Azeotropy : Optimizer

OP2024-06

Omkar Shirpure (22B0910), Akshit Pavagadhi (22B0369)

March 16, 2024

Contents

1 Introduction

This problem statement is all about minimizing a function which is unknown based upon the values of the parameters which we need to optimize to get the minimum value. In our case the two variables were risk and cost for which the total cost of T which is not known.

There are multiple approaches but since in our case the constraints are the requests are costly and we need to minimize them as much as possible and find the minimum value for T.

Several approaches can be considered for this optimization task. One option is to utilize polynomial regression, which is a common technique for fitting a function to data points. However, if the number of data points is limited, polynomial regression may not provide accurate results.

We could also choose another method where we uniformly distribute the points all over the range to calculate the values and take the minimum of those values but since those values may not be accurate and are highly inaccurate as the function may not always be lying on those points.

For this question I chose to use Bayesian estimation using Gaussian process regression method. What this method basically does is It calculates probability of the values at some points given the value of some fixed known data points. We start by fitting a model through the gather points and we somehow calculate the next best estimate for the point So as to get to the minimum.

2 Usage

The final solution is given in the "sols" folder named as final-solution.py. The required packages are mentioned in it.

Run the Python file to get the plot of the estimated curve and also the minimum of that function and for what values of C and R.

There are also multiple test Python notebooks that I made in order to test our methods of Naive queries, Gaussian Regression, etc. Those are also given in the same folder.

All this can be found at the GitHub link:
`https://github.com/sudo-boo/Azeotropy-optimizer`.
There are multiple images that help in visualization of the regressions.

3 Methodology

3.1 Modeling the Objective Function

We model the objective function using Gaussian process regression, which provides a distribution over possible functions that could describe the data. This allows us to quantify uncertainty and make informed decisions about where to sample next.

So, Gaussian Process is just a way to fit the unknown curve (given some or very few data points)

$$\begin{pmatrix} f(\mathbf{x}) \\ f(\mathbf{x}_*) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(\mathbf{x}) \\ m(\mathbf{x}_*) \end{pmatrix}, \begin{pmatrix} k(\mathbf{x},\mathbf{x}) & k(\mathbf{x},\mathbf{x}_*) \\ k(\mathbf{x}_*,\mathbf{x}) & k(\mathbf{x}_*,\mathbf{x}_*) \end{pmatrix} \right)$$

The mean function $m(\mathbf{x})$ represents the prior belief about the function values, and the covariance function $k(\mathbf{x}, \mathbf{x}')$ governs the smoothness and correlations between different points in the input space.

Given each values, for plotting the function, the GP calculates the value (predicted mean μ_* and predicted variance σ_*^2) for discrete points as:

$$\mu_* = \mu + \mathbf{K}(x_*, x)(\mathbf{K})^{-1}\mathbf{y}$$
$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T(\mathbf{K})^{-1}\mathbf{k}_*$$

where,

- μ_*: Predictive mean at test point x_*.
- $\mathbf{k}_*$: Kernel vector between x_* and observed data.
- $\mathbf{K}$: Kernel matrix from observed data.
- σ_n^2: Noise variance parameter.
- $\mathbf{y}$: Vector of observed target values.
- $k(\mathbf{x}_*, \mathbf{x}_*)$: Kernel function evaluated at x_* with itself.

Since, we assumed no noise, the (K) kernel function would simply had been $(\mathbf{K} + \sigma_n^2\mathbf{I})$.

3.2 Selection of kernel function

The kernel function is basically what decides the criterion for the shape of the predicted function, its smoothness and the confidence intervals.It exactly controls the behavior of the function we need to predict.

In our case, we needed to find the minima of the function, hence, I chose to select RBF (Radial Basis Function): $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$

where, σ^2 is the changeable width parameter for the variance at the points of prediction.

3.3 Acquisition Function

An acquisition function is a heuristic that guides the selection of the next point to evaluate. In this code, the expected improvement is used as the acquisition function. It quantifies the potential improvement over the current best observation and weighs it by the uncertainty in the prediction.

The goal is to find the point with the highest expected improvement, indicating that it has the potential to lead to a better solution.

To find the minima, we need to find both types of points where the :

- variance is very high **(exploration)**
- mean is to be calculated accurately **(exploitation)**

We can formulate this by considering an acquisition function on which the Expected Inprovement is calculated which gives us the next best possible values for which the data point is to be calculated to gain maximum information or improvement. One such example for finding the minima is:

$$\mathrm{EI}(\mathbf{x}) = E\left[\max(f(\mathbf{x}) - f(\mathbf{x}^+), 0)\right]$$

3.4 Iterative Optimization

The optimization process is iterative. We start with some initial data points and train the Gaussian process model. Then, we iteratively select new points to evaluate using the acquisition function, evaluate the objective function at these points, and update the model with the new data. This process continues until a stopping criterion is met (in our case the number of data points hits max), such as reaching a maximum number of epochs or convergence to a satisfactory solution.

3.5 Visualization

Finally, we evaluate the optimized solution by predicting the minimum value and corresponding variables using the trained GP model.

We recursively find new best values and plot the fitted model. We choosing the values where confidence interval is high and also where the function is minimum. There we follow a trade-off between those two values.

For a general case you can assume, given enough data points, the model can be plotted in 2D by recursive estimation considering the prior points. The figure given is an example of that

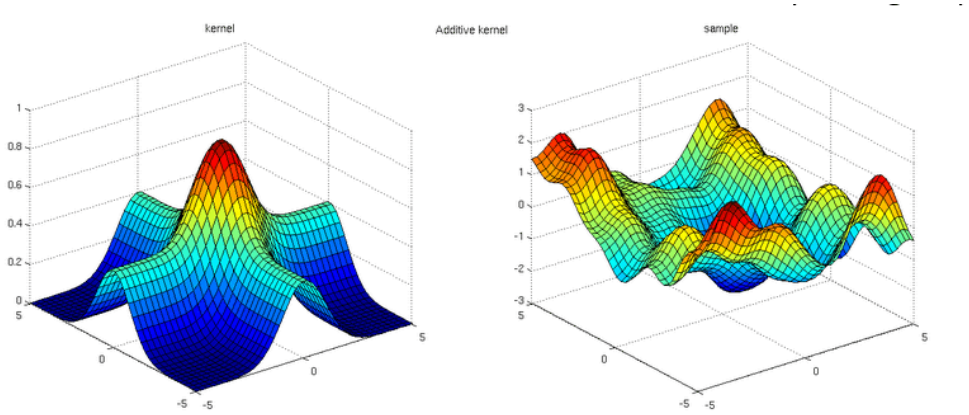

Figure 1: Curve fit after 6 data points

In our given figures, the predicted curve is given and the 95% confidence interval is also represented by light-blue layer which is same as the range $\mu \pm 2\sigma$.

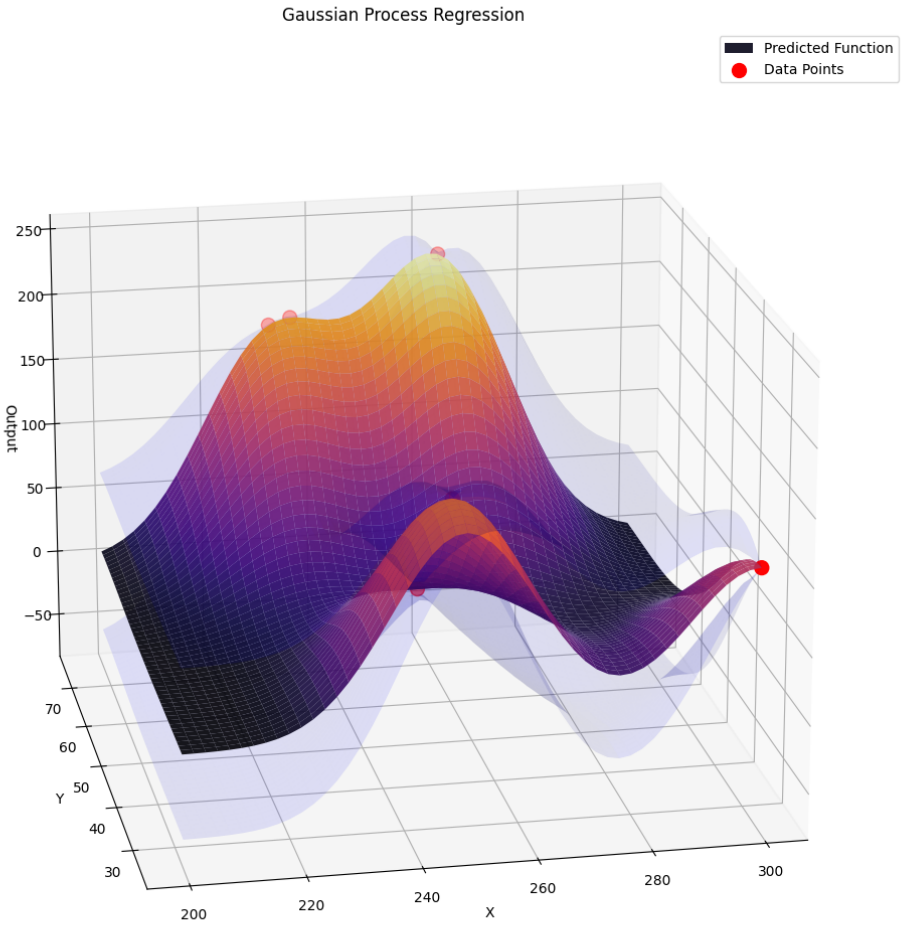

Figure 2: Curve fit after 6 data points

- For the first 6 points, the model focused on the exploration part, as seen, considering sparse and endpoints of the data ranges.

Note: The two close points $(240, 75)$ *were due to the reason that the model was restarted and due to randomness, it gave slightly different value for which the value is calculated*

- After 8 points the model found the maxima at $(300, 75)$ and as you can be seen, no other points were searched nearby that. This is a proof that the model was working.

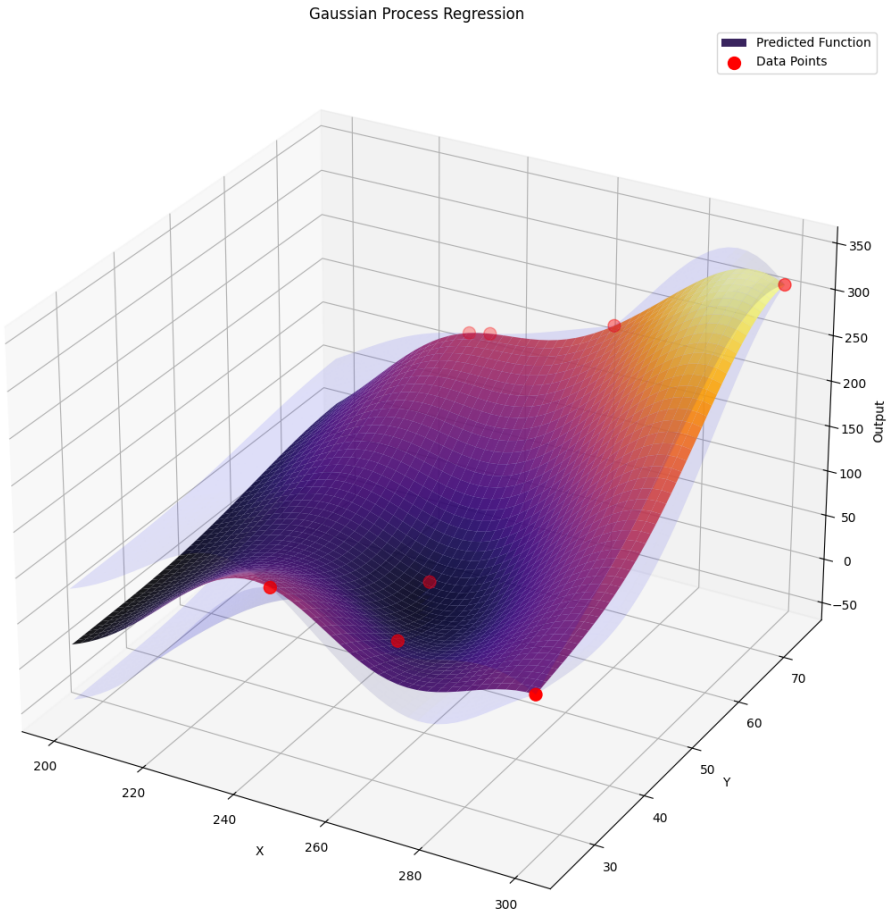

Figure 3: Curve fit after 8 data points

The bendiness to the curves was due to the difference in the initial consideration of the mean while starting the GP. I assumed it to be 0 *but the actual graph had minima at* 32*. But there was n way of knowing it so can't do anything about it.*

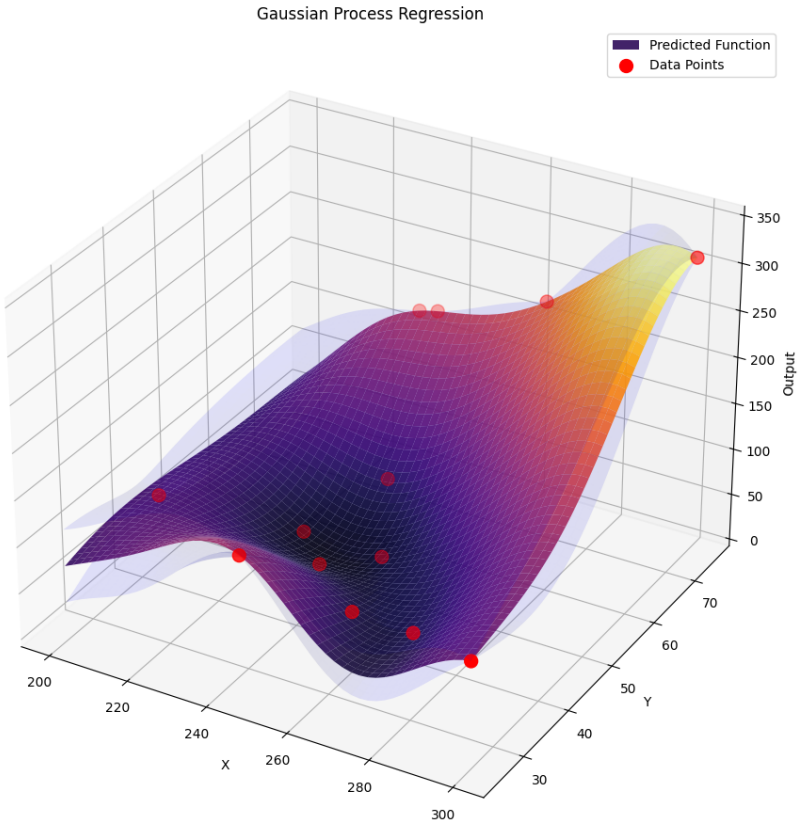

Figure 4: Curve fit after 11 data points

- After 11 predictions, it started focusing on the minima of the function in the ranges $[220, 270]$ and $[35, 65]$. While also focusing on the corners points but with less weightage.

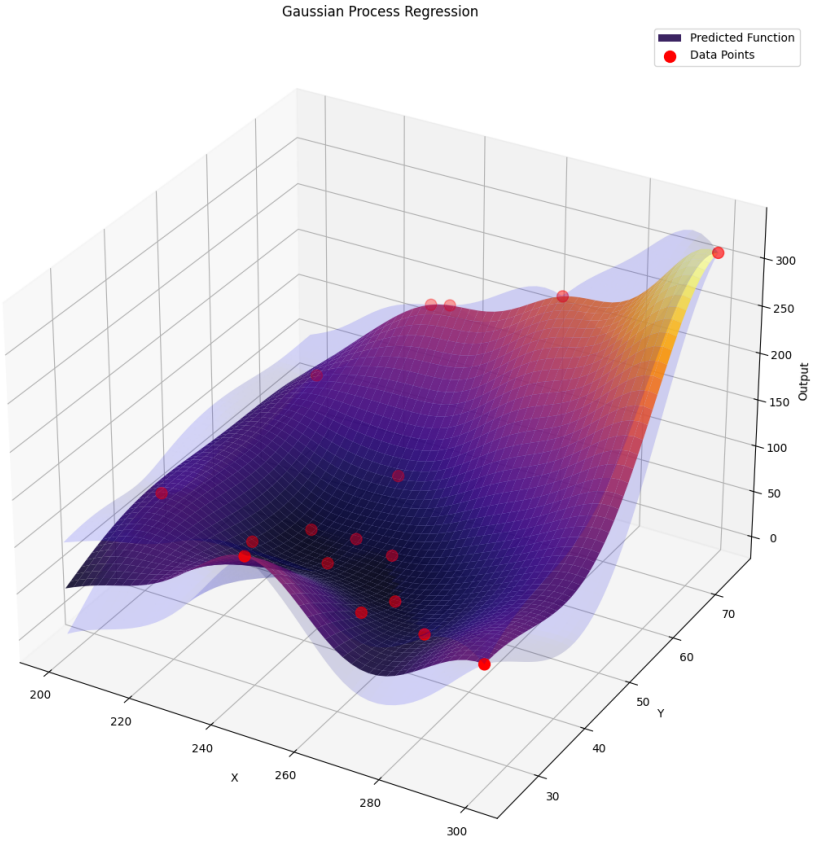

Figure 5: Curve fit after 16 data points

- Most of the values here are now saturated in the region of minima and the confidence interval can also be seen to minimize and the search in that area can be stopped to get the minima.

Note: I tend to round up the suggested points for the sake of simplicity and ease. As I was sure that the minima won't me missed as we are calculating the minima for the function by calculating mean of the GP model which was pretty accurate in the minima region.

3.6 Result Calculation

Table 1: Data Points

C	R	Values
250	50	34.9326
263.4400959	75	223.5943
231.7284842	75	176.11
244.3812422	25	143.2747
235.7397	75	181
300	25	107.794
300	75	311.9724
260	35	57.69
230	50	36.7465
205	40	89.3282
240	45	36.5198
240	60	63.5974
275	35	55.3876
240	51	35.1882
210	68	102.1441
225	42	53.8587
260	42	36.3241

The minimum value and corresponding input variables represent the optimized solution found by the Bayesian optimization algorithm.

This here in our case is calculated by forming multiple random Gaussian curves formed by our trained model within the 95% confidence interval, then taking its mean and choosing the values where the value of those mean of the functions is minimum.

In summary, the code implements Bayesian optimization using Gaussian process regression to find the optimal solution for a given objective function within a defined search space. It iteratively refines its estimate of the optimal solution based on the observed data points and the probabilistic model of the objective function.

4 Results

The values for which I got minimum were:

$$C = 253.76884422110552$$

$$R = 46.35678391959799$$

And the minimum cost found was:

$$T = 32.433675076245216$$

These values can also be seen when you run the *final-solutions.py*.

It is highly suggested to run the code as it is more intuitive that just the description of confidence intervals and the prediction of the data points.

5 Other Approaches

Some of the other approaches that I considered for this project but discarded due to some reasons. Those were:

- **Polynmial Regression** - Less data, more complexity for higher dimensions, form of predicted function not known.
- **Genetic Algorithm** - First choice earlier. Could be used if limit for data points was 50, though simple to implement, may escape minima or maxima if not chosen correctly.
- **Naive Tests** - Trail and Error. No metrics so no reason to use in practically complex cases.
- **Equal Intervals** - Highly likely to escape the minima.

6 References

- https://see.stanford.edu/materials/aimlcs229/cs229-gp.pdf
- https://en.wikipedia.org/wiki/Bayesian_optimization
- https://www.youtube.com/playlist?list=PLE6Wd9FR--EdyJ5lbFl8UuGjecvVw66F6