A tutorial on Bayesian Optimization

Daniel Hernández–Lobato
Computer Science Department
Universidad Autónoma de Madrid

http://dhnzl.org, daniel.hernandez@uam.es
Challenges in Engineering Design

The society demands new products of better quality, functionality, usability, etc.!
Challenges in Engineering Design

The society demands new products of better quality, functionality, usability, etc.!
Challenges in Engineering Design

The society demands new products of better quality, functionality, usability, etc.!

- Many choices at each step.
Challenges in Engineering Design

The society demands new products of better quality, functionality, usability, etc.!

- Many choices at each step.
- Complicated and high dimensional.
Challenges in Engineering Design

The society demands new products of better quality, functionality, usability, etc.!

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

The society demands new products of better quality, functionality, usability, etc.!

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

The society demands new products of better quality, functionality, usability, etc.!

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

Optimization is a challenging task in new products design!
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: 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.

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

\[ y = f(x) + \epsilon \]
Optimization Problems: Common Features

- Very expensive evaluations.

- The objective is a black-box.

- The evaluation can be noisy.

Bayesian optimization methods can be used to solve these problems!
Bayesian Optimization in Practice

1. Get initial sample.
Bayesian Optimization in Practice

Get initial sample.
1 Get initial sample.

2 Fit a model to the data:

\[ p(y|x, \mathcal{D}_n). \]
1. Get initial sample.

2. Fit a model to the data:

\[ p(y|x, \mathcal{D}_n) \].

Bayesian Optimization in Practice
Bayesian Optimization in Practice

1. Get initial sample.

2. **Fit a model to the data:**
   \[ p(y|\mathbf{x}, \mathcal{D}_n). \]

3. **Select data collection strategy:**
   \[ \alpha(\mathbf{x}) = \mathbb{E}_{p(y|\mathbf{x}, \mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]. \]
Bayesian Optimization in Practice

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

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

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

1. Get initial sample.
2. Fit a model to the data:
   \[ p(y|x, D_n). \]
3. Select data collection strategy:
   \[ \alpha(x) = \mathbb{E}_{p(y|x, D_n)}[U(y|x, D_n)]. \]
4. Optimize acquisition function \( \alpha(x) \).
5. Collect data and update model.
6. Repeat!
**Bayesian Optimization in Practice**

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

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

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

1. Get initial sample.
2. Fit a model to the data:
   \[ p(y|x, D_n) \].
3. Select data collection strategy:
   \[ \alpha(x) = E_{p(y|x, D_n)}[U(y|x, D_n)] \].
4. Optimize acquisition function \( \alpha(x) \).
5. Collect data and update model.
6. Repeat!
1. Get initial sample.
2. Fit a model to the data:
   \[ p(y|x, D_n) \].
3. Select data collection strategy:
   \[ \alpha(x) = E_{p(y|x, D_n)}[U(y|x, D_n)] \].
4. Optimize acquisition function \( \alpha(x) \).
5. Collect data and update model.
6. Repeat!
1. Get initial sample.
2. Fit a model to the data:  
   \[ p(y|x, D_n) . \]
3. Select data collection strategy:  
   \[ \alpha(x) = E_{p(y|x, D_n)}[U(y|x, D_n)] . \]
4. Optimize acquisition function \( \alpha(x) \).
5. Collect data and update model.
6. Repeat!
Bayesian Optimization in Practice

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

2 Fit a model to the data:

\[ p(y|x, D_n) \].

3 Select data collection strategy:

\[ \alpha(x) = E_{p(y|x,D_n)}[U(y|x, D_n)] \].

4 Optimize acquisition function \( \alpha(x) \).

5 Collect data and update model.

6 Repeat!
Bayesian Optimization in Practice

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

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

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

1. Get initial sample.
2. Fit a model to the data:
   $$p(y|x, D_n).$$
3. Select data collection strategy:
   $$\alpha(x) = E_{p(y|x, D_n)}[U(y|x, D_n)].$$
4. Optimize acquisition function $$\alpha(x).$$
5. Collect data and update model.
6. Repeat!
Bayesian Optimization vs. Uniform Exploration

Tuning LDA on a collection of Wikepida articles (Snoek et al., 2012).
Fitting a Model to the Data
Fitting a Model to the Data

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

\[ f(x) = \sum_{j=1}^{H} v_j h_j(x) \]
Fitting a Model to the Data

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

\[ f(x) = \sum_{j=1}^{H} v_j h_j(x) \]
Fitting a Model to the Data

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

\[ f(x) = \sum_{j=1}^{H} v_j h_j(x) \]
Fitting a Model to the Data

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

\[ f(x) = \sum_{j=1}^{H} v_j h_j(x) \]

**Posterior Dist.**

\[ p(W|\text{Data}) = \frac{p(W)p(\text{Data}|W)}{p(\text{Data})} \]

**Predictive Dist.**

\[ p(y|\text{Data}, x) = \int p(y|W, x)p(W|\text{Data})dW \]
Fitting a Model to the Data

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

$\begin{align*}
  h_j(x) &= \tanh \left( \sum_{i=1}^{I} x_i w_{ji} \right) \\
  f(x) &= \sum_{j=1}^{H} v_j h_j(x)
\end{align*}$

Posterior Dist. \hspace{1cm} p(W|\text{Data}) = \frac{p(W)p(\text{Data}|W)}{p(\text{Data})}$

Predictive Dist. \hspace{1cm} p(y|\text{Data}, x) = \int p(y|W, x)p(W|\text{Data})dW$
Fitting a Model to the Data

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

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

Distribution over functions $f(\cdot)$ so that for any finite \(\{x_i\}_{i=1}^N\), \((f(x_1), \ldots, f(x_N))^T\) follows an $N$-dimensional Gaussian distribution.
Gaussian Processes

Distribution over functions $f(\cdot)$ so that for any finite $\{x_i\}_{i=1}^N$, $(f(x_1), \ldots, f(x_N))^T$ follows an $N$-dimensional Gaussian distribution.
Gaussian Processes

Distribution over functions $f(\cdot)$ so that for any finite $\{x_i\}_{i=1}^N$, $(f(x_1), \ldots, f(x_N))^T$ follows an $N$-dimensional Gaussian distribution.

When $H \to \infty$, $(f(x_1), \ldots, f(x_N))^T$ follows an $N$-dimensional Gaussian where $\mathbb{E}[f(x_i)f(x_k)] = \sigma^2 \mathbb{E}[h_j(x_i)h_j(x_k)]$ by the central limit theorem.
Gaussian Processes

Distribution over functions $f(\cdot)$ so that for any finite $\{x_i\}_{i=1}^N$, $(f(x_1), \ldots, f(x_N))^T$ follows an $N$-dimensional Gaussian distribution.

When $H \to \infty$, $(f(x_1), \ldots, f(x_N))^T$ follows an $N$-dimensional Gaussian where $\mathbb{E}[f(x_i)f(x_k)] = \sigma^2\mathbb{E}[h_j(x_i)h_j(x_k)]$ by the central limit theorem.

Due to Gaussian form, there are closed-form solutions for many useful questions about finite data.
Gaussian Processes

- The **joint distribution** for \( y^* \) at test points \( \{x^*_m\}_{m=1}^M \) and \( y \):

\[
p(y^*, y) = \mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_\theta & K_\theta \\ \kappa_\theta & k_\theta^T \end{bmatrix} \right)
\]
Gaussian Processes

- The **joint distribution** for \( y^* \) at test points \( \{x_m^*\}_{m=1}^M \) and \( y \):

\[
p(y^*, y) = \mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_\theta & K_\theta \\ \kappa_\theta & k_\theta^T \end{bmatrix} \right)
\]

- These **matrices** are computed from the covariance \( C(\cdot, \cdot; \theta) \):

\[
\begin{align*}
[K_\theta]_{n,n'} &= C(x_n, x_{n'}; \theta) \\
[k_\theta]_{n,m} &= C(x_n, x_m^*; \theta) \\
[\kappa_\theta]_{m,m'} &= C(x_m^*, x_m'^*; \theta)
\end{align*}
\]
Gaussian Processes

- The joint distribution for \( y^* \) at test points \( \{x^*_m\}_{m=1}^M \) and \( y \):

\[
p(y^*, y) = \mathcal{N}\left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_\theta & K_\theta \\ \kappa_\theta & k_\theta^T \end{bmatrix} \right)
\]

- These matrices are computed from the covariance \( C(\cdot, \cdot; \theta) \):

\[
[K_\theta]_{n,n'} = C(x_n, x_{n'}; \theta) \\
k_{\theta}]_{n,m} = C(x_n, x^*_m; \theta), \quad [\kappa_\theta]_{m,m'} = C(x^*_m, x^*_{m'}; \theta),
\]

- The predictive distribution for \( y^* \) given \( y \), \( p(y^*|y) \), is:

\[
y^* \sim \mathcal{N}(m, \Sigma)
\]

\[
m = k_\theta^T K_\theta^{-1} y, \quad \Sigma = \kappa_\theta - k_\theta^T K_\theta^{-1} k_\theta,
\]
Gaussian Processes

- The **joint distribution** for \(y^*\) at test points \(\{x_m^*\}_{m=1}^M\) and \(y\):

\[
p(y^*, y) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_\theta & K_\theta \\ k_\theta^T & \kappa_\theta \end{bmatrix}\right)
\]

- These **matrices** are computed from the covariance \(C(\cdot, \cdot; \theta)\):

\[
[K_\theta]_{n,n'} = C(x_n, x_{n'}; \theta)
\]

\[
[k_\theta]_{n,m} = C(x_n, x_m^*; \theta), \quad [\kappa_\theta]_{m,m'} = C(x_m^*, x_m^{*'}; \theta),
\]

- The **predictive distribution** for \(y^*\) given \(y\), \(p(y^*|y)\), is:

\[
y^* \sim \mathcal{N}(m, \Sigma)
\]

\[
m = k_\theta^T K_\theta^{-1} y, \quad \Sigma = \kappa_\theta - k_\theta^T K_\theta^{-1} k_\theta,
\]

- The log of the **marginal likelihood**, \(p(y|\theta)\), is:

\[
\log p(y) = -\frac{N}{2} \log 2\pi - \frac{1}{2} \log |K_\theta| - \frac{1}{2} y^T K_\theta^{-1} y
\]
Some Covariance Functions

Squared Exponential

\[ C(x, x') = \sigma^2 \exp \left\{ \frac{1}{2} \sum_j \left( \frac{x_j - x'_j}{l_j} \right)^2 \right\} \]
Some Covariance Functions

Squared Exponential

\[ C(x, x') = \sigma^2 \exp \left\{ \frac{1}{2} \sum_j \left( \frac{x_j - x'_j}{l_j} \right)^2 \right\} \]

Matérn

\[ C(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu r}}{l} \right)^\nu K_\nu \left( \frac{\sqrt{2\nu r}}{l} \right) \]
Some Covariance Functions

Squared Exponential

$$C(x, x') = \sigma^2 \exp \left\{ \frac{1}{2} \sum_j \left( \frac{x_j - x_j'}{l_j} \right)^2 \right\}$$

Matérn

$$C(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu r}}{l} \right)^\nu K_\nu \left( \frac{\sqrt{2\nu r}}{l} \right)$$

Neural Network

$$C(x, x') = \frac{2}{\pi} \sin^{-1} \left( \frac{2x^T \Sigma x'}{\sqrt{(1+2x^T \Sigma x)(1+2x'^T \Sigma x')}} \right)$$
Some Covariance Functions

Squared Exponential

\[ C(x, x') = \sigma^2 \exp \left\{ \frac{1}{2} \sum_j \left( \frac{x_j - x'_j}{l_j} \right)^2 \right\} \]

Matérn

\[ C(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu r}}{l} \right)^\nu K_\nu \left( \frac{\sqrt{2\nu r}}{l} \right) \]

Neural Network

Periodic

\[ C(x, x') = \frac{2}{\pi} \sin^{-1} \left( \frac{2x^T \Sigma x'}{\sqrt{(1+2x^T \Sigma x)(1+2x'^T \Sigma x')}} \right) \]

\[ C(x, x') = \exp \left\{ -\frac{2 \sin^2 \left( \frac{|x - x'|}{2} \right)}{l^2} \right\} \]
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 
GP regression provides a \textbf{closed-form} posterior distribution for $f(\cdot)$. 

\textbf{Ground Truth}
GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 
GP regression provides a \textbf{closed-form} posterior distribution for $f(\cdot)$. 
GP regression provides a *closed-form* posterior distribution for $f(\cdot)$. 

**Ground Truth**
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 

![Graph showing a comparison between the ground truth and the GP regression prediction.](image-url)
From the Prior to the Posterior

GP regression provides a \textbf{closed-form} posterior distribution for $f(\cdot)$.
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 

![Diagram showing prediction intervals and ground truth](image_url)
From the Prior to the Posterior

GP regression provides a closed-form posterior distribution for $f(\cdot)$. 
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 
GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 

![Diagram showing the relationship between the prior and posterior distributions in GP regression. The red line represents the ground truth, while the black line depicts the posterior distribution. The shaded area illustrates the uncertainty.](image-url)
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 

![Diagram showing function approximation with ground truth and posterior distributions.](image-url)
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for \( f(\cdot) \).
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 

![Graph showing the regression and ground truth curves](chart.png)
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$.
GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 

![Ground Truth](image.png)
From the Prior to the Posterior

GP regression provides a **closed-form** posterior distribution for $f(\cdot)$. 
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!

\[ \alpha(x) = \mathbb{E}_{p(y^*|D_N,x)} [U(y^*|x, D_N)] \]
Some Acquisition Functions

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

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

- **Probability of Improvement:**

  $$U(y^*|D_N, x) = \mathbb{I}(y^* < \nu), \quad \alpha(x) = \Phi(\gamma(x))$$
Some Acquisition Functions

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

- **Probability of Improvement:**
  $$U(y^*|\mathcal{D}_N, x) = \mathbb{I}(y_* < \nu), \quad \alpha(x) = \Phi(\gamma(x))$$

- **Expected Improvement:**
  $$U(y^*|\mathcal{D}_N, x) = \max(0, \nu - y^*), \quad \alpha(x) = \sigma(x) \left( \gamma(x) \Phi(\gamma(x)) + \phi(\gamma(x)) \right)$$
Some Acquisition Functions

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

- **Probability of Improvement:**

  \[
  U(y^*|D_N, x) = \mathbb{I}(y_* < \nu), \quad \alpha(x) = \Phi(\gamma(x))
  \]

- **Expected Improvement:**

  \[
  U(y^*|D_N, x) = \max(0, \nu - y^*), \quad \alpha(x) = \sigma(x) (\gamma(x)\Phi(\gamma(x)) + \phi(\gamma(x)))
  \]

- **Lower Confidence Bound:**

  \[
  \alpha(x) = - (\mu(x) - \kappa \sigma^2(x))
  \]
Some Acquisition Functions

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

- **Probability of Improvement:**
  $$U(y^*|\mathcal{D}_N, x) = \mathbb{I}(y_\star < \nu), \quad \alpha(x) = \Phi(\gamma(x))$$

- **Expected Improvement:**
  $$U(y^*|\mathcal{D}_N, x) = \max(0, \nu - y^*), \quad \alpha(x) = \sigma(x)(\gamma(x)\Phi(\gamma(x)) + \phi(\gamma(x)))$$

- **Lower Confidence Bound:**
  $$\alpha(x) = -\left(\mu(x) - \kappa\sigma^2(x)\right)$$

- **Entropy Search:**
  $$U(y^*|\mathcal{D}_N, x) = H[p(x_{\text{min}}|\mathcal{D}_N)] - H[p(x_{\text{min}}|\mathcal{D}_N \cup \{x, y^*\})]$$
Some Acquisition Functions:
Some Acquisition Functions: Prob. Improvement
Some Acquisition Functions: Exp. Improvement
Some Acquisition Functions: Entropy Search
B\text{ayesian Optimization and Model Selection}

- \textbf{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 $p(y|\theta)$ can give too confident uncertainty estimates.
Bayesian Optimization and Model Selection

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

- **Sampling the hyper-parameters:** computing $p(\theta|y)$ is **intractable!** Alternative: generate a few samples form $p(\theta|y)$ using MCMC.
Bayesian Optimization and Model Selection

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

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

*Slice sampling means no additional hyper-parameters!*
Bayesian Optimization and Model Selection

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

- **Sampling the hyper-parameters:** computing $p(\theta|y)$ is intractable! Alternative: generate a few samples from $p(\theta|y)$ 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 $p(y|\theta)$ can give **too confident** uncertainty estimates.

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

**Slice sampling means no additional hyper-parameters!**

\[ \tilde{p}(\theta) \]

\[ \theta_{\text{min}} \quad u \quad \theta_{\text{max}} \]

\[ \theta^{(\tau)} \quad \theta \]

(Neal, 2003)
Integrated Acquisition Function

\[ \hat{\alpha}(x) = \int \alpha(x; \theta)p(\theta|y) \, d\theta \approx \frac{1}{K} \sum_{k=1}^{K} \alpha(x; \theta^{(k)}) \quad \theta^{(k)} \sim p(\theta|y), \]
Integrated Acquisition Function

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

Posterior samples with three different length-scales

(Snoek et al., 2012)
Integrated Acquisition Function

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

(Snoek et al., 2012)
Integrated Acquisition Function

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

(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:**

\[
\alpha(x) = \frac{\sigma(x) (\gamma(x) \Phi (\gamma(x)) + \phi(\gamma(x)))}{\exp \{\mu_{\log\text{-}time}(x)\}}
\]

(Snoek et al., 2012)
Cost-sensitive Bayesian Optimization
Cost-sensitive Bayesian Optimization
Cost-sensitive Bayesian Optimization
Cost-sensitive Bayesian Optimization

\[ f(x) \]

\[ EI(x) \]

\[ \frac{duration(x)}{EI(x)} \cdot s \]
Cost-sensitive Bayesian Optimization
Cost-sensitive Bayesian Optimization

\[ f(x) \]
\[ \text{EI}(x) \]
\[ \text{duration}(x) / s \]
\[ \text{EI}(x) / s \]
Cost-sensitive Bayesian Optimization

\[ f(x) \]

\[ EI(x) \]

\[ \frac{EI(x)}{s} \]

\[ \text{duration}(x) \]
Deep neural network on the CIFAR dataset (Snoek et al., 2012)
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*.

![Diagram of neural network and hardware accelerator](image)
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

Objective 1

Objective 2

Pareto Set (Input space)
Constrained Multi-Objective Optimization

Objective 1

Objective 2

Pareto Set (Input space)

Pareto Frontier (value space)

Values for Domain Points

Values for Optimal Points

Pareto Points
Constrained Multi-Objective Optimization

Objective 1

Objective 2

Constraint 1

Pareto Frontier (value space)

Pareto Set (input space)
Constrained Multi-Objective Optimization

Objective 1

Objective 2

Constraint 1

Pareto Set (Input space)

Pareto Frontier (value space)
Constrained Multi-Objective Optimization

Objective 1

Objective 2

Constraint 1

Pareto Set (Input space)

Pareto Frontier (value space)

Values for Domain Points

Values for Optimal Points

Pareto Points
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**

\[
\begin{align*}
  \text{Black-box 1} & \rightarrow Y_t^1 \\
  \text{Black-box 2} & \rightarrow Y_t^2 \\
  x_t & \rightarrow \\
\end{align*}
\]
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.
Information-based Approach

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

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

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

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

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

![Diagram showing actual objectives and constraints with $\mathcal{X}^*$ marked.]
Information-based Approach

The Pareto set $\mathcal{X}^*$ in the feasible space is a random variable! Information is measured by the entropy of $p(\mathcal{X}^*|\mathcal{D}_N)$.
Information-based Approach

The Pareto set $\mathcal{X}^*$ in the feasible space is a random variable! Information is measured by the entropy of $p(\mathcal{X}^*|D_N)$.
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 $\mathcal{X}^*$ in the feasible space is a random variable!

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

---

**Actual Objectives and Constraints**

**Posterior of each Objective and Constraint**

**Optimized Samples Drawn from the Posterior**
Information-based Approach

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

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

![Graph showing the actual objectives and constraints, the posterior of each objective and constraint, and the optimized samples drawn from the posterior.](image)
Information-based Approach

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

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

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

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

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

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

The Pareto set \( \mathcal{X}^* \) in the feasible space is a random variable! Information is measured by the entropy of \( p(\mathcal{X}^*|\mathcal{D}_N) \).
Information-based Approach

The Pareto set $\mathcal{X}^*$ in the feasible space is a random variable! Information is measured by the entropy of $p(\mathcal{X}^*|\mathcal{D}_N)$. 
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

$$\alpha(x) = H[\mathcal{X}^\star|\mathcal{D}_t] - \mathbb{E}_y \left[ H[\mathcal{X}^\star|\mathcal{D}_t \cup \{x, y\}] \bigg| \mathcal{D}_t, x \right]$$ (1)
Information-based Approach

The Pareto set $\mathcal{X}^*$ in the feasible space is a random variable! Information is measured by the entropy of $p(\mathcal{X}^*|\mathcal{D}_N)$.

The acquisition function is

$$\alpha(x) = H[\mathcal{X}^*|\mathcal{D}_t] - \mathbb{E}_y \left[ H[\mathcal{X}^*|\mathcal{D}_t \cup \{x, y\}|\mathcal{D}_t, x] \right]$$

(1)

How much we know about $\mathcal{X}^*$ now.
Information-based Approach

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

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

The acquisition function is

$$
\alpha(x) = H[\mathcal{X}^*|\mathcal{D}_t] - \mathbb{E}_y \left[ H[\mathcal{X}^*|\mathcal{D}_t \cup \{x,y\}] \bigg| \mathcal{D}_t, x \right] \tag{1}
$$

How much we know about $\mathcal{X}^*$ now.

How much we will know about $\mathcal{X}^*$ after collecting $y$ at $x$. 
Information-based Approach

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

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

![High Entropy vs Low Entropy](image)

The acquisition function is

$$\alpha(x) = H[\mathcal{X}^*|\mathcal{D}_t] - \mathbb{E}_y \left[ H[\mathcal{X}^*|\mathcal{D}_t \cup \{x, y\}] | \mathcal{D}_t, x \right]$$ (1)

How much we know about $\mathcal{X}^*$ now.

How much we will know about $\mathcal{X}^*$ after collecting $y$ at $x$. 
Information-based Approach

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

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

![High Entropy Low Information](chart1)

![Low Entropy High Information](chart2)

The acquisition function is

$$\alpha(x) = H[\mathcal{X}^*|\mathcal{D}_t] - \mathbb{E}_{y}[H[\mathcal{X}^*|\mathcal{D}_t \cup \{x, y\}]|\mathcal{D}_t, x]$$

(1)

- How much we know about $\mathcal{X}^*$ now.
- How much we will know about $\mathcal{X}^*$ after collecting $y$ at $x$.
- Computing (1) is very difficult in practice!
Predictive Entropy Search (PES)

We swap $y$ and $\mathcal{X}^*$ to obtain a reformulation of the acquisition function.
Predictive Entropy Search (PES)

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

\[
H[\mathcal{X}^*|\mathcal{D}_t] - \mathbb{E}_y\left[ H[\mathcal{X}^*|\mathcal{D}_t \cup \{x, y\}] | \mathcal{D}_t, x \right] \equiv \text{MI}(y, \mathcal{X}^*) \quad (\text{ESMOC})
\]
Predictive Entropy Search (PES)

We swap $y$ and $x^*$ to obtain a reformulation of the acquisition function.

\[
H[x^* | D_t] - \mathbb{E}_y \left[ H[x^* | D_t \cup \{x, y\}] \middle| D_t, x \right] \equiv \text{MI}(y, x^*) \quad \text{(ESMOC)}
\]

\[
H[y | D_t, x] - \mathbb{E}_{x^*} \left[ H[y | D_t, x, x^*] \middle| D_t, x \right] \equiv \text{MI}(x^*, y) \quad \text{(PESMOC)}
\]
Predictive Entropy Search (PES)

We swap $y$ and $x^*$ to obtain a reformulation of the acquisition function.

\[
\begin{align*}
H[x^*|D_t] - \mathbb{E}_y \left[ H[x^*|D_t \cup \{x, y\}] | D_t, x \right] & \equiv \text{MI}(y, x^*) \quad \text{(ESMOC)} \\
H[y|D_t, x] - \mathbb{E}_{x^*} \left[ H[y|D_t, x, x^*] | D_t, x \right] & \equiv \text{MI}(x^*, y) \quad \text{(PESMOC)}
\end{align*}
\]
Predictive Entropy Search (PES)

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

\[
H[\mathcal{X}^*|\mathcal{D}_t] - \mathbb{E}_y\left[H[\mathcal{X}^*|\mathcal{D}_t \cup \{x, y\}]|\mathcal{D}_t, x\right] \equiv \text{MI}(y, \mathcal{X}^*) \quad \text{(ESMOC)}
\]

\[
H[y|\mathcal{D}_t, x] - \mathbb{E}_{\mathcal{X}^*}\left[H[y|\mathcal{D}_t, x, \mathcal{X}^*]|\mathcal{D}_t, x\right] \equiv \text{MI}(\mathcal{X}^*, y) \quad \text{(PESMOC)}
\]
Predictive Entropy Search (PES)

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

\[
H[\mathcal{X}^*|\mathcal{D}_t] - \mathbb{E}_y\left[H[\mathcal{X}^*|\mathcal{D}_t \cup \{x, y\}]|\mathcal{D}_t, x\right] \equiv \text{MI}(y, \mathcal{X}^*) \quad \text{(ESMOC)}
\]

\[
H[y|\mathcal{D}_t, x] - \mathbb{E}_{\mathcal{X}^*}\left[H[y|\mathcal{D}_t, x, \mathcal{X}^*]|\mathcal{D}_t, x\right] \equiv \text{MI}(\mathcal{X}^*, y) \quad \text{(PESMOC)}
\]
Predictive Entropy Search (PES)

We swap $y$ and $x^*$ to obtain a reformulation of the acquisition function.

\[
H[x^*|D_t] - E_y[H[x^*|D_t \cup \{x, y\}|D_t, x] \equiv MI(y, x^*) \quad \text{(ESMOC)}
\]

\[
H[y|D_t, x] - E_{x^*}[H[y|D_t, x, x^*]|D_t, x] \equiv MI(x^*, y) \quad \text{(PESMOC)}
\]
Predictive Entropy Search (PES)

We swap $y$ and $x^*$ to obtain a reformulation of the acquisition function.

\[
H[x^*|\mathcal{D}_t] - \mathbb{E}_y \left[ H[x^*|\mathcal{D}_t \cup \{x, y\}] \big| \mathcal{D}_t, x \right] \equiv \text{MI}(y, x^*) \quad \text{(ESMOC)}
\]

\[
H[y|\mathcal{D}_t, x] - \mathbb{E}_{x^*} \left[ H[y|\mathcal{D}_t, x, x^*] \big| \mathcal{D}_t, x \right] \equiv \text{MI}(x^*, y) \quad \text{(PESMOC)}
\]
Predictive Entropy Search (PES)

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

\[
H[\mathcal{X}^* | \mathcal{D}_t] - \mathbb{E}_y \left[ H[\mathcal{X}^* | \mathcal{D}_t \cup \{x, y\}] | \mathcal{D}_t, x \right] \equiv \text{MI}(y, \mathcal{X}^*) \quad \text{(ESMOC)}
\]

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

Gaussian distribution

Approximated by sampling from $p(\mathcal{X}^* | \mathcal{D}_t)$
Predictive Entropy Search (PES)

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

\[
H[\mathcal{X}^*|\mathcal{D}_t] - \mathbb{E}_y \left[ H[\mathcal{X}^*|\mathcal{D}_t \cup \{x, y\}] | \mathcal{D}_t, x \right] \equiv \text{MI}(y, \mathcal{X}^*) \quad \text{(ESMOC)}
\]

\[
H[y|\mathcal{D}_t, x] - \mathbb{E}_{\mathcal{X}^*} \left[ H[y|\mathcal{D}_t, x, \mathcal{X}^*] | \mathcal{D}_t, x \right] \equiv \text{MI}(\mathcal{X}^*, y) \quad \text{(PESMOC)}
\]
Predictive Entropy Search (PES)

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

\[
\begin{align*}
H[\mathcal{X}^*|D_t] - \mathbb{E}_y \left[ H[\mathcal{X}^*|D_t \cup \{x, y\}] | D_t, x \right] & \equiv \text{MI}(y, \mathcal{X}^*) \quad \text{(ESMOC)} \\
H[y|D_t, x] - \mathbb{E}_{\mathcal{X}^*} \left[ H[y|D_t, x, \mathcal{X}^*] | D_t, x \right] & \equiv \text{MI}(\mathcal{X}^*, y) \quad \text{(PESMOC)}
\end{align*}
\]

\( \mathcal{X}^* \) dominates any other point in \( \mathcal{X} \).

\( \text{(Minka, 2001)} \)
Predictive Entropy Search (PES)

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

\[ H[\mathcal{X}^*|D_t] - \mathbb{E}_y \left[ H[\mathcal{X}^*|D_t \cup \{x, y\}] \big| D_t, x \right] \equiv \text{MI}(y, \mathcal{X}^*) \quad \text{(ESMOC)} \]

\[ H[y|D_t, x] - \mathbb{E}_{\mathcal{X}^*} \left[ H[y|D_t, x, \mathcal{X}^*] \big| D_t, x \right] \equiv \text{MI}(\mathcal{X}^*, y) \quad \text{(PESMOC)} \]

Gaussian distribution

Approximated by sampling from $p(\mathcal{X}^*|D_t)$

Factorized Gaussian approximation with 

\textbf{expectation propagation.}

$\mathcal{X}^*$ dominates any other point in $\mathcal{X}$.

\[
\alpha(x) \approx \sum_{c=1}^{C} \log v_c^{PD}(x) - \frac{1}{M} \sum_{m=1}^{M} \left( \sum_{c=1}^{C} \log v_c^{CPD}(x|\mathcal{X}^*_m) \right) + \\
\sum_{k=1}^{K} \log v_k^{PD}(x) - \frac{1}{M} \sum_{m=1}^{M} \left( \sum_{k=1}^{K} \log v_k^{CPD}(x|\mathcal{X}^*_m) \right)
\]

(Minka, 2001)
Predictive Entropy Search (PES)

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

$$H[\mathcal{X}^*|D_t] - \mathbb{E}_y \left[ H[\mathcal{X}^*|D_t \cup \{x, y\}] | D_t, x \right] \equiv \text{MI}(y, \mathcal{X}^*) \quad \text{(ESMOC)}$$

$$H[y|D_t, x] - \mathbb{E}_{\mathcal{X}^*} \left[ H[y|D_t, x, \mathcal{X}^*] | D_t, x \right] \equiv \text{MI}(\mathcal{X}^*, y) \quad \text{(PESMOC)}$$

$\mathcal{X}^*$ dominates any other point in $\mathcal{X}$.

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

(Minka, 2001)
Predictive Entropy Search (PES)

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

\[
H[\mathcal{X}^*|\mathcal{D}_t] - \mathbb{E}_y[H[\mathcal{X}^*|\mathcal{D}_t \cup \{x, y\}]|\mathcal{D}_t, x] \equiv \text{MI}(y, \mathcal{X}^*) \quad \text{(ESMOC)}
\]

\[
H[y|\mathcal{D}_t, x] - \mathbb{E}_{\mathcal{X}^*}[H[y|\mathcal{D}_t, x, \mathcal{X}^*]|\mathcal{D}_t, x] \equiv \text{MI}(\mathcal{X}^*, y) \quad \text{(PESMOC)}
\]

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

(Minka, 2001)
Example of PES' acquisition

\[ f_1(x) \]

\[ f_2(x) \]
Example of PES’ acquisition
Example of PES’ acquisition

\[ v_{1}^{PD}(x) \] Sample of \( X^* \)

\[ f_1(x) \]

\[ v_{2}^{PD}(x) \] Sample of \( X^* \)

\[ f_2(x) \]
Example of PES’ acquisition

\[ v_{1}^{PD}(x) \quad \text{Sample of } \mathcal{X}^* \quad v_{1}^{CPD}(x|\mathcal{X}_1^*) \]

\[ v_{2}^{PD}(x) \quad \text{Sample of } \mathcal{X}^* \quad v_{2}^{CPD}(x|\mathcal{X}_1^*) \]
Example of PES’ acquisition

$v^\text{PD}_1(x)$

Sample of $\mathcal{X}^*$

$v^\text{CPD}_1(x|\mathcal{X}^*)$

$\alpha_1(x)$

$f_1(x)$

$v^\text{PD}_2(x)$

Sample of $\mathcal{X}^*$

$v^\text{CPD}_2(x|\mathcal{X}^*)$

$\alpha_2(x)$

$f_2(x)$
Example of PES' acquisition

$v_{1}^{PD}(x)$  Sample of $\mathcal{X}^*$  $v_{1}^{CPD}(\mathcal{X}^{*})$  $\alpha_1(x)$

$v_{2}^{PD}(x)$  Sample of $\mathcal{X}^*$  $v_{2}^{CPD}(\mathcal{X}^{*})$  $\alpha_2(x)$
Finding a Fast and Accurate Neural Network

Average Pareto Front 100 Function Evaluations

Methods
- EHI
- ParEGO
- SMSego
- SUR

PES decoupled

coupled
Finding a Fast and Accurate Neural Network

Average Pareto Front 100 Function Evaluations

Evaluations PES decoupled

(Hernández-Lobato et al., 2016)
Finding a Fast and Accurate Neural Network

Average Pareto Front 100 Function Evaluations

Average Pareto Front 200 Function Evaluations

(Hernández-Lobato et al., 2016)
Low energy hardware accelerator

Pareto Fronts 600 Function Evaluations

Methods
- PES coupled
- PES decoupled
- Random search

Energy vs. Error

Energy: 23.75, 19.86, 15.97, 12.08, 8.19, 4.30
Error: 0.025, 0.027, 0.030, 0.033, 0.035, 0.038
Low energy hardware accelerator

Pareto Fronts 600 Function Evaluations

Methods
- PES coupled
- PES decocoupled
- Random search

Evaluations Performed by PES decoupled

Black boxes
- Energy
- Error

(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 $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$.

$$H[x^*|D_t] - \mathbb{E}_y[H[x^*|D_t \cup \{x_q, y_q\}_{q=1}^Q]|D_t, x] \equiv \text{MI}(y, x^*) \quad \text{(Parallel ES)}$$

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search

Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$.

\[
H[x^*|D_t] - \mathbb{E}_y \left[ H[x^*|D_t \cup \{x_q, y_q\}_{q=1}^Q]|D_t, x \right] \equiv \text{MI}(y, x^*) \quad \text{(Parallel ES)}
\]

\[
H[y|D_t, x] - \mathbb{E}_{x^*} \left[ H[y|D_t, x, x^*]|D_t, x \right] \equiv \text{MI}(x^*, y) \quad \text{(Parallel PES)}
\]

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search

Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$.

\[
H[x^* | D_t] - \mathbb{E}_y \left[ H[x^* | D_t \cup \{x_q, y_q\}_{q=1}^Q] | D_t, x \right] \equiv \text{MI}(y, x^*) \quad \text{(Parallel ES)}
\]

\[
H[y | D_t, x] - \mathbb{E}_{x^*} \left[ H[y | D_t, x, x^*] | D_t, x \right] \equiv \text{MI}(x^*, y) \quad \text{(Parallel PES)}
\]

(Shah and Ghahramani, 2015)
Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$.

\[
H[x^*|D_t] - \mathbb{E}_y \left[ H[x^*|D_t \cup \{x_q, y_q\}_{q=1}^Q]|D_t, x \right] \equiv \text{MI}(y, x^*) \quad \text{(Parallel ES)}
\]

\[
H[y|D_t, x] - \mathbb{E}_{x^*} \left[ H[y|D_t, x, x^*]|D_t, x \right] \equiv \text{MI}(x^*, y) \quad \text{(Parallel PES)}
\]

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search

Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$.

$$H[x^*|D_t] - \mathbb{E}_y[H[x^*|D_t \cup \{x_q, y_q\}_{q=1}^Q]|D_t, x] \equiv \text{MI}(y, x^*) \quad \text{(Parallel ES)}$$

$$H[y|D_t, x] - \mathbb{E}_{x^*}[H[y|D_t, x, x^*]|D_t, x] \equiv \text{MI}(x^*, y) \quad \text{(Parallel PES)}$$

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search

Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$.

$$H[x^*|D_t] - \mathbb{E}_y \left[ H[x^*|D_t \cup \{x_q, y_q\}_{q=1}^Q] | D_t, x \right] \equiv \text{MI}(y, x^*) \quad \text{(Parallel ES)}$$

$$H[y|D_t, x] - \mathbb{E}_{x^*} \left[ H[y|D_t, x, x^*] | D_t, x \right] \equiv \text{MI}(x^*, y) \quad \text{(Parallel PES)}$$

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search

Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$.

$$H[x^*|D_t] - \mathbb{E}_y \left[ H[x^*|D_t \cup \{x_q, y_q\}_{q=1}^Q] | D_t, x \right] \equiv \text{MI}(y, x^*) \quad \text{(Parallel ES)}$$

$$H[y|D_t, x] - \mathbb{E}_{x^*} \left[ H[y|D_t, x, x^*] | D_t, x \right] \equiv \text{MI}(x^*, y) \quad \text{(Parallel PES)}$$

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search

Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$.

$$H[x^*|D_t] - \mathbb{E}_y \left[ H[x^*|D_t \cup \{x_q, y_q\}_{q=1}^Q]|D_t, x \right] \equiv \text{MI}(y, x^*) \quad (\text{Parallel ES})$$

$$H[y|D_t, x] - \mathbb{E}_{x^*} \left[ H[y|D_t, x, x^*]|D_t, x \right] \equiv \text{MI}(x^*, y) \quad (\text{Parallel PES})$$

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search

Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$.

\[
\begin{align*}
H[x^*|D_t] - \mathbb{E}_y H[x^*|D_t \cup \{x_q, y_q\}_{q=1}^Q|D_t, x] &\equiv \text{MI}(y, x^*) \quad \text{(Parallel ES)} \\
H[y|D_t, x] - \mathbb{E}_{x^*} H[y|D_t, x, x^*|D_t, x] &\equiv \text{MI}(x^*, y) \quad \text{(Parallel PES)}
\end{align*}
\]

Multi-variate Gaussian distribution

Approximated by sampling from $p(x^*|D_t)$

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

(Shah and Ghahramani, 2015)
Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^\star$.

$$H[x^\star|D_t] - \mathbb{E}_y \left[ H[x^\star|D_t \cup \{x_q, y_q\}_{q=1}^Q] | D_t, x \right] \equiv \text{MI}(y, x^\star) \quad \text{(Parallel ES)}$$

$$H[y|D_t, x] - \mathbb{E}_{x^\star} \left[ H[y|D_t, x, x^\star] | D_t, x \right] \equiv \text{MI}(x^\star, y) \quad \text{(Parallel PES)}$$

Multi-variate Gaussian distribution

Approximated by sampling from $p(x^\star|D_t)$

Multivariate Gaussian approximation with **expectation propagation**

$x^\star$ is better than any other point in $\mathcal{X}$

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

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search

Choose a set $Q$ points $S_t = \{x_q\}_{q=1}^Q$ to minimize the entropy of $x^*$. 

\[ H[x^*|D_t] - \mathbb{E}_y \left[ H[x^*|D_t \cup \{x_q, y_q\}_{q=1}^Q]|D_t, x \right] \equiv \text{MI}(y, x^*) \quad \text{(Parallel ES)} \]

\[ H[y|D_t, x] - \mathbb{E}_{x^*} \left[ H[y|D_t, x, x^*]|D_t, x \right] \equiv \text{MI}(x^*, y) \quad \text{(Parallel PES)} \]

- Multi-variate Gaussian distribution
- Approximated by sampling from $p(x^*|D_t)$
- Multivariate Gaussian approximation with **expectation propagation**
- $x^*$ is better than any other point in $\mathcal{X}$

\[ \alpha(S_t) = \log |V^{PD}(S_t)| - \frac{1}{M} \sum_{m=1}^{M} \log |V^{CPD}(S_t|x^*_m)| \]

It is possible to compute the gradient of $\alpha(\cdot)$ w.r.t. each $x_q \in S_t$!

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search: Level Curves

(Shah and Ghahramani, 2015)
Parallel Predictive Entropy Search: Results

(Shah and Ghahramani, 2015)
Software for Bayesian Optimization

Many of the methods described are implemented into **Spearmint** using Python.
https://github.com/HIPS/Spearmint
Software for Bayesian Optimization

Many of the methods described are implemented into **Spearmint** using Python.

https://github.com/HIPS/Spearmint

Spearmint’s super-nice features:

1. Easy problem configuration (JSON file) and wrapper calling.
Software for Bayesian Optimization

Many of the methods described are implemented into Spearmint using Python.
https://github.com/HIPS/Spearmint

Spearmint’s super-nice features:

1. Easy problem configuration (JSON file) and wrapper calling.
2. GP hyper-parameter sampling via slice sampling (MCMC).
Software for Bayesian Optimization

Many of the methods described are implemented into **Spearmint** using Python.
https://github.com/HIPS/Spearmint

Spearmint’s super-nice features:

1. Easy problem configuration (JSON file) and wrapper calling.
2. GP hyper-parameter sampling via slice sampling (MCMC).
3. Allows for non-stationary functions via betawarp.
Software for Bayesian Optimization

Many of the methods described are implemented into Spearmint using Python.
https://github.com/HIPS/Spearmint

Spearmint’s super-nice features:

1. Easy problem configuration (JSON file) and wrapper calling.
2. GP hyper-parameter sampling via slice sampling (MCMC).
3. Allows for non-stationary functions via betawarp.
4. Supports ignoring variables in some objectives or constraints.
Software for Bayesian Optimization

Many of the methods described are implemented into **Spearmint** using Python.  
https://github.com/HIPS/Spearmint

Spearmint’s super-nice features:

1. Easy problem configuration (JSON file) and wrapper calling.
2. GP hyper-parameter sampling via slice sampling (MCMC).
3. Allows for non-stationary functions via betawarp.
4. Supports ignoring variables in some objectives or constraints.
5. Supports different schedulers including SLURM.
Software for Bayesian Optimization

Many of the methods described are implemented into **Spearmint** using Python.

https://github.com/HIPS/Spearmint

Spearmint’s super-nice features:

1. Easy problem configuration (JSON file) and wrapper calling.
2. GP hyper-parameter sampling via slice sampling (MCMC).
3. Allows for non-stationary functions via betawarp.
4. Supports ignoring variables in some objectives or constraints.
5. Supports different schedulers including SLURM.
6. Stores all results (observations and suggestions) in a database.
Software for Bayesian Optimization

Many of the methods described are implemented into Spearmint using Python.
https://github.com/HIPS/Spearmint

Spearmint’s super-nice features:

1. Easy problem configuration (JSON file) and wrapper calling.
2. GP hyper-parameter sampling via slice sampling (MCMC).
3. Allows for non-stationary functions via betawarp.
4. Supports ignoring variables in some objectives or constraints.
5. Supports different schedulers including SLURM.
6. Stores all results (observations and suggestions) in a database.

Other tools: SMAC (Java), Hyperopt (Python), Bayesopt (C++), PyBO (Python), MOE (Python / C++).
Further Extensions and Open Issues

1. **High-dimensionality**: BO is restricted to problems of moderate dimension. However, many big problems have low effective dimensionality which can be exploited (Wang *et al.*, 2013).
Further Extensions and Open Issues

1. **High-dimensionality**: BO is restricted to problems of moderate dimension. However, many big problems have low effective dimensionality which can be exploited (Wang et al., 2013).

2. Consider **dependencies** among objectives / tasks (Swersky et al., 2013) (Shah and Ghahramani, 2016). Most of the times the GPs are simply assumed to be independent, which is suboptimal.
Further Extensions and Open Issues

1. **High-dimensionality**: BO is restricted to problems of moderate dimension. However, many big problems have low effective dimensionality which can be exploited (Wang et al., 2013).

2. Consider **dependencies** among objectives / tasks (Swersky et al., 2013) (Shah and Ghahramani, 2016). Most of the times the GPs are simply assumed to be independent, which is suboptimal.

3. Most acquisition functions consider an evaluation **horizon equal to one**. We can do better by considering a particular evaluation budget and taking decisions accordingly (González et al., 2016).
Further Extensions and Open Issues

1. **High-dimensionality**: BO is restricted to problems of moderate dimension. However, many big problems have low effective dimensionality which can be exploited (Wang et al., 2013).

2. Consider **dependencies** among objectives / tasks (Swersky et al., 2013) (Shah and Ghahramani, 2016). Most of the times the GPs are simply assumed to be independent, which is suboptimal.

3. Most acquisition functions consider an evaluation **horizon equal to one**. We can do better by considering a particular evaluation budget and taking decisions accordingly (González et al., 2016).

4. **Safe Bayesian Optimization**: Sometimes we should avoid evaluating the objective at particular input locations (system failure) where it falls below some critical value (Berkenkamp et al., 2016).
Conclusions

Bayesian optimization:
Conclusions

Bayesian optimization:

1. Is a **powerful tool** that can be used to optimize black-box objectives that are very expensive to evaluate and noisy.
Conclusions

Bayesian optimization:

1. Is a **powerful tool** that can be used to optimize black-box objectives that are very expensive to evaluate and noisy.

2. Provides **significantly better results** than a random / uniform exploration of the input space.
Conclusions

Bayesian optimization:

① Is a powerful tool that can be used to optimize black-box objectives that are very expensive to evaluate and noisy.

② Provides significantly better results than a random / uniform exploration of the input space.

③ Can deal with complicated optimization problems with several objectives and / or multiple constraints.
**Conclusions**

**Bayesian optimization:**

1. Is a **powerful tool** that can be used to optimize black-box objectives that are very expensive to evaluate and noisy.

2. Provides **significantly better results** than a random / uniform exploration of the input space.

3. Can deal with complicated optimization problems with **several objectives** and / or **multiple constraints**.

**Thank you very much!**
References

References