Introduction to Optimization for Statistical Model Fitting

Luigi Acerbi

Department of Computer Science
University of Helsinki
Finnish Center for Artificial Intelligence FCAI

BAMB! Summer School – Day 2
September 2022

1. Introduction and recap
   - Of models and likelihoods
   - The psychometric function
2. Model fitting
   - A statistical estimation problem
   - Model fitting via point estimation
3. Intro to optimization
   - The problem
   - Optimization algorithms
   - Optimization cheat sheets

Learning objectives

By the end of this lecture/tutorial, we will:

- Explain how model fitting is an optimization problem
- Describe several major optimization algorithms
- Set up and run optimization on real dataset and models
- Summarize practical tips & tricks for optimization

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:

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

  - 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:

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

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

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

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

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!

What is a mathematical model?

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

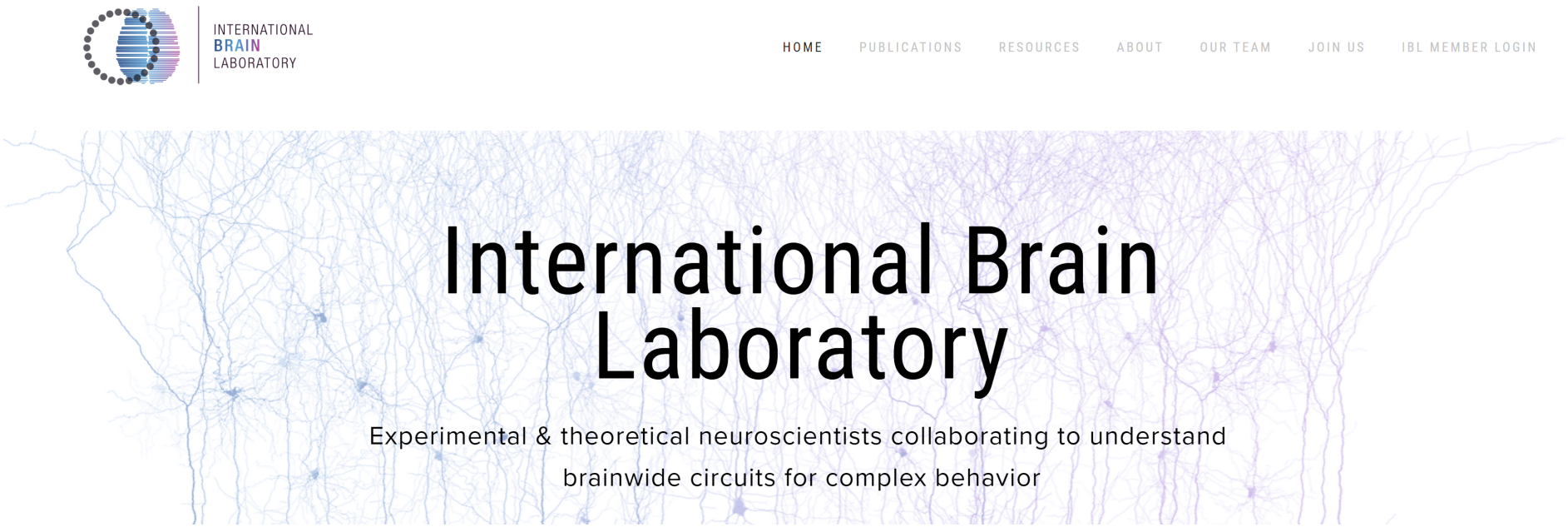
$$p\left(\text{data}|\boldsymbol{\theta}\right)$$

  - 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!

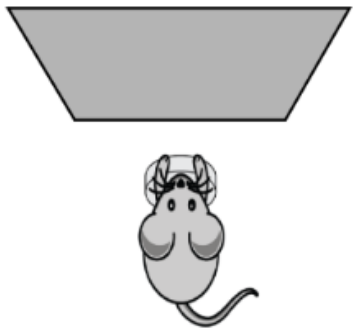
We need some data

Data from International Brain Laboratory (IBL)



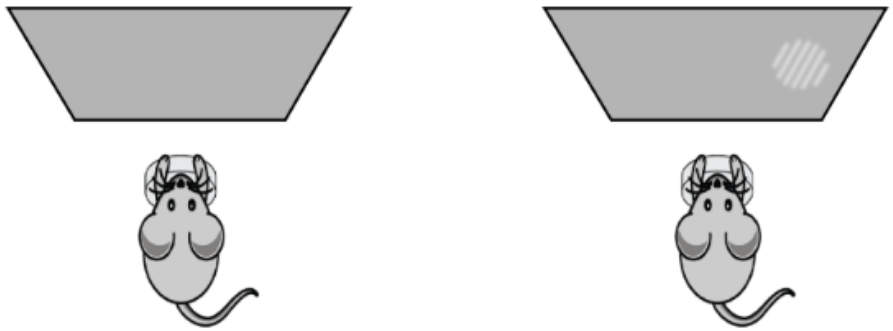
`https://www.internationalbrainlab.com`

IBL Task



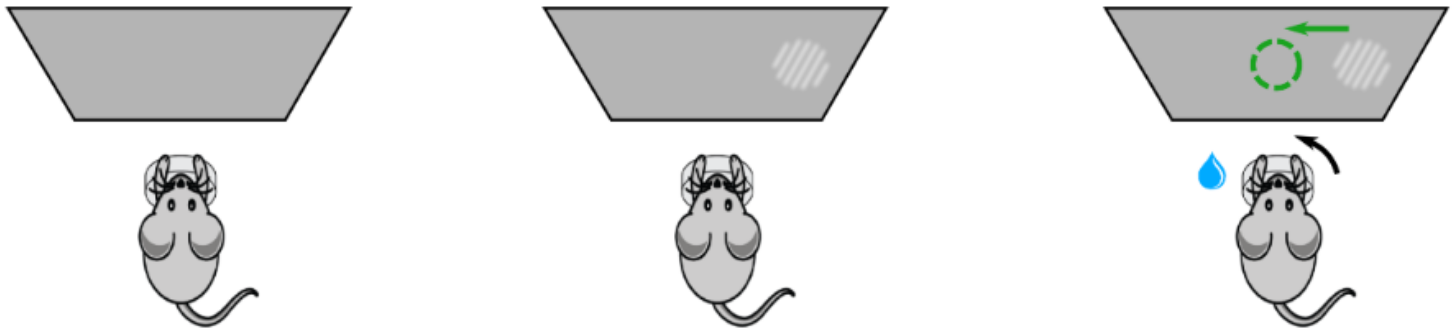
(IBL et al., *eLife*, 2021)

IBL Task



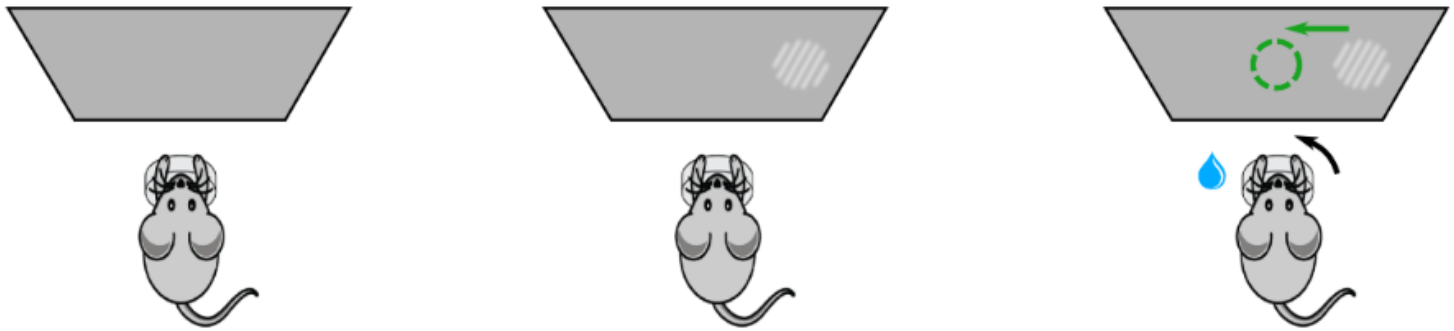
(IBL et al., *eLife*, 2021)

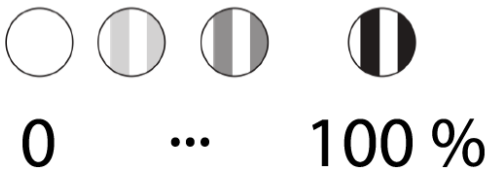
IBL Task



(IBL et al., *eLife*, 2021)

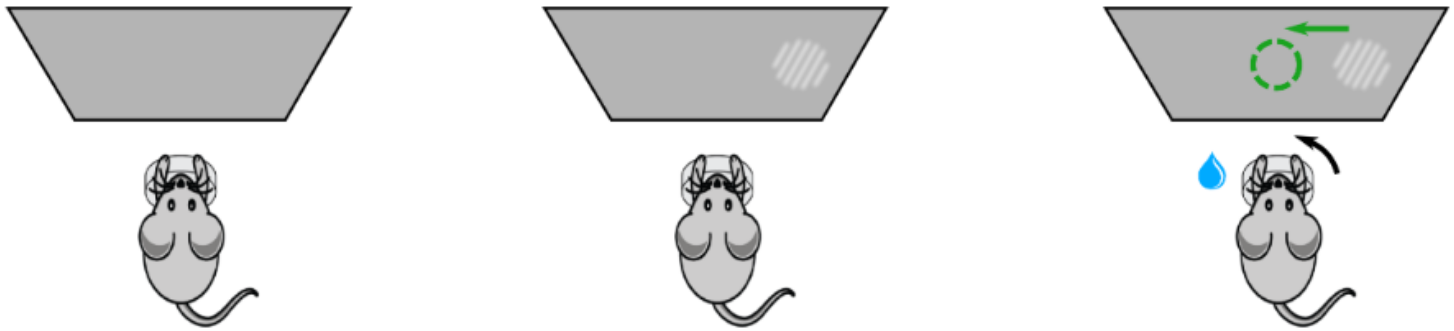
IBL Task

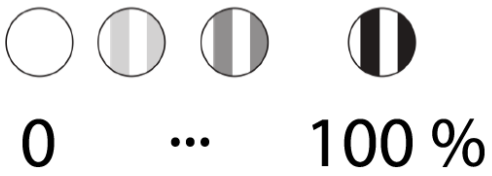


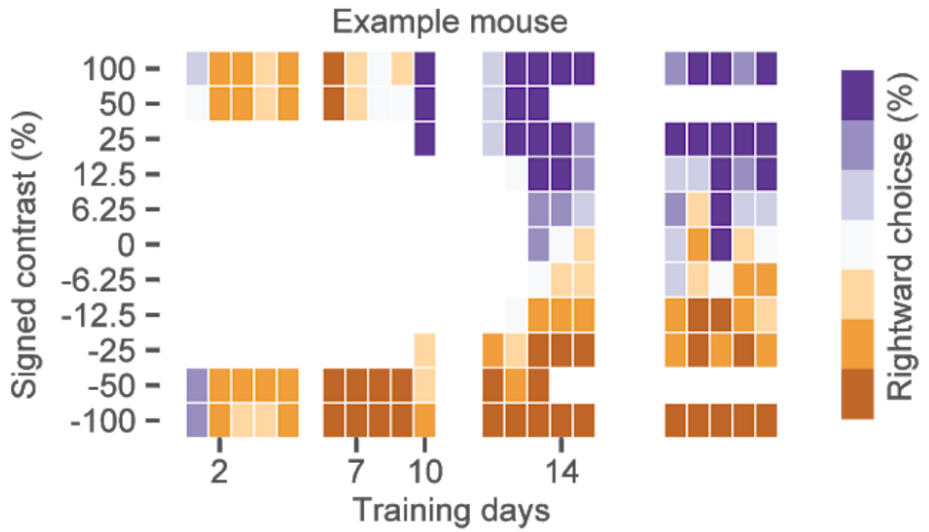


(IBL et al., *eLife*, 2021)

IBL Task





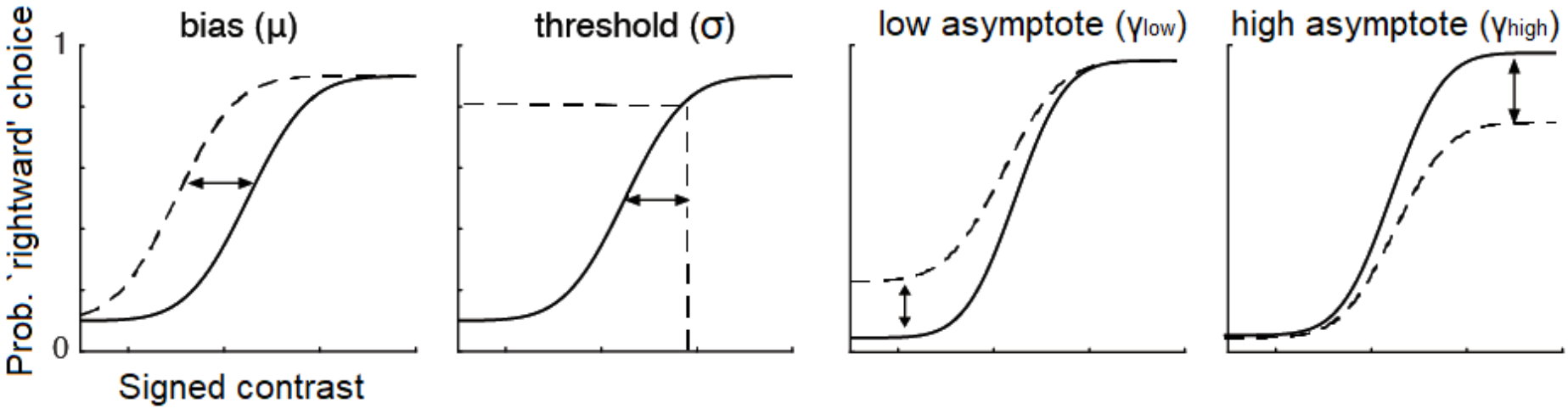


(IBL et al., *eLife*, 2021)

Hacking time I

Let's have a look at the data

The psychometric function



- Data: (signed contrast, choice) for each trial
- Parameters $\boldsymbol{\theta}$: $(\mu, \sigma, \gamma_{\text{low}}, \gamma_{\text{high}})$

$$p(\text{rightward choice}|s, \boldsymbol{\theta}) = \gamma_{\text{low}} + (1 - \gamma_{\text{low}} - \gamma_{\text{high}}) \cdot F(s; \mu, \sigma)$$

The psychometric function (alt version)

- Default decision process $F(s; \mu, \sigma)$
- Lapses with probability $\lambda \in [0, 1]$ (*lapse rate*)
- If lapse, respond 'rightward' with probability $\gamma \in [0, 1]$ (*lapse bias*)
- Parameters $\boldsymbol{\theta}$: $(\mu,\ \sigma,\ \lambda,\ \gamma)$

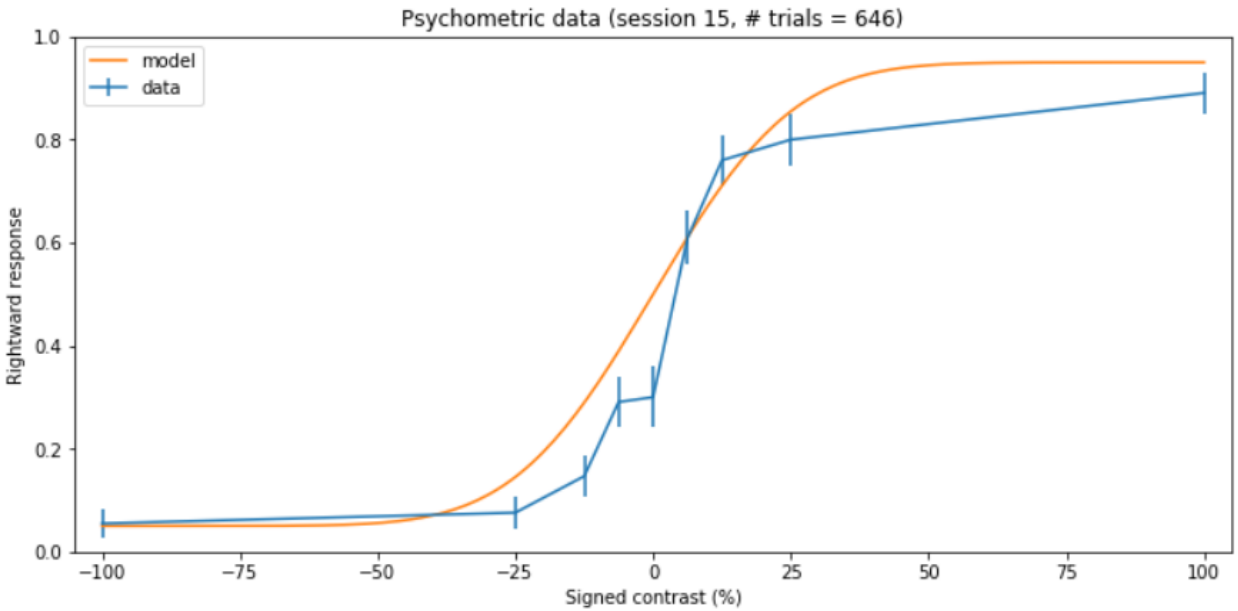
$$p(\text{rightward choice}|s, \boldsymbol{\theta}) = \lambda\gamma + (1 - \lambda) \cdot F(s; \mu, \sigma)$$

Hacking time II

Let's have a look at the psychometric function

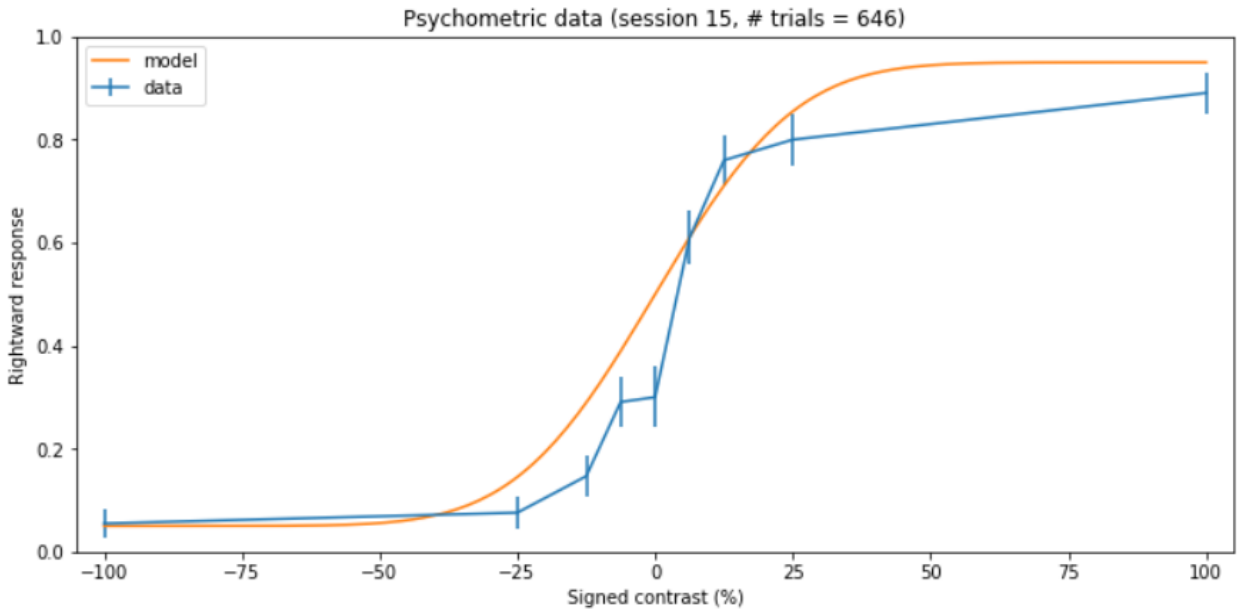
Metric for model fitting

We need a quantity to measure *goodness of fit*



Metric for model fitting

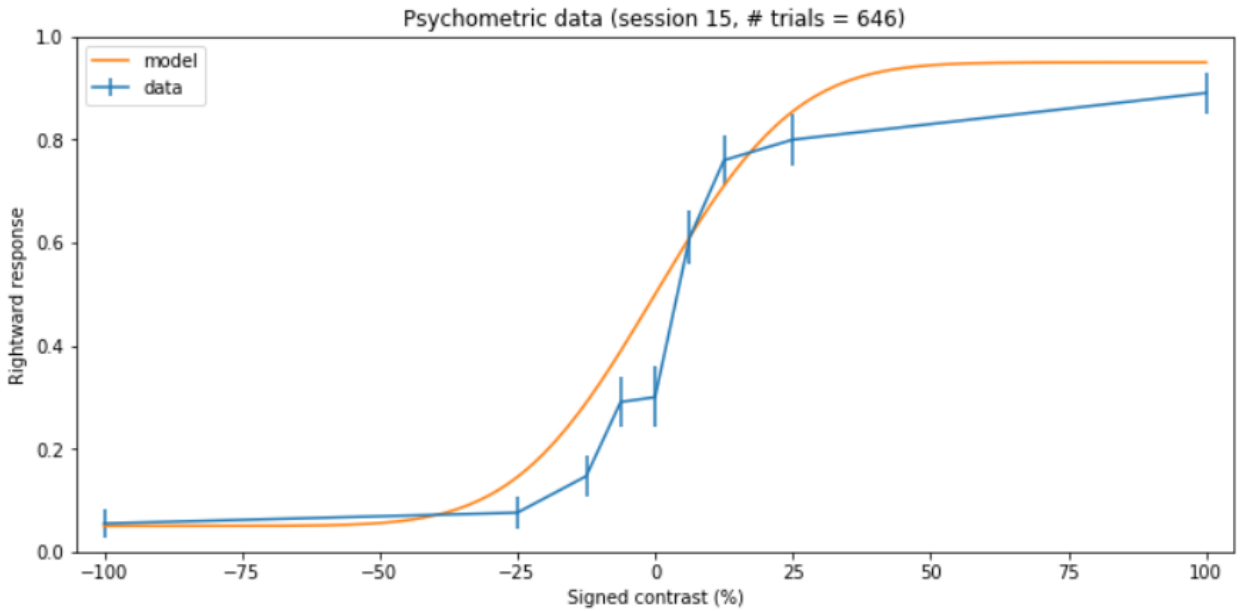
We need a quantity to measure *goodness of fit*



- Mean squared error?

Metric for model fitting

We need a quantity to measure *goodness of fit*



- ~~Mean squared error?~~
- The likelihood $p(\text{data}|\boldsymbol{\theta})$

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 $\log p(\text{data}|\theta)$

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

Hacking time III

Let's play with a log-likelihood function

1. Introduction and recap
   - Of models and likelihoods
   - The psychometric function

2. Model fitting
   - A statistical estimation problem
   - Model fitting via point estimation

3. Intro to optimization
   - The problem
   - Optimization algorithms
   - Optimization cheat sheets

Model fitting $\sim$ statistical estimation problem

Model fitting $\sim$ statistical estimation problem

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 $\sim$ statistical estimation problem

Maximum likelihood estimation (MLE)

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

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

Bayesian posterior estimation

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

Model fitting $\sim$ statistical estimation problem

Maximum likelihood estimation (MLE)

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

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

Bayesian posterior estimation

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

Maximum-a-posteriori (MAP) estimation

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\text{data}) = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\text{data}) \\ &= \arg\max_{\boldsymbol{\theta}} \{\log p(\text{data}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\}\end{aligned}$$

Two ways of model fitting

Two ways of model fitting

"Point estimation"

- Maximum likelihood estimation (MLE), Maximum-a-posteriori (MAP)
- Model fitting $\sim$ *optimization problem*
- Returns a single 'best' parameter vector $\hat{\boldsymbol{\theta}}$ ($\hat{\boldsymbol{\theta}}_{ML}$ or $\hat{\boldsymbol{\theta}}_{MAP}$)

Two ways of model fitting

"Point estimation"

- Maximum likelihood estimation (MLE), Maximum-a-posteriori (MAP)
- Model fitting $\sim$ *optimization problem*
- Returns a single 'best' parameter vector $\hat{\boldsymbol{\theta}}$ ($\hat{\boldsymbol{\theta}}_{ML}$ or $\hat{\boldsymbol{\theta}}_{MAP}$)

Bayesian posterior estimation

- True Bayesian way
- Returns a *distribution* (the posterior distribution $p(\boldsymbol{\theta}|\text{data})$)
- For $n \to \infty$ converges to MLE (if $p(\hat{\boldsymbol{\theta}}_{\text{ML}}) \neq 0$)
- In practice, it returns an *approximation* of the posterior

Two ways of model fitting

"Point estimation"

- Maximum likelihood estimation (MLE), Maximum-a-posteriori (MAP)
- Model fitting $\sim$ *optimization problem*
- Returns a single 'best' parameter vector $\hat{\boldsymbol{\theta}}$ ($\hat{\boldsymbol{\theta}}_{ML}$ or $\hat{\boldsymbol{\theta}}_{MAP}$)

Bayesian posterior estimation

- True Bayesian way
- Returns a *distribution* (the posterior distribution $p(\boldsymbol{\theta}|\text{data})$)
- For $n \to \infty$ converges to MLE (if $p(\hat{\boldsymbol{\theta}}_{\text{ML}}) \neq 0$)
- In practice, it returns an *approximation* of the posterior
  1. In parametric form $q(\boldsymbol{\theta})$ (e.g., *variational inference*)
  2. As a bunch of discrete samples (e.g., *Markov-Chain Monte Carlo*)

Two ways of model fitting

"Point estimation"

- Maximum likelihood estimation (MLE), Maximum-a-posteriori (MAP)
- Model fitting $\sim$ *optimization problem*
- Returns a single 'best' parameter vector $\hat{\boldsymbol{\theta}}$ ($\hat{\boldsymbol{\theta}}_{ML}$ or $\hat{\boldsymbol{\theta}}_{MAP}$)

Bayesian posterior estimation

- True Bayesian way
- Returns a *distribution* (the posterior distribution $p(\boldsymbol{\theta}|\text{data})$)
- For $n \to \infty$ converges to MLE (if $p(\hat{\boldsymbol{\theta}}_{\text{ML}}) \neq 0$)
- In practice, it returns an *approximation* of the posterior
  1. In parametric form $q(\boldsymbol{\theta})$ (e.g., *variational inference*)
  2. As a bunch of discrete samples (e.g., *Markov-Chain Monte Carlo*)
- Next lecture!

Model fitting via point estimation

- Find single θ that best describes the data

Model fitting via point estimation

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

Model fitting via point estimation

- 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 point estimation

- 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 point estimation

- 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 point estimation

- 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

Model fitting via point estimation

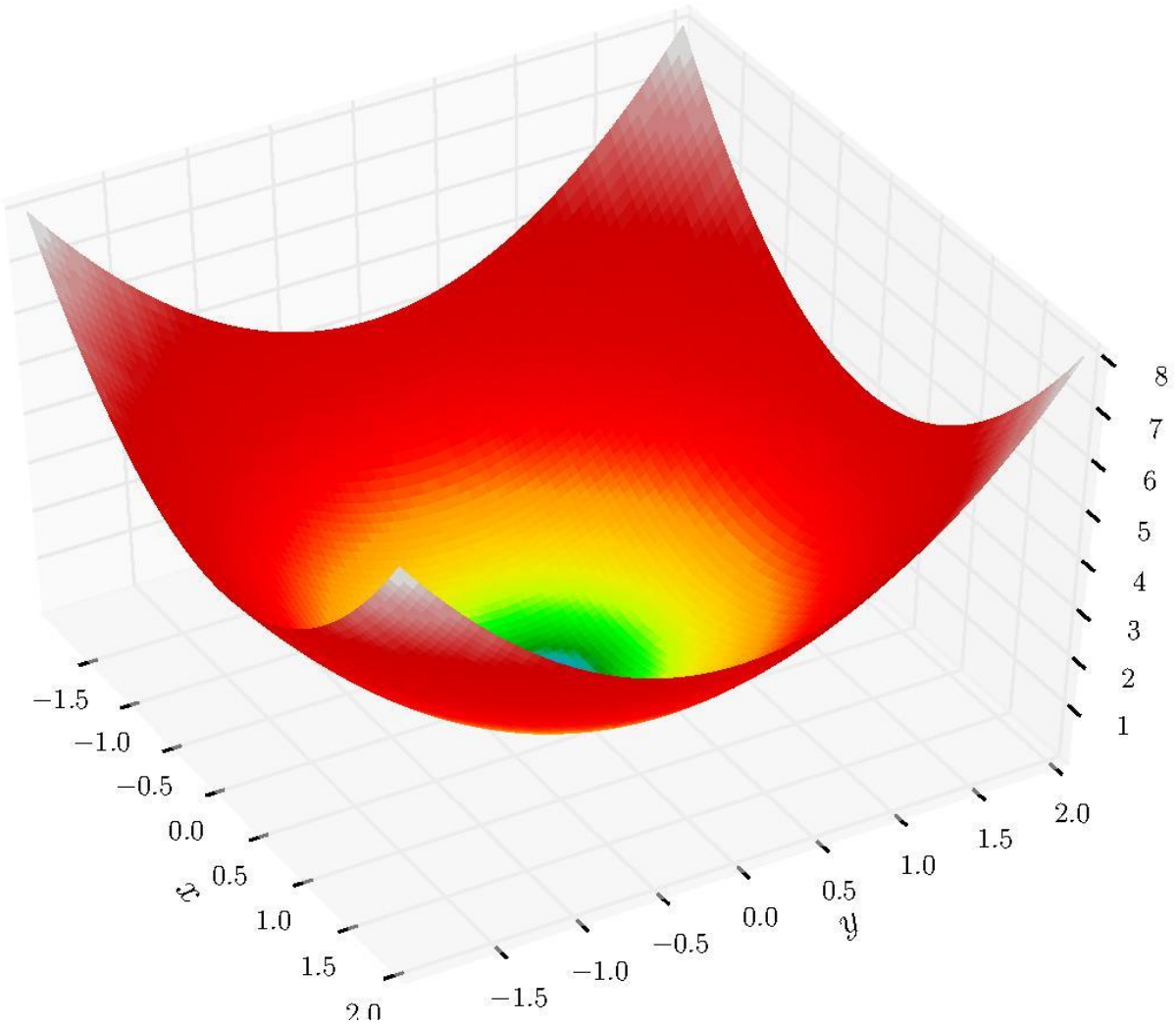
- Find single $\boldsymbol{\theta}$ that best describes the data
- (For this section we switch notation from $\boldsymbol{\theta}$ to $\boldsymbol{x}$)
- Given $\tilde{f}(\boldsymbol{x}) \equiv \begin{cases} \log p(\text{data}|\boldsymbol{x}) & \text{maximum likelihood} \\ \log p(\text{data}|\boldsymbol{x})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

General solution

Feed $f(\boldsymbol{x})$ to an optimization algorithm

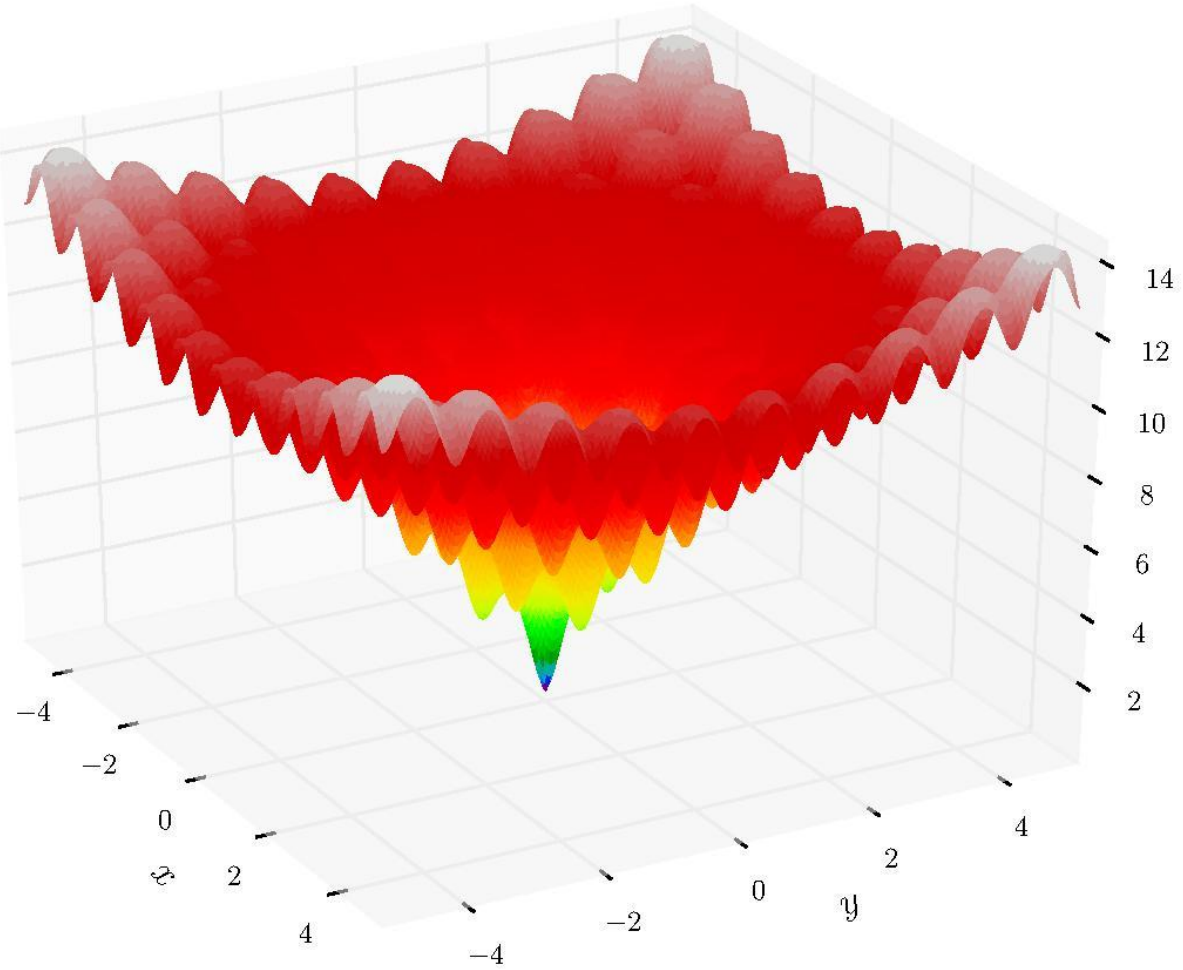
1. Introduction and recap
   - Of models and likelihoods
   - The psychometric function
2. Model fitting
   - A statistical estimation problem
   - Model fitting via point estimation
3. Intro to optimization
   - The problem
   - Optimization algorithms
   - Optimization cheat sheets

How hard can it be?



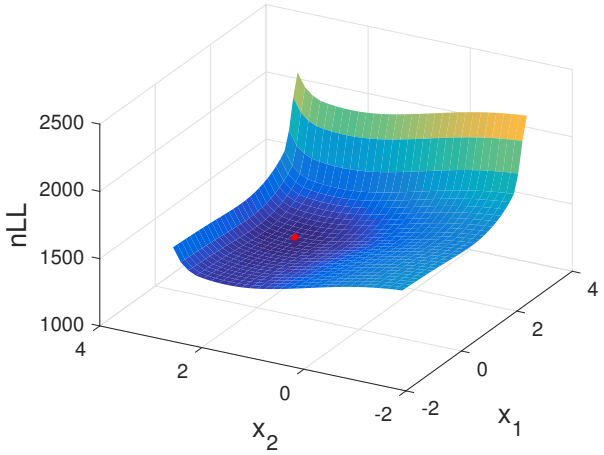
Source: Wikimedia Commons

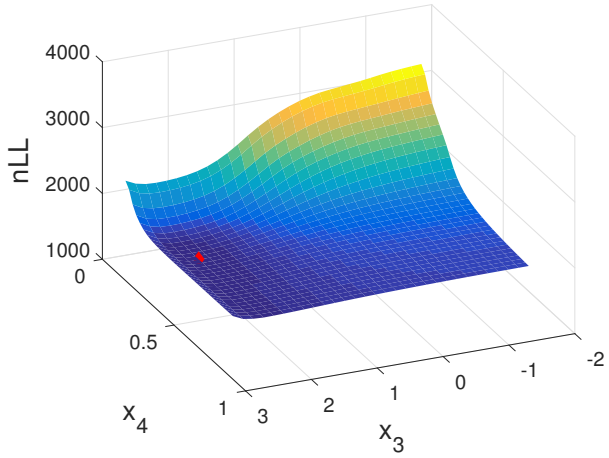
How hard can it be?

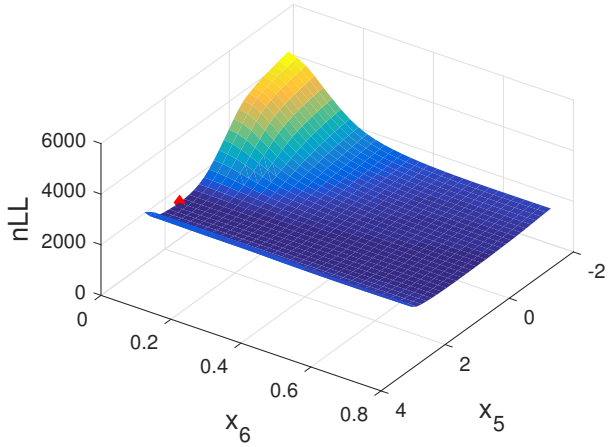


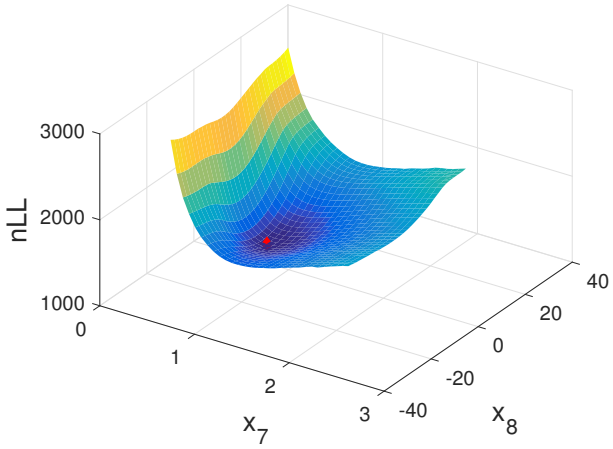
Source: Wikimedia Commons

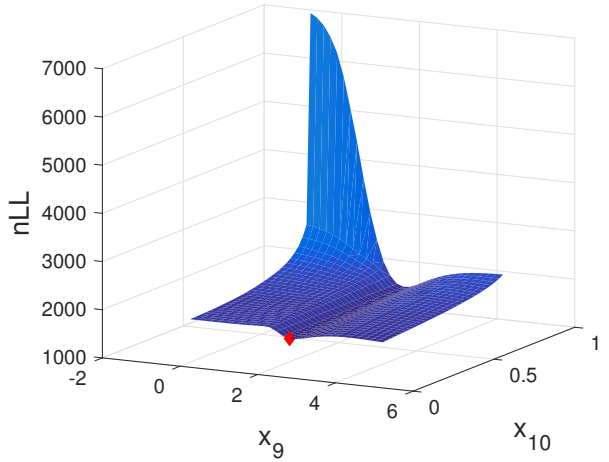
How hard can it be?

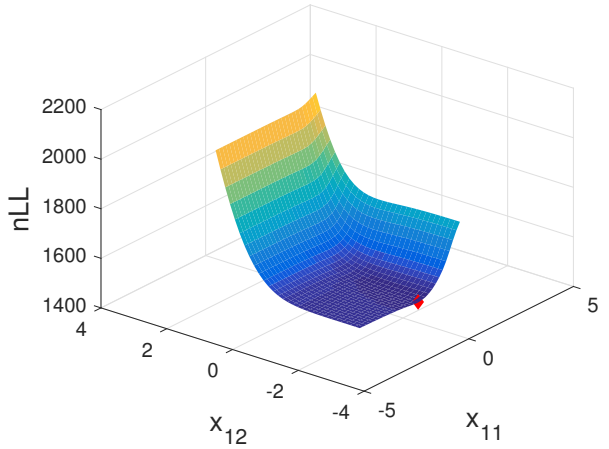




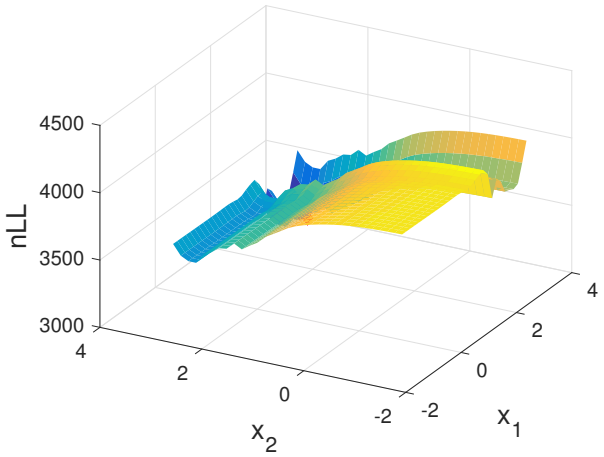


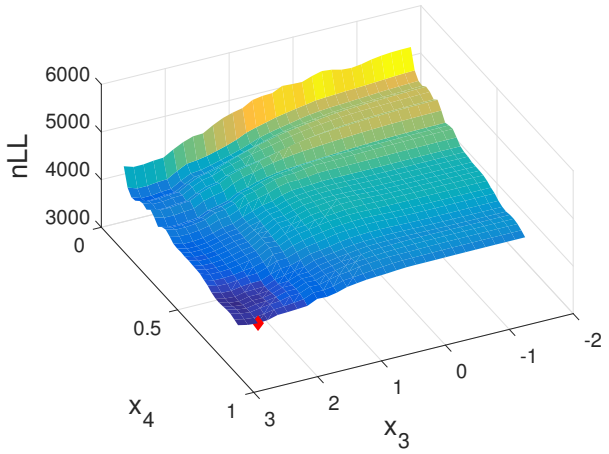


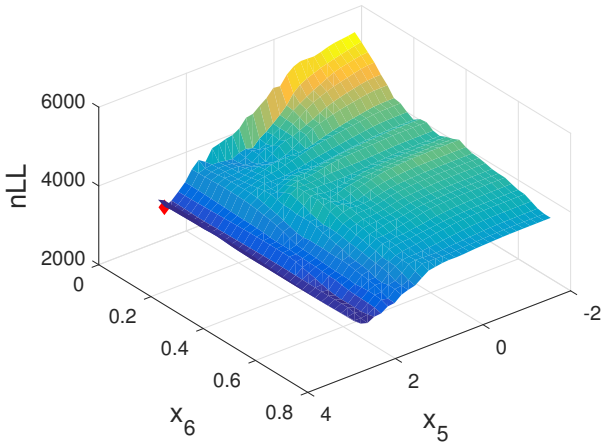


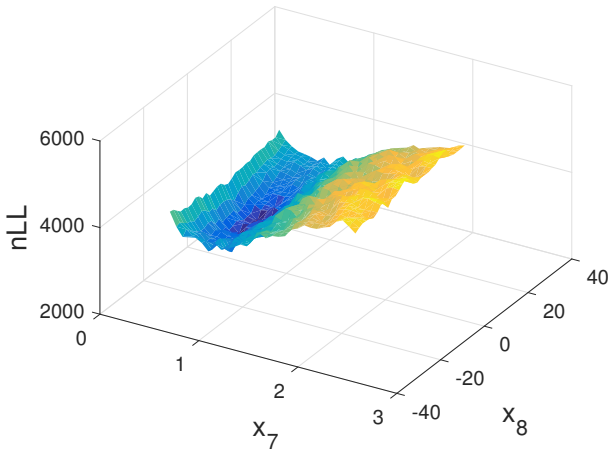


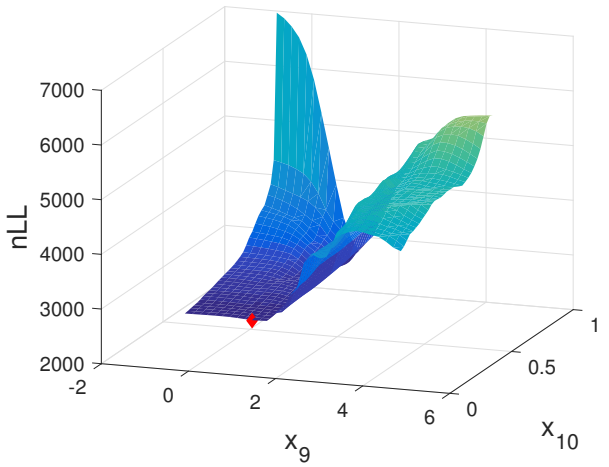
How hard can it be?

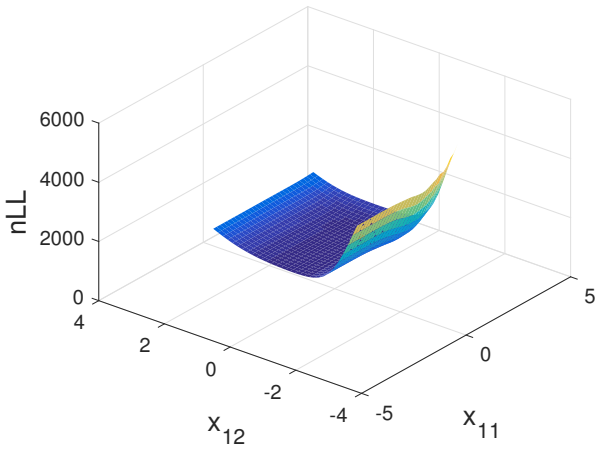












How hard can it be?

neval	x_1	x_2	$f(x)$
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

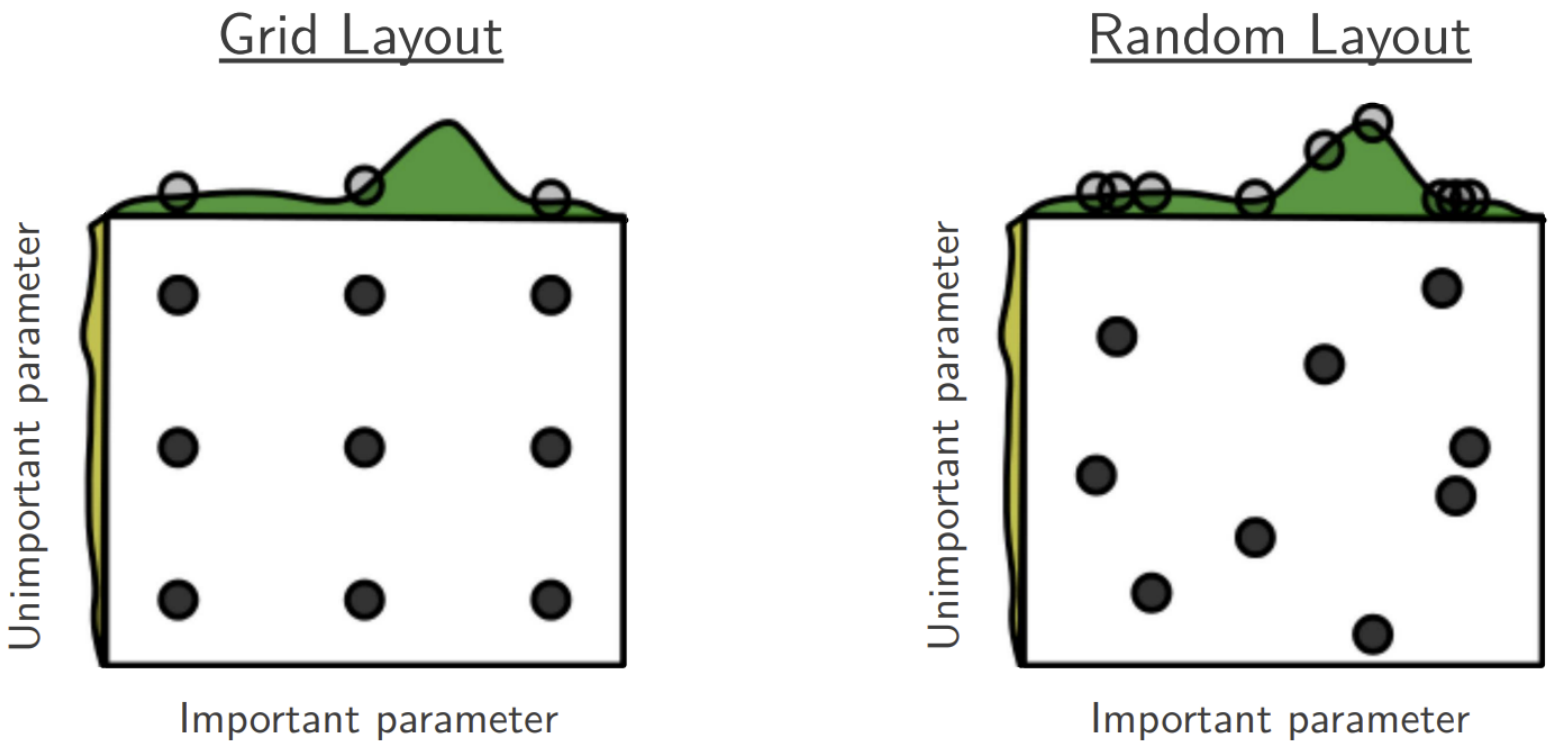
- Stochastic gradient descent (e.g., ADAM)
- Quasi-Newton methods (e.g., BFGS aka `fminunc`/`fmincon`)

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`

Grid and random search

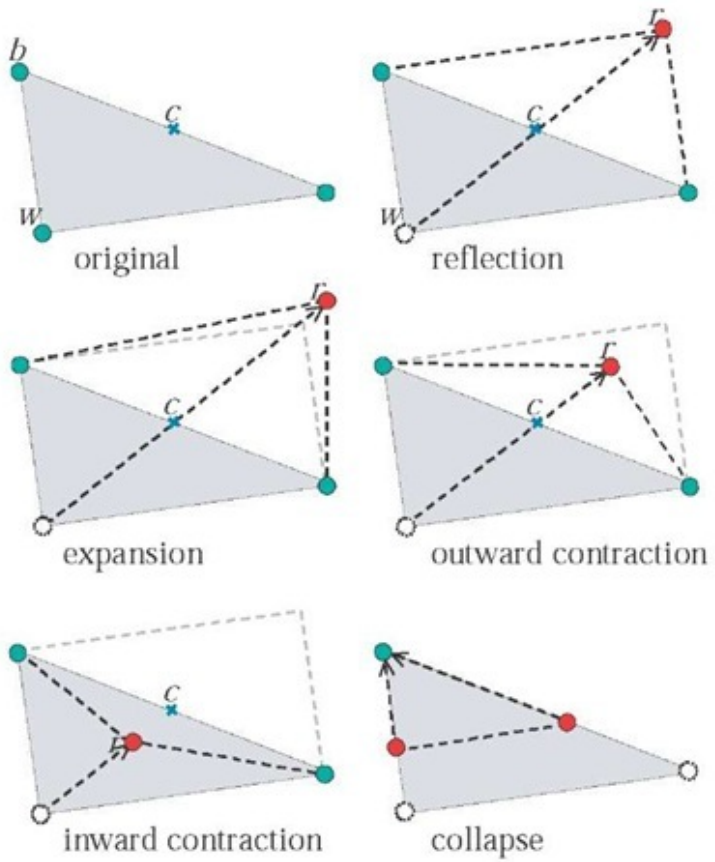


Source: Bergstra & Bengio (2012)

- Curse of dimensionality
- N points per (important) dimension $\Rightarrow N^D$ points

Nelder-Mead (`fminsearch`)

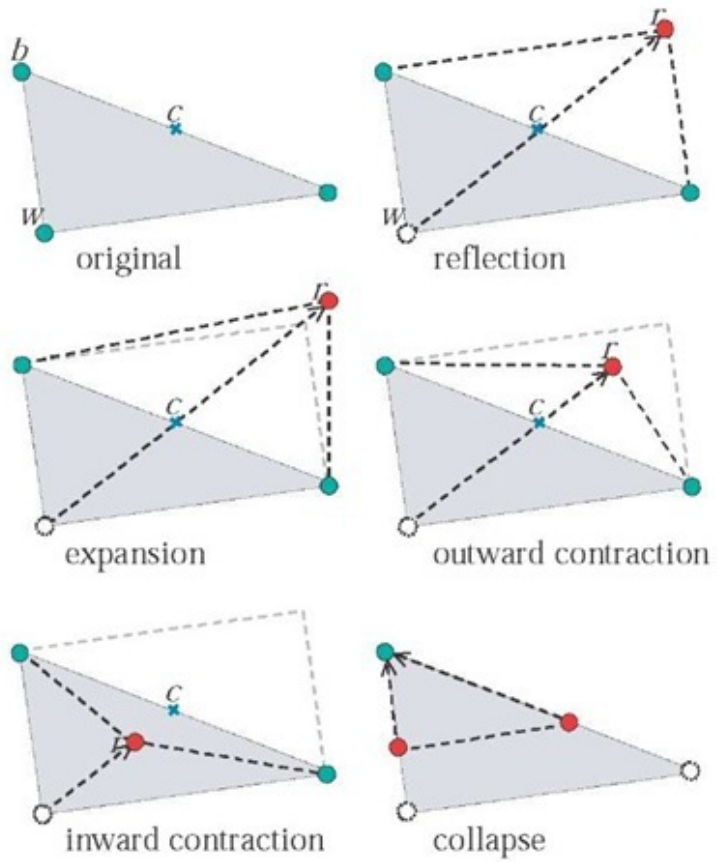
J. A. Nelder & R. Mead, A simplex method for function minimization (1965)



Source: Encyclopedia of Artificial Intelligence (2009)

Nelder-Mead (`fminsearch`)

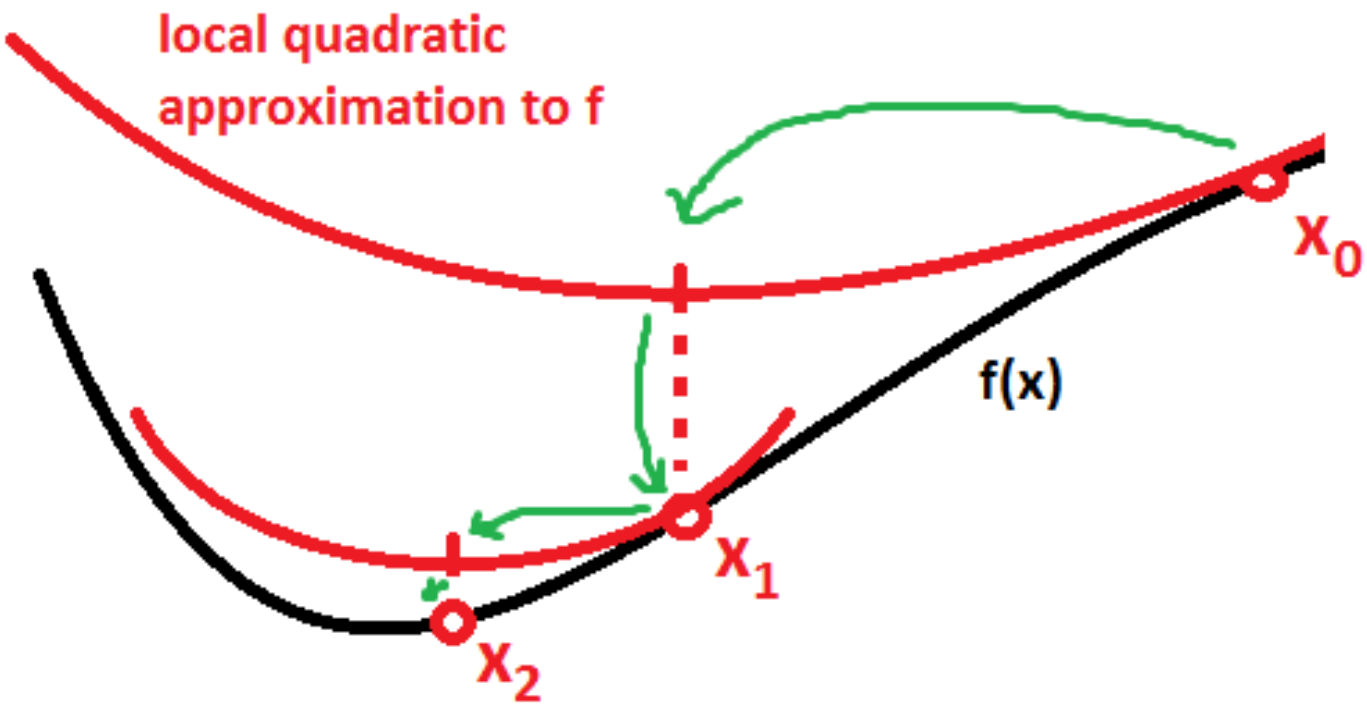
J. A. Nelder & R. Mead, A simplex method for function minimization (1965)



Source: Encyclopedia of Artificial Intelligence (2009)

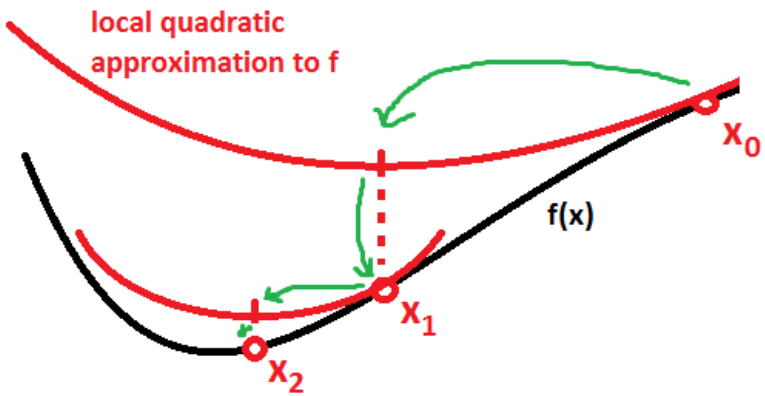
Bounded optimization: `fminsearchbnd` (John d'Errico)

Newton method



Source: StackExchange

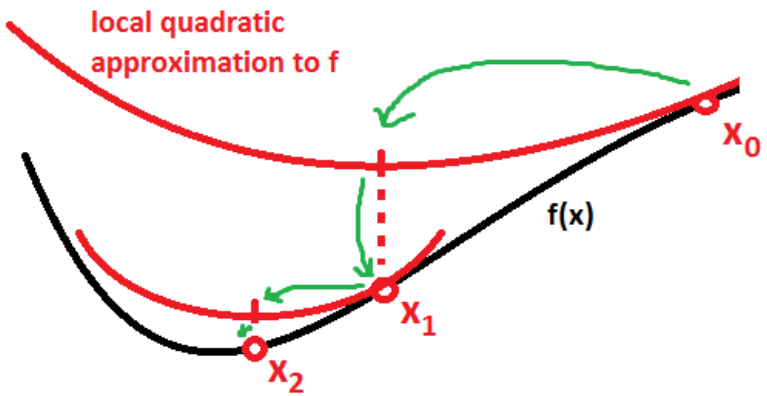
Newton method



Source: StackExchange

Needs the inverse of the curvature (inverse Hessian)
Very expensive in high dimension

Quasi-Newton methods (`fminunc`, `fmincon`)

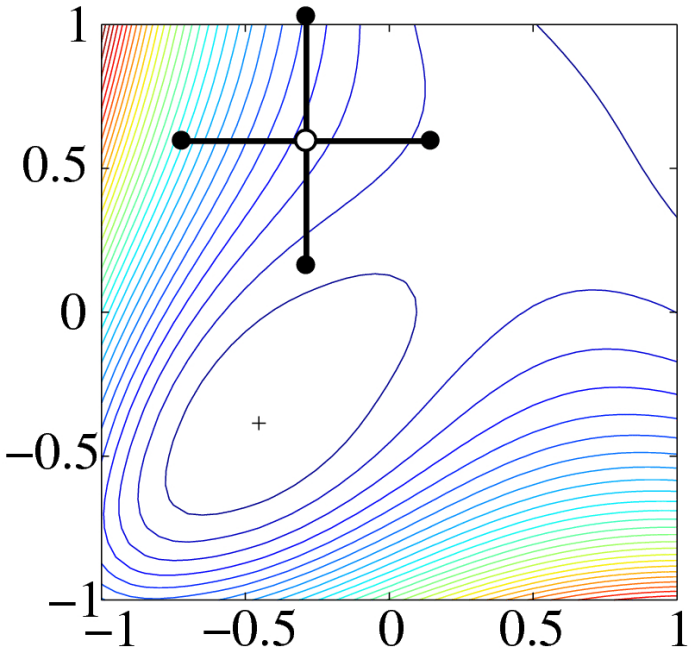


Source: StackExchange

Approximate Hessian (DFP) or inverse Hessian (BFGS) via gradient
Very fast and efficient on smooth problems

Direct search (`patternsearch`)

R. Hooke and T.A. Jeeves, "Direct search" solution of numerical and statistical problems (1961)



Source: Wikimedia Commons

Genetic Algorithms (ga)

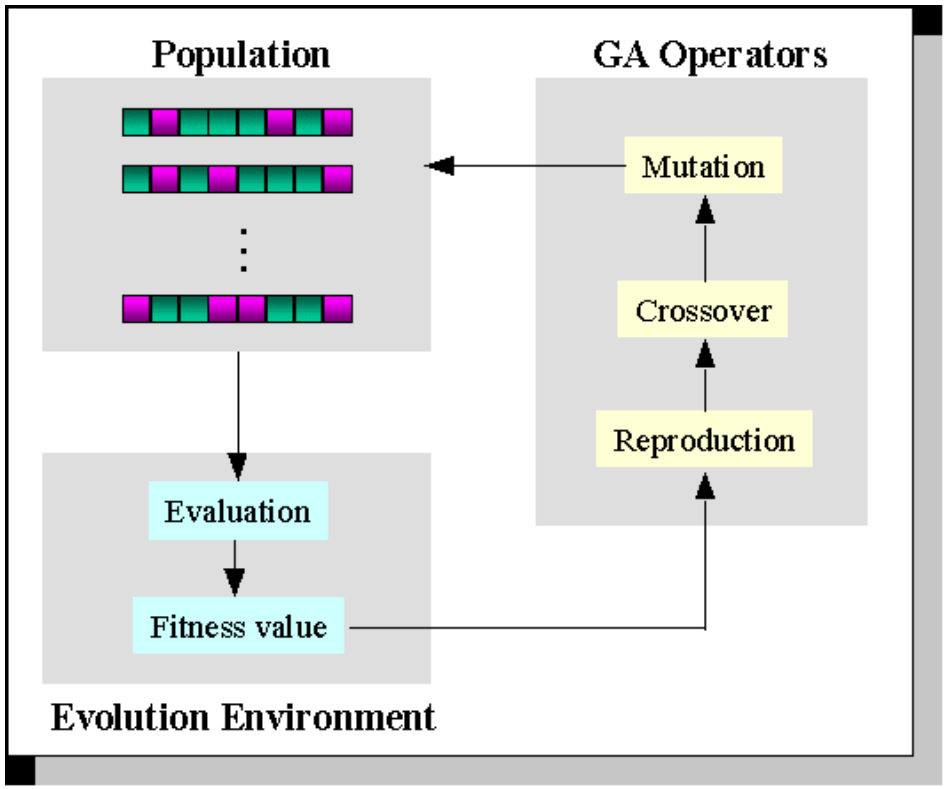
J.H. Holland, Adaptation in Natural and Artificial Systems (1975)

- Evolutionary algorithm
- Population based

Genetic Algorithms (ga)

J.H. Holland, Adaptation in Natural and Artificial Systems (1975)

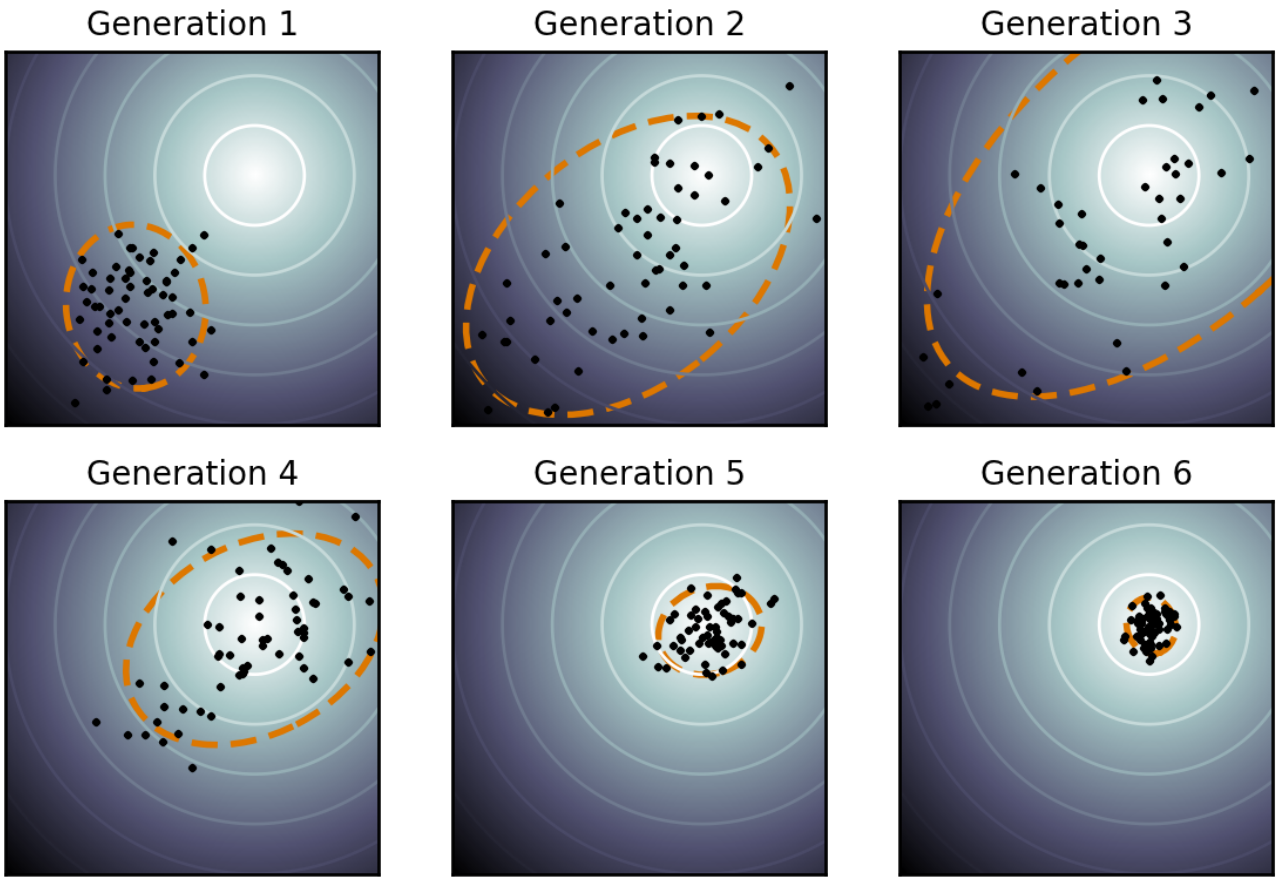
- Evolutionary algorithm
- Population based



Source: An Educational GA Learning Tool (IEEE)

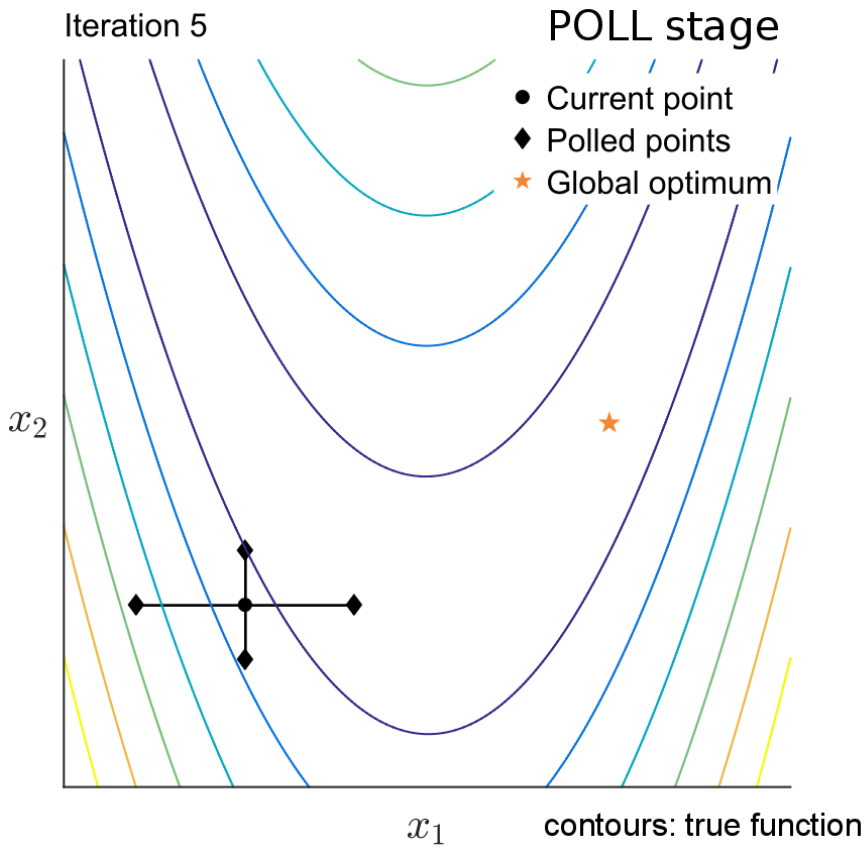
Cov. Matrix Adaptation - Evolution Strategies (cmaes)

[*] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), (2003)



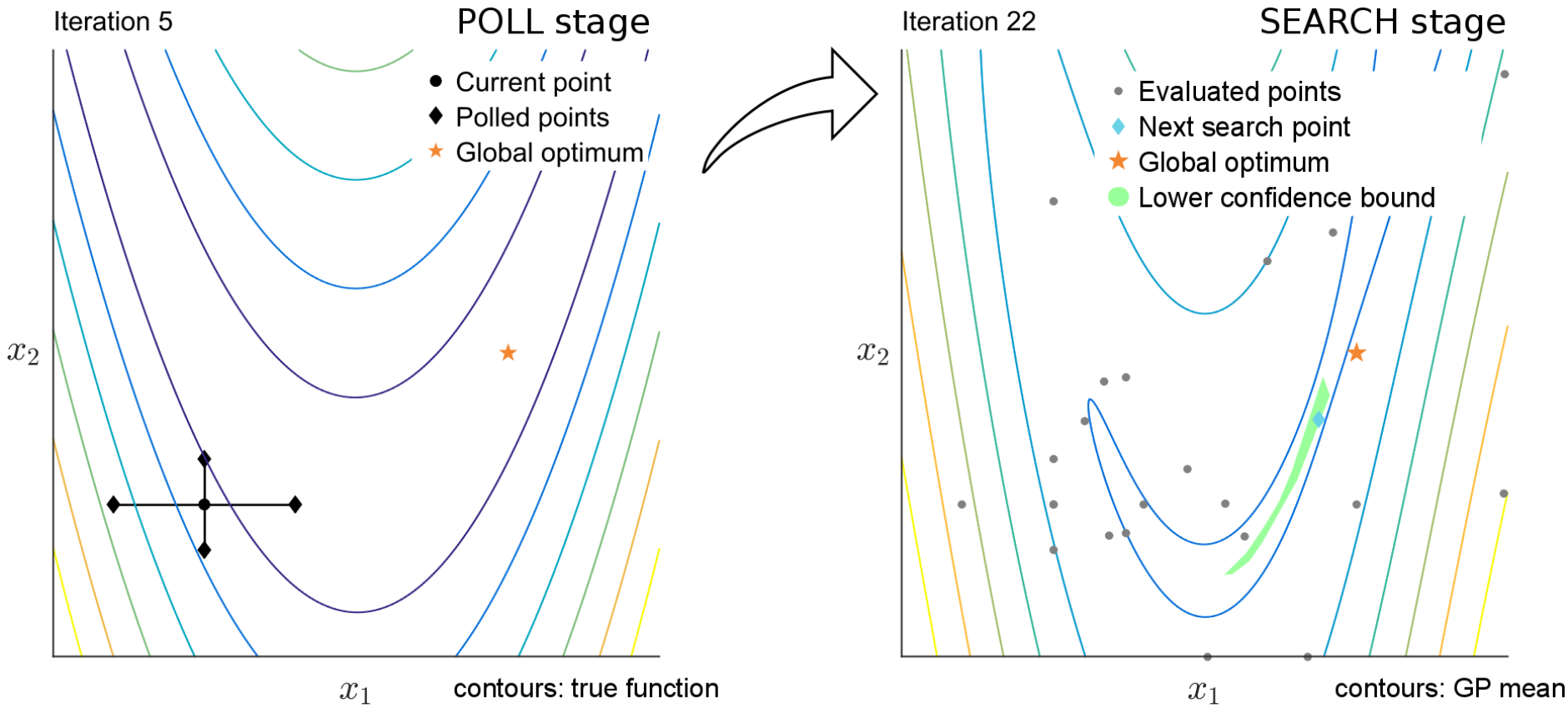
Source: Wikipedia

Bayesian Adaptive Direct Search (bads)



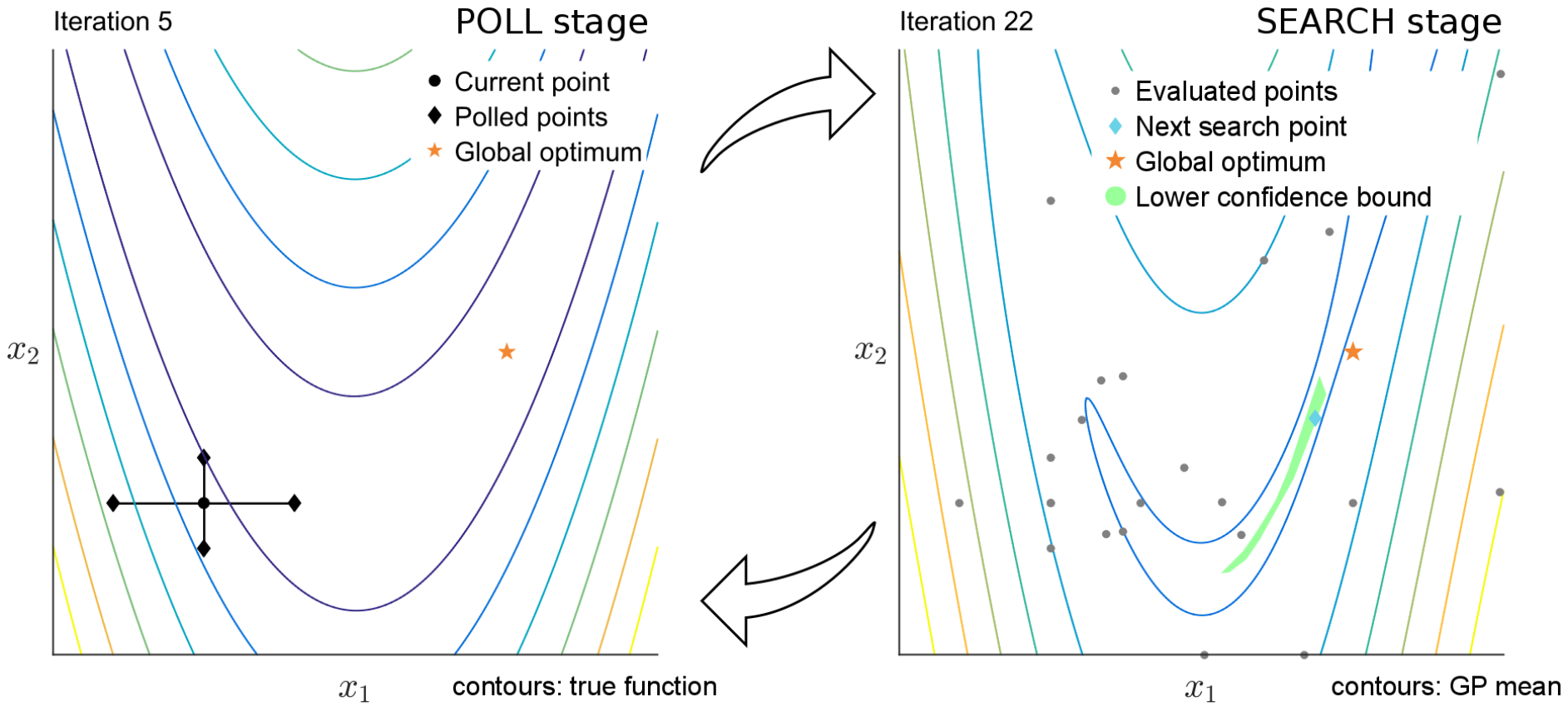
Acerbi & Ma, *NeurIPS* (2017)

Bayesian Adaptive Direct Search (bads)



Acerbi & Ma, *NeurIPS* (2017)

Bayesian Adaptive Direct Search (bads)



Acerbi & Ma, *NeurIPS* (2017)

Local vs. global optimization

BFGS (`fmincon`)

Nelder-Mead (`fminsearch`)

`patternsearch`

SGD

BADS

CMA-ES

Genetic algorithms

Simulated annealing

Bayesian Optimization

MCS

Preparing for optimization

- *Domain* of parameter vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_D) \in \boldsymbol{\Theta}$

Preparing for optimization

- *Domain* of parameter vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_D) \in \boldsymbol{\Theta}$

  In practice, for each θ_d, define
  - The *hard bounds* of the parameter.
    - ★ Mathematical constraints (e.g., $\sigma > 0$; $0 \leq p \leq 1$)
    - ★ Effective physical limitations

Preparing for optimization

- *Domain* of parameter vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_D) \in \boldsymbol{\Theta}$

  In practice, for each θ_d, define
  - The *hard bounds* of the parameter.
    - ★ Mathematical constraints (e.g., $\sigma > 0$; $0 \leq p \leq 1$)
    - ★ Effective physical limitations
  - The *plausible bounds* of the parameter
    - ★ Should span parameter values for most datasets (e.g., 95% prior interval)
    - ★ Built from pilot studies, literature, guesswork
    - ★ If in doubt, start larger

Preparing for optimization

- *Domain* of parameter vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_D) \in \boldsymbol{\Theta}$

  In practice, for each θ_d, define
  - The *hard bounds* of the parameter.
    - ★ Mathematical constraints (e.g., $\sigma > 0$; $0 \leq p \leq 1$)
    - ★ Effective physical limitations
  - The *plausible bounds* of the parameter
    - ★ Should span parameter values for most datasets (e.g., 95% prior interval)
    - ★ Built from pilot studies, literature, guesswork
    - ★ If in doubt, start larger
- Consider reparameterizations to achieve
  - Uniformity of effects across parameter range
  - Independence between parameters

Hacking time IV

Let's optimize the log-likelihood for the psychometric model

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

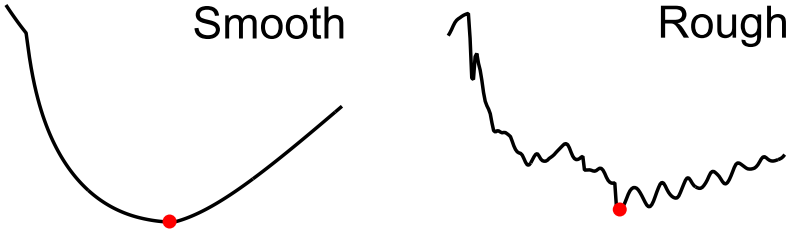
Understand your problem $\Longrightarrow$ often a *gray* box

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

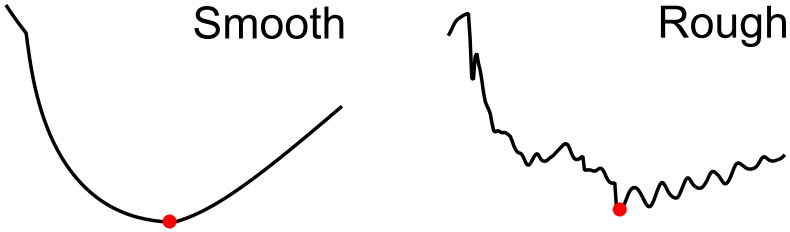
Understand your problem $\Longrightarrow$ often a *gray* box

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

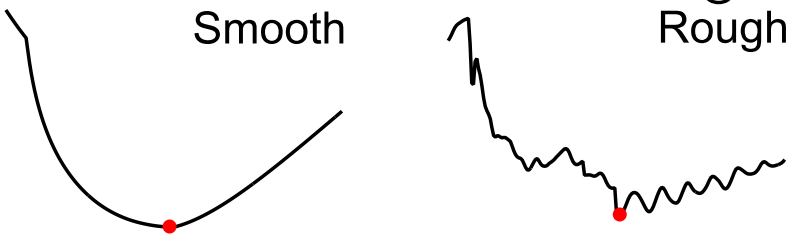
Understand your problem $\Longrightarrow$ often a *gray* box

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 $\mathbb{E}[f(\boldsymbol{x})]$

Optimization cheat sheet, page 1

Rule zero

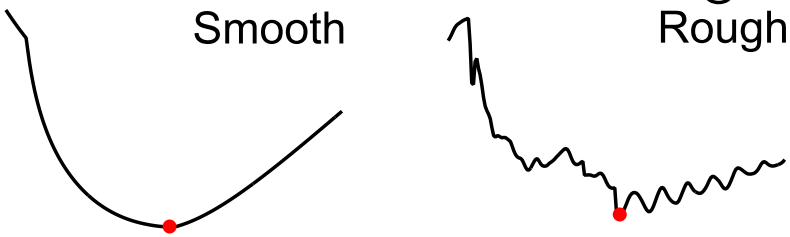
Understand your problem $\Longrightarrow$ often a *gray* box

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

'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
- 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?
  - Tough luck

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

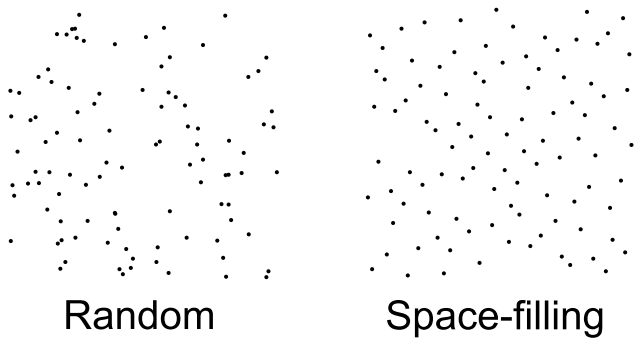
Optimization cheat sheet, page 3

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)



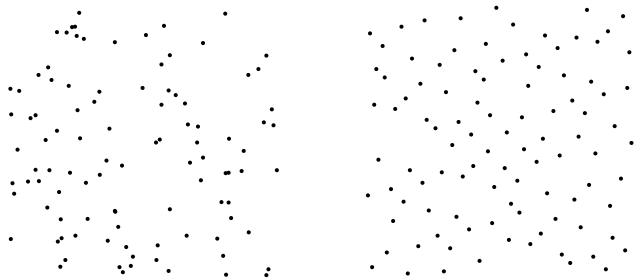
Optimization cheat sheet, page 3

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?

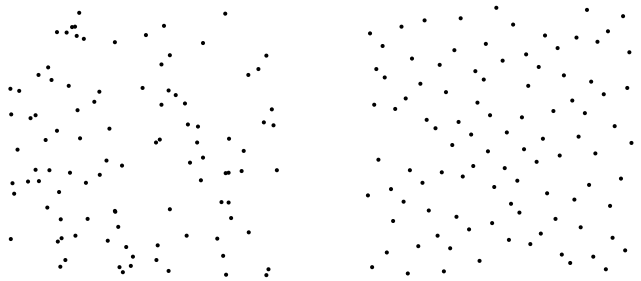
Optimization cheat sheet, page 3

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

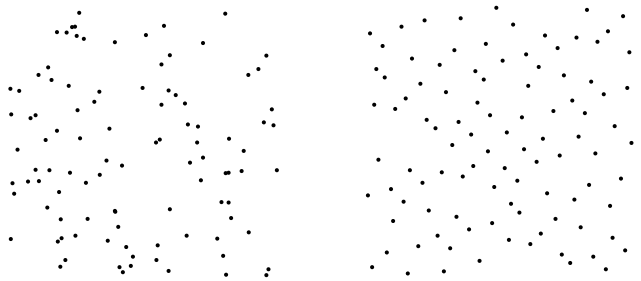
Optimization cheat sheet, page 3

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)



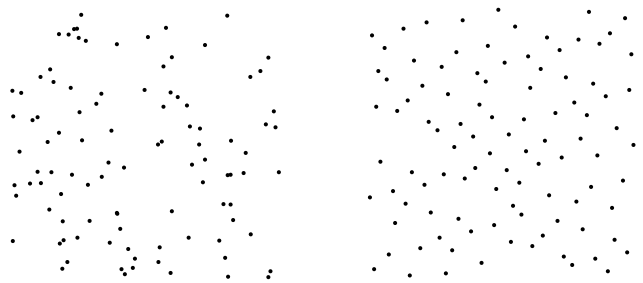
- How many restarts?
  - As many as you need
  - Informally, check that 'most' points converge to the same solution

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)

Learning objectives (did I lie?)

By the end of this lecture/tutorial, we will:

- Explain how model fitting is an optimization problem
- Describe several major optimization algorithms
- Set up and run optimization on real dataset and models
- Summarize practical tips & tricks for optimization

Final slide

- Contacts: `luigi.acerbi@helsinki.fi`, `@AcerbiLuigi`

Code:

- BADS (MATLAB): `github.com/lacerbi/bads`
- PyBADS (Python): In the works!
- Optimization demos: `github.com/lacerbi/optimviz`

Final slide

- Contacts: `luigi.acerbi@helsinki.fi`, `@AcerbiLuigi`

Code:

- BADS (MATLAB): `github.com/lacerbi/bads`
- PyBADS (Python): In the works!
- Optimization demos: `github.com/lacerbi/optimviz`

Thanks!

(Time for questions?)