Hyperparameter Optimization by Bayesian Optimization

Jee-Hyong Lee

Sungkyunkwan Univ.

Bayesian Optimization

- **Definition**

$$\arg\max_{\mathrm{x}} f(x)$$

  - You don't know anything about $f(x)$
  - You can query but it is very expensive

  - Any good idea??

Bayesian Optimization

- **Any Good Idea??**
  - No information on $f(x)$ ...
  - First choose a random point, x_1, and evaluate $f(x_1)$
  - Guess the shape of $f(x)$ based on $\left(x_1, f(x_1)\right)$
  - Based on the guess,
    choose the next point, x_2, and evaluate $f(x_2)$
  - Guess shape of $f(x)$ based on $\left\{\left(x_1, f(x_1)\right), \left(x_2, f(x_2)\right)\right\}$
  - Repeat those steps

Bayesian Optimization

- **Overall Description**

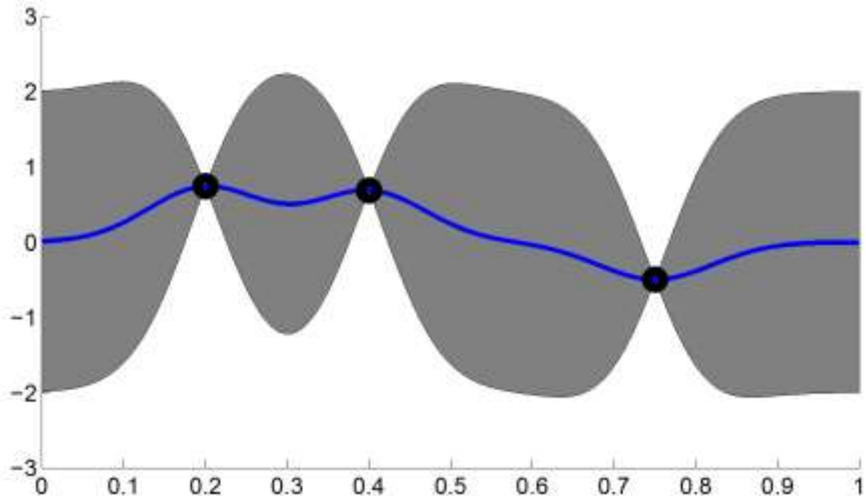
1. Guess the underlying function with known data points (Gaussian process)

2. Select the next point to query based on the guess (Acquisition function)

3. Repeat

Bayesian Optimization

- **Where to Query Next**
  - You expect to be good: Exploitation
  - You are uncertain about: Exploration



  - How to choose the next point to query?

Bayesian Optimization

- **Acquisition Function: $\alpha(x)$**
  - Return the fitness of x to be evaluated next
  - Choosing the next point

$$next\_point = \underset{x}{\operatorname{argmax}}\, \alpha(x)$$

- **Requirement of Acquisition Function**
  - Needs to balance Exploitation and Exploration
  - Need to be easy to optimize

- **How to optimize $\alpha(x)$?**

Bayesian Optimization

- **GP posterior gives $\boldsymbol{\mu(x), \sigma^2(x), p(y)}$**
  - »
- **Closed-Form Acquisition Functions**
  - Probability of Improvement

$$\alpha(x) = \frac{f(x_{best}) - \mu(x)}{\sigma(x)}$$

  - Expected Improvement

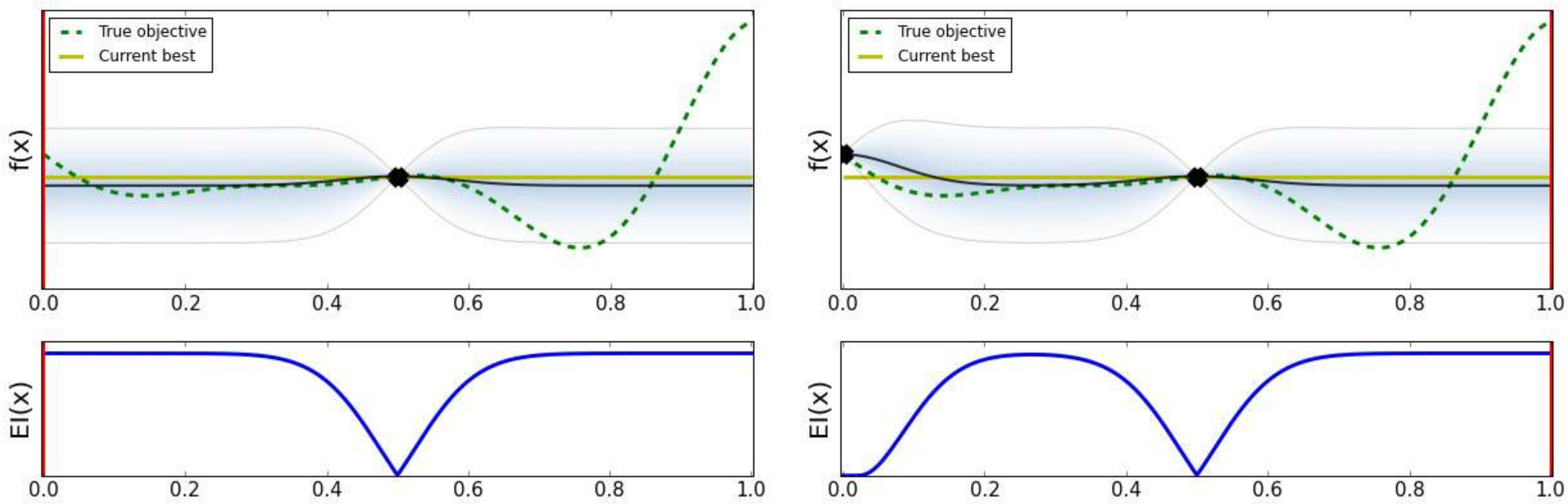
$$\alpha(x) = \int_y \max(0, y_{best} - y) \cdot p(y))\, dy$$

  - GP Upper Confidence Bound

$$\alpha(x) = \mu(x) - \beta\sigma(x)$$

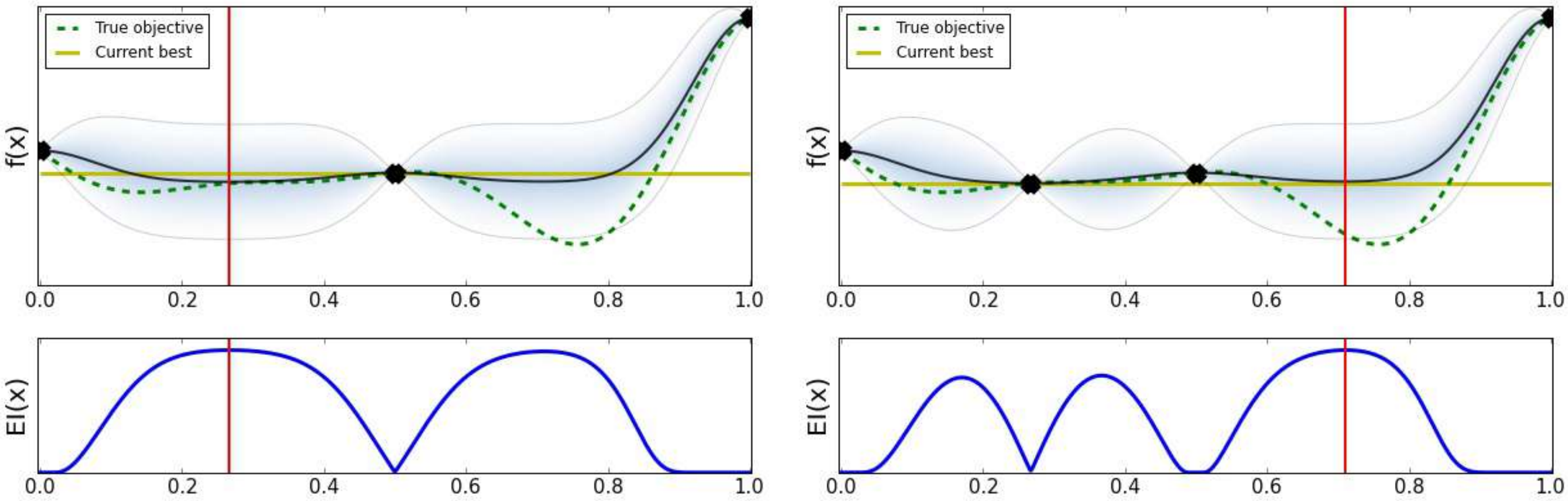
Bayesian Optimization

- **Illustration**



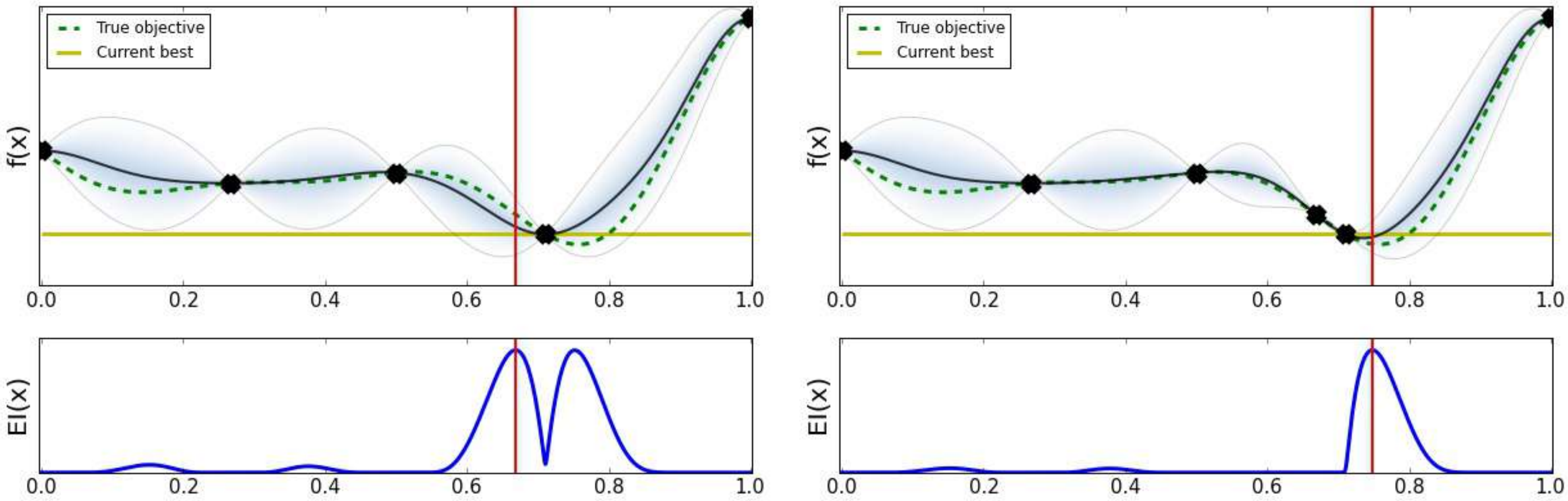
Bayesian Optimization

- **Illustration**



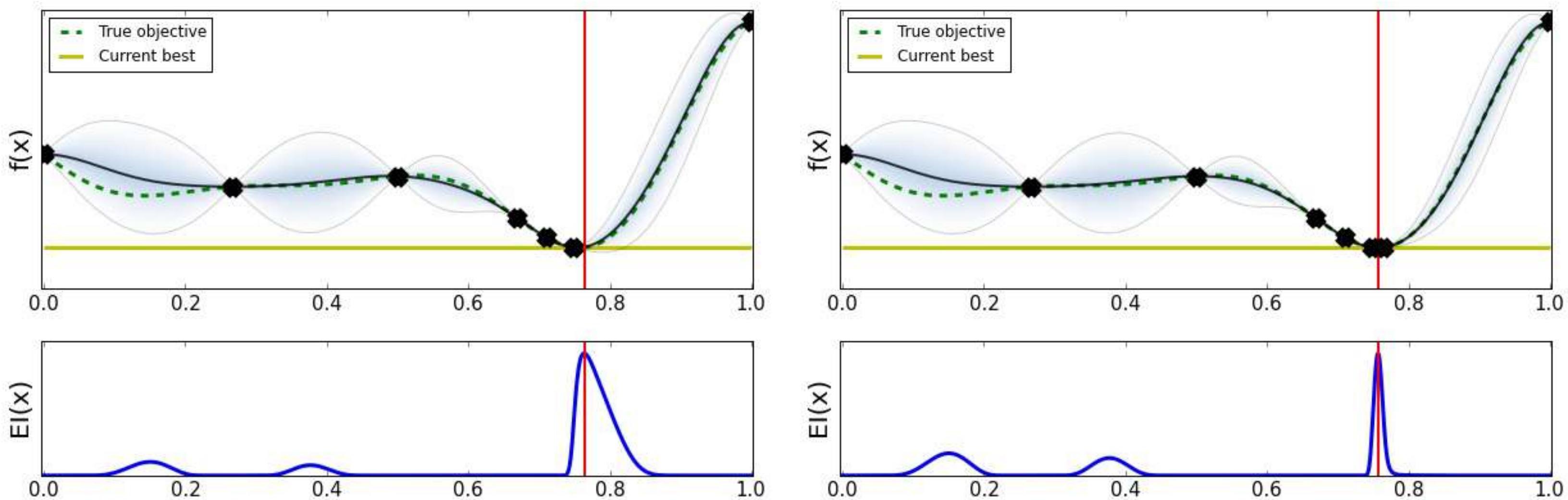
Bayesian Optimization

- **Illustration**



Bayesian Optimization

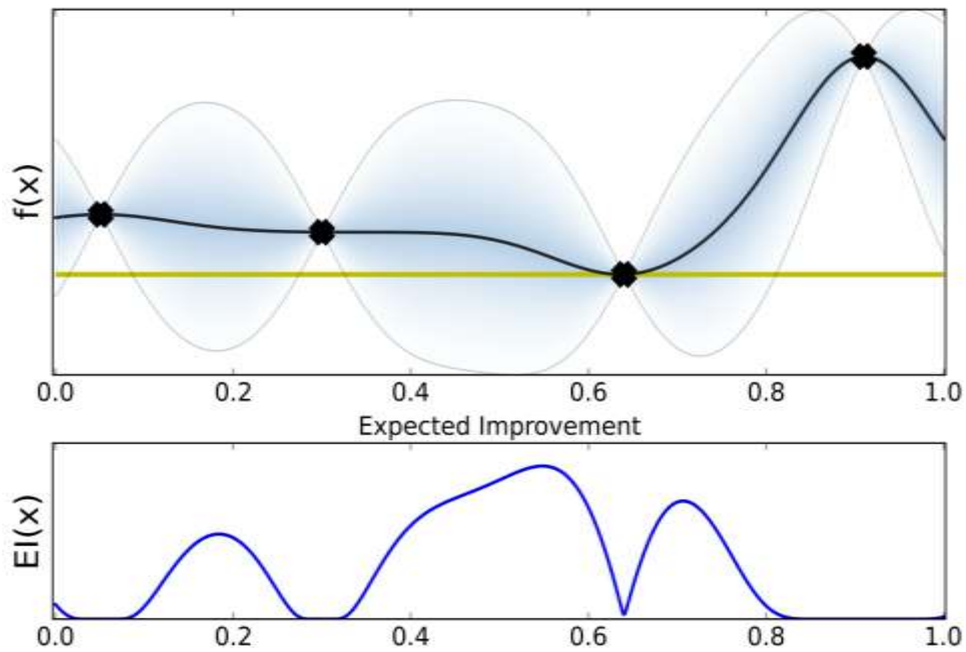
- **Illustration**



Acquisition Function

- **Expected Improvement**
  - The most used acquisition.

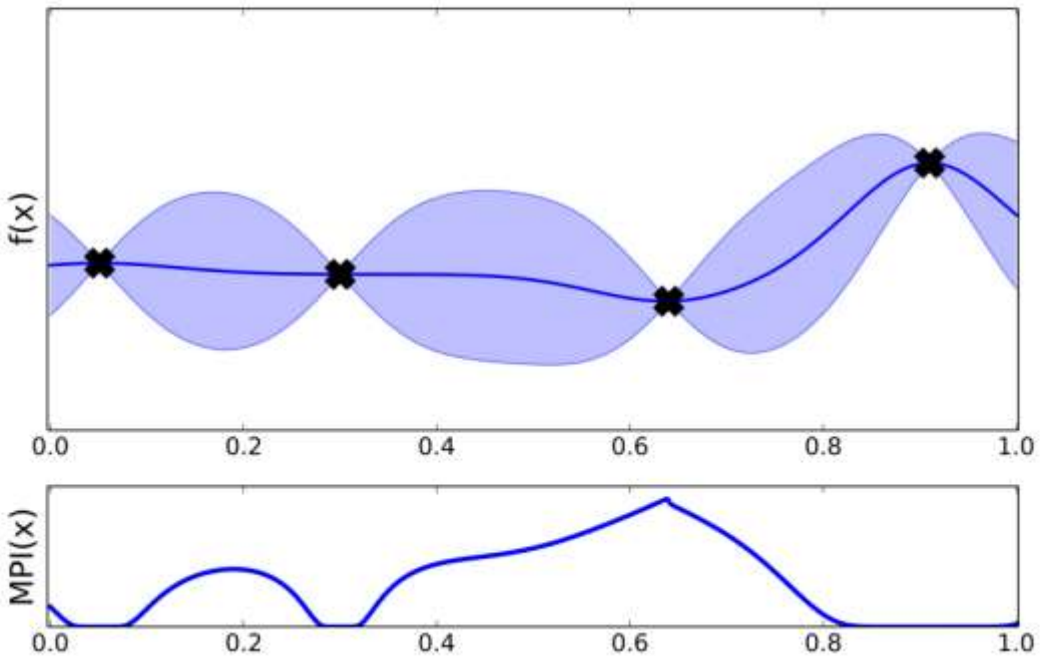
$$\alpha(x) = \int_y \max(0, y_{best} - y) \cdot p(y))\, dy$$



Acquisition Function

- **Probability of Improvement**
  - First used acquisition but Less used in practice.

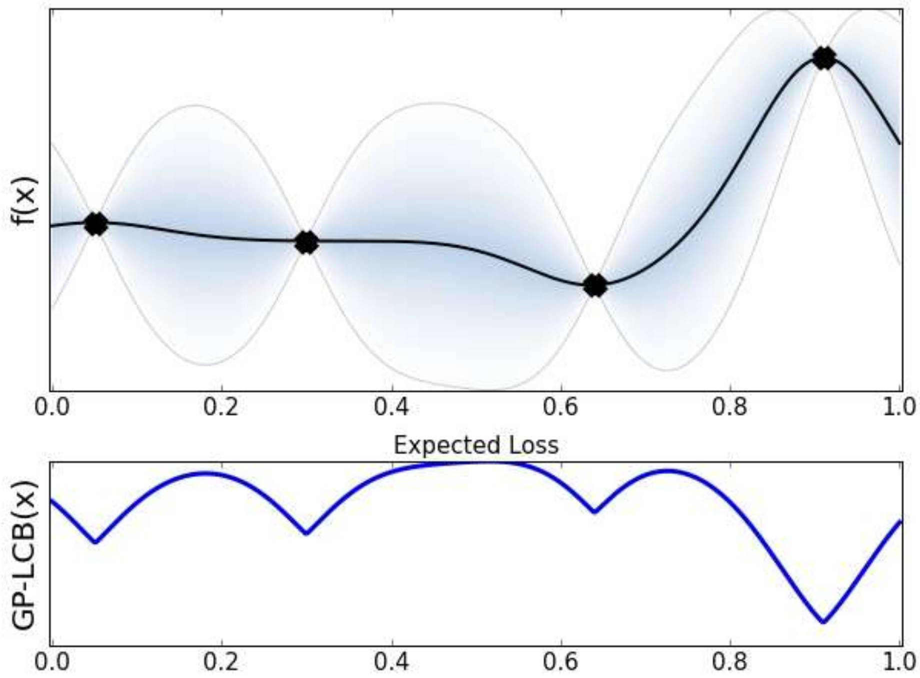
$$\alpha(x) = \frac{f(x_{best}) - \mu(x)}{\sigma(x)}$$



Acquisition function

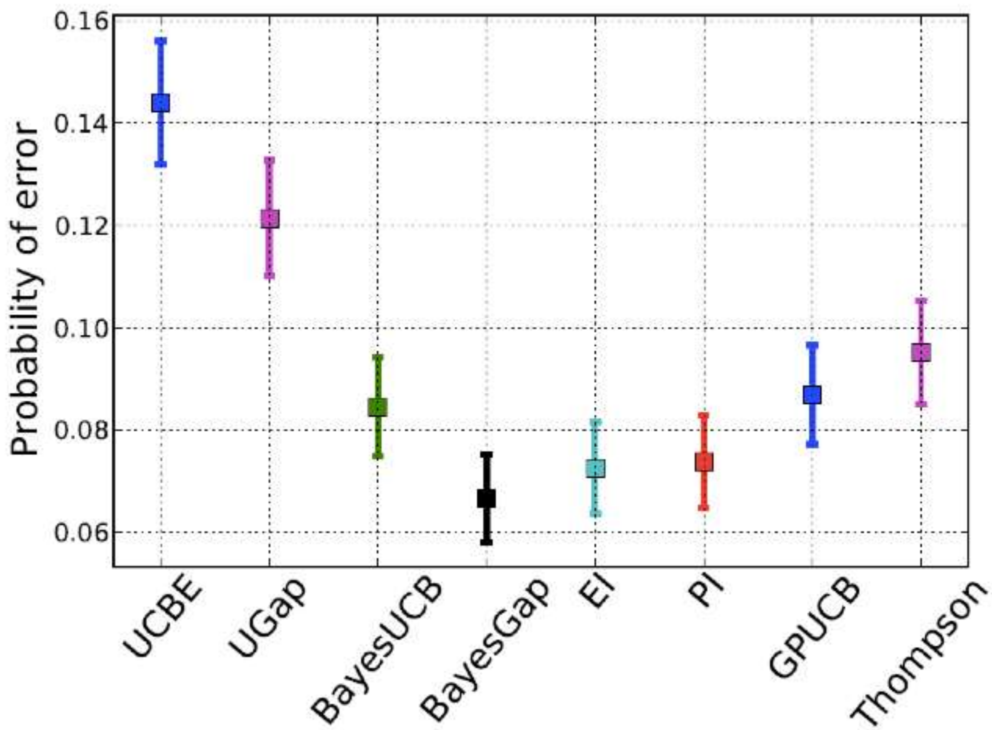
- **UP Upper (lower) Confidence Band**
  - Direct Balance between exploration and exploitation

$$\alpha(x) = \mu(x) - \beta\sigma(x)$$



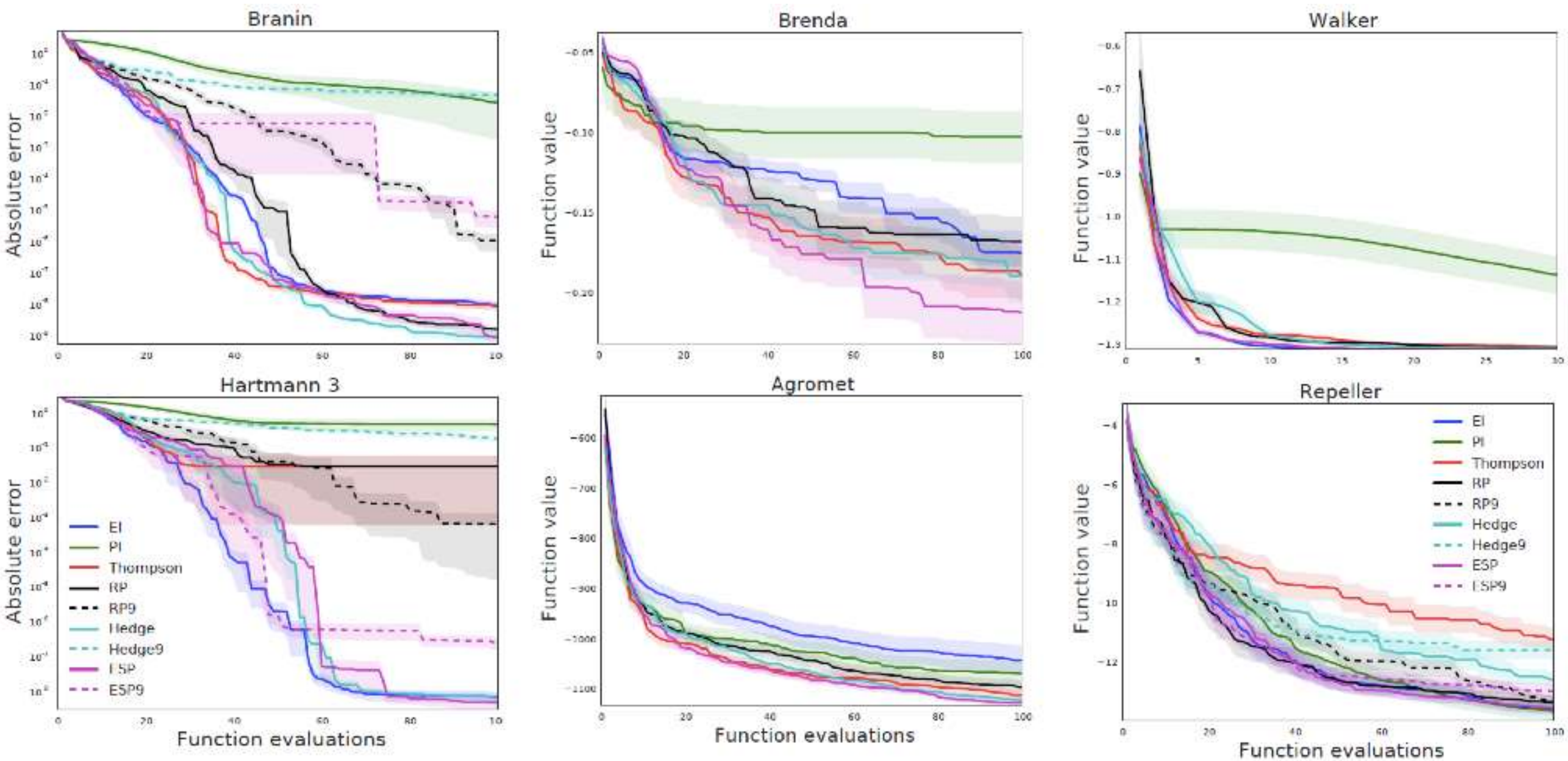
Acquisition Function

- **The choice of the utility may change a lot the result of the optimization.**



Acquisition Function

- **The best utility depends on the problem and the level of exploration/exploitation required**



How to Optimize Acquisition Function

- **Gradient descent methods**
  - Eg: Conjugate gradient.
- **Lipschitz based heuristics**
  - DIRECT
- **Evolutionary algorithms:**

- **Some of these methods can also be used to directly optimize the function**

How to Optimize Acquisition Function

- **Some acquisition functions are differentiable**

Algorithm 2: Gradient Descent

---

input : $f : \mathbb{R}^n \rightarrow \mathbb{R}$ a differentiable function
$\mathbf{x}^{(0)}$ an initial solution

output: $\mathbf{x}^{\star}$, a local minimum of the cost function f.

begin
    $k \leftarrow 0$;
    while STOP-CRIT **and** $(k < k_{max})$ **do**
        $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \alpha^{(k)} \nabla f(\mathbf{x})$;
        with $\alpha^{(k)} = \arg \min_{\alpha \in \mathbb{R}_+} f\left(\mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x})\right)$;
        $k \leftarrow k + 1$;
    return $\mathbf{x}^{(k)}$
end

How to Optimize Acquisition Function

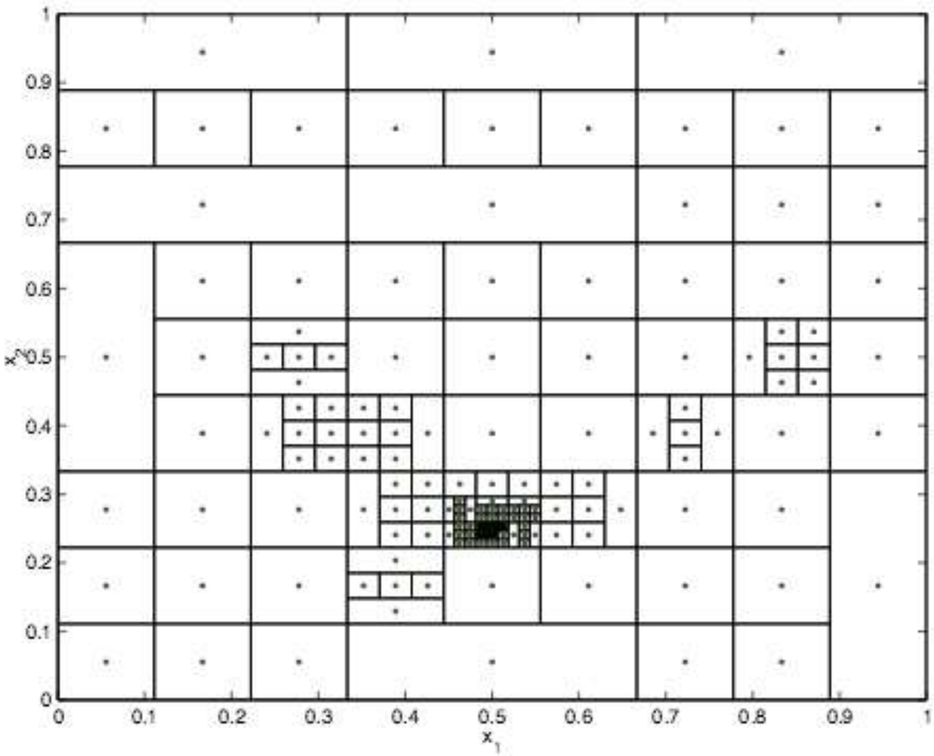
- **DIRECT**

Algorithm `DIRECT`('myfcn',bounds,opts)

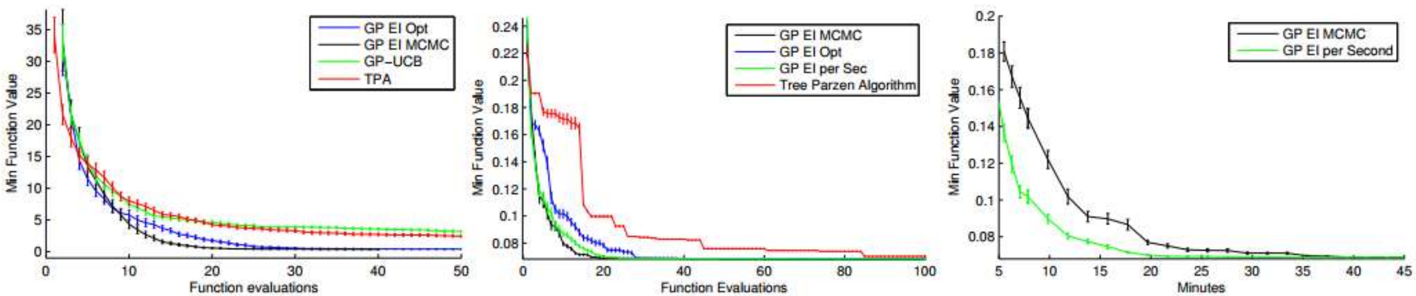
Normalize the domain to be the unit hyper-cube with center c_1
Find $f(c_1)$, $f_{min} = f(c_1)$, $i = 0$, $m = 1$
Evaluate $f(c_1 \pm \delta e_i$, $1 \leq i \leq n$, and divide hyper-cube
while $i \leq maxits$ and $m \leq maxevals$ **do**
    Identify the set S of all pot. optimal rectangles/cubes
    for all $j \in S$
        Identify the longest side(s) of rectangle j
        Evaluate myfcn at centers of new rectangles, and divide j into smaller rectangles
        Update f_{min}, $xatmin$, and m
    end for
    $i = i + 1$
end while

How to Optimize Acquisition Function

- **DIRECT**
  - Finds good solution in general and doesn't need gradient. Not generalizable to non-squared domains.

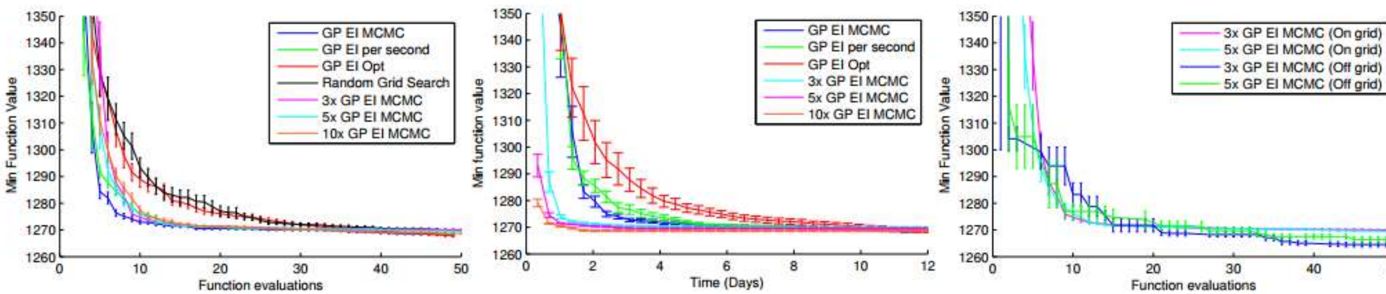


Performance



(a) (b) (c)

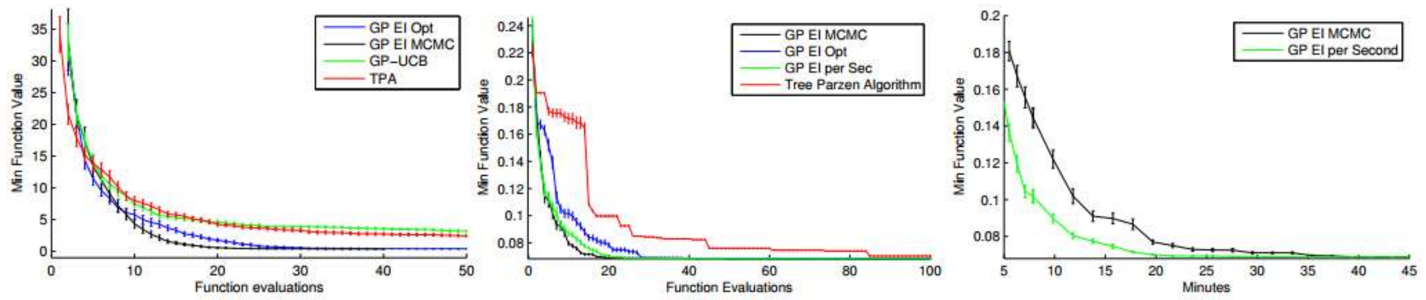
Figure 3: Comparisons on the Branin-Hoo function (3a) and training logistic regression on MNIST (3b). (3c) shows GP EI MCMC and GP EI per Second from (3b), but in terms of time elapsed.



(a) (b) (c)

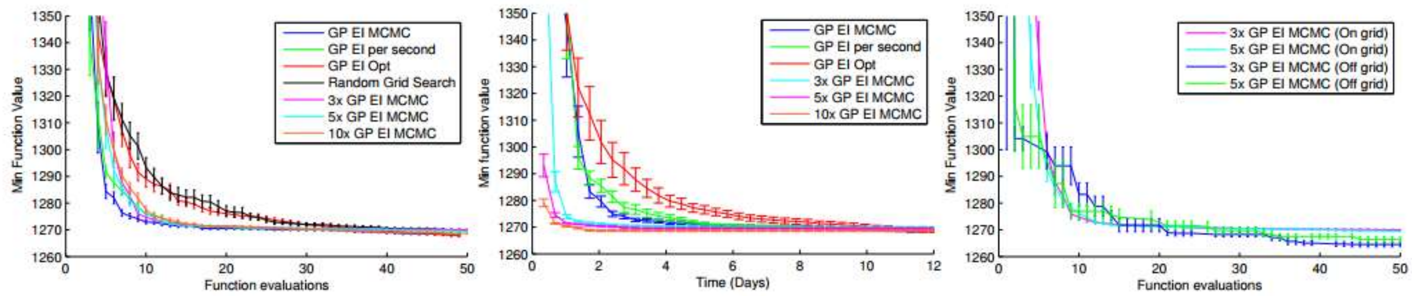
Figure 4: Different strategies of optimization on the Online LDA problem compared in terms of function evaluations (4a), walltime (4b) and constrained to a grid or not (4c).

Performance



(a) (b) (c)

Figure 3: Comparisons on the Branin-Hoo function (3a) and training logistic regression on MNIST (3b). (3c) shows GP EI MCMC and GP EI per Second from (3b), but in terms of time elapsed.



(a) (b) (c)

Figure 4: Different strategies of optimization on the Online LDA problem compared in terms of function evaluations (4a), walltime (4b) and constrained to a grid or not (4c).

Performance

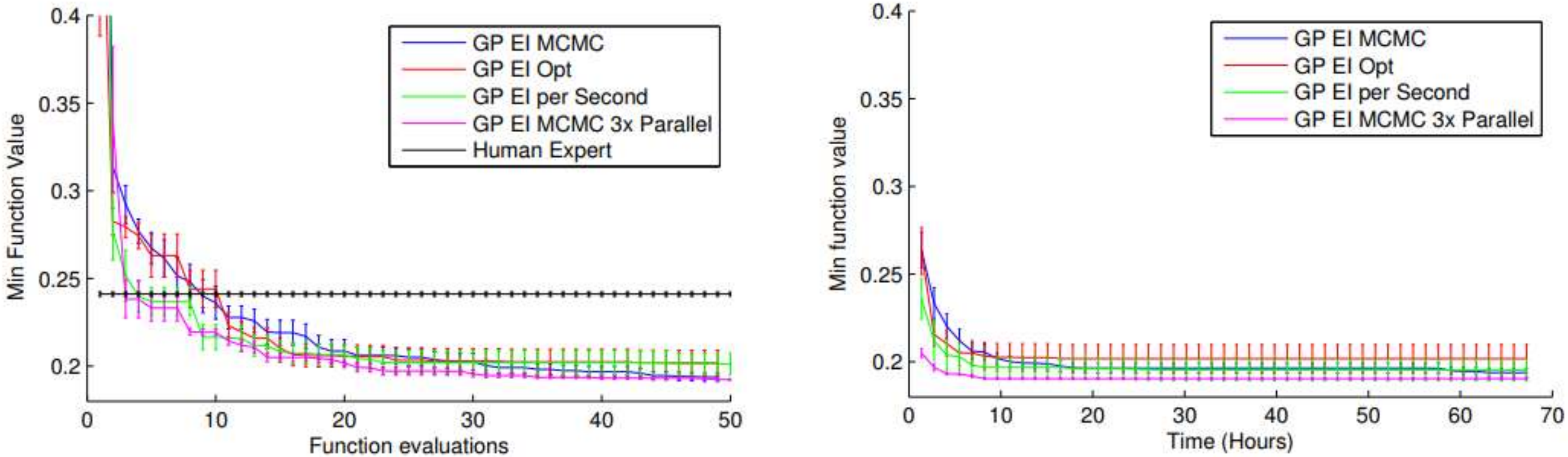


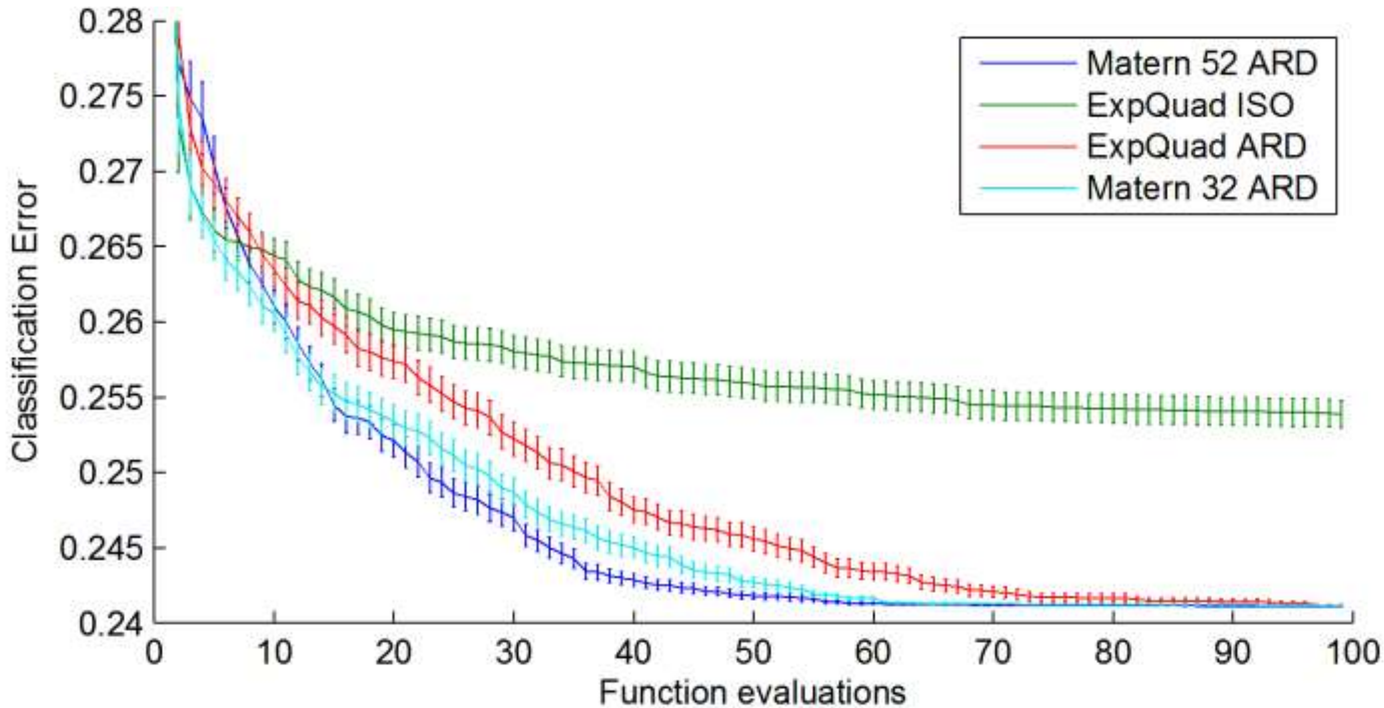
Figure 6: Validation error on the CIFAR-10 data for different optimization strategies.

Disadvantage

- **Fragility and poor default choices.**
  - Getting the function model wrong can be catastrophic.!
- **Experiments are run sequentially.**
  - We want to take advantage of cluster computing.!
- **Limited scalability in dimensions and evaluations.**
  - We want to solve big problems.

Disadvantage

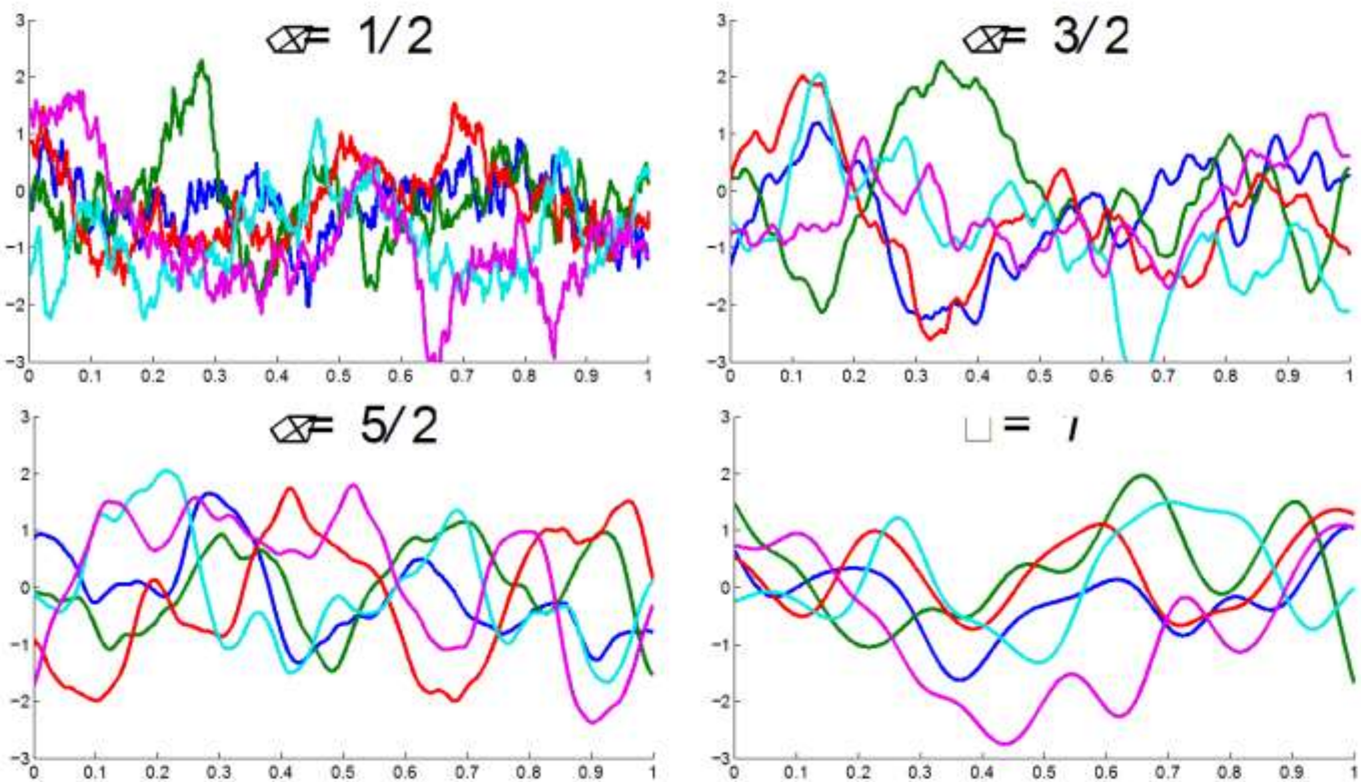
- **Covariance function selection**
  - This turns out to be crucial to good performance.
  - Usually use adaptive Matèrn 3/5 kernel.!



Disadvantage

- **Choosing Covariance Function**

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



High Dimensional BO Using Dropout

- **Bayesian Optimization**
  - Not good if we have many hyperparameters!!

- **Why don't we drop some values?**

High Dimensional Bayesian Optimization Using Dropout

Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen,
Svetha Venkatesh, Alistair Shilton
IJCAI 2017

High Dimensional BO Using Dropout

- **Randomly drop some variables**

Sample **d** out of **D** dimensions of the input x

I_d : $Indices\ of\ d\ out\ of\ D\ dimensions$

I_{D-d} : $Indices\ of\ leftout\ D-d\ dimensions$

$x^d = x^{I_d},\ x^{D-d} = x^{I_{D-d}}$

$\mathrm{x} = [x^d,\ x^{D-d}]$

→ Use only d dimensions to optimize acquisition function

$$a(\boldsymbol{x}^d) = \mu_{t-1}(\boldsymbol{x}^d) + \sqrt{\beta_t^d}\sigma_{t-1}(\boldsymbol{x}^d)$$ (UCB function)

High Dimensional BO Using Dropout

three "fill-in" strategies for x_t^{D-d}

- **Dropout-Random**: use a random value in the domain:

$$x_t^{D-d} \sim u(x^{D-d}) \tag{4}$$

- **Dropout-Copy:** copy the value of the variables from the best function value so far:

$$x_t^{+} = \text{argmax}_{t' \leq t} f(x_{t'})$$
$$x_t^{D-d} = (x_t^{+})^{D-d} \tag{5}$$

→ This may be stuck in local optimum

where x_t^{+} is the variables of the best found function value till t iterations.

→ this problem is solved by third strategy

- **Dropout-Mix:** use a mixture of the above two methods. We use a random value with probability p or copy the value from the variables of the best found function value so far with the probability $1 - p$.

High Dimensional BO Using Dropout

▪ Dropout Algorithm

Algorithm 1 Dropout Algorithm for High-dimensional Bayesian Optimization

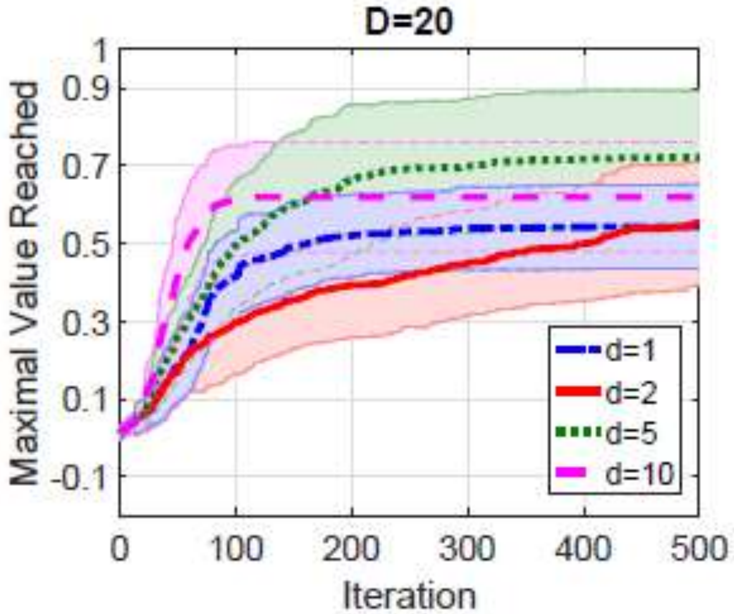
---

Input: $\mathcal{D}_1 = \{\boldsymbol{x}_0, y_0\}$

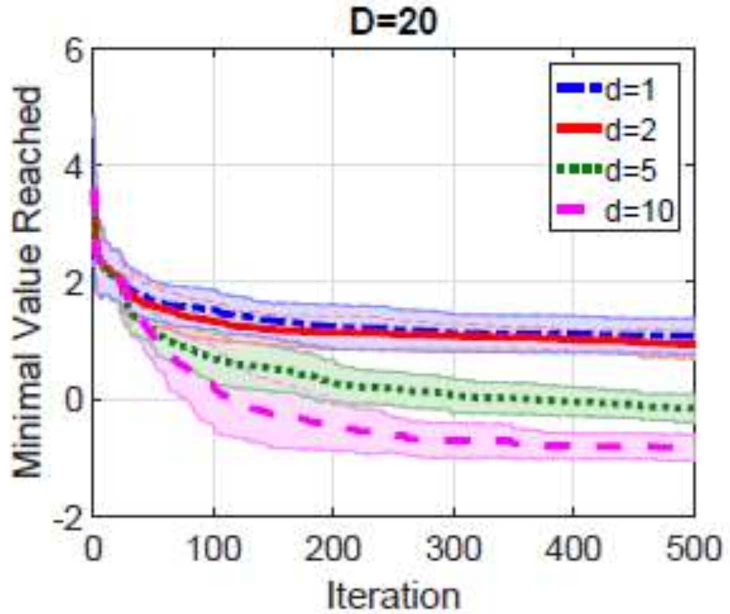
1: **for** $t = 1, 2, \cdots$ **do**
2:     randomly select d dimensions
3:     $\boldsymbol{x}_t^d \leftarrow \text{argmax}_{\boldsymbol{x}_t^d \in \mathcal{X}^d} a(\boldsymbol{x}^d \mid \mathcal{D}_t)$ (Eq.(3))
4:     $\boldsymbol{x}_t^{D-d} \leftarrow$one of three "fill-in" strategies (Sec 2.)
5:     $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_t^d \cup \boldsymbol{x}_t^{D-d}$
6:     $y_t \leftarrow$ Query y_t at $\boldsymbol{x}_t$
7:     $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{\boldsymbol{x}_t, y_t\}$
8: **end for**

High Dimensional BO Using Dropout

- **Effect of d**



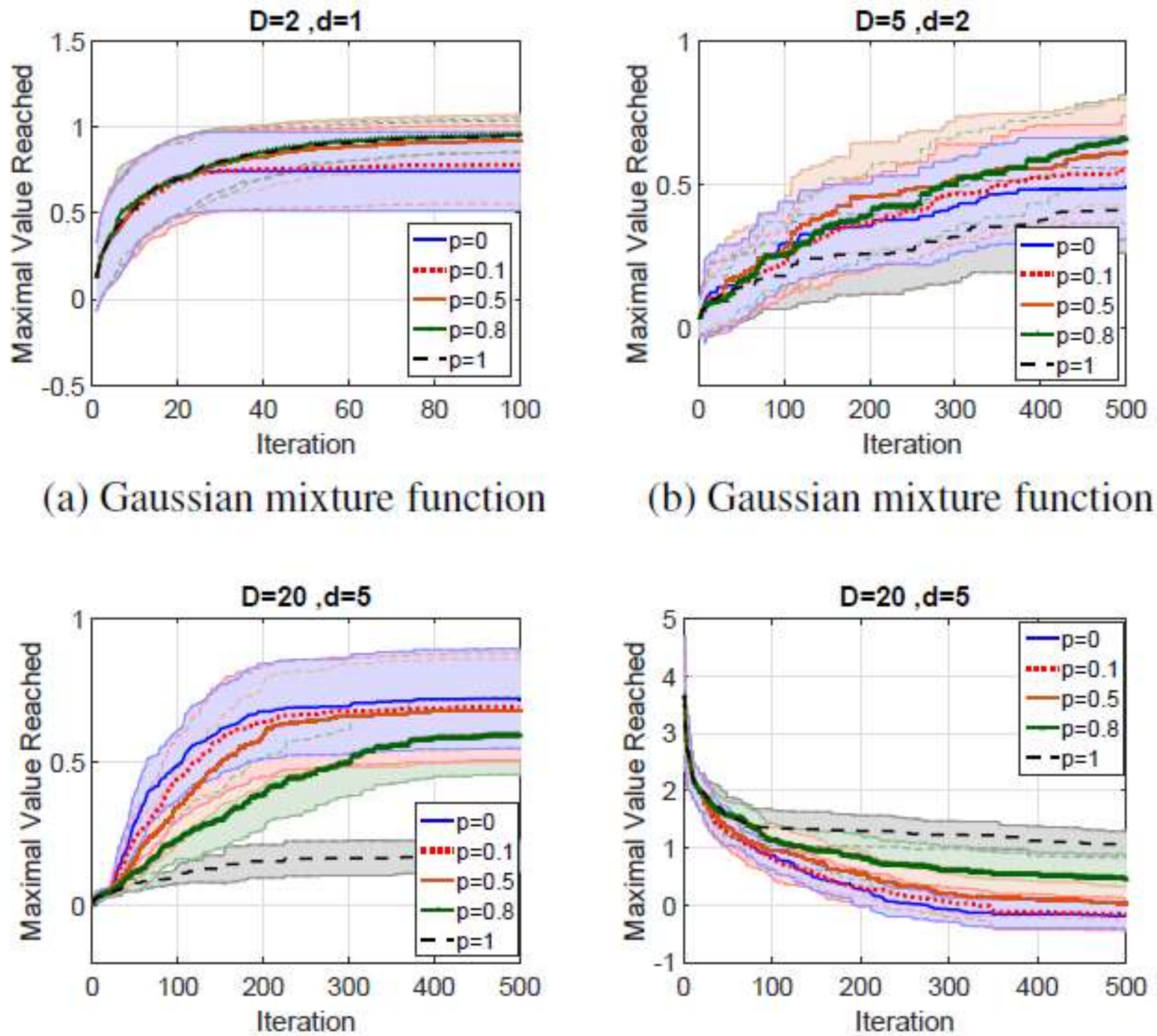
(a) Gaussian mixture function



(b) Schwefel's 1.2 function

High Dimen

- **Effect of p**



(a) Gaussian mixture function (b) Gaussian mixture function

(c) Gaussian mixture function (d) Schwefel's 1.2 function

High Dimensional BO Using Dropout

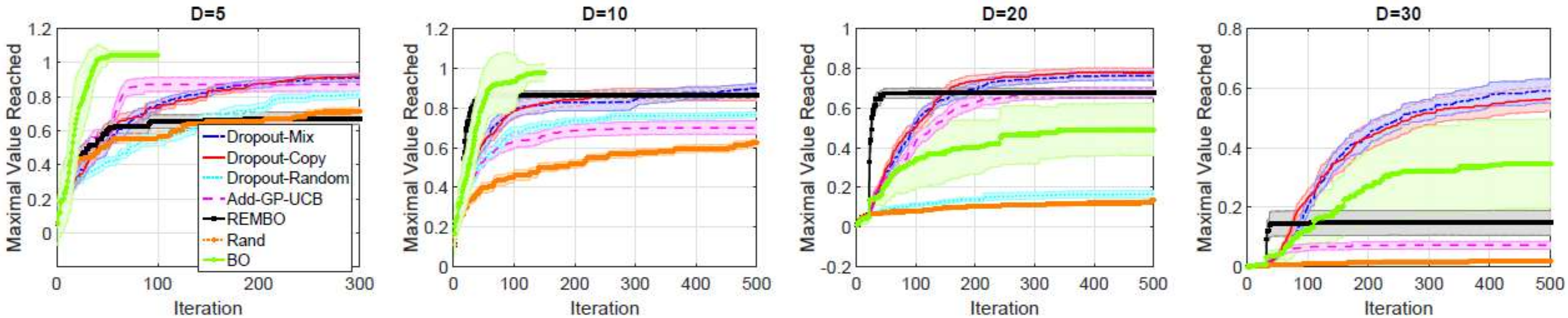


Figure 3: The optimization for the Gaussian mixture function. Higher value is better. Four different dimensions are tested from left to right (a) $D = 5$ (b) $D = 10$ (c) $D = 20$ (d) $D = 30$. The BO for $D = 5$ and $D = 10$ is terminated once it converges. The graphs are best seen in color.

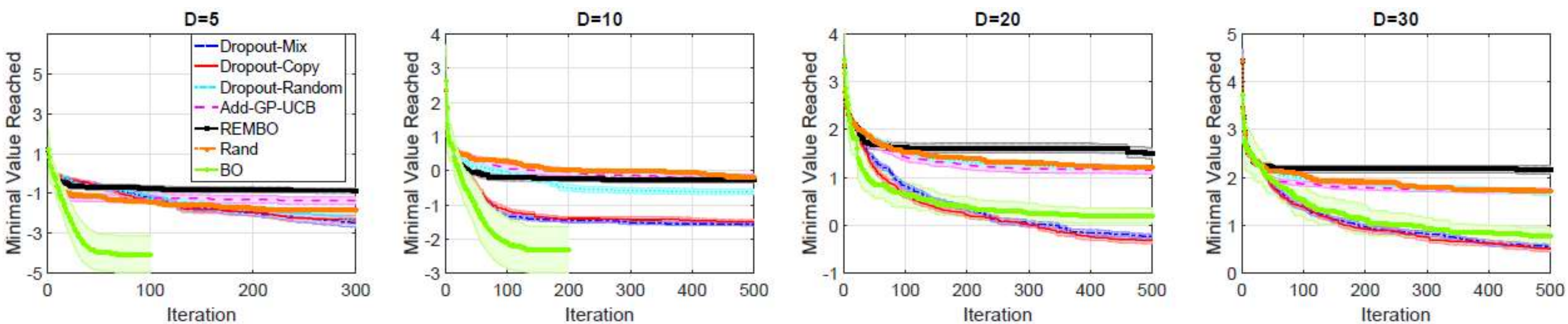


Figure 4: The optimization for Schwefel's 1.2 function. Lower value is better. Four different dimensions are tested from left to right (a) $D = 5$ (b) $D = 10$ (c) $D = 20$ (d) $D = 30$. The graphs are best seen in color.

High Dimensional BO Using Dropout

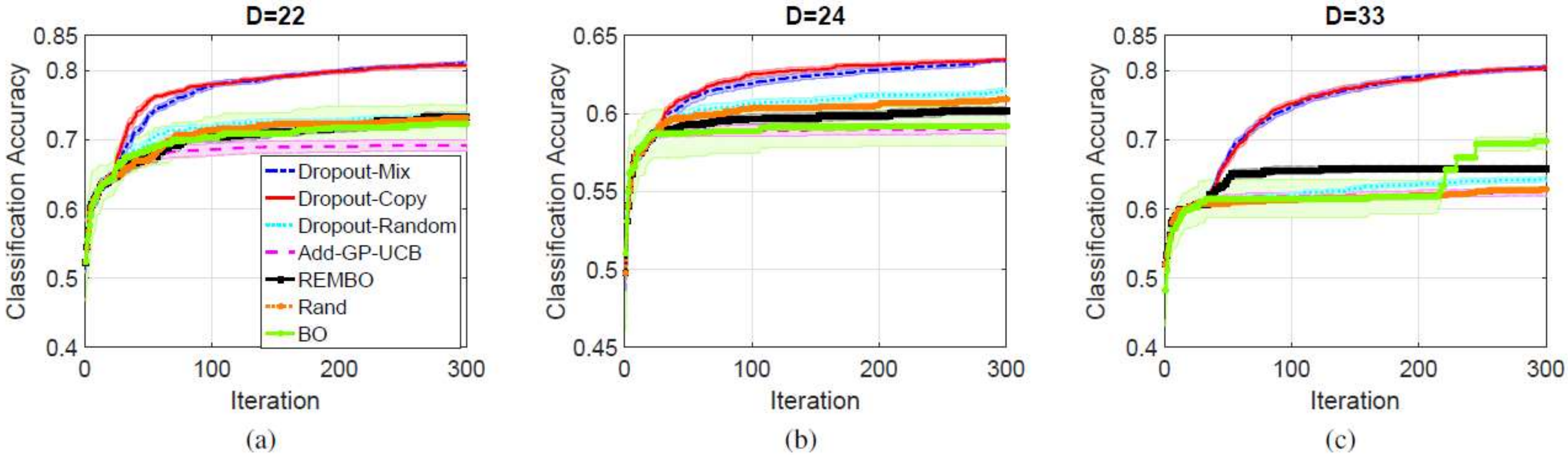


Figure 5: Maximum classification accuracy for training data as a function of Bayesian optimization iteration. The number of stages in a cascade classifier is equal to the number of features in three datasets (a) IJCNN1 $D = 22$, (b) German $D = 24$, (c) Ionosphere $D = 33$.