Optimization Subject to Hidden Constraints via Statistical Emulation

Herbert K.H. Lee
Applied Math & Statistics
University of California, Santa Cruz
herbie@ams.ucsc.edu

Robert B. Gramacy
Statistical Laboratory
University of Cambridge
bobby@statslab.cam.ac.uk

Crystal Linkletter
Center for Statistical Sciences
Brown University
cdlinkle@stat.brown.edu

Genetha A. Gray
Predictive Simulation R&D
Sandia National Laboratories
gagray@sandia.gov

March 24, 2010

Abstract

We present new methodology for constrained optimization based on building a combination of models, one for the objective function and one for the constraint region. We use a treed Gaussian process as a statistical emulator for the complex objective function, and a random forest to model the probability of meeting the constraints. By combining these models, we can guide the optimization search to promising areas in terms of both the objective function and the constraint. This approach avoids the problem of becoming stuck in a local mode, as well as being able to deal with unconnected viable regions. We demonstrate our methodology on a simulated problem and an example from hydrology.

Key words: constrained optimization, surrogate model, Gaussian process, sequential design, expected improvement

Mathematics Subject Classification: 62M30 62C10 49M30 90C56

1 Introduction

Optimization of complex functions, where many separated local optima abound, is a challenging problem with a long history. Further complication arises when the function can only be successfully evaluated over a subset of the input space, and this region is not known in advance. In this case, it might also be of interest to learn about the feasibility region itself. Our motivating example comes from hydrology, where we are using a computer simulation for designing placement of a set of wells. We want to minimize the cost of the setup. However, for most runs of the simulator, the response is that there has been a constraint violation, and no further information about cost is provided. Thus we need to simultaneously learn about the constraint region and solve the minimization.

We note that constraints come in several kinds. When the constraints are known restrictions on the valid set of inputs, it is reasonably straightforward to focus the optimization on only the valid set. However when the constraints depend on the outputs, the problem becomes considerably more difficult (for example, Finkel and Kelley, 2009, and the references therein). We further distinguish between physical or hidden constraints, where it is not possible to get an output value, and policy constraints, where an output value is obtained and then deemed not valid. In this paper we focus on the former case, where the computer simulation fails to return a value for some runs. In that case, no information about the objective function can be learned from a run outside the valid set, and one wants to learn the boundaries of valid regions to avoid wasting runs on input settings outside that set.

Our approach is to use statistical emulation, where we build a statistical model as an approximation to the complex simulator, and use this model to guide the optimization. This model has two parts: one to predict the response surface when the simulator gives a valid response, and one to predict whether or not the simulator will return a valid response at all. We use treed Gaussian processes (Gramacy and Lee, 2008) for response surface prediction, and random forests (Breiman, 2001) for constraint violation prediction. The

former is becoming widely recognized as a flexible surrogate model, or *emulator*, for the design and analysis of computer experiments. The latter is a powerful classification algorithm from the machine learning literature.

In Section 2 we review the statistical emulation techniques. In Section 3 we apply the statistical predictions to optimization using expected improvement. In Section 4 we then provide an illustrative example before finally applying our methods to the motivating example in hydrology in Section 5.

2 Statistical Emulation

The use of stochastic or statistical models to approximate a complex function, such as the output of a computer simulator, is now well established in the literature (Sacks et al., 1989; Kennedy and O'Hagan, 2000; Santner et al., 2003; Fang et al., 2006). In particular, the standard model used for emulation is a Gaussian process (GP). The functional response is treated as a random variable $Z(\mathbf{x})$ that depends on the input vector $\mathbf{x}$ such that the response varies smoothly. The degree of smoothness is determined by the covariance structure of the GP. GPs have the property that any finite set of locations has a joint multivariate normal (Gaussian) distribution, thus the process is completely determined by its mean and covariance function. If we have observations at a set of locations $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, then

$$(Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_n)) \sim \mathrm{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{1}$$

where $\boldsymbol{\mu} = (\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_n))$ is a mean function and $\boldsymbol{\Sigma}$ is the variance-covariance matrix. The mean function is typically taken as constant, linear, or a low-order polynomial. The covariance is typically given a simple parameterization such that correlation decreases with distance in the input space. More details on GPs are available in references such as Cressie (1993) and Stein (1999). We take a fully Bayesian approach, allowing for estimation of

uncertainty, which is critical when trying to determine the probability that an unsampled location will be an improvement over the current known optimum.

In some cases standard GP models are adequate. But in others, their limitations, such as strong assumptions of stationarity and poor computational scaling, can be problematic. To reduce these problems, we instead use treed Gaussian process (TGP) models, wherein the input space is partitioned using a recursive tree structure and independent GP models are fit within each partition. More details on TGP are available in Gramacy and Lee (2008). Such models are a natural extension of standard GP models, and combine partitioning ideas with Bayesian methods to produce smooth fitted functions (Chipman et al., 2002). The partitions can be fit simultaneously with the parameters of the embedded GP models using reversible jump Markov chain Monte Carlo (Green, 1995). There is software available in the form of a `tgp` library for the open source statistical package R (see `http://www.cran.r-project.org/src/contrib/Descriptions/tgp.html`).

In addition to emulating the simulator, we also need to predict whether the simulator will return a valid response or whether there will be a constraint violation. For this task we turn to a machine learning tool, random forests. A random forest is built from a collection of tree models. Each tree model partitions the space into disjoint regions and gives a single class probability for all points in that region. In our application, the classes are (1) "returns a valid response", or (2) "not". A large number of CART trees (Breiman et al., 1984) are randomly instantiated, and the random forest classifier is created by taking weighted votes across all of the trees in the collection. It has been shown that this process converges asymptotically as the number of trees grows large, but that in practice a relatively small number of trees provides a good approximation. More details on random forests are available in Breiman (2001). We use the R implementation from the `randomForest` library (Liaw and Wiener, 2002).

3 Optimization

Our approach toward optimization is that of expected improvement (EI, Jones et al., 1998; Taddy et al., 2009). We use the statistical emulator, or surrogate model, to guide the search for a new optimum, sequentially investigating locations with the largest probability of being an improvement. To be concrete, we focus on the case of minimization (but it can clearly be applied to maximization as well). Herein we assume that the functional response is deterministic, but the method can be generalized for stochastic response functions as well. After N runs, denote the current minimum value observed as $f_{\min} = \min\{z_1, \ldots, z_N\}$, where $z_i = Z(\mathbf{x}_i)$. Define the improvement statistic at a proposed input location $\mathbf{x}$ by

$$I(\mathbf{x}) = \max\{f_{\min} - Z(\mathbf{x}), 0\}. \tag{2}$$

Since the $\mathbf{x}$ of interest are previously unobserved locations, $Z(\mathbf{x})$ is unknown and we represent its distribution using the posterior predictive distribution from the treed Gaussian process emulator. A simple algorithm for optimization sequentially selects the points $\mathbf{x}'$ that maximize the expected improvement (EI)

$$\mathbf{x}' = \arg\max_{\mathbf{x} \in \mathcal{X}} \mathbb{E}\{I(\mathbf{x})\}. \tag{3}$$

Conditional on a particular TGP parameterization, the expectation calculation is available in closed form as a function of the mean and variance of the (Gaussian) posterior predictive distribution of $Z(\mathbf{x})$. For the particular parameterization, the EI provides a combined measure of how promising a candidate point is, balancing points where the emulator predicts a minimal response with those points where the uncertainty in the emulator is large. For the latter locations, there is a probability of a minimal response even though the mean prediction may not be as small due to the emulator uncertainty. It is trivial to average over the posterior EI by sampling from the TGP posterior and posterior predictive distributions and

taking an average of the corresponding EI calculations that arise from each sample. Our implementation takes advantage of the functionality of the `tgp` R library, which provides an argument for evaluating the expected improvement, essentially automating the EI calculation as just described via Markov chain Monte Carlo (MCMC). MCMC is the standard algorithm for fitting Bayesian statistical models, simulating realizations from the posterior distribution via a Markov Chain to allow for Monte Carlo estimation of any desired posterior quantities (Gelman et al. (1995)), which meshes well with our EI approach.

In the case where there are no constraints, one can just iteratively evaluate the next point that maximizes the EI, and then update the statistical emulator and the improvement function. There are several algorithms that may be used to aid in the search of the $\mathbf{x}$ input that maximizes the EI. In the case of a simple (non-treed) GP surrogate model and maximum likelihood inference, there is a branch and bound algorithm that may be used (Jones et al., 1998). This leads to the so-called expected global optimization (EGO) algorithm. The more sophisticated and fully Bayesian TGP model requires a more sophisticated search method. Taddy et al. (2009) used the TGP EI calculations on random Latin hypercube candidate designs (LHDs, McKay et al., 1979) as an "oracle" sub-routine within a direct/pattern-search optimizer called APPS (Gray and Kolda, 2006; Kolda, 2005) and illustrate how these tools may be used to obtain the optimal design for a circuit device. Gramacy and Taddy (2010, Section 3) propose a simpler, R-centric variant via the opposite embedding—where the LHD candidates are augmented by an oracle point obtained by searching the maximum *a posteriori* TGP surface for minima via the `optim` function—that has been shown to perform well in many examples.

In any case, the presence of constraints complicates the notion of expected improvement. It does not make any sense to knowingly waste time evaluating the function in a region expected to return a constraint violation. Thus we need to trade off between the EI and the probability that a valid response will be returned. To do this, we simply multiply the EI by the probability that we will get a valid response. At each tried location, we can record

whether or not a valid response was returned by the simulator. This information can be used to inform a random forest classifier. Let $h(\mathbf{x})$ be the predicted probability of a valid response at input $\mathbf{x}$ estimated by the random forest classifier. We extend the algorithm in Equation (3) by now choosing the point $\mathbf{x}'$ that maximizes the expected constrained improvement:

$$\mathbf{x}' = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{E}\{I(\mathbf{x})\}h(\mathbf{x}). \tag{4}$$

With this modification, we can focus our efforts where they are most likely to be fruitful. Note that this approach takes a global view of the optimization problem, as both the function emulator and the constraint probability map cover the full input space. As with vanilla EI, there are several ways one can proceed to search for the $\mathbf{x}$ which maximizes the EI. All of the techniques mentioned above would apply in the expected constrained improvement context. Our approach is based on the R implementation described by Gramacy and Taddy (2010, Section 3).

4 Illustrative Example

We provide a relatively simple example here to illustrate our methods. Suppose we want to minimize the function

$$\begin{aligned} Z(x_1, x_2) &= -w(x_1)w(x_2) && (5)\\ w(x) &= \exp\left(-(x-1)2\right) + \exp\left(-0.8(x+1)2\right) - 0.05\sin\left(8(x+0.1)\right). && (6) \end{aligned}$$

Figure 1 shows the negative of the surface (i.e., as a maximization problem) for visibility (it is much easier to see the peaks than the valleys). There are four separated modal regions, and local modes within each of the regions. The true global minimum is shown at $x_1 = -1.0408, x_2 = -1.0408$. Now suppose that when we try to evaluate this function, it will only return valid values inside of an ellipse, but that we don't know anything about this valid

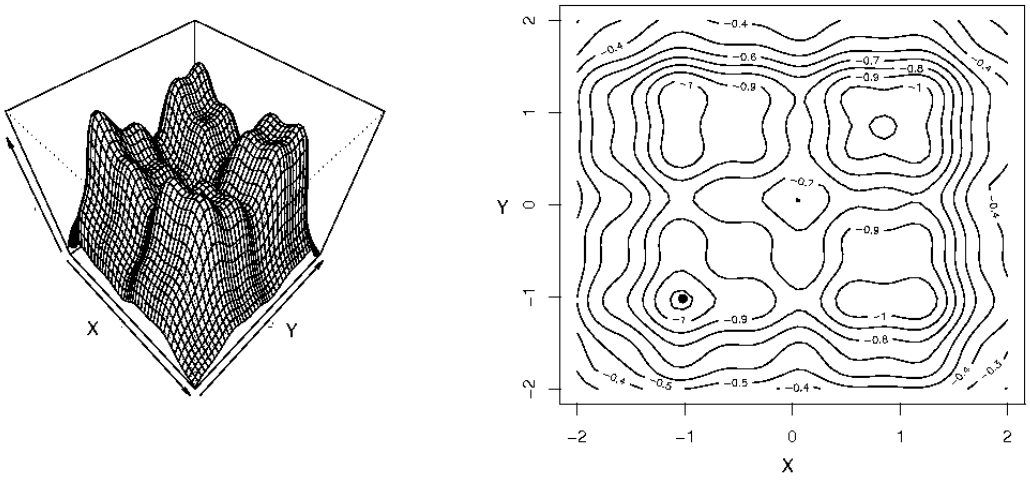

Figure 1: The function to be optimized, shown without constraints and shown as a maximization problem for visibility. The optimum is shown with the dot on the right hand plot.

region in advance and need to learn about it from the data.

To begin our search, we generate an initial design of 20 random points from a LHD and evaluate the function at those points. We use this initial dataset to fit the treed Gaussian process emulator and the random forest classifier. We then compute the EI at a fresh set of 100 LHD locations, multiply it by the predicted probability of obtaining a valid response at those candidate locations, and choose the maximum of that result as the next point to evaluate. If we obtain a valid response, that completes the iteration. Otherwise, we sample at the location with the next highest expected constrained improvement until we obtain a valid response (note that until a valid response is obtained, there is no need to update the emulator). We only retrain the random forests classifier and TGP emulator once per iteration, i.e., only after a valid response is obtained. We do fifty iterations, which provides us with the answer.

Figures 2 and 3 show the results from this implementation. Figure 2 shows an intermediate stage after twenty iterations. The upper left plot shows the truth along with the

bounding ellipse (only points inside the ellipse return a valid value) which we are treating as unknown and arbitrarily-shaped. The lower left plot shows the current emulated surface, which is a reasonable approximation of the truth within the valid region, although it will continue to improve as more locations are sampled (and no data is available outside the valid region, so we cannot expect a perfect match outside that region). The solid dots show the locations that have been evaluated, and the open circles show a Latin hypercube over which we conduct our search for the next point with highest expected constrained improvement. The top middle plot shows the EI surface. The bottom middle plot shows the predicted probability of returning a valid value obtained from fitting the random forest classifier. The open circles are the locations which have been evaluated successfully, and the filled circles are the locations which have been evaluated but produced a constraint violation. The upper right plot shows the product of the EI and the probability of being valid, and it is this surface that is maximized. The location shown at the crosshairs is the chosen point. The bottom right plot shows the progress in minimization. As the process proceeds, the emulator and the classifier better learn about their respective problems and seek out lower minima until they cannot find anything better. Figure 3 shows the result after fifty iterations, at which point we have found our minimum.

5 Hydrology Example

A key motivating example for this work is the hydraulic capture problem from the community problems (Mayer et al., 2002). The goal of the problem is to find a configuration of up to four wells to control the direction of groundwater movement to contain a contaminant plume, and to do so at minimal cost. The constraints in this problem are in the form of specifications on the hydraulic head differences at designated locations, and the objective function is the monetary cost of installing and operating the wells. More details on this problem are available in Mayer et al. (2002) and Fowler et al. (2004). A number of solutions have been proposed

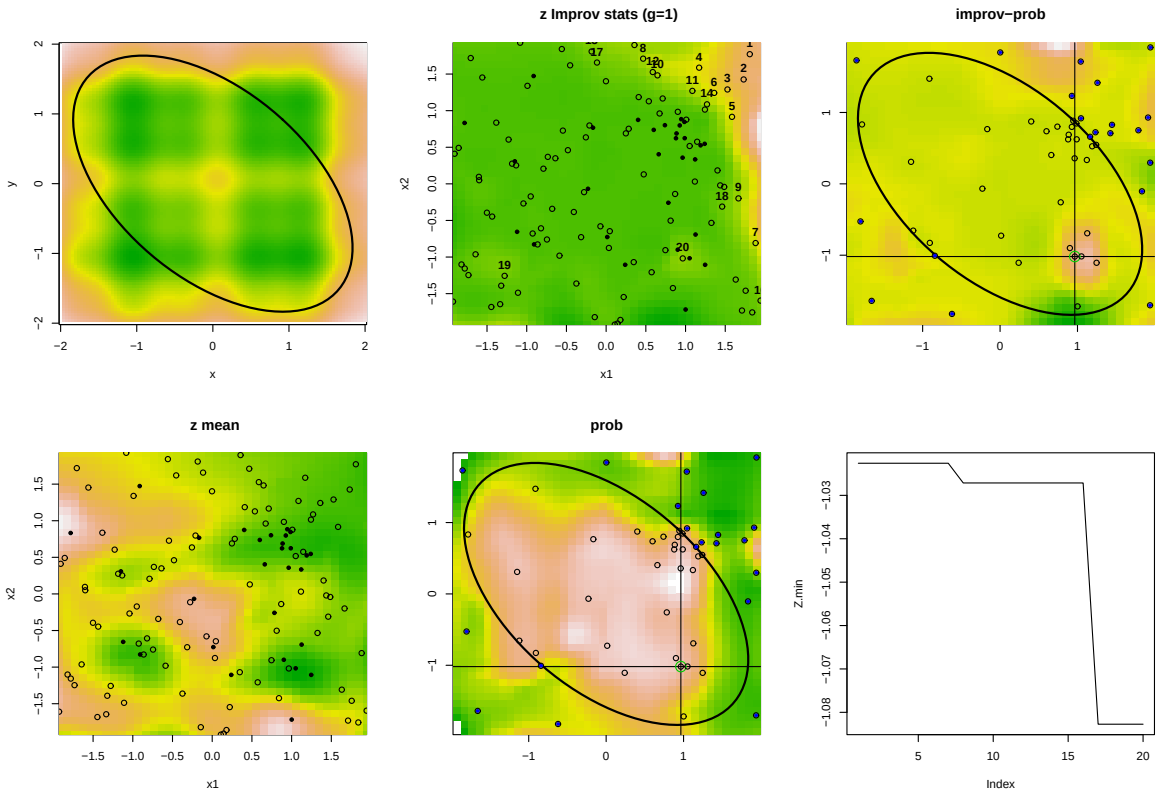

Figure 2: After 20 iterations: The function to be optimized with valid region, the fitted emulator, the EI surface, the fitted probability of being valid, the expected constrained improvement, and the progress in minimization. White shows high values and green shows low values.

for this problem, and some comparisons and discussion of the difficulties are available in references such as Fowler et al. (2008); Gray et al. (2009); Hemker et al. (2008). This problem is particularly challenging for two reasons. First, the objective function is quite irregular and difficult to predict. Second, the dimension of the space is not fixed, varying from twelve dimensions if four wells are used, down to three dimensions if only one well is used (the three parameters corresponding to each well are detailed below).

The hydraulic capture problem is described in Mayer et al. (2002) and the implementation details can be found in Fowler et al. (2008). Intuitively, the problem is illustrated in Figure 4. The green oval shows the relative location of a contaminant plume in a rectangular aquifer. The circles around the exterior of the plume represent the gradient constraints on

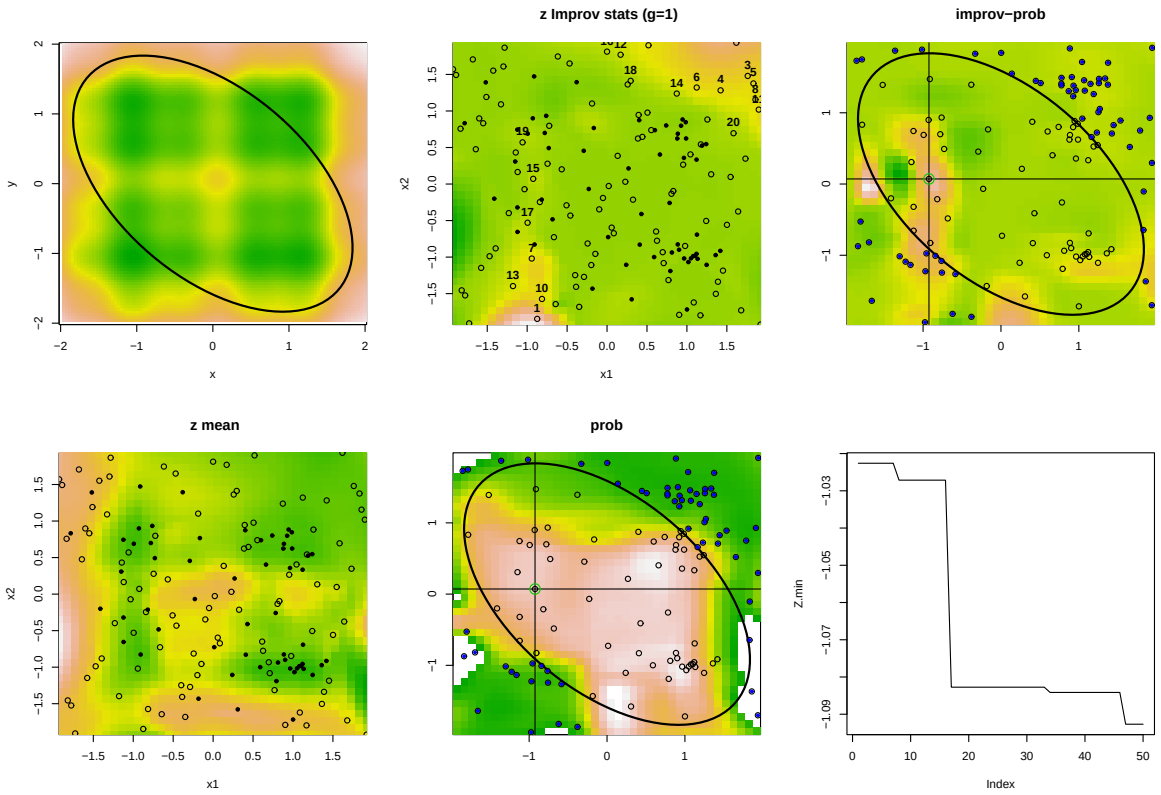

Figure 3: After 50 iterations: The function to be optimized with valid region, the fitted emulator, the EI surface, the fitted probability of being valid, the expected constrained improvement, and the progress in minimization. White shows high values and green shows low values.

the hydraulic head values, aligned so that flow will be forced towards the interior of the plume to prevent migration. Groundwater flow in the natural system in the absence of wells is towards the northeast, indicated by the arrow in the lower left corner. In this illustration, we also use stars to show a reasonable configuration of four wells–two extracting water inside the plume and two injecting water outside the plume. The inputs of the objective function are the well locations (in x-y coordinates) and the flow rates of each of the wells, so that there are three parameters corresponding to each well. The function output is the total cost of installing and operating the wells. Note that the calculation of this cost requires the results from the groundwater simulator.

Our first approach is to start with four wells and allow the search to work its way down

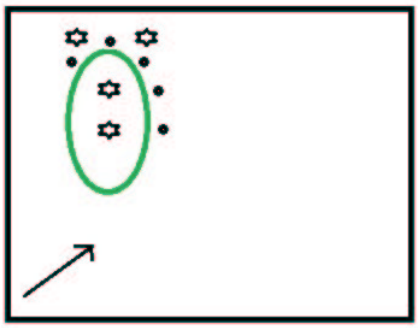

Figure 4: Pictorial representation of the constrained optimization problem

to a single well iteratively. When the flow rate of a well is less than $1 * 10^{-4}$, the well is turned off and the flow rate is set to zero. The search is initialized by seeding the space with a Latin hypercube to obtain some points to fit the emulator. We then propose a set of candidate points from a Latin hypercube, but we augment the candidate set over which we evaluate the expected improvement in two ways. First, following Taddy et al. (2009), we add fifty points in a ball of small radius around the best point. Second, we add a candidate point equal to the best point with the smallest well set to flow zero. We then fit the TGP emulator and the random forest classifier, and rank the candidate points based on the product of their expected improvement and probability of being valid, as per Equation (4). We evaluate the points in rank order until one returns a valid output, then we move to the next iteration, updating the statistical models and repeating. This algorithm is able to easily reduce the number of wells from four to two, but it has more difficulty moving to a single well solution. The best solution it finds is a cost of \$36,389.83 using two wells. We note that this solution is better than the solutions found by six of the nine methods used in a comparison study of techniques on this problem by Fowler et al. (2008).

However, this two well solution is not the best solution. We take a second approach to searching the space, using our expected improvement approach for each of one-, two-, three-, and four-well spaces separately. With this fixed-dimension approach, we find a best one-well solution of \$22,952.77, which is better than all of the solutions found in Fowler

et al. (2008), with the best solution reported there being $23,421. We note that the set up of the experiment here is different than in the alternatives; our method requires more than one initial valid point to initialize the statistical models. The comparison methods might perform differently under the same initial conditions we used here.

6 Conclusions

In conclusion, we have proposed a new approach for constrained optimization based on statistical models. By simultaneously learning about both the objective function and the boundary of the constraints, we are able to better focus our efforts on the valid regions.

Some areas for further work include the consideration of other classification models. For example, when the constraint boundary is relatively smooth, a Gaussian process classifier may learn it more quickly. Another approach is to move beyond expected improvement at a point and consider a more global improvement function, such as one that integrates improvement over the whole space. Convergence diagnostics could also be better explored, such as those along the lines of Gramacy and Taddy (2010, Section 3).

Acknowledgments

This research was initiated at a workshop at the American Institute of Mathematics on Derivative-Free Hybrid Optimization Methods for Solving Simulation-Based Problems in Hydrology, and was also partially supported by NSF grant DMS-0906720.

References

Breiman, L. (2001). "Random Forests." *Machine Learning*, 45, 5–32.

Breiman, L., Friedman, J. H., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.

Chipman, H., George, E., and McCulloch, R. (2002). "Bayesian Treed Models." *Machine Learning*, 48, 303–324.

Cressie, N. A. C. (1993). *Statistics for Spatial Data, revised edition*. John Wiley & Sons.

Fang, K.-T., Li, R., and Sudjianto, A. (2006). *Design and Modeling for Computer Experiments*. Boca Raton: Chapman & Hall/CRC.

Finkel, D. E. and Kelley, C. T. (2009). "Convergence Analysis of Sampling Methods for Perturbed Lipschitz Functions." *Pacific Journal of Optimization*, 5, 339–350.

Fowler, K. R., Kelley, C. T., Kees, C. E., and Miller, C. T. (2004). "A Hydraulic Capture Application for Optimal Remediation Design." In *Proceedings of the XV International conference on Computational Methods in Water Resources*, eds. C. T. Miller, M. W. Farthing, G. G. W., and G. F. Pinder.

Fowler, K. R., Reese, J. P., Kees, C. E., Dennis Jr., J. E., Kelley, C. T., Miller, C. T., Audet, C., Booker, A. J., Couture, G., Darwin, R. W., Farthing, M. W., Finkel, D. E., Gablonsky, G., Gray, G., and Kolda, T. G. (2008). "A Comparison of Derivative-free Optimization Methods for Water Supply and Hydraulic Capture Community Problems." *Advances in Water Resources*, 31, 5, 743–757.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian Data Analysis*. London: Chapman and Hall.

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.

Gramacy, R. B. and Taddy, M. A. (2010). "Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with `tgp` Version 2, an R Package for Treed Gaussian Process Models." *Journal of Statistical Software*, 33, 6, 1–48.

Gray, G. A., Fowler, K. R., and Griffin, J. D. (2009). "A Hybrid Optimization Scheme for Solving the Hydraulic Capture Problem with an Unknown Number of Wells." In *The First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering*.

Gray, G. A. and Kolda, T. G. (2006). "Algorithm 856: APPSPACK 4.0: Asynchronous Parallel Pattern Search for Derivative-Free Optimization." *ACM Transactions on Mathematical Software*, 32, 3, 485–507.

Green, P. (1995). "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination." *Biometrika*, 82, 711–732.

Hemker, T., Fowler, K. R., Farthing, M. W., and von Stryk, O. (2008). "A Mixed-Integer Simulation-based Optimization Approach with Surrogate Functions in Water Resources Management." *Opt. Eng.*, 9, 4, 341–360.

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. and O'Hagan, A. (2000). "Predicting the Output from a Complex Computer Code when Fast Approximations are Available." *Biometrika*, 87, 1–13.

Kolda, T. G. (2005). "Revisiting asynchronous parallel pattern search for nonlinear optimization." *SIAM Journal of Optimization*, 16, 2, 563–586.

Liaw, A. and Wiener, M. (2002). "Classification and Regression by randomForest." *R News*, 2, 3, 18–22.

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.

McKay, M. D., Conover, W. J., and Beckman, R. J. (1979). "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code." *Technometrics*, 21, 239–245.

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.

Stein, M. L. (1999). *Interpolation of Spatial Data*. New York, NY: Springer.

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.