Model fitting like a boss

Luigi Acerbi

Department of Basic Neuroscience, University of Geneva
Center for Neural Science, New York University
International Brain Lab

September 9, 2019

What is a model?

What is a model?

The best material model of a cat is another, or preferably the same, cat.

Wiener, *Philosophy of Science* (1945) (with Rosenblueth)

What is a mathematical model?

- Quantitative stand-in for a theory

What is a mathematical model?

- Quantitative stand-in for a theory
- A *family of probability distributions* over possible datasets:

$$p\left(\text{data}|\boldsymbol{\theta}\right)$$

  - data is a dataset with n data points (e.g., trials)
  - $\boldsymbol{\theta}$ is a parameter vector

What is a mathematical model?

- Quantitative stand-in for a theory
- A *family of probability distributions* over possible datasets:

$$p\,(\text{data}|\boldsymbol{\theta})$$

  - data is a dataset with n data points (e.g., trials)
  - $\boldsymbol{\theta}$ is a parameter vector
- **Why?**

What is a mathematical model?

- Quantitative stand-in for a theory
- A *family of probability distributions* over possible datasets:

$$p\,(\text{data}|\boldsymbol{\theta})$$

  - data is a dataset with n data points (e.g., trials)
  - $\boldsymbol{\theta}$ is a parameter vector

- **Why?** Description, prediction, and explanation

What is a mathematical model?

- Quantitative stand-in for a theory
- A *family of probability distributions* over possible datasets:

$$p\left(\mathrm{data}|\boldsymbol{\theta}\right)$$

  - data is a dataset with n data points (e.g., trials)
  - $\boldsymbol{\theta}$ is a parameter vector

- **Why?** Description, prediction, and explanation
- Defining $p(\mathrm{data}|\boldsymbol{\theta})$ is the core of model building

What is a mathematical model?

- Quantitative stand-in for a theory
- A *family of probability distributions* over possible datasets:

$$p\left(\text{data}|\boldsymbol{\theta}\right)$$

  - data is a dataset with n data points (e.g., trials)
  - $\boldsymbol{\theta}$ is a parameter vector
- **Why?** Description, prediction, and explanation
- Defining $p(\text{data}|\boldsymbol{\theta})$ is the core of model building
  - Wait, what?

What is a mathematical model?

- Quantitative stand-in for a theory
- A *family of probability distributions* over possible datasets:

$$p\left(\text{data}|\boldsymbol{\theta}\right)$$

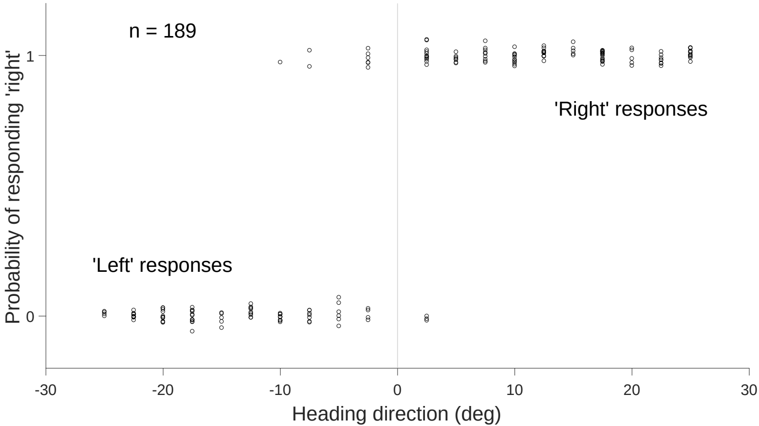
  - data is a dataset with n data points (e.g., trials)
  - $\boldsymbol{\theta}$ is a parameter vector
- **Why?** Description, prediction, and explanation
- Defining $p(\text{data}|\boldsymbol{\theta})$ is the core of model building
  - Wait, what?
- **How?** Think about the data generation process!

Example: Psychometric function

Task: heading direction 'discrimination' task

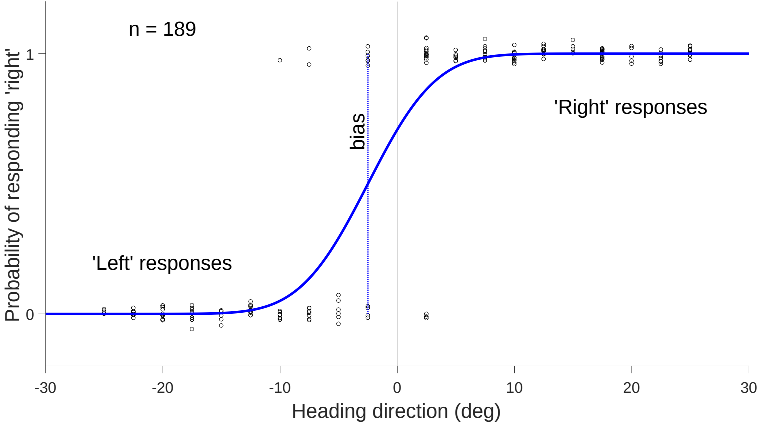
Example: Psychometric function

Task: heading direction 'discrimination' task



(data from Acerbi*, Dokka*, et al., *PLoS Comput Biol*, 2018)

Example: Psychometric function



- data: (heading direction, choice) for each trial
- parameters $\boldsymbol{\theta}$: $(\mu,\ \sigma,\ \lambda)$

The (log) likelihood

- $p(\text{data}|\boldsymbol{\theta})$ is a *probability density* as you vary data for a fixed $\boldsymbol{\theta}$
- $p(\text{data}|\boldsymbol{\theta})$ is the *likelihood*, a function of $\boldsymbol{\theta}$ for fixed data

The (log) likelihood

- For numerical reasons we work with $\log p(\text{data}|\boldsymbol{\theta})$

The (log) likelihood

- For numerical reasons we work with $\log p(\text{data}|\boldsymbol{\theta})$
- Using the rules of probability and logarithms:

$$
\begin{aligned}
\log p\left(\text{data}|\boldsymbol{\theta}\right) &= \log p(\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(n)} | \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(n)}, \boldsymbol{\theta}) \\
&= \log \prod_{i=1}^{n} p_i \left(\boldsymbol{r}^{(i)} | \boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i-1)}, \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(n)}, \boldsymbol{\theta} \right) \\
&= \sum_{i=1}^{n} \log p_i \left(\boldsymbol{r}^{(i)} | \boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i-1)}, \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(n)}, \boldsymbol{\theta} \right)
\end{aligned}
$$

The (log) likelihood

- For numerical reasons we work with $\log p(\text{data}|\boldsymbol{\theta})$
- Using the rules of probability and logarithms:

$$\begin{aligned}\log p\left(\text{data}|\boldsymbol{\theta}\right) &= \log p(\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(n)}|\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(n)}, \boldsymbol{\theta}) \\ &= \log \prod_{i=1}^{n} p_i \left(\boldsymbol{r}^{(i)}|\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i-1)}, \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(n)}, \boldsymbol{\theta}\right) \\ &= \sum_{i=1}^{n} \log p_i \left(\boldsymbol{r}^{(i)}|\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i-1)}, \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(n)}, \boldsymbol{\theta}\right)\end{aligned}$$

- Simplest case: $\log p\left(\text{data}|\boldsymbol{\theta}\right) = \sum_{i=1}^{n} \log p_i \left(\boldsymbol{r}^{(i)}|\boldsymbol{s}^{(i)}, \boldsymbol{\theta}\right)$

The (log) likelihood

- For numerical reasons we work with $\log p(\text{data}|\boldsymbol{\theta})$
- Using the rules of probability and logarithms:

$$\begin{aligned}
\log p\left(\text{data}|\boldsymbol{\theta}\right) &= \log p(\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(n)}|\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(n)}, \boldsymbol{\theta}) \\
&= \log \prod_{i=1}^{n} p_i\left(\boldsymbol{r}^{(i)}|\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i-1)}, \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(n)}, \boldsymbol{\theta}\right) \\
&= \sum_{i=1}^{n} \log p_i\left(\boldsymbol{r}^{(i)}|\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i-1)}, \boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(n)}, \boldsymbol{\theta}\right)
\end{aligned}$$

- Simplest case: $\log p\left(\text{data}|\boldsymbol{\theta}\right) = \sum_{i=1}^{n} \log p_i\left(\boldsymbol{r}^{(i)}|\boldsymbol{s}^{(i)}, \boldsymbol{\theta}\right)$
- Model building: Write function with
  - Input: $\boldsymbol{\theta}$ and data
  - Output: $\log p(\text{data}|\boldsymbol{\theta})$

Model fitting

Model fitting $\sim$ *statistical estimation* problem

Model fitting

Model fitting $\sim$ *statistical estimation* problem

1. Maximum likelihood estimation (MLE)

Model fitting

Model fitting $\sim$ *statistical estimation* problem

1. Maximum likelihood estimation (MLE)

- Find maximum of $p(\text{data}|\boldsymbol{\theta})$

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p(\text{data}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log p(\text{data}|\boldsymbol{\theta})$$

Model fitting

Model fitting $\sim$ *statistical estimation* problem

1. Maximum likelihood estimation (MLE)

- Find maximum of $p(\text{data}|\boldsymbol{\theta})$

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p(\text{data}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log p(\text{data}|\boldsymbol{\theta})$$

2. Bayesian posterior

$$p(\boldsymbol{\theta}|\text{data}) = \frac{p(\text{data}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\text{data})} \propto p(\text{data}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

Model fitting

Model fitting $\sim$ *statistical estimation* problem

1. Maximum likelihood estimation (MLE)

- Find maximum of $p(\text{data}|\boldsymbol{\theta})$

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p(\text{data}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log p(\text{data}|\boldsymbol{\theta})$$

2. Bayesian posterior

$$p(\boldsymbol{\theta}|\text{data}) = \frac{p(\text{data}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\text{data})} \propto p(\text{data}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

- For $n \to \infty$ converges to MLE (if $p(\hat{\boldsymbol{\theta}}_{\text{ML}}) \neq 0$)

Model fitting

Model fitting $\sim$ *statistical estimation* problem

1. Maximum likelihood estimation (MLE)

- Find maximum of $p(\text{data}|\boldsymbol{\theta})$

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p(\text{data}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log p(\text{data}|\boldsymbol{\theta})$$

2. Bayesian posterior

$$p(\boldsymbol{\theta}|\text{data}) = \frac{p(\text{data}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\text{data})} \propto p(\text{data}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

- For $n \to \infty$ converges to MLE (if $p(\hat{\boldsymbol{\theta}}_{\text{ML}}) \neq 0$)
- *Full posterior*: informative about parameter uncertainty and trade-offs

Model fitting

Model fitting $\sim$ *statistical estimation* problem

1. Maximum likelihood estimation (MLE)

- Find maximum of $p(\text{data}|\boldsymbol{\theta})$

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p(\text{data}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log p(\text{data}|\boldsymbol{\theta})$$

2. Bayesian posterior

$$p(\boldsymbol{\theta}|\text{data}) = \frac{p(\text{data}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\text{data})} \propto p(\text{data}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

- For $n \to \infty$ converges to MLE (if $p(\hat{\boldsymbol{\theta}}_{\text{ML}}) \neq 0$)
- *Full posterior*: informative about parameter uncertainty and trade-offs
- *Maximum-a-posteriori (MAP)*: $\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\text{data})$

How to do model fitting?

How to do model fitting?

Maximum likelihood estimation (MLE), Maximum-a-posteriori (MAP)

- Model fitting $\sim$ *optimization problem*

How to do model fitting?

Maximum likelihood estimation (MLE), Maximum-a-posteriori (MAP)

- Model fitting $\sim$ *optimization problem*

Bayesian posterior

- How do we represent/approximate an arbitrary posterior distribution?

How to do model fitting?

Maximum likelihood estimation (MLE), Maximum-a-posteriori (MAP)

- Model fitting $\sim$ *optimization problem*

Bayesian posterior

- How do we represent/approximate an arbitrary posterior distribution?
    1. Use a known (easier) distribution (*variational inference*)

How to do model fitting?

Maximum likelihood estimation (MLE), Maximum-a-posteriori (MAP)

- Model fitting $\sim$ *optimization problem*

Bayesian posterior

- How do we represent/approximate an arbitrary posterior distribution?
  1. Use a known (easier) distribution (*variational inference*)
  2. Use a bunch of discrete samples (*Markov-Chain Monte Carlo*)

Model fitting via optimization

- Find single $\boldsymbol{\theta}$ that best describes the data

Model fitting via optimization

- Find single $\boldsymbol{\theta}$ that best describes the data
- (For this section we switch notation from $\boldsymbol{\theta}$ to $\boldsymbol{x}$)

Model fitting via optimization

- Find single $\boldsymbol{\theta}$ that best describes the data
- (For this section we switch notation from $\boldsymbol{\theta}$ to $\boldsymbol{x}$)
- Given $\tilde{f}(\boldsymbol{x}) \equiv \begin{cases} \log p(\text{data}|\boldsymbol{x}) & \text{maximum likelihood} \\ \log p(\text{data}|\boldsymbol{x}) + \log p(\boldsymbol{x}) & \text{maximum-a-posteriori} \end{cases}$

Model fitting via optimization

- Find single $\boldsymbol{\theta}$ that best describes the data
- (For this section we switch notation from $\boldsymbol{\theta}$ to $\boldsymbol{x}$)
- Given $\tilde{f}(\boldsymbol{x}) \equiv \begin{cases} \log p(\text{data}|\boldsymbol{x}) & \text{maximum likelihood} \\ \log p(\text{data}|\boldsymbol{x}) + \log p(\boldsymbol{x}) & \text{maximum-a-posteriori} \end{cases}$
- By convention, we *minimize* $f(x) \equiv -\tilde{f}(\boldsymbol{x})$

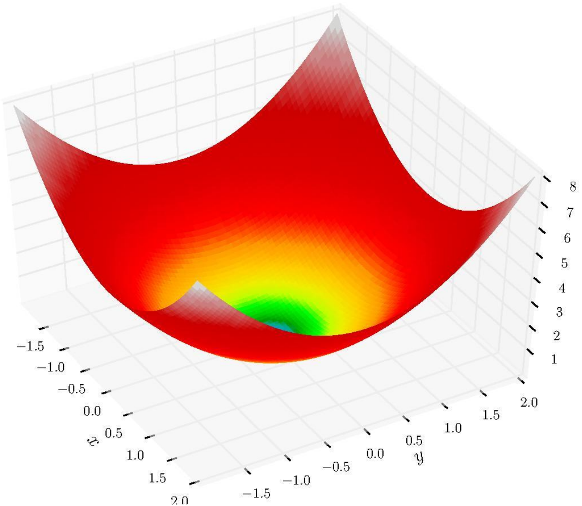
Model fitting via optimization

- Find single $\boldsymbol{\theta}$ that best describes the data
- (For this section we switch notation from $\boldsymbol{\theta}$ to $\boldsymbol{x}$)
- Given $\tilde{f}(\boldsymbol{x}) \equiv \begin{cases} \log p(\text{data}|\boldsymbol{x}) & \text{maximum likelihood} \\ \log p(\text{data}|\boldsymbol{x}) + \log p(\boldsymbol{x}) & \text{maximum-a-posteriori} \end{cases}$
- By convention, we *minimize* $f(x) \equiv -\tilde{f}(\boldsymbol{x})$
- $\Longrightarrow$ Find $\boldsymbol{x}_{opt} \approx \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$ as fast as possible

Model fitting via optimization

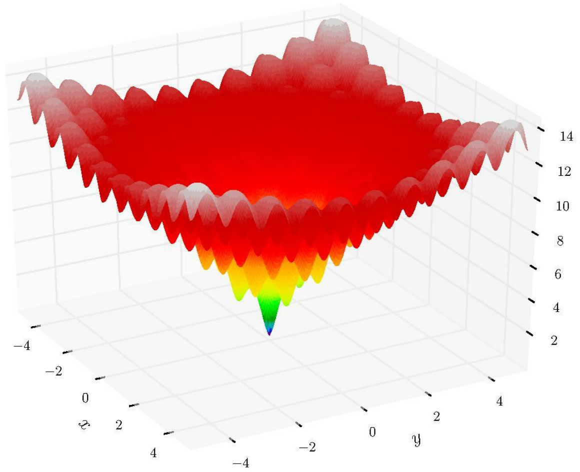
- Find single $\boldsymbol{\theta}$ that best describes the data
- (For this section we switch notation from $\boldsymbol{\theta}$ to $\boldsymbol{x}$)
- Given $\tilde{f}(\boldsymbol{x}) \equiv \begin{cases} \log p(\text{data}|\boldsymbol{x}) & \text{maximum likelihood} \\ \log p(\text{data}|\boldsymbol{x}) + \log p(\boldsymbol{x}) & \text{maximum-a-posteriori} \end{cases}$
- By convention, we *minimize* $f(x) \equiv -\tilde{f}(\boldsymbol{x})$
- $\Longrightarrow$ Find $\boldsymbol{x}_{opt} \approx \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$ as fast as possible
- General case: $f(\boldsymbol{x})$ is a *black box*
  - Sometimes we can compute the gradient

How hard can it be?



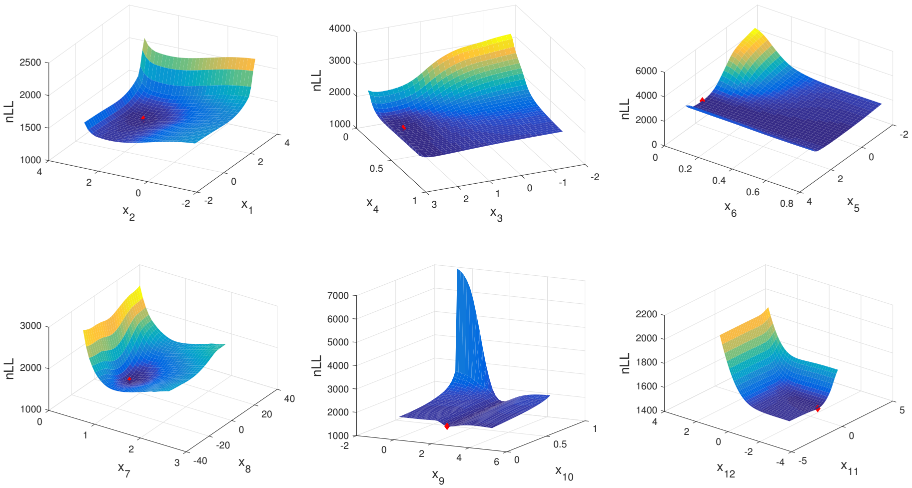
Source: Wikimedia Commons

How hard can it be?

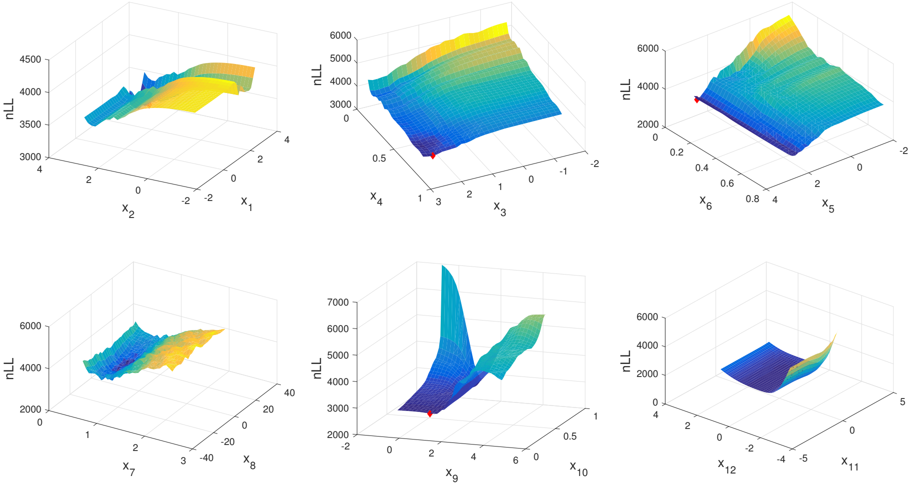


Source: Wikimedia Commons

How hard can it be?



How hard can it be?



How hard can it be?

neval	x_1	x_2	$f(\boldsymbol{x})$
1	-0.500	2.500	508.500
2	-0.525	2.500	497.110
3	-0.500	2.625	566.313
4	-0.525	2.375	443.063
5	-0.537	2.250	386.953
6	-0.563	2.250	376.320
7	-0.594	2.125	316.702
8	-0.606	1.875	229.824
9	-0.647	1.563	133.598
10	-0.703	1.438	91.847
11	-0.786	1.031	20.292
12	-0.839	0.469	8.918
13	-0.962	-0.359	168.785
14	-0.978	-0.063	107.796
15	-0.895	0.344	24.553
16	-0.730	1.156	41.905
17	-0.854	0.547	6.760
18	-0.907	-0.016	73.917
19	-0.816	0.770	4.366
20	-0.831	0.848	5.818
21	-0.793	1.070	22.655
22	-0.839	0.678	3.448
23	-0.824	0.600	3.955
24	-0.846	0.508	7.766
25	-0.824	0.704	3.391
26	-0.839	0.782	4.004
27	-0.828	0.645	3.497
28	-0.835	0.737	3.523
29	?	?	?

Optimization can be hard

1. Optimizer does not see the landscape!

Optimization can be hard

1. Optimizer does not see the landscape!
2. Multiple local minima or saddle points ('non-convex')

Optimization can be hard

1. Optimizer does not see the landscape!
2. Multiple local minima or saddle points ('non-convex')
3. Expensive function evaluation

Optimization can be hard

1. Optimizer does not see the landscape!
2. Multiple local minima or saddle points ('non-convex')
3. Expensive function evaluation
4. Noisy function evaluation
5. Rough landscape (numerical approximations, etc.)

Optimization algorithms

Gradient-based methods

- Stochastic gradient descent (e.g., ADAM)
- Quasi-Newton methods (e.g., BFGS aka `fminunc`/`fmincon`)

Gradient-free methods

- Nelder-Mead (`fminsearch`)
- Pattern/direct search (`patternsearch`)
- Simulated annealing
- Genetic algorithms
- CMA-ES
- Bayesian optimization
- Bayesian Adaptive Direct Search (BADS; Acerbi & Ma, *NeurIPS* 2017)

Optimization algorithms

Gradient-based methods

- Stochastic gradient descent (e.g., ADAM)
- Quasi-Newton methods (e.g., BFGS aka `fminunc`/`fmincon`)

Gradient-free methods

- Nelder-Mead (`fminsearch`)
- Pattern/direct search (`patternsearch`)
- Simulated annealing
- Genetic algorithms
- CMA-ES
- Bayesian optimization
- Bayesian Adaptive Direct Search (BADS; Acerbi & Ma, *NeurIPS* 2017)

Demos: `https://github.com/lacerbi/optimviz`

Local vs. global optimization

Local $\longleftrightarrow$ Global

BFGS (`fmincon`)	Nelder-Mead (`fminsearch`)	BADS	Bayesian Optimization
	`patternsearch`	CMA-ES	MCS
	SGD	Genetic algorithms	
		Simulated annealing	

Bayesian Optimization

1. Start with a prior over functions (Gaussian process)

Bayesian Optimization

1. Start with a prior over functions (Gaussian process)
2. Find $\tilde{\boldsymbol{x}}$ that maximizes *acquisition function* (exploration/exploitation)

Bayesian Optimization

1. Start with a prior over functions (Gaussian process)
2. Find $\tilde{\boldsymbol{x}}$ that maximizes *acquisition function* (exploration/exploitation)
3. Evaluate $f(\tilde{\boldsymbol{x}})$

Bayesian Optimization

1. Start with a prior over functions (Gaussian process)
2. Find $\tilde{\boldsymbol{x}}$ that maximizes *acquisition function* (exploration/exploitation)
3. Evaluate $f(\tilde{\boldsymbol{x}})$
4. Compute posterior over functions (Gaussian process)

Bayesian Optimization

1. Start with a prior over functions (Gaussian process)
2. Find $\tilde{\boldsymbol{x}}$ that maximizes *acquisition function* (exploration/exploitation)
3. Evaluate $f(\tilde{\boldsymbol{x}})$
4. Compute posterior over functions (Gaussian process)
5. `goto 2`

Bayesian Optimization

1. Start with a prior over functions (Gaussian process)
2. Find $\tilde{\boldsymbol{x}}$ that maximizes *acquisition function* (exploration/exploitation)
3. Evaluate $f(\tilde{\boldsymbol{x}})$
4. Compute posterior over functions (Gaussian process)
5. `goto 2`

J. Mockus, *Journal of Global Optimization* (1994)

Bayesian Optimization

- Good for expensive ($\gtrsim 1$ min), noisy functions up to $D \approx 10$

Bayesian Optimization

- Good for expensive ($\gtrsim 1$ min), noisy functions up to $D \approx 10$
- Scales badly with n, computation time $\sim O(n^3)$

Bayesian Optimization

- Good for expensive ($\gtrsim 1$ min), noisy functions up to $D \approx 10$
- Scales badly with n, computation time $\sim O(n^3)$
- Performance depends on quality of global approximation

Bayesian Adaptive Direct Search (`bads`)

- Combines Mesh-Adaptive Direct Search (MADS) with Bayesian Optimization (BO)

Bayesian Adaptive Direct Search (`bads`)

- Combines Mesh-Adaptive Direct Search (MADS) with Bayesian Optimization (BO)

Algorithm

1. Take as input f, `x0`, `LB`, `UB`, `PLB`, `PUB`

Bayesian Adaptive Direct Search (`bads`)

- Combines Mesh-Adaptive Direct Search (MADS) with Bayesian Optimization (BO)

Algorithm

1. Take as input f, `x0`, `LB`, `UB`, `PLB`, `PUB`
2. Evaluate f on an initial design and $x \leftarrow \arg\min_i f(x_i)$

Bayesian Adaptive Direct Search (`bads`)

- Combines Mesh-Adaptive Direct Search (MADS) with Bayesian Optimization (BO)

Algorithm

1. Take as input f, `x0`, `LB`, `UB`, `PLB`, `PUB`
2. Evaluate f on an initial design and $x \leftarrow \arg\min_i f(x_i)$
3. Until convergence or `MaxFunEvals` do

Bayesian Adaptive Direct Search (`bads`)

- Combines Mesh-Adaptive Direct Search (MADS) with Bayesian Optimization (BO)

Algorithm

1. Take as input f, `x0`, `LB`, `UB`, `PLB`, `PUB`
2. Evaluate f on an initial design and $x \leftarrow \arg\min_i f(x_i)$
3. Until convergence or `MaxFunEvals` do
   - POLL STEP: Evaluate up to $2D$ points around x, update x

Bayesian Adaptive Direct Search (`bads`)

- Combines Mesh-Adaptive Direct Search (MADS) with Bayesian Optimization (BO)

Algorithm

1. Take as input f, `x0`, `LB`, `UB`, `PLB`, `PUB`
2. Evaluate f on an initial design and $x \leftarrow \arg\min_i f(x_i)$
3. Until convergence or `MaxFunEvals` do
   - POLL STEP: Evaluate up to $2D$ points around x, update x
   - (TRAIN STEP: Train GP on neighborhood of x)

Bayesian Adaptive Direct Search (`bads`)

- Combines Mesh-Adaptive Direct Search (MADS) with Bayesian Optimization (BO)

Algorithm

1. Take as input f, `x0`, `LB`, `UB`, `PLB`, `PUB`
2. Evaluate f on an initial design and $x \leftarrow \arg\min_i f(x_i)$
3. Until convergence or `MaxFunEvals` do
   - Poll Step: Evaluate up to $2D$ points around x, update x
   - (Train Step: Train GP on neighborhood of x)
   - Search Step: Perform multiple iterations of BO in neighborhood of x

Bayesian Adaptive Direct Search (`bads`)

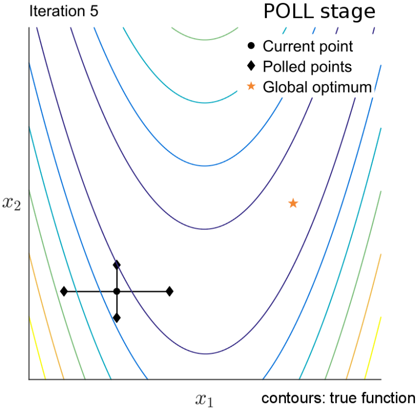
- Combines Mesh-Adaptive Direct Search (MADS) with Bayesian Optimization (BO)

Algorithm

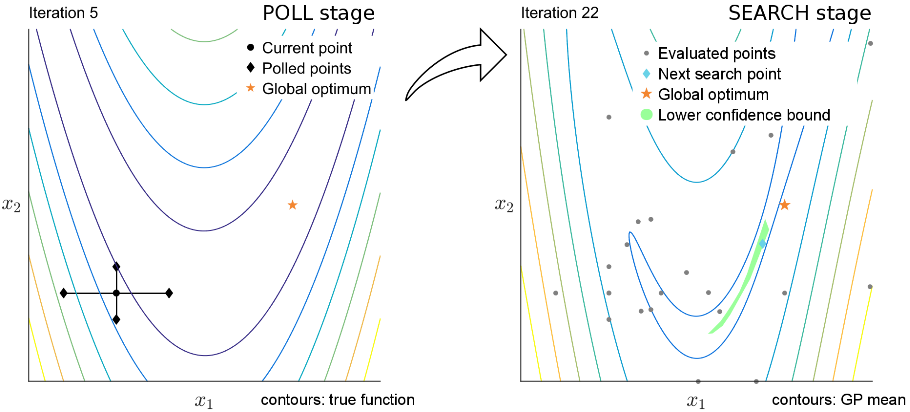
1. Take as input f, `x0`, `LB`, `UB`, `PLB`, `PUB`
2. Evaluate f on an initial design and $x \leftarrow \arg\min_i f(x_i)$
3. Until convergence or `MaxFunEvals` do
    - POLL STEP: Evaluate up to $2D$ points around x, update x
    - (TRAIN STEP: Train GP on neighborhood of x)
    - SEARCH STEP: Perform multiple iterations of BO in neighborhood of x

Acerbi & Ma, *NeurIPS* (2017)

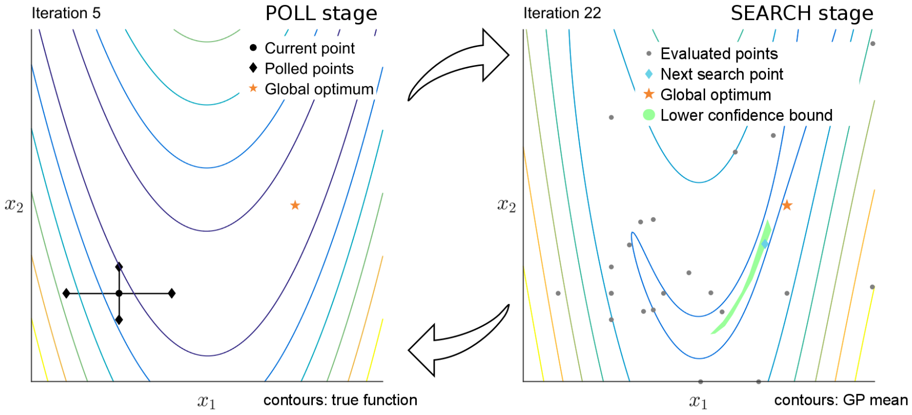
BADS algorithm



BADS algorithm



BADS algorithm



BADS algorithm

Algorithm 1 Bayesian Adaptive Direct Search

Input: objective function f, starting point $\boldsymbol{x}_0$, hard bounds LB, UB, (*optional*: plausible bounds PLB, PUB, barrier function c, additional `options`)

Initialization: $\Delta_0^{\text{mesh}} \leftarrow 2^{-10}$, $\Delta_0^{\text{poll}} \leftarrow 1$, $k \leftarrow 0$, evaluate f on initial design ▷ Section 3.1
repeat
    (update GP approximation at any step; refit hyperparameters if necessary) ▷ Section 3.2
    for $1 \dots n_{\text{search}}$ **do** ▷ SEARCH stage, Section 3.3
        $\boldsymbol{x}_{\text{search}} \leftarrow$ SEARCHORACLE ▷ local Bayesian optimization step
        Evaluate f on $\boldsymbol{x}_{\text{search}}$, **if** improvement is *sufficient* **then break**
    if SEARCH is NOT *successful* **then** ▷ optional POLL stage, Section 3.3
        compute poll set P_k
        evaluate opportunistically f on P_k sorted by acquisition function
    if iteration k is *successful* **then**
        update incumbent $\boldsymbol{x}_{k+1}$
        if POLL was *successful* **then** $\Delta_k^{\text{mesh}} \leftarrow 2\Delta_k^{\text{mesh}}$, $\Delta_k^{\text{poll}} \leftarrow 2\Delta_k^{\text{poll}}$
    else
        $\Delta_k^{\text{mesh}} \leftarrow \frac{1}{2}\Delta_k^{\text{mesh}}$, $\Delta_k^{\text{poll}} \leftarrow \frac{1}{2}\Delta_k^{\text{poll}}$
    $k \leftarrow k+1$
until `fevals` $>$ `MaxFunEvals` **or** $\Delta_k^{\text{poll}} < 10^{-6}$ **or** stalling ▷ stopping criteria
return $\boldsymbol{x}_{\text{end}} = \arg\min_k f(\boldsymbol{x}_k)$ (or $\boldsymbol{x}_{\text{end}} = \arg\min_k q_\beta(\boldsymbol{x}_k)$ for noisy objectives, Section 3.4)

BADS properties

- Good for moderately costly ($\gtrsim 0.1$ s) or noisy functions
- Scales okay with n (uses only local neighborhood)
- Local approximation deals with nonstationarity
- Explicit support for noise
- Outperforms other algorithms (Acerbi & Ma, 2017)

BADS summary

- POLL stage: Similar to `patternsearch`
- SEARCH stage: Local Bayesian optimization
- Initial POLL/SEARCH scale $\sim$ plausible box
- BADS supports:
  - Unbounded variables (deprecated)
  - Bounded variables
  - Non-bound constraints
  - Fixed variables
  - Periodic variables
- BADS treats *stochastic* target functions differently
  - Ensure that the noise SD is $\lesssim 1$

Optimization cheat sheet, page 1

Rule zero

Optimization cheat sheet, page 1

Rule zero

Understand your problem $\Longrightarrow$ often a *gray* box

Optimization cheat sheet, page 1

Rule zero

Understand your problem $\Longrightarrow$ often a *gray* box

Input variables:

- Dimensionality: low ($D \lesssim 10$) or high ($D \gg 20$)

Optimization cheat sheet, page 1

Rule zero

Understand your problem $\Longrightarrow$ often a *gray* box

Input variables:

- Dimensionality: low ($D \lesssim 10$) or high ($D \gg 20$)
- Bounds: Think of *hard* and *plausible* bounds

Optimization cheat sheet, page 1

Rule zero

Understand your problem $\Longrightarrow$ often a *gray* box

Input variables:

- Dimensionality: low ($D \lesssim 10$) or high ($D \gg 20$)
- Bounds: Think of *hard* and *plausible* bounds
- Parameterization: Not all parameterizations are created equal

Optimization cheat sheet, page 1

Rule zero

Understand your problem $\Longrightarrow$ often a *gray* box

Input variables:

- Dimensionality: low ($D \lesssim 10$) or high ($D \gg 20$)
- Bounds: Think of *hard* and *plausible* bounds
- Parameterization: Not all parameterizations are created equal

Target function:

- Convexity: convex or non-convex

Optimization cheat sheet, page 1

Rule zero

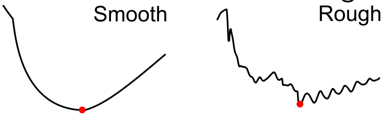
Understand your problem $\Longrightarrow$ often a *gray* box

Input variables:

- Dimensionality: low ($D \lesssim 10$) or high ($D \gg 20$)
- Bounds: Think of *hard* and *plausible* bounds
- Parameterization: Not all parameterizations are created equal

Target function:

- Convexity: convex or non-convex
- Smoothness: smooth or rough



Optimization cheat sheet, page 1

Rule zero

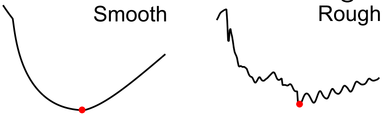
Understand your problem $\Longrightarrow$ often a *gray* box

Input variables:

- Dimensionality: low ($D \lesssim 10$) or high ($D \gg 20$)
- Bounds: Think of *hard* and *plausible* bounds
- Parameterization: Not all parameterizations are created equal

Target function:

- Convexity: convex or non-convex
- Smoothness: smooth or rough



- Deterministic or stochastic

Optimization cheat sheet, page 1

Rule zero

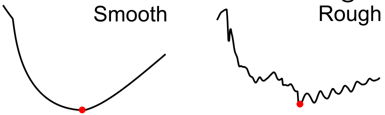
Understand your problem $\Longrightarrow$ often a *gray* box

Input variables:

- Dimensionality: low ($D \lesssim 10$) or high ($D \gg 20$)
- Bounds: Think of *hard* and *plausible* bounds
- Parameterization: Not all parameterizations are created equal

Target function:

- Convexity: convex or non-convex
- Smoothness: smooth or rough



- Deterministic or stochastic
  - If stochastic $\Longrightarrow$ minimize $\mathbb{E}\left[f(\boldsymbol{x})\right]$

Optimization cheat sheet, page 1

Rule zero

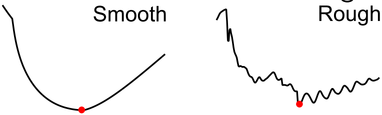
Understand your problem $\Longrightarrow$ often a *gray* box

Input variables:

- Dimensionality: low ($D \lesssim 10$) or high ($D \gg 20$)
- Bounds: Think of *hard* and *plausible* bounds
- Parameterization: Not all parameterizations are created equal

Target function:

- Convexity: convex or non-convex
- Smoothness: smooth or rough



- Deterministic or stochastic
  - If stochastic $\Longrightarrow$ minimize $\mathbb{E}[f(\boldsymbol{x})]$
- Computational cost:
  cheap ($\ll 0.01$ s), moderate (0.01-1 s), or expensive ($\gg 1$ s)

Optimization cheat sheet, page 2

Fundamental theorem

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences
  - If low-D and (moderately) costly $\Longrightarrow$ BADS

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences
  - If low-D and (moderately) costly $\Longrightarrow$ BADS

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences
  - If low-D and (moderately) costly $\Longrightarrow$ BADS
- Is your problem rough or noisy?

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences
  - If low-D and (moderately) costly $\Longrightarrow$ BADS
- Is your problem rough or noisy?
  - First, try and make it smooth and deterministic!

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences
  - If low-D and (moderately) costly $\Longrightarrow$ BADS
- Is your problem rough or noisy?
  - First, try and make it smooth and deterministic!
  - If gradient is available and high-D $\Longrightarrow$ SGD (e.g., ADAM)

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences
  - If low-D and (moderately) costly $\Longrightarrow$ BADS
- Is your problem rough or noisy?
  - First, try and make it smooth and deterministic!
  - If gradient is available and high-D $\Longrightarrow$ SGD (e.g., ADAM)
  - If high-D and cheap $\Longrightarrow$ CMA-ES

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences
  - If low-D and (moderately) costly $\Longrightarrow$ BADS
- Is your problem rough or noisy?
  - First, try and make it smooth and deterministic!
  - If gradient is available and high-D $\Longrightarrow$ SGD (e.g., ADAM)
  - If high-D and cheap $\Longrightarrow$ CMA-ES
  - If low-D and (moderately) costly $\Longrightarrow$ BADS

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences
  - If low-D and (moderately) costly $\Longrightarrow$ BADS
- Is your problem rough or noisy?
  - First, try and make it smooth and deterministic!
  - If gradient is available and high-D $\Longrightarrow$ SGD (e.g., ADAM)
  - If high-D and cheap $\Longrightarrow$ CMA-ES
  - If low-D and (moderately) costly $\Longrightarrow$ BADS
- Is your problem high-D, costly, and you do not have the gradient?

Optimization cheat sheet, page 2

Fundamental theorem

'No Free Lunch' theorem $\Longrightarrow$ no single best optimizer for all problems
(But not all methods are created equal!)

- Is your problem smooth?
  - If you have the gradient $\Longrightarrow$ BFGS
  - If low-D and cheap $\Longrightarrow$ BFGS with finite differences
  - If low-D and (moderately) costly $\Longrightarrow$ BADS
- Is your problem rough or noisy?
  - First, try and make it smooth and deterministic!
  - If gradient is available and high-D $\Longrightarrow$ SGD (e.g., ADAM)
  - If high-D and cheap $\Longrightarrow$ CMA-ES
  - If low-D and (moderately) costly $\Longrightarrow$ BADS
- Is your problem high-D, costly, and you do not have the gradient?
  - Give up and pray

Optimization cheat sheet, page 3

The golden rule

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

- How to choose starting points?

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

- How to choose starting points?
  - Draw from prior distribution

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

- How to choose starting points?
  - Draw from prior distribution
  - Draw from a 'plausible' box

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

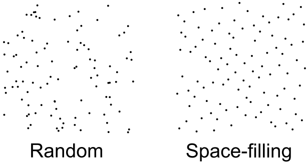
- How to choose starting points?
  - Draw from prior distribution
  - Draw from a 'plausible' box
  - Sieve method

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

- How to choose starting points?
  - Draw from prior distribution
  - Draw from a 'plausible' box
  - Sieve method
  - Use space-filling designs (quasi-random sequences)

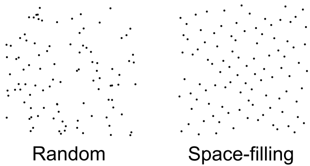


Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

- How to choose starting points?
  - Draw from prior distribution
  - Draw from a 'plausible' box
  - Sieve method
  - Use space-filling designs (quasi-random sequences)



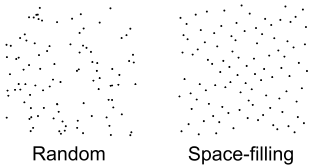
- How many restarts?

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

- How to choose starting points?
  - Draw from prior distribution
  - Draw from a 'plausible' box
  - Sieve method
  - Use space-filling designs (quasi-random sequences)



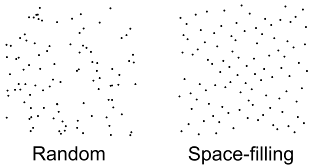
- How many restarts?
  - As many as you need

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

- How to choose starting points?
  - Draw from prior distribution
  - Draw from a 'plausible' box
  - Sieve method
  - Use space-filling designs (quasi-random sequences)



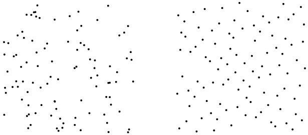
- How many restarts?
  - As many as you need
  - Informally, check that 'most' points converge to the same solution

Optimization cheat sheet, page 3

The golden rule

No optimizer can guarantee to find the global optimum
$\Longrightarrow$ Always perform multiple distinct optimization runs ('restarts')

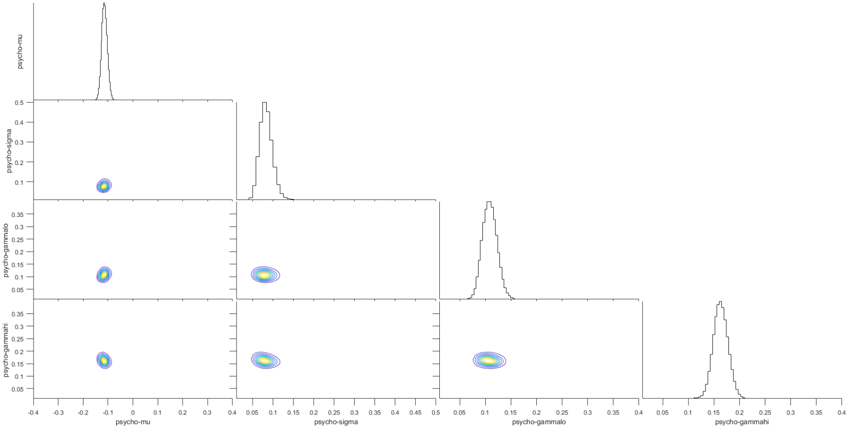
- How to choose starting points?
  - Draw from prior distribution
  - Draw from a 'plausible' box
  - Sieve method
  - Use space-filling designs (quasi-random sequences)



- How many restarts?
  - As many as you need
  - Informally, check that 'most' points converge to the same solution
  - *Bootstrap* approach (Acerbi, Dokka et al., *PLoS Comp Biol* 2018)

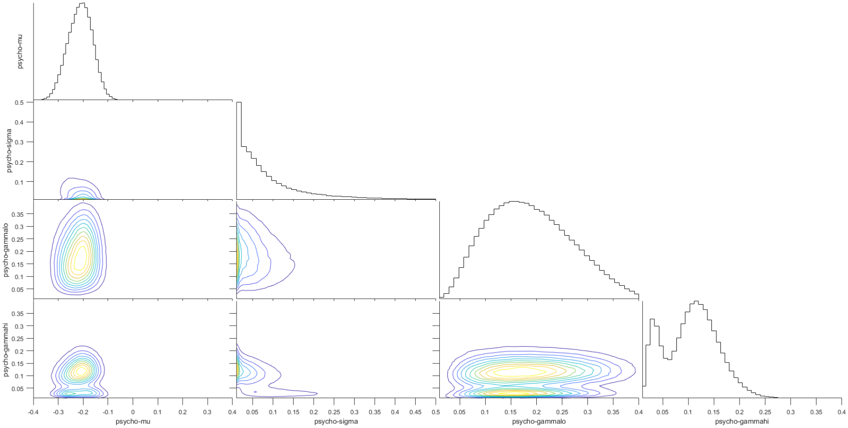
Bayesian posteriors

Bayesian posteriors



$n = 1353$ trials

Bayesian posteriors



$n = 90$ trials

Benefits of Bayesian posteriors

- Check for parameter uncertainty, trade-offs, identifiability
  - Deeper understanding of your model
  - Robustness of claims (Acerbi, Ma, Vijayakumar, *NeurIPS* 2014)

Benefits of Bayesian posteriors

- Check for parameter uncertainty, trade-offs, identifiability
  - Deeper understanding of your model
  - Robustness of claims (Acerbi, Ma, Vijayakumar, *NeurIPS* 2014)
- Less overfitting

Benefits of Bayesian posteriors

- Check for parameter uncertainty, trade-offs, identifiability
  - Deeper understanding of your model
  - Robustness of claims (Acerbi, Ma, Vijayakumar, *NeurIPS* 2014)
- Less overfitting
- Use posterior samples to compute model comparison metrics
  - DIC, WAIC, LOO-CV

Benefits of Bayesian posteriors

- Check for parameter uncertainty, trade-offs, identifiability
  - Deeper understanding of your model
  - Robustness of claims (Acerbi, Ma, Vijayakumar, *NeurIPS* 2014)
- Less overfitting
- Use posterior samples to compute model comparison metrics
  - DIC, WAIC, LOO-CV
- Fully taking into account uncertainty is just *better*

Benefits of Bayesian posteriors

- Check for parameter uncertainty, trade-offs, identifiability
  - Deeper understanding of your model
  - Robustness of claims (Acerbi, Ma, Vijayakumar, *NeurIPS* 2014)
- Less overfitting
- Use posterior samples to compute model comparison metrics
  - DIC, WAIC, LOO-CV
- Fully taking into account uncertainty is just *better*

How do I get Bayesian posteriors?

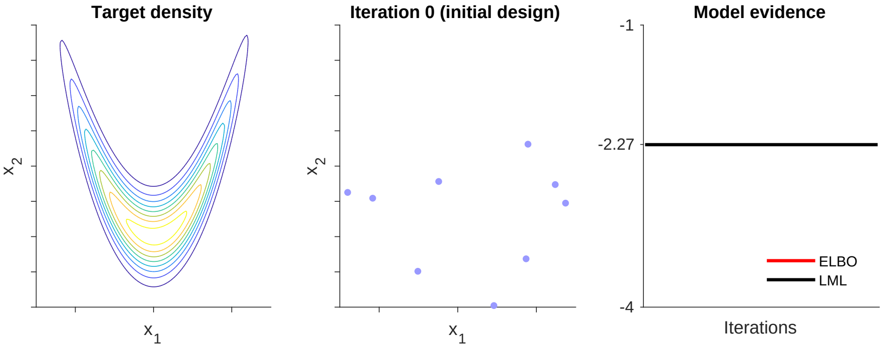
- MCMC (slice sampling, NUTS)
- Variational inference

Variational Bayesian Monte Carlo

Alternative to MCMC (for low-D, moderately costly problems)

Variational Bayesian Monte Carlo

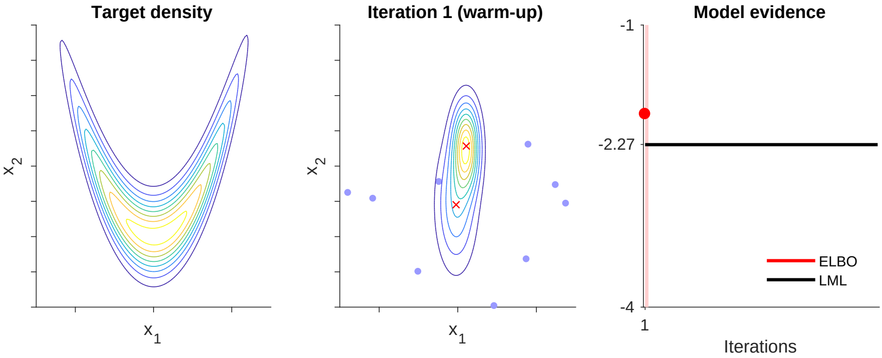
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

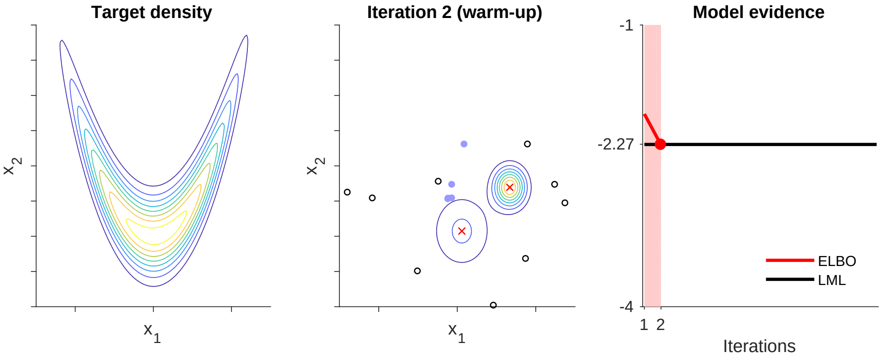
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

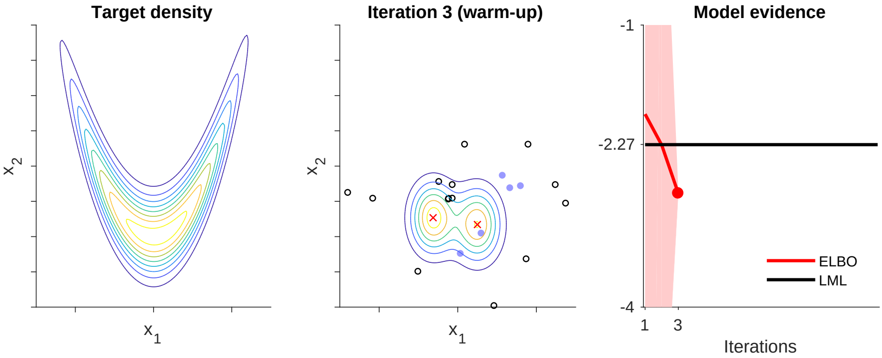
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

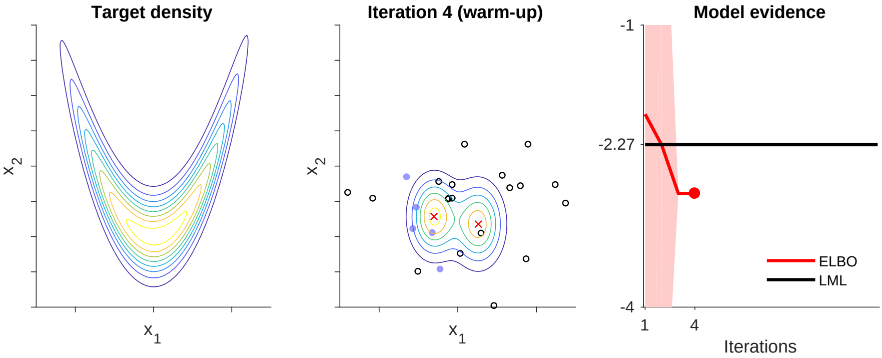
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

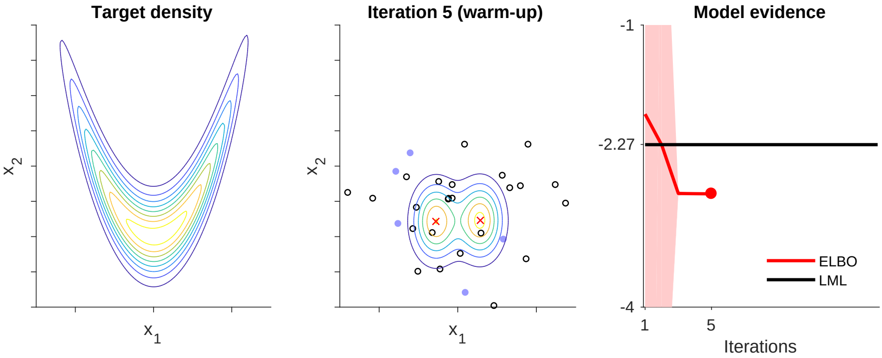
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

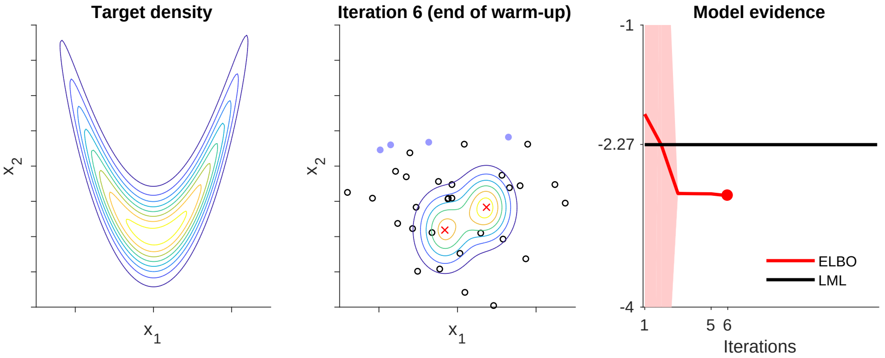
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

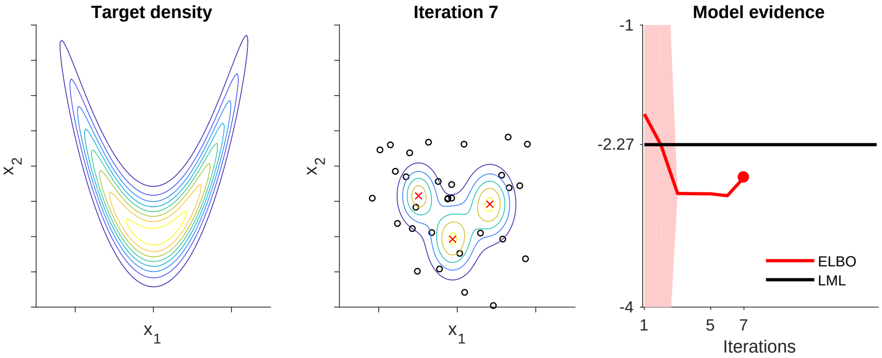
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

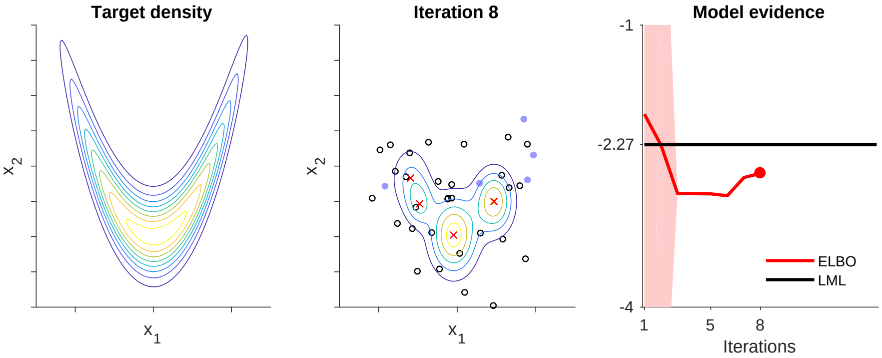
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

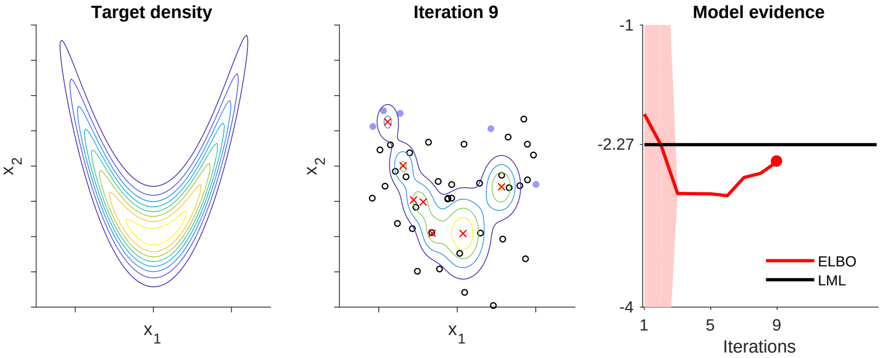
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

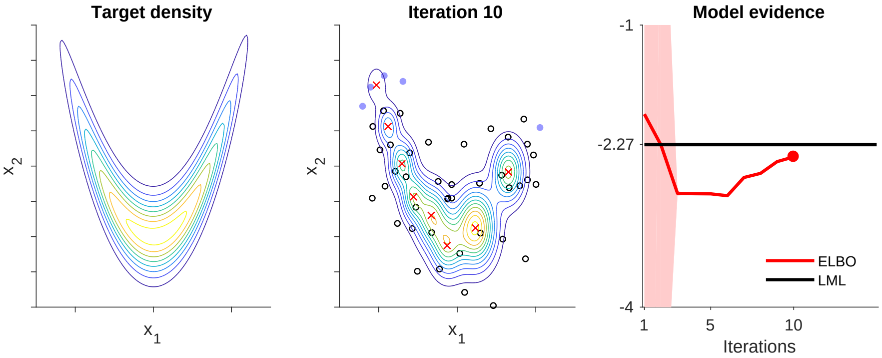
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

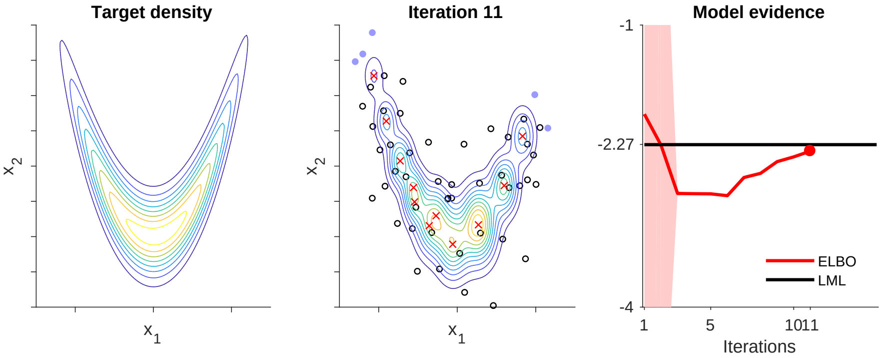
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

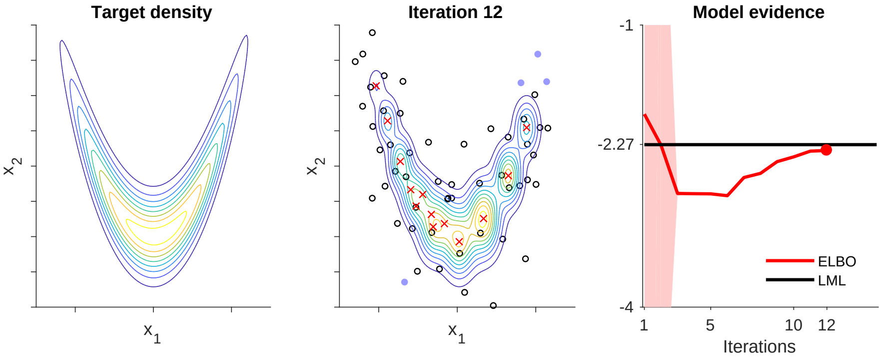
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

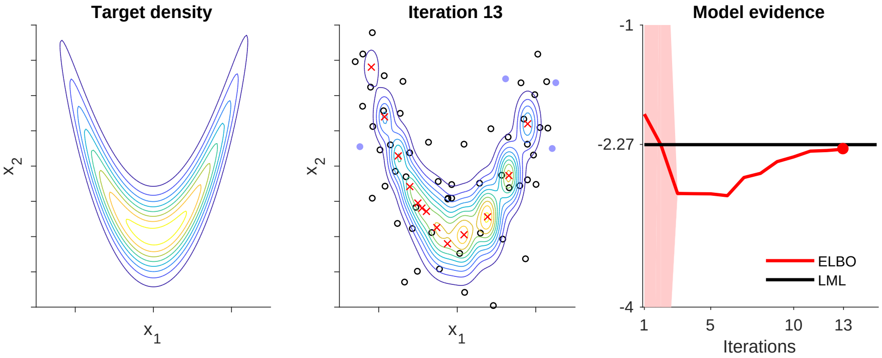
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

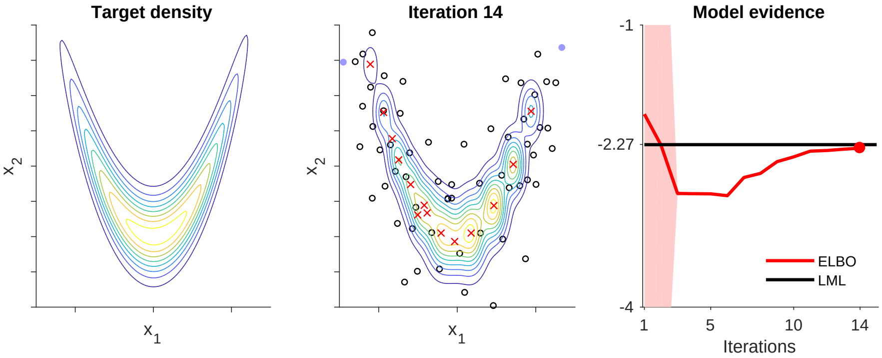
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

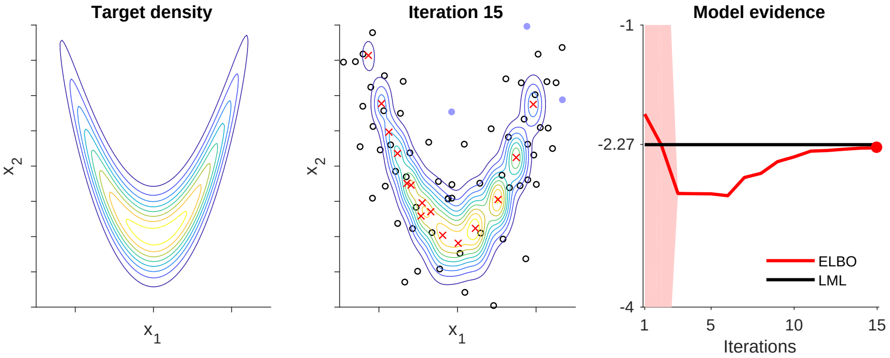
Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Variational Bayesian Monte Carlo

Alternative to MCMC (for low-D, moderately costly problems)



Acerbi, *NeurIPS* 2018

Applied example

OPEN ACCESS PEER-REVIEWED

RESEARCH ARTICLE

Bayesian comparison of explicit and implicit causal inference strategies in multisensory heading perception

Luigi Acerbi, Kalpana Dokka, Dora E. Angelaki, Wei Ji Ma

Published: July 27, 2018 • https://doi.org/10.1371/journal.pcbi.1006110

Final slide

- Contact me at `luigi.acerbi@gmail.com`
- Optimization demos: `github.com/lacerbi/optimviz`
- BADS available at `github.com/lacerbi/bads`
- VBMC available at `github.com/lacerbi/vbmc`
- Tutorial: `github.com/lacerbi/workshop-nyu-2019`

Final slide

- Contact me at `luigi.acerbi@gmail.com`
- Optimization demos: `github.com/lacerbi/optimviz`
- BADS available at `github.com/lacerbi/bads`
- VBMC available at `github.com/lacerbi/vbmc`
- Tutorial: `github.com/lacerbi/workshop-nyu-2019`

Thanks!

(Time for questions?)