Sequential Model-Based Optimization

– of –

Expensive Blackbox Functions

Drew Blount

Mathematics Department, Reed College

dblount@reed.edu

March 11, 2015

Chapter 1

The SMBO Paradigm

Sequential model-based optimization is an accurately named process, because it attempts to optimize objective functions by developing a sequence of models. The broad idea is this: you want to understand some process's global behavior from a few sample points, and you have a limited ability to collect more data–think of additional samples as available, but expensive to gather. This is often the case with processes that are difficult to observe or simulate.

Given a few sample points, you use the available data to develop a model of the process's behavior. With this model alone, you could perform simple *model*-based optimization, using it to predict the locations of global optima. In *sequential* model-based optimization, however, we use predictive models in a more clever, bootstrapping way: to predict what further data, if collected, would allow us to improve our model—and thus our predictive ability—the most. In other words, each model is used to generate another, better model. Global optima, or at least very good solutions, are then found by recursively improving the predictive ability of a model.

To make this process more tangible, consider the case of gold-mining, which is a surprisingly deep analogy to the kind of data mining which is explored in this thesis. Imagine that you are a gold-miner, and you own a claim to Valley X. You want to understand where in the valley you could find the most gold, where best to start a mine. Your mining company has drilled five exploratory shafts throughout the valley. You make a map of the valley's gold distribution based on these five samples.

Were you, the gold miner, only interested in model-based optimization, you would use this rudimentary map to predict where the most gold in the valley is, and start your mine there. Say, however, that you have enough resources to drill five more exploratory shafts first. The clever miner then asks themself, "where could I drill the next exploratory shaft, to best learn about where the gold is most concentrated in the valley?" By leveraging what they have learned from the first five data points, the miner finds the region they would like most to learn about. After drilling the sixth hole in this region, the miner can improve their model of the gold distribution yet again, and ask the same question: "where should I investigate next to find the most gold?" The goal of this thesis is to automate this decision-making process.

1.1 History

[Just a few paragraphs here, talk first about actual kriging (which was done for gold) and maybe get a fun historical anecdote]

[Talk about the EGO paper (Jones et al., 1998)–Hutter calls it the beginning of SMBO, "limited to optimizing continuous parameters for noise-free functions (i.e., the performance of deterministic algorithms)." (Hutter et al., 2011, p. 509) See citation for further discussion of SMBO history. Talk about the background of EGO authors, how Jones worked for GM and one of the case studies in the original paper involved 3D engine component design.]

Mention "PDT^{TM}," ProtoLife's "predictive design technology," which is illustrated by a figure very similar to Fig 1.1 ProtoLife (2013)

Talk about Hutter's recent work, maybe Google talk? Perhaps some words about how the ML community isn't particularly interested right now, and is more interested in accomplishing extremely huge-dimensional mapping problems like vision and translation, which are being accomplished by deep nets.

1.2 The SMBO Cycle

This thesis will explore several algorithms that are classified as sequential model-based optimizers Hutter et al. (2011); Hamadi et al. (2012); Jones et al. (1998); Rasmussen and Williams, C. K. I. (2006). They differ in the space of model functions they explore, their assumptions regarding determinism, and their procedural relationship to the objective functions being optimized. Nonetheless, they can all be roughly summarized by a three-part while loop,

1. the objective function is evaluated at some set of input points
2. a working model of the objective function (the "predictor function") is generated from the evaluated input-output pairs
3. new points are chosen to be evaluated by the objective function, so that the working model's utility in optimizing the objective function will be maximized.

[note: should the above be in pseudocode? It would at least be nice to reference it like a figure.]

The loop is run until results are satisfactory, or the expected improvement from further iterations falls below a user-defined threshold. Because of the circular, bootstrapping nature of the algorithm, it is tempting to illustrate it with a triangle like we see on recycling bins, as shown in Figure 1.1

Tho company ProtoLife uses a similar illustration to Fig. 1.1 to describe their analytical product, which appears to be a powerful research applications of SMBO ProtoLife (2013). The top node in the triangle is determined by each domain-specific application. Like ProtoLife, I am interested in trying several different model modules (the bottom-right node in the figure) to optimize different objective functions (the top-right node).

[Hutter aslo includes in his introduction an image of a predictor function w/ expected improvement overlayed, like in Jones, right here]

1.3 Terms, Symbols

The function which we wish to optimize is the *objective function*. The *predictor* or *model function* is that generated in each iteration of the *SMBO loop* (or *cycle*).

Sample points are those points where we have evaluated the objective function–the points on which we base our model function. Throughout this thesis, I'll use several letters and symbols for specific concepts and indices. So that there is a definitive reference, I have listed them here (make this an official table/figure?).

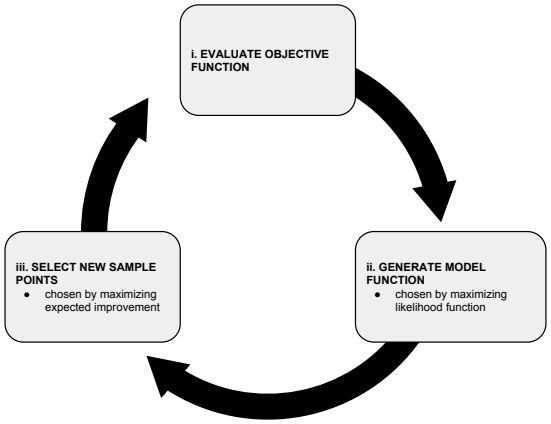

Figure 1.1: the three iterated stages of the SMBO process

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

$\hat{y}$: the predictor function

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

Chapter 2

Case study: the EGO Algorithm

The EGO algorithm is named for the paper in which it was presented, the informatively titled, "Efficient Global Optimization of Expensive Blackbox Functions" (Jones et al., 1998). I treat the EGO algorithm as the quintessential SMBO–it was the first, and it makes simple and intuitive assumptions about both the objective function and the best method to model it. [Kinda a lame explanation, but there has got to be more justification for treating EGO so centrally].

2.1 The DACE Predictor

The model that is sequentially fit in the EGO algorithm is known as the DACE predictor. Like EGO, the DACE acronym comes from a somewhat generally-titled paper, in this case, "Design and Analysis of Computer Experiments" (Sacks et al., 1989). [Might want to elaborate later. For now, here's the bare info:] The assumptions that the DACE predictor makes are these (following closely Jones et al's presentation of DACE Jones et al. (1998)):

First, we assume what is called a *stochastic process model* [get cites/explanation from Jones p.456], i.e., that,

$$y(\mathbf{x}^{(i)}) = \mu + \epsilon(\mathbf{x}^{(i)}) \quad \text{for } i \in (1, 2, ..., n). \tag{2.1}$$

As is common in statistics, μ represents the mean of the process. Note that the above equation appears simpler than even linear regression, as it has no functional component. The DACE model, and stochastic processes in general, instead contains its predictive power in the 'error terms' $\epsilon(\mathbf{x}^{(i)})$. These terms are assumed to be distributed normally,

$$\epsilon(\mathbf{x}^{(i)}) = \text{Normal}(0, \sigma^2) \quad \text{for } i \in (1, 2, ..., n), \tag{2.2}$$

for a process-wide σ^2. Despite the normal distribution, the $\epsilon(\mathbf{x}^{(i)})$ are very much *not* independent of each other: it is in a complex error-correlation structure that the DACE model encodes the contours of its response surface. Specifically,

$$\text{Corr}\left(\epsilon(\mathbf{x}^{(i)}), \epsilon(\mathbf{x}^{(i)})\right) = \text{Exp}\left[-\sum_{h=1}^{k} \theta_h \left|\mathbf{x}_h^{(i)} - \mathbf{x}_h^{(j)}\right|^{p_h}\right] = \prod_{h=1}^{k} \text{Exp}\left[-\theta_h \left|\mathbf{x}_h^{(i)} - \mathbf{x}_h^{(j)}\right|^{p_h}\right], \tag{2.3}$$

where the free parameters $\{(\theta_i, p_i)$ for $i \in (1, 2, ..., k)$ determine the shape of the DACE predictor. Note that these three constraints completely describe the DACE model: we assume that we are

modeling a stochastic process with mean μ, standard deviation σ^2, and inter-sample correlation described by Eq. 2.3.

Here I should discuss what error correlation means with some pretty pictures like in Jones p.459.

Much more can (and should, and will) be said about the shape of the predictor implied by this correlation equation, but for now it suffices to say that it encodes the heuristic, "points near each other in input-space should have nearby function values", with a concept of nearness along each input dimension that is gaussian in shape, with magnitude and falloff-steepness determined by θ and p.

2.1.1 Selection of $\{(\theta_i, p_i)\}$

To fit a DACE predictor to sample data, the $k+2$ free parameters μ, σ^2, and $\{(\theta_i, p_i)$ for $i \in (1, 2, ..., k)$ are set by maximizing the likelihood function which is implied by the prior assumptions **??**. I will here walk through the derivation of that likelihood function.

Likelihood, I'll remind the reader, is a statistical concept very similar to probability. Imagine a scenario where parameters ψ give rise to some model f_ψ. Given some empirical observations Z, the concept of likelihood allows us to quantify how well the parameter choice ψ and the generated model f_ψ match our empirical observation—likelihood tests how well a model matches data. This is done quite simply: by defining the likelihood of a set of parameters given an observation, as the probability of that observation, given those parameters:

$$L(\psi|Z) = Pr(Z|\psi). \tag{2.4}$$

Note that this works intuitively well with our notion of probability: a set of parameters is a good one, i.e. it is likely, if the model that it generates is one wherein our observed data are relatively probable. The best model is that which is most likely, meaning that under no other model would the observed data Z appear more probable.

Here, our observed data is the vector of witnessed function values $\mathbf{y}$, so finding an equation for the likelihood of our parameters is equivalent to deriving the joint probability distribution of this n-vector, given the assumptions laid out in the previous section. I will now derive this joint distribution, using techniques that should be familiar to those schooled in college-level statistics. I'll first derive a simpler distribution, then use the change of variables formula to arrive at the result.

Consider a set of independent, identically-distributed gaussian random variables $\mathbb{Z} = Z_1, ..., Z_n$, with mean 0 and standard deviation σ^2. For the sake of being explicit, the expectation function f_{Z_i} for each Z_i is then,

$$f_{Z_i} = \frac{1}{\sigma\sqrt{2\pi}} e^{Z_i^2/\sigma^2}. \tag{2.5}$$

The joint probability distribution of $\mathbb{Z}$ is simply the product of each of its [independent] components:

$$
\begin{aligned}
f_{\mathbb{Z}}(\mathbb{Z}) &= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{Z_i^2/\sigma^2} \qquad (2.6)\\
&= \frac{1}{(2\pi\sigma^2)^{n/2}} Exp\left[\frac{1}{\sigma^2}\sum_{i=1}^{n} Z_i^2\right]\\
&= \frac{1}{(2\pi\sigma^2)^{n/2}} e^{\mathbb{Z}^T\mathbb{Z}/\sigma^2},
\end{aligned}
$$

where the last line uses vector multiplication to denote the sum of the square components of $\mathbb{Z}$.

Now, we use the change of variables formula to derive from this an equation for our actual data. For convenience, I'll consider the probability distribution of the vector of error terms, ϵ, rather than simply the output vector $\mathbb{y}$. If $\mathbb{1}$ denotes the 1-vector, then,

$$\epsilon = \mathbb{y} - \mathbb{1}\mu. \qquad (2.7)$$

I will also consider $\mathbb{R}$, the *correlation matrix* of the errors ϵ. $\mathbb{R}$ is simply the $n \times n$ matrix whose $(i, j)^{th}$ entry is $Corr[\epsilon_i, \epsilon_j]$ as defined in Eq 2.3. By construction, we know that $\mathbb{R}$ is symmetric and positive-definite, which means that a Cholesky decomposition exists CITE**?**, i.e., that there exists a lower-triangular matrix A such that,

$$\mathbb{R} = AA^T. \qquad (2.8)$$

Now, because of [WHY? is it the lower-triangularity of A? I don't see how...], we know that the matrix multiplication of A and ϵ will produce a vector of mutually independent variables sharing the Normal$(0, \epsilon)$ distribution of each component of ϵ. That is,

$$A\epsilon = \mathbb{Z}, \qquad (2.9)$$

where $\mathbb{Z}$ is and independent, identically distributed vector with each component Normal$(0, \epsilon)$. We will use this fact, along with the change of variables formula and the above-derived probability density function for such a $\mathbb{Z}$ (Eq 2.6), to arrive at the likelihood equation. Letting g denote the linear transformation equivalent to left-multiplication by the matrix A, we see that,

$$
\begin{aligned}
g(\epsilon) &= \mathbb{Z} \qquad (2.10)\\
\epsilon &= g^{-1}(\mathbb{Z}) = A^{-1}\mathbb{Z} \qquad (2.11)
\end{aligned}
$$

Now recall the change of variables formula, which given Eq. 2.11 states that,

$$
\begin{aligned}
f_\epsilon &= f_{\mathbb{Z}}\left(g^{-1}(\mathbb{Z})\right) \times \left|g^{-1}(\mathbb{Z})\right| \qquad (2.12)\\
&= f_{\mathbb{Z}}\left(A^{-1}\epsilon\right)|A^{-1}| \qquad (2.13)\\
&= f_{\mathbb{Z}}\left(A^{-1}\epsilon\right)\frac{1}{|A|} \qquad (2.14)
\end{aligned}
$$

Now plugging in Eq. 2.6, the formula for $f_{\mathbb{Z}}$,

$$f_\epsilon = \frac{1}{(2\pi\sigma^2)^{n/2}|A|} Exp\left[\frac{1}{\sigma^2}(A^{-1}\epsilon)^T(A^{-1}\epsilon)\right]. \tag{2.15}$$

$$\tag{2.16}$$

Now I will do some linear algebra to the matrices in the exponent,

$$\begin{aligned}(A^{-1}\epsilon)^T(A^{-1}\epsilon) &= \epsilon^T(A^{-1})^T A^{-1}\epsilon \\ &= \epsilon^T((A^T)^{-1}A^{-1})\epsilon \\ &= \epsilon^T(AA^T)^{-1}\epsilon \\ &= \epsilon^T\mathbb{R}^{-1}\epsilon \end{aligned} \tag{2.17}$$

Also rewriting $|A|$ as $|\mathbb{R}|^{1/2}$, because the determinant of a product is the product of the determinants, and Eq. 2.8, we get the clean formulation,

$$\frac{1}{(2\pi\sigma^2)^{n/2}|\mathbf{R}|^{1/2}}\ Exp\left[\frac{\epsilon^T\mathbb{R}^{-1}\epsilon}{\sigma^2}\right]. \tag{2.18}$$

[ASK ALBYN:Why am I missing a factor of $-\frac{1}{2}$ over Jones's result (Eq 2.21)?]
Recall the definition of correlation, that for random variables a and b,

$$\text{Corr}(a,b) = \frac{E\left[(a - E(a))\,(b - E(b))\right]}{\sigma_a\sigma_b}, \tag{2.19}$$

where E is the expected-value function. Now considering Eq. 2.2, we have a simple equation for the correlation of two errors,

$$\text{Corr}(a,b) = \frac{E\left[\epsilon(\mathbf{x}^{(i)})\epsilon(\mathbf{x}^{(i)})\right]}{\sigma^2} \tag{2.20}$$

From here, you can derive the likelihood equation,

$$\frac{1}{(2\pi\sigma^2)^{n/2}|\mathbf{R}|^{1/2}}\ exp\left[-\frac{(\mathbf{y} - \mathbf{1}\mu)'\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}\right]. \tag{2.21}$$

If we fix the $\{(\theta_i, p_i)\}$, the above equation can be analytically maximized in μ and σ^2 (Jones et al., 1998) to get,

$$\hat{\mu} = \frac{\mathbf{1}'\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}, \tag{2.22}$$

and,

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})'\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu})}{n}. \tag{2.23}$$

Note that when $\mathbf{R} = \mathbf{I}$, the identity matrix, there is no correlation between each variable. As you would hope, in this case Eq. 2.22 and Eq. 2.23 reduce to the standard statistical definitions of mean and standard deviation.

$$asdfasdfasd \tag{2.24}$$

Chapter 3

Implementation

3.1 Visual Examples with EGO

3.1.1 1D

–I've already made the software for this. Narrate a few examples. Probably a one-page grid of figures showing initial sample points, selection of next points, update of the model, etc.

3.1.2 2D

–Perhaps the Branin function shown in the original Jones paper? the same grid would be good to look at in 2D.

3.2 Performance Tests

–Optimize a SAT solver

–Compare to other optimizers and perhaps SMBOs besides EGO

–Maybe use a neural net model function?? (how do we measure expected improvement–how does Norman?)

–Here is where I'll dump all of my time for the rest of March once I'm done with the example implementations.

Bibliography

Hamadi, Y., Monfroy, E., and Saubion, F., editors (2012). *Autonomous Search*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hutter, F., Hoos, H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In Coello, C., editor, *Learning and Intelligent Optimization*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer Berlin Heidelberg.

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.

ProtoLife (2013). ProtoLife PDT overview. http://protolife.com/npdtov.html. Accessed: 2015-02-26.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statist. Sci.*, 4(4):409–423.