Incorporate search phase to design phase in surrogate-based methods for black-box optimization

Ky Vu

1 LIX, École Polytechnique, France.
Email: vukhacy@lix.polytechnique.fr

June 17, 2015

Abstract

In all surrogate-based methods for black-box optimization, we have to find an initial set of points to evaluate (design phase) before starting to search for optimal solutions (search phase). The purpose of the design phase is to predict the global behavior of the black-box function, which helps us identify potential locations to explore. However, the design phase is done without using any information from function evaluations. In practical point of view, by doing that way we waste invaluable information to improve the quality of the design set. In his paper, we propose several methods to combine the two phases, so that at the end of the design phase, we have a set of points which are both good in term of experimental design (e.g. evenly distributed, non-collapsing...) and in term of function improvement.

1 Black-box optimization and surrogate-based methods

Black-box optimization is a branch of mathematical programming in which objective functions are expensive to evaluate. In black-box optimization, no analytical and derivative information about objective functions or constraints is available. Formally, it is defined by

$$\min_{x \in D} f(x)$$

in which $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a continuous expensive function and $D$ is a compact set in $\mathbb{R}^d$. The continuity and compactness conditions ensure the existence of a global minimum of the problem.

Recently, the most successful methods for solving black-box optimization are based on surrogate models. The idea is to iteratively construct models to approximate the black-box functions and then use them to search for optimal solutions. The methods begin with choosing and evaluating a set $S$ of initial points ($the design set$). A surrogate model is built to fit the data and based on that we select new point(s) to evaluate regarding certain criteria. We update the design set and repeat the procedure until the stopping criterion is met (or until we reach a certain number of function evaluations).

The purpose of the design phase is to locate in advance some points, with the hope that if we evaluate the black-box here, we will gain as much information about the function as possible. However, no information from these function evaluations is actually used at this step. Instead, only the structure of the domain $D$ is taken into account. We often require that the set $S$ has the following properties, in which we most important are:

- **Space**: The design points should be evenly spread over the entire design space.
- **Noh**: Two design points should not share any coordinate value when it is not known a priori which dimensions are important.
In order to find a good design set, we often have to solve complicated optimization problems. For example, if we need a design set with two above properties, we have to solve a maximin (or minimax) distance problem over the set of all Latin hypercube designs (LHD). The problem is very time-consuming to solve to optimality, especially in high dimension. However, even if we find a very good design set, it still does not ensure that we will get useful information from evaluations at these points. As long as function values from previous evaluations have been used, we might waste computational time by searching at regions less likely containing (local) minimizers.

But the idea is not to neglect the design phase and start to find optimal solutions (search phase) right after evaluating just two points. We still need function values at uniform set of points in order to obtain a global picture of the black-box function. In this way we can use surrogate models to approximate black-box functions more accurately, which is important in all surrogate-based methods.

In the following, we propose several methods to adaptively construct design sets by using information from previous evaluations. At the end, we obtain a good design set both in terms of experimental design and function improvement by inserting the general surrogate-based optimization framework.

Incorporatesearchphasetodesignphase

Assume the design phase requires us to construct a set of \( k \) points that are evenly distributed over the domain and (or) are non-collapsing. The methods we propose begin with evaluating a set of 2 points and iteratively insert points until we reach the number \( k \). The points being inserted are based on information from previous function evaluations. In particular, simple local or global approximation models (weak models) are used to fit the data and we utilize these models to search for the next point.

### 2.1 Search over a set of good designs

In order to find an optimal design, we need to solve an optimization problem: minimize a function \( \sigma \), which can be seen as a measure of non-uniformity of point sets. Optimal (or near-optimal) designs therefore are not unique. It is because there are many choices of measure \( \sigma \) (minimax, maximin, IMSE, MMSE,...) and solving the problem associated with each measure leads us to a different design. Even if we use the same measure \( \sigma \), there might be multiple optimal solutions. For example, by rotating a design set, we can obtain different sets with exactly the same property (see Figure 1).

**Figure 1:** The two equivalent optimal Latin hypercube designs

Since for a certain size, there are many optimal design sets, it is natural to select the ones which produce the minimal objective function values. In particular, assume we have a large set \( D_k \) of good designs of cardinality \( k \). If we have already chosen and evaluated \( x_1, x_2, \ldots, x_i \), then the next point to be chosen is the one that minimizes the weak surrogate model \( \sigma_l \) built from the data \((x_1, y_1), \ldots, (x_l, y_l)\) while
INCORPORATE SEARCH PHASE TO DESIGN PHASE

still belongs to some design set of $D_k$. It means that we select

$$x_{i+1} = \arg \min_{D \in S_i} \min_{x \in D \setminus \{x_1, \ldots, x_i\}} \sigma_1(x)$$

where $S_i$ is the set of all designs in $D_k$ that contain $x_1, x_2, \ldots, x_i$, i.e.

$$S_i = \{D \in D_k | x_1, \ldots, x_i \in D\}.$$

Obviously $S_2 \supseteq S_3 \supseteq \ldots \supseteq S_k$, so the search space for $x_1$ will be reduced when $i$ increases. Thus, if the family of design sets $D_k$ is not too big, it might happen that $S_i = S_{i+1} = \ldots = S_k = \{D^*\}$ for some $i$, i.e. all search spaces have only one element. In this case, we don’t have to continue doing any search: $D^*$ is the design set we need.

Since finding optimal designs is a difficult problem, especially in high dimension, we often do not have a family of design sets $D_k$ that is large enough. Thus, the case above occurs quite frequently, even when $i$ is small. Then there is not too many things we can do to improve the design set based on the function evaluation information. We propose other methods in subsequences, that deal particularly with Latin hypercube designs. In both cases, the methods work by perturbing a current design set, so that space-fillingness and non-collapsingness are preserved, the function improvement can be obtained. While in the first method, a Latin Hypercube design is maintained at each iteration, that is not the case in the second method.

2.2 Perturb a Latin hypercube design

Any Latin hypercube design $D = \{x_1, x_2, \ldots, x_n\}$ in $\mathbb{R}^d$ is represented by a LHD matrix

$$M = \begin{bmatrix}
    x_{11} & x_{12} & \ldots & x_{1d} \\
    x_{21} & x_{22} & \ldots & x_{2d} \\
    \vdots & \vdots & \ddots & \vdots \\
    x_{n1} & x_{n2} & \ldots & x_{nd}
\end{bmatrix},$$

where each column of $M$ is a permutation of $\{1, 2, \ldots, n\}$. Here $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T$ for $1 \leq i \leq n$ and $x_{ij} \in \{1, 2, \ldots, n\}$. Note that if we pick up any column and exchange elements in that column, we still obtain another Latin hypercube design.

Assume that we know $D$ is an optimal (or good) Latin hypercube design set and we have already evaluated some of them (e.g. $k = 2$). Based on the evaluation values of the black-box $f(.)$ at these $k$ points, we would like to evaluate at different locations other than those remaining in $D$. This means we have to sequentially update the set of points until we find a LHD.

Assume that $x_1, x_2, \ldots, x_k$ have already been evaluated. Remove all the rows corresponding to $x_1, x_2, \ldots, x_k$, we get a saturated matrix $M'$ with $n - k$ rows, which represent all the remaining points. Now we want to select a new point that differs from one of the points. Since we only change of component of the point we expect that uniformity of the design is still ensured and we obtain a good design again.

The above idea is presented in Algorithm 1.

2.3 Relax a Latin hypercube design

With LHDs, we can only evaluate a function in a lattice of finite points. This is because a LHD is found by dividing the domain into a large number of cells, applying optimal search to select $n$ cells and then evaluating at the centers of these cells. This practice is too strict since centers are not necessarily the most informative points in the cells, especially if some previous function evaluations are known.
3 COMPUTATIONAL RESULTS

Algorithm: Perturb a Latin hypercube design

Data: A black-box function \( f \), a Latin hypercube design \( D^* \), a subset \( S \subset D^* \) of evaluated points.

Result: A new Latin hypercube design \( D \) such that \( S \subset D \).

Initialization

Construct the LHD matrix \( M \) representing \( D^* \).

while \( |S| < n \) do

• Step 1: Evaluate \( f \) at all points in \( S \). Construct a surrogate model \( g_S \) that interpolates (approximates) the data \( \{(x, f(x)) \mid x \in S\} \).

• Step 2: Let \( M' \) be the saturated matrix of \( M \) by removing all the rows corresponding to \( x_1, \ldots, x_k \). Then \( x_{k+1} \) is selected by \( x_{k+1} = \arg \min \{ g_S(x) \mid x \in \Omega_k \} \), in which \( \Omega_k \) is the set of all possible rows of any matrices generated from \( M' \) by exchanging elements in one random column.

• Step 3: Assign \( S := S \cup \{x_{k+1}\} \).

end

return \( S \).

We propose an idea to improve the classical LHD approach. The general structure of a Latin hypercube design is preserved (i.e., we keep all the selected cells), but instead of using the centers for evaluation, we search over their neighborhoods to find more appropriate points. The search is based on the function evaluations from previously chosen points, in particular a surrogate model built upon them.

Now assume \( x_1, x_2, \ldots, x_n \) are the centers in an optimal design and \( \Delta_1, \Delta_2, \ldots, \Delta_n \) are the corresponding cells containing them. Assume hatwe already evaluated at \( x_1^*, x_2^*, \ldots, x_k^* \) for some \( x_i^* \in \Delta_i \), \( 1 \leq i \leq k \). For example, they can be chosen as centers \( x_i \) in the initial step, and updated in later iterations.

We associate each \( x \in \{x_{k+1}, x_{k+2}, \ldots, x_n\} \) with a ball \( B(x, r) \) of radius \( r > 0 \) such that \( B(x_i, r) \subseteq \Delta_i \).

We will talk about how to choose the radius \( r \) later. The balls \( B(x, r) \) can be defined by any of \( l_p \) norms, but we take the \( l_\infty \) norm instead of \( l_1 \) or \( l_2 \) for simplicity.

First we build a surrogate model \( g_k \) that fits the data \( (x_1^*, f(x_1^*)), \ldots, (x_k^*, f(x_k^*)) \). Assume \( y_{k+1} \) is the point that minimizes the surrogate function \( g_k(\cdot) \) over \( D \). We do not choose \( y_{k+1} \) as the next candidate point. Instead, we select the projection of \( y_{k+1} \) onto the union of all remaining balls. In other words,

\[
x_{k+1}^* = \arg \min_{z \in B_x} \|z - y_{k+1}\|
\]

where \( B_k = \bigcup_{i=k+1}^n B(x_i, r) \).

The above idea is presented in the Algorithm 2.

3 Computational results

Ideas:

• Test the two algorithms on standard test problems to see how much improvement they can make.

• Optimal design library from [Husslage11].
Algorithm 2: Relax a Latin hypercube design

Data: A black-box function $f$ over domain $D$, a Latin hypercube design $D^*$, a set $S$ of evaluated points.
Result: A new design $D$ such that $S \subset D \subset D$.

while $|S| < n$ do
  • **Step 1**: Evaluate $f$ at all points in $S$. Construct a surrogate model $g_S$ that interpolates (approximates)
    
    $$\{(x, f(x)) \mid x \in S\}.$$ 

  • **Step 2**: Find the minimum of the surrogate model by solving
    
    $$y_{k+1} = \min_{x \in D} g_S(x).$$ 

  • **Step 3**: Choose the radius $r > 0$. Let $S^*$ be the set of all centers associated with points in $S$. Select $x^*_{k+1}$ by
    
    $$x^*_{k+1} = \min_{z \in B_{k}} \|z - y_{k+1}\|,$$
    
    where $B_k = \bigcup_{x_i \in D \cap S^*} B(x_i, r)$.

  • **Step 3**: Assign $S := S \cup \{x^*_{k+1}\}$.

end

return $S$. 