Optimization Under Constraints by Applying an Asymmetric Entropy Measure

David V. Lindberg and Herbert K.H. Lee

Complex functions, such as the output of computer simulators, can be difficult to optimize. The task becomes even more difficult when only some of the function evaluations return real numbers and others simply fail to return a value. We combine statistical emulation, classification, sequential design, and optimization with an asymmetric entropy measure to solve the thorny problem of finding an optimum along a constraint boundary. This approach is demonstrated on simulated examples and a real problem in groundwater remediation.

Key words: Emulator, Expected Improvement, Gaussian Process, Hidden Constraints, Sequential Design.

1. INTRODUCTION

One recurring but difficult problem that arises in many contexts is constrained optimization. We want to find the minimum or maximum of a function $f(\mathbf{x}) \in \mathbb{R}$ where $\mathbf{x} \in \mathbb{R}^m$ subject to hidden constraints where the output of f is only defined for $\mathbf{x} \in \Omega \subset \mathbb{R}^m$ where Ω might be a non-convex subset. When $\mathbf{x} \notin \Omega$, then $f(\mathbf{x})$ is not defined. This is a critical distinction from the case subject to unknown constraints where the function can be evaluated outside the valid region. When f is expensive to evaluate, such as for a computer simulator, it is critical to use as few evaluations as possible that are outside the valid region, as they are a complete waste of computational efforts. Thus one needs to have a good understanding of both the function and the boundary of the valid region. Statistical models can approximate both the function and the boundary, and can then guide the optimization.

We propose herein an approach that combines statistical emulation, statistical classification, sequential design via asymmetric entropy, and optimization. We build upon a number of

David V. Lindberg is PhD student, Department of Mathematical Sciences, Norwegian University of Science and Technology, Trondheim 7491, Norway (E-mail: *davidlin@math.ntnu.no*). Herbert K.H. Lee is Professor in Applied Mathematics & Statistics, Jack Baskin School of Engineering, University of California, Santa Cruz, CA 95064 (E-mail: *herbie@soe.ucsc.edu*)

earlier works that look at pieces of this problem. Emulation is the approximation of a complex function with a statistical model, typically a Gaussian process. Santner et al. (2003) provides a good overview of emulation, including its use for optimization, and considers simple convex constraints. Jones et al. (1998) introduces expected improvement as the statistical approach to unconstrained optimization. Optimization subject to unknown constraints is a well-studied problem. Schonlau et al. (1998) offers an initial probability adjusted attempt, seeking to find the optimum at a location with high probability of being valid. This method hence tends to push the exploration into the valid region. However, in most real problems, the optimum will occur along the boundary, which increases the difficulty of the problem. If the solution is far from a boundary, finding the unconstrained solution may work just as well, and is a simpler problem. Thus we focus here on finding an optimum on the constraint boundary. Parr et al. (2012) propose a method that treats the objective and constraint functions separately, using a Pareto front approach to control the trade-off, but does not specifically look for optimum points on the boundary. Sasena et al. (2002) propose a penalty adjusted method as an alternative to the probability adjusted method, but focuses on objective functions that can be evaluated even when the constraint is not satisfied. Gramacy and Polson (2011) propose the use of entropy during active learning to hone in on the boundary, however entropy alone does not tend to fully incorporate learning behavior, and when combined with optimization tends to push the exploration too much into the invalid region.

Recognizing both the strengths and the issues with using entropy, we incorporate the idea of asymmetric entropy (Marcellin et al., 2006) to focus our exploration close to the constraint boundary but with a bias of staying inside the valid region, which improves the efficiency of the optimization and is particularly important when the function cannot be evaluated in the invalid region. Asymmetric entropy was originally developed in the context of growing decision trees when one class is rare compared to the dominant class. We believe our approach is the first to use statistical emulation to focus on an optimum along a constraint boundary for hidden constraints, and the first to incorporate asymmetric entropy in statistical emulation to focus on an optimum along a constraint boundary. This approach also leads to a convergence criterion.

Our focus here is on derivative-free optimization (Kolda et al., 2003), where function evaluations do not provide derivative information, a fairly common situation for computer experiments. We note our approach is applicable to both deterministic simulators as well as stochastic simulators or functions observed with noise, and it can deal with noisy constraint boundaries. In this paper we focus on deterministic simulators only.

Building off of this context from the literature, we next review Gaussian process emulation and classification. We then discuss the key concepts and innovations in sequential design under hidden constraints, and finally we examine simulated and real examples.

2. GAUSSIAN PROCESS MODEL FOR REGRESSION

The typical model for statistical emulation is a Gaussian process (GP) (Sacks et al., 1989; Kennedy and O'Hagan, 2001; Santner et al., 2003), which provides a good combination of nonparametric flexibility and structure induced through correlation. Suppose we have n observed data points $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)'$ where $\mathbf{x}_i \in \mathbb{R}^m$ are input vectors and $y_i = f(\mathbf{x}_i) \in \mathbb{R}$ are observed scalar outputs from an unknown deterministic output function f. We model f by a GP surrogate model:

$$Y(\mathbf{x}) = \boldsymbol{\beta}'\mathbf{h}(\mathbf{x}) + Z(\mathbf{x}) + \epsilon. \tag{2.1}$$

Here Y is the modeled output function, $\boldsymbol{\beta} \in \mathbb{R}^l$ is a regression coefficient parameter, $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_l(\mathbf{x}))'$ is a known transformation $\{h_k(\mathbf{x}) : \mathbb{R}^m \to \mathbb{R} : k = 1, \ldots, l\}$, $Z(\cdot)$ is a zero-mean GP with spatial covariance kernel $C(\cdot, \cdot)$, and $\epsilon \sim N(0, \sigma_\epsilon^2)$ is a possible white noise term.

While our approach is quite general, in this paper we use a linear regression mean with intercept, i.e., $l = m + 1$, $h_k(\mathbf{x}) = x_k$ for $k = 1, \ldots, m$ and $h_{m+1}(\mathbf{x}) = 1$, however other choices would work as well. Hence, $Y(\mathbf{x})$ is a GP with mean $\boldsymbol{\beta}'\mathbf{h}(\mathbf{x})$ and covariance kernel $C(\cdot, \cdot) + \sigma_\epsilon^2 I_n$.

If interpolation is desired, σ_ϵ^2 can be set to zero. In practice, this term can account for possible noise as well as making the algorithms more numerically stable, and can be advantageous even for deterministic simulators (Gramacy and Lee, 2012). Following the computer modeling literature, we use the nugget effect parameterization (rather than a replication error parameterization), resulting in a covariance function parameterization

$$C(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \times [K(\mathbf{x}_i, \mathbf{x}_j) + g\delta_{ij}] .$$

Here $g > 0$ is the nugget parameter, δ_{ij} is the delta function, and σ^2 is a covariance scale parameter. We choose to model the correlation structure $K(\cdot, \cdot)$ by a Gaussian correlation function, which is the standard in the computer modeling literature (Santner et al., 2003):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left\{ -\sum_{k=1}^{m} \frac{(x_{ik} - x_{jk})^2}{d_k} \right\},$$

where $\mathbf{d} = (d_1, \ldots, d_m)$ are smoothing parameters. See also Abrahamsen (1997) for an excellent review on correlation functions. The resulting $n \times n$ matrix with entries $K(\mathbf{x}_i, \mathbf{x}_j) + g\delta_{ij}$ is denoted $\mathbf{K}$.

When all parameters are known, the predictive distribution of the unknown output value $y^* = y(\mathbf{x}^*)$ for some input $\mathbf{x}^*$ is Gaussian $p(y^* | \mathbf{y}, \boldsymbol{\beta}, \sigma^2, \mathbf{K}) = N(\hat{y}(\mathbf{x}^*)\ ,\ \hat{\sigma}^2(\mathbf{x}^*))$ where

$$\hat{y}(\mathbf{x}^*) = \boldsymbol{\beta}'\mathbf{h}(\mathbf{x}^*) + \mathbf{k}_*'\mathbf{K}^{-1}(\mathbf{y} - \boldsymbol{\beta}'\mathbf{h}(\mathbf{X})) \quad , \quad \hat{\sigma}_y^2(\mathbf{x}^*) = \sigma^2 \times \left[1 + g - \mathbf{k}_*'\mathbf{K}^{-1}\mathbf{k}_*\right] \tag{2.2}$$

is the prediction mean and variance. Here

$$\mathbf{k}_* = k(\mathbf{x}^*)\ :\ k_i(\mathbf{x}^*) = K(\mathbf{x}^*, \mathbf{x}_i), \quad i = 1 \ldots, n\ . \tag{2.3}$$

We can thus represent the unknown response surface f by our statistical surrogate model $\hat{y}$ with necessary confidence bounds available through $\hat{\sigma}_y$. Hence, GP modeling provides simultaneous prediction and uncertainty quantification in an easy interpretable fashion.

We take a Bayesian approach, which allows full estimation of uncertainty, but also requires specification of priors. We use an improper uniform prior for $\boldsymbol{\beta} \propto 1$, and an inverse gamma prior for the covariance scale $\sigma^2 \sim IG\left(\alpha_\sigma/2, q_\sigma/2\right)$. The matrix $\mathbf{K}$ is defined by $\mathbf{d}$ and g for which we choose independent exponential priors, $p(d) = p(g) = Exp(\lambda)$ assuming equal priors for the smoothness parameters $p(d_k) = p(d)\ \forall\ k$. The hyperparameters $\alpha_\sigma, q_\sigma, \lambda$ are assumed known. We choose to do parameter estimation by particle learning, which is a better inferential method for sequential design than Markov chain Monte Carlo (MCMC) (Gramacy and Polson, 2011). Implementation details are in the Appendix.

3. GAUSSIAN PROCESS MODEL FOR BINARY CLASSIFICATION

Suppose we have n observed data points $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and $\mathbf{t} = (t_1, \ldots, t_n)$ where $\mathbf{x}_i \in \mathbb{R}^m$ are input vectors and $t_i = f(\mathbf{x}_i) \in \{-1, 1\}$ are observed binary categorical outputs from an unknown function f. When modeling f in the surrogate classification GP (CGP) framework, we introduce latent variables $z \in \mathbb{R}$ through a link function on t (Neal, 1998). We thus turn the classification problem for the binary output variable t into a regression problem on z and assign a GP prior to z according to Eqn (2.1). We choose a linear logistic regression link function

$$p(t|z) = \frac{\exp\{tz\}}{1 + \exp\{tz\}}\ . \tag{3.1}$$

The purpose of the latent variable z is just to allow a convenient formulation of the model, we are interested in $p(t|\mathbf{x}, z)$ through the link function and not the actual values of z. As we consider binary classification problems, we can solve for one class only. For the latent variable GP model, there is little benefit in allowing a linear mean (Gramacy and Polson, 2011), so we use a zero-mean GP prior, simplifying the computations.

In order to fit a binary CGP surrogate model, training samples from both categories $t \in \{-1, 1\}$ need to be present in the initial data set $(\mathbf{X}, \mathbf{t})$. Given the data $(\mathbf{X}, \mathbf{t})$ we do inference for a new input $\mathbf{x}^*$ in two steps, first computing

$$p(z^*|\mathbf{X}, \mathbf{t}, \mathbf{x}^*) = \int p(z^*|\mathbf{X}, \mathbf{x}^*, \mathbf{z}) \times p(\mathbf{z}|\mathbf{X}, \mathbf{t})\; d\mathbf{z}\;, \tag{3.2}$$

where $\mathbf{z} = (z_1, \ldots, z_n)$ are the unobserved latent variables which we integrate out, z_i corresponding to the observation $(\mathbf{x}_i, t_i)$. The posterior distribution of the latent variables given the data is

$$p(\mathbf{z}|\mathbf{X}, \mathbf{t}) \propto p(\mathbf{t}|\mathbf{z}) \times p(\mathbf{z}|\mathbf{X})\;, \tag{3.3}$$

where $p(\mathbf{t}|\mathbf{z})$ has conditional independent marginals given in Eqn (3.1) and $p(\mathbf{z}|\mathbf{X})$ is the GP prior. Second, we compute the probability of interest, i.e. the probability that the unknown class $f(\mathbf{x}^*)$ is 1, by

$$p(t^* = 1|\mathbf{X}, \mathbf{t}, \mathbf{x}^*) = \int p(t^* = 1|z^*) \times p(z^*|\mathbf{X}, \mathbf{t}, \mathbf{x}^*)\; dz^*\;. \tag{3.4}$$

Here, $p(z^*|\mathbf{X}, \mathbf{t}, \mathbf{x}^*)$ is the posterior predictive distribution given by Eqn (3.2). The parameters are estimated by particle learning as for the regression problem, see the Appendix.

The integrals in Eqn (3.2) and (3.4) are analytically intractable, but simple numerical integration is possible for the last integral which is in one dimension. For the first integral, we solve it by Monte Carlo integration, sampling a set of latent variables from the posterior distribution in Eqn (3.3) and passing the samples through the predictive distribution, see Neal (1998) for more details.

4. SEQUENTIAL DESIGN UNDER HIDDEN CONSTRAINTS

We consider here (unsupervised) sequential design, in particular optimization of an expensive black-box output function f subject to hidden constraints. Lee et al. (2011) provide an early attempt at this difficult problem, but do not capitalize on the fact that in most constrained optimization problems, the solution lies along the constraint boundary. Suppose we have n inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ where $\mathbf{x}_i \in \mathbb{R}^m$ are input vectors for which we define $\Omega \subset \mathbb{R}^m$ as the part of the input space where the response function $f(\mathbf{x}) \in \mathbb{R}$ is defined. Introducing binary constraint variables $(t_1, \ldots, t_n) : \; t_i \in \{-1, 1\}$, for which $t = -1$ if constraints are violated and $t = 1$ if not, the outputs are defined as

$$y_i = \begin{cases} f(\mathbf{x}_i) & , \; \mathbf{x}_i \in \Omega \; : \; t_i = 1 \\ \text{undefined} & , \; \mathbf{x}_i \notin \Omega \; : \; t_i = -1 \end{cases} .$$

We model the unknown output function f by a GP as described in Section 2, and the probability of meeting the constraints, i.e. $p(t = 1)$, by a CGP as described in Section 3. We begin by describing the expected improvement statistic for unconstrained global optimization. Next, we describe how to employ the entropy measure for classification problems. We then combine asymmetric entropy with expected improvement for an efficient algorithm for constrained optimization.

4.1 Global Optimization

We describe here a sequential updating algorithm for unconstrained optimization as introduced by Jones et al. (1998). From here onwards, we focus on minimization for concreteness, but maximization can be obtained by minimizing the negative of the response function. From an initial design $(\mathbf{X}, \mathbf{y})$ to which we fit a GP, we sequentially add a point to our design that optimizes the improvement statistic $I(\mathbf{x}) = \max\{y_{min} - Y(\mathbf{x}) \; , \; 0\}$ where

$y_{min} = \min\{y_1, \ldots, y_n\}$. This improvement statistic thus favors points with output value lower than the current minimum, hence searching for a global minimum. But since the true value of f is unknown at points that have not yet been evaluated, we use the posterior expectation from our statistical model. The expected improvement (EI) can be computed by

$$EI(\mathbf{x}) = (y_{min} - \hat{y}(\mathbf{x})) \times \Phi\left(\frac{y_{min} - \hat{y}(\mathbf{x})}{\hat{\sigma}_y(\mathbf{x})}\right) + \hat{\sigma}_y(\mathbf{x}) \times \phi\left(\frac{y_{min} - \hat{y}(\mathbf{x})}{\hat{\sigma}_y(\mathbf{x})}\right),$$

where $\hat{y}$ and $\hat{\sigma}_y$ is the prediction mean and variance given in Eqn (2.2), and $\Phi(\cdot)$ and $\phi(\cdot)$ are the normalized Gaussian cdf and pdf respectively.

4.2 Binary Classification Boundary Detection

Suppose we want to focus on detecting the true classification boundary through a sequential design algorithm. From an initial data design, we want the updates in our algorithm to add a point to our design that gives us maximal information about the boundary. For a binary classification problem, the Shannon entropy is defined as

$$S(\mathbf{x}) = -p_1(\mathbf{x}) \times \log(p_1(\mathbf{x})) - (1 - p_1(\mathbf{x})) \times \log(1 - p_1(\mathbf{x})),$$

where $p_1(\mathbf{x}) = p(t(\mathbf{x}) = 1)$. Entropy is thus non-negative with minimum entropy when $p_1 = 0 \vee 1$, and maximum entropy at the boundary where $p_1 = p_{-1} = 0.5$. Hence entropy, as a measure of uncertainty, is minimized when we are 100% sure of the class and maximized when we are most unsure. A sequential algorithm for boundary detection could thus add a point with maximum entropy at each update, i.e. a point at the predicted boundary. Sequential design by entropy is however a greedy algorithm, and it tends to select new points in areas which have already been explored (Gramacy and Polson, 2011), thus making it less efficient for either a global understanding of the boundary or for optimization.

4.3 Combining EI and Asymmetric Entropy

Consider an optimization problem subject to hidden constraints. A natural idea would be to maximize a statistic of the form $T(\mathbf{x}) = EI(\mathbf{x})^{\alpha_1} \times S(\mathbf{x})^{\alpha_2}$. Here α_1 and α_2 are specified weights with the property that for $\alpha_1 = 0$ we put all of the emphasis on trying to explore the constraint boundary, while for increasing α_1 we put more emphasis on locating the global maximum, similarly for α_2.

For output functions where the solution is expected to be located on the constraint boundary, the unconstrained global optimum is likely located outside the constrained region. Hence the EI statistic will often favor regions outside the constraints, while the entropy weighs points symmetrically across the boundary. The proposed statistic $T(\mathbf{x})$ will thus tend to favor points outside the constrained region, which will be a waste of computational resources because the function is not defined there. A natural solution would be to put more emphasis on the entropy by setting $\alpha_2 > \alpha_1$, but the entropy part would still be symmetric. We address this problem by using the asymmetric entropy measure proposed in Marcellin et al. (2006) in the rather different context of growing decision trees when one class is rare compared to the dominant class, and thus one class needs to be favored over another. For our binary constraint classification problem, the asymmetric entropy is defined as

$$S_a(\mathbf{x}) = \frac{2p_1(\mathbf{x})(1 - p_1(\mathbf{x}))}{p_1(\mathbf{x}) - 2wp_1(\mathbf{x}) + w^2} \ , \tag{4.1}$$

where w is a mode location parameter. Here maximum uncertainty (maximum entropy) is reached when $p_1 = w$ instead of $p_1 = 0.5$ for the standard Shannon entropy measure. In order to favor points inside the valid region while not pushing exploration too far from the boundary, we need a w which is larger than $1/2$, but not too much larger. Our empirical results have found $w = 2/3$ to work consistently well. The scaled asymmetric entropy is compared to the scaled Shannon entropy in Figure 4.1 for our choice of $w = 2/3$. The statistic we want to maximize in each update, with weights α_1 and α_2, is thus

$$T(\mathbf{x}) = EI(\mathbf{x})^{\alpha_1} \times S_a(\mathbf{x})^{\alpha_2}, \tag{4.2}$$

where the weights can be defined as dynamic parameters, changing throughout the algorithm.

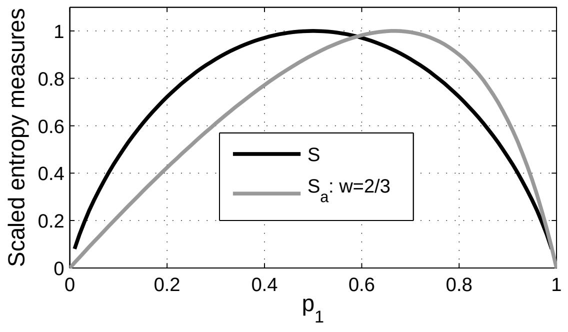

Figure 4.1: *Comparison of the scaled standard entropy S and the asymmetric entropy S_a for $w = 2/3$.*

5. TEST STUDY: CONSTRAINED OPTIMIZATION

In this empirical study, we consider optimization by sequential design of a simple constrained output function. For a m-dimensional input $\mathbf{x} = (x_1, \ldots, x_m)$, consider the function $f(\mathbf{x}) = \sum_{j=1}^m x_j/m$. We set as constraints a circular boundary centered in $(c_1, \ldots, c_m)$ with radius r, hence the valid region in higher dimensions is a hypersphere. The deterministic output is then defined as

$$f(\mathbf{x}) = \begin{cases} \sum_{j=1}^m x_j/m & , \sum_{j=1}^m (x_j - c_j)^2 \leq r^2 \ : \ t = 1 \\ \text{undefined} & , \sum_{j=1}^m (x_j - c_j)^2 > r^2 \ : \ t = -1 \end{cases} . \tag{5.1}$$

In particular, we set $c_j = 0.5 \; \forall j$ and $r = 0.5$. It is trivial to show that the true minimum keeping the constraints is for identical inputs $x_j = (1 - 1/\sqrt{m})/2 \; \forall j$. An example for $m = 2$ is displayed in Figure 5.1, with grey circular constraint boundary and true minimum as a black point.

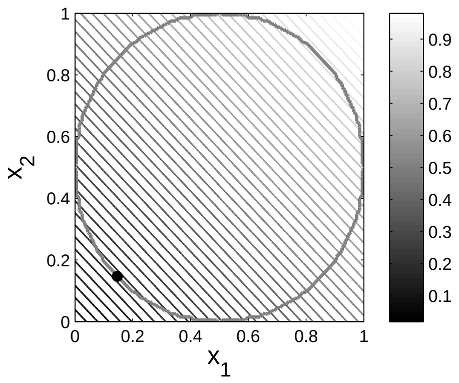

Figure 5.1: *Original function with circular constraint, true global minimum as black point.*

Our training sets will thus constitute $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $\mathbf{y} = (y_1, \ldots, y_n)$, $\mathbf{t}) = (t_1, \ldots, t_n)$ where $\mathbf{x}_i \in [0, 1]^m$ are the inputs, $y_i = f(\mathbf{x}_i)$ are the outputs defined by Eqn (5.1) only valid within the constrained region and $t_i(\mathbf{x_i}) \in \{1, -1\}$ is a binary constraint class variable. Hence we discard outputs outside the feasible region. We follow the ideas described in Section 4, declaring a GP model on $(\mathbf{X}, \mathbf{y})$ and a CGP model on $(\mathbf{X}, \mathbf{t})$ to obtain constraint probability estimates $\hat{p}(t = 1)$. As hyperparameters we set $(\alpha_\sigma, q_\sigma) = (5, 1)$ for $p(\sigma^2)$ and $\lambda = 5$ for $p(d) = p(g)$ which we believe should work well in general with unity range on the input and output. Uniform sliding windows are set as correlation parameter proposals, e.g. $q(d^*|d) = U[ld/u, ud/l]$, with $(u, l) = (4, 3)$.

5.1 Constrained Optimization for Increasing Dimension

We here consider three test studies, for dimensions $m = 2, 4, 6$ respectively. Due to the curse of dimensionality, the volume V of the valid input space rapidly decreases for higher order problems (e.g., $V = 0.0807$ for $m = 6$). Starting with an initial design of size n_0, we therefore ensure that it contains at least $m + 1$ data points in each class t; if our sampled design does not, then we resample until it does. Throughout this paper we choose designs by Latin Hypercube (LH) random samples, although other designs as maximin or minimax would apply as well. We choose to do PL updates with $N = 1000$ particles, setting the nugget to $g = 0.00001$ for numerical stability. In the sequential update results presented we compare the updates for the test statistic in Eqn (4.2) for $\alpha_1 = 1$ and for the six values $\alpha_2 = \{1, 3, 5, 7, 9, 11\}$, and we set $w = 2/3$ in the asymmetric entropy S_a in Eqn (4.1). The resulting powered asymmetric entropies gets more focused around w when α_2 increases. For all six runs, we use the same initial LH design with equal initiated particles.

For each update, we sample a LH candidate set of size 10000 and select the points that optimize the test statistics. In the plots presented, we keep track of $\log[T(\mathbf{x}^*)] : \mathbf{x}^* = \arg\max_{\mathbf{x}} \{T(\mathbf{x})\}$ for the statistics in each update, see for example the upper plot in Figure 5.2, which we expect to decrease with the updates due to the learning. In the plots of $\log[T(\mathbf{x}^*)]$, a grey marker indicates that the computed output violates the constraints, while a larger black marker indicates that they are met. We also keep track of the corresponding computed minimum output value $f_{min} = \min\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$ in each update, see the lower plot in Figure 5.2 where it is compared to the true global minimum plotted in dashed black.

When updating by PL, for each candidate point we could compute the heuristic $T(\cdot)$ for each particle and average (Gramacy and Polson, 2011). We choose instead to set the correlation matrix in each update to $\mathbf{K}(\hat{\mathbf{d}}, \hat{g})$ where we have averaged the parameters over the particles, e.g. $\hat{g} = \sum_{t=1}^{N} g^{(t)}/N$. In our experience, this works just as well, providing a very close approximation and reducing the computational time to a single evaluation of $T(\cdot)$ for each candidate point.

Results for problems of dimension $m = 2, 4, 6$ are shown in Figures 5.2, 5.3, 5.4 respectively, with minimum values computed and the ratio of the computed outputs located within the valid region presented in Table 5.1. Consider the results for $m = 2$ in Figure 5.2, we notice how $T(\mathbf{x}^*)$ tends to decrease with the updates as expected. In the run for the first order statistics, the learning algorithm seems to emphasize the expected improvement over the entropy, selecting more inputs outside the valid region, and locate the optimum slower than higher order statistics. For $m = 4$ in Figure 5.3, the first order run did not even return any valid outputs, showing clearly the need for additional emphasis on asymmetric entropy. The selected points for higher orders of α_2 however, seem to behave more as desired, concentrated in the region around the boundary optimum when this is located. We notice how the runs for higher orders of α_2 are almost identical, as the statistics often favor the same input (out of 10000 candidates). The runs for higher orders also identify the constrained minimum quite quickly. As a result of this study, as well as additional simulations, we choose $\alpha_1 = 1$ and $\alpha_2 = 5$ for use in the rest of this paper.

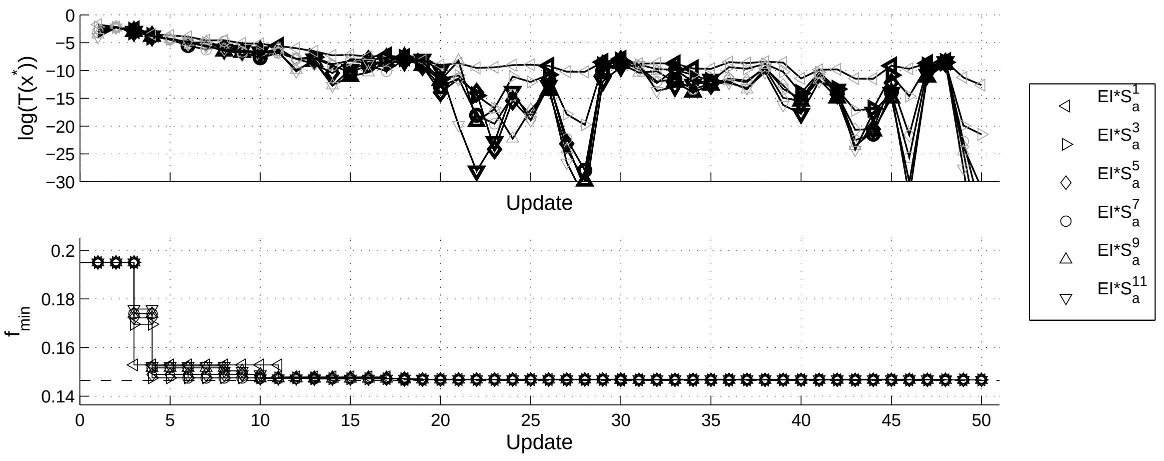

Figure 5.2: *Updates for test case $m = 2$, $n_0 = 21$. The top plot shows the objective function and the bottom plot shows the output function being minimized, with dark symbols for points meeting the constraint and light points not meeting the constraint.*

5.2 Variability and Convergence

We here consider the test statistic $T(\mathbf{x}) = EI(\mathbf{x}) \times S_a(\mathbf{x})^5$ for inputs of order $m = 4$, and we explore the variability that can be inherent in our algorithm. From the same initial design, we have run the sequential design algorithm ten times, with independent LH candidate sets

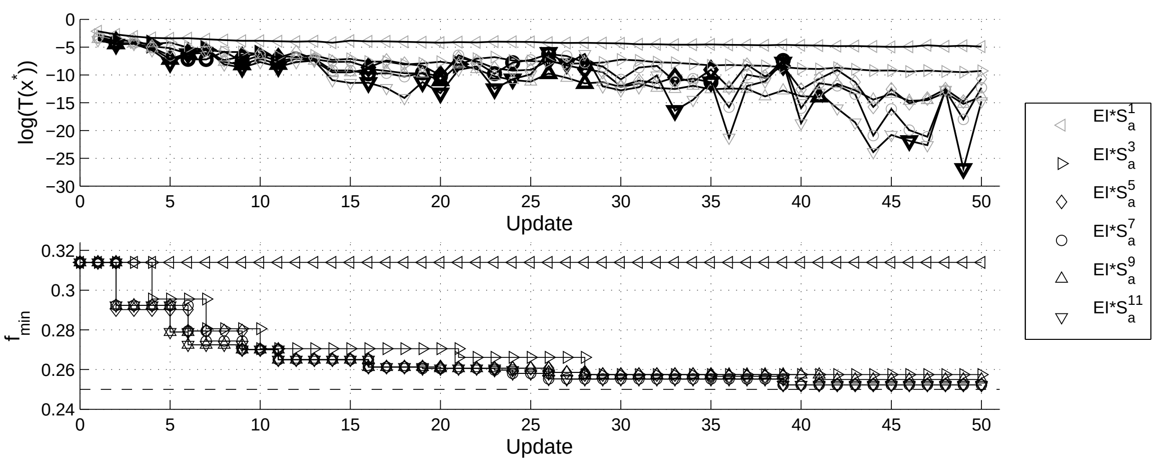

Figure 5.3: *Updates for test case* $m = 4$, $n_0 = 43$.

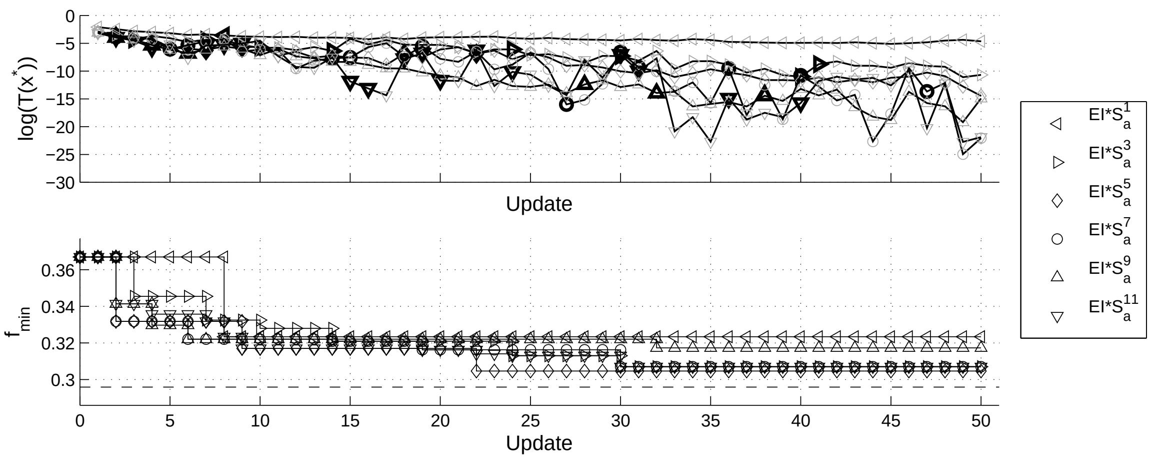

Figure 5.4: *Updates for test case* $m = 6$, $n_0 = 65$.

in each update between the runs. The ten runs are displayed in Figure 5.5, and they show that the Monte Carlo variability is relatively small, with a few larger deviations. The minimum computed during the algorithm has a smaller range, from 0.2514 to 0.2535, indicating stability of our proposed algorithm in finding the optimum. From the plots of $\log[T(\mathbf{x}^*)]$, we observe how the statistic converges towards 0, although not necessarily monotonically. Figure 5.6 displays the distribution of $T(\mathbf{x}) = EI(\mathbf{x}) \times S_a(\mathbf{x})^5$ over the candidate set for the updates $5, 10, 15, 20, 25, 30$ for a single run of dimension $m = 4$. Only values $T(\mathbf{x}) > 10^{-12}$ are included. We again observe how the distribution converges towards 0. Thus we can use

α_2	$m=2$		$m=4$		$m=6$	
	f_{min}	ratio	f_{min}	ratio	f_{min}	ratio
1	0.1467	24%	-	0%	0.3234	2%
3	0.1467	38%	0.2575	10%	0.3070	16%
5	0.1467	50%	0.2523	22%	0.3047	10%
7	0.1467	46%	0.2523	26%	0.3070	24%
9	0.1467	50%	0.2550	20%	0.3173	12%
11	0.1467	54%	0.2523	34%	0.3070	32%
True minimum	0.1464	-	0.2500	-	0.2959	-

Table 5.1: *Computed minimum values f_{min} and the ratio of valid outputs.*

T to create a convergence criterion, looking at either the full distribution or the maximum value. Typical optimization convergence criteria have a tolerance threshold δ, and declare convergence when the criterion goes below the threshold. Here we can declare convergence when $\max\{T(\mathbf{x})\} < \delta$, where $\delta = 10^{-4}$ would be a logical choice. This approach makes intuitive sense because we stop our search when we predict that there is little potential for noticeable improvement in the objective function.

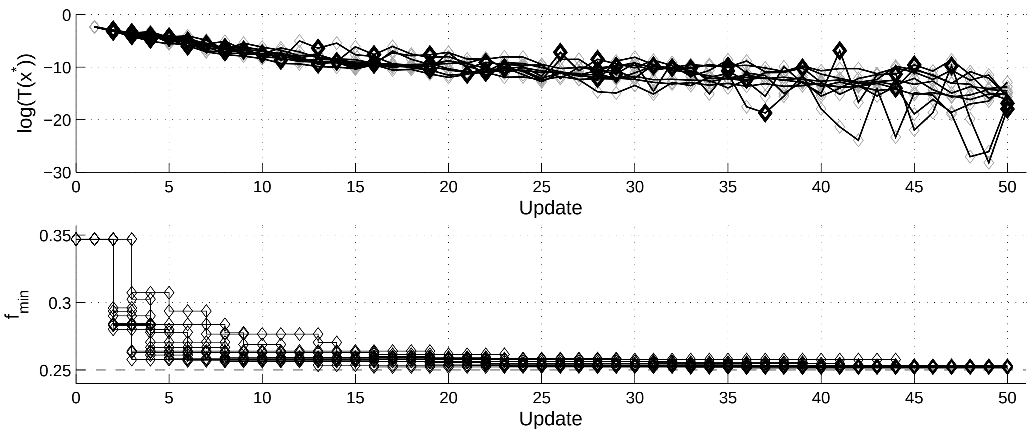

Figure 5.5: *Updates from ten runs for the MC variability test study, $m = 4$.*

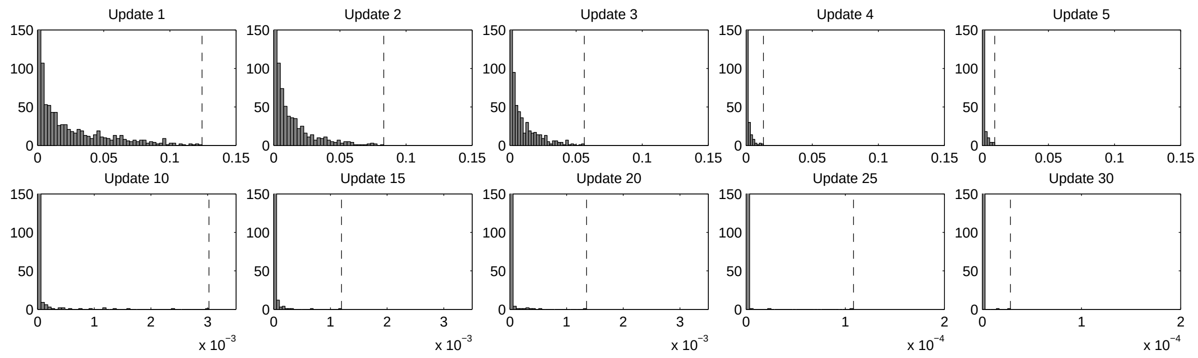

Figure 5.6: *Distribution of* $T(\mathbf{x})$ *for* $m = 4$*, displayed for the updates* $1, 2, 3, 4, 5, 10, 15, 20, 25, 30$ *with change of scale at updates* 10 *and* 25*. Dashed line is* $\max\{T(\mathbf{x})\}$*.*

5.3 Comparison study

We here compare the three alternative statistics

$$T_1(\mathbf{x}) = EI(\mathbf{x}) \times p_1(\mathbf{x}) \qquad T_2(\mathbf{x}) = EI(\mathbf{x}) \times p_1(\mathbf{x})^5 \qquad T_3(\mathbf{x}) = EI(\mathbf{x}) \times S(\mathbf{x})^5$$

to our proposed statistic denoted by $T_4(\mathbf{x}) = EI(\mathbf{x}) \times S_a(\mathbf{x})^5$. Here $T_1(\mathbf{x})$ corresponds to the original probability adjusted statistic of Schonlau et al. (1998), $T_2(\mathbf{x})$ is its powered adjustment and $T_3(\mathbf{x})$ is the symmetric entropy version of our proposed statistic. We expect the powered alternatives $T_2(\mathbf{x})$ and $T_3(\mathbf{x})$ to be more comparable to $T_4(\mathbf{x})$ due to their larger emphasis on learning the hidden constraints.

We have run the sequential algorithm 100 times with 15 updates for the problem of dimension $m = 2$, with different initial LH designs of size $n_0 = 10$. The resulting average value of f_{min} and average ratio of valid outputs for the four statistics are displayed in Figure 5.7, where the last 10 updates are enlarged in the middle plot. The final ratio of valid outputs were 21.67%, 64.20%, 30.27%, 44.53%. Observe how $T_1(\mathbf{x})$ selects the fewest valid outputs, and locates the minimum the slowest. Its powered adjustment $T_2(\mathbf{x})$ puts most emphasis on selecting valid points as expected, and f_{min} decreases fast in the early updates, but then slower as the selected inputs are likely well within the constrained region further away from the boundary. The symmetric entropy statistic $T_3(\mathbf{x})$ selects quite few valid outputs as expected, but f_{min} decreases quite well. Our proposed statistic $T_4(\mathbf{x})$ seems to hone in on the minimum the best, and has a fairly high ratio of valid outputs.

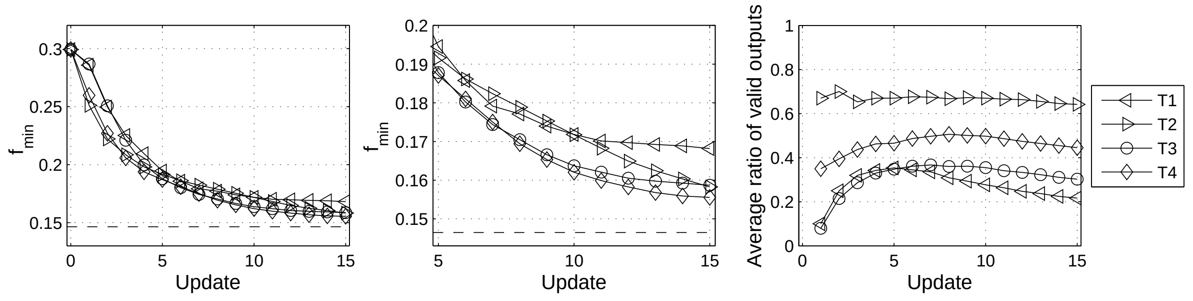

Figure 5.7: *Average f_{min} and ratio of valid outputs for the first 15 updates over 100 runs. True minimum as dashed line.*

6. CASE STUDY: HYDRAULIC CAPTURE PROBLEM

We consider here sequential design for the hydraulic capture problem from the community problems in Mayer et al. (2002). Up to four wells can be drilled to control a plume of contaminated groundwater. The objective is to find the lowest installation and operation cost configuration (location and pumping rates) of the wells while avoiding spread of the contamination. It turns out that only a single well is necessary to contain the plume, and the installation costs of additional wells dominate the cost function, so we focus on the single-well version of the problem. Thus our input vector $\mathbf{x}$ contains the location coordinates (x_1, x_2) and the pumping rate x_3. The cost objective function is computed from the hydraulic head necessary to maintain the specified pumping rate, where the relationship is highly nonlinear. Thus the simulator needs to be run to evaluate the cost. The constraint combines restrictions on the hydraulic head and the need to contain the plume. We expect the solution to be located on the constraint boundary and hence use the statistic in Eqn (4.2) with $\alpha_1 = 1$ and $\alpha_2 = 5$.

The physical setup determines bounds of $10 \leq x_1, x_2 \leq 990$ on the coordinates and $-0.0064 \leq x_3 \leq 0.0064$ on the pumping rate, where negative and positive rates indicate extraction and injection respectively. The total cost is computed by the deterministic simulator MODFLOW (McDonald and Harbaugh, 1996), `http://water.usgs.gov/ogw/modflow/`, returning the total cost as the output value if the constraints are met, and a constraint violation indicator otherwise, a classic hidden constraints problem. Each simulation run in about 1.1s on a 2.80 GHz CPU.

Initial experiments gave very low ratios (less than 2%) of valid outputs in the runs, because the constraint boundary is extremely complex. Thus we focused our efforts on the region with $235 \leq x_1 \leq 270$, $580 \leq x_2 \leq 680$, and $-0.0064 \leq x_3 \leq -0.0051$. Updates from an initial LH design of size $n_0 = 65$ are displayed in Figure 6.1, where we have run 300 PL updates, thus we run the simulator a total of 365 times. While most of the selected inputs still returned invalid outputs, the ratio of valid outputs is 4.7%, a large improvement. The minimum found over all runs had coordinates $(x_1, x_2) = (259.65, 638.03)$ and extraction rate $x_3 = -0.0053$ with total cost 22952.78. Lee et al. (2011) found this same minimum, but required several thousand simulator calls. The direction provided by asymmetric entropy appears to significantly improve the efficiency of the search.

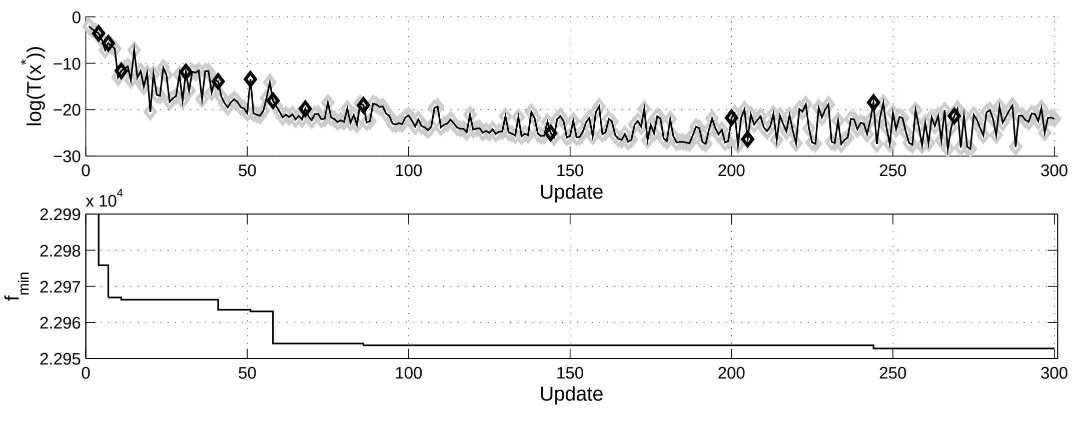

Figure 6.1: *Updates for the hydraulic capture problem. Darker points are the valid runs.*

7. CONCLUSION

Finding an optimum along a constraint boundary is a difficult problem. In this paper, we introduced an algorithm that uses statistical modeling for emulation of both the response function and the classification function, and brings in a new use for an asymmetric entropy measure. Asymmetry is the key for efficient exploration, because we need to focus our efforts where the function can be successfully evaluated, rather than wasting computational effort on unsuccessful runs.

Based on numerous experiments, we have fixed the entropy mode parameter $w = 2/3$ and the weights $\alpha_1 = 1$ and $\alpha_2 = 5$. One could look further into the optimality of these choices, including the possibility of allowing them to change dynamically during the algorithm. Future research could explore an automated dynamical approach for determining these parameters.

The functions considered in this paper are deterministic, where standard EI is applied in the optimization. An extension to noisy computer simulators should be investigated further, for example with a modified EI in the fashion of Huang et al. (2006) or Picheny et al. (2013), or by the approach of Gramacy and Lee (2011).

One potential extension would be to move beyond the standard assumption of stationarity in the GP model, for example with a treed Gaussian process emulator (Gramacy and Lee, 2008). Our approach is clearly extensible in this way, although the computational efficiencies of particle learning will no longer be available.

Another extension would be to improve optimization by use of a hybrid algorithm that incorporates a local direct method running in parallel with the statistical model, along the lines of Taddy et al. (2009) for unconstrained optimization. The statistical model guides both understanding of the constraint boundary and global exploration to locate smaller regions of interest, whereas the numerical direct method simultaneously explores more efficiently within these local regions.

ACKNOWLEDGMENTS

The work is partially funded by the URE - consortium at Department of Mathematical Sciences, NTNU, Trondheim, Norway, and by U. S. National Science Foundation grant DMS-0906720. We thank Robert Gramacy for discussion of the use of entropy in constrained optimization, and the American Institute of Mathematics (AIM) for sponsoring a workshop and small group meetings that allowed these and other discussions on constrained optimization. We would like to thank two referees, the associate editor, and the editor for helpful comments that have improved this paper.

APPENDIX: PARTICLE LEARNING

Here we describe a particle learning approach for GPs (PLGP) in regression and binary classification problems, as introduced by Gramacy and Polson (2011) with R software available (Gramacy, 2012). The main advantage of PLGP is that it updates online, whereas MCMC needs to be restarted and reiterated every time a new data point is added, which would be computationally burdensome in a sequential design or optimization setting.

For the regression problem described in Section 2, assume we have an initial set of correlation matrices $P_n = \{\mathbf{K}_n^{(h)}\}_{h=1}^N$ simulated from the posterior model given the data $(\mathbf{X}^n, \mathbf{y}^n)$ and all other parameters. The superscript n is included here to keep track of the data dimension, i.e. the number of data points. This could be obtained by storing N sampled values of $\mathbf{K}$ after burn-in from an MCMC algorithm. In the regression setting, for each particle h, we set as point estimates of $\hat{\boldsymbol{\beta}}_n^{(h)}$ and $\hat{\sigma}_n^{(h)}$ their posterior means, which only depend on $\mathbf{K}_n^{(h)}$. Hence P_n contains the sufficient information about all uncertainties given the data. When adding a new data point $(\mathbf{x}_{n+1}, y_{n+1})$ to our design, the idea is to update the particles P_n only to account for it, which will be approximated samples from the posterior distribution $p(\mathbf{K}|\mathbf{y}^{n+1})$. We proceed by the two-step update through resampling and propagation as described in Gramacy and Polson (2011). First we resample the particles from a multinomial pdf with weights $v_h \propto p\left(y_{n+1}|\mathbf{y}^n, \mathbf{K}_n^{(h)}\right)$, which is a Student-t predictive distribution. Next, the propagation step updates the particles to account for the new data by

$$\mathbf{K}_{n+1}^{(h)} = \begin{pmatrix} \mathbf{K}_n^{(h)} & k^{(h)}(\mathbf{x}_{n+1}) \\ k^{(h)'}(\mathbf{x}_{n+1}) & K_n^{(h)}(\mathbf{x}_{n+1}, \mathbf{x}_{n+1}) \end{pmatrix},$$

where $k^{(h)}(\mathbf{x}_{n+1})$ is defined in Eqn (2.3). The new output y_{n+1} is accounted for through $\hat{\boldsymbol{\beta}}_n^{(h)}$ and $\hat{\sigma}_n^{(h)}$. In practice, the predictive Student-t distribution is well approximated by a Gaussian distribution, as the degrees of freedom are typically large enough, particularly as a sequential algorithm progresses.

For the binary classification problem described in Section 3, given data $(\mathbf{X}^n, \mathbf{t}^n)$ all sufficient information is contained in $\mathbf{K}$, but we should also store the latent variables $\mathbf{z}^n = \{z_1, \ldots, z_n\}$. We thus let our initial particle set be $\{\mathbf{K}_n^{(h)}, [\mathbf{z}^n]^{(h)}\}_{h=1}^N$ which can be sampled by MCMC. When adding a new data point $(\mathbf{x}_{n+1}, t_{n+1})$ to our design, we proceed by a two-step update through resampling and propagation. In the resampling step, we compute weights which depend on the latent variable z_{n+1}:

$$v_h \propto p\left(t_{n+1} = 1 \left| [\mathbf{z}^n]^{(h)}, \mathbf{K}_n^{(h)}\right.\right) = \int p\left(t_{n+1} = 1 | z_{n+1}\right) \times p\left(z_{n+1} \left| [\mathbf{z}^n]^{(h)}, \mathbf{K}_n^{(h)}\right.\right) dz_{n+1}.$$

Here, $p\left(z_{n+1} \left| [\mathbf{z}^n]^{(h)}, \mathbf{K}_n^{(h)}\right.\right)$ is the Student-t predictive distribution. To approximate this weight integral, we simulate S samples of z_{n+1} from its predictive distribution, next pass the samples through the link likelihood function and then average. We thus resample the indices with replacement from a multinomial pdf with these approximated weights, obtaining new indices for the particles. According to Gramacy and Polson (2011), $S = 100$ should suffice. In the propagation step, we need to update the particles to account for the new data. For each resampled particle h, we first sample $z_{n+1}^{(h)}$ from its predictive distribution and set $[\mathbf{z}^{n+1}]^{(h)} = \left([\mathbf{z}^{n}]^{(h)}, z_{n+1}^{(h)}\right)$. Next, we update the latent variables in a MH-step as described in Section 3. The correlation matrices are propagated and rejuvenated as described above for the regression problem.

To avoid particle depletion in future resample steps, we choose to rejuvenate the particles (Gramacy and Polson, 2011). We then resample the updated correlation matrix particles in a MH-step from the posterior distribution, where we have to recompute all N correlation matrices. This recomputation negates some of the traditional computational advantages of PL over MCMC, but we still benefit from the online updating, eliminating the need for burning-in MCMC after each new data point is collected.

REFERENCES

Abrahamsen, P. (1997). “A review of Gaussian random fields and correlation functions.” Tech. Rep. 917, Norwegian Computing Center, Box 114 Blindern, N-0314 Oslo, Norway.

Gramacy, R. B. (2012). `plgp`*: Particle Learning of Gaussian Processes*. R package v.1.1-5.

Gramacy, R. B. and Lee, H. K. H. (2008). “Bayesian treed Gaussian process models with an application to computer modeling.” *Journal of the American Statistical Association*, 103, 1119–1130.

— (2011). “Optimization under unknown constraints.” In *Bayesian Statistics 9*, eds. J. Bernardo, S. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West, 229–256. Oxford University Press.

— (2012). "Cases for the nugget in modeling computer experiments." *Statistics and Computing*, 22, 3, 713–722.

Gramacy, R. B. and Polson, N. G. (2011). "Particle learning of Gaussian process models for sequential design and optimization." *Journal of Computational and Graphical Statistics*, 20, 1, 102–118.

Huang, D., Allen, T. T., Notz, W. I., and Zeng, N. (2006). "Global optimization of stochastic Black-Box systems via sequential Kriging meta-models." *Journal of Global Optimization*, 34, 441–466.

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

Kennedy, M. C. and O'Hagan, A. (2001). "Bayesian calibration of computer models." *Journal of the Royal Statistical Society, Series B*, 63, 425–464.

Kolda, T. G., Lewis, R. M., and Torczon, V. (2003). "Optimization by direct search: New perspectives on some classical and modern methods." *SIAM Review*, 45, 385–482.

Lee, H. K. H., Gramacy, R. B., Linkletter, C., and Gray, G. A. (2011). "Optimization subject to hidden constraints via statistical emulation." *Pacific Journal of Optimization*, 7, 3, 467–478.

Marcellin, S., Zighed, D. A., and Ritschard, G. (2006). "An asymmetric entropy measure for decision trees." In *11th Information Processing and Management of Uncertainty in knowledge-based systems (IPMU 06)*, 1292–1299.

Mayer, A. S., Kelley, C. T., and Miller, C. T. (2002). "Optimal design for problems involving flow and transport phenomena in saturated subsurface systems." *Advances in Water Resources*, 25, 1233–1256.

McDonald, M. G. and Harbaugh, A. W. (1996). "Programmer's documentation for MODFLOW-96, an update to the U. S. geological survey modular finite difference ground-water flow model." Tech. Rep. Open-File Report 96-486, U. S. Geological Survey.

Neal, R. M. (1998). "Regression and classification using Gaussian process priors." In *Bayesian Statistics 6*, ed. J. M. Bernardo et al., 476–501. Oxford University Press.

Parr, J. M., Keane, A. J., Forrester, A. I. J., and Holden, C. M. E. (2012). "Infill sampling criteria for surrogate-based optimization with constraint handling." *Engineering Optimization*, 44, 10, 1147–1166.

Picheny, V., Ginsbourger, D., Richet, Y., and Caplin, G. (2013). "Quantile-based optimization of noisy computer experiments with tunable precision." *Technometrics*, 55, 1, 2–13.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). "Design and analysis of computer experiments." *Statistical Science*, 4, 409–435.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. New York, NY: Springer-Verlag.

Sasena, M. J., Papalambros, P., and Goovaerts, P. (2002). "Exploration of metamodeling sampling criteria for constrained global optimization." *Engineering Optimization*, 34, 263–278.

Schonlau, M., Jones, D. R., and Welch, W. J. (1998). "Global versus local search in constrained optimization of computer models." In *New Developments and applications in experimental design*, no. 34 in IMS Lecture Notes - Monograph Series, 11–25.

Taddy, M., Lee, H. K. H., Gray, G. A., and Griffin, J. D. (2009). "Bayesian guided pattern search for robust local optimization." *Technometrics*, 51, 389–401.