Stochastic Kriging for Bermudan Option Pricing

International Conference on Monte Carlo Techniques, Paris

Mike Ludkovski

Dept of Statistics & Applied Probability UC Santa Barbara

July 6 2016
Work supported by NSF DMS-1222262
Optimal Stopping via Monte Carlo

- \((X_t)\): Markov state process, \(t = 0, 1, 2, \ldots\)
- Dynamics \(X_{t+1} = F(X_t, \varepsilon_t)\), smooth transition density \(p(t, y|0, x)\)
- Wish to maximize expected reward \(V(t, x) = \sup_{\tau \leq T} \mathbb{E}[h(\tau, X_\tau)]\)
- from stopping at \(\tau\)
- Optimization is over hitting times \(\tau = \min\{s : X_s \in \mathcal{S}_s\} \land T\)
- Timing Value \(T(t, x) := \mathbb{E}_{t,x} [V(t+1, X_{t+1})] - h(t, x)\)
- Stopping set \(\mathcal{S}_t = \{x : T(t, x) < 0\}\)
Simulation Approach

- Stochastic grid $x^n$, $n = 1, \ldots, N \Rightarrow$ Trajectories/scenarios $x_{t:T}^{1:N}$
- Evaluate future pathwise payoff $h(\tau_{t+1}, x_{\tau_{t+1}}^n)$ where
  $\tau_{t+1}^n := \min\{ s > t : x_s^n \in \hat{S}_s \}$
- Compare to immediate payoff: $y^n := h(\tau_{t+1}, x_{\tau_{t+1}}^n) - h(t, x_t^n)$
- Then $\mathbb{E}[Y(x)] = \mathbb{E}_{t,x}[h(\tau_{t+1}, X_{\tau_{t+1}})] - h(t, x) = T(t, x)$
- Rank expected future payoff vs present reward
- Policy search vs Value-function-approximation
Abstract Statistical Problem

- Have a stochastic simulator $Y(x) = f(x) + \varepsilon$, $\mathbb{E}[\varepsilon] = 0$
- Input space $x \in \mathcal{X} \subset \mathbb{R}^d$ (continuous, multi-dimensional)
- Goal: learn $\mathcal{G} := \{x : f(x) \leq 0\}$
- Discriminate between positive and negative values of the latent function
- Precise loss function:

$$L(\hat{\mathcal{G}}) = \mathbb{E} \left[f(x) 1_{\mathcal{G} \Delta \hat{\mathcal{G}}}(x)\right]$$

where the expectation is over a given measure $\mathbb{P}$
- The responses $Y$ are pathwise costs-to-go (aka $q$-value); has intrinsic noise $\varepsilon$ due to the particular trajectory of $X$
What are the Challenges?

- How to approximate $\hat{f}$?
- How to measure goodness-of-fit?
- How to handle non-standard statistical context?
- How to generate simulations?
- How to prove/guarantee convergence?
- How to speed-up convergence?
- How to achieve scalability?
What are the Challenges?

- How to approximate $\hat{f}$?  
  Approximation architecture $\mathcal{H}$
- How to measure goodness-of-fit?  
  Loss function $\inf_{\mathcal{H}} \mathbb{E}[L(\hat{f}, f)]$
- How to handle non-standard statistical context?
- How to generate simulations?
- How to prove/guarantee convergence?
- How to speed-up convergence?
- How to achieve scalability?
What are the Challenges?

- How to approximate \( \hat{f} \)? Approximation architecture \( \mathcal{H} \)
- How to measure goodness-of-fit? Loss function \( \inf_{\mathcal{H}} \mathbb{E}[L(\hat{f}, f)] \)
- How to handle non-standard statistical context? Properties of \( \varepsilon \)
- How to generate simulations? Experimental design
- How to prove/guarantee convergence?
- How to speed-up convergence?
- How to achieve scalability?
What are the Challenges?

- How to approximate $\hat{f}$?  Approximation architecture $\mathcal{H}$
- How to measure goodness-of-fit?  Loss function $\inf_{\mathcal{H}} \mathbb{E}[L(\hat{f}, f)]$
- How to handle non-standard statistical context?  Properties of $\varepsilon$
- How to generate simulations?  Experimental design
- How to prove/guarantee convergence?  Behavior as $n \to \infty$
- How to speed-up convergence?  Non-asymptotics for a given $N$
- How to achieve scalability?

Minimize dependence on specific dimension, payoff $h(\cdot)$, dynamics $F$
What are the Challenges?

- How to approximate $\hat{f}$? Approximation architecture $\mathcal{H}$
- How to measure goodness-of-fit? Loss function $\inf_{\mathcal{H}} \mathbb{E}[L(\hat{f}, f)]$
- How to handle non-standard statistical context? Properties of $\varepsilon$
- How to generate simulations? Experimental design
- How to prove/guarantee convergence? Behavior as $n \to \infty$
- How to speed-up convergence? Non-asymptotics for a given $N$
- How to achieve scalability?
  Minimize dependence on specific dim $d$, payoff $h(\cdot)$, dynamics $F$
Statistical Learning

- Step I: experimental design – generate $x^{1:N}$
- Step II: sample $y^{1:N} = Y(x^{1:N})$ and estimate $\hat{S}$

- Low signal-to-noise ratio
- Strong heteroscedasticity
- Non-standard noise distribution

$(x, y)^{1:N}$ with $N = 10^4$, $X = [28, 40]$
Existing State-of-the-Art

- Approximation architectures: basis expansions; nonparametric regression; hierarchical methods; ...
- Goodness-of-fit: least squares; penalized least-squares; opportunity cost
- Heteroscedasticity, non-Gaussian noise: regularization, batching
- Experimental design: space-filling; sequential adaptive; importance sampling
Existing State-of-the-Art (cont)

- **Convergence proofs**: Belomestny, Bouchard, Clement, Gobet, Lamberton, Lapeyre, Pagès, Stentoft, Warin, ...
  Intuitively: policy-iteration is better...

- To **Speed-up** convergence: ASK the RIGHT questions to identify opportunities for improvement

- **Scalability**: used in a wide variety of contexts, often as a sub-procedure. Would like to have a smart algorithm that doesn’t require too much fine-tuning (e.g adaptive dictionary selection)
Contributions

1. A *nice* modeling framework is available in GP/kriging. One of the new tools emerging from machine learning. Arguably “smarter” and more flexible than working with basis functions.

2. Experimental design is arguably *more important* than the regression model. Default “density-based” sampling is highly inefficient. Investigate space-filling and adaptive designs. Replicated design.

3. The loss function resembles *classification*. Build a classification model by converting observations into 0/1 labels. Modifies the statistical behavior of the simulator. Promising in combination with adaptive design.
Formalize Statistical Learning

- Capture the idea that $f$ is learned from the data: $\mathcal{Z}^{(n)} \equiv (x, y)_1^n$ induces $\hat{F}^{(n)} = \mathbb{E}[f|\mathcal{Z}^{(n)}]$ posterior distribution (measure on $\mathcal{H}$)
- Treat the true map $f \in \mathcal{H}$ as a random function
- Specify prior distribution and then use Bayesian updating
- $\hat{F}_x^{(n)} = \mathbb{E}[f(x)|\mathcal{Z}^{(n)}]$ posterior at $x$ (measure on $\mathcal{X}$)

$\hat{f}^{(n)}$ and its 95% CI for 3 different $n$, L. (2016)
Stochastic Kriging

- $f$ is a realization of a **Gaussian random field** with a covariance structure defined by $K$, function space $\mathcal{H}_K = \text{span}(K(\cdot, x) : x \in \mathcal{X})$
- $K(x, x') := \mathbb{E}[f(x)f(x')]$ controls the spatial smoothness
- e.g. Gaussian kernel $K(x, x') = \tau^2 \exp(-\|x - x'\|^2/\theta^2)$ – elements of $\mathcal{H}_K$ are $C^\infty$, with lengthscale $\theta$ and fluctuation scale $\tau$.
- The **posterior conditional** on $Z \equiv (x, y)^{1:N}$ is also **Gaussian** $f(x)|Z \sim N(m(x), v^2(x))$

$$m(x) = \tilde{k}(x)^T(K + \Sigma)^{-1}\tilde{y}$$
$$v(x, x') = K(x, x') - \tilde{k}(x)^T(K + \Sigma)^{-1}\tilde{k}(x')$$

- $K_{ij} = K(x^i, x^j)$, $\Sigma = \text{diag}(\sigma^2(x^i))$, $k_i = K(x, x^i)$
GP Modeling

- Given the kernel, the posterior is in closed-form
- Lengthscale $\theta$ controls correlation decay = spatial smoothness of $f$
- Can incorporate a non-zero mean/trend
- Global consistency – converge to the truth as $N \to \infty$
- Fitted Matern-5/2 kernel
  \[ K(x, x'; \tau, \theta) = \tau^2 \left( 1 + \sqrt{5}\|x - x'\|_\theta + \frac{5}{3}\|x - x'\|^2_\theta \right) \cdot e^{-\sqrt{5}\|x - x'\|_\theta} \]
Fitting a GP

- Need to pick the kernel family
- Need to know the kernel hyperparameters – $\tau$, $\theta$’s, et cetera.
- **Solution I:** Use MLE (nonlinear optimization problem) or cross-validation
- **Solution II:** Specify priors and use a fully Bayesian method (requires MCMC)
- Need the sampling noise $\sigma^2(x)$ – use batching/replications to estimate
- GP is expensive compared to e.g LM; complexity is $O(N^3)$ for a design of size $N$
- We used DiceKriging package in R – off-the-shelf use
Batched Designs

- **Re-use** same site $x$ for multiple paths – like a MC forest
- *(pre)-Average* the pathwise payoffs: $\bar{y}(x) = \frac{1}{M} \sum_{i=1}^{M} y^{(i)}(x)$ where $y^{(1)}(x), \ldots, y^{(M)}(x)$ are $M$ independent replicates
- Sample variance estimator: $\tilde{\sigma}^2(x) := \frac{1}{M-1} \sum_{i=1}^{M} (y^{(i)}(x) - \bar{y}(x))^2$
- *(More proper is to train another metamodel for $\sigma(\cdot)$)*
- *(M can be chosen adaptively)*
- Plug-in $\tilde{\sigma}^2(x)/M$ for variance of $\bar{Y}(x)$. Only need to regress $(x, \bar{y})$’s
- When $M$ is big, can just *interpolate* averaged payoffs
Batched Kriging Metamodel for $T(t, \cdot)$

LHS design $\mathcal{Z}$ of size $N = 3000$ with $M = 100$ replications. The vertical “error” bars indicate the 95% quantiles of the simulation batch at $x$, while the dotted lines indicate the 95% credibility interval (CI) of the kriging metamodel fit.
Advantages of GP

- Adapts to the structure of the problem. Need to pick the kernel family but the rest is automatic
- Has an extensive “ecosystem”: local GP, treed GP, t-noise GP, et cetera
- Works well with sequential design by providing online local goodness-of-fit metrics; also is updateable
- Implemented in multiple R packages
- Clarifies the twin requirements of smoothing and interpolation
- Smooth \( \hat{f} \), can also set/get gradient estimates
- **Disadvantage**: slow; less analytically understood
Experimental Design

- Global design: \( \inf_{\mathcal{Z}: |\mathcal{Z}|=N} \mathbb{E}_{0,\mathcal{X}_0} \left[ \mathcal{L}(\hat{f}(\mathcal{Z}^{(N)}), f) \right] \)
- Above is NP-hard, so need heuristics
- **Idea 1:** need to learn \( f(x) \) over the input space \( \mathcal{X} \)
  - Space-filling designs – grid-based, low-discrepancy (Sobol), **LHS**
  - Loss is weighted according to \( \mathbb{P} \) – sample \( x_t^{1:N} \sim X_t \) from \( \mathbb{P} \)
    - (“empirical” design as originally proposed by Longstaff-Schwartz)
Experimental Design

Global design: \( \inf_{\mathcal{Z} : |\mathcal{Z}| = n} \mathbb{E}_{0, x_0} \left[ \mathcal{L}(\hat{f}(\mathcal{Z}^{(N)}), f) \right] \)

Above is NP-hard, so need heuristics

Idea 1: need to learn \( f(x) \) over the input space \( \mathcal{X} \)

Space-filling designs – grid-based, low-discrepancy (Sobol), LHS

Loss is weighted according to \( \mathbb{P} \) – sample \( x^1_t, \ldots, x^n_t \sim X_t \) from \( \mathbb{P} \) ("empirical" design as originally proposed by Longstaff-Schwartz)

Idea 2: The geometry of the design affects the local accuracy of the response surfaces

Denser design – smaller local error

Goal is to learn the sign of \( f(x) \)

\( \Rightarrow \) preferentially target regions where \( f(\cdot) \) changes signs

Adaptive designs
Proposed Designs

Based on $S_t | S_0$

Monte Carlo forest

Uniform in [30, 40]

Adaptive Grid
Space-Filling Designs

LHS $M = 20, N' = 150$  
LHS $M = 100, N' = 30$  
Emp $M = 100, N' = 30$

Three different designs for fitting a kriging metamodel of the continuation value for the 1-D Bermudan Put ($t = 0.6, T = 1$). Top panels show the fitted $\hat{T}(t, \cdot)$ and sites $x^{1:N'}$. Middle panels plot the corresponding surrogate standard deviation $\nu(x)$. Bottom panels display the loss metric $\ell(x; \mathcal{Z})$. 

Ludkovski  
RMC Kriging
Adaptive Design for Optimal Stopping

- Recall that aim to learn the sign of \( T(t, \cdot) \)
- Gradually grow \( \mathcal{Z}(k) \), \( k = N_0, \ldots, N \)
- Add new locations greedily according to acquisition function
  \[ x^{k+1} = \arg \max EI_k(x) \]
- Favor points where \( m^{(k)}(x) \approx 0 \) (close to zero-contour) or \( v^{(k)}(x) \)
  is large (reduce uncertainty)
- Loss from making the wrong stopping decision at \((t, x)\) is
  \[
  \ell(x; \mathcal{Z}) := \int_{\mathbb{R}} |y - h(t, x)| 1\{m(x) < h(t, x) < y \cup y < h(t, x) < m(x)\} \mathcal{M}_x(dy)
  \]
- Analytic integral if assume the posterior distribution is Gaussian
  \( \mathcal{M}_x \sim N(m(x), v^2(x)) \).
ZC-SUR Strategy

- **ZC-SUR (zero-contour stepwise uncertainty reduction):** Maximize stepwise expected reduction in local loss.

- **Analytic expression for** 
  \[ EI_k(x) := \mathbb{E}[\ell^{(k)}(x) - \ell^{(k+1)}(x) | Z^{(k)}, x^{k+1} = x] \]

- (Approximately) maximize \( EI_k(x) \); see Gramacy-L. (SIFIN 2015)

- Related ideas in machine learning/simulation optimization:
  - AL (Cohn et al ’96, MacKay ’92): Minimizing integrated posterior variance
  - EGO (Jones et al ’98): Learning \( \inf_x f(x) \)
  - Exploration/Exploitation trade-off (Auer et al ’02): UCB policies
  - Contour-finding: Ranjan et al ’08
  - SUR (Picheny et al ’10): Myopically maximizing loss reduction
Adaptive Designs

1-D Put

2-D Max Put

Adaptive designs. **Color-coded** according to $T(t, x)$; **red** contour indicates the stopping boundary.
Adaptive and LHS designs. Bermudan Put $e^{-rt}(100 - X_1)_+$ with a Heston stochastic volatility model. Both designs used $N = 10000$ simulations. Color-coded according to $T(t, x)$; red contour indicates the stopping boundary.
Effect of Design

- Probabilistic design: $x^n \sim p(\cdot, t|x_0, 0)$ (Classical approach)
- Highly sensitive to initial condition, often mis-aligned with $\mathcal{G}$
- Adaptive design gains are modest

<table>
<thead>
<tr>
<th>Design/Batch Size</th>
<th>$M = 4$</th>
<th>$M = 20$</th>
<th>$M = 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic</td>
<td>1.458 (0.002)</td>
<td>1.448 (0.003)</td>
<td>1.443 (0.006)</td>
</tr>
<tr>
<td>LHS</td>
<td>1.453 (0.002)</td>
<td>1.446 (0.004)</td>
<td>1.416 (0.033)</td>
</tr>
<tr>
<td>Sobol QMC</td>
<td>1.454 (0.002)</td>
<td>1.448 (0.002)</td>
<td>1.454 (0.002)</td>
</tr>
<tr>
<td>Sequential ZC-SUR</td>
<td>$N/A$</td>
<td>1.428 (0.004)</td>
<td>1.439 (0.005)</td>
</tr>
</tbody>
</table>

Performance of different DoE approaches to RMC for the 2-D Bermudan Put. The table reports $\hat{V}(0, X_0)$ and its Monte Carlo (StDev). All methods utilize $|Z_t| = 3000$. Results are based on averaging 100 runs of each method, and evaluating $\hat{V}(0, X_0)$ on a fixed out-of-sample database of $N_{out} = 100,000$ scenarios. For comparison, LSMC-BW11 algorithm yielded estimates of $\hat{V}^{BW11}(0, X_0) = 1.431$ with $N = 10,000$ and $\hat{V}^{BW11}(0, X_0) = 1.452$ with $N = 50,000$. 
## Simulation Savings

<table>
<thead>
<tr>
<th>Method</th>
<th>( \hat{V}(0, X_0) ) (StDev.)</th>
<th>#Sims</th>
<th>Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2D Max call</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSMC BW11</td>
<td>7.89 (0.023)</td>
<td>360 \cdot 10^3</td>
<td>4.0</td>
</tr>
<tr>
<td>LSMC BW11</td>
<td>7.95 (0.015)</td>
<td>1125 \cdot 10^3</td>
<td>7.7</td>
</tr>
<tr>
<td>Krig + LHS</td>
<td>7.85 (0.073)</td>
<td>59 \cdot 10^3</td>
<td>1.2</td>
</tr>
<tr>
<td>Krig + LHS</td>
<td>7.90 (0.037)</td>
<td>117 \cdot 10^3</td>
<td>5.2</td>
</tr>
<tr>
<td>Krig + SUR</td>
<td>7.91 (0.024)</td>
<td>102 \cdot 10^3</td>
<td>15.6</td>
</tr>
<tr>
<td>Krig + SUR</td>
<td>7.95 (0.05)</td>
<td>246 \cdot 10^3</td>
<td>28.7</td>
</tr>
<tr>
<td><strong>3D Max Call</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSMC BW11</td>
<td>11.07 (0.01)</td>
<td>2.7 \cdot 10^6</td>
<td>22</td>
</tr>
<tr>
<td>Krig + LHS</td>
<td>11.09 (0.02)</td>
<td>0.48 \cdot 10^6</td>
<td>27</td>
</tr>
<tr>
<td>Krig + SUR</td>
<td>11.05 (0.02)</td>
<td>0.51 \cdot 10^6</td>
<td>161</td>
</tr>
<tr>
<td><strong>5D Max Call</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSMC BW11</td>
<td>16.32 (0.02)</td>
<td>5.76 \cdot 10^6</td>
<td>87</td>
</tr>
<tr>
<td>Krig + LHS</td>
<td>16.32 (0.03)</td>
<td>0.81 \cdot 10^6</td>
<td>317</td>
</tr>
<tr>
<td>Krig + SUR</td>
<td>16.33 (0.02)</td>
<td>0.85 \cdot 10^6</td>
<td>952</td>
</tr>
</tbody>
</table>

Comparison of RMC methods for different max-Call models. Results are averages across 100 runs of each algorithm, with third column reporting the corresponding standard deviations of \( \hat{V}(0, X_0) \). Time is based on running the R code on a 1.9 MHz laptop with 8Gb of RAM. The BW11 method used \( 10^2 \) partitions for \( d = 2 \), \( 5^3 \) partitions for \( d = 3 \) and \( 4^5 \) partitions for \( d = 5 \).
Adaptive Design: Is It Worth It?

- Significant **memory savings**, increased computation time
- Kriging metamodel is an **updateable** representation of $\mathbb{S}$ – can be used “**anytime**” or with adaptive termination
- Outputs empirical self-assessment to monitor performance
- New connections to **statistics/machine learning**
- Sequential design is intermediate step – can sacrifice accuracy (e.g. use one regression method during seq design and another for final metamodel)
- Or can use other **importance sampling** ideas (build a rough fit, then refine)
Sign Classification

- Convert pathwise rewards into 0/1 labels:
  \[ z_t^n = I(h(\tau_{t+1}, x_{\tau_{t+1}}^n) > h(t, x_t^n)) \]
- Let \( p(x) = P(Z(x) = 1) \). Then \( S_t \sim \{ p(x) > 0.5 \} \).
- Build a statistical model for \( p(x) \) and hence approximate \( S \).
  (Picazo 2002)
- Tools: Logistic regression; support vector machines.
- Probit GP model: \( p(x) = \Phi(\tilde{f}(x)) \) where \( \tilde{f} \sim GP(m(x), v^2(x)) \)
- Likelihood log \( \log p(\tilde{f}|x, z) \propto \frac{1}{2} \tilde{f}^T K^{-1} \tilde{f} + \sum_i \log \Phi((2Z_i - 1)\tilde{f}_i) \)
GP Classification

Classification Pros/Cons

- Classification modifies the statistical “noise”; smoothes non-Gaussian $\varepsilon$ and heteroskedasticity.
- Note that $p(x) = 0.5$ is when the median of $Y$ is zero. When $Y$ is skewed, median $\neq$ mean. Significant concern in financial applications where skew is very severe (ATM: usually pathwise payoff is less than immediate one, but sometimes it’s MUCH bigger).
- There is necessarily loss of information in discarding the magnitude of $Y$ when switching to $Z$.
- Better targets the loss function.
- Directly models the stopping boundary (eg SVM: adaptive representation of $\partial\mathcal{G}$ as a collection of hyperplanes).
- Natural approach for sequential design construction?
Next Steps

- Structured regression (with X. Lyu)
- Root-finding (with S. Rodriguez)
- Multiple responses (with R. Hu)
- Related control problems
- Common library of examples for benchmarking
Next Steps

- Structured regression (with X. Lyu)
- Root-finding (with S. Rodriguez)
- Multiple responses (with R. Hu)
- Related control problems
- Common library of examples for benchmarking

THANK YOU!
References I


References II


