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

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

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.

Comparing Architectures in Conditional Spaces

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

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 $(x_1, \delta_1(x), x_2)$ and assume that $\delta_1(x) = \text{true}$; 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 $x_2$:

- If we are comparing two points for which the same parameters are relevant, the value of any unused parameters shouldn’t matter,
  $$k((x_1, \text{false}, x_2), (x'_1, \text{false}, x'_2)) = k((x_1, \text{false}, x_2), (x'_1, \text{false}, x'_2)),$$
  $\forall x_1, x'_1, x_2, x'_2.$

- The covariance between a point using both parameters and a point using only one should again only depend on their shared parameters,
  $$k((x_1, \text{false}, x_2), (x'_1, \text{true}, x'_2)) = k((x_1, \text{false}, x_2), (x'_1, \text{true}, x'_2)).$$

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

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

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 $\delta_i(x) = \text{false}$ are mapped onto a line varying only along $x_i$. 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 $\delta$ but the same values of $x_2$.

The figure above shows a visualization of the embedding of points $(x_1, \delta_1(x), x_2, x_2')$ into $\mathbb{R}^4$. In this space, we have the Euclidean distance,

$$d((x, \delta(x)), (x', \delta(x'))) = ||g(x) - g(x')||_2 = \begin{cases} 0 & \text{if } \delta(x) = \delta(x') = \text{false} \\ \omega_i \sqrt{2(1 - \cos(\pi \rho_i \frac{z_i}{\sqrt{l_i u_i}}))} & \text{if } \delta(x) = \delta(x') \text{ and } \delta(x) \neq \delta(x') \text{ or vice versa.} \\ \omega_i \sqrt{2} & \text{otherwise.} \end{cases}$$

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Original data</th>
<th>Log outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separate Linear</td>
<td>0.812 ± 0.041</td>
<td>0.737 ± 0.047</td>
</tr>
<tr>
<td>Separate GP</td>
<td>0.546 ± 0.038</td>
<td>0.446 ± 0.041</td>
</tr>
<tr>
<td>Separate Arc GP</td>
<td>0.535 ± 0.030</td>
<td>0.440 ± 0.031</td>
</tr>
<tr>
<td>Linear</td>
<td>0.876 ± 0.043</td>
<td>0.831 ± 0.047</td>
</tr>
<tr>
<td>GP</td>
<td>0.481 ± 0.031</td>
<td>0.401 ± 0.028</td>
</tr>
</tbody>
</table>

Normalised 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.

Future Work

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