Likelihood-free Inference and Optimization using Stochastic Gradient Approximations

Abstract

In this paper we apply recent techniques from Bayesian inference that use gradient information to sample efficiently from the true posterior distribution to the *likelihood-free* or *approximate Bayesian computation* (ABC) setting. To do this for ABC we adopt a *gradient-free* stochastic approximation algorithm by Spall [**?**]. Together these algorithms provide both optimization and inference for likelihood-free models as the algorithm ABC-SGLD transitions from optimization to sampling. We demonstrate ABC-SGLD on problems where the true gradient information is known and on challenging ABC simulators.

1 INTRODUCTION

- Arguably the two most useful procedures in simulation-based science are optimization and Bayesian inference.
- Optimization can take the form of simple grid-search to sophisticated techniques like factorial design [**?**], Bayesian experiment design [**?**], (mention others).
- Recently, Bayesian optimization techniques have shown success in optimizing very expensive black-box simulators [**?**].
- The main approach of Bayesian inference of simulator parameters is ABC. ABC is largely based a few sampling algorithms: SMC and MCMC.
- Though gradients are not directly computable for simulators, they can be computed analytically by using finite differences (see [**?**]). This quickly becomes infeasible for large p problems. There is however an alternative stochastic approximation algorithm by Spall [**?**] that requires only 2 simulation calls independent of p.
- By using this approximation, gradient-based algorithms can be adopted for simulators: for optimization, using analogous stochastic gradient descent algorithms and for Bayesian inference using Langevin dynamics [**?**].
- Recently, the SGLD [**?**] algorithm has efficiently combined optimization with inference using ideas from SGD with Langevin dynamics.
- This paper describes ABC-SGLD.

2 GRADIENTS FROM FORWARD SIMULATIONS

2.1 Robbin's Monro: SGD

2.2 Spall's method: SPSA

In the gradient-free setting, Spall [**?**] provides a stochastic approximate to the true gradient using only 2 forward simulations (function evaluations). This is in contrast to multivariate finite-difference stochastic approximation FDSA [**?**] requiring $2p$ evaluations.

The gradient estimate is

$$\hat{g}_t\left(\boldsymbol{\theta}_t\right) = \begin{bmatrix} \frac{y_t^+ - y_t^-}{2c_{t1}\Delta_{t1}} \\ \vdots \\ \frac{y_t^+ - y_t^-}{2c_{tp}\Delta_{tp}} \end{bmatrix} \tag{1}$$

where c_{tp} is a step-size that is usually constant for all dimensions p, but can be different (as shown in this case); $\Delta_{tp} \in -1, +1$ is a *perturbation mask* (called symmetric Bernouilli variables by Spall), i.e. $\Delta_{tp} \sim 2 * \text{Bernouilli}(0.5) - 1$; and $y_t^{\pm}$ are function evaluations:

$$y_t^+ = L\left(\boldsymbol{\theta} + c_t\Delta_t\right) \tag{2}$$

$$y_t^- = L\left(\boldsymbol{\theta} - c_t\Delta_t\right) \tag{3}$$

A way of reducing the noise in the gradient is by averaging

over q draws of Δ_t:

$$\hat{g}_t(\boldsymbol{\theta}_t) = \frac{1}{q}\sum_{j=1}^{q}\hat{g}_t^j(\boldsymbol{\theta}_t) \tag{4}$$

where for each j new perturbation masks are drawn.

Another variance reduction technique called the method of *common random numbers* (CRN) [**?**] sets a common seed for the random number generator (RNG) for both calls to the simulator. This technique can remove the effect of the simulator noise in the gradient estimate, leaving on the randomness of the perturbation masks as the source of noise. Consider using SPSA instead of SGD: the CRN technique is equivalent of using the same batch of data vectors to evaluate the log-likelihood which is a sensible approach.

2.2.1 Variations

- Varying c_q: use a different per repeat. Seems to converge faster. What is the correct noise process? Right now trying this: flip coin, if heads perturb $c* = 1 + U(0,1)$, else perturb $c/ = 1 + U(0,1)$.

3 LANGEVIN DYNAMICS

4 STOCHASTIC-GRADIENT LD

$$\begin{aligned}
h &= \frac{N}{n}\sum_{i=1}^{n}\nabla\log p(\mathbf{x}_i|\boldsymbol{\theta}) - \sum_{i=1}^{N}\nabla\log p(\mathbf{x}_i|\boldsymbol{\theta}) && (5)\\
&= \frac{N}{n}\sum_{i=1}^{n} s_i && (6)\\
s_i &= \nabla\log p(\mathbf{x}_i|\boldsymbol{\theta}) + \frac{1}{N}\nabla p(\boldsymbol{\theta}) && (7)\\
\mathrm{V}(h) &= \frac{N^2}{n^2}\sum_{i=1}^{n}\mathrm{Var}(s_i) && (8)\\
&= \frac{N^2}{n^2}\sum_{i=1}^{n}\frac{1}{n}\left(\sum_{j=1}(s_i-\bar{s})^2\right) && (9)\\
&= \frac{N^2}{n^2}\sum_{i=1}^{n} V_s && (10)\\
&= \frac{N^2}{n^2} n V_s && (11)\\
&= \frac{N^2}{n} V_s && (12)
\end{aligned}$$

$$\begin{aligned}
\mathrm{Var}(\boldsymbol{\theta}) &= \frac{\epsilon^2}{4}\mathrm{Var}(h) + \epsilon && (13)\\
\frac{\epsilon^2}{4}\mathrm{Var}(h) &<< \epsilon && (14)\\
\frac{\epsilon N^2}{4n} V_s &<< 1 && (15)
\end{aligned}$$

5 SGLD-ABC

5.1 ABC Cost Function

The ABC likelihood is proportional to the convolution between a kernel around the observation statistics $\mathbf{y}$ and the generator of pseudo-statistics $\mathbf{x}$, i.e. the simulator. Since pseudo-statistics can only be generated, we resort to a Monte Carlo estimate for the ABC likelihood based on S draws from the simulator using the same parameter vector $\boldsymbol{\theta}$ but different random seeds:

$$\begin{aligned}
\pi(\mathbf{y}|\boldsymbol{\theta}) &= \int \pi_{\boldsymbol{\epsilon}}(\mathbf{y}|\mathbf{x})\,\pi(\mathbf{x}|\boldsymbol{\theta})\,d\mathbf{x} && (16)\\
&\approx \frac{1}{S}\sum_{s=1}^{S}\pi_{\boldsymbol{\epsilon}}\left(\mathbf{y}|\mathbf{x}^{(s)}\right) && (17)
\end{aligned}$$

where $\mathbf{x}^{(s)} = \pi(\mathbf{x}|\boldsymbol{\theta},\omega_s)$. Note we have passed a unique random seed ω_s into the simulator, making $\mathbf{x}^{(s)}$ a deterministic function of the simulator with parameters $\{\boldsymbol{\theta},\omega_s\}$. It will be useful to rewrite this deterministic simulator function as f: $\mathbf{x}^{(s)} = f(\boldsymbol{\theta},\omega_s)$, allowing us to rewrite the likelihood as

$$\pi(\mathbf{y}|\boldsymbol{\theta}) \approx \frac{1}{S}\sum_{s=1}^{S}\pi_{\boldsymbol{\epsilon}}(\mathbf{y}|f(\boldsymbol{\theta},\omega_s)) \tag{18}$$

For optimization purposes we can search for the maximum a posteriori (MAP) estimator $\hat{\boldsymbol{\theta}}$ using the log-joint distribution $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}}\mathcal{MAP}$, where $\mathcal{MAP}$ is

$$\begin{aligned}
\mathcal{LJ} &= \log\pi(\boldsymbol{\theta}) + \log\pi(\mathbf{y}|\boldsymbol{\theta}) && (19)\\
&\approx \log\pi(\boldsymbol{\theta}) + \log\sum_{s=1}^{S}\pi_{\boldsymbol{\epsilon}}(\mathbf{y}|f(\boldsymbol{\theta},\omega_s)) - \log S && (20)
\end{aligned}$$

we can also maximize the lower-bound of $\mathcal{LJ} \geq \mathcal{LB}$, where

$$\mathcal{LB} \approx \log\pi(\boldsymbol{\theta}) + \sum_{s=1}^{S}\log\pi_{\boldsymbol{\epsilon}}(\mathbf{y}|f(\boldsymbol{\theta},\omega_s)) \tag{21}$$

which will be more convenient for optimization for typical kernels (i.e. π_ϵ is Gaussian).

The simplest optimization is to set the objective function C to be $-\mathcal{LJ}$ or $-\mathcal{LB}$ and compute stochastic gradients using 2SPSA with q repeats (negative log-likelihood because we will assume minimization):

$$\frac{\partial C}{\partial\boldsymbol{\theta}} = \frac{1}{q}\sum_{j=1}^{q}\frac{C(\boldsymbol{\theta}+c\Delta_j) - C(\boldsymbol{\theta}-c\Delta_j)}{2c\Delta_j} \tag{22}$$

or 1SPSA

$$\frac{\partial C}{\partial\boldsymbol{\theta}} = \frac{1}{q}\sum_{j=1}^{q}\frac{C(\boldsymbol{\theta}+c\Delta_j) - C(\boldsymbol{\theta})}{c\Delta_j} \tag{23}$$

which is less accurate but is less computation if $C(\boldsymbol{\theta})$ is computed at each step (TODO: possible algorithm). TODO: rewrite $\frac{\partial C}{\partial\boldsymbol{\theta}}$ as $\hat{g}$ or similar.

5.2 Special Case: Gaussian Kernels

When the kernel is known we can use the chain rule to compute the gradients with the first part from the kernel and the second from the variation of the simulator pseudo-statistics, whose gradient can be estimated using SPSA. Expanding the likelihood over J statistics, the lower bound on the loglikelihood becomes:

$$\begin{aligned}
\hat{L} &= \sum_s \log \pi_\epsilon \left(\mathbf{y}|f(\boldsymbol{\theta},\omega_s)\right) && (24)\\
&= \sum_s \log \prod_{j=1}^{J} \pi_{\epsilon_j} \left(y_j|f_j(\boldsymbol{\theta},\omega_s)\right) && (25)\\
&= \sum_s \sum_j \log \pi_{\epsilon_j} \left(y_j|f_j(\boldsymbol{\theta},\omega_s)\right) && (26)\\
&= \sum_s \sum_j -\frac{1}{2}\frac{\left(y_j - f_j(\boldsymbol{\theta},\omega_s)\right)^2}{\epsilon_j^2} - Z_j && (27)\\
& && (28)
\end{aligned}$$

The gradient:

$$\begin{aligned}
\nabla \hat{L} &= \sum_s \sum_j \frac{\left(y_j - f_j(\boldsymbol{\theta},\omega_s)\right)}{\epsilon_j^2} \cdot \frac{\partial f_j(\boldsymbol{\theta},\omega_s)}{\partial \boldsymbol{\theta}} && (29)\\
&= \sum_s \sum_j e_{js} \nabla_{js} && (30)
\end{aligned}$$

The first part e_{js} is analytic, the second part ∇_{js} requires estimation by 2SPSA:

$$\hat{\nabla}_{js} = \frac{1}{q}\sum_{r=1}^{q} \frac{f_j(\boldsymbol{\theta} + c\Delta_r, \omega_s) - f_j(\boldsymbol{\theta} - c\Delta_r, \omega_s)}{2c\Delta_r} \quad (31)$$

or 1SPSA

$$\hat{\nabla}_{js} = \frac{1}{q}\sum_{r=1}^{q} \frac{f_j(\boldsymbol{\theta} + c\Delta_r, \omega_s) - f_j(\boldsymbol{\theta}, \omega_s)}{c\Delta_r} \quad (32)$$

the benefit of 1SPSA in this case is that the first gradients e_{js} can be computed from $f_j(\boldsymbol{\theta}, \omega_s)$.

5.3 SGD, AdaGrad, RMSProp, AdaM

With our stochastic gradient in hand, we can make use of robust stochastic gradient rules, including recent advances from machine learning.

5.3.1 SGD

The basic update rule is at time t for minimization is

$$\Delta\boldsymbol{\theta} = -\alpha_t \hat{g}_t \quad (33)$$

with conditions $\sum_{t=1}^{\infty} \alpha_t = \infty$ and $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$ required for convergence. An additional momentum parameter β_1 can be added to improve convergence

$$\begin{aligned}
g_m &= \beta_1 g_m + (1-\beta_1) * \hat{g}_t && (34)\\
\Delta\boldsymbol{\theta} &= -\alpha_t g_m && (35)
\end{aligned}$$

5.3.2 AdaGrad

AdaGrad uses the total cumulative element-wise square of the gradients to automatically both decrease α_t and scale the learning rates for each dimension:

$$\begin{aligned}
g_m &= \beta_1 g_m + (1-\beta_1) * \hat{g}_t && (36)\\
v_m &= \sum_{t'=1}^{t} {\hat{g}_t}^2 && (37)\\
\Delta\boldsymbol{\theta} &= -\alpha_t g_m / \sqrt{v_m} && (38)
\end{aligned}$$

5.3.3 RMSProp

$$\begin{aligned}
g_m &= \beta_1 g_m + (1-\beta_1) * \hat{g}_t && (39)\\
v_m &= \beta_2 v_m + (1-\beta_2) * {\hat{g}_t}^2 && (40)\\
\Delta\boldsymbol{\theta} &= -\alpha_t g_m / \sqrt{v_m} && (41)
\end{aligned}$$

5.3.4 AdaM

$$\begin{aligned}
g_m &= \beta_1 g_m + (1-\beta_1) * \hat{g}_t && (42)\\
v_m &= \beta_2 v_m + (1-\beta_2) * {\hat{g}_t}^2 && (43)\\
\gamma &= \frac{\sqrt{1-(1-\beta_2)^t}}{(1-(1-\beta_1)^t)} && (44)\\
\Delta\boldsymbol{\theta} &= -\alpha_t \gamma g_m / \sqrt{v_m} && (45)
\end{aligned}$$

6 Applying SGLD to ABC

There are two main sources of randomness in the ABC gradient computation: the random perturbation mask and the random seed of the simulator model, which controls the simulator noise. This noise can be eliminated by using common random numbers [**?**]. On the other hand, it is not clear that this is the best strategy, as the noise from the masks at a fixed seed may be higher than

$$\begin{aligned}
V_{\hat{g}_t^j(\boldsymbol{\theta}_t)} &= \frac{1}{q-1}\sum_{j=1}^{q} (\hat{g}_t^j(\boldsymbol{\theta}_t) - \hat{g}_t(\boldsymbol{\theta}_t))^2 && (46)\\
V_{\hat{g}_t(\boldsymbol{\theta}_t)} &= \frac{1}{q-1} V_{\hat{g}_t^j(\boldsymbol{\theta}_t)} && (47)\\
\frac{\epsilon^2}{4} V_{\hat{g}_t(\boldsymbol{\theta}_t)} &= \frac{\epsilon^2}{4q} V_{\hat{g}_t^j(\boldsymbol{\theta}_t)} && (48)\\
&<< \epsilon && (49)\\
\frac{\epsilon}{4(q-1)} V_{\hat{g}_t^j(\boldsymbol{\theta}_t)} &<< 1 && (50)
\end{aligned}$$

Check for $g(\boldsymbol{\theta}) + h(\boldsymbol{\theta}) + \eta$ that $h(\boldsymbol{\theta}) + \eta \approx \eta$

Also $\theta_{t+1} = \theta_t + \frac{\epsilon}{2}\hat{g}_t(\boldsymbol{\theta}_t) + \sqrt{\epsilon}\mathcal{N}(0,1)$

6.1 Statistical Tests for Gradients

From Byrd:

$$\frac{||V_{\hat{g}_t^j(\boldsymbol{\theta}_t)}||_1}{q} \leq r^2||\hat{g}_t||_2^2 \tag{51}$$

$$\hat{q} = \frac{||V_{\hat{g}_t^j(\boldsymbol{\theta}_t)}||_1}{r^2||\hat{g}_t||_2^2} \tag{52}$$

TODO:

- Check that q gradients are normal. Especially as q increases.
- Merge Byrd's test for $\hat{q}$ and condition on SGLD-ABC.
- Variance check for increasing q can also be used to stop optimization since the variance is less than the noise in the gradients.

7 EXPERIMENTS

7.1 Logistic Regression with Stochastic Gradients

7.2 Optimization of Blowfly Dynamics

7.2.1 Using AdaM

7.2.2 Using AdaM with "SGLD" injected noise

7.3 Likelihood-free Inference of Blowfly Dynamics

7.4 Other Useful ABC problem

8 DISCUSSION

9 CONCLUSION

References

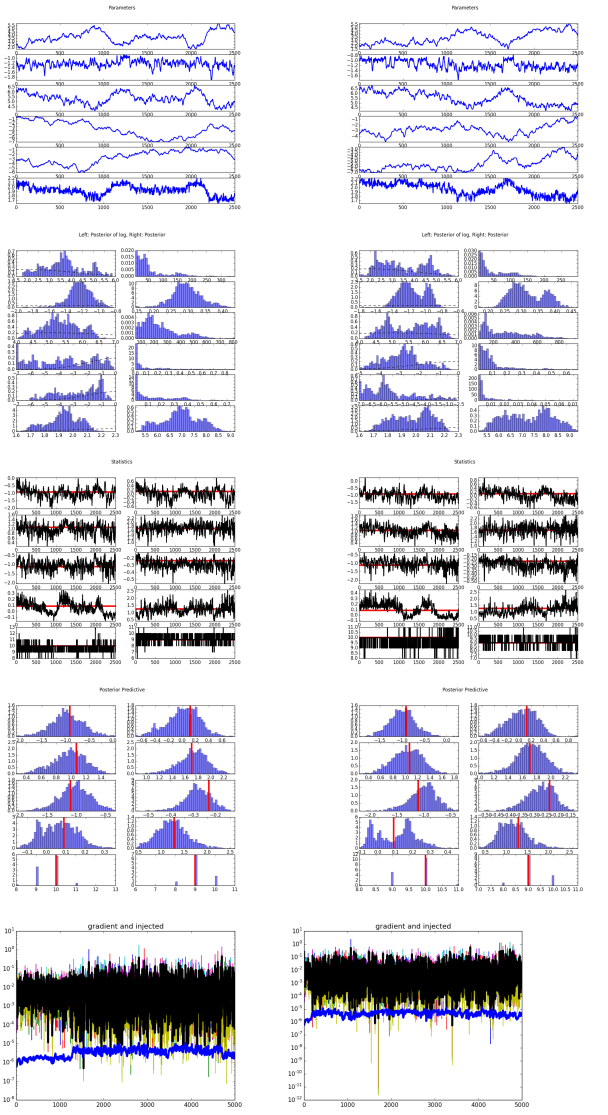

Figure 1: Optimizing Blowfly using $q = 5$, $\epsilon = 0.01$, $\beta_1 = 0.75$,$\beta_2 = 0.9$,$\alpha = 0.1$ and $c = 1$. Left: Optimization using AdaM, Right: "SGLD" with estimated noise.