Advanced Methods for Bayesian Optimization in Complex Scenarios Course

Presented by: Eduardo C. Garrido–Merchán.

Assistant Professor at Universidad Pontificia Comillas

Quantitative Methods Department

Former PhD student of Daniel Hernandez Lobato

June, 2024.

Index

- Introduction and Motivation.

Index

- Introduction and Motivation.
- Fundamentals of Bayesian Optimization.

Index

- Introduction and Motivation.
- Fundamentals of Bayesian Optimization.
- Hands on Bayesian optimization with BOTorch.

Index

- Introduction and Motivation.
- Fundamentals of Bayesian Optimization.
- Hands on Bayesian optimization with BOTorch.
- Parallel Constrained Multi-objective Bayesian optimization.

Index

- Introduction and Motivation.
- Fundamentals of Bayesian Optimization.
- Hands on Bayesian optimization with BOTorch.
- Parallel Constrained Multi-objective Bayesian optimization.
- Advanced scenarios: Mixed, multi-fidelity, high-dim, topologies.

Index

- Introduction and Motivation.
- Fundamentals of Bayesian Optimization.
- Hands on Bayesian optimization with BOTorch.
- Parallel Constrained Multi-objective Bayesian optimization.
- Advanced scenarios: Mixed, multi-fidelity, high-dim, topologies.
- Advanced hands on Bayesian optimization with BOTorch.

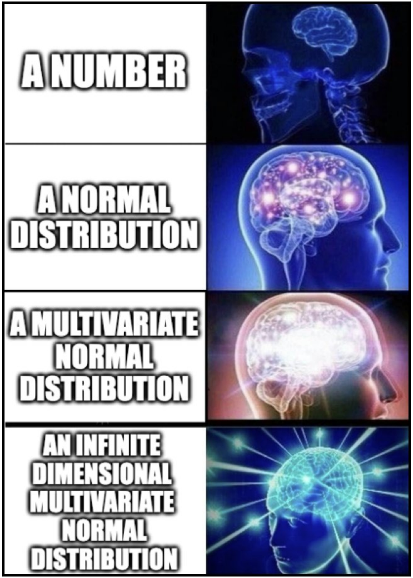
Index

- Introduction and Motivation.
- Fundamentals of Bayesian Optimization.
- Hands on Bayesian optimization with BOTorch.
- Parallel Constrained Multi-objective Bayesian optimization.
- Advanced scenarios: Mixed, multi-fidelity, high-dim, topologies.
- Advanced hands on Bayesian optimization with BOTorch.
- References.

First Session: Bayesian optimization fundamentals

Introduction.

GPs are great... so now... how can they be used for optimization?



Challenges in Machine Learning, Finance, Robotics, Engineering, Business...

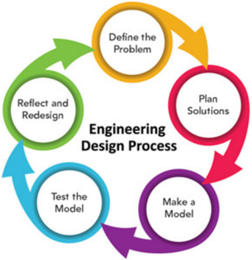
Companies face complex scenarios dealing with lots of scenarios!

Challenges in Machine Learning, Finance, Robotics, Engineering, Business...

Companies face complex scenarios dealing with lots of scenarios!

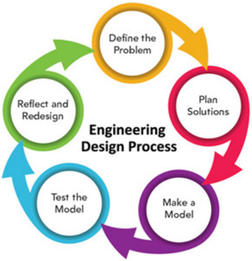


Challenges in Machine Learning, Finance, Robotics, Engineering, Business...

Companies face complex scenarios dealing with lots of scenarios!



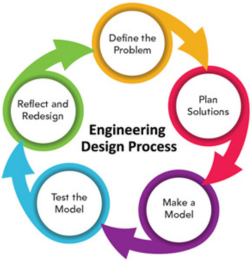



- Many choices at each step.

Challenges in Machine Learning, Finance, Robotics, Engineering, Business...

Companies face complex scenarios dealing with lots of scenarios!

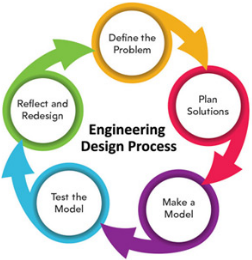


- Many choices at each step.
- Complicated and high dimensional.

Challenges in Machine Learning, Finance, Robotics, Engineering, Business...

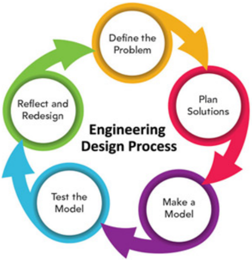
Companies face complex scenarios dealing with lots of scenarios!



- Many choices at each step.
- Complicated and high dimensional.
- Difficult for individuals to reason about.

Challenges in Machine Learning, Finance, Robotics, Engineering, Business...

Companies face complex scenarios dealing with lots of scenarios!

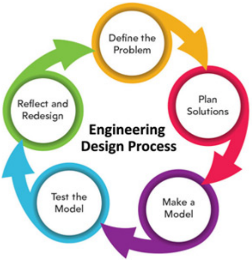


- Many choices at each step.
- Complicated and high dimensional.
- Difficult for individuals to reason about.
- Prone to human bias.

Challenges in Machine Learning, Finance, Robotics, Engineering, Business...

Companies face complex scenarios dealing with lots of scenarios!

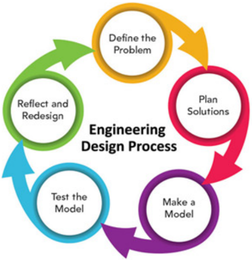


- Many choices at each step.
- Complicated and high dimensional.
- Difficult for individuals to reason about.
- Prone to human bias.
- It may be possible to test various models in parallel.

Challenges in Machine Learning, Finance, Robotics, Engineering, Business...

Companies face complex scenarios dealing with lots of scenarios!

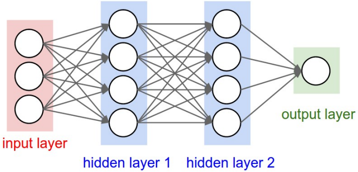


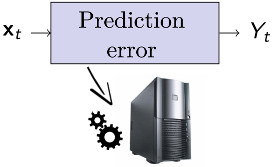



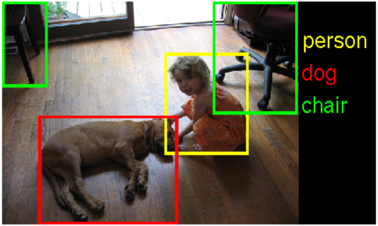
- Many choices at each step.
- Complicated and high dimensional.
- Difficult for individuals to reason about.
- Prone to human bias.
- It may be possible to test various models in parallel.

Optimization is a challenging task in real-life choices!

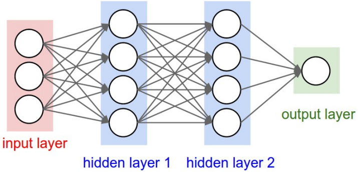
Example: **Deep Neural Network** for object recognition.

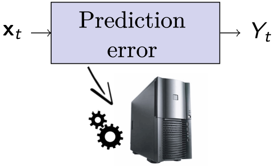


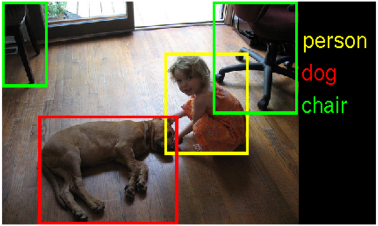




Example: **Deep Neural Network** for object recognition.

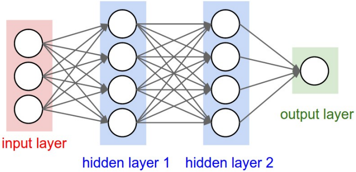


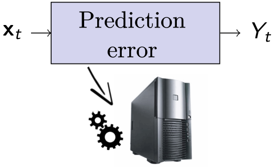


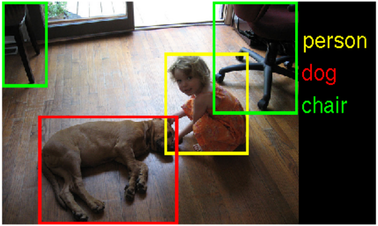


Parameters to tune: Number of neurons, number of layers, learning-rate, level of regularization, momentum, etc.

Example: **Deep Neural Network** for object recognition.

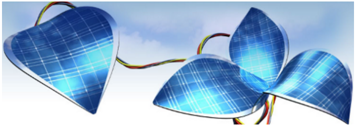


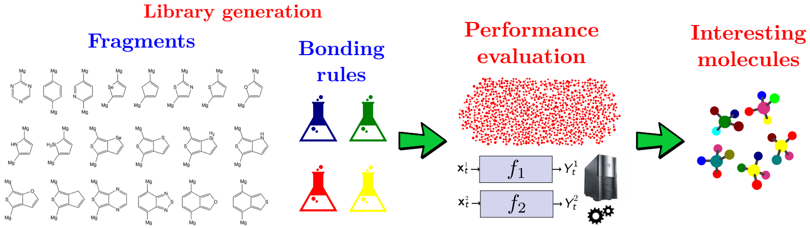




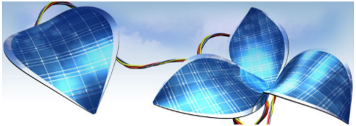
Parameters to tune: Number of neurons, number of layers, learning-rate, level of regularization, momentum, etc. If **multiple processors** were available, we could test **various configurations in parallel**, in order to **gain more information**.

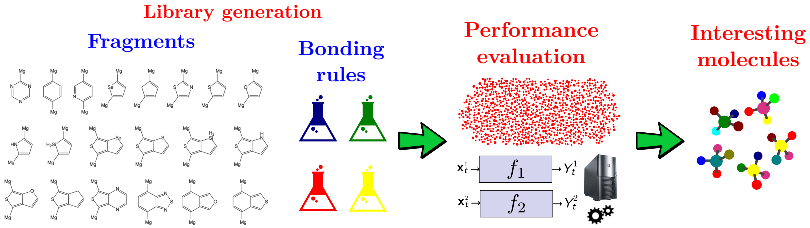
Example: new **plastic solar cells** for transforming light into electricity.





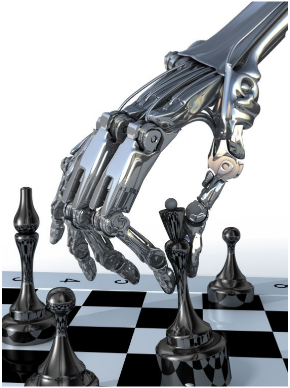
Example: new **plastic solar cells** for transforming light into electricity.



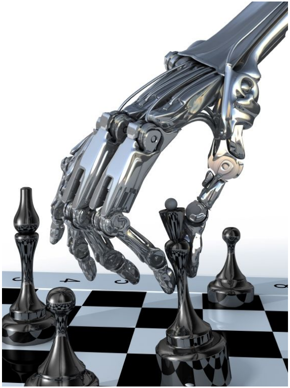


Explore **millions of candidate molecule structures** to identify the compounds with the best properties.

Example: **control system** for a robot that is able to grasp objects.



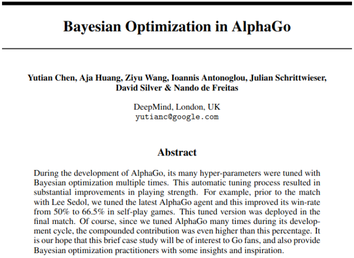
Example: **control system** for a robot that is able to grasp objects.



Finger Joint Trajectories

Parameters to tune: initial pose for the robot's hand and finger joint trajectories.

Bayesian optimization, the secret sauce of AlphaGo (and Alphazero)



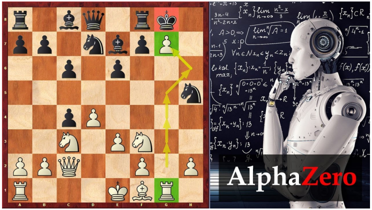
Bayesian Optimization in AlphaGo

Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver & Nando de Freitas

DeepMind, London, UK
yutianc@google.com

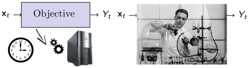
Abstract

During the development of AlphaGo, its many hyper-parameters were tuned with Bayesian optimization multiple times. This automatic tuning process resulted in substantial improvements in playing strength. For example, prior to the match with Lee Sedol, we tuned the latest AlphaGo agent and this improved its win-rate from 50% to 66.5% in self-play games. This tuned version was deployed in the final match. Of course, since we tuned AlphaGo many times during its development cycle, the compounded contribution was even higher than this percentage. It is our hope that this brief case study will be of interest to Go fans, and also provide Bayesian optimization practitioners with some insights and inspiration.



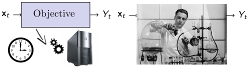
Optimization Problems: Common Features

- Very expensive evaluations.

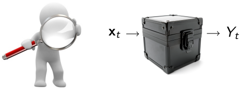


Optimization Problems: Common Features

- Very expensive evaluations.

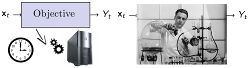


- The objective is a black-box.

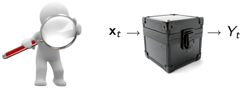


Optimization Problems: Common Features

- Very expensive evaluations.

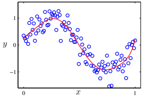


- The objective is a black-box.



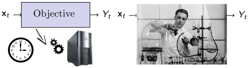
- The evaluation can be noisy.

$y = f(\mathbf{x}) + \epsilon$

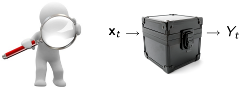


Optimization Problems: Common Features

- Very expensive evaluations.

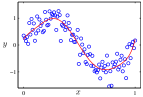


- The objective is a black-box.

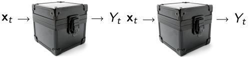


- The evaluation can be noisy.

$y = f(\mathbf{x}) + \epsilon$



- Evaluations may be done in parallel.



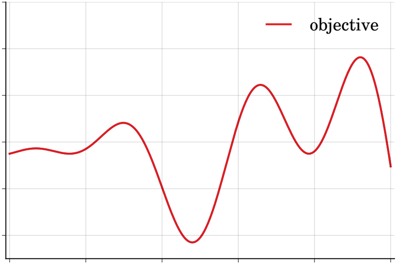
Bayesian optimization methods can be used to solve these problems!

Activity

Think about a real problem that can be solved with Bayesian optimization. 15 mins individual. 10 mins in groups. 3 presentations with slides.

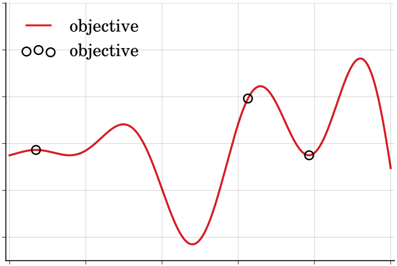
- Which is the function to optimize?
- Does it satisfy the BO assumptions?
- Which are the independent variables that are taken into account?
- Which are the lower and upper bounds in the optimization?
- Is f(X) a sample of a Gaussian process?
- What is the prior mean of the Gaussian process?
- Which kernel would you use?

Bayesian Optimization



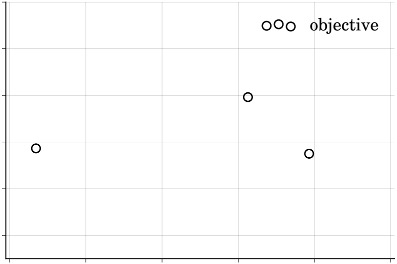
1. Get initial sample.

Bayesian Optimization



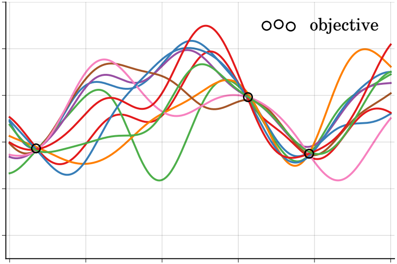
1. Get initial sample.

Bayesian Optimization



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$

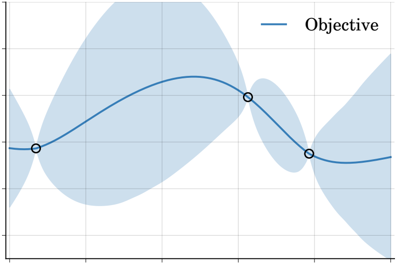
Bayesian Optimization



1. Get initial sample.
2. Fit a model to the data:

$$p(y|\mathrm{x}, \mathcal{D}_n) \,.$$

Bayesian Optimization



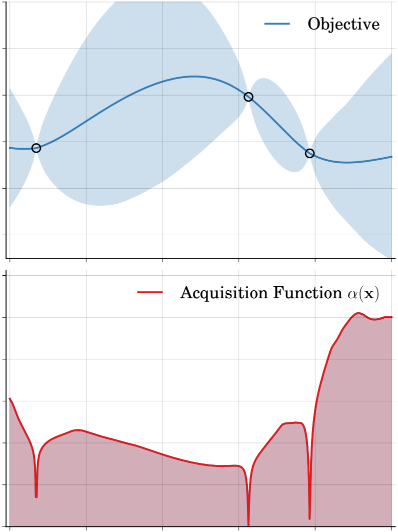
1. Get initial sample.
2. Fit a model to the data:

$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$

3. Select data collection strategy:

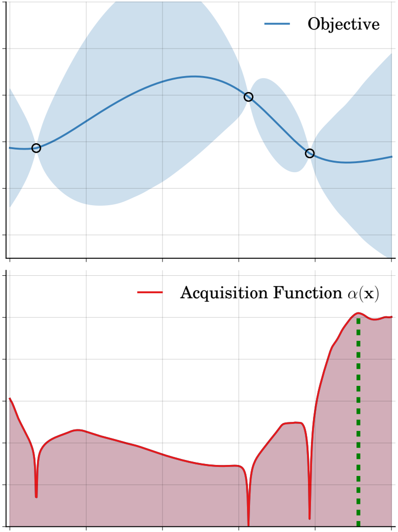
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$

Bayesian Optimization



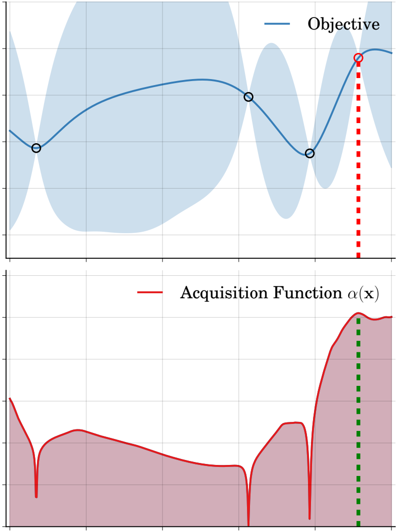
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.

Bayesian Optimization



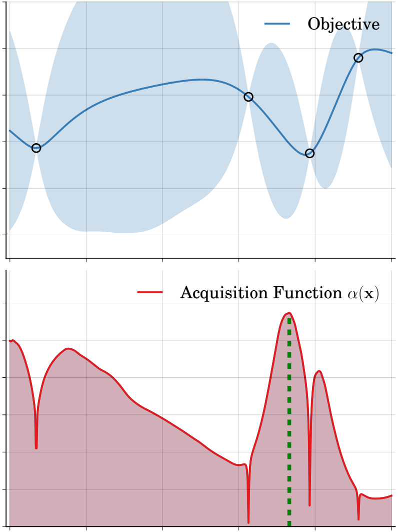
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x}, \mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.

Bayesian Optimization



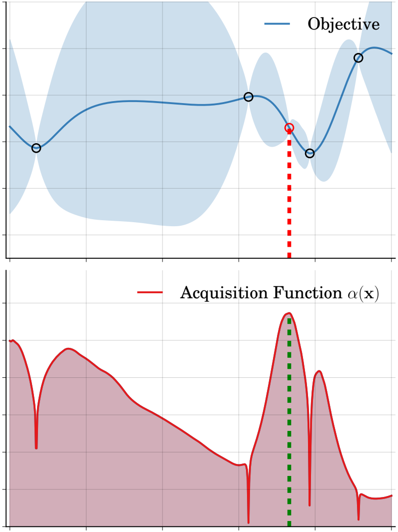
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n) .$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)] .$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



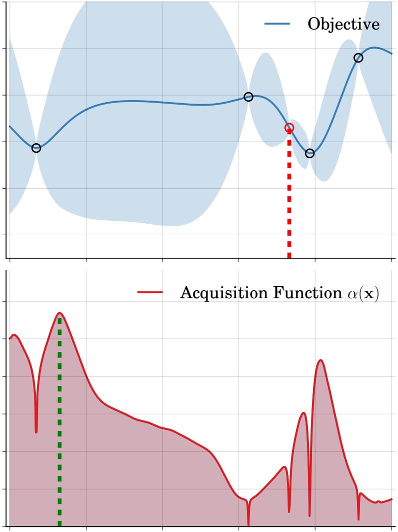
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



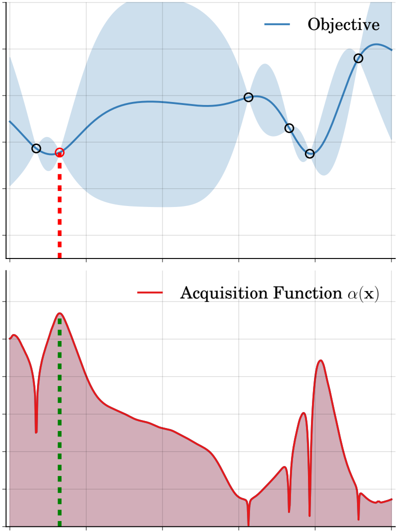
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



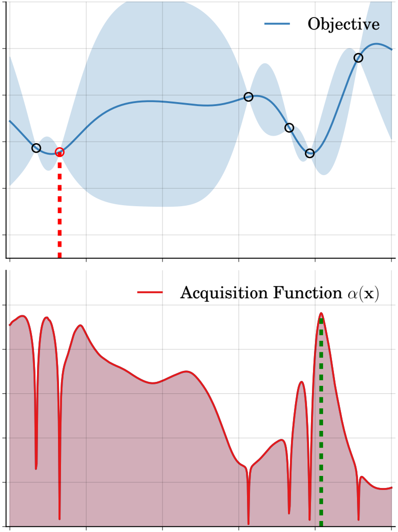
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



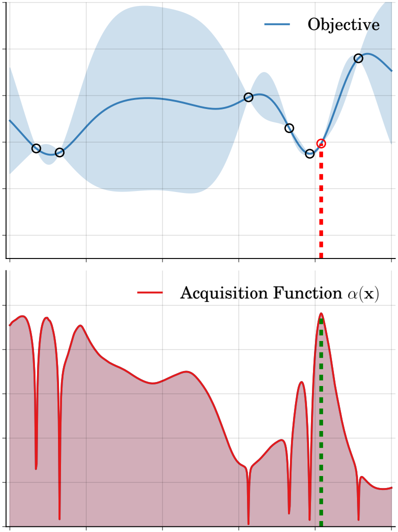
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



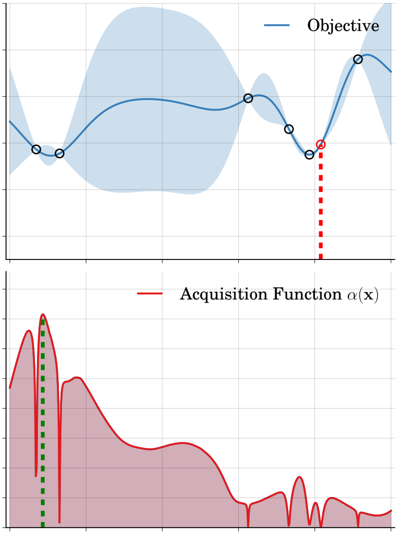
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



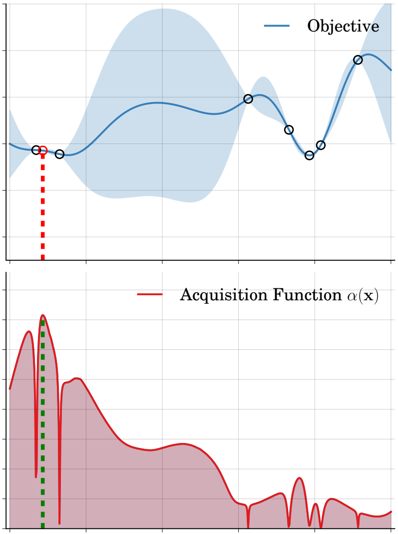
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



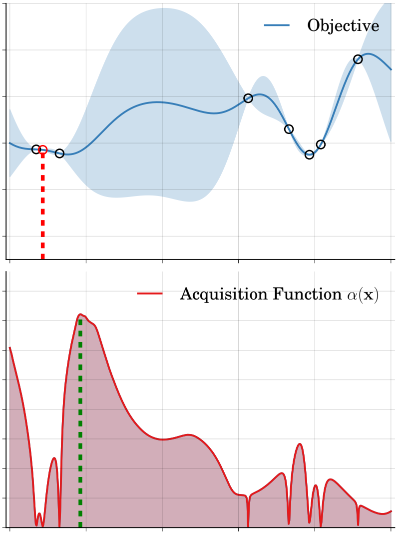
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



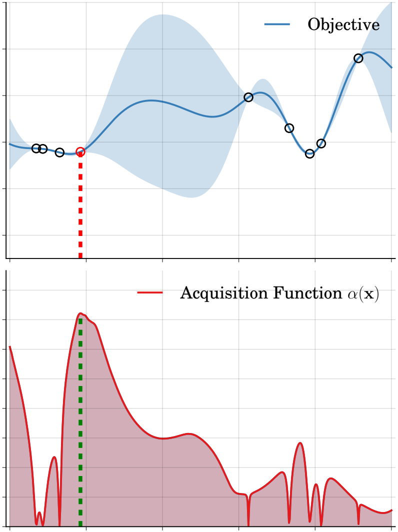
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



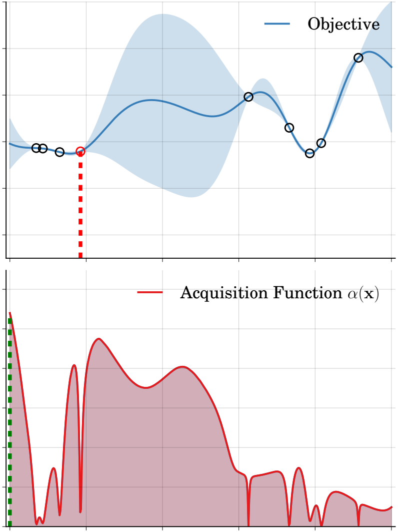
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



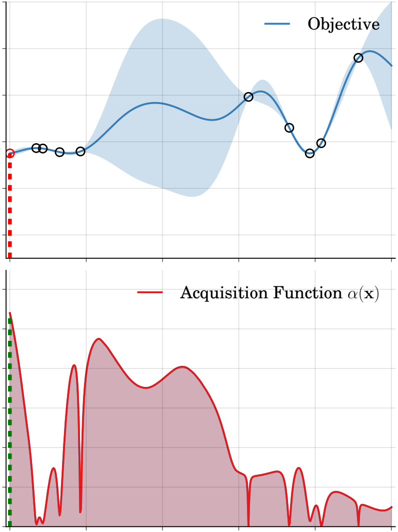
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



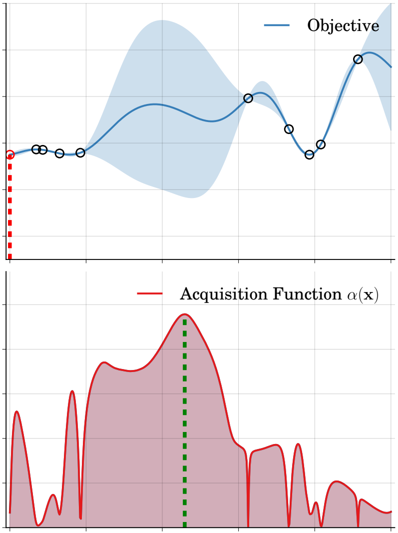
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



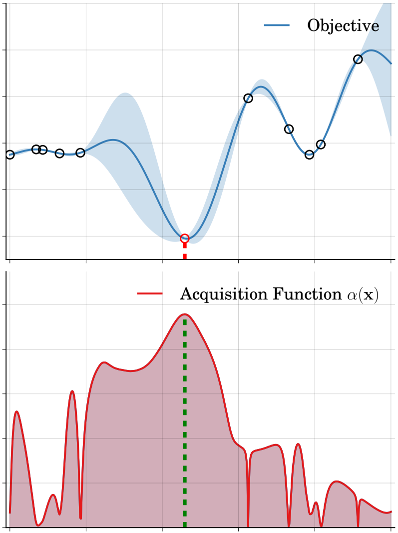
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization



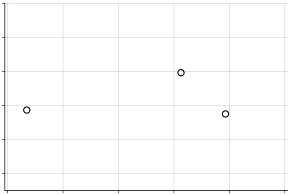
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Bayesian Optimization

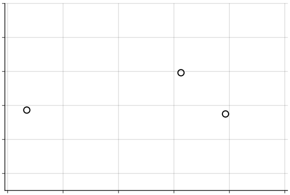


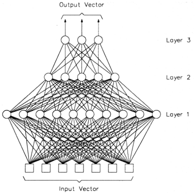
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathrm{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathrm{x}) = \mathrm{E}_{p(y|\mathrm{x},\mathcal{D}_n)}[U(y|\mathrm{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathrm{x})$.
5. Collect data and update model.
6. Repeat!

Fitting a Model to the Data



Fitting a Model to the Data

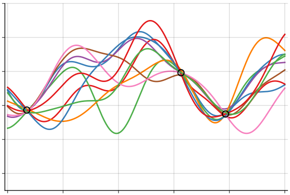


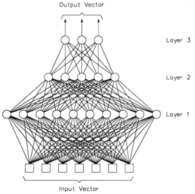


$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

Fitting a Model to the Data

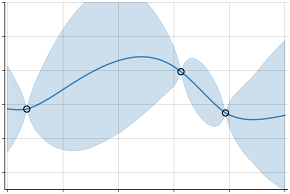


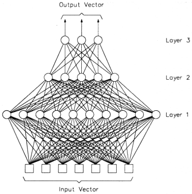


$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

Fitting a Model to the Data

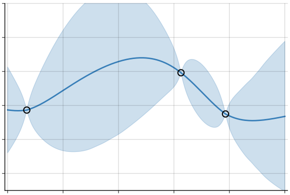


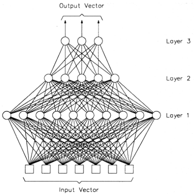


$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

Fitting a Model to the Data





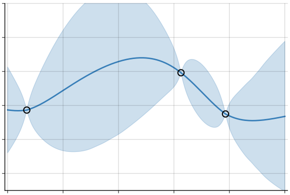
$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

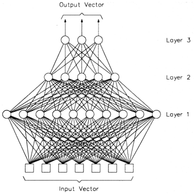
$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

Posterior Dist. $\quad p(\mathbf{W}|\mathrm{Data}) = p(\mathbf{W})p(\mathrm{Data}|\mathbf{W})/p(\mathrm{Data})$

Predictive Dist. $\quad p(y|\mathrm{Data}, x) = \int p(y|\mathbf{W}, x)p(\mathbf{W}|\mathrm{Data})d\mathbf{W}$

Fitting a Model to the Data





$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

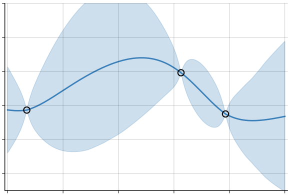
$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

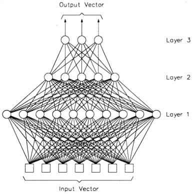
Posterior Dist. $p(\mathbf{W}|\mathrm{Data}) = p(\mathbf{W})p(\mathrm{Data}|\mathbf{W})/p(\mathrm{Data})$

Predictive Dist. $p(y|\mathrm{Data}, x) = \int p(y|\mathbf{W}, x)p(\mathbf{W}|\mathrm{Data})d\mathbf{W}$

Challenges: The model should be non-parametric (the world is complicated) and computing $p(\mathrm{Data})$ is intractable!

Fitting a Model to the Data





$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

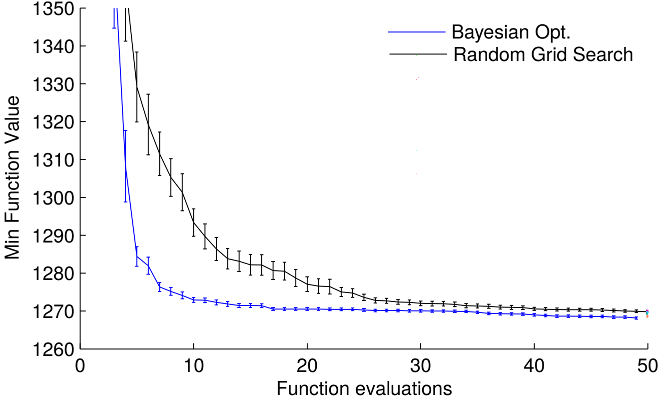
Posterior Dist. $p(\mathbf{W}|\text{Data}) = p(\mathbf{W})p(\text{Data}|\mathbf{W})/p(\text{Data})$

Predictive Dist. $p(y|\text{Data}, x) = \int p(y|\mathbf{W}, x)p(\mathbf{W}|\text{Data})d\mathbf{W}$

Challenges: The model should be non-parametric (the world is complicated) and computing $p(\text{Data})$ is intractable!

Solved by setting $p(\mathrm{W}) = \prod_{ij} \mathcal{N}(w_{ji}|0, \sigma^2 H^{-1})$ and letting $H \to \infty$!

Bayesian Optimization vs. Uniform Exploration



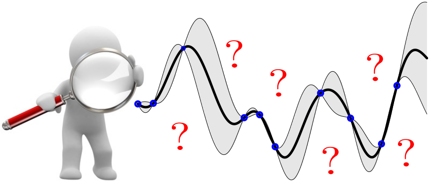
Tuning LDA on a collection of Wikepida articles (Snoek *et al.*, 2012).

Using the GP Uncertainty in Optimization

Where to evaluate **next**?

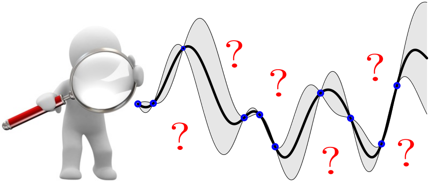
Using the GP Uncertainty in Optimization

Where to evaluate **next**?



Using the GP Uncertainty in Optimization

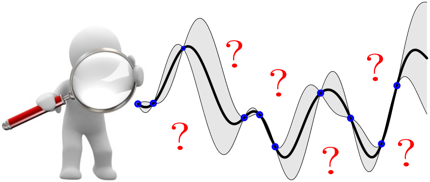
Where to evaluate **next**?



- **Exploration:** seek places with high variance.

Using the GP Uncertainty in Optimization

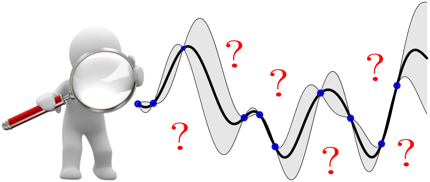
Where to evaluate **next**?



- **Exploration:** seek places with high variance.
- **Exploitation:** seek places with low mean.

Using the GP Uncertainty in Optimization

Where to evaluate **next**?

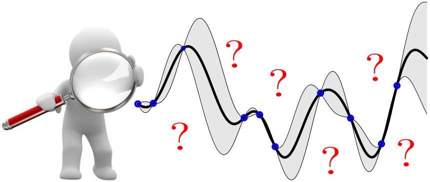


- **Exploration:** seek places with high variance.
- **Exploitation:** seek places with low mean.

The acquisition function balances these two, to choose in an intelligent way the next evaluation point!

Using the GP Uncertainty in Optimization

Where to evaluate **next**?



- **Exploration:** seek places with high variance.
- **Exploitation:** seek places with low mean.

The acquisition function balances these two, to choose in an intelligent way the next evaluation point!

$$\alpha(\mathrm{x}) = \mathbb{E}_{p(y^\star|\mathcal{D}_N,\mathrm{x})}\left[U(y^\star|\mathrm{x},\mathcal{D}_N)\right]$$

Some Acquisition Functions

Let $\nu = \min\{y_1, \ldots, y_N\}$ and $\gamma(\mathsf{x}) = \frac{\nu - \mu(\mathsf{x})}{\sigma(\mathsf{x})}$.

Some Acquisition Functions

Let $\nu = \min\{y_1, \ldots, y_N\}$ and $\gamma(\mathsf{x}) = \frac{\nu - \mu(\mathsf{x})}{\sigma(\mathsf{x})}$.

- **Probability of Improvement**:

$$U(y^\star | \mathcal{D}_N, \mathsf{x}) = \mathbb{I}(y_\star < \nu)\,, \qquad \alpha(\mathsf{x}) = \Phi\left(\gamma(\mathsf{x})\right)$$

Some Acquisition Functions

Let $\nu = \min\{y_1, \ldots, y_N\}$ and $\gamma(\mathrm{x}) = \frac{\nu - \mu(\mathrm{x})}{\sigma(\mathrm{x})}$.

- **Probability of Improvement**:

$$U(y^\star | \mathcal{D}_N, \mathrm{x}) = \mathbb{I}(y_\star < \nu)\,, \qquad \alpha(\mathrm{x}) = \Phi\left(\gamma(\mathrm{x})\right)$$

- **Expected Improvement**:

$$U(y^\star | \mathcal{D}_N, \mathrm{x}) = \max(0, \nu - y^\star)\,, \alpha(\mathrm{x}) = \sigma(\mathrm{x})\left(\gamma(\mathrm{x})\Phi\left(\gamma(\mathrm{x})\right) + \phi(\gamma(\mathrm{x}))\right)$$

Some Acquisition Functions

Let $\nu = \min\{y_1, \ldots, y_N\}$ and $\gamma(\mathrm{x}) = \frac{\nu - \mu(\mathrm{x})}{\sigma(\mathrm{x})}$.

- **Probability of Improvement**:

$$U(y^\star|\mathcal{D}_N, \mathrm{x}) = \mathbb{I}(y_\star < \nu), \qquad \alpha(\mathrm{x}) = \Phi\left(\gamma(\mathrm{x})\right)$$

- **Expected Improvement**:

$$U(y^\star|\mathcal{D}_N, \mathrm{x}) = \max(0, \nu - y^\star), \alpha(\mathrm{x}) = \sigma(\mathrm{x})\left(\gamma(\mathrm{x})\Phi\left(\gamma(\mathrm{x})\right) + \phi(\gamma(\mathrm{x}))\right)$$

- **Lower Confidence Bound**:

$$\alpha(\mathrm{x}) = -\left(\mu(\mathrm{x}) - \kappa\sigma(\mathrm{x})\right)$$

Some Acquisition Functions

Let $\nu = \min\{y_1, \ldots, y_N\}$ and $\gamma(\mathrm{x}) = \frac{\nu - \mu(\mathrm{x})}{\sigma(\mathrm{x})}$.

- **Probability of Improvement**:

$$U(y^\star | \mathcal{D}_N, \mathrm{x}) = \mathbb{I}(y_\star < \nu)\,, \qquad \alpha(\mathrm{x}) = \Phi\left(\gamma(\mathrm{x})\right)$$

- **Expected Improvement**:

$$U(y^\star | \mathcal{D}_N, \mathrm{x}) = \max(0, \nu - y^\star)\,, \alpha(\mathrm{x}) = \sigma(\mathrm{x})\left(\gamma(\mathrm{x})\Phi\left(\gamma(\mathrm{x})\right) + \phi(\gamma(\mathrm{x}))\right)$$

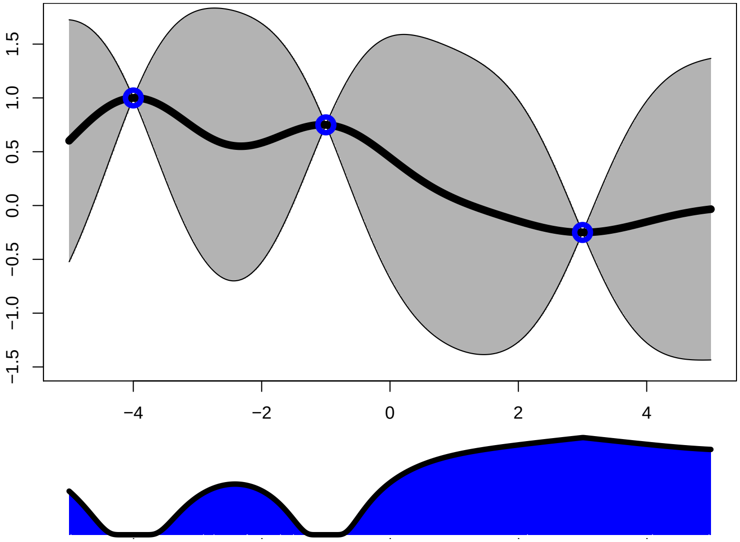
- **Lower Confidence Bound**:

$$\alpha(\mathrm{x}) = -\left(\mu(\mathrm{x}) - \kappa\sigma(\mathrm{x})\right)$$

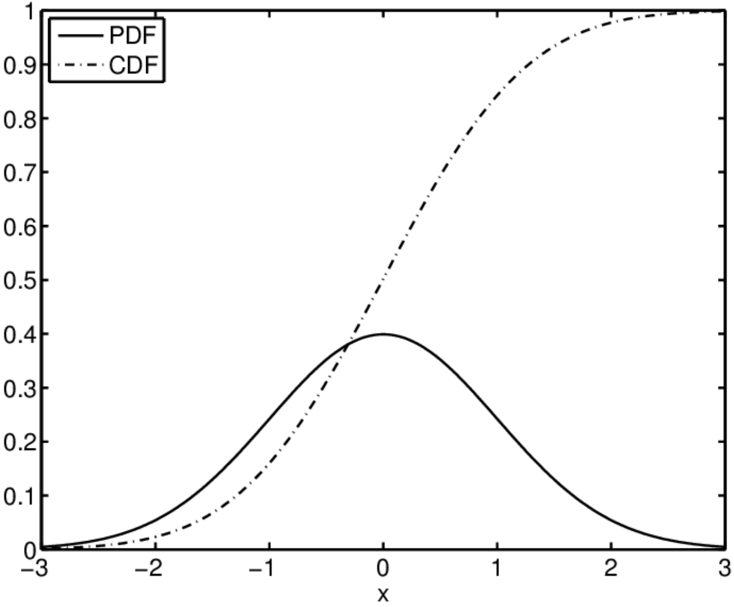
- **Entropy Search**:

$$U(y^\star | \mathcal{D}_N, \mathrm{x}) = \mathrm{H}[p(\mathrm{x}_{\min} | \mathcal{D}_N)] - \mathrm{H}[p(\mathrm{x}_{\min} | \mathcal{D}_N \cup \{\mathrm{x}, y^\star\})]$$

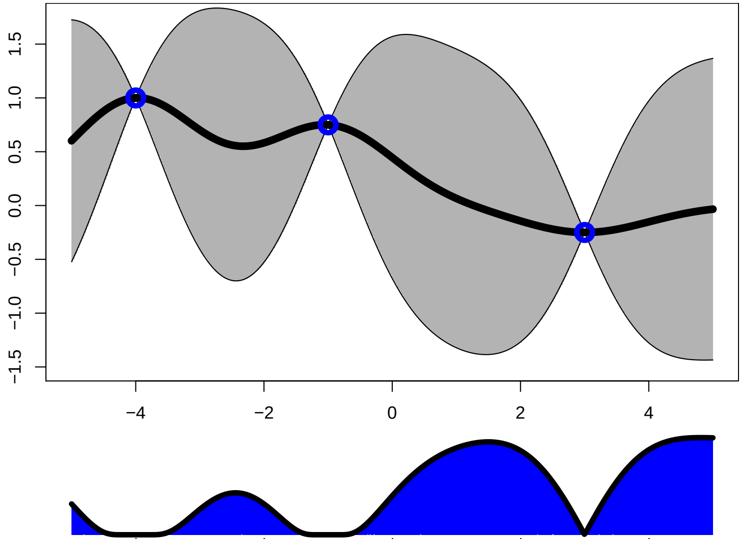
Some Acquisition Functions: Prob. Improvement



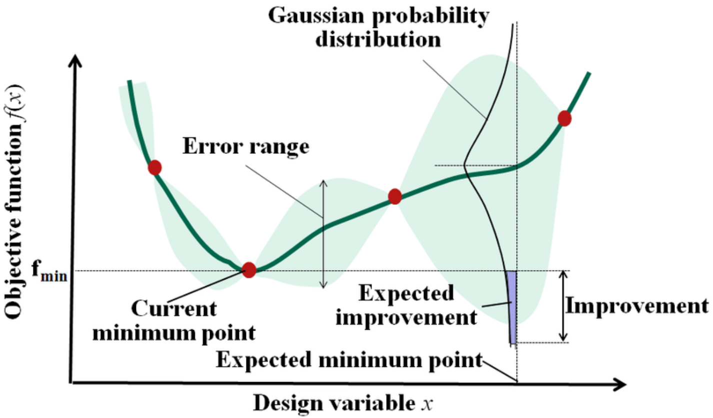
Prob. Improvement: CDF



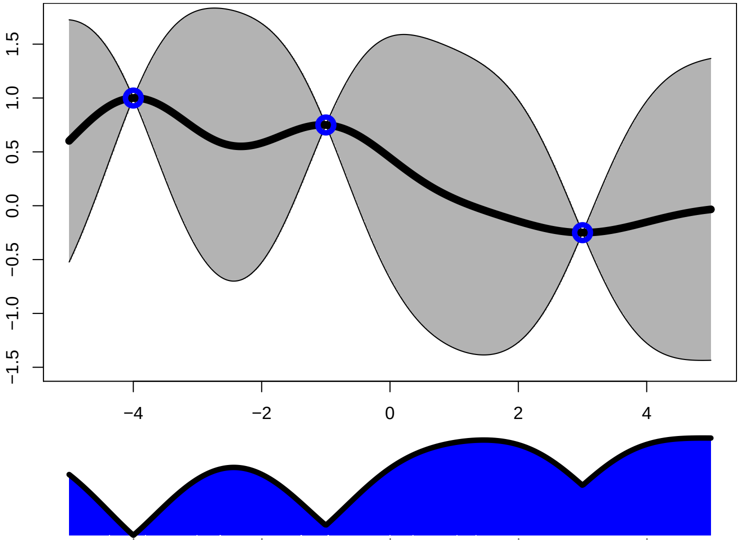
Some Acquisition Functions: Exp. Improvement



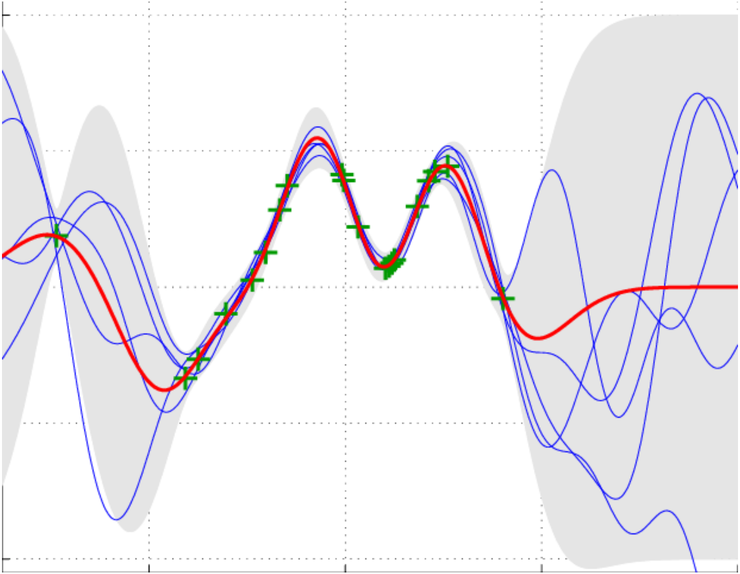
Expected Improvement: intuition



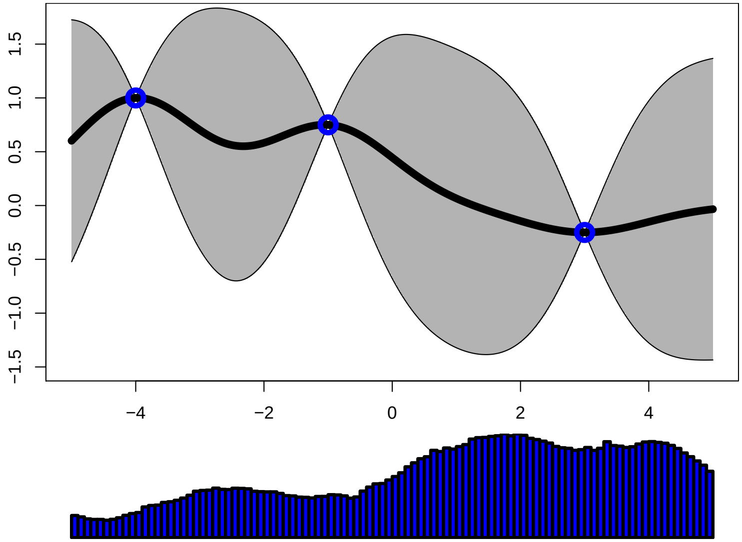
Some Acquisition Functions: Lower Conf. Bound

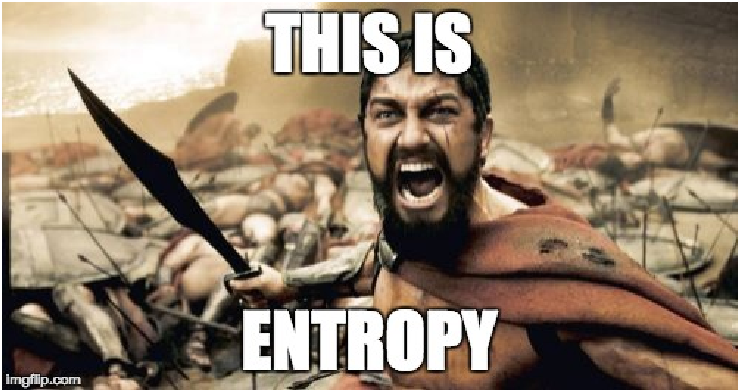


kappa=0. Use only GP samples and optimize to suggest!



Some Acquisition Functions: Entropy Search




THIS IS
ENTROPY
imgflip.com

Information-based Approach

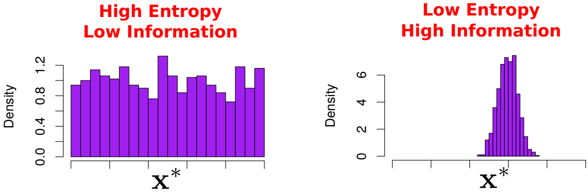
The minimizer, x^*, can be modelled as a **random variable**!

Information is measured by the **entropy** of $p(\mathrm{x}^*|\mathcal{D})$.

Information-based Approach

The minimizer, x^*, can be modelled as a **random variable**!

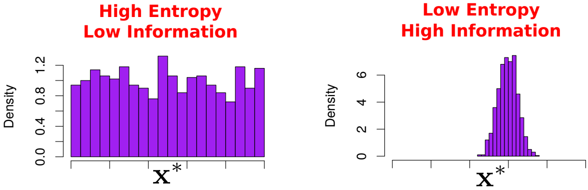
Information is measured by the **entropy** of $p(x^*|\mathcal{D})$.



Information-based Approach

The minimizer, x^*, can be modelled as a **random variable**!

Information is measured by the **entropy** of $p(x^*|\mathcal{D})$.



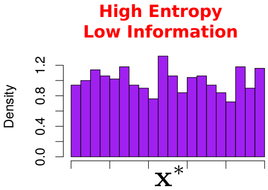
The acquisition function is

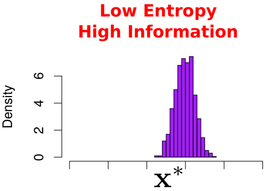
$$\alpha(\mathbf{x}) = \mathsf{H}\left[\mathbf{x}^*\middle|\mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathbf{x}^*\middle|\mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right]\middle|\mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

Information-based Approach

The minimizer, x^*, can be modelled as a **random variable**!

Information is measured by the **entropy** of $p(x^*|\mathcal{D})$.





The acquisition function is

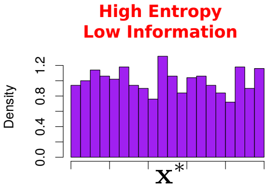
$$\alpha(\mathbf{x}) = \mathsf{H}\left[\mathbf{x}^*\middle|\mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathbf{x}^*\middle|\mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right]\middle|\mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

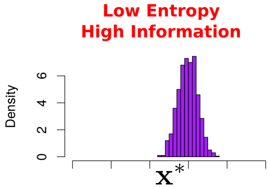
How much we know about $\mathbf{x}^*$ now.

Information-based Approach

The minimizer, x^*, can be modelled as a **random variable**!

Information is measured by the **entropy** of $p(x^*|\mathcal{D})$.





The acquisition function is

$$\alpha(\mathbf{x}) = \mathsf{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

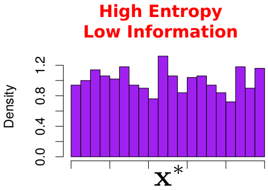
How much we know about $\mathbf{x}^*$ now.

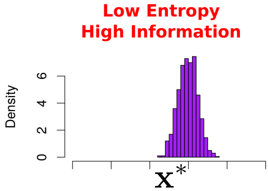
How much we will know about $\mathbf{x}^*$ after collecting y at $\mathbf{x}$.

Information-based Approach

The minimizer, x^*, can be modelled as a **random variable**!

Information is measured by the **entropy** of $p(\mathrm{x}^*|\mathcal{D})$.





The acquisition function is

$$\alpha(\mathbf{x}) = \mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

How much we know about $\mathbf{x}^*$ now.

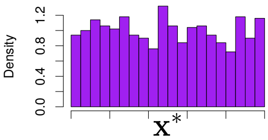
How much we will know about $\mathbf{x}^*$ after collecting y at $\mathbf{x}$.

Information-based Approach

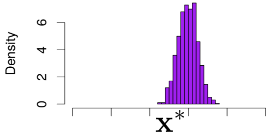
The minimizer, x^*, can be modelled as a **random variable**!

Information is measured by the **entropy** of $p(x^*|\mathcal{D})$.

High Entropy
Low Information



Low Entropy
High Information



The acquisition function is

$$\alpha(\mathbf{x}) = \mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

How much we know about $\mathbf{x}^*$ now.

How much we will know about $\mathbf{x}^*$ after collecting y at $\mathbf{x}$.

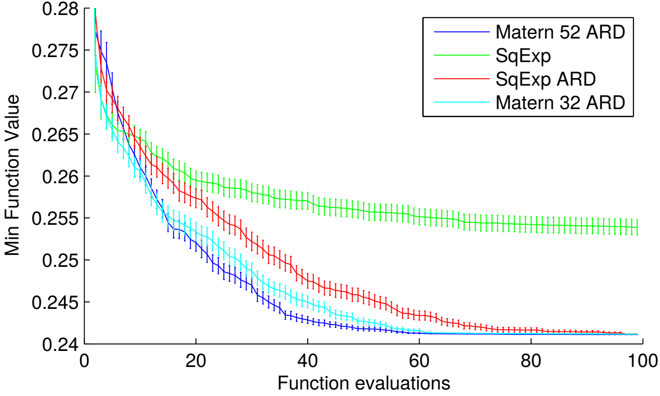
Computing (1) is **very difficult in practice**!

Bayesian Optimization and Model Selection

- **Covariance function selection:** **critical** to achieve good performance. The default choice for regression (squared exponential) is too smooth. Matérn $\nu = 5/2$ kernel works better.

Bayesian Optimization and Model Selection

- **Covariance function selection: critical** to achieve good performance. The default choice for regression (squared exponential) is too smooth. Matérn $\nu = 5/2$ kernel works better.



Structured SVM for protein motif finding (Snoek *et al.*, 2012).

Bayesian Optimization and Model Selection

- **Hyper-parameter selection:** with a small number of observations maximizing $p(\mathrm{y}|\theta)$ can give **too confident** uncertainty estimates.

Bayesian Optimization and Model Selection

- **Hyper-parameter selection:** with a small number of observations maximizing $p(\mathrm{y}|\theta)$ can give **too confident** uncertainty estimates.
- **Sampling the hyper-parameters:** computing $p(\theta|\mathrm{y})$ is **intractable!** Alternative: generate a few samples form $p(\theta|\mathrm{y})$ using MCMC.

Bayesian Optimization and Model Selection

- **Hyper-parameter selection:** with a small number of observations maximizing $p(\mathrm{y}|\theta)$ can give **too confident** uncertainty estimates.

- **Sampling the hyper-parameters:** computing $p(\theta|\mathrm{y})$ is **intractable!** Alternative: generate a few samples form $p(\theta|\mathrm{y})$ using MCMC.

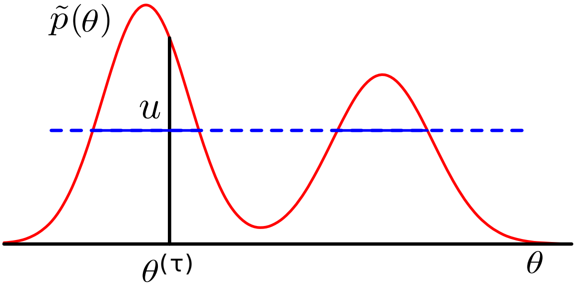
Slice sampling means no additional hyper-parameters!

(Neal, 2003)

Bayesian Optimization and Model Selection

- **Hyper-parameter selection:** with a small number of observations maximizing $p(\mathrm{y}|\theta)$ can give **too confident** uncertainty estimates.
- **Sampling the hyper-parameters:** computing $p(\theta|\mathrm{y})$ is **intractable!** Alternative: generate a few samples form $p(\theta|\mathrm{y})$ using MCMC.

Slice sampling means no additional hyper-parameters!

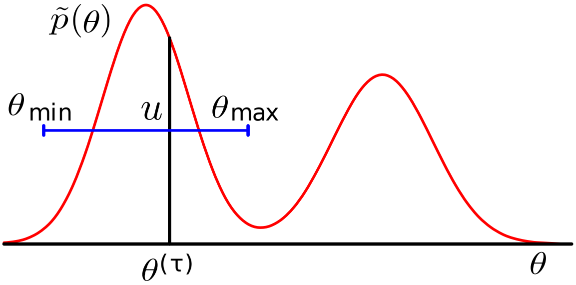


(Neal, 2003)

Bayesian Optimization and Model Selection

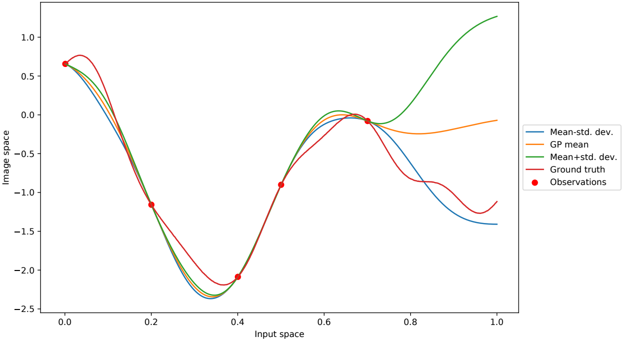
- **Hyper-parameter selection:** with a small number of observations maximizing $p(\mathrm{y}|\theta)$ can give **too confident** uncertainty estimates.
- **Sampling the hyper-parameters:** computing $p(\theta|\mathrm{y})$ is **intractable!** Alternative: generate a few samples form $p(\theta|\mathrm{y})$ using MCMC.

Slice sampling means no additional hyper-parameters!

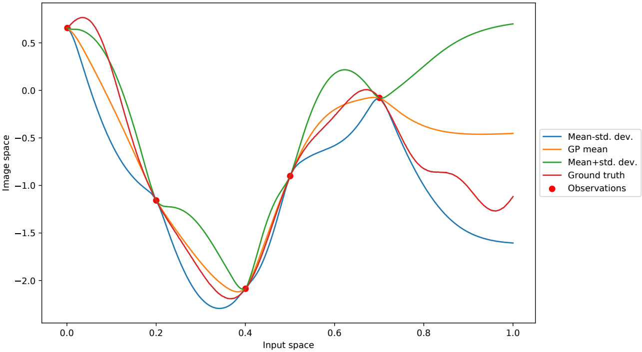


(Neal, 2003)

GP fitting an unknown function using Maximum Likelihood



GP fitting an unknown function sampling hyperparameters



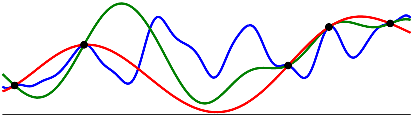
Integrated Acquisition Function

$$\hat{\alpha}(\mathrm{x}) = \int \alpha(\mathrm{x};\theta) p(\theta|\mathrm{y}) d\theta \approx \frac{1}{K}\sum_{k=1}^{K} \alpha(\mathrm{x};\theta^{(k)}) \quad \theta^{(k)} \sim p(\theta|\mathrm{y}),$$

Integrated Acquisition Function

$$\hat{\alpha}(\mathrm{x}) = \int \alpha(\mathrm{x};\theta)p(\theta|\mathrm{y})d\theta \approx \frac{1}{K}\sum_{k=1}^{K}\alpha(\mathrm{x};\theta^{(k)}) \quad \theta^{(k)} \sim p(\theta|\mathrm{y})\,,$$

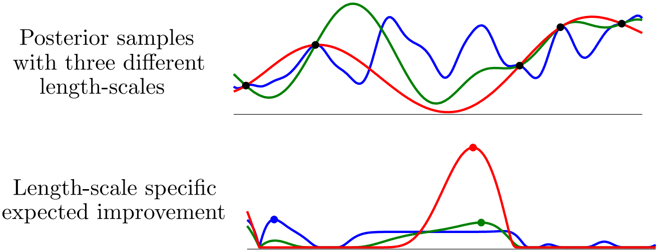
Posterior samples
with three different
length-scales



(Snoek *et al.*, 2012)

Integrated Acquisition Function

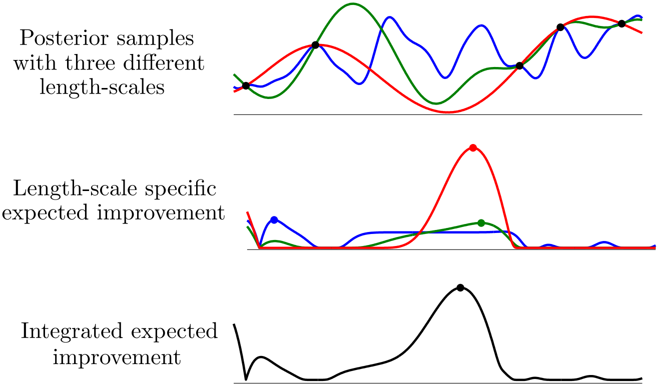
$$\hat{\alpha}(\mathrm{x}) = \int \alpha(\mathrm{x};\theta)p(\theta|\mathrm{y})d\theta \approx \frac{1}{K}\sum_{k=1}^{K}\alpha(\mathrm{x};\theta^{(k)}) \quad \theta^{(k)} \sim p(\theta|\mathrm{y}),$$



(Snoek *et al.*, 2012)

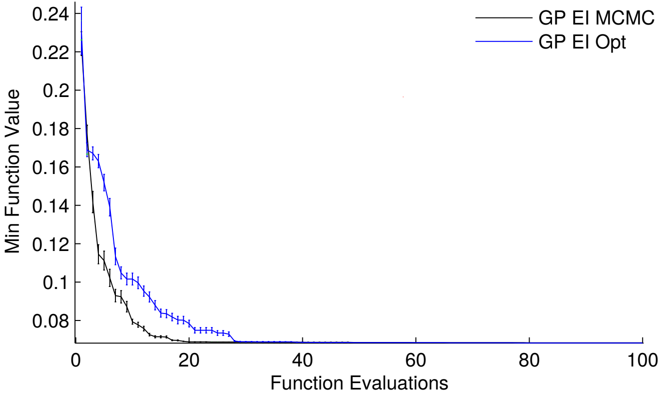
Integrated Acquisition Function

$$\hat{\alpha}(\mathsf{x}) = \int \alpha(\mathsf{x};\theta)p(\theta|\mathsf{y})d\theta \approx \frac{1}{K}\sum_{k=1}^{K}\alpha(\mathsf{x};\theta^{(k)}) \quad \theta^{(k)} \sim p(\theta|\mathsf{y}),$$



(Snoek *et al.*, 2012)

MCMC estimation vs. Maximization



Logistic regression on the MNIST (Snoek *et al.*, 2012).

Cost-sensitive Bayesian Optimization

- Different inputs may have **different computational costs**, *e.g.*, training a neural network of increasing hidden layers and units.

Cost-sensitive Bayesian Optimization

- Different inputs may have **different computational costs**, *e.g.*, training a neural network of increasing hidden layers and units.
- Better to do **cheap evaluations** before expensive ones!

Cost-sensitive Bayesian Optimization

- Different inputs may have **different computational costs**, *e.g.*, training a neural network of increasing hidden layers and units.
- Better to do **cheap evaluations** before expensive ones!
- The evaluation costs are **unknown** but they can be **recorded** and then **modeled** with an additional **Gaussian process**.

Cost-sensitive Bayesian Optimization

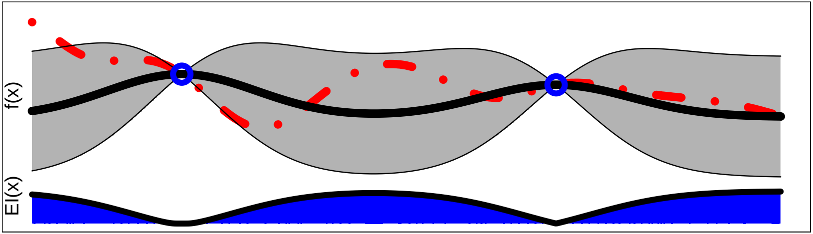
- Different inputs may have **different computational costs**, *e.g.*, training a neural network of increasing hidden layers and units.
- Better to do **cheap evaluations** before expensive ones!
- The evaluation costs are **unknown** but they can be **recorded** and then **modeled** with an additional **Gaussian process**.

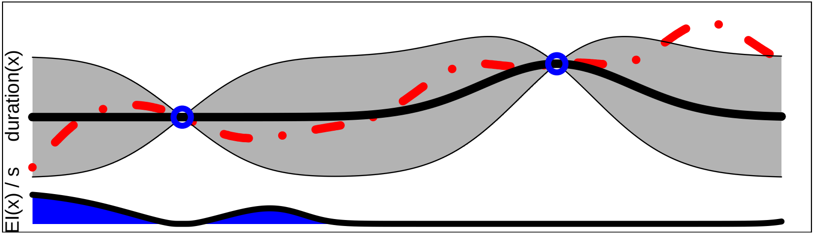
Expected Improvement per-second:

$$\alpha(\mathrm{x}) = \frac{\sigma(\mathrm{x})\left(\gamma(\mathrm{x})\Phi\left(\gamma(\mathrm{x})\right) + \phi(\gamma(\mathrm{x}))\right)}{\exp\left\{\mu_{\text{log-time}}(\mathrm{x})\right\}}$$

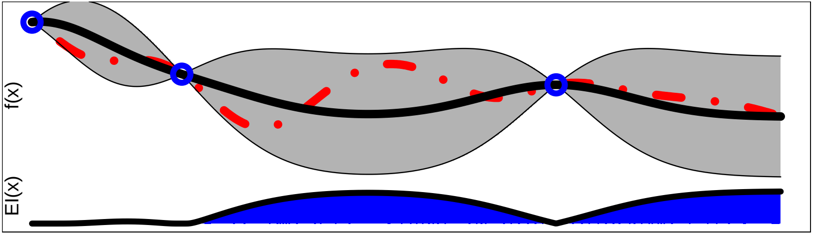
(Snoek *et al.*, 2012)

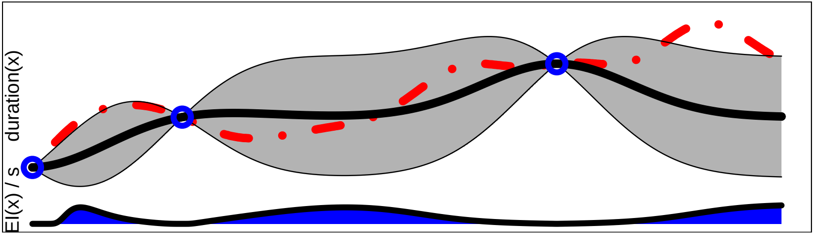
Cost-sensitive Bayesian Optimization



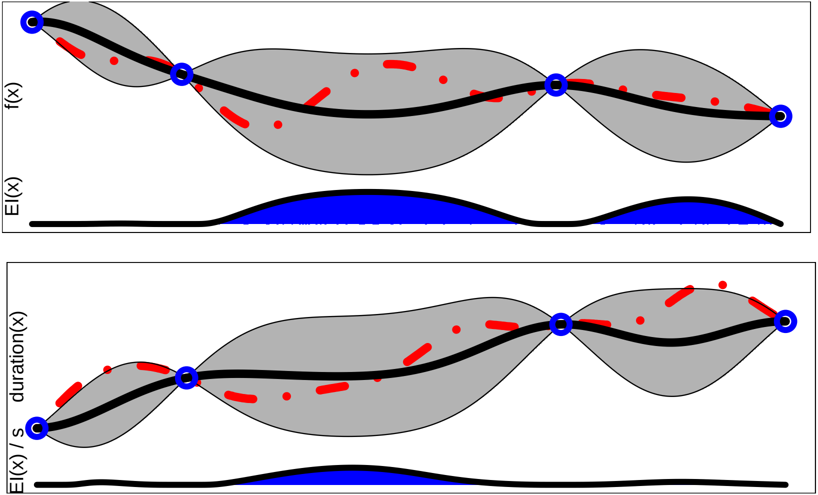


Cost-sensitive Bayesian Optimization

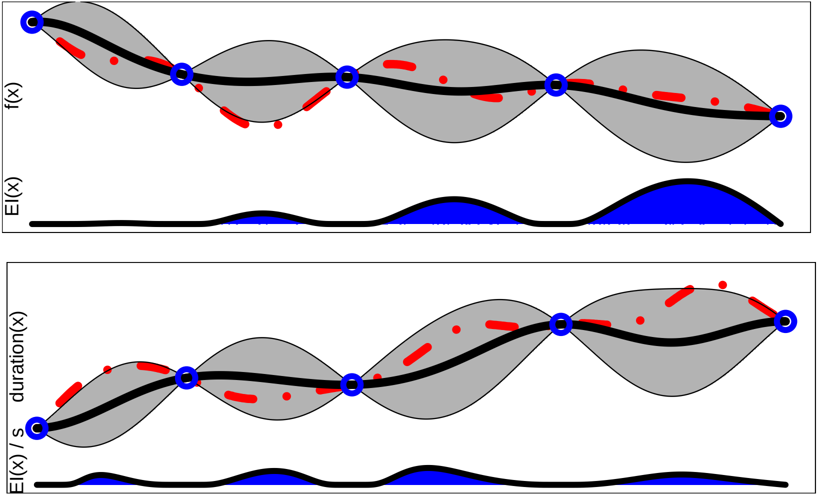




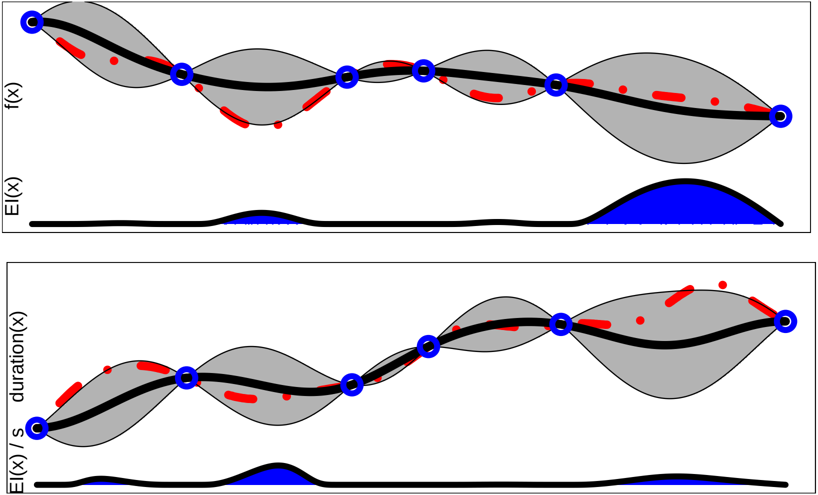
Cost-sensitive Bayesian Optimization



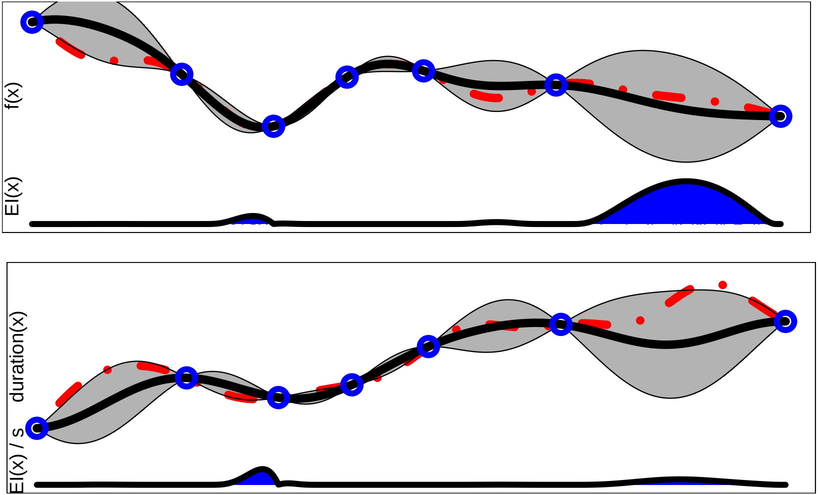
Cost-sensitive Bayesian Optimization



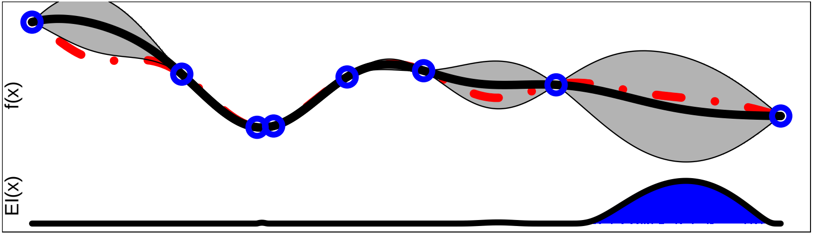
Cost-sensitive Bayesian Optimization

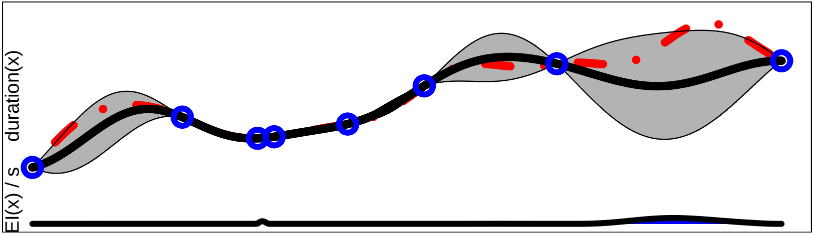


Cost-sensitive Bayesian Optimization

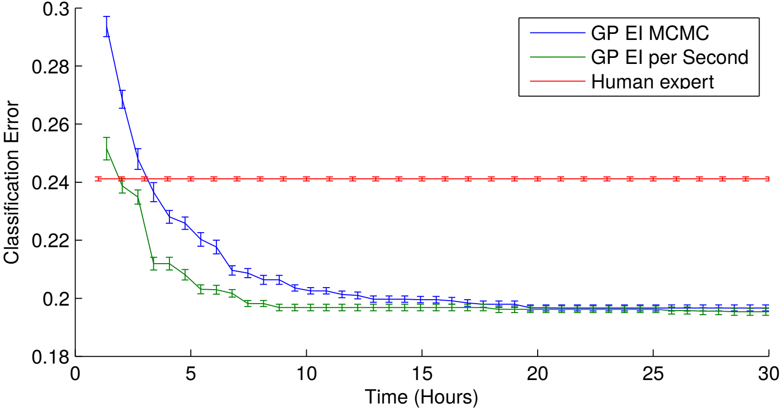


Cost-sensitive Bayesian Optimization





Cost-sensitive Bayesian Optimization



Deep neural network on the CIFAR dataset (Snoek *et al.*, 2012)

Gaussian processes are not the only probabilistic surrogate model!

1. **Computationally expensive**: GPs scale poorly due to cubic observation complexity.

Gaussian processes are not the only probabilistic surrogate model!

1. **Computationally expensive**: GPs scale poorly due to cubic observation complexity.
2. **Fixed covariance structure**: GPs assume a fixed structure in data covariance (stationarity), limiting their ability to model complex data patterns.

Gaussian processes are not the only probabilistic surrogate model!

1. **Computationally expensive**: GPs scale poorly due to cubic observation complexity.
2. **Fixed covariance structure**: GPs assume a fixed structure in data covariance (stationarity), limiting their ability to model complex data patterns.
3. **Difficulties with high dimensionality**: GPs struggle to perform well in high-dimensional search spaces.

Gaussian processes are not the only probabilistic surrogate model!

1. **Computationally expensive**: GPs scale poorly due to cubic observation complexity.
2. **Fixed covariance structure**: GPs assume a fixed structure in data covariance (stationarity), limiting their ability to model complex data patterns.
3. **Difficulties with high dimensionality**: GPs struggle to perform well in high-dimensional search spaces.
4. **Random Forests**: We use its empirical predictive distribution. Perform well in hierarchical spaces, are robust to outliers and scale well.

Gaussian processes are not the only probabilistic surrogate model!

1. **Computationally expensive**: GPs scale poorly due to cubic observation complexity.
2. **Fixed covariance structure**: GPs assume a fixed structure in data covariance (stationarity), limiting their ability to model complex data patterns.
3. **Difficulties with high dimensionality**: GPs struggle to perform well in high-dimensional search spaces.
4. **Random Forests**: We use its empirical predictive distribution. Perform well in hierarchical spaces, are robust to outliers and scale well.
5. **Bayesian neural networks**: Able to model complex patterns and scalable.

Gaussian processes are not the only probabilistic surrogate model!

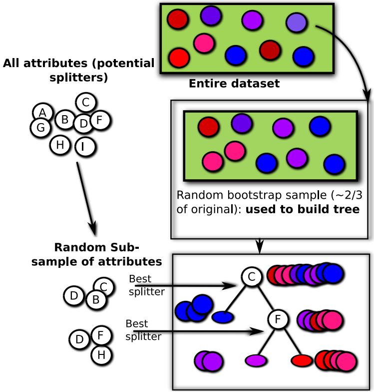
1. **Computationally expensive**: GPs scale poorly due to cubic observation complexity.
2. **Fixed covariance structure**: GPs assume a fixed structure in data covariance (stationarity), limiting their ability to model complex data patterns.
3. **Difficulties with high dimensionality**: GPs struggle to perform well in high-dimensional search spaces.
4. **Random Forests**: We use its empirical predictive distribution. Perform well in hierarchical spaces, are robust to outliers and scale well.
5. **Bayesian neural networks**: Able to model complex patterns and scalable.
6. **Deep Gaussian Process**: Increased expressivity, advantages of GPs.

Other Models: Random Forest

Ensemble method where the predictors are random regression trees trained on random subsamples of the data.

Other Models: Random Forest

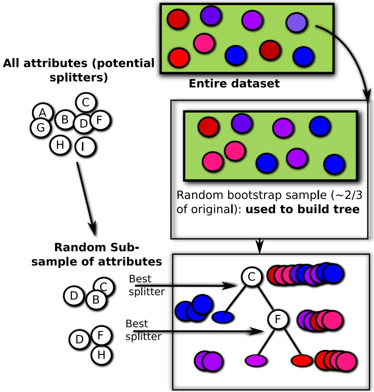
Ensemble method where the predictors are random regression trees trained on random subsamples of the data.



- Trees are grown on different bootstrap samples of the data.

Other Models: Random Forest

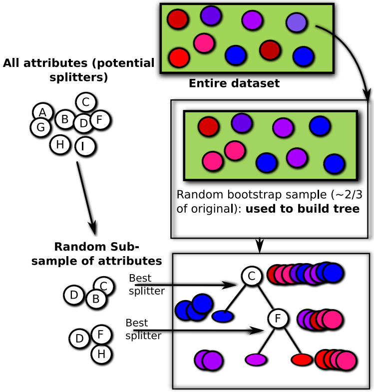
Ensemble method where the predictors are random regression trees trained on random subsamples of the data.



- Trees are grown on different bootstrap samples of the data.
- At each node the best splitter is chosen randomly.

Other Models: Random Forest

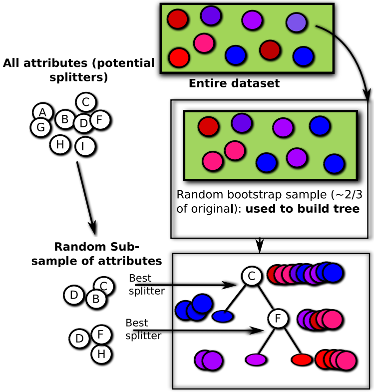
Ensemble method where the predictors are random regression trees trained on random subsamples of the data.



- Trees are grown on different bootstrap samples of the data.
- At each node the best splitter is chosen randomly.
- Leaf nodes predict the average value of the points reaching that node.

Other Models: Random Forest

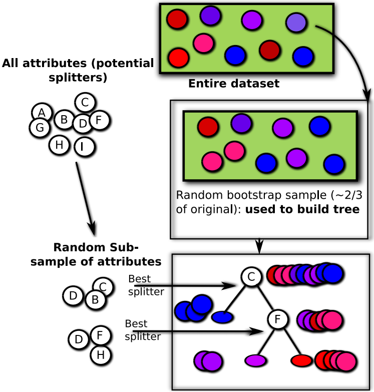
Ensemble method where the predictors are random regression trees trained on random subsamples of the data.



- Trees are grown on different bootstrap samples of the data.
- At each node the best splitter is chosen randomly.
- Leaf nodes predict the average value of the points reaching that node.
- This guarantees that each tree is slightly different.

Other Models: Random Forest

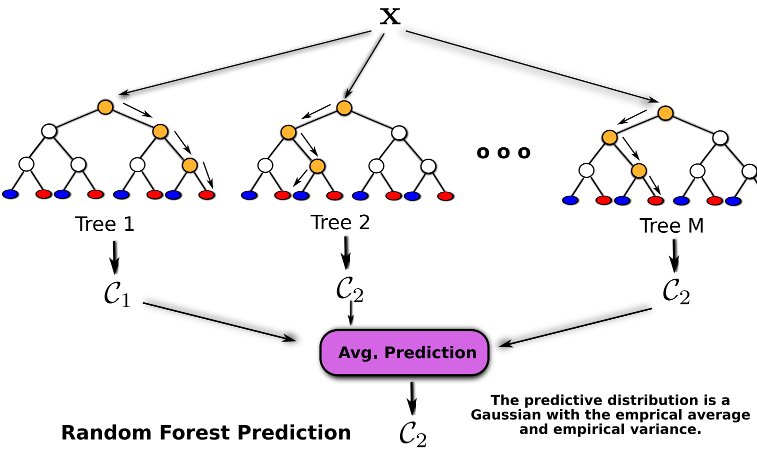
Ensemble method where the predictors are random regression trees trained on random subsamples of the data.



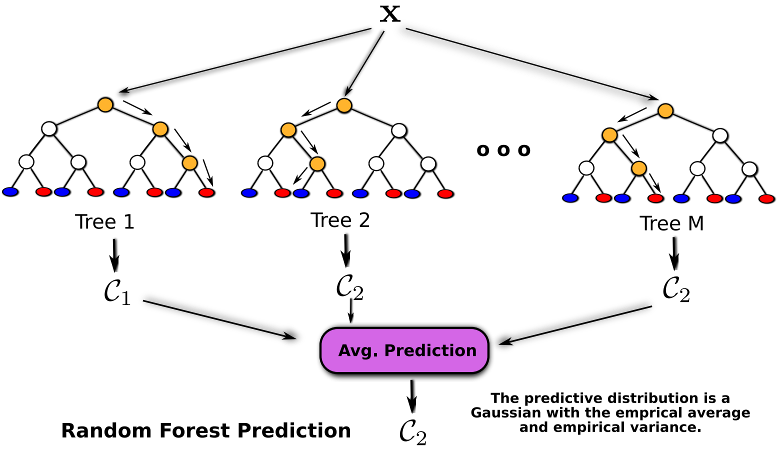
- Trees are grown on different bootstrap samples of the data.
- At each node the best splitter is chosen randomly.
- Leaf nodes predict the average value of the points reaching that node.
- This guarantees that each tree is slightly different.

Very cheap to compute and massively paralelizable!

Random Forest: Predictive Distribution



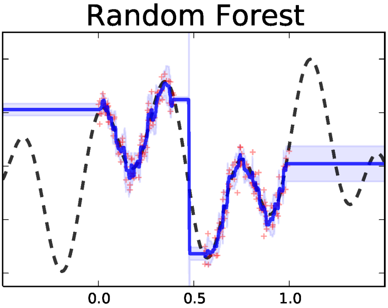
Random Forest: Predictive Distribution



$$p(f^\star|\mathcal{D}_n) = \mathcal{N}(f^\star|\overline{\mu}, \overline{\nu}^2)$$

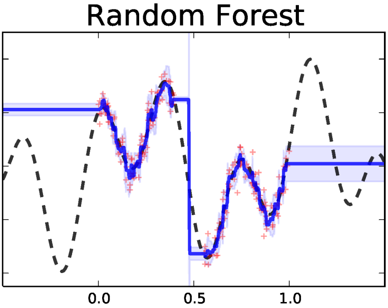
(Hutter et al., 2011)

Random Forest in Practice



(Shahriari et al., 2016)

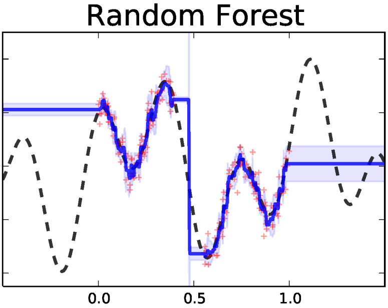
Random Forest in Practice



(Shahriari et al., 2016)

- Allows for a lot of evaluations (good when the objective is cheap).

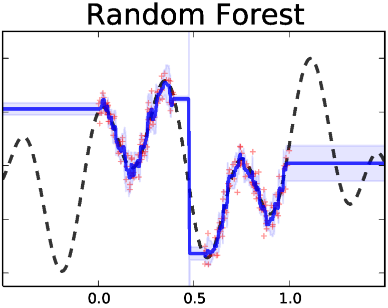
Random Forest in Practice



(Shahriari et al., 2016)

- Allows for a lot of evaluations (good when the objective is cheap).
- Too confident intervals in far away from the data regions.

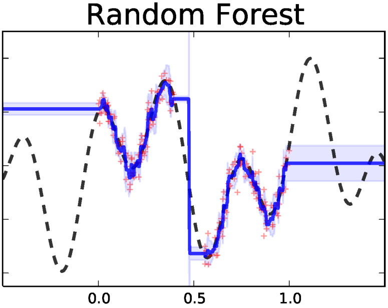
Random Forest in Practice



(Shahriari et al., 2016)

- Allows for a lot of evaluations (good when the objective is cheap).
- Too confident intervals in far away from the data regions.
- Conflictive predictions can cause the variance to be too high.

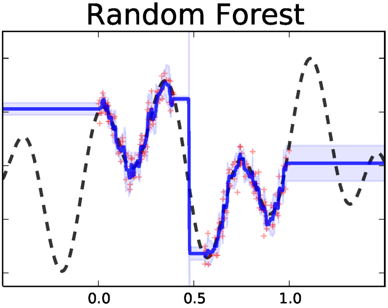
Random Forest in Practice



(Shahriari et al., 2016)

- Allows for a lot of evaluations (good when the objective is cheap).
- Too confident intervals in far away from the data regions.
- Conflictive predictions can cause the variance to be too high.
- Discontinuous: Difficult to optimize the acquisition function.

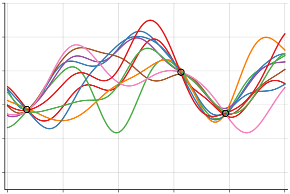
Random Forest in Practice

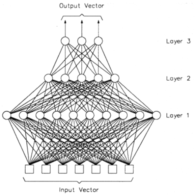


(Shahriari et al., 2016)

- Allows for a lot of evaluations (good when the objective is cheap).
- Too confident intervals in far away from the data regions.
- Conflictive predictions can cause the variance to be too high.
- Discontinuous: Difficult to optimize the acquisition function.
- No parameters to tune.

Other Models: Bayesian Neural Networks



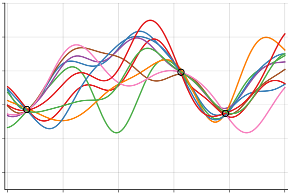


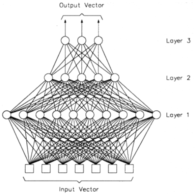
$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

- Neural networks scale well to the training data (linear cost).

Other Models: Bayesian Neural Networks



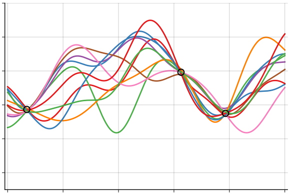


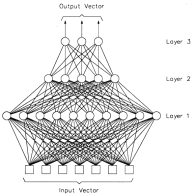
$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

- Neural networks scale well to the training data (linear cost).
- Trained very fast on GPUs.

Other Models: Bayesian Neural Networks



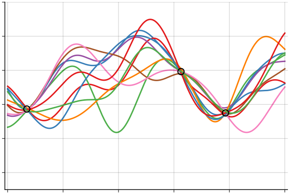


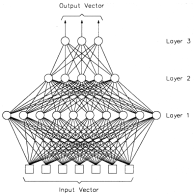
$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

- Neural networks scale well to the training data (linear cost).
- Trained very fast on GPUs.
- State of the art prediction results.

Other Models: Bayesian Neural Networks





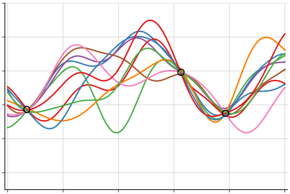
$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

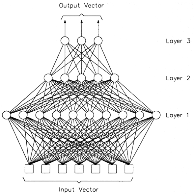
$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

- Neural networks scale well to the training data (linear cost).
- Trained very fast on GPUs.
- State of the art prediction results.

They are an alternative to GPs to allow for a large number observations!

Other Models: Bayesian Neural Networks





$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

- Neural networks scale well to the training data (linear cost).
- Trained very fast on GPUs.
- State of the art prediction results.

They are an alternative to GPs to allow for a large number observations!

The posterior distribution of the networks weights W **is intractable!**

Bayesian Neural Networks: Predictive Distribution

Several techniques considered to approximate the predictive distribution:

Bayesian Neural Networks: Predictive Distribution

Several techniques considered to approximate the predictive distribution:

- Markov Chain Monte Carlo methods.

Bayesian Neural Networks: Predictive Distribution

Several techniques considered to approximate the predictive distribution:

- Markov Chain Monte Carlo methods.
- Variational Inference.

Bayesian Neural Networks: Predictive Distribution

Several techniques considered to approximate the predictive distribution:

- Markov Chain Monte Carlo methods.
- Variational Inference.
- Expectation Propagation.

Bayesian Neural Networks: Predictive Distribution

Several techniques considered to approximate the predictive distribution:

- Markov Chain Monte Carlo methods.
- Variational Inference.
- Expectation Propagation.
- Reinterpretations of dropout.

Bayesian Neural Networks: Predictive Distribution

Several techniques considered to approximate the predictive distribution:

- Markov Chain Monte Carlo methods.
- Variational Inference.
- Expectation Propagation.
- Reinterpretations of dropout.
- Point estimates and Bayesian linear-models in the last layer.

Bayesian Neural Networks: Predictive Distribution

Several techniques considered to approximate the predictive distribution:

- Markov Chain Monte Carlo methods.
- Variational Inference.
- Expectation Propagation.
- Reinterpretations of dropout.
- Point estimates and Bayesian linear-models in the last layer.

Trade-off between accuracy of the predictive distribution and scalability! Still a lot of research going on!

Activity: Bayesian optimization of a ML algorithm

Use a generative AI to generate a BO code of a ML hyperparameter problem.

- Try to execute the code in your machine and fix it.
- Try to interpret everything.
- Tune it and improve the quality of the error estimator.
- Try to improve it with additional acquisition functions.
- Try to obtain the best score in the titanic problem with BO-ML!

Software for Bayesian Optimization

Many of the methods described are implemented into **BOTorch** using Python.
`https://botorch.org/`

Software for Bayesian Optimization

Many of the methods described are implemented into **BOTorch** using Python.
`https://botorch.org/`

BOTorch's super-nice features:

1. **Modularity**: Plugin new models, acquisition functions and optimizers.

Software for Bayesian Optimization

Many of the methods described are implemented into **BOTorch** using Python.
`https://botorch.org/`

BOTorch's super-nice features:

1. **Modularity**: Plugin new models, acquisition functions and optimizers.
2. **PyTorch**: Easily integrate neural network modules. Native GPU and Autograd support.

Software for Bayesian Optimization

Many of the methods described are implemented into **BOTorch** using Python.
`https://botorch.org/`

BOTorch's super-nice features:

1. **Modularity**: Plugin new models, acquisition functions and optimizers.
2. **PyTorch**: Easily integrate neural network modules. Native GPU and Autograd support.
3. **Scalability**: Support for scalable GPs via GPyTorch.

Software for Bayesian Optimization

Many of the methods described are implemented into **BOTorch** using Python.
`https://botorch.org/`

BOTorch's super-nice features:

1. **Modularity**: Plugin new models, acquisition functions and optimizers.
2. **PyTorch**: Easily integrate neural network modules. Native GPU and Autograd support.
3. **Scalability**: Support for scalable GPs via GPyTorch.
4. **Advanced scenarios**: Advanced research coded into tutorials about complex BO scenarios.

Software for Bayesian Optimization

Many of the methods described are implemented into **BOTorch** using Python.
`https://botorch.org/`

BOTorch's super-nice features:

1. **Modularity**: Plugin new models, acquisition functions and optimizers.
2. **PyTorch**: Easily integrate neural network modules. Native GPU and Autograd support.
3. **Scalability**: Support for scalable GPs via GPyTorch.
4. **Advanced scenarios**: Advanced research coded into tutorials about complex BO scenarios.

Other tools: SMAC3 (Python-RFs), GPyOpt (Python3), Spearmint (Python2.7-sampling), mlrMBO (R).

Time to practice!

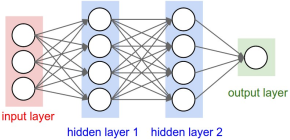
1. Bayesian optimization of a benchmark optimization function.
2. Bayesian optimization of the hyper-parameters of a machine learning model.
3. Bayesian optimization of the hyper-paramaters of a deep reinforcement learning algorithm.

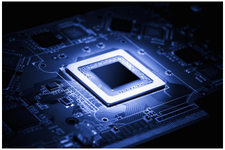
Second Session: Advanced Bayesian optimization.

(Parallel) Multi-objective Bayesian optimization with constraints.

Several Objectives and Constraints

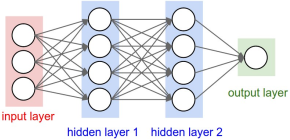
Optimal design of **hardware accelerator** for neural network predictions.

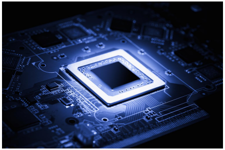




Several Objectives and Constraints

Optimal design of **hardware accelerator** for neural network predictions.



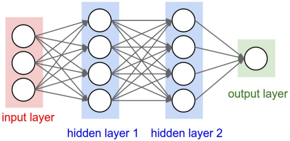


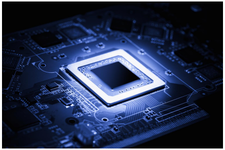
Goals:

- Minimize **prediction error**.
- Minimize **prediction time**.

Several Objectives and Constraints

Optimal design of **hardware accelerator** for neural network predictions.





Goals:

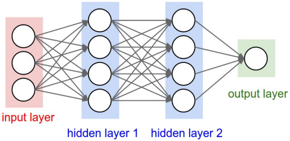
- Minimize **prediction error**.
- Minimize **prediction time**.

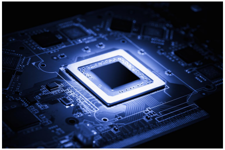
Constrained to:

- **Chip area** below a value.
- **Power consumption** below a level.

Several Objectives and Constraints

Optimal design of **hardware accelerator** for neural network predictions.



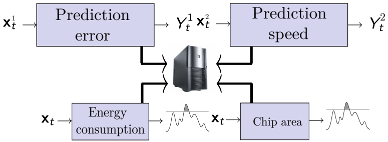


Goals:

- Minimize **prediction error**.
- Minimize **prediction time**.

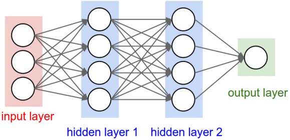
Constrained to:

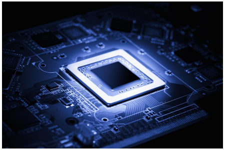
- **Chip area** below a value.
- **Power consumption** below a level.



Several Objectives and Constraints

Optimal design of **hardware accelerator** for neural network predictions.



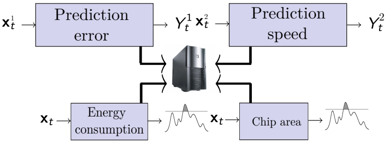


Goals:

- Minimize **prediction error**.
- Minimize **prediction time**.

Constrained to:

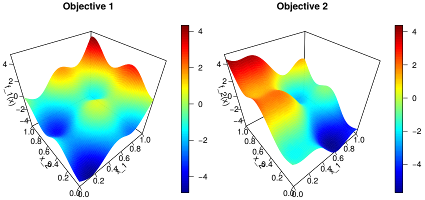
- **Chip area** below a value.
- **Power consumption** below a level.



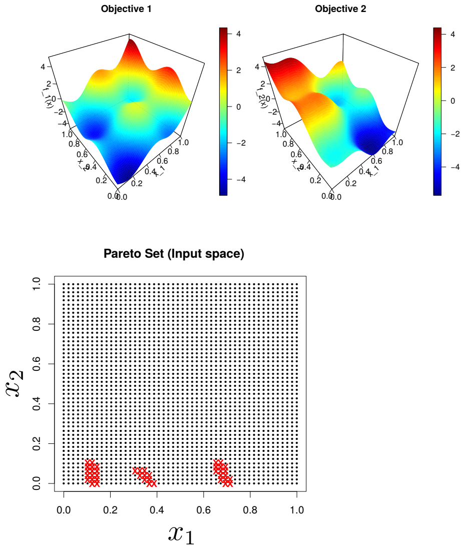
Challenges:

- **Complicated** constraints.
- **Conflictive** objectives.

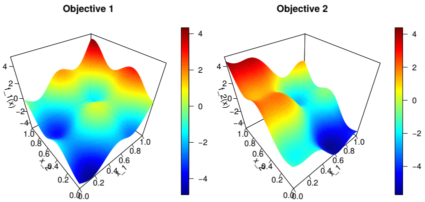
Constrained Multi-Objective Optimization

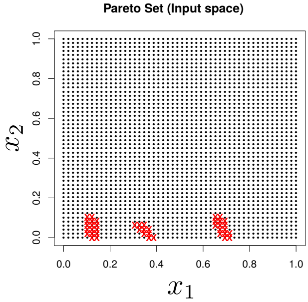


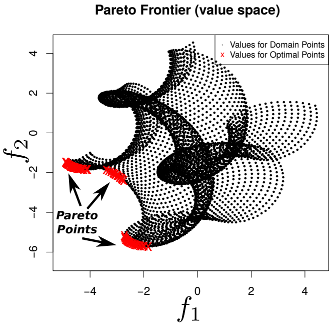
Constrained Multi-Objective Optimization



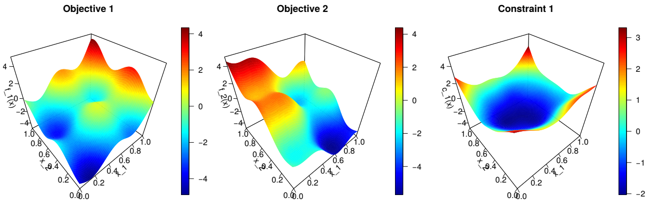
Constrained Multi-Objective Optimization

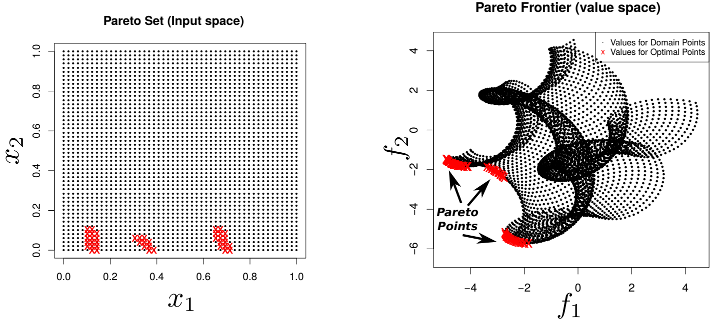




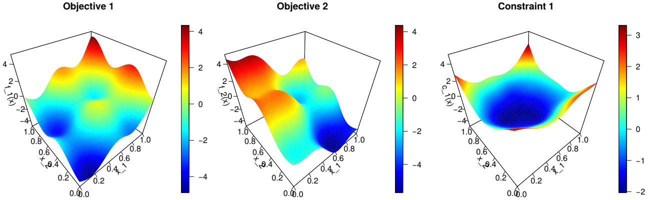


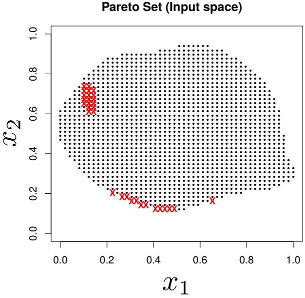
Constrained Multi-Objective Optimization

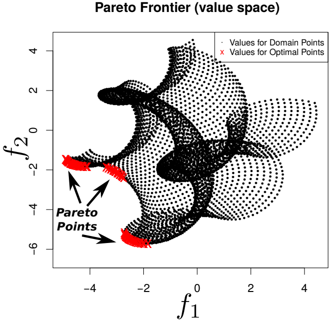




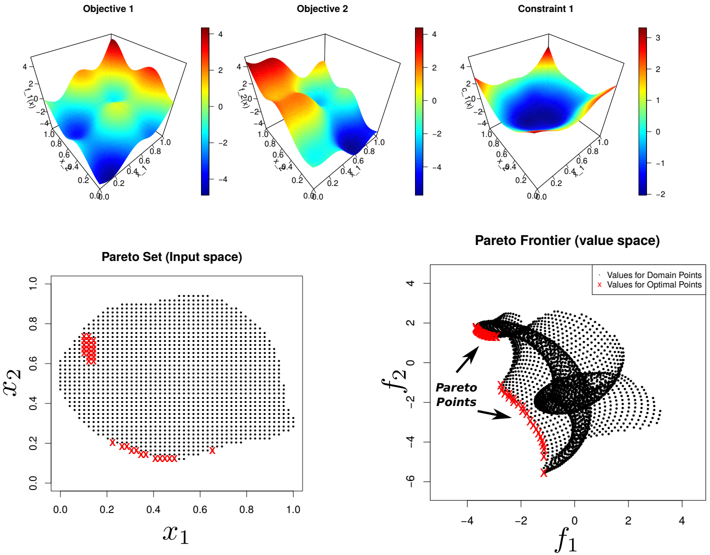
Constrained Multi-Objective Optimization







Constrained Multi-Objective Optimization



Bayesian Optimization Methods

Additional challenges when dealing with several black-boxes.

Bayesian Optimization Methods

Additional challenges when dealing with several black-boxes.

- Simple approach: evaluate **all** the objectives and constraints at the **same input location**. Expected to be sub-optimal.

Bayesian Optimization Methods

Additional challenges when dealing with several black-boxes.

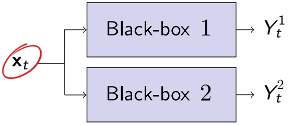
- Simple approach: evaluate **all** the objectives and constraints at the **same input location**. Expected to be sub-optimal.
- Advanced approach: make **intelligent** decisions about what **black-box** to evaluate **next** and on **which location**.

Bayesian Optimization Methods

Additional challenges when dealing with several black-boxes.

- Simple approach: evaluate **all** the objectives and constraints at the **same input location**. Expected to be sub-optimal.
- Advanced approach: make **intelligent** decisions about what **black-box** to evaluate **next** and on **which location**.

Coupled evaluations

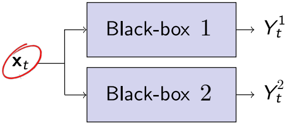


Bayesian Optimization Methods

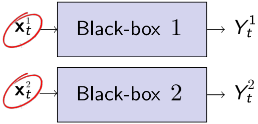
Additional challenges when dealing with several black-boxes.

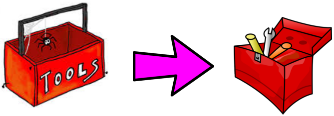
- Simple approach: evaluate **all** the objectives and constraints at the **same input location**. Expected to be sub-optimal.
- Advanced approach: make **intelligent** decisions about what **black-box** to **evaluate next** and on **which location**.

Coupled evaluations



Decoupled evaluations





Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information-based Approach

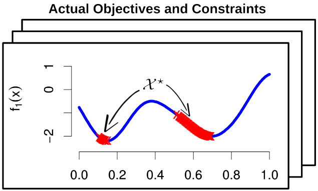
The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.

Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

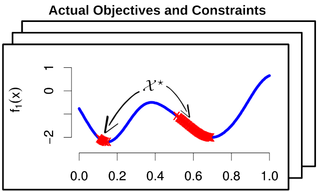
Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.

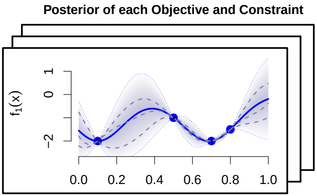


Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.

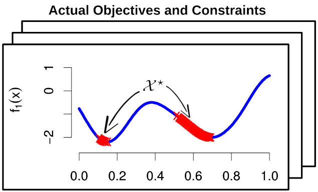


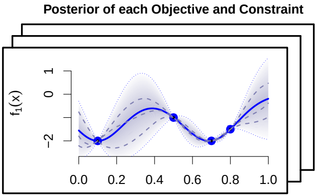


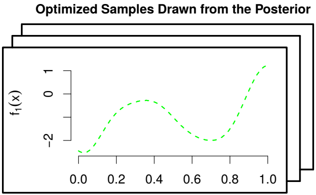
Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.



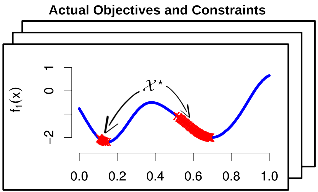


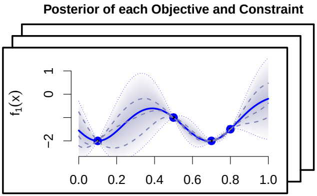


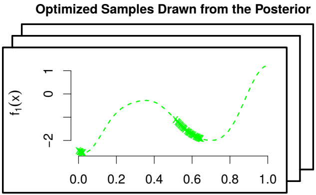
Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.





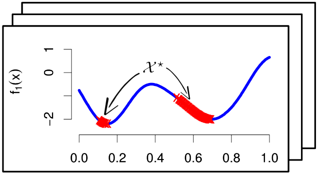


Information-based Approach

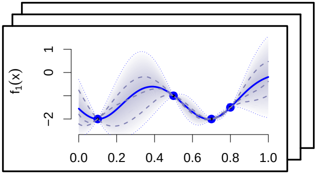
The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star | \mathcal{D}_N)$.

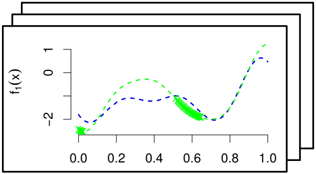
Actual Objectives and Constraints



Posterior of each Objective and Constraint



Optimized Samples Drawn from the Posterior

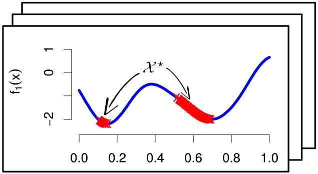


Information-based Approach

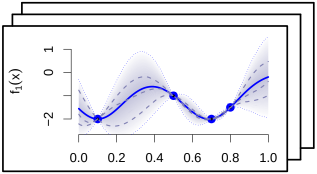
The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star | \mathcal{D}_N)$.

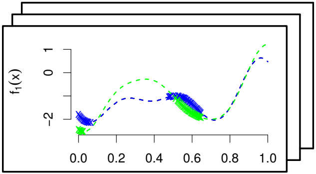
Actual Objectives and Constraints



Posterior of each Objective and Constraint



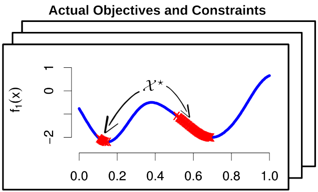
Optimized Samples Drawn from the Posterior

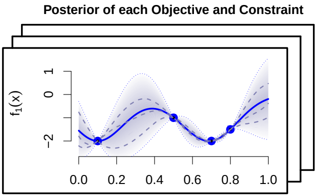


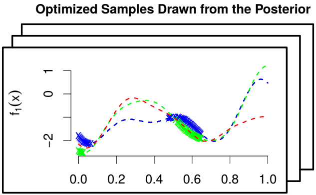
Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.





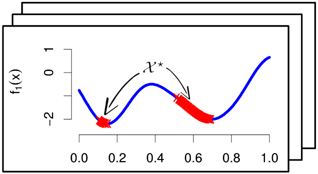


Information-based Approach

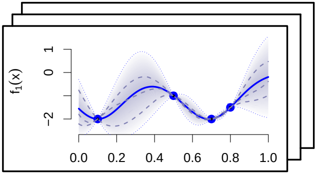
The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.

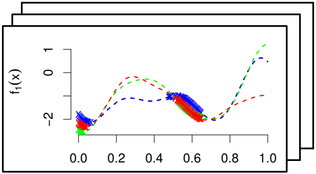
Actual Objectives and Constraints



Posterior of each Objective and Constraint



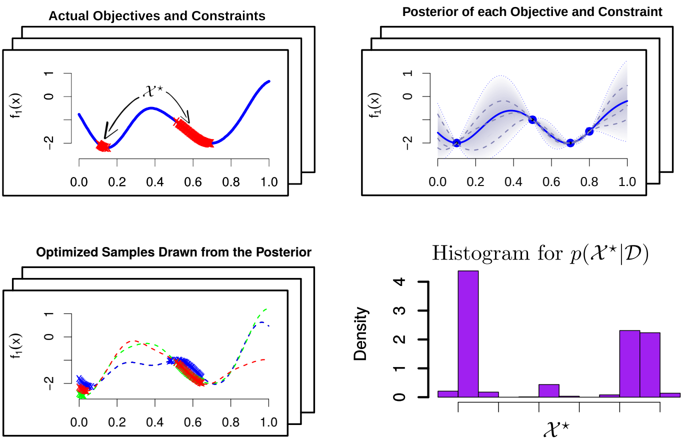
Optimized Samples Drawn from the Posterior



Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star | \mathcal{D}_N)$.



Information-based Approach

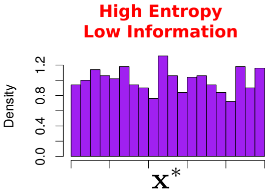
The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

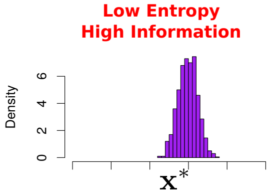
Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.

Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.

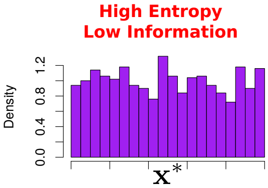


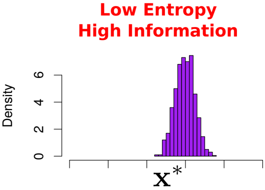


Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.





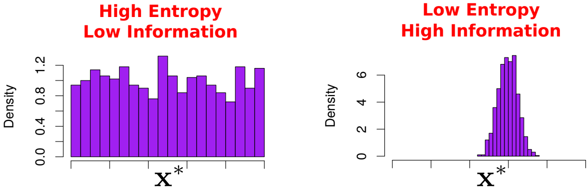
The acquisition function is

$$\alpha(\mathbf{x}) = \mathsf{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.



The acquisition function is

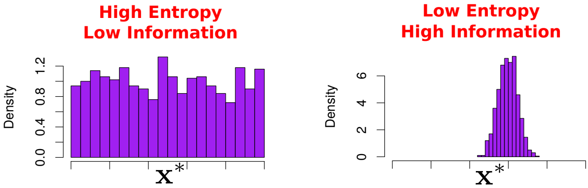
$$\alpha(\mathbf{x}) = \mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

How much we know about $\mathbf{x}^*$ now.

Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.



The acquisition function is

$$\alpha(\mathbf{x}) = \mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

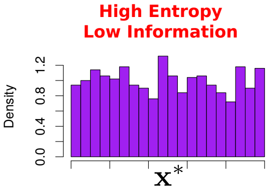
How much we know about $\mathbf{x}^*$ now.

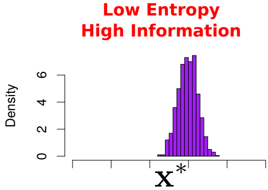
How much we will know about $\mathbf{x}^*$ after collecting y at $\mathbf{x}$.

Information-based Approach

The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.





The acquisition function is

$$\alpha(\mathbf{x}) = \mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^* \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

How much we know about $\mathbf{x}^*$ now.

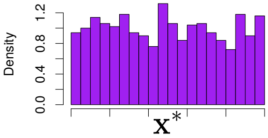
How much we will know about $\mathbf{x}^*$ after collecting y at $\mathbf{x}$.

Information-based Approach

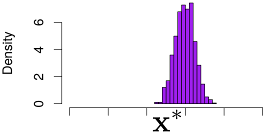
The Pareto set $\mathcal{X}^\star$ in the feasible space is a **random variable**!

Information is measured by the **entropy** of $p(\mathcal{X}^\star|\mathcal{D}_N)$.

High Entropy
Low Information



Low Entropy
High Information



The acquisition function is

$$\alpha(\mathbf{x}) = \mathrm{H}\left[\mathbf{x}^*\middle|\mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^*\middle|\mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right]\middle|\mathcal{D}_t, \mathbf{x}\right] \quad (1)$$

How much we know about $\mathbf{x}^*$ now.

How much we will know about $\mathbf{x}^*$ after collecting y at $\mathbf{x}$.

Computing (1) is **very difficult in practice**!

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathsf{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

Predictive Entropy Search (PES)

We **swap** $\mathbf{y}$ **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathsf{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathsf{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathsf{H}\big[\mathcal{X}^\star\big|\mathcal{D}_t\big] - \mathbb{E}_{\mathbf{y}}\Big[\mathsf{H}\big[\mathcal{X}^\star\big|\mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\big]\Big|\mathcal{D}_t, \mathbf{x}\Big] \equiv \mathsf{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathsf{H}\big[\mathbf{y}\big|\mathcal{D}_t, \mathbf{x}\big] - \mathbb{E}_{\mathcal{X}^\star}\Big[\mathsf{H}\big[\mathbf{y}\big|\mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\big]\Big|\mathcal{D}_t, \mathbf{x}\Big] \equiv \mathsf{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Predictive Entropy Search (PES)

We **swap** $\mathbf{y}$ **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathsf{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathsf{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Gaussian distribution

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Gaussian distribution

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Gaussian distribution

Approximated by sampling from $p(\mathcal{X}^\star | \mathcal{D}_t$

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathsf{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathsf{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Gaussian distribution

Approximated by sampling from $p(\mathcal{X}^\star|\mathcal{D}_t$

Predictive Entropy Search (PES)

We **swap** $\mathbf{y}$ **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Gaussian distribution

Approximated by sampling from $p(\mathcal{X}^\star | \mathcal{D}_t)$

Factorized Gaussian approximation with **expectation propagation**. $\mathcal{X}^\star$ dominates any other point in $\mathcal{X}$.

(Minka, 2001)

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Gaussian distribution

Approximated by sampling from $p(\mathcal{X}^\star|\mathcal{D}_t)$

Factorized Gaussian approximation with **expectation propagation**. $\mathcal{X}^\star$ dominates any other point in $\mathcal{X}$.

$$\boxed{\alpha(\mathbf{x}) \approx \sum_{c=1}^{C} \log v_c^{PD}(\mathbf{x}) - \frac{1}{M}\sum_{m=1}^{M}\left(\sum_{c=1}^{C} \log v_c^{CPD}(\mathbf{x}|\mathcal{X}^\star_{(m)})\right) + \sum_{k=1}^{K} \log v_k^{PD}(\mathbf{x}) - \frac{1}{M}\sum_{m=1}^{M}\left(\sum_{k=1}^{K} \log v_k^{CPD}(\mathbf{x}|\mathcal{X}^\star_{(m)})\right)}$$

(Minka, 2001)

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Gaussian distribution

Approximated by sampling from $p(\mathcal{X}^\star | \mathcal{D}_t)$

Factorized Gaussian approximation with **expectation propagation**. $\mathcal{X}^\star$ dominates any other point in $\mathcal{X}$.

$$\alpha(\mathbf{x}) \approx \sum_{c=1}^{C} \log v_c^{PD}(\mathbf{x}) - \frac{1}{M}\sum_{m=1}^{M}\left(\sum_{c=1}^{C} \log v_c^{CPD}(\mathbf{x}|\mathcal{X}^\star_{(m)})\right) + \sum_{k=1}^{K} \log v_k^{PD}(\mathbf{x}) - \frac{1}{M}\sum_{m=1}^{M}\left(\sum_{k=1}^{K} \log v_k^{CPD}(\mathbf{x}|\mathcal{X}^\star_{(m)})\right) = \sum_{i=1}^{C+K} \alpha_i(\mathbf{x})$$

(Minka, 2001)

Predictive Entropy Search (PES)

We **swap** y **and** $\mathcal{X}^\star$ to obtain a reformulation of the acquisition function.

$$\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathcal{X}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}, \mathbf{y}\}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathcal{X}^\star) \quad \text{(ESMOC)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathcal{X}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathcal{X}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathcal{X}^\star, \mathbf{y}) \quad \text{(PESMOC)}$$

Gaussian distribution

Approximated by sampling from $p(\mathcal{X}^\star|\mathcal{D}_t)$

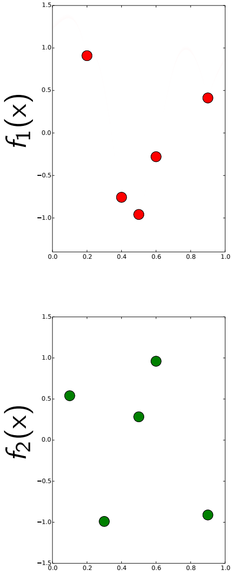
Factorized Gaussian approximation with **expectation** ... $\mathcal{X}^\star$ dominates any ...

One acquisition per black-box

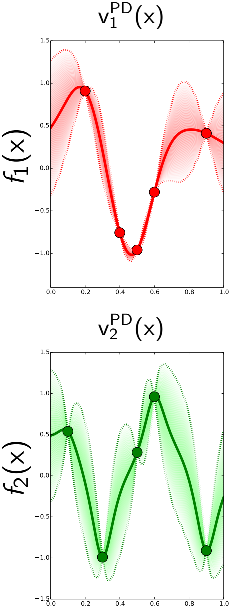
$$\alpha(\mathbf{x}) \approx \sum_{c=1}^{C} \log v_c^{PD}(\mathbf{x}) - \frac{1}{M}\sum_{m=1}^{M}\left(\sum_{c=1}^{C} \log v_c^{CPD}(\mathbf{x}|\mathcal{X}^\star_{(m)})\right) + \sum_{k=1}^{K} \log v_k^{PD}(\mathbf{x}) - \frac{1}{M}\sum_{m=1}^{M}\left(\sum_{k=1}^{K} \log v_k^{CPD}(\mathbf{x}|\mathcal{X}^\star_{(m)})\right) = \sum_{i=1}^{C+K} \alpha_i(\mathbf{x})$$

(Minka, 2001)

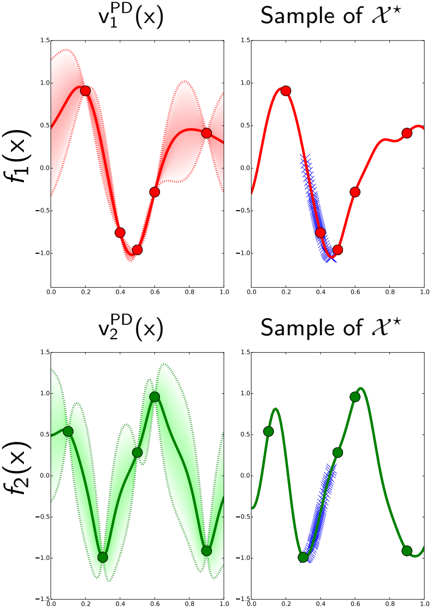
Example of PES' acquisition



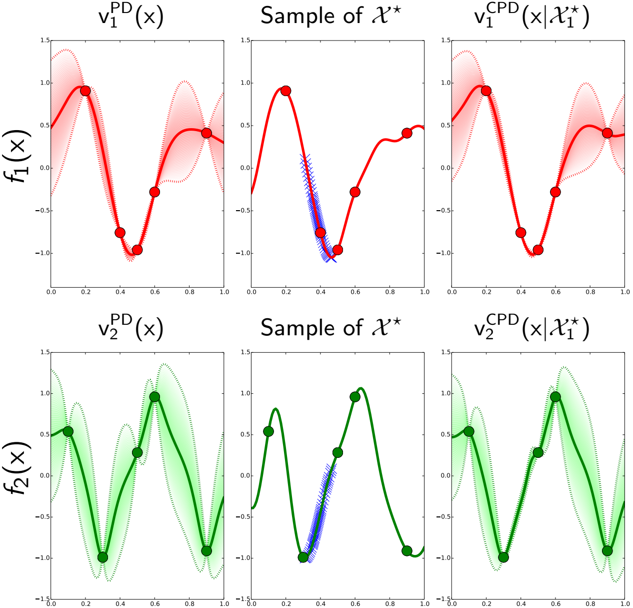
Example of PES' acquisition



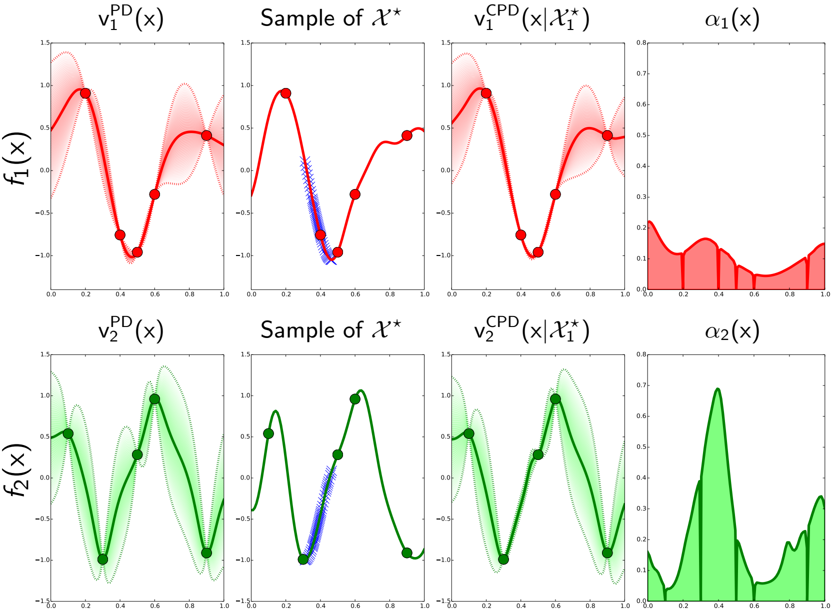
Example of PES' acquisition



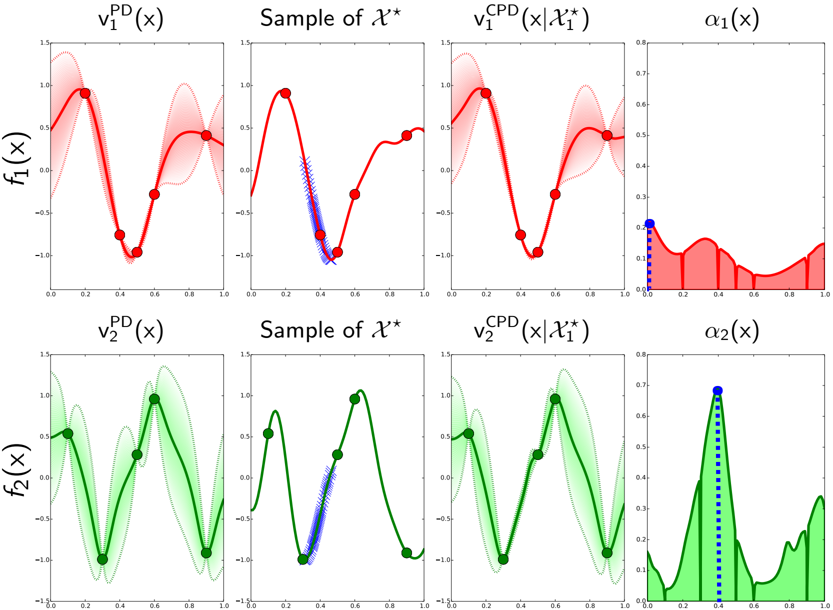
Example of PES' acquisition



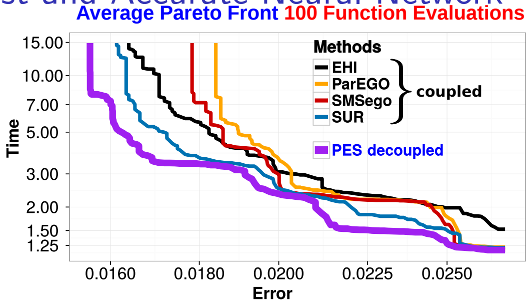
Example of PES' acquisition



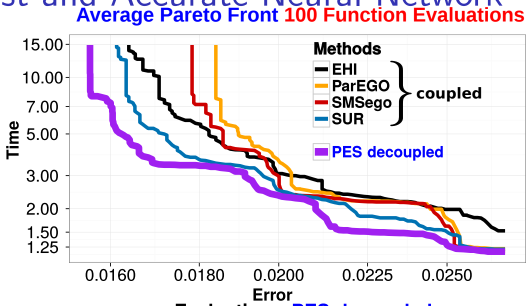
Example of PES' acquisition

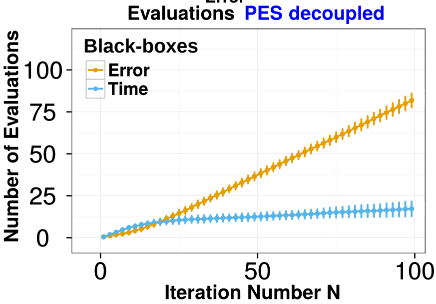


Finding a Fast and Accurate Neural Network



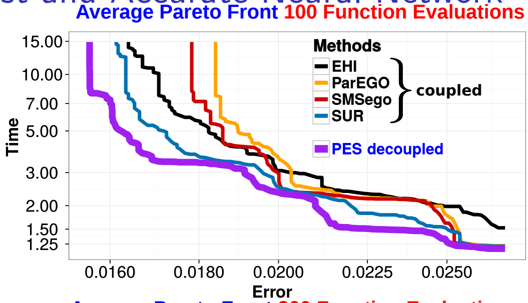
Finding a Fast and Accurate Neural Network

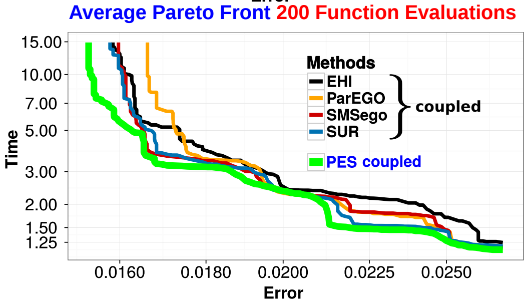




(Hernández-Lobato *et al.*, 2016)

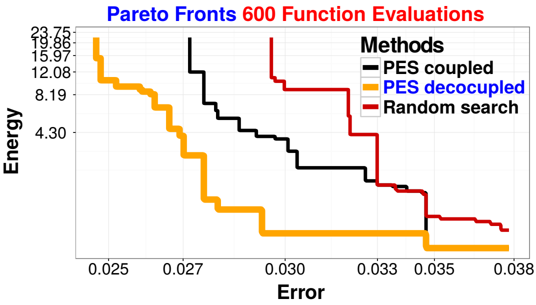
Finding a Fast and Accurate Neural Network



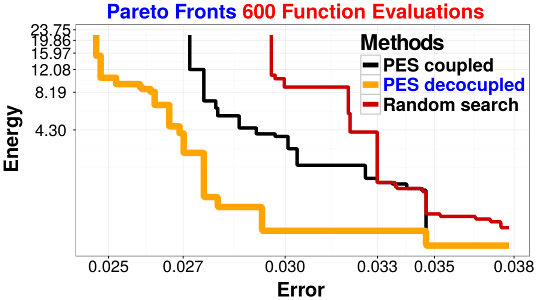


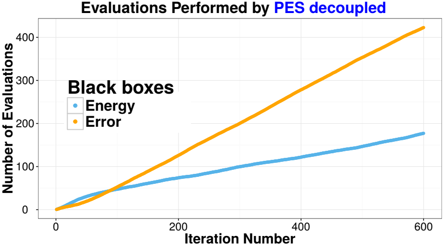
(Hernández-Lobato *et al.*, 2016)

Low energy hardware accelerator



Low energy hardware accelerator





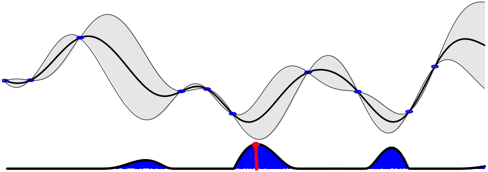
(Hernández-Lobato *et al.*, 2016)

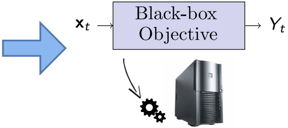
Parallel Bayesian Optimization

Traditional Bayesian optimization is **sequential**!

Parallel Bayesian Optimization

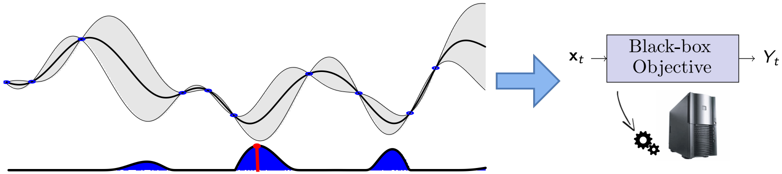
Traditional Bayesian optimization is **sequential**!





Parallel Bayesian Optimization

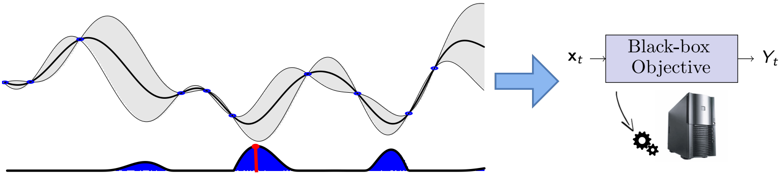
Traditional Bayesian optimization is **sequential**!



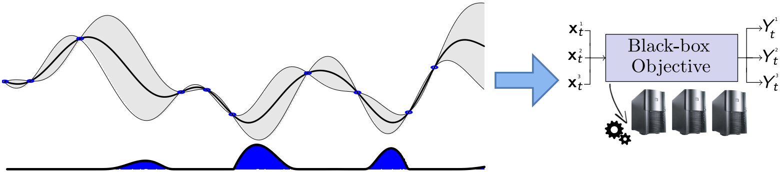
Computing clusters let us do **many things** at once!

Parallel Bayesian Optimization

Traditional Bayesian optimization is **sequential**!

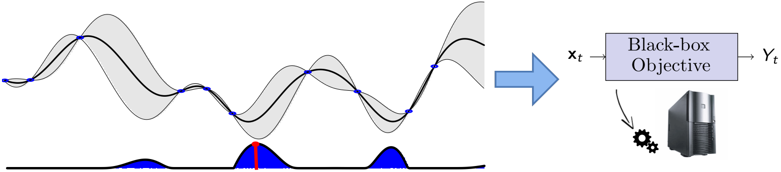


Computing clusters let us do **many things** at once!

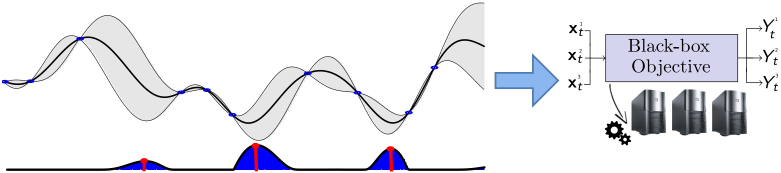


Parallel Bayesian Optimization

Traditional Bayesian optimization is **sequential**!

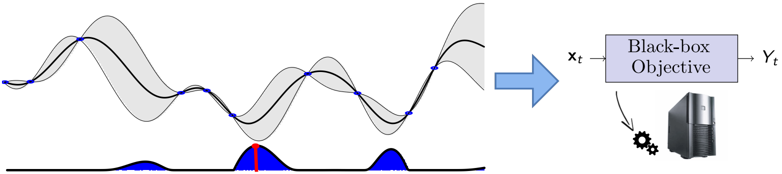


Computing clusters let us do **many things** at once!

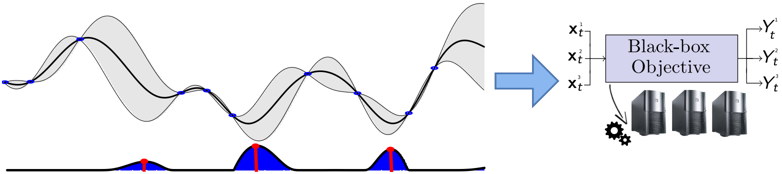


Parallel Bayesian Optimization

Traditional Bayesian optimization is **sequential**!



Computing clusters let us do **many things** at once!



Parallel experiments should be highly informative but different!

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathsf{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathsf{x}^\star$.

$$\mathsf{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathrm{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathrm{x}^\star$.

$$\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathbf{x}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathrm{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathrm{x}^\star$.

$$\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathbf{x}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathrm{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathrm{x}^\star$.

$$\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathbf{x}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathrm{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathrm{x}^\star$.

$$\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathbf{x}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

Multi-variate Gaussian distribution

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathrm{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathrm{x}^\star$.

$$\mathsf{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathsf{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathbf{x}^\star}\left[\mathsf{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

Multi-variate Gaussian distribution

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathrm{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathrm{x}^\star$.

$$\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathbf{x}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

Multi-variate Gaussian distribution

Approximated by sampling from $p(\mathbf{x}^\star | \mathcal{D}_t)$

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathrm{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathrm{x}^\star$.

$$\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}\right] - \mathbb{E}_{\mathbf{x}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

Multi-variate Gaussian distribution

Approximated by sampling from $p(\mathbf{x}^\star | \mathcal{D}_t)$

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathrm{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathrm{x}^\star$.

$$\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathrm{H}\left[\boxed{\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}}\right] - \boxed{\mathbb{E}_{\mathbf{x}^\star}}\left[\mathrm{H}\left[\boxed{\mathbf{y} \middle| \mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star}\right] \middle| \mathcal{D}_t, \mathbf{x}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

Multi-variate Gaussian distribution

Approximated by sampling from $p(\mathbf{x}^\star|\mathcal{D}_t)$

Multivariate Gaussian approximation with **expectation propagation**
$\mathbf{x}^\star$ is better than any other point in $\mathcal{X}$

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{x_q\}_{q=1}^{Q}$ to minimize the entropy of $x^\star$.

$$\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{X}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{X}\right] - \mathbb{E}_{\mathbf{x}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{X}, \mathbf{x}^\star\right] \middle| \mathcal{D}_t, \mathbf{X}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

Multi-variate Gaussian distribution

Approximated by sampling from $p(\mathbf{x}^\star|\mathcal{D}_t)$

Multivariate Gaussian approximation with **expectation propagation**
$\mathbf{x}^\star$ is better than any other point in $\mathcal{X}$

$$\alpha(\mathbf{X}) = \log|\mathbf{V}^{\mathrm{PD}}(\mathbf{X})| - \frac{1}{M}\sum_{m=1}^{M} \log|\mathbf{V}^{\mathrm{CPD}}(\mathbf{X}|\mathbf{x}^\star_{(m)})|$$

(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search

Choose a set Q points $\mathcal{S}_t = \{\mathrm{x}_q\}_{q=1}^{Q}$ to minimize the entropy of $\mathrm{x}^\star$.

$$\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t\right] - \mathbb{E}_{\mathbf{y}}\left[\mathrm{H}\left[\mathbf{x}^\star \middle| \mathcal{D}_t \cup \{\mathbf{x}_q, y_q\}_{q=1}^{Q}\right] \middle| \mathcal{D}_t, \mathbf{X}\right] \equiv \mathrm{MI}(\mathbf{y}, \mathbf{x}^\star) \quad \text{(Parallel ES)}$$

$$\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{X}\right] - \mathbb{E}_{\mathbf{x}^\star}\left[\mathrm{H}\left[\mathbf{y} \middle| \mathcal{D}_t, \mathbf{X}, \mathbf{x}^\star\right] \middle| \mathcal{D}_t, \mathbf{X}\right] \equiv \mathrm{MI}(\mathbf{x}^\star, \mathbf{y}) \quad \text{(Parallel PES)}$$

Multi-variate Gaussian distribution

Approximated by sampling from $p(\mathbf{x}^\star|\mathcal{D}_t)$

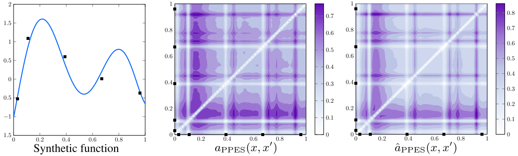
Multivariate Gaussian approximation with **expectation propagation**
$\mathbf{x}^\star$ is better than any other point in $\mathcal{X}$

$$\alpha(\mathbf{X}) = \log|\mathbf{V}^{\mathrm{PD}}(\mathbf{X})| - \frac{1}{M}\sum_{m=1}^{M}\log|\mathbf{V}^{\mathrm{CPD}}(\mathbf{X}|\mathbf{x}^\star_{(m)})|$$

It is possible to compute the gradient of $\alpha(\cdot)$ w.r.t. each $\mathrm{x}_q \in \mathcal{S}_t$!

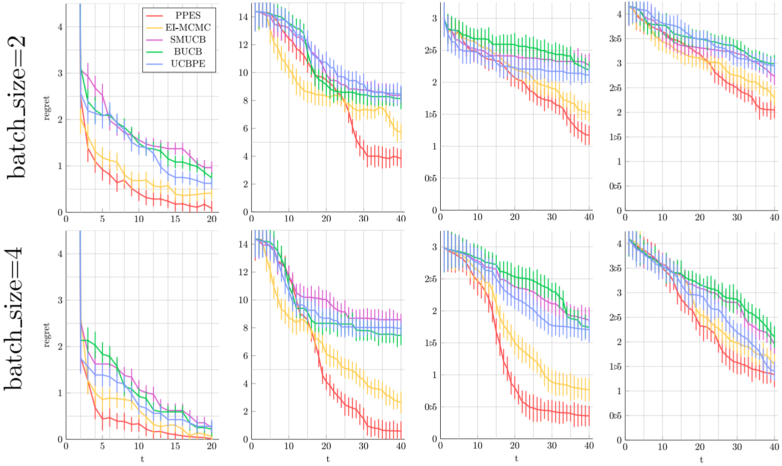
(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search: Level Curves



(Shah and Ghahramani, 2015)

Parallel Predictive Entropy Search: Results



(a) boston (b) hydrogen (c) rocket (d) robot

(Shah and Ghahramani, 2015)

BO with Integer-valued and Categorical Variables

Standard GPs assume continuous input variables which makes BO with integer-valued or categorical challenging.

BO with Integer-valued and Categorical Variables

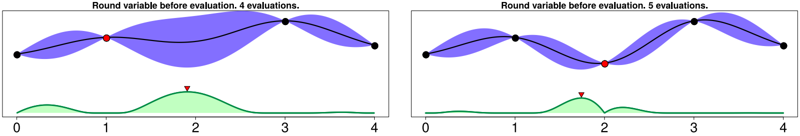
Standard GPs assume continuous input variables which makes BO with integer-valued or categorical challenging.

A naive approach is to round the suggested value to the closest integer or to the closest one-hot encoding.

BO with Integer-valued and Categorical Variables

Standard GPs assume continuous input variables which makes BO with integer-valued or categorical challenging.

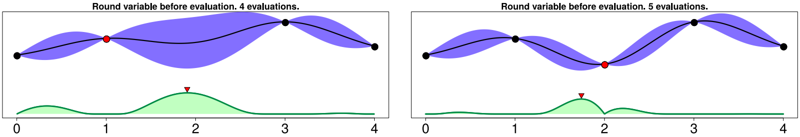
A naive approach is to round the suggested value to the closest integer or to the closest one-hot encoding.



BO with Integer-valued and Categorical Variables

Standard GPs assume continuous input variables which makes BO with integer-valued or categorical challenging.

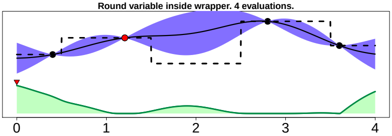
A naive approach is to round the suggested value to the closest integer or to the closest one-hot encoding.

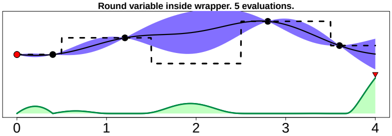


The BO algorithm may get stuck and may always perform the next evaluation at the same input location!

BO with Integer-valued and Categorical Variables

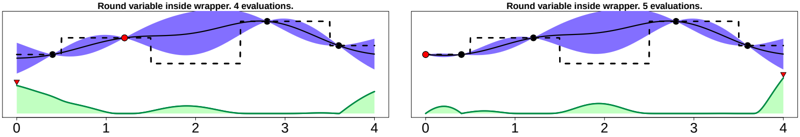
Rounding inside of the wrapper works but makes the objective flat!





BO with Integer-valued and Categorical Variables

Rounding inside of the wrapper works but makes the objective flat!



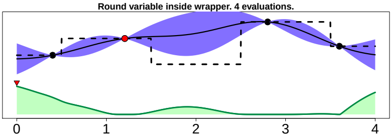
A modified GP covariance function accounts for this:

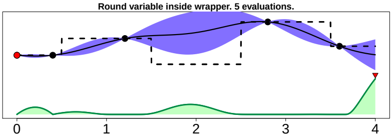
$$C_{\text{new}}(\mathrm{x}_n, \mathrm{x}_{n'}) = C(T(\mathrm{x}_n), T(\mathrm{x}_{n'}); \boldsymbol{\theta})$$

where $T(\cdot)$ does the rounding to the closest integer or one-hot encoding.

BO with Integer-valued and Categorical Variables

Rounding inside of the wrapper works but makes the objective flat!

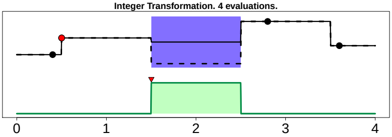


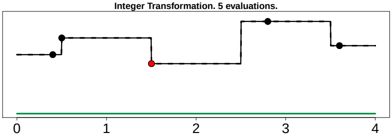


A modified GP covariance function accounts for this:

$$C_{\text{new}}(\mathrm{x}_n, \mathrm{x}_{n'}) = C(T(\mathrm{x}_n), T(\mathrm{x}_{n'}); \boldsymbol{\theta})$$

where $T(\cdot)$ does the rounding to the closest integer or one-hot encoding.



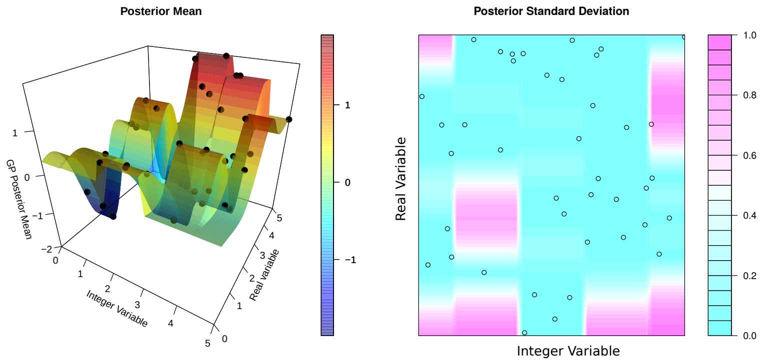


BO with Integer-valued and Categorical Variables

The GP predictive distribution is constant across all variables that lead to the same integer or one-hot-encoding.

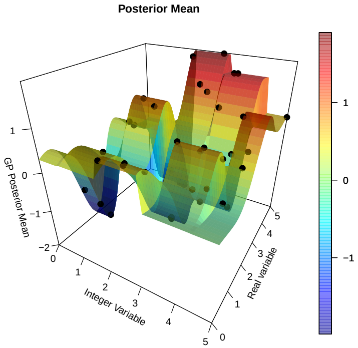
BO with Integer-valued and Categorical Variables

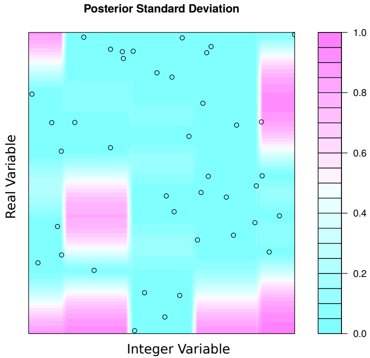
The GP predictive distribution is constant across all variables that lead to the same integer or one-hot-encoding.



BO with Integer-valued and Categorical Variables

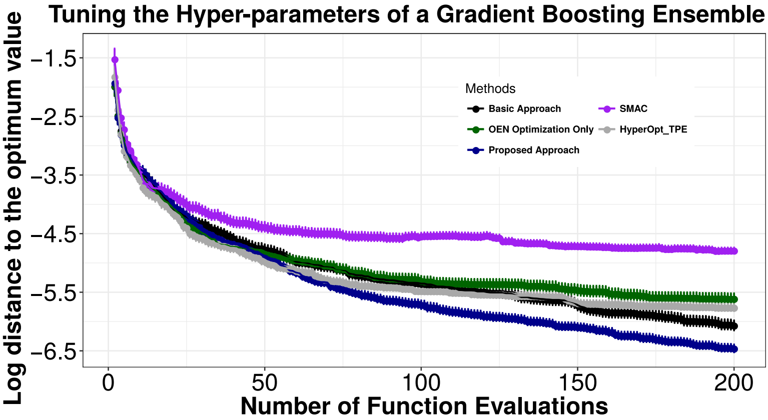
The GP predictive distribution is constant across all variables that lead to the same integer or one-hot-encoding.





Similar results for categorical variables!

BO with Integer-valued and Categorical Variables



One continuous variable and two integer-valued variables.

Freeze-Thaw Bayesian Optimization

Common aspects of many machine learning algorithms:

Freeze-Thaw Bayesian Optimization

Common aspects of many machine learning algorithms:

1. A minimization step must be performed with, *e.g.*, gradient descent.

Freeze-Thaw Bayesian Optimization

Common aspects of many machine learning algorithms:

1. A minimization step must be performed with, *e.g.*, gradient descent.

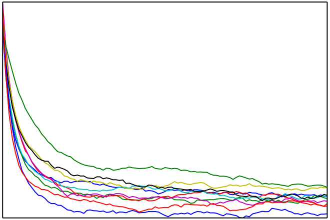
2. There are hyper-parameters that impact the final performance.

Freeze-Thaw Bayesian Optimization

Common aspects of many machine learning algorithms:

1. A minimization step must be performed with, *e.g.*, gradient descent.

2. There are hyper-parameters that impact the final performance.

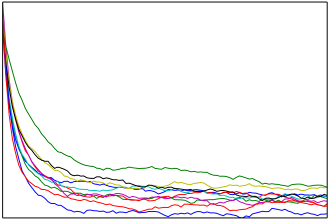


Freeze-Thaw Bayesian Optimization

Common aspects of many machine learning algorithms:

1. A minimization step must be performed with, *e.g.*, gradient descent.

2. There are hyper-parameters that impact the final performance.



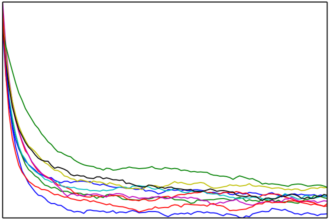
Can we use partial training information and a model to determine which hyper-parameter configuration is going to be optimal?

Freeze-Thaw Bayesian Optimization

Common aspects of many machine learning algorithms:

1. A minimization step must be performed with, *e.g.*, gradient descent.

2. There are hyper-parameters that impact the final performance.



Can we use partial training information and a model to determine which hyper-parameter configuration is going to be optimal?

Yes, that is precisely what Freeze-Thaw BO does!

(Swersky et al., 2014)

A GP Kernel for Training Curves

We want to specify a kernel that supports exponentially decaying functions of the form $\exp\{-\lambda t\}$ for $t, \lambda \geq 0$.

A GP Kernel for Training Curves

We want to specify a kernel that supports exponentially decaying functions of the form $\exp\{-\lambda t\}$ for $t, \lambda \geq 0$.

The covariance between inputs t and t' is:

$$C(t, t') = \int_0^\infty e^{-\lambda t} e^{-\lambda t'} \psi(\lambda; \alpha, \beta) d\lambda = \frac{\beta^\alpha}{(t + t' + \beta)^\alpha}$$

where $\psi(\lambda; \alpha, \beta)$ is a gamma distribution with parameters α and β.

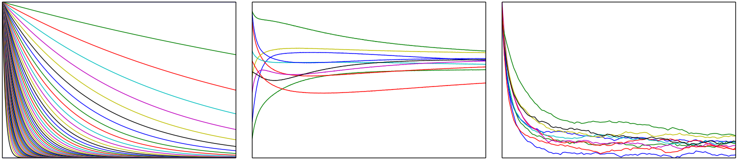
A GP Kernel for Training Curves

We want to specify a kernel that supports exponentially decaying functions of the form $\exp\{-\lambda t\}$ for $t, \lambda \geq 0$.

The covariance between inputs t and t' is:

$$C(t, t') = \int_0^\infty e^{-\lambda t} e^{-\lambda t'} \psi(\lambda; \alpha, \beta) d\lambda = \frac{\beta^\alpha}{(t + t' + \beta)^\alpha}$$

where $\psi(\lambda; \alpha, \beta)$ is a gamma distribution with parameters α and β.



(a) Exponential Decay Basis (b) Samples (c) Training Curve Samples

Inference on Asymptotic Values

A standard GP is used as the prior for the asymptotic values of each training curve.

Inference on Asymptotic Values

A standard GP is used as the prior for the asymptotic values of each training curve.

Hierarchical generative model:

$$p(\{\mathrm{y}_n\}_{n=1}^N | \{\mathrm{x}_n\}_{n=1}^N) = \int \left[\prod_{n=1}^N \mathcal{N}(\mathrm{y}_n | f_n \mathrm{1}, \mathrm{K}_{t_n}) \right] \mathcal{N}(\mathrm{f}|\mathrm{m}, \mathrm{K}_\mathrm{x}) d\mathrm{f}$$

where

$$\mathrm{x}_n \equiv n \text{ configuration}\,, \qquad \mathrm{y}_n \equiv n \text{ observed curve}\,,$$

$$f_n \equiv n \text{ asymptotic value}\,, \qquad \mathrm{m} \equiv \text{prior asymptotic mean values}\,,$$

$$\mathrm{K}_{t_n} \equiv \text{covariances for curve values}\,, \quad \mathrm{K}_\mathrm{x} \equiv \text{cov. for asymptotic values}$$

Inference on Asymptotic Values

A standard GP is used as the prior for the asymptotic values of each training curve.

Hierarchical generative model:

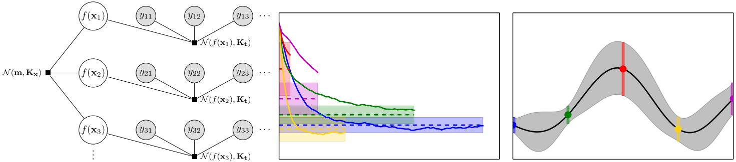
$$p(\{\mathrm{y}_n\}_{n=1}^N | \{\mathrm{x}_n\}_{n=1}^N) = \int \left[\prod_{n=1}^N \mathcal{N}(\mathrm{y}_n | f_n \mathrm{1}, \mathrm{K}_{t_n}) \right] \mathcal{N}(\mathrm{f} | \mathrm{m}, \mathrm{K}_\mathrm{x}) d\mathrm{f}$$

where

$$\begin{aligned} \mathrm{x}_n &\equiv n \text{ configuration}\,, & \mathrm{y}_n &\equiv n \text{ observed curve}\,, \\ f_n &\equiv n \text{ asymptotic value}\,, & \mathrm{m} &\equiv \text{prior asymptotic mean values}\,, \\ \mathrm{K}_{t_n} &\equiv \text{covariances for curve values}\,, & \mathrm{K}_\mathrm{x} &\equiv \text{cov. for asymptotic values} \end{aligned}$$

The joint distribution of $\{\mathrm{y}\}_{n=1}^N$ and f is Gaussian and hence so it is the predictive distribution $p(\mathrm{f}|\{\mathrm{y}\}_{n=1}^N)$!

Inference on Asymptotic Values and BO

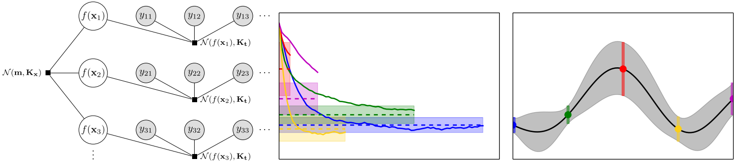


(a) Graphical Model

(b) Training curve predictions

(c) Asymptotic GP

Inference on Asymptotic Values and BO



(a) Graphical Model

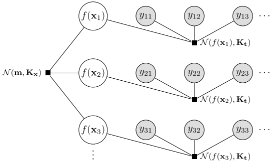
(b) Training curve predictions

(c) Asymptotic GP

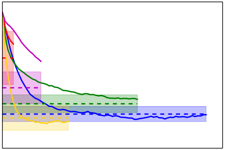
Bayesian Optimization:

- $p(\mathrm{f} | \{\mathrm{y}_n\}_{n=1}^N, \{\mathrm{x}_n\}_{n=1}^N)$ determines asymptotic values.

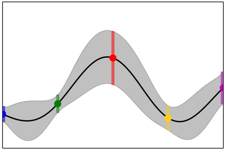
Inference on Asymptotic Values and BO



(a) Graphical Model



(b) Training curve predictions

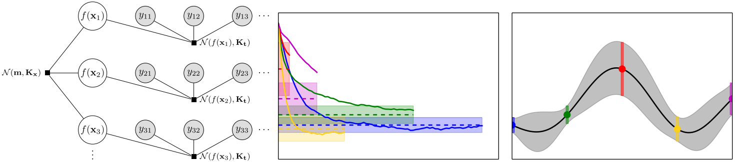


(c) Asymptotic GP

Bayesian Optimization:

- $p(\mathrm{f} | \{\mathrm{y}_n\}_{n=1}^N, \{\mathrm{x}_n\}_{n=1}^N)$ determines asymptotic values.
- This distribution can be used to make intelligent decisions!

Inference on Asymptotic Values and BO

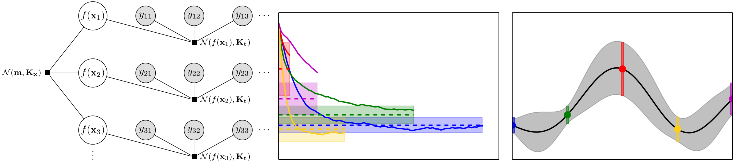


(a) Graphical Model (b) Training curve predictions (c) Asymptotic GP

Bayesian Optimization:

- $p(\mathrm{f}|\{\mathrm{y}_n\}_{n=1}^N, \{\mathrm{x}_n\}_{n=1}^N)$ determines asymptotic values.
- This distribution can be used to make intelligent decisions!
- Shall we train more one configuration or shall we start a new one?

Inference on Asymptotic Values and BO



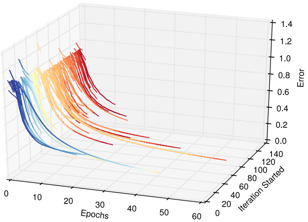
(a) Graphical Model (b) Training curve predictions (c) Asymptotic GP

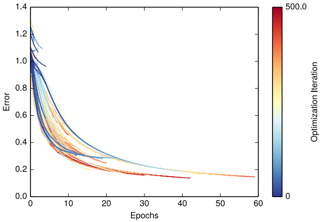
Bayesian Optimization:

- $p(\mathrm{f} | \{\mathrm{y}_n\}_{n=1}^N, \{\mathrm{x}_n\}_{n=1}^N)$ determines asymptotic values.
- This distribution can be used to make intelligent decisions!
- Shall we train more one configuration or shall we start a new one?
- A combination of EI and ES is used as the acquisition function.

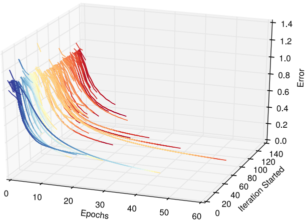
(Swersky et al., 2014)

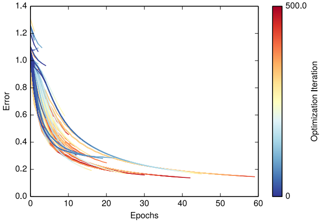
Freeze-Thaw BO in practice

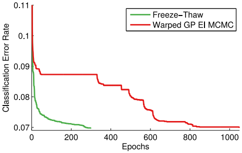




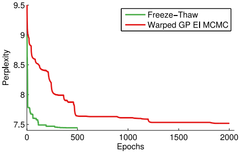
Freeze-Thaw BO in practice



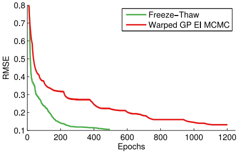




(a) Logistic Regression



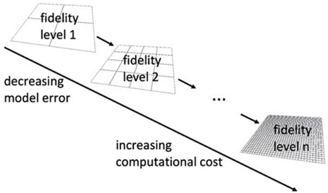
(b) Online LDA

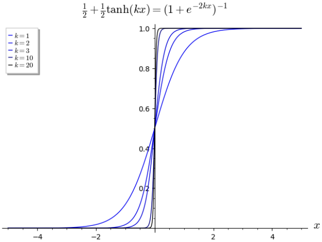


(c) PMF

(Swersky et al., 2014)

Multi-fidelity Bayesian optimization





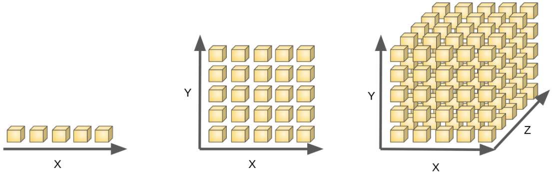
- **Concept of Fidelity:** Fidelity refers to the accuracy or reliability of the information, we use different levels of accuracy.
- **Example:** # layers in NNs, # of timesteps for DRL.
- *We assume that lower fidelities are correlated with higher.*
- **Cost-Efficiency:** Leveraging computationally cheaper versions of the functions to guide the search process.
- **The trick:** The acquisition function should balance exploration at cheaper fidelities with exploitation at the highest fidelity level.

Multi-fidelity Bayesian optimization example

- Trace-aware knowledge-gradient acquisition function.
- It values observations of a point (x) and a set of fidelities ($\mathcal{S}$) according to the ratio of the reduction in expected loss that it induces, to its computational cost.
- It measures the value of information per unit cost of sampling.
- It uses a function $L()$ to measure the extent to which observing trace information improves the quality of the solution. $L(0)$ will be the minimum.
- Its analytical expression basically penalizes the cost wrt the information obtained:

$$takg(x, \mathcal{S}) = \frac{L(0) - L(x, \mathcal{S})}{cost(x, max(\mathcal{S}))}$$

High-dimensional Bayesian optimization



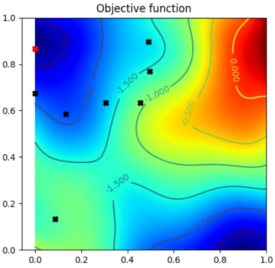
- GPs empirical performance tends to be lower if $d > 7$.
- **The problem:** The search space grows exponentially with the number of dimensions.
- **The trick:** We can project the high-dimensional problem into a lower-dimensional subspace that explains it well using embeddings.
- *Find the hidden most explicative manifold for the data, then optimize there!*
- Approaches differ in the type of embeddings (e.g. random), assumptions about the function (e.g. being a sum of functions), or use of models (Deep GP, GP-LVM.)

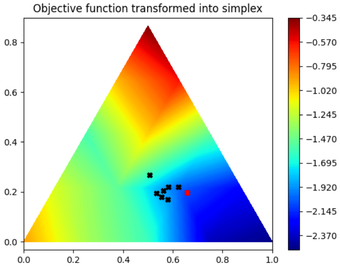
High-dimensional Bayesian optimization example

- Sparse Axis-Aligned Subspace Bayesian Optimization (SAASBO)
- **Goal:** Identify sparse subspaces relevant to modeling the unknown objective function.
- **Assumption:** High function variability being captured by axis-aligned blocks of input features.
- **Method:** Use complex GP prior to consider a smaller class of functions.
- **Effect:** Turn most non-explicative dims off,
- **Integration:** Perform hyper-parameter sampling with NUTS and EI.

[kernel variance]	$\sigma_k^2 \sim \mathcal{LN}(0, 10^2)$
[global shrinkage]	$\tau \sim \mathcal{HC}(\alpha)$
[length scales]	$\rho_i \sim \mathcal{HC}(\tau)$
[function values]	$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, K^{\psi}_{\mathbf{XX}})$
[observations]	$\mathbf{y} \sim \mathcal{N}(\mathbf{f}, \sigma^2 \mathbb{1}_N)$

Bayesian optimization in a simplex (Portfolio optimization)

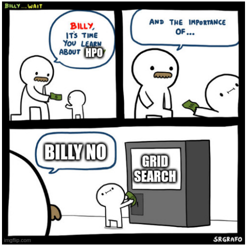




- Common financial metrics as Sharpe or Sortino ratio can be configured for a specific portfolio.
- For example using real-time ESG values of the assets.
- Garrido-Merchán, E. C., Piris, G. G., & Vaca, M. C. (2023). Bayesian optimization of ESG (Environmental Social Governance) financial investments. Environmental Research Communications, 5(5), 055003.

Conclusions and Further Work

- BO is a state-of-the-art class of methods used to optimize expensive and noisy black-box functions.
- We can generalize BO to tackle advanced scenarios: parallel constrained multi-obj, high-dim, multi-fidelity...
- BO can be applied in a wide array of applications: ML, DRL, finance, robotics, materials, business operations...
- Further work topics: causality, meta-BO, transfer learning, adaptation to specific domains.



Referencias

- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive Entropy Search for Multi-Objective Bayesian Optimization with Constraints. Neurocomputing, 361, 50-68.

Referencias

- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive Entropy Search for Multi-Objective Bayesian Optimization with Constraints. Neurocomputing, 361, 50-68.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2023). Parallel Predictive Entropy Search for Multi-objective Bayesian Optimization with Constraints. Expert Systems with Applications, pre-print.

Referencias

- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive Entropy Search for Multi-Objective Bayesian Optimization with Constraints. Neurocomputing, 361, 50-68.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2023). Parallel Predictive Entropy Search for Multi-objective Bayesian Optimization with Constraints. Expert Systems with Applications, pre-print.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2020). Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes. Neurocomputing, 380, 20-35.

Referencias

- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive Entropy Search for Multi-Objective Bayesian Optimization with Constraints. Neurocomputing, 361, 50-68.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2023). Parallel Predictive Entropy Search for Multi-objective Bayesian Optimization with Constraints. Expert Systems with Applications, pre-print.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2020). Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes. Neurocomputing, 380, 20-35.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2018). Bayesian Optimization of a Hybrid System for Robust Ocean Wave Features Prediction. Neurocomputing, 275, 818-828.

Referencias

- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive Entropy Search for Multi-Objective Bayesian Optimization with Constraints. Neurocomputing, 361, 50-68.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2023). Parallel Predictive Entropy Search for Multi-objective Bayesian Optimization with Constraints. Expert Systems with Applications, pre-print.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2020). Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes. Neurocomputing, 380, 20-35.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2018). Bayesian Optimization of a Hybrid System for Robust Ocean Wave Features Prediction. Neurocomputing, 275, 818-828.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2017, June). Bayesian optimization of a hybrid prediction system for optimal wave energy estimation problems. In International Work-Conference on Artificial Neural Networks (pp. 648-660). Springer, Cham.

Referencias

- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive Entropy Search for Multi-Objective Bayesian Optimization with Constraints. Neurocomputing, 361, 50-68.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2023). Parallel Predictive Entropy Search for Multi-objective Bayesian Optimization with Constraints. Expert Systems with Applications, pre-print.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2020). Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes. Neurocomputing, 380, 20-35.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2018). Bayesian Optimization of a Hybrid System for Robust Ocean Wave Features Prediction. Neurocomputing, 275, 818-828.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2017, June). Bayesian optimization of a hybrid prediction system for optimal wave energy estimation problems. In International Work-Conference on Artificial Neural Networks (pp. 648-660). Springer, Cham.
- Balázs, C., van Beekveld, M., Caron, S., Dillon, B. M., Farmer, B., Fowlie, A., Garrido-Merchán, E. C., ... & White, M. (2021). A Comparison of Optimisation Algorithms for High-Dimensional Particle and Astrophysics Applications. Journal of High Energy Physics, 2021(5), 1-46.

Referencias

- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive Entropy Search for Multi-Objective Bayesian Optimization with Constraints. Neurocomputing, 361, 50-68.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2023). Parallel Predictive Entropy Search for Multi-objective Bayesian Optimization with Constraints. Expert Systems with Applications, pre-print.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2020). Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes. Neurocomputing, 380, 20-35.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2018). Bayesian Optimization of a Hybrid System for Robust Ocean Wave Features Prediction. Neurocomputing, 275, 818-828.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2017, June). Bayesian optimization of a hybrid prediction system for optimal wave energy estimation problems. In International Work-Conference on Artificial Neural Networks (pp. 648-660). Springer, Cham.
- Balázs, C., van Beekveld, M., Caron, S., Dillon, B. M., Farmer, B., Fowlie, A., Garrido-Merchán, E. C., ... & White, M. (2021). A Comparison of Optimisation Algorithms for High-Dimensional Particle and Astrophysics Applications. Journal of High Energy Physics, 2021(5), 1-46.
- Córdoba, I., Garrido-Merchán, E. C., Hernández-Lobato, D., Bielza, C., & Larranaga, P. (2018, October). Bayesian Optimization of the PC Algorithm for Learning Gaussian Bayesian Networks. In Conference of the Spanish Association for Artificial Intelligence (pp. 44-54). Springer, Cham.

Referencias

- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive Entropy Search for Multi-Objective Bayesian Optimization with Constraints. Neurocomputing, 361, 50-68.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2023). Parallel Predictive Entropy Search for Multi-objective Bayesian Optimization with Constraints. Expert Systems with Applications, pre-print.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2020). Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes. Neurocomputing, 380, 20-35.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2018). Bayesian Optimization of a Hybrid System for Robust Ocean Wave Features Prediction. Neurocomputing, 275, 818-828.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2017, June). Bayesian optimization of a hybrid prediction system for optimal wave energy estimation problems. In International Work-Conference on Artificial Neural Networks (pp. 648-660). Springer, Cham.
- Balázs, C., van Beekveld, M., Caron, S., Dillon, B. M., Farmer, B., Fowlie, A., Garrido-Merchán, E. C., ... & White, M. (2021). A Comparison of Optimisation Algorithms for High-Dimensional Particle and Astrophysics Applications. Journal of High Energy Physics, 2021(5), 1-46.
- Córdoba, I., Garrido-Merchán, E. C., Hernández-Lobato, D., Bielza, C., & Larranaga, P. (2018, October). Bayesian Optimization of the PC Algorithm for Learning Gaussian Bayesian Networks. In Conference of the Spanish Association for Artificial Intelligence (pp. 44-54). Springer, Cham.
- Garrido-Merchán, E. C., & Albarca-Molina, A. (2018, November). Suggesting Cooking Recipes through Simulation and Bayesian Optimization. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 277-284). Springer, Cham.

Referencias

- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive Entropy Search for Multi-Objective Bayesian Optimization with Constraints. Neurocomputing, 361, 50-68.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2023). Parallel Predictive Entropy Search for Multi-objective Bayesian Optimization with Constraints. Expert Systems with Applications, pre-print.
- Garrido-Merchán, E. C., & Hernández-Lobato, D. (2020). Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes. Neurocomputing, 380, 20-35.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2018). Bayesian Optimization of a Hybrid System for Robust Ocean Wave Features Prediction. Neurocomputing, 275, 818-828.
- Cornejo-Bueno, L., Garrido-Merchán, E. C., Hernández-Lobato, D., & Salcedo-Sanz, S. (2017, June). Bayesian optimization of a hybrid prediction system for optimal wave energy estimation problems. In International Work-Conference on Artificial Neural Networks (pp. 648-660). Springer, Cham.
- Balázs, C., van Beekveld, M., Caron, S., Dillon, B. M., Farmer, B., Fowlie, A., Garrido-Merchán, E. C., ... & White, M. (2021). A Comparison of Optimisation Algorithms for High-Dimensional Particle and Astrophysics Applications. Journal of High Energy Physics, 2021(5), 1-46.
- Córdoba, I., Garrido-Merchán, E. C., Hernández-Lobato, D., Bielza, C., & Larranaga, P. (2018, October). Bayesian Optimization of the PC Algorithm for Learning Gaussian Bayesian Networks. In Conference of the Spanish Association for Artificial Intelligence (pp. 44-54). Springer, Cham.
- Garrido-Merchán, E. C., & Albarca-Molina, A. (2018, November). Suggesting Cooking Recipes through Simulation and Bayesian Optimization. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 277-284). Springer, Cham.
- Jariego-Pérez, L. C., & Garrido-Merchán, E. C. (2020). Towards Automatic Bayesian Optimization: A first step involving acquisition functions. CAEPIA. 2021. Accepted.

Thank you for your attention.