Modifications of PI and EI under Gaussian Noise Assumption in Current Optima

Huabing Wang

School of Informatics, University of Edinburgh, Edinburgh, UK
s1893474@ed.ac.uk

Abstract

Bayesian optimisation is a widely used tool for the hyper-parameter optimisations of black box functions. It implements a cheaper surrogate model such as Gaussian processes (GPs) to model search space. Acquisition functions on the top of GPs such as Probability Improvement (PI) and Expected Improvement (EI) are used to query the distribution of loss at all unevaluated positions in order to find the best one in theory. Traditionally, both acquisition functions use current optima in computations directly, but GPs assume that observations are noise corrupted. In this work, we mathematically derive modify PI and EI under Gaussian noise assumption. Modified PI and EI are compared with original versions on benchmark functions. We show that modified versions converge faster in same number of iterations and can achieve better performance in complex loss functions with reduced iterations.

Keywords

Bayesian Optimization, Acquisition Functions, Benchmark Functions

1. Introduction

Machine learning has achieved phased success. However, almost all machine learning models need to optimize hyper-parameters, such as Neural Networks, Topic Model and Random Forest. In practice, tuning the hyper-parameters includes methods such as Grid Search, Random Search [1], and Gradient-based Optimizations [2]. These mehthods are then designed in order to minimize empirical risk with the desired efficiency or convergence speed. Bayesian Optimization [3] is acted as a probabilistic approach that majorly implements Gaussian Process (GP) and utilizes its property of both prediction and uncertainty measure to achieve derivative-free optimization. It can be used when the gradient of function for optimizing is not accessible.

For Bayesian optimization, J Snoek et al. summarizes the applications in the field of machine learning, and numerical simulation shows that Bayesian optimization has the characteristics of high efficiency and strong convergence [4]. Martin Pelikan further find that hierarchy can be used to reduce problem complexity in black box optimization [5]. K Swersky et al. extends multi-task Gaussian processes to the framework of Bayesian optimization, and aims to transfer the knowledge gained from previous optimizations to new tasks in order to find optimal hyperparameter settings more efficiently [6]. J Snoek et al. further explores the use of neural networks as an alternative to GPs to model distributions over functions [7].

In fact, the principle of Bayesian Optimization is like reinforcement learning, it updates the modelling to hyper-parameters after each evaluation and then calculate the location for the next evaluation. Acquisition functions are used to calculate the desirability of each unevaluated locations, which also trades off between exploration and exploitation. Typical Acquisition functions are Upper Confidence Bound (UCB), Probability of Improvement (PI) and Expected Improvement (EI). However, the above acquisition functions does not fully take into account the deviation in the machine learning data collection process, that is, the noise contained in the current optimal. Based on the assumption of normal noise, we propose the corresponding modified version of PI and EI acquisitions, derive the corresponding explicit equations, and through a large number of numerical simulations and comparisons, the results indicates the feasibility of our proposed acquisition function.

2. Algorithms

2.1. Gaussian Process

Gaussian process [8] can be considered as a proxy for a black-box function which enables uncertainty quantification. Gaussian process (GP) is infinite-dimensional multivariate Gaussian distribution. Covariance matrix of this distribution is defined by kernel functions $k(\cdot,\cdot)$. Imagining $\mathbf{x}$ forms a finite discretization of input space. Assuming the distribution has zero mean, prior draws $\mathbf{f}$ can be simulated:

$$f|\mathbf{x} \sim \mathcal{N}(0, \mathbf{K}) \tag{1}$$

Statistical assumptions about the GP prior are represented in kernel functions. A commonly adopted kernel function is Matern kernel [8], where ν controls the smoothness of gaussian process. Let $r = \mathbf{x}_i - \mathbf{x}_j$, Matern class of covariance function has the following definition:

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

with positive parameters ν and length-scale ℓ, Gamma function Γ and modified Bessel function K_ν. Smooth GP kernels assumes that if x and x' are close by, then f(x) and f(x') have similar values.

Given noise observations $\mathbf{y}_{1:n}$ at $\mathbf{x}_{1:n}$ where $y_i \sim \mathcal{N}(f_i, \sigma_y^2)$. For a new point $\mathbf{x}_{n+1}$, the joint probability distribution is given by

$$\begin{pmatrix}\mathbf{y}_{1:n} \\ f_{n+1}\end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K} + \sigma_y^2\mathbb{I} & \mathbf{k}(\mathbf{x}_{1:n}, \mathbf{x}_{n+1}) \\ \mathbf{k}(\mathbf{x}_{n+1}, \mathbf{x}_{1:n}) & k(\mathbf{x}_{n+1}, \mathbf{x}_{n+1})\end{bmatrix}\right) \tag{3}$$

where $\mathbf{K} = \mathbf{K}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n})$. After applying the rule for conditional gaussians, we can gather the posterior over function values $f_{n+1}|\mathbf{y}_{1:n}$, which follows a univariate Gaussian distribution $\mathcal{N}(\mu(\mathbf{x}_{n+1}), \sigma^2(\mathbf{x}_{n+1}))$ with

$$\mu(\mathbf{x}_{n+1}) = \mathbf{k}(\mathbf{x}_{1:n}, \mathbf{x}_{n+1})(\mathbf{K} + \sigma_y^2\mathbb{I})^{-1}\mathbf{y}_{1:n}$$
$$\sigma^2(\mathbf{x}_n + 1) = k(\mathbf{x}_{n+1}, \mathbf{x}_{n+1}) - \mathbf{k}(\mathbf{x}_{1:n}, \mathbf{x}_{n+1})$$
$$(\mathbf{K} + \sigma_y^2\mathbb{I})^{-1}\mathbf{k}(\mathbf{x}_{n+1}, \mathbf{x}_{1:n})$$

GP regression estimates the probability distribution of function values on unevaluated points. For each prediction location $\mathbf{x}^*$, mean $\mu(\mathbf{x}^*)$ gives the best estimate of the function value, and variance $\sigma^2(\mathbf{x}^*)$ models the uncertainty at the point. Acquisition functions utilizes the computed distribution to guide the search for the optimal function value.

2.2. Acquisition Functions

Acquisition functions take the mean and variance at each unevaluated point as input and compute a value indicating how favorable it is to sample at this point. It trades off between exploitation and exploration:

- Exploitation: looking for locations that minimize the posterior mean $\mu(\mathbf{x})$.
- Exploration: looking for locations that maximize posterior variance $\sigma^2(\mathbf{x})$

Given the data $C = [(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_t, y_t)]$ observed, the next point $\mathbf{x}_{t+1}$ is chosen by the ranking the value returned by acquisition function at candidate points.

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} a(\mathbf{x}|C) \tag{4}$$

Acquisition functions is defined as the expected utility u at the unevaluated location $\mathbf{x}$:

$$\begin{aligned} a(\mathbf{x}|C) &= \mathbb{E}(u(\mathbf{x}, y)|\mathbf{x}, C) \\ &= \int u(\mathbf{x}, y)\, p(y|\mathbf{x}, C)\, dy \end{aligned} \tag{5}$$

The probability $p(y|\mathbf{x}, C)$ here is gathered from posterior distribution $\mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$ calculated by GP regression.

Probability Improvement (PI), Expected Improvement (EI) and Entropy Search employ different utility functions. Other acquisition functions such as Upper confidence bound(UCB) directly invoke the mean and variance instead.

2.2.1. Probability Improvement

PI computes the likelihood that the function at unevaluated locations $\mathbf{x}$ will return a value lower than the current minimum $\tilde{y}$. Utility function of PI is defined as:

$$u(\mathbf{x}) = \begin{cases} 0, \text{if}(\mathbf{x}) > \tilde{y} \\ 1, \text{if}(\mathbf{x}) \leq \tilde{y} \end{cases} \tag{6}$$

We can understand utility function as a reward, when $f(\mathbf{x}) \leq \tilde{y}$, a certain amount of value is rewarded, here the reward is 1. According to the utility function above, the expected utility x can be written as the normal commutative density function of $\frac{\tilde{y}-\mu(x)}{\sigma(x)}$:

$$\begin{aligned} a_{PI}(\mathbf{x}) &= \mathbb{E}[u(\mathbf{x})|C] \\ &= \int_{-\infty}^{\tilde{y}} \mathcal{N}_{f(\mathbf{x})}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))\mathrm{d}\, f(\mathbf{x}) \\ &= \Phi\left(\frac{\tilde{y}-\mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) \mathrm{PI}(\mathbf{x}) \end{aligned} \tag{7}$$

PI only cares whether $f(\mathbf{x})$ is greater than $\tilde{y}$, but does not count the quantity of improvement. This will result PI very likely to pick points near to previously sampled locations. As the searching trajectory reach local minimum, it will be stuck here and hardly jump out. Therefore, PI only cares about exploitation.

2.2.2. Expected Improvement

EI leverages better between exploration and exploitation. The amount of improvement with respect to the recent global optima $\tilde{y} - f(\mathbf{x})$ is taken into account. Utility function of EI is defined as:

$$u(\mathbf{x}) = \max(0, \tilde{y} - f(\mathbf{x})) \tag{8}$$

Therefore, the expression of expected utility can be derived:

$$\begin{aligned} a_{EI}(\mathbf{x}) &= \mathbb{E}[u(\mathbf{x})|C] \\ &= \int_{-\infty}^{\infty} \max(0, \tilde{y} - f(\mathbf{x}))\, \mathcal{N}_{f(\mathbf{x})}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))\mathrm{d}\, f(\mathbf{x}) \\ &= \int_{-\infty}^{\tilde{y}} (\tilde{y} - f(\mathbf{x}))\, \mathcal{N}_{f(\mathbf{x})}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))\mathrm{d}\, f(\mathbf{x}) \\ &= (\tilde{y} - \mu(\mathbf{x}))\Phi\left(\frac{\tilde{y}-\mu(x)}{\sigma(x)}\right) + \sigma(\mathbf{x})\phi\left(\frac{\tilde{y}-\mu(x)}{\sigma(x)}\right) \end{aligned} \tag{9}$$

where $\phi(\cdot)$ is the probability density function. In order to get higher value, at the left side of equation, we want to minimize $\mu(\mathbf{x})$; and at the right side, we want to maximize $\sigma(\mathbf{x})$. A basic equation based trade off between exploitation and exploration are achieved here.

The trade off between exploration and exploitation can be adjusted by tunning a parameter ξ at the deduction part($\tilde{y} - \mu(\mathbf{x}) - \xi$). Larger ξ will favour exploration in early steps and exploitaion later does not work well experimentally[9].

2.2.3. Modified Probability Improvement

If evaluations are noise corrupted $y_i|f_i \sim \mathcal{N}(f_i, \sigma_y^2)$, the current loss optimum $\tilde{y}$ is not a reliable sample. Instead of using the optimum directly, we consider to use the posterior distribution $\mathcal{N}(\mu(\tilde{\mathbf{x}}), \sigma^2(\tilde{\mathbf{x}}))$ at the current optimum. PI can be modified under the noise corrupted conditions in order to increase the robustness at sampling process. Let

- $k(\mathbf{x}, \mathbf{x})$ denotes the posterior variance $\sigma^2(\mathbf{x})$ of an unevaluated point $\mathbf{x}$ computed from gaussian process.
- $k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}})$ denotes the posterior variance $\sigma^2(\tilde{\mathbf{x}})$ of loss optimum .
- $k(\mathbf{x}, \tilde{\mathbf{x}})$ denotes the posterior covariance between unevaluated point and loss optimum.

According to the rule of variance deduction of two dependent random variables:

$$\mathrm{Var}[X - Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] - 2 \times \mathrm{Cov}[X, Y] \tag{10}$$

Distribution of $f(\mathbf{x}) - f(\tilde{\mathbf{x}})$ can be derived:

$$\mathcal{N}_{f(\mathbf{x})-f(\tilde{\mathbf{x}})}(\mu(\mathbf{x}) - \mu(\tilde{\mathbf{x}}), k(\mathbf{x}, \mathbf{x}) + k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) - 2k(\mathbf{x}, \tilde{\mathbf{x}})) \tag{11}$$

Utility function of Modified Probability Improvement (MPI) is rewritten as:

$$u(\mathbf{x}) = \begin{cases} 0, \text{if} f(\mathbf{x}) - f(\tilde{\mathbf{x}}) > 0 \\ 1, \text{if} f(\mathbf{x}) - f(\tilde{\mathbf{x}}) \leq 0 \end{cases} \tag{12}$$

Since the utility function only counts the improvement when $f(\mathbf{x}) - f(\tilde{\mathbf{x}}) \leq 0$, PI can be written as the probability of $f(\mathbf{x}) - f(\tilde{\mathbf{x}}) \leq 0$. As if $X \sim \mathcal{N}(\mu, \sigma^2)$, then $\mathbb{P}(X < x) = \Phi(\frac{x-\mu}{\sigma})$. Cumulative density function for MPI can be derived:

$$\begin{aligned} a_{\mathrm{MPI}}(x) &= \mathbb{P}(f(\mathbf{x}) - f(\tilde{\mathbf{x}}) \leq 0) \\ &= \Phi\left(\frac{0 - (\mu(\mathbf{x}) - \mu(\tilde{\mathbf{x}}))}{\sqrt{k(\mathbf{x},\mathbf{x}) + k(\tilde{\mathbf{x}},\tilde{\mathbf{x}}) - 2k(\mathbf{x},\tilde{\mathbf{x}})}}\right) \\ &= \Phi\left(\frac{\mu(\tilde{\mathbf{x}}) - \mu(\mathbf{x})}{\sqrt{k(\mathbf{x},\mathbf{x}) + k(\tilde{\mathbf{x}},\tilde{\mathbf{x}}) - 2k(\mathbf{x},\tilde{\mathbf{x}})}}\right) \end{aligned} \tag{13}$$

2.2.4. Modified Expected Improvement

Same as MPI, $\tilde{y}$ is replaced by the posterior distribution at $\tilde{\mathbf{x}}$ in Modified Expected Improvement (MEI). Expression of utility function is:

$$u(\mathbf{x}) = \max(0, f(\tilde{\mathbf{x}}) - f(\mathbf{x})) \tag{14}$$

A lemma of expectation on max function applied on normal distributed random variables [10] can be directly employed to get the expression of MEI:

$$\begin{aligned} &\text{If } s \sim \mathcal{N}(\mu, \sigma^2) \\ &\mathbb{E}[\max(0, s)] \quad = \int_0^\infty s\mathcal{N}(S; \mu, \sigma^2)\, ds \\ &\qquad\qquad\quad = \Phi\left(\frac{\mu}{\sigma}\right)\mu + \phi\left(\frac{\mu}{\sigma}\right)\sigma \end{aligned} \tag{15}$$

We already know the mean and variance of normal distribution of $f(\mathbf{x}) - f(\tilde{\mathbf{x}})$ from equation 11. Mean of $f(\tilde{\mathbf{x}}) - f(\mathbf{x})$ is $\mu(\tilde{\mathbf{x}}) - \mu(\mathbf{x})$, and the variance remines the same. Let ρ denotes $\sqrt{k(\mathbf{x}, \mathbf{x}) + k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) - 2k(\mathbf{x}, \tilde{\mathbf{x}})}$, applying Lemma from equation 15:

$$\begin{aligned} a_{\mathrm{MEI}} &= \mathbb{E}[u(x)|C] \\ &= \Phi(\tfrac{\mu(\tilde{\mathbf{x}}) - \mu(\mathbf{x})}{\rho})(\mu(\tilde{\mathbf{x}}) - \mu(\mathbf{x})) + \phi(\tfrac{\mu(\tilde{\mathbf{x}}) - \mu(\mathbf{x})}{\rho})\rho \end{aligned} \tag{16}$$

3. Experiments

Performance of modified versions of PI and EI are compared with the traditional PI and EI on 3 selected 2D benchmark functions. Variables including kernel functions, kernel parameters and position of pre-samplings are controlled to be the same for each set of experiment. We will visualise sampling position and global optima in search space, and current minimal loss at each iteration. Performance of 4 acquisition functions on each benchmark function will be discussed by sections.

3.1. Testing on Spere Function

Sphere function[11] has 1 global minima. It is bowl-shaped, convex and unimodal. Sphere function in d dimensions is:

$$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2 \tag{17}$$

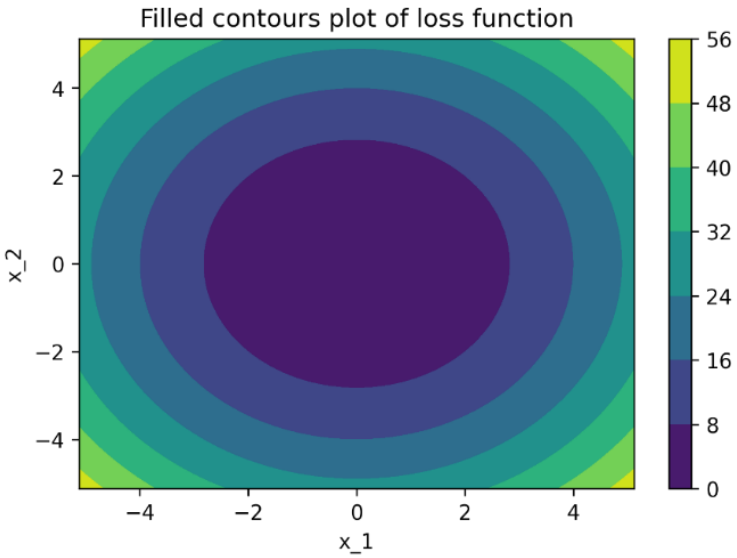
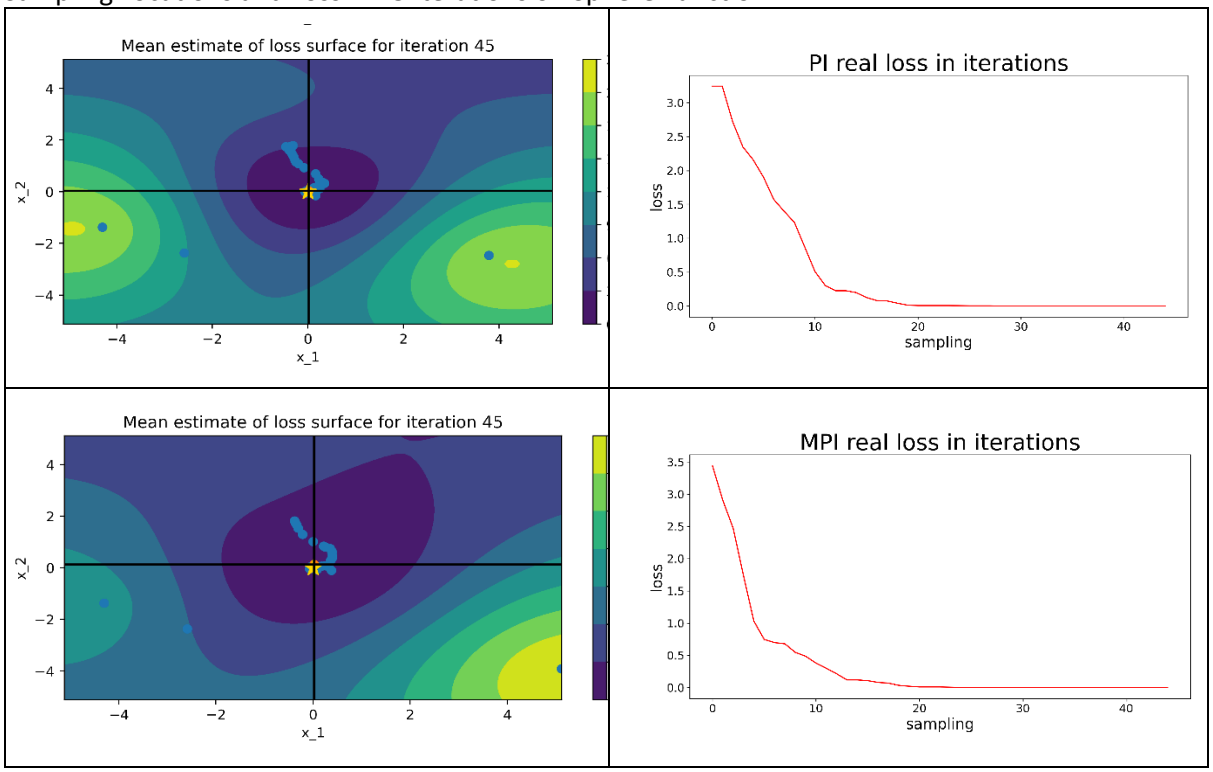


Figure 1: Coutours of Sphere Function

Figure 1 shows the contour of this function. Sampling locations and loss in 45 iterations for 4 acuqisation functions are in table 1, where star points represents the global optima and blue points are the sampling locations. We will compare acquisition functions in pair. PI performed competitively in the given environment, its sampling trajectory to the minima almost follows the gradient direction. After it reaches the global minima, it only sample the locations close to it. MPI shows similar performance, the difference is that it takes longer time (more iterations) to reach optima, and it will occationally jump out and sample locations far from current minima. EI and MEI puts more leverage at exploration side. Both of them will search globally before start to exploit near to loss optimum. Unlike EI, MEI converges faster after it had sampled locations close to global minima, it does not frequently jump out and searching locations far from current optima.

Table 1

Sampling Locations and Loss in 45 iterations on Sphere Function



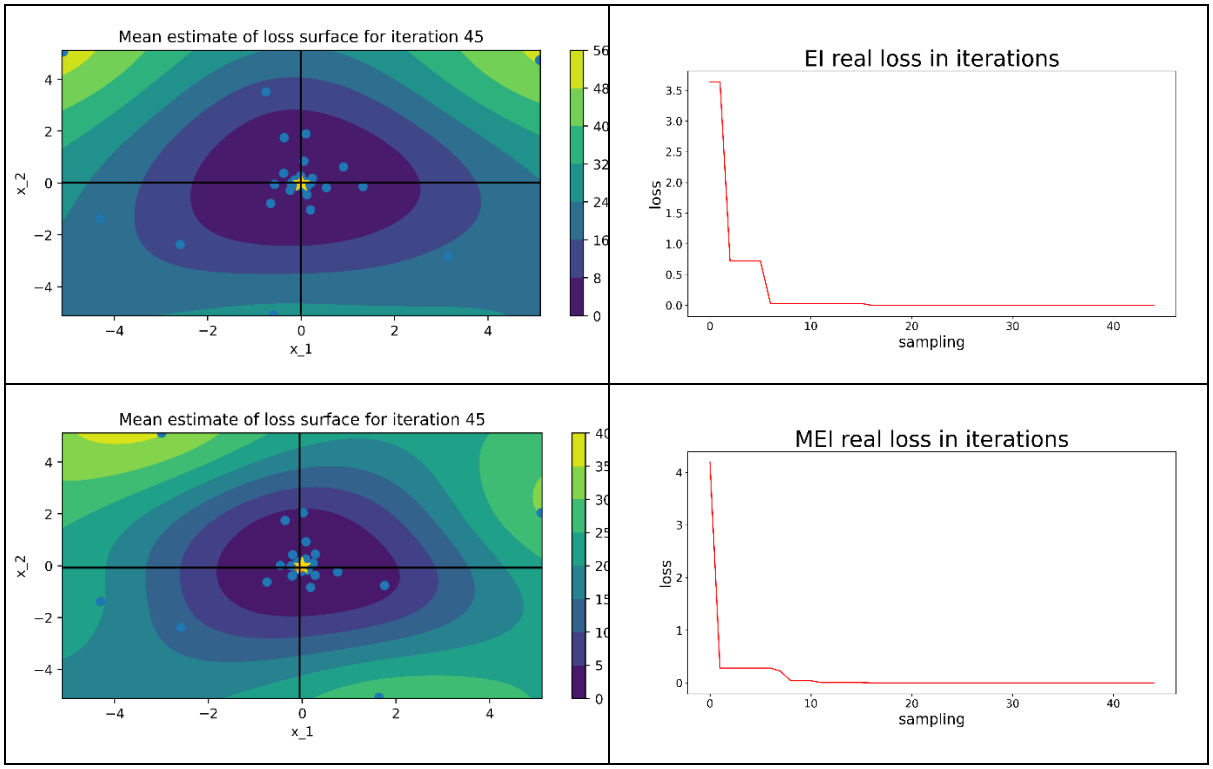


Table 2
Loss on Sphere Function Averaged in 10 Trails

Acquisition Function	Mean Loss $\pm$ Standard Deviation
PI	$2.15 \times 10^{-4} \pm 3.04 \times 10^{-5}$
MPI	$7.13 \times 10^{-5} \pm 7.07 \times 10^{-5}$
EI	$1.42 \times 10^{-4} \pm 6.61 \times 10^{-5}$
MEI	$1.81 \times 10^{-3} \pm 1.93 \times 10^{-4}$

Table 2 gives the mean and standard deviations of loss that averaged in 10 trails for each acquisation function. Each trail has 45 iterations and a unique random seed. MPI is better performed than PI, but MEI is worse than EI. Sphere function provides a simple situation where every acquisition function is able to sample near to the global optima. MPI shows a better exploitation ability in such a situation.

3.2. Testing on Six-Hump Camel Function

Six-hump camel function[11] has 6 local minima, and 2 of them are global minima. Six-hump camel function in 2 dimensions is defined as:

$$f(\mathbf{x}) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \tag{18}$$

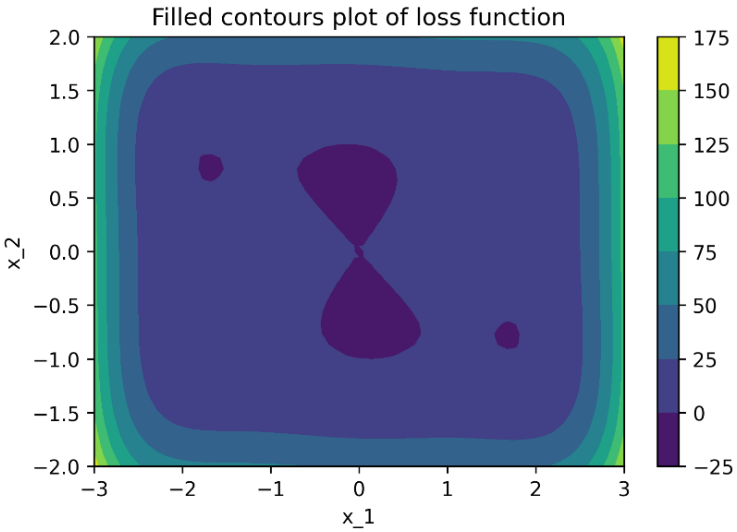
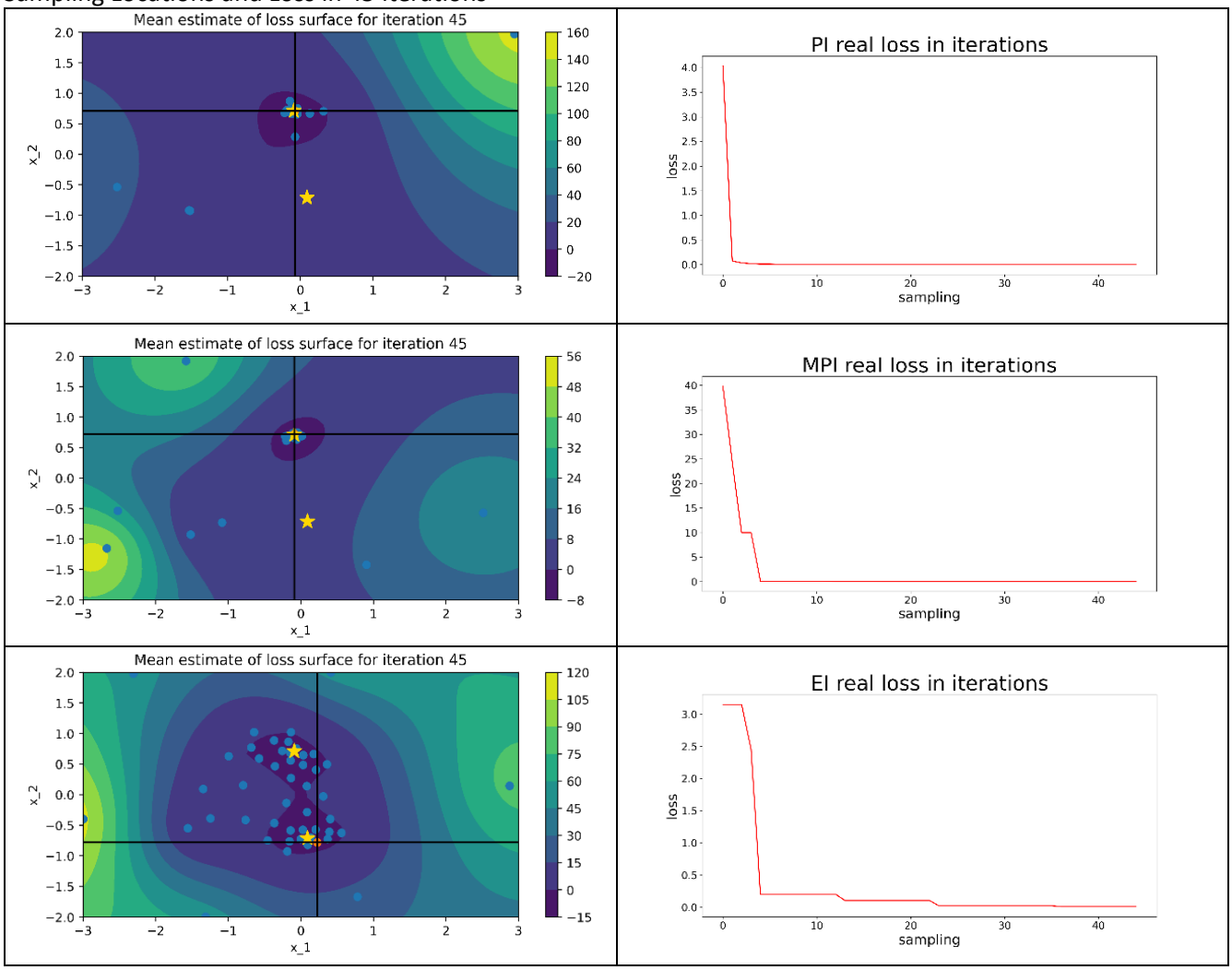


Figure 2: Coutours of Six-Hump Camel Function

Table 3

Sampling Locations and Loss in 45 iterations



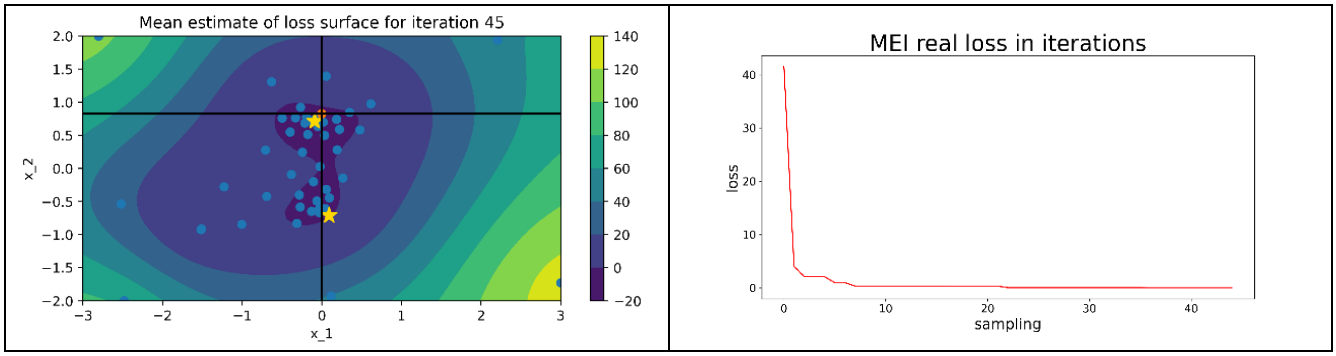


Figure 2 shows the contour of this function. According to table 3, both PI and MPI could only find 1 global optima, but PI converges faster than MPI. EI and MEI is able to sample points equally near to both global optimum. MEI converges faster than EI at finding the first global minima.

Table 4
Loss on Six-Hump Camel Function Averaged in 10 Trails

Acquisition Function	Mean Loss $\pm$ Standard Deviation
PI	$1.1 \times 10^{-4} \pm 8.62 \times 10^{-5}$
MPI	$7.19 \times 10^{-5} \pm 1.34 \times 10^{-4}$
EI	$7.41 \times 10^{-3} \pm 5.73 \times 10^{-3}$
MEI	$1.15 \times 10^{-2} \pm 9.02 \times 10^{-3}$

Table 4 gives the mean of loss and its standard deviation with 45 iterations averaged in 10 trails. Six-hump camel function is more complex compared with sphere function Through EI and MEI is able to find multiple optima, PI and MPI still achieve smaller loss by exploiting single one. Comparing acquisition functions in pairs, PI and MPI achieves similar performance, and EI still performs better than MEI.

3.3. Testing on Rastrigin Function

Rastrigin function[11] is a multimodal function, and its local minimas grid distribute through out the search space. It only has 1 global minima at the center. Rastrigin function in d dimensional is defined as:

$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{19}$$

Figure 3 shows the contour of this function. As rastrigin function is complicated, we set two set of experiments in order to test the performance of acquisitions in 45 and 100 iterations respectatively.

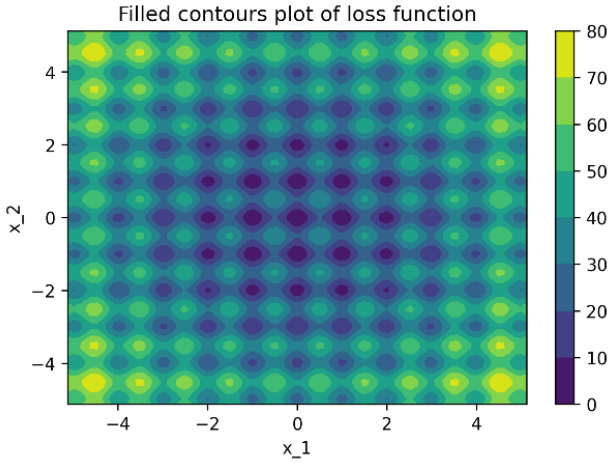


Figure 3: Coutours of Rastrigin Function

3.3.1. Testing on Rastrigin Function in 45 Iterations

Table 5 shows the sampling locations and loss of 4 acquisition functions. Both PI and MPI exploit at a local optima close to global optima, and sampling points globally at the same time. EI and MEI find 4 local optima to exploit respectively, but since they has exploited too much, both of them do not have enough iterations to explore globally. The loss of PI and MPI converges faster than EI and MEI, and MPI is the fastest in the four acquisition functions.

Table 5

Sampling Locations and Loss in 45 iterations

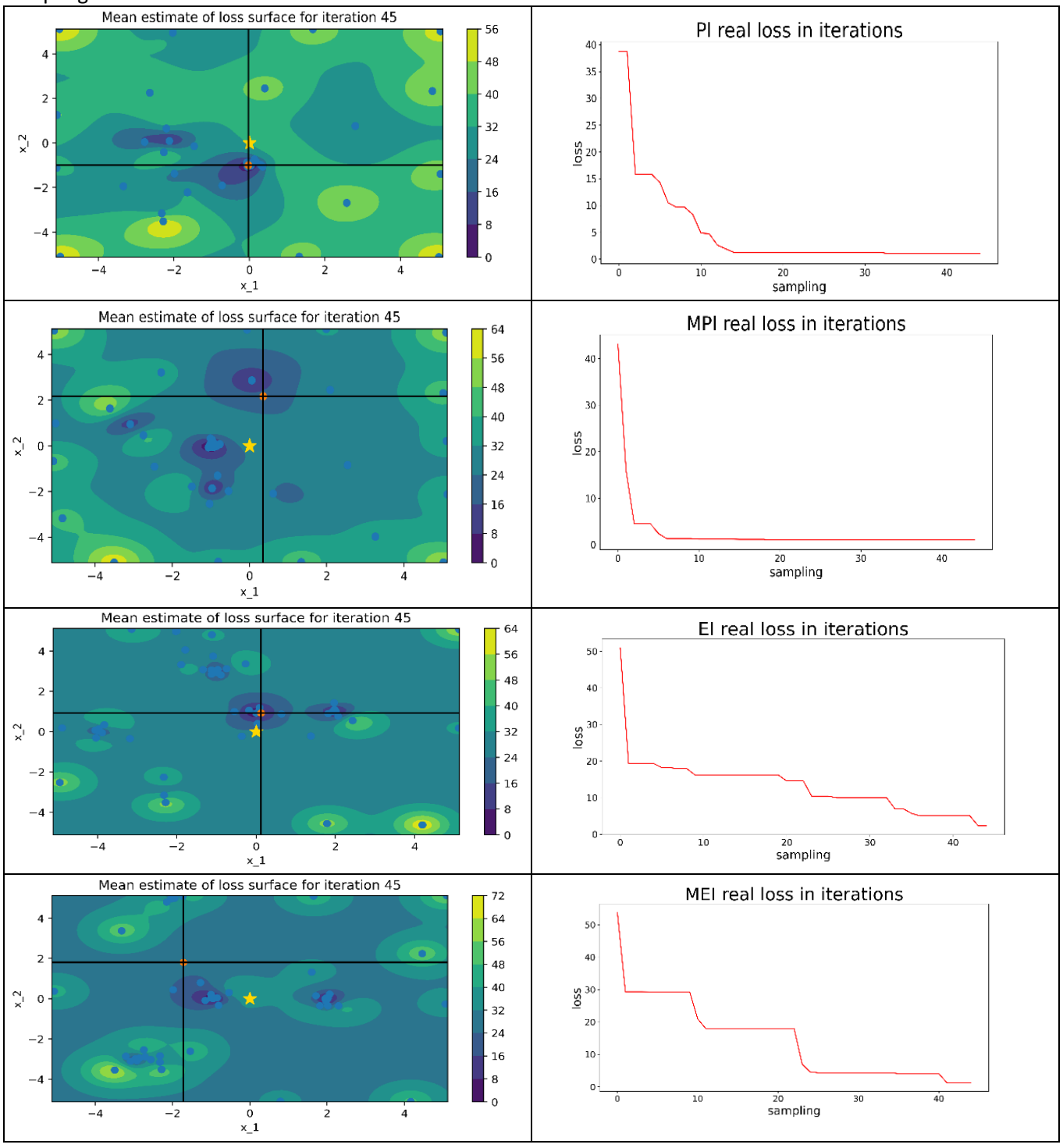


Table 6
Loss on Rastrigin Function Averaged in 10 Trails

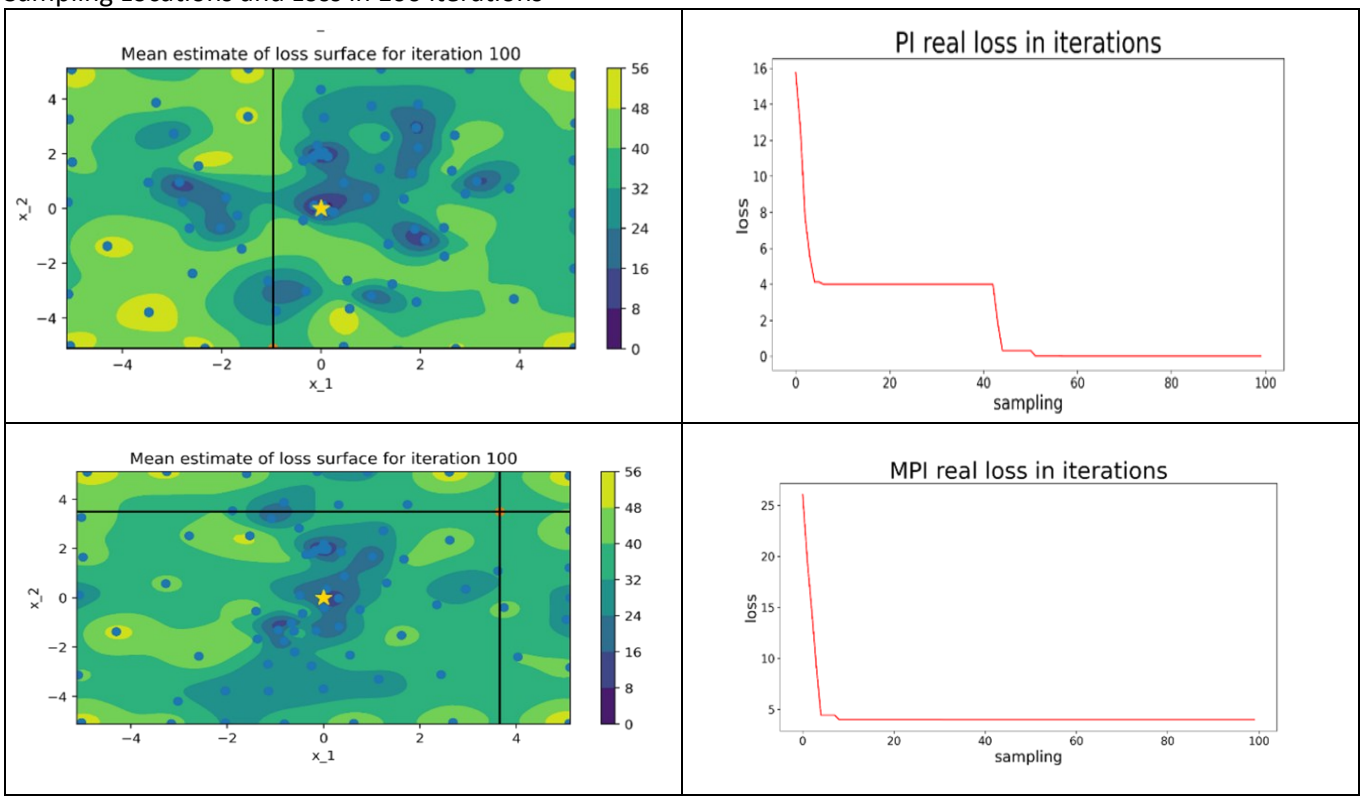
Acquisition Function	Mean Loss $\pm$ Standard Deviation
PI	9.41 ± 4.41
MPI	2.57 ± 2.17
EI	4.22 ± 1.65
MEI	2.17 ± 2.20

Table 6 gives the comparison of loss averaged in 10 trails. Here both MPI and MEI achieve a better loss than original acquisition functions. MPI is also more robust than PI as the standard deviation is smaller.

3.3.2. Testing on Rastrigin Function in 100 Iterations

Table 7 shows the sampling locations and loss of 4 acquisition functions in 100 iterations. PI and MPI can sample locations near to global optima, but only PI actually exploits at the optima. EI and MEI exploits at several good local optimas close to global optima but did not exploit at global optima. All 4 acquisation functions do explore search space with a number of sampling locations. PI spends more iterations to get a relatively small loss, both MPI and MEI converge faster than PI and EI.

Table 7
Sampling Locations and Loss in 100 iterations



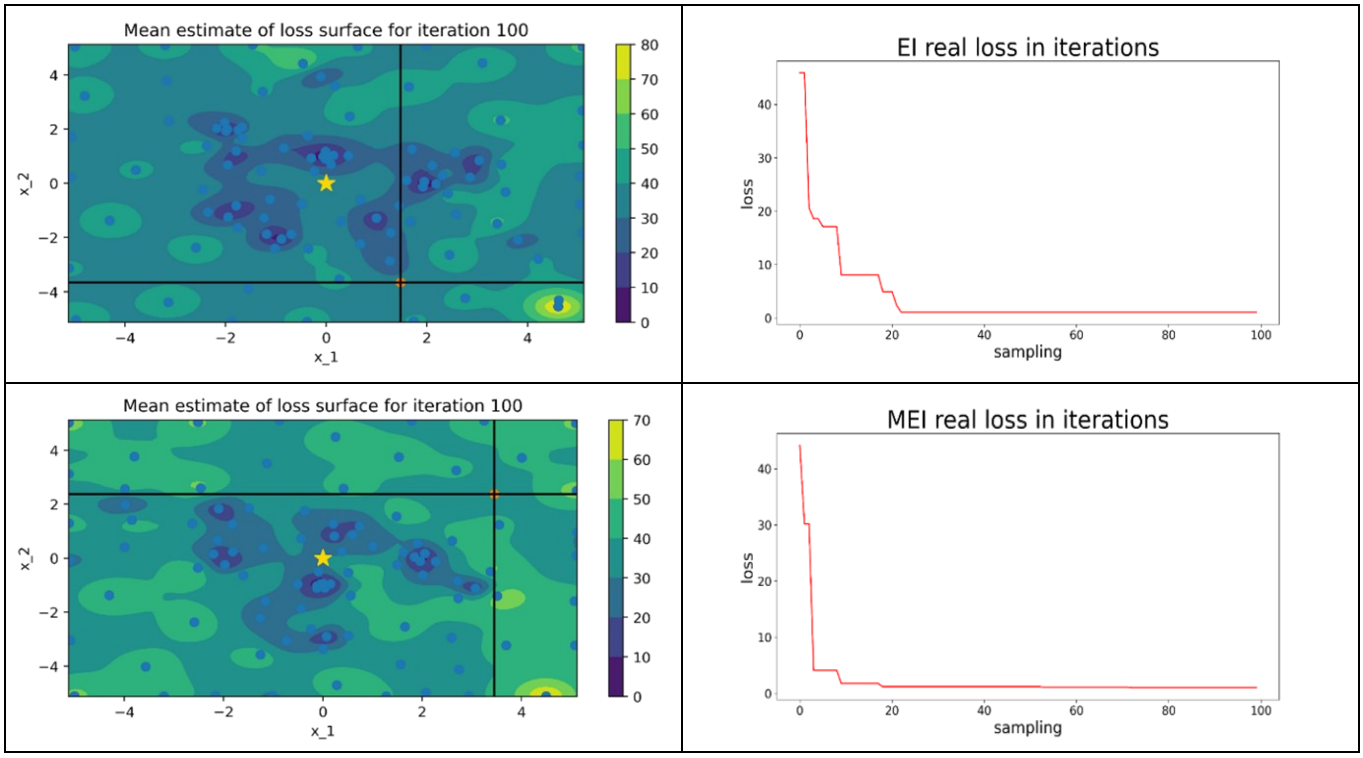


Table 8
Loss on Rastrigin Functions in 100 Iterations Averaged in 10 Trails

Acquisition Function	Mean Loss $\pm$ Standard Deviation
PI	1.18 ± 1.38
MPI	1.70 ± 1.26
EI	0.73 ± 0.45
MEI	1.14 ± 0.53

Table 8 gives the comparison of loss averaged in 10 trails. PI and EI are better than MPI and MEI. EI has the lowest loss with best robustness to get the good result. This shows in complex situation with sufficient iterations, EI shows its superiority on finding optimised result.

3.4. Experiment Summary

In simple loss functions with only a small number of minimas, MPI performs better than PI, EI and MEI. MEI is the worst one with much bigger loss and high standard deviations. In complicated loss functions with insufficient iterations, MEI and MPI is better than EI and PI. With sufficient iterations, EI is better than other acquisition functions, and MEI is the worst. In most of the conditions, loss of MPI and MEI converge faster than PI and EI.

4. Conclusions

This paper discusses the acquisition function in Bayesian Optimization in machine learning applications. Based on the traditional acquisition function, the systematic noise between the observation data and the ground truth is not fully considered. When the noise satisfies the Gaussian distribution assumption, we propose modified acquisition functions for EI and PI respectively. In addition, we believe that the following perspectives can be used as future work:

- When the number of iterations increases beyond a threshold, we should consider using a more complex hypothesis space to construct the prediction of unknown points, such as Gaussian mixture distribution or depth neural network with complex structure.

- When calculating the collection function of a point, the information of nearby points should be weighed at the same time, which can be realized by an algorithm similar to random forest, in which the nearby points will be assigned to a leaf node.
- When the data contains non Gaussian noise, acquisition functions should be constructed correspondingly to achieve better balancing the exploration and exploitation, so as to improve the optimization efficiency.

5. References

[1] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," Journal of machine learning research, vol. 13, no. 2, 2012.

[2] Y. Bengio, "Gradient-based optimization of hyperparameters," Neural computation, vol. 12, no. 8, pp. 1889–1900, 2000.

[3] M. Pelikan, D. E. Goldberg, E. Cantú-Paz, et al., "Boa: The bayesian optimization algorithm," in Proceedings of the genetic and evolutionary computation conference GECCO-99, vol. 1, pp. 525–532, Citeseer, 1999.

[4] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," Advances in neural information processing systems, vol. 25, 2012.

[5] M. Pelikan, "Hierarchical bayesian optimization algorithm," in Hierarchical Bayesian optimization algorithm, pp. 105–129, Springer, 2005.

[6] K. Swersky, J. Snoek, and R. P. Adams, "Multi-task bayesian optimization," 2013.

[7] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, "Scalable bayesian optimization using deep neural networks," in International conference on machine learning, pp. 2171–2180, PMLR, 2015.

[8] C. E. Rasmussen and C. K. I. Williams, Gaussian process for Machine Learning. The MIT Press, 2005.

[9] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," arXiv preprint arXiv:1012.2599, 2010.

[10] S. Nadarajah and S. Kotz, "Exact distribution of the max/min of two gaussian random variables," IEEE Transactions on very large scale integration (VLSI) systems, vol. 16, no. 2, pp. 210–212, 2008.

[11] M. Molga and C. Smutnicki, "Test functions for optimization needs." https://robertmarks.org/Classes/ENGR5358/Papers/functions.pdf year=2005.