Project Report

Alexis Park, Jonathan Chen, Kevin Xie
CSE515T: Bayesian Methods in Machine Learning

December 4, 2019

1 Data visualization

The Branin unction is defined as follows:

$$f(\boldsymbol{x}) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s \tag{1.1}$$

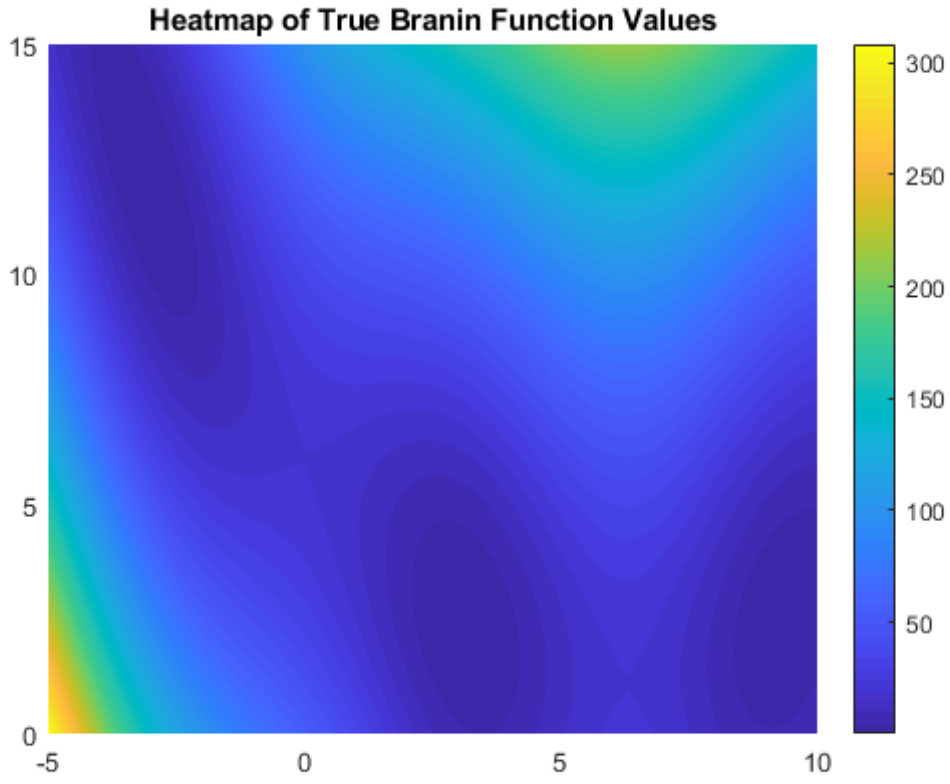

Figure 1: Heatmap of True Branin function values with domain $\mathcal{X} = [-5, 10] \times [0, 15]$ with 1000 values per dimension.

From figure 1, we can see that the function's values fluctuate in a somewhat sinusoidal manner, with a large minimal "trench" spanning diagonally across the center of the domain with steadily

increasing regions along the trench's sides. Therefore the function does not appear to be stationary.

In order to make the function more stationary, we tried a variety of transformations. By analyzing the equation, we noted that the sinusoidal fluctuations can be attributed to both the added cosine term and the 4th order polynomial term. We thus attempted to pass the data through a transformation that combined an inverse cosine function (arccos) with the cumulative density function of the normal probability distribution ($\Phi(x)$). However, taking the log of the data gave us the most stationary result as shown in figure 2.

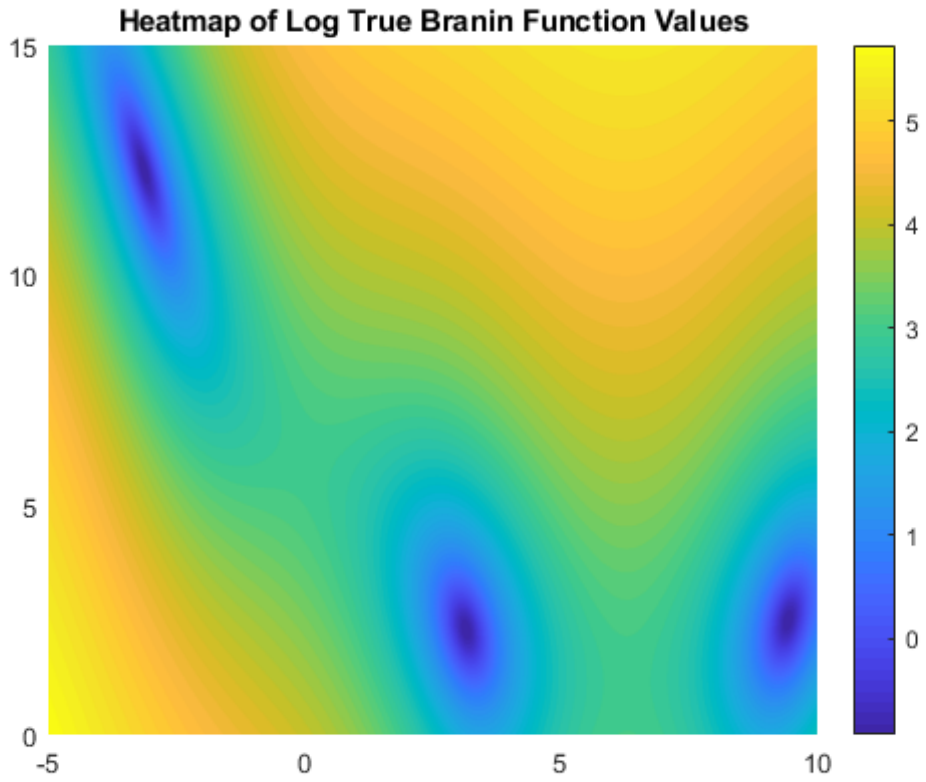

Figure 2: Heatmap of log Branin function values with domain $\mathcal{X} = [-5, 10] \times [0, 15]$ with 1000 values per dimension. The plot shows the behavior of the Branin function is more constant throughout the domain when compared to the original function's behavior.

Next, we plotted a kernel density estimate (KDE) of LDA and SVM benchmarks (Fig. 3A and 3C). By analyzing the resultant probability density graphs, we identified that KDE of LDA show log-normal like distribution. Also, both estimates have relatively similar behavior but on significantly different scales.

Since KDE of LDA show log-normal like distribution, the log of LDA values should produce the normal distribution, which resulted in better performance (Fig. 3B). As we expected, the (log) LDA KDE is more distributed throughout the graph and relatively less concentrated around the peak. However, the (log) SVM KDE made the distribution less normal as shown in figure 3D.

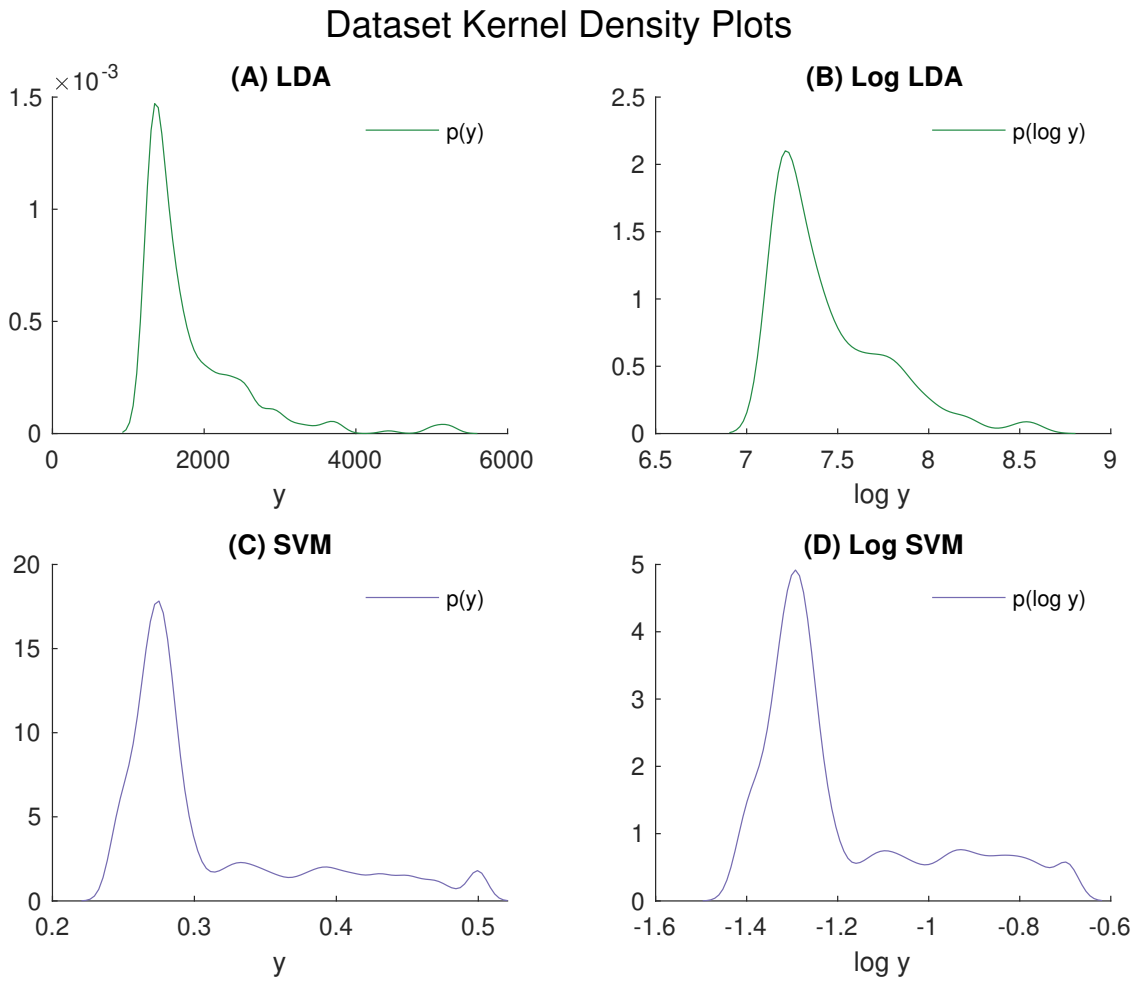

Figure 3: Kernel density estimates for LDA (green, A and B) and SVM (purple, C and D) datasets, along with the estimates for their logs.

2 Model fitting

Model fitting with Branin function

For model fitting, we used 32 Sobol sequence Branin function training points and fit the data to a Gaussian Process (GP) model with constant mean with initial value of 0, squared exponential covariance with initial values of length scale and output scale of 1. After optimization by maximizing the marginal likelihood, we obtained the following values for the hyperparameters:

- mean: 137.19
- length scale: 4.00
- output scale: 114.05

Although we assumed that there is no noise in the data, the exact Gaussian likelihood is parameterized and the optimization procedure also fit it's value (1e-3). This is fine because the number is small, but also because this increases the numerical stability of the GP.

Based on figure 4, both the mean and the length scale values are reasonable. The range of the Branin function extends from approximately 0 to 350, with more mass below the approximate "midpoint" value of 175. Thus, an optimized function mean of 137.19 is also reasonable. The overall slowly oscillating behavior of the Branin function also appears to have a "periodicity" of approximately six units; thus, a length scale of 4.00 is acceptable. The oscillations themselves have some combination of amplitudes that range from small towards the center of the function's domain to very large towards the edges; thus, an output scale of 114.05 is acceptable.

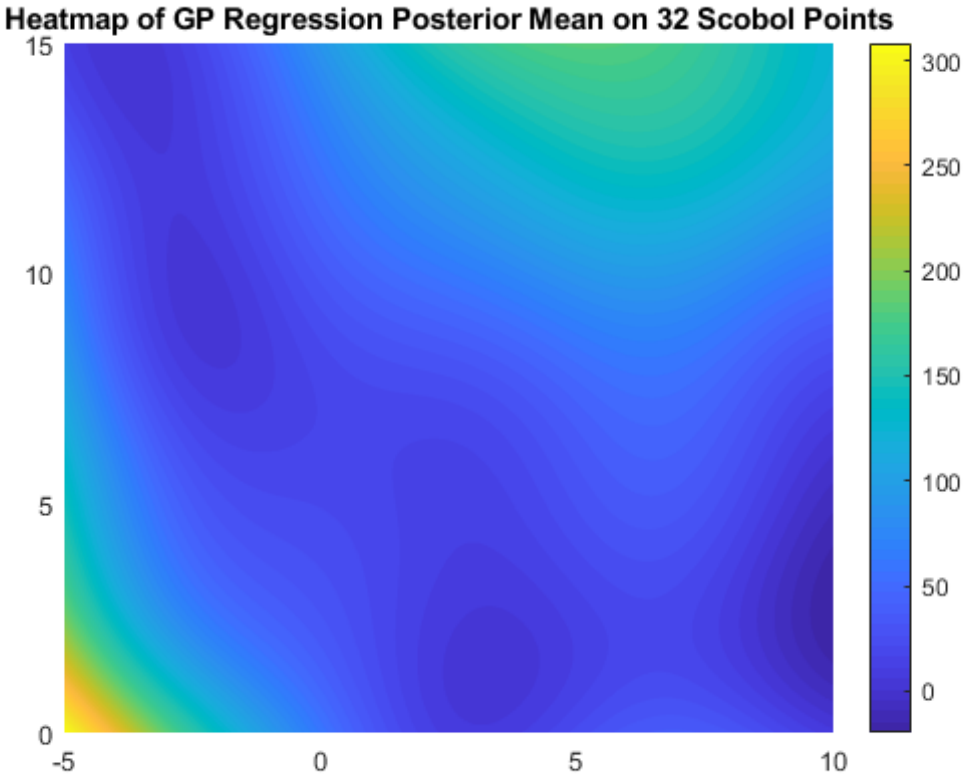

Figure 4: Heatmap of GP posterior mean

Comparing the predicted GP posterior mean to the true Branin function values, figure 5 shows that the training points lie in the darkest spots on the heatmap, where the predicted and true values are almost identical. Similarly, as the predictive mean's location departs from the areas where the training values are known, the absolute differences increase.

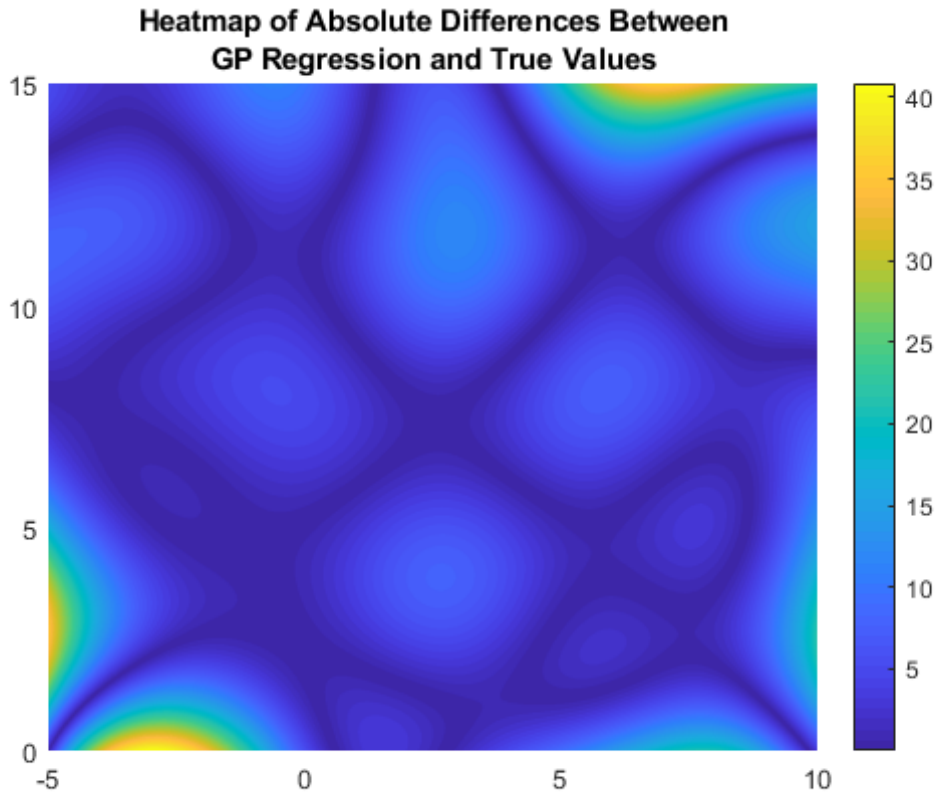

Figure 5: Heatmap of GP posterior mean and True Branin values

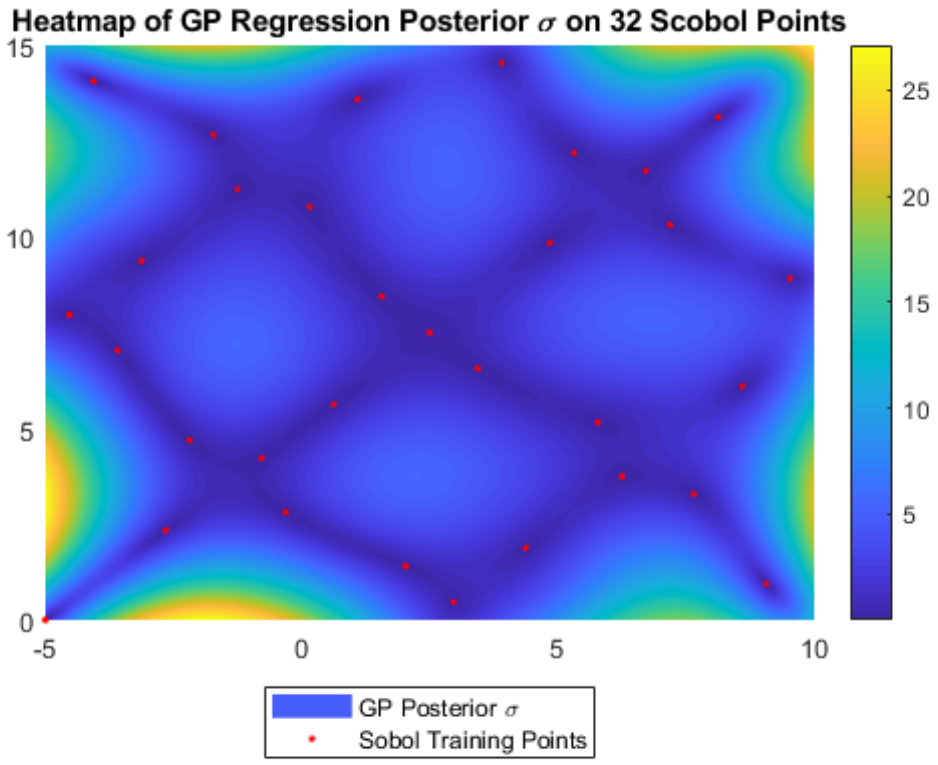

Figure 6: Heatmap of the GP posterior standard deviation

As shown in figure 6, as you approach the training data points, the standard deviation approaches zero; Areas far from the training points therefore have significantly higher uncertainty. Furthermore, the scale of the standard deviation extends from 0 to 25. This behavior is expected, as one would expect the GP model to be certain near the training points (where it knows the underlying function passes through), and more uncertain as it gets farther from these points. The scale of this uncertainty is similarly reasonable. As previously mentioned, the Branin function has both small and large variations. Thus, the GP must return a relatively high standard deviation at the locations where it is uncertain in order to properly accommodate the large range of possible fluctuations.

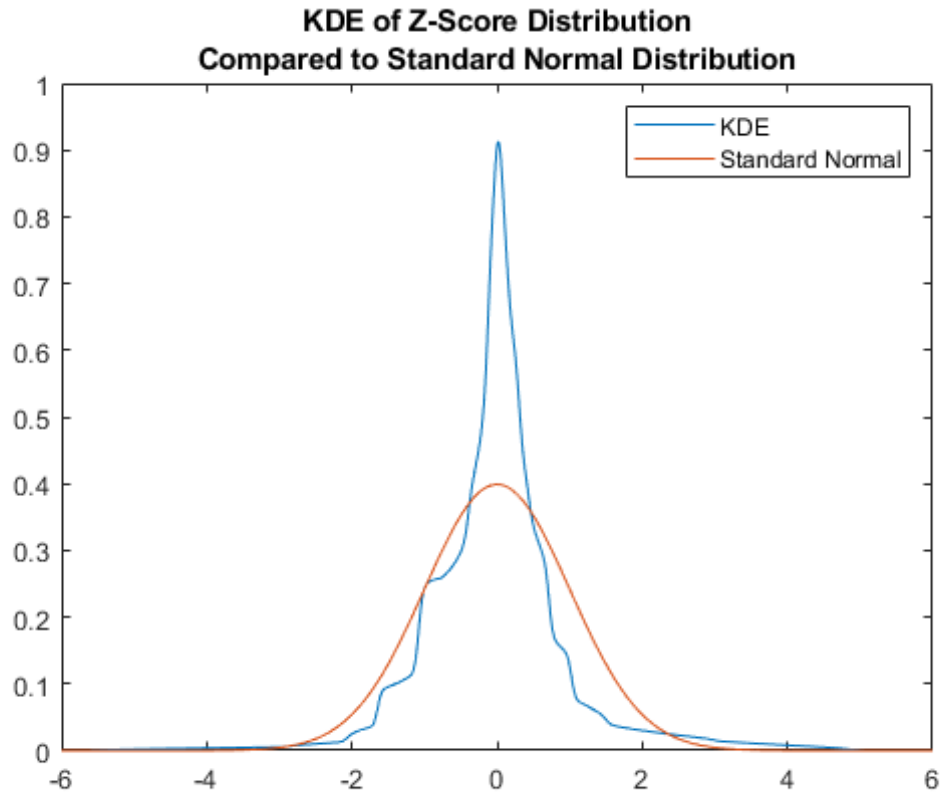

Figure 7: Kernel density estimate of the z-scores of the residuals between the GP posterior mean and true values.

Based on figure 7, the KDE of the Z-score distribution follows an approximately standard normal distribution with more concentrated density at the peak.

Model fitting with log transformation of Branin function

We repeated model fitting using a log transformation of the Branin function. After maximizing the marginal likelihood under the same conditions as the previous fitting attempt, we obtained the following hyperparameter values:

- mean: 3.62
- length scale: 3.32
- output scale: 1.63

Again, the optimization procedure also fit the likelihood, resulting in a hyperparameter value of 0.001.

The log Branin posterior mean ranges from approximately 0.5 to 6 as shown in figure 8, with more area above the midpoint value of 3.25. Thus, the hyperparameter mean value with a slight increment from the "midpoint" follows our expectation. The range of the function also supports an output scale of 1.63, which indicates a relatively "flat" function, as expected. Finally, the oscillatory behavior of the log branin function appears to be slightly less frequent; thus, a length scale of 3.32 is reasonable.

Figure 8: Heatmap of GP posterior mean of log transformed Branin function values

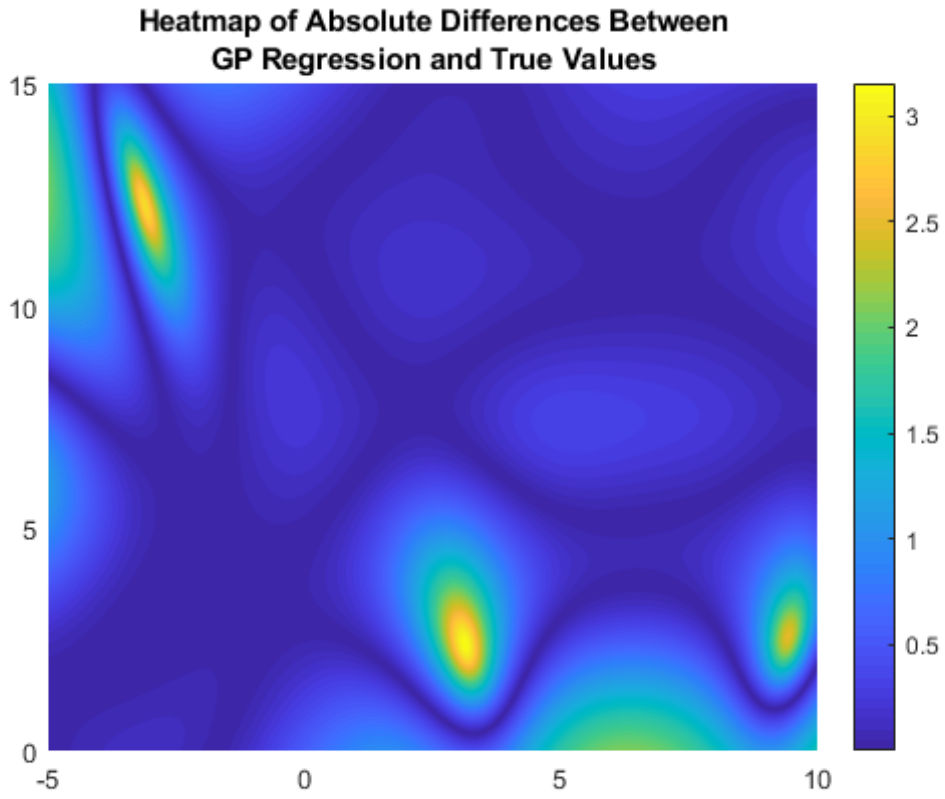

Figure 9: Heatmap of GP posterior mean and true log transformed Branin function values

The absolute difference between the predicted values and the true values are nearly zero as shown in the darkest areas of figure 9 like in figure 5. Additionally, the values farther away from the training points have increasing absolute differences, including areas which show very high absolute differences.

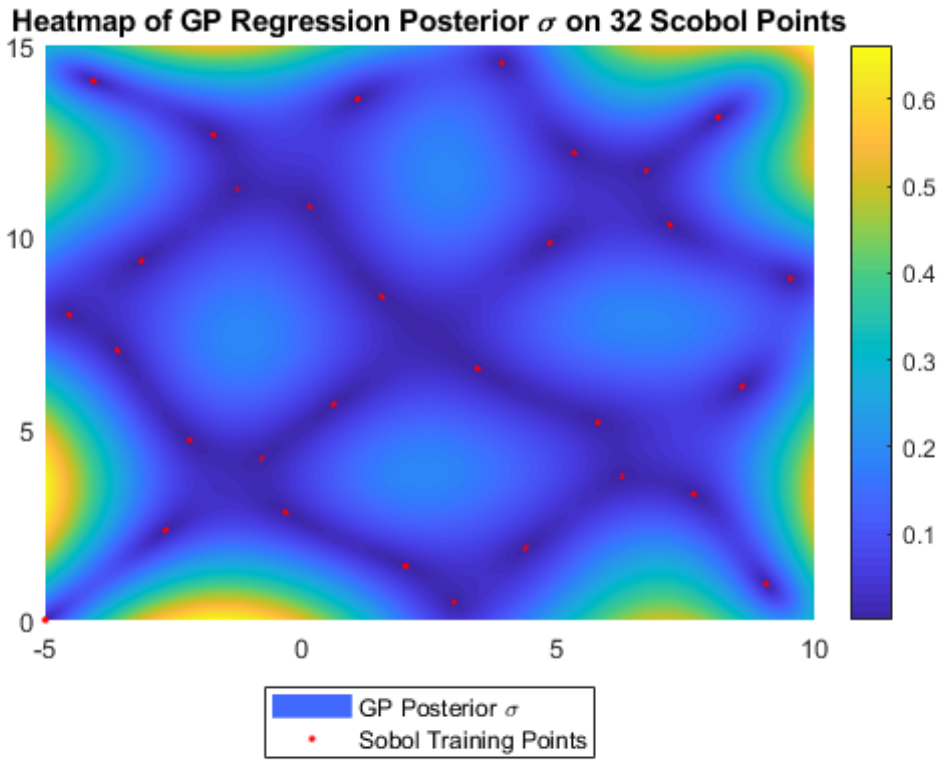

Figure 10: Heatmap of GP posterior standard deviation of log transformed Branin function values

As shown in figure 10, the standard deviation again approaches zero as the predicted values approach the training points. The scale of the standard deviation extends from 0 to approximately 0.7. As explained previously, this behavior is expected: the GP model should be more confident the closer it is to a training point. The application of the log function to the Branin model should also reduce the scale of the standard deviation, as seen.

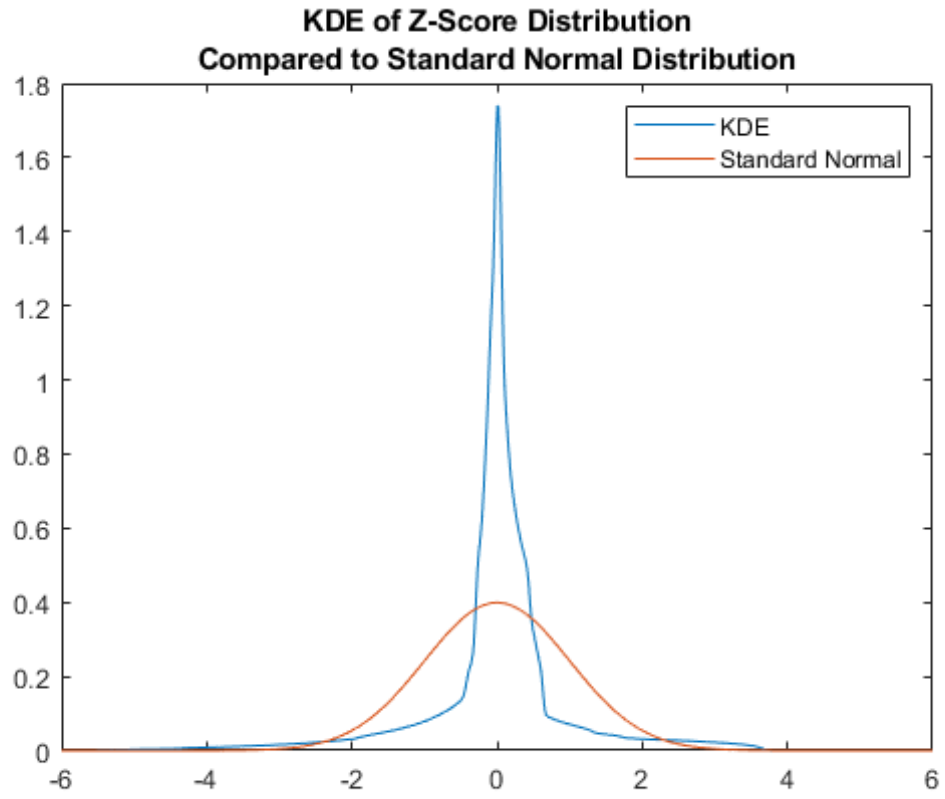

Figure 11: Kernel density estimate of z-score of log transformed Branin function values

Based on figure 11, the KDE of the Z-score distribution is similar a "focused" Gaussian distribution.

By using a log transformation of the Branin function, the marginal likelihood is improved and more calibrated: the absolute difference between predicted and true values are much lower in figure 9 than in figure 5, which indicates that the model trained from log transformed data has higher marginal likelihood.

Bayesian Information Criterion (BIC)

In order to compare which model fits better with the dataset, we used the BIC to evaluate how well those choices fit a given dataset. To compute the BIC, we found the values of the hyperparameters maximizing the (log) marginal likelihood using 2.1, and then 2.2, where $|\theta|$ is the total number of hyperparameters and $|\mathcal{D}|$ is the number of observations. We used three for $|\theta|$ and 32 for $|\mathcal{D}|$.

From equation 2.2, it can be seen that the first term should be minimized to prevent overfitting, while the second term should be maximized, since it corresponds to the model evidence. Overall, lower BIC values imply better model fit.

$$\hat{\theta} = \arg\max_{\theta} \log p(y \mid X, \theta) \tag{2.1}$$

$$\text{BIC} = |\theta| \log |\mathcal{D}| - 2 \log p(y \mid X, \hat{\theta}) \tag{2.2}$$

Data	BIC score
Branin	304.3118
log Branin	61.4710

Table 1: BIC score of data from model fitting (Branin and (log) Branin)

Now considering BIC as a function of the choice of mean and covariance functions (μ, K), we used MeanConst and four different covariance functions, Squared Exponential (SE), Rational Quadratic (RQ), sum of SE and RQ, and product of SE and RQ. Table 2 shows the best model and BIC scores we found for each function.

Data	Model	BIC score
Branin	Rational Quadratic	299.367
log Branin	Squared Exponential	**61.471**
LDA	Product of SE and RQ	6.777e6
log LDA	Rational Quadratic	**17.298**
SVM	Product of SE and RQ	**-138.894**
log SVM	Rational Quadratic	-77.547

Table 2: Computed BIC scores and best model for different functions and their log transformations. The best value for each function is shown in bold.

3 Bayesian Optimization

The best-fitting models were selected from the previous experiments. We then used the Expected Improvement (EI) acquisition function, defined as:

$$a_{EI}(\boldsymbol{x}) = (f' - \mu(\boldsymbol{x}))\Phi(f'; \mu(\boldsymbol{x}), K(\boldsymbol{x},\boldsymbol{x})) + K(\boldsymbol{x},\boldsymbol{x})\mathcal{N}(f'; \mu(\boldsymbol{x}), K(\boldsymbol{x},\boldsymbol{x})), \tag{3.1}$$

where $\Phi(\boldsymbol{x})$ is the cumulative probability density of the Normal distribution, and f' is the minimum value of the current observations. Note that the EI acquisition function combines both exploitation and exploration by using the posterior mean and known minimum value, and the posterior standard deviation, respectively.

Using the previously selected 32 points, a GP model was fit using the aforementioned optimized settings and the following heatmaps of the posterior mean and standard deviation of the log Branin function were created:

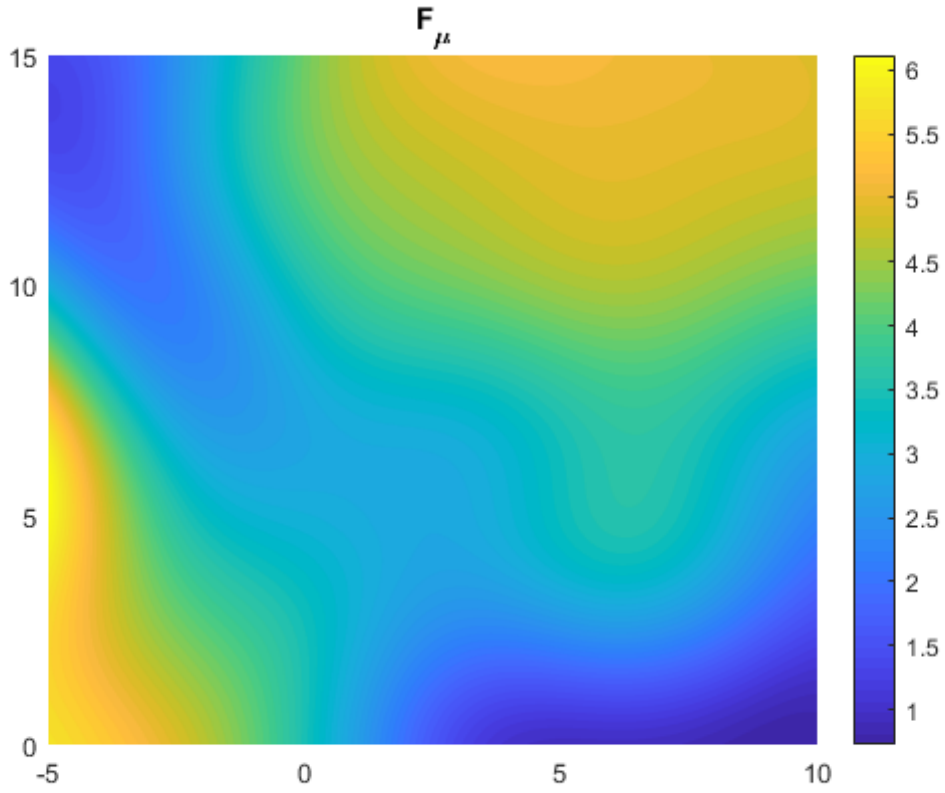

Figure 12: Predictive posterior mean of the log Branin function, calculated using a previously optimized GP model trained on 32 Sobol sequence points. Warmer colors indicate higher values. Note the predicted minimum areas in dark blue at the top left and bottom right corners of the plot.

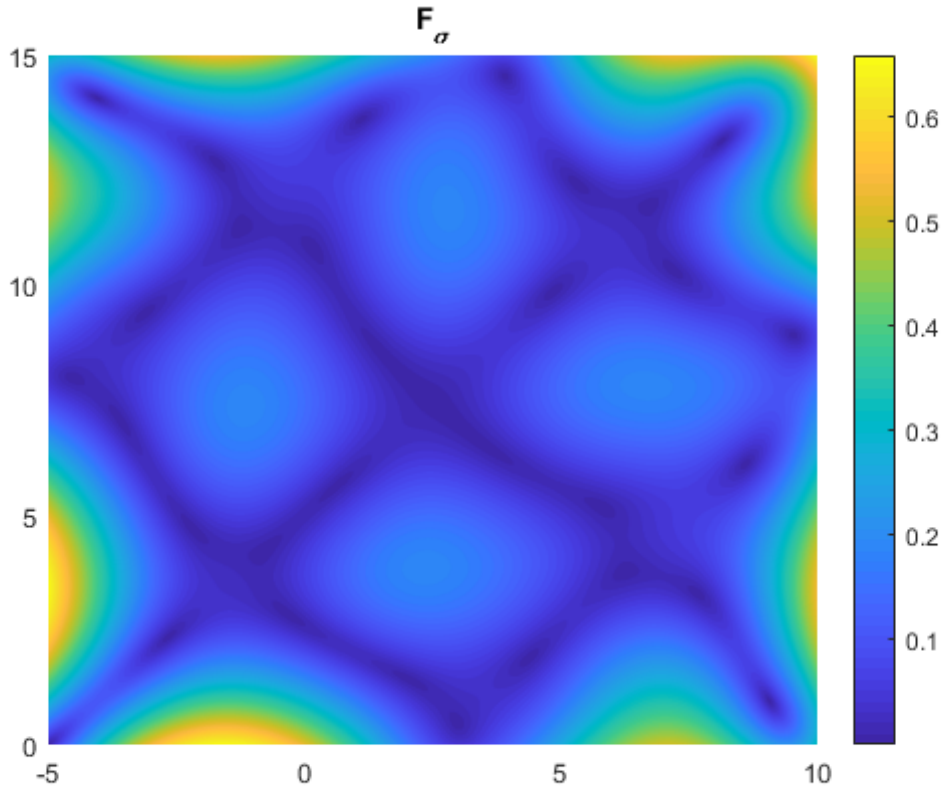

Figure 13: Predictive posterior standard deviation of the log Branin function, calculated using a previously optimized GP model trained on 32 Sobol sequence points. Warmer colors indicate higher deviations and imply greater uncertainty in the prediction.

The EI function values were then calculated using the posteriors and the point $(7.658, 0)$ was identified as the optimal point to observe next. A heatmap of the EI values is shown below with the optimal observation location marked as a black point:

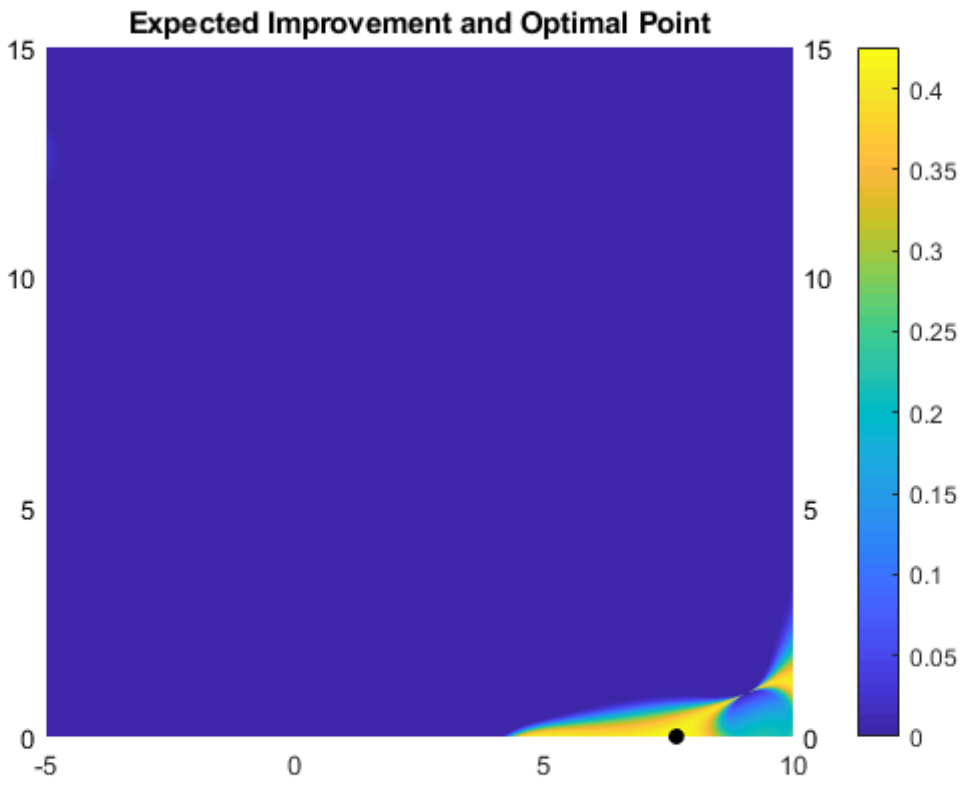

Figure 14: Expected Improvement acquisition values of the optimized GP log Branin model trained on 32 Sobol sequence points. Warmer colors indicate higher expected improvement. The maximum expected improvement is marked as a black point and denotes the optimal location for the next observation.

Analyzing figures 12, 13, and 14 above, we reason that the proposed optimal testing point is ideal. From the posterior mean, it can be seen that the point $(7.658, 0)$ is within a region of predicted minimal values. Furthermore, from the posterior standard deviation, it can be seen that the point is also within a region of higher uncertainty and therefore reduced predictive confidence. Thus, it is plausible that the EI acquisition function would seek to test this area and this specific point in order to identify a possible global minimum and improve confidence in an area of uncertainty.

The following Bayesian active learning experiment was then applied independently to the log Branin, log LDA, and SVM functions:

1. Five initial observations were randomly selected, constituting the initial dataset $\mathcal{D}$.
2. For the log Branin function only, a dense, evenly spaced grid of 250,000 points was generated within the domain of the function for use in calculating the GP predictive posterior.
3. A GP model using the respective aforementioned optimized settings was fit to $\mathcal{D}$.
4. A new point x was found using the EI acquisition function and the GP predictive posterior.
5. The function value $f(x)$ was calculated and the point $(x, f(x))$ was added to $\mathcal{D}$.

6. Steps 3-5 were repeated 30 times, resulting in a final dataset $\mathcal{D}$ of 35 points.

The performance of each of the above experiments was evaluated using the "gap" measure, defined for minimization as:

$$\text{gap} = \frac{f(\text{best found}) - f(\text{best initial})}{f(\text{minimum}) - f(\text{best initial})} \tag{3.2}$$

The gap can be interpreted as a measure of the accuracy or performance of the optimization; as the gap value approaches one, the attempted optimization approaches the true global minimum.

The gaps for the log Branin, log LDA, and SVM models were calculated to be 1.00, 0.79, and 0.96, respectively. This implies that EI successfully found a global minimum of the log Branin function, but missed the global minimum of the log LDA and SVM functions.

The above bayesian active learning experiment was then modified as such:

1. A seed for the random number generator (RNG) was chosen.
2. Five initial observations were randomly selected, constituting the initial dataset $\mathcal{D}$.
3. For the log Branin function only, a dense, evenly spaced grid of 250,000 points was generated within the domain of the function for use in calculating the GP predictive posterior.
4. A GP model using the respective aforementioned optimized settings was fit to $\mathcal{D}$.
5. A new point x was found using the EI acquisition function and the GP predictive posterior.
6. The function value $f(x)$ was calculated and the point $(x, f(x))$ was added to $\mathcal{D}$.
7. Steps 4-6 were repeated 150 times, resulting in a final dataset $\mathcal{D}$ of 155 points.

The above procedure was duplicated, only the Random Search acquisition function (which selects a point at random) was used in place of EI. These experiments were repeated 20 times with 20 different RNG seeds to create different random initializations for the log Branin, log LDA, and SVM functions, resulting in a total of 60 experiments. The learning curves of these 60 trials using only the first 30 new observations for both EI and RS acquisition functions are shown in the figures below:

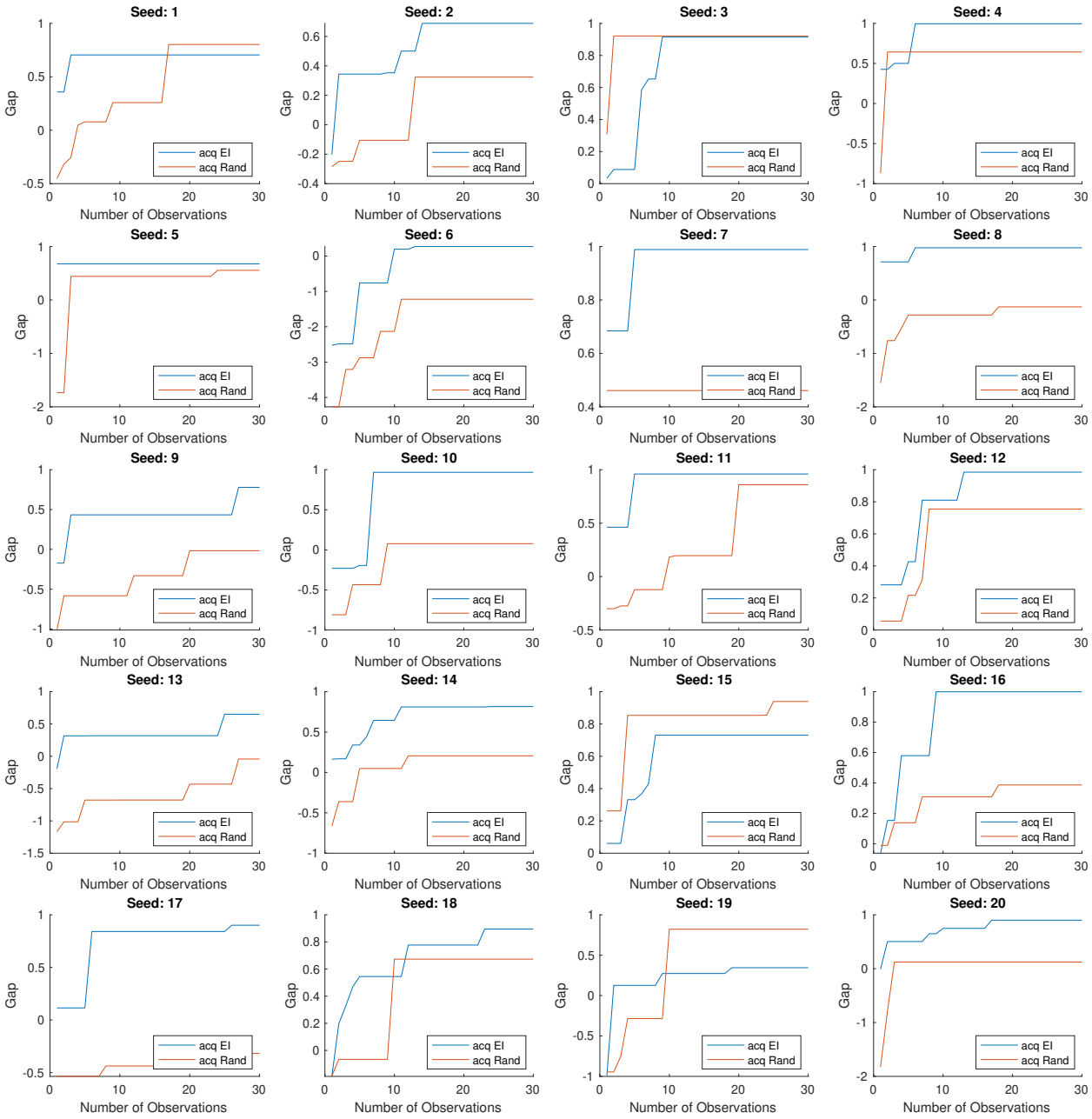

Figure 15: Learning curves for 20 log Branin function experiments. The EI acquisition function is shown in blue, and the RS acquisition function is shown in orange. Observe that EI consistently outperforms RS in both rate of gap increase and final gap convergence at 30 observations.

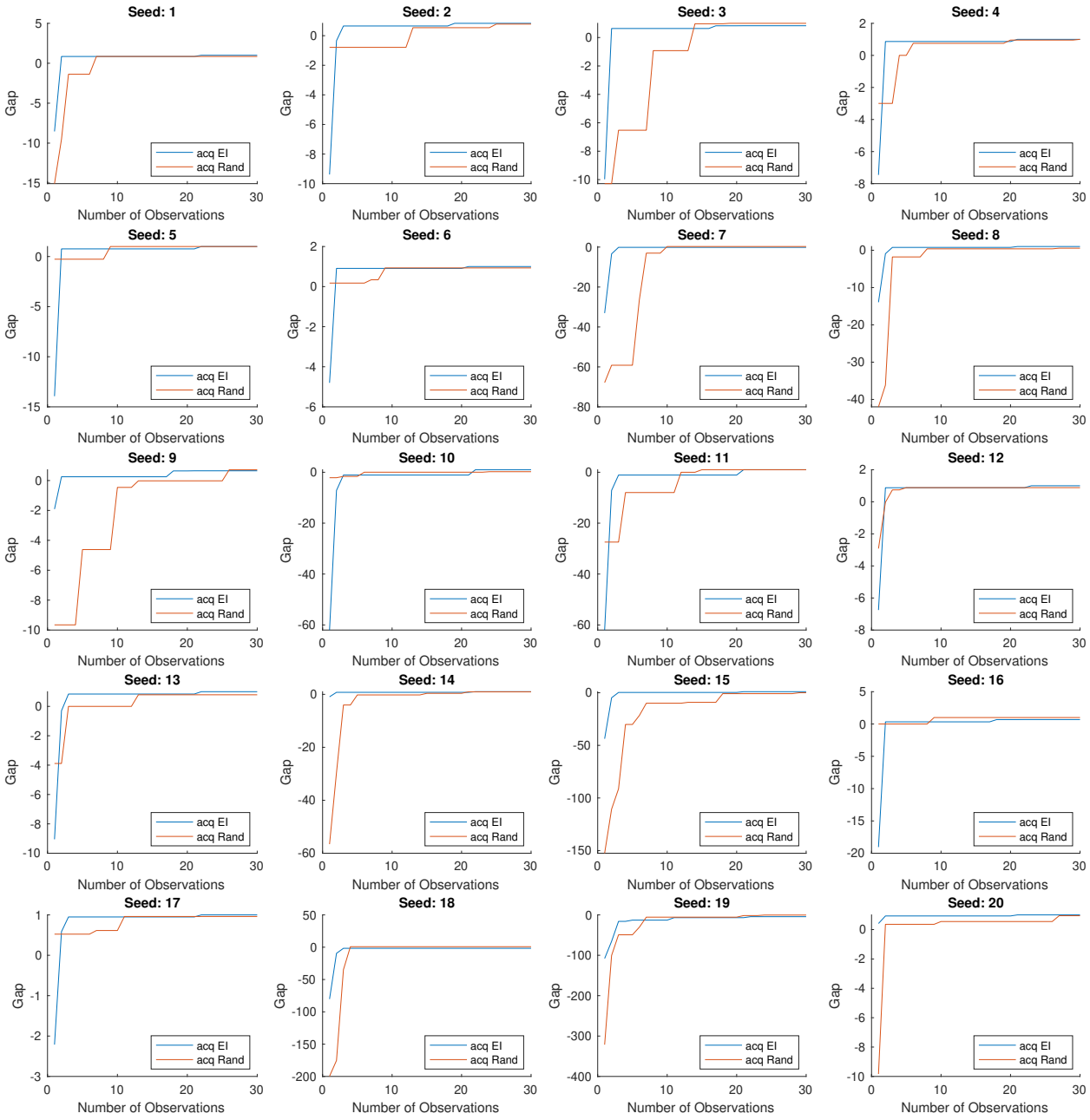

Figure 16: Learning curves for 20 log LDA function experiments. The EI acquisition function is shown in blue, and the RS acquisition function is shown in orange. Note that EI and RS both consistently converge towards the same gap value at 30 observations

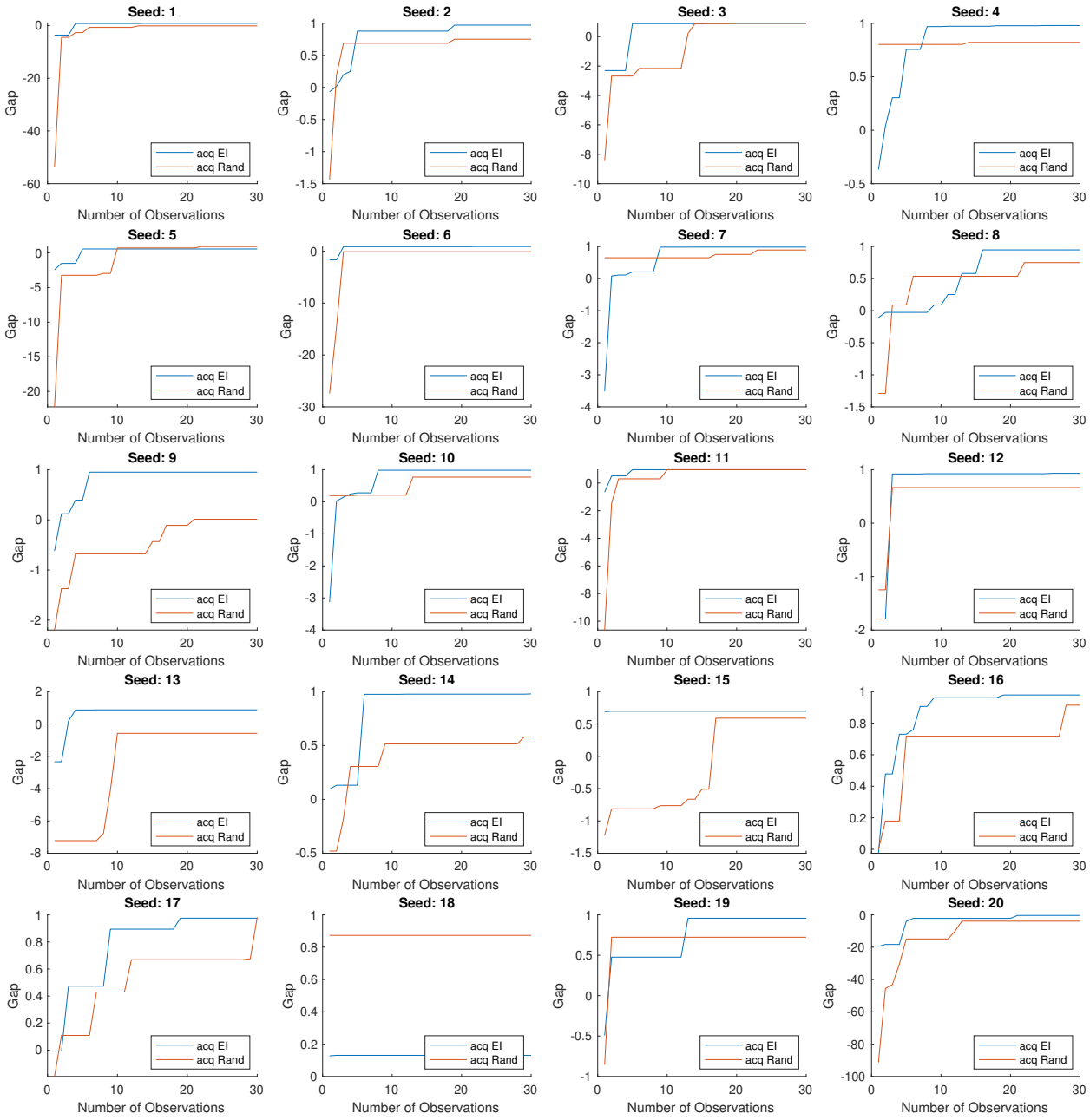

Figure 17: Learning curves for 20 SVM function experiments. The EI acquisition function is shown in blue, and the RS acquisition function is shown in orange. It can be seen that EI performs at least as well as RS in the majority of experiments, but can find and get stuck in a local minima, as seen in Seed 18.

From figures 15, and 17 above, it can be seen that EI significantly outperforms RS on both the log Branin and the SVM functions. It is evident that in these experiments, EI often converges to its final gap value in significantly fewer observations than RS does, and that its final gap values at 30 observations are generally higher than those of RS. This behavior was expected, as EI combines both exploitation and exploration together to identify locations with higher expected benefit, whereas RS only applies (random) exploration. The addition of exploitation and non-random exploration in EI allows it to actually navigate towards optima, whereas RS can only approach op-

tima through chance. However, there are cases where EI and RS have similar performance: from figure 16, it can be seen that the two acquisition functions perform similarly to each other in the majority of experiments at 30 observations. We believe that EI's lack of performance on the log LDA function can be attributed to a large amount of local minima; EI can often become stuck in local minima, thereby decreasing its performance, whereas RS is purely exploratory and therefore unhindered by the presence of any local optima.

The difference in performance between EI and RS can be further seen when one considers their mean gaps at 30, 60, 90, 120 and 150 observations, as shown in the table below:

# Observations	EI (Branin)	RS (Branin)	EI (LDA)	RS (LDA)	EI (SVM)	RS (SVM)
30	0.9072	0.3409	0.4877	0.7122	0.9106	0.3728
60	0.9532	0.6086	0.9611	0.7944	0.9415	0.5676
90	0.9532	0.7124	0.9753	0.9174	0.9592	0.9190
120	0.9532	0.9096	0.9753	0.9633	0.9018	0.9716
150	0.9532	0.9202	0.9473	0.9016	0.9018	0.9737

Table 3: Mean Gaps of EI and RS on the log Branin, log LDA, and SVM functions at 30, 60, 90, 120 and 150 observations. Note that EI reaches a higher gap at 150 observations in all functions when compared to RS, signifying better performance overall. Furthermore, for the log Branin and SVM functions, EI achieves these higher values significantly faster than RS does.

It is evident that EI outperforms RS on all three functions, including log LDA, at 150 observations. Additionally, EI generally requires less observations to achieve higher gap values, implying a faster learning rate than RS. We also note that EI likely spends considerable time early on navigating local minima in the log LDA function, as evident by its noticeably worse performance at 30 observations when compared to that of RS. However, EI for log LDA rapidly improves following this local-minima exploration and quickly outperforms RS. It is also important to note that on average, no method was able to find the global minimum. This result was expected for RS, which would need to be extraordinarily lucky to find the global minimum, but was somewhat surprising for EI. It appears that in the limit, EI may become entrenched in local minima and thus converges towards a suboptimal result.

We more rigorously compare the performance of EI and RS on all three functions using a paired t-test and evaluate the null hypothesis that the gap values attained for EI and RS come from the same distribution (i.e. the performance of EI and RS are equivalent or similar). We begin the test using only 30 observations for both EI and RS and calculate the corresponding p-values, as seen in the table below:

Function	P-Value
Log Branin	5.39e-4
Log LDA	0.38
SVM	0.039

Table 4: P-Values comparing the similarity of performance between EI and gap on the log Branin, log LDA and SVM functions. Higher p-values indicate more similar performance.

From the table above, it is quite obvious that at 30 observations, EI and RS have significantly

different performance on the log Branin and SVM functions; however, the two acquisition functions have similar performance on the LDA function. Such behavior was supported visually in figures 15, 16 and 17.

To identify the number of observations required for RS and EI to exhibit similar performance, we then began RS at one observation and increment the number of observations for RS while holding EI at 30 observations until the p-value exceeds 0.05. The resultant number of observations before RS attains possibly similar performance to EI is shown below:

Function	Observations Required	P-Value
Log Branin	59	0.0624
Log LDA	10	0.0599
SVM	3	0.0951

Table 5: Number of observations required for RS before it achieves possibly similar performance to EI with 30 observations for the log Branin, log LDA, and SVM functions.

Looking at tables 4 and 5, one can see that in the log Branin function, EI significantly outperforms RS. Specifically, 59 points must be observed before RS and EI show possibly similar performance. However, it is also evident that RS likely performs as well as EI on the log LDA function. This was expected, given their similar learning curves above in figure 16. Perhaps more interestingly, the performance of RS varies significantly relative to that of EI for the SVM function: when there are less observations, the two acquisition methods, on average, demonstrate similar performance. However, as the number of observations increase, the two methods begin to differ before gradually converging with each other again. Another iterative paired t-test experiment was performed, and it was determined that such oscillatory behavior continued until 44 new observations were attained, after which the performance of EI and RS remained consistently comparable.

4 Bonus

For the bonus section, we implemented two more acquisition functions: max variance and lower confidence bound (LCB). We also tried optimizing hyperparameters with the addition of each observation to the data set, dubbed "online optimization" (OO) for fun, and injected more exploration by randomly sampling the next point instead of using the acquisition function suggested point with some probability.

More Acquisition Functions

We chose to implement max variance (eq. 4.1) as one of our acquisition functions because it seemed like an intuitive way to explore a function's values and shape. If our objective is to build a model of a target function by actively searching the parameter space, then it makes sense to use our current belief about f (the posterior) to pick the next point to sample. In the case of max variance, we would sample where our model is least certain. *A priori*, we wouldn't expect max variance to be optimal for identifying the minima of f since the objective it is optimizing only incentivizes exploration and not exploitation.

$$\alpha_{\max \mathrm{Var}}(x) = K(x, x) \tag{4.1}$$

We also selected the minimization formulation of the upper confidence bound acquisition function, dubbed lower confidence bound (eq. 4.2). This computes the trade off between exploitation and exploration directly by mixing naive greedy exploitation with just the minimum of the mean function and max variance from before.

$$\alpha_{LCB}(x;\beta) = \beta\sigma(x) - \mu(x), \text{ where } \sigma(x) = \sqrt{K(x,x)} \tag{4.2}$$

Looking at it's formulation, we can see that it is the difference between the standard deviation and the mean function value of the posterior. It is also parameterized by β, which allows us to control the amount of exploration we want to do. We can see that the function will be greatest when the mean function value is low and when the variance is high. LCB effectively addresses the limitation of the max variance acquisition function by extending it to do exploitation as well.

We evaluated the performance of each acquisition function on a 300-by-300 grid from the Branin function using a procedure similar to the one used in Section 3: starting with five randomly selected points, we fit a GP, computed acquisition values, used the maximum of those values to choose our next point, and computed the gap value of the new dataset. We did ten trials with 100 iterations of active observation for each function and averaged the gap values to get those shown in Figure 18 as solid lines.

On their own, max variance and LCB are slower to find the global minimum than expected improvement. Although not shown in the figure, max variance performed similarly to random search. This is what we expected since max variance does not do any exploitation. LCB performs relatively better, but is slower to move out of local minima. This is likely because of it's naive-greedy nature and is somewhat expected.

Forcing More Exploration

It is possible for our acquisition values to get stuck in local minima for too long. To mitigate this, we force each acquisition function to do more exploration by using a randomly sampled point with probability $p = 0.1$ instead of the one suggested by the acquisition function. From Figure 18, we can see that occasionally picking random samples did not help our models converge. It is possible that this is true only for the Branin function, but we can't say from this experiment alone.

Online Optimization

We also tried optimizing the model hyperparameters after each iteration. This is representative of how Bayesian optimization is performed in practice. It makes sense to do this for acquisition functions that rely on the model posterior. If we are evaluating an expectation, like in EI, then it makes sense to use the posterior of the most-likely model. For LCB, we use the mean and variance of the posterior directly, which means that its performance will improve if we have better estimates of both from a more likely model. We didn't expect max variance to improve because it's lack of exploitation still constrains its performance even with optimized hyperparameters. These hypotheses are confirmed in Figure 18. The gap values with online optimization converge to one faster for both EI and LCB. Here we see that LCB actually converges faster that EI with online optimization.

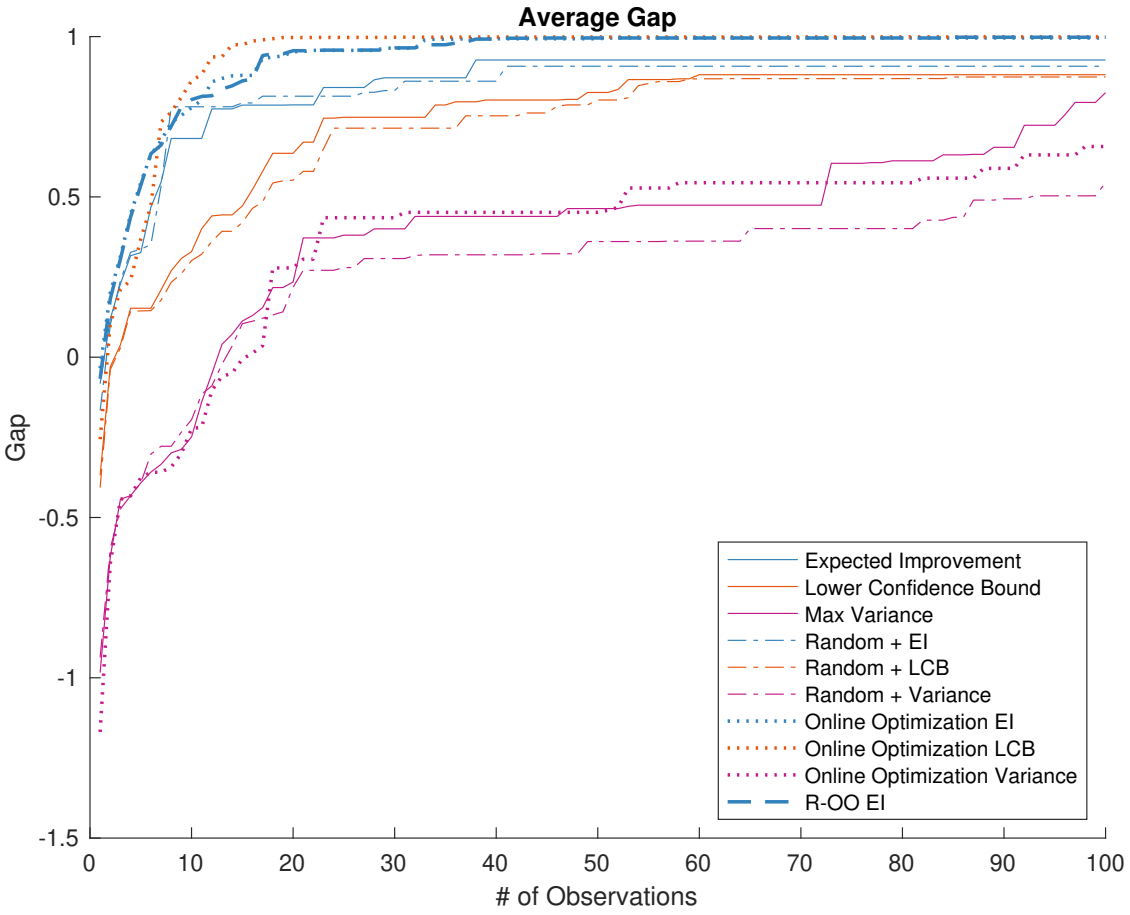

Figure 18: A busy plot showing gap as a function of the number of observations made.