Panel: Statistical Emulation & Optimizaion

Robert Wolpert
Duke University Department of Statistical Science
wolpert@stat.duke.edu

SAMSI | Bayesian Nonparametrics Workshop

Outline

Deterministic Computer Simulation Models

Reality:	$Y^R(x),$	$x \in \mathcal{X}$
Field Measurements:	$Y^F(x_j) = Y^R(x_j) + \epsilon_j^F,$	$x_j \in \mathcal{F}$
Computer Model:	$Y^R(x_j) = Y^M(x_j, u) + \delta_u(x_j),$	$x_j \in \mathcal{D},\ u \in \mathcal{U}$

Model typicaly has tuning parameters u, discrepancy $\delta_u(\cdot)$.

Deterministic Computer Simulation Models

Reality: $Y^R(x), \quad x \in \mathcal{X}$

Field Measurements: $Y^F(x_j) = Y^R(x_j) + \epsilon_j^F, \quad x_j \in \mathcal{F}$

Model typicaly has tuning parameters u, discrepancy $\delta_u(\cdot)$.
We'll ignore (or fix) the tuning parameter for now, and write

Computer Model: $Y^R(x_j) = Y^M(x_j) + \delta(x_j), \quad x_j \in \mathcal{D}.$

Deterministic Computer Simulation Models

Reality: $Y^R(x), \quad x \in \mathcal{X}$
Field Measurements: $Y^F(x_j) = Y^R(x_j) + \epsilon_j^F, \quad x_j \in \mathcal{F}$

Model typicaly has tuning parameters u, discrepancy $\delta_u(\cdot)$.
We'll ignore (or fix) the tuning parameter for now, and write

Computer Model: $Y^R(x_j) = Y^M(x_j) + \delta(x_j), \quad x_j \in \mathcal{D}.$

- Field Measurements are expensive. That's one reason to use a model.

Deterministic Computer Simulation Models

Reality: $Y^R(x), \quad x \in \mathcal{X}$
Field Measurements: $Y^F(x_j) = Y^R(x_j) + \epsilon_j^F, \quad x_j \in \mathcal{F}$

Model typicaly has tuning parameters u, discrepancy $\delta_u(\cdot)$.
We'll ignore (or fix) the tuning parameter for now, and write

Computer Model: $Y^R(x_j) = Y^M(x_j) + \delta(x_j), \quad x_j \in \mathcal{D}.$

- Field Measurements are expensive. That's one reason to use a model.
- Model runs are expensive too.

Goals

Optimization

- Find input values $x \in \mathcal{X}$ to minimize some functional, to find
$$\operatorname*{argmin}_x F\big(Y^M(x)\big)$$
usually in the hope this will also minimize reality $F\big(Y^R(x)\big)$ (highest concentration, elevation, best fishing spot, etc)
- Methods: Iterative search (BFGS, N/M) if objective is smooth; otherwise Stochastic Search (expensive).

Goals

Optimization

- Find input values $x \in \mathcal{X}$ to minimize some functional, to find
$$\operatorname*{argmin}_x F\big(Y^M(x)\big)$$
usually in the hope this will also minimize reality $F\big(Y^R(x)\big)$ (highest concentration, elevation, best fishing spot, etc)
- Methods: Iterative search (BFGS, N/M) if objective is smooth; otherwise Stochastic Search (expensive).

Bayesian Expectation

- Try to estimate some probability, fraction, or mean quantity, to find
$$\mathrm{E}\Big[G\big(Y^M(x)\big)\Big]$$
given prior $\pi(x)$ and perhaps observations.
- Methods: Importance sampling or MCMC (expensive).

How expensive?

A few examples:

Ast Galaxy formation: **Galform** model for Galaxy Formation costs about 1.5h for each model run, on large multiprocessor array;

Geo Volcano Simulation: **Titan2D** takes about 30-60m for each run, on 64-processor array (it's slower on larger arrays due to IPC needs);

HEP Heavy ion collision: **UrQMD** takes about 1–2h to reconstruct one collision (which lasts 10^{-24}s), on the OSG using hundreds of nodes.

How expensive?

A few examples:

Ast Galaxy formation: **Galform** model for Galaxy Formation costs about 1.5h for each model run, on large multiprocessor array;

Geo Volcano Simulation: **Titan2D** takes about 30-60m for each run, on 64-processor array (it's slower on larger arrays due to IPC needs);

HEP Heavy ion collision: **UrQMD** takes about 1–2h to reconstruct one collision (which lasts 10^{-24}s), on the OSG using hundreds of nodes.

Either Stochastic Search or MCMC posterior evaluations will entail from 10^4 to 10^6 model evaluations

How expensive?

A few examples:

Ast Galaxy formation: **Galform** model for Galaxy Formation costs about 1.5h for each model run, on large multiprocessor array;

Geo Volcano Simulation: **Titan2D** takes about 30-60m for each run, on 64-processor array (it's slower on larger arrays due to IPC needs);

HEP Heavy ion collision: **UrQMD** takes about 1–2h to reconstruct one collision (which lasts 10^{-24}s), on the OSG using hundreds of nodes.

Either Stochastic Search or MCMC posterior evaluations will entail from 10^4 to 10^6 model evaluations

And 10^5 hours is about 11.4 years. It would take three orders of magnitude improvement ("three logs") to bring this down to four days.

A Cheap Way Out

Even though $Y^M(x)$ is deterministic, we can treat it as a stochastic random field— because *we don't know* the values of $Y^M(x)$ for any x outside the design set $\mathcal{D}$.

A Cheap Way Out

Even though $Y^M(x)$ is deterministic, we can treat it as a stochastic random field— because *we don't know* the values of $Y^M(x)$ for any x outside the design set $\mathcal{D}$.

Usual choice: Gaussian Process

- $Y^M(x) \sim \mathrm{GP}\big(\mu(x),\ \lambda^{-1}R(x,y)\big)$
- Mean $\mu(x)$ taken to be zero; or uncertain constant; or regression $H\big(X(x)'\beta\big)$ for some known function $H(\cdot)$, known covariates $X(x)$ and uncertain β;
- Correlation $R(x,y)$ from some small isotropic parametric family, like Matérn or power-exponential

$$R(x,y) = \exp\Big(-\sum[\ |x_i - y_i|/\rho_i\ ]^\alpha\Big)$$

  for some shape parameter $0 < \alpha \le 2$ and distance scales $\{\rho_i > 0\}$
- Precision $\lambda > 0$ also uncertain.

How does this help?

Now we can:

1. Select a design set $\mathcal{D}$ of a few hundred (or fewer) points;
2. Find $y_j = Y^M(x_j)$ at each of them (and set $\mathbf{y} = \{y_j\}$);
3. "Train" the emulator, *i.e.*, find estimates for α and $\{\beta_i\}$ and λ and $\{\rho_i\}$;
4. Use $\mathsf{E}[Y^M(x) \mid Y^M(\mathcal{D}) = \mathbf{y}]$ as a surrogate for $Y^M(x)$ in the Optimization and Expectation goals above

How does this help?

Now we can:

1. Select a design set $\mathcal{D}$ of a few hundred (or fewer) points;
2. Find $y_j = Y^M(x_j)$ at each of them (and set $\mathbf{y} = \{y_j\}$);
3. "Train" the emulator, *i.e.*, find estimates for α and $\{\beta_i\}$ and λ and $\{\rho_i\}$;
4. Use $\mathrm{E}[Y^M(x) \mid Y^M(\mathcal{D}) = \mathbf{y}]$ as a surrogate for $Y^M(x)$ in the Optimization and Expectation goals above
5. Take comfort in the Advantages:
   - Emulator evaluates in (milli)seconds, while Model takes hours— we get our three logs, and more;
   - Emulator comes with built-in approximation error estimates based on the conditional variance of the GP.

Summary & Next Steps

- Contemporary scientific models are too slow for direct use in optimization or (simulation-based) Bayesian analysis;
- In many application areas people have successfully applied Gaussian Process Emulators to speed things up

Summary & Next Steps

- Contemporary scientific models are too slow for direct use in optimization or (simulation-based) Bayesian analysis;
- In many application areas people have successfully applied Gaussian Process Emulators to speed things up

Now, let's discuss the open questions—

- Are GPs the best surrogates to use?
- Does it matter what covariance structure we use?
- How can isotropy fail? When can it be recovered by changes in variables?
- What if I want to model an n-dimensional Model output, for HUGE dimension n?
- What if I think I need a big design set $\mathcal{D}$ with more than a few hundred pionts? I'll need to invert a big matrix inside a loop...
- More questions? Suggestions? Concerns?

Let's talk!