Thesis Working Whitepaper

Drew Blount

Mathematics Department, Reed College

dblount@reed.edu

March 9, 2015

A gold miner knows that there's gold in a valley, so she'd like to dig a mine. The valley is large, and she knows that choosing the right site for her mine is crucial. She has equipment that can drill an exploratory shaft, and report how much gold is underneath a specific spot in the valley (telling her how profitable it would be to build her mine there). By drilling many such holes, she could get a complete map of the gold in the valley, and use this map to find the perfect spot for the mine—but she doesn't nearly have the resources to blanket the valley in exploratory shafts. How can our miner find the best site for her mine, while drilling as few exploratory holes as possible?

This thesis is an exploration of problems analogous to the miner's—the optimization of expensive black-box functions. These are functions that you can execute. there is a function whose inner workings are unknown (our miner isn't schooled enough in geology to have any a priori theory about the distribution of gold in her valley)

This thesis is an exploration of an algorithm that optimizes black-box functions, which are functions that you can execute, but do not understand. We're concerned only with deterministic functions here, and we assume that points that are close in input space will be close in output-space, too. Other than those requirements, there are no assumptions made about the functions we wish to optimize. The only tool of information-retrieval, in this optimization

This thesis is an exploration of the question, is there a general way to do things well, when you don't know what you're doing?

To formalize the framing, how do you optimize a function ("accomplish a task well") in general? For simplicity, we assume that the function is a deterministic mapping of inputs to outputs. The task of optimization is then a matter of finding the set of inputs that produces the most desirable output. In order for an optimization strategy to be truly general, it must

In this context, we assume that there is some deterministic process at play, though we do not know anything about that process besides what

For the strategy to be general, it must require no particular kind of domain-specific knowledge about the function being optimized. Of course, the question of optimization

1 Notes on Automated Algorithm Configuration and Parameter Tuning

NP-hard problems come up often in the real world, so practicable solutions are needed to make brute force searches tractable. This is often achieved with complex heuristic techniques, often several at once, interacting in complex ways. These heuristics for pruning and exploring search-space are themselves (the heuristics) defined by certain parameters, and the setting of these parameters greatly affects the efficiency and effectiveness of the techniques.

There's a good framing of the *algorithm configuration* or *parameter tuning problem* on p.38.

Three classes of methods for solving this problem:

- *Racing procedures*: Very simple. In parallel (or sequentially), evaluate candidate solutions with some series of benchmarks, and continually prune candidates whose performance is a certain measure worse than the current leader.
- *ParamILS*: An 'iterative first-improvement search in the space of configurations'; Starting at one configuration, individual params are changed one-by-one, and any change that doesn't improve some fitness measure is immediately undone.
- *Sequential Model-Based Optimization*: I'll want to essentially include Figure 3.4 (page 56).

2 Notes on Gaussian Processes for Global Optimization

2.1 The Model Selection Problem

For a given set of data (points in design space where the value of the objective function is known), there are an infinite number of possible functions that could have generated them, and an infinite number of models we might adopt which would interpolate correctly. The task of choosing from this set of potential models, of selecting the best model for given data, is the model selection problem.

[It would be good to include a quick figure here showing a few data points in a 2d function with two different DACE models fit to them, and a caption along the lines of, "each model interpolates correctly, so on what basis do you choose one over the other?"]

As this thesis is concerned with deterministic processes, an intuitive constraint is that any model must interpolate correctly, that is, $\hat{y}(x) = y(x)$ for x an evaluated point in input-space. Rephrased again, we're only interested in model functions that accurately predict the objective function values we have seen so far. As illustrated in Figure [ref figure mentioned above], a given data set allows for several such functions, and indeed, the number of relatively simple model functions that interpolate correctly is in general infinite.

Therefore, the model selection problem can only be resolved with some notion of a model's *prior probability*, that is, a probabilistic framework that is independent of the data. This framework and its assumptions If one is hoping to optimize a black box function, it might seem paradoxical to have some prior basis

3 The EGO Algorithm

The EGO algorithm Jones et al. (1998) proceeds in three steps:

This steps proceed in a cycle until a global optimum is found.

4 The Predictor Function

The predictor function used by the EGO algorithm has also been called the DACE model, for "design and analysis of computer experiments". The starting point of the DACE model is parameterized distance function for points in input-space,

$$d(\mathbf{x}, \mathbf{x}') = \sum_{h=1}^{k} \theta_h |x_h - x'_h|^{p_h}, \tag{1}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are k-dimensional vectors. The parameters $\theta_1, ..., \theta_k \equiv \theta$ and $p_1, ..., p_k \equiv \mathbf{p}$ effectively define a particular DACE predictor function [1]. When using the DACE predictor, then, the second stage of the EGO algorithm process diagrammed in Fig. **??** consists of choosing $\mathbf{p}$ and θ in a manner that best fits whatever data is known about the objective function. Before we can discuss the selection of these parameters, we must understand how they are used to generate a predictor function.

Intuitively, the DACE predictor works by correlating the output of the objective function with variation along each (assumed-to-be-) independent dimension of the input space. This correlation function is defined as simply the negative exponent of the parameterized distance:

$$Corr(\epsilon(\mathbf{x}), \epsilon(\mathbf{x}')) = e^{-d(\mathbf{x},\mathbf{x}')} \tag{2}$$

This correlation function behaves as we wish it would. In the original words of Jones et al, "... when the distance between $\mathbf{x}^i$ and $\mathbf{x}^j$ is small, the correlation is near one. Similarly, when the distance between the points is large, the correlation will approach zero.

We define the correlation matrix $\mathbf{R}$ as the $n \times n$ matrix, where

$$\mathbf{R}_{(i,j)} = Corr(\epsilon(\mathbf{x}^{(i)}), \epsilon(\mathbf{x}^{(j)})) \tag{3}$$

between objective function values at different points in input space[2]. When two points are close in input-space, their function value should be close. The details of this relationship will be sensitive to the data and hand. Thus the starting point of the DACE process is constructing a measure of distance between input values from the already-evaluated input-output pairs.

First, some definitions and variables. I've copied the conventions of Jones et al.

Sample Point : a point in input space where the blackbox function has already been evaluated

n : the number of sample points

k : the dimensionality of input space

$\mathbf{x}^{(i)}$: the ith sample point (a k-vector)

$\mathbf{x}$: the vector of sample points (n k-vectors, an $n \times k$ matrix)

y : the objective function

$\mathbf{y}$: the vector of evaluated outputs, i.e. $\mathbf{y}_i = y(\mathbf{x}^{(i)})$

[1] Jones et al p. 460 describes how the two additional parameters, the output y's mean and standard deviation, can be predicted from

[2] Precisely, the DACE predictor deals not with correlation between objective function values, but the deviation of objective function values from the mean.[is that correct?]

$p_i \in [1, 2\ ], \theta_i \geq 0$: two parameters associated with the ith input dimension

Tuning the parameters $\theta_1, ..., \theta_k$ and $p_1, ..., p_k$ to the empirical data $\mathbf{x}$ and $\mathbf{y}$ is the real computational task in constructing the DACE predictor.

4.1 Likelihood

The parameters p and θ are chosen by maximizing the likelihood function,

$$\frac{1}{(2\pi\sigma^2)^{n/2}|\mathbf{R}|^{\frac{1}{2}}}\ exp\left[-\frac{(\mathbf{y}-\mathbf{1}\mu)'\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\mu)}{2\sigma^2}\right] \tag{4}$$

Where the best estimates of μ, σ^2 are,

$$\hat{\mu} = \frac{\mathbf{1}'\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}, \tag{5}$$

and,

$$\hat{o}^2 = \frac{(\mathbf{y}-\mathbf{1}\hat{\mu})'\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\hat{\mu})}{n}. \tag{6}$$

Plugging Eqs **??** into Eq. 4, gives the concentrated likelihood function that is in optimized to fit the parameters p and θ to a given dataset.

4.2 Distance and Correlation Between Sample Points

4.3 Predictor

The best linear unbiased predictor of the function's output at $\mathbf{x}^*$ is,

$$\hat{y}(\mathbf{x}^*) = \hat{\mu} + \mathbf{r}'\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\mu) \tag{7}$$

4.4 Examples

Any application of the DACE model requires an initial set of sample points. Jones et al use the heuristic of sampling

5 Choosing the Next Sample Point

Here, I'll apply the algorithm to a very simple case where

References

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.