Parametric Optimization with Gaussian Process Upper Confidence Bounding

Michael Willett
mwillett@seas.upenn.edu

Abstract—Optimization theory is a critically important field of study for building robust software packages and tools. Often times engineers and scientists are faced with the problem of selecting a set of control variables to yield to best performance output for some system, however, the exact dynamics of the system are unknown. In these cases, the designer must sample various points within the configuration space. If the cost sampling a system is high, then it is obvious that designer will want to minimize the total number of sampled points before convergence on the global optima. The goal of the Gaussian Process Upper Confidence Bound (GP-UCB) algorithm is to provide probabilistic guarantees that the total cumulative regret of the sampling process falls below sub-linear bounds. This paper shows that while we can follow sub-linear regret bounds in higher dimensional space, the dimensionality of the system greatly influences the regret of the final convergence point.

1 Gaussian Process Optimization

This paper focuses it's work on the task of identifying optima of an unknown function in a finite domain using a Gaussian processes as the function estimator. Gaussian processes are an implementation of kernel machines that estimate the shape of the function by sampling the measured mean value of the function at some set of discrete points, and using a kernel function to estimate the variance of the function in unmeasured regions. Gaussian processes are particularly useful when generating functions in high dimensional spaces that can be efficiently calculated with matrix operations, and where some prior knowledge of the search domain can be used to choose the appropriate smoothing kernels.

Due to the structure of Gaussian processes, it is easy to identify regions in the domain that should be explored due to high uncertainty. For this case study, we will be exploring the multi-armed bandit problem, where the algorithm must effectively negotiate when to explore uncertain regions in the search space, and when it should start exploiting the best state it has found through the exploration process.

Prior to the work of Srinivas et al., several groups have investigated using the variance from the Gaussian process as the primary metric for determining the best point for exploration when trying to model the entire function. However, often times with the context of the optimization problem, the designer does not need to know fine details of the function in regions that are highly unlikely to contain the global optima in the search space. In this case, Srinivas proposes the intuitive solution of only sampling the point with the highest potential value based off of the mean and variance. As a method of assurance to properly explore uncertain areas, Srinivas adds a scaling factor to the variance parameter that allows the exploration process to be bounded entirely by the potential information gain of the sampled point. The details of this algorithm are shown in 1.

Algorithm 1 The GP-UCB algorithm

1: **Input:** Input space D; GP Prior $\mu_0 = 0, \sigma_0, k$
2: **for** $t = 1, 2, ...$ **do**
3: Choose $\boldsymbol{x}_t = \text{argmax } \mu_{t-1}(\boldsymbol{x}) + \sqrt{\beta_t}\sigma_{t-1}(\boldsymbol{x})$
4: Sample $y_t = f(\boldsymbol{x}_t) + \epsilon_t$
5: Perform Bayesian update to obtain μ_t and σ_t
6: **end for**

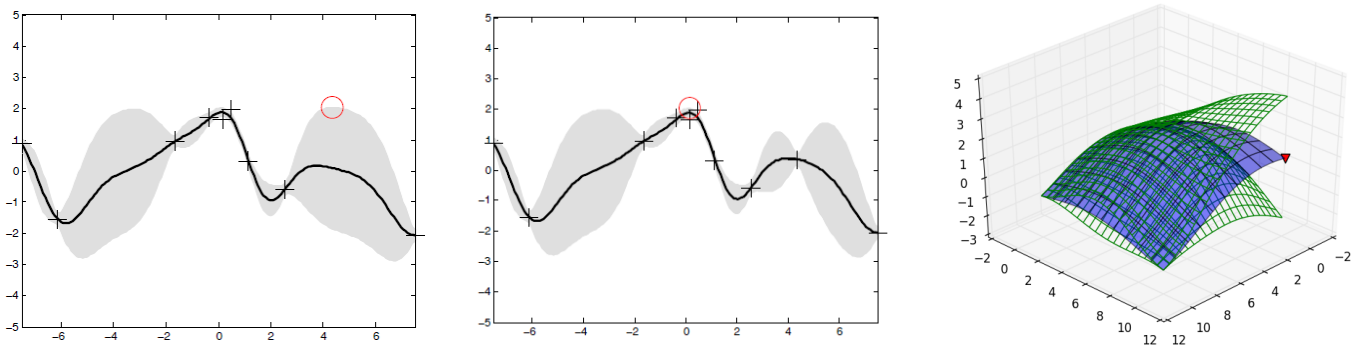
While Srinivas et al. suggest several values for β_t depending on the kernel and assumed "complexity" of the function measured with RKHS norms, Lipschitz continuity, and compactness, we chose to limit this analysis to most general case of simply assuming finite D. In all tests we assume a GP with zero mean and radial basis function kernel. According to Srinivas, we use the following equations in the GP-UCB algorithm.

$$\beta_t = 2\log\left(\frac{|D|t^2\pi^2}{6\delta}\right), \quad \delta \in (0,1)$$

$$Pr\left\{R_T \leq \sqrt{\frac{8T\beta_T\gamma_T}{\log\left(1+\sigma^{-2}\right)}}, \quad \forall T \geq 1\right\} \geq 1-\delta$$

$$\gamma_T = \sqrt{T(\log T)^{d+1}}$$

The process can be easily visualized in figures 1. The figure shows two steps in the GP-UCB algorithm. In the first image the red circle highlights the point with the highest mean and uncertainty. After the function has been sampled at this point, we see the variance decrease



(a) Iteration t (b) Iteration $t+1$ (c) Sampling in 2D

Fig. 1: Sampling Points with GP-UCB in 1D and 2D

and the mean function is updated, and the next point to be sampled is near the global optima.

Some preliminary testing was performed with linear, squared exponential, and matern kernels, but RBF proved to be most stable in converging on accurate results. We used the zero mean function for the Gaussian process, though constant mean can be used to limited success depending on the input function. The zero mean function was stable under the three different functions tested.

2 Test Implementation

The tool-chain pyGPs by Neumann et al. was used for updating model parameters and samples are taken, as it has an extensive library of built in kernels and mean functions for fitting various models. There exist various other libraries that may reduce run time, but this platform was sufficient for ease of development, and performance appeared to scale well as more CPU cores are available on the machine. All tests were run on a quad-core Intel Core i5 with 16GB without hyper-threading.

One deficit of the library is that the built in hyper-parameter optimization is inconsistent. This will lead to the optimization process starting to converge at a final point, but then will re-optimize the scaling parameters on the covariance matrix, which disturbs the GP-UCB method by inappropriately over-weighting the exploration trade off. The side effect was that the algorithm would randomly start to explore many points on the out limits of the search domain, before coming back to explore previously identified optima.

The solution to this problem was to initialize the model with some amount of random points and optimize the hyper-parameters once before running the GP-UCB process. This solution resulted in the expected transition from exploration to exploitation overtime, and in all test cases up to 10 dimensions, the algorithm converged after

Trial	Final Regret	x_1	x_2	x_3
1	.5806	2.5853	2.9809	3.0986
2	.3585	3.4895	3.2191	3.1798
3	.5183	3.6415	3.1124	3.2754
4	.6157	3.0104	2.7447	2.6896
5	1.4334	3.5163	4.5239	3.2015
6	1.7530	4.0333	4.3604	4.0318
7	.7194	2.7957	2.7078	3.5995
8	.4518	3.2616	3.4188	3.4776
9	.5138	3.1922	2.9562	3.6181
10	6.8490	10.0000	3.2637	3.4098
11	1.7090	3.8073	4.5249	3.8926
12	.6415	3.3850	2.6516	2.8067
13	6.9219	10.0000	3.2083	4.0749
14	.5092	3.259	3.1193	3.6366
15	.6307	3.4989	3.6491	3.2535
16	.6605	2.5715	2.8723	3.3385
17	1.7246	4.3349	4.0436	3.9998
18	6.8701	10.0000	3.4721	3.3680
19	1.7796	4.2127	4.2237	4.0630
20	3.1470	2.9872	0.0000	3.0397
average	**1.2809**	**4.3791**	**3.2526**	**3.45275**
outliers removed	**.4866**	**3.3872**	**3.4238**	**3.4528**

TABLE 1: Sine Wave, 3D, 300 iterations, no noise

500 samples. While each test trial recalculated hyper-parameters during the initialization sequence, experimental results showed that these values remained consistent between trials of the same function and dimensionality.

Several test sets were collected to measure the effects of different control variables. Accuracy of the final optima were calculated while only adjusting one parameter at a time, as well as analysis over multiple iterations. Tests were done while varying different δ values, dimensionality, β functions, and noise levels. Additional testing on the scalability of this platform was performed as well.

3 Results

Unless otherwise specified, tests were performed on simple sinusoidal function for easy calculation of regret:

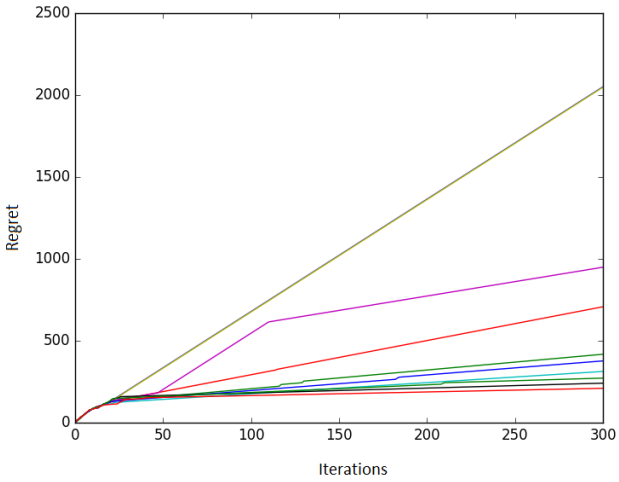


Fig. 2: Cumulative Regret Over 10 Samples

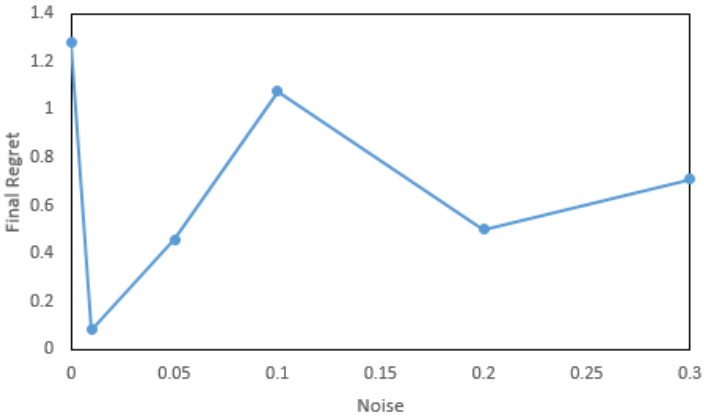


Fig. 3: Final Regret With Gaussian Noise

$$y(\boldsymbol{x}) = \sum_{i=1}^{d} \sin \frac{\boldsymbol{x}_i}{2} + \epsilon, \quad 0 \leq \boldsymbol{x}_i \leq 10$$

3.1 Convergence and Final Regret

While initial results seemed promising in low dimensionality, the method never appeared to fully converge on the global optimum in the function domain. Instead, it usually would converge a point near the optimum with a relatively low regret compared to random sampling, but not the true global optimum. Table 1 shows this result in more detail.

In this test, 20 trials were run where the GP-UCB algorithm was allowed to run for 300 iterations. In most cases, the algorithm would stop exploring new points within 100 samples as can be inferred by Figure 3. Over the 20 trials, the closest the GP-UCB algorithm got to the true optimum was trial 2, which had a final regret of .3585. While relatively close to the true optimum at $x_i = \pi$, considering that the worst case regret is 11.88, this convergence point falls well outside of general computational rounding errors of finding the optima.

More importantly, this test shows a major flaw in the algorithm that was never fully resolved during implementation. Trials $10, 13, 18,$ and 20, all generated a final point that was on a search domain limit. It was very common during the trial to see initial exploration along the domain limits across all dimensionality tests due to high uncertainty, and the major question was if the algorithm would eventually sample enough points along each dimension that were not on the limits to lead it to converge on the true optima.

Knowing that the implementation from Srinivas only provides probabilistic guarantees on the regret, it was assumed that the final optima would best be determined by averaging over multiple trials, this domain limit problem greatly skewed this final average point. Table 1 shows the difference in final regret if averaging all trials versus selectively pruning out points on the domain edges. Without any pruning of data, the regret of the average point is heavily skewed towards the center point of the search domain (this assumes it will generate points on the lower and upper bound of a variable with equal likelihood, which tests suggest). If manually cleaned, the final regret of the average does show improved performance over most single trials, however, this does assume the the optimization problem is framed such that it is known that the optima cannot occur at the domain limits, which may be restrictive for many real world problems.

3.2 Adding Noise

When adding a zero mean normal distribution to the test function, some interesting behavior starts to emerge. For very small noise ($\sigma^2 = .01, .05$), we observed no cases where the GP-UCB process converged on a point that lied on one of the domain limits over 10 trials. In fact, every trial for $\sigma^2 = .01$ resulted in substantially lower final regret than any other noise lowers. When σ^2 was $.05$ and higher, we started to see final regret values similar to the no noise tests.

Further investigation is required, but these results could be due to some of the default parameters in the pyGPs package which calculates variance updates relative to some initialize prior. In all tests we used the default values, but repeating the noise tests for different start states may result in more accurate convergence values for each trial.

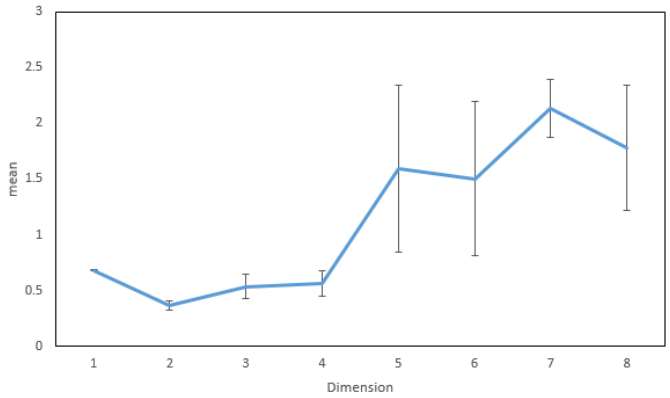


Fig. 4: Regret Variance Over 10 Trials

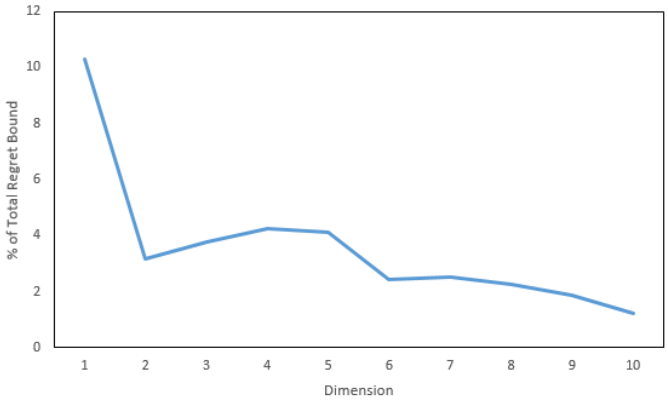


Fig. 5: Measured Total Regret as Measured by Regret Upper Bound

3.3 Modifying β_t

Several ad hoc tests were preformed to test the final regret if the exact formulation of β_t is modified at the expense of loosing probabilistic guarantees on the cumulative regret.

$$\beta_t = 2\log\left(\frac{|D|\pi^2}{6\delta}\right), \quad \delta \in (0,1) \tag{1}$$

$$\beta_t = 2\log\left(\frac{dt^2\pi^2}{6\delta}\right), \quad \delta \in (0,1) \tag{2}$$

$$\beta_t = 2C\log\left(\frac{|D|t^2\pi^2}{6\delta}\right), \quad \delta, C \in (0,1) \tag{3}$$

In the first test, the goal was to investigate what a constant β_t term might have, while equations 2 and 3 tested how different scaling β_t might effect rate of convergence. Equation 2 hopes to remove the size of the search domain as a factor, and equation 3 just test general scaling down of β_t.

Modification of the β_t function had little effect on the final convergence point. For particularly large changes to the order of magnitude of β_t, one could control the amount relative amount of exploration the algorithm would perform before convergence. In theory, adding more priority to the exploration process should result in lower final regret, though no significant improvements were found during this process. The greatest effect these modulations had was the number of iterations that passed before the algorithm converged on a final point. It may be possible that further improvements in the $argmax$ operation would improve accuracy in tandem with these effects, but that is left for further study.

3.4 Scalability

From a run time perspective, the algorithm scales quite well. Assuming that the initial calculation of the hyper-parameters for the Gaussian process is not intractable, Figure 6 shows that the run time appears to scale linearly up to 10 dimensional space.

The majority of computation time occurred during the $argmax$ operation to find the next sampling point (in most tests, this took over 98% of calculation time). In the test implementation, this operation was performed using basin hopping constrained gradient decent, with three restarts to avoid traditional problems with the fact that the optimization problem is not guaranteed to be convex. Increasing the number of restarts or using different optimization methods can change the run time. However, in these tests it had little effect on the final regret of the GP-UCB algorithm.

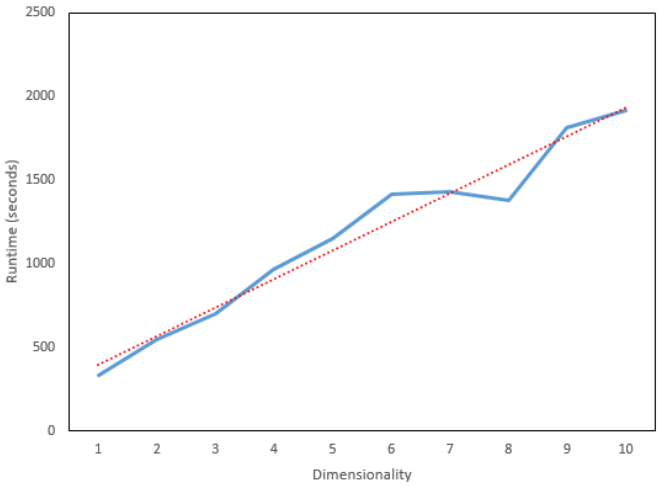


Fig. 6: Run Time for 500 iterations (red line denotes linear trend)

The biggest problem facing the scalability concern is that the average final regret over multiple trials appears to increase significantly as dimensionality increases. Figure 4 shows the sharp change in regret variance when

going from 4 to 5 dimensions. In general, accuracy of the final convergence point significantly decreased as dimensionality increase. One would expect the regret metric to grow on the order of $O(\sqrt{N})$ since regret was calculated using N-Dimensional euclidean distance. While there were not enough trials performed in high-dimensional space to state that average regret does not grow according to $O(\sqrt{N})$, the variance in the 10 trials conducted at each dimensionality suggest that the true final regret will be difficult to measure without an excessive number of trials.

3.5 Total Regret Bounds

While Srinivas's paper only guarantees zero-regret for a very specific class of functions, it still provides generalized cumulative regret bounds dependant on potential information gain for several kernel functions. Using this, we can compare the final total regret for any trial and compare it to the theoretical bounds.

Figure 5 shows that while we do see worse final regret as the dimensionality increases, the margin of total cumulative regret for individual trials compared to the theoretical limit seems to improve, which is promising if accuracy can be improved for higher dimensional problems.

4 Conclusion

Overall, this implementation for using Gaussian processes for optimization shows initial promise for the multi-armed bandit problem for limiting the number of exploration samples, though if the most accurate measurement of true optima is required, the algorithm needs some improvement. Preliminary tests show that if the control parameters are set appropriately relative to prior knowledge of the function domain and expected noise, we can get repeatably low regret values for each trial, but as soon as the optimization problem changes, these values need to be modified again. Since the goal of this algorithm is to reduce the number of samples of the optimization function as much as possible, rerunning trials to get the correct control parameters appropriate for the noise and dimensionality is not ideal. If a strong correlation could be found that would provide context for initializing the GP-UCB process, this could be a valuable tool for optimizing high dimensionality problems with a relatively small amount of training samples.

References

[1] Srinivas et al., *Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design*, arXiv:0912.3995v4.

[2] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, the MIT Press, 2006.

[3] M. Neumann et al., *pyGPs* (Version 1.3.2) http://www-ai.cs.uni-dortmund.de/weblab/static/api_docs/pyGPs/, Feb.2014.