## Breast Cancer Epidemiology: Calibrating Simulations via Optimization

Michael C. Ferris
(joint work with Geng Deng, Dennis G. Fryback, Vipat Kuruchittham)

We investigate the use of optimization and data mining techniques for calibrating the input parameters to a discrete event simulation code. In the context of a breast-cancer epidemiology model we show how a hierarchical classifier can accurately predict those parameters that ensure the simulation replicates benchmark data within 95% confidence intervals. We formulate an optimization model that evaluates solutions based on an integer valued score function. The scores are determined from a simulation run (and are therefore subject to stochastic variations), and are expensive to calculate.

The Wisconsin Breast Cancer Epidemiology Simulation uses detailed individual-woman-level discrete event simulation of four processes (breast cancer natural history, detection, treatment and nonbreast cancer mortality among U.S. women) to replicate breast cancer incidence rates according to the Surveillance, Epidemiology, and End Results (SEER) Program data from 1975 to 2000. Incidence rates are calculated for four different stages of tumor growth, namely, in situ, localized, regional, and distant; these correspond to increasing size and/or progression of the disease. Each run involves the simulation of 3 million women and takes approximately 8 minutes to execute on a 1 GHz Pentium machine with 1 GB of RAM.

The four simulated processes overlap in complex ways, and thus it is very difficult to formulate analytical models of their interactions. However, each can be modeled by simulation; these models need to take into account the increase in efficiency of screening processes that has occurred since 1975, the changes in nonscreen detection as a result of increased awareness of the disease, and a variety of other changes during that time. The simulations are grounded in mathematical and statistical models that are formulated by using a parameterization. For example, the natural history process in the simulation can be modeled by using a Gompertzian growth model that is parameterized by a mean and variance typically unknown exactly but for which a range of reasonable values can be estimated. The overall simulation facilitates interaction between the various components, but it is extremely difficult to determine values for the parameters that ensure the simulation replicates known data patterns across the time period studied. In all, there are 37 parameters, most of which interact with each other and are constrained by linear relationships. Further details can be found in [1, 3].

A score is calculated that measures how well the simulation output replicates an estimate of the incidence curves in each of the four growth stages. Using SEER and Wisconsin Cancer Reporting System (WCRS) data, we generate an envelope that captures the variation in the data that might naturally be expected in a population of the size we simulated. For the 26 years considered, the four growth stages give a total of 104 points, each of which is tested to see whether it lies in the envelope.

The number of points outside the envelope is summed to give the score (0 is ideal). While one could argue that distance to the envelope might be a better measure, such calculations are scale dependent and were not investigated. Unfortunately, the score function also depends on the "history" of breast cancer incidence and mortality that is generated in the simulation based on a random seed value $\omega$. We will adopt the notation $f_\omega(v)$, where $v$ represents the vector of parameters and $\omega$ indexes the replication. While we are interested in the distribution (over $\omega$) of $f_\omega(v)$, we will focus here on the problem

$$\min_v \max_\omega f_\omega(v).$$

The purpose of this study is to determine parameter values $v$ that generate small values for the scoring function. Prior to the work described here, acceptance sampling had been used to fit the parameters. Essentially, the simulation was run tens of thousands of times with randomly chosen inputs to determine a set of good values. With over 450,000 simulations, only 363 were found that had a score no more than 10. That is, for a single replication $\omega$, 363 vectors $v$ had $f_\omega(v) \leq 10$.

Our first goal was to generate many more vectors $v$ with scores no more than 10. To do this, we attempted to use the given scoring function data to generate a classifier that quickly predicts whether a given vector $v$ is in

$$L(\lambda) = \{v | f_\omega(v) \leq \lambda\}, \text{for a fixed replication } \omega.$$

We typically use $\lambda = 5$ to indicate good fit and $\lambda = 10$ for acceptable parameter choices. Our approach is as follows:

– Split the data into a training (90%) and testing (10%) set.
– Given the training set, generate a (hierarchical) classifier that predicts membership of $L(\lambda)$. Validate this classifier on the testing set.
– Generate 100,000 potential values for $v$, uniformly at random.
– For those vectors $v$ that the classifier predicts are in $L(\lambda)$, evaluate $f_\omega(v)$ via simulation.

Since the classifier is cheap to evaluate, this process facilitates a more efficient exploration of the parameter space. Clearly, instead of using a single replication $\omega$, we could replace $f_\omega(v)$ by $\max_{\omega\in\Omega} f_\omega(v)$ where $\Omega = \{\omega_1, \ldots, \omega_m\}$ for some $m > 1$. In fact this approach was carried out. The difficulty is that we require replication data (for our experiments we choose $m = 10$) and we update the definition of $L(\lambda)$ appropriately. However, the process we follow is identical to that outlined here.

In our setting, $v$ has dimension 37. Using expert advice, we allowed only 9 dimensions to change; the other 28 values were fixed to the feasible values that have highest frequency of occurrence over the "positive" samples. For example, if $v_{37}$ can take possible values from $[\phi_1, \phi_2, \ldots, \phi_n]$, then we set the value of $v_{37}$ to be $\arg\min_{i=1}^n \frac{P_i}{W_i}$, where $P_i$ and $W_i$ are the number of appearances of $\phi_i$ in the positive and whole sample set. This is similar to using a naive Bayesian classifier to determine which value has the highest likelihood to be "positive". Our experiments showed this choice of values outperformed even the values that experts deemed appropriate for these 28 values; a posteriori analysis confirmed their superiority.

We generated a hierarchical classifier. The key difficulty in generating a classifier is the fact that we have a vast majority of "negative" data points (i.e., vectors $v \notin L(\lambda)$). By successively projecting our training data into two-dimensional slices, we identified two pairs of planes (meanGamma/varGamma and onsetProp/lag) in which only "negative" data points in our training set were present outside a small band of values. The top level of the classifier labels points outside these bands as "negative". The remaining points (within the bands, the positive and negative points are intermingled) are classified by using the following procedure.

Given a particular training set $A$, a variety of support vector machine classifiers can be generated by solving an optimization problem for values $u$ and $\gamma$, and using the kernel classifier [7]

$$K(v', A')u - \gamma \leq 0$$

to imply that a new point $v$ is "negative", where $K$ is a given kernel function. We used the following kernels: linear, polynomial degree 2, polynomial degree 3, and Gaussian. We also used the C45 decision tree classifier and $k$-nearest neighbor classifier with $k = 5$. All of these classifiers are publicly available [6, 9].

Furthermore, the one-sided sampling approach [4] was used to generate a number of different training sets; the sampling approach iteratively removes "negative" points in a rigorously defined manner, and we stop this process when there are approximately 500 "negative" points remaining (there are around 300 "positive" points in each training set). The resulting classifier is evaluated on the testing set by using the measures

$$TP = \frac{\#\text{ correctly classified positives}}{\text{total }\#\text{ of positives}} \text{ and } TN = \frac{\#\text{ correctly classified negatives}}{\text{total }\#\text{ of negatives}}.$$

(Note that cross-validation accuracy is inappropriate to use in this setting because it can be made large by classifying all points as "negative" on account of the imbalanced nature of the data.) Classifiers are discarded if the value of $TP$ is less than 0.9 (typically $TN$ is around 0.4). This value was chosen to guarantee the probability of removing positive points in error is small. We also generate training sets by resampling with replacement.

For a uniform sample of 100,000 potential values of $v$, the naive banding classifier removes all but 8 640. Each of the above classifiers was used successively to determine whether the point $v$ was "negative" (and hence removed from consideration); if not, $v$ was passed onto the next classifier. This process was repeated until the number of points being removed decreased to zero. At that stage there were 788 points that were hypothesized to be "positive". These 788 points were tested using simulation, and 65% were found to be "positive". This is a significant improvement over the random sampling scheme.

A further sequence of classifiers was determined from a training set generated by using the above sampling schemes, but where we adjusted the number of "negative" points in the training set so that the resulting values for $TP$ and $TN$ were approximately 0.6 and 0.7. These additional classifiers have a larger chance of removing "positive" samples in error, but they reduce the number of remaining points in our sample much more quickly. For our example set the remaining 788

points was reduced to 220 points. Evaluating these remaining points by simulation, 195 were found to be in $L(10)$. Thus, with very high success rate (89%), our classifier is able to predict values of $v$ that have a low score $f_\omega(v)$.

We employed the classifier technique above to generate a large number of samples in $L(30)$. Given these samples, we used the DACE toolbox [5] to fit a kriging model to the data, which we consider a surrogate function [2] for our objective. We used the Nelder-Mead simplex method [8] to optimize this surrogate and generated several local minimizers for this function based on different trial starting points. These local minimizers were evaluated by simulation. To improve our results further, we updated the surrogate function with the simulation results of the local minimizers and repeated the optimization. The parameter values found by using this process outperform all previous values found. Furthermore, expert analysis of various output curves generated from the simulation results with the best set of parameter values confirms the quality of this solution.

While our procedure is somewhat ad hoc, the following conclusions are evident:

– The classifier technique is cheap to use and predicts good parameter values very accurately without performing additional simulations.
– A hierarchical classifier significantly improves classification accuracy.
– Imbalanced training data has a detrimental effect on classifier behavior. Ensuring the data is balanced in size is crucial before generating classifiers.

The classifier facilitates easy generation of parameter settings within a given level set of score values and potentially allows investigation of such level sets and good parameter settings from a biological perspective. Future work will investigate characterizing the level set more precisely with the aim of enhancing biological understanding of the model parameters.

## References

[1] CISNET, `http://cisnet.cancer.gov/profiles/`.
[2] J. E. Dennis, A. Booker, P. Frank, D. Serafini, V. Torczon, and M. Trosset, A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization,* 17(1):1–13, 1999.
[3] D. G. Fryback, N. K. Stout, M. A. Rosenberg, A. Trentham-Dietz, V. Kuruchittham, and P. L. Remington. The Wisconsin breast cancer epidemiology simulation model. Preprint, January 2005.
[4] M. Kubat and S. Matwin. Addressing the curse of imbalanced data sets: One sided sampling. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997, pp. 179–186.
[5] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. DACE – A Matlab kriging toolbox, Informatics and Mathematical Modelling. Technical University of Denmark, DTU, 2002.
[6] J. Ma, Y. Zhao, and A. Stanley. OSU SVM classifier Matlab Toolbox. `http://www.ece.osu.edu/~maj/osu_svm/`.
[7] O. L. Mangasarian, Generalized support vector machines. In *Advances in Large Margin Classifiers,* edited by A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, MIT Press, Cambridge, MA, 2000, pp. 135–146.
[8] J. A. Nelder and R. Mead, A simplex method for function minimization. *Computer Journal,* 7:308–313, 1965.

[9] J. Weston, A. Elisseeff, G. Bakir, and F. Sinz. The Spider, Matlab Machine Learning Toolbox. `http://www.kyb.tuebingen.mpg.de/bs/people/spider/`.

## $L^1$-Optimal Boundary Control of a String to Rest in Finite Time

Martin Gugat

Hyperbolic partial differential equations often appear as models in engineering, for example as systems of conservation laws that model fluid flow.

The control of such systems is usually possible only with boundary controls, which in the mathematical model corresponds to control via the boundary conditions. To get some insight into the nature of optimal controls for such systems, we consider the following problem of optimal Dirichlet boundary control for the wave equation:

$$\text{(P)} \qquad \begin{cases} \min\limits_{u_1,u_2\in L^1(0,T)} \displaystyle\int_0^T |u_1(t)| + |u_2(t)|\, dt \quad \text{suject to} \\ y_{tt}(x,t) = c^2 y_{xx}(x,t), & (x,t) \in (0,L)\times(0,T) \\ y(0,t) = u_1(t), \quad y(L,t) = u_2(t), & t \in (0,T) \\ y(x,0) = y_0(x), \quad y_t(x,0) = y_1(x), & x \in (0,L) \\ y(x,T) = 0, \qquad y_t(x,T) = 0, & x \in (0,L). \end{cases}$$

The functions $y_0$ and $y_1$ are given, as well as the real numbers $T > 0$, $L > 0$, $c > 0$.

In general, this problem does not have a unique solution. An explicit representation of all solutions is given in the following theorem.

**Theorem 1.** *Assume that $T \geq t_0 = L/c$, that $y_0 \in L^1(0,L)$, and that $Y_1(x) = \int_0^x y_1(s)\, ds \in L^1(0,L)$. For $t \in (0,t_0)$, let*

$$\begin{aligned} \alpha_0(t) &= y_0(ct) + (1/c)\int_0^{ct} y_1(s)\, ds, \\ \beta_0(t) &= y_0(L-ct) - (1/c)\int_0^{L-ct} y_1(s)\, ds. \end{aligned}$$

*Choose a real number $r$ that minimizes*

$$I(r) = \frac{1}{2}\int_0^{t_0} |\alpha_0(t) - r| + |\beta_0(t) + r|\, dt. \tag{1}$$

*Let $k = \max\{j \in \mathbb{N} : jt_0 \leq T\}$ and $\Delta = T - kt_0$. For $j \in \{0,\dots,k\}$ and $t \in (0,\Delta)$, let $\lambda_j(t) \geq 0$, $\nu_j(t) \geq 0$ almost everywhere be such that $\lambda_j(\alpha_0 - r) \in L^1(0,\Delta)$, $\nu_j(\beta_0 + r) \in L^1(0,\Delta)$, and*

$$\sum_{j=0}^{k} \lambda_j(t) = 1 = \sum_{j=0}^{k} \nu_j(t) \quad \textit{almost everywhere on } (0,\Delta).$$