Distill ABOUT PRIZE SUBMIT

Exploring Bayesian Optimization

Breaking Bayesian Optimization into small, sizeable chunks.

AUTHORS

Apoorv Agnihotri

Nipun Batra

AFFILIATIONS

Indian Insitute of Technology Gandhinagar

Indian Insitute of Technology Gandhinagar

PUBLISHED

May 5, 2020

DOI

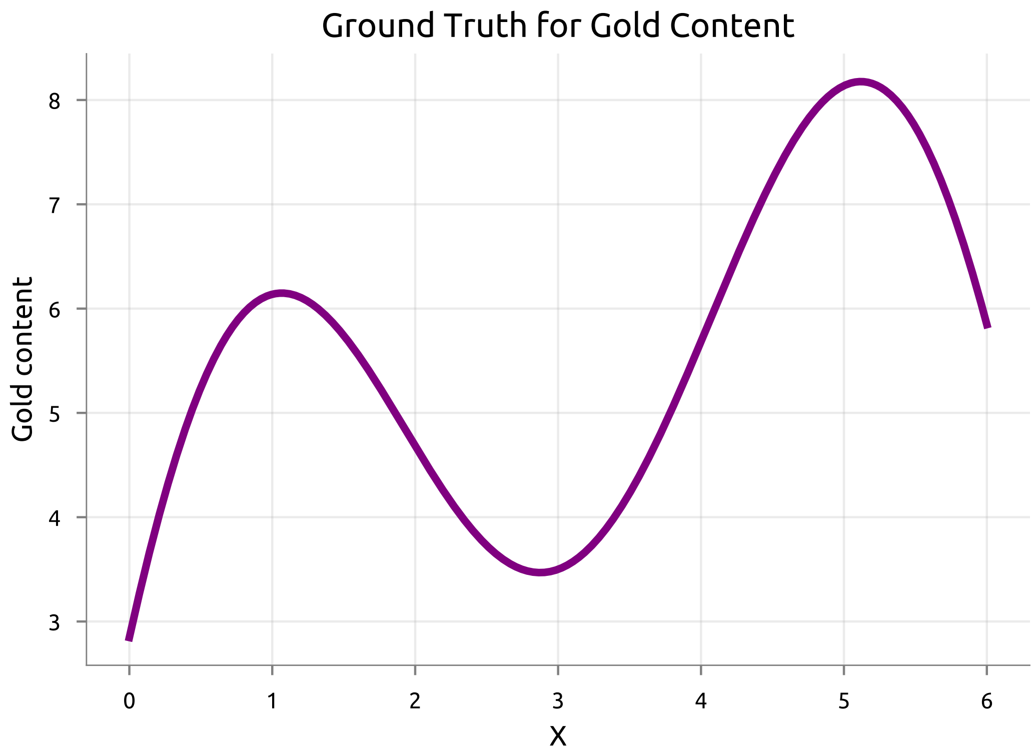
10.23915/distill.00026

Many modern machine learning algorithms have a large number of hyperparameters. To effectively use these algorithms, we need to pick good hyperparameter values. In this article, we talk about Bayesian Optimization, a suite of techniques often used to tune hyperparameters. More generally, Bayesian Optimization can be used to optimize any black-box function.

Mining Gold!

Let us start with the example of gold mining. Our goal is to mine for gold in an unknown land [1] . For now, we assume that the gold is distributed about a line. We want to find the location along this line with the maximum gold while only drilling a few times (as drilling is expensive).

Let us suppose that the gold distribution $f(x)$ looks something like the function below. It is bi-modal, with a maximum value around $x = 5$. For now, let us not worry about the X-axis or the Y-axis units.



Initially, we have no idea about the gold distribution. We can learn the gold distribution by drilling at different locations. However, this drilling is costly. Thus, we want to minimize the number of drillings required while still finding the location of maximum gold quickly.

We now discuss two common objectives for the gold mining problem.

- **Problem 1: Best Estimate of Gold Distribution (Active Learning)**
  In this problem, we want to accurately estimate the gold distribution on the new land. We can not drill at every location due to the prohibitive cost. Instead, we should drill at locations providing **high information** about the gold distribution. This problem is akin to **Active Learning** [2, 3] .

- **Problem 2: Location of Maximum Gold (Bayesian Optimization)**
  In this problem, we want to find the location of the maximum gold content. We, again, can not drill at every location. Instead, we should drill at locations showing **high promise** about the gold content. This problem is akin to **Bayesian Optimization** [4, 5] .

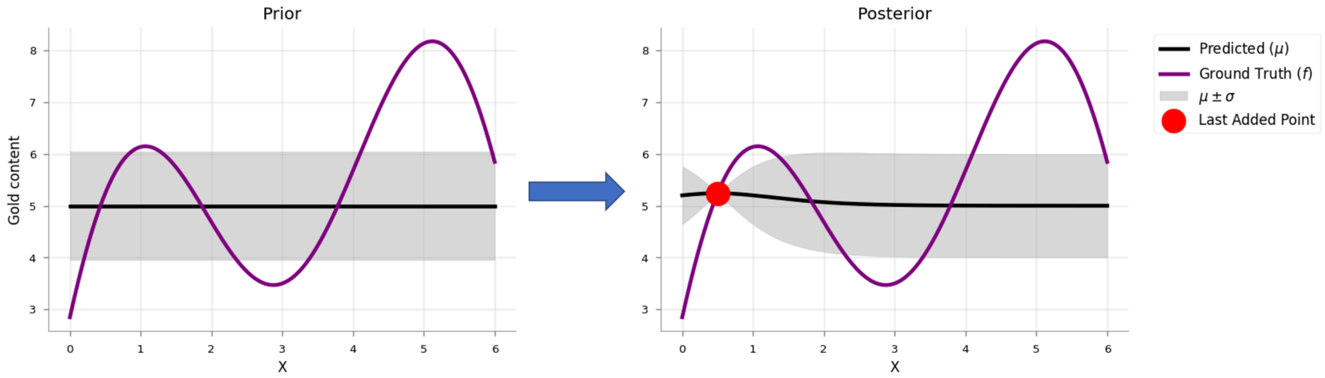
We will soon see how these two problems are related, but not the same.

Active Learning

For many machine learning problems, unlabeled data is readily available. However, labeling (or querying) is often expensive. As an example, for a speech-to-text task, the annotation requires expert(s) to label words and sentences manually. Similarly, in our gold mining problem, drilling (akin to labeling) is expensive. Active learning minimizes labeling costs while maximizing modeling accuracy. While there are various methods in active learning literature, we look at **uncertainty reduction**. This method proposes labeling the point whose model uncertainty is the highest. Often, the variance acts as a measure of uncertainty.

Since we only know the true value of our function at a few points, we need a *surrogate model* for the values our function takes elsewhere. This surrogate should be flexible enough to model the true function. Using a Gaussian Process (GP) is a common choice, both because of its flexibility and its ability to give us uncertainty estimates [2].

Our surrogate model starts with a prior of $f(x)$ — in the case of gold, we pick a prior assuming that it's smoothly distributed [3]. As we evaluate points (drilling), we get more data for our surrogate to learn from, updating it according to Bayes' rule.



Each new data point updates our surrogate model, moving it closer to the ground truth. The black line and the grey shaded region indicate the mean (μ) and uncertainty $(\mu \pm \sigma)$ in our gold distribution estimate before and after drilling.

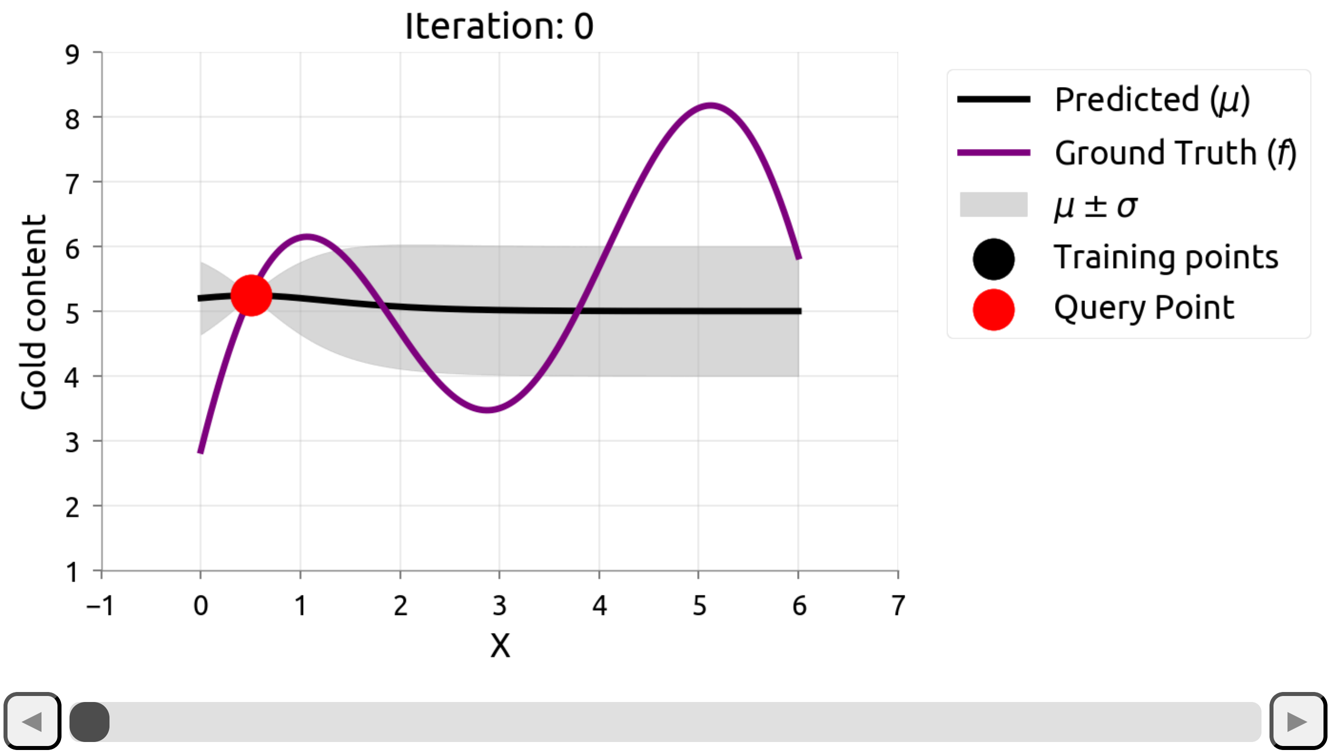
In the above example, we started with uniform uncertainty. But after our first update, the posterior is certain near $x = 0.5$ and uncertain away from it. We could just keep adding more training points and obtain a more certain estimate of $f(x)$.

However, we want to minimize the number of evaluations. Thus, we should choose the next query point "smartly" using active learning. Although there are many ways to pick smart points, we will be picking the most uncertain one.

This gives us the following procedure for Active Learning:

1. Choose and add the point with the highest uncertainty to the training set (by querying/labeling that point)
2. Train on the new training set
3. Go to #1 till convergence or budget elapsed

Let us now visualize this process and see how our posterior changes at every iteration (after each drilling).



The visualization shows that one can estimate the true distribution in a few iterations. Furthermore, the most uncertain positions are often the farthest points from the current evaluation points. At every iteration, active learning **explores** the domain to make the estimates better.

Bayesian Optimization

In the previous section, we picked points in order to determine an accurate model of the gold content. But what if our goal is simply to find the location of maximum gold content? Of course, we could do active learning to estimate the true function accurately and then find its maximum. But that seems pretty wasteful — why should we use evaluations improving our estimates of regions where the function expects low gold content when we only care about the maximum?

This is the core question in Bayesian Optimization: “Based on what we know so far, which point should we evaluate next?” Remember that evaluating each point is expensive, so we want to pick carefully! In the active learning case, we picked the most uncertain point, exploring the function. But in Bayesian Optimization, we need to balance exploring uncertain regions, which might unexpectedly have high gold content, against focusing on regions we already know have higher gold content (a kind of exploitation).

We make this decision with something called an acquisition function. Acquisition functions are heuristics for how desirable it is to evaluate a point, based on our present model [4] . We will spend much of this section going through different options for acquisition functions.

This brings us to how Bayesian Optimization works. At every step, we determine what the best point to evaluate next is according to the acquisition function by optimizing it. We then update our model and repeat this process to determine the next point to evaluate.

You may be wondering what's "Bayesian" about Bayesian Optimization if we're just optimizing these acquisition functions. Well, at every step we maintain a model describing our estimates and uncertainty at each point, which we update according to Bayes' rule[2] at each step. Our acquisition functions are based on this model, and nothing would be possible without them!

Formalizing Bayesian Optimization

Let us now formally introduce Bayesian Optimization. Our goal is to find the location ($x \in \mathbb{R}^d$) corresponding to the global maximum (or minimum) of a function $f : \mathbb{R}^d \mapsto \mathbb{R}$. We present the general constraints in Bayesian Optimization and contrast them with the constraints in our gold mining example [5].

General Constraints	Constraints in Gold Mining example
f's feasible set A is simple, e.g., box constraints.	Our domain in the gold mining problem is a single-dimensional box constraint: $0 \leq x \leq 6$.
f is continuous but lacks special structure, e.g., concavity, that would make it easy to optimize.	Our true function is neither a convex nor a concave function, resulting in local optimums.
f is derivative-free: evaluations do not give gradient information.	Our evaluation (by drilling) of the amount of gold content at a location did not give us any gradient information.
f is expensive to evaluate: the number of times we can evaluate it is severely limited.	Drilling is costly.
f may be noisy. If noise is present, we will assume it is independent and normally distributed, with common but unknown variance.	We assume noiseless measurements in our modeling (though, it is easy to incorporate normally distributed noise for GP regression).

To solve this problem, we will follow the following algorithm:

1. We first choose a surrogate model for modeling the true function f and define its **prior**.
2. Given the set of **observations** (function evaluations), use Bayes rule to obtain the **posterior**.
3. Use an acquisition function $\alpha(x)$, which is a function of the posterior, to decide the next sample point $x_t = \text{argmax}_x \alpha(x)$.

4. Add newly sampled data to the set of **observations** and goto step #2 till convergence or budget elapses.

Acquisition Functions

Acquisition functions are crucial to Bayesian Optimization, and there are a wide variety of options [6]. In the following sections, we will go through a number of options, providing intuition and examples.

PROBABILITY OF IMPROVEMENT (PI)

This acquisition function chooses the next query point as the one which has the highest *probability of improvement* over the current max $f(x^+)$. Mathematically, we write the selection of next point as follows,

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

where,

$P(\cdot)$ indicates probability

ϵ is a small positive number

And, $x^+ = \text{argmax}_{x_i \in x_{1:t}} f(x_i)$ where x_i is the location queried at i^{th} time step.

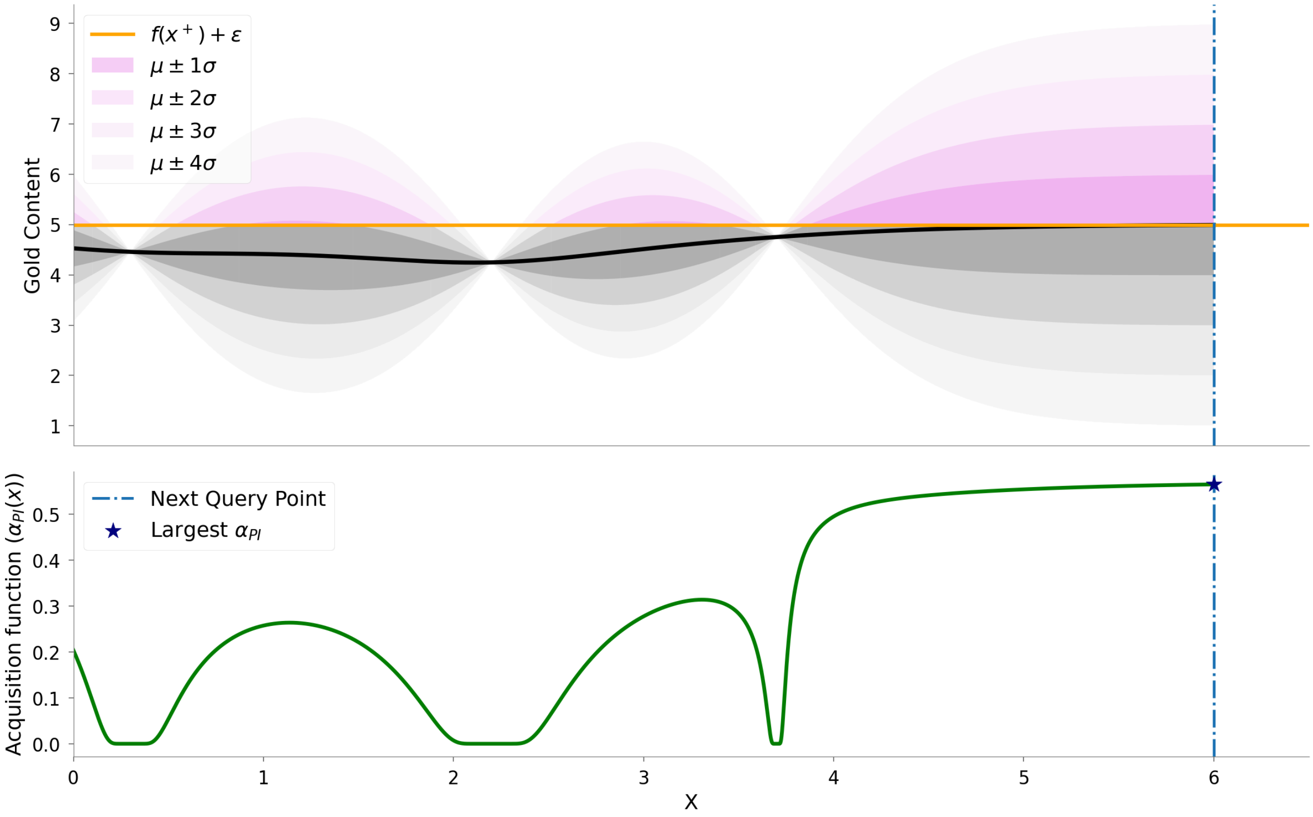
Looking closely, we are just finding the upper-tail probability (or the CDF) of the surrogate posterior. Moreover, if we are using a GP as a surrogate the expression above converts to,

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

where,

$\Phi(\cdot)$ indicates the CDF

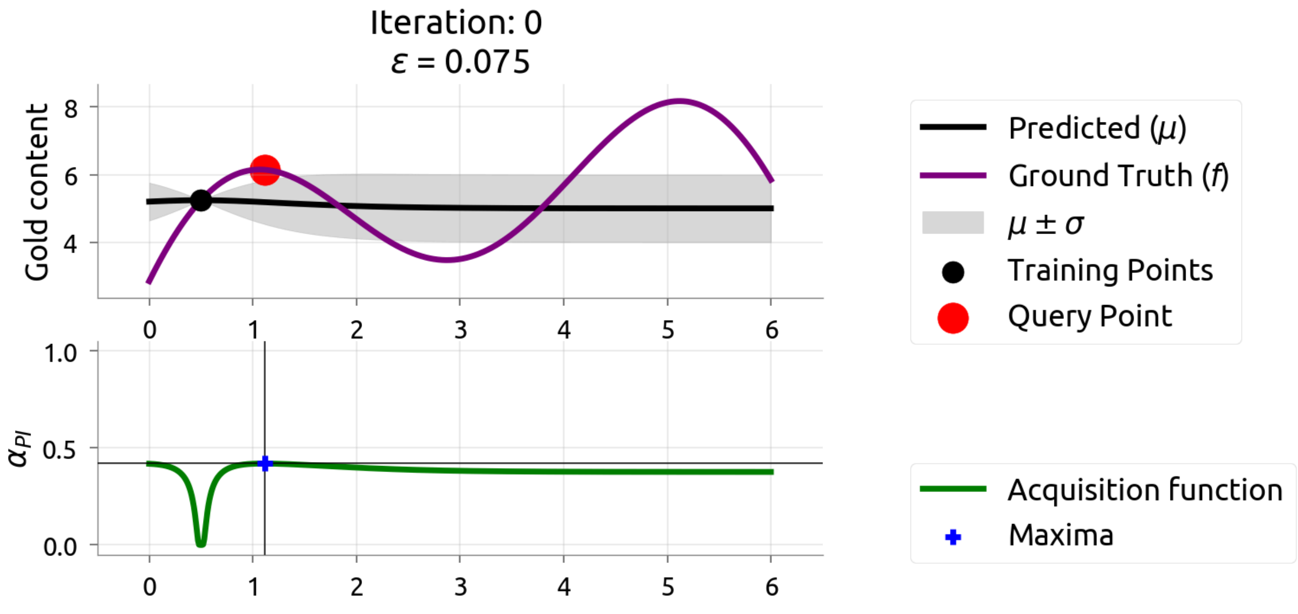
The visualization below shows the calculation of $\alpha_{PI}(x)$. The orange line represents the current max (plus an ϵ) or $f(x^+) + \epsilon$. The violet region shows the probability density at each point. The grey regions show the probability density below the current max. The “area” of the violet region at each point represents the “probability of improvement over current maximum”. The next point to evaluate via the PI criteria (shown in dashed blue line) is $x = 6$.



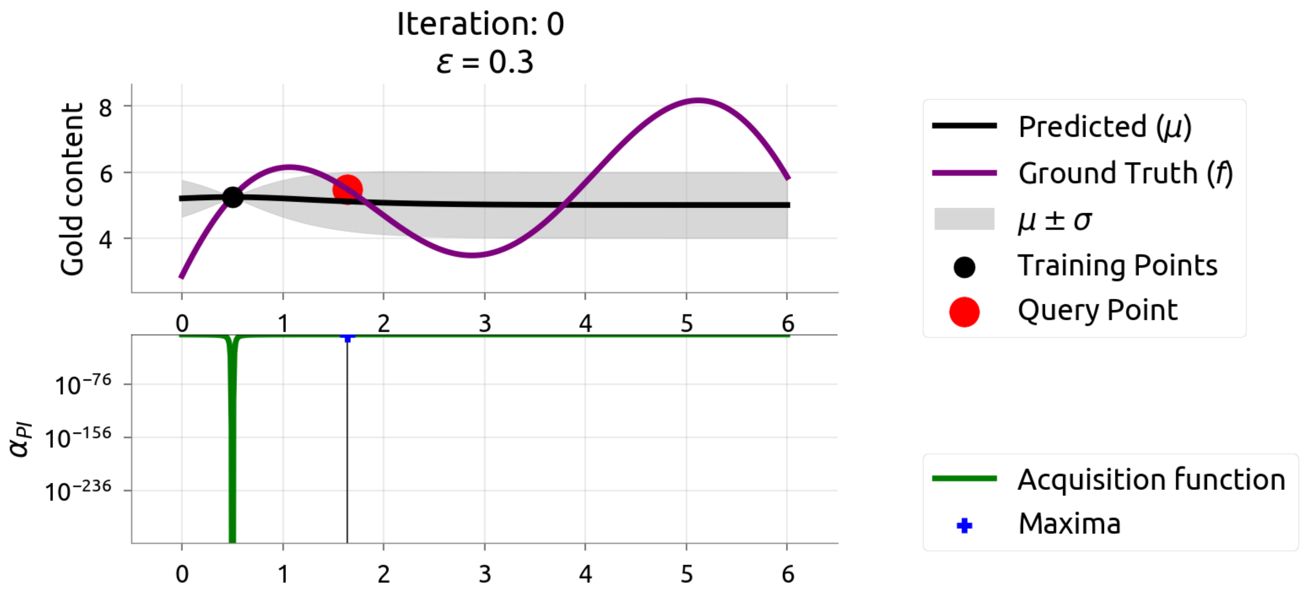
Intuition behind ϵ in PI

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

Let us now see the PI acquisition function in action. We start with $\epsilon = 0.075$.

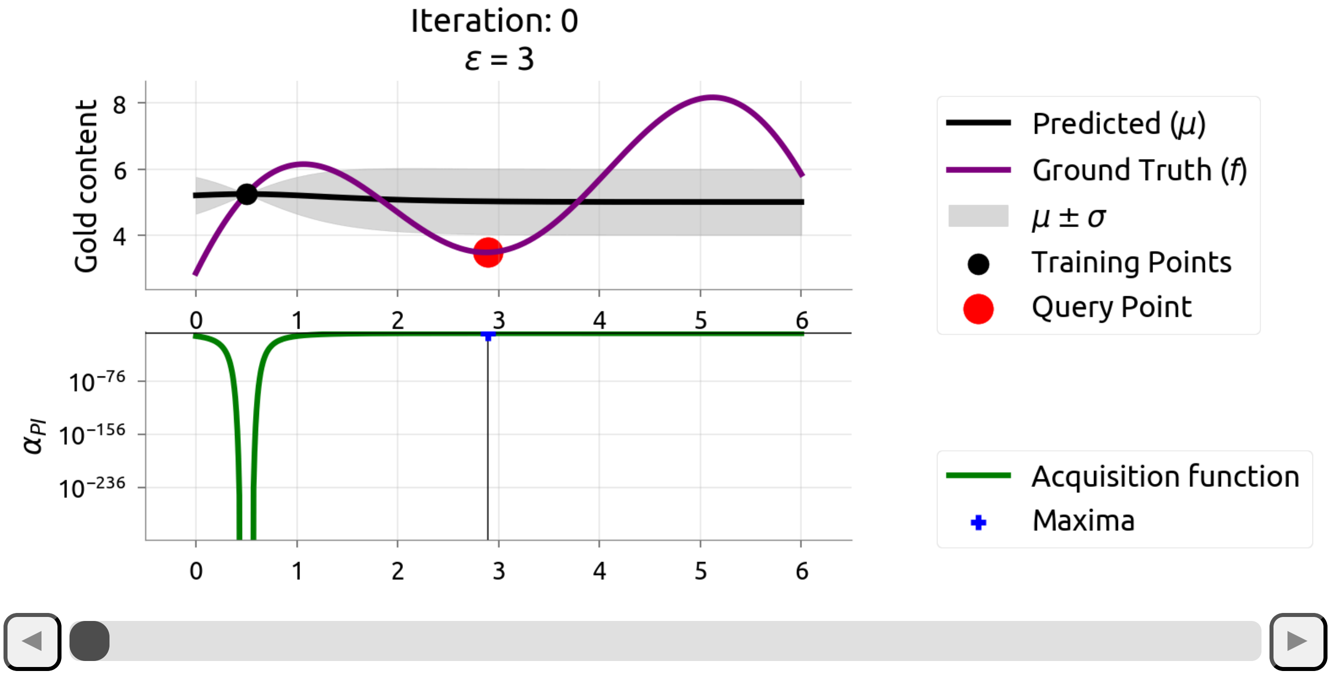


Looking at the graph above, we see that we reach the global maxima in a few iterations [7]. Our surrogate possesses a large uncertainty in $x \in [2, 4]$ in the first few iterations [8]. The acquisition function initially **exploits** regions with a high promise [9], which leads to high uncertainty in the region $x \in [2, 4]$. This observation also shows that we do not need to construct an accurate estimate of the black-box function to find its maximum.



The visualization above shows that increasing ϵ to 0.3, enables us to **explore** more. However, it seems that we are exploring more than required.

What happens if we increase ϵ a bit more?



We see that we made things worse! Our model now uses $\epsilon = 3$, and we are unable to exploit when we land near the global maximum. Moreover, with high exploration, the setting becomes similar to active learning.

Our quick experiments above help us conclude that ϵ controls the degree of exploration in the PI acquisition function.

EXPECTED IMPROVEMENT (EI)

Probability of improvement only looked at *how likely* is an improvement, but, did not consider *how much* we can improve. The next criterion, called Expected Improvement (EI), does exactly that [10]! The idea is fairly simple — choose the next query point as the one which has the highest expected improvement over the current max $f(x^+)$, where $x^+ = \text{argmax}_{x_i \in x_{1:t}} f(x_i)$ and x_i is the location queried at i^{th} time step.

In this acquisition function, $t + 1^{th}$ query point, x_{t+1}, is selected according to the following equation.

$$x_{t+1} = argmin_x \mathbb{E}\left(||h_{t+1}(x) - f(x^\star)|| \mid \mathcal{D}_t\right)$$

Where, f is the actual ground truth function, h_{t+1} is the posterior mean of the surrogate at $t + 1^{th}$ timestep, $\mathcal{D}_t$ is the training data $\{(x_i, f(x_i))\} \ \forall x \in x_{1:t}$ and $x^\star$ is the actual position where f takes the maximum value.

In essence, we are trying to select the point that minimizes the distance to the objective evaluated at the maximum. Unfortunately, we do not know the ground truth function, f. Mockus[8] proposed the following acquisition function to overcome the issue.

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

where $f(x^+)$ is the maximum value that has been encountered so far. This equation for GP surrogate is an analytical expression shown below.

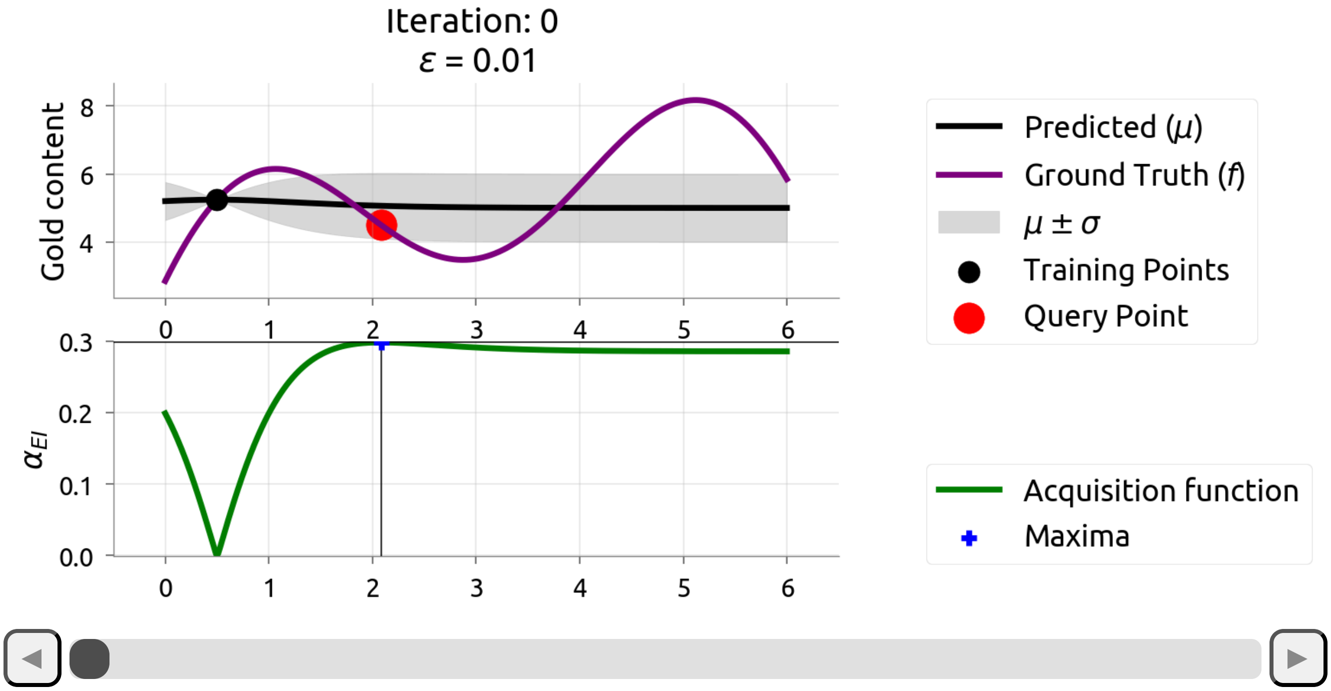
$$EI(x) = \begin{cases} (\mu_t(x) - f(x^+) - \epsilon)\Phi(Z) + \sigma_t(x)\phi(Z), & \text{if}\ \sigma_t(x) > 0 \\ 0, & \text{if}\ \sigma_t(x) = 0 \end{cases}$$

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

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

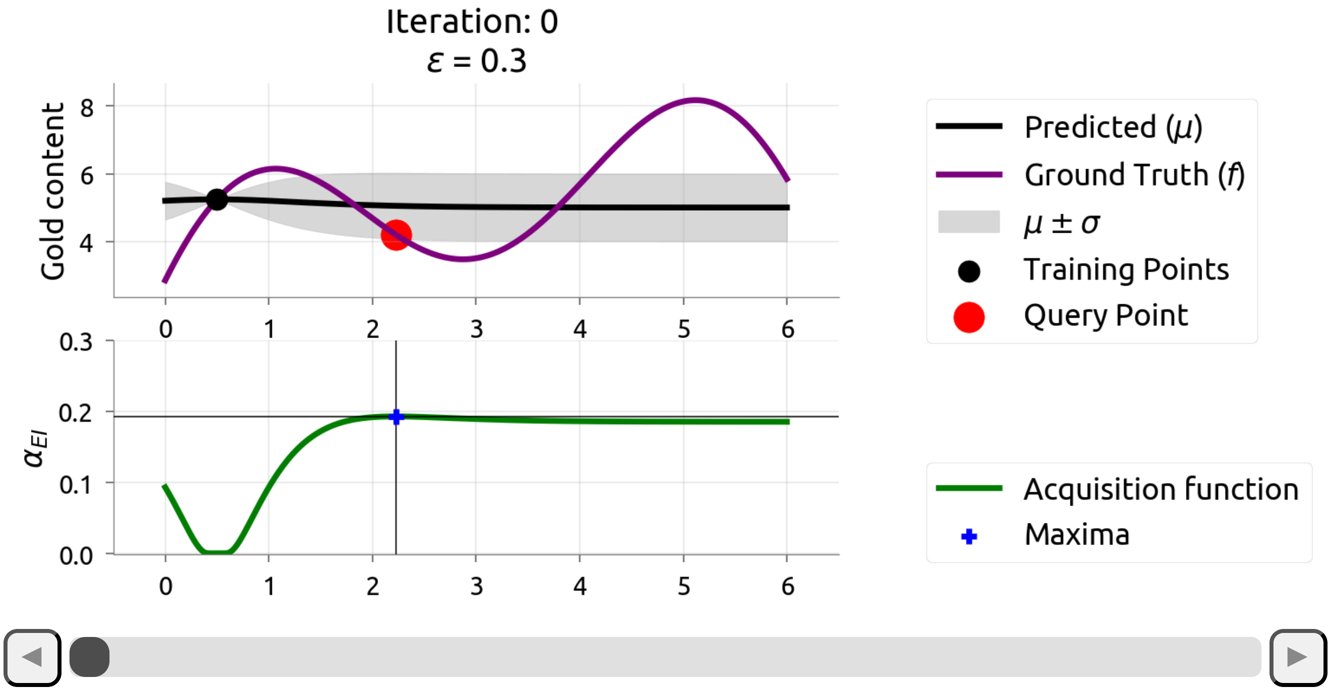
From the above expression, we can see that *Expected Improvement* will be high when: i) the expected value of $\mu_t(x) - f(x^+)$ is high, or, ii) when the uncertainty $\sigma_t(x)$ around a point is high.

Like the PI acquisition function, we can moderate the amount of exploration of the EI acquisition function by modifying ϵ.



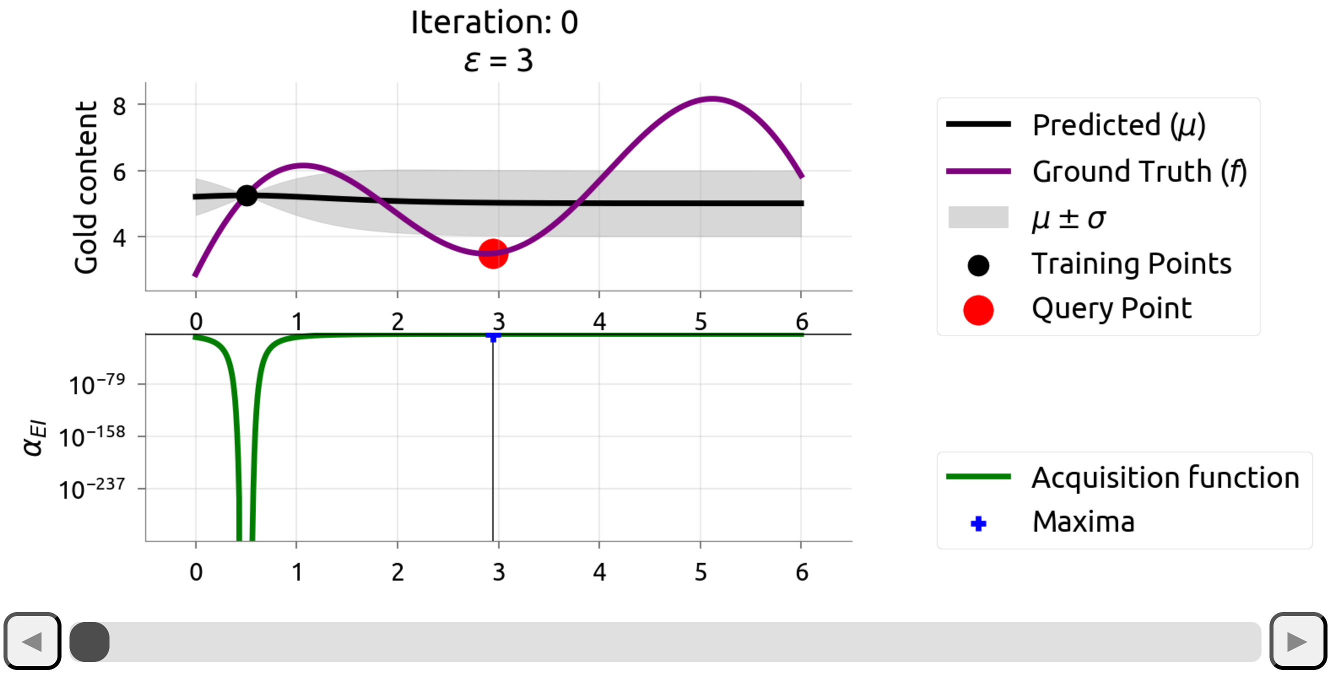
For $\epsilon = 0.01$ we come close to the global maxima in a few iterations.

We now increase ϵ to explore more.



As we expected, increasing the value to $\epsilon = 0.3$ makes the acquisition function explore more. Compared to the earlier evaluations, we see less exploitation. We see that it evaluates only two points near the global maxima.

Let us increase ϵ even more.



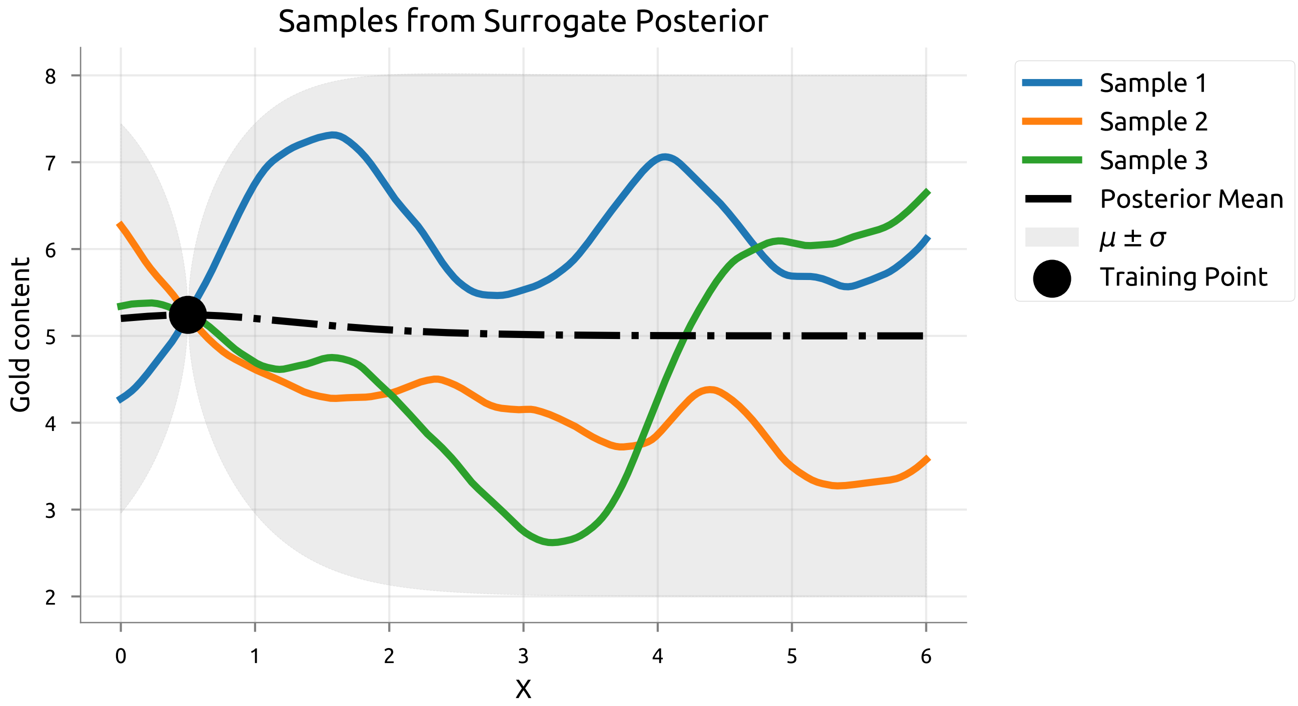
Is this better than before? It turns out a yes and a no; we explored too much at $\epsilon = 3$ and quickly reached near the global maxima. But unfortunately, we did not exploit to get more gains near the global maxima.

PI VS EI +

Thompson Sampling

Another common acquisition function is Thompson Sampling [9]. At every step, we sample a function from the surrogate's posterior and optimize it. For example, in the case of gold mining, we would sample a plausible distribution of the gold given the evidence and evaluate (drill) wherever it peaks.

Below we have an image showing three sampled functions from the learned surrogate posterior for our gold mining problem. The training data constituted the point $x = 0.5$ and the corresponding functional value.



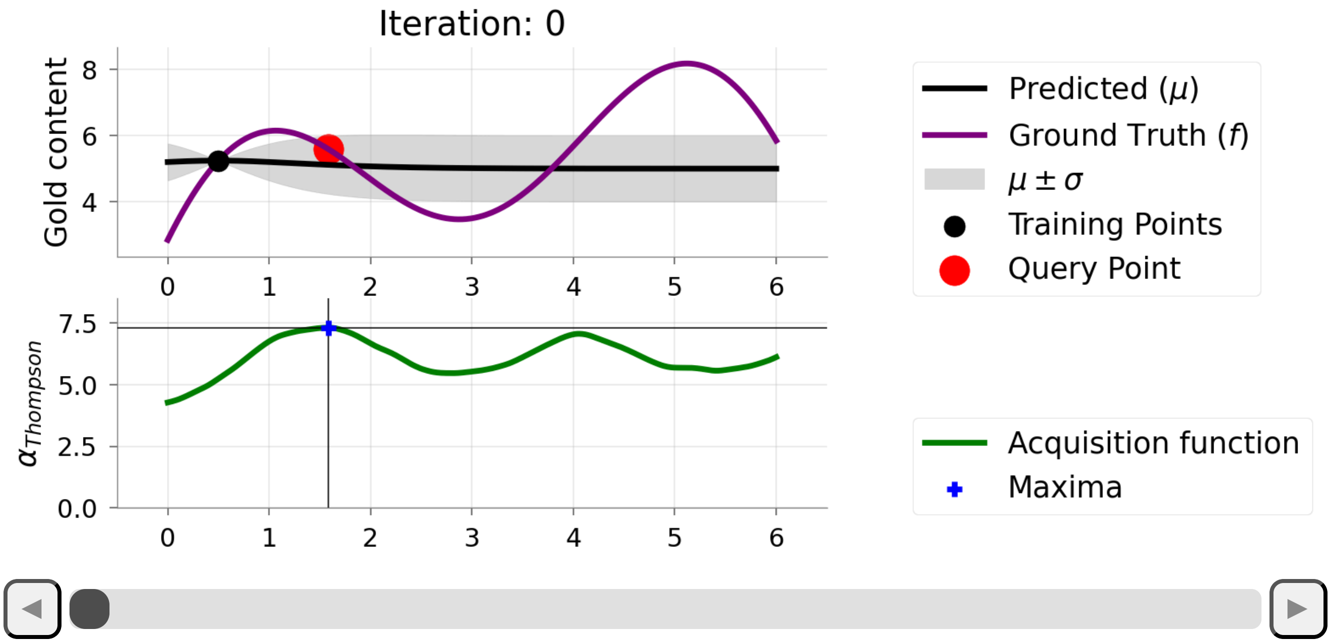
We can understand the intuition behind Thompson sampling by two observations:

- Locations with high uncertainty ($\sigma(x)$) will show a large variance in the functional values sampled from the surrogate posterior. Thus, there is a non-trivial probability that a sample can take high value in a highly uncertain region. Optimizing such samples can aid **exploration**.

  As an example, the three samples (sample #1, #2, #3) show a high variance close to $x = 6$. Optimizing sample 3 will aid in exploration by evaluating $x = 6$.

- The sampled functions must pass through the current max value, as there is no uncertainty at the evaluated locations. Thus, optimizing samples from the surrogate posterior will ensure **exploiting** behavior.

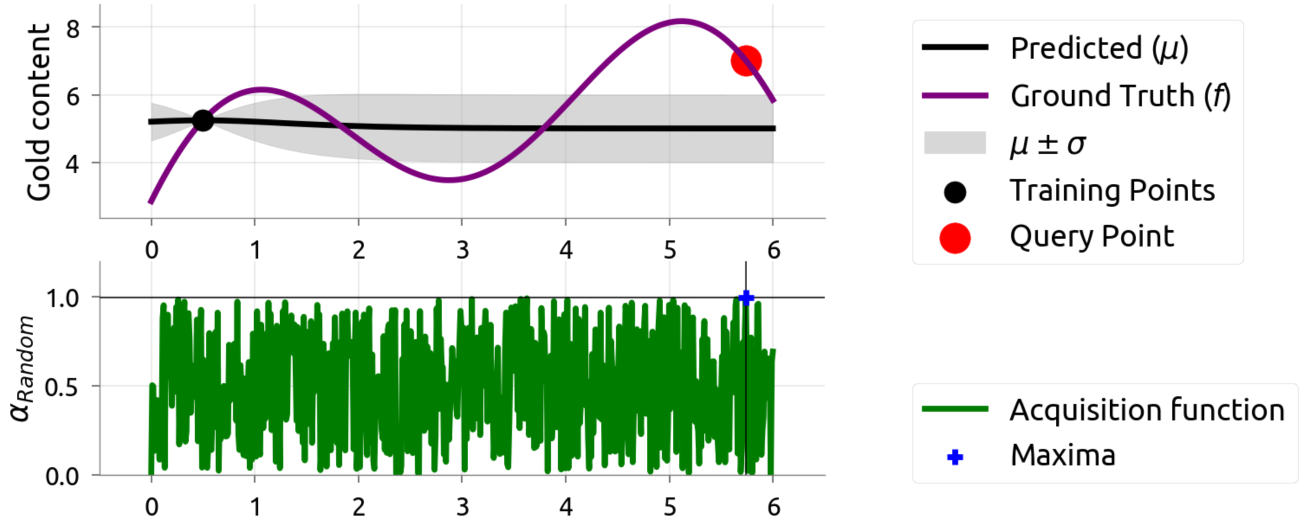
As an example of this behavior, we see that all the sampled functions above pass through the current max at $x = 0.5$. If $x = 0.5$ were close to the global maxima, then we would be able to **exploit** and choose a better maximum.



The visualization above uses Thompson sampling for optimization. Again, we can reach the global optimum in relatively few iterations.

Random

We have been using intelligent acquisition functions until now. We can create a random acquisition function by sampling x randomly.



◀ ▶

The visualization above shows that the performance of the random acquisition function is not that bad! However, if our optimization was more complex (more dimensions), then the random acquisition might perform poorly.

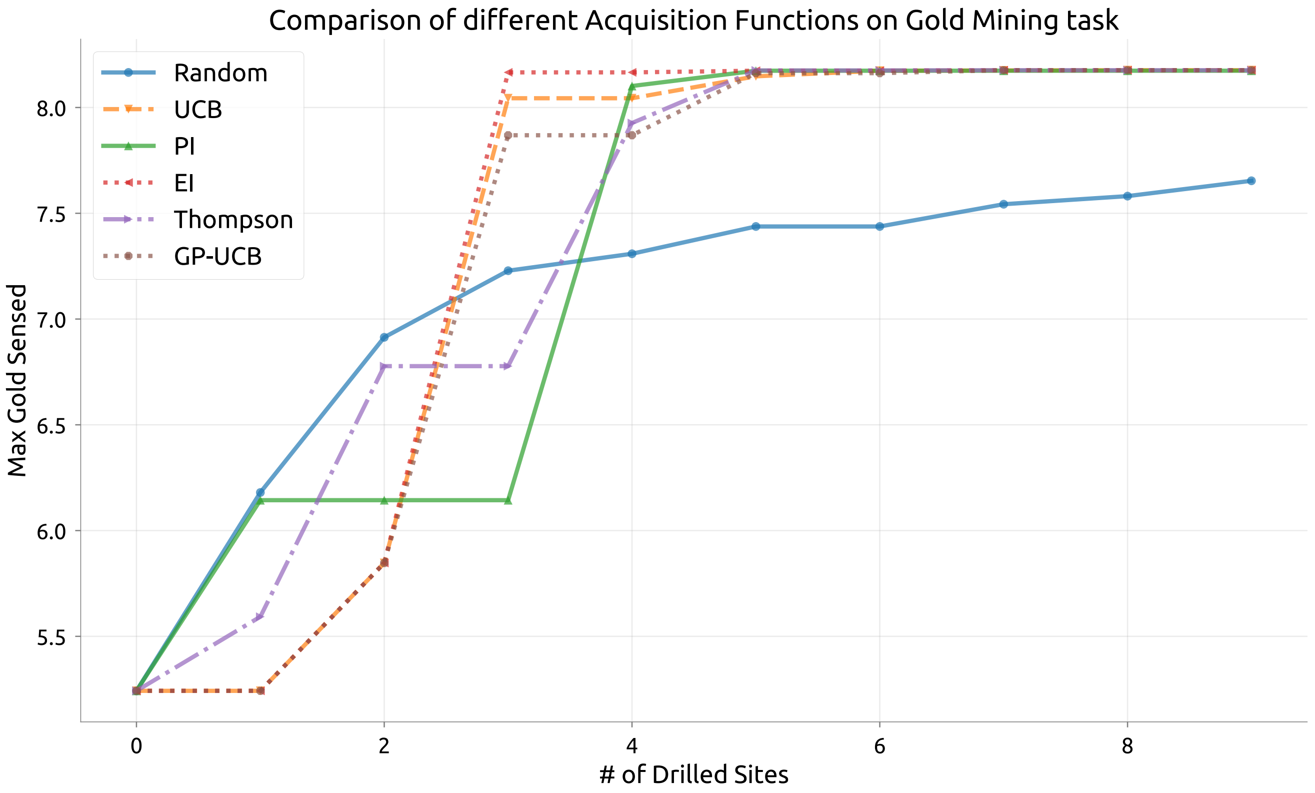
Summary of Acquisition Functions

Let us now summarize the core ideas associated with acquisition functions: i) they are heuristics for evaluating the utility of a point; ii) they are a function of the surrogate posterior; iii) they combine exploration and exploitation; and iv) they are inexpensive to evaluate.

OTHER ACQUISITION FUNCTIONS +

Comparison

We now compare the performance of different acquisition functions on the gold mining problem [14]. We have used the optimum hyperparameters for each acquisition function. We ran the random acquisition function several times with different seeds and plotted the mean gold sensed at every iteration.



The *random* strategy is initially comparable to or better than other acquisition functions [15]. However, the maximum gold sensed by *random* strategy grows slowly. In comparison, the other acquisition functions can find a good solution in a small number of iterations. In fact, most acquisition functions reach fairly close to the global maxima in as few as three iterations.

Hyperparameter Tuning

Before we talk about Bayesian optimization for hyperparameter tuning [12, 13, 14], we will quickly differentiate between hyperparameters and parameters: hyperparameters are set before learning and the parameters are learned from the data. To illustrate the difference, we take the example of Ridge regression.

$$\hat{\theta}_{ridge} = argmin_{\theta \in \mathbb{R}^p} \sum_{i=1}^{n} \left(y_i - x_i^T \theta\right)^2 + \lambda \sum_{j=1}^{p} \theta_j^2$$

In Ridge regression, the weight matrix θ is the parameter, and the regularization coefficient $\lambda \geq 0$ is the hyperparameter.
If we solve the above regression problem via gradient descent optimization, we further introduce another optimization parameter, the learning rate α.

The most common use case of Bayesian Optimization is *hyperparameter tuning*: finding the best performing hyperparameters on machine learning models.

When training a model is not expensive and time-consuming, we can do a grid search to find the optimum hyperparameters. However, grid search is not feasible if function evaluations are costly, as in the case of a large neural network that takes days to train. Further, grid search scales poorly in terms of the number of hyperparameters.

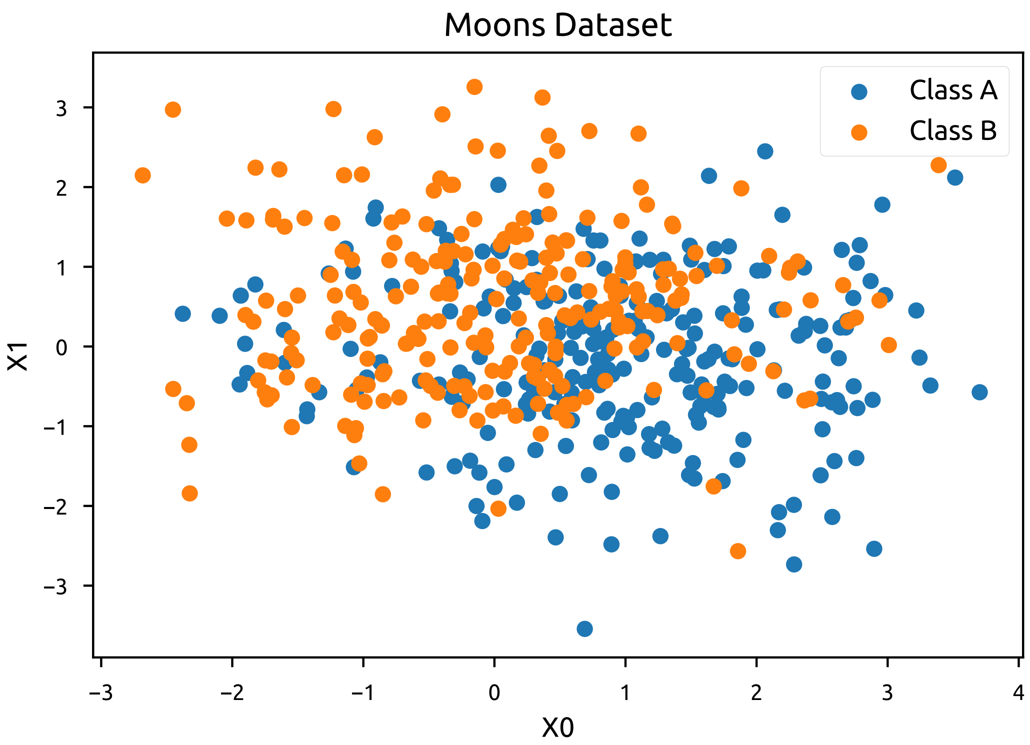
We turn to Bayesian Optimization to counter the expensive nature of evaluating our black-box function (accuracy).

Example 1 — Support Vector Machine (SVM)

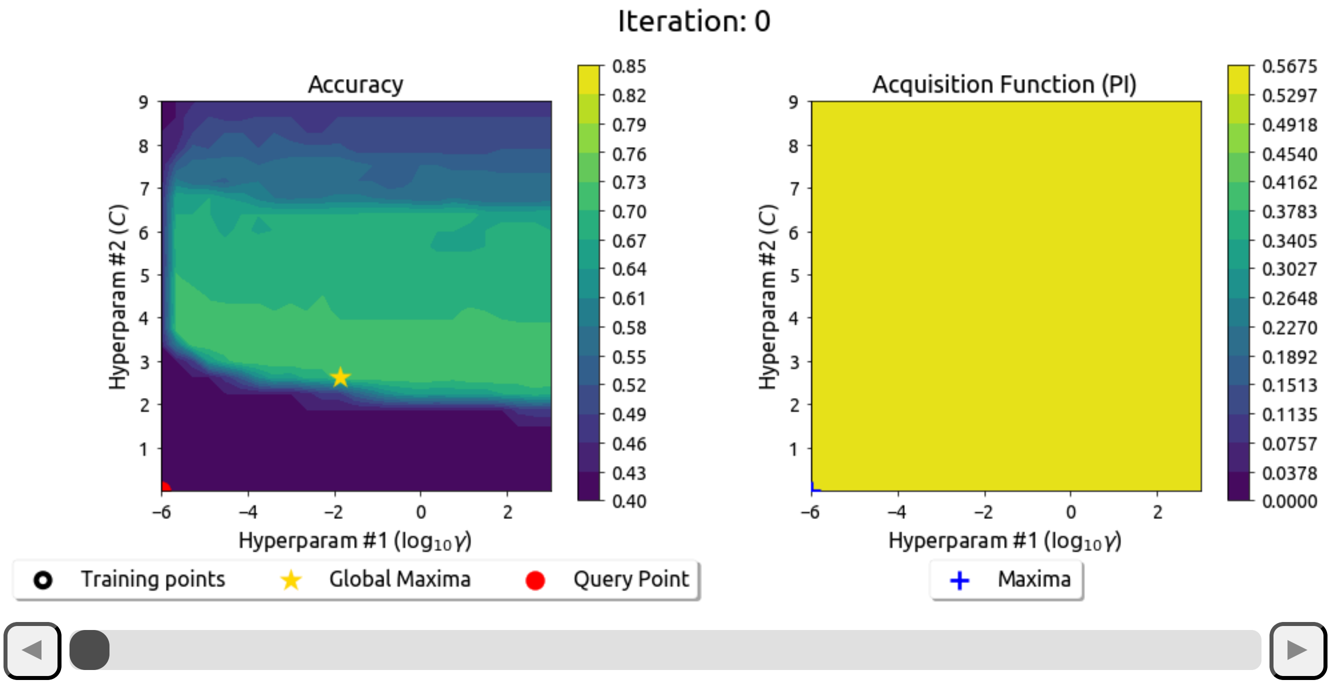
In this example, we use an SVM to classify on sklearn’s moons dataset and use Bayesian Optimization to optimize SVM hyperparameters.

- γ — modifies the behavior of the SVM’s kernel. Intuitively it is a measure of the influence of a single training example [16].
- C — modifies the slackness of the classification, the higher the C is, the more sensitive is SVM towards the noise.

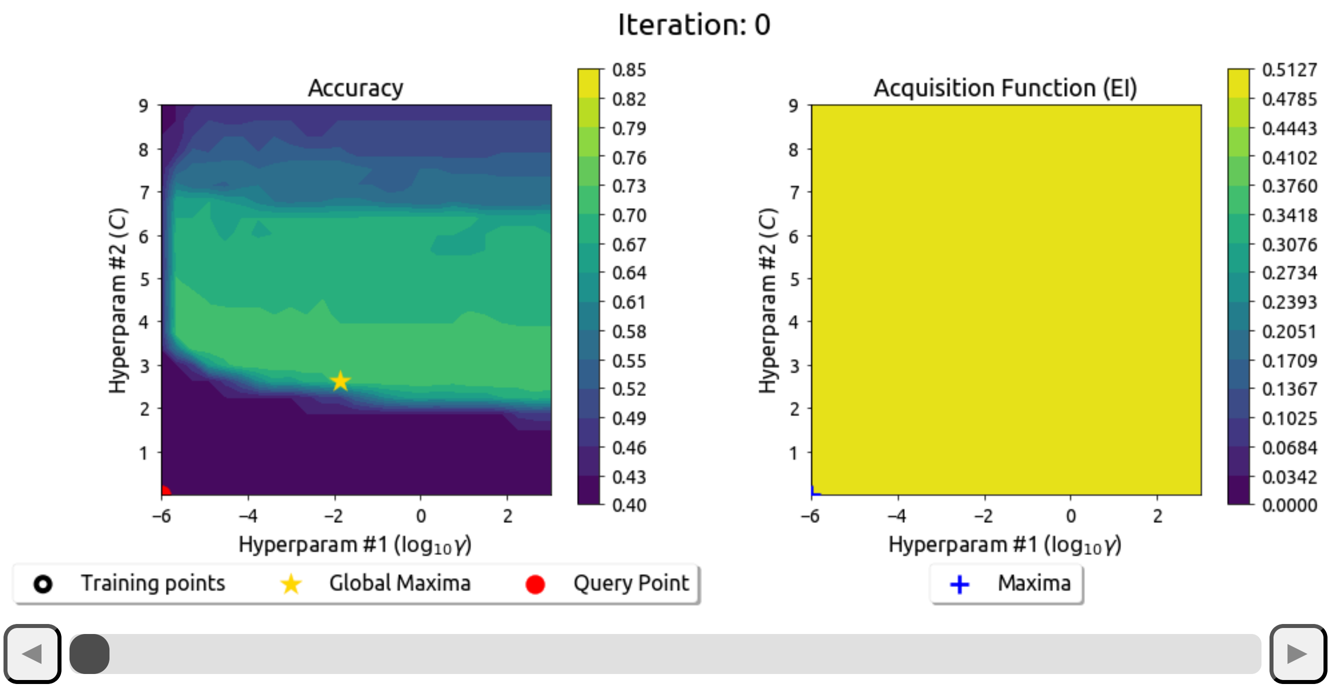
Let us have a look at the dataset now, which has two classes and two features.



Let us apply Bayesian Optimization to learn the best hyperparameters for this classification task [17]. The optimum values for <C, γ> have been found via running grid search at high granularity.



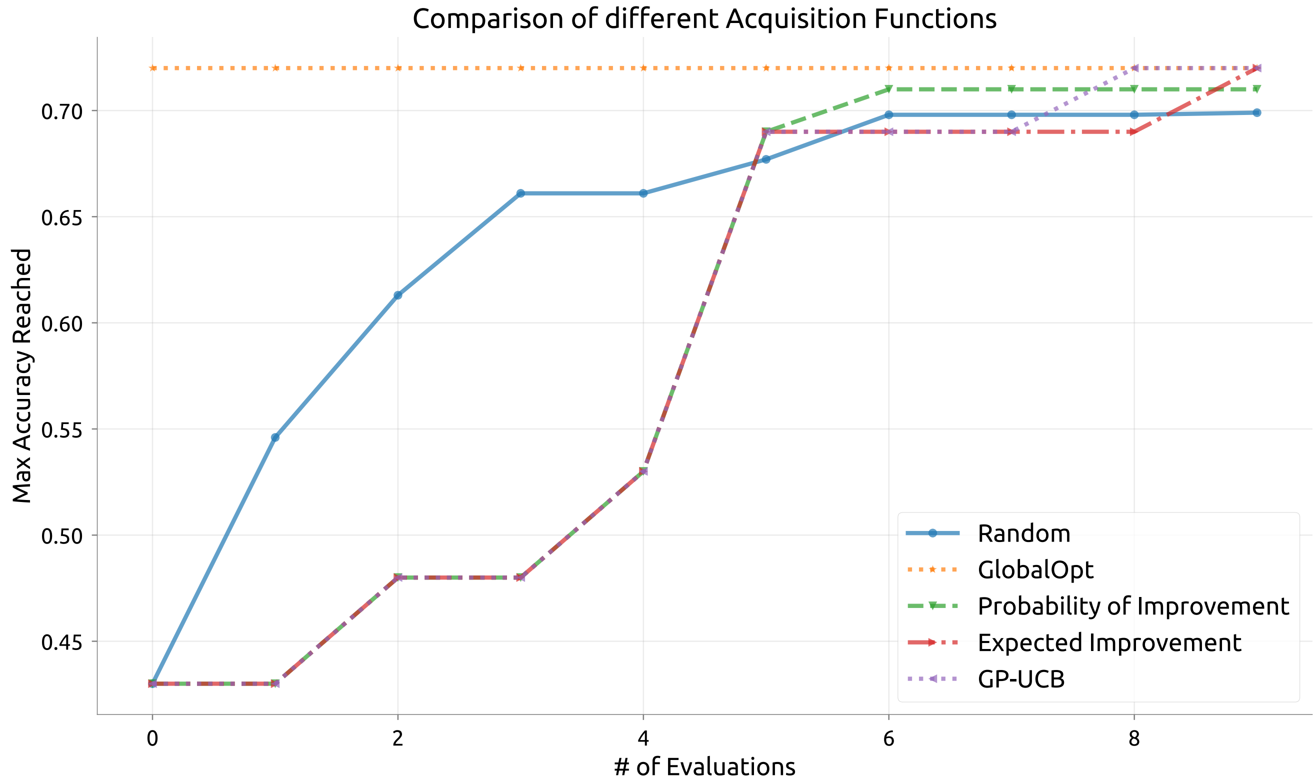
Above we see a slider showing the work of the *Probability of Improvement* acquisition function in finding the best hyperparameters.



Above we see a slider showing the work of the *Expected Improvement* acquisition function in finding the best hyperparameters.

Comparison

Below is a plot that compares the different acquisition functions. We ran the *random* acquisition function several times to average out its results.



All our acquisition beat the *random* acquisition function after seven iterations. We see the *random* method seemed to perform much better initially, but it could not reach the global optimum, whereas Bayesian Optimization was able to get fairly close. The initial subpar performance of Bayesian Optimization can be attributed to the initial exploration.

OTHER EXAMPLES +

Conclusion and Summary

In this article, we looked at Bayesian Optimization for optimizing a black-box function. Bayesian Optimization is well suited when the function evaluations are expensive, making grid or exhaustive search impractical. We looked at the key components of Bayesian Optimization. First, we looked at the notion of using a surrogate function (with a prior over the space of objective functions) to model our black-box function. Next, we looked at the "Bayes" in Bayesian Optimization — the function evaluations are used as data to obtain the surrogate posterior. We look at acquisition functions, which are functions of the surrogate posterior and are optimized sequentially. This new sequential optimization is in-expensive and thus of utility of us. We also looked at a few acquisition functions and showed how these different functions balance exploration and exploitation. Finally, we looked at some practical examples of Bayesian Optimization for optimizing hyper-parameters for machine learning models.

We hope you had a good time reading the article and hope you are ready to **exploit** the power of Bayesian Optimization. In case you wish to **explore** more, please read the Further Reading section below. We also provide our repository to reproduce the entire article.

Embrace Bayesian Optimization

Having read all the way through, you might have been sold on the idea about the time you can save by asking Bayesian Optimizer to find the best hyperparameters for your fantastic model. There are a plethora of Bayesian Optimization libraries available. We have linked a few below. Do check them out.

- scikit-optimize [20]
- sigopt
- hyperopt
- spearmint
- MOE
- BOTorch
- GPyOpt
- DragonFly

Acknowledgments

This article was made possible with inputs from numerous people. Firstly, we would like to thank all the Distill reviewers for their punctilious and actionable feedback. These fantastic reviews immensely helped strengthen our article. We further express our gratitude towards the Distill Editors, who were extremely kind and helped us navigate various steps to publish our work. We would also like to thank Dr. Sahil Garg for his feedback on the flow of the article. We would like to acknowledge the help we received from Writing Studio to improve the script of our article. Lastly, we sincerely thank Christopher Olah. His inputs, suggestions, multiple rounds of iterations made this article substantially better.

Further Reading

1. Using gradient information when it is available.
   - Suppose we have gradient information available, we should possibly try to use the information. This could result in a much faster approach to the global maxima. Please have a look at the paper by Wu, et al.[16] to know more.
2. To have a quick view of differences between Bayesian Optimization and Gradient Descent, one can look at this amazing answer at StackOverflow.

3. We talked about optimizing a black-box function here. If we are to perform over multiple objectives, how do these acquisition functions scale? There has been fantastic work in this domain too! We try to deal with these cases by having multi-objective acquisition functions. Have a look at this excellent notebook for an example using `gpflowopt`.

4. One of the more interesting uses of hyperparameters optimization can be attributed to searching the space of neural network architecture for finding the architectures that give us maximal predictive performance. One might also want to consider nonobjective optimizations as some of the other objectives like memory consumption, model size, or inference time also matter in practical scenarios.

5. When the datasets are extremely large, human experts tend to test hyperparameters on smaller subsets of the dataset and iteratively improve the accuracy for their models. There has been work in Bayesian Optimization, taking into account these approaches [17, 18] when datasets are of such sizes.

6. There also has been work on Bayesian Optimization, where one explores with a certain level of “safety”, meaning the evaluated values should lie above a certain security threshold functional value [19]. One toy example is the possible configurations for a flying robot to maximize its stability. If we tried a point with terrible stability, we might crash the robot, and therefore we would like to explore the configuration space more diligently.

7. We have been using GP in our Bayesian Optimization for getting predictions, but we can have any other predictor or mean and variance in our Bayesian Optimization.

   - One can look at this slide deck by Frank Hutter discussing some limitations of a GP-based Bayesian Optimization over a Random Forest based Bayesian Optimization.

   - There has been work on even using deep neural networks in Bayesian Optimization [20] for a more scalable approach compared to GP. The paper talks about how GP-based Bayesian Optimization scales cubically with the number of observations, compared to their novel method that scales linearly.

8. Things to take care when using Bayesian Optimization.

   - While working on the blog, we once scaled the accuracy from the range $[0,\ 1]$ to $[0,\ 100]$. This change broke havoc as the Gaussian Processes we were using had certain hyperparameters, which needed to be scaled with the accuracy to maintain scale invariance. We wanted to point this out as it might be helpful for the readers who would like to start using on Bayesian Optimization.

   - We need to take care while using Bayesian Optimization. Bayesian Optimization based on Gaussian Processes Regression is highly sensitive to the kernel used. For example, if you are using Matern kernel, we are implicitly assuming that the function we are trying to optimize is first order differentiable.

   - Searching for the hyperparameters, and the choice of the acquisition function to use in Bayesian Optimization are interesting problems in themselves. There has been amazing work done, looking at this problem. As mentioned previously in the post, there has been work done in strategies using multiple acquisition function [21] to deal with these interesting issues.

   - A nice list of tips and tricks one should have a look at if you aim to use Bayesian Optimization in your workflow is from this fantastic post by Thomas on Bayesian Optimization with sklearn.

9. Bayesian Optimization applications.

   - Bayesian Optimization has been applied to Optimal Sensor Set selection for predictive accuracy [22].

   - Peter Frazier in his talk mentioned that Uber uses Bayesian Optimization for tuning algorithms via backtesting.

   - Facebook [23] uses Bayesian Optimization for A/B testing.

   - Netflix and Yelp use Metrics Optimization software like Metrics Optimization Engine (MOE) which take advantage of Parallel Bayesian Optimization [24].

Footnotes

1. Interestingly, our example is similar to one of the first use of Gaussian Processes (also called kriging) [1], where Prof. Krige modeled the gold concentrations using a Gaussian Process. [↩]
2. Gaussian Process supports setting of priors by using specific kernels and mean functions. One might want to look at this excellent Distill article [6] on Gaussian Processes [7] to learn more.

   Please find this amazing video from Javier González on Gaussian Processes. [↩]
3. Specifics: We use a Matern 5/2 kernel due to its property of favoring doubly differentiable functions. See Rasmussen and Williams 2004 and scikit-learn, for details regarding the Matern kernel. [↩]
4. More details on acquisition functions can be accessed at on this link. [↩]
5. The section below is based on the slides/talk from Peter Fraizer at Uber on Bayesian Optimization: Youtube talk, slide deck [↩]
6. Please find these slides from Washington University in St. Louis to know more about acquisition functions. [↩]
7. Ties are broken randomly. [↩]
8. The proportion of uncertainty is identified by the grey translucent area. [↩]
9. Points in the vicinity of current maxima [↩]
10. A good introduction to the Expected Improvement acquisition function is by this post by Thomas Huijskens and these slides by Peter Frazier [↩]
11. Each dot is a point in the search space. Additionally, the training set used while making the plot only consists of a single observation $(0.5, f(0.5))$ [↩]
12. Since “Probability of Improvement” is low [↩]
13. Since “Expected Improvement” is high [↩]
14. To know more about the difference between acquisition functions look at these amazing slides from Nando De Freitas [↩]
15. UCB and GP-UCB have been mentioned in the collapsible [↩]
16. StackOverflow answer for intuition behind the hyperparameters. [↩]
17. **Note**: the surface plots you see for the Ground Truth Accuracies below were calculated for each possible hyperparameter for showcasing purposes only. We do not have these values in real applications. [↩]
18. **Note**: One will need to negate the accuracy values as we are using the minimizer function from `scikit-optim`. [↩]
19. The example above has been inspired by Hvass Laboratories’ Tutorial Notebook showcasing hyperparameter optimization in TensorFlow using `scikit-optim`. [↩]
20. Really nice tutorial showcasing hyperparameter optimization on a neural network available at this link. [↩]

References

1. **A statistical approach to some basic mine valuation problems on the Witwatersrand** [link]
   Krige, D., 1951. Journal of the Southern African Institute of Mining and Metallurgy, Vol 52(6), pp. 119-139. Southern African Institute of Mining and Metallurgy.

2. **Active Learning Literature Survey** [PDF]
Settles, B., 2009.

3. **Active learning: theory and applications** [PDF]
Tong, S., 2001.

4. **Taking the Human Out of the Loop: A Review of Bayesian Optimization**
Shahriari, B., Swersky, K., Wang, Z., Adams, R.P. and Freitas, N.d., 2016. Proceedings of the IEEE, Vol 104(1), pp. 148-175. DOI: 10.1109/JPROC.2015.2494218

5. **A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning**
Brochu, E., M. Cora, V. and De Freitas, N., 2010. CoRR, Vol abs/1012.2599.

6. **A Visual Exploration of Gaussian Processes** [link]
Görtler, J., Kehlbeck, R. and Deussen, O., 2019. Distill. DOI: 10.23915/distill.00017

7. **Gaussian Processes in Machine Learning** [PDF]
Rasmussen, C.E., 2004. Advanced Lectures on Machine Learning, pp. 63--71. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-28650-9_4

8. **Bayesian approach to global optimization and application to multiobjective and constrained problems**
B. Mockus, J. and Mockus, L., 1991. Journal of Optimization Theory and Applications, Vol 70, pp. 157-172. DOI: 10.1007/BF00940509

9. **On The Likelihood That One Unknown Probability Exceeds Another In View Of The Evidence Of Two Samples** [link]
Thompson, W.R., 1933. Biometrika, Vol 25(3-4), pp. 285-294. DOI: 10.1093/biomet/25.3-4.285

10. **Using Confidence Bounds for Exploitation-Exploration Trade-Offs**
Auer, P., 2003. J. Mach. Learn. Res., Vol 3(null), pp. 397–422. JMLR.org.

11. **Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design**
Srinivas, N., Krause, A., Kakade, S.M. and Seeger, M., 2009. arXiv e-prints, pp. arXiv:0912.3995.

12. **Practical Bayesian Optimization of Machine Learning Algorithms** [link]
Snoek, J., Larochelle, H. and Adams, R.P., 2012. Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2, pp. 2951--2959. Curran Associates Inc.

13. **Algorithms for Hyper-Parameter Optimization**
Bergstra, J., Bardenet, R., Bengio, Y. and K\'{e}gl, B., 2011. Proceedings of the 24th International Conference on Neural Information Processing Systems, pp. 2546–2554. Curran Associates Inc.

14. **Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures** [link]
Bergstra, J., Yamins, D. and Cox, D.D., 2013. Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, pp. I–115–I–123. JMLR.org.

15. **Scikit-learn: Machine Learning in {P}ython**
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., 2011. Journal of Machine Learning Research, Vol 12, pp. 2825--2830.

16. **Bayesian Optimization with Gradients** [PDF]
Wu, J., Poloczek, M., Wilson, A.G. and Frazier, P., 2017. Advances in Neural Information Processing Systems 30, pp. 5267--5278. Curran Associates, Inc.

17. **Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization** [PDF]
Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. and Talwalkar, A., 2018. Journal of Machine Learning Research, Vol 18-185, pp. 1-52.

18. **Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets** [HTML]
Klein, A., Falkner, S., Bartels, S., Hennig, P. and Hutter, F., 2017. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Vol 54, pp. 528--536. PMLR.

19. **Safe Exploration for Optimization with Gaussian Processes** [link]
Sui, Y., Gotovos, A., Burdick, J.W. and Krause, A., 2015. Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, pp. 997--1005. JMLR.org.

20. **Scalable Bayesian Optimization Using Deep Neural Networks** [link]
Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M.M.A., Prabhat, P. and Adams, R.P., 2015. Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, pp. 2171--2180. JMLR.org.

21. **Portfolio Allocation for Bayesian Optimization** [link]
Hoffman, M., Brochu, E. and de Freitas, N., 2011. Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, pp. 327--336. AUAI Press.

22. **Bayesian Optimization for Sensor Set Selection** [link]
Garnett, R., Osborne, M.A. and Roberts, S.J., 2010. Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, pp. 209--219. ACM. DOI: 10.1145/1791212.1791238

23. **Constrained Bayesian Optimization with Noisy Experiments** [link]
Letham, B., Karrer, B., Ottoni, G. and Bakshy, E., 2019. Bayesian Anal., Vol 14(2), pp. 495--519. International Society for Bayesian Analysis. DOI: 10.1214/18-BA1110

24. **Parallel Bayesian Global Optimization of Expensive Functions**
Wang, J., Clark, S.C., Liu, E. and Frazier, P.I., 2016. arXiv e-prints, pp. arXiv:1602.05149.

Updates and Corrections

If you see mistakes or want to suggest changes, please create an issue on GitHub.

Reuse

Citation

For attribution in academic contexts, please cite this work as

```
Agnihotri & Batra, "Exploring Bayesian Optimization", Distill, 2020.
```

BibTeX citation

```
@article{agnihotri2020exploring,
  author = {Agnihotri, Apoorv and Batra, Nipun},
  title = {Exploring Bayesian Optimization},
  journal = {Distill},
  year = {2020},
  note = {https://distill.pub/2020/bayesian-optimization},
  doi = {10.23915/distill.00026}
}
```

Distill is dedicated to clear explanations of machine learning

About Submit Prize Archive RSS GitHub Twitter ISSN 2476-0757