding giving rise to the pseudo-metric. 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.

The figure above 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,

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

Experimental Setup

- We infer all GP parameters using MCMC with 100 steps of burn-in and 25 steps in between each trial.
- We use a Matérn kernel using the pseudo-metric described above.
- Our experiments involved optimizing a neural network with 23 hyperparameters over 6 architectures corresponding to 0 to 5 hidden layers.
- The hyperparameters are:
  - Learning rates.
  - L2 norm constraints.
  - Dropout rates.
  - Number of hidden units in each layer.
- Baseline embeds irrelevant dimensions randomly, roughly corresponds to a uniform prior over irrelevant dimensions.

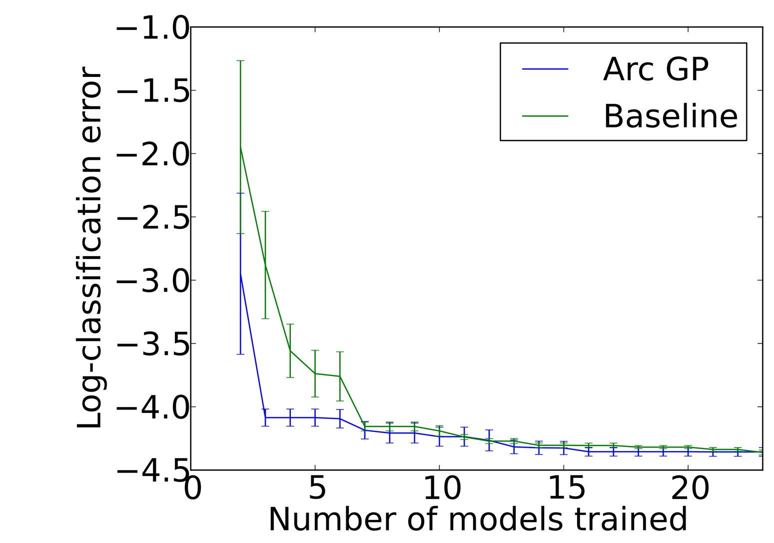
Regression Results

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

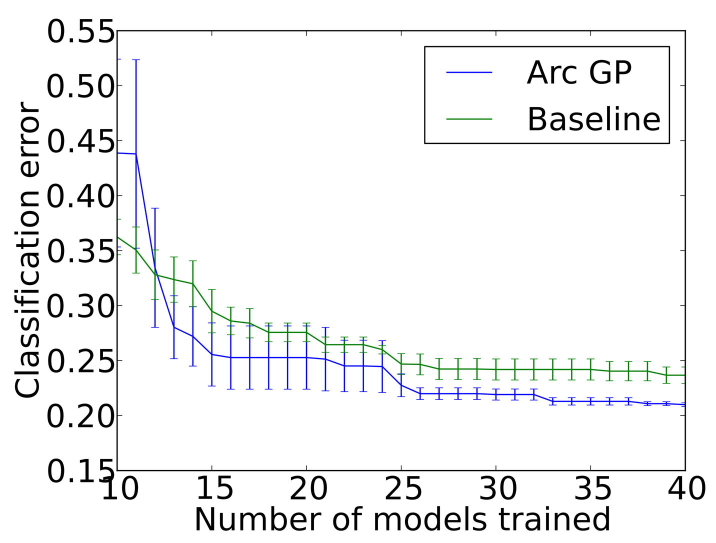
Normalized Mean Squared Error on MNIST Bayesian optimization data

Optimization Results

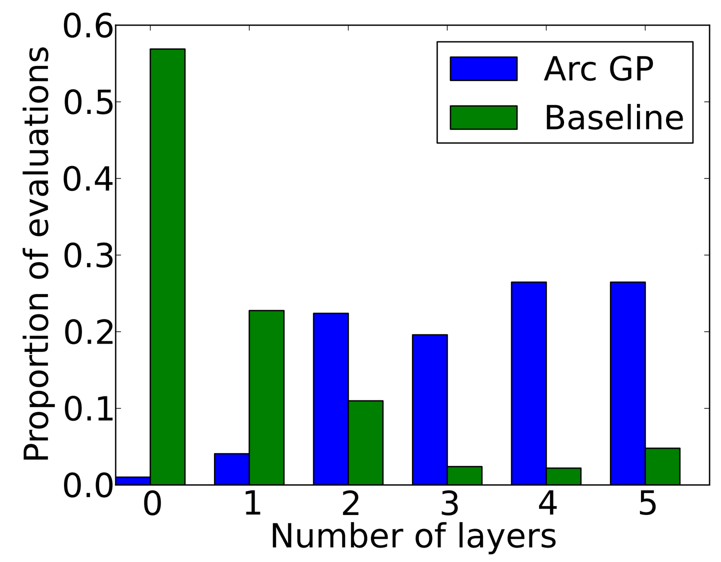
- Optimize a densely connected neural network on several datasets.
- Use k-means features (Coates et. al, *AISTATS* 2011) for CIFAR-10.



MNIST



CIFAR-10



Architectures searched

Future Work

- Comparison to more baselines.
- Use separable kernel for each parameter group.
- Extension to general DAG structures and other machine learning models.