Locating Infinite Discontinuities in Computer Experiments

Article *in* SIAM/ASA Journal on Uncertainty Quantification · January 2020

DOI: 10.1137/18M1209076

CITATIONS
0

READS
26

3 authors, including:

Ying-Chao Hung
National Chengchi University
36 PUBLICATIONS **182** CITATIONS
SEE PROFILE

George Michailidis
University of Michigan
366 PUBLICATIONS **6,417** CITATIONS
SEE PROFILE

Locating Infinite Discontinuities in Computer Experiments*

Ying-Chao Hung†, George Michailidis‡, and Horace PakHai Lok§

Abstract. Identification of input configurations so as to meet a pre-specified output target under a limited experimental budget has been an important task for computer experiments. Such a task often involves the development of response models and design of experimental trials that rely on the models exhibiting continuity and differentiability properties. Motivated by two canonical examples in systems and manufacturing engineering, we propose a strategy for locating the boundary of the response surface in computer experiments, wherein on one side the response is finite, whereas on the other side is infinite, leveraging ideas from active learning and quasi-Monte Carlo methods. The strategy is illustrated on an example from computer networks engineering and one from precision manufacturing and shown to allocate experimental trials in a fairly effective manner. We conclude by discussing extensions of the proposed strategy to characterize other types of output discontinuity or non-differentiability in high-cost experiments, including jump discontinuities in the target output response or pathological structures such as kinks and cusps.

Key words. Computer experiments, Infinite discontinuity, Active learning, Support vector machines, Quasi-Monte Carlo methods

AMS subject classifications. 60G15, 62M20, 62K99, 65Y20, 91B74

1. Introduction.

Computer experiments involve complex computer codes underlying some physical phenomenon or experiment. They act as surrogates to explore the relationship between the corresponding input factors and an output measure of interest. However, running such computer codes is usually computationally expensive in terms of CPU time [47]. A natural question arising is how to obtain a comprehensive understanding of output performance over the input domain by utilizing a limited number of experimental trials, which leads to how to best model the input-output relationship and select the corresponding computer code trials.

The Gaussian Stochastic Process (GASP) model has been the most popular technique for modeling the input-output relationship of computer experiments [40, 45, 46, 47], briefly summarized next. Denote the code output corresponding to K-dimensional inputs (locations) $x_1, \ldots, x_n$ by $y(x_1), \ldots, y(x_n)$. The stationary GASP model models such pairs of inputs-outputs as $y(x_i) = \mu + z(x_i)$, where μ is the output mean and $z(x_i)$ is a spatial process with mean zero, constant variance σ^2, and covariance $\mathrm{Cov}(z(x_i), z(x_j)) = \sigma^2 R_{ij}$, where R_{ij} is a correlation function defined by a measure of spatial distance between inputs x_i and x_j. Further, it is assumed that $y(X)$ follows a multivariate normal distribution with mean vector $\mathbf{1_n}\mu$ and covariance matrix $\sigma^2[R_{ij}]$. The benefit of adopting the GASP model is that

*Submitted to the editors DATE.

Funding: This work was supported in part by research grant MOST 106-2118-M-004-002-MY2 (YCH) and NSF IIS 1632730 and NSF DMS 1830175 (GM).

†Department of Statistics, National Chengchi University, Taipei 11605, Taiwan (hungy@nccu.edu.tw).

‡Department of Statistics and the Informatics Institute, University of Florida, Gainesville, FL 32611, USA (gmichail@ufl.edu).

§Department of Statistics, National Chengchi University, Taipei 11605, Taiwan (105354008@nccu.edu.tw).

the explicit form of the best linear unbiased predictor (BLUP) at a new input location can be obtained in closed form [46, 47], thus allowing researchers to explore the experiment's output performance over the entire input space. There is a considerable body of work on related models, including limit kriging [27], co-kriging [16], blind kriging [28], Bayesian treed Gaussian process models [19] and scaled Gaussian stochastic process [20], just to name a few. Based on the above-mentioned characteristics, there exists a number of design methods that aim to select the experimental trials so as to minimize uncertainty in estimating the GASP model estimation, as well as output predictions. Examples include a sequential design based on a designated improvement function for contour estimation [44], a design over non-convex regions based on multidimensional scaling to the geodesic distance [43], and a sequential design strategy based on maximum mutual information [2], etc. A large portion of such design strategies belongs to the framework of space filling designs [26, 47].

Note that GASP based modeling and the associated design techniques developed for computer experiments automatically assume that the output response surface is *smooth* (*continuous* and *differentiable*) [2]. This assumption provides a convenient mathematical formulation and corresponding solution, but may not be suitable for certain settings as outlined next. For example, a fundamental issue in many service engineering systems is to identify possible input configurations so that some performance measure of interest does not increase in an unbounded manner to *infinity* [4, 8, 15, 34, 50]. This is related to the problem of system *stability* and often needs to be validated through a long-run simulation. Another example is the laser cutting epoxy film in manufacturing engineering, for which the goal is to detect the physical cutting limits by characterizing a number control factors. Since the cutting limits refer to the boundary of input configurations that produce "nonzero" outputs (or the boundary of the no-cut region), by taking the reciprocal of the output value, the goal here is exactly the same as identifying input configurations that produce finite outputs. Due to the complexity of the laser cutting process, a computer surrogate model is often utilized to understand how the control factors affect the outputs. Note that experimental trials needed to identify the input configurations for such problems are often computationally expensive. A detailed description of these two examples is provided in Section 2.

The broad objective of this paper is to develop methodology to identify input configurations in computer experiments that trace the boundary that separates infinite valued outputs from finite valued ones. Technically, it aims to locate the boundary in the input domain (henceforth referred to as the set of *infinite discontinuities*) in an efficient and accurate manner. As pointed out in the literature [19], the popular GASP model cannot predict well non-continuous changes in the output (e.g. jumps). To overcome this limitation, we propose to formulate the problem of locating infinite outputs as a classification one. In this case, input configurations that lead to finite output values are labeled as belonging to class "+1", while those that lead to infinite output values are labeled as belonging to another class "−1". Hence, the problem becomes to identify the decision boundary separating the two classes. Based on this, the proposed strategy leverages ideas from *active learning*, a semi-supervised machine learning technique (see [33] and references therein), whose primary goal is to train a good classifier for predicting the output labels by using a small amount of sampled data. Note that rather than randomly obtaining a full training data set, active learning iteratively selects the most "informative" instances and query their labels to train the classifier. Further, the use

of active learning for the problem at hand belongs to the *pool-based sampling* [36, 53] and is related to sequential experimental designs. Further, active learning in our strategy is based on the powerful formalism of support vector machines (SVM) [3, 10, 42, 48, 54] for label predictions. However, the sampling technique is different from those employed by conventional methods in active learning. In order to obtain a good initial training set (i.e., the passive learning process) and accommodate well the training samples selected at later learning stages (i.e., the active learning process), our sampling strategy utilizes a robust quasi-Monte Carlo (QMC) method - called uniform design (UD) in the statistical literature [12, 13].

Note that conceptually the problem addressed in this paper shares similarities to the so-called regression discontinuity design (RDD) that has many applications in economics and engineering [25, 32, 35, 51]. The remainder of the paper is organized as follows. In Section 2, a motivating example that addresses the stability issue of queueing service engineering systems is described in detail. In Section 3, the strategy based on the concept of active learning is introduced. The focus is on (i) how to choose an adequate initial training set and unlabeled instances at later learning stages from a pool of candidate instances; and (ii) how to develop sophisticated SVM based active learners for label prediction. Note that there are two iteratively updated weight functions associated with data involved in this active learning process - one is designed to improve the accuracy of the SVM learner, while the other to represent the sample informativeness/importance based on which the training data can be best selected by UD. In Section 4, the proposed strategy is illustrated on a queueing service engineering system and numerical evaluations are provided. A discussion and some concluding remarks are drawn in Section 5. Finally, an accelerated algorithm for implementing UD in higher dimensional spaces and the Hierarchical Mixing Linear SVMs (HMLSVM, see [55]) for locating infinite discontinuities with piecewise linear shapes are given in Appendix A and Appendix C, respectively.

2. Motivating Examples.

Next, we describe in considerable detail two motivating examples from systems engineering and precision manufacturing.

2.1. Example 1: Stability of Queueing Systems.

Consider a multi-class queueing system comprised of Q parallel, infinite capacity, and first-in-first-out (FIFO) queues, with each queue containing traffic from a different job class. Each job of class q arrives according to a random process, carries a random amount of workload (or service requirement), and waits in line for service. For generosity here we assume each arrival process has a *time-heterogeneous rate* (i.e., the instantaneous input rate can vary over time). Thus, it is reasonable to define a time-averaged arrival rate for each input class q, i.e.,

$$x_q = \lim_{T\to\infty} \frac{E[A_q(T)]}{T},$$

where $A_q(T)$ is the amount of workload arriving at queue q before time T, $q = 1, \ldots, Q$. These time-averaged input rates can be collected in a vector $x = (x_1, \ldots, x_Q)$, where each element x_q is assumed to be non-degenerate. At any point in time, the system can be in one of M ($M \geq 2$) service modes indexed by $m \in \{1, \ldots, M\}$. When the system switches into the m-th service mode, the jobs in queue q receive service at a constant rate μ_{mq} (i.e., the amount of work that can be processed in one time unit). Therefore, mode m is asso-

ciated with the service rate vector $U_m = (\mu_{m1}, \ldots, \mu_{mQ})$. Note that the switching scheme between service modes automatically introduces intricate dependencies amongst the queues. For a general real-life system, we assume the switching time between any two service modes i and j (denoted by Δ_{ij}) is non-negligible, while no service is provided during each switching epoch. This particular queueing system is known as the Switched Processing System (SPS) and captures the essence of the fundamental resource allocation problem in many modern service engineering applications involving heterogeneous processors and multiple classes of job traffic flows; examples include network switches for routing traffic through the Internet, modern flexible manufacturing processes, and automatic call distributor (ACD) (see [1, 23] and references therein).

A fundamental issue for SPS studies is to characterize the allowable range of input rates (i.e. x) so that system can achieve a certain level of "stability" under a given service-mode switching policy. For example, denote by $W_q(t)$ the workload (or remaining service requirement) of queue q at time t, which is obviously a function of the input rate vector x. Since each arrival process is time-heterogeneous, system stability can be defined based on the time-averaged expected amount of workload [57]. That is, given an input-rate vector $x = (x_1, \ldots, x_Q)$, the system is *stable* if

$$\theta_q(x) = \lim_{T\to\infty} \int_0^T \frac{E[W_q(t)]}{T} dt < \infty \quad \text{for all } q = 1, \ldots, Q. \tag{2.1}$$

On the other hand, the system is characterized as "unstable" if $\theta_q(x)$ is infinite for at least one queue q. Given a service-mode switching policy π, the maximum set of input-rate vectors x satisfying (2.1) is called the *stability region*.

Note that based on the definition in (2.1), the outer *boundary* of the stability region clearly represents the set of infinite discontinuities. However, identifying this boundary is mathematically intractable for even very small systems and hence in practice is approximated through computer simulations of the underlying system. To illustrate, let us consider a simple 2-queue system with three service modes $U_1 = (0, 5)$, $U_2 = (6, 4)$, $U_3 = (7, 0)$ and assume that the switching time between any two service modes is constant, say, $\Delta_{ij} \equiv \Delta$ for all i, j. Jobs of each input arrive according to a compound Poisson process with exponentially distributed service requirements, having a rate of one. Suppose a well-known service-mode switching policy called the MaxProduct is employed [1, 23, 24], the estimated stability regions are shown in Figure 1 for two selected switching times $\Delta = 0.5$ and 1.0, where stability is validated based on (2.1) by simulating the system at a set of superimposed input-rate grids over an $\mathbb{R}^2$ region $[7, 0] \times [0, 5]$. Note that for comparison purpose, the maximum stability region (i.e. when $\Delta = 0$) is also included in Figure 1.

Note that for a system with the stability region shown in Figure 1, we can simply define $\theta(x) = \theta_1(x)$, $\theta_2(x)$, or $\theta_1(x) + \theta_2(x)$. Thus, by (2.1) $\theta(x)$ is characterized as "finite" when the input-rate vector $x = (x_1, x_2)$ is inside the stability region. Specifically, $\theta(x)$ becomes substantially larger when x gets fairly close to the boundary of the stability region and jumps to "infinity" when x reaches or crosses the boundary. The later case naturally leads to an output model characterized by jump discontinuities. The main challenge of identifying the stability region based on simulation is the associated computational cost, since for each input configuration the system needs to be simulated for a large number of events (job arrivals,

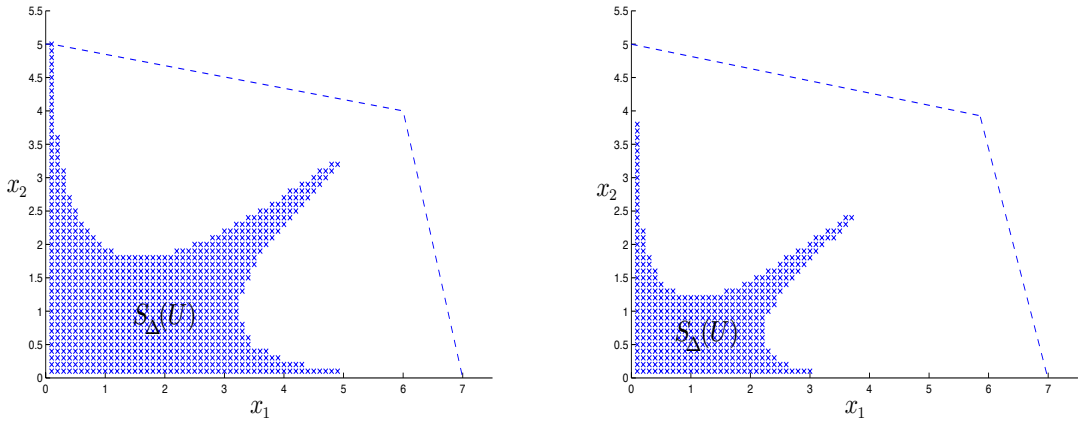

Figure 1. *The estimated stability regions $S_\Delta(U)$ with service-mode switching times $\Delta = 0.5$ (the dotted area in the left panel) and $\Delta = 1.0$ (the dotted area in the right panel) under the MaxProduct policy. The polygon with the boundaries described by the dashed lines refers to the maximum stability region with $\Delta = 0$.*

switching between service modes epochs, job service and departures), so that the long term average defined in equation (2.1) can be estimated accurately. This computational time grows fast for larger systems compromising of more queues and more switching modes, or for input configurations close to the boundary of the stability region.

2.2. Example 2: Limits of Laser Cutting Process. Laser cutting epoxy film has replaced the conventional die cutting film and is being widely used for circuit board attachment to carriers and housings in defense electronics, commercial radio frequency (RF) microwave and microelectronics industries. An important issue in such manufacturing process is to detect the physical cutting limits by characterizing the control factors such as laser power, beam radius, pulse duration, focal position, and Rayleigh length. Denote the cutting width at the bottom of the material by $y(x_1, x_2)$, where x_1 represents the pulse duration (e.g. in μs), x_2 represents the laser power (e.g. in watts), while other factors are kept constant. The goal is to identify the control factor settings (x_1, x_2) so that cut through does not occur (i.e. the cutting width $y(x_1, x_2) = 0$). However, identifying the limits of the cutting process is a challenging task since (i) the relationship between the cutting width and other control factors may not be trivial; and (ii) each experimental trial is physically expensive [32]. Thus, simulation of such complex processes along with an adequate surrogate model is often used to understand how the control factors affect the outputs.

Suppose that the experiment is emulated, say, with a proper transformation of scales, by the following functional relationship (see [32] for a similar function):

$$y(x_1,x_2) = \begin{cases} \frac{1}{100}\left\{\left[(x_1-7)^2+(x_2-7)^2-62\right]^2+\left[(x_1-7)^2-0.5(x_1-7)-0.5(x_2-7)-1.5\right]^2\right\}+10 & \text{if } x_1^2+x_2^2>80, \\ 0 & \text{if } x_1^2+x_2^2\le 80. \end{cases} \tag{2.2}$$

The interpolated surface and the contour plot of function (2.2) over the 2-dimensional square domain $[0, 14] \times [0, 14]$ are depicted in Figure 2. As can be seen, the boundary of the blue-color

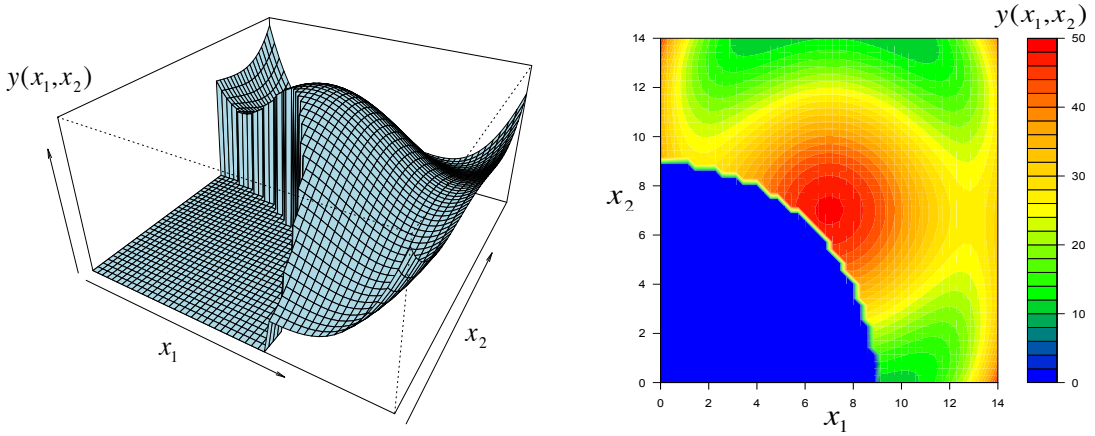

Figure 2. *The interpolated surface (left panel) and the contour plot (right panel) of function (2.2) based on* $50 \times 50 = 2,500$ *grids superimposed over the input domain* $[0, 14] \times [0, 14]$. *The blue-color area in the contour plot represents the settings of "no cut" (i.e.* $y(x_1, x_2) = 0$*) in the input domain.*

area in the right panel of Figure 2 represents the set of discontinuities in the input domain, which is the focus of this work. Also note that identifying the limits (i.e. discontinuities) of the cutting process can be thought as a "dual" problem to that given in Example 1, since one can consider the reciprocal of $y(x_1, x_2)$ as the response in Example 1. In words, by letting $\theta(x) = 1/y(x_1, x_2)$, the input points that produce the "zero" output are equivalent to the infinite discontinuities described in Example 1.

3. Locating Infinite Valued Discontinuities via Active Learning.

Since the GASP model is not suitable for finding the infinite valued discontinuities, we propose next an alternative strategy based on the concept of active learning. The strategy is also sequential and formulates the primary problem as a classification one. Further, it includes two main components: (i) a quasi-Monte Carlo method called uniform design (UD) for sampling the training data (or allocating experimental trials); and (ii) an SVM classifier for estimating the discontinuities. Note that there are two well-designed weight functions associated with these two components. The first one represents the importance of data to be sampled, while the second one represents the importance of sampled data for training the SVM classifier. We start with introducing the sampling technique based on UD, while active learning with SVM is discussed afterwards.

3.1. Sampling Based on Quasi-Monte Carlo Methods.

The quasi-Monte Carlo (QMC) method is a deterministic version of the Monte Carlo method. It yields a faster convergence rate for numerical integration and has proved to be a robust technique for exploring relationships between input factors and experiments outputs. The key idea of QMC is to choose a set of n points $\mathcal{P} = \{p_1, \ldots, p_n\}$ that are uniformly distributed on a bounded input domain D, $D \subset \mathbb{R}^K$. Let $Z(n)$ be the collection of all possible sets $\{p_1, \ldots, p_n\}$ on D. A uniform design (UD) is an efficient QMC that seeks the best set $\mathcal{P} \in Z(n)$ such that a measure of uniformity

$M(\mathcal{P})$ is minimized. That is,

$$\mathcal{P}^* = \arg \min_{\mathcal{P} \in Z(n)} M(\mathcal{P}). \tag{3.1}$$

The most popular choice for the measure of uniformity is the so-called "discrepancy" [13, 14, 21]. In this study, we employ a measure named the *central composite discrepancy* (CCD), which is developed to overcome the limitations of conventional space filling designs [9, 38]. The CCD is attractive from the following perspectives: (i) the solution is robust to the output model and has a faster convergence rate than traditional random sampling procedures (i.e. the property of quasi-Monte Carlo methods); (ii) it can be applied to any shape of design domains; (iii) the solution is invariant to rotations; (iv) the optimal design $\mathcal{P}^*$ can be obtained to explore a designated output response by placing different weights at the input configurations. Note that (iv) is the key element of our proposed framework, since all allocated experimental points may not be equally important in locating the desired discontinuities. The formulation of the CCD with weights is given next.

At a particular learning stage s, let $f_s(x)$ be the weight function such that $f_s(x) > 0$ for all $x \in D$ and assume $\int_D f_s(x)dx = 1$ (this can be done by a simple normalization). With this formulation, the weight function $f_s(x)$ can also be viewed as the probability density function of all data points defined on D. To find a set of n points $\mathcal{P} = \{p_1, \ldots, p_n\}$ in D so that they have a "good representation" for $f_s(x)$, we define

$$WCCD_{f_s,p}(n, \mathcal{P}) = \left\{ \frac{1}{v(D)} \int_D \frac{1}{2^K} \sum_{k=1}^{2^K} \left| \frac{N(D_k(x), \mathcal{P})}{n} - F(D_k(x)) \right|^p dx \right\}^{1/p}, \tag{3.2}$$

where $v(D)$ is the volume of D, $N(D_k(x), \mathcal{P})$ is the number of points allocated in the subregion $D_k(x)$ (note that D is partitioned into 2^K subregions at each x), $F(D_k(x)) = \int_{D_k(x)} f_s(x)dx$ is the proportion of points expected to be allocated in $D_k(x)$, $k = 1, \ldots, 2^K$. Therefore, a good representation for $f_s(x)$ will be the set of points $\mathcal{P}^*$ that minimizes (3.2). Note that solving $\mathcal{P}^*$ is an NP-hard problem as the number of allocated design points goes to infinity. In practice, one can superimpose a set of grid points Z over the input domain D and approximate the solution by using sophisticated search algorithms [6, 9, 38]. The solution of such a discretized optimization problem is known as the nearly uniform design (NUD), which minimizes the approximation of $WCCD_{f_s,p}(n, \mathcal{P})$,

$$\widehat{WCCD}_{f_s,p}(n, \mathcal{P}) = \left\{ \frac{1}{|Z|} \sum_{z \in Z} \frac{1}{2^K} \sum_{k=1}^{2^K} \left| \frac{N(D_k(z), \mathcal{P})}{n} - F(D_k(z)) \right|^p \right\}^{1/p}, \tag{3.3}$$

where $\mathcal{P} \in Z(n) \subset Z$, and $|Z|$ represents the cardinality of Z. It should be noted that the solution $\mathcal{P}^*$ here is deterministic and is different from the solution obtained by stochastic sampling methods (e.g. random sampling).

3.2. Acceleration of the Sampling Procedure. Next, we discuss how to accelerate the associated sampling procedure. Suppose a UD of size n needs to be found based on N

grid points (i.e. let $|Z| = N$) superimposed over the input domain D. Under the natural assumption that $K < n \ll N$, an exhaustive search requires computational time of order $O(N^n)$ for finding $\mathcal{P}^*$. To reduce the computation load, Chuang and Hung (2010) [9] proposed a Switching Algorithm and showed that an NUD can be constructed within the computational time of order $O(N^{(2+p)})$. Although the algorithm provides a good approximation to $\mathcal{P}^*$ in lower dimensions, it may still be computationally intensive for many problems with multi-dimensional inputs. To see this, let us look at a rectangular domain with $K = 3, N = 10^3, n = 6$ and $p = 2$ (wherein an ℓ_2-distance is considered). Note that even for such a simple example, exhaustive search requires computation of order 10^{18}, while the Switching Algorithm requires computation up to order of 10^{12}, which represents a big improvement in absolute terms, but not still sufficient for most applications.

In order to implement UD with a more manageable time line, we introduce next an accelerated algorithm for finding an NUD. Let ψ_i be a non-degenerate subset of $\{1, \ldots, K\}$ and denote by Z^{ψ_i} the projection of Z onto the corresponding axes in a lower $|\psi_i|$-dimensional space, $|Z^{\psi_i}| \geq n$. The idea of the proposed accelerated algorithm is to first locate the design points based on the lower $|\psi_i|$-dimensional uniformity. Then, by augmenting the selected locations in the lower $|\psi_i|$-dimensional space, we sequentially increase the dimensionality of Z^{ψ_i} so that the design points are relocated step by step to achieve better uniformity in the primary input space. It should be highlighted that the acceleration algorithm is developed to search a flexible and bigger area than that of the well-known Latin Hypercube Designs (LHD) [11], while retaining the property of uniformity and a fairly fast computation speed. A detailed description of the acceleration algorithm and discussion of its performance are given in Appendix A.

We next introduce how to develop an adequate response model $g(x)$ at each design stage, so as to improve the estimation of the discontinuities based on the concept of active learning.

3.3. Active Learning with SVM. Let us recall the example of the queueing system introduced in Section 2 and unify some notations. Denote the input rate vector by x, the labeled response, say, $\theta(x) = \theta_1(x)$ by $y(x)$, and the stability region $S_\Delta(U)$ by $T(x)$, respectively. To overcome the difficulty of modeling infinite outcomes, we formulate the problem as a binary classification one. That is, label $y(x) = +1$ represents the case that the input point x is inside the target region (or equivalently $\theta(x) < \infty$), while label $y(x) = -1$ represents the case that x is outside the target region (or $\theta(x) = \infty$). Therefore, the objective of locating/estimating the boundary of $T(x)$ becomes that of identifying the decision boundary between design points with labels $y(x) = +1$ and -1 in the primary data space, wherein the allocated experimental trials are treated as training samples. Since labeling data is time consuming or costly, instead of obtaining the training samples at random, active learning selects "informative" samples so as to maximize the classification accuracy with less training data. We focus on active learning based on an SVM classifier since: (i) SVM is a proven powerful classification tool capable of dealing with different shapes of decision boundaries; and (ii) the distance of a sample x to the decision boundary of $T(x)$ can be easily computed (see below) and subsequently used as a measure of information about input x.

Denote the training data set (i.e. the existing experimental trials) at a particular learning stage by $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i = (x_{i1}, \ldots, x_{iK})$ is a K-dimensional input vector

and $y_i = y(x_i) \in \{-1, +1\}$ is the corresponding output label. In order to obtain flexible decision boundaries (as shown in Figure 1), the SVM algorithm projects the training data into a higher dimensional feature space $\mathcal{H}$ using a mapping function $\Phi(\cdot)$. With this mapping, it is shown that the optimal solution depends on data merely through the inner products in $\mathcal{H}$ [54], that is, on functions of the form $\langle \Phi(x_i), \Phi(x_j) \rangle$. Hence, a computationally less expensive approach is to use a kernel function $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ instead of explicit mappings $\Phi(x_i)$ and $\Phi(x_j)$. This is known as the *kernel trick* - a commonly used technique in machine learning [49].

Since different data points may have different contributions to the training of a classifier, we use the weighted version of SVM (termed as WSVM) for active learning. Let W_i represent the weight assigned to each training sample x_i, $i = 1, \ldots, n$. To obtain the WSVM based on all weighted training data (x_i, y_i, W_i) with a mapping function $\Phi(\cdot)$, one solves the constrained convex optimization problem [59]:

$$\begin{aligned} &\underset{w,b}{\text{minimize}} \ \ \frac{1}{2} w'w + C \sum_{i=1}^{n} W_i \xi_i \\ &\text{subject to} \ \begin{cases} y_i \left(\langle w, \Phi(x_i) \rangle + b \right) \geq 1 - \xi_i, & i = 1, \ldots, n, \\ \xi_i \geq 0, & i = 1, \ldots, n. \end{cases} \end{aligned} \tag{3.4}$$

where $\langle w, \Phi(x) \rangle + b$ represents the hyperplane separating the y_i's in the feature space $\mathcal{H}$, w is the coefficient vector, b controls the offset of the decision boundary from the origin, ξ_i are known as *slack variables* that penalize wrong classifications of y_i, and $C > 0$ controls the tradeoff between classification accuracy and the margin between two bounding planes. The standard approach for solving (3.4) is to formulate its dual given by [59]:

$$\begin{aligned} &\underset{\alpha_i}{\text{maximize}} \ \ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ &\text{subject to} \ \begin{cases} 0 \leq \alpha_i \leq C W_i, & i = 1, \ldots, n, \\ \sum_{i=1}^{n} \alpha_i y_i = 0, \end{cases} \end{aligned} \tag{3.5}$$

where α_i are the associated Lagrange multipliers. Solving (3.5) yields the decision function

$$g(x) = \text{sign} \left(\langle w, \Phi(x) \rangle + b \right) = \text{sign} \left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \right), \tag{3.6}$$

where $w = \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i)$ (readers can refer to LIBSVM [5] for how to obtain the WSVM learner in practice). Note that a popular choice of the kernel function is the radial basis function (RBF), which has the form $K(x_i, x_j) = \exp\left\{ -\gamma \|x_i - x_j\|^2 \right\}, \quad \gamma > 0$. The RBF kernel offers the following two advantages in practice: (i) it can produce nonlinear decision boundaries by mapping data into high (or infinite) dimensional space; and (ii) it has relatively low complexity for model selection. Therefore, it is particularly suitable for exploring high-dimensional data structures. In practice, the two parameters (C, γ) can be chosen by performing an appropriate grid search in R_+^2 and utilizing the idea of cross-validation (CV) so

that the "prediction error rate" is minimized [22, 29, 31]. However, there are some alternative methods that have been shown to achieve better efficiency and/or accuracy than traditional grid search based on CV. For example, the optimal value of γ can be found by minimizing the Fisher discriminant function [56], by considering parameter selection as a recognition problem [58], and by optimizing simultaneously the overall accuracies (for within-class samples) and kappa accuracies (for between-class samples) [37], just to name a few.

Let us denote the decision function obtained at learning stage s by $g_s(x)$, $x \in D$. Analogous to the idea of boosting [17, 18], the weight for each sampled point x_i is chosen as

$$W_i = 1 + M_i^s = 1 + \sum_{t=1}^{s-1} I\{y_i \cdot g_t(x_i) < 0\}, \tag{3.7}$$

where M_i^s represents the total number of times that x_i was wrongfully labeled (i.e. the decision function yields a different label from y_i, such as $y_i = +1$ and $g_t(x_i) < 0$, or $y_i = -1$ and $g_t(x_i) > 0$) in the past $s-1$ stages, $M_i^1 = 0$ for all i. Note that if M_i^s is large, the weight W_i at stage s is set to be large, while W_i is set to be small if M_i^s is small. Intuitively, this will force the decision function $g_s(x)$ to accommodate all the training samples based on their classification history, thus improving the overall classification accuracy. Once $g_s(x)$ is obtained at a particular learning stage s, the boundary of $T(x)$ can be simply estimated by

$$\partial \tilde{T}_s(x) = \{x \in D : g_s(x) = 0\}. \tag{3.8}$$

Note that the goal of obtaining $\partial \tilde{T}_s(x)$ is to quantify the "informativeness" (or importance) of all unlabeled samples, based on which a suitable weight function can be incorporated into UD for choosing the next-stage instance from the pool.

Next, we proceed to introduce how the weight function $f_s(x)$ in (3.2) is defined in accordance with the quantified measure of information. For each instance x, compute its minimum Euclidean distance to the estimated boundary of the target region, that is,

$$d(x, \partial \tilde{T}_s(x)) = \min_{\delta \in \partial \tilde{T}_s(x)} \|x - \delta\|. \tag{3.9}$$

If x is unlabeled and has a smaller $d(x, \partial \tilde{T}_s(x))$, we say that it is more informative for training the SVM classifier and thus more likely to be selected at the next stage. This idea is quite similar to that of the Simple Margin querying method [53], which chooses the unlabeled sample closest to the decision boundary in $\mathcal{H}$. Based on this argument, the weight function $f_s(x)$ for implementing UD is given by

$$f_s(x) = \exp\left\{-\frac{d(x, \partial \tilde{T}_s(x))}{\beta(s)}\right\}. \tag{3.10}$$

As can be seen in (3.10), the weight of an instance x becomes exponentially larger when it gets closer to the estimated decision boundary and becomes exponentially smaller when it gets farther away from the boundary. This will push the next-stage learning to query an instance near the estimated decision boundary $\partial \tilde{T}_s(x)$. Further, $\beta(s)$ is chosen as a *decreasing*

function of learning stage s, which is similar to the concept of *learning rate* [41]. The goal of this setup is to accelerate the process of querying unlabeled instances near the "true" decision boundary, thus improving the estimation efficiency at subsequent stages. To normalize the weight function, we simply set

$$f_s^*(x) = \frac{f_s(x)}{\int_{x\in D} f_s(x)dx}. \tag{3.11}$$

Remark:
(a) For nonlinear decision boundaries the computation of $d(x, \partial\tilde{T}_s(x))$ may not be trivial, especially when the data dimension is high. A computationally cheaper way is to represent the input domain D as a pool of superimposed grid points Z (as done in Subsection 3.1) so that $\partial\tilde{T}_s(x)$ can be approximated by a subset of Z, say,

$$\partial\tilde{T}_s(z) = \{z \in Z : -\epsilon < g_s(z) < \epsilon\}, \tag{3.12}$$

where $\epsilon > 0$ is known as the *tolerance level* and chosen by the designer. Therefore, $d(x, \partial\tilde{T}_s(x))$ can be approximated by

$$d(x, \partial\tilde{T}_s(z)) = \min_{z\in\partial\tilde{T}_s(z)} \|x - z\|. \tag{3.13}$$

To evaluate the quality of the approximation based on (3.12), let us denote $|Z| = N$ by the total number of equal-size grids z superimposed over a typical K-dimensional input domain $D = [0, d]^K$. The value of N reflects the total computational cost one can afford so as to implement the quasi-Monte Carlo method. By definition, we then have

$$N = (d/l)^K, \tag{3.14}$$

and thus $l = dN^{-(1/K)}$, where l is the side-length of each K-dimensional grid $z \in Z$. Note that if we use (3.12) to approximate the decision boundary obtained in (3.8), the maximum error corresponds to the "diameter" of each superimposed grid z, which is given by

$$\sqrt{K}l = \sqrt{K}dN^{-(1/K)}. \tag{3.15}$$

It is easy to see that the approximation is influenced by the number of input dimensions (K), domain size (d) and maximum affordable computational cost (N).
(b) Instead of using $d(x, \partial\tilde{T}_s(x))$ to formulate the weight function $f_s(x)$ in (3.10), for computational simplicity, one may consider the following two alternatives: (i) consider the distance between x and the obtained decision boundary $\langle w, \Phi(x)\rangle + b = 0$ in $\mathcal{H}$, which has a simple mathematical expression $|\langle w, \Phi(x)\rangle + b| / ||w||$. Then, assign the weight for each x as

$$f_s(x) = \exp\left\{-\frac{|\langle w, \Phi(x)\rangle + b| / ||w||}{\beta(s)}\right\}; \tag{3.16}$$

(ii) consider the predictive value $g_s(x)$ and assign the weight for each x as

$$f_s(x) = \exp\left\{-\frac{\|g_s(x)\|}{\beta(s)}\right\}. \tag{3.17}$$

Note that assigning the weight based on the decision value is straightforward since $g_s(x) = 0$ on the decision boundary. Further, the decision values are readily available from software packages (e.g. the R package e1071). Thus, for computational simplicity purposes, we strongly recommend assigning the weight for each x by using (3.17).

We summarize the detailed steps of our strategy for estimating the target region with nonlinear boundaries in Algorithm 3.1. Note that we denote by $\mathcal{P}_s^*$ the training set obtained at learning stage s, where $|\mathcal{P}_s^*| = n_s$, $s = 1, 2, \ldots$.

Algorithm 3.1 Active Learning with SVM

1: Determine an appropriate training domain D associated with a set of superimposed grid points Z. Set the learning stage to $s = 1$ and determine the size of the initial training set n_1. Find the initial training set $\mathcal{P}_1^*$ of size n_1 that minimizes (3.3) based on Z, where $p = 2$ and the weight function is initially placed as $f_0(x) \equiv 1$. Train a kernel WSVM classifier based on $\mathcal{P}_1^*$ with the weights $W_i \equiv 1$ and obtain the decision boundary $\partial\tilde{T}_1(z)$ by (3.12).
2: **while** the stopping criterion is not met **do**
3: Update the weight functions $f_s(x)$ and $f_s^*(x)$ by (3.17) and (3.11), respectively. Set $s = s + 1$.
4: Select an unlabeled instance z^* from the pool $Z \setminus \mathcal{P}_{s-1}^*$ so that $\mathcal{P}_s^* = \mathcal{P}_{s-1}^* \cup \{z^*\}$ minimizes (3.3). Query the label of z^*.
5: Update the weights W_i for all instances in $\mathcal{P}_s^*$ by (3.7) and train a kernel WSVM classifier based on $\mathcal{P}_s^*$. Obtain a new decision boundary $\partial\tilde{T}_s(z)$.
6: **end while**
7: **return** $\partial\tilde{T}_s(z)$

We highlight a few important issues regarding the implementation of Algorithm 3.1.

- A suitable input domain D should be considered, so as to accurately and efficiently estimate the desired decision boundary. This often relies on the practitioner's knowledge about the response model. If possible, one can choose D in the form of hyper-rectangles, so as to accelerate the implementation of Algorithm 3.1.
- The number of grid points superimposed on D depends on the computational complexity of the proposed framework and the executing processor's performance. For example, if the input domain is high dimensional and/or the executing processor has moderate efficiency, a smaller number of grid points in Z should be considered.
- In cases where there is lack of knowledge regarding the output label, it is recommended to allocate a larger portion of the experimental budget to the initial training set (i.e., for the passive learning process) so as to better accommodate the model/sample uncertainty and improve the estimation efficiency. A rule of thumb for this proportion is about $20 \sim 50\%$, which was suggested by a large number of experimental scenarios.
- The choice of $\beta(s)$ in (3.10) controls the speed of convergence for the estimated decision boundary - it is basically a positive decreasing function of the learning stage s. However, if $\beta(s)$ decays too fast, the algorithm will be forced to stop at an earlier

stage (since almost all the experimental trials are allocated near the estimated decision boundaries). This will apparently accelerate the convergence of the algorithm, but lose the opportunity of exploring the output model at some unlabeled input locations. Due to the sampling feature of the proposed framework (i.e. only one query at each learning stage), it is recommended that $\beta(s)$ be chosen to have a moderately slow decay. For example, one may consider

$$\beta(s) = a_0 \times (c)^{-s},$$

where a_0 is a positive constant and $c \gtrapprox 1$ (i.e. c is a positive constant greater than (but close to) one). Note that such $\beta(s)$ apparently has a slower decay than the exponential function. Numerical results also show that it performs well for various types of computer experiments (see Section 4).

- The algorithm stops usually when the experimental budget is expended or the estimated decision boundary has achieved a stable status.
- Note that one can speed up the learning of Algorithm 3.1 by selecting a batch of samples at each learning stage. Such sampling process refers to the so-called batch-mode active learning [52].

It should be mentioned that if we know a priori that the target region has a piecewise linear boundary (e.g. the stability region when $\Delta = 0$ in Figure 1 or target regions comprising of a number of unidentifiable linear constraints), the strategy of using the RBF kernel SVM may not be plausible when the size of the training data is small. An alternative strategy is to employ a technique analogous to the so-called Hierarchical Mixing Linear SVMs (HMLSVMs) introduced in [55]. The idea of HMLSVMs is quite similar to that of the hierarchical classification tree, but it assigns different weights to the training data and uses the linear SVM (thus known as the WLSVM) as the decision function to split the nodes. The detailed description of how this technique is implemented at each learning stage of our proposed framework is given in Appendix C.

4. Performance Assessment. In this section, we illustrate the proposed strategy on the two motivating examples and evaluate its performance. All numerical results presented here were performed by using the statistical software package R (version 3.2.5) and executed on a 3.2 GHz Intel® Core™ i5-6500 processors with 16 GB of memory under the operating system of Microsoft Windows 7 64-bit Service Pack 1 (SP1). The SVM techniques were implemented by utilizing the R package "e1071" and "wSVM", while all the tuning parameters used in the proposed framework were determined based on a number of sensitivity tests.

4.1. Queueing Example. Let us consider the queueing example with a service-mode switching time $\Delta = 0.5$ introduced in Subsection 2.1. The "true" stability region was obtained by simulating the system at $50 \times 70 = 3,500$ superimposed grid points, as shown in the left panel of Figure 1. We now demonstrate how to estimate the stability region based on active learning with SVM, as given in Algorithm 3.1. Since no evidence indicates that the target region has a linear (or piecewise linear) boundary, the RBF kernel is a reasonable choice for fitting the WSVM model. Note that the initial training set $\mathcal{P}_1^*$ is obtained by UD, whose size is chosen to be $n_1 = 10$ (see Figure 3). For fitting the RBF kernel WSVM model,

the parameters (C, γ) are chosen to minimize the prediction error (based on the 10-fold cross-validation) based on the set of candidate grid points (e^i, e^j), where $i, j = -10, -9, \ldots, 9, 10$. As a result, the grid search yields the best parameters $(C, \gamma) = (1096.633, 0.0183)$. Since all computations now are based on the superimposed grid points, at each learning stage the estimated boundary of the target region has the form of (3.12), wherein ϵ is chosen to be 0.01. We then update the weight function by using (3.17), wherein we choose $\beta(s) = 100 \times (1.1)^{-s}$ so that it has a moderately slow decay as the design stage moves. The estimated boundaries $\partial\tilde{T}_s(z)$ and the contour lines of the associated weight functions $f_s(x)$ at learning stages $s = 8$ $(n_s = 17)$, $s = 19$ $(n_s = 28)$, $s = 46$ $(n_s = 55)$, $s = 66$ $(n_s = 75)$ are depicted in Figure 3 and Figure 4, respectively.

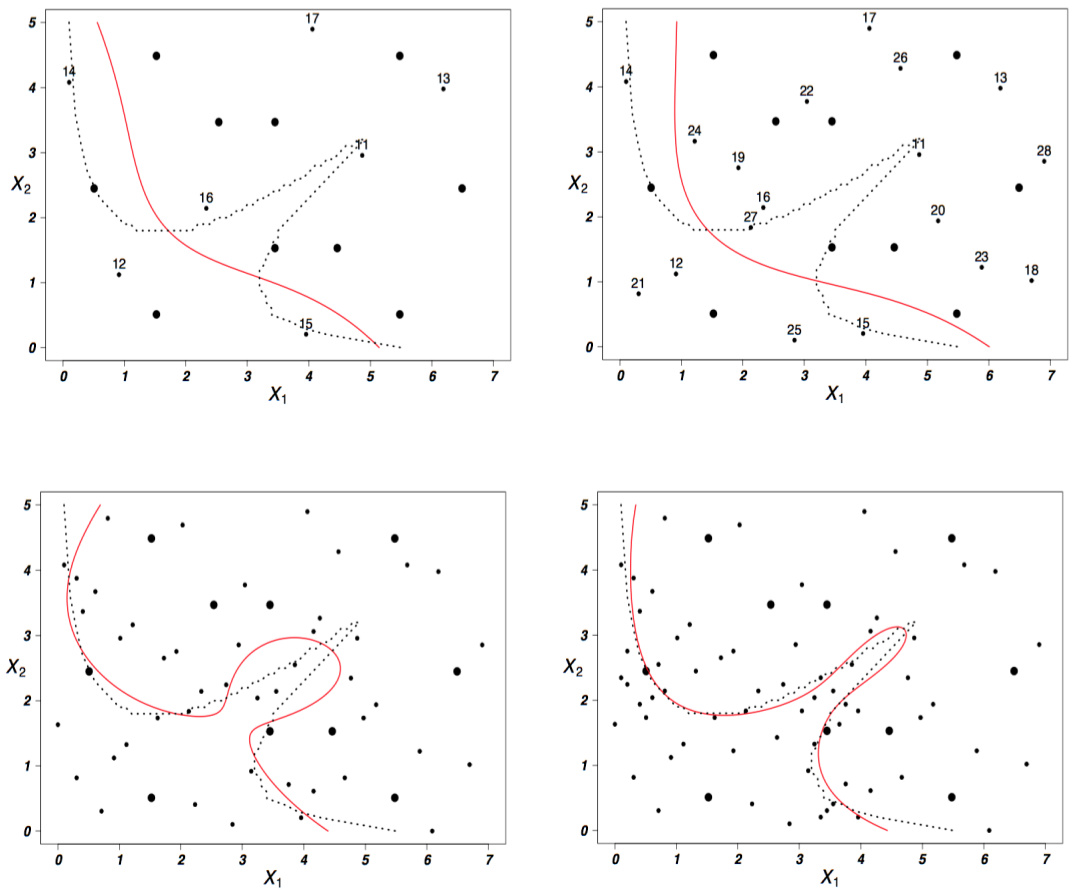

Figure 3. *The estimated boundary $\partial\tilde{T}_s(z)$ (the red curve) based on WSVM at the learning stage $s = 8$ (upper left), $s = 19$ (upper right), $s = 46$ (lower left) and $s = 66$ (lower right), respectively. The dotted lines represent the true stability region, the big dots represent the selected initial training data, while the small numbered dots represent the selected samples at later learning stages.*

As can be seen from Figure 3, the estimated target region gets closer to the true one as

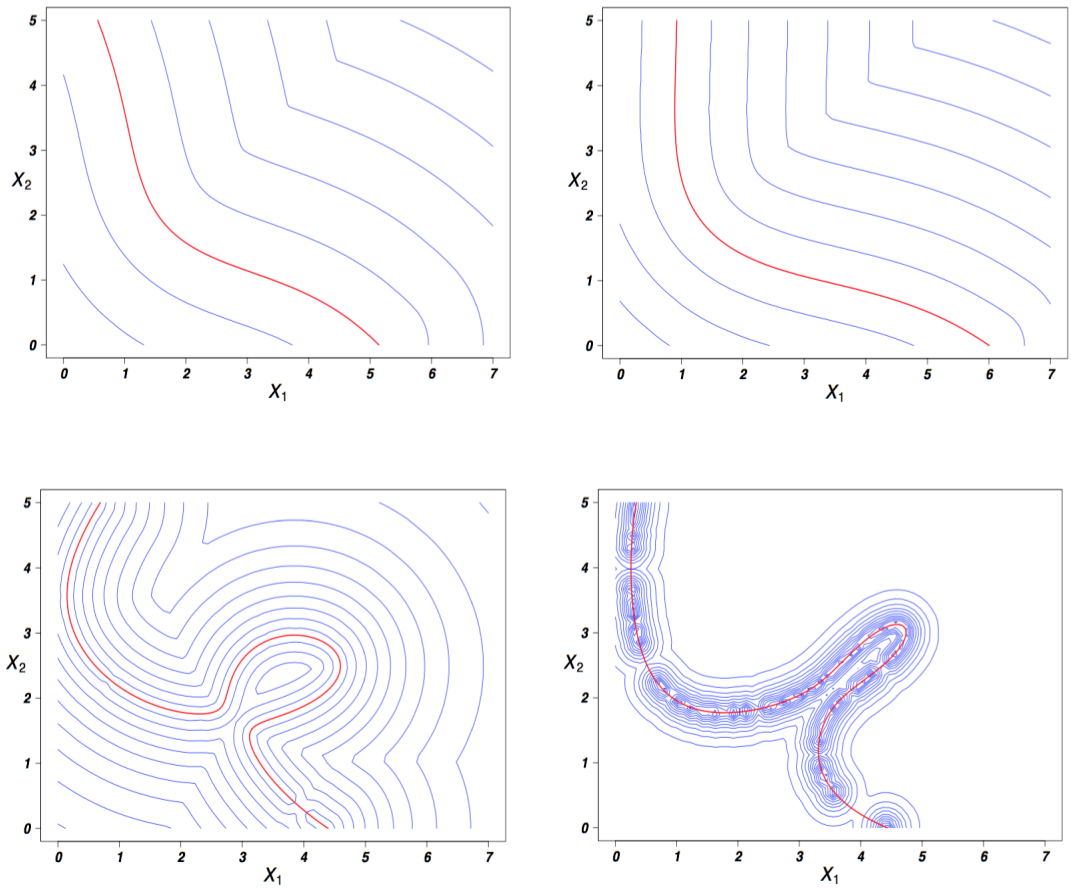

Figure 4. *The contour lines of the weight functions $f_s(x)$ at the learning stages $s = 8$ (upper left), $s = 19$ (upper right), $s = 46$ (lower left) and $s = 66$ (lower right). The red line represents the weight function on the estimated boundary.*

the learning stage increases. In addition, the initially selected training set uniformly spreads out the experimental trials over the input domain, while thereafter it tends to allocate the experimental trials closer and closer to the boundary of the estimated target region. Note that an adequate choice of the control process $\beta(s)$ in the weight function $f_s(x)$ allows us to allocate experimental trials in the areas not been explored at early design stages. These phenomena are also supported by the contour plots of the weight function shown in Figure 4.

To evaluate the efficiency of Algorithm 3.1, we compute the label error rate of the estimated target region based on the superimposed 3,500 grid points and compare with that obtained by (i) passive learning with SVM (i.e. allocate all experimental trials at once based on the UD); (ii) GASP modeling technique along with a sequential sampling strategy for contour estimation (a modification of the method introduced in [44], see Appendix B for detailed steps); and (iii) the GASP modeling technique based on pure UD (i.e. allocate all experimental trials at once).

Note that for (ii), we take the reciprocal of the primary response (i.e. $y(x)=1/\theta(x)=1/\theta_1(x)$) for building the GASP model. Thus, the contour of interest has height "close to zero" (but not equal to zero) and can be estimated in a stepwise manner by using a stage-dependent improvement function. The numerical results with respect to different sizes of training samples are depicted in Figure 5, where each GASP model is estimated by using the R package "GPfit" [39].

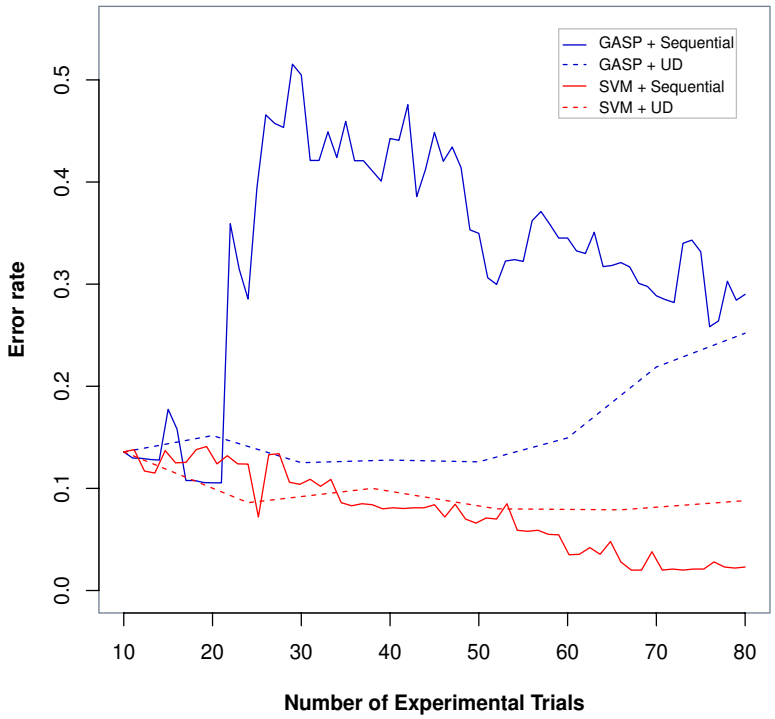

Figure 5. *The error rates for estimating the stability region based on passive and active learning with SVM (red lines) and the GASP model for contour estimation (blue lines). The solid lines refer to the results by using the sequential sampling strategy; the dashed lines refer to the results by sampling at once based on UD.*

As can be seen from Figure 5, the error rate of the proposed strategy based on active learning reduces in a steady manner to zero as the experimental budget increases. Note that the convergence may not be particularly fast, which is due to the spiky shape of the target region in the middle part of the experimental domain. However, the estimation error rate is no more than 6% when the total number of experimental trials exceeds 52. On the other hand, the error rate based on the GASP model is significantly larger. The estimation error rate remains high (above 26%), even when the number of experimental trials is increased to 80. Such a slow convergence rate may be induced by over-allocating experimental trials outside the target region, which is a consequence of utilizing the sampling strategy based on improvement function. Not surprisingly, the strategy of allocating all experimental trials at once based on UD does not work well for estimating the desired boundary. Even if we increase the size of training data to 80, the estimation error rate is always above 9% for both the GASP and SVM modeling methodologies. In conclusion, the strategy of employing active

learning with SVM outperforms that of employing passive learning and the GASP model for estimating the desired input boundary of infinite valued outcomes.

4.2. The Laser Cutting Process. Let us consider the laser cutting problem introduced in Subsection 2.1, where the process has four control factors and the experiment is emulated by the following functional relationship

(4.1)

$$y(x_1,x_2,x_3,x_4) = \begin{cases} \frac{\left[(x_1+x_2-7)^2+(x_3+x_4-7)^2-62\right]^2+\left[(x_1+x_2-7)^2-0.5(x_1+x_2-7)-0.5(x_3+x_4-7)-1.5\right]^2}{100}+10 & \text{if } (x_1+x_2)^2+(x_3+x_4)^2>80, \\ 0 & \text{if } (x_1+x_2)^2+(x_3+x_4)^2\leq 80. \end{cases}$$

Suppose now we are interested in identifying the limits of the cutting process (i.e. the input boundary that results in $y(x) = 0$) over the 4D hypercube domain $D = [0, 7]^4$ and 10^4 grid points are superimposed over D. The prediction error rates for the methods based on GASP and SVM with two sampling strategies are given in Figure 6, where the remaining simulation setups are the same as those given in Subsection 4.1. As can be seen from Figure 6, the error rate of the proposed strategy based on active learning drops down stably to zero as the experimental budget increases. It is no more than 5% when the total number of experimental trials exceeds 50. On the other hand, the error rate based on GASP with the sequential sampling strategy appears to be significantly larger and the decay is relatively slow. The estimation error rate remains high (above 17%), even when the number of experimental trials is increased to 80. Interestingly, estimation does not benefit much by using the sequential sampling strategy based on contour extraction. In summary, the strategy of utilizing active learning with SVM (i) performs slightly better than that of utilizing passive learning (i.e. sampling at once by utilizing UD); and (ii) significantly outperforms both sampling strategies based on GASP.

We next examine the performance of the proposed method for larger-scale systems. In order to reduce the computational burden increased by high dimensionality, here we provide some useful implementation guidelines.

Guidelines for Conducting Experiments for Large-scale Systems

• Consider a set of rather "coarse grid" Z points over the experimental domain D. For example, here we consider 10 grid points (0.7, 1.4, 2.1, 2.8, 3.5, 4.2, 4.9, 5.6, 6.3, 7.0) for each dimension x_i. Thus, there are 10^4 grid points in D. A quick numerical examination shows that there are 2,082 grid points with the output $y(x) = 0$.

• Select a simpler version of (3.3) to implement the quasi-Monte Carlo method. For example, by choosing $K = 0$ in (3.3), we consider "uniformity" merely through one subregion $D_1(z)$ (i.e. the hyper-rectangular subregion from 0 to a grid point z) instead of through 2^K subregions.

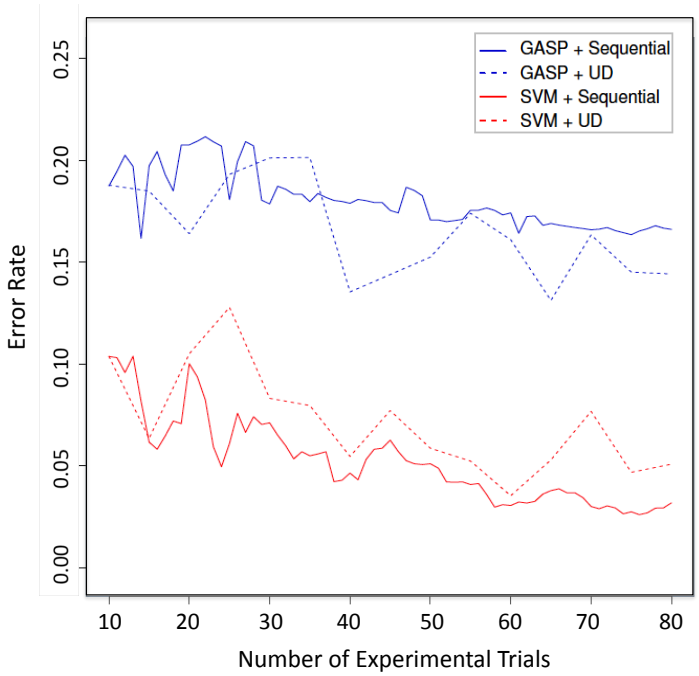

Figure 6. *The error rates for estimating the 4D laser cutting boundaries based on passive and active learning with SVM (red lines) and the GASP model for contour estimation (blue lines). The solid lines refer to the results by using the sequential sampling strategy; the dashed lines refer to the results by sampling at once based on UD.*

That is, the NUD minimizes

$$\widehat{WCCD}_{f_s,p}(n,\mathcal{P}) = \left\{ \frac{1}{|Z|} \sum_{z\in Z} \left| \frac{N(D_1(z),\mathcal{P})}{n} - F(D_1(z)) \right|^p \right\}^{1/p}, \tag{4.2}$$

where $\mathcal{P} \in Z(n) \subset Z$, $|Z|$ represents the cardinality of Z, and $F(D_1(z)) = \sum_{g\in D_1(z)} f_s^*(g)$ is the proportion of points expected to be allocated in $D_1(z)$, where $\sum_{z\in Z} f_s^*(z) = 1$ (summation of all normalized weights is equal to one). This simplified measure is the hybrid of the so-called L_p-discrepancy and F-discrepancy [9], based on which the optimal design requires significantly less computation load. For simplicity, we can choose $p = 1$ in (4.2).

• To further reduce the computation load of finding the best new design point, we can implement the accelerated algorithm shown in Appendix A. The idea is to augment on the lower-dimensional design and increase one dimension each step to find the location of a new design point.

Let us now consider the laser cutting problem with eight control factors and the experiment

is emulated by the following functional relationship

$$
\begin{aligned}
&y(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \\
&= I\{(\textstyle\sum_{i=1}^{4} x_i)^2 + (\sum_{i=5}^{8} x_i)^2 > 120\} \times \\
&\tfrac{1}{100}\Big\{\big[(\textstyle\sum_{i=1}^{4} x_i - 7)^4 + (\sum_{i=5}^{8} x_i - 7)^2 - 62\big]^2 + \big[(\textstyle\sum_{i=1}^{4} x_i - 7)^2 - 0.5(\sum_{i=1}^{4} x_i - 7) - 0.5(\sum_{i=5}^{8} x_i - 7) - 1.5\big]^2\Big\} + 10.
\end{aligned} \tag{4.3}
$$

Analogously, we are interested in identifying the limits of the cutting process (i.e. the input boundary that results in $y(x) = 0$) over the 8D hypercube domain $D = [0, 3]^8$. To accelerate the implementation of the proposed procedure by using the above guidelines, we consider 3 coarse grid points $(1.0, 2.0, 3.0)$ for each input dimension x_i. Thus, there are $6,561$ grid points superimposed over D, among which $2,299$ grid points have the output value $y(x) = 0$. Further, the quasi-Monte Carlo method is implemented by using a simpler version of the discrepancy measure (say, (4.2) with $p = 1$) and the accelerated search algorithm shown in Appendix A, with an initial design of size $n_0 = 20$. The prediction error rates for the methods based on GASP and SVM with two different sampling strategies are given in Figure 7.

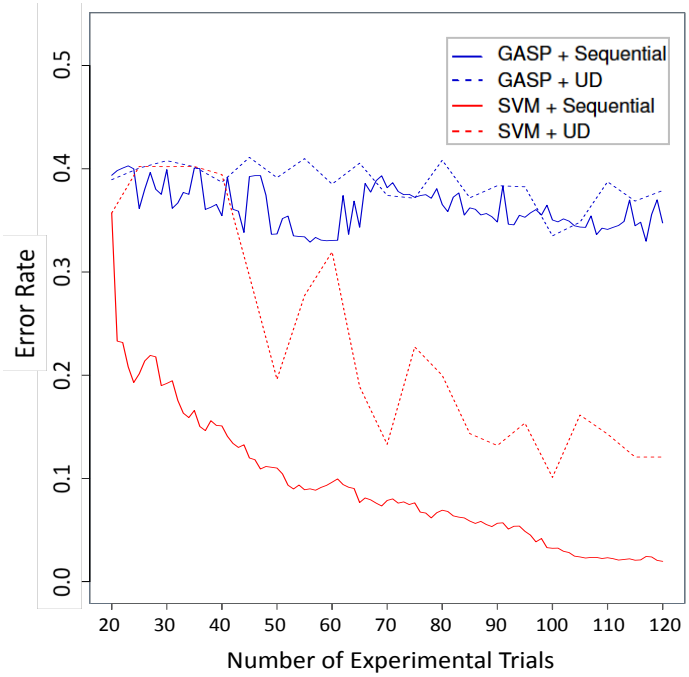

Figure 7. *The error rates for estimating the 8D laser cutting boundaries based on passive and active learning with SVM (red lines) and the GASP model for contour estimation (blue lines). The solid lines refer to the results by using the sequential sampling strategy; the dashed lines refer to the results by sampling at once based on UD.*

It can be seen that Figure 7 reveals similar patterns to those shown in Figure 6. The error rate of the proposed method based on active learning and SVM drops down quickly to zero, as the experimental budget increases. It is no more than 5% when the total number of experimental trials exceeds 100. On the other hand, the error rates based on GASP with both sampling strategies appear to be significantly larger and their decay is rather slow. It is also

worth noting that for the method based on SVM, active learning significantly outperforms passive learning (i.e. sampling at once based on UD) in terms of both estimation efficiency and stability.

5. Conclusion and Discussion. Although the GASP model is particularly popular for output analysis of computer experiments, it nevertheless exhibits certain drawbacks in addressing the problem under study: (i) it cannot predict well outputs exhibiting sharp changes in nearby locations due to its underlying assumption of continuity and differentiability [19]; (ii) it may encounter serious computational issues (i.e. parameter estimation) for large designs [30]; (iii) the associated design methods may not provide efficient allocations for estimating the desired outputs. In this article, we propose a strategy for locating infinite discontinuities in computer experiments leveraging ideas from active learning. The strategy formulates the problem as a binary classification one and employs active learning with an SVM classifier and UD to predict the output discontinuities and allocate the experimental trials in a more effective manner. Numerical results show that, given a limited experimental budget, the proposed strategy is superior to the modeling strategy based on GASP, as well as strategies based on a pure uniform design (i.e. with all experimental trials allocated at once by UD) in terms of both efficiency and prediction accuracy.

It should be mentioned that the sampling technique here is different from the conventional methods in active learning, for which the initial training set (for passive learning) is usually obtained by random sampling or LHD. In general, the quality and cost of the initially selected training set in active learning are beyond consideration. However, we believe that for computer experiments significant attention needs to be paid to the selection of the initial design, since it can take up a large portion of the total experimental budget and thus significantly influences the output analysis. The sampling method employed in our strategy belongs to the class of quasi-Monte Carlo methods and has proven more robust than random sampling and other space filling designs. Further, with the placement of a well-designed weight function, the UD can be automatically incorporated into the sampling process of active learning. For practical implementation, we have also provided an accelerated algorithm so that the solution of UD can be fairly well approximated within a more reasonable computational time. Such acceleration is particularly useful and necessary for designs with a large number of input factors. Finally, another learning technique called HMLSVMs is introduced so as to better deal with infinite valued discontinuities that lead to piecewise linear shapes.

We believe that the proposed strategy can be extensively used to identify other types of output discontinuity or non-differentiability in computer or other physical experiments. For example, to identify (finite) *jump discontinuities* one can examine the magnitude of response $y(x)$ in the neighborhood of each input point x; to identify pathological structures like *kinks* and *cusps* (which are essential in manufacturing processes and computer graphics) one may examine the "gradients" of response in the neighborhood of each input point x and determine suitable class labels (note that the gradients for kinks and cusps are not continuous). However, these problems will require a more delicate/complicated formulation.

Appendix A. Accelerated Algorithm for Finding an NUD.

The accelerated algorithm for constructing an NUD is shown as follows.

The following theorem describes how much computation can be saved by Algorithm A.1

Algorithm A.1 Acceleration for Finding an NUD

1: Consider a set of superimposed grid points Z over D. Select $\psi_0 \subset \{1, \ldots, K\}$, find the optimal design $\mathcal{P}_0^*$ in the selected $|\psi_0|$-dimensional space (e.g. by using the exhaustive search or Switching Algorithm). Set $i = 0$.
2: **while** $|\psi_{i+1}| < K$ **do**
3: Select $\psi_{i+1} \subset \{1, \ldots, K\} \setminus \psi_i$, set $\psi_{i+1} = \psi_{i+1} \cup \psi_i$.
4: Find the optimal design $\mathcal{P}_{i+1}^*$ in $Z^{\psi_{i+1}}$ by augmenting the design $\mathcal{P}_i^*$ in the lower dimensional input domain Z^{ψ_i}.
5: Set $i = i + 1$.
6: **end while**
7: **return** $\mathcal{P}_{i+1}^*$

for finding an NUD.

Theorem A.1 (Computational time of Algorithm A.1). *If the number of superimposed grid points in each dimension of D is taken to be $N^{\frac{1}{K}}$ and $|\psi_i| = d(i+1)$ for each iteration i (where d is divisible by K), the accelerated exhaustive search has the computational time $O(N^{\frac{nd}{K}})$, while the accelerated Switching Algorithm has the computational time up to $O(N^{\frac{d}{K}(2+p)})$.*

Proof. Note that $|\psi_i| = d(i+1)$ indicates that at each iteration i, an additional d-dimensional subspace of D is aggregated with the current design space so as to find the optimal design $\mathcal{P}_i^*$. Therefore, for fixed values of K, d and n, the accelerated exhaustive search requires the computational time

$$\binom{K}{d}\binom{N^{\frac{d}{K}}}{n} + \binom{K-d}{d}\binom{N^{\frac{d}{K}}}{n} + \cdots + \binom{d}{d}\binom{N^{\frac{d}{K}}}{n} = O(N^{\frac{nd}{K}}).$$

On the other hand, the accelerated Switching Algorithm requires the computational time up to

$$(K/d) \cdot O(N^{\frac{d}{K}(2+p)}) = O(N^{\frac{d}{K}(2+p)}).$$

Note that in practice the accelerated algorithm would have a dramatic reduction on the computational time of finding an NUD (though there may be a minor loss of uniformity in the primary space). To see this, let us recall the early example with $K = 3$, $d = 1$, $n = 6$, $p = 2$ and $N = 10^3$. The accelerated exhaustive search requires merely 10^6 computations for finding an NUD, while the search requires 10^{18} computations without acceleration. The accelerated Switching Algorithm requires up to 10^4 computations for finding an NUD, while the search requires up to 10^{12} computations without acceleration.

To evaluate the numerical performance of Algorithm A.1, we consider a simple design area $D = [0, 1] \times [0, 1]$. Suppose there are 100 candidate grids superimposed over D (so $100^{1/2} = 10$ grids for each dimension), and we choose $d = 1$ and $p = 2$. The CCD are then computed by utilizing (3.3) (with F chosen as an identity function) based on the four algorithms: Exhaustive Search, Switching, Accelerated Exhaustive Search, and Accelerated Switching. The results for $n = 1, \ldots, 10$ are shown in Figure 8. Note that the CCD for the Switching algorithm is computed by taking the average of 1,000 initial designs based on random sampling, while

the CCD for the Accelerated Switching algorithm is computed by taking the average of all solutions based on a set of NUDs found in the first dimension (i.e. a set of $\mathcal{P}_1^*$). As can be seen, for this small design area both the Switching and Accelerated Switching algorithms approximate very well the optimal design based on Exhaustive Search. This provides a strong numerical evidence that the proposed accelerated algorithm is adequate for finding an NUD.

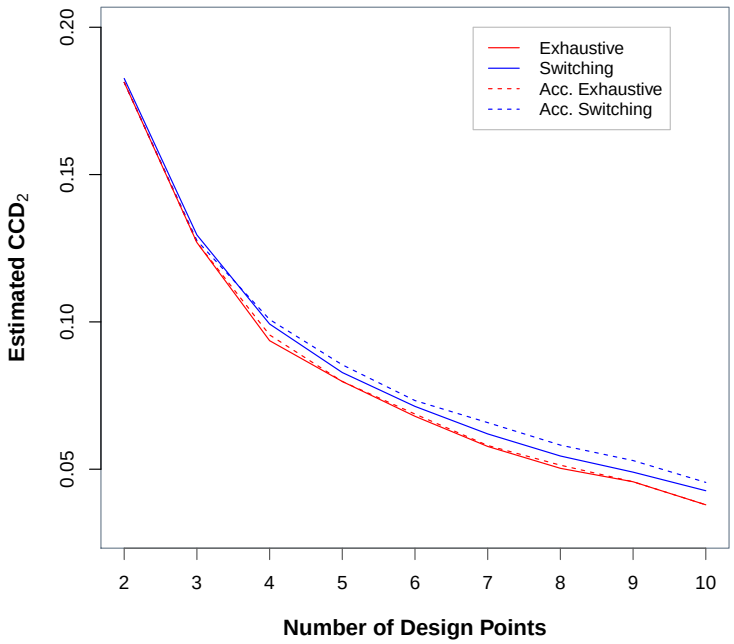

Figure 8. *The estimated CCD (with $p = 2$) for the Exhaustive, Switching, Accelerated Exhaustive and Accelerated Switching algorithms with respect to the number of allocated experimental trials. Here the input domain is $D = [0, 1]^2$.*

We next consider the laser cutting example in Subsection 4.2 with an 8D hypercube domain $D = [0, 3]^8$, while 3 grid points $(0.5, 1.5, 2.5)$ are considered for each input dimension so as to evaluate the performance of the above algorithms. Thus, there are $N = 3^8 = 6,561$ grid points superimposed over D. To accelerate the computation of Algorithm A.1, we choose $d = 2$ (augmenting two dimensions one at a time to find the optimal design), $K = 0$ and $p = 1$ in (3.3), i.e., a simplified version of WCCD with merely one subregion $D_1(z)$ and the L_1-norm distance is considered. Since now the design is unweighted (i.e. F is chosen as an identity function), it is clear that $F(D_1(z)) = |D_1(z)|/|D|$, which corresponds to the proportion of the volume taken up by the subregion $D_1(z)$. Under such setup, the WCCD reduces to the form of so-called L_1-discrepancy. Figure 9 yields the values of L_1-discrepancy associated with the number of design points allocated in D based on the above algorithms. As can be seen, for this 8D domain with coarse grid points, the trend of discrepancy measure is somewhat different from that shown in Figure 8 - it does not drop down monotonically as we

increase the number of design points. However, numerical values show that both the Switching and the Accelerated Switching algorithms approximate fairly well the optimal design based on Exhaustive Search. The empirical evidence renders strong support that our proposed accelerated algorithm performs well in finding an NUD for larger scale systems.

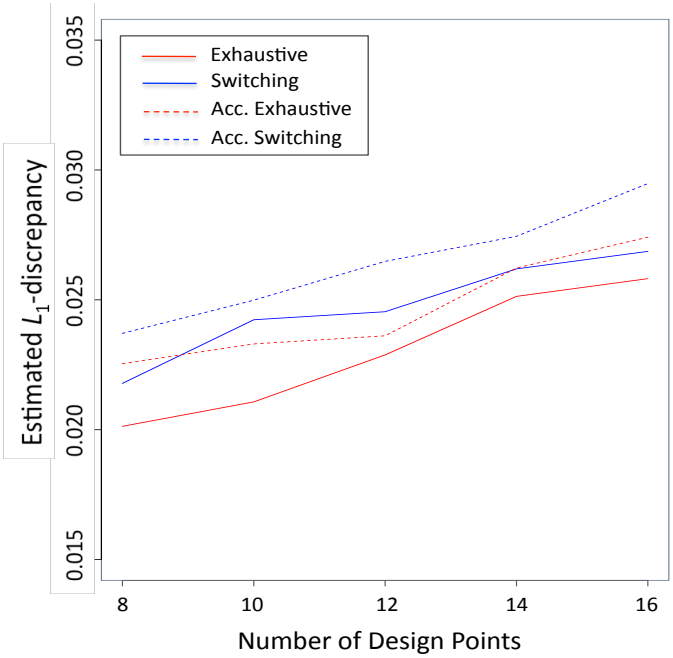

Figure 9. *The estimated L_1-discrepancy for the Exhaustive, Switching, Accelerated Exhaustive and Accelerated Switching algorithms with respect to the number of allocated design points. Here the input domain is $D = [0, 3]^8$.*

Appendix B. A sequential sampling strategy based on GASP and contour estimation.

Suppose one would like to estimate a contour $S(a) = \{x : y(x) = a\}$ (the set of input points with the response value a) based on the GASP model, a useful strategy is to start with a relatively small experimental design (called initial design) and then sequentially allocate experimental trials near the estimated contour and/or input locations with large prediction uncertainty [44]. Specifically, consider the following *improvement function*

$$I(x) = \epsilon^2(x) - \min\{(y(x) - a)^2, \epsilon^2(x)\}, \tag{B.1}$$

where $\epsilon(x) = \alpha s(x)$ for some $\alpha > 0$ and $y(x) \sim \mathrm{N}(\hat{y}(x), s^2(x))$. Here $\hat{y}(x)$ and $s^2(x)$ represent the best linear predictor and the associated mean squared error at any non-sampled point x, respectively. Therefore, at each design stage one tends to select an untried point that maximizes $I(x)$ over the input space. However, due to the uncertainty of un-sampled design points, the expected value of the improvement function is considered instead. The following mathematical expression, which was given in [7], provides a simpler way for calculating

$E[I(x)]$:

$$E[I(x)] = \left[\alpha^2 s^2(x) - (\hat{y}(x) - a)^2 - s^2(x)\right] \cdot [\Phi(u_2) - \Phi(u_1)] + s^2(x)\left[u_2\phi(u_2) - u_1\phi(u_1)\right] + 2s(x)(\hat{y}(x) - a)\left[\phi(u_2) - \phi(u_1)\right], \tag{B.2}$$

where $u_1 = (a - \hat{y}(x) - \alpha s(x))/s(x)$, $u_2 = (a - \hat{y}(x) + \alpha s(x))/s(x)$, $\Phi(\cdot)$ and $\phi(\cdot)$ represent the cdf and pdf of a standard normal random variable, respectively. Thus, in the standard contour estimation problem a not yet tested point x is chosen to maximize (B.2) at each design stage.

Note that the goal is to locate the "boundary" for the set of input points x with $y(x) = 0$, which is different from identifying the contour line of height $a = 0$ associated with a continuous response surface. Thus, we need to modify accordingly the above sequential sampling procedure. Suppose there are N_j available experimental trials at a particular design stage j and we denote by $x_1, \ldots, x_{N_j}$ and $y(x_1), \ldots, y(x_{N_j})$ the input locations and their associated outputs, respectively. After fitting the GASP model based on the N_j experimental trials, the estimated contour (i.e. target boundary) is given by

$$\partial\tilde{T}_j(x) = \{x \in D : 0 < \hat{y}(x) \leq a_j\}, \tag{B.3}$$

where

$$a_j = \frac{1}{2} \cdot \min\{y(x_i) : \ y(x_i) > 0, \ i = 1, \ldots, N_j\}. \tag{B.4}$$

We next introduce a *stage-dependent improvement function*

$$I_j(x) = \epsilon^2(x) - \min\{(y(x) - a_j)^2, \epsilon^2(x)\}, \tag{B.5}$$

and thus the next-stage design point is chosen as

$$x^* = \arg \max_{x \notin \{x_i, \ldots, x_{N_j}\}} E[I_j(x)], \tag{B.6}$$

where the computation of $E[I_j(x)]$ can be done by using (B.2).

Since $y(x) = 0$ for all $x \notin T(x)$, the sampling strategy based on the concept of improvement function tends to over-allocate experimental trials outside the target region (this is also validated by numerical investigations). Unfortunately, certain portions of such trials are not particularly informative for estimating the boundary $\partial T(x)$ (e.g. data far away from the boundary). The goal of utilizing a_j instead of 0 in (B.5) is to reduce the amount of over-sampled data outside the target region $T(x)$. Intuitively, the approximation of $\partial T(x)$ should then be done in a more efficient manner. Now we proceed to prove that the estimated $\partial\tilde{T}_j(x)$ converges to the true boundary $\partial T(x)$ as $N_j \to \infty$. It is clear that the GASP model is not able to fit well the primary output surface $y(x)$, which is neither continuous nor differentiable on $\partial T(x)$. However, as we learn that $y(x)$ becomes continuous on $\partial T(x)$ after a reciprocal transformation (though it is still not differentiable), it suffices to show [44]:

$$\lim_{j \to \infty} \sup_{x \in D \setminus \partial T(x)} E[I_j(x)] = 0. \tag{B.7}$$

Let us now break the input domain D into two parts: one for $x \in T(x)$ and the other for $x \notin T(x)$ (both not including the boundary $\partial T(x)$). Since the output surfaces $y(x)$ for both parts are continuous and differentiable, the proof of Theorem 1 in [44] yields

$$\sup_{x \in D \setminus \partial T(x)} s^2(x) \to 0 \quad \text{as } N_j \to \infty,$$

and thus

$$\sup_{x \in D \setminus \partial T(x)} I_j(x) \to 0 \quad \text{as } N_j \to \infty.$$

This then implies (B.7) and the proof of convergence is complete. The detailed steps of the above procedure based on a set of superimposed grid points Z are shown in Algorithm B.1.

Algorithm B.1 Sampling based on contour estimation and GASP

1: Determine an appropriate experimental domain D associated with a set of superimposed grid points Z. Set the design stage $j = 1$ and determine the size of the initial design n_1. Allocate an adequate initial design $\mathcal{P}_1^*$ (e.g. by UD) based on Z and obtain the experimental outcomes $y(z)$ for all $z \in \mathcal{P}_1^*$. Compute a_1 by (B.4) and fit the GASP model based on all $z \in \mathcal{P}_1^*$. Compute $E[I_1(z)]$ by (B.2) for all $z \in Z \setminus \mathcal{P}_1^*$.
2: **while** the stopping criterion is not met **do**
3: Select the next-stage design point z^* which maximizes $E[I_j(z)]$ over the set $Z \setminus \mathcal{P}_j^*$.
4: Obtain the experimental outcome for z^* and set $\mathcal{P}_{j+1}^* = \mathcal{P}_j^* \cup \{z^*\}$.
5: Compute a_{j+1} by (B.4) and fit the GASP model based on all $z \in \mathcal{P}_{j+1}^*$.
6: Set $j = j + 1$.
7: **end while**
8: **return** $\partial \tilde{T}_{j+1}(z)$

Appendix C. Active Learning with HMLSVMs.

We briefly introduce how to develop a learner based on HMLSVMs by utilizing the idea of classification tree. Let $[I]$ be a node of the hierarchical tree and denote the root node by $[0]$, the left child node of $[I]$ by $[I]^+$, the right child node of $[I]$ by $[I]^-$, respectively. Let $g_{[I]}(x)$ be the decision function of fitting an LSVM at node $[I]$ and denote the associated "order of split" in the hierarchical tree by i. We simply write $g_{[I]}(x) = g^{(i)}(x)$ and, it is clear that $g_{[0]}(x) = g^{(1)}(x)$. For the classification tree having m splits, we denote the set of all decision functions by $F^m = \{g^{(1)}(x), \ldots, g^{(m)}(x)\}$. We next introduce how to split the nodes so as to grow the decision tree.

Denote the set of training data associated with a node $[I]$ by $\mathcal{T}_{[I]}$, where $\mathcal{T}_{[0]} = \{(y_1, x_1), \ldots, (y_n, x_n)\}$. The training data associated with the left child and right child node of $[I]$ are then denoted by $\mathcal{T}_{[I]^+} = \{x \in \mathcal{T}_{[I]} : g_{[I]}(x) > 0\}$ and $\mathcal{T}_{[I]^-} = \{x \in \mathcal{T}_{[I]} : g_{[I]}(x) < 0\}$, respectively. In order to best split the training data in a node, we need to define the so-called *host training center* and *guest training center*. Given the current tree with m splits, denote the set of training samples having "wrong classifications" by $\mathcal{W}$. For any $x_i \in \mathcal{T}_{[0]}$, denote the set of its k "nearest" training samples with the same labels as y_i by $N_k(x_i)$. The host training

center is then defined as

$$x_h = \arg\max_{x_i \in \mathcal{W}} \sum_{x_j \in N_k(x_i)} \exp\left\{-\frac{\|x_j - x_i\|^2}{\sigma^2}\right\} E_j, \tag{C.1}$$

where $E_j = \min\left\{1, \max\{0, 1 - y_j \cdot G^{(m)}(x_j)\}\right\}$ and σ^2 is a positive constant. By this definition, the selected host training center appears to have the largest density of misclassified training samples over its k nearest neighbors. Suppose now $x_h \in \mathcal{T}_{[I]}$ for some node $[I]$, the guest training center is then defined as

$$x_g = \arg\max_{x_i \in \bar{N}_k(x_h)} \sum_{x_j \in N_k(x_i)} \exp\left\{-\frac{\|x_i - x_j\|^2 + \|x_j - x_h\|^2}{\sigma^2}\right\}, \tag{C.2}$$

where $\bar{N}_k(x_h)$ represents the set of x_h's k nearest training samples, with different labels from y_h. In short, the guest training center is a training sample that is close to the host training center and at the same time has a large density of correctly classified training samples over its k nearest neighbors. Note that such setup is quite similar to the concept of "local likelihood", while the goal is to split node $[I]$ based on the two selected centers x_h and x_g.

To develop the decision function to split node $[I]$, we first define the weights of the training samples in $N_k(x_h)$ and $N_k(x_g)$ as

$$W_i = \begin{cases} \exp\left\{-\frac{\|x_i - x_h\|^2}{\sigma^2}\right\}, & \text{if } x_i \in N_k(x_h); \\ \exp\left\{-\frac{\|x_i - x_g\|^2}{\sigma^2}\right\}, & \text{if } x_i \in N_k(x_g). \end{cases} \tag{C.3}$$

As can be seen, the weight of each training sample in both sets becomes exponentially larger as it gets closer to x_h or x_g and exponentially smaller as it gets farther away. The WLSVM is then used to develop a decision function $g_{N_k}(x)$ based on the two sets $N_k(x_h)$ and $N_k(x_g)$. Once we have $g_{N_k}(x)$, the decision function for all training samples in node $[I]$ is obtained by mixing $g_{N_k}(x)$ with the decision function of its parent node $[I_p]$, that is,

$$g_{[I]}(x) = \begin{cases} \max\{g_{[I_p]}(x), g_{N_k}(x)\}, & \text{if } [I] = [I_p]^-; \\ \min\{g_{[I_p]}(x), g_{N_k}(x)\}, & \text{if } [I] = [I_p]^+. \end{cases} \tag{C.4}$$

Denote by $g_{[I]}(x) = g^{(m+1)}(x)$, then after the $(m+1)$-th split, the set of all decision functions for the tree is given by

$$F^{m+1} = F^m \cup \{g^{(m+1)}(x)\}. \tag{C.5}$$

Denote by $F_s^m = \{g_s^{(1)}(x), \ldots, g_s^{(m)}(x)\}$ the set of all decision functions obtained by HMLSVMs at a particular learning stage s, where

$$g_s^{(i)}(x) = \text{sign}\left(\langle w_s^{(i)}, x)\rangle + b_s^{(i)}\right)$$

is the linear decision function for the i-th split at stage s, $i = 1, \ldots, m$. Then, for each input point x we can analogously compute the distance (3.9) and the weight (3.10) so that the

weighted UD can be utilized to sample an instance at the next stage. Note that due to the structure of the developed decision tree, the computation of (3.9) can be intricate as the size of the tree becomes large - even when the input dimension is low. However, for some cases where the estimated target regions have a "convex" shape, the computation can be done in a simpler way. To see this, for any input point x inside the estimated target region $\tilde{T}_s(x)$, its distance to the boundary can be directly computed by

$$d(x, \partial\tilde{T}_s(x)) = \min_{i\in\{1,\dots,m\}} \frac{|\langle w_s^{(i)}, x\rangle + b_s^{(i)}|}{\|w_s^{(i)}\|}. \tag{C.6}$$

If an input point x is outside the estimated target region, one will have to identify which facet/vertex of $\partial\tilde{T}_s(x)$ is closest to x and then compute the distance accordingly. We summarize the detailed steps of our framework for estimating the target region with piecewise linear boundaries in Algorithm C.1.

Algorithm C.1 Active Learning with HMLSVMs

1: Determine an appropriate training domain D associated with a set of superimposed grid points Z. Set the learning stage $s = 1$ and determine the size of the initial training set n_1. Find the initial training set $\mathcal{P}_1^*$ of size n_1 that minimizes (3.3) based on Z, where $p = 2$ and the weight function is initially placed as $f_0(x) \equiv 1$. Train an HMLSVMs classifier based on $\mathcal{P}_1^*$ with the weights $W_i \equiv 1$ in the root node [0] and (C.3) for later splits, obtain the decision boundary $\partial\tilde{T}_1(z)$ based on the set of resulting decision functions F_1^m.
2: **while** the stopping criterion is not met **do**
3: Update the weight functions $f_s(x)$ and $f_s^*(x)$ by (3.10) and (3.11), set $s = s + 1$.
4: Select a sample $z^* \in Z \setminus \mathcal{P}_{s-1}^*$ so that $\mathcal{P}_s^* = \mathcal{P}_{s-1}^* \cup \{z^*\}$ minimizes (3.3). Query the label of z^*.
5: Train an HMLSVMs classifier based on $\mathcal{P}_s^*$ with the weights $W_i \equiv 1$ in the root node [0] and (C.3) for later splits, obtain the decision boundary $\partial\tilde{T}_s(z)$ based on the set of resulting decision functions F_s^m.
6: **end while**
7: **return** $\partial\tilde{T}_s(z)$

Figure 10 yields the decision tree and estimated stability region for the queueing system shown in Figure 1 (with $\Delta = 0$) based on HMLSVMs with two splits, where the host center x_h and guest center x_g for the second split were computed based on two nearest samples. To evaluate the performance of Algorithm C.1, the prediction error rate along with the design stage is shown in Figure 11. Note that the prediction error rate here is calculated based on $40 \times 40 = 1,600$ grid points and the control process $\beta(s) = 100 \times (1.3)^{-s}$, while for comparison purpose the result based on the GASP model and Algorithm B.1 is also attached. As can be seen, for such a small system the prediction error rate of our proposed method is below 2% as the number of experiments exceeds 27 and remains stable after that. On the other hand, the prediction error rate of the GASP model appears to have a relatively slow decay by using the same number of experimental trials.

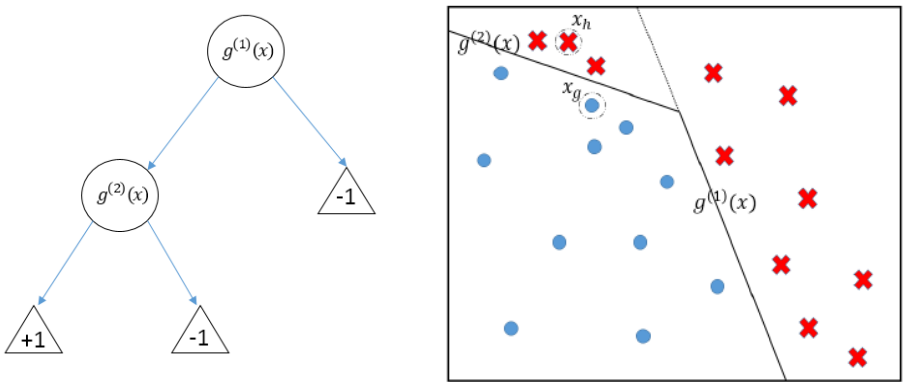

Figure 10. *Illustration of the decision tree based on HMLSVMs with two splits (left panel) and the estimated stability region for the queueing system shown in Figure 1 (with $\Delta = 0$) along with the host center x_h and guest center x_g by choosing $k = 2$ (right panel). Here the red symbol "$\times$" represents the objects with a true class label "-1", while the blue symbol "$\circ$" represents objects with a true class label "$+1$".*

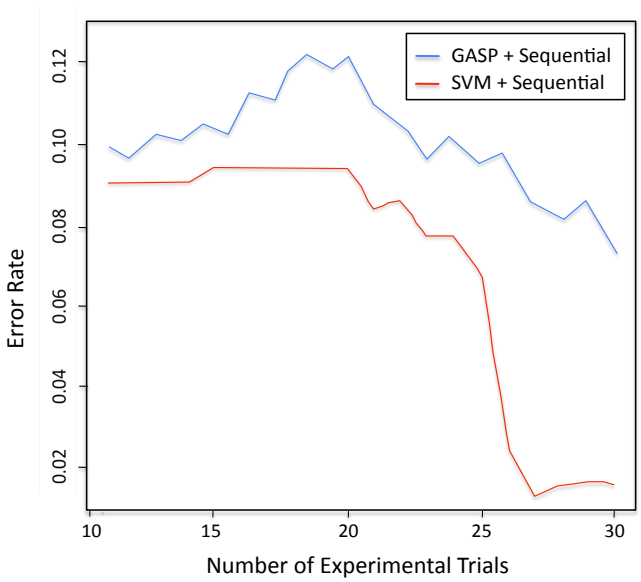

Figure 11. *The error rates for estimating the piecewise linear boundaries based on passive and active learning with SVM (red line) and the GASP model for contour estimation (blue line). Note that both methods here utilize the associated sequential sampling strategy.*

Remark: If we know a priori that the true target region is *convex*, one would not expect the decision tree to end up with a non-convex shape. To possibly avoid this issue, one can either (i) assign larger weights W_i to the training samples with label $y(x) = +1$ (i.e. points inside the true target region) when fitting the LSVM model or (ii) assign larger weights $f_s(x)$ to the input points inside the estimated target region so as to allocate the experimental trials based on WCCD. Intuitively, such strategy will push the decision boundary at each node of the tree to move "outward", thus retaining as much as possible the convex shape of the estimated

target region.

Acknowledgement. The authors would like to thank Chia-Li Lin and Ruby Chiu-Hsing Weng for their technical support and assistance with the work.

REFERENCES

[1] M. Armony and N. Bambos, *Queueing dynamics and maximal throughput scheduling in switched processing systems*, Queueing Systems, 44 (2003), pp. 209–252, https://doi.org/10.1023/A:1024714024248.

[2] J. Beck and S. Guillas, *Sequential design with mutual information for computer experiments (mice): Emulation of a tsunami model*, SIAM/ASA Journal on Uncertainty Quantification, 4 (2016), pp. 739–766, https://doi.org/10.1137/140989613.

[3] B. E. Boser, I. M. Guyon, and V. N. Vapnik, *A training algorithm for optimal margin classifiers*, in Proceedings of the 5th Annual ACM Workshop on COLT, 1992, pp. 144–152, https://doi.org/10.1145/130385.130401.

[4] M. Bramson, *Stability of Queueing Networks*, vol. 5, Springer Verlag, Berlin, 2008, pp. 169–345, https://doi.org/10.1214/08-PS137.

[5] C. C. Chang and C. J. Lin, *LIBSVM: A library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, 2 (2011), pp. 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[6] R. B. Chen, Y. W. Hsu, Y. Hung, and W. C. Wang, *Discrete particle swarm optimization for constructing uniform design on irregular regions*, Computational Statistics & Data Analysis, 72 (2014), pp. 282–297, https://doi.org/10.1016/j.csda.2013.10.015.

[7] R. B. Chen, Y. C. Hung, W. C. Wang, and S. W. Yen, *Contour estimation via two fidelity computer simulators under limited resources*, Computational Statistics, 28 (2013), pp. 1813–1834, https://doi.org/10.1007/s00180-012-0380-7.

[8] W. J. Chen, *Instability threshold and stability boundaries of rotor-bearing systems*, Journal of Engineering for Gas Turbines and Power, 118 (1996), pp. 115–121, https://doi.org/10.1115/1.2816526.

[9] S. C. Chuang and Y. C. Hung, *Uniform design over general input domains with applications to target region estimation in computer experiments*, Computational Statistics & Data Analysis, 54 (2010), pp. 219–232, https://doi.org/10.1016/j.csda.2009.08.008.

[10] C. Cortes and V. N. Vapnik, *Support vector networks*, Machine Learning, 20 (1995), pp. 273–297, https://doi.org/10.1023/A:1022627411411.

[11] H. Dette and A. Pepelyshev, *Generalized latin hypercube design for computer experiments*, Technometrics, 52 (2010), pp. 421–429, https://doi.org/10.1198/TECH.2010.09157.

[12] K. T. Fang and D. K. J. Lin, *Uniform experimental designs and their applications in industry*, vol. 22, Elsevier, Amsterdam, 2003, pp. 131–170, https://doi.org/10.1016/S0169-7161(03)22006-X.

[13] K. T. Fang, D. K. J. Lin, P. Winker, and Y. Zhang, *Uniform design: Theory and applications*, Technometrics, 42 (2000), pp. 237–248, https://doi.org/10.2307/1271079.

[14] K. T. Fang and Y. Wang, *Number-theoretic Methods in Statistics*, Chapman and Hall, London, 1994.

[15] Y. Fang and K. A. Loparo, *Stochastic stability of jump linear systems*, IEEE Transactions on Automatic Control, 47 (2002), pp. 1204–1208, https://doi.org/10.1109/TAC.2002.800674.

[16] A. I. J. Forrester, A. Sóbester, and A. J. Keane, *Multi-fidelity optimization via surrogate modelling*, in Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 463, Dec. 2007, pp. 3251–3269, https://doi.org/10.1098/rspa.2007.1900.

[17] Y. Freund, *Boosting a weak learning algorithm by majority*, Information and Computation, 121 (1995), pp. 256–285, https://doi.org/10.1006/inco.1995.1136.

[18] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, Journal of Computer and System Sciences, 55 (1997), pp. 119–139, https://doi.org/10.1006/jcss.1997.1504.

[19] R. B. Gramacy and H. H. Lee, *Bayesian treed gaussian process models with an application to computer modeling*, Journal of the American Statistical Association, 103 (2008), pp. 1119–1130, https://doi.org/10.1198/016214508000000689.

[20] M. Gu and L. Wang, *Scaled gaussian stochastic process for computer model calibration and prediction*, May 2017, https://arxiv.org/abs/1707.08215.

[21] F. J. Hickernell, *A generalized discrepancy and quadrature error bound*, Mathematics of Computation, 67 (1998), pp. 299–322, https://doi.org/10.1090/S0025-5718-98-00894-1.

[22] C. M. Huang, Y. J. Lee, D. K. J. Lin, and S. Y. Huang, *Model selection for support vector machines*

via uniform design, Computational Statistics & Data Analysis, 52 (2007), pp. 335–346, https://doi.org/10.1016/j.csda.2007.02.013.

[23] Y. C. Hung and C. C. Chang, *Dynamic scheduling for switched processing systems with substantial service-mode switching times*, Queueing Systems, 60 (2008), pp. 87–109, https://doi.org/10.1007/s11134-008-9088-3.

[24] Y. C. Hung and G. Michailidis, *Stability and control of acyclic stochastic processing networks with shared resources*, IEEE Transactions on Automatic Control, 57 (2012), pp. 489–494, https://doi.org/10.1109/TAC.2011.2164012.

[25] G. Imbens and T. Lemieux, *Regression discontinuity designs: A guide to practice*, Journal of Econometrics, 142 (2008), pp. 615–635, https://doi.org/10.1016/j.jeconom.2007.05.001.

[26] M. E. Johnson, L. M. Moore, and D. Ylvisaker, *Minimax and maximin distance designs*, Journal of Statistical Planning and Inference, 26 (1990), pp. 131–148, https://doi.org/10.1016/0378-3758(90)90122-B.

[27] V. R. Joseph, *Limit kriging*, Technometrics, 48 (2006), pp. 458–466, https://doi.org/10.1198/004017006000000011.

[28] V. R. Joseph, Y. Hung, and A. Sudjianto, *Blind kriging: a new method for developing metamodels*, Journal of Mechanical Design, 130 (2008), 0311021 (8 pages), https://doi.org/10.1115/1.2829873.

[29] A. Karatzoglou, D. Meyer, and K. Hornik, *Support vector machines in r*, Journal of Statistical Software, 15 (2006), pp. 1–28, https://doi.org/10.18637/jss.v015.i09.

[30] C. G. Kaufman, D. Bingham, S. Habib, K. Heitmann, and J. A. Frieman, *Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology*, Annals of Applied Statistics, 5 (2011), pp. 2470–2492, https://doi.org/10.1214/11-AOAS489.

[31] S. S. Keerthi and C. J. Lin, *Asymptotic behaviors of support vector machines with gaussian kernel*, Neural Computation, 15 (2003), pp. 1667–1689, https://doi.org/10.1162/089976603321891855.

[32] T. A. Khawli, U. Eppelt, and W. Schulz, *Advanced metamodeling techniques applied to multidimensional applications with piecewise response*, vol. 9432, Springer International Publishing, Switzerland, 2015, pp. 93–104, https://doi.org/10.1007/978-3-319-27926-8_9.

[33] J. Kremer, K. S. Pedersen, and C. Igel, *Active learning with support vector machines*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4 (2014), pp. 313–326, https://doi.org/10.1002/widm.1132.

[34] P. R. Kumar and S. P. Meyn, *Stability of queueing networks and scheduling policies*, IEEE Transactions on Automatic Control, 40 (1995), pp. 251–260, https://doi.org/10.1109/9.341782.

[35] D. Lee and T. Lemieux, *Regression discontinuity designs in economics*, Journal of Economic Literature, 48 (2010), pp. 281–355, https://doi.org/10.1257/jel.48.2.281.

[36] D. Lewis and J. Catlett, *Heterogeneous uncertainty sampling for supervised learning*, in Proceedings of the Eleventh International Conference on Machine Learning, July 1994, pp. 148–156, https://doi.org/10.1016/B978-1-55860-335-6.50026-X.

[37] C. H. Li, C. T. Lin, and B. C. Kuo, *An automatic method for selecting the parameter of the RBF kernel function to support vector machines*, in Proceedings of the IEEE International Conference on Geoscience and Remote Sensing Symposium, July 2010, pp. 836–839, https://doi.org/10.1109/IGARSS.2010.5649251.

[38] D. K. J. Lin, C. Sharpe, and P. Winker, *Optimized u-type designs on flexible regions*, Computational Statistics & Data Analysis, 54 (2010), pp. 1505–1515, https://doi.org/10.1016/j.csda.2010.01.032.

[39] B. McDonald, P. Ranjan, and H. Chipman, *Gpfit: An r package for fitting a gaussian process model to deterministic simulator outputs*, Journal of Statistical Software, 64 (2015), pp. 1–23, https://doi.org/10.18637/jss.v064.i12.

[40] M. D. Morris, T. J. Mitchell, and D. Ylvisaker, *Bayesian design and analysis of computer experiments: use of derivatives in surface prediction*, Technometrics, 35 (1993), pp. 243–255, https://doi.org/10.2307/1269517.

[41] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, Cambridge, 2012.

[42] J. Platt, *Probabilistic outputs for support vector machines and comparison to regularized likelihood methods*, vol. 10, MIT Press, Cambridge, 1999, pp. 61–74, https://doi.org/10.1.1.41.1639.

[43] M. T. Pratola, O. Harari, D. Bingham, and G. E. Flowers, *Design and analysis of experiments on nonconvex regions*, Technometrics, 59 (2017), pp. 36–47, https://doi.org/10.1080/00401706.2015.

1115674.
[44] P. Ranjan, D. Bingham, and G. Michailidis, *Sequential experimental design for contour estimation from complex computer codes*, Technometrics, 50 (2008), pp. 527–541, https://doi.org/10.1198/004017008000000541.
[45] J. Sacks, S. B. Schiller, and W. J. Welch, *Design for computer experiments*, Technometrics, 31 (1989), pp. 41–47, https://doi.org/10.2307/1270363.
[46] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statistical Science, 4 (1989), pp. 409–423, https://doi.org/10.1214/ss/1177012420.
[47] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*, Springer Verlag, New York, 2003.
[48] B. Schölkopf, *Support Vector Learning*, Oldenbourg Verlag, Munich, 1997.
[49] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, 2004, https://doi.org/10.1017/CBO9780511809682.
[50] M. A. L. Thathachar and P. Viswanath, *On the stability of fuzzy systems*, IEEE Transactions on Fuzzy Systems, 5 (1997), pp. 145–151, https://doi.org/10.1109/91.554461.
[51] D. Thistlewaite and D. Campbell, *Regression-discontinuity analysis: An alternative to the ex post facto experiment*, Journal of Educational Psychology, 51 (1960), pp. 309–317, https://doi.org/10.1037/h0044319.
[52] S. Tong and E. Chang, *Support vector machine active learning for image retrieval*, in Proceedings of the International Conference on Multimedia (MM), 2001, pp. 107–118, https://doi.org/10.1145/500141.500159.
[53] S. Tong and D. Koller, *Support vector machine active learning with applications to text classification*, Journal of Machine Learning Research, 2 (2001), pp. 45–66, https://doi.org/10.1162/153244302760185243.
[54] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.
[55] D. Wang, X. Zhang, M. Fan, and X. Ye, *Hierarchical mixing linear support vector machines for nonlinear classification*, Pattern Recognition, 59 (2016), pp. 255–267, https://doi.org/10.1016/j.patcog.2016.02.018.
[56] W. Wang, Z. Xu, W. Lu, and X. Zhang, *Determination of the spread parameter in the gaussian kernel for classification and regression*, Neurocomputing, 55 (2003), pp. 643–663, https://doi.org/10.1016/S0925-2312(02)00632-X.
[57] J. Wieland, R. Pasupathy, and B. W. Schmeiser, *Queueing-network stability: simulation-based checking*, in Proceedings of the Winter Simulation Conference, 2003, pp. 520–527, https://doi.org/10.1109/WSC.2003.1261464.
[58] Z. Xu, M. Dai, and D. Meng, *Fast and efficient strategies for model selection of gaussian support vector machine*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 39 (2009), pp. 1292–1307, https://doi.org/10.1109/TSMCB.2009.2015672.
[59] X. Yang, Q. Song, and Y. Wang, *A weighted support vector machine for data classification*, Journal of Pattern Recognition and Artificial Intelligence, 21 (2007), pp. 961–976, https://doi.org/10.1142/S0218001407005703.