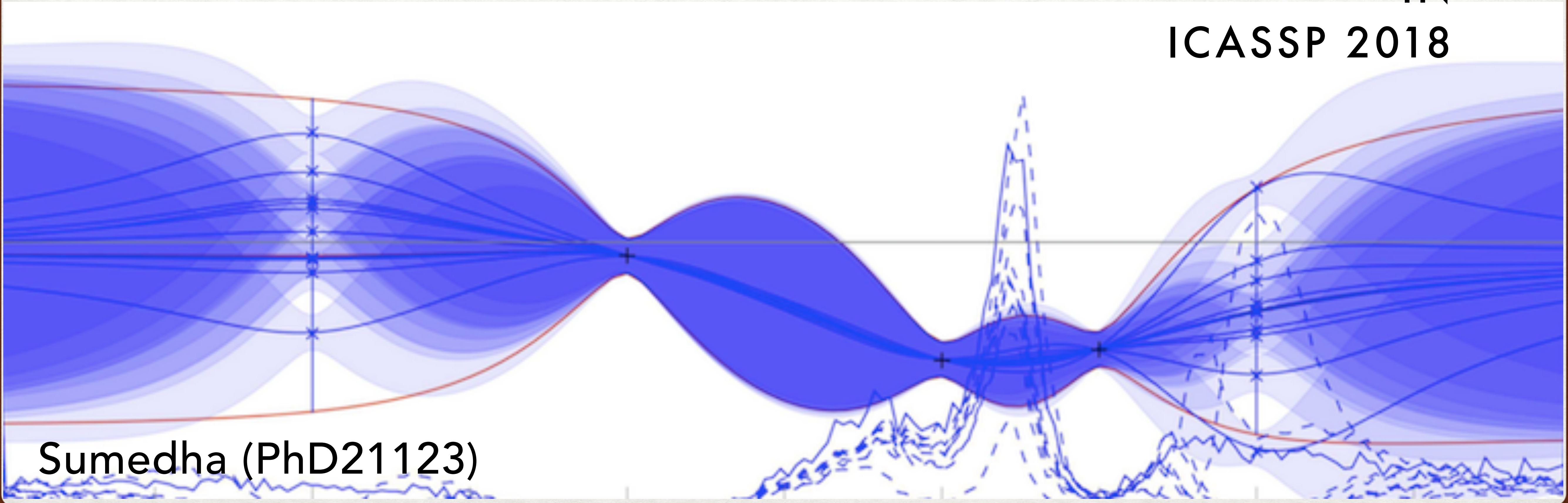
CLUSTERING-GUIDED GP-UCB FOR BAYESIAN OPTIMIZATION

BML Mid-Term Paper Presentation

JUNGTAEK KIM AND SEUNGJIN CHOI
IN
ICASSP 2018

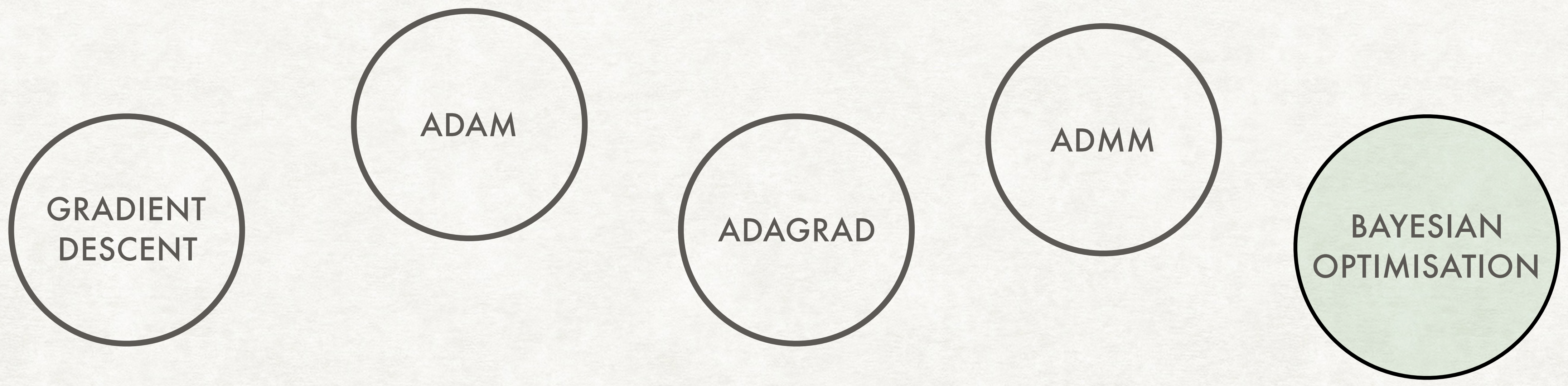


Sumedha (PhD21123)

CONTENTS

TO BAYE OR NOT TO BAYE

WHY BAYESIAN OPTIMIZATION



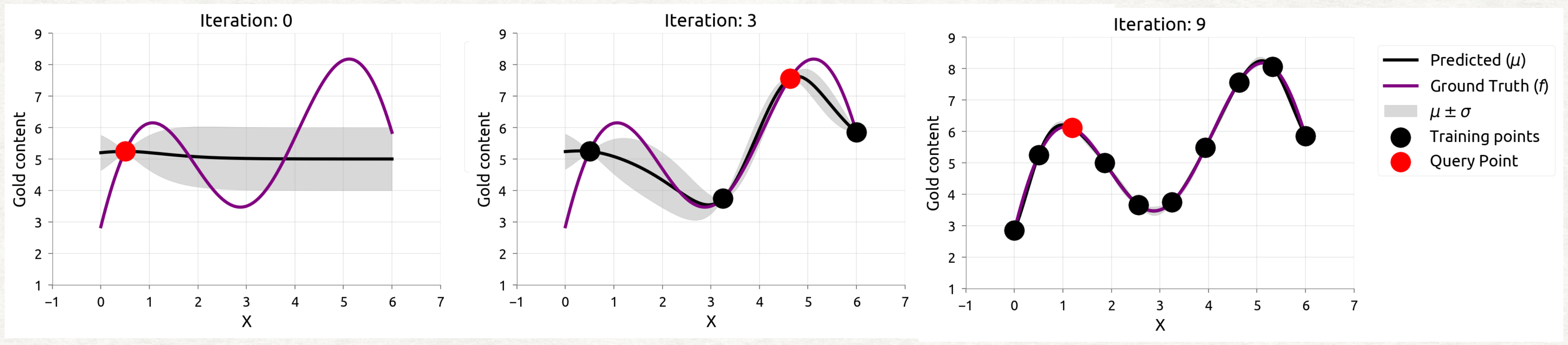
- At the times, when a large amount data is available, an approximation for original function y = f(x) can be made and hence it’s maxima or minima can be found.

- 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)

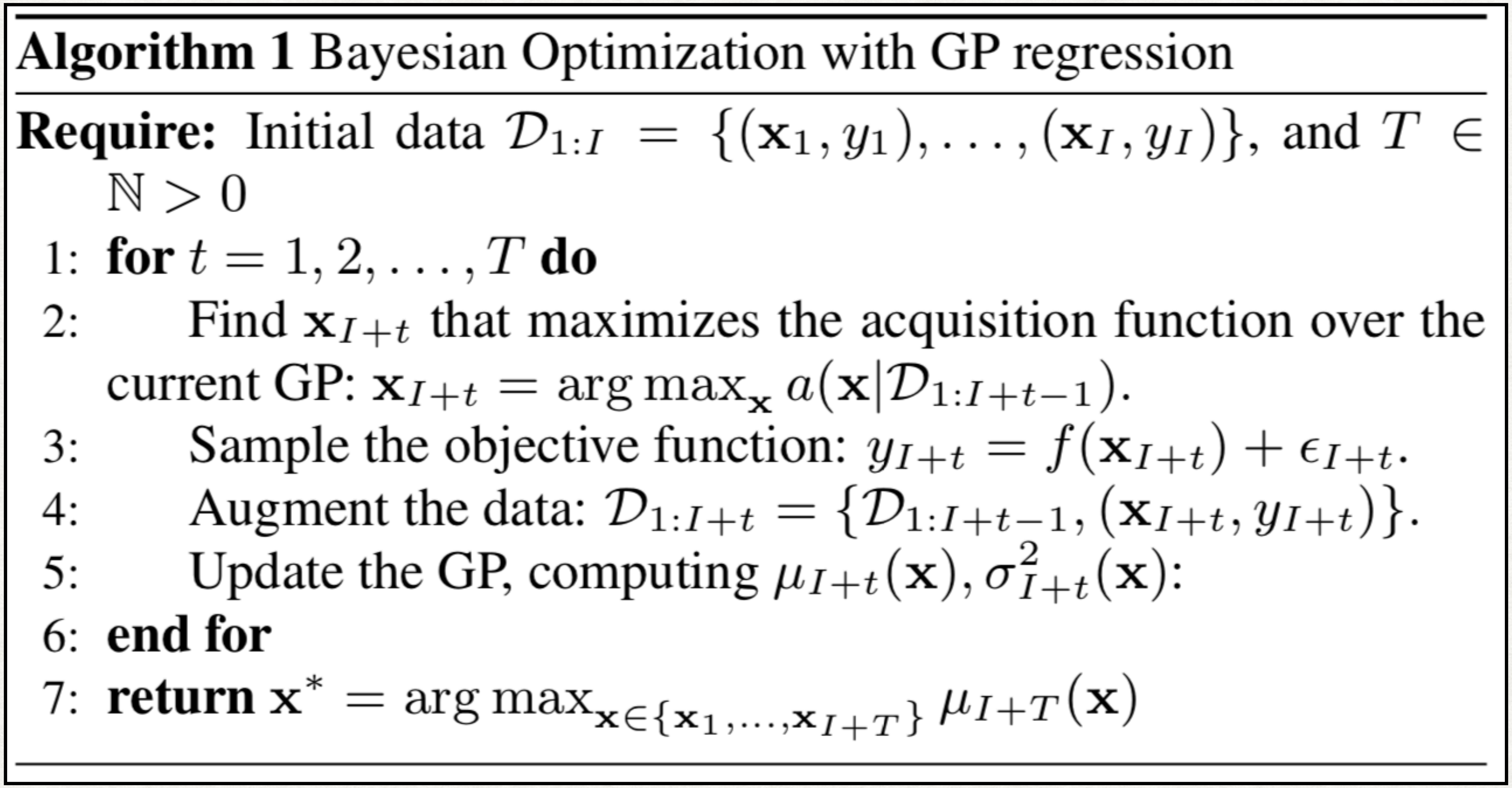
OBTAIN POSTERIOR MEAN AND POSTERIOR VARIANCE FOR GIVEN SET OF OBESERVATIONS

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

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



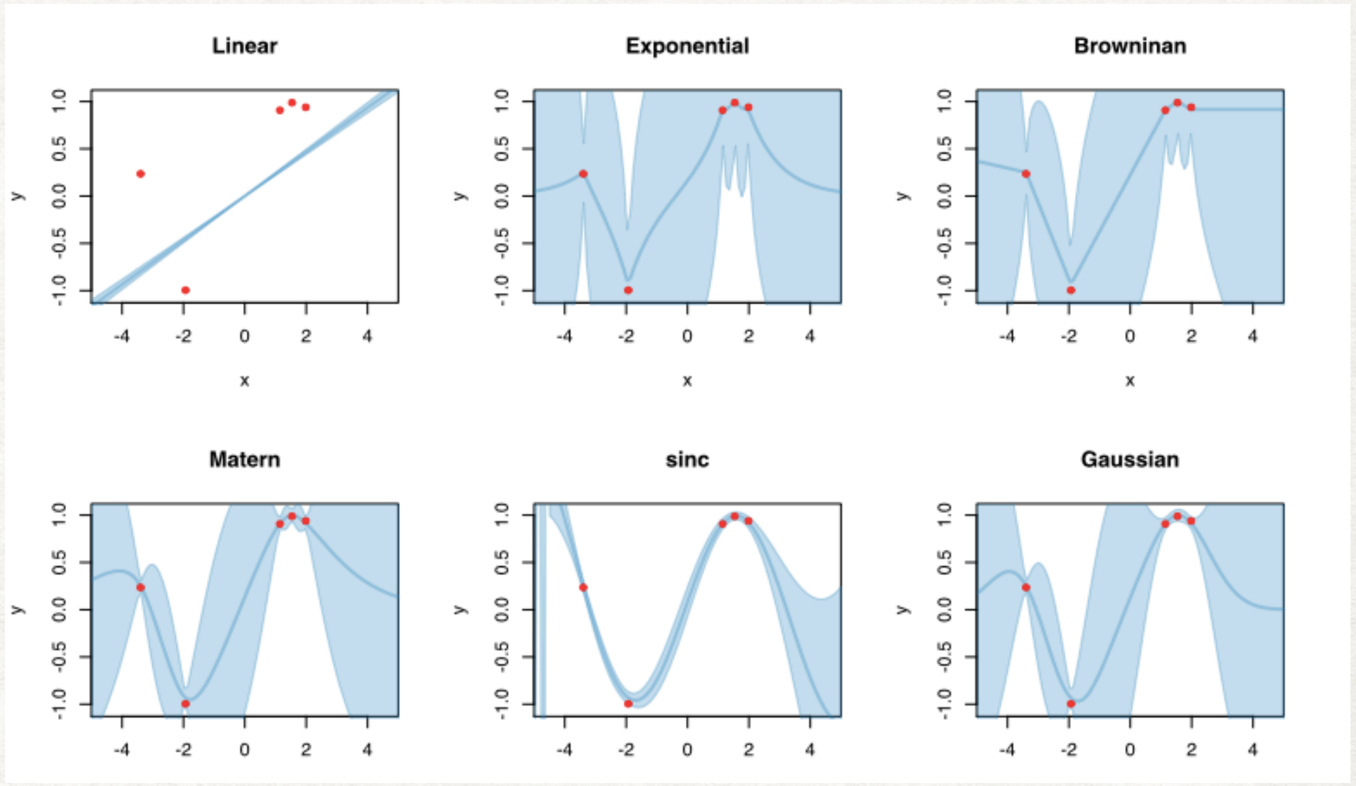
Algorithm 1 Bayesian Optimization with GP regression

```
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.
- $p(f|X)= \quad N(f|\mu,K)$
- $f=(f(x_1),\ldots,f(x_N))$, $\mu=(m(x_1),\ldots,m(x_N))$ and $K_{ij}=\kappa(x_i,x_j)$.
- 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:

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

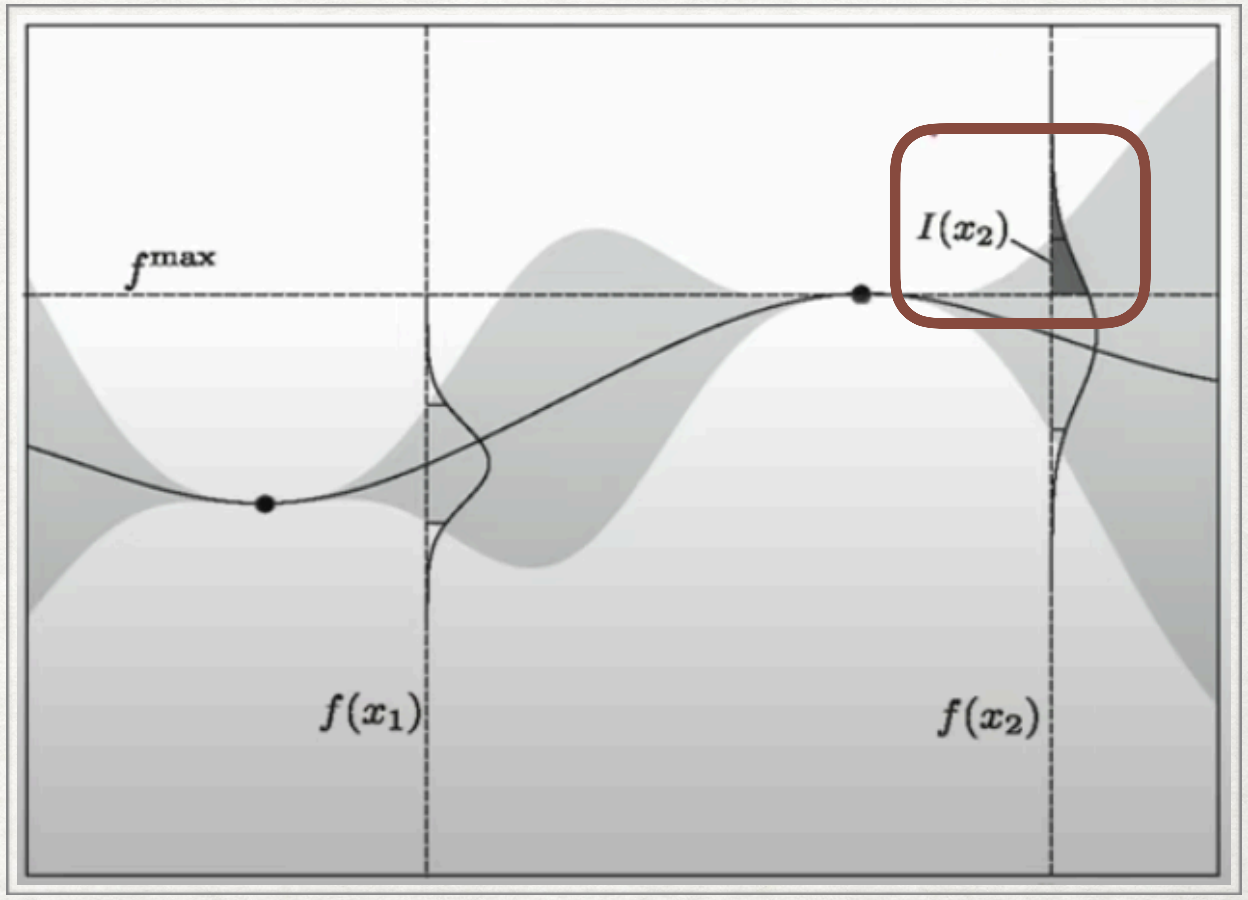
  where, ϵ is a small positive number and $f(x^+)$ is current max

  - 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:

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

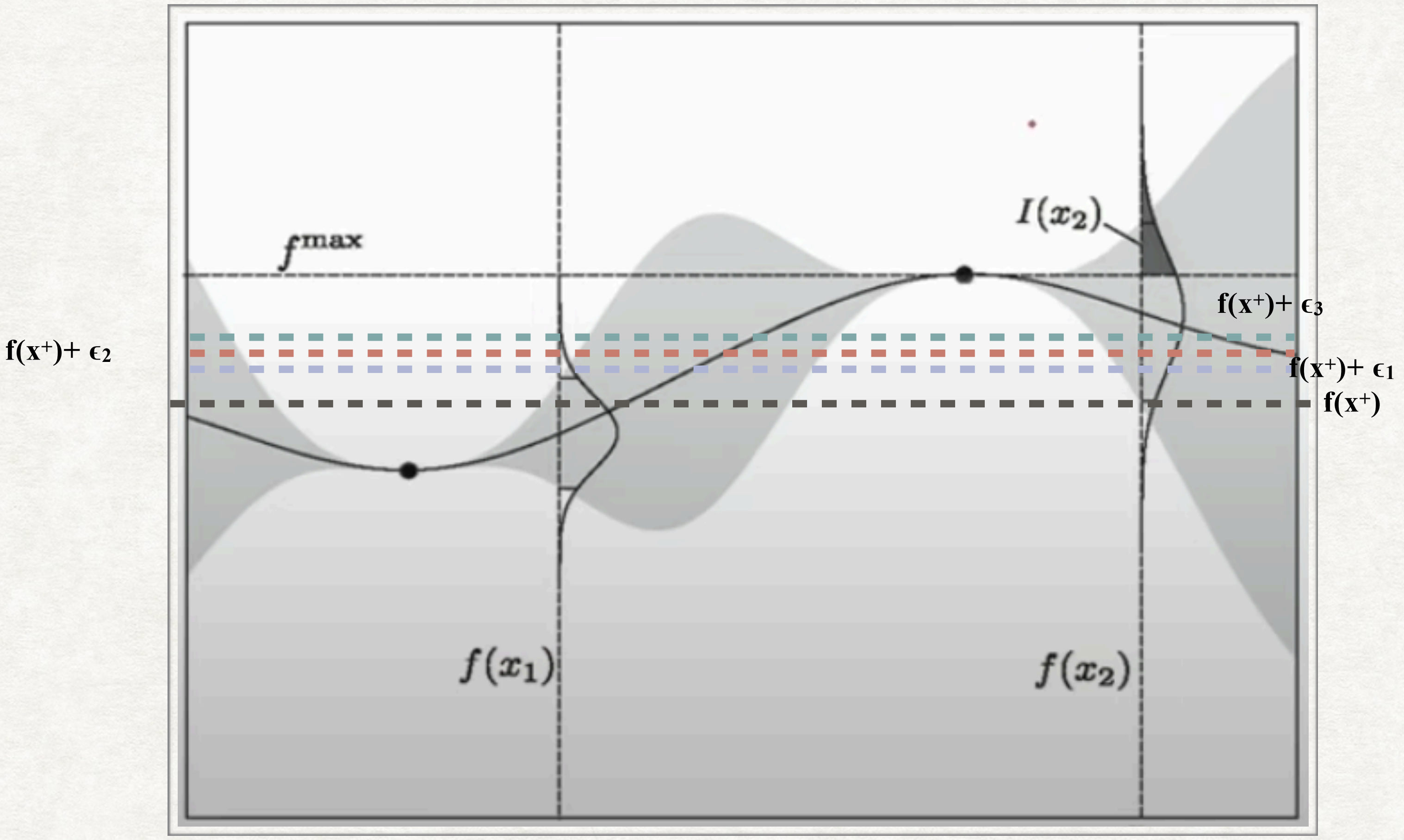
$\Phi(\cdot)$ indicates the CDF



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:

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

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.

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

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

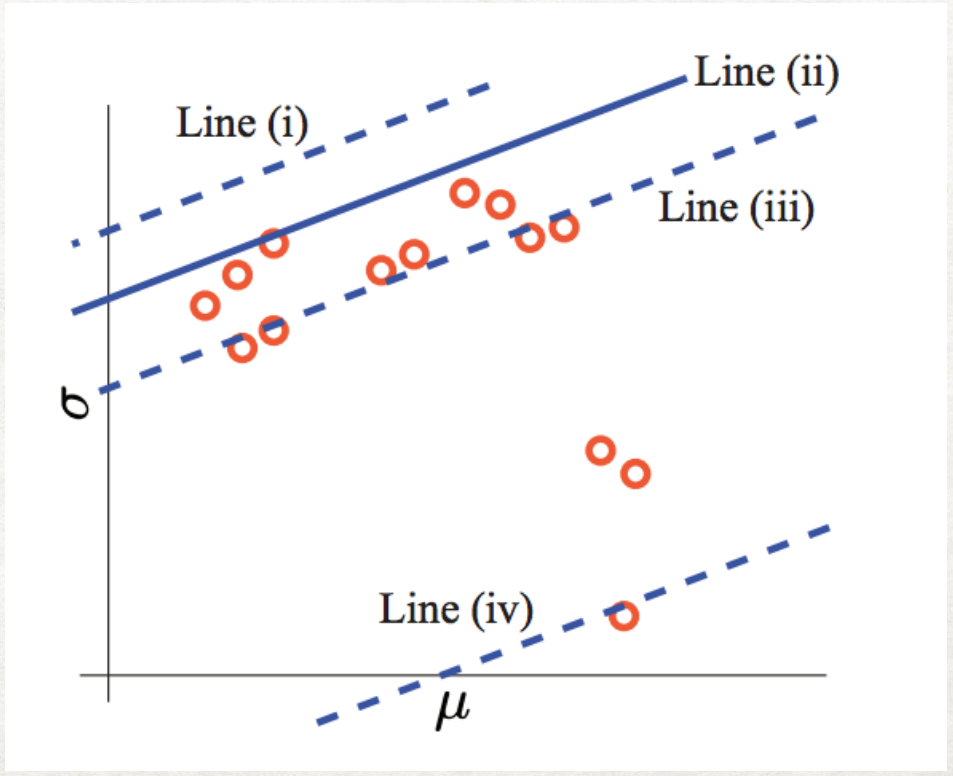
- Presents A novel geometric view of GP-UCB based on posterior mean and variance has been presented in the paper
- Reduces the search space of GP-UCB acquisition function to single cluster chosen by method proposed

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

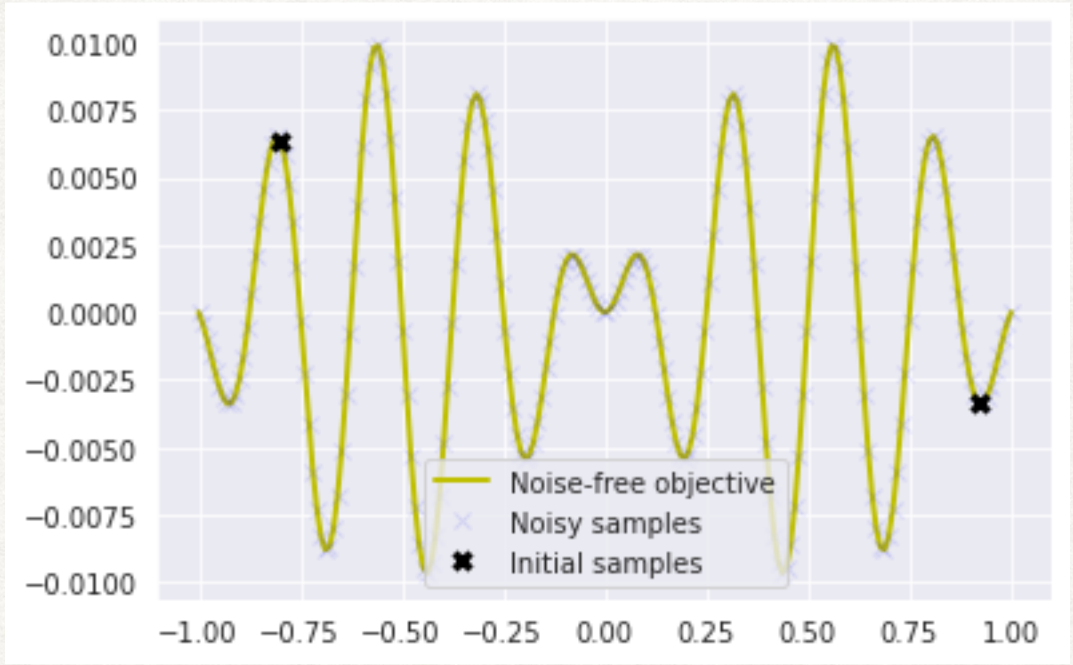
↓

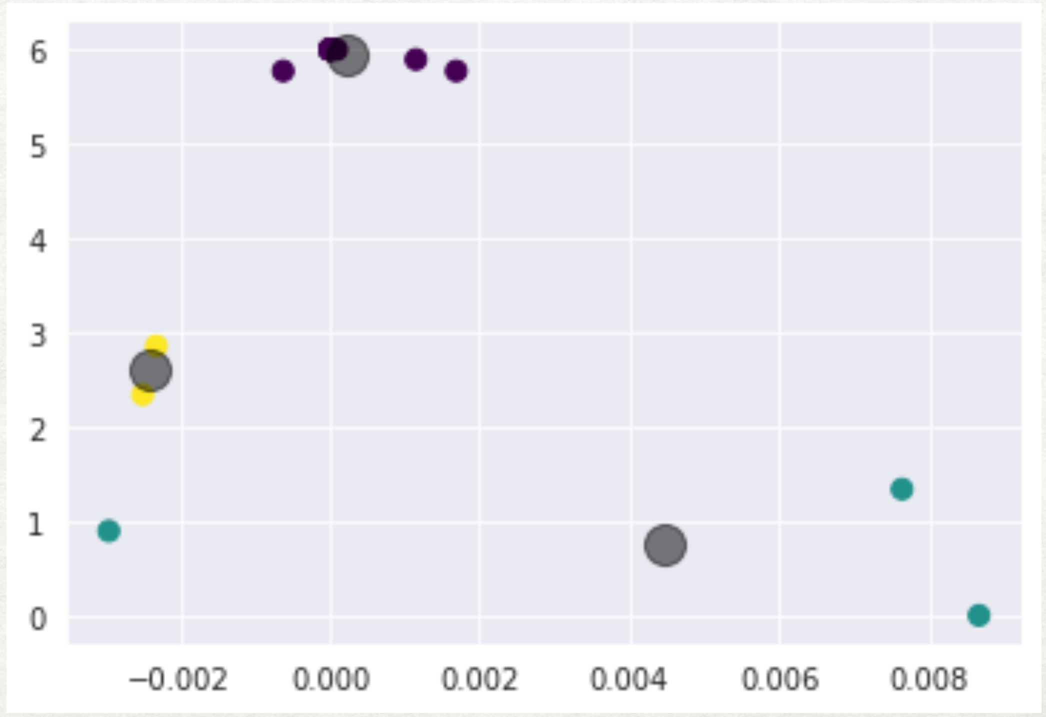
Randomly sample a few points from posterior mean and variance and plot them in these in μ-σ plane

↓

Cluster these points and calculate their centres

↓





CLUSTERING GUIDED GP-UCB

WORK IN DETAIL

Find the best cluster C_{i*}

$$i^* = \arg\max_{i=1,2,\ldots,k} [-c_{i,1} + \kappa c_{i,2}]$$

k being the number of clusters

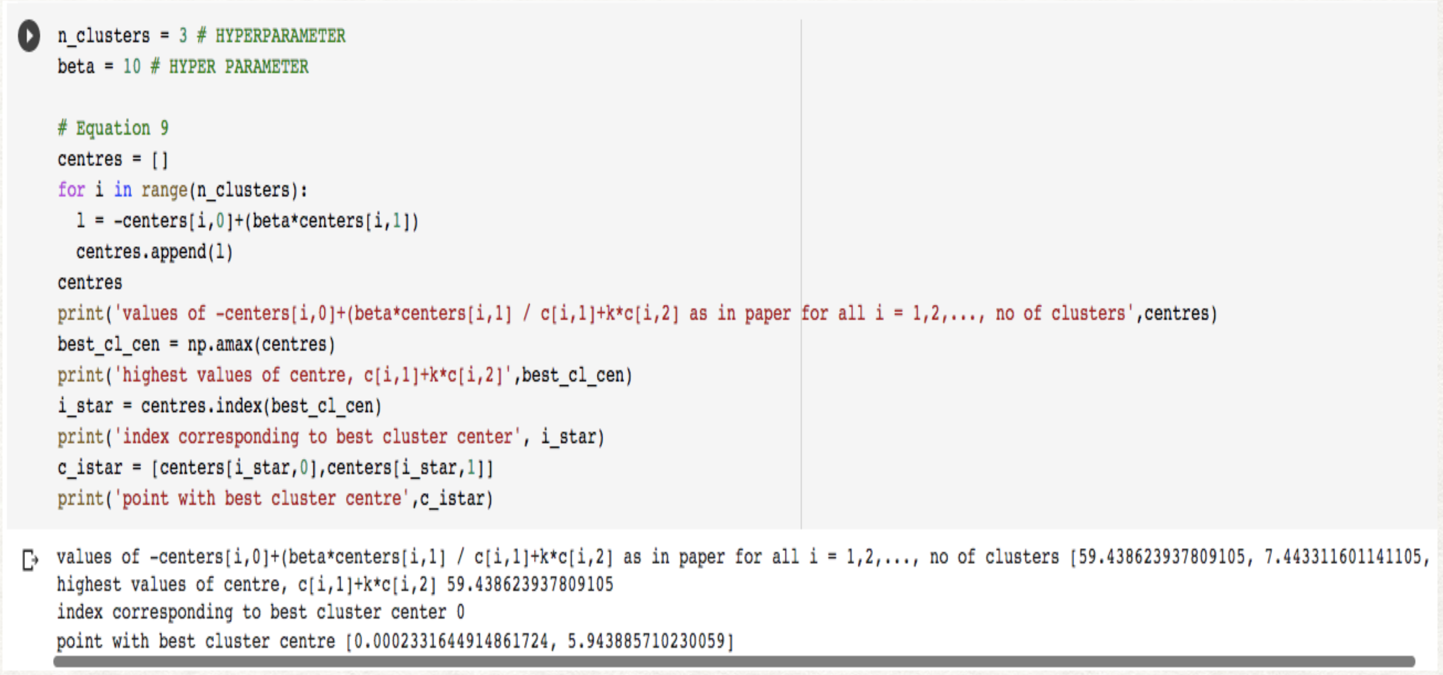
Calculate x_{1+t}

GPUCBNN:

$$x_{t+1} = \arg\min_{x \in C_{i_*}} \| [\mu(x), \sigma(x)]^T - c_{i_*} \|_2^{\,2} \,,$$

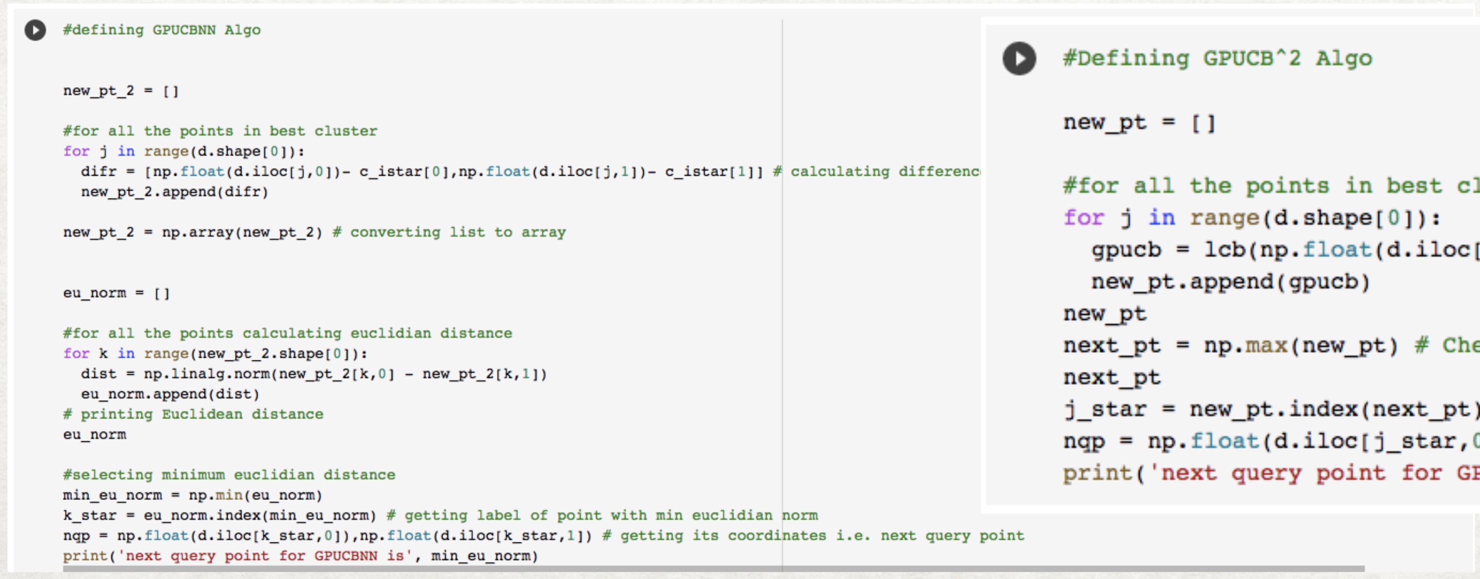
GPUCB²:

$$x_{t+1} = \arg\max_{x \in C_{i_*}} a(x|D_{1:t})$$

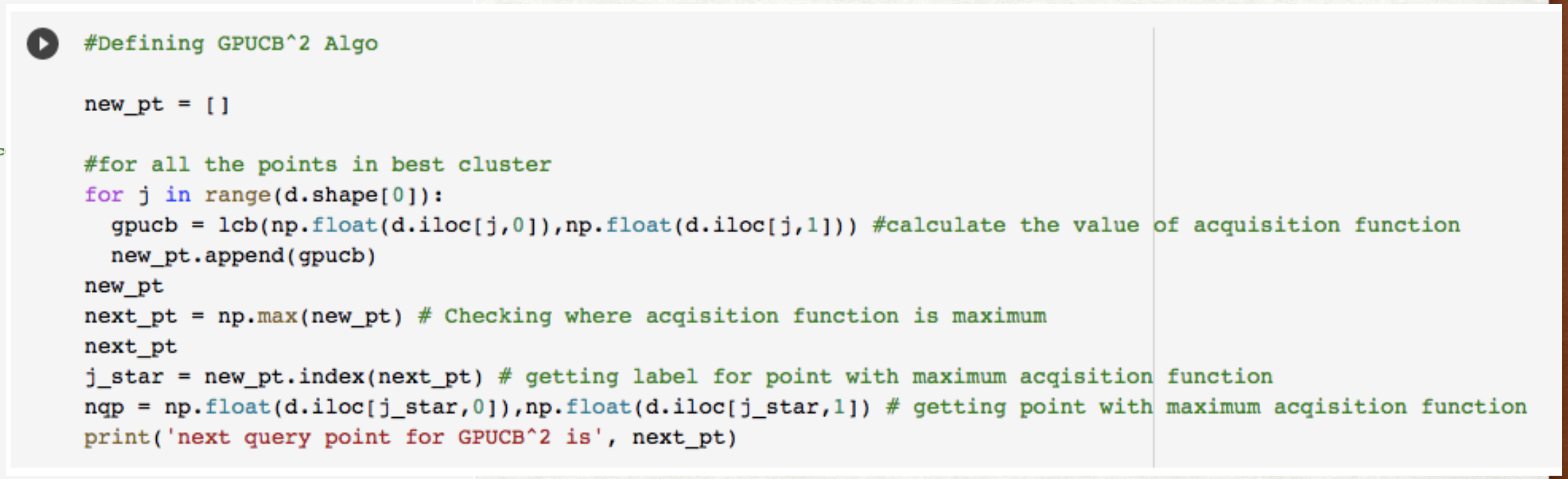


```
n_clusters = 3 # HYPERPARAMETER
beta = 10 # HYPER PARAMETER

# Equation 9
centres = []
for i in range(n_clusters):
  l = -centers[i,0]+(beta*centers[i,1])
  centres.append(l)
centres
print('values of -centers[i,0]+(beta*centers[i,1] / c[i,1]+k*c[i,2] as in paper for all i = 1,2,..., no of clusters',centres)
best_cl_cen = np.amax(centres)
print('highest values of centre, c[i,1]+k*c[i,2]',best_cl_cen)
i_star = centres.index(best_cl_cen)
print('index corresponding to best cluster center', i_star)
c_istar = [centers[i_star,0],centers[i_star,1]]
print('point with best cluster centre',c_istar)
```

```
values of -centers[i,0]+(beta*centers[i,1] / c[i,1]+k*c[i,2] as in paper for all i = 1,2,..., no of clusters [59.438623937809105, 7.443311601141105,
highest values of centre, c[i,1]+k*c[i,2] 59.438623937809105
index corresponding to best cluster center 0
point with best cluster centre [0.0002331644914861724, 5.943885710230059]
```



```
#defining GPUCBNN Algo


new_pt_2 = []

#for all the points in best cluster
for j in range(d.shape[0]):
  difr = [np.float(d.iloc[j,0])- c_istar[0],np.float(d.iloc[j,1])- c_istar[1]] # calculating differenc
  new_pt_2.append(difr)

new_pt_2 = np.array(new_pt_2) # converting list to array


eu_norm = []

#for all the points calculating euclidian distance
for k in range(new_pt_2.shape[0]):
  dist = np.linalg.norm(new_pt_2[k,0] - new_pt_2[k,1])
  eu_norm.append(dist)
# printing Euclidean distance
eu_norm

#selecting minimum euclidian distance
min_eu_norm = np.min(eu_norm)
k_star = eu_norm.index(min_eu_norm) # getting label of point with min euclidian norm
nqp = np.float(d.iloc[k_star,0]),np.float(d.iloc[k_star,1]) # getting its coordinates i.e. next query point
print('next query point for GPUCBNN is', min_eu_norm)
```



```
#Defining GPUCB^2 Algo

new_pt = []

#for all the points in best cluster
for j in range(d.shape[0]):
  gpucb = lcb(np.float(d.iloc[j,0]),np.float(d.iloc[j,1])) #calculate the value of acquisition function
  new_pt.append(gpucb)
new_pt
next_pt = np.max(new_pt) # Checking where acqisition function is maximum
next_pt
j_star = new_pt.index(next_pt) # getting label for point with maximum acqisition function
nqp = np.float(d.iloc[j_star,0]),np.float(d.iloc[j_star,1]) # getting point with maximum acqisition function
print('next query point for GPUCB^2 is', next_pt)
```

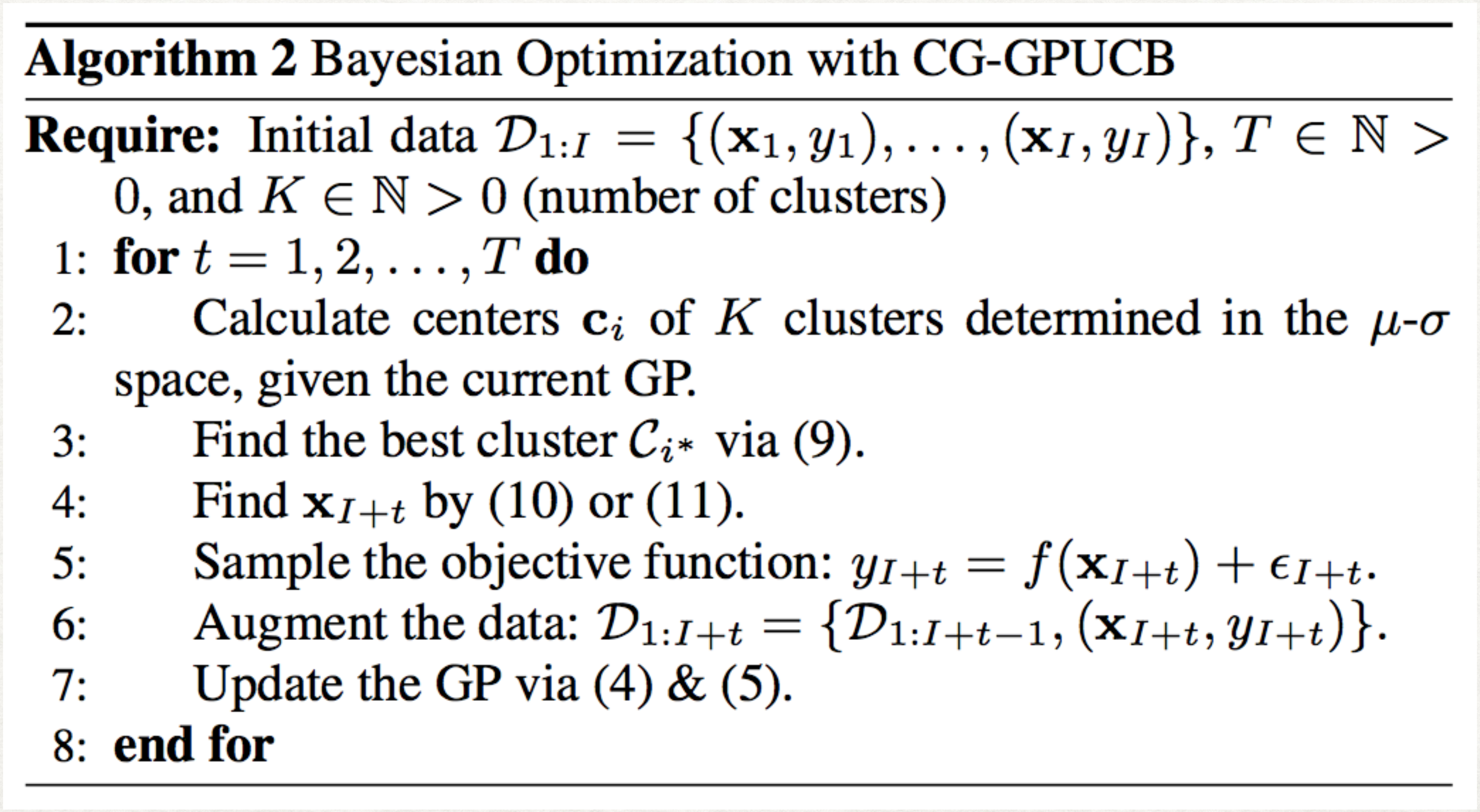
CLUSTERING GUIDED GP-UCB

WORK IN DETAIL

$$\begin{tikzcd}
\text{Find the best cluster } C_{i*} \arrow[d] \\
\text{Calculate } x_{1+t} \arrow[d] \\
\text{Sample objective function } y_{1+t} = f_{1+t} + \epsilon_{1+t} \arrow[d] \\
\text{Update GP with new dataset containing } \{D_{1:t}, (x_{1+t}, y_{1+t})\} \arrow[d] \\
\text{Repeat steps 4-9 for T iterations}
\end{tikzcd}$$

CLUSTERING GUIDED GP-UCB

WORK IN DETAIL

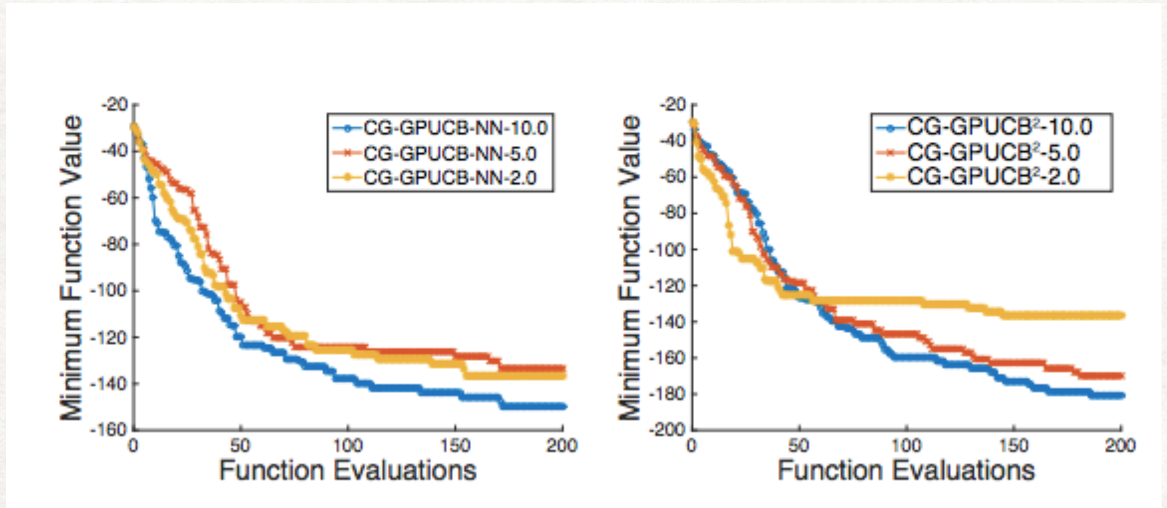


Algorithm 2 Bayesian Optimization with CG-GPUCB

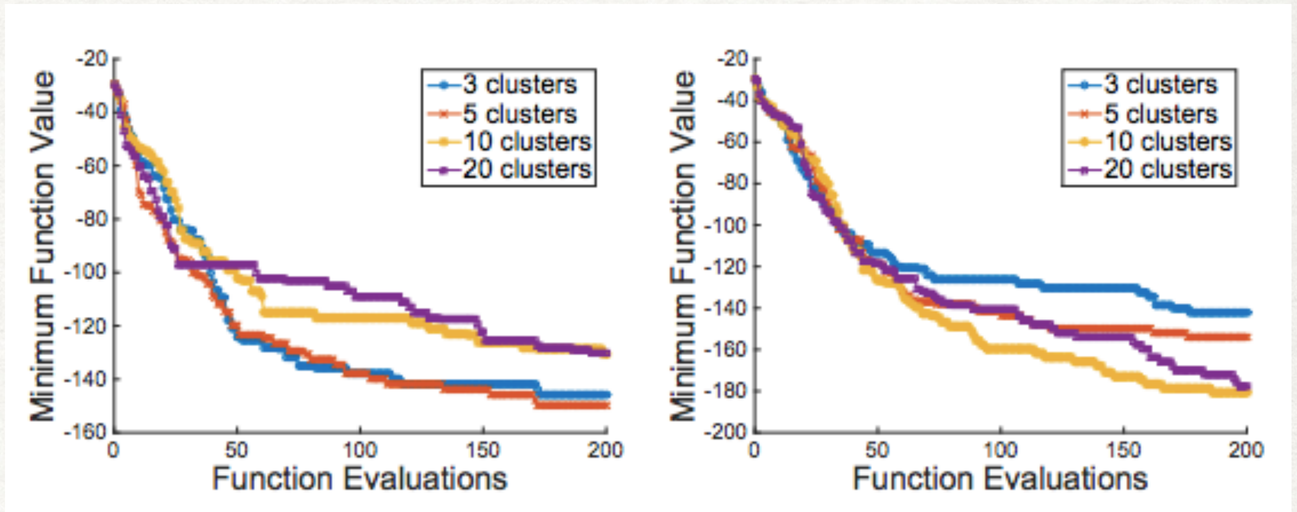
```
Require: Initial data $\mathcal{D}_{1:I} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_I, y_I)\}$, $T \in \mathbb{N} > 0$, and $K \in \mathbb{N} > 0$ (number of clusters)
1: for $t = 1, 2, \ldots, T$ do
2:     Calculate centers $\mathbf{c}_i$ of $K$ clusters determined in the $\mu$-$\sigma$ space, 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
```

PAPER IN DETAIL

EXPERIMENTS & RESULTS



(a) Effect of scaling-down hyp. for CG-GPUCB-NN (b) Effect of scaling-down hyp. for CG-GPUCB^2



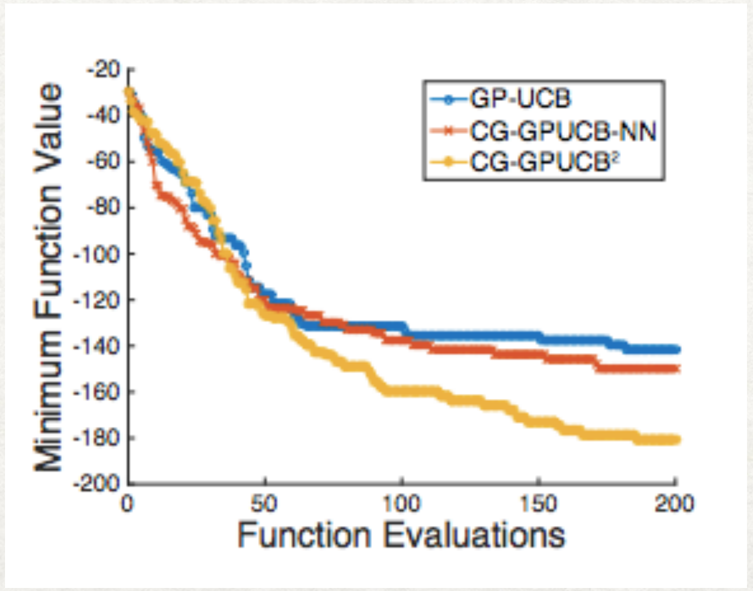
(c) Effect of the number of clusters for CG-GPUCB-NN (d) Effect of the number of clusters for CG-GPUCB^2

For synthetic function κ was set to 2.0, 5.0, and 10.0 where the number of clusters was fixed to 5 for CGGPUCB-NN and 10 for CG-GPUCB^2.

Shows best results for κ = 10

For synthetic function the number of clusters was also set to 3, 5, 10, and 20 where the scaling-down hyperparameter was fixed to 10.0.

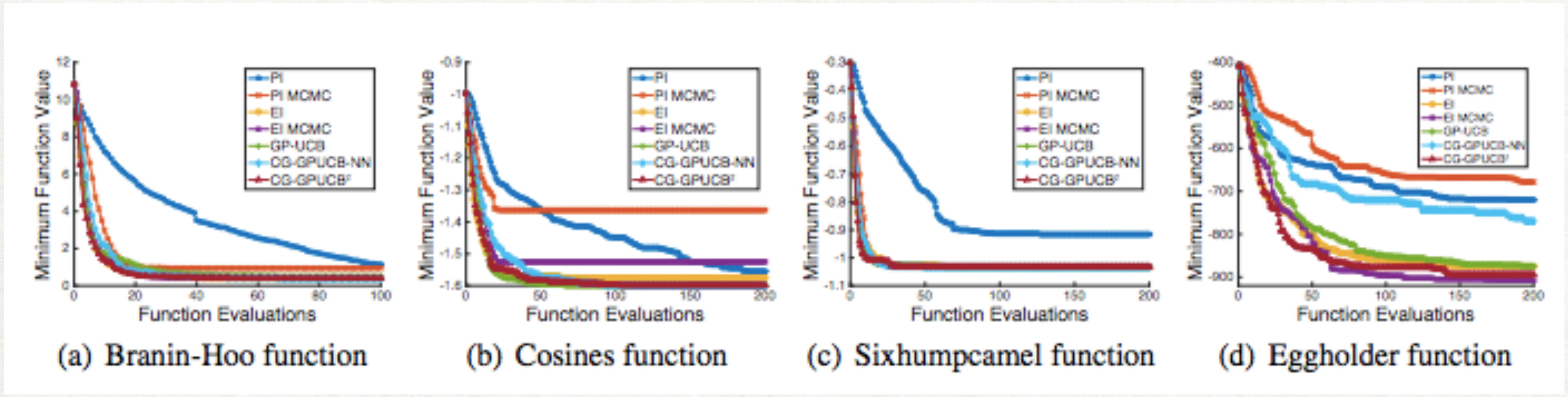
Shows best results for number of clusters = 5 for CGGPUCB-NN and 10 for CG-GPUCB^2



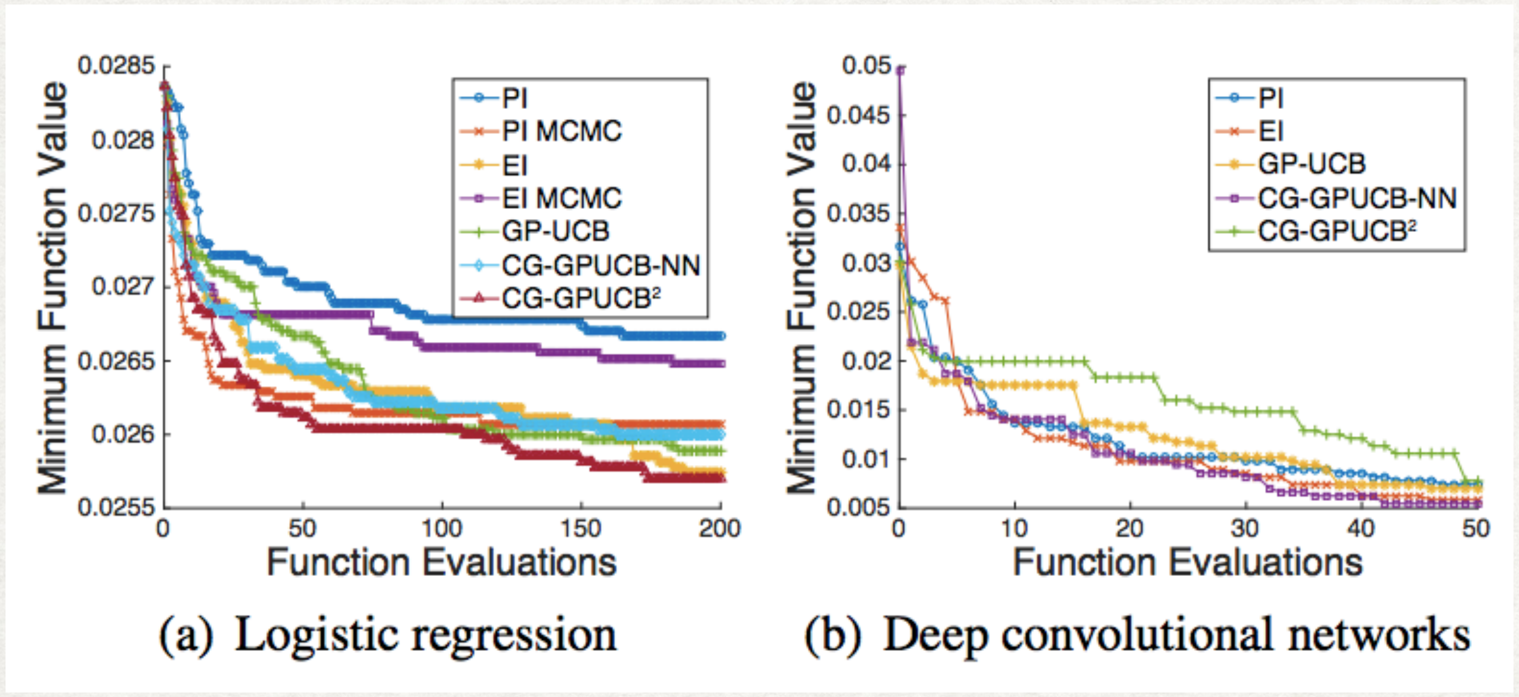
Ran GP-UCB, CGGPUCB-NN and CG-GPUCB^2 on synthetic function. It can be seen that CGGPUCB-NN performs slightly better than GP-UCB and CG-GPUCB^2 outperforms the two

PAPER IN DETAIL

EXPERIMENTS & RESULTS



(a) Branin-Hoo function (b) Cosines function (c) Sixhumpcamel function (d) Eggholder function



(a) Logistic regression (b) Deep convolutional networks

- For Branin-Hoo function, all acquisition functions except PI and EI showed that minimum function values were converged to almost the global optimum for 200 evaluations.
- For Cosines function, EI, GP-UCB, and CG-GPUCB2 found the global optimum for less than 100 iterations.
- For Sixhumpcamelfunction, most acquisition functions, except PI found the global optimum with similar performance.
- For Eggholder function, EI MCMC and CG-GPUCB2 converged to almost −900 for 200 function evaluations.
- For Logistic Regression EI and CG-GPUCB2 showed almost similar convergence performance, but CG-GPUCB2 was slightly better.
- For Deep convolution networks EI and CG-GPUCB-NN converged to near error rate 0.005% during 50 evaluations

FUTURE WORK

- Replace K-Means clustering with Bayesian clustering to remove hyper-parameter number of cluesters
- Extend the code to multiple iterations and run all experiments done in the paper and compare the results
- Try code with different kernels and hyper-parameters to see how we can get best results

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

THANK YOU !