EECS 553 Lecture 17
Machine Learning (ECE)
[class version]

Jeff Fessler

University of Michigan
©Laura Balzano and Jeff Fessler, UM, October 28, 2026, 23:42

F26

Logistics

© 2026 Fessler

- HW 6 due Wed. Nov. 4
- PA3: Fri. Nov. 6: draft report
- Presentations - see previous lecture
- Mid-semester feedback - thanks!

Recap / Outline

© 2026 Fessler

- Recap
  - Self attention
  - Hyperparameter selection / Holdout
- Today

  - Hyperparameter selection
    - Holdout
    - Bayesian optimization
    - Derivative-free optimization
    - Black-box optimization
    - Cross validation

- Upper level: minimize holdout risk estimate:

$$\boldsymbol{\lambda}_{\text{HO}} \triangleq \arg\min_{\boldsymbol{\lambda}} \hat{R}_{\text{HO}}\left(\hat{f}_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}\right), \quad \hat{R}_{\text{HO}}(\hat{f}) = \frac{1}{n-m} \sum_{i=m+1}^{n} L(y_i, \hat{f}(\boldsymbol{x}_i))$$

- Lower level: optimize model parameters (for given $\boldsymbol{\lambda}$):

$$\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^p} J(\boldsymbol{\theta}; \boldsymbol{\lambda}).$$

  For example:

$$J(\boldsymbol{\theta}; \boldsymbol{\lambda}) = \frac{1}{m} \sum_{i=1}^{m} L\left(y_i, \hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right) + \lambda \left\|\boldsymbol{\theta}\right\|_2^2 .$$

- Earlier we considered $\boldsymbol{\lambda}$ to be "given" and focused on model fitting: $\arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Now we write $J(\boldsymbol{\theta}; \boldsymbol{\lambda})$ and $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ to remind us that $\boldsymbol{\lambda}$ matters.
- Both upper level and lower level are often expensive!

Hyperparameter selection strategies: Overview

© 2026 Fessler

- Different methods for selecting $\boldsymbol{\lambda}$ using $\hat{R}_{\text{HO}}$
  - Analyzing $\hat{R}_{\text{HO}}$ theoretically (rare)
  - Grid search

    Example. For SVM one might try
    $(C, \sigma) \in \{1, 10, 100, 1000\} \times \{10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}\}$.
  - Random search
  -
- Other strategies
  - Common knowledge
  - Tuning by hand
  - Variational methods [2, p. 486, Fig. 10.9]

Hyperparameter selection: Analytically

© 2026 Fessler

- Analyzing $\hat{R}_{\text{HO}}$ theoretically
  - *e.g.*, step size is often driven by Lipshitz constant of your cost function
  - or smoothness properties / properties of Hessian
  - drawback: needs information about your optimization landscape
  - drawback: often provides worst case guarantees when average case can be better
  - limited to one or very few hyperparameters
- Common knowledge
- Tuning by hand
- Grid Search
- Random search
- Bayesian optimization

Hyperparameter selection: Folklore

© 2026 Fessler

- Analyzing $\hat{R}_{\mathrm{HO}}$ theoretically
- Common knowledge
  - many ML researchers might "know" what works and what does not!
  - *e.g.*, typical Adam learning rates are 10^{-4} to 10^{-3}
  - *e.g.*, typical dropout rates are $0.5 - 0.9$
  - drawback: the huge variety of ML problems often means we should not limit ourselves to what worked before
- Tuning by hand
- Grid search
- Random search
- Bayesian optimization

Hyperparameter selection: GSD

© 2026 Fessler

- Analyzing $\hat{R}_{\mathrm{HO}}$ theoretically
- Common knowledge
- Tuning by hand (aka graduate student descent GSD)
  - Fiddle with the hyperparams until you get results you want
  - (unfortunately) very commonly used!
  - drawback: lots of effort usually required
  - drawback: not principled / generalizable / reproducible
  - tempting to cheat by looking at test data
  - but can contribute to "common knowledge"
- Grid search
- Random search
- Bayesian optimization

Hyperparameter selection: Grid search

© 2026 Fessler

- Analyzing $\hat{R}_{\text{HO}}$ theoretically
- Common knowledge
- Tuning by hand
- **Grid search** or **parameter sweep**
  - Define a range of parameters to try
  - *e.g.*, step size from 10^{-4} to 10^{-2} and $\lambda = 0.5 - 2$
  - then create a grid with several options between the endpoints
  - essentially brute force and you must choose the grid properly
  - can be viable for 1-3 dimensional hyperparameters
  - drawback: as the number of hyperparams increases, the cost of grid search increases exponentially.
- Random search
- Bayesian optimization

CQ: Grid search

© 2026 Fessler

Q17.1 Here is an example risk contour plot for hyperparameter grid search. What should the axes labels be?

A: x, y B: x_1, x_2 C: y_1, y_2 D: θ_1, θ_2 E: λ_1, λ_2

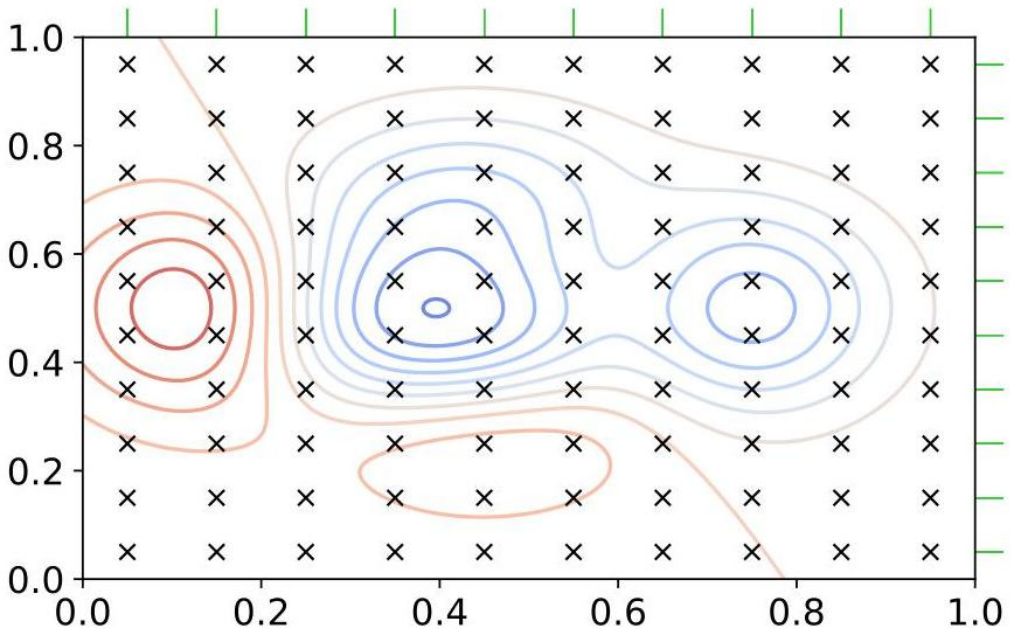


Image courtesy A. Elevers

Accelerating grid search

© 2026 Fessler

- One can speed up grid search using early stopping and successive halving
- Run all grid points for one epoch, discard the worst half and continue
  (Specific to GD/SGD training.)

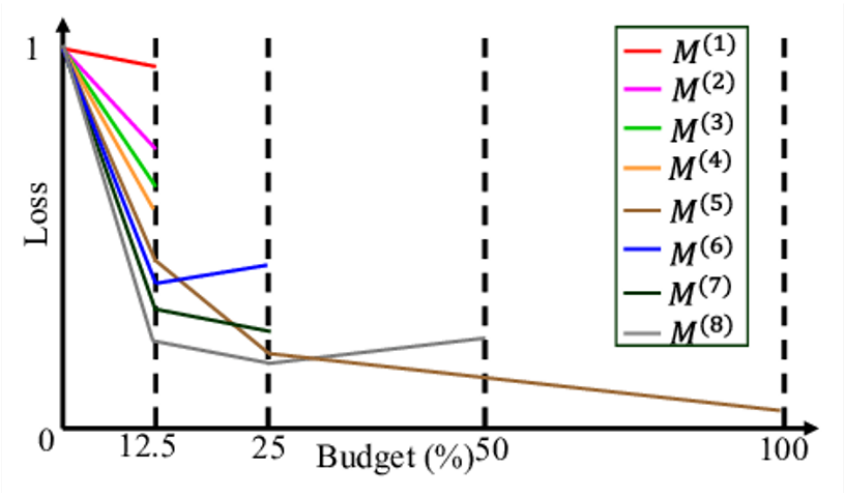


Figure courtesy A. Krunz

- Benefit: highly parallel
- Drawback:
- Alternative grid search:

- Analyzing $\hat{R}_{\text{HO}}$ theoretically
- Common knowledge
- Tuning by hand
- Grid search
- **Random search**
  - Select hyperparameter vectors $\boldsymbol{\lambda}$ randomly from some range
  - Try however many that you can afford computationally
  - Benefit: simple
  - Drawback: unlikely to get close to optimal in a high-dimensional space
- Bayesian optimization

Bayesian optimization: Overview

© 2026 Fessler

- **Bayesian optimization**: a sequential statistical approach for minimizing noisy black-box functions like $F(\boldsymbol{\lambda}) = \hat{R}_{\text{HO}}\left(\hat{f}_{\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})}\right)$
- **Key idea**: learn a statistical model of the function $F(\boldsymbol{\lambda})$ that maps hyperparameter values $\boldsymbol{\lambda}$ to the HO risk
  - *e.g.*, **Gaussian process** model (also used in ML more broadly)
- Choose parameters to (try to) minimize F under this model
- **Main benefit:** choose candidate hyperparameter vectors

  somewhat reducing the curse of dimensionality.
- Empirically it has been demonstrated to get better results in fewer "experiments" ($\boldsymbol{\lambda}$ values)
- Somewhat widely used, with lots of opportunity for innovation/research.

Bayesian optimization: Basics

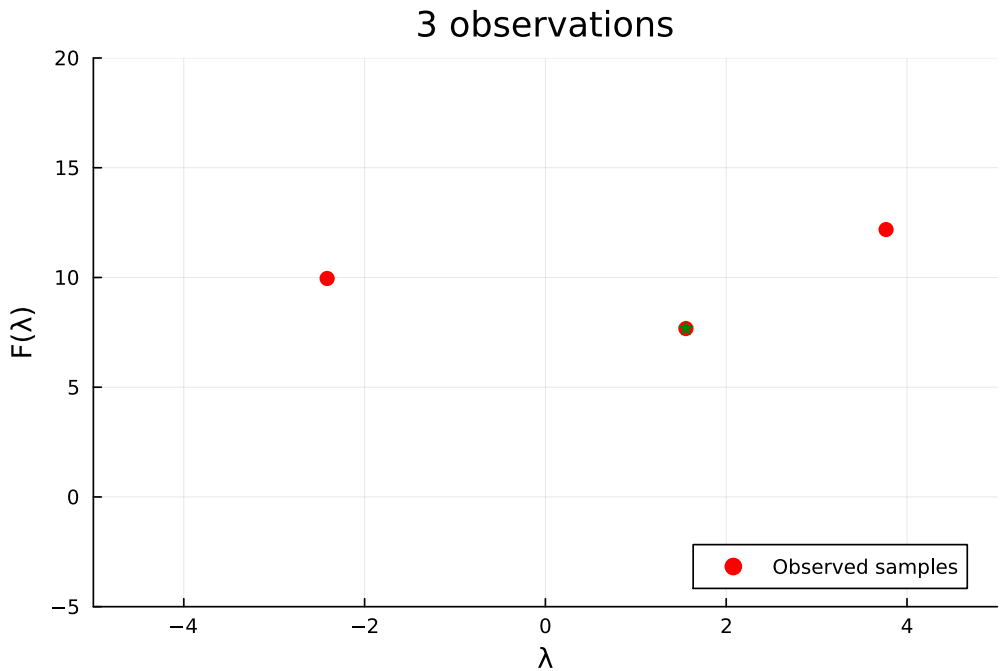
© 2026 Fessler

- It could deserve a whole lecture. Fundamental topic...
- Model the target function F as a **Gaussian Process**
- **Bayesian optimization** uses a practical recursive approach:
  - Future HP "trials" depend on the past $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t$ and
  - Past trials are used to:
    - determine subsequent HP choices with good accuracy
    - quantify the uncertainty of F for candidate HP choices
- **Exploration-exploitation tradeoff**
  - Next sample $\boldsymbol{\lambda}_{t+1}$ should identify promising HP choices
  - Uncertain regions might appear to have low accuracy (high risk), but they might contain the optimal HP choice!
- **Acquisition function**: Optimizes exploration-exploitation tradeoff of the next sample given the history.

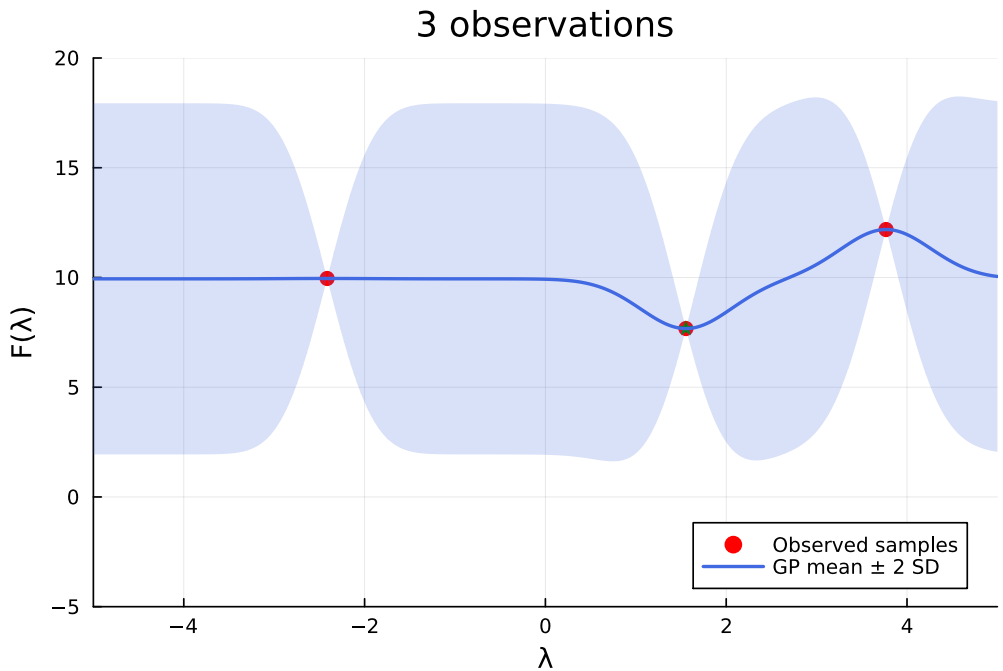
Start of slides from Prof. Samet Oymak, who in turn got them from someone else

Bayesian optimization: Example

© 2026 Fessler



- Having obtained initial function values $F(\boldsymbol{\lambda}_1), F(\boldsymbol{\lambda}_2), F(\boldsymbol{\lambda}_3)$ at some (random) locations, what value of $\boldsymbol{\lambda}$ should we try next?
- This is the universal challenge in black-box optimization.
- We need to make some model for F.
- A simple piecewise linear model would be useless.



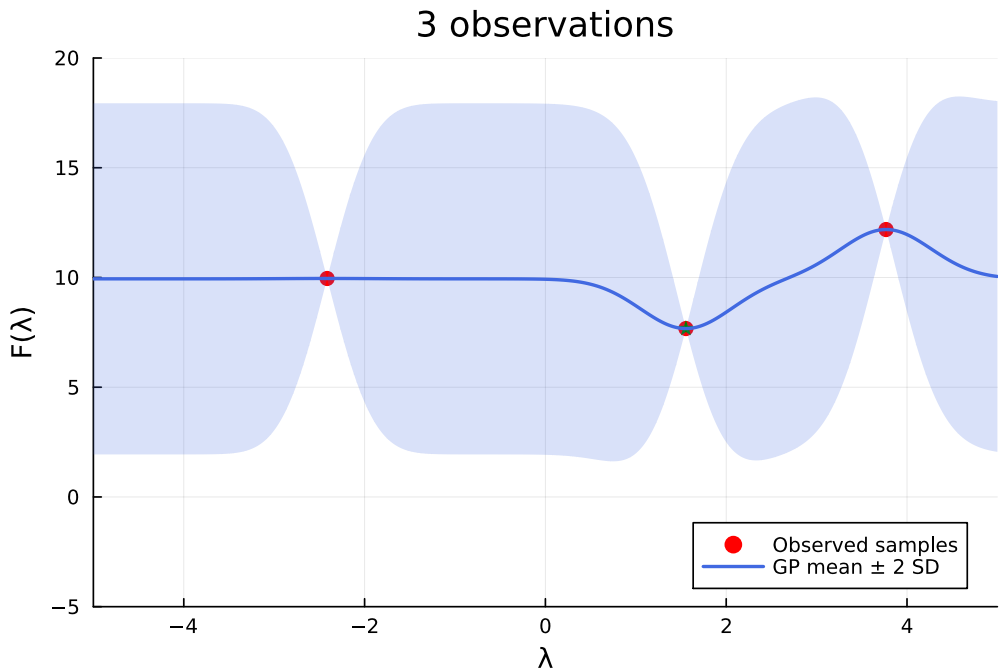
- BO uses a Gaussian process model for $F(\cdot)$ with **kernel** $k(\cdot, \cdot)$:

$$\boldsymbol{y} = \boldsymbol{F} + \varepsilon, \quad \boldsymbol{F} = \begin{bmatrix} F(\boldsymbol{\lambda}_1) \\ \vdots \\ F(\boldsymbol{\lambda}_t) \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}), \quad K_{ij} = k(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j).$$

- Blue line: posterior mean $\mu(\boldsymbol{\lambda}) \triangleq \mathbb{E}\{F(\boldsymbol{\lambda}) \mid \boldsymbol{y}\}$
- Shaded area shows uncertainty: $\mu(\boldsymbol{\lambda}) \pm 2\sigma(F(\boldsymbol{\lambda}) \mid \boldsymbol{y})$

Bayesian optimization: Update

© 2026 Fessler

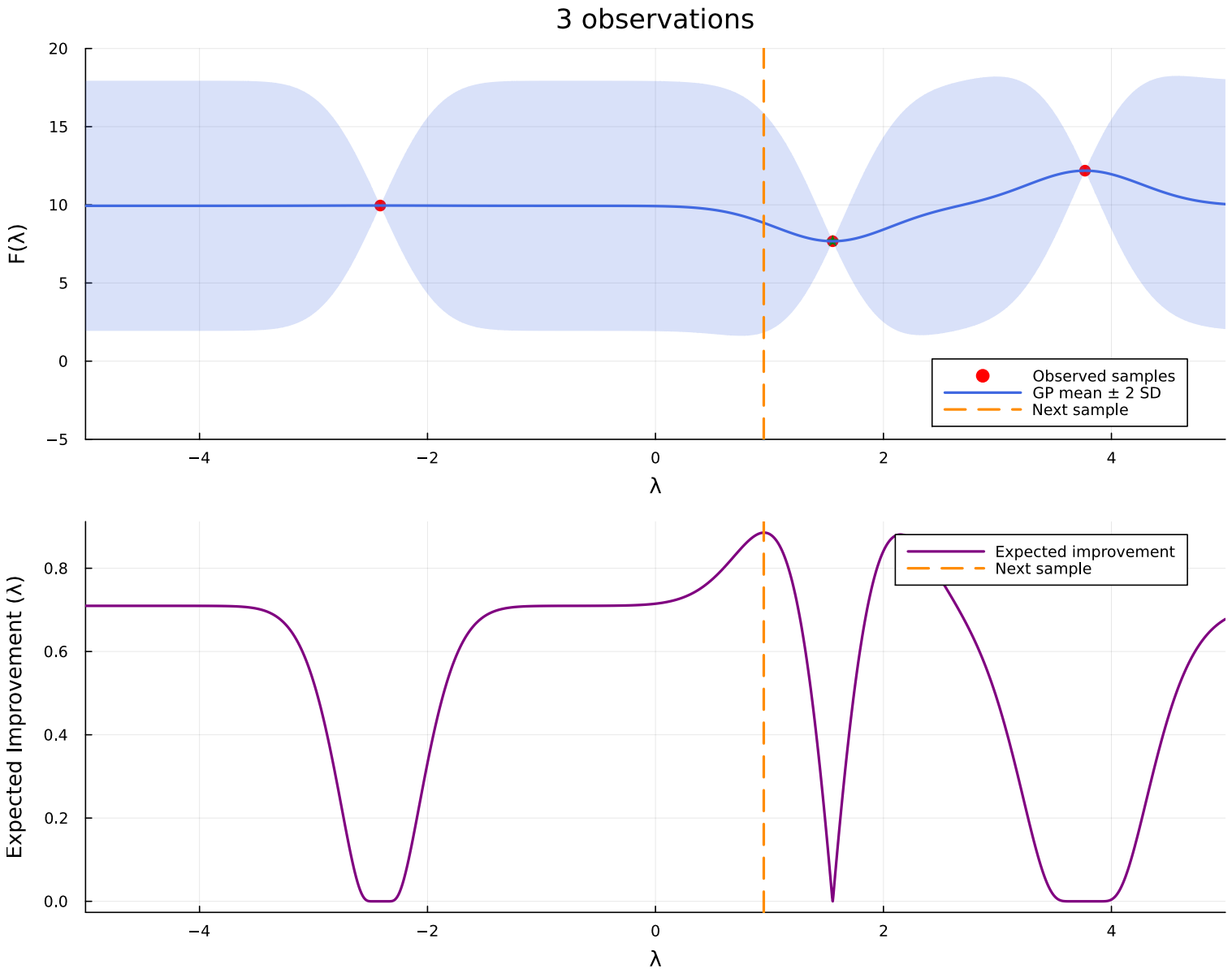


Options for next sample $\boldsymbol{\lambda}_{t+1}$ involve an

- **Lower confidence bound**: $\arg\min_{\boldsymbol{\lambda}} \mu(\boldsymbol{\lambda}) - 2\sigma(F(\boldsymbol{\lambda}) \mid \boldsymbol{y})$.
- **Probability of improvement**:
  $\arg\max_{\boldsymbol{\lambda}} \mathrm{P}\{F(\boldsymbol{\lambda}) < F_{\min}\}$, $F_{\min} = \min_{i \in \{1,\dots,t\}} F(\boldsymbol{\lambda}_i)$.
- **Expected improvement** approach:
  $\arg\min_{\boldsymbol{\lambda}} \mathbb{E}\{\max(0, F_{\min} - F(\boldsymbol{\lambda})) \mid \boldsymbol{y}\}$.
- All of these easily computable for Gaussian model; easy to optimize over $\boldsymbol{\lambda}$.

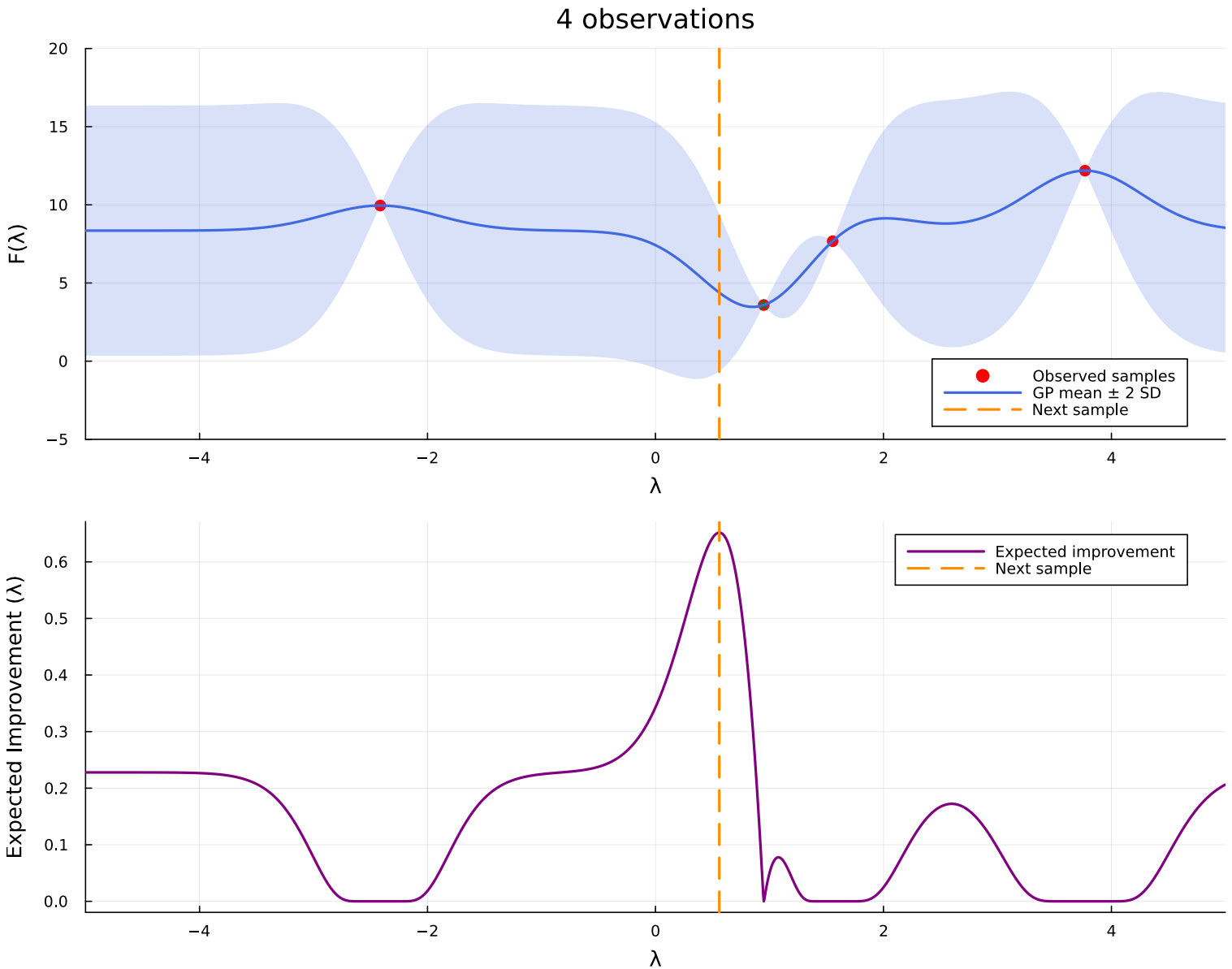
Bayesian optimization: Iteration 00

© 2026 Fessler



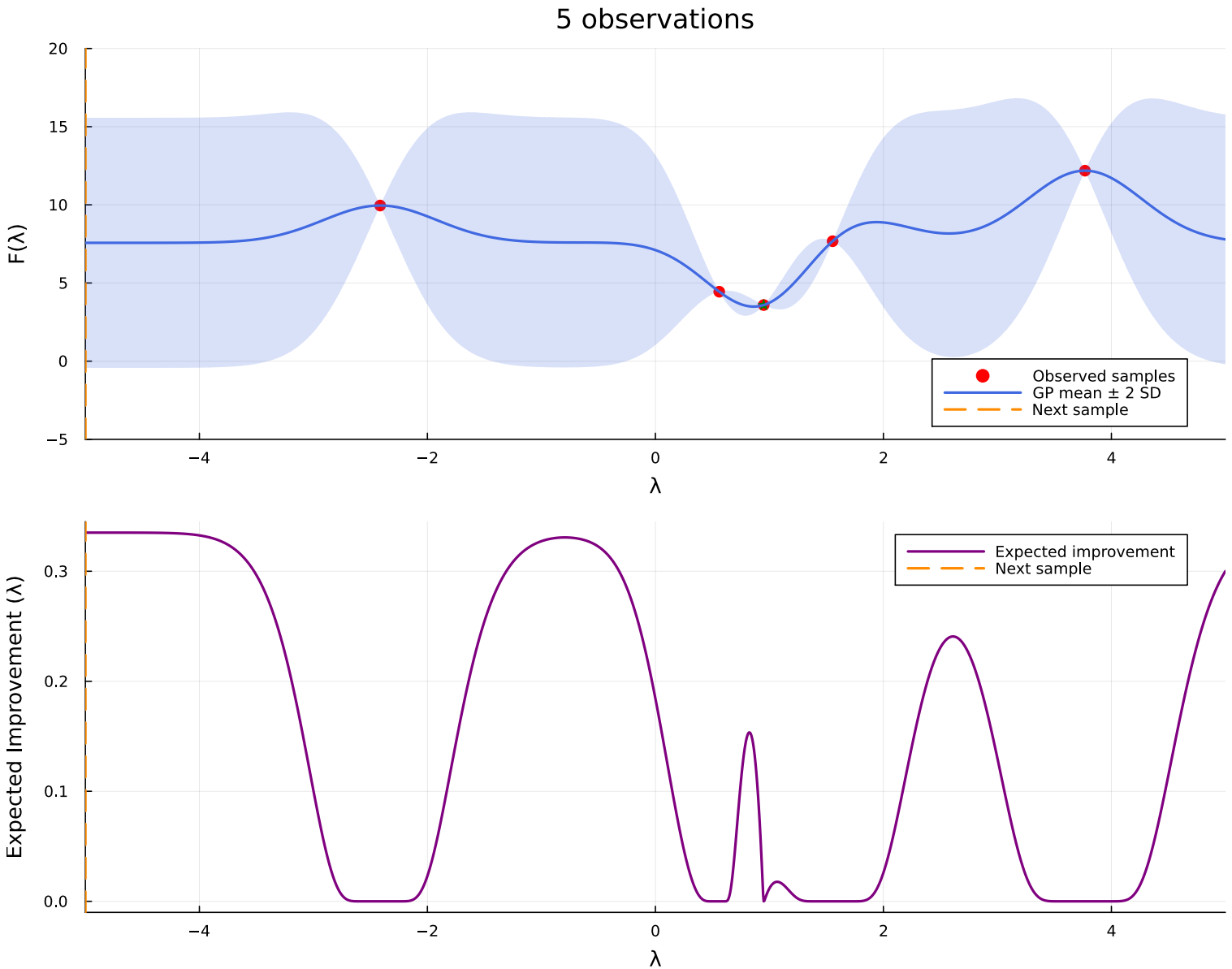
Bayesian optimization: Iteration 01

© 2026 Fessler



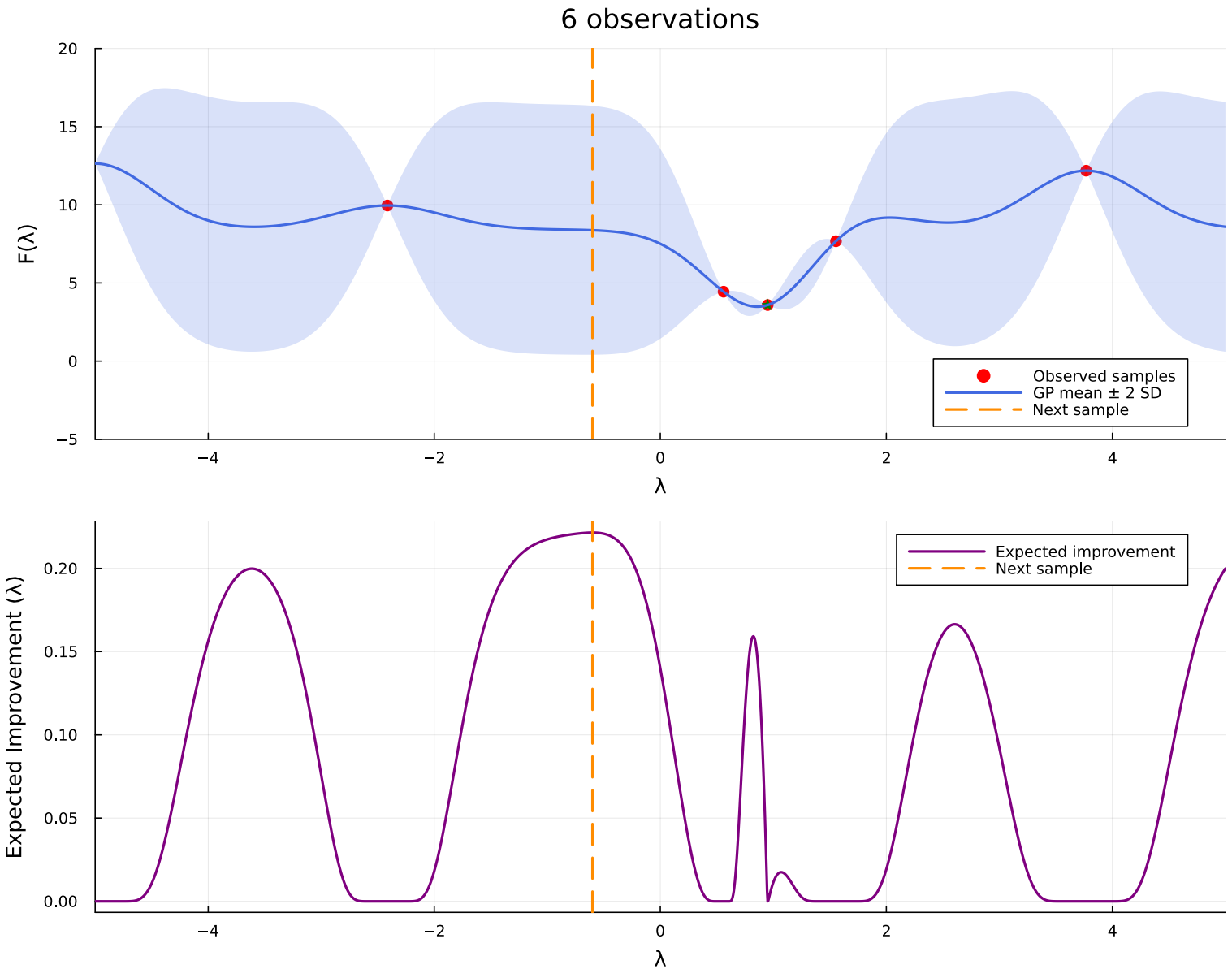
Bayesian optimization: Iteration 02

© 2026 Fessler



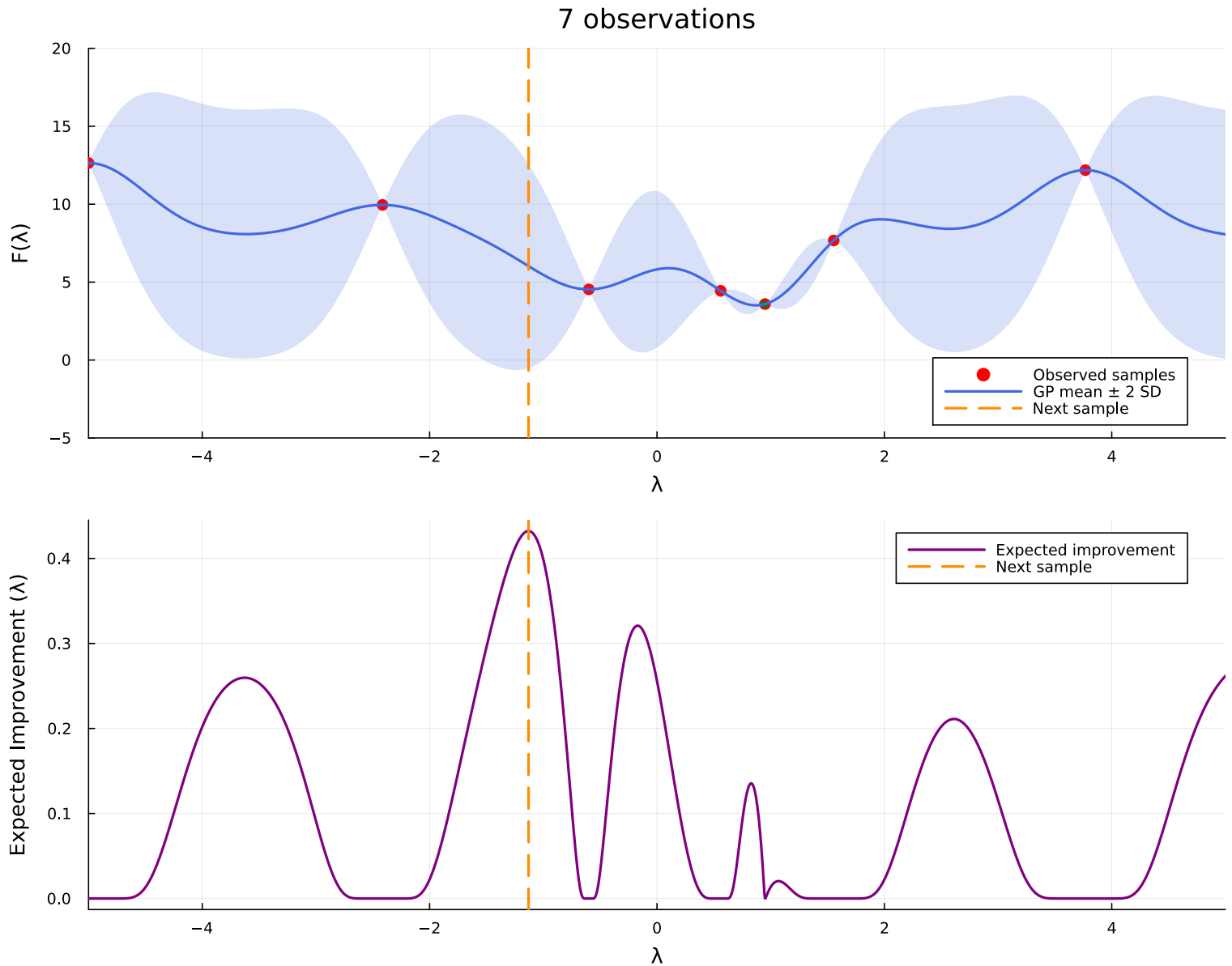
Bayesian optimization: Iteration 03

© 2026 Fessler



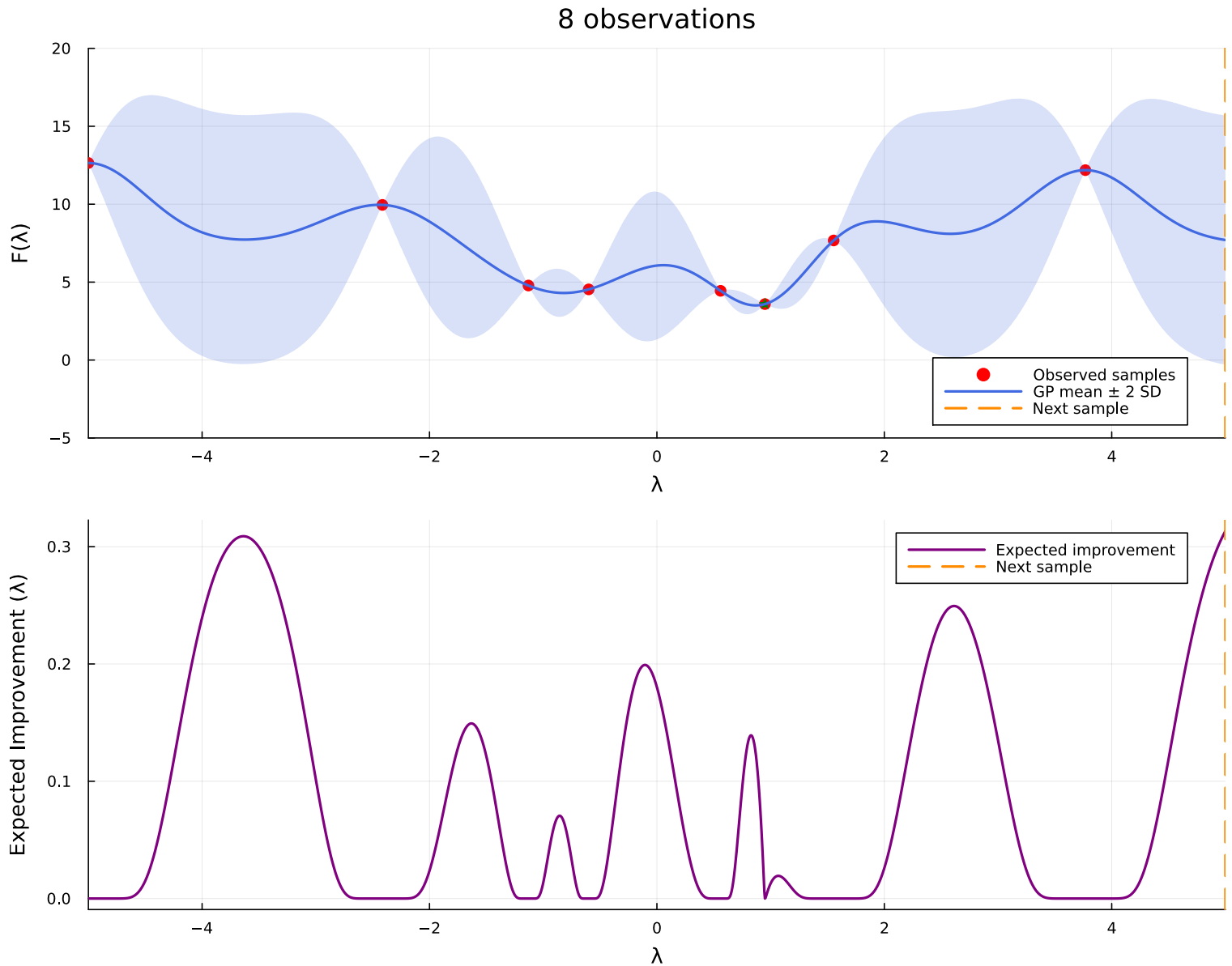
Bayesian optimization: Iteration 04

© 2026 Fessler



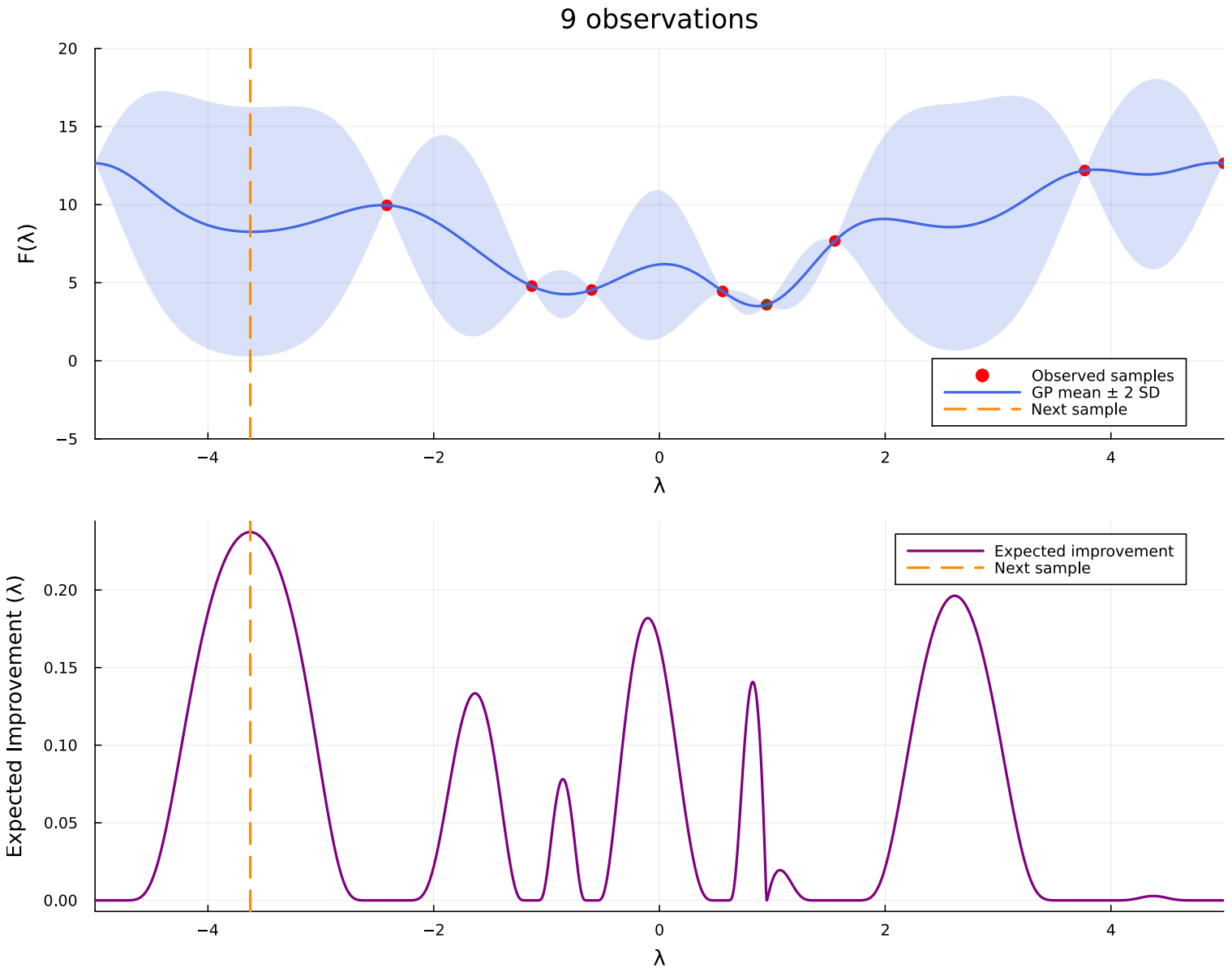
Bayesian optimization: Iteration 05

© 2026 Fessler



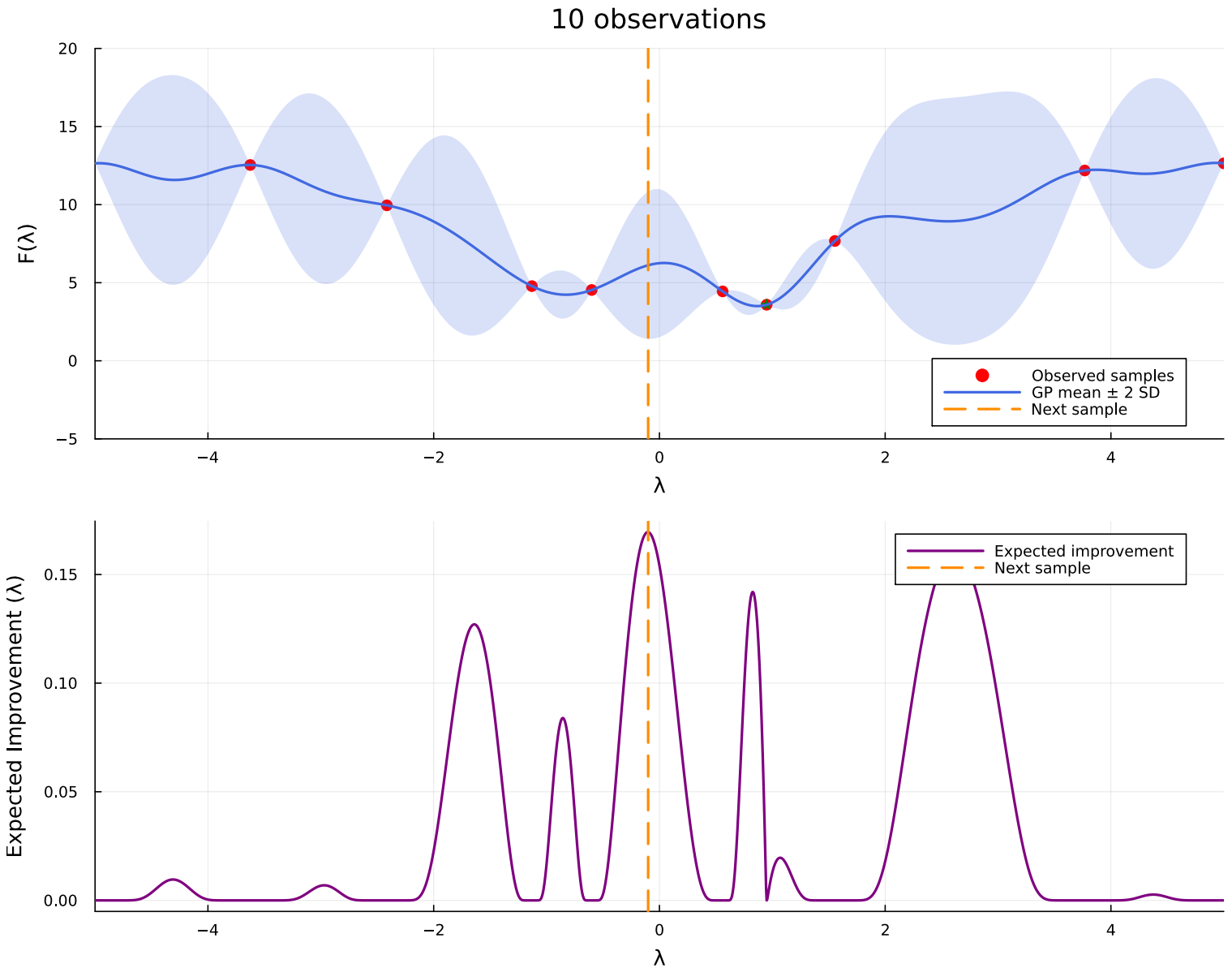
Bayesian optimization: Iteration 06

© 2026 Fessler



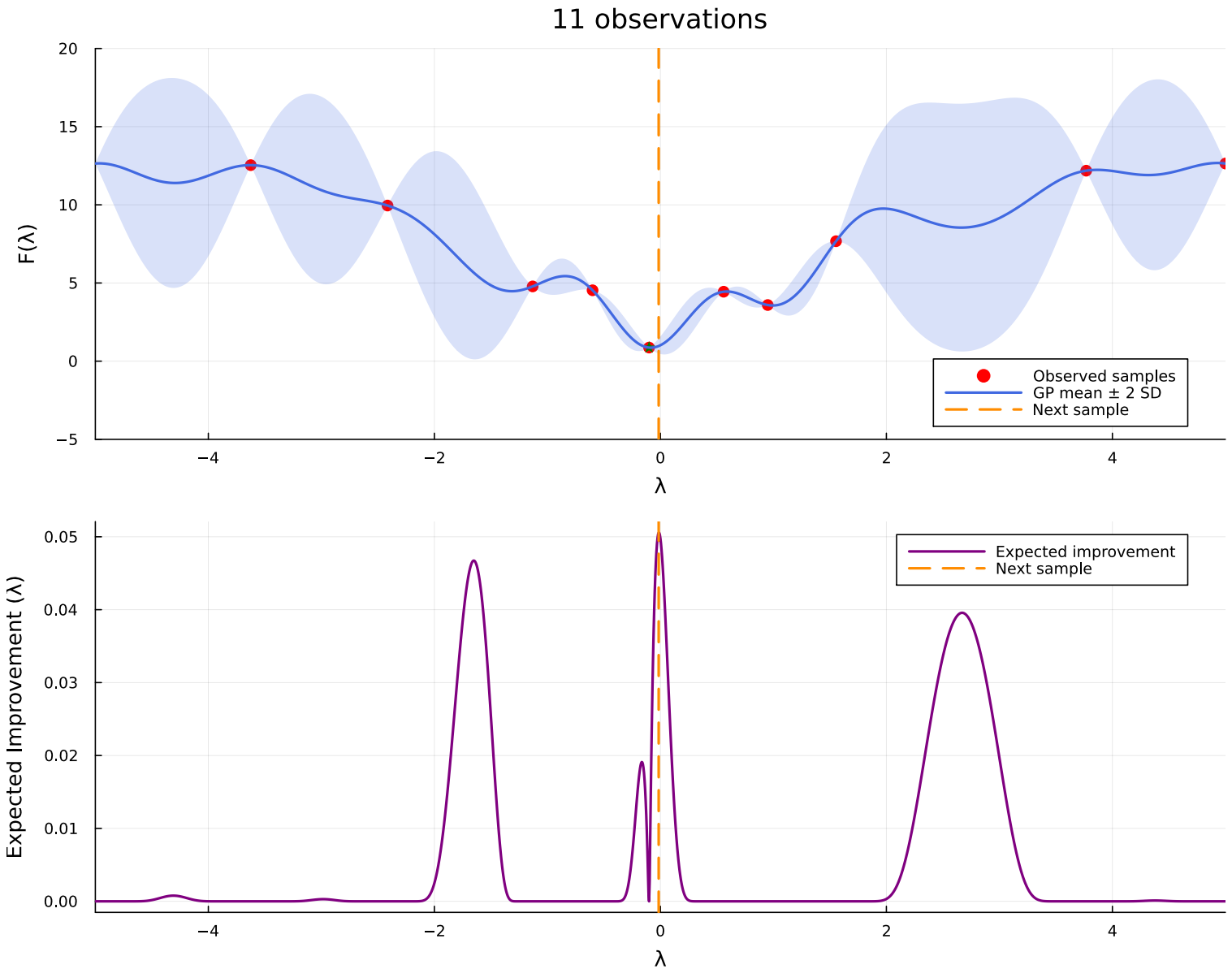
Bayesian optimization: Iteration 07

© 2026 Fessler



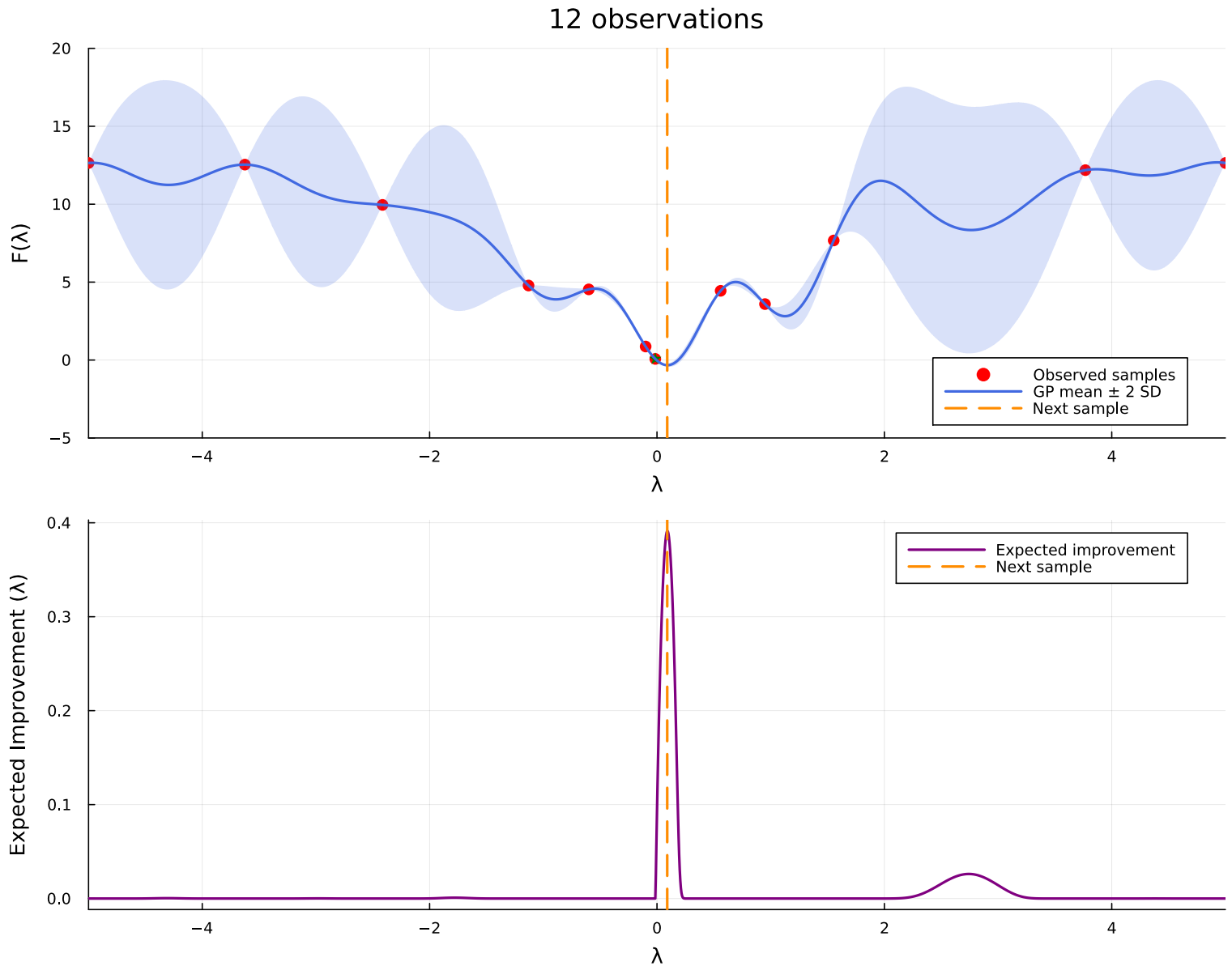
Bayesian optimization: Iteration 08

© 2026 Fessler



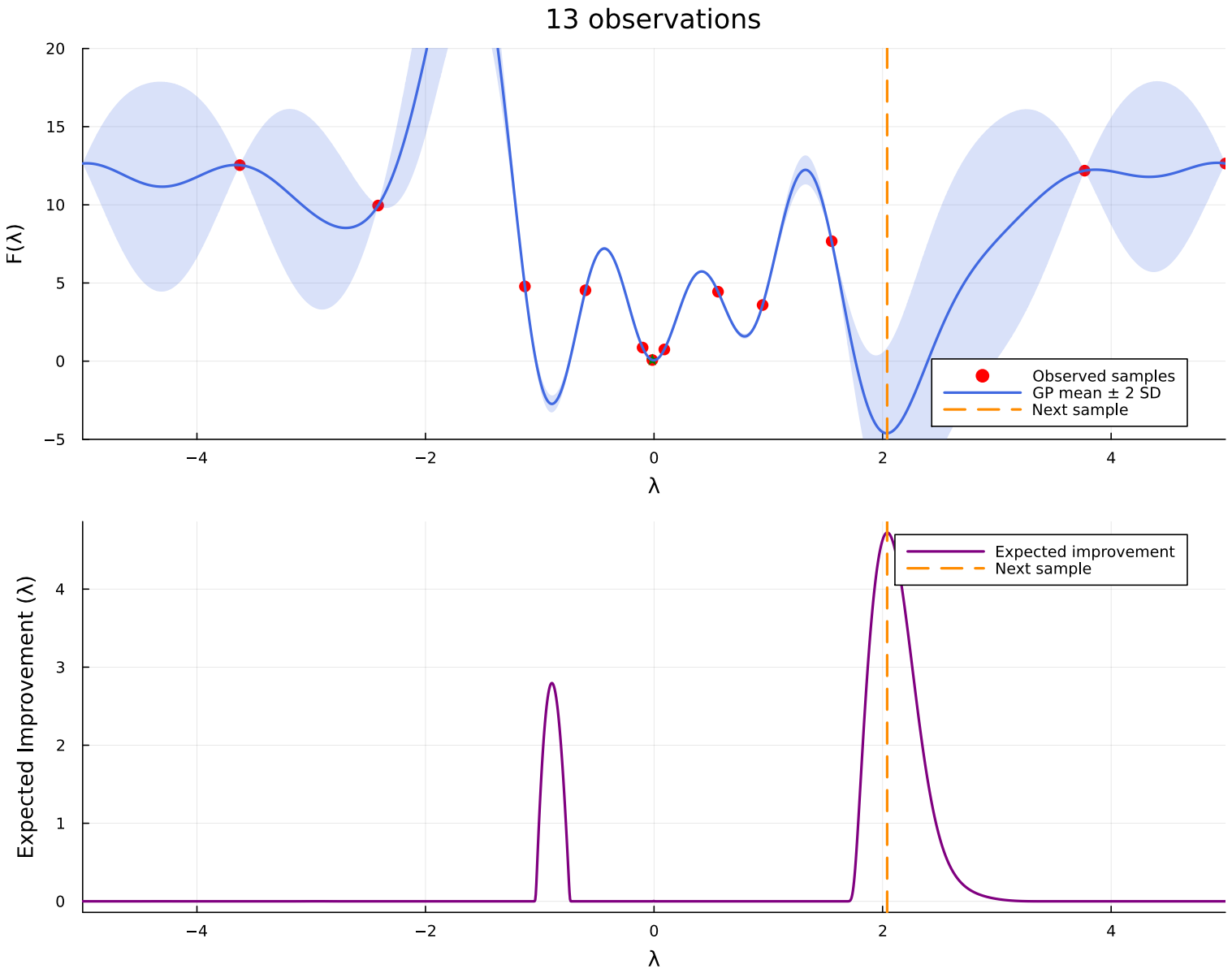
Bayesian optimization: Iteration 09

© 2026 Fessler



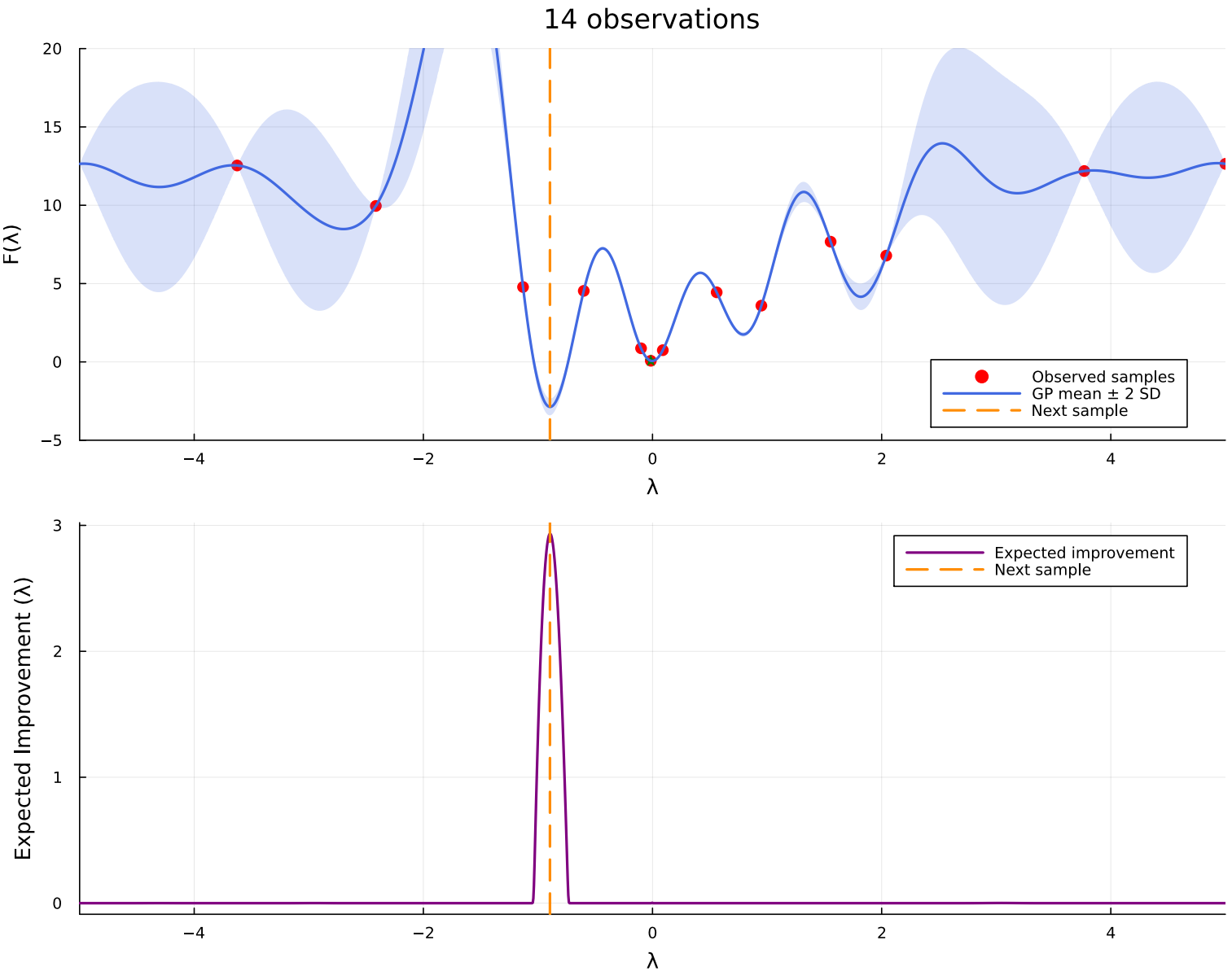
Bayesian optimization: Iteration 10

© 2026 Fessler



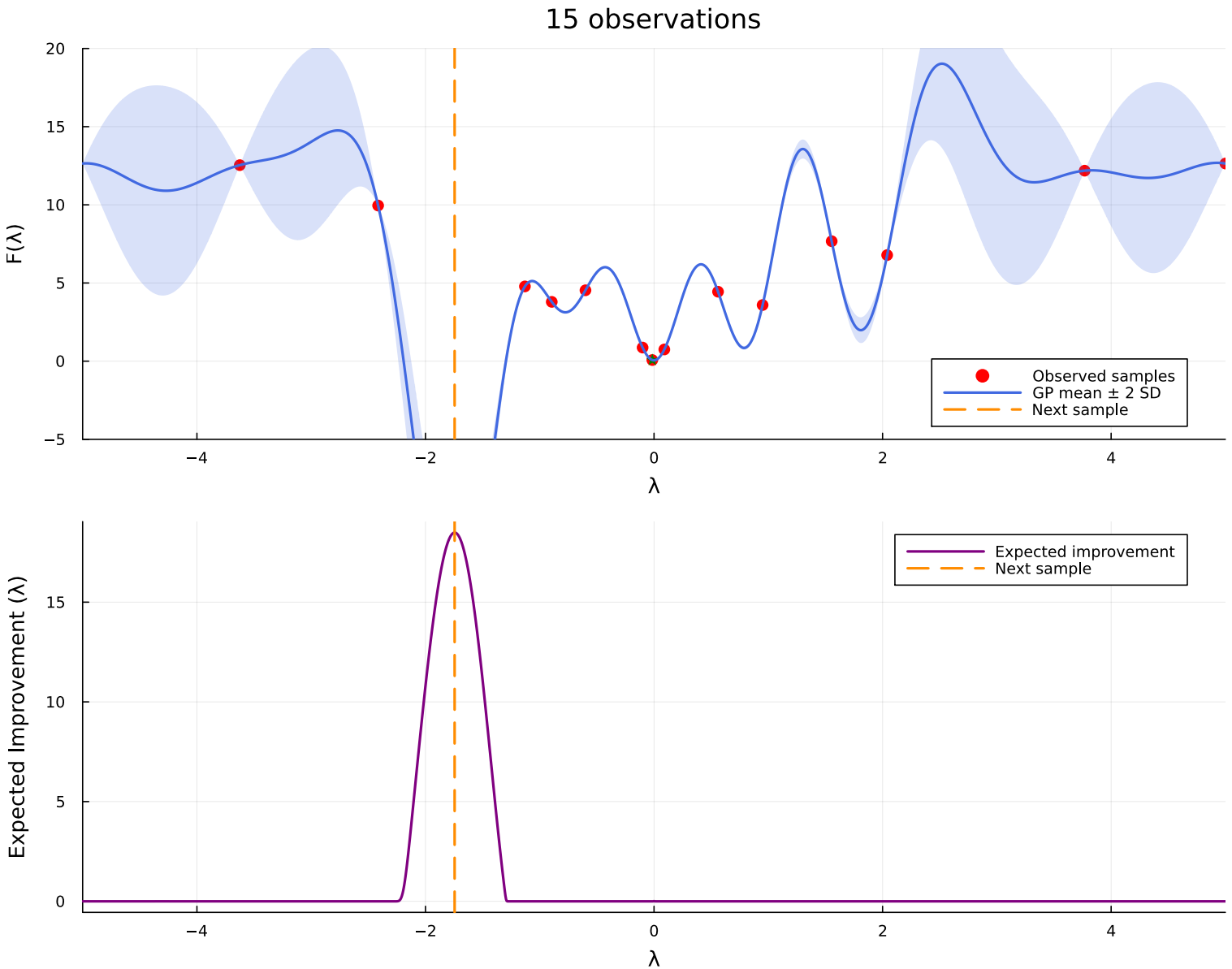
Bayesian optimization: Iteration 11

© 2026 Fessler



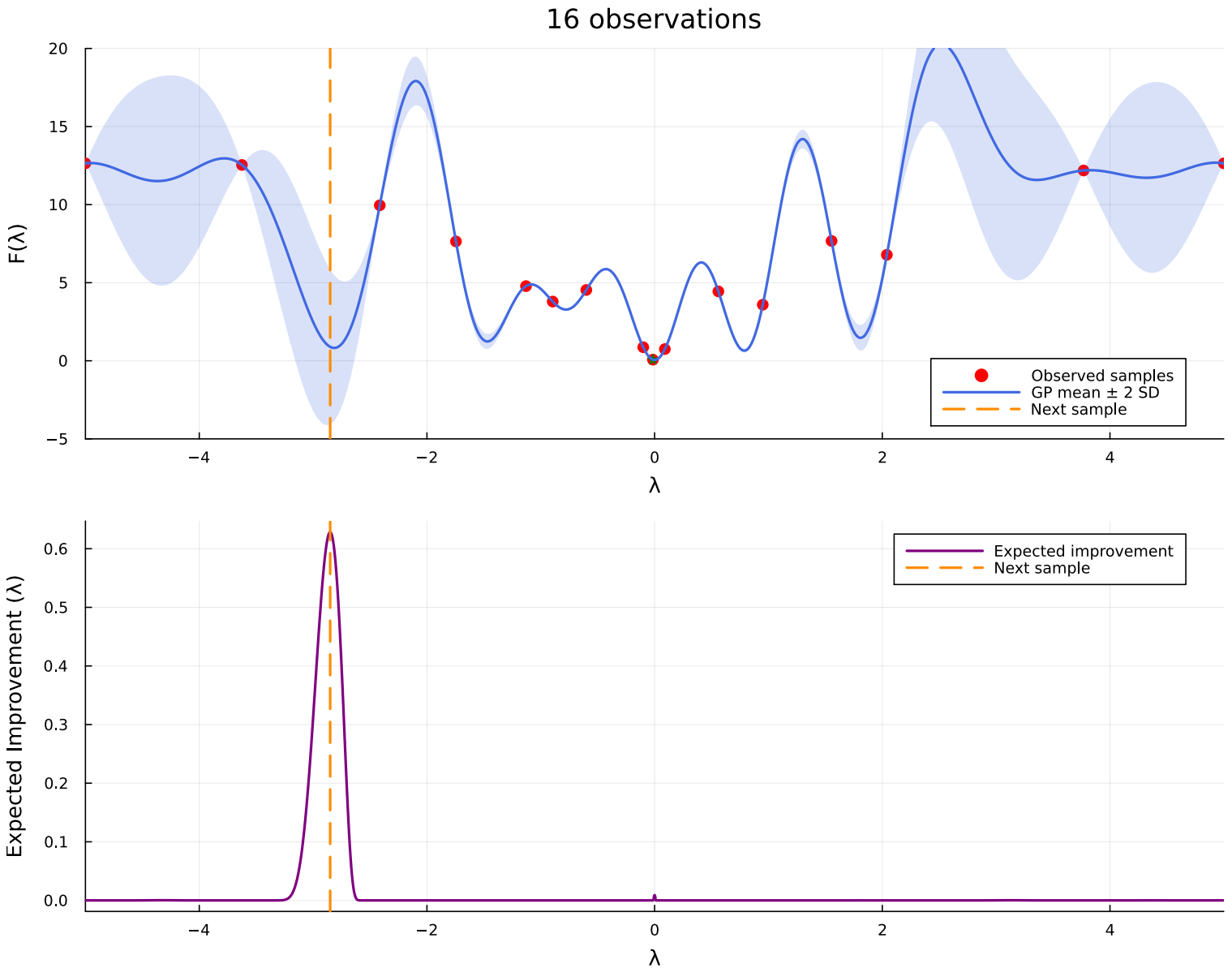
Bayesian optimization: Iteration 12

© 2026 Fessler



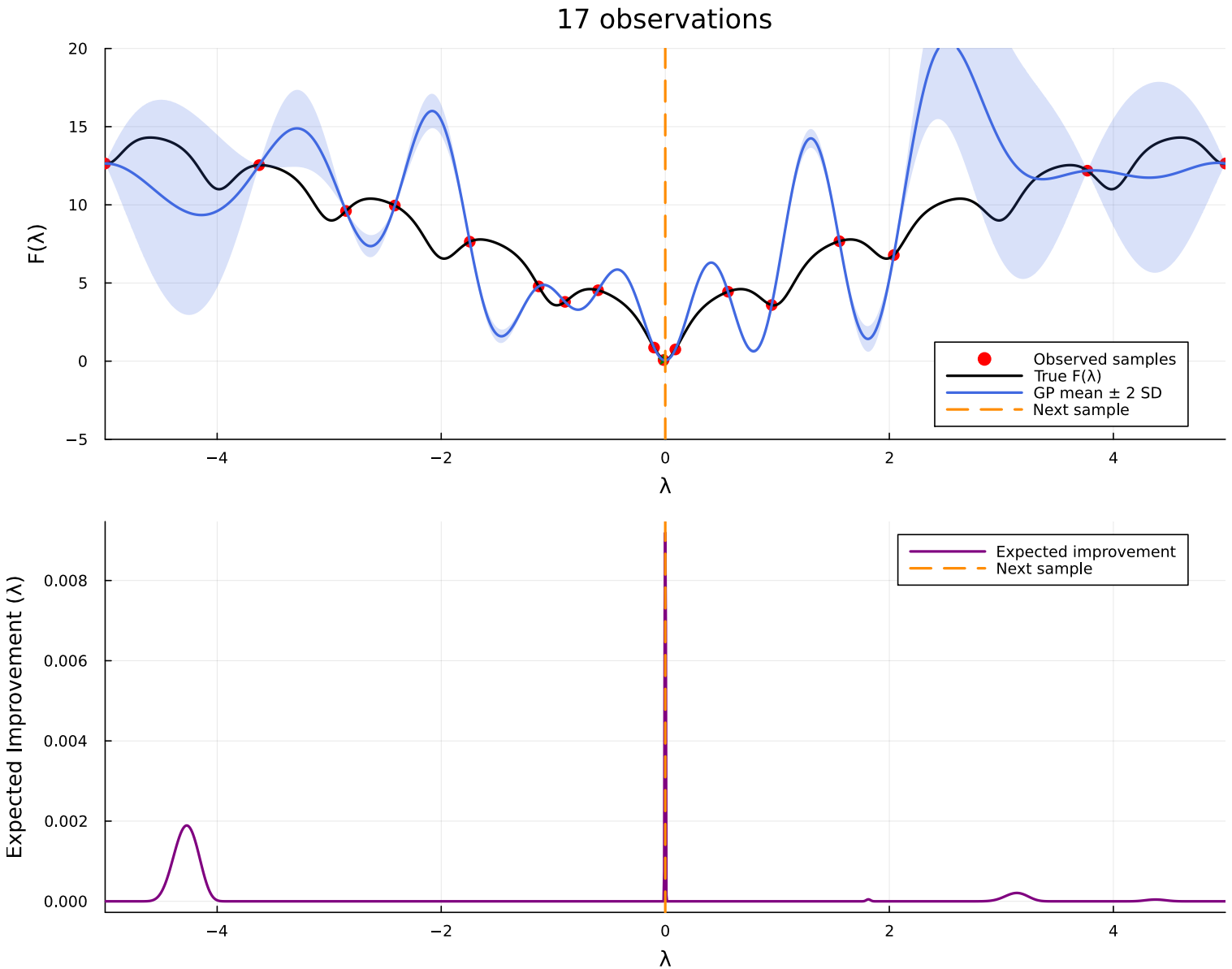
Bayesian optimization: Iteration 13

© 2026 Fessler



Bayesian optimization: Iteration 14

© 2026 Fessler



Bayesian optimization: Summary

© 2026 Fessler

- Samples are allowed to be noisy: $y_i = F(\boldsymbol{\lambda}_i) + \varepsilon_i$
  Think "stochastic" in SGD.
- Example was 1D but approach extends to $\boldsymbol{\lambda}$
- Many details in practice, including kernel parameters.
- GP updated posterior mean and variance every step

Related method: DFO

© 2026 Fessler

- Derivative-free optimization
- Also called zeroth-order optimization
- These methods optimize a function using only evaluations, no derivatives or gradients
- Appropriate for HP optimization
- Appropriate for reinforcement learning

- Suppose F is a black-box function: difficult to optimize
- Idea: Approximate the gradient of F
- Use the definition of derivative
- For a one dimensional function: $F : \mathbb{R} \mapsto \mathbb{R}$

$$\lim_{\delta \to 0} \frac{F(x+\delta) - F(x)}{\delta} = F'(x)$$

- Think of δ as a small perturbation
- Here we do not consider noise in evaluating F
- This works for 1 dimensional functions. What about high dim?
- Recall each entry of $\boldsymbol{\lambda}$ is a HP choice:
  - λ_1 = Learning rate
  - λ_2 = Dropout
  - λ_3 = CNN filter width
  - ...

- Suppose $F : \mathbb{R}^q \mapsto \mathbb{R}$
- Each input variable can be a different HP choice
- Let us approximate ∇F
- Consider multivariate normal perturbation $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$
- It can be shown that

$$\mathbb{E}\left\{ \lim_{\delta \to 0} \frac{F(\boldsymbol{\lambda} + \delta \boldsymbol{g}) - F(\boldsymbol{\lambda})}{\delta} \right\} =$$

where expectation is over $\mathrm{p}(\boldsymbol{g})$.

- Theory:

$$\mathbb{E}\left\{ \lim_{\delta \to 0} \frac{F(\boldsymbol{\lambda} + \delta \boldsymbol{g}) - F(\boldsymbol{\lambda})}{\delta} \boldsymbol{g} \right\} = \nabla F(\boldsymbol{\lambda}).$$

- In practice: Approximate expectation via $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n$:

$$\frac{1}{n} \sum_{i=1}^{n} \frac{F(\boldsymbol{\lambda} + \delta \boldsymbol{g}_i) - F(\boldsymbol{\lambda})}{\delta} \boldsymbol{g}_i \approx \nabla F(\boldsymbol{\lambda}),$$

  if n is large enough and δ is small enough
- Intuition: Each $\boldsymbol{g}_i$ explores a potentially promising direction
- Gradient is found (approximately) by combining these explorations
- Of course if function evaluation F is expensive...

Given a hyperparameter choice $\boldsymbol{\lambda}$, we can evaluate the HO risk which is the error of the model with those hyperparameters: $\hat{R}_{\text{HO}}(\hat{f}_{\boldsymbol{\lambda}})$

- This is difficult to evaluate: you need to train a neural net to calculate it!
- you do not have access to the gradient of this function $\nabla_{\boldsymbol{\lambda}} \hat{R}_{\text{HO}}(\hat{f}_{\boldsymbol{\lambda}})$

These types of problems are called **black box optimization**

Black box optimization: Other examples

- Simulating complex systems
  - You have a simulator that computes complex physical dynamics, and you treat that as a black box
  - Goal is to, *e.g.*, predict the fluid flow with a smaller ("reduced order") model
- Reinforcement learning
  - You typically do not know the function mapping a policy (action choices) to an expected reward
  - You must run episodes ("rollouts") and observe rewards to update your policy

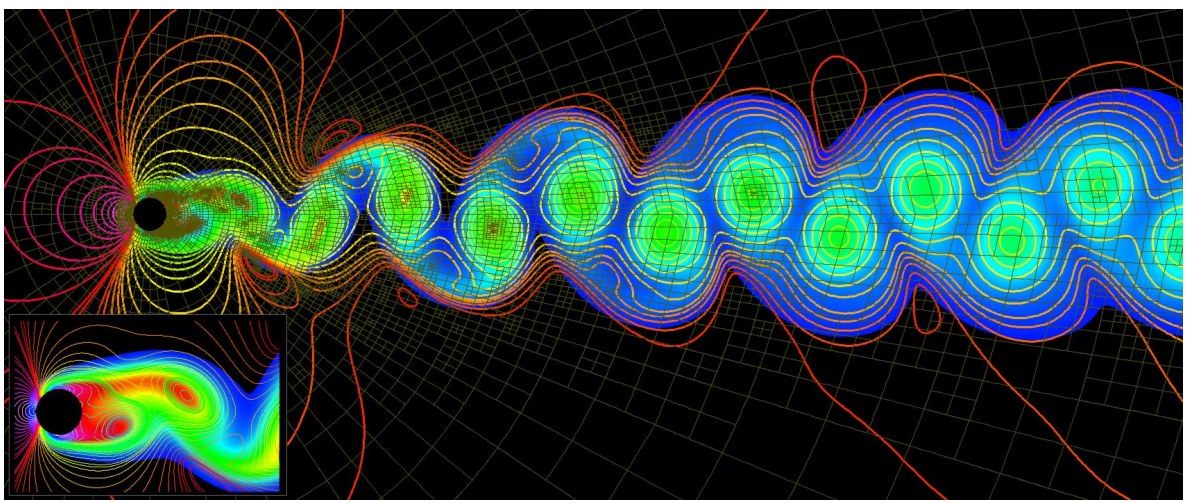


Figure courtesy https://docanco.com/what-is-fluid-dynamics.

CQ: Validation

© 2026 Fessler

Q17.2 Validation data is used to (?)

A: train our model parameters.

B: test our model parameters and report how accurate our model is on a task.

C: train our model with a given hyperparameter setting.

D: test our model hyperparameters for tuning.

E: none of these

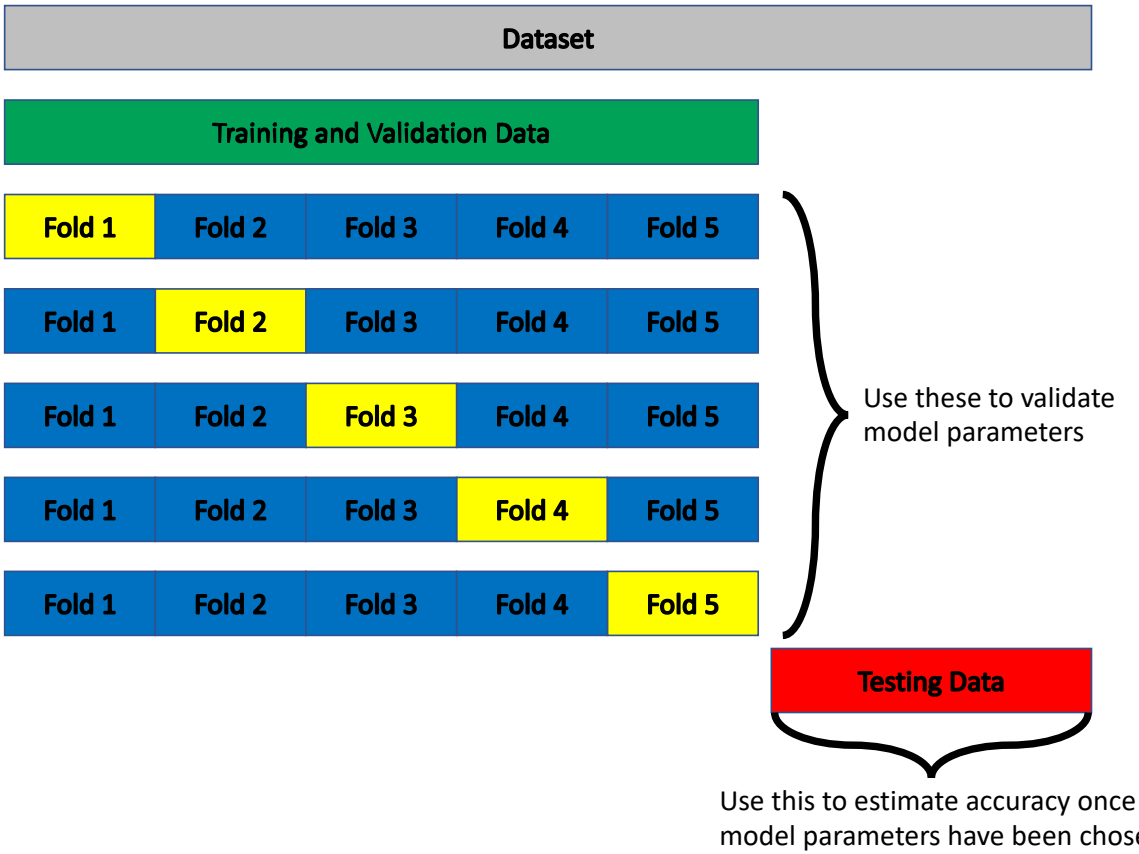
Cross validation

© 2026 Fessler

- If data are relatively scarce, splitting into training, validation, and test can mean there are too few samples
- **Cross-validation** splits the training data into
- We then train and validate on every subset and aggregate the risk estimates.

K-fold cross validation

© 2026 Fessler



We use each of the K training sets for model fitting and the corresponding validation set for model assessment. Here $K = 5$. $K = n$ is cross validation (expensive for large n).

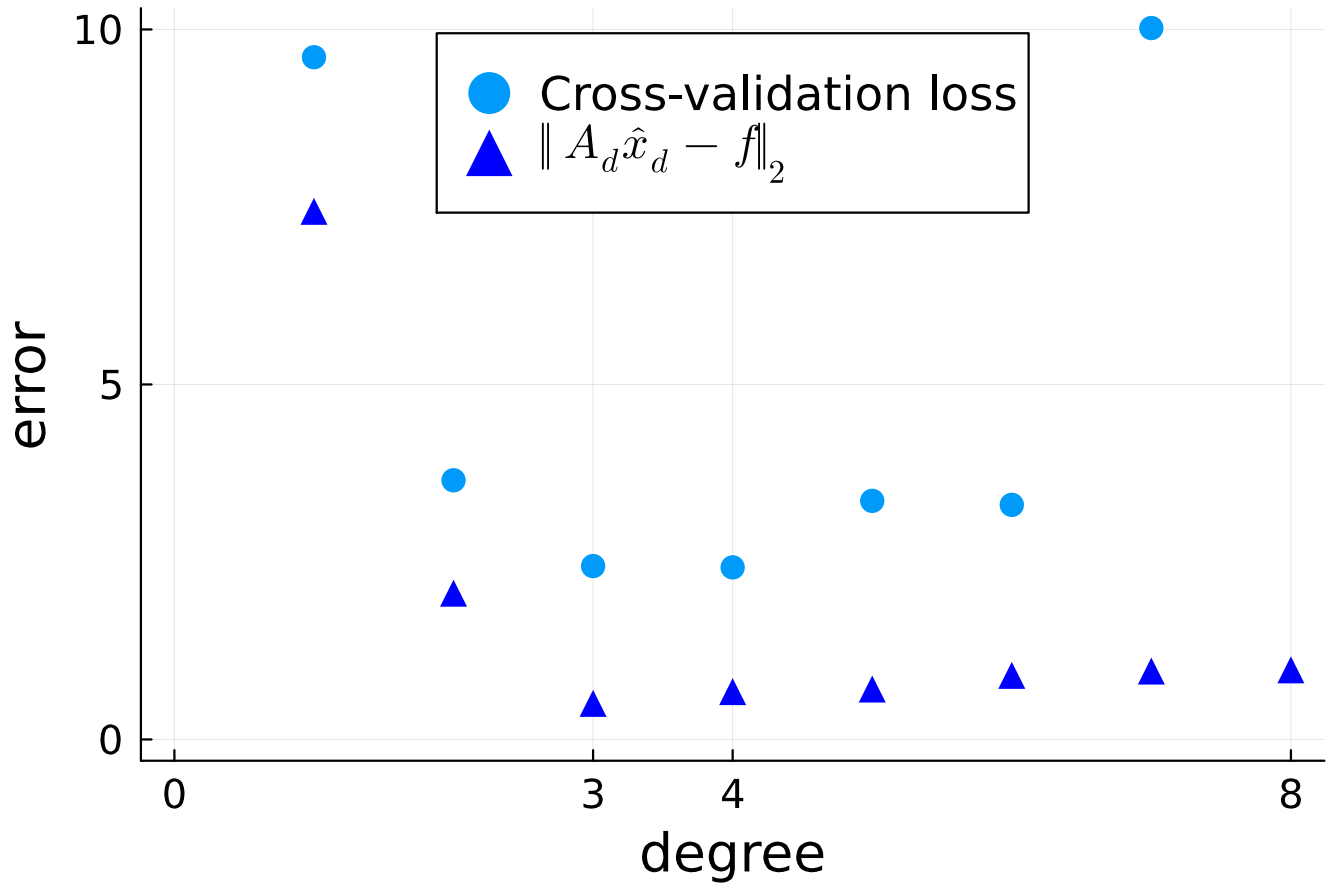
Figure courtesy [4]

K-fold cross validation

© 2026 Fessler

- Theoretical guarantees of this approach assume i.i.d. data drawn from a single population.
- For data from timeseries and/or with potentially other confounding factors (coming from multiple experiments, different data collection processes), one should carefully choose folds with samples from the mixture of populations.

Demo with n-fold cross-validation



References I

[1] K. P. Murphy.
Probabilistic machine learning: An introduction.
MIT Press, 2022.

[2] C. Bishop.
Pattern recognition and machine learning.
Springer, 2006.

[3] H. Yu, J. A. Fessler, and Y. Jiang.
Bilevel optimized implicit neural representation for scan-specific accelerated MRI reconstruction.
IEEE Trans. Med. Imag., 45(7):3792–808, July 2026.

[4] scikit learn.
3.1. cross-validation: evaluating estimator performance, 2011.
https://scikit-learn.org/stable/modules/cross_validation.html.