Kernel Recursive ABC: Point Estimation with Intractable Likelihood

Motonobu Kanagawa

EURECOM, Sophia Antipolis, France
(Previously U. Tübingen)

ISM-UUlm Workshop, October 2019

Contents of This Talk

1. **Kernel Recursive ABC: Point Estimation with Intractable Likelihood** (ICML 2018)
   T. Kajihara, M. Kanagawa, K. Yamazaki and K. Fukumizu.

Outline

Machine Learning for Computer Simulation

Computer simulation has been used to study time-evolving complex phenomena in various scientific fields.

- Climate science, social science, economics, ecology, epidemiology, etc. etc...

Machine Learning for Computer Simulation

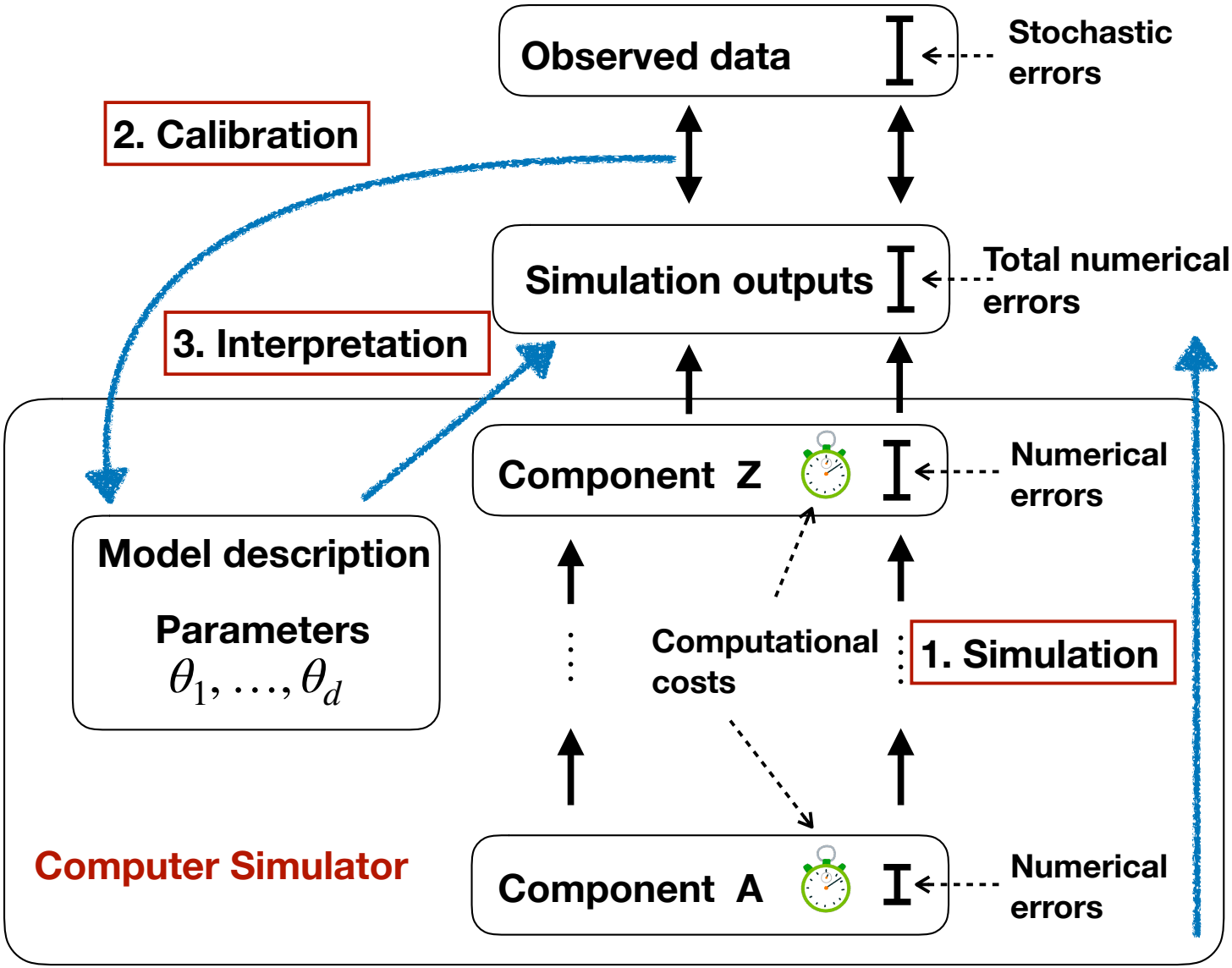
Computer simulation has been used to study time-evolving complex phenomena in various scientific fields.

- Climate science, social science, economics, ecology, epidemiology, etc. etc...

The power of computer simulation is extrapolation, which enables

- predictions of quantities/phenomena in the future.
- gaining understanding of the phenomena of interest.

Machine Learning for Computer Simulation



Example: Tsunami Simulation [Saito, 2019, p.211]

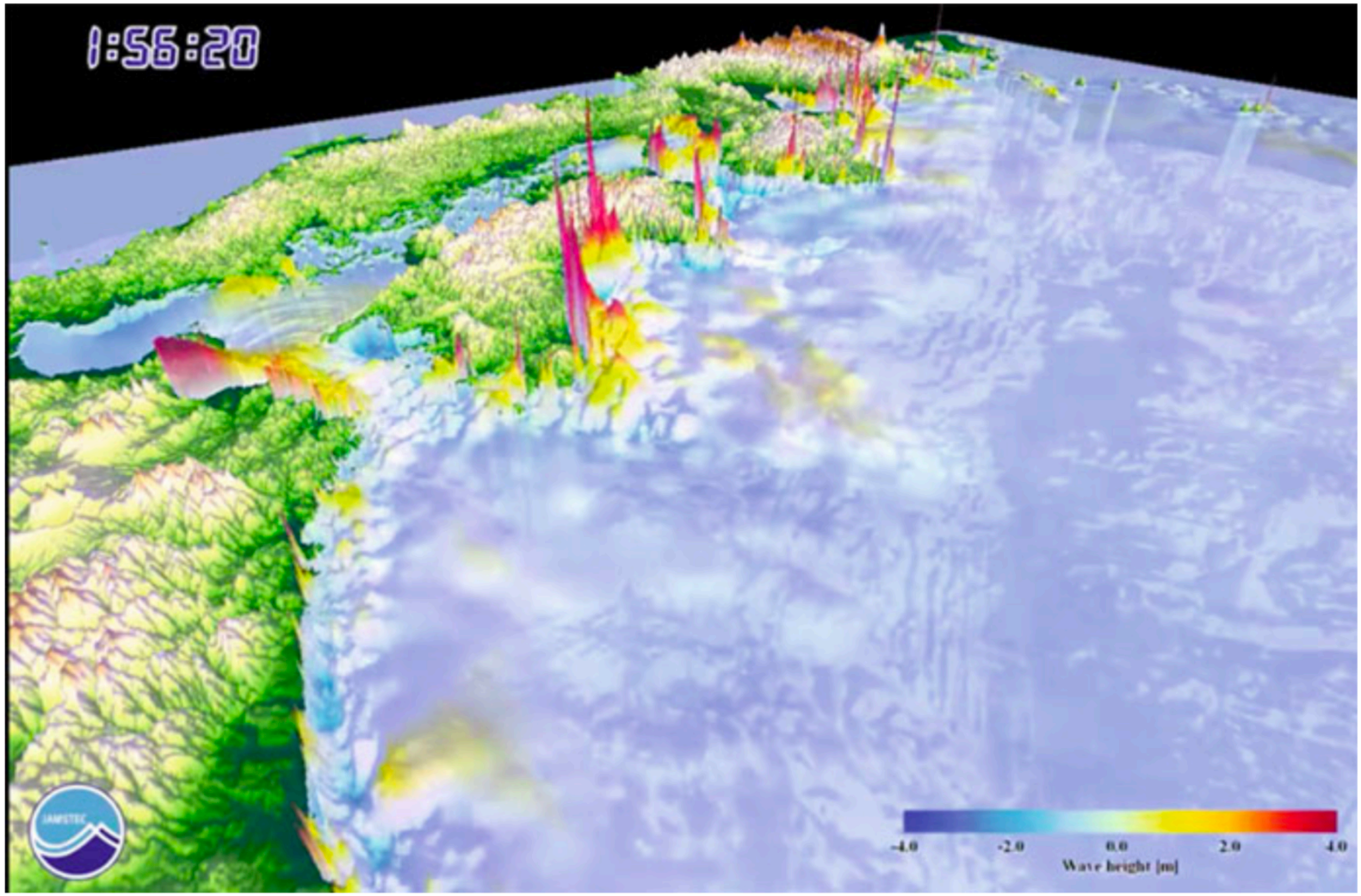


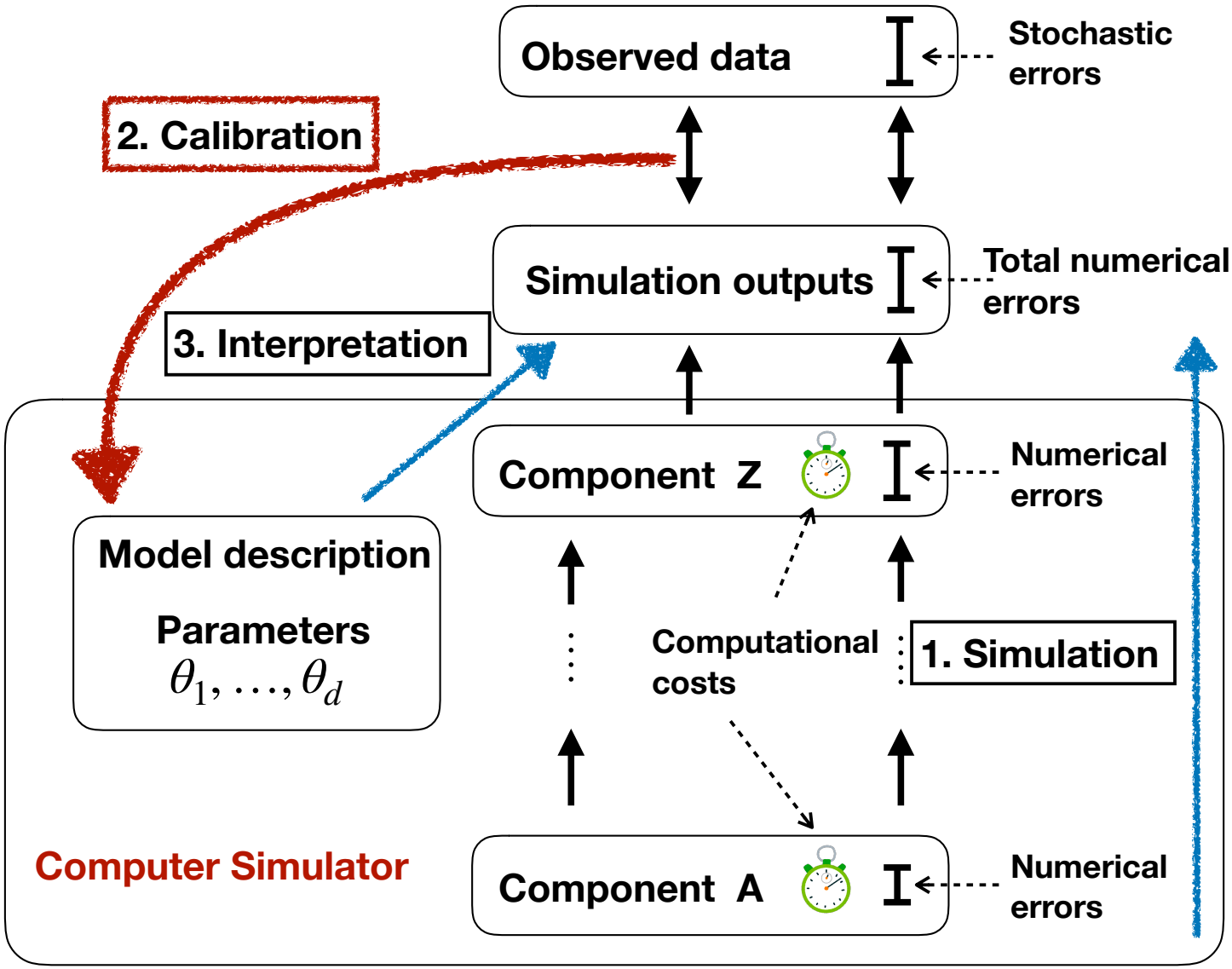
Fig. 6.5 An example of a tsunami simulation (JAGURS code) in the Nankai subduction zone, southwestern Japan. (Courtesy of T. Baba)

Example: Pedestrian Flow Simulation [Yamashita et al., 2010]

Multi-agent systems for pedestrians walking in Ginza.

Figure 1: Points representing individual pedestrians. (red = slow)

Calibration: Parameter Estimation and Model Selection



Calibration: Parameter Estimation and Model Selection

To obtain a "good" simulator, the following two tasks regarding calibration to observed data must be addressed.

Calibration: Parameter Estimation and Model Selection

To obtain a "good" simulator, the following two tasks regarding calibration to observed data must be addressed.

1. **Parameter estimation**
   - Estimate parameters θ of a simulation model $p(\boldsymbol{y}^*|\theta)$. ($\boldsymbol{y}^*$ denotes observed data.)

Calibration: Parameter Estimation and Model Selection

To obtain a "good" simulator, the following two tasks regarding calibration to observed data must be addressed.

1. **Parameter estimation**
   - Estimate parameters θ of a simulation model $p(\boldsymbol{y}^*|\theta)$. ($\boldsymbol{y}^*$ denotes observed data.)
2. **Model selection**
   - Select one model from multiple ($K \geq 2$) candidate models:

$$p_1(\boldsymbol{y}^*|\theta_1),\ p_2(\boldsymbol{y}^*|\theta_2),\ \ldots, p_K(\boldsymbol{y}^*|\theta_K)$$

Calibration: Parameter Estimation and Model Selection

To obtain a "good" simulator, the following two tasks regarding calibration to observed data must be addressed.

1. **Parameter estimation**
   - Estimate parameters θ of a simulation model $p(\boldsymbol{y}^*|\theta)$. ($\boldsymbol{y}^*$ denotes observed data.)
2. **Model selection**
   - Select one model from multiple ($K \geq 2$) candidate models:

$$p_1(\boldsymbol{y}^*|\theta_1),\ p_2(\boldsymbol{y}^*|\theta_2),\ \ldots, p_K(\boldsymbol{y}^*|\theta_K)$$

In the language of statistics, computer simulation can be defined as sampling from a probabilistic model $p(\boldsymbol{y}|\theta)$.

Calibration: Parameter Estimation and Model Selection

These tasks are harder than standard statistical problems, since the likelihood function

$$\ell(\theta) := p(\boldsymbol{y}^*|\theta)$$

is not available.

Calibration: Parameter Estimation and Model Selection

These tasks are harder than standard statistical problems, since the likelihood function

$$\ell(\theta) := p(\boldsymbol{y}^*|\theta)$$

is not available. This is because

- The mapping $\theta \to \boldsymbol{y}$ is usually very complicated. (e.g., it involves solving differential equations)

Calibration: Parameter Estimation and Model Selection

These tasks are harder than standard statistical problems, since the likelihood function

$$\ell(\theta) := p(\boldsymbol{y}^*|\theta)$$

is not available. This is because

- The mapping $\theta \to \boldsymbol{y}$ is usually very complicated. (e.g., it involves solving differential equations)

Thus one needs to solve these tasks by likelihood-free inference, making use of sampling/forward simulations.

Calibration: Parameter Estimation and Model Selection

These tasks are harder than standard statistical problems, since the likelihood function

$$\ell(\theta) := p(\boldsymbol{y}^*|\theta)$$

is not available. This is because

- The mapping $\theta \to \boldsymbol{y}$ is usually very complicated. (e.g., it involves solving differential equations)

Thus one needs to solve these tasks by likelihood-free inference, making use of sampling/forward simulations.

Approaches to likelihood-free inference include

- Approximate Bayesian Computation (ABC) [Sisson et al., 2018].
- Bayesian optimization [Gutmann and Corander, 2016].

Approximate Bayesian Computation (ABC)

– Set $\varepsilon > 0$ and $J := \{\}$.

Approximate Bayesian Computation (ABC)

- Set $\varepsilon > 0$ and $J := \{\}$.
- Generate parameter-data pairs from the model:

$$(\theta_1, \boldsymbol{y}_1), \ldots, (\theta_n, \boldsymbol{y}_n) \sim \underbrace{p(\boldsymbol{y}|\theta)}_{\text{simulator}} \underbrace{\pi(\theta)}_{\text{prior}}, \quad \text{i.i.d.}$$

Approximate Bayesian Computation (ABC)

– Set $\varepsilon > 0$ and $J := \{\}$.

– Generate parameter-data pairs from the model:

$$(\theta_1, \boldsymbol{y}_1), \ldots, (\theta_n, \boldsymbol{y}_n) \sim \underbrace{p(\boldsymbol{y}|\theta)}_{\text{simulator}} \underbrace{\pi(\theta)}_{\text{prior}}, \quad \text{i.i.d.}$$

– For $j = 1, \ldots, n$, set

$$J \leftarrow J \cup \{j\} \quad \text{if} \quad \mathrm{dist}(\boldsymbol{y}_i, \boldsymbol{y}^*) \leq \varepsilon,$$

where $\boldsymbol{y}^*$ is observed data.

Approximate Bayesian Computation (ABC)

– Set $\varepsilon > 0$ and $J := \{\}$.

– Generate parameter-data pairs from the model:

$$(\theta_1, \boldsymbol{y}_1), \ldots, (\theta_n, \boldsymbol{y}_n) \sim \underbrace{p(\boldsymbol{y}|\theta)}_{\text{simulator}} \underbrace{\pi(\theta)}_{\text{prior}}, \quad \text{i.i.d.}$$

– For $j = 1, \ldots, n$, set

$$J \leftarrow J \cup \{j\} \quad \text{if} \quad \operatorname{dist}(\boldsymbol{y}_i, \boldsymbol{y}^*) \leq \varepsilon,$$

where $\boldsymbol{y}^*$ is observed data.

– Monte Carlo approximation of the posterior:

$$p(\theta|\boldsymbol{y}^*) \approx \hat{p}_\varepsilon(\theta|\boldsymbol{y}^*) := \frac{1}{|J|} \sum_{j \in J} \underbrace{\delta_{\theta_j}}_{\text{Dirac at } \theta_j}$$

Contributions

We propose a kernel-based method for point estimation of simulation-based statistical models.

Contributions

We propose a kernel-based method for point estimation of simulation-based statistical models.

The proposed approach (termed **kernel recursive ABC**)

Contributions

We propose a kernel-based method for point estimation of simulation-based statistical models.

The proposed approach (termed **kernel recursive ABC**)

- is based on kernel mean embeddings,

Contributions

We propose a kernel-based method for point estimation of simulation-based statistical models.

The proposed approach (termed **kernel recursive ABC**)

- is based on kernel mean embeddings,
- is a combination of kernel ABC and kernel herding, and

Contributions

We propose a kernel-based method for point estimation of simulation-based statistical models.

The proposed approach (termed **kernel recursive ABC**)

- is based on kernel mean embeddings,
- is a combination of kernel ABC and kernel herding, and
- recursively applies Bayes' rule to the same observed data.

Contributions

We propose a kernel-based method for point estimation of simulation-based statistical models.

The proposed approach (termed **kernel recursive ABC**)

- is based on kernel mean embeddings,
- is a combination of kernel ABC and kernel herding, and
- recursively applies Bayes' rule to the same observed data.

It should be useful when point estimation is more desirable than the fully Bayesian approach.

Contributions

We propose a kernel-based method for point estimation of simulation-based statistical models.

The proposed approach (termed **kernel recursive ABC**)

- is based on kernel mean embeddings,
- is a combination of kernel ABC and kernel herding, and
- recursively applies Bayes' rule to the same observed data.

It should be useful when point estimation is more desirable than the fully Bayesian approach. For instance:

- when your prior distribution $\pi(\theta)$ is not fully reliable,

Contributions

We propose a kernel-based method for point estimation of simulation-based statistical models.

The proposed approach (termed **kernel recursive ABC**)

- is based on kernel mean embeddings,
- is a combination of kernel ABC and kernel herding, and
- recursively applies Bayes' rule to the same observed data.

It should be useful when point estimation is more desirable than the fully Bayesian approach. For instance:

- when your prior distribution $\pi(\theta)$ is not fully reliable,
- when one simulation is computationally very expensive, and

Contributions

We propose a kernel-based method for point estimation of simulation-based statistical models.

The proposed approach (termed **kernel recursive ABC**)

- is based on kernel mean embeddings,
- is a combination of kernel ABC and kernel herding, and
- recursively applies Bayes' rule to the same observed data.

It should be useful when point estimation is more desirable than the fully Bayesian approach. For instance:

- when your prior distribution $\pi(\theta)$ is not fully reliable,
- when one simulation is computationally very expensive, and
- when your purpose is on predictions based on simulations.

Outline

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function on a set $\mathcal{X}$.

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function on a set $\mathcal{X}$.

The function $k(x, x')$ is called a **positive definite kernel**, if

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \quad \text{holds}$$

for all $\quad n \in \mathbb{N}, \quad c_1, \ldots, c_n \in \mathbb{R}, \quad x_1, \ldots, x_n \in \mathcal{X}.$

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function on a set $\mathcal{X}$.

The function $k(x, x')$ is called a **positive definite kernel**, if

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \quad \text{holds}$$

$$\text{for all} \quad n \in \mathbb{N}, \quad c_1, \ldots, c_n \in \mathbb{R}, \quad x_1, \ldots, x_n \in \mathcal{X}.$$

Examples of positive definite kernels on $\mathcal{X} = \mathbb{R}^d$:

$$\begin{aligned}
\text{Gaussian} \quad & k(x, x') &&= \exp(-\|x - x'\|^2/\gamma^2). \\
\text{Laplace (Matérn)} \quad & k(x, x') &&= \exp(-\|x - x'\|/\gamma). \\
\text{Linear} \quad & k(x, x') &&= \langle x, x' \rangle. \\
\text{Polynomial} \quad & k(x, x') &&= (\langle x, x' \rangle + c)^m.
\end{aligned}$$

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function on a set $\mathcal{X}$.

The function $k(x, x')$ is called a **positive definite kernel**, if

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \quad \text{holds}$$

$$\text{for all} \quad n \in \mathbb{N}, \quad c_1, \ldots, c_n \in \mathbb{R}, \quad x_1, \ldots, x_n \in \mathcal{X}.$$

Examples of positive definite kernels on $\mathcal{X} = \mathbb{R}^d$:

$$\begin{aligned}
\text{Gaussian} \quad k(x, x') &= \exp(-\|x - x'\|^2/\gamma^2). \\
\text{Laplace (Matérn)} \quad k(x, x') &= \exp(-\|x - x'\|/\gamma). \\
\text{Linear} \quad k(x, x') &= \langle x, x' \rangle. \\
\text{Polynomial} \quad k(x, x') &= (\langle x, x' \rangle + c)^m.
\end{aligned}$$

In this talk, I will simply call k a kernel.

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

For any kernel k, there is a **uniquely associated Hilbert space $\mathcal{H}$** consisting of functions on $\mathcal{X}$ such that

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

For any kernel k, there is a **uniquely associated Hilbert space $\mathcal{H}$** consisting of functions on $\mathcal{X}$ such that

(i) $k(\cdot, x) \in \mathcal{H}$ for all $x \in \mathcal{X}$

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

For any kernel k, there is a **uniquely associated Hilbert space $\mathcal{H}$** consisting of functions on $\mathcal{X}$ such that

$$\text{(i)} \quad k(\cdot, x) \in \mathcal{H} \quad \text{for all } x \in \mathcal{X}$$

where $k(\cdot, x)$ is the function of the first argument with x fixed:

$$x' \in \mathcal{X} \to k(x', x).$$

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

For any kernel k, there is a **uniquely associated Hilbert space $\mathcal{H}$** consisting of functions on $\mathcal{X}$ such that

$$\text{(i)} \quad k(\cdot, x) \in \mathcal{H} \quad \text{for all } x \in \mathcal{X}$$

where $k(\cdot, x)$ is the function of the first argument with x fixed:

$$x' \in \mathcal{X} \to k(x', x).$$

$$\text{(ii)} \quad f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}} \quad \text{for all } f \in \mathcal{H} \text{ and } x \in \mathcal{X},$$

which is called the **reproducing property**.

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

For any kernel k, there is a **uniquely associated Hilbert space $\mathcal{H}$** consisting of functions on $\mathcal{X}$ such that

$$\text{(i)} \quad k(\cdot, x) \in \mathcal{H} \quad \text{for all } x \in \mathcal{X}$$

where $k(\cdot, x)$ is the function of the first argument with x fixed:

$$x' \in \mathcal{X} \to k(x', x).$$

$$\text{(ii)} \quad f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}} \quad \text{for all } f \in \mathcal{H} \text{ and } x \in \mathcal{X},$$

which is called the **reproducing property**.

– $\mathcal{H}$ is called the **RKHS** of k.

Kernels and Reproducing Kernel Hilbert Spaces (RKHS)

For any kernel k, there is a **uniquely associated Hilbert space $\mathcal{H}$** consisting of functions on $\mathcal{X}$ such that

$$\text{(i)} \quad k(\cdot, x) \in \mathcal{H} \quad \text{for all } x \in \mathcal{X}$$

where $k(\cdot, x)$ is the function of the first argument with x fixed:

$$x' \in \mathcal{X} \to k(x', x).$$

$$\text{(ii)} \quad f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}} \quad \text{for all } f \in \mathcal{H} \text{ and } x \in \mathcal{X},$$

which is called the **reproducing property**.

– $\mathcal{H}$ is called the **RKHS** of k.
– $\mathcal{H}$ can be written as

$$\mathcal{H} = \overline{\text{span}\,\{k(\cdot, x) \mid x \in \mathcal{X}\}}$$

Kernel Mean Embeddings [Smola et al., 2007]

A framework for representing distributions in an RKHS.

Kernel Mean Embeddings [Smola et al., 2007]

A framework for representing distributions in an RKHS.

– Let $\mathcal{P}$ be the set of **all probability distributions** on $\mathcal{X}$.

Kernel Mean Embeddings [Smola et al., 2007]

A framework for representing distributions in an RKHS.

– Let $\mathcal{P}$ be the set of **all probability distributions** on $\mathcal{X}$.
– Let k be a kernel on $\mathcal{X}$, and $\mathcal{H}$ be its RKHS.

Kernel Mean Embeddings [Smola et al., 2007]

A framework for representing distributions in an RKHS.

– Let $\mathcal{P}$ be the set of **all probability distributions** on $\mathcal{X}$.
– Let k be a kernel on $\mathcal{X}$, and $\mathcal{H}$ be its RKHS.

For each distribution $P \in \mathcal{P}$, define the **kernel mean**:

$$\mu_P := \int k(\cdot, x) dP(x) \in \mathcal{H}.$$

which is a **representation** of P in $\mathcal{H}$.

Kernel Mean Embeddings [Smola et al., 2007]

A framework for representing distributions in an RKHS.

– Let $\mathcal{P}$ be the set of **all probability distributions** on $\mathcal{X}$.
– Let k be a kernel on $\mathcal{X}$, and $\mathcal{H}$ be its RKHS.

For each distribution $P \in \mathcal{P}$, define the **kernel mean**:

$$\mu_P := \int k(\cdot, x) dP(x) \in \mathcal{H}.$$

which is a **representation** of P in $\mathcal{H}$.

A key concept: Characteristic kernels [Fukumizu et al., 2008].

– The kernel k is called **characteristic**, if for any $P, Q \in \mathcal{P}$,

$$\mu_P = \mu_Q \quad \text{if and only if} \quad P = Q.$$

Kernel Mean Embeddings [Smola et al., 2007]

A framework for representing distributions in an RKHS.

– Let $\mathcal{P}$ be the set of **all probability distributions** on $\mathcal{X}$.
– Let k be a kernel on $\mathcal{X}$, and $\mathcal{H}$ be its RKHS.

For each distribution $P \in \mathcal{P}$, define the **kernel mean**:

$$\mu_P := \int k(\cdot, x) dP(x) \in \mathcal{H}.$$

which is a **representation** of P in $\mathcal{H}$.

A key concept: Characteristic kernels [Fukumizu et al., 2008].

– The kernel k is called **characteristic**, if for any $P, Q \in \mathcal{P}$,

$$\mu_P = \mu_Q \quad \text{if and only if} \quad P = Q.$$

– In other words, k is characteristic if

$$\text{the mapping} \quad P \in \mathcal{P} \to \mu_P \in \mathcal{H} \quad \text{is injective}.$$

Kernel Mean Embeddings [Smola et al., 2007]

Intuitively, k being **characteristic** implies that $\mathcal{H}$ **is large enough.**

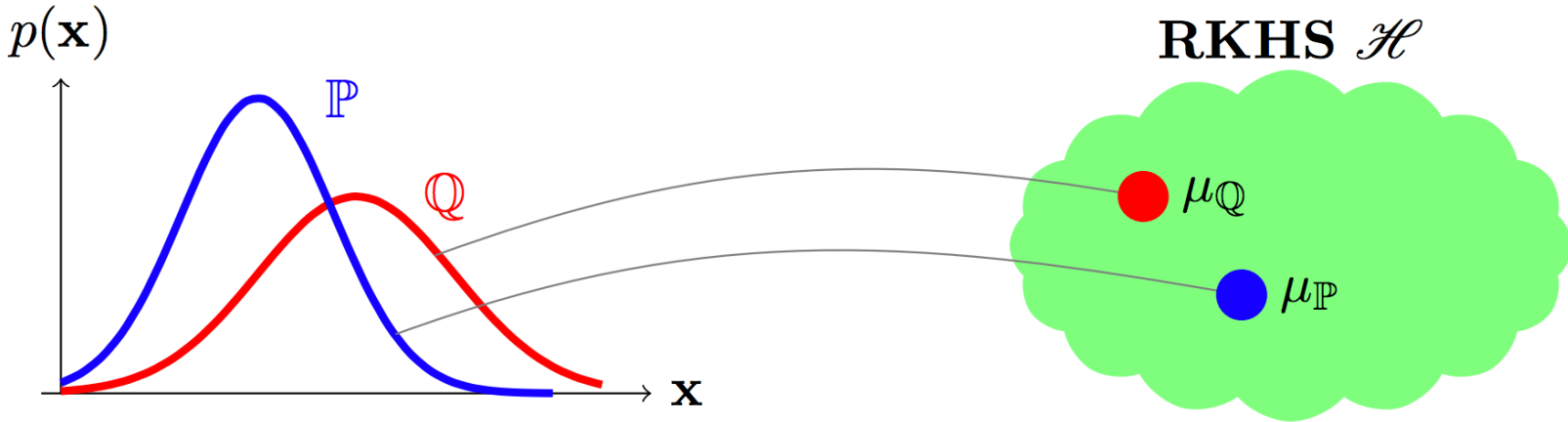


Figure 2: Injective embedding [Muandet et al., 2017, Figure 2.3]

Kernel Mean Embeddings [Smola et al., 2007]

Intuitively, k being **characteristic** implies that $\mathcal{H}$ **is large enough.**

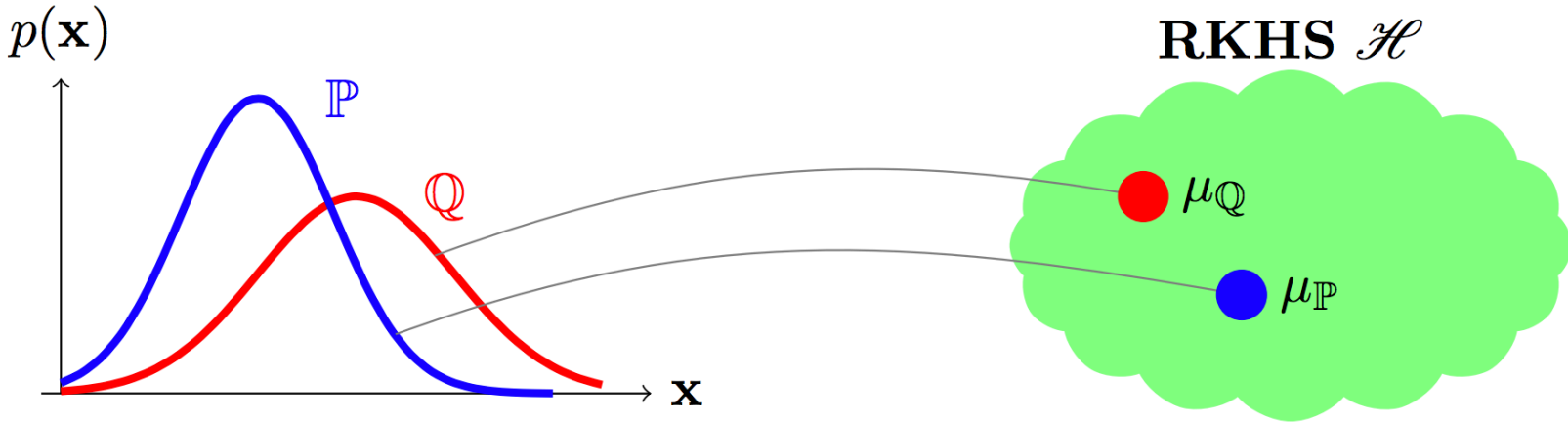


Figure 2: Injective embedding [Muandet et al., 2017, Figure 2.3]

Examples of characteristic kernels on $\mathcal{X} = \mathbb{R}^d$:

– Gaussian and Matérn kernels [Sriperumbudur et al., 2010].

Kernel Mean Embeddings [Smola et al., 2007]

Intuitively, k being **characteristic** implies that $\mathcal{H}$ **is large enough.**

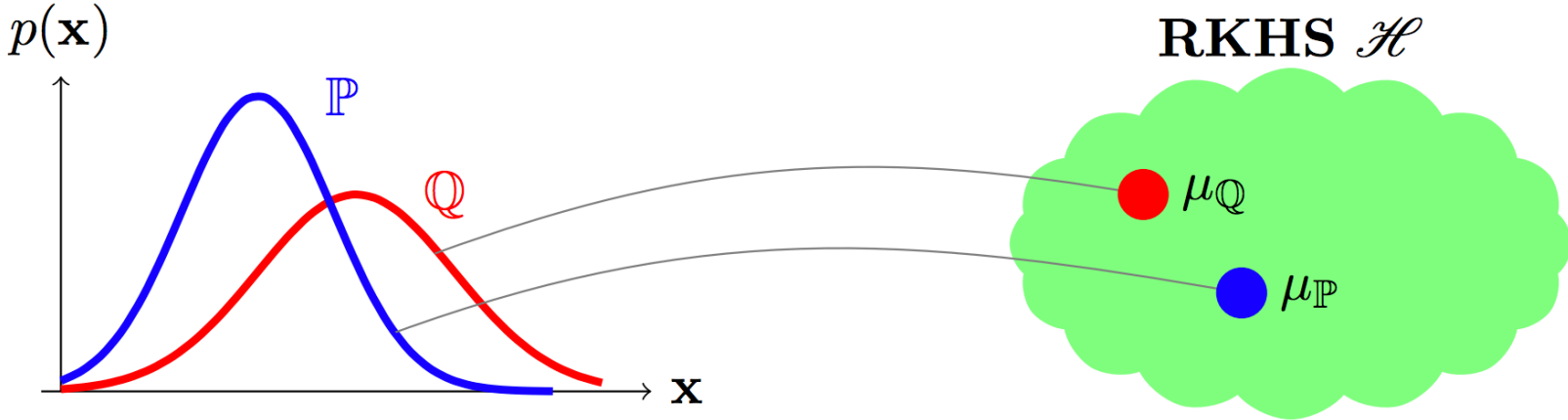


Figure 2: Injective embedding [Muandet et al., 2017, Figure 2.3]

Examples of characteristic kernels on $\mathcal{X} = \mathbb{R}^d$:
– Gaussian and Matérn kernels [Sriperumbudur et al., 2010].

Examples of **non**-characteristic kernels on $\mathcal{X} = \mathbb{R}^d$:
– Linear and polynomial kernels.

Outline

Recursive Bayes Updates and Power Posteriors

Given observed data $\boldsymbol{y}^*$, **Bayes' rule** yields a posterior distribution:

$$\underbrace{p(\theta|\boldsymbol{y}^*)}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{y}^*|\theta)}_{\text{Likelihood}} \underbrace{\pi(\theta)}_{\text{Prior}}.$$

Recursive Bayes Updates and Power Posteriors

Given observed data $\boldsymbol{y}^*$, **Bayes' rule** yields a posterior distribution:

$$\underbrace{p(\theta|\boldsymbol{y}^*)}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{y}^*|\theta)}_{\text{Likelihood}} \underbrace{\pi(\theta)}_{\text{Prior}}.$$

Recursive Bayes updates: Apply **Bayes' rule recursively** to the **same observed data $\boldsymbol{y}^*$**.

$$\textbf{1st recursion} \quad \pi_1(\theta) := p_1(\theta|\boldsymbol{y}^*) \quad \propto \quad p(\boldsymbol{y}^*|\theta)\pi(\theta).$$

Recursive Bayes Updates and Power Posteriors

Given observed data $\boldsymbol{y}^*$, **Bayes' rule** yields a posterior distribution:

$$\underbrace{p(\theta|\boldsymbol{y}^*)}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{y}^*|\theta)}_{\text{Likelihood}} \underbrace{\pi(\theta)}_{\text{Prior}}.$$

Recursive Bayes updates: Apply **Bayes' rule recursively** to the **same observed data $\boldsymbol{y}^*$**.

$$\begin{array}{rrcl} \textbf{1st recursion} & \pi_1(\theta) := p_1(\theta|\boldsymbol{y}^*) & \propto & p(\boldsymbol{y}^*|\theta)\pi(\theta). \\ \textbf{2nd recursion} & \pi_2(\theta) := p_2(\theta|\boldsymbol{y}^*) & \propto & p(\boldsymbol{y}^*|\theta)\pi_1(\theta) \end{array}$$

Recursive Bayes Updates and Power Posteriors

Given observed data $\boldsymbol{y}^*$, **Bayes' rule** yields a posterior distribution:

$$\underbrace{p(\theta|\boldsymbol{y}^*)}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{y}^*|\theta)}_{\text{Likelihood}} \underbrace{\pi(\theta)}_{\text{Prior}}.$$

Recursive Bayes updates: Apply **Bayes' rule recursively** to the **same observed data $\boldsymbol{y}^*$**.

$$\begin{aligned}
\textbf{1st recursion} \quad \pi_1(\theta) := p_1(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi(\theta). \\
\textbf{2nd recursion} \quad \pi_2(\theta) := p_2(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_1(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^2\pi(\theta).
\end{aligned}$$

Recursive Bayes Updates and Power Posteriors

Given observed data $\boldsymbol{y}^*$, **Bayes' rule** yields a posterior distribution:

$$\underbrace{p(\theta|\boldsymbol{y}^*)}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{y}^*|\theta)}_{\text{Likelihood}} \underbrace{\pi(\theta)}_{\text{Prior}}.$$

Recursive Bayes updates: Apply **Bayes' rule recursively** to the **same observed data $\boldsymbol{y}^*$**.

$$\begin{aligned}
\textbf{1st recursion} \quad \pi_1(\theta) := p_1(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi(\theta). \\
\textbf{2nd recursion} \quad \pi_2(\theta) := p_2(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_1(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^2\pi(\theta). \\
\textbf{3rd recursion} \quad \pi_3(\theta) := p_3(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_2(\theta)
\end{aligned}$$

Recursive Bayes Updates and Power Posteriors

Given observed data $\boldsymbol{y}^*$, **Bayes' rule** yields a posterior distribution:

$$\underbrace{p(\theta|\boldsymbol{y}^*)}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{y}^*|\theta)}_{\text{Likelihood}} \underbrace{\pi(\theta)}_{\text{Prior}}.$$

Recursive Bayes updates: Apply **Bayes' rule recursively** to the **same observed data $\boldsymbol{y}^*$**.

$$\begin{aligned}
\textbf{1st recursion} \quad \pi_1(\theta) := p_1(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi(\theta). \\
\textbf{2nd recursion} \quad \pi_2(\theta) := p_2(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_1(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^2\pi(\theta). \\
\textbf{3rd recursion} \quad \pi_3(\theta) := p_3(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_2(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^3\pi(\theta).
\end{aligned}$$

Recursive Bayes Updates and Power Posteriors

Given observed data $\boldsymbol{y}^*$, **Bayes' rule** yields a posterior distribution:

$$\underbrace{p(\theta|\boldsymbol{y}^*)}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{y}^*|\theta)}_{\text{Likelihood}} \underbrace{\pi(\theta)}_{\text{Prior}}.$$

Recursive Bayes updates: Apply **Bayes' rule recursively** to the **same observed data $\boldsymbol{y}^*$**.

$$\begin{aligned}
\textbf{1st recursion} \quad \pi_1(\theta) := p_1(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi(\theta). \\
\textbf{2nd recursion} \quad \pi_2(\theta) := p_2(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_1(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^2\pi(\theta). \\
\textbf{3rd recursion} \quad \pi_3(\theta) := p_3(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_2(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^3\pi(\theta). \\
&\dots
\end{aligned}$$

Recursive Bayes Updates and Power Posteriors

Given observed data $\boldsymbol{y}^*$, **Bayes' rule** yields a posterior distribution:

$$\underbrace{p(\theta|\boldsymbol{y}^*)}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{y}^*|\theta)}_{\text{Likelihood}} \underbrace{\pi(\theta)}_{\text{Prior}}.$$

Recursive Bayes updates: Apply **Bayes' rule recursively** to the **same observed data $\boldsymbol{y}^*$**.

$$\begin{aligned}
\textbf{1st recursion} \quad \pi_1(\theta) := p_1(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi(\theta). \\
\textbf{2nd recursion} \quad \pi_2(\theta) := p_2(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_1(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^2\pi(\theta). \\
\textbf{3rd recursion} \quad \pi_3(\theta) := p_3(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_2(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^3\pi(\theta). \\
&\dots \\
N\textbf{-th recursion} \quad \pi_N(\theta) := p_N(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_{N-1}(\theta)
\end{aligned}$$

Recursive Bayes Updates and Power Posteriors

Given observed data $\boldsymbol{y}^*$, **Bayes' rule** yields a posterior distribution:

$$\underbrace{p(\theta|\boldsymbol{y}^*)}_{\text{Posterior}} \propto \underbrace{p(\boldsymbol{y}^*|\theta)}_{\text{Likelihood}} \underbrace{\pi(\theta)}_{\text{Prior}}.$$

Recursive Bayes updates: Apply **Bayes' rule recursively** to the **same observed data $\boldsymbol{y}^*$**.

$$\begin{aligned}
\textbf{1st recursion} \quad \pi_1(\theta) := p_1(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi(\theta). \\
\textbf{2nd recursion} \quad \pi_2(\theta) := p_2(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_1(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^2\pi(\theta). \\
\textbf{3rd recursion} \quad \pi_3(\theta) := p_3(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_2(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^3\pi(\theta). \\
&\dots \\
N\textbf{-th recursion} \quad \pi_N(\theta) := p_N(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi_{N-1}(\theta) \\
&= p(\boldsymbol{y}^*|\theta)^N\pi(\theta).
\end{aligned}$$

Power Posteriors and Maximum Likelihood Estimation

N recursive Bayes updates yield the **power posterior**

$$p_N(\theta|\boldsymbol{y}^*) \propto p(\boldsymbol{y}^*|\theta)^N \pi(\theta)$$

Power Posteriors and Maximum Likelihood Estimation

N recursive Bayes updates yield the **power posterior**

$$p_N(\theta|\boldsymbol{y}^*) \propto p(\boldsymbol{y}^*|\theta)^N \pi(\theta)$$

Theorem [Lele et al., 2010].
Assume that $p(\boldsymbol{y}^*|\theta)$ has a unique global maximizer:

$$\theta^* := \arg\max_{\theta \in \Theta} p(\boldsymbol{y}^*|\theta).$$

Power Posteriors and Maximum Likelihood Estimation

N recursive Bayes updates yield the **power posterior**

$$p_N(\theta|\boldsymbol{y}^*) \propto p(\boldsymbol{y}^*|\theta)^N \pi(\theta)$$

Theorem [Lele et al., 2010].
Assume that $p(\boldsymbol{y}^*|\theta)$ has a unique global maximizer:

$$\theta^* := \arg\max_{\theta \in \Theta} p(\boldsymbol{y}^*|\theta).$$

Then, if $\pi(\theta^*) > 0$, under mild conditions on $\pi(\theta)$ and $p(\boldsymbol{y}|\theta)$,

$$p_N(\theta|\boldsymbol{y}^*) \to \underbrace{\delta_{\theta^*}}_{\text{Dirac at } \theta^*} \quad \text{as } N \to \infty \quad \text{(weak convergence)}.$$

Power Posteriors and Maximum Likelihood Estimation

N recursive Bayes updates yield the **power posterior**

$$p_N(\theta|\boldsymbol{y}^*) \propto p(\boldsymbol{y}^*|\theta)^N \pi(\theta)$$

Theorem [Lele et al., 2010].
Assume that $p(\boldsymbol{y}^*|\theta)$ has a unique global maximizer:

$$\theta^* := \arg\max_{\theta \in \Theta} p(\boldsymbol{y}^*|\theta).$$

Then, if $\pi(\theta^*) > 0$, under mild conditions on $\pi(\theta)$ and $p(\boldsymbol{y}|\theta)$,

$$p_N(\theta|\boldsymbol{y}^*) \to \underbrace{\delta_{\theta^*}}_{\text{Dirac at } \theta^*} \quad \text{as } N \to \infty \quad \text{(weak convergence)}.$$

This implies that recursive Bayes updates provide a way of Maximum Likelihood Estimation.

Proposed Method: Kernel Recursive ABC (Sketch)

– **Recursive Applications of 1.Bayes' rule and 2.sampling.**

Proposed Method: Kernel Recursive ABC (Sketch)

– Recursive Applications of 1.Bayes' rule and 2.sampling.

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedures:

Proposed Method: Kernel Recursive ABC (Sketch)

– Recursive Applications of 1.Bayes' rule and 2.sampling.

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedures:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

Proposed Method: Kernel Recursive ABC (Sketch)

– **Recursive Applications of 1.Bayes' rule and 2.sampling.**

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedures:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

– Simulate pseudo-data for each θ_i:

$$\boldsymbol{y}_i \sim p(\boldsymbol{y}|\theta_i) \quad (i = 1, \ldots, n).$$

Proposed Method: Kernel Recursive ABC (Sketch)

– **Recursive Applications of 1.Bayes' rule and 2.sampling.**

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedures:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

– Simulate pseudo-data for each θ_i:

$$\boldsymbol{y}_i \sim p(\boldsymbol{y}|\theta_i) \quad (i = 1, \ldots, n).$$

– Estimate the kernel mean of the power posterior using $(\theta_i, \boldsymbol{y}_i)_{i=1}^n$:

$$\mu_{P_N} := \int \underbrace{k_\Theta(\cdot, \theta)}_{\text{Kernel on } \Theta} p_N(\theta|\boldsymbol{y}^*) d\theta \tag{1}$$

where $p_N(\theta|\boldsymbol{y}^*) \propto p^N(\boldsymbol{y}|\theta)\pi(\theta)$

Proposed Method: Kernel Recursive ABC (Sketch)

– **Recursive Applications of 1.Bayes' rule and 2.sampling.**

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedures:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

– Simulate pseudo-data for each θ_i:

$$\boldsymbol{y}_i \sim p(\boldsymbol{y}|\theta_i) \quad (i = 1, \ldots, n).$$

– Estimate the kernel mean of the power posterior using $(\theta_i, \boldsymbol{y}_i)_{i=1}^n$:

$$\mu_{P_N} := \int \underbrace{k_\Theta(\cdot, \theta)}_{\text{Kernel on } \Theta} p_N(\theta|\boldsymbol{y}^*) d\theta \qquad (1)$$

$$\text{where} \quad p_N(\theta|\boldsymbol{y}^*) \propto p^N(\boldsymbol{y}|\theta)\pi(\theta)$$

2. Kernel Herding: Sampling $\theta'_1, \ldots, \theta'_n$ from the estimate of (1):

Proposed Method: Kernel Recursive ABC (Sketch)

– **Recursive Applications of 1.Bayes' rule and 2.sampling.**

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedures:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

– Simulate pseudo-data for each θ_i:

$$\boldsymbol{y}_i \sim p(\boldsymbol{y}|\theta_i) \quad (i = 1, \ldots, n).$$

– Estimate the kernel mean of the power posterior using $(\theta_i, \boldsymbol{y}_i)_{i=1}^n$:

$$\mu_{P_N} := \int \underbrace{k_\Theta(\cdot, \theta)}_{\text{Kernel on } \Theta} p_N(\theta|\boldsymbol{y}^*) d\theta \qquad (1)$$

$$\text{where} \quad p_N(\theta|\boldsymbol{y}^*) \propto p^N(\boldsymbol{y}|\theta)\pi(\theta)$$

2. Kernel Herding: Sampling $\theta'_1, \ldots, \theta'_n$ from the estimate of (1):

$$\textbf{Set: } N \leftarrow N + 1 \text{ and } (\theta_1, \ldots, \theta_n) \leftarrow (\theta'_1, \ldots, \theta'_n)$$

Kernel ABC [Nakagome et al., 2013]

– Define

- a kernel $k_{\mathcal{Y}}(\boldsymbol{y}, \boldsymbol{y}')$ on the data space $\mathcal{Y}$,
- a kernel $k_{\Theta}(\theta, \theta')$ on the parameter space Θ, and
- a regularisation constant $\lambda > 0$.

Kernel ABC [Nakagome et al., 2013]

– Define

- a kernel $k_{\mathcal{Y}}(\boldsymbol{y}, \boldsymbol{y}')$ on the data space $\mathcal{Y}$,
- a kernel $k_{\Theta}(\theta, \theta')$ on the parameter space Θ, and
- a regularisation constant $\lambda > 0$.

1. Sampling: Generate parameter-data pairs from the model:

$$(\theta_1, \boldsymbol{y}_1), \ldots, (\theta_n, \boldsymbol{y}_n) \sim p(\boldsymbol{y}|\theta)\pi(\theta), \quad \text{i.i.d.}$$

Kernel ABC [Nakagome et al., 2013]

– Define

- a kernel $k_{\mathcal{Y}}(\boldsymbol{y}, \boldsymbol{y}')$ on the data space $\mathcal{Y}$,
- a kernel $k_{\Theta}(\theta, \theta')$ on the parameter space Θ, and
- a regularisation constant $\lambda > 0$.

1. Sampling: Generate parameter-data pairs from the model:

$$(\theta_1, \boldsymbol{y}_1), \ldots, (\theta_n, \boldsymbol{y}_n) \sim p(\boldsymbol{y}|\theta)\pi(\theta), \quad \text{i.i.d.}$$

2. Weight computation: Given observed data $\boldsymbol{y}^*$, compute

$$\begin{aligned} \boldsymbol{k}_Y(\boldsymbol{y}^*) &:= (k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_1), \ldots, k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_n))^\top \in \mathbb{R}^n. \\ (w_1(\boldsymbol{y}^*), \ldots, w_n(\boldsymbol{y}^*))^\top &:= (G_Y + n\lambda I_n)^{-1} \boldsymbol{k}_Y(\boldsymbol{y}^*) \in \mathbb{R}^n, \end{aligned}$$

where $G_Y := (k_{\mathcal{Y}}(\boldsymbol{y}_i, \boldsymbol{y}_j)) \in \mathbb{R}^{n \times n}$ is the kernel matrix.

Kernel ABC [Nakagome et al., 2013]

– Define

- a kernel $k_{\mathcal{Y}}(\boldsymbol{y}, \boldsymbol{y}')$ on the data space $\mathcal{Y}$,
- a kernel $k_{\Theta}(\theta, \theta')$ on the parameter space Θ, and
- a regularisation constant $\lambda > 0$.

1. Sampling: Generate parameter-data pairs from the model:

$$(\theta_1, \boldsymbol{y}_1), \ldots, (\theta_n, \boldsymbol{y}_n) \sim p(\boldsymbol{y}|\theta)\pi(\theta), \quad \text{i.i.d.}$$

2. Weight computation: Given observed data $\boldsymbol{y}^*$, compute

$$\begin{aligned} \boldsymbol{k}_Y(\boldsymbol{y}^*) &:= (k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_1), \ldots, k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_n))^\top \in \mathbb{R}^n. \\ (w_1(\boldsymbol{y}^*), \ldots, w_n(\boldsymbol{y}^*))^\top &:= (G_Y + n\lambda I_n)^{-1} \boldsymbol{k}_Y(\boldsymbol{y}^*) \in \mathbb{R}^n, \end{aligned}$$

where $G_Y := (k_{\mathcal{Y}}(\boldsymbol{y}_i, \boldsymbol{y}_j)) \in \mathbb{R}^{n \times n}$ is the kernel matrix.

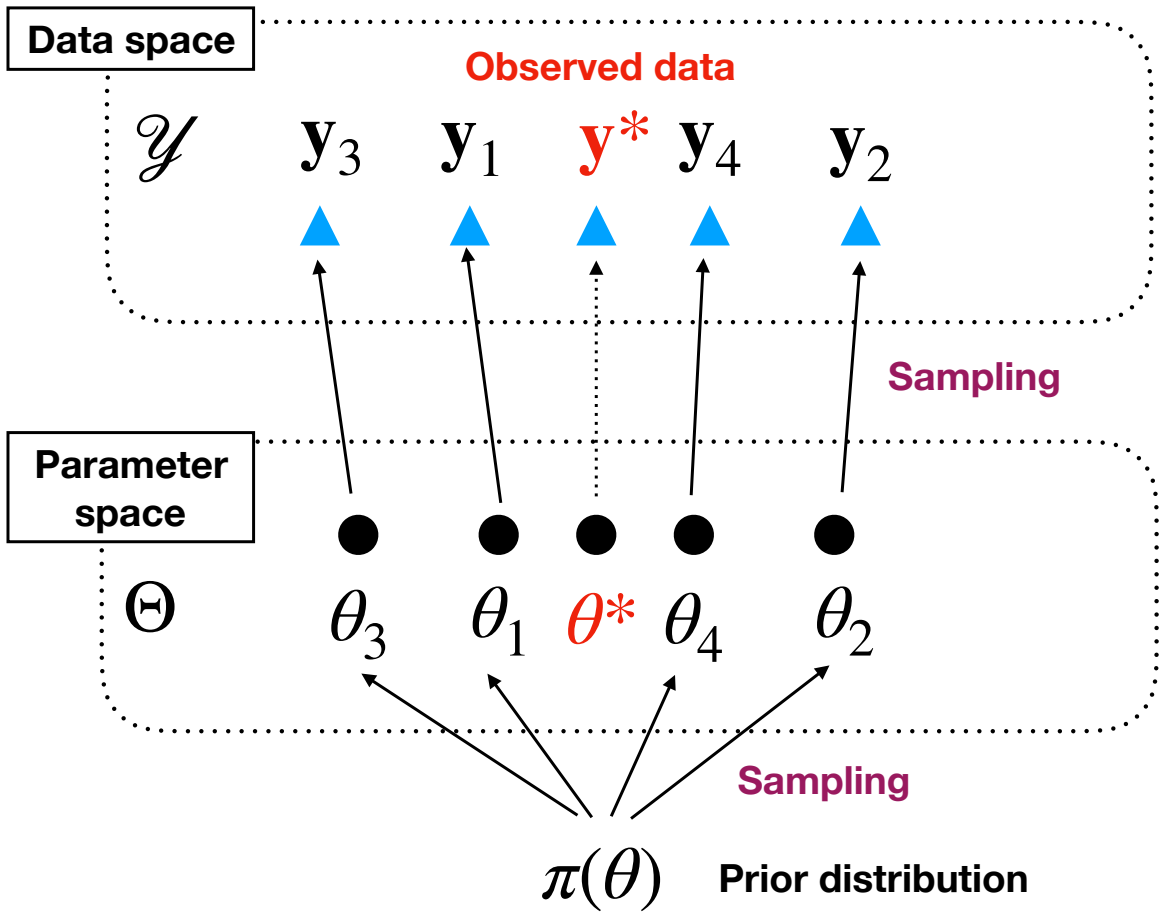
Output: An estimate of the posterior kernel mean:

$$\begin{aligned} \int k_{\Theta}(\cdot, \theta) p(\theta|\boldsymbol{y}^*) d\theta &\approx \sum_{i=1}^{n} w_i(\boldsymbol{y}^*) k_{\Theta}(\cdot, \theta_i), \\ p(\theta|\boldsymbol{y}^*) &\propto p(\boldsymbol{y}^*|\theta)\pi(\theta). \end{aligned}$$

Kernel ABC: The Sampling Step

1. **Sampling**: Generate parameter-data pairs from the model:

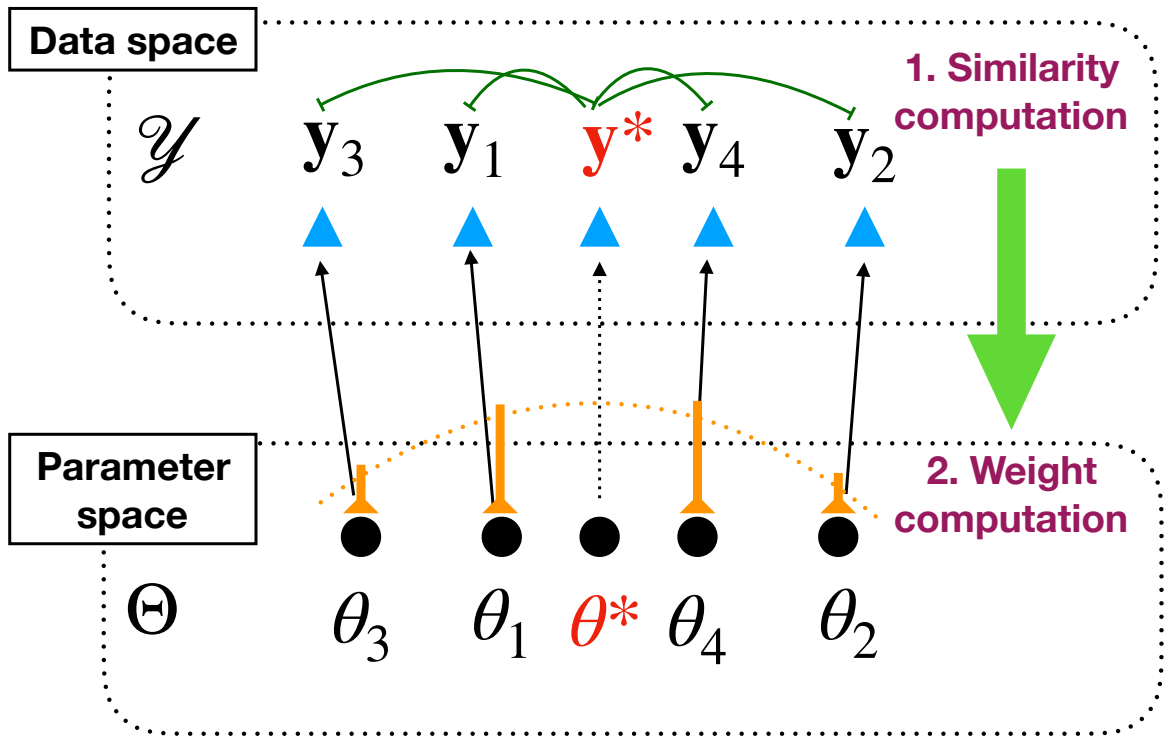
$$(\theta_1, \boldsymbol{y}_1), \ldots, (\theta_n, \boldsymbol{y}_n) \sim p(\boldsymbol{y}|\theta)\pi(\theta), \quad \text{i.i.d.}$$



Kernel ABC: The Weight Computation Step

2. Weight computation: Given observed data $\boldsymbol{y}^*$, compute

1. Similarities: $\boldsymbol{k}_Y(\boldsymbol{y}^*) = (k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_1), \ldots, k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_n))^\top$,
2. Weights: $(w_1(\boldsymbol{y}^*), \ldots, w_n(\boldsymbol{y}^*))^\top = (G_Y + n\lambda I_n)^{-1} \boldsymbol{k}_Y(\boldsymbol{y}^*)$.



$$\int k_\Theta(\cdot, \theta) p(\theta|\boldsymbol{y}^*) d\theta \approx \sum_{i=1}^{n} w_i(\boldsymbol{y}^*) k_\Theta(\cdot, \theta_i).$$

Kernel Herding [Chen et al., 2010]

Let

– P be a known probability distribution on Θ; and

– $\mu_P = \int k_\Theta(\cdot, \theta) dP(\theta)$ be its kernel mean.

Kernel Herding [Chen et al., 2010]

Let

– P be a known probability distribution on Θ; and

– $\mu_P = \int k_\Theta(\cdot, \theta) dP(\theta)$ be its kernel mean.

Kernel herding is a deterministic sampling method that

Kernel Herding [Chen et al., 2010]

Let
– P be a known probability distribution on Θ; and
– $\mu_P = \int k_\Theta(\cdot, \theta) dP(\theta)$ be its kernel mean.

Kernel herding is a deterministic sampling method that
– sequentially generates sample points $\theta'_1, \ldots, \theta'_n$ from P as

$$\theta'_1 \quad := \quad \arg\max_{\theta \in \Theta} \mu_P(\theta),$$

Kernel Herding [Chen et al., 2010]

Let
– P be a known probability distribution on Θ; and
– $\mu_P = \int k_\Theta(\cdot, \theta) dP(\theta)$ be its kernel mean.

Kernel herding is a deterministic sampling method that
– sequentially generates sample points $\theta'_1, \ldots, \theta'_n$ from P as

$$\theta'_1 := \arg\max_{\theta \in \Theta} \mu_P(\theta),$$

$$\theta'_T := \arg\max_{\theta \in \Theta} \underbrace{\mu_P(\theta)}_{\text{mode seeking}} - \frac{1}{T} \sum_{\ell=1}^{T-1} \underbrace{k_\Theta(\theta, \theta'_\ell)}_{\text{repulsive force}} \quad (T = 2, \ldots, n).$$

Kernel Herding [Chen et al., 2010]

Let

– P be a known probability distribution on Θ; and

– $\mu_P = \int k_\Theta(\cdot, \theta) dP(\theta)$ be its kernel mean.

Kernel herding is a deterministic sampling method that

– sequentially generates sample points $\theta'_1, \ldots, \theta'_n$ from P as

$$\theta'_1 := \arg\max_{\theta \in \Theta} \mu_P(\theta),$$

$$\theta'_T := \arg\max_{\theta \in \Theta} \underbrace{\mu_P(\theta)}_{\text{mode seeking}} - \frac{1}{T} \sum_{\ell=1}^{T-1} \underbrace{k_\Theta(\theta, \theta'_\ell)}_{\text{repulsive force}} \quad (T = 2, \ldots, n).$$

– is equivalent to greedily approximating the kernel mean μ_P:

$$\theta'_T = \arg\min_{\theta \in \Theta} \left\| \mu_P - \frac{1}{T} \left(k_\Theta(\cdot, \theta) + \sum_{i=1}^{T-1} k_\Theta(\cdot, \theta'_i) \right) \right\|_{\mathcal{H}_\Theta},$$

if k_Θ is shift-invariant. ($\mathcal{H}_\Theta$ is the RKHS of k_Θ.)

Kernel Herding [Chen et al., 2010]

Red squares: Sample points generated from kernel herding

Purple circles: Randomly generated i.i.d. sample points.

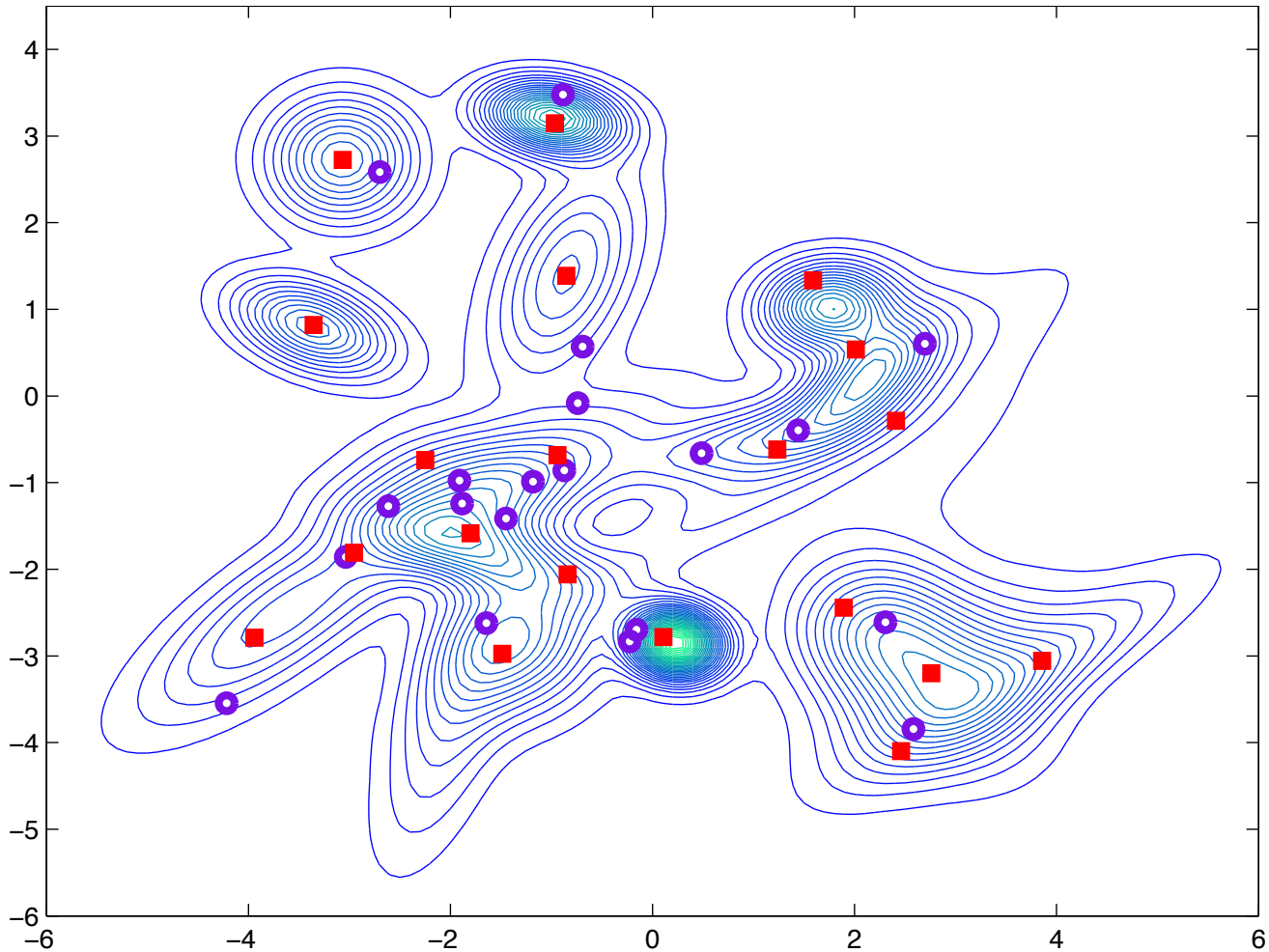


Figure 3: [Chen et al., 2010, Fig 1]

Proposed Method: Kernel Recursive ABC (Algorithm)

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedure:

Proposed Method: Kernel Recursive ABC (Algorithm)

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedure:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

Proposed Method: Kernel Recursive ABC (Algorithm)

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedure:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

– Generate pseudo-data from each θ_i:

$$\boldsymbol{y}_i \sim p(\boldsymbol{y}|\theta_i) \quad (i = 1, \ldots, n),$$

Proposed Method: Kernel Recursive ABC (Algorithm)

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedure:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

– Generate pseudo-data from each θ_i:

$$\boldsymbol{y}_i \sim p(\boldsymbol{y}|\theta_i) \quad (i = 1, \ldots, n),$$

– Compute weights for $\theta_1, \ldots, \theta_n$:

$$\boldsymbol{k}_Y(\boldsymbol{y}^*) = (k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_1), \ldots, k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_n))^\top,$$
$$(w_1(\boldsymbol{y}^*), \ldots, w_n(\boldsymbol{y}^*))^\top = (G_Y + n\lambda I_n)^{-1} \boldsymbol{k}_Y(\boldsymbol{y}^*).$$

Proposed Method: Kernel Recursive ABC (Algorithm)

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedure:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

– Generate pseudo-data from each θ_i:

$$\boldsymbol{y}_i \sim p(\boldsymbol{y}|\theta_i) \quad (i = 1, \ldots, n),$$

– Compute weights for $\theta_1, \ldots, \theta_n$:

$$\boldsymbol{k}_Y(\boldsymbol{y}^*) = (k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_1), \ldots, k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_n))^\top,$$
$$(w_1(\boldsymbol{y}^*), \ldots, w_n(\boldsymbol{y}^*))^\top = (G_Y + n\lambda I_n)^{-1} \boldsymbol{k}_Y(\boldsymbol{y}^*).$$

2. Kernel Herding: Sampling from $\hat{\mu}_{P_N} := \sum_{i=1}^n w_i(\boldsymbol{y}^*) k_\Theta(\cdot, \theta_i)$:

$$\theta'_T := \arg\max_{\theta \in \Theta} \hat{\mu}_{P_N}(\theta) - \frac{1}{T} \sum_{\ell=1}^{T-1} k(\theta, \theta'_\ell) \quad (T = 1, \ldots, n).$$

Proposed Method: Kernel Recursive ABC (Algorithm)

For $N = 1, 2, \ldots, N_{\mathrm{iter}}$, iterate the following procedure:

1. Kernel ABC: If $N = 1$: generate $\theta_1, \ldots, \theta_n \sim \pi(\theta)$, i.i.d.

– Generate pseudo-data from each θ_i:

$$\boldsymbol{y}_i \sim p(\boldsymbol{y}|\theta_i) \quad (i = 1, \ldots, n),$$

– Compute weights for $\theta_1, \ldots, \theta_n$:

$$\boldsymbol{k}_Y(\boldsymbol{y}^*) = (k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_1), \ldots, k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_n))^\top,$$
$$(w_1(\boldsymbol{y}^*), \ldots, w_n(\boldsymbol{y}^*))^\top = (G_Y + n\lambda I_n)^{-1} \boldsymbol{k}_Y(\boldsymbol{y}^*).$$

2. Kernel Herding: Sampling from $\hat{\mu}_{P_N} := \sum_{i=1}^n w_i(\boldsymbol{y}^*) k_\Theta(\cdot, \theta_i)$:

$$\theta'_T := \arg\max_{\theta \in \Theta} \hat{\mu}_{P_N}(\theta) - \frac{1}{T} \sum_{\ell=1}^{T-1} k(\theta, \theta'_\ell) \quad (T = 1, \ldots, n).$$

Set: $N \leftarrow N + 1$ and $(\theta_1, \ldots, \theta_n) \leftarrow (\theta'_1, \ldots, \theta'_n)$

Why Kernels?

– The combination of Kernel ABC and Kernel Herding leads to robustness against misspecfication of the prior $\pi(\theta)$.

Outline

Prior Misspecification

Assume that

- there is a **"true" parameter** θ^* such that

$$\boldsymbol{y}^* \sim p(\boldsymbol{y}|\theta^*)$$

- but you **don't know** much about θ^*.

Prior Misspecification

Assume that

- there is a **"true" parameter** θ^* such that

$$\boldsymbol{y}^* \sim p(\boldsymbol{y}|\theta^*)$$

- but you **don't know** much about θ^*.

In such a case, you may **misspecify the prior** $\pi(\theta)$.

Prior Misspecification

Assume that

- there is a **"true" parameter** θ^* such that

$$\boldsymbol{y}^* \sim p(\boldsymbol{y}|\theta^*)$$

- but you **don't know** much about θ^*.

In such a case, you may **misspecify the prior** $\pi(\theta)$.

- e.g., the support of $\pi(\theta)$ may not contain θ^*.

Prior Misspecification

Assume that

- there is a **"true" parameter** θ^* such that

$$\boldsymbol{y}^* \sim p(\boldsymbol{y}|\theta^*)$$

- but you **don't know** much about θ^*.

In such a case, you may **misspecify the prior** $\pi(\theta)$.

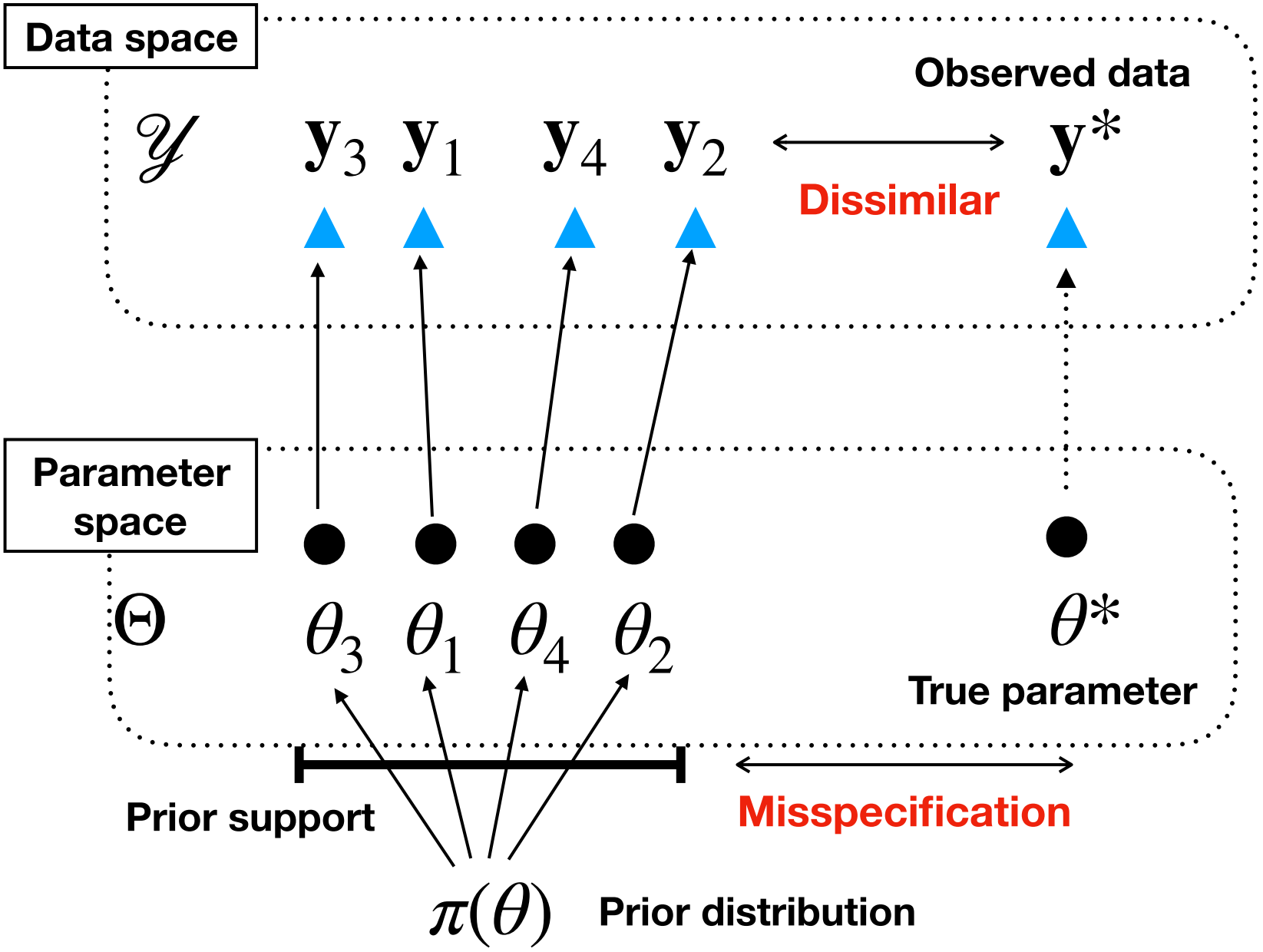
- e.g., the support of $\pi(\theta)$ may not contain θ^*.

As a result, simulated data

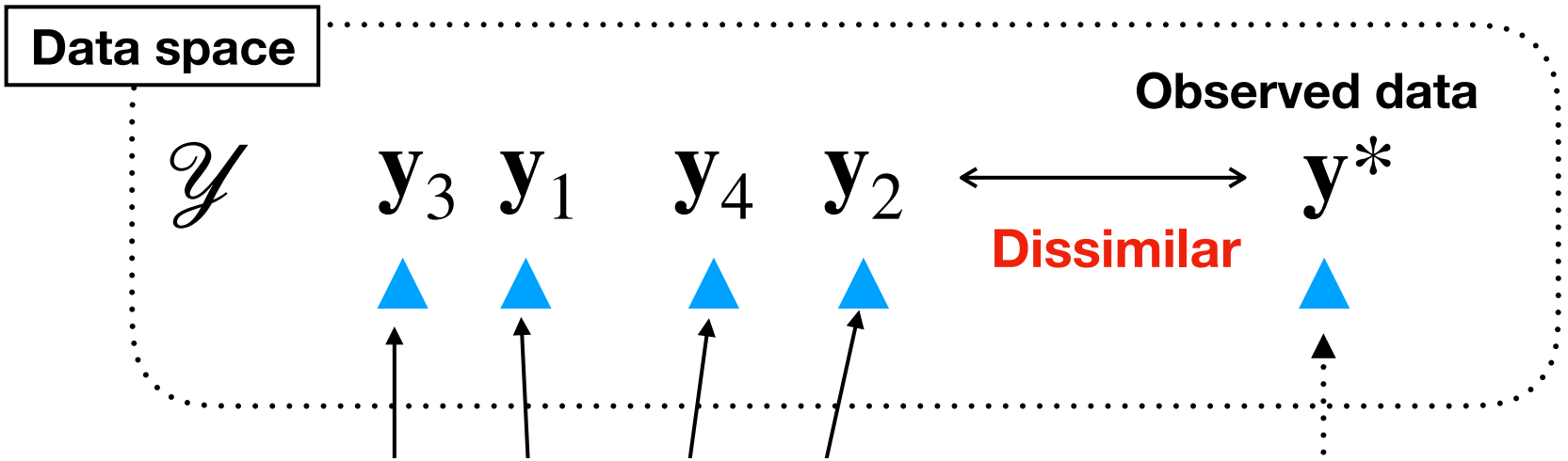
$$\boldsymbol{y}_i \sim p(\boldsymbol{y}|\theta_i), \quad \theta_i \sim \pi(\theta) \quad (i = 1, \ldots, n).$$

may become **far apart** from observed data $\boldsymbol{y}^*$.

Prior Misspecification

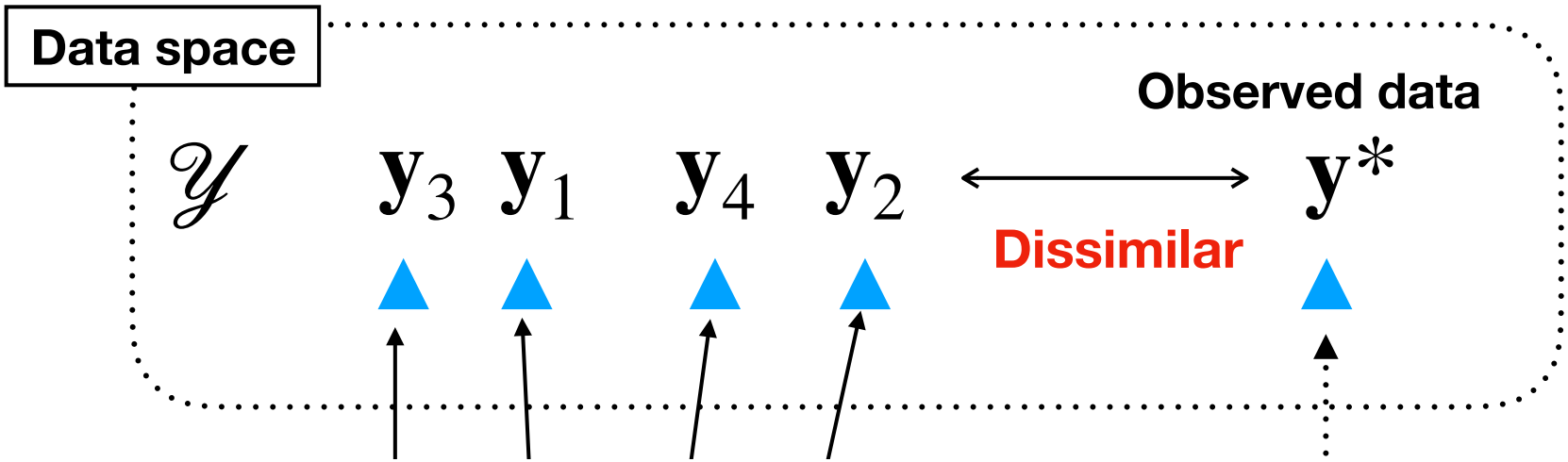


Auto-Correction Mechanism: The Kernel ABC Step



– Recall $k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_i)$ quantifies the **similarity** between $\boldsymbol{y}^*$ and $\boldsymbol{y}_i$.

Auto-Correction Mechanism: The Kernel ABC Step

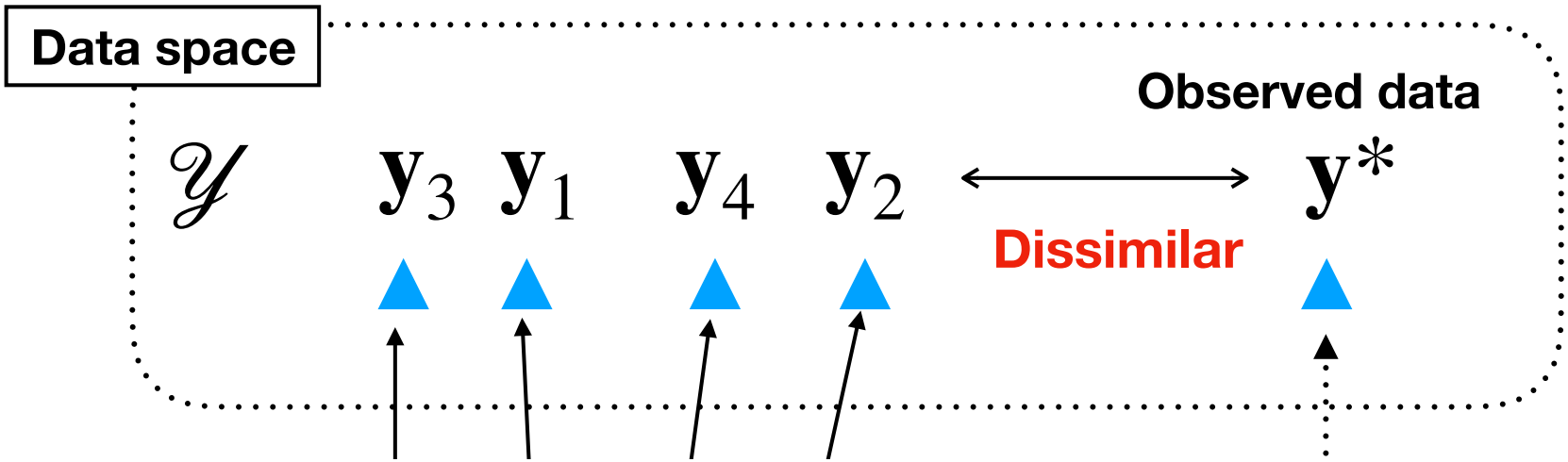


– Recall $k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_i)$ quantifies the **similarity** between $\boldsymbol{y}^*$ and $\boldsymbol{y}_i$.

— e.g. a Gaussian kernel:

$$k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_i) = \exp(-\mathrm{dist}^2(\boldsymbol{y}^*, \boldsymbol{y}_i)/\gamma^2)$$

Auto-Correction Mechanism: The Kernel ABC Step



– Recall $k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_i)$ quantifies the **similarity** between $\boldsymbol{y}^*$ and $\boldsymbol{y}_i$.

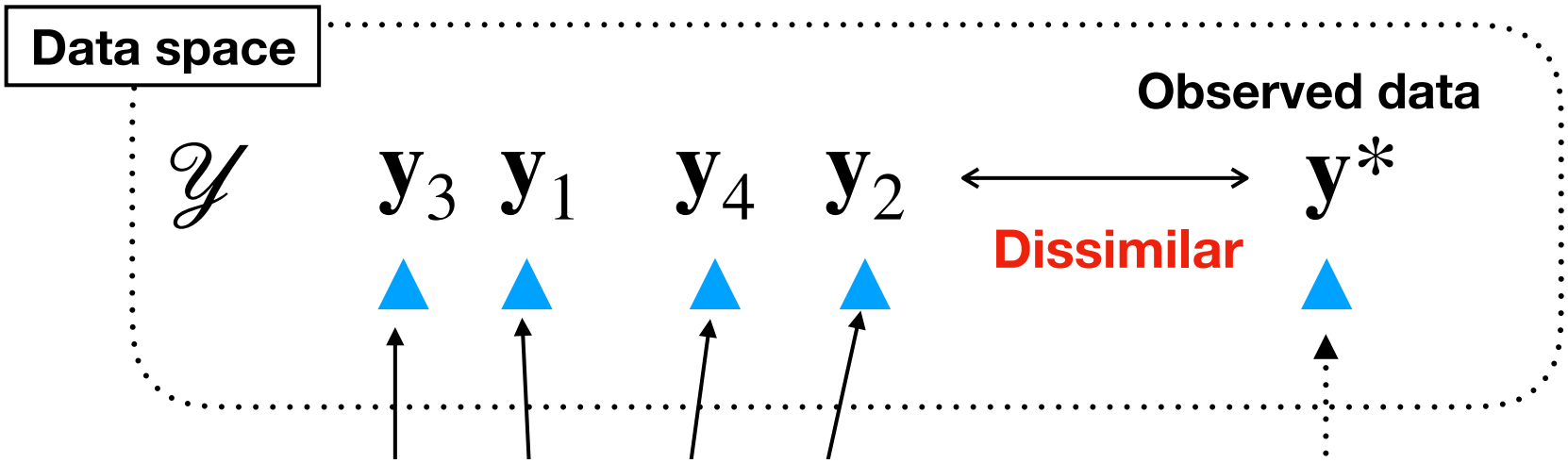
— e.g. a Gaussian kernel:

$$k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_i) = \exp(-\mathrm{dist}^2(\boldsymbol{y}^*, \boldsymbol{y}_i)/\gamma^2)$$

– Therefore, if $\boldsymbol{y}^*$ and each $\boldsymbol{y}_i$ are **dissimilar**, we have

$$\boldsymbol{k}_Y(\boldsymbol{y}^*) = (k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_1), \ldots, k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_n))^\top \approx \mathbf{0}$$

Auto-Correction Mechanism: The Kernel ABC Step



– Recall $k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_i)$ quantifies the **similarity** between $\boldsymbol{y}^*$ and $\boldsymbol{y}_i$.
— e.g. a Gaussian kernel:

$$k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_i) = \exp(-\mathrm{dist}^2(\boldsymbol{y}^*, \boldsymbol{y}_i)/\gamma^2)$$

– Therefore, if $\boldsymbol{y}^*$ and each $\boldsymbol{y}_i$ are **dissimilar**, we have

$$\boldsymbol{k}_Y(\boldsymbol{y}^*) = (k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_1), \ldots, k_{\mathcal{Y}}(\boldsymbol{y}^*, \boldsymbol{y}_n))^\top \approx \mathbf{0}$$

– As a result, the weights by Kernel ABC become

$$(w_1(\boldsymbol{y}^*), \ldots, w_n(\boldsymbol{y}^*))^\top = (G_Y + n\lambda I_n)^{-1} \boldsymbol{k}_Y(\boldsymbol{y}^*) \approx \mathbf{0}$$

Auto-Correction Mechanism: The Kernel Herding Step

Deterministically sample $\theta'_1, \ldots, \theta'_n$

$$\theta'_1 := \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} w_i(\boldsymbol{y}^*) k(\theta, \theta_i)$$

Auto-Correction Mechanism: The Kernel Herding Step

Deterministically sample $\theta'_1, \ldots, \theta'_n$

$$\theta'_1 := \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} w_i(\boldsymbol{y}^*) k(\theta, \theta_i)$$

For $T = 2, \ldots, n$,

$$\theta'_T := \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \underbrace{w_i(\boldsymbol{y}^*)}_{\approx 0} k(\theta, \theta_i) - \frac{1}{T} \sum_{\ell=1}^{T-1} k(\theta, \theta'_\ell)$$

Auto-Correction Mechanism: The Kernel Herding Step

Deterministically sample $\theta'_1, \ldots, \theta'_n$

$$\theta'_1 := \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} w_i(\boldsymbol{y}^*) k(\theta, \theta_i)$$

For $T = 2, \ldots, n$,

$$\begin{aligned} \theta'_T &:= \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \underbrace{w_i(\boldsymbol{y}^*)}_{\approx 0} k(\theta, \theta_i) - \frac{1}{T} \sum_{\ell=1}^{T-1} k(\theta, \theta'_\ell) \\ &\approx \arg\min_{\theta \in \Theta} \sum_{\ell=1}^{T-1} k(\theta, \theta'_\ell) \end{aligned}$$

Auto-Correction Mechanism: The Kernel Herding Step

Deterministically sample $\theta'_1, \ldots, \theta'_n$

$$\theta'_1 := \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} w_i(\boldsymbol{y}^*) k(\theta, \theta_i)$$

For $T = 2, \ldots, n$,

$$\begin{aligned} \theta'_T &:= \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \underbrace{w_i(\boldsymbol{y}^*)}_{\approx 0} k(\theta, \theta_i) - \frac{1}{T} \sum_{\ell=1}^{T-1} k(\theta, \theta'_\ell) \\ &\approx \arg\min_{\theta \in \Theta} \sum_{\ell=1}^{T-1} k(\theta, \theta'_\ell) \end{aligned}$$

Therefore, θ'_T is chosen to be distant from $\theta'_1, \ldots, \theta'_{T-1}$.

Auto-Correction Mechanism: A Numerical Illustration

– Parameter space: $\Theta = \mathbb{R}$.

Auto-Correction Mechanism: A Numerical Illustration

- Parameter space: $\Theta = \mathbb{R}$.
- Observed data $\boldsymbol{y}^* = \{y_1, \ldots, y_{100}\} \subset \mathbb{R}$, where

$$y_1, \ldots, y_{100} \sim \underbrace{\mathcal{N}(\theta^*, 40)}_{\text{Normal dist.}}, \text{ i.i.d.} \quad \text{with} \quad \underbrace{\theta^* = 0}_{\text{Unknown, to be estimated}} .$$

Auto-Correction Mechanism: A Numerical Illustration

– Parameter space: $\Theta = \mathbb{R}$.

– Observed data $\boldsymbol{y}^* = \{y_1, \ldots, y_{100}\} \subset \mathbb{R}$, where

$$y_1, \ldots, y_{100} \sim \underbrace{\mathcal{N}(\theta^*, 40)}_{\text{Normal dist.}}, \text{ i.i.d.} \quad \text{with} \quad \underbrace{\theta^* = 0}_{\text{Unknown, to be estimated}} .$$

– Assume your prior about θ^* is **severely misspecified**: let

$$\pi(\theta) = \text{uniform}[2000, 3000].$$

(The support of $\pi(\theta)$ does not contain θ^*.)

Auto-Correction Mechanism: A Numerical Illustration

– Parameter space: $\Theta = \mathbb{R}$.

– Observed data $\boldsymbol{y}^* = \{y_1, \ldots, y_{100}\} \subset \mathbb{R}$, where

$$y_1, \ldots, y_{100} \sim \underbrace{\mathcal{N}(\theta^*, 40)}_{\text{Normal dist.}}, \text{ i.i.d. with } \underbrace{\theta^* = 0}_{\text{Unknown, to be estimated}}.$$

– Assume your prior about θ^* is **severely misspecified**: let

$$\pi(\theta) = \text{uniform}[2000, 3000].$$

(The support of $\pi(\theta)$ does not contain θ^*.)

We applied the proposed method to estimate θ^*, with

Auto-Correction Mechanism: A Numerical Illustration

– Parameter space: $\Theta = \mathbb{R}$.
– Observed data $\boldsymbol{y}^* = \{y_1, \ldots, y_{100}\} \subset \mathbb{R}$, where

$$y_1, \ldots, y_{100} \sim \underbrace{\mathcal{N}(\theta^*, 40)}_{\text{Normal dist.}}, \text{ i.i.d. with } \underbrace{\theta^* = 0}_{\text{Unknown, to be estimated}}.$$

– Assume your prior about θ^* is **severely misspecified**: let

$$\pi(\theta) = \text{uniform}[2000, 3000].$$

(The support of $\pi(\theta)$ does not contain θ^*.)

We applied the proposed method to estimate θ^*, with
– k_Θ, $k_\mathcal{Y}$ being Gaussian, the latter based on the energy distance [Székely and Rizzo, 2013].

Auto-Correction Mechanism: A Numerical Illustration

– Parameter space: $\Theta = \mathbb{R}$.
– Observed data $\boldsymbol{y}^* = \{y_1, \ldots, y_{100}\} \subset \mathbb{R}$, where

$$y_1, \ldots, y_{100} \sim \underbrace{\mathcal{N}(\theta^*, 40)}_{\text{Normal dist.}}, \text{ i.i.d. with } \underbrace{\theta^* = 0}_{\text{Unknown, to be estimated}}.$$

– Assume your prior about θ^* is **severely misspecified**: let

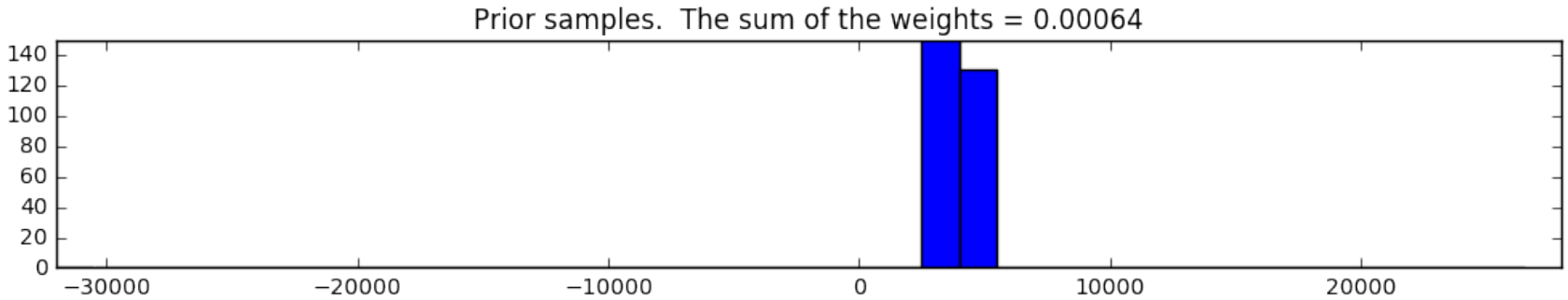
$$\pi(\theta) = \text{uniform}[2000, 3000].$$

(The support of $\pi(\theta)$ does not contain θ^*.)

We applied the proposed method to estimate θ^*, with
– k_Θ, $k_\mathcal{Y}$ being Gaussian, the latter based on the energy distance [Székely and Rizzo, 2013].
In each iteration, 300 $(\theta_i, \boldsymbol{y}_i)$-pairs are simulated.

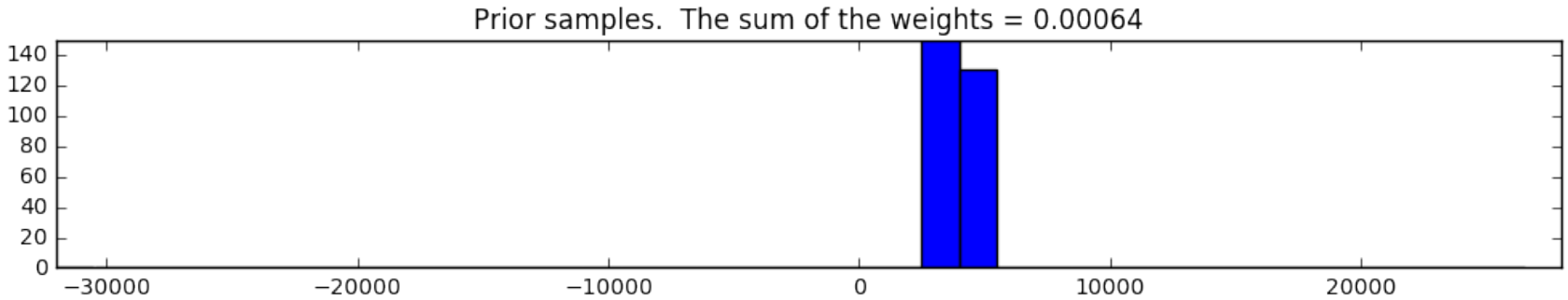
Auto-Correction Mechanism: A Nummerical Illustration

Initial sampling from the prior $\pi(\theta) = \text{uniform}[2000, 3000]$:
(Recall $\theta^* = 0$.)

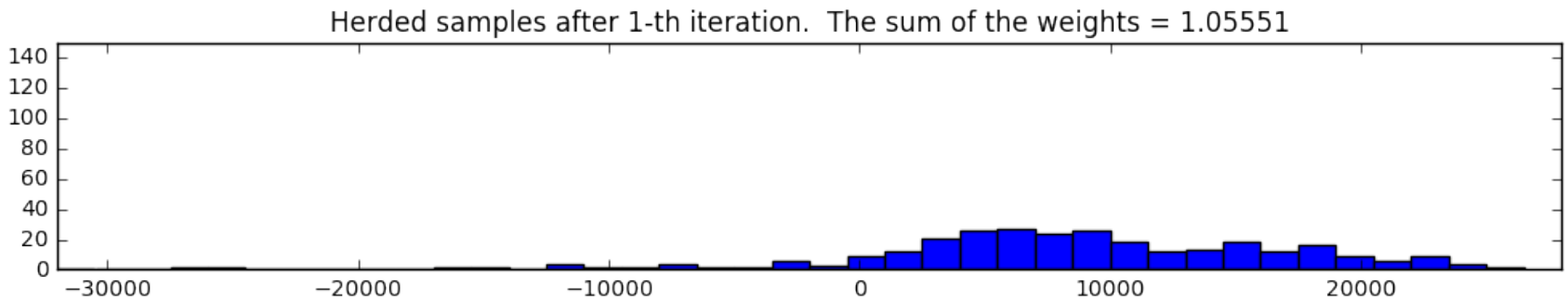


Auto-Correction Mechanism: A Nummerical Illustration

Initial sampling from the prior $\pi(\theta) = \text{uniform}[2000, 3000]$:
(Recall $\theta^* = 0$.)

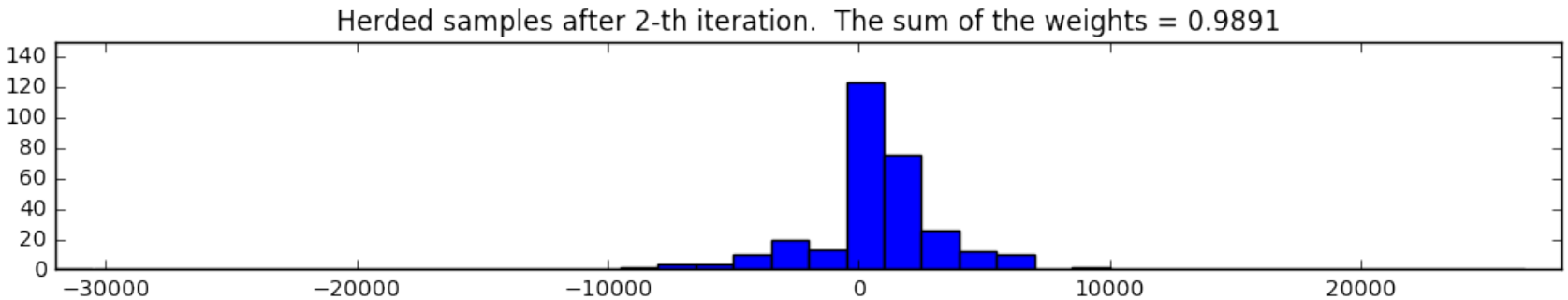


After **1 recursion** of Kernel ABC + Kernel Herding:



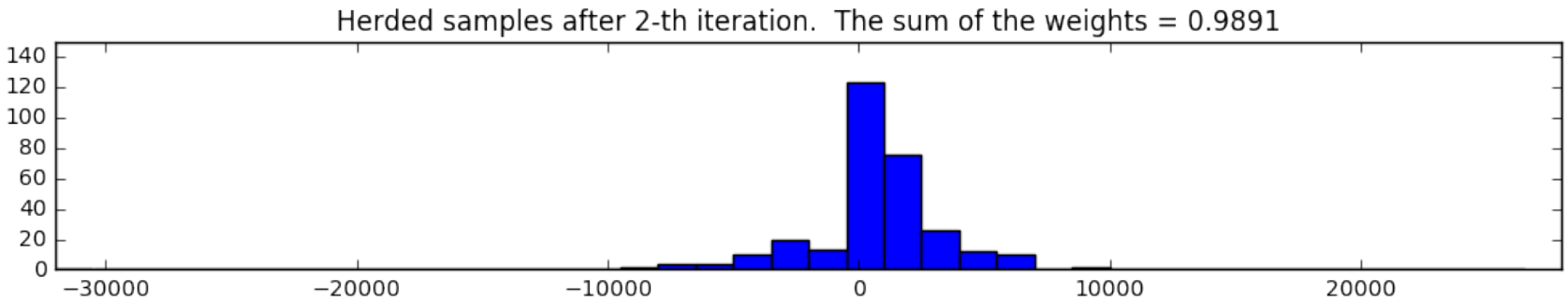
Auto-Correction Mechanism: A Nummerical Illustration

After **2 recursions** of Kernel ABC + Kernel Herding:

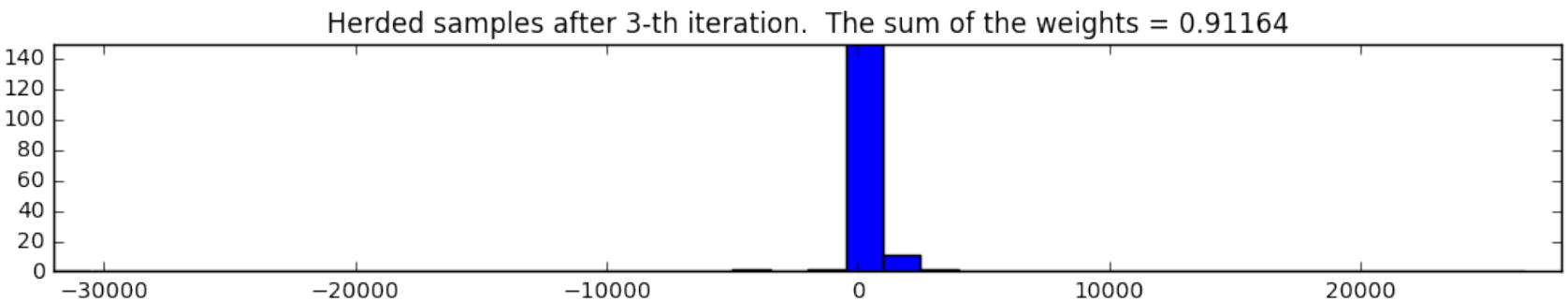


Auto-Correction Mechanism: A Nummerical Illustration

After **2 recursions** of Kernel ABC + Kernel Herding:



After **3 recursions** of Kernel ABC + Kernel Herding:



Outline

Summary

The proposed method outperformed competitors in most cases ...

Please look at the paper for details!!

Point Estimation with a Misspecified Prior Distribution

The task: Estimate the **mean vector** of a 20-dim Gaussian distribution using **a severely misspecified prior**.

Point Estimation with a Misspecified Prior Distribution

The task: Estimate the **mean vector** of a 20-dim Gaussian distribution using **a severely misspecified prior**.

– **The true mean vector:** $\mu \in [0, 2000]^{20} \subset \mathbb{R}^{20}$.

Point Estimation with a Misspecified Prior Distribution

The task: Estimate the **mean vector** of a 20-dim Gaussian distribution using **a severely misspecified prior**.

– **The true mean vector:** $\mu \in [0, 2000]^{20} \subset \mathbb{R}^{20}$.

– **The prior:** the uniform distribution on $[9 \times 10^6, 10^7]^{20} \subset \mathbb{R}^{20}$.

Point Estimation with a Misspecified Prior Distribution

The task: Estimate the **mean vector** of a 20-dim Gaussian distribution using **a severely misspecified prior**.

– **The true mean vector:** $\mu \in [0, 2000]^{20} \subset \mathbb{R}^{20}$.

– **The prior:** the uniform distribution on $[9 \times 10^6, 10^7]^{20} \subset \mathbb{R}^{20}$.

Algorithm	parameter error	data error	cputime
KR-ABC	0.70(0.29)	0.008(0.004)	866.02(26.12)
KR-ABC (less samples)	7.22(3.28)	0.02(0.24)	353.498(23.05)
K2-ABC	>1e+6 (>1e+3)	>1e+5 (>1e+3)	209.51(11.49)
K-ABC	>1e+6 (>1e+3)	>1e+5 (>1e+3))	403.93(24.97)
SMC-ABC (mean)	>1e+6 (>1e+3)	>1e+5 (>1e+3)	590.41(29.54)
SMC-ABC (MAP)	>1e+6 (>1e+3)	>1e+5 (>1e+3)	590.41(29.54)
ABC-DC	>1e+6 (>1e+3)	>1e+5 (>1e+3)	313.99(16.85)
BO	>1e+5(>1e+4)	>1e+5 (>1e+4)	25940.86(936.40)
MSM	>1e+5(>1e+4)	>1e+5(>1e+4)	307.42(67.94)

Parameter Estimation of a Pedestrian Flow Simulator [Yamashita et al., 2010]

Estimate certain parameters characterising groups of pedestrians.

Figure 4: Points representing individual pedestrians. (red = slow)

Parameter Estimation of a Pedestrian Flow Simulator

Algorithm	$\theta^{(N)}$ error	$\theta^{(T)}$ error	data error	cputime
KR-ABC	61.58(74.42)	70.93(102.08)	0.008(0.009)	2233.45(97.54)
KR-ABC (less samples)	82.46(75.05)	134.00(161.85)	0.014(0.014)	1875.32(147.16)
K2-ABC	298.94(120.71)	308.95(109.43)	0.10(0.10)	1547.32(56.31)
K-ABC	354.72(145.76)	389.52(140.91)	0.12(0.09)	1773.74(84.91)
SMC-ABC (mean)	271.51(104.64)	363.12(91.28)	0.09(0.07)	2017.89(110.02)
SMC-ABC (MAP)	255.15(139.33)	348.43(104.74)	0.09(0.1)	2017.89(110.02)
ABC-DC	273.93(136.14)	327.48(98.12)	0.09(0.14)	1984.43(59.12)
BO	194.57(65.83)	291.73(105.33)	0.04(0.06)	37541.23(3047.46)
MSM	453.58(89.43)	510.04(55.10)	0.24(0.17)	1869.83(49.51)

Outline

Conclusions

We proposed the **Kernel Recursive ABC**, a method for point estimation of simulator-based statistical models that is **robust to misspecification of a prior distribution**.

Conclusions

We proposed the **Kernel Recursive ABC**, a method for point estimation of simulator-based statistical models that is **robust to misspecification of a prior distribution**.

Extension to Model Selection:
Model Selection for Simulator-based Statistical Models: A Kernel Approach (ArXiv, 2019)
T. Kajihara and M. Kanagawa and Y. Nakaguchi and K. Khandelwal and K. Fukumizu.

Conclusions

We proposed the **Kernel Recursive ABC**, a method for point estimation of simulator-based statistical models that is **robust to misspecification of a prior distribution**.

Extension to Model Selection:
Model Selection for Simulator-based Statistical Models: A Kernel Approach (ArXiv, 2019)
T. Kajihara and M. Kanagawa and Y. Nakaguchi and K. Khandelwal and K. Fukumizu.

– Perform model selection by mixture modelling, using the Kernel Recursive ABC.

Conclusions

We proposed the **Kernel Recursive ABC**, a method for point estimation of simulator-based statistical models that is **robust to misspecification of a prior distribution**.

Extension to Model Selection:
Model Selection for Simulator-based Statistical Models: A Kernel Approach (ArXiv, 2019)
T. Kajihara and M. Kanagawa and Y. Nakaguchi and K. Khandelwal and K. Fukumizu.

– Perform model selection by mixture modelling, using the Kernel Recursive ABC.

Future Work:
– Statistical convergence analysis.
– Scalability to large scale problems.

Collaborators

- Takafumi Kajihara (NEC/AIST/RIKEN)
- Keisuke Yamazaki (AIST)
- Kenji Fukumizu (ISM)
- Yuuki Nakaguchi (NEC)
- Kanishk Khandelwal (NEC)

Chen, Y., Welling, M., and Smola, A. (2010).
Supersamples from kernel-herding.
In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI 2010)*, pages 109–116.

Fukumizu, K., Gretton, A., Sun, X., and Schölkopf, B. (2008).
Kernel measures of conditional dependence.
In *Advances in Neural Information Processing Systems 20*, pages 489–496.

Gutmann, M. U. and Corander, J. (2016).
Bayesian optimization for likelihood-free inference of simulator-based statistical models.
Journal of Machine Learning Research, 17(125):1–47.

Lele, S. R., Nadeem, K., , and Schmuland, B. (2010).
Estimability and likelihood inference for generalized linear mixed models using data cloning.
Journal of the American Statistical Association, 105(492):1617–1625.


Muandet, K., Fukumizu, K., Sriperumbudur, B. K., and Schölkopf, B. (2017).
Kernel mean embedding of distributions : A review and beyond.
Foundations and Trends in Machine Learning, 10(1–2):1–141.


Nakagome, S., Fukumizu, K., and Mano, S. (2013).
Kernel approximate Bayesian computation in population genetic inferences.
Statistical Applications in Genetics and Molecular Biology, 12(6):667–678.


Saito, T. (2019).
Tsunami Generation and Propagation.
Springer.


Sisson, S. A., Fan, Y., and Beaumont, M. (2018).
Handbook of Approximate Bayesian Computation.
Chapman and Hall/CRC.


Smola, A., Gretton, A., Song, L., and Schölkopf, B. (2007).
A Hilbert space embedding for distributions.

In *Proceedings of the International Conference on Algorithmic Learning Theory*, volume 4754, pages 13–31. Springer.


Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. (2010).
Hilbert space embeddings and metrics on probability measures.
Jounal of Machine Learning Research, 11:1517–1561.


Székely, G. J. and Rizzo, M. L. (2013).
Energy statistics: A class of statistics based on distances.
Journal of Statistical Planning and Inference, 143:1249–1272.


Yamashita, T., Soeda, S., and Noda, I. (2010).
Assistance of evacuation planning with high-speed network model-based pedestrian simulator.
In *Proceedings of Fifth International Conference on Pedestrian and Evacuation Dynamics (PED 2010)*, page 58. PED 2010.