# Approximate Bayesian Computations via Sufficient Dimension Reduction

Harris Quach

November 28, 2018

# Approximate Bayesian Computation: Context

- Observed sample of size $n$, $y_{obs} \in \mathbb{R}^n \sim f_\theta$, prior $\pi(\theta)$; we want draws from posterior $\pi(\theta|y_{obs}) \propto \pi(\theta) f(y_{obs}|\theta)$
- Problem: $f(y|\theta)$ is intractable - computationally expensive, no analytic form, etc. **But** we can simulate from the $f_\theta$.
- Idea: To sample from posterior, find $\theta$ that generate simulations $y_{sim}$ matching $y_{obs}$, i.e. $y_{sim} = y_{obs}$
- Matching $y_{sim}$ to $y_{obs}$ difficult if $n$ is large, especially if $y$ is continuous.

# ABC: Role of Sufficiency

- Easier if we have a lower-dimensional summary statistic $\varphi = \varphi(y)$; ideally, $\varphi(y)$ is a sufficient statistic; $\varphi(y)$ being informative on $\theta$ also works.
- Easier if instead of matching, we settle for close enough: $\rho(\varphi(y_{sim}), \varphi(y_{obs})) < \varepsilon$ for $\rho$ a metric, $\varepsilon > 0$

**Algorithm 1:** ABC

**Given:** proposal $g(\theta)$; a summary statistic $\varphi(\cdot)$; a metric $\rho$ with some tolerance $\varepsilon$; your acceptance rule as a function of closeness

1. Draw $\theta_{sim} \sim g(\theta)$ for $sim = 1, ..., S$
2. Draw $y_{sim} \in \mathbb{R}^n \sim f_{\theta_{sim}}$ for $sim = 1, ..., S$
3. Accept $\theta_{sim}$ according to your rule depending on closeness, e.g $\rho(\varphi(y_{sim}), \varphi(y_{obs})) < \varepsilon$

**Result:** $S$ Draws from a posterior that approximates $\pi(\theta|y_{obs})$

$$\pi(\theta|y_{obs}) \approx \pi(\theta|\varphi_{obs}) \approx \pi_{ABC}(\theta|\varphi_{obs})$$

# Sufficient Dimension Reduction: Context

- But what if we have no idea about $\varphi$?
- We have $\theta \in \mathbb{R}^B$, $Y \in \mathbb{R}^{n \times B}$
- Objective: Find an transformation $\varphi(Y)$ such that

$$\theta \perp\!\!\!\perp Y|\varphi(Y), \quad or \quad P(\theta|Y) = P(\theta|\varphi(Y))$$

- Finding an informative summary $\varphi = \varphi(y)$ such that $\pi(\theta|y) = \pi(\theta|\varphi(Y))$ is a sufficent dimension reduction problem!

# Sufficient Dimension Reduction: Heuristics and Methods

- Being informative means $\varphi(Y)$ explains variation in $\theta$
- Work with matrices like $\Lambda_{sdr} = E($"Variation between $\theta$ and $Y$" $)$;
- The SDR methods we consider are will generally produce estimated functions of the form:

$$\hat{\varphi}(Y) = eigen.vec_d(\hat{\Lambda}_{sdr})(\text{``}\widehat{VCov_Y}\text{''})^{-1}\text{``}Y\text{''}$$

# Estimating Sufficient Statistic via Simulation

**Algorithm 2:** Estimating Sufficient Statistic for ABC via Simulation

**Given:** proposal $g(\theta)$;

Draw $\theta_{sim} \sim g(\theta)$ for $sim = 1, \ldots, B$

For each $\theta_{sim}$: Draw $y_{sim}^{(r)} \in \mathbb{R}^n \sim f_{\theta_{sim}}$ for $r = 1, \ldots, R$

**begin** For each $r = 1, \ldots, R$:

  Let $\theta^{(r)} = (\theta_1^r, \ldots, \theta_B^r) \in \mathbb{R}^B$ and $Y^{(r)} = (y_1^r, \ldots, y_B^r) \in \mathbb{R}^{n \times B}$

  Estimate $\hat{\Lambda}_{sdr}^{(r)}$

**end**

Construct $\hat{\Lambda}_{sdr} = \frac{1}{R} \sum_{r=1}^{R} \hat{\Lambda}_{sdr}^{(r)}$; and

$\hat{\varphi}(y) = eigen.vec_d(\hat{\Lambda}_{sdr})(\text{“}\widehat{VCov_Y}\text{”})^{-1}\text{“}y\text{”}$

**Output:** $\hat{\varphi}$: an estimated sufficient statistic for $\theta$

# Example: AR(1) in Ghosh & Zhong (2016)

- For $t = 1, ..., n$, $n = 100$, $\sigma = 0.5$, $B = 1000$;

$$y_{t+1} = \theta y_t + \eta_t, \quad \eta_t \sim N(0, \sigma^2), \quad \theta \sim Unif(-1, 1) \implies \theta | y, \sigma \sim N\left(\frac{\sum_2^n y_t y_{t-1}}{\sum_2^n y_{t-1}^2}, \frac{\sigma^2}{\sum_2^n y_{t-1}^2}\right)$$

**Algorithm 3:** ABC with SDR for AR(1)

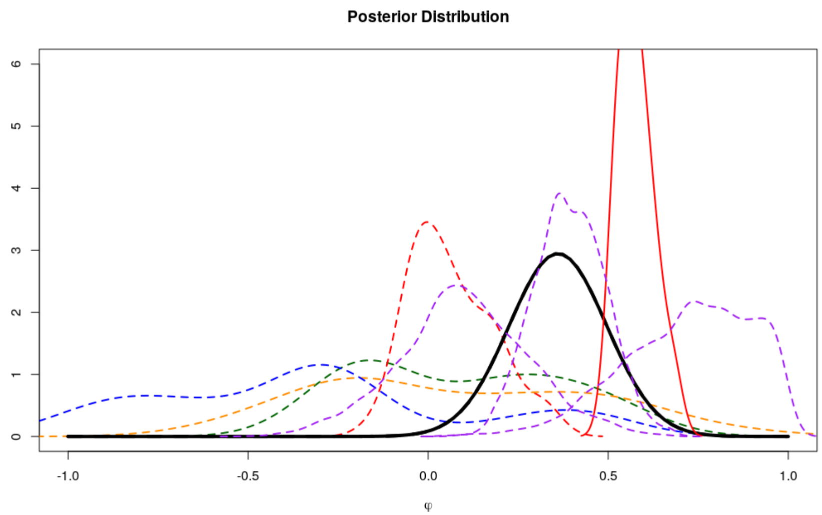
**Given:** proposal $g(\theta)$; estimated summary statistic $\hat{\varphi}(\cdot)$ ; a metric $\rho$ with some tolerance $\varepsilon$; your acceptance rule as a function of closeness

1 Draw $\theta_{sim} \sim g(\theta)$ for $sim = 1, ..., S$

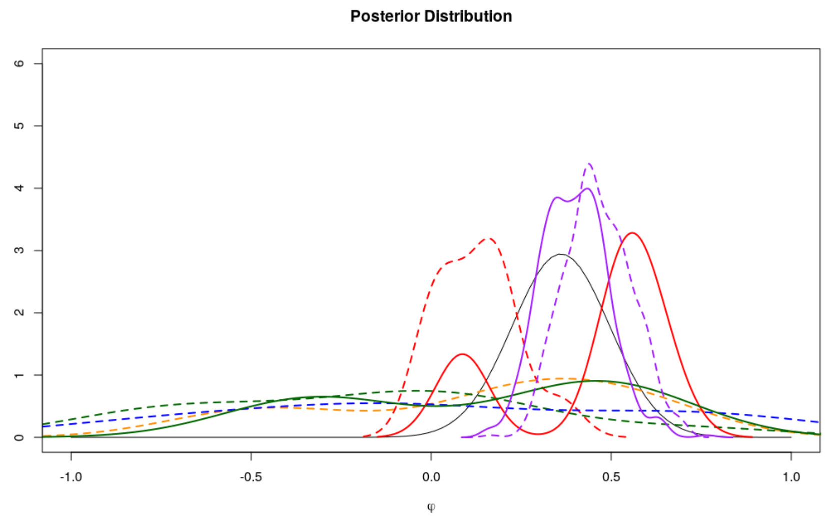
2 Accept $\theta_{sim}$ according to your rule depending on closeness, e.g $\rho(\hat{\varphi}(y_{sim}), \hat{\varphi}(y_{obs})) < \varepsilon$

**Output:** $S$ Draws from $\pi_{ABC}(\theta | \hat{\varphi}_{obs}) \approx \pi(\theta | \varphi(y_{obs}) \approx \pi(\theta | y_{obs})$

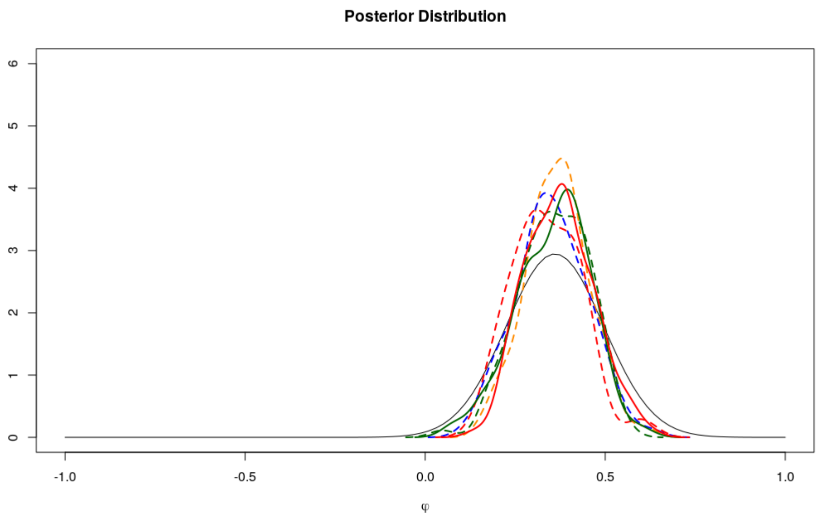
# **True Posterior** vs ABC via ( DR, SIR, SAVE, IHT dashed), **GSIR**, **GSAVE**



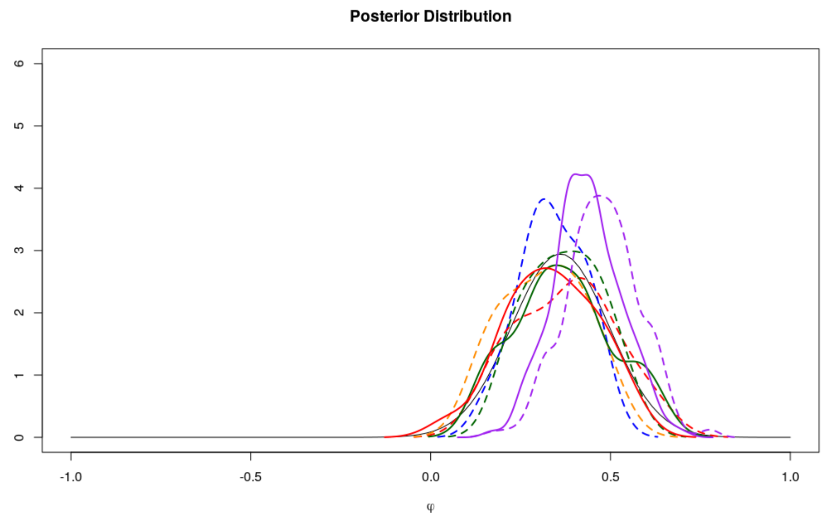

$g = \pi$; $R = 30$ no rep; $B = 1000$, $S = 1000$




$g = \pi$; $R = 200$; $S = 1000$; $MCMC - TruncNorm$




$g = TruncNorm$; $R = 100$; $S = 1000$




$g = TruncNorm$; $R = 200$; $S = 1000$

# Conclusion

- Averaging to estimate $\Lambda_{sdr}$ enables use of SDR by speeding up computation;
- Repeated drawing for each $\theta_{sim}$ improves effectiveness of non-linear SDR;
- SDR provides an automated way to construct useful summary statistics for ABC;
- Need to develop better SDR implementation within an MCMC ABC framework;

# Revisiting Gradient-based Meta-learning Optimization via Stochastic Composition Optimization

Huaxiu Yao

College of Information Science and Technology

28th Nov 2018

# Introduction: Background (Meta-learning)

## Supervised Learning

Input: $\mathbf{x}$;Output: $\mathbf{y}$; Data: $(\mathbf{x}, \mathbf{y})_i$

Relation: $\mathbf{y} = f(\mathbf{x}; \theta)$

## Meta-Supervised Learning

Input: $\mathcal{D}_{train}$, $\mathbf{x}_{test}$; Output: $\mathbf{y}_{test}$; Data: $\mathcal{D}_i = (\mathbf{x}, \mathbf{y})_j$

Relation: $\mathbf{y} = f(\mathcal{D}_{train}, \mathbf{x}_{test}; \theta)$

**Why it is so important?**

Reduces the problem to the design & optimization of f.

**Applications**

1. **Learning to optimization**: learn how to design the hyperparameters (e.g., step size)

2. **Few-shot Learning**: learning how to classify images with a few samples.

# Gradient-based Meta-learning

**Key Idea**: Train over many tasks, to learn parameter vector $\theta$ that transfers [Finn et al. 2017]

$\theta$: parameter vector being meta-learned (i.e., the learnable parameters of $f$)

$\phi_{\mathbf{i}}^*$: optimal parameter vector for task i

**Loss Function (one gradient as exemplarity)**:

$$\min_{\theta} \sum_{task\ i} \mathcal{L}_{test}^{i}(\theta - \alpha \nabla_\theta \mathcal{L}_{train}^{i}(\theta))$$

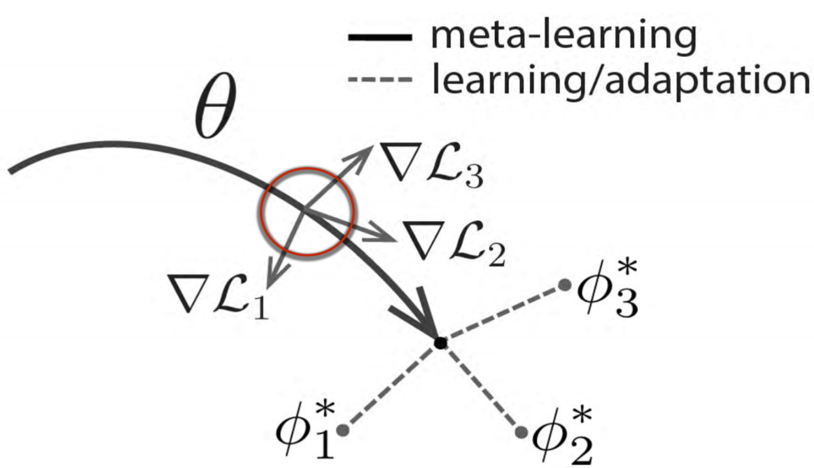



Figure 1: Meta-learning

**How to solve this optimization problem?**

Stochastic Gradient Descent (e.g., Adam)?

Simply using stochastic gradient descent may introduce biased estimation.

# Challenge: How to Optimize the Loss Function

**Loss Function**:

$$\min_{\theta} \sum_{task\ i} \mathcal{L}^{i}_{test}(\theta - \alpha\nabla_{\theta}\mathcal{L}^{i}_{train}(\theta))$$

The loss function can be regarded as a nested function, which can be revised as:

$$\min_{\theta} \sum_{task\ i} \mathcal{L}^{i}_{test}(\theta - \alpha\nabla_{\theta}\mathcal{L}^{i}_{train}(\theta)) \Rightarrow \mathbb{E}[g_1(\mathbb{E}[g_2(\theta)])]$$

$$g_1(\theta) = \mathcal{L}_{test}(\cdot);\ g_2(\theta) = \theta - \alpha\nabla_{\theta}\mathcal{L}_{train}(\theta)$$

where $\mathcal{L}_{train}(\theta)$ is a complex function and the variable is $\theta$.
The gradient of loss function is:

$$\nabla G(\theta) = \nabla g_2(\theta)\nabla g_1(g_2(\theta))$$

**Challenge**: $g_2(\theta)$ is unknown with finite sample oracles (SO).

## Solution: Stochastic Composition Optimization

Optimality condition of problem (assuming that the problem is convex) is [Wang et al. 2017]:

$$\nabla G(\theta^*)'(\theta - \theta^*) \geq 0$$

The unbiased sample of $\nabla G$ is difficult to obtain. We introduce a new function $z$ and the optimality condition can be:

$$\begin{aligned} &(\nabla g_2(\theta)\nabla g_1(z))'(\theta - \theta^*) \geq 0 \\ &z = g_2(\theta) \end{aligned}$$

For a given $(\theta, z)$, we can get unbiased results.
**Multi-level Optimization**: we can also easily extend to multi-level optimization, which is (assuming there are T steps):

$$\begin{aligned} &(\nabla g_T(\theta)\nabla g_{T-1}(z_{T-1})\cdots\nabla g_1(z_1))'(\theta - \theta^*) \geq 0 \\ &z_{T-1} = g_T(\theta) \\ &z_1 = g_2(z_2) \end{aligned}$$

# Solution: Stochastic Composition Optimization

**Stochastic Composition Optimization** The formulation is:

$$\min_{\theta} \mathbb{E}[g_1(\mathbb{E}[g_2(\theta)])]$$

**Input: SO, K, stepsize** $\{\alpha_k\}_{k=1}^{K}$, $\{\beta_k\}_{k=1}^{K}$, **data,** $z_0$

1: **for** $k = 1$ to $K$ **do**

2: Query the SO to obtain $\nabla g_2(\theta_k)$, $g_2(\theta_k)$, $g_1(y_{k+1})$

Update $y_{k+1} = (1 - \beta_k)y_k + \beta_k g_2(\theta_k)$

Update $x_{k+1} = x_k - \alpha \nabla g_2(\theta_k) \nabla g_1(y_{k+1})$

3: **end for**

**Algorithm 1:** Pseudocode for Stochastic Composition Gradient Descent

# Experiments: Synthetic Dataset

**Dataset Description**: We generate the task from a familiy functions. The base function is:

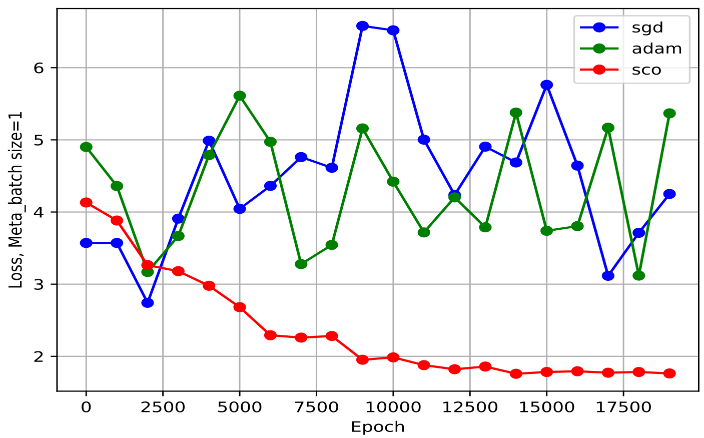
$$y = A_1 sin(wx + b_1) + A_2 sin(wx + b_2)$$
$$A_1 \sim U[1.0, 5.0], A_2 \sim U[-1.0, 2.0], b_1 \sim U[0, 2\pi], w \sim U[0.5, 2.0]$$

Each composition represents one task. For each task, 5 samples are used for training and 15 samples for testing.

**Compared Methods**: We compare the results with stochastic gradient descent (SGD) and Adam.

**Base model**: Multiple Layer Perception with two hidden layers (each layer has 40 neurons).

**Results**:



# Experiments: Real-world Dataset

**Dataset Description**: We selected 100 class from Imagenet, 64, 16, 20 classes are used for training, validation and testing. For each task, we randomly select 5 classes (5-way), each class has 1 or 5 (1-shot or 5-shot) sample for training and 15 samples for testing.
**Compared Methods**: Adam.
**Base model**: Four layers convolutional neural network.
**Results**: The accuracy and training loss are shown in Table 1 and Figure 2.

Table 1: Classification Accuracy

| Model | 5-way 1-shot | 5-way 5-shot |
|---|---|---|
| Adam | 48.70±1.68 | 63.11±0.92 |
| SCO | **50.01±1.65** | **64.02±0.83** |

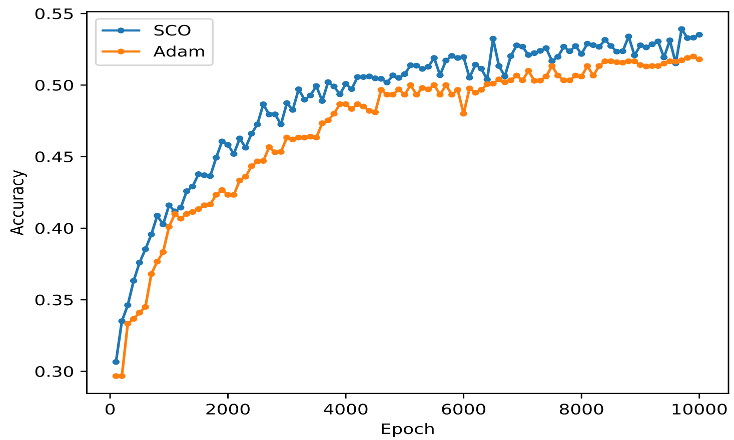



Figure 3: Accuracy of 1-shot

# Discussion & Conclusion

**Theoretical Analysis**: Since the meta-learning problem is non-convex case, the convergence error bound of stochastic gradient descent algorithm is $\mathcal{O}(n^{-4/(7+T)})$.

**Weakness**:

1. The computational cost of this algorithm is still high, usually take several hours to train.
2. Empirically, sometimes the algorithms is a little unstable.

**Conclusion & Discussion**:

1. By using stochastic composition gradient descent on gradient-based meta-learning problem, we can alleviate the effect of biased SO. Empirically, we achieve better performance on synthetic and real-world datasets.
2. In the future, we can apply the stochastic composition optimization to more meta-learning applications, especially reinforcement learning case. In addition, we can apply to more meta-learning frameworks (e.g., recurrent meta-learning)