A Two-Phase Approach for Simulation-Based Optimization

Geng Deng Michael C. Ferris

University of Wisconsin-Madison

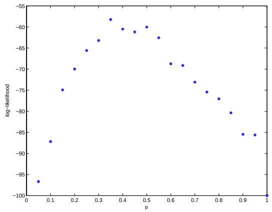
INFORMS International Hong Kong 2006

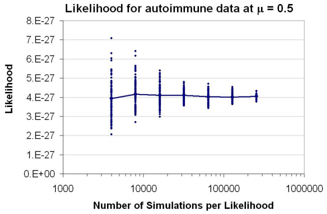
Simulation-based optimization problem

- Computer simulations are used as substitute to evaluate complex real systems.
- Simulations have been used efficiently in supply chain management, engineering design, medical treatment, and many other fields.
- The goal: Optimization finds the best values of the decision variables (design parameters or controls) that minimize some performance measure of the simulation.

A simulation example: Hypermutation rate

- Estimate mutation rate by maximizing a likelihood function
- The objective function is highly volatile. For $\mu = 0.5$:





What makes a simulation response function different?

The objective function

- typically does not have a closed form, thus cannot provide gradient or hessian information.
- is normally computationally expensive.
- is affected by uncertainties in simulations.
- typically does not have an interface to optimization routines. Therefore, the optimization routines can only utilize the existing simulation results (offline optimization).

A general problem formulation

We formulate the simulation optimization problem as

$$\min_x F(x) = E[f(x, \omega(x))]. \tag{1}$$

Comments:

- This is a stochastic optimization problem.
- The underlying function $F(\cdot)$ is unknown and to be estimated.
- Instead of expectation, other formulations such as minimum or maximum of $f(x, \omega(x))$ are acceptable in different contexts.
- For simplicity, we assume

$$f(x, \omega(x)) = F(x) + \omega(x).$$

- $\omega(x)$ are independently distributed.

Design surrogate models

- Surrogate model approximates F with a mathematical model A.
- The goal: Optimization procedures are executed over the inexpensive model A, instead of the expensive cost function.
- Determine a surrogate model by minimizing sum of difference errors:

$$\begin{array}{ll} \min & \sum_{x_i \in \mathcal{S}} \Theta \left(\frac{1}{J_i} \sum_{j=1}^{J_i} f(x_i, \omega_j) - A(x_i) \right) \\ s.t. & A(\cdot) \in \mathcal{A} \end{array} \tag{2}$$

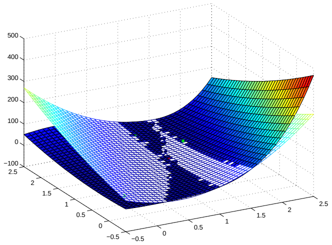
  Here $\Theta(\cdot)$ is a merit function, which is typically chosen as an l^2-norm, and the set $\mathcal{A}$ is a class of tunable functional forms.
- The functional forms $\mathcal{A}$ should be carefully specified.

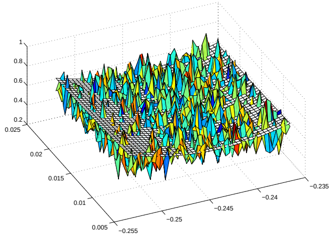
Model error and random error

The total error comes from two sources:

- Model error is due to an inappropriate choice of functional forms $\mathcal{A}$ to fit the data.
- Random error is directly induced by the uncertainty ω in the sample response.

Illustrations of global view and local view:





Two-phase framework

1. **Phase I is a global exploration step.** The algorithm explores the entire domain and proceeds to determine potentially good subregions for future investigation.
2. **Phase II is a local exploitation step.** Local optimization algorithms are applied to determine the final solution.

- Different phases serve different purposes.
- Algorithms are specially designed in either phase.

Phase I offline method: Classifier

Instead of approximating F, a classifier works as a surrogate function to the indicator function

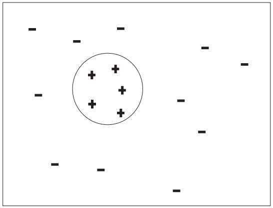
$$I(x) = \begin{cases} 1, & \text{for } x \text{ in a promising subregion;} \\ 0, & \text{otherwise.} \end{cases}$$

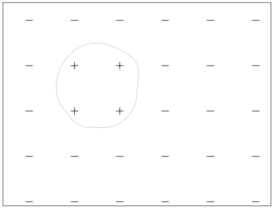
Why do we use classifiers?

- In Phase I, we are really concerned about function $I(x)$.
- Use a much smaller amount of samples.
- Works in high dimensional cases.

Phase I offline method: Classifier

The process of training and evaluating classifiers:





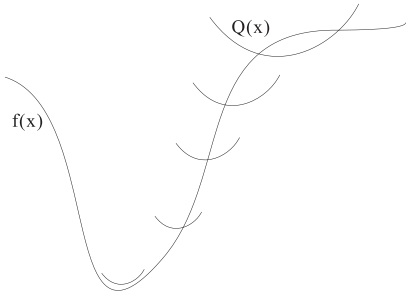
(Left figure) For the training data, samples in the level set $L(c)$ are classified as positive and the others are classified as negative. The solid circle represents the level set $L(c)$. (Right figure) Classification is performed on the new data, which are evenly distributed on a mesh grid. Four points are predicted as positive and rest are negative. The dotted circle represents an estimated boundary for the level set $L(c)$.

Phase II method

- The goal: The Phase II algorithm solves the optimization problem in the subregion, in which the objective function is subject to relatively large noise.
- A typical method is to fit a local quadratic model, either by interpolation or regression.
- Modify existing deterministic algorithm. General approach: Estimate the underlying function by averaging multiple replications.
- Potentially difficulty:
  efficiency of algorithm VS number of simulation runs

The Noisy UOBYQA algorithm (Powell)

The UOBYQA algorithm is based on a trust region method. It constructs a series of local quadratic approximation models of the underlying function f. Each quadratic model is constructed by interpolating a set of points.



G. Deng, M. C. Ferris. Adaptation of the UOBYQA algorithm for noisy functions. In *Proceedings of the 2006 Winter Simulation Conference*, 2006

Where is the problematic parts in the deterministic algorithm?

...
for iteration $k = 1, 2, \cdots$
(a) Construct a quadratic function
$Q(x) = f(x_k) + g_Q^T(x - x_k) + \frac{1}{2}(x - x_k)^T G_Q(x - x_k)$.
(b) Solve the trust region subproblem:

$$\begin{aligned} s_k = \arg\min_s \quad & Q(x_k + s) \\ s.t. \quad & \|s\|_2 \leq \Delta \end{aligned}$$

(Reduce the quadratic model variance)

...
(e) Update a new iterate x_{k+1} by comparing function values $f(x_k)$ and $f(x_k + s_k^*)$. (Use pairwise comparisons)

...

How to stabilize the quadratic model?

- Given an interpolation set $\mathcal{I} = \{y^1, y^2, \cdots, y^L\}$. Quadratic interpolation model is a linear combination of Lagrange functions:

$$Q(x) = \sum_{j=1}^{L} f(y^j) l_j(x), x \in \mathbb{R}^n, \tag{3}$$

- Each piece $l_j(x)$ is a quadratic polynomial, satisfying

$$l_j(y^i) = \delta_{ij}, i = 1, 2, \cdots, L.$$

- The coefficients of l_j are uniquely determined, regardless of the random objective function.

Bayesian estimation of coefficients c_Q, g_Q, G_Q

In Bayesian approach, the mean of function output $\mu(y^j)$ is considered as a random variable:
Normal posterior distributions:

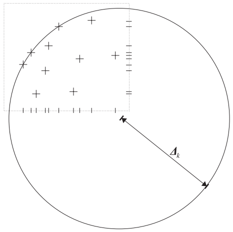
$$\mu(y^j)|X \sim N(\bar{\mu}(y^j), \hat{\sigma}^2(y^j)/r_j) \tag{4}$$

Thus the coefficients of the quadratic model are estimated as:

$$\begin{array}{rcl} g_Q|X & = & \sum_{j=1}^{L}(\mu(y^j)|X)g_j, \\ G_Q|X & = & \sum_{j=1}^{L}(\mu(y^j)|X)G_j. \end{array} \tag{5}$$

- g_j, G_j are coefficients of Lagrange functions l_j.
- g_j, G_j are determined by points y^j.

Bayesian validation of solutions



Trial solutions are generated within a trust region. The standard deviation of the solutions are constrained.

$$\max_{j=1}^{n} std([s^{*(1)}(j), s^{*(2)}(j), \cdots, s^{*(N_t)}(j)]) \leq \beta\Delta_k. \tag{6}$$

Selecting the next iterate and the new termination criterion

- Compare two points x_k and $x_k + s_k^*$ using pairwise comparison. The new iterate is set as the better point. (refer to previous slide)
- New termination criterion to stop the algorithm appropriately.

How to select the best system?

Selecting the best system is equivalent to solving a discrete optimization problem:

$$\arg\min_i E(X_i), \tag{7}$$

Existing approaches:

- Indifference-zone ranking and selection: Find the best solution within δ.
- The Bayesian approach uses posterior distributions to estimate the probability of correct selection (PCS).

Calculating PCS via posterior distributions

- Normal posterior distribution:

$$\mu_i|X \sim N(\bar{x}_i, \hat{\sigma}_i^2/r_i). \tag{8}$$

- Pairwise comparison:

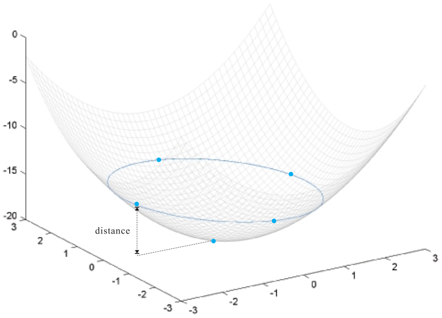
$$PCS = Pr(\mu_1 \leq \mu_2) \sim Pr(\mu_1 \leq \mu_2|\, X) = Pr(\mu_1|X - \mu_2|X \leq 0). \tag{9}$$

- Multiple comparisons (Bonferroni inequality):

$$\begin{aligned} PCS &= Pr(\mu_b - \mu_i \leq 0, i = \{1, 2, \cdots, K\} \setminus \{b\}) \\ &= 1 - \textstyle\sum_{i=1, i \neq b}^{K} Pr(\mu_b - \mu_i > 0) \end{aligned} \tag{10}$$

- Benefit of Bayesian approaches:
  - Utilize mean and variance information
  - Simple and direct to implement
  - Without using indifference-zone parameter δ

New termination criterion



The ‘distance’ between two points is computed via the quadratic model. It is compared with the separable distance d to test the separability between the points.

A numerical test

Table: The Performance of Noisy UOBYQA for the Rosenbrock Function, with $n = 2$ and $\sigma^2 = 0.01$.

Iteration (k)	FN	$F(x_k)$	Δ_k
1	1	404	2
20	78	3.56	9.8×10^{-1}
40	140	0.75	1.2×10^{-1}
60	580	0.10	4.5×10^{-2}
80	786	0.0017	5.2×10^{-3}
✓ Stops here with the new termination criterion			
100	1254	0.0019	2.8×10^{-4}
120	2003	0.0016	1.1×10^{-4}
✓ Stops here with the termination criterion $\Delta_k \leq 10^{-4}$			