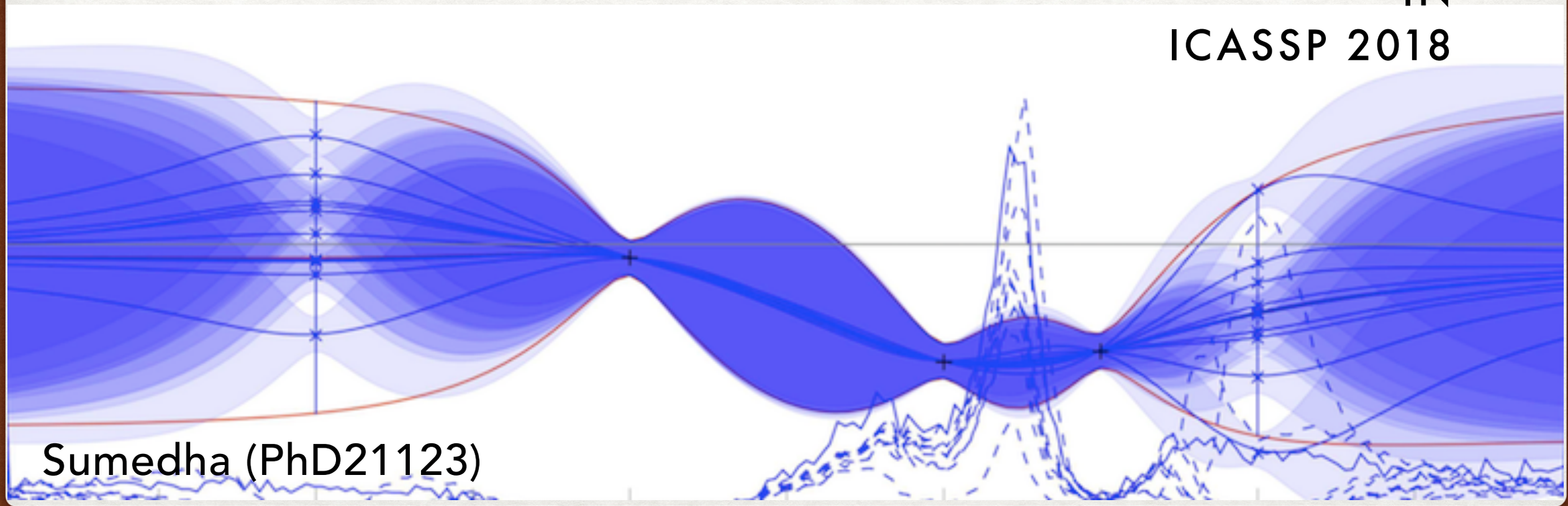
CLUSTERING-GUIDED GP-UCB FOR BAYESIAN OPTIMIZATION

BML Mid-Term Paper Presentation

JUNGTAEK KIM AND SEUNGJIN CHOI
IN
ICASSP 2018

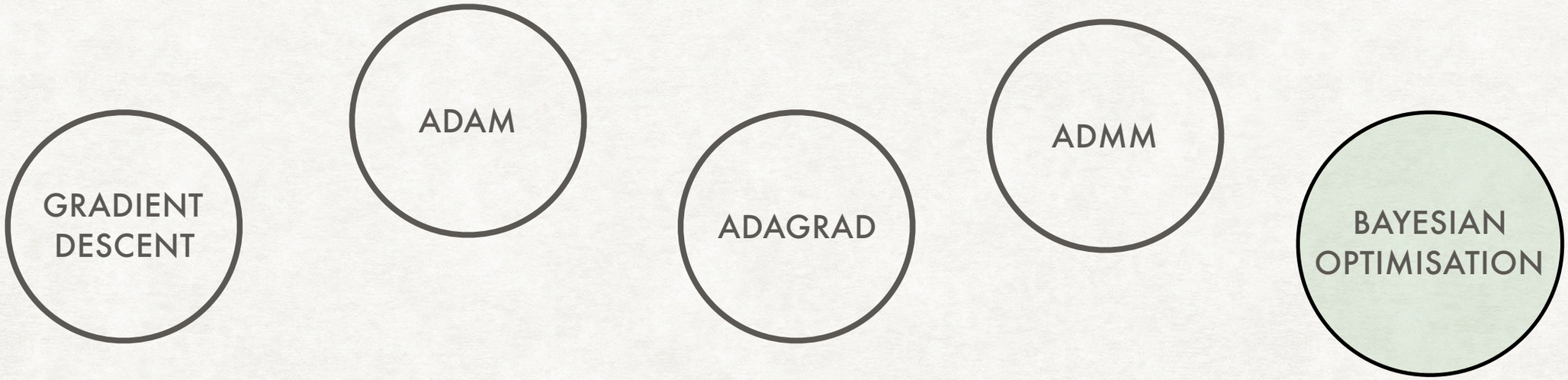


Sumedha (PhD21123)

CONTENTS

TO BAYE OR NOT TO BAYE

WHY BAYESIAN OPTIMIZATION

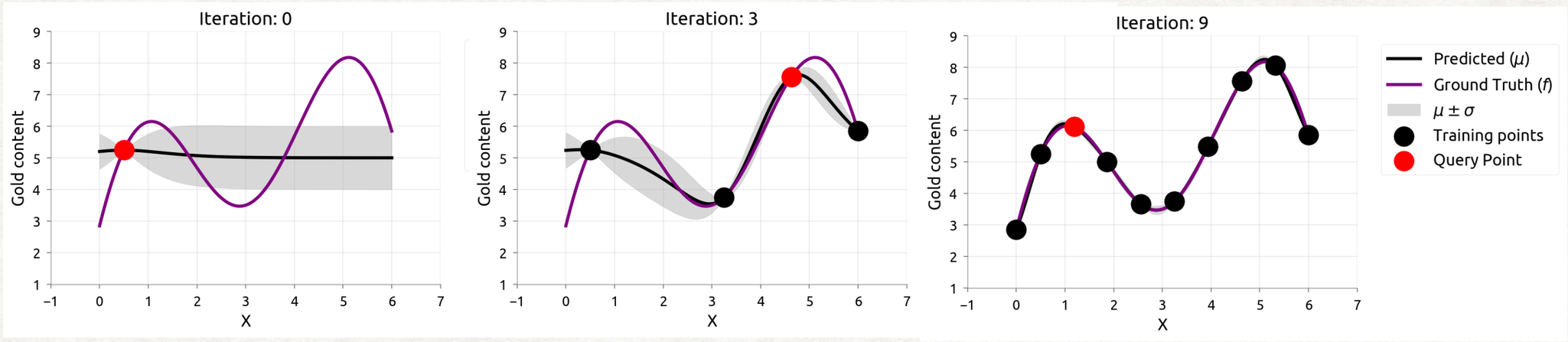


- At the times, when a large amount data is available, an approximation for original function y = f(x) can be made and hence it's maxima or minima can be found.
- But in the cases where data is not available in abundance and obtaining each data point costs a lot such as in case of gold drilling or bandit problem, f(x) is a black box function and can't be approximated

TO BAYE OR NOT TO BAYE

WHY BAYESIAN OPTIMIZATION

- One way to get this is by obtaining a surrogate function of f(x), by fitting a prior on it; which is usually Gaussian. Then choose the next point with highest uncertainty (variance) by exploring the domain (to minimise the uncertainity) and then update the GP using Bayes rule. Doing this repeatedly gives us a good approximation of function and this process is known as Active learning



- In the cases where we don't want the model, we just want maxima or minima i.e. location with maximum gold, estimating the whole function seems wasteful

- Balancing exploration and exploitation, based on what we know so far to evaluate next point is known as Bayesian Optimization (Explained Later)

BASICS

BAYESIAN OPTIMIZATION

TAKE A FEW SAMPLES VALUES & GET THEIR FUNCTION EVALUATIONS

↓

FIT A PRIOR ON THESE SAMPLE VALUES (USUALLY GAUSSIAN PROCESS BECAUSE OF SMOOTH NATURE OF REAL WORLD DATA)

↓

OBTAIN POSTERIOR MEAN AND POSTERIOR VARIANCE FOR GIVEN SET OF OBESERVATIONS

↓

USE ACQUISITION FUNCTION α TO DECIDE NEXT POINT SUCH THAT $X_{T+1} = \text{ARGMAX}(\alpha)$

↓

ADD NEWLY SAMPLED DATA (X_{T+1} & CORRESPONDING OBSERVATION)TO SET OF OBSERVATIONS, UPDATE THE PRIOR AND GO TO STEP3 UNTIL CONVERGENCE

BASICS

BAYESIAN OPTIMIZATION

Algorithm 1 Bayesian Optimization with GP regression

Require: Initial data $\mathcal{D}_{1:I} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_I, y_I)\}$, and $T \in \mathbb{N} > 0$

1: **for** $t = 1, 2, \ldots, T$ **do**

2: Find $\mathbf{x}_{I+t}$ that maximizes the acquisition function over the current GP: $\mathbf{x}_{I+t} = \arg\max_{\mathbf{x}} a(\mathbf{x}|\mathcal{D}_{1:I+t-1})$.

3: Sample the objective function: $y_{I+t} = f(\mathbf{x}_{I+t}) + \epsilon_{I+t}$.

4: Augment the data: $\mathcal{D}_{1:I+t} = \{\mathcal{D}_{1:I+t-1}, (\mathbf{x}_{I+t}, y_{I+t})\}$.

5: Update the GP, computing $\mu_{I+t}(\mathbf{x}), \sigma^2_{I+t}(\mathbf{x})$:

6: **end for**

7: **return** $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \{\mathbf{x}_1, \ldots, \mathbf{x}_{I+T}\}} \mu_{I+T}(\mathbf{x})$

BAYESIAN OPTIMIZATION

GAUSSIAN PROCESS

- A Gaussian process is a random process, where any point $x \in \mathbb{R}^d$ is assigned a random variable $f(x)$ and where the joint distribution of a finite number of these variables $p(f(x_1),\ldots,f(x_N))$ is itself Gaussian.

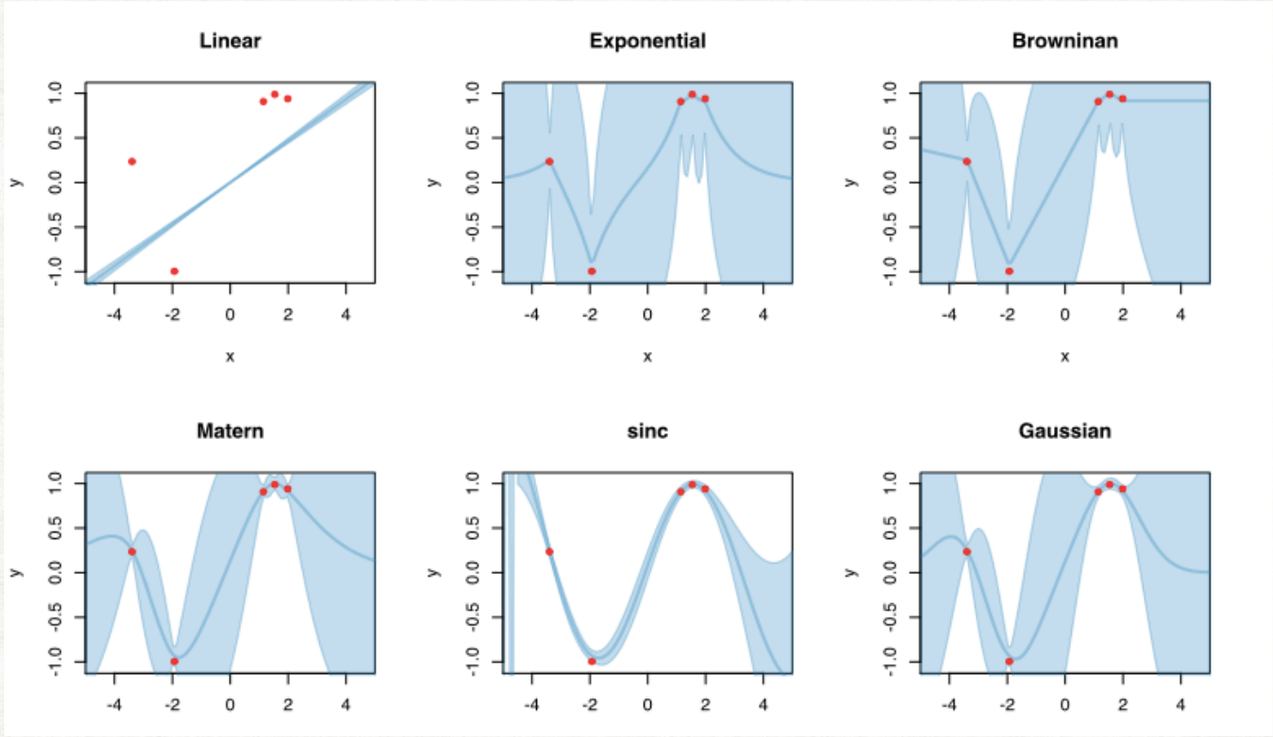
- $p(f|X) = N(f|\mu,K)$

- $f=(f(x_1),\ldots,f(x_N))$, $\mu=(m(x_1),\ldots,m(x_N))$ and $Kij=\kappa(x_i,x_j)$.

- m is the mean function, κ is a positive definite kernel function or covariance function.

- Thus, a Gaussian process is a distribution over functions whose shape is defined by K.

- Different Types of Gaussian Process Kernels are: White noise kernel, Gaussian kernel or radial basis function kernel, Rational quadratic kernel, Periodic kernel, Matern kernel etc.



BAYESIAN OPTIMIZATION

ACQUISITION FUNCTION

Tells how to acquire data, balancing exploration (looking at points with high mean) & exploitation(looking at points where variance is high)

PI (PROBABILITY OF IMPROVEMENT)

- This acquisition function chooses the next query point as the one which has the highest probability of improvement over the current max.

Mathematically it is expressed as:

$$x_{t+1} = argmax(\alpha_{PI}(x)) = argmax(P(f(x) \geq (f(x^+) + \epsilon)))$$

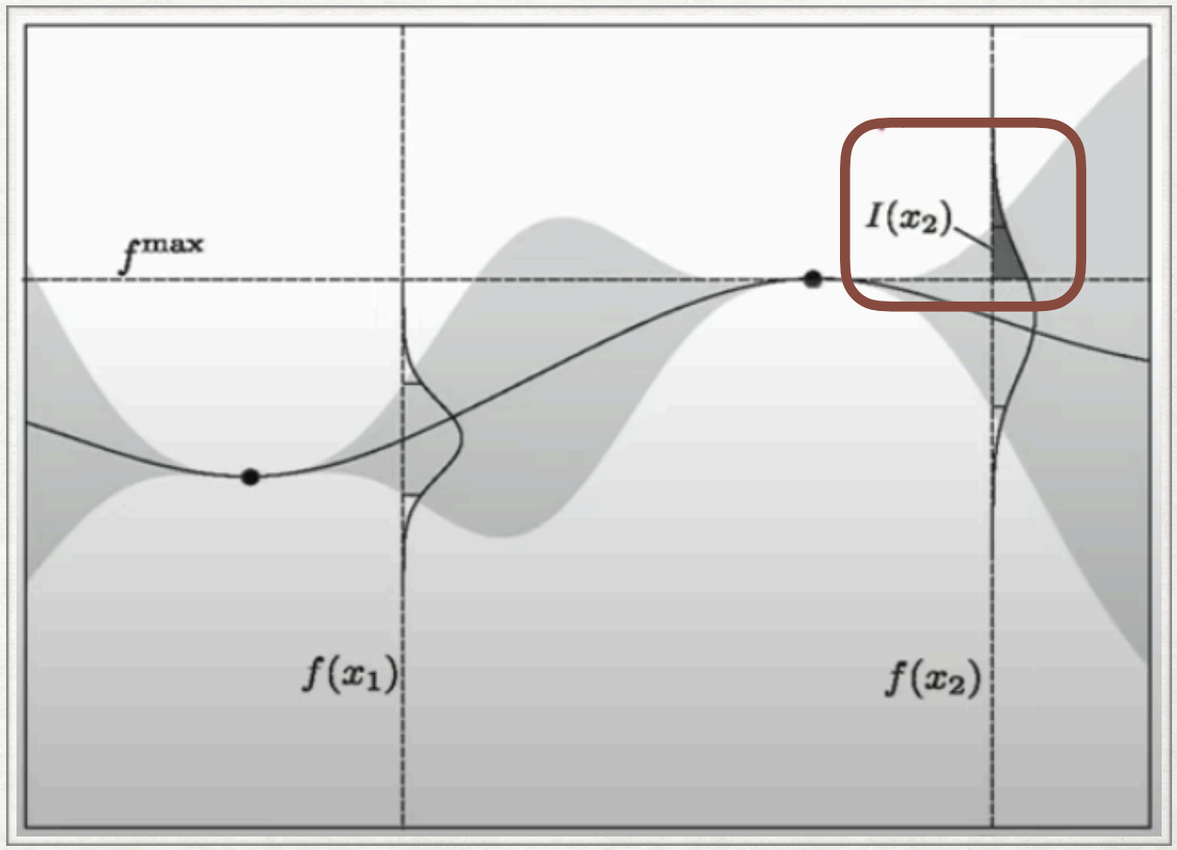
where, ϵ is a small positive number and $f(x^+)$ is current max

- Can be seen as finding the upper-tail probability (or the CDF) of the surrogate posterior.

- Using a GP as a surrogate the expression above converts to:

$$x_{t+1} = argmax_x \Phi\left(\frac{\mu_t(x) - f(x^+) - \epsilon}{\sigma_t(x)}\right)$$

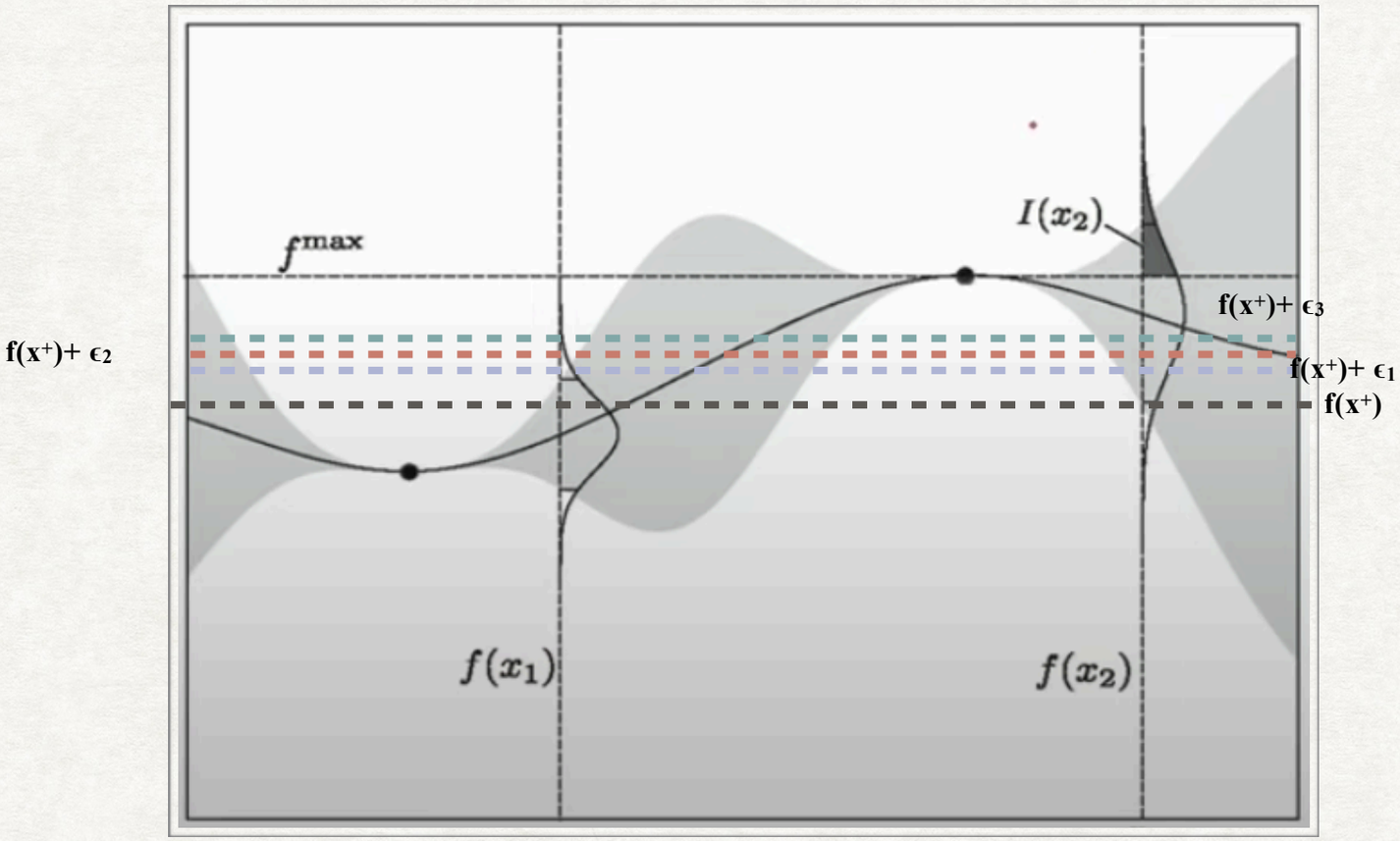
$\Phi(\cdot)$ indicates the CDF



ACQUISITION FUNCTION

PI (PROBABILITY OF IMPROVEMENT)

- PI uses ϵ to strike a balance between exploration and exploitation. Increasing ϵ results in querying locations with a larger variance as their probability density is spread



- As ϵ increases the area for exploitation remains (along x axis) same but area for exploration decreases (along y axis)

- Has been observed to work well only in cases when we already know function maximum

ACQUISITION FUNCTION

EI (EXPECTED IMPROVEMENT)

- PI only looked at how likely is an improvement, but, did not consider how much we can improve. Expected Improvement (EI), does exactly that. It choose the next query point as the one which has the highest expected improvement over the current max $f(x^+)$

- A/c to Expected Utility in decision theory, the expected value of an action to an action is calculated by multiplying the weight of each action by probability of that action, hence expectation is used

- It selects the point that minimizes the distance to the objective evaluated at the maximum.

- It is mathematically given as:

$$\begin{aligned} \mathbf{x}_{n+1} &= \arg\min_{\mathbf{x}} \mathbb{E}(\|f_{n+1}(\mathbf{x}) - f(\mathbf{x}^\star)\| \mid \mathcal{D}_n) \\ &= \arg\min_{\mathbf{x}} \int \|f_{n+1}(\mathbf{x}) - f(\mathbf{x}^\star)\| p(f_{n+1}|\mathcal{D}_n) df_{n+1} \end{aligned}$$

Where, f(x) is the actual ground truth function, f_{n+1} is the posterior mean of the surrogate, D_n is the training data, $x^\star$ is the actual position where f(x) takes the maximum value.

Since, we do not know $x^\star$, the following acquisition function is used to overcome the issue.

$$\mathbf{x} = \arg\max_{\mathbf{x}} \mathbb{E}(\max\{0, f_{n+1}(\mathbf{x}) - f^{\max}\} \mid \mathcal{D}_n)$$

Where, f^{max} is the current maximum value

ACQUISITION FUNCTION

EI (EXPECTED IMPROVEMENT)

Using a GP as a surrogate the expression above converts to:

$$EI(x) = \begin{cases} (\mu_t(x) - f(x^+) - \epsilon)\Phi(Z) + \sigma_t(x)\phi(Z), & \text{if } \sigma_t(x) > 0 \\ 0, & \text{if } \sigma_t(x) = 0 \end{cases}$$

$$Z = \frac{\mu_t(x) - f(x^+) - \epsilon}{\sigma_t(x)}$$

where $\Phi(\cdot)$ indicates CDF and $\phi(\cdot)$ indicates pdf.

GP-UCB (GAUSSIAN PROCESS-UPPER CONFIDENCE BOUND)

- This acquisition function considers linear combination of the mean and uncertainty of surrogate model.
- $\alpha(x) = -\mu(x) + \kappa \times \sigma(x)$
- The intuition behind the UCB acquisition function is weighing of the importance between the posterior mean vs. the posterior uncertainty.
- The κ above is the hyper-parameter that can control the preference between mean and variance.

CLUSTERING GUIDED GP-UCB

WHY THIS WORK

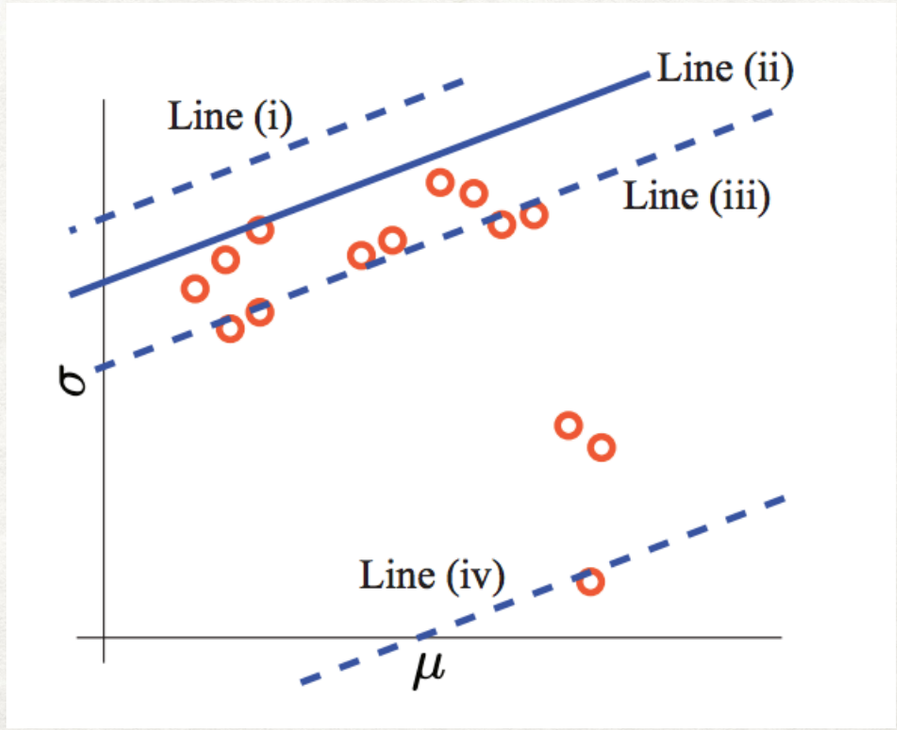
- Presents A novel geometric view of GP-UCB based on posterior mean and variance has been presented in the paper
- Reduces the search space of GP-UCB acquisition function to single cluster chosen by method proposed

GEOMETRIC VIEW OF GP-UCB BASED ON POSTERIOR MEAN AND VARIANCE

- GP-UCB Acquisition function: $a(x|D_{1:t}) = -\mu(x) + \kappa\sigma(x)$,
- Rewriting the above equation we get:

$$\sigma(x) = 1/\kappa \ (\mu(x) + a(x|D_{1:t}))$$

Same as the equation of line with y axis as $\sigma(x)$ and x axis as $\mu(x)$



CLUSTERING GUIDED GP-UCB

WORK IN DETAIL

Take a few samples values & get their function evaluations

↓

Fit a Gaussian prior on these sample values

↓

Obtain posterior mean and posterior variance for given set of obeservations

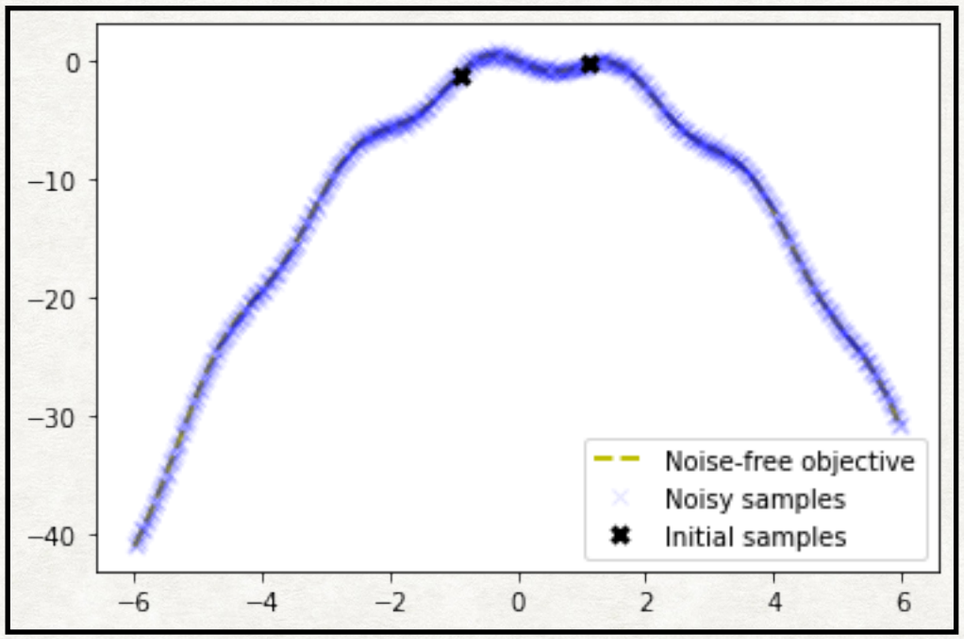
↓

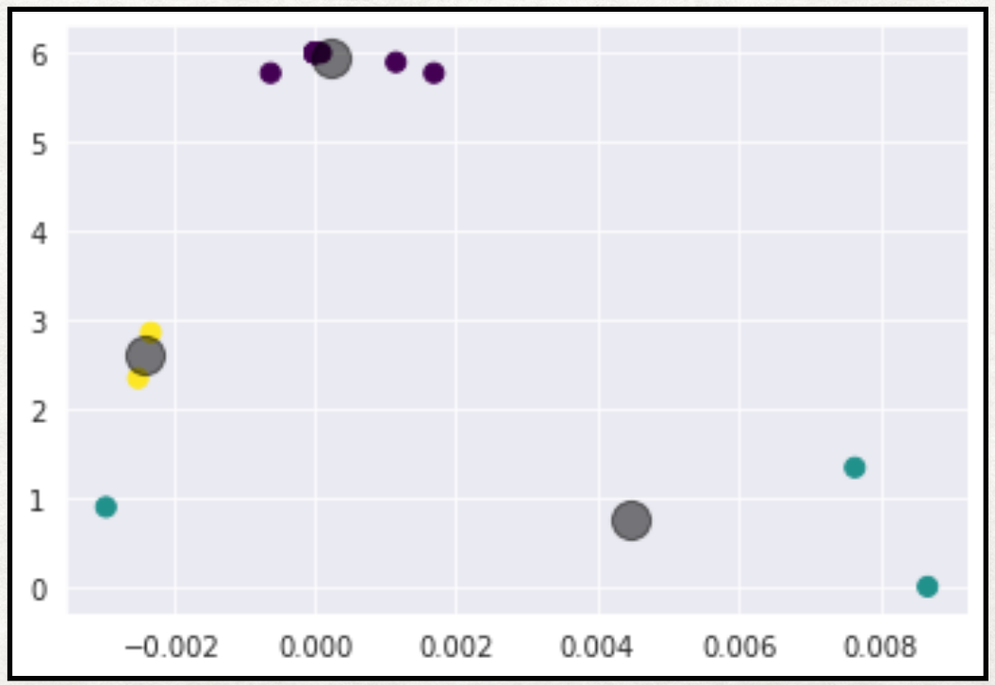
Randomly sample a few points from posterior mean and variance and plot them in these in μ-σ plane

↓

Cluster these points and calculate their centres

↓





CLUSTERING GUIDED GP-UCB

WORK IN DETAIL

Find the best cluster C_{i*}

↓

Calculate x_{1+t}

↓

Sample objective function $y_{1+t} = f_{1+t} + \epsilon_{1+t}$

↓

Update GP with new dataset containing $\{D_{1:t}, (x_{1+t}, y_{1+t})\}$

↓

Repeat steps 4-9 for T iterations

CLUSTERING GUIDED GP-UCB

WORK IN DETAIL

Algorithm 2 Bayesian Optimization with CG-GPUCB

Require: Initial data $\mathcal{D}_{1:I} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_I, y_I)\}$, $T \in \mathbb{N} > 0$, and $K \in \mathbb{N} > 0$ (number of clusters)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Calculate centers $\mathbf{c}_i$ of K clusters determined in the μ-σ space, given the current GP.
3: Find the best cluster $\mathcal{C}_{i*}$ via (9).
4: Find $\mathbf{x}_{I+t}$ by (10) or (11).
5: Sample the objective function: $y_{I+t} = f(\mathbf{x}_{I+t}) + \epsilon_{I+t}$.
6: Augment the data: $\mathcal{D}_{1:I+t} = \{\mathcal{D}_{1:I+t-1}, (\mathbf{x}_{I+t}, y_{I+t})\}$.
7: Update the GP via (4) & (5).
8: **end for**

EXPERIMENTS

We ran GPUCB^2, GPUCB_NN, GPUCB, GPUCB_MCMC, E1, EI_MCMC, PI, PI_MCMC on the synthetic function -np.sin(3*X) - X**2 + 0.7*X + noise * np.random.randn(*X.shape)

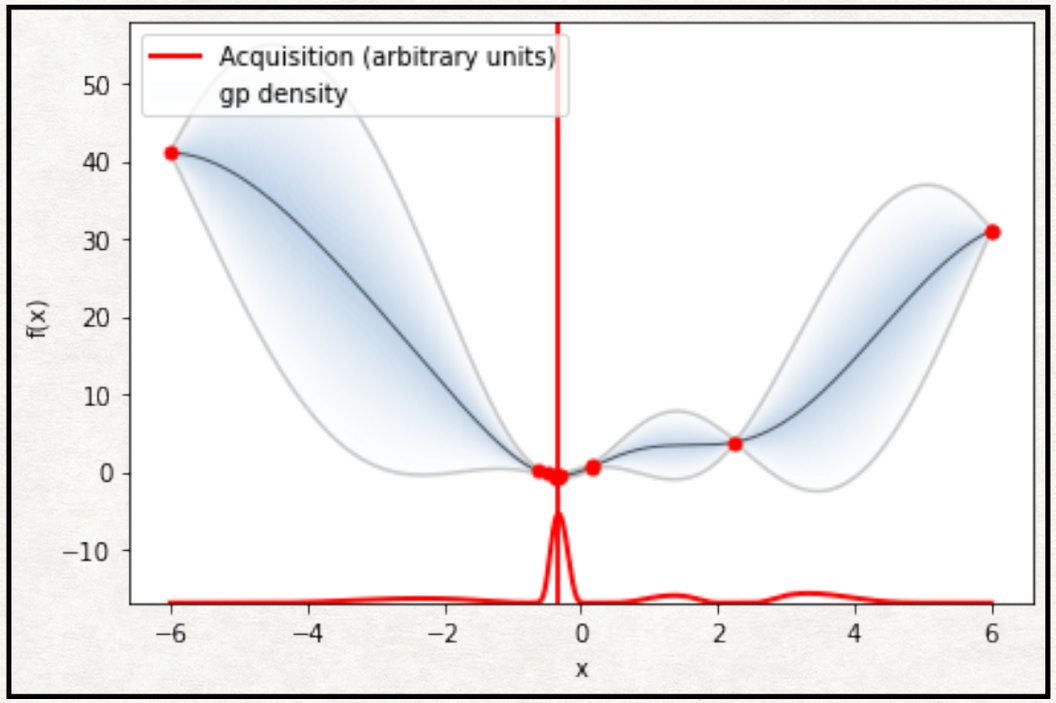
	GPUCB_Scratch	GPUCB_Gpyopt	GPUCB_MCMC	EI	EI_MCMC	PI	PI_MCMC	GPUCB^2_ra	GPUCB^2_gp	GPUCB_NN_ra	GPUCB_NN_gp
f(min)	-5.01	-0.52	-0.6	0.2	-0.6	-0.1	-0.6	-5.96	-5.97	-5.96	-5.99

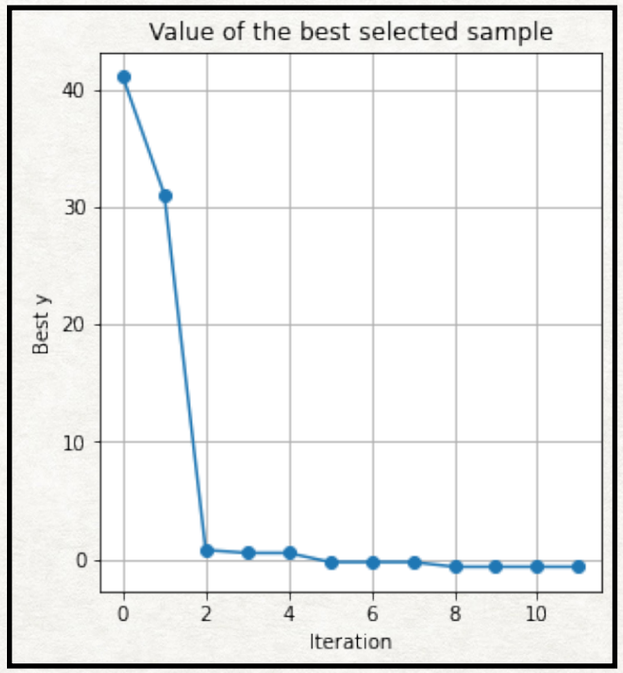
We performed these experiments using random sampling & Selecting using GPUCB

In later case search space reduces further which is beneficial while performing experiments on complex functions as shown later

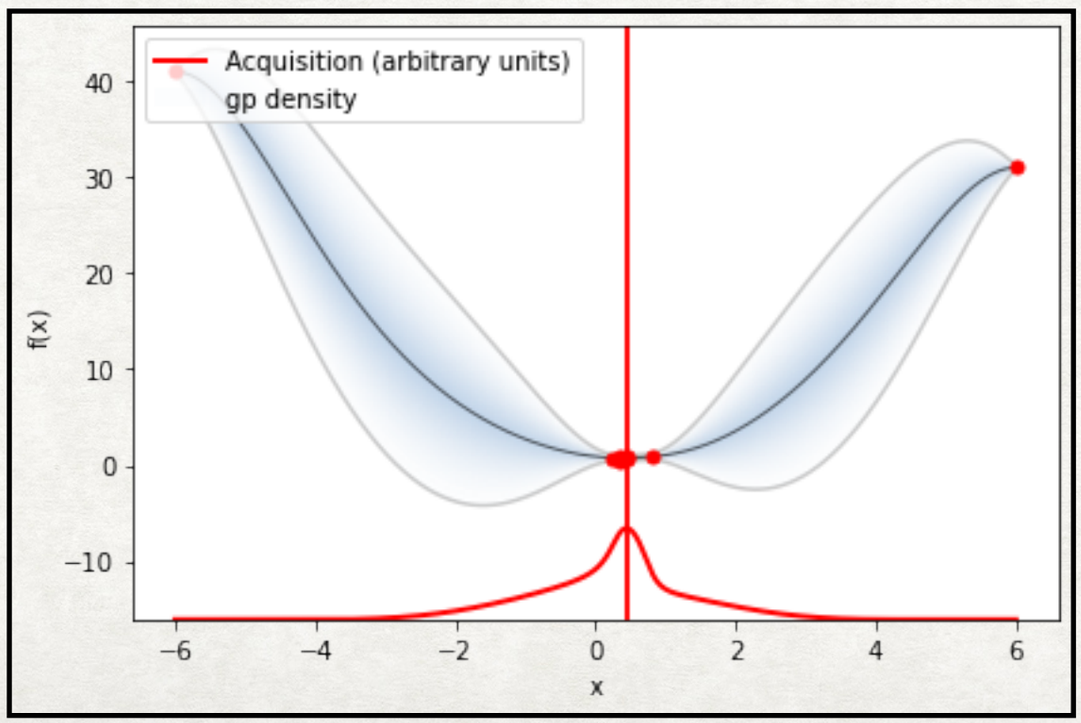
RESULTS

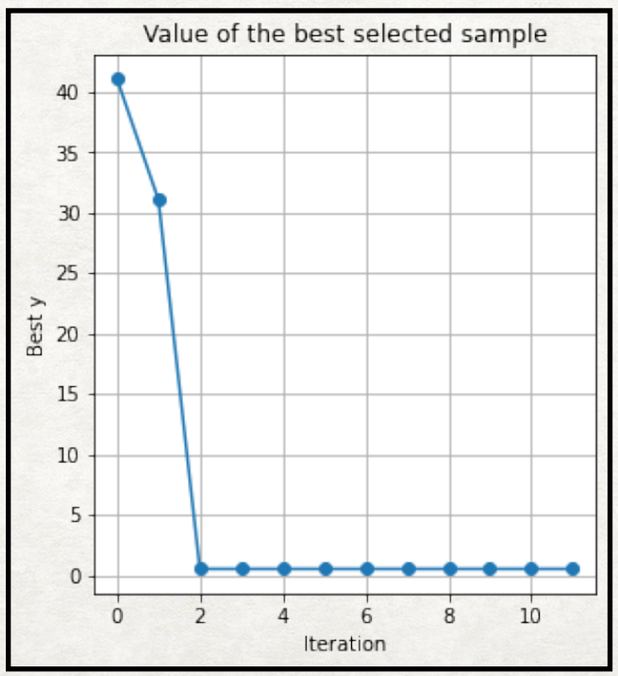
PI





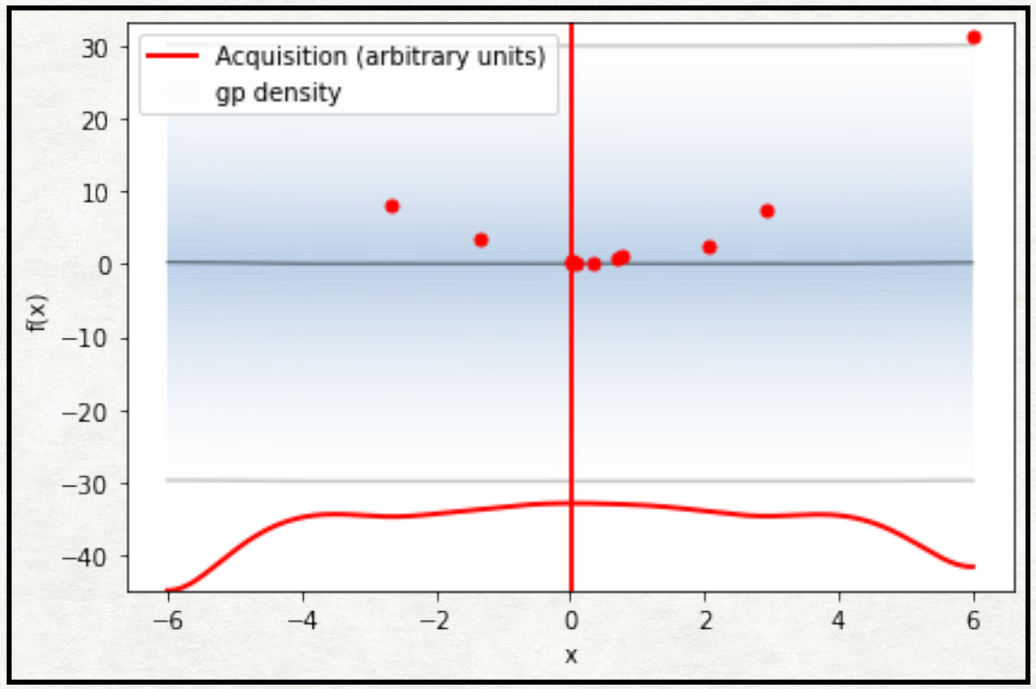
PI_MCMC

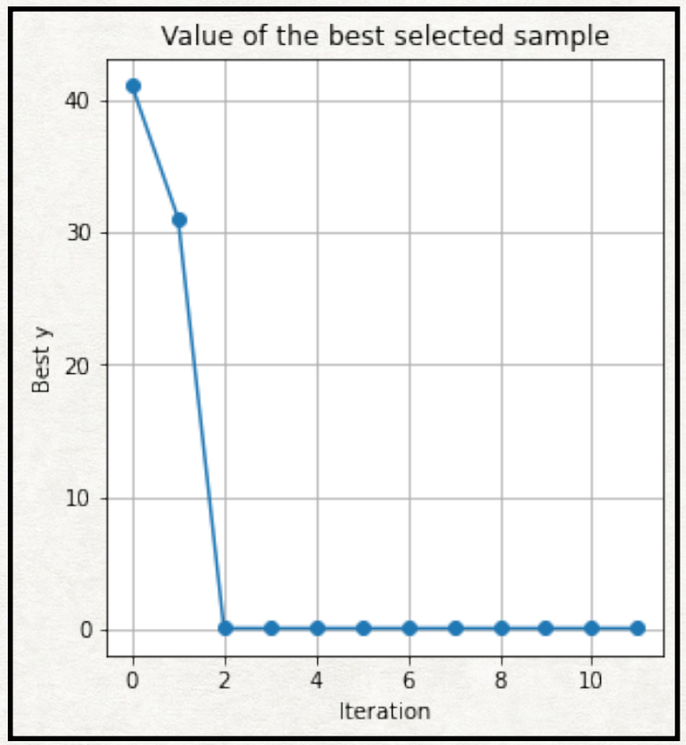




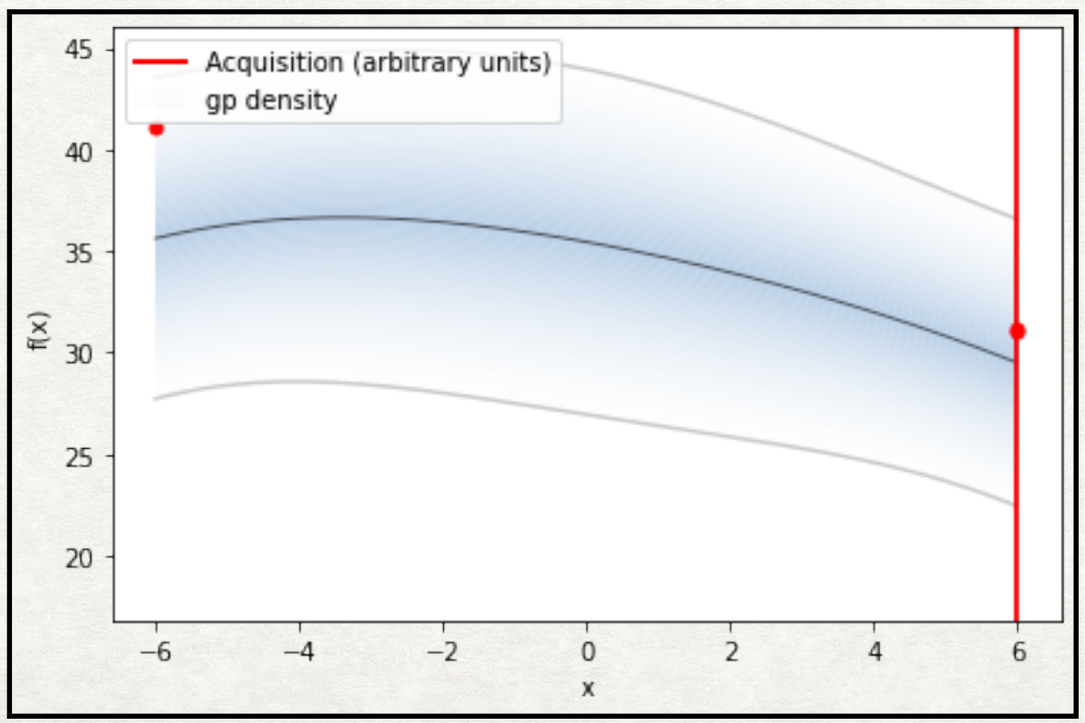
RESULTS

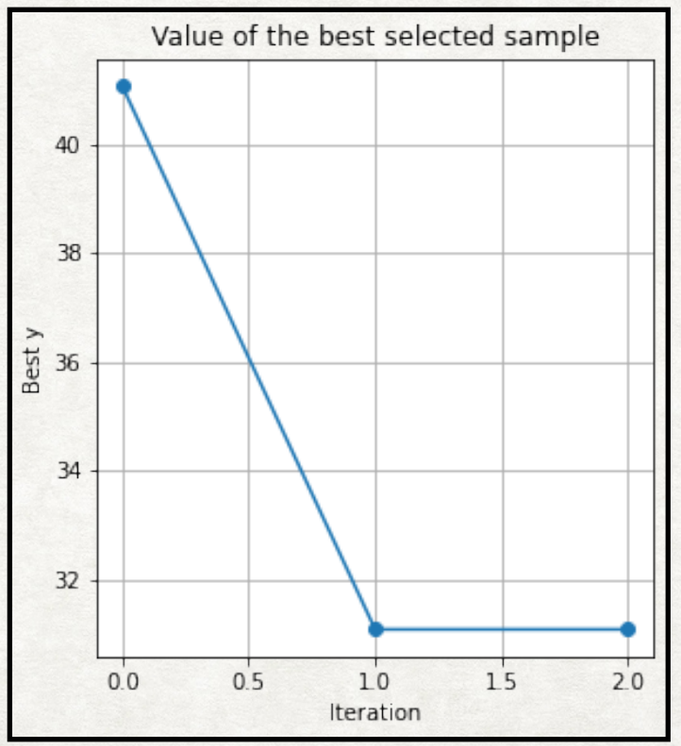
EI





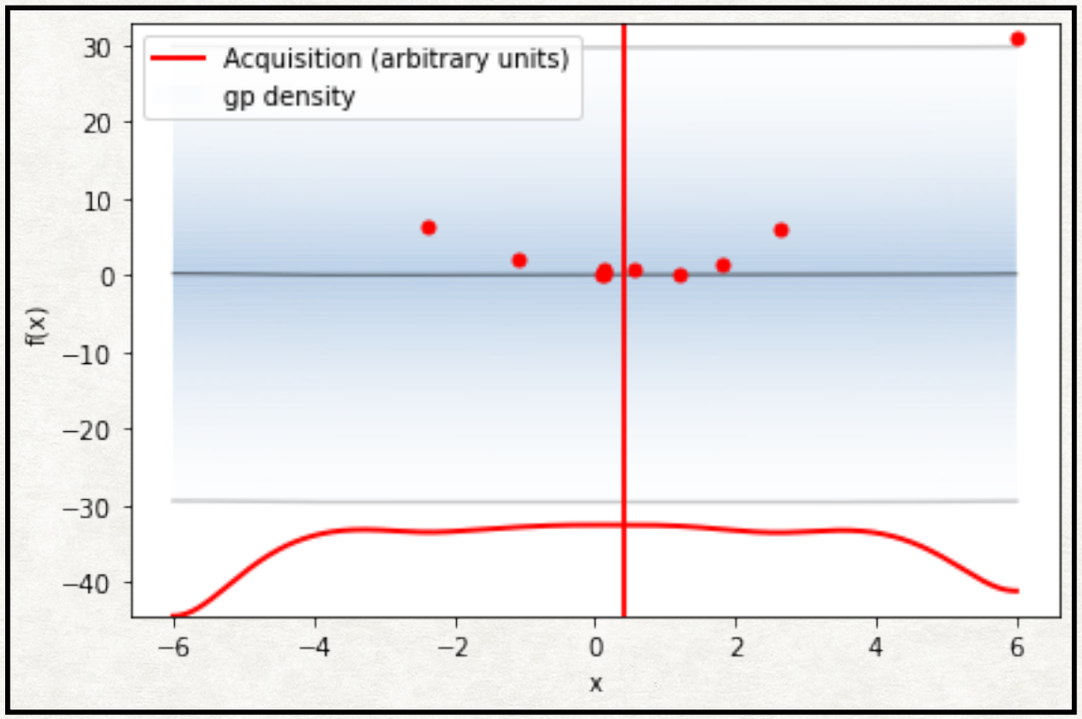
EI_MCMC

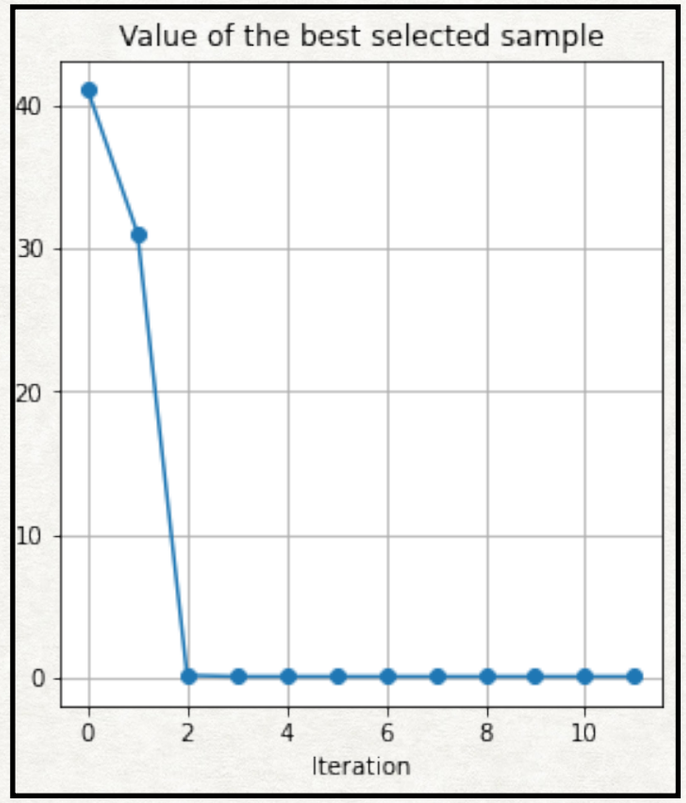




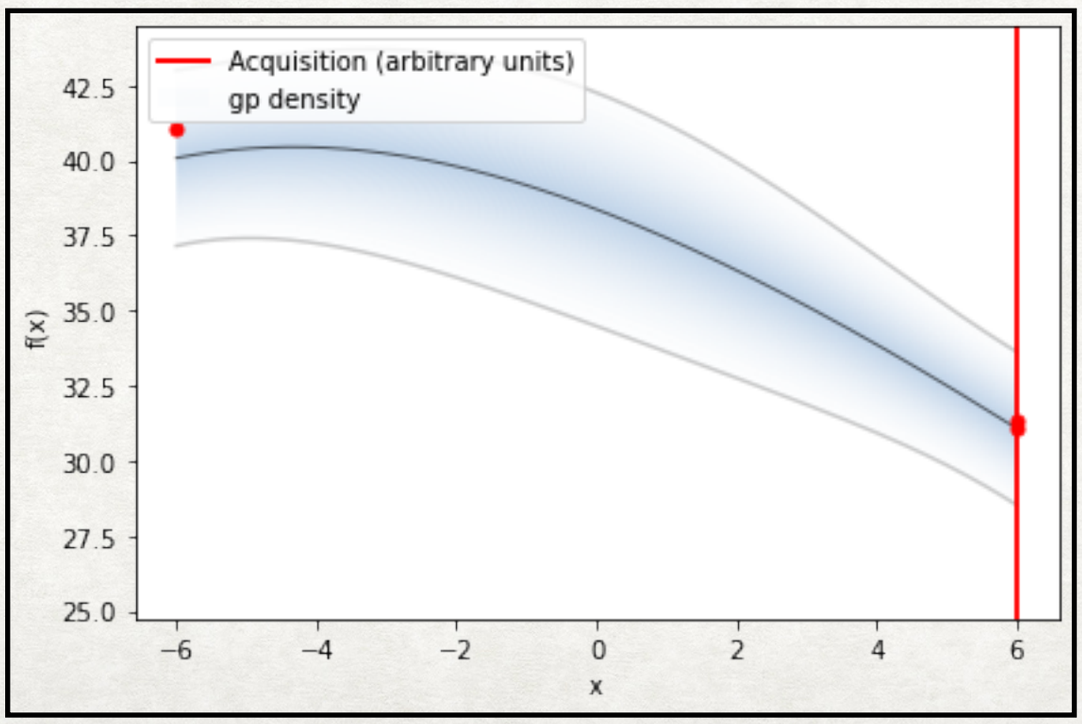
RESULTS

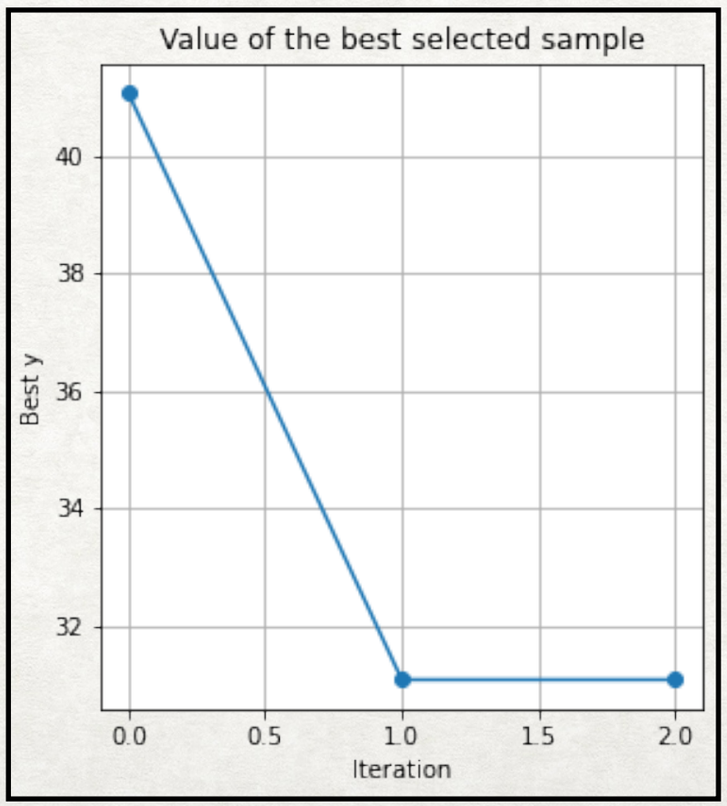
LCB





LCB_MCMC

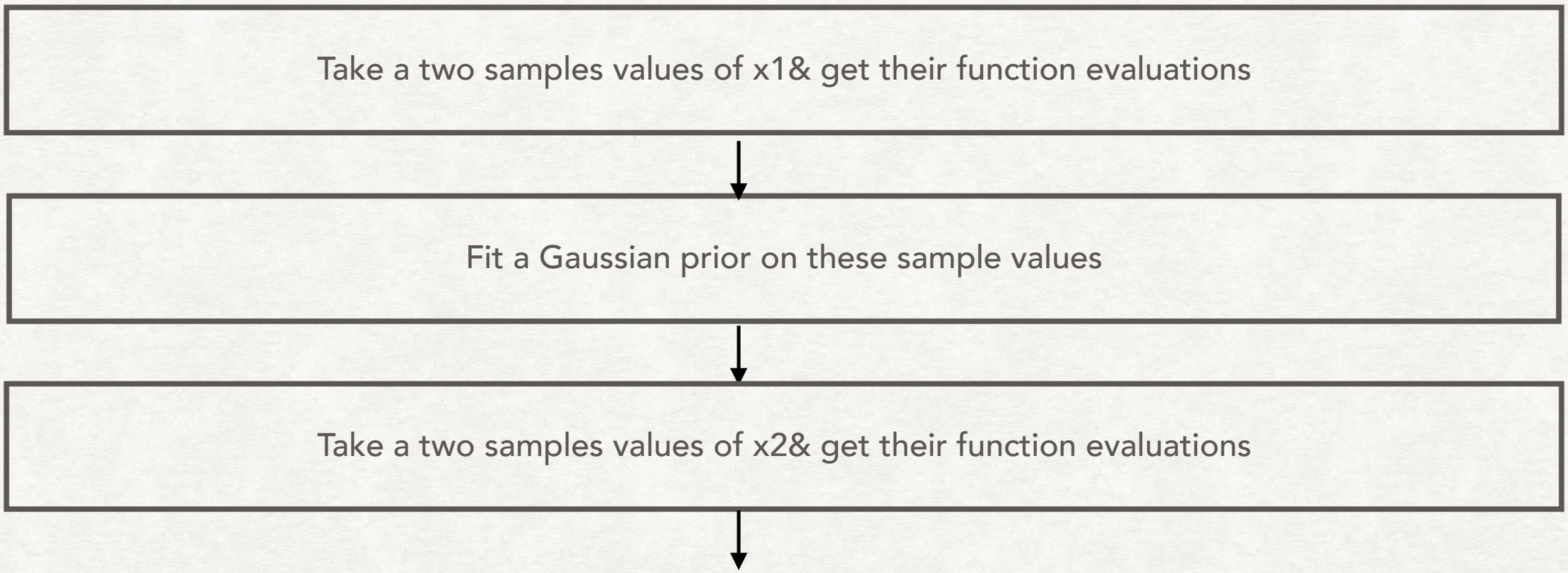




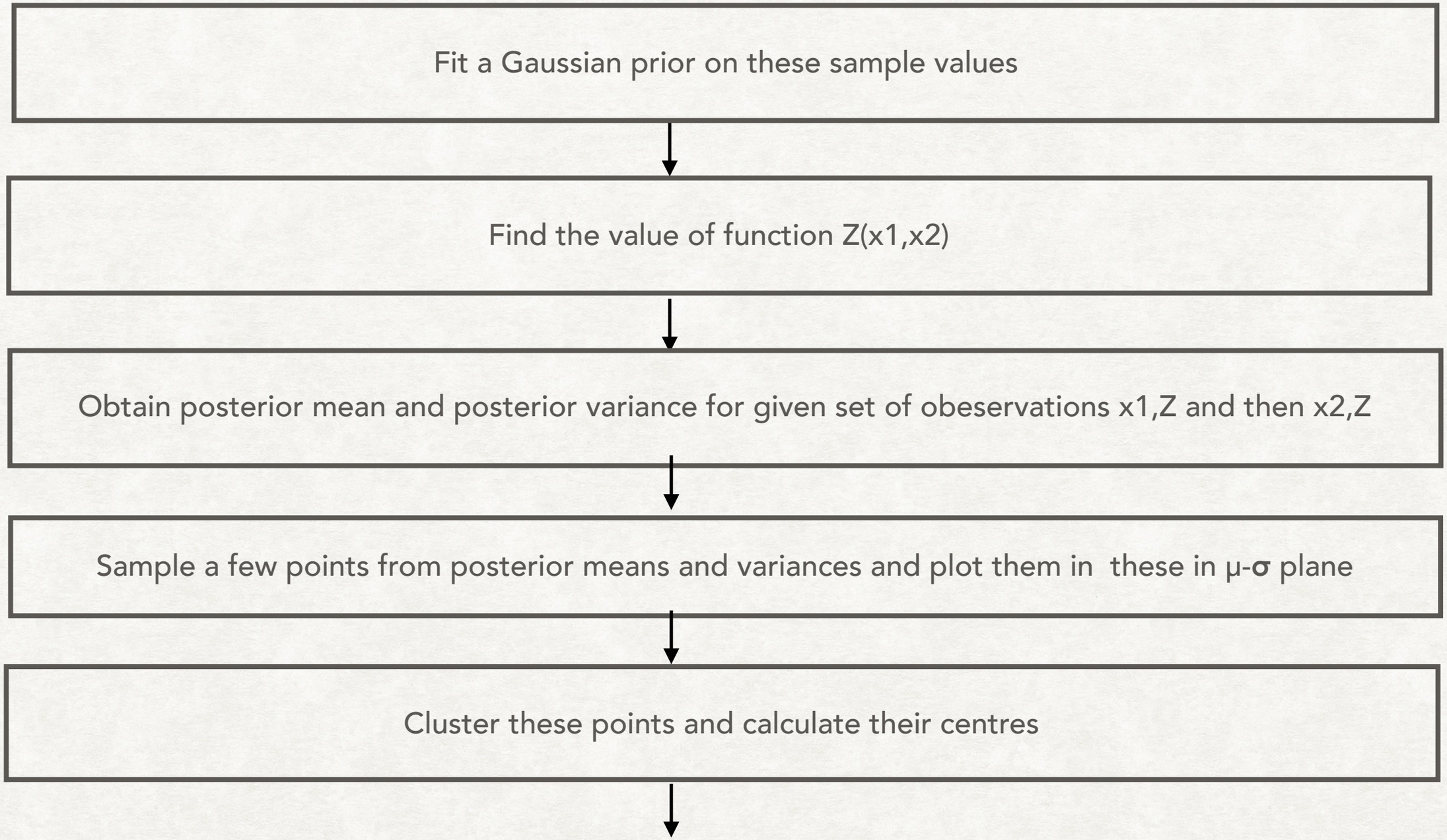
EXPERIMENTS

We also mentioned acquisition functions on three Baseline multivariate functions namely Branin Hoo Function, Six-Hump Camel Function, Egg Holder Function

In the paper authors don't provide algorithm to work with multi-variate functions, so we update algo as follows



EXPERIMENTS



EXPERIMENTS

Find the best cluster C_{i*} for both x1 and x2 axis

↓

Calculate $x1_{1+t}$, $x2_{1+t}$

↓

Sample objective function $z_{1+t} = f(x1,x2) + \epsilon_{1+t}$

↓

Update GP with new dataset containing $\{D_{1:t}, (x_{1+t}, y_{1+t})\}$

↓

Repeat steps 4-9 for T iterations

EXPERIMENTS

BRANIN HOO FUNCTION

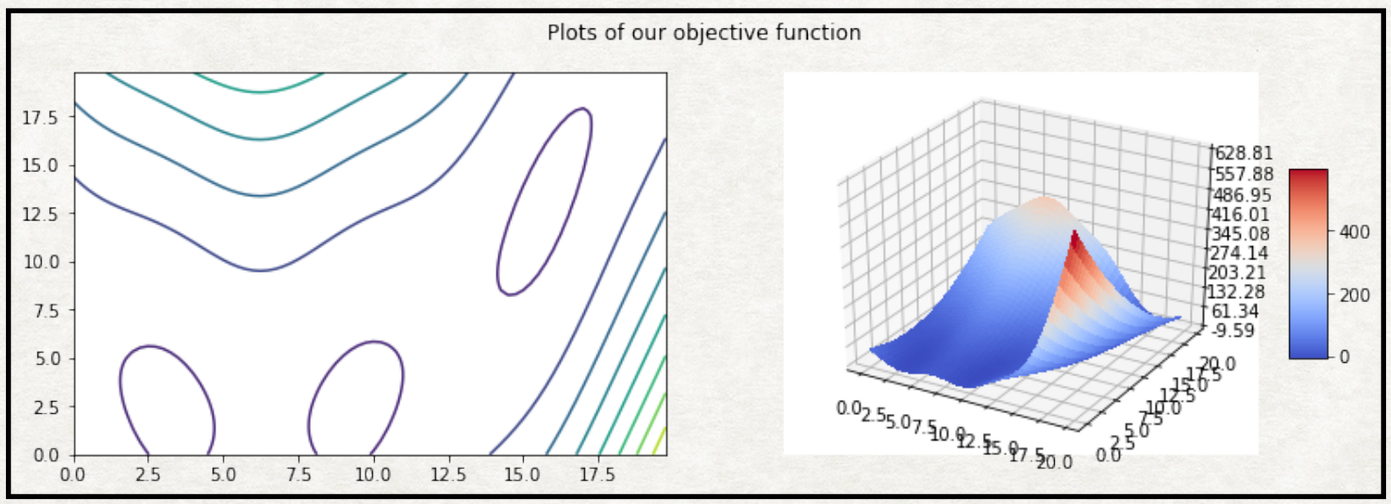
Mathematical Representation

```
def objfunc2d(x1,x2):


    a = 1
    b = 5.1/(4*np.pi**2)
    c = 5/np.pi
    r = 6
    s = 10
    t = 1/(8*np.pi)

    return (a * (x2 - b*x1**2 + c*x1 - r)**2)+(s*(1-t)*np.cos(x1))
```

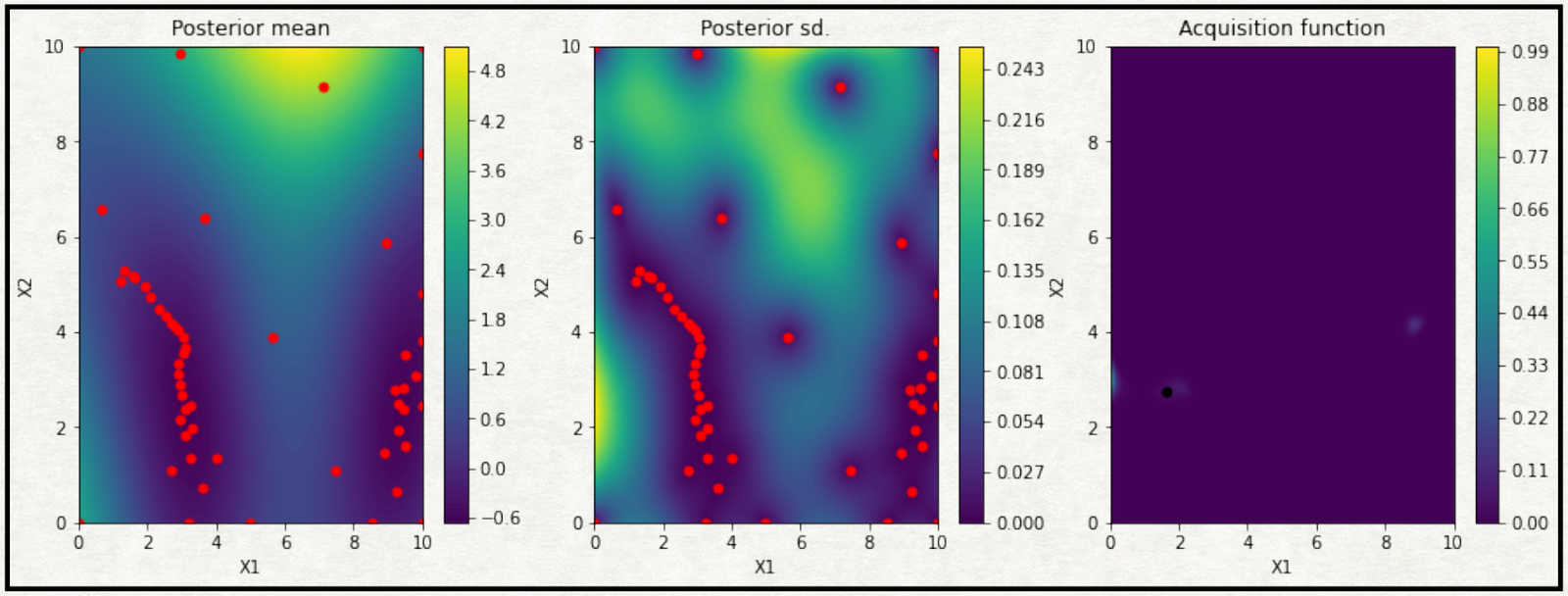
Vizualization

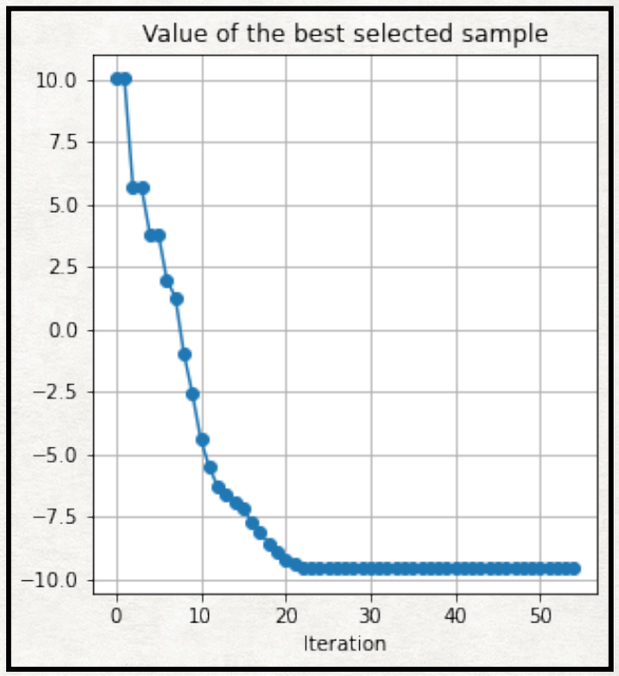


EXPERIMENTS

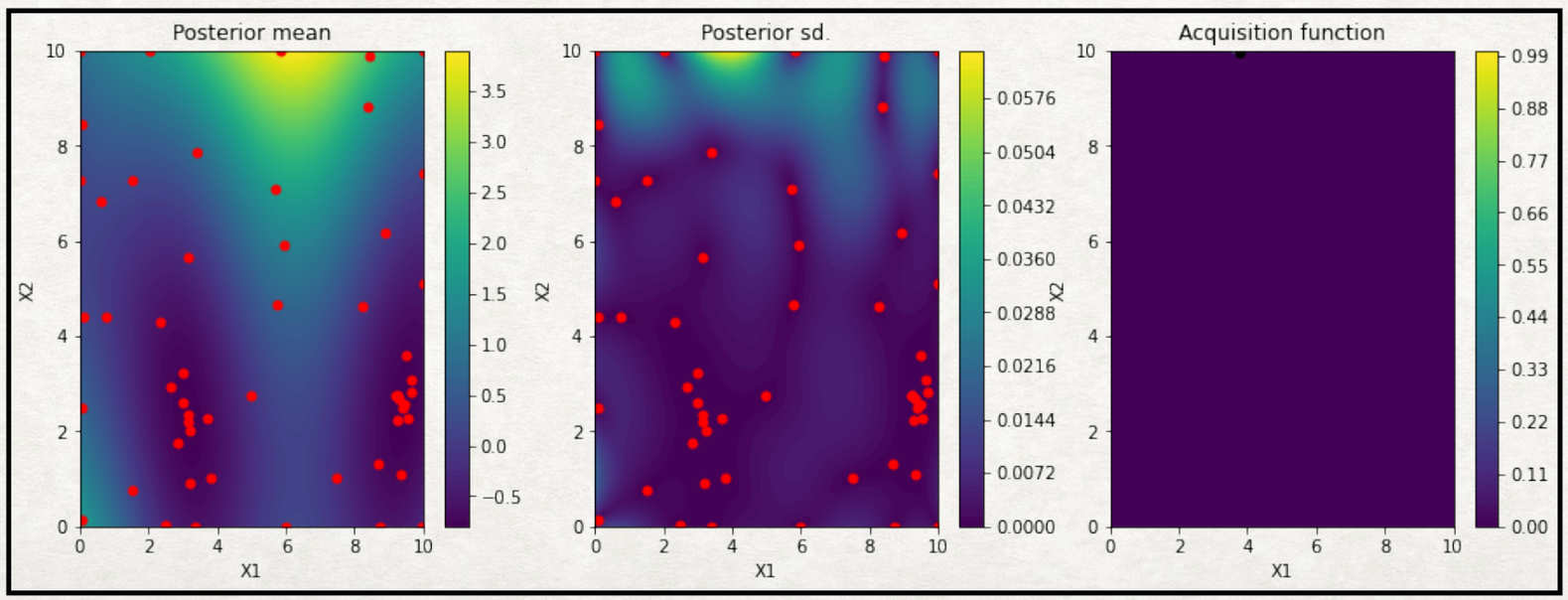
BRANIN HOO FUNCTION-RESULTS

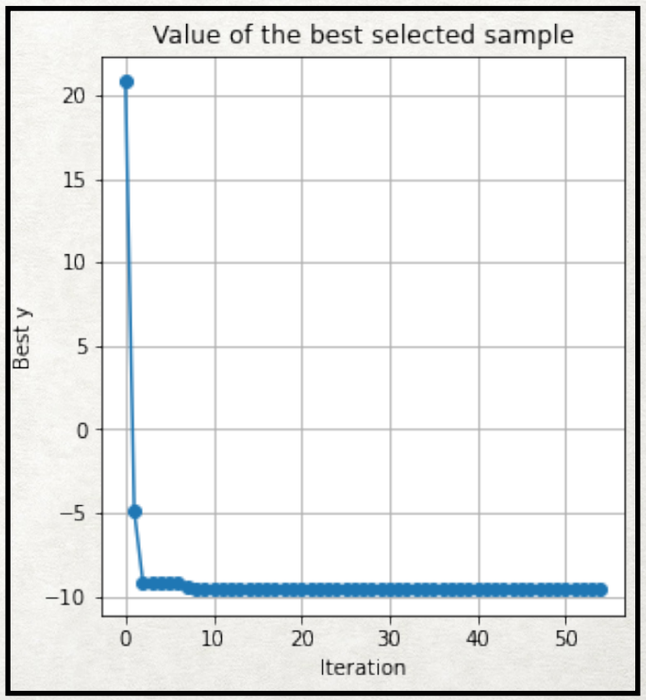
PI





PI_MCMC

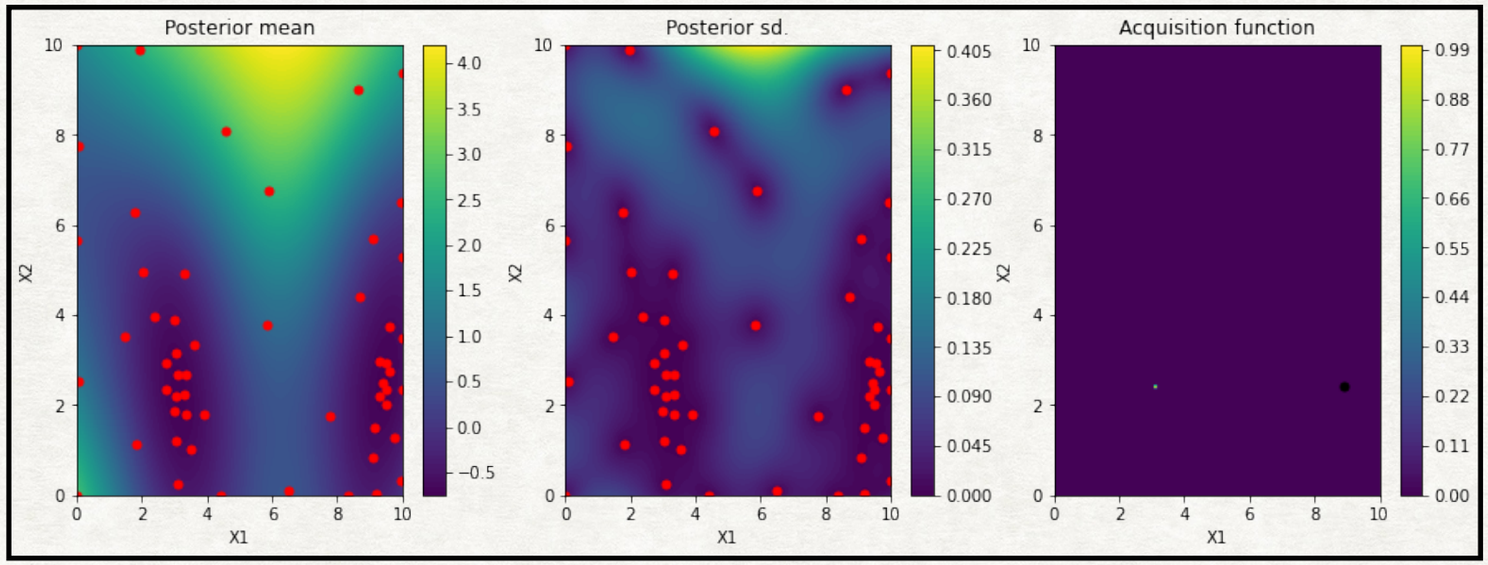


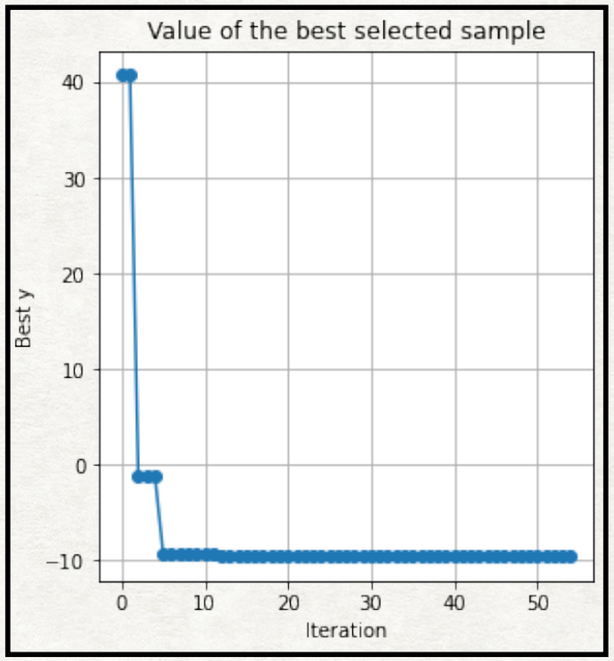


EXPERIMENTS

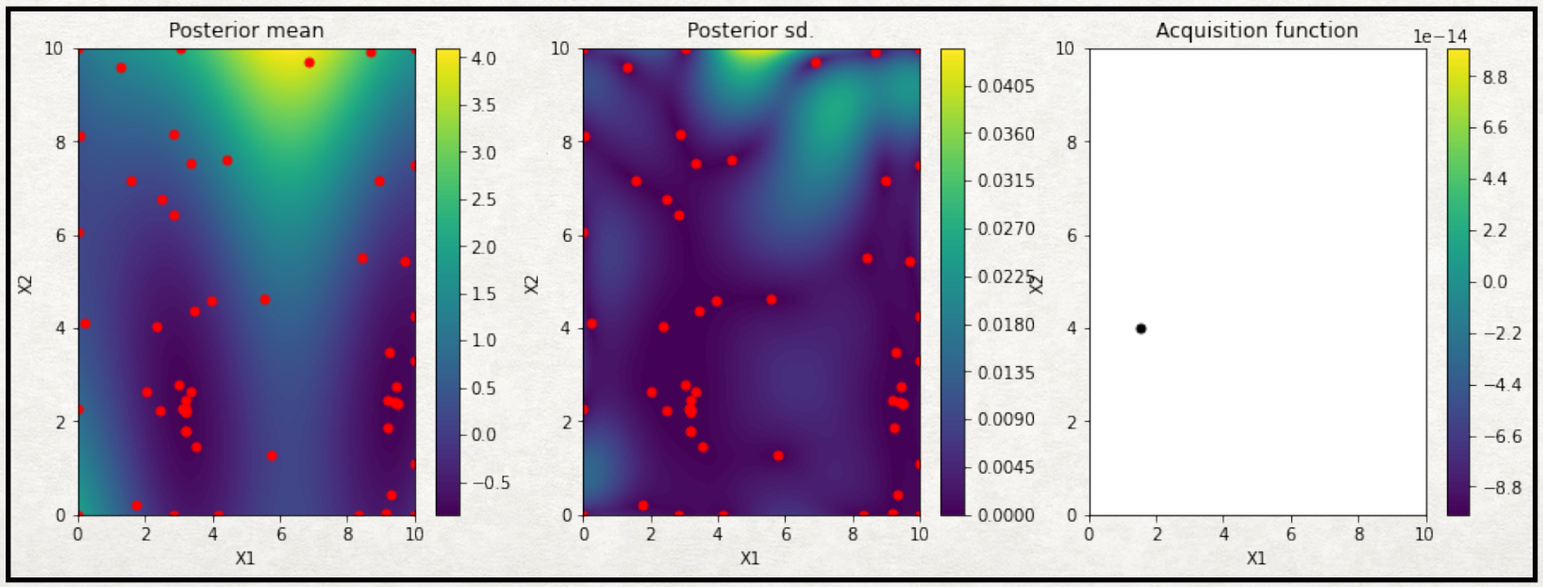
BRANIN HOO FUNCTION-RESULTS

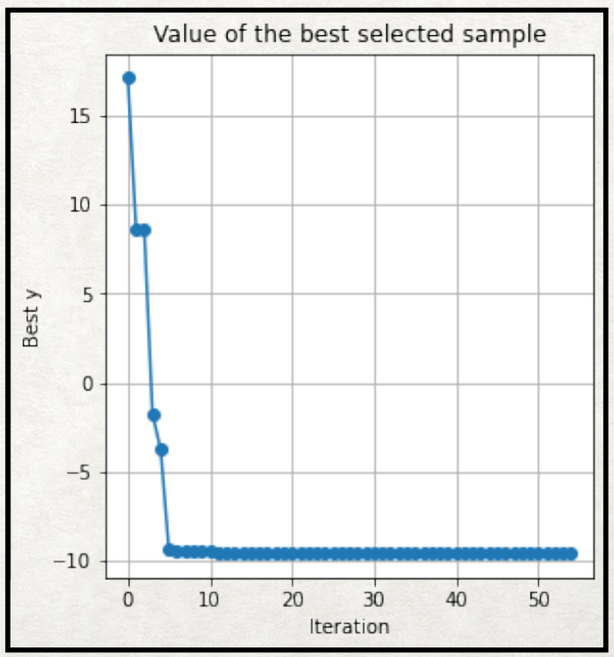
EI





EI_MCMC

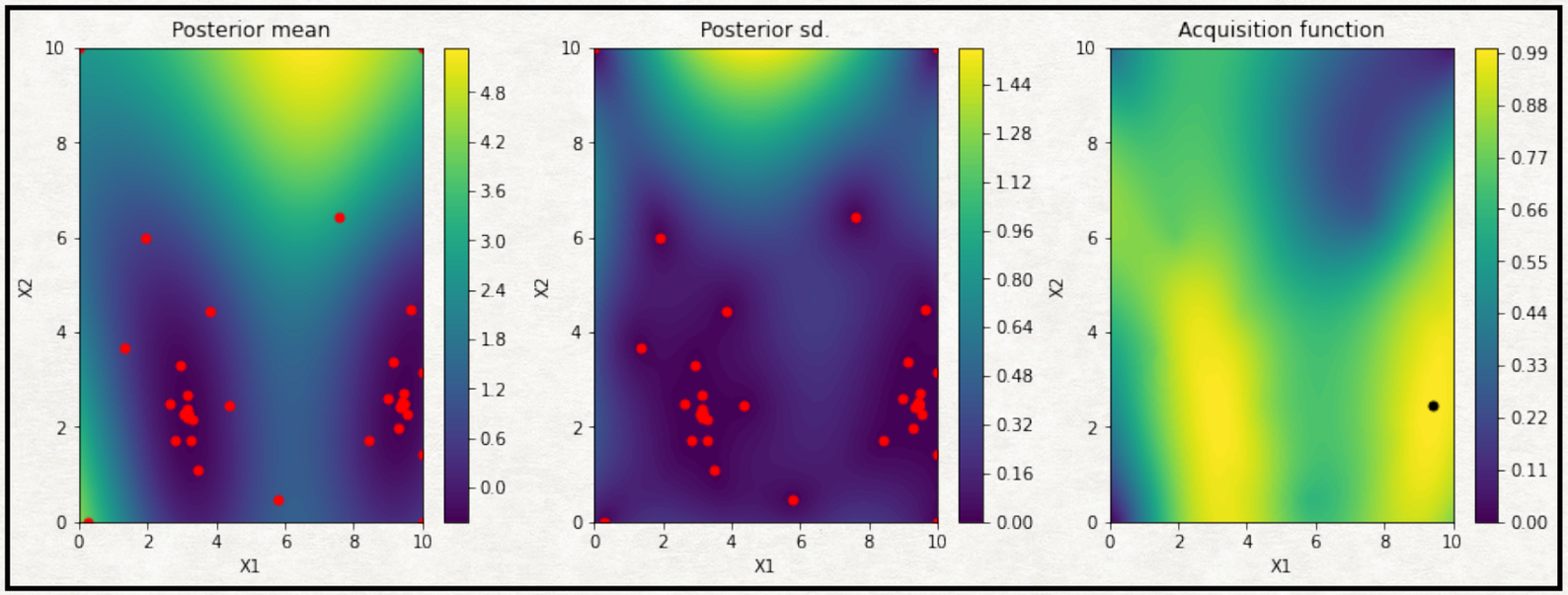


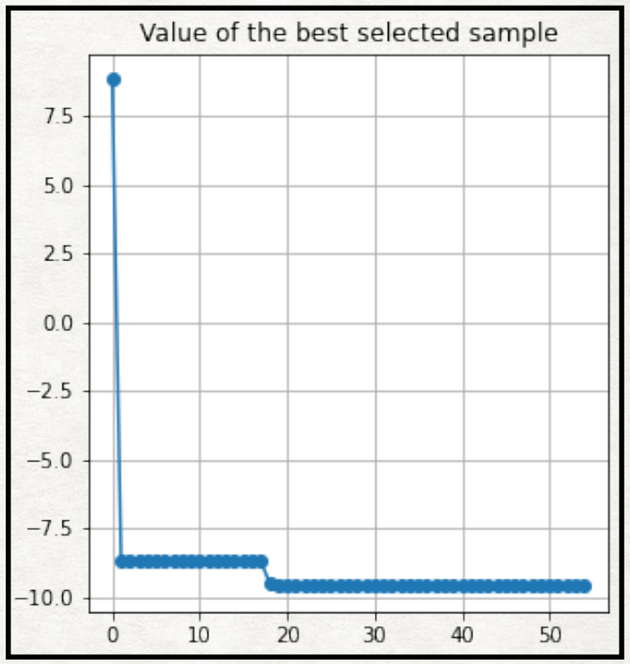


EXPERIMENTS

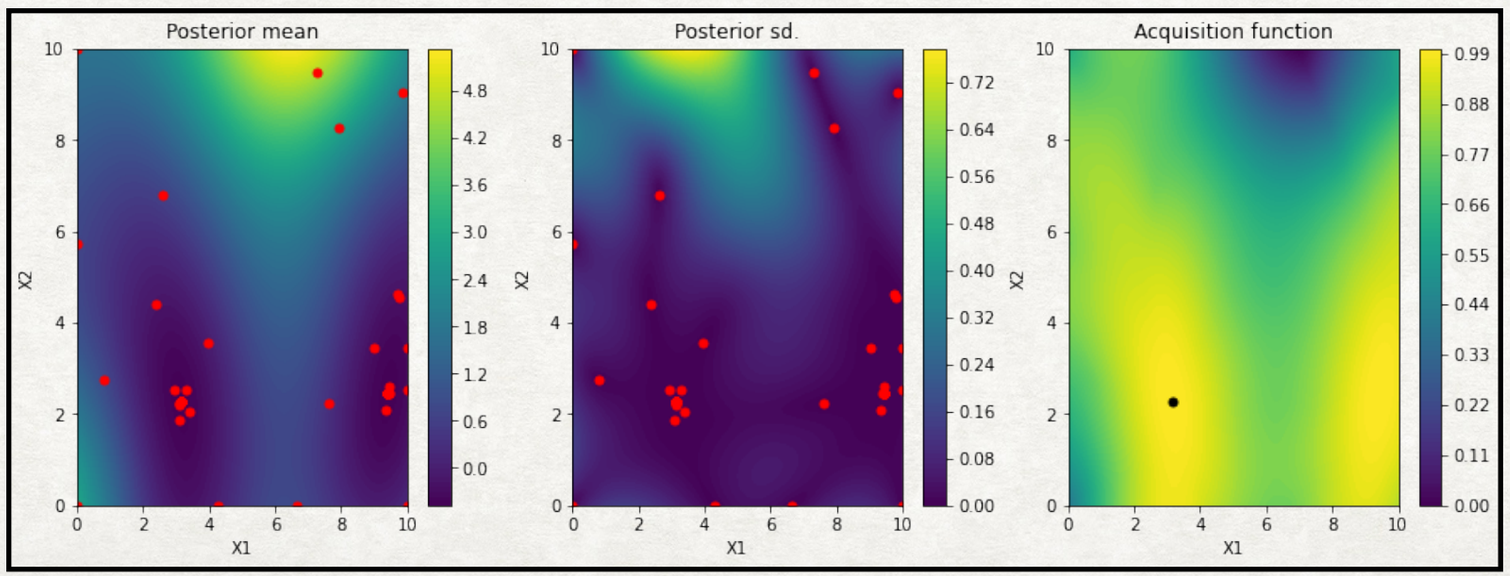
BRANIN HOO FUNCTION-RESULTS

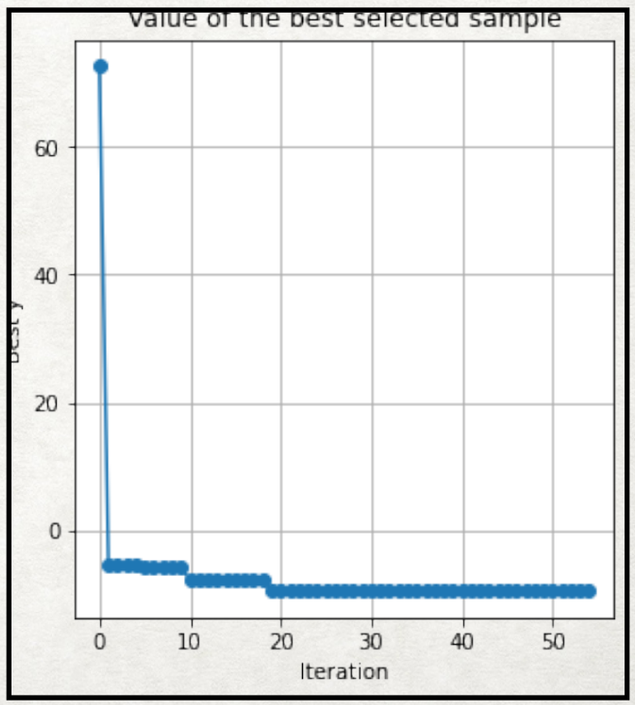
UCB





UCB_ MCMC





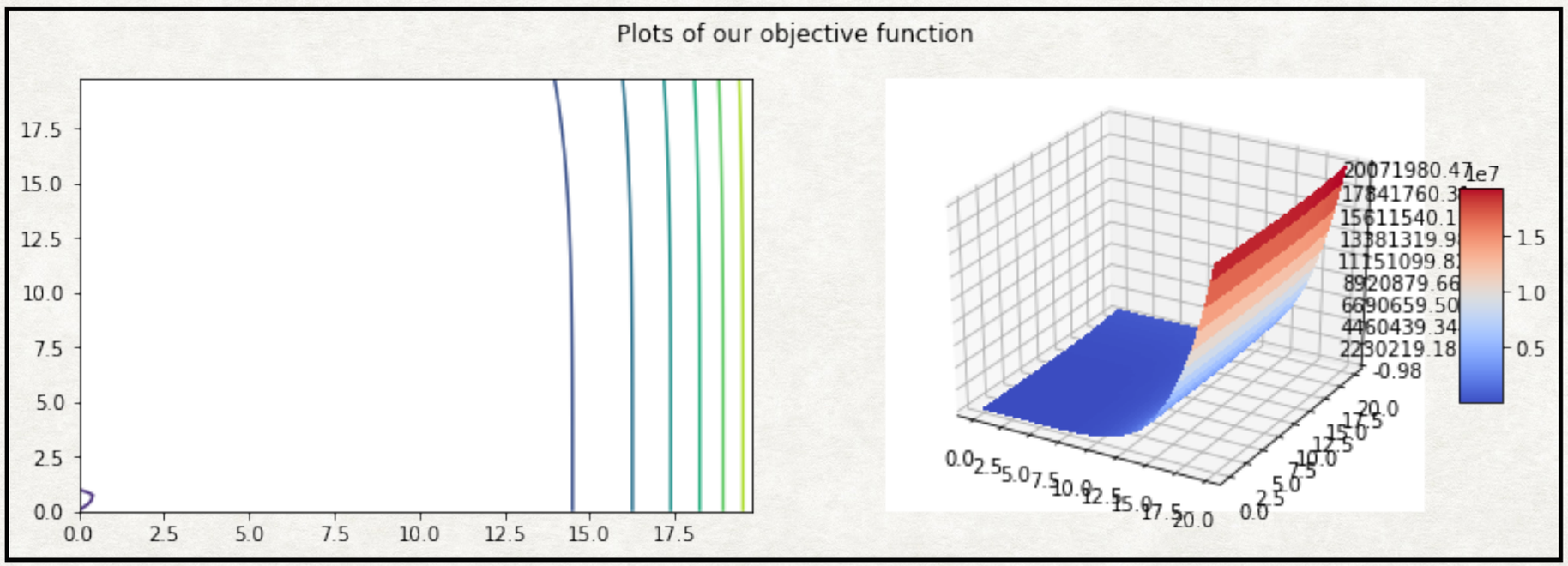
EXPERIMENTS

SIX-HUMP CAMEL FUNCTION

Mathematical Representation

```
def objfunc2d(x1,x2):
  return((4-2.1*(x1**2)+(x1**4)/3) * x1**2)+ (x1*x2)+((-4+4*x2**2) * x2**2)
```

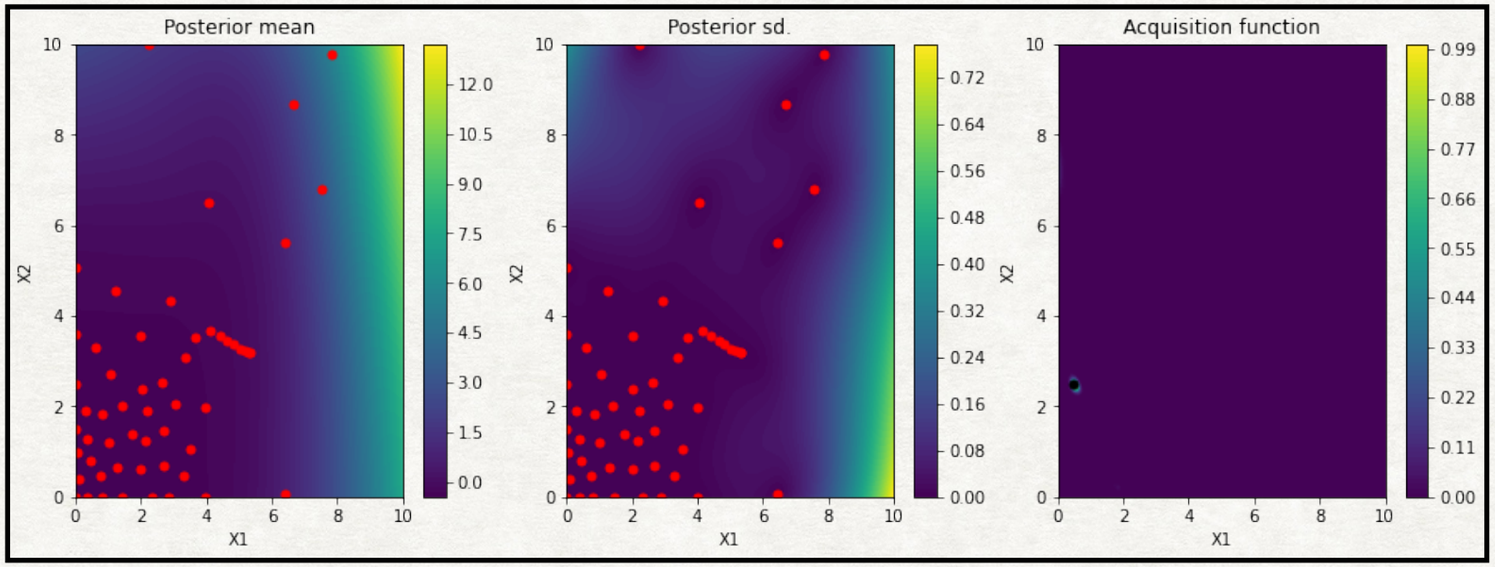
Vizualization

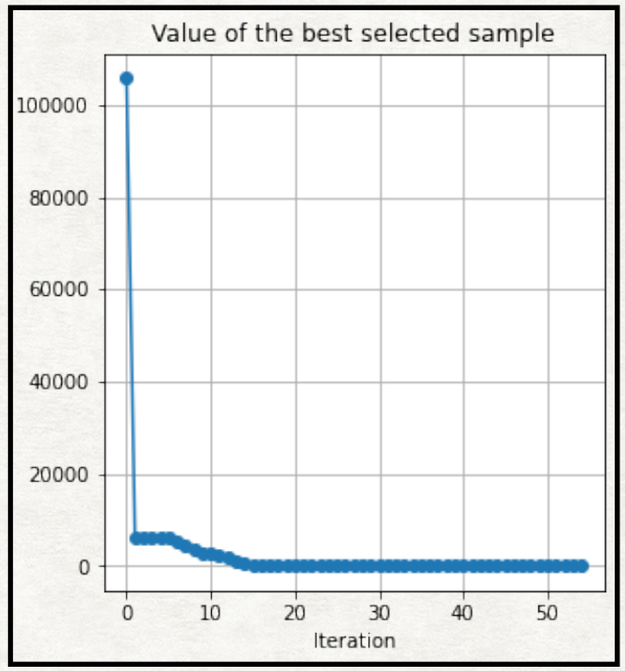


EXPERIMENTS

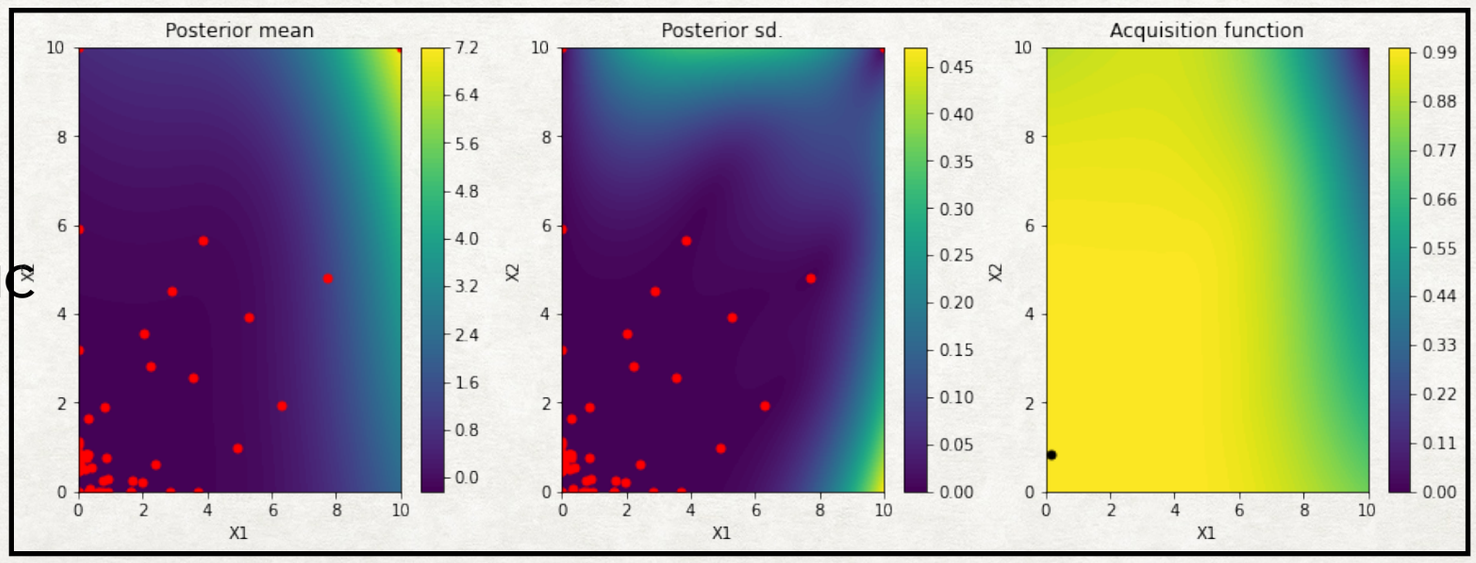
SIX-HUMP CAMEL FUNCTION-RESULTS

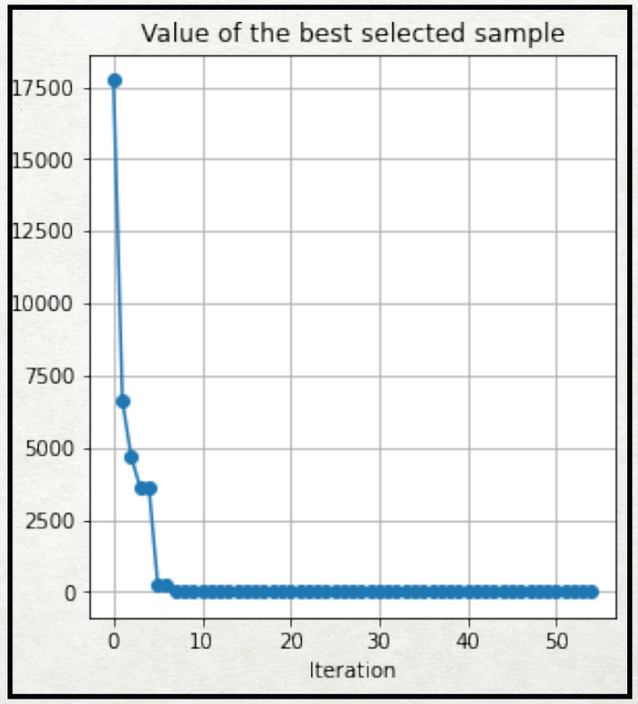
PI





PI_MCMC

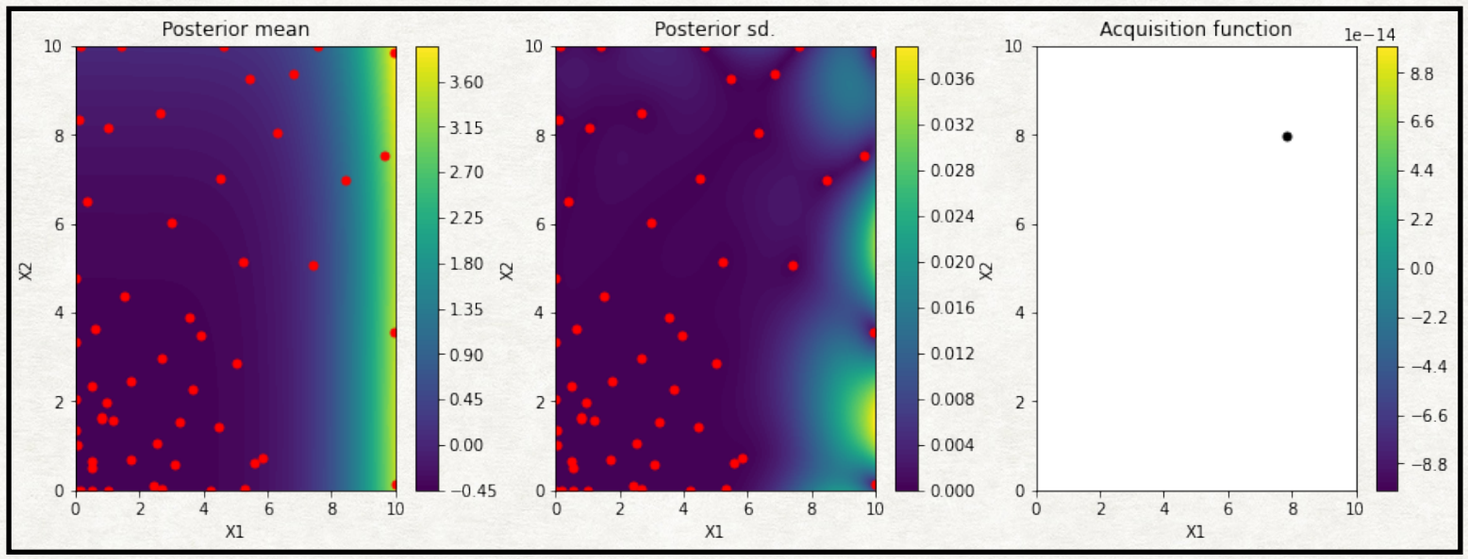


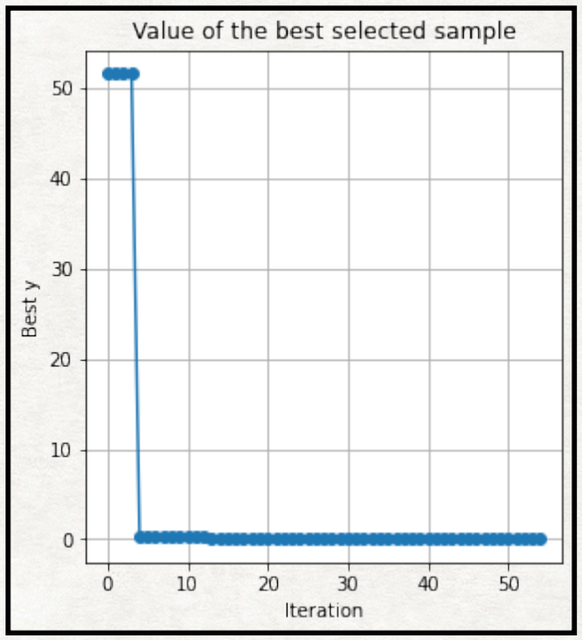


EXPERIMENTS

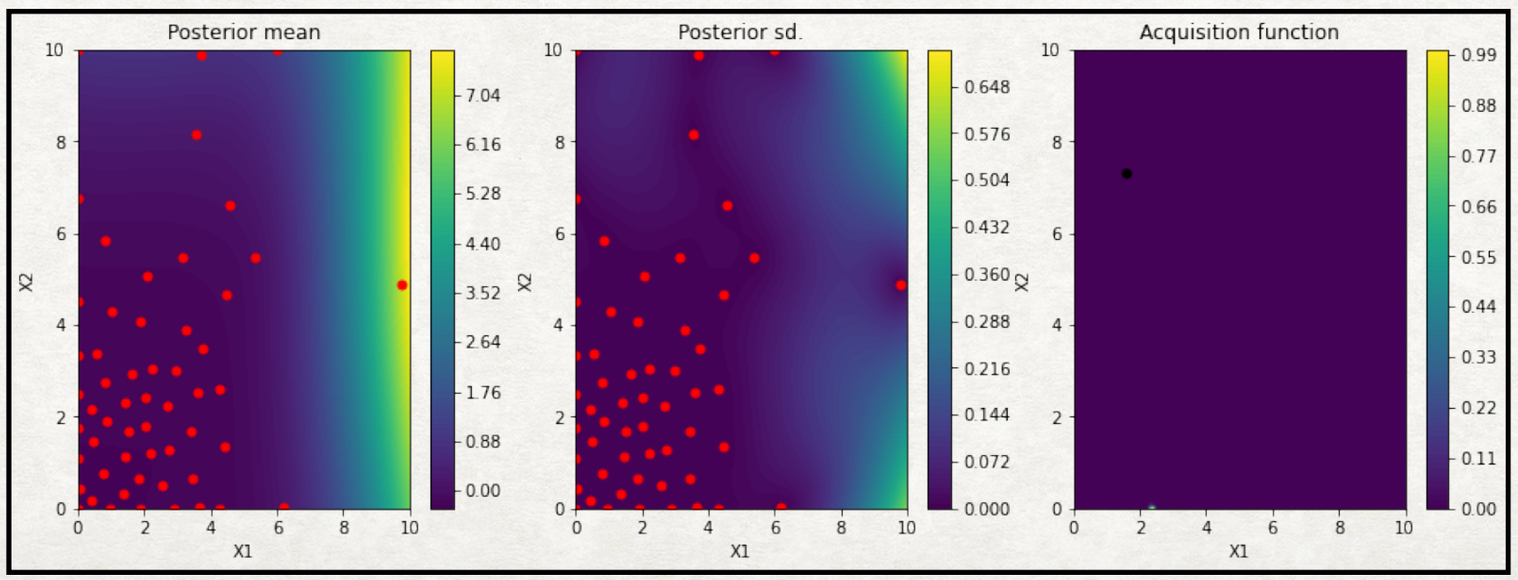
SIX-HUMP CAMEL FUNCTION-RESULTS

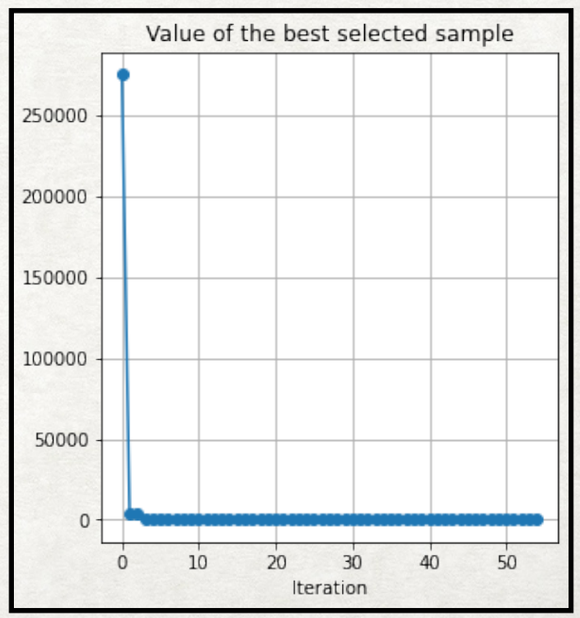
EI





EI_MCMC

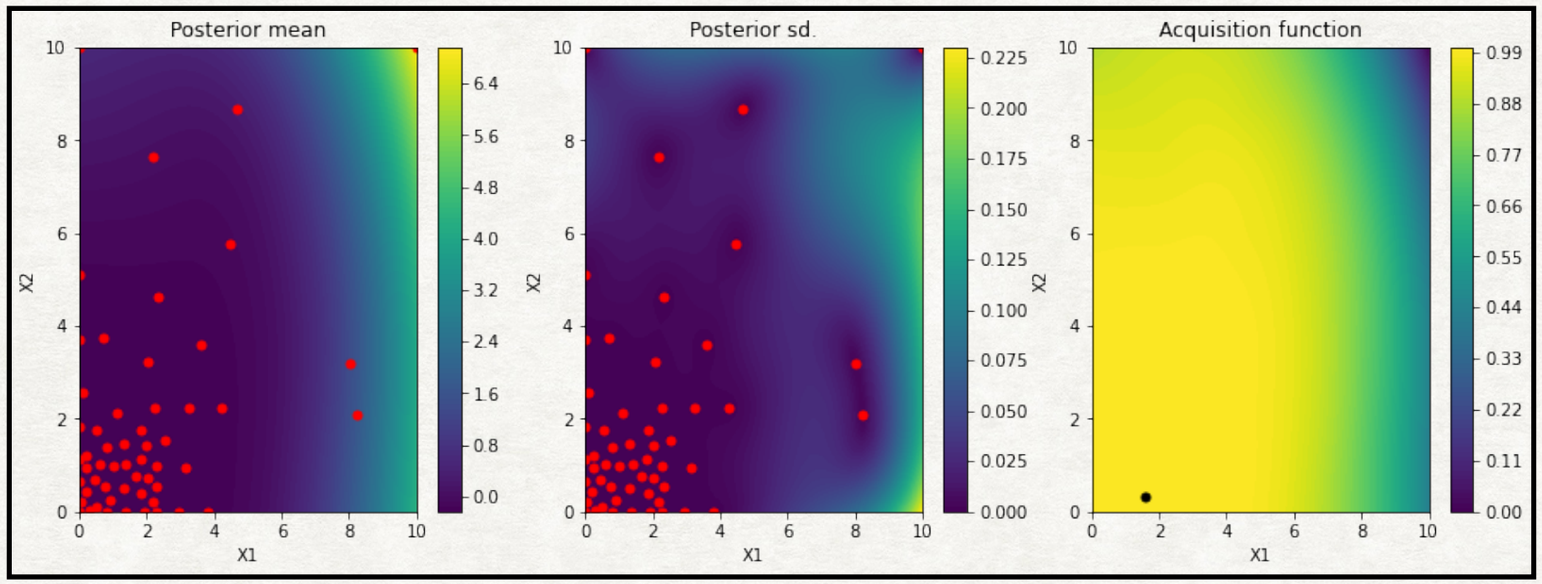


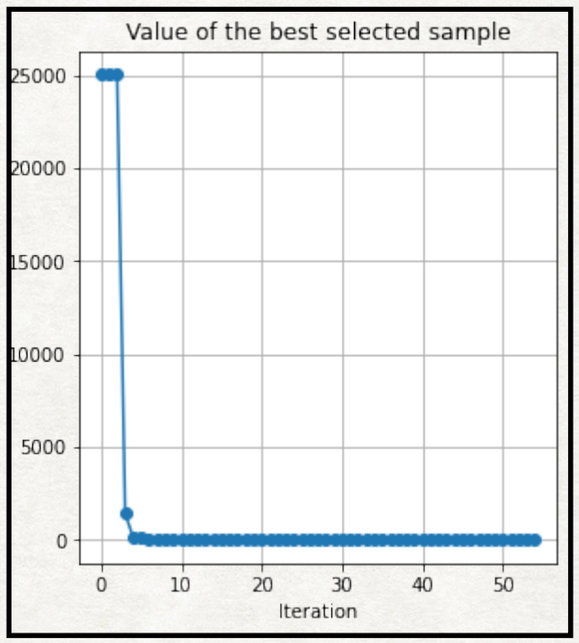


EXPERIMENTS

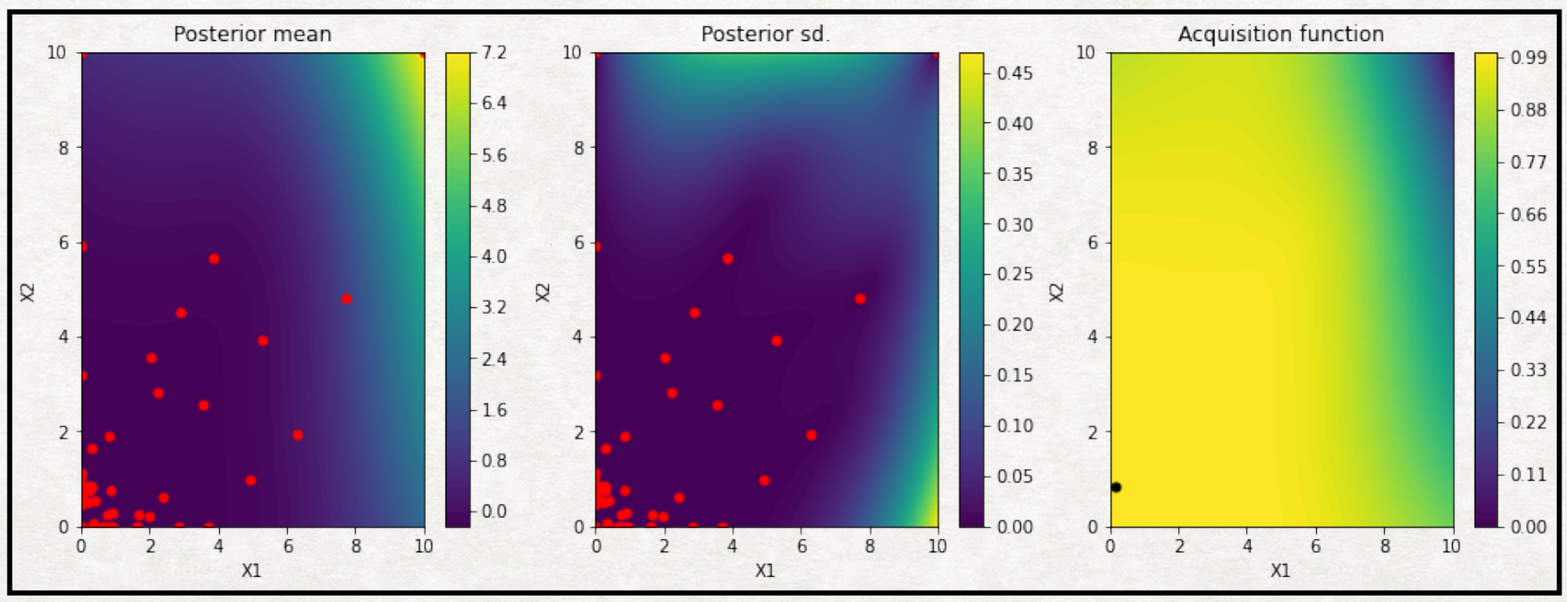
SIX-HUMP CAMEL FUNCTION-RESULTS

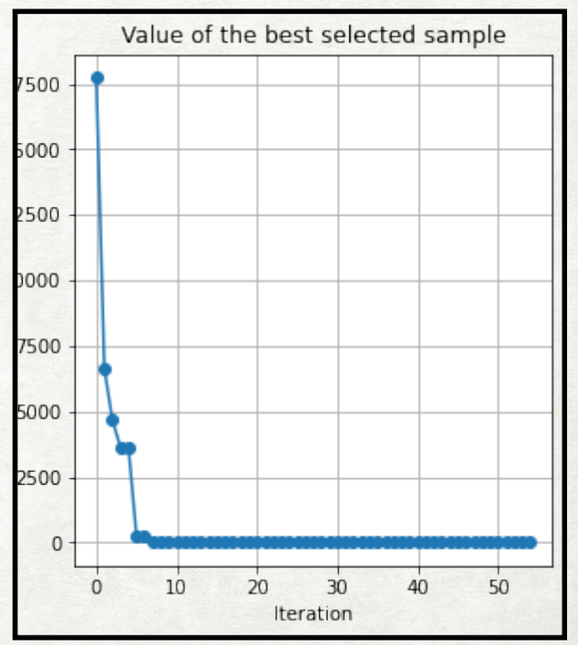
UCB





UCB_ MCMC





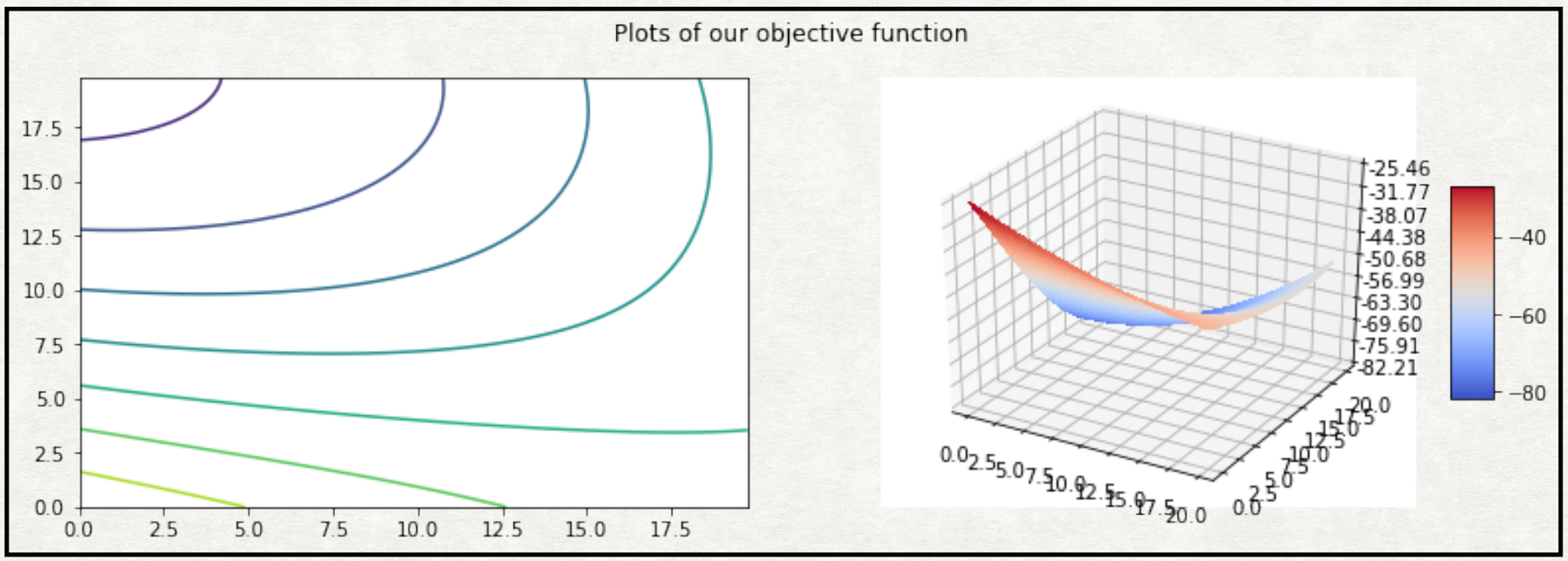
EXPERIMENTS

EGG HOLDER FUNCTION

Mathematical Representation

```
def objfunc2d(x1,x2):
    return -(x2+47) * np.sin(np.sqrt(np.abs(x2+x1/2+47)))+ -x2 * np.sin(np.sqrt(np.abs(x1-(x2+47))))
```

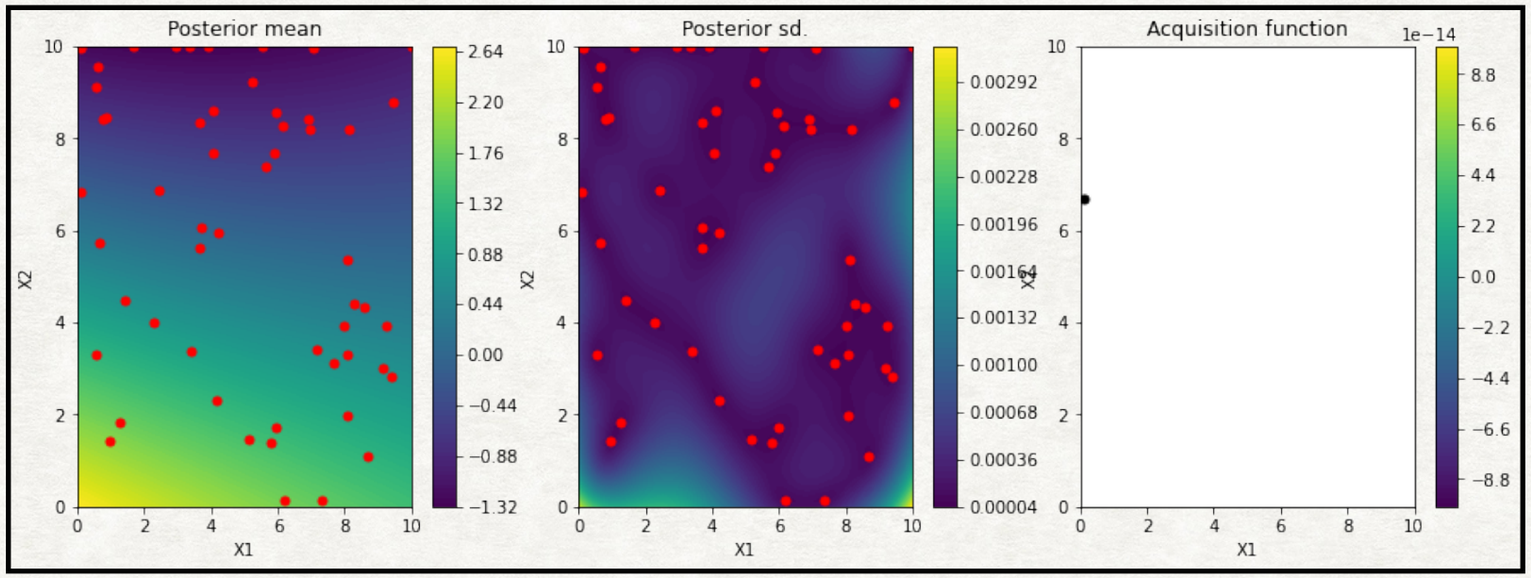
Vizualization

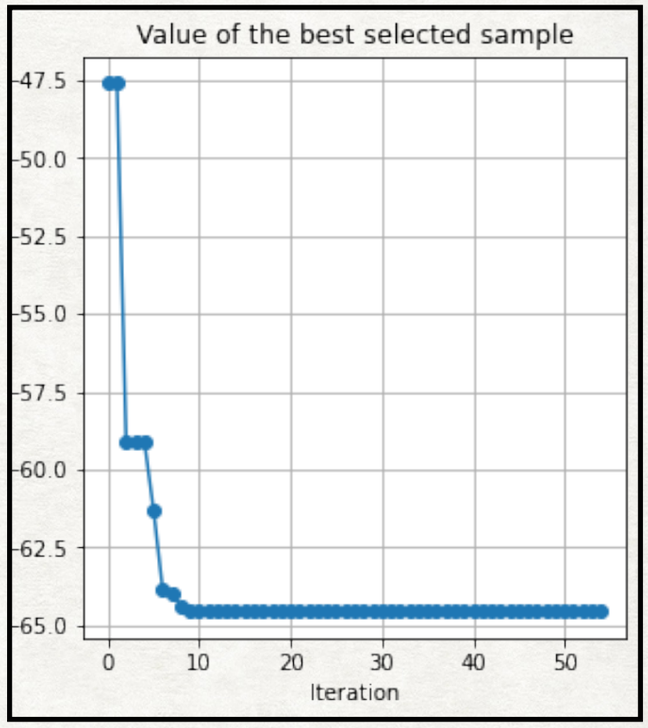


EXPERIMENTS

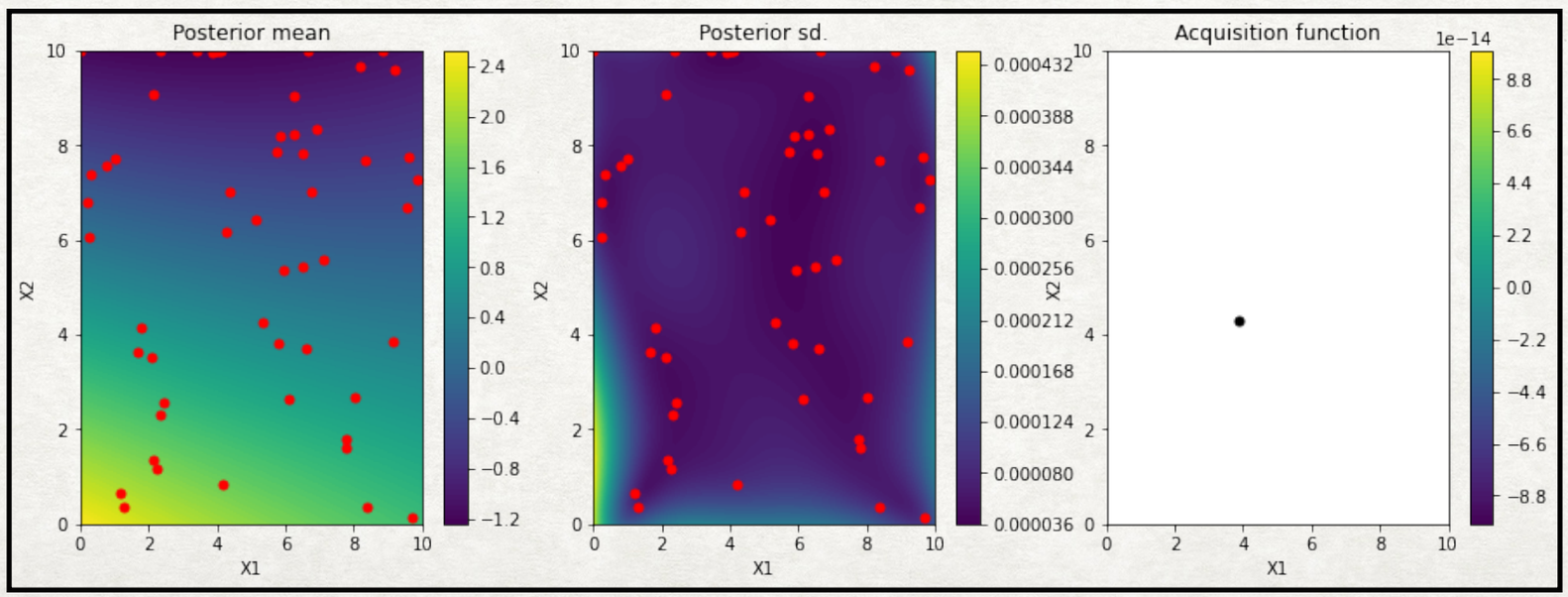
EGG HOLDER FUNCTION-RESULTS

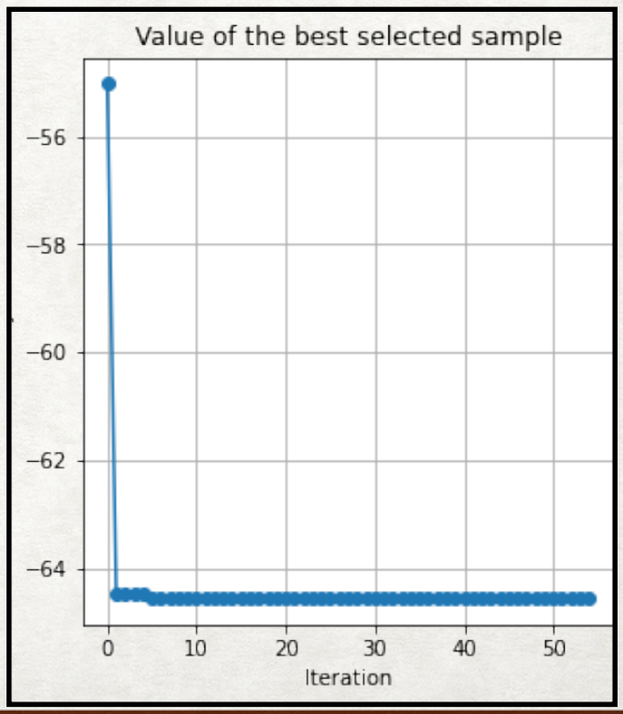
PI





PI_MCMC

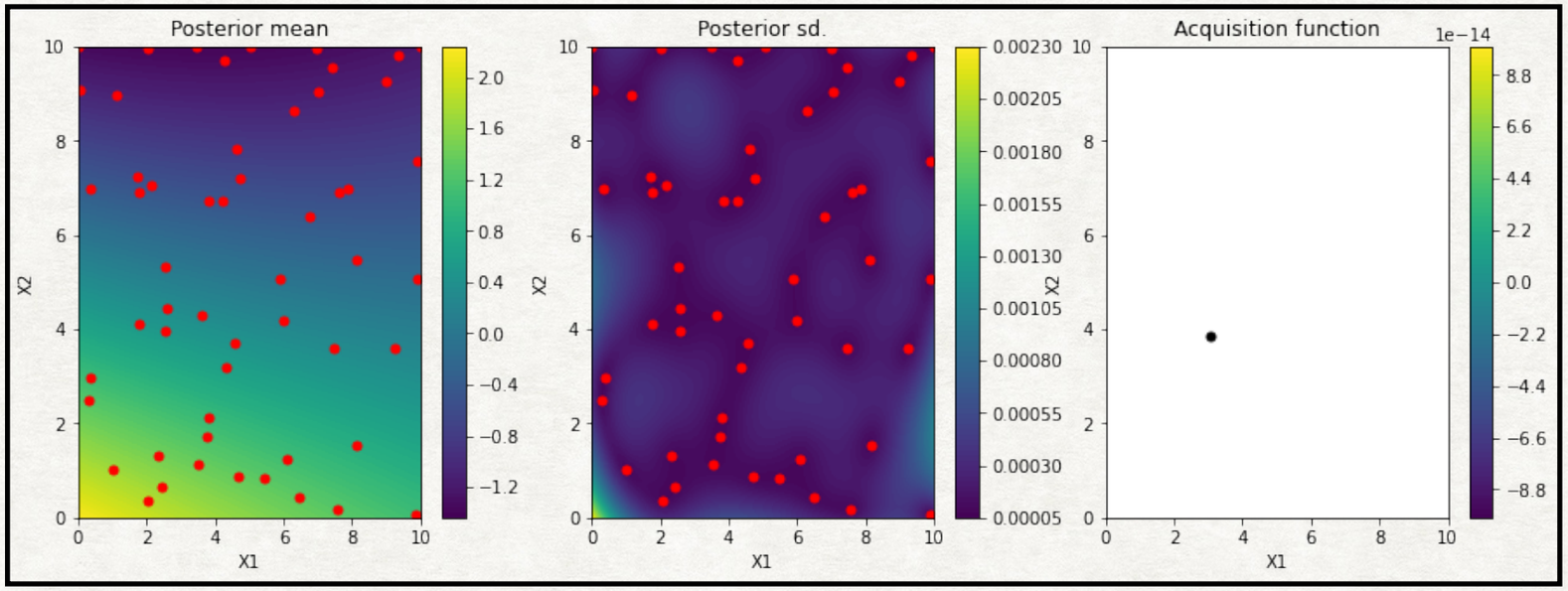


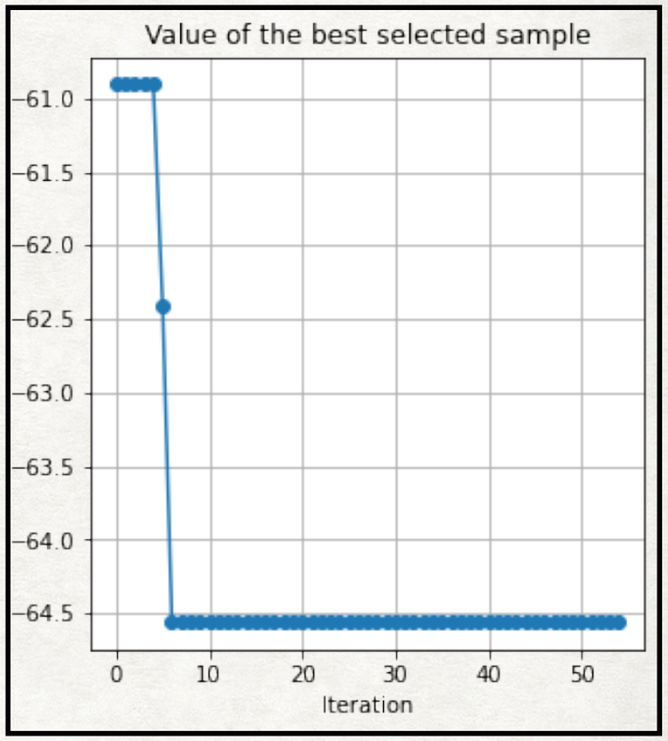


EXPERIMENTS

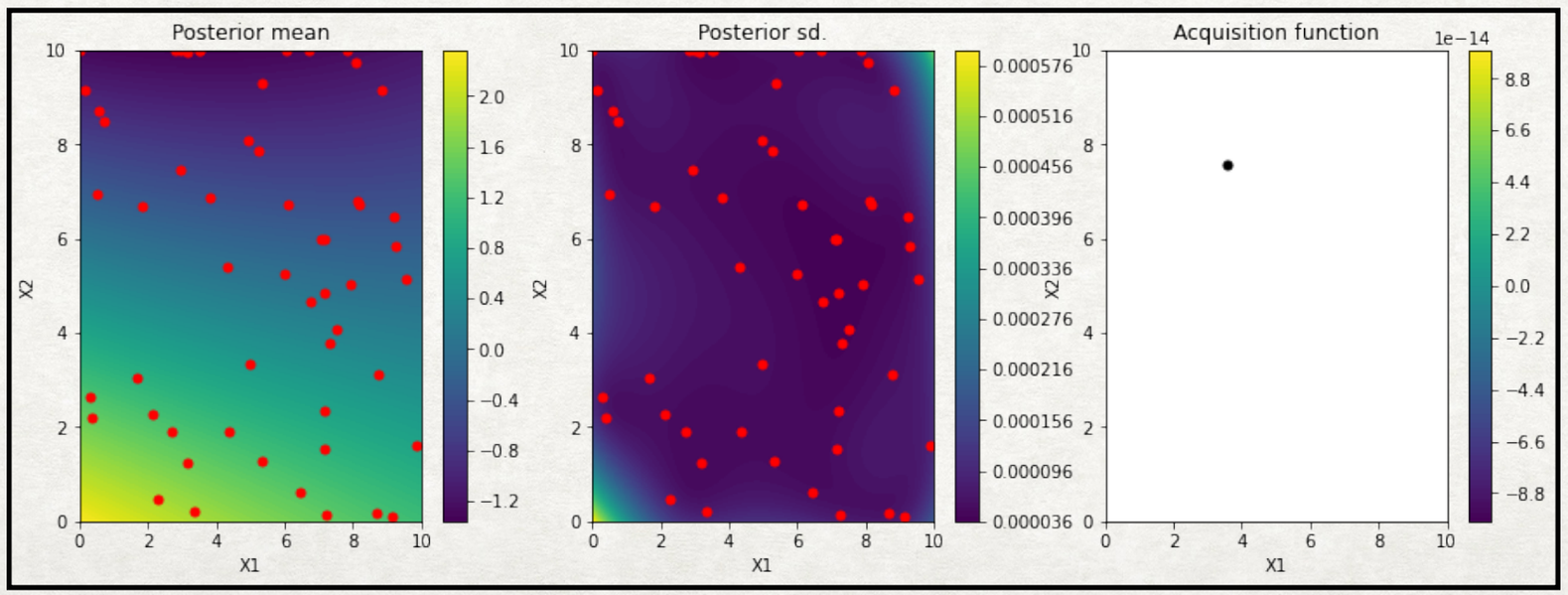
EGG HOLDER FUNCTION-RESULTS

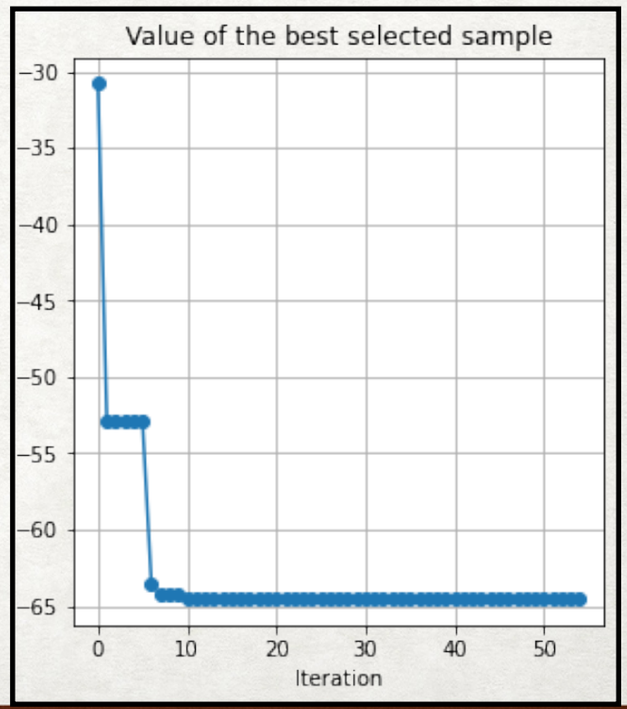
EI





EI_MCMC

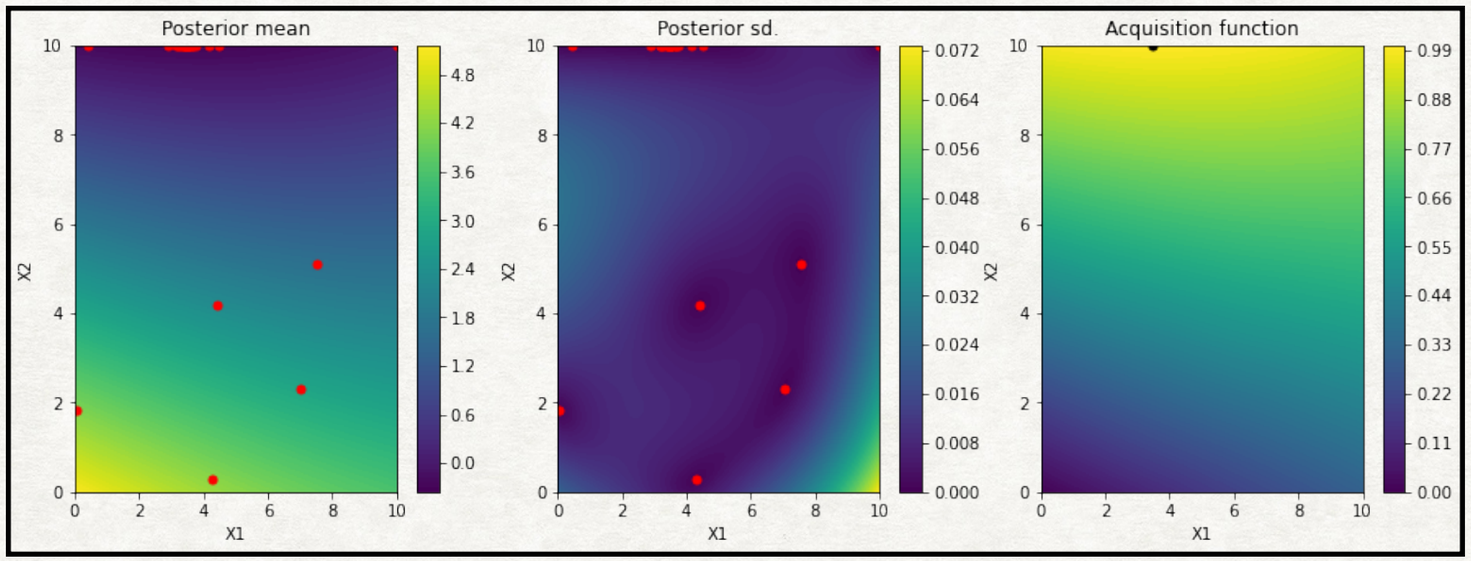


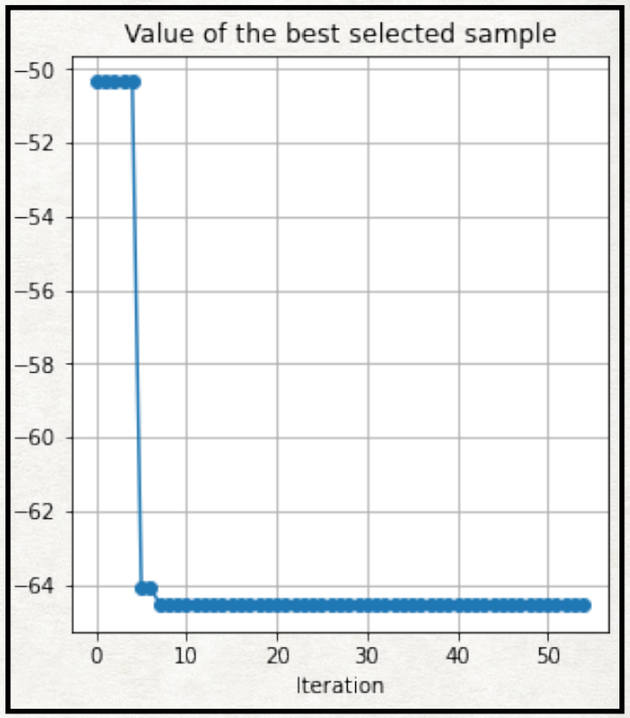


EXPERIMENTS

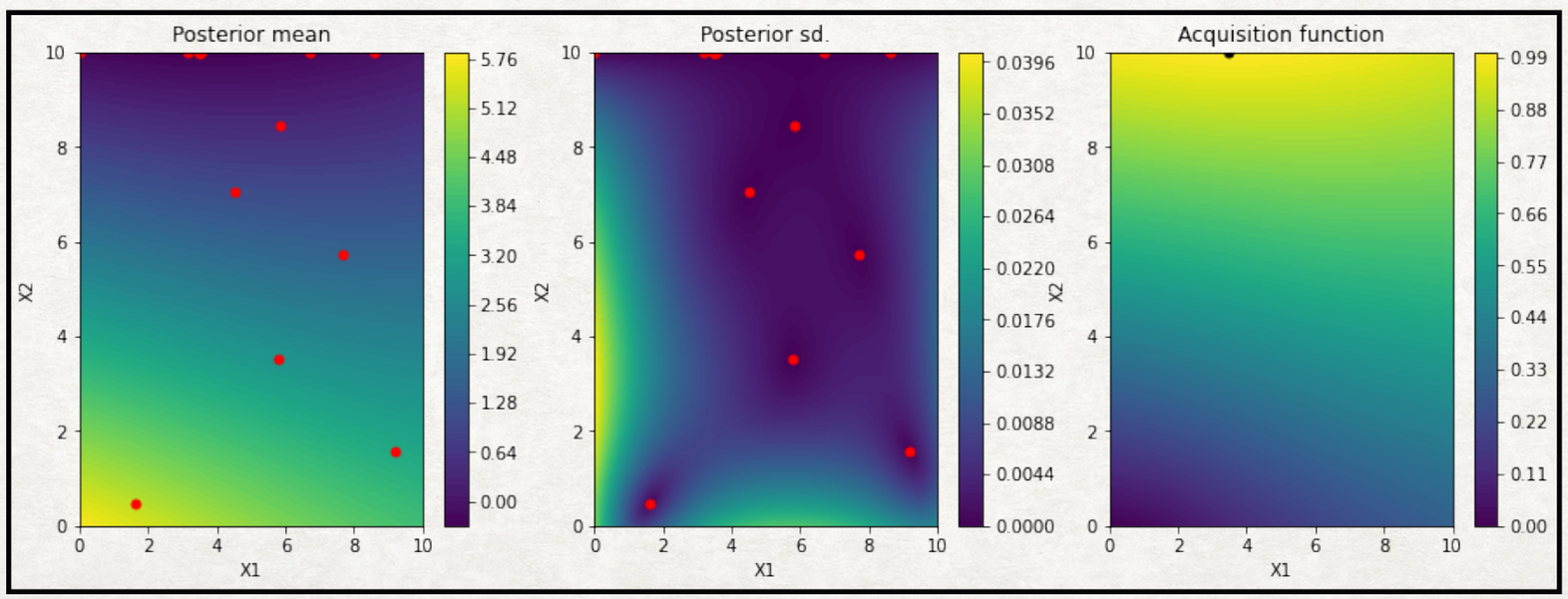
EGG HOLDER FUNCTION-RESULTS

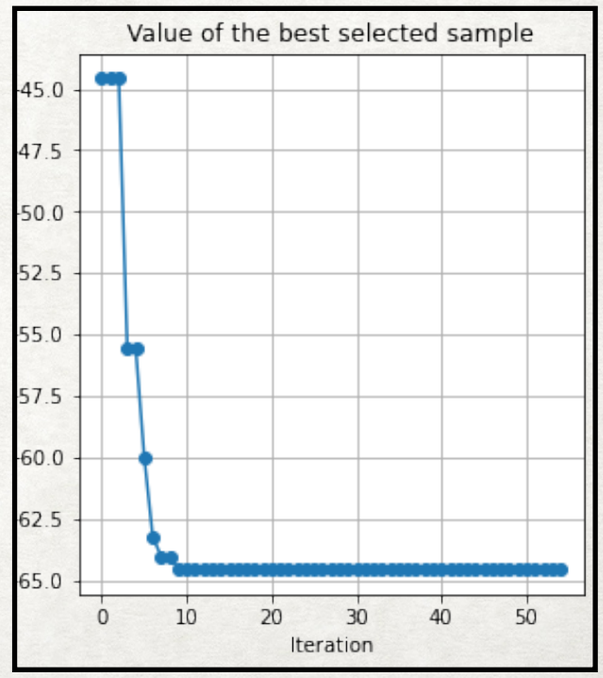
UCB





UCB_ MCMC





RESULTS

Function	**Branin-Hoo**		**Egg Holder**		**Six-Hump Camel**	
Acquisition	(x,y)	Fmin	(x,y)	Fmin	(x,y)	Fmin
GPUCB	3.14168267, 2.27518949	-9.60211254	3.71680763, 10.	-64.5	0, 0.8985604	-0.62199444
GPUCB_MCMC	3.14132133, 2.27679613	-9.60210978	3.64703385, 10.	-64.56132069	0, 0.60437904	-0.92739565
EI	3.15129929 2.32885665	-9.59788858	3.62480693, 10.	-64.56167395	0, 0.90340291	-0.60023005
EI_MCMC	9.42448631, 2.45532295	-9.60173467]	3.48714407,10.	-64.56271354	0, 0.83546759	0.84317435
PI	9.42807156, 2.39934847	-9.59590912	2.99442148, 10.	-64.55020213]	0, 0.52149062	-0.79197728
PI_MCMC	3.08856032, 2.3485409	-9.58760068	3.41143713, 10	-64.56244196	0, 0.58556976	-0.90126823
GPUCB^2	_	-8.51	_	-50.32	_	-0.6
GPUCB_NN	_	_	_	_	_	_

REFERENCE

- http://mlg.postech.ac.kr/~seungjin/publications/icassp18_KimJT.pdf

For Gaussian Process:

- https://www.cs.cmu.edu/~epxing/Class/10708-15/notes/10708_scribe_lecture21.pdf
- https://peterroelants.github.io/posts/gaussian-process-kernels/#Rational-quadratic-kernel
- https://krasserm.github.io/2018/03/19/gaussian-processes/
- https://colab.research.google.com/github/krasserm/bayesian-machine-learning/blob/dev/bayesian-optimization/bayesian_optimization.ipynb#scrollTo=53523pBsUFOL

For Bayesian Optimization:

- https://distill.pub/2020/bayesian-optimization/
- https://arxiv.org/pdf/0912.3995.pdf
- https://www.cs.ubc.ca/~nando/540-2013/lectures/l7.pdf
- https://paperswithcode.com/paper/gaussian-process-optimization-in-the-bandit

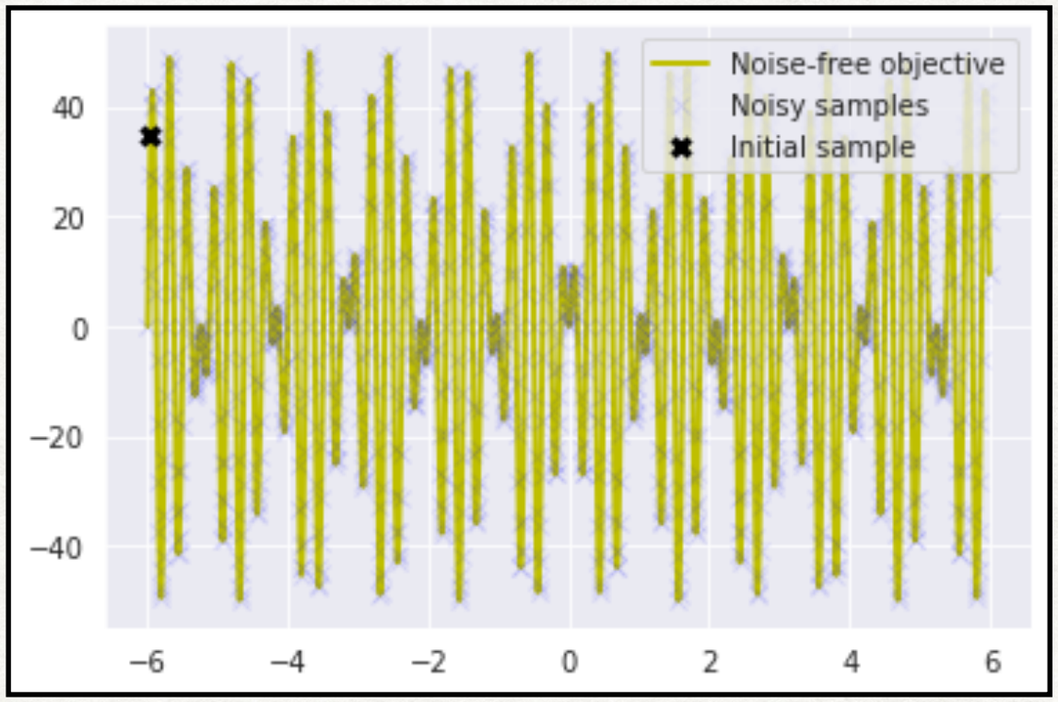
Functions Details

- https://www.sfu.ca/~ssurjano/branin.html
- https://www.sfu.ca/~ssurjano/egg.html
- https://www.sfu.ca/~ssurjano/camel6.html

THANK YOU !

EXPERIMENTS

We experimented on another synthetic function: (50*(np.sin(8*np.pi*X))*(np.sin(3*X)))+noise* np.random.randn(*X.shape)



All acqsn funs performed poorly on this one

Add -log to this and tryyy