School of Computing
FACULTY OF ENGINEERING

Batched Bayesian Optimization

Ruoxuan Wang

Submitted in accordance with the requirements for the degree of B.Sc Computer Science

2020/21

40 credits

The candidate confirms that the following have been submitted.

Items	Format	Recipient(s) and Date
Deliverable 1, 2, 3	Report	SSO (11/05/21)
Deliverable 4	URL of code and datasets	SSO (10/05/21)
Deliverable 5	Report	Minerva (10/05/21)

Type of project: Theoretical Study

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of Student) _____

Summary

Bayesian optimization is commoly used in machine learning fields. In each iteration, it selects one more sample to evaluate, and update the probability model. However, with the developmet of modern technology, parallel computing is available and this traditional way is not suitable for usage. It is more efficient to select a batch of points and evaluate them at the same time.

Therefore, in this project, the author will compare different batched Bayesian optimization algorithms to study their advantages and drawbacks, as well as ideas for extending the traditional Bayesian optimization to parallel computing.

Acknowledgements

I am extremely grateful to my supervisor, Dr. Dawei Zhan for his guidance and assistance at every stage of this project. Honestly speaking, completion of this report would have not been possible without him.

I would like to extend my sincere thanks to Dr. Mark Walkley for his detailed feedback on the intermediate report, provided me with useful comments and suggestions.

Moreover, thanks to those generous people who kindly shared their teaching material and course recording online (see appendix B), enabling me to learn the mathematical principles behind this research topic by myself.

Finally, a special 'thank you' to my family for their continued support and encouragement in the past 21 years.

Contents

Chapter 1

Project Management and Planning

1.1 Report structure

In this report, the following sections will be covered: Project management and planning will take up the first section, mainly covering the project aim and objectives, things to be submitted, and the initial schedule. Then, the author will introduce basic knowledge needed for the topic, including gaussian process model, bayesian optimization and batched bayesian optimization. In chapter 3, three popular algorithms and conducted experiments will be explained in detail, where the author will compare and learn from the result. Finally, the author will make self apprasial, reflecting on time management, obstacles and how she handled them, as well as things learned.

1.2 Aim

This project aims to carry out a comprehensive analysis, both theoretical and empirical, of different algorithms used for batched Bayesian optimization. In this project, the author learned principles from gaussian process model to traditional Bayesian optimization, and investigated ways to implement batched Bayesian optimization based on literature review. To test the performance of different algorithms, experiments was carried out on a variaty of functions with different batch size. Different algorithms were compared, by analyzing the outcome of experiments, and comparing the advanteges and drawbacks.

Hopefully, after reading this report, the reader can have a general understanding of algorithms that are widely used nowadays, as well as the principles they base on.

1.3 Objectives

- Have a deep understanding of Gaussian process, Bayesian optimization, batched Bayesian optimization, and understand the principles behind usages.
- Conduct background research into different algorithms of batched Bayesian optimization, learning from the pre-existing work.
- Implement three popular algorithms - Kriging Believer, Constant Liar, and Multi-Modal Expected Improvement
- Conduct experiments on different functions, and compare the performances on finding minimum value of these functions.
- Make comparison and contrast of the results from different algorithms, and analyze the advantage and drawback.

1.4 Deliverables

At the end of the project, the following things will be delivered:

- Source code to implement different algorithms
- Datasets generated for testing each algorithm
- Plots indicating the outcome of each experiment
- The final report

1.5 Schedule

Schedule of the project can be divided into three consecutive phrases: knowledge accumulation, coding, and analyzing.

In the first two months, the author will supplement mathematical knowledge, and read papers about the topic. Then the implementation of three popular algorithms will take three months, including not only coding but also detailed testing on different functions with different parameters. Phrase 3 will be conducted in the mean time: The author will analyze the result of each experiment and make conparisons. And writing the academic report will take two months, from late March to early May.

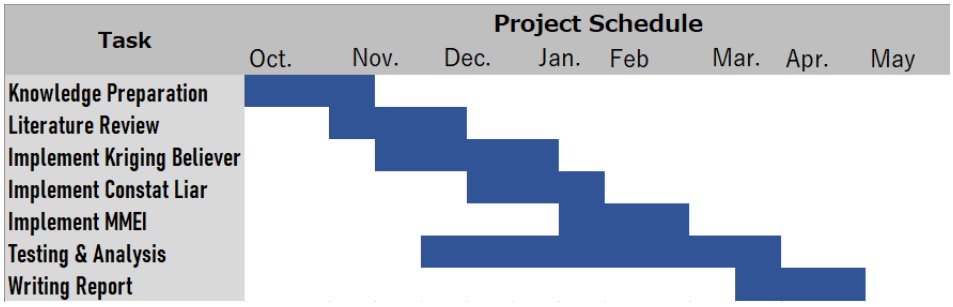


Figure 1.1: Schedule

Chapter 2

Background

In the engineering field, optimization problems are omnipresent. And in real life, this kind of optimization problems have objective functions that are expensive to evaluate, for example, conducting geological exploration for petrol or natural gas at given geographic coordinates, evaluating the efectiveness of a certain composition taken from a chemical search space, and adjusting the hyperparamenter of a neutrual network that has many layers (Krasser, 2018). For an expensive objective function, it is vital to approach the optimal solution in an efficient way.

Bayesian optimization is a proven method, it iteratively select a new point and evaluate the objective function using the existing points (newly-selected point included), then update the probabilistic surrogate model before the next iteration.[1]

In this chapter, knowledge related to this project will be introduced in a bottom-up approach. Hope that the reader can get to know essentiall background of my research topic in this way.

2.1 Gaussian Process Model

2.1.1 Brief

In the field of probability theory and statistics, Gaussian process is a stochastic process that any finite collection of random variables has a joint distribuiton that is multivariate Gaussian[4].

In many machine learning algorithms which involves a Gaussian process, the Gaussian process indicates the distribution of function. With a kernel function (will be covered in 2.1.3) defining the similarity between point in training dataset and predicted point, we make predictions and train the model.

2.1.2 Gaussian Process

Given any point $\mathbf{x} \in \mathbf{R}^d$ in Gaussian process, there is an assigned corresponding random variable $f(\mathbf{x})$, where the joint probability distribution of every finite subset of random variables $f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_N)$ will conform to Gaussian distribution.

Let

$$\mathbf{f} = (f(\mathbf{x_1}), f(\mathbf{x_2}), \cdots, f(\mathbf{x_N})), \quad \text{and} \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \cdots \\ \mathbf{x}_N \end{pmatrix}$$

Then the multivariate Gaussian distribution can be written as

$$p(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}, \mathbf{K}) \tag{2.1}$$

with μ representing the mean vector $m(X) = (m(\mathbf{x_1}), m(\mathbf{x_2}), \cdots, m(\mathbf{x_N}))$ and $\mathbf{K}$ representing

the covariance matrix. In the field of machine learning, $\mathrm{m}(X)$ is usually set to be zero because we assume that the arbitrarily can be well-modeled by Gaussian process itself.

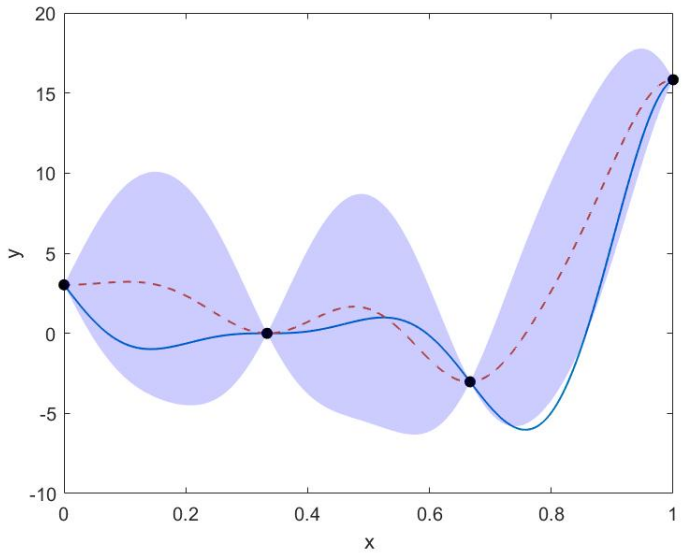


Figure 2.1: A function with its gaussian process

2.1.3 Kernel Function

Given a multivariate Gaussian distribution, the covariance matrix $\mathbf{K}$ in equation (1) can be denoted as $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, where each element in the matrix is the covariance of $\mathbf{x}_i$ and $\mathbf{x}_j$, and κ is the covariance function, also called kernel function.

The value of kernel function will reflect similarity between two points, $\mathbf{x}_i$ and $\mathbf{x}_j$: if the kernal function decided that $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, then we can infer that $f(\mathbf{x_1})$ and $f(\mathbf{x_2})$ are also similar. Thus, selection of kernel function defines the behaviour and performance of gaussian process.

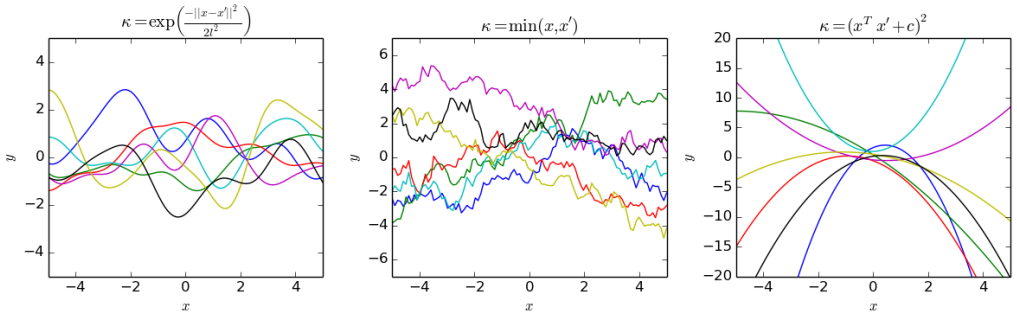


Figure 2.2: Using different kernels on prior function distribution of the Gaussian process [5]

A kernel function can define basic aspects such as process' stationarity, isotropy, smoothness and periodicity[3][4]. However, we are not using different kernel functions, as it is not central to this project. Instead, for all the experiments of this project, the *dace* Kriging toolbox will be used, where the kernel function was designed to be Gaussian kernel function.

2.1.4 Gaussian Process Regression

Gaussian process regression is a non-parameteric model that uses the Gaussian process prior to conduct regression analysis on given dataset.

To have a graphical overview, the author first use an example [5]: There are samples from a Gaussian process prior and posterior, where green lines indicate samples from the gaussian process, dots indicate past observations, and the red triangle indicates the next point to be selected.

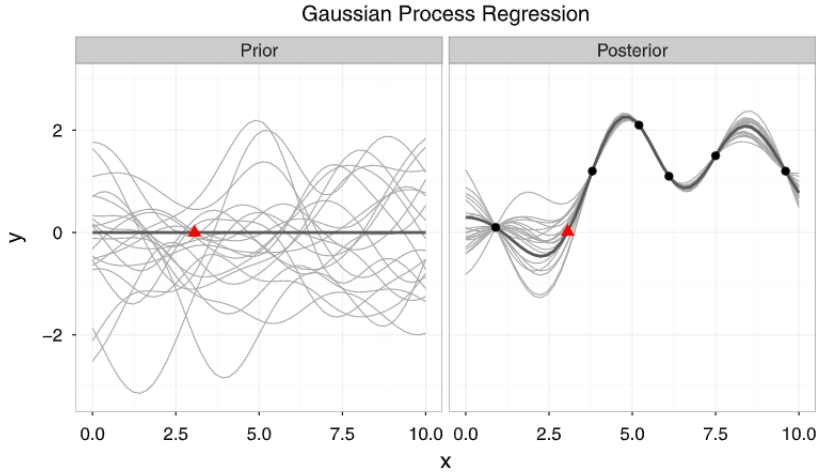


Figure 2.3: Gaussian process regression, an external example[7]

Given that the Gaussian process prior is $p(\mathbf{f} \mid \mathbf{X})$, the posterior of Gaussian process can easily yield as we know the corresponding $\mathbf{y}$. The posterior can be deoted as $\mathrm{p}\left(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}\right)$, where $\mathbf{X}_*$ is the new input and $\mathbf{f}_*$ is our new prediction.

As the prior and posterior are both Gaussian, random distribution of the prodiction $\mathbf{X}_*$ is still a Gaussian distribution, denoted as:

$$p\left(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}\right) = \int p\left(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{f}\right) p(\mathbf{f} \mid \mathbf{X}, \mathbf{y}) d\mathbf{f} = \mathcal{N}\left(\mu_*, \mathbf{\Sigma}_*\right) \tag{2.2}$$

where μ_* represents the expectation vector:

$$\mu_* = \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{y} \tag{2.3}$$

and $\mathbf{\Sigma}_*$ represents the covariance matrix:

$$\Sigma_* = \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{K}_* \tag{2.4}$$

As the joint distribution of collections of finite random variable in a Gaussian process is a Gaussian distribution, the joint distribution of $\mathbf{y}$ and $\mathbf{f}_*$ still follows Gaussian distribution:

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_y & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right) \tag{2.5}$$

where

$$\mathbf{K}_y = \kappa(\mathbf{X}, \mathbf{X}) + \sigma_y I \tag{2.6}$$

$$\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*) \tag{2.7}$$

$$\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*) \tag{2.8}$$

In equation (6), I is the identity matrix of size $N \times N$, and σ_y is the noise to the diagonal of $\mathbf{K}_y$. In this project, to simplify, we assume that there's no noise and σ_y was set to be 0.

2.2 Bayesian Optimization

2.2.1 Brief

In the field of machine learning, optimization problems are everywhere. Many optimization problems have the objective function that is a black box function. It is hard to evaluate a black box function $f(\mathbf{x})$ that only input and output are known. Therefore, we need to focus on a model that is somehow 'forseeable', and use that model to approximate the objective function.

Bayesian optimization is applied typically in optimization problems like this: $\max_{x \in A} f(x)$, with A representing a set of points whose membership can easily be evaluated

Method of bayesian optimization is to assume the objective function to be a random function, construct a probability model over it, and use the model to select the promising point that most likely to bring hugest improvement to the model in the next evaluation. Then in the next loop we add that point to the set of sample points and update the model... Going round and round till it meet the condition to stop.

2.2.2 Surrogate Model

A surrogate model is the model we used to find the next sample point $x_t = \arg\max_x \alpha(x)$ where x_t represents the acquisition function to be covered next section.

In this project, we use the Gaussian process model as surrogate model. By using the pre-defined prior over the objective function, Gaussian process allows us to use the pre-defined prior to incorporate prior beliefs (e.g.: shape) about the objective function[8].(Krasser, 2019) And in a Gaussian process, the posterior can be used to make prediction of the next promising point in search space that is likely to yield bigest improvement.

Gaussian process model is an efficient surrogate model, as the posterior of Gaussian process is cheap to evaluate.

2.2.3 Acquisition Function

Acquisition function acts like a 'measure' that used to select the next promising point (i.e.: point that most likely to bring huge improvement).

In general, there are two strategies[9] to select a new point:

- Explore: find a new point on the function, this way is helpful to get an accurate point.
- Exploit: make selection near the already selected points, this way is helpful to find max value.

The above two strategies are contradict to each other, thus we need to make trade-off and find a balance between them.

In this project, we will use the Experiment Improvement function, as it not only has an analytic form that is easy to compute, but also has good practical performance.

As what it was called, experiment improvement function elect the point with the greatest expected improvement as the next sample point, that is:

$$x_{t+1} = \arg\min_x \mathbb{E}\left(\|h_{t+1}(x) - f\left(x^*\right)\| \mid \mathcal{D}_t\right) \tag{2.9}$$

where f represents the objective function, h_{t+1} represents the mean of posterior of the surrogate model at step $t+1$. And $\mathcal{D}_t = \{(x_i, f(x_i))\}, \forall x \in x_{1:t}$ is the sample used for training, and x^* is the place where the function f reaches its maximum value.

To select the point that near to the max point of objective function, we apply Mockus's way:

$$x_{t+1} = \arg\max_x \mathbb{E}\left(\max\left\{0, h_{t+1}(x) - f\left(x^+\right)\right\} \mid \mathcal{D}_t\right) \tag{2.10}$$

where $f(x^+)$ represents the point among we already selected, where the function value reaches max.

Using gaussian process as surrogate model, yields:

$$EI(x) = \begin{cases} \left(\mu_t(x) - f\left(x^+\right) - \epsilon\right)\Phi(Z) + \sigma_t(x)\phi(Z), & \text{if } \sigma_t(x) > 0 \\ 0 & \text{if } \sigma_t(x) = 0 \end{cases} \tag{2.11}$$

$$Z = \frac{\mu_t(x) - f\left(x^+\right) - \epsilon}{\sigma_t(x)} \tag{2.12}$$

where $\Phi(Z)$ represents cumulative density function (CDF) and $\phi(Z)$ represents Probability density function (PDF).

2.2.4 Process

The following pseudocode briefly introduced the process of Sequantial Model-Based Optimization, which is the simplest form of Bayesian optimization:

Algorithm 1 Sequential Model-Based Optimization

Input: $f, \mathcal{X}, S, \mathcal{M}$
$\mathcal{D} \leftarrow$ INITSAMPLES $(f, \mathcal{X})$
for $i \leftarrow |\mathcal{D}|$ to T do
$p(y \mid \mathbf{x}, \mathcal{D}) \leftarrow$ FITMODEL $(\mathcal{M}, \mathcal{D})$
  $\mathbf{x}_i \leftarrow \arg\max_{\mathbf{x}\in\mathcal{X}} S(\mathbf{x}, p(y \mid \mathbf{x}, \mathcal{D}))$
$y_i \leftarrow f(\mathbf{x}_i)$ ▷ Expensive step
$\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{x}_i, y_i)$
end for

where f represents the black box function, X represents the domain of f, D represents the dataset containing pairs of data (x, y) where x is the input and y is the corresponding output. And S is the acquisition function used to select the next sample point (to be introduced in 2.2.3). M is the model evaluated from the dataset D, Gaussian process model will be used in this project.

From above, easy to know that bayesian optimization has the following steps:

1. Model the objective function f using the surrogate model, and define the prior of objective function (i.e.: sample points).

2. Use the acquisition function (EI function at this project) to find the promising point $\mathbf{x}_i$ that has the best performance on the surrogate model (i.e.: x of the new point).

3. Calculate value of the objective function at that new point (i.e.: y).

4. Add point (x, y) to the dataset D, update surrogate model with updated D

5. Repeat step 2-4 till reaches the maximum teration.

Here's a visualization of Bayesian optimization. We can see that, in each eveluation, the point where Experiment Improvement function reaches maximum was selected as the next sample point.

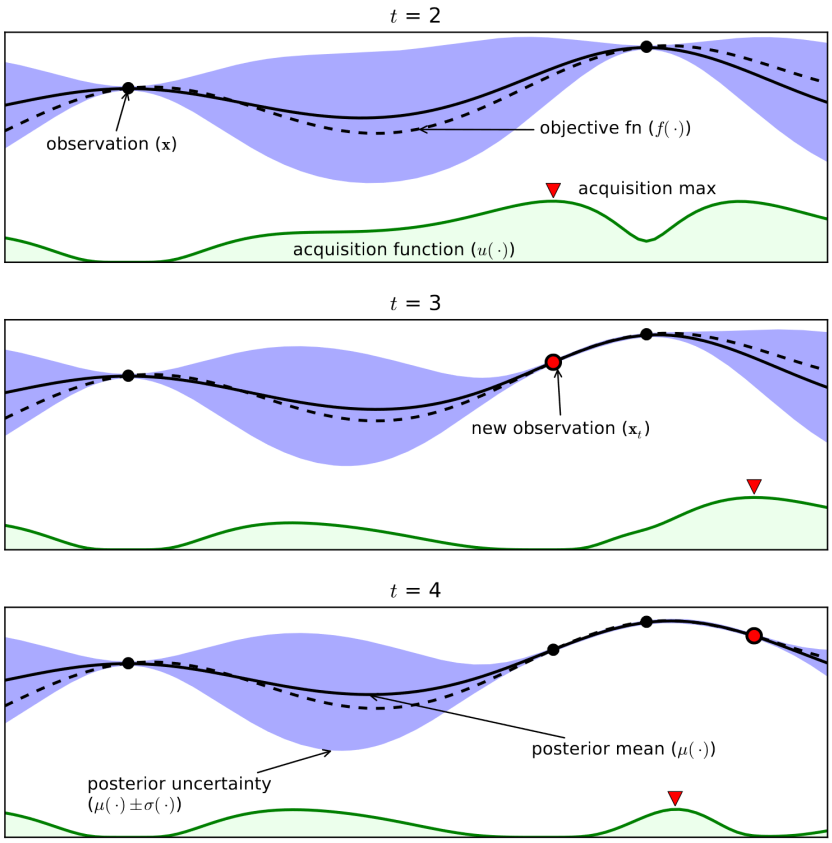


Figure 2.4: An example, visualizing bayesian optimization [12]

2.2.5 Useful Conditions

Bayesian optimization techniques are most useful in the following type of functions:

1. Objective function that expensive to evaluate:
   This can be put down to two facts: Firstly, the majority of time taken is the time used for calculating the value of objective function in the newly-selected sample point at each iteration. Secondly, compared to grid search or random search, Bayesian optimization help

us to select the promising point for evaluatin and thus we calculate fewer points. As a result, Bayesian optimization reduces both the time and number of times for calculations.

2. Objective function that is a black box function and we do not know its mathematical expression.

3. Objective function that is derivative-free (or else we can use other methods like the gradient descent method)

2.3 Batched Bayesian Optimization

From the flowchart below, it cam be seen that there are three steps in traditional bayesian optimization:

1. Using EI function to find the point.

2. Calculate objective function at that point.

3. Add point to sample and update the model.

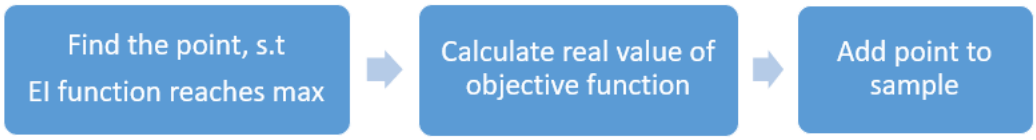


Figure 2.5: An iteration of traditional bayesian optimization

Changing from traditional Bayesian optimization to batched Bayesian optimization is simple in theory, as it only requires the change in step 2:

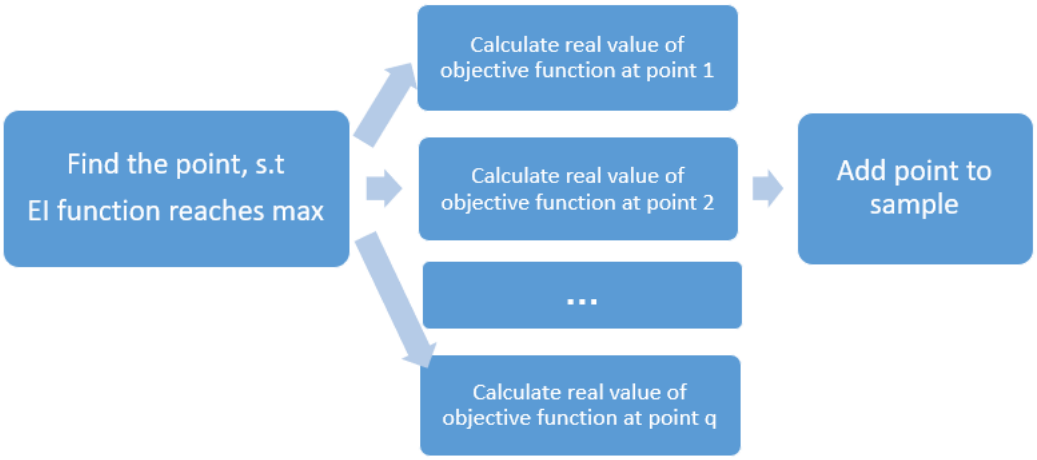


Figure 2.6: An iteration of batched Bayesian optimization

In step 2, a batch of points will be selected at one time (suppose the batch size is q), and will be calculated in parallel, then in step 3, value of the opjective function at these q points will be added to the sample at each iteration.

Chapter 3

Experimental Design

This chapter will cover the implementation of three most popular algorithms, as well as the experiments conducted.

3.1 Improving Reliability

In order to test and campare the performance of those algorithms, the author will apply each of them on five existing functions, use the algorithm to find the minimum value, and make comparison with real minimum value of the functions.

Function name	Num of dimensions	Minimum value
Sixhump	2	-1.0316
Sasena	2	1.4565
Ellipsoid	10	0
Rosenbrock	20	0
Ackley	20	0

The author made following efforts to make the experiments convincing and credible:

1. Used functions with mathematical expression, instead of black box functions. Due to the fact that, minimum value of these functionsare already known, and thus the comprison between real minimum value and minimum found by alrorithms will based on hard facts. As a result, it will be more scientific.

2. Used functions of different varieties (e.g.: some are exponential, some has the shape of a ellipsoid, some is widely used for Kriging Approximations), which ensured the breath of experiment.

3. Used average performance as result. As sample points are selected randomly, which will affect the input data, the author conducted each experiment for 10 times, and calculate the average value of outcomes.

4. Comprehensive attempts of q. Despite knowing that larger batch size will improve the performance, for each algorithm, the author set the batch size to be 1,2,4,8, and made experiments separately. This helps the author to explore marginal effect of improving q.

3.2 Experiment Process

The testing process of each algorithm including the following steps:

1. Initializing: Before testing, upper and lower bound of the function was set. Then the author set dimension of the random dataset to be selected. Then, the threshold number to stop iteration was set to 20.

2. Selecting sample points: as this is a controlled experiment, we need to omit the influence of irrelevant variables. Thus, for each algorithm, the same method, Latin hypercube sampling (LHS) was applied.

3. Apply the surrogate model and acquisition function: In this experiment, Kriging toolbox was used, where zero order polynomial was used as regression model, and gaussian kernel (as mentioned in 2.1.3) was used as kernel function.

4. Trying different algorithms: For each function, different algorithms was applied, in order to select a batch of points, and update the model.

Chapter 4

Multi-Model Expected Improvement

4.1 Method

From Chapter 2, it can be known that the promising point is the point where Experiment Improvement function reaches its maximum. Therefore, to select a batch of (q) points at one iteration. A. S´obester, S.J. Leary et al [7] put forward an algorithm for selecting the top q points where experiment improvement (EI) function reaches its top q maximum value.

From figure 2.4, it can be noticed that EI function has many local maximum values:

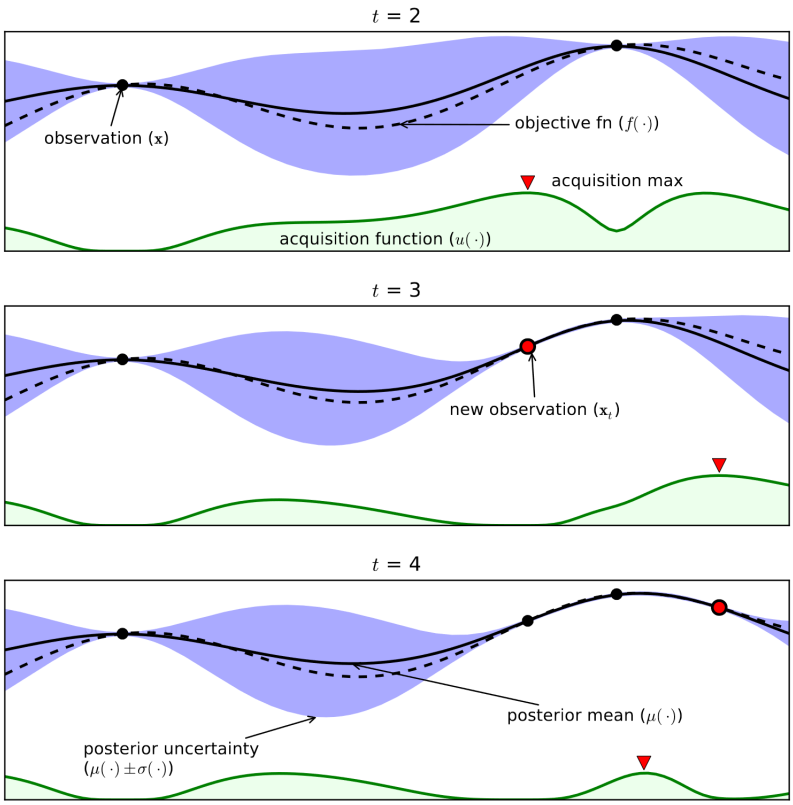


Figure 4.1: An example, visualizing bayesian optimization

This kind of shape brings the idea that when searching the next point to be evaluated, the "peak" on EI function, whose local maximum was already selected can be excluded, and when selecting the next point, only other peaks need to be searched.

To implement this idea, an algorithm was developed: to set a constraint at the first "peak", and ignore points within the constraint when searching. From pre-existing works, we know that there is an relationship between constraint and difference between lower and upper bound:

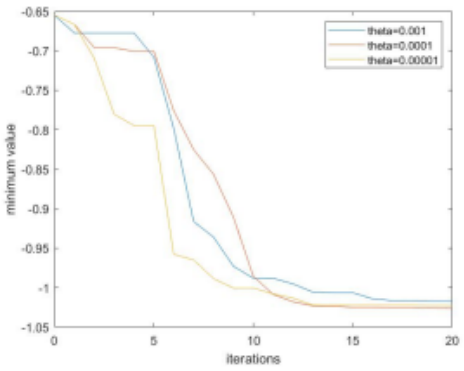
$$C = \theta \times (U - L) \tag{4.1}$$

where C is the constraint, U is the upper bound and L is the lower bound. The author tried θ

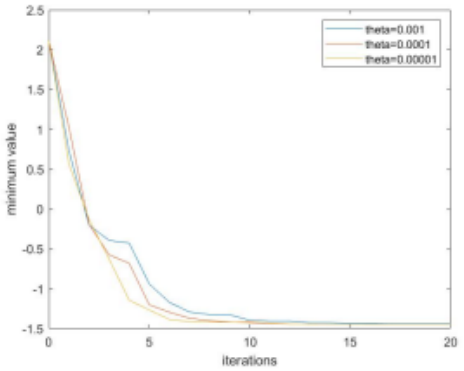
to be 0.001, 0.0001, 0.00001, with a changing q on each function:

4.2 Experiment 1.1

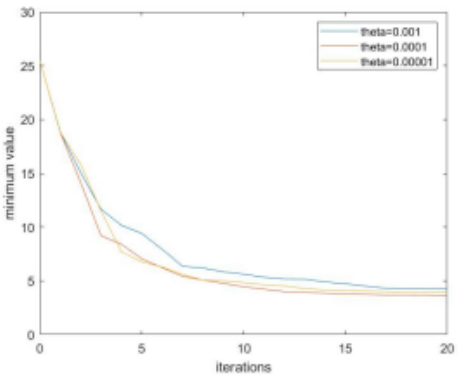
Setting $q = 2$ on five functions, yields $\theta = 0.0001$ brings the best performance on all the five functions:



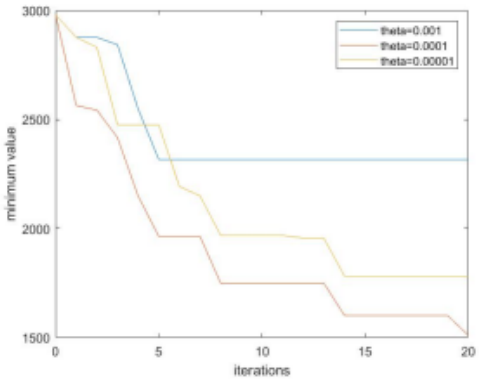
Sixhump, q=2 min=-1.03



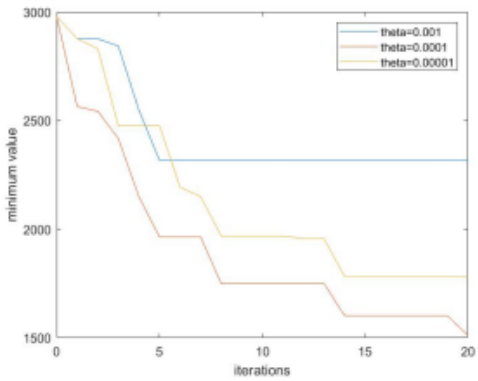
Sasena, q=2 min=1.45



Ellipsoid, q=2, min=0



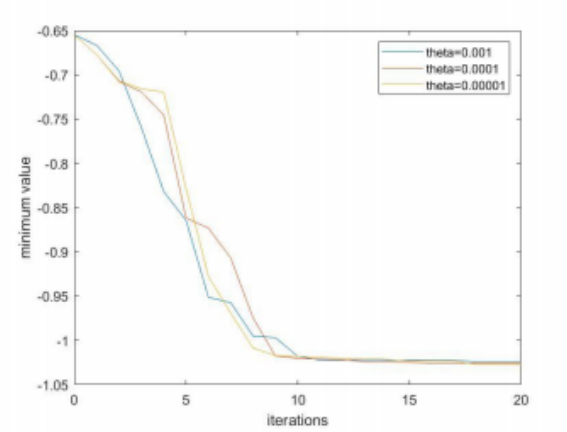
Rosenbrock, q=2 min=0



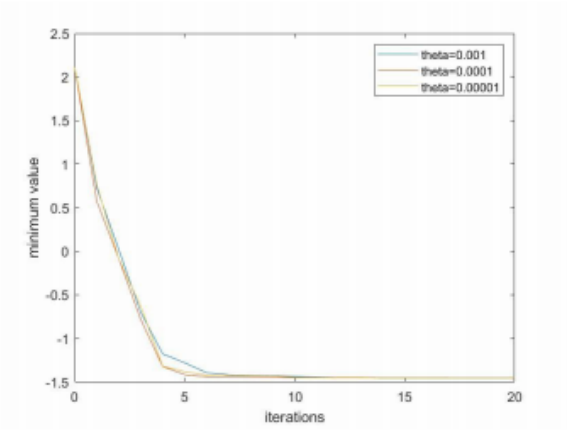
Ackley, q=2, min = 0

4.3 Experiment 1.2 and 1.3

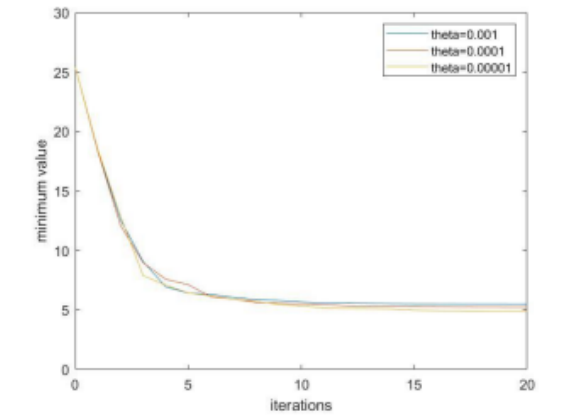
In experiment 2 and 3, setting $q = 4$ and $q = 8$ respectively on five functions, also yields $\theta = 0.0001$ brings the best performance on all the five functions. However, for function *Sixhump*, *Sasena* and *Ellipsoid*, whose dimensions were set to be lower (2, 2 and 10 respectively), the speed of convergence got diamatically faster: the three lines on the plot almost coinside with each other.



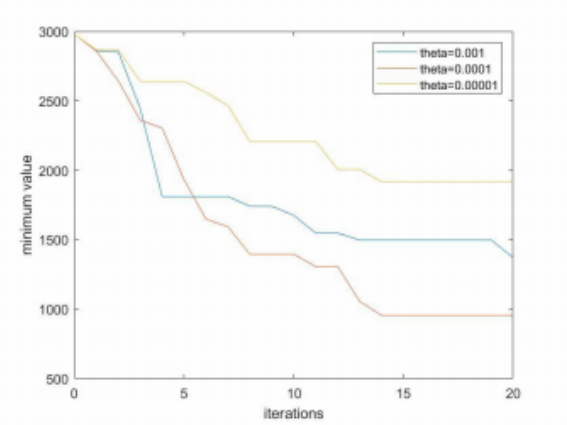
Sixhump, q=2 min=-1.03



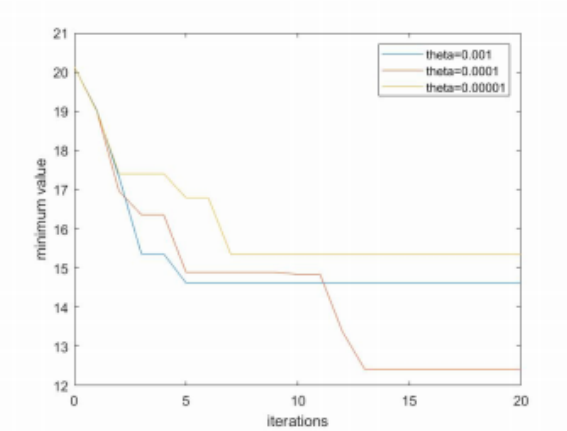
Sasena, q=2 min=1.45



Ellipsoid, q=2, min=0



Rosenbrock, q=2 min=0

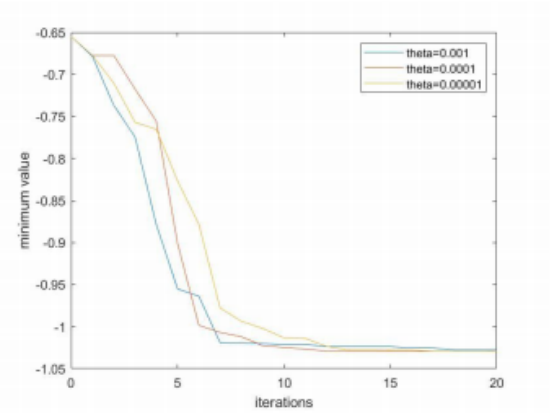


4.4 Conclusion of Experiment 1

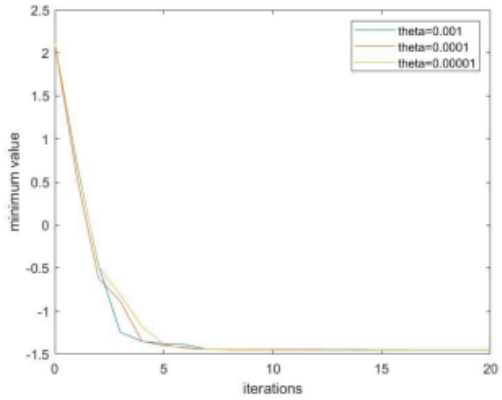
From experiment 1, conclusion can be drawn that:

1.For the constraint we set in equation(3.1), $\theta = 0.0001$ brings the best performance.

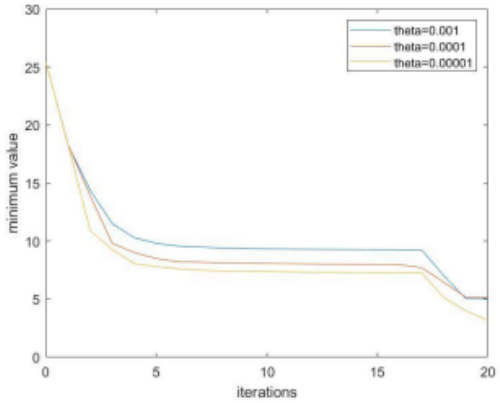
2. As batch size goes up, the speed of convergence also goes faster, especially in functions with imput data of lower dimensions.



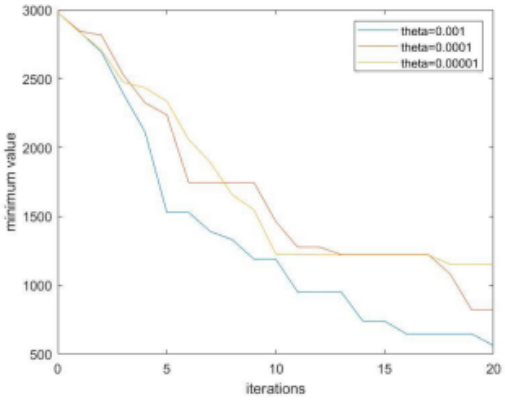
Sixhump, q=8 min=-1.03



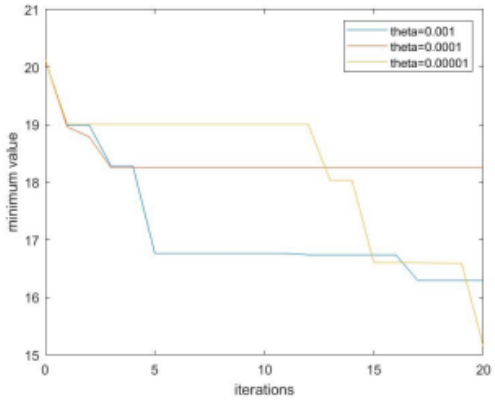
Sasena, q=8 min=1.45



Ellipsoid, q=8, min=0



Rosenbrock, q=8 min=0



Ackley, q=8, min = 0

Chapter 5

Kriging Believer

5.1 Method

Due to the fact that the objective function is expensive to calculate, again look at the flowchart:

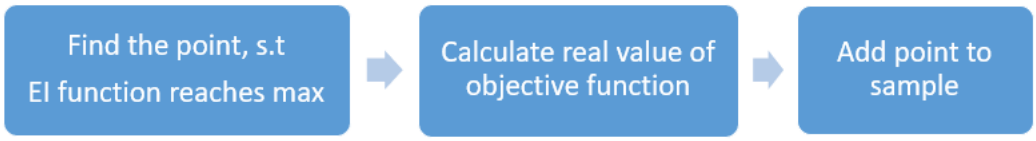


Figure 5.1: An iteration of traditional bayesian optimization

before putting the newly selected point to the set of sample points in step 3 (at the flowchart), the value of objective function at this point needs to be calculated. Calculating these values one by one will take a lot of time. Thus, we need to calculate them in parallel.

Nevertheless, we have known that bayesian optimization contains iterations, each iteration base on the past. That is, the value obtain in step 3 is the value to be used in step 1 in next iteration, we can't execute step 1 in the next iteration without a new sample point being added. To tackle this problem, Mocus [3] put forward an algorithm:

using the using the outcome of Gaussian Process model instead of the real value of objective function as selected points, and then in step 3 we calculate the real value in parallel. Which means that, within a batch, in step 3 of each iteration we use the value that evaluated by GP model as the 'fake' point, and use that 'fake' point in the next iteration. After all the 'fake' points in that iteration were selected, we then calculate the objective function's real value at these points in parallel.

This algorithm saves time not only because we calculate those points in parallel, but also due to the fact that Gaussian Process model is cheap to evaluate.

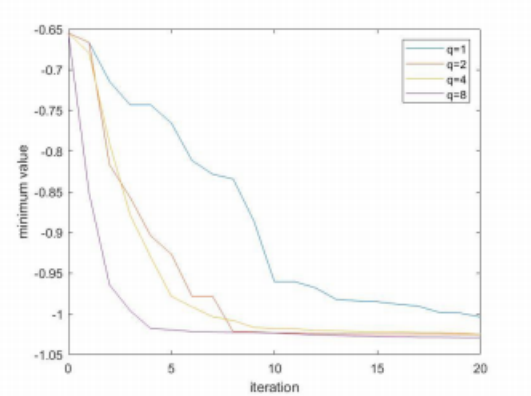
5.2 Experiment

In this experiment, the author also tried the value of q to be 1,2,4,and 8. But at this time, the author draw a plot for each function, withi different lines representing the outcome of different q. Uncer this circumstance, the author can not only compare the influence on different functions, but also can compare the influence of q in each function

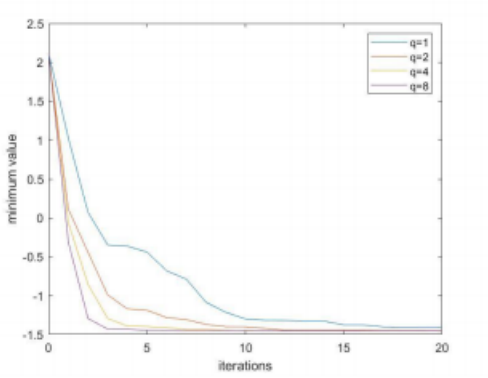
5.3 Conclusion

This experiment gives the conclusion from a straight-forward way. From each seperate graph, it is also obvious that, an increasing q leads to a fast convergence speed.

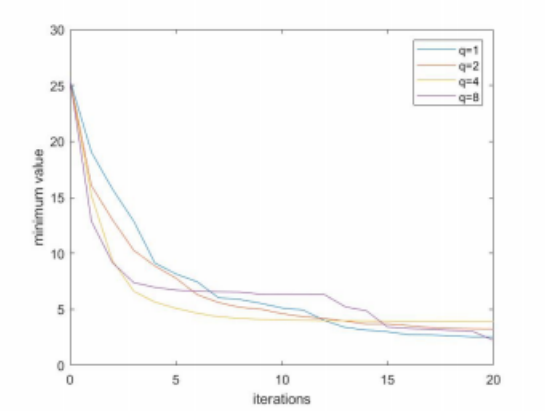
As for the accuracy, Kriging Believer is accurate on function Sixhump, Sasena and Ellipsoid. But on function Rosenbrock and Ackley, it turned out that it is not accurate sometimes, and it is amazing to find out value of q will even influence accuracy at these two functions.



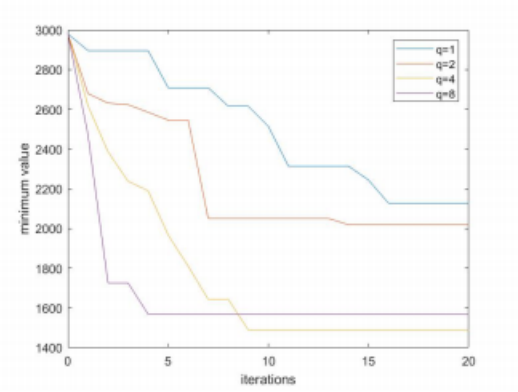
Sixhump, min=-1.03



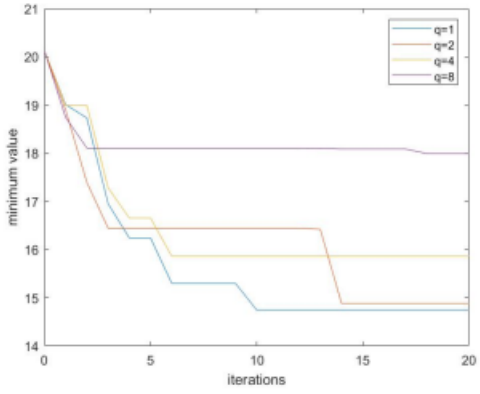
Sasena, min=1.45



Ellipsoid, min=0



Rosenbrock, min=0



Ackley, min = 0

Chapter 6

Constant Liar

6.1 Method

Constant Liar is a similar way as Kriging Believer, in layman's terms this algorithm also has the method of using a 'fake' point at each iteration, and calculate the real value at 'fake' points after we iterate through each iteration inside one batch. But Constant Liar works even more direct: instead of using a model to evaluete those 'fake' points, originator of this algorithm tend to choose real values (e.g.: maximum value) in sample points, and use these values directly as 'fake' points. [2] In the paper, Ginsbourger Riche et al tried to use sample points that has maximum value, which turned to have a great performance. In this experiment, the author will try maximum value, minimum value, and average value.

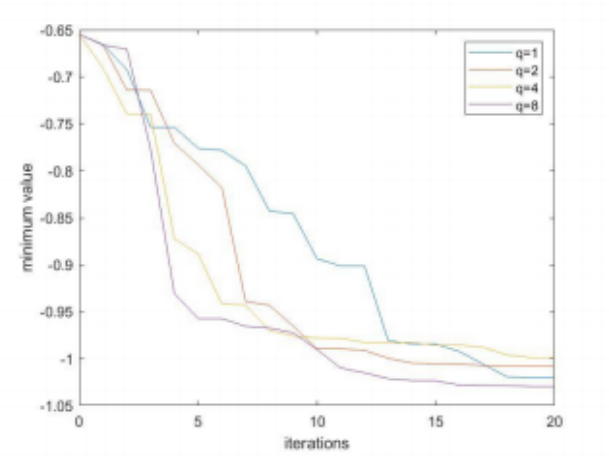
6.2 Experiments

The author tried maximum value, minimum value, and average value of sample points as the value used as 'fake' point:

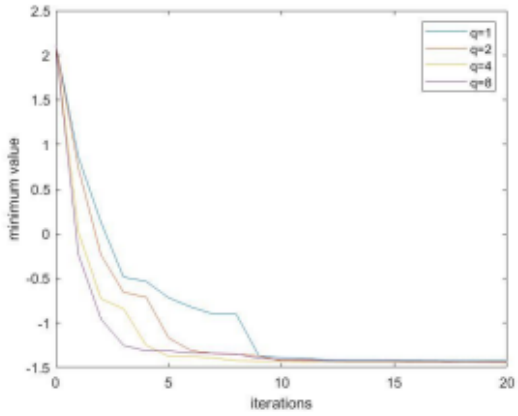
6.3 Conclusion

In this experiment, consider the performance of the algorithm on all the five function sit tiurs out that using average value of sample points directly as 'fake' points has the best performance.

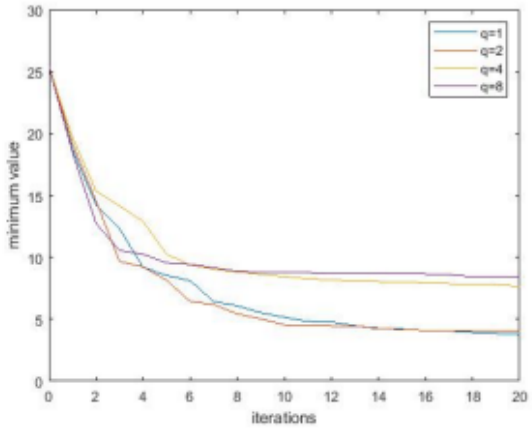
Average Value



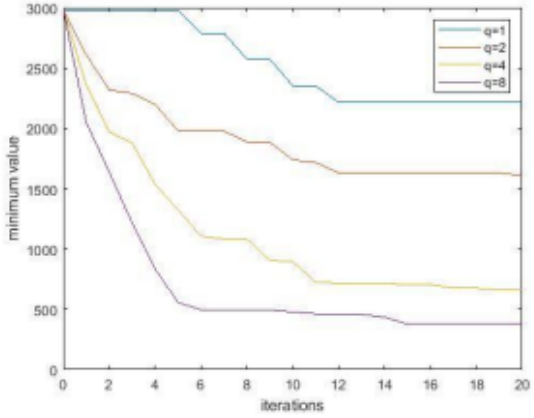
Sixhump, min=-1.03



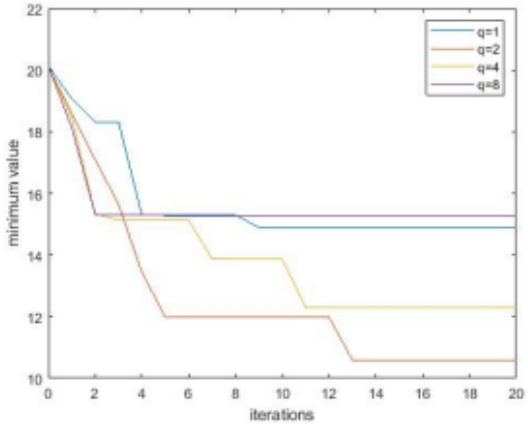
Sasena, min=1.45



Ellipsoid, min=0

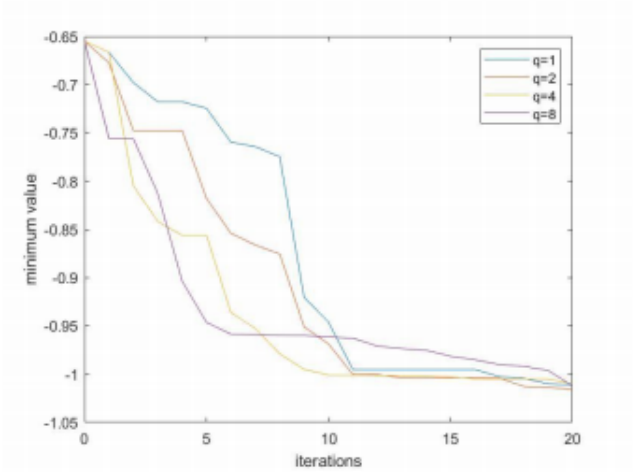


Rosenbrock, min=0

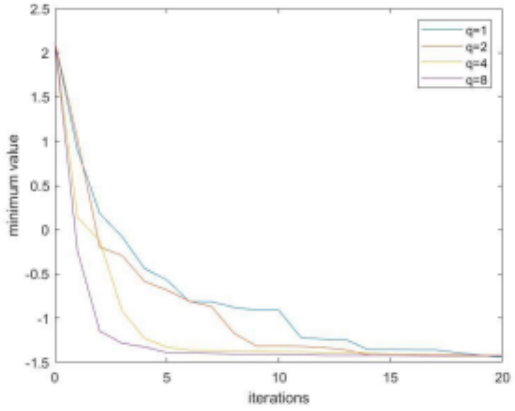


Ackley, min = 0

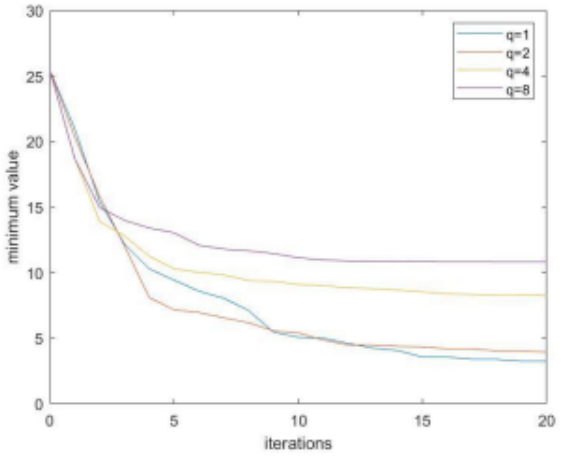
Maximum Value



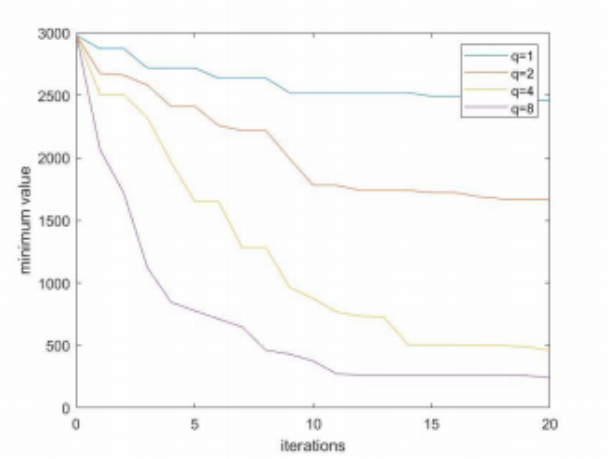
Sixhump, min=-1.03



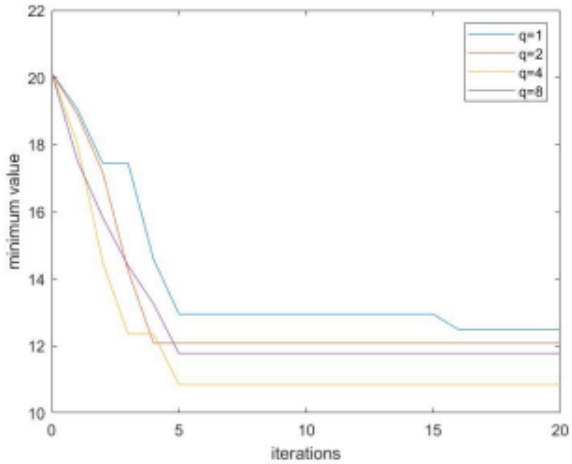
Sasena, min=1.45



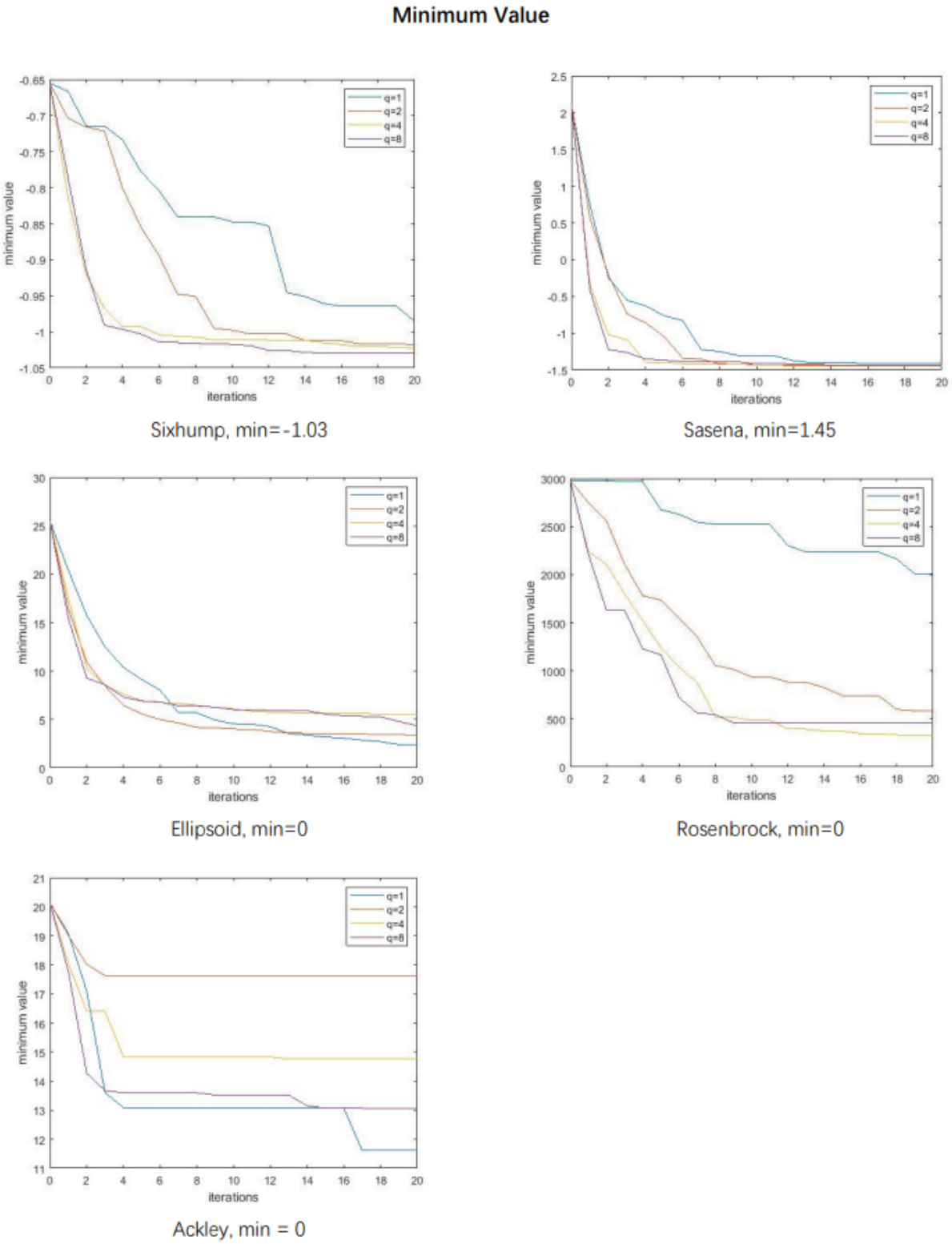
Ellipsoid, min=0



Rosenbrock, min=0



Ackley, min = 0


Minimum Value
minimum value
iterations
q=1
q=2
q=4
q=8
Sixhump, min=-1.03
Sasena, min=1.45
Ellipsoid, min=0
Rosenbrock, min=0
Ackley, min = 0

Chapter 7

Self Appraisal

7.1 Timeline

As the mathematical formula is a bit hard for me, the time used for background research significantly increased:

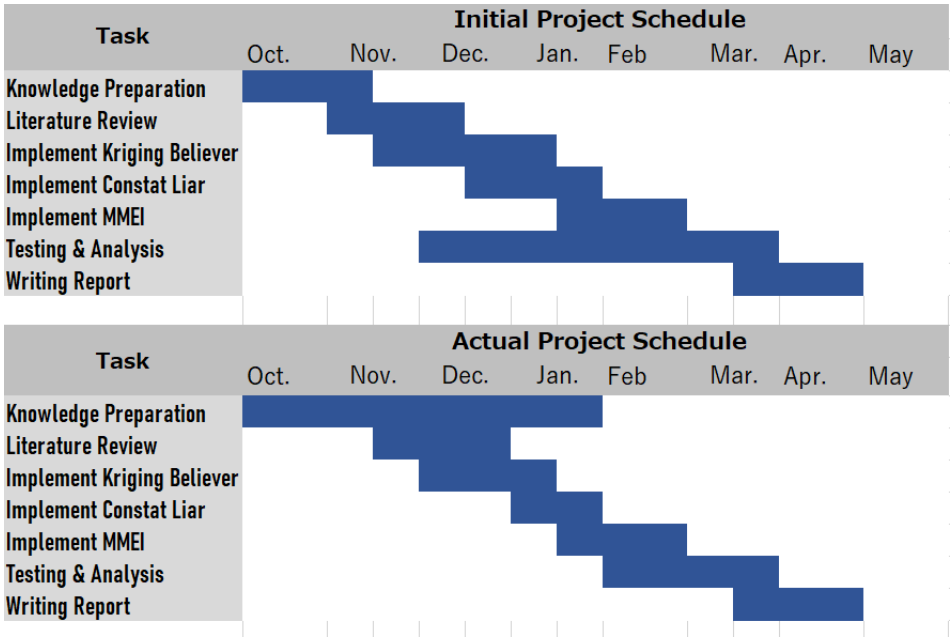


Figure 7.1: initial vs actural chedule

7.2 Obstacles

The biggest obstacle lies on Mathematics:both trying to understand the mathematical formula behind background knowledge and the mathematical deriviation of equations in the paper. To tackle this problem, I managed to find resources on youtube, and read some of the blogs (see appendix B). Again, thanks to those people who generously shared these material online!!!

the next obstacle is to use LATEX. I have to admit that I'm not very good at it. The last day before deadline, my LATEX file could not compile, and all the pictures were in a mass. I used two days to fix this issue. Which lead to the late submit. Some pictures are a bit obscure, as i used Microsoft word to align them, and took a screenshot, then put them into LATEX. Sorry for the inconvenience.

References

[1] D. Barber. *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[2] D. Ginsbourger, R. Le Riche, and L. Carraro. *Kriging Is Well-Suited to Parallelize Optimization*, volume 2 of *Adaptation Learning and Optimization*, book section 6, pages 131–162. Springer Berlin Heidelberg, 2010.

[3] J. Mockus and L. Mockus. Bayesian approach to global optimization and application to multiobjective and constrained problems. *Journal of optimization theory and applications*, 70(1):157–172, 1991.

[4] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning.* The MIT Press, Cambridge, MA, USA, 2006.

[5] E. Schulz, M. Speekenbrink, and A. Krause. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

[6] E. Schulz, M. Speekenbrink, and A. Krause. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

[7] A. Sóbester, S. J. Leary, and A. J. Keane. A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Structural and Multidisciplinary Optimization*, 27(5):371–383, 2004.

Appendices

Appendix A

External Deliverable

Source code and dataset repository:
`https://github.com/MileyWrx/Fnial-year-project`

Appendix B

Learning Material

1. Lecture and notes on random variables, course CS70 at UC Berkeley:
https://www.eecs70.org/static/notes/n15.pdf
https://www.youtube.com/watch?v=Txj62swC6rM&list=PLzAv_uHZw7dTI2e0F8-lxxOWV9zXMzwNE&index=18

2. Lecture and explaination on Gaussian process, course CS4780, Cornell University:
https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote15.html
https://www.youtube.com/watch?v=R-NUdqxKjos&feature=emb_imp_woyt

3. Tutorial: GP model for machine learning, University of Cambridge:
http://mlg.eng.cam.ac.uk/zoubin/tut06/snelson-gp.pdf

4. Recording of a talk about Bayesian optimization at Gaussian Process Summer School at Sheffield:
https://www.youtube.com/watch?v=YB64VoGQsK8

5. The following blogs:
Yeliang Fan https://leovan.me/cn/
Gao Peng https://ggaaooppeenngg.github.io/zh-CN/
Martin Krasser http://krasserm.github.io/

6. Refenence of other resources:
picture on Bayesian Optimization [Online]. [Accessed April 22, 2021]
Available from: https://miro.medium.com/max/3074/1*PhKGj_bZlND8IEfII426wA.png