Optimization Under Constraints by Applying an Asymmetric Entropy Measure

David 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 $\mathbf{x} \in \mathbb{R}^m$ but the output of f is only defined for $\mathbf{x} \in \Omega \subset \mathbb{R}^m$. When $\mathbf{x} \notin \Omega$, then $f(\mathbf{x})$ is not defined. 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 earlier works that look at pieces of this problem. Emulation is the approximation of a complex

David 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*)

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. Lee et al. (2011) offers an initial attempt at constrained optimization, seeking to find the optimum at a location with high probability of being valid. 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. 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. Marcellin et al. (2006) introduce an asymmetric entropy measure in the context of growing decision trees when one class is rare compared to the dominant class. We incorporate asymmetric entropy 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. We believe our approach is the first to use statistical emulation to focus on an optimum along a constraint boundary, and the first to incorporate asymmetric entropy. Many computer models result in highly complex constraint boundaries, so our methodology has the potential to be of high impact in the field of computer simulation experiments.

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. The format for the rest of this paper starts with a review of Gaussian process

emulation and classification, then discusses the key concepts and innovations in sequential design under hidden constraints, and finally examines 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. Assume we have n observed data points $(\mathbf{X}, \mathbf{y}) = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ 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 output function f. We model f by a GP surrogate model with linear regression mean

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

Here Y is the modeled output function, $\boldsymbol{\beta} \in \mathbb{R}^m$ is a regression coefficient parameter, $Z(\cdot)$ is a zero-mean GP with spatial covariance matrix $C(\cdot, \cdot)$ and $\epsilon \sim N(0, \sigma_\epsilon^2)$ is a possible white noise term. Hence, $Y(\mathbf{x})$ is a GP with mean $\boldsymbol{\beta}' f(\mathbf{x})$ and covariance matrix $C(\cdot, \cdot) + \sigma_\epsilon^2 I_m$. In our actual computations, we expand the linear regression mean to include an intercept b_0, i.e. $b_0 + \boldsymbol{\beta}'\mathbf{x} := \boldsymbol{\beta}'\mathbf{x}$ in Eqn (2.1) for notational convenience.

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, resulting in a covariance function parametrization

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

Here $g > 0$ is the nugget parameter, δ_{ij} is the delta function, and σ^2 is a covariance scale

parameter. The correlation structure K is modeled by a Gaussian correlation function, which is the standard in the computer modeling literature (Santner et al., 2003; Higdon et al., 2008)

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

where $\mathbf{d} = (d_1, \ldots, d_m)$ are smoothing parameters. See also Abrahamsen (1997) for an excellent review on correlation functions. For notational convenience, we let $K := K + gI_n$.

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

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

is the prediction mean and variance. Here

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

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.

2.1 Parameter Estimation

The parameters in our GP regression model constitute $\boldsymbol{\theta} = (\boldsymbol{\beta}, \sigma^2, \mathbf{d}, g)$. We can take a Bayesian approach, estimating the parameters by MCMC, following the model presented in Gramacy and Polson (2011). For priors, we use an improper uniform prior on $\boldsymbol{\beta} \propto 1$ and an inverse gamma prior for the covariance scale $\sigma^2 \sim IG\left(\alpha_\sigma/2, q_\sigma/2\right)$. The correlation matrix K is defined by $\mathbf{d}$ and g for which we choose independent exponential prior models, $p(d) = p(g) = Exp(\lambda)$ 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 can update β and σ^2 in Gibbs steps, for which we can compute the full conditional posterior distributions analytically. It can be shown that the full conditionals are

$$\boldsymbol{\beta}|\mathbf{y}, \sigma^2, K \sim N_m(\tilde{\boldsymbol{\beta}}\ ,\ \sigma^2 V_{\tilde{\beta}}) \quad , \quad \sigma^2|\mathbf{y}, K \sim IG\left(\frac{\alpha_\sigma + n}{2}\ ,\ \frac{q_\sigma + \psi}{2}\right) \tag{2.4}$$

where $\boldsymbol{\beta}$ is integrated out in the conditional distribution of σ^2. Here

$$\tilde{\boldsymbol{\beta}} = V_{\tilde{\beta}}\left(\mathbf{X}K^{-1}\mathbf{y}\right) \quad , \quad V_{\tilde{\beta}} = \left(\mathbf{X}K^{-1}\mathbf{X}'\right)^{-1} \quad , \quad \psi = \mathbf{y}'K^{-1}\mathbf{y} - \tilde{\boldsymbol{\beta}}' V_{\tilde{\beta}}^{-1}\tilde{\boldsymbol{\beta}}$$

For the correlation matrix posterior, analytically integrating out $\boldsymbol{\beta}$ and σ^2 gives

$$p(K|\mathbf{y}) \propto \left(\frac{|V_{\tilde{\beta}}|}{|K|}\right)^{1/2} \times \left(\frac{q_\sigma + \psi}{2}\right)^{(\alpha_\sigma + n)/2} \times\ p(K) \tag{2.5}$$

where $p(K) = p(\mathbf{d}, g)$. We can thus update K in a MH-step where Eqn (2.5) is the part of the posterior that appears in the acceptance probability. As proposals for $\mathbf{d}$ and g we choose independent uniform sliding windows.

Integrating out $\boldsymbol{\beta}$ and σ^2, it can be shown that the predictive distribution $p(y^*|\mathbf{y}, K)$ is Student-t with $n-m-1$ degrees of freedom and with mean and scale given by Eqn (2.2). We note here that the posterior means for $\boldsymbol{\beta}$ and σ^2 are $\hat{\boldsymbol{\beta}} = \tilde{\boldsymbol{\beta}}(K)$ and $\hat{\sigma}^2 = \left(q_\sigma + \psi\left(K\right)\right)/(\alpha_\sigma + n)$ by Eqn (2.4), which depend only on K, a property we will make use of in the next section when defining particles.

2.2 Particle Learning for Regression GPs

For a single data analysis, MCMC works well. However for sequential design or optimization, the model needs to be repeatedly updated after each new data point is obtained. Thus a sequential inference approach such as particle learning (PL) becomes computationally advantageous (Carvalho et al., 2010). Here we describe a PL approach for GPs (PLGP) in regression 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 in each update step.

Assume we have an initial set of covariance parameters $P_n = \{K_n^{(h)}\}_{h=1}^N$ simulated from its 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 K after burn-in from the MCMC algorithm above. For each particle h, we set as point estimates of $\hat{\boldsymbol{\beta}}_n^{(h)}$ and $\hat{\sigma}_n^{(h)}$ their posterior means (defined in the previous section), which only depend on $K_n^{(h)}$. 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 $p(K|\mathbf{y}^{n+1})$. We proceed by the two-step PL update through resampling and propagation as described in Gramacy and Polson (2011). First we resample the particles from a multinomial pdf with weights $v_h \propto p\left(y_{n+1}|\mathbf{y}^n, K_n^{(h)}\right)$, which is the Student-t predictive distribution defined at the end of the previous section. Next, the propagation step updates the particles to account for the new data by

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

where $k^{(h)}(\mathbf{x}_{n+1})$ is defined in Eqn (2.3). The new output y_{n+1} is then 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.

To avoid particle depletion in future resample steps, we choose to rejuvenate the particles (Gramacy and Polson, 2011). We then resample the updated covariance matrix particles in a MH-step from the posterior distribution in Eqn (2.5), where we have to recompute all N covariance 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.

3. GAUSSIAN PROCESS MODEL FOR BINARY CLASSIFICATION

Assume we have n observed data points $(\mathbf{X}, \mathbf{t}) = (\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)$ 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; Rasmussen and Williams, 2006). 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

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

with $p(t = -1|z) = 1 - p(t = 1|z)$. 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.

Given 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).

3.1 Sampling Latent Variables by MCMC

The integrals in Eqn (3.2) and (3.4) are analytically intractable, but simple numerical integration is possible for the last integral which is one-dimensional. A thorough overview of analytic approximations to these integrals is given by Nickisch and Rasmussen (2008). We choose to solve it by Monte Carlo integration, sampling from the latent posterior distribution in Eqn (3.3) in a MH-step and passing the samples (after some burn-in) through the predictive distributions. In the MH-steps, assuming that all parameters are known and given the previous latent sample $\mathbf{z}$, we propose new latent states $\mathbf{z}^*$ by

$$q(\mathbf{z}^*|\mathbf{z}) = N_n\left((1-\eta^2)^{1/2}\mathbf{z}\ ,\ \eta^2\sigma^2 K\right) \tag{3.5}$$

where η is some constant. Smaller η results in small changes in the proposed values, but with higher acceptance probability. This specific proposal is chosen to enhance the acceptance probability, which can be shown to depend on the link likelihood ratio only (Neal, 1998). As proposed in Neal (1998), we also choose to do multiple updates of the latent variables for each change of the parameters, choosing the last update. When estimating the GP parameters we first sample $\mathbf{z}$ in the described MH-step and given $\mathbf{z}$ we sample all parameters as described in Section 2.1. With estimated parameters, the predictive latent distribution $p(z^*|\mathbf{X}, \mathbf{x}^*, \mathbf{z})$ in Eqn (3.2) is Gaussian with mean and variance given by Eqn (2.2).

When estimating the parameters for the latent variable GP model, there is little benefit in allowing a linear mean (Gramacy and Polson, 2011). We thus use a zero-mean GP prior, simplifying the computations.

3.2 Particle Learning for Binary CGPs

We describe here the PL approach for Gaussian processes in binary classification problems similar to PLGP for regression problems as described in Section 2.2 (Gramacy and Polson, 2011). Given data $(\mathbf{X}^n, \mathbf{t}^n)$ all sufficient information is contained in 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 $\{K_n^{(h)}, [\mathbf{z}^n]^{(h)}\}_{h=1}^N$ which can be sampled by MCMC as described above. When adding a new data point $(\mathbf{x}_{n+1}, t_{n+1})$ to our design, we proceed by the two-step PL update through resampling and propagation.

In the resampling step, we now need to compute weights which depend on the latent variable z_{n+1}

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

Here, $p\left(z_{n+1} \left| [\mathbf{z}^n]^{(h)}, K_n^{(h)} \right.\right)$ 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 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 the previous section. The correlation matrices are propagated and rejuvenated as described in Section 2.2.

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 some hidden constraints, where f is not defined outside the valid region. The hidden constraints are assesed through a binary classification problem. Lee et al. (2011) provides an early attempt at this difficult problem, but does not capitalize on the fact that in most constrained optimization problems, the solution lies along the constraint boundary. (When the solution is well within the interior, the problem is essentially equivalent to an easier unconstrained problem.) Here we consider problems where the optimum is expected to be located on the constraint boundary. Hence we need our algorithm to both explore the output function over the input space within the constraints, as well as to efficiently determine the constraint boundary. Because function calls outside the valid region do not provide any information about the function, we have a strong bias for sampling along or inside the boundary. This is a critical distinction from the case where the function can be evaluated outside the valid region (Gramacy and Lee, 2011).

Assume we have n observed data points $(\mathbf{X}, \mathbf{y}) = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ 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 constraints are not violated, and $y_i = f(\mathbf{x}_i) \in \mathbb{R}$ are observed scalar outputs only computable within the constrained region. Introducing corresponding binary constraint variables $\mathbf{t} = (t_1, \ldots, t_n) : t_i \in \{-1, 1\}$, for which $t = -1$ if constraints are violated and $t = 1$ if not, the output is 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 an expected improvement statistic for unconstrained global optimization by sequential design. Next, we describe how to employ the entropy measure in sequential design for classification problems. An appealing approach is then to combine the probability of meeting the constraints through the entropy measure with the expected improvement statistic of interest.

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 $I(\mathbf{x}) = \max_x \{y_{min} - Y(\mathbf{x}) \ , \ 0\}$ 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 give 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 $p = w$ instead of $p = 0.5$ for the standard Shannon entropy measure. The scaled asymmetric entropy is compared to the scaled Shannon entropy in Figure 4.1 for $w = 2/3$, which we have found to work well for our problem, as it sufficiently favors points inside the valid region while not pushing exploration too far from the boundary. Thus the statistic we want to maximize in each update is

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

where the weights α_1 and α_2 can be defined as dynamic parameters, changing throughout the algorithm.

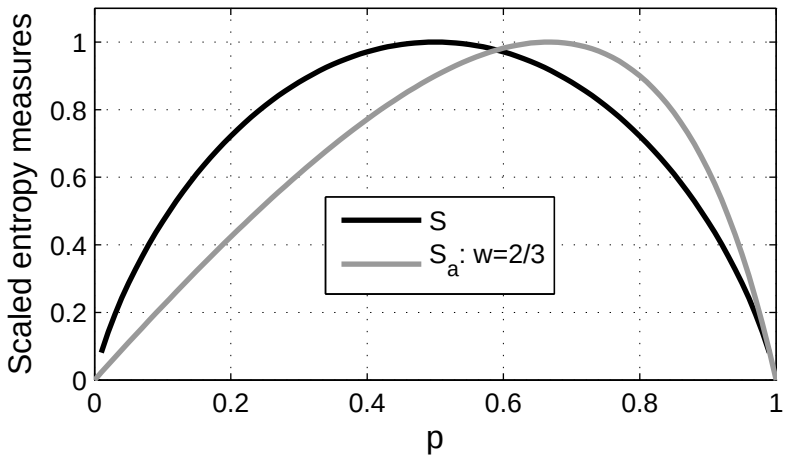

Figure 4.1: *Comparison of the scaled standard entropy S and the asymmetric entropy S_a for $w = 2/3$.*

5. TEST STUDY: CONSTRAINED OPTIMIZATION

In this empirical study, we consider optimization by sequential design of a simple output function assuming that the output is subject to hidden constraints. For a m-dimensional input $\mathbf{x} = (x_1, \ldots, x_m)$, consider the function $f(\mathbf{x}) = \sum_{j=1}^{m} x_j$ (see Section 6 for a real world example of this function). 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 output is then defined as

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

In particular, we set the circular constraints with center $c_j = 0.5 \ \forall j$ and radius $r = 0.5$. 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.

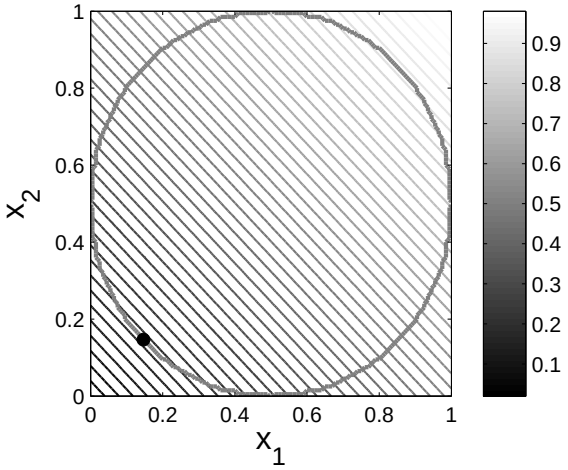

Figure 5.1: *Original function with circular constraint, true global minimum as black point.*

Our training sets will thus constitute $(\mathbf{X}, \mathbf{y}, \mathbf{t}) = (\mathbf{x}_1, y_1, t_1), \ldots, (\mathbf{x}_n, y_n, t_n)$ where $\mathbf{x}_i \in [0, 1]^m$ are the inputs, $y_i = f(\mathbf{x}_i) \in [0, 1]$ are the outputs defined by Eqn (5.1) and $t_i \in \{1, -1\}$

is a binary constraint class varible. 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 $p(d) = p(g)$. Uniform sliding windows are set as covariance parameter proposals, e.g. $q(d^*|d) = U[ld/u, ud/l]$, with $(u, l) = (4, 3)$. When sampling the latent variables for classification, we set $\eta = 0.05$ in the Gaussian proposal in Eqn (3.5).

We 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 Latin Hybercube (LH) design, we therefore ensure that it contains at least $m+1$ data points in each class t, if not we redraw. 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 are displayed in Figure 5.2, showing how the entropy gets more focused around w when α_2 increases. For all six runs, we use the same initial LH design with equal initiated particles.

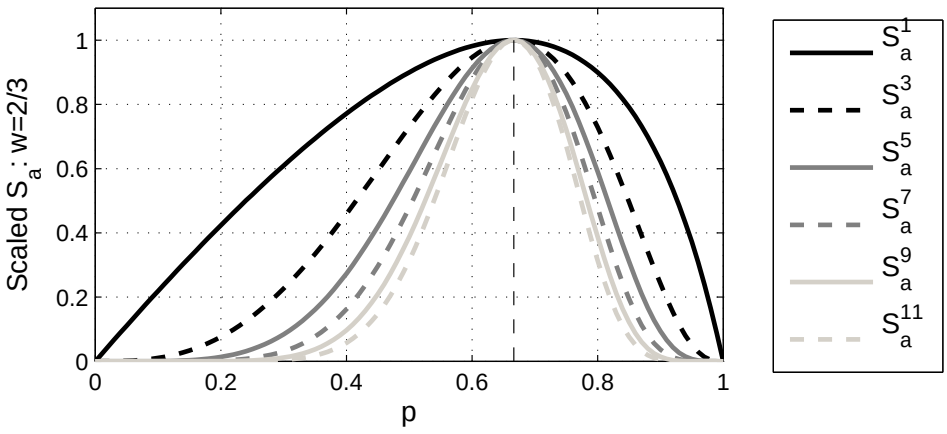

Figure 5.2: *Asymmetric entropy measure $S_a^{\alpha_2}$ for increasing orders α_2 with $w = 2/3$.*

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 5.3, which we expect to decrease with the updates due to the learning. We also keep track of the corresponding computed output value $f(\mathbf{x}^*)$ in each update, see the lower plot in Figure 5.3 where it is compared to the true global minimum plotted in dashed black. In all plots, a grey marker indicates that the computed output violates the constraints, while a larger black marker indicates that they are met.

When updating by PL, for each candidate point we could compute compute the heuristic $T(\cdot)$ for each particle and average (Gramacy and Polson, 2011). We choose instead to set the covariance matrix in each update to $K(\hat{\mathbf{d}}, \hat{g})$ 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.

5.1 Results

Results for problems of dimension $m = 2, 4, 6$ are shown in Figures 5.3, 5.4, 5.5 respectively, with minimum values computed presented in Table 5.1. Consider the results for $m = 2$ in Figure 5.3, 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 and locate the optimum slower than higher order statistics. For $m = 4$ in Figure 5.4, 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 chose $\alpha_1 = 1$ and $\alpha_2 = 5$ for use in the rest of this paper.

α_2	$m = 2$	$m = 4$	$m = 6$
1	0.1467	-	0.3234
3	0.1467	0.2575	0.3070
5	0.1467	0.2523	0.3047
7	0.1467	0.2523	0.3070
9	0.1467	0.2550	0.3173
11	0.1467	0.2523	0.3070
True value	0.1464	0.2500	0.2959

Table 5.1: *Mimimum values computed during the sequential algorithm.*

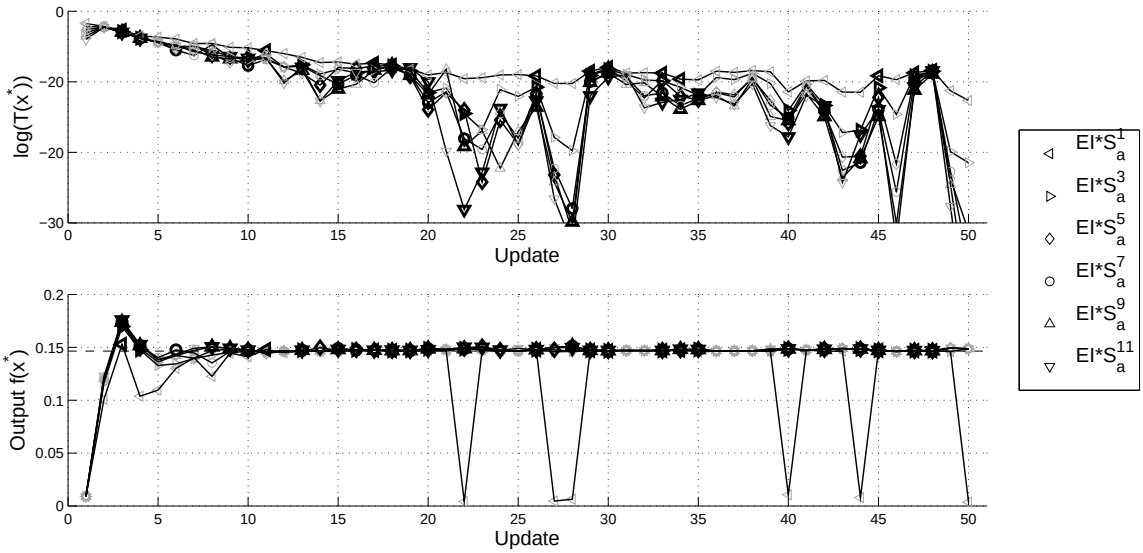

Figure 5.3: *Updates for test case $m = 2$, $n_0 = 21$. The top plot shows the objective function and the bottom plot shows the output function being minimized, with dark symbols for points meeting the constraint and light points not meeting the constraint.*

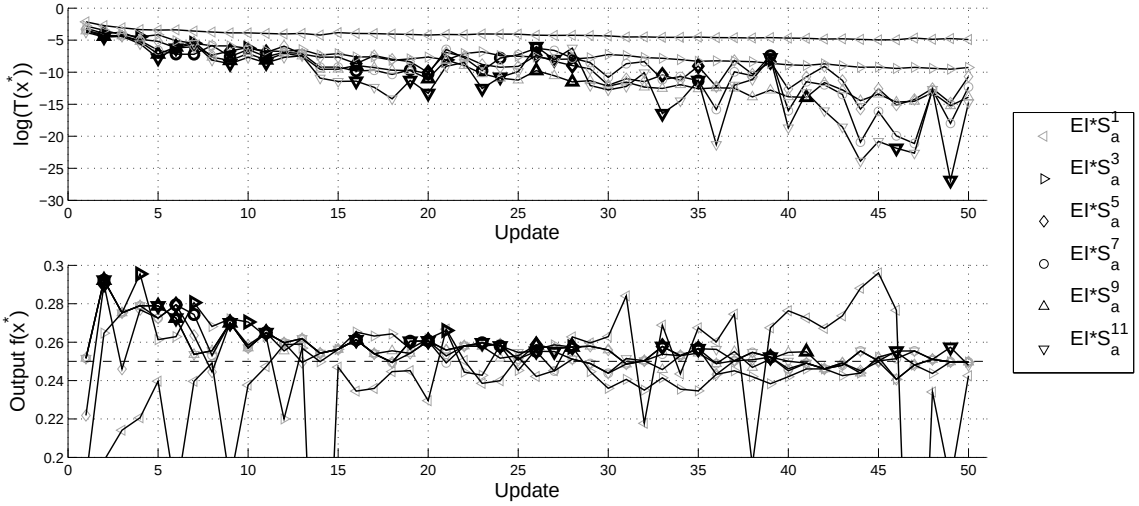

Figure 5.4: *Updates for test case* $m = 4$. $n_0 = 43$

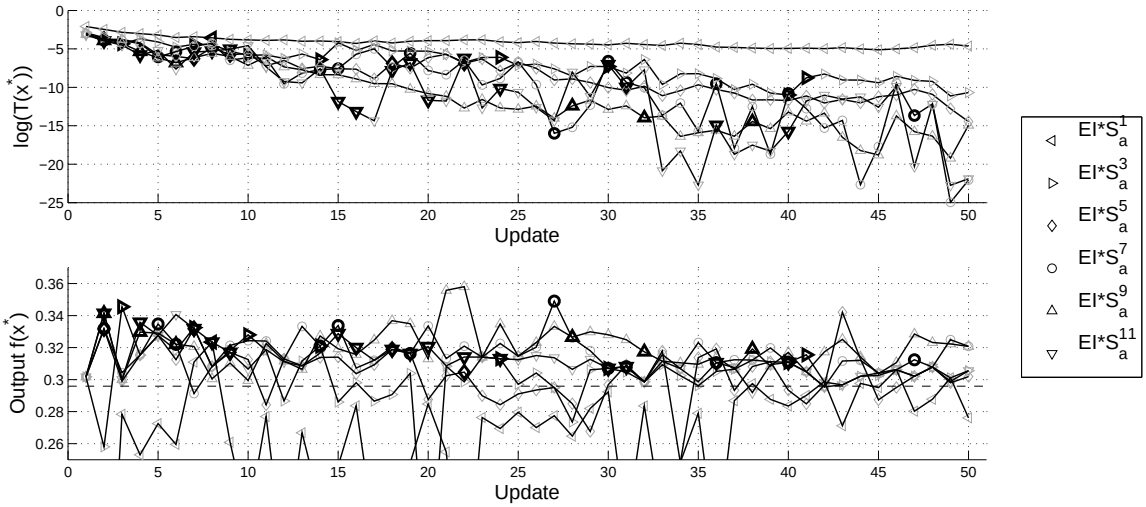

Figure 5.5: *Updates for test case* $m = 6$. $n_0 = 65$

5.2 Variability Study

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 in each update between the runs. The ten runs are displayed in Figure 5.6, 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.

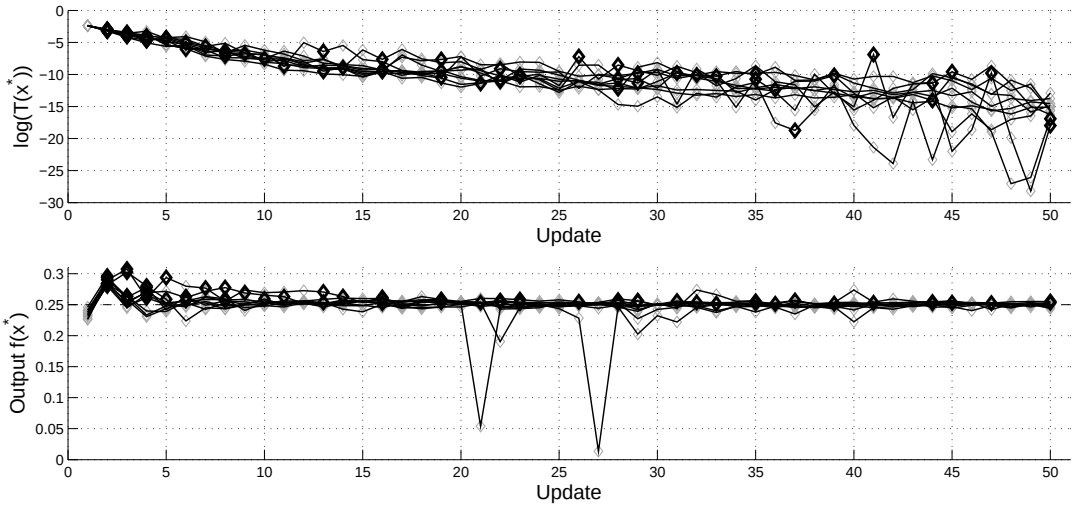

Figure 5.6: *Updates from ten runs for the MC variability test study, $m = 4$.*

6. CASE STUDY: PUMP-AND-TREAT PROBLEM

We consider here sequential design for the pump-and-treat problem case study presented in Matott et al. (2011) based on the Lockwood Solvent groundwater plume site located near Billings, Montana (Tetra Tech Inc., 2003). The area under study is presented in Figure 6.1. We have two plumes A and B containing chlorinated solvents, with two and four pumping wells respectively which we name wells A1 and A2 and B1 through B4. The objective is to find the lowest cost pumping rates of the six wells, subject to the hidden constraints such that the plumes of contamination do not reach the Yellowstone river.

The output function $f(\mathbf{x})$ to be minimized is here the total cost in USD of operating the wells, where $\mathbf{x} = (x_1, \ldots, x_6) := (Q_{A1}, Q_{A2}, Q_{B1}, Q_{B2}, Q_{B3}, Q_{B4})$ are the pumping rates in the six wells. We assume a constant cost per water pumped of 35USD/m^3 and set simple bounds $0 \leq x_i \leq 20000$ $\forall i$. The total cost $f(\mathbf{x})$ is computed by the simulator described

in Matott et al. (2011) which we will treat as a Black-Box function, although under certain conditions, it turns out to be simply the sum of the inputs. The simulator also returns constraint variables $t_1(\mathbf{x})$ and $t_2(\mathbf{x})$ for the amount of pollution escaping from plumes A and B repectively. A contraint variable of zero corresponds to meeting that constraint. Hence we define a common constraint indicator $t(\mathbf{x}) = 1$ when $t_1(\mathbf{x}) = t_2(\mathbf{x}) = 0$ and $t(\mathbf{x}) = -1$ otherwise. Each simulation run in about 2.5s, which is fairly slow when we need to explore the whole 6-dimensional input space. Higher precision simulations can take even longer. Hence, the need to model the output by a fast computational surrogate GP model.

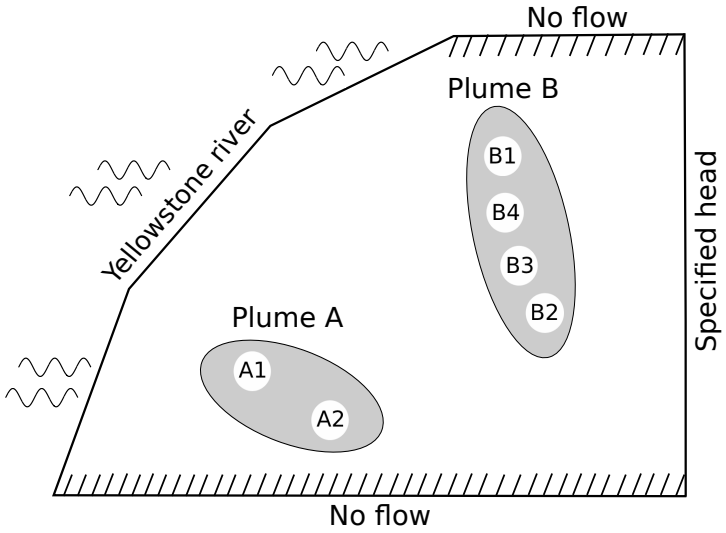

Figure 6.1: *Simplified map of the Lockwood site.*

6.1 Two-Dimensional Single Plume Problem

As an initial simplification, we first consider wells A1 and A2 only, fixing the pumping rates for each well in plume B at 10,000. We expect the solution to be located on the constraint boundary and hence choose to optimize the statistic in Eqn (4.2) with $\alpha_1 = 1$ and $\alpha_2 = 5$. Updates from an initial LH design of size $n = 21$ is displayed in Figure 6.2. The optimal pumping rate found in the runs was $(Q_{A1}, Q_{A2}) = (179, 5269)$ matching the results in Matott et al. (2011), where optimized low cost was achieved with low rate for well

A1 and higher rate for well A2. A graphical display of the first four updates is shown in Figure 6.3, where the grey line indicates the predicted boundary and the white point is the input maximizing the heuristic objective function $T(\cdot)$. Notice how the algorithm selects points on or slightly within the predicted boundary in the region of interest, as desired.

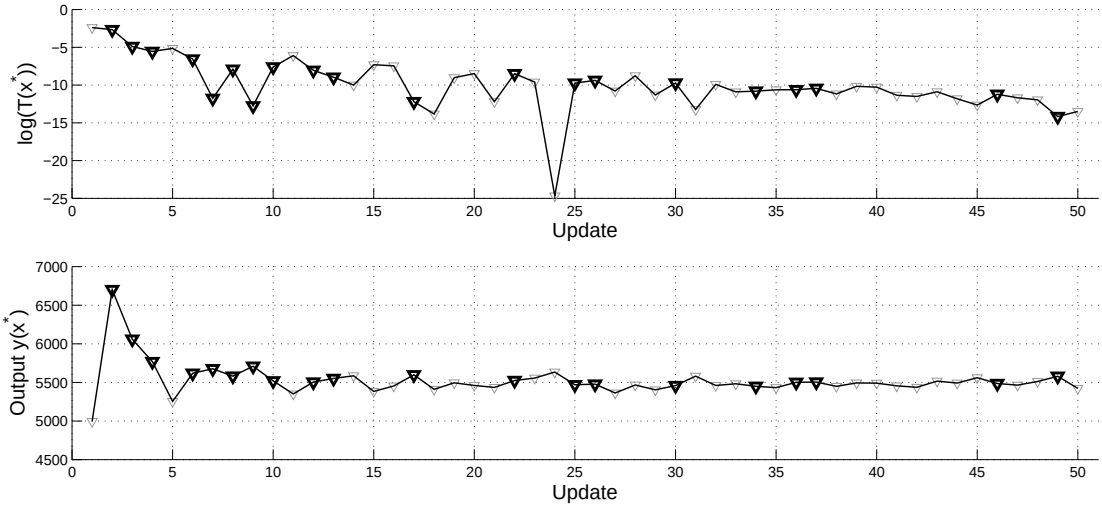

Figure 6.2: *Updates for the pump-and-treat problem for the wells in plume A. The top plot shows the objective function and the bottom plot shows the output function being minimized, with dark symbols for points meeting the constraint and light points not meeting the constraint.*

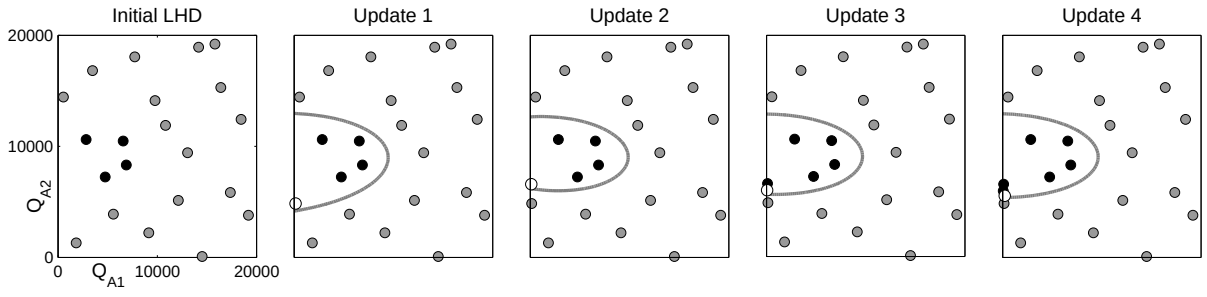

Figure 6.3: *Sequential PL updates for pumping rates in plume A. Dark points meet the constraint, light points do not, the gray line is the predicted boundary, and the open circle shows the point of highest objective $T(\cdot)$.*

6.2 Full Six-Dimensional Problem

We now consider the full problem with six wells, with the weights in Eqn (4.2) again set to $\alpha_1 = 1,\ \alpha_2 = 5$. The hidden constraints for the 6D problem are known to be a lot more complex than the seemingly elliptical boundary for the 2D problem. Updates from an initial LH design of size $n = 65$ is displayed in Figure 6.4, where we have run 300 PL updates, thus we run the simulator a total of 365 times. The minimum found over all runs is $(Q_{A1}, Q_{A2}, Q_{B1}, Q_{B2}, Q_{B3}, Q_{B4}) = (54, 6617, 13892, 2385, 1845, 818)$ with total cost $25,612$, which noticeably is the last valid output obtained (update 186). In comparison, the best optimzer in Matott et al. (2011) obtained a minimum value of $23,714$ in 993 simulator runs. Of the algorithms completing in 500 runs or less, the best value found was 27,137. Hence our algorithm proposed here could be preferred with respect to the number of function evaluations, which is particularly important for highly CPU-demanding computer simulators.

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.

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.

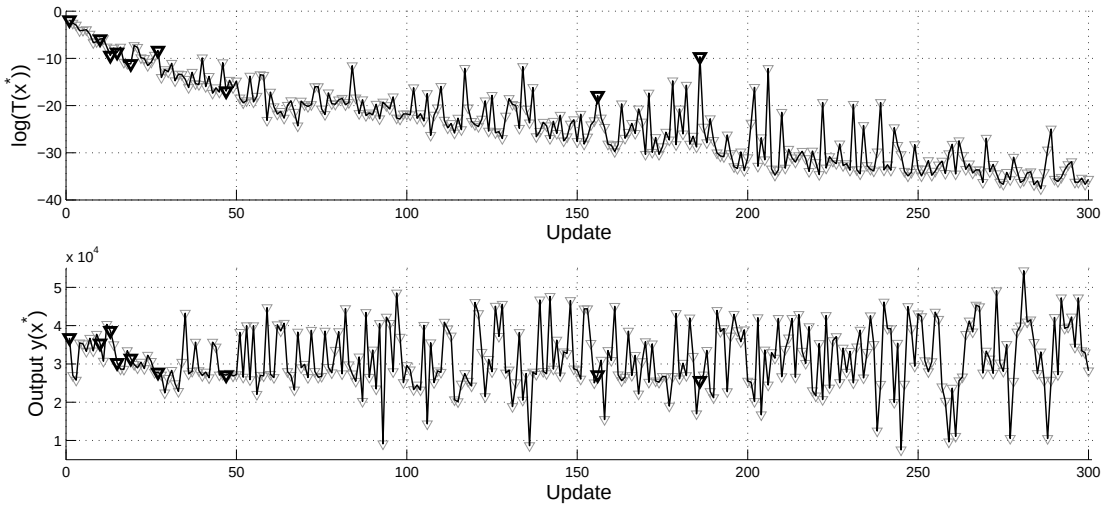

Figure 6.4: *Updates for the pump-and-treat problem for all six wells.*

Another potential extension would be to improve optimization by use of a hybrid algorithm that incorporates a local direct numerical method, along the lines of Taddy et al. (2009) for unconstrained optimization. The local algorithm is more efficient at exploring locally, and the statistical methods guide both global exploration and understanding of the constraint boundary.

ACKNOWLEDGMENTS

The work was funded by the URE - consortium at Department of Mathematical Sciences, NTNU, Trondheim, Norway, and by U. S. National Science Foundation grant DMS-0906720.

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.

Carvalho, C. M., Johannes, M., Lopes, H. F., and Polson, N. (2010). "Particle learning and smoothing." *Statistical Science*, 25, 1, 88–106.

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

Gramacy, R. B. (2012). `plgp`*: Particle Learning of Gaussian Processes*. R package version 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.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008). "Computer Model Calibration Using High-Dimensional Output." *Journal of the American Statistical Association*, 103, 570–583.

Jones, D., 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., Lewis, R., and Torczon, V. (2003). "Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods." *SIAM Review*, 45, 385–482.

Lee, H., Gramacy, R., Linkletter, C., and Gray, G. (2011). "Optimization Subject to Hidden Constraints via Statistical Emulation." *Pacific Journal of Optimization*, 7, 3, 467–478.

Marcellin, S., Zighed, D., 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.

Matott, L., Leung, K., and Sim, J. (2011). "Application of MATLAB and Python Optimizers to Two Case-Studies Involving Groundwater Flow and Contaminant Transport Modeling." *Computers & Geosciences*, 37, 11, 1894–1899.

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.

Nickisch, H. and Rasmussen, C. (2008). "Approximations for Binary Gaussian Process Classification." *Journal of Machine Learning Reasearch*, 9, 10, 2035–2078.

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

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.

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.

Tetra Tech Inc. (2003). "Remedial Investigation Report Lockwood Solvent Groundwater Plume Site." Tech. rep., Montana Department of Environmental Quality, P.O.Box 200901 Helena, Montana 59620.