Bayesian Optimization is a powerful technique for finding the global optimizer of blackbox functions.

Users want to know more: which inputs are important, the effects of which ones are correlated?

We use functional ANOVA to provide such information, based on efficient operations in random forests.

Efficient Marginal Performance Predictions in Random Forests

Basic Definitions (only for reference)

Let $A$ be an algorithm having $n$ parameters with domains $\Theta_1,\ldots,\Theta_n$. We use integers to denote the parameters, and $N$ to refer to the set $\{1,\ldots,n\}$ of all parameters of $A$.

Definition 1 (Configuration Space $\Theta^N$). $A$'s configuration space is $\Theta = \Theta_1 \times \cdots \times \Theta_n$.

Definition 2 (Parameter Instanciation). A complete instanciation of an algorithm's $n$ parameters is a vector $\theta = (\theta_1,\ldots,\theta_n)$ with $\theta_i \in \Theta_i$. We also refer to complete parameter instanciations as parameter configurations. A partial instanciation of a subset $U = \{i_1,\ldots,i_k\} \subseteq N$ of $A$'s parameters is a vector $\theta_U = (\theta_{i_1},\ldots,\theta_{i_k})$ with $\theta_i = \theta_i$.

Definition 3 (Extension Set). Let $\theta_U = (\theta_{i_1},\ldots,\theta_{i_k})$ be a partial instanciation of the parameters $U = \{1,\ldots,n\}$. The extension set $X(\theta_U)$ of $\theta_U$ is then the set of parameter configurations $\Theta_U := \{\theta_U : \theta_{i_k} \in R \}$ such that $\sum \theta_{i_k} = w = l$. For cross-products $S = S_1 \times \cdots \times S_n$, $|S| = \prod |S_i|$.

Definition 4 (Range size). The range size $|S|$ of an empty set $S$ is $|0| = 1$; for other finite sets, the range size equals the cardinality: $|S| = |S|$; and for closed intervals $S = [a,b] \subseteq R$, we have $|S| = b - a$.

Main Definition and Results

Definition 5 (Marginal performance). Let $A$'s (true) performance be $y(\theta) \rightarrow R, U \subseteq N$, and $T = W$. $A$'s marginal performance $\omega_U(\theta_U)$ is then defined as

$$\omega_U(\theta_U) = \mathbb{E}(y(\Theta_{\bar{U}}) | \Theta_U \in X(\theta_U)) = \frac{1}{|\Theta_U|} \sum y(\Theta_{\bar{U}})$$

Similarly, $A$'s marginal predicted performance $\hat{\omega}_U(\theta_U)$ under a model $\hat{y}(\Theta_{\bar{U}})$ is

$$\hat{\omega}_U(\theta_U) = \frac{1}{|\Theta_U|} \sum y(\Theta_{\bar{U}})$$

Theorem 6. Given the partitioning $T$ of a regression tree $\hat{T}$ that defines a predictor $\hat{y} : \Theta \rightarrow R$, and a partial instanciation $\theta_U$ of $N$, $T$'s marginal prediction $\hat{\omega}_U(\theta_U)$ can be computed as

$$\hat{\omega}_U(\theta_U) = \frac{1}{|\Theta_U|} \sum y(\Theta_{\bar{U}})$$

Complexity with Random Forests

We can use our efficient marginal computations to conduct these importance indices efficiently:

Theorem 9. Given a configuration space $\Theta$ consisting of $n$ categorical parameters of maximal domain size $D$ and a regression tree $T$ with $L$ leaves that defines a predictor $\hat{y} : \Theta \rightarrow R$, we can exactly compute the fractions of variance explained by all subsets $U$ of $\Theta$'s parameters. Of sets up to $K$, with space complexity $O(D^2 \cdot D + n \cdot \log D)$. Additional marginal predictions cost additional space $O(1)$ and time $O(D \cdot n \cdot \log D)$.

How to Use This in Practice

- Collect performance data by running the algorithm with different parameter settings (e.g., run Bayesian Optimization)
- Fit a random forest model on that data (can e.g., be the model already used in BayesOpt)
- Determine important (pairs of) variables
- Inspect important main and interaction effects
- Future work: use within Bayesian optimization to iteratively focus on important parameters

Efficient Decomposition of Variance

Functional ANOVA (not new)

Functional ANOVA decomposes a function $f : \Theta_1 \times \cdots \times \Theta_n \rightarrow R$ into additive components that only depend on subsets of its parameters $N$.

$$f(\Theta) = \sum_{U \subseteq N} f_U(\Theta_U)$$

The components $f_U(\Theta_U)$ are defined as follows:

$$f_U(\Theta_U) = \frac{1}{|\Theta_U|} \sum_{\Theta_{\bar{U}} \in X(\theta_U)} f(\Theta)$$

The constant $f_U$ is the function's mean across its domain.

The many functions $f_U(\Theta_U)$ are called main effects and capture the effect of varying parameter $\mu$ across all instanciations of all other parameters.

By definition, the variance of $f_U$ across its domain $\Theta_U$ is

$$V_U = \frac{1}{|\Theta_U|} \sum_{\Theta_{\bar{U}} \in X(\theta_U)} (f(\Theta_{\bar{U}}) - f_U(\Theta_U))^2$$

and functional ANOVA decomposes this variance into contributions by all subsets of variables $U$ (see, e.g., Hooke, 2007, for derivation):

$$V = \sum_{U \subseteq N} V_U$$

The importance of all main and interaction effects $f_U$ can thus be quantified by the fraction of variance they explain: $f_U = V_U / V$.

Application to Auto-WEKA

768 (!) parameters

Four parameters consistently turned out to be important:
- Machine learning algorithm (out of 31 choices)
- Base algorithm to use in an ensemble
- Feature selection: scoring mechanism for feature subsets
- Feature search: search mechanism through feature subsets

Main effect of the choice of machine learning algorithm

for CIFAR-10

for MNIST

Application to solvers for hard combinatorial problems (SAT, MIP, TSP)

- State-of-the-art solvers for NP-hard problems SAT, MIP, and TSP
- Between 4 and 76 parameters (choices of heuristics, etc)
- Performance highly dependent on these parameters
- Ran SMAC [Hutter et al., 2011] ten times for each benchmark
- Achieved speedups between 1.02x and 857x over default
- Fitted random forests on the union of the performance data
- Ran functional ANOVA on the random forest

Main effects explained a large fraction of variance: see table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor 1</td>
<td>Factor 2</td>
<td>Factor 3</td>
</tr>
<tr>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
</tr>
</tbody>
</table>

Main effect of Spear’s most important parameter

(its variable selection heuristic) on instances from hardware verification (left) and software verification (right)