Gaussian Processes for Episodic Experimental Design

Ari Fiorino AFIORINO@ALUMNI.CMU.EDU
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Ojash Neopane ONEOPANE@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Aarti Singh AARTI@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract

We present a method of finding the optimal series of actions in an experimental episode. An episode consists of a series of contexts, a series of actions, and a series of rewards. Within an episode, the rewards are only shown after the actions have been made. It is assumed there is a delayed effect where an action will only affect a certain amount of rewards that come after it. Our algorithm approaches this problem setup using Gaussian Processes. We present experiments on synthetic data that show our algorithm has much higher performance than suboptimal algorithms with weaker assumptions. We also present two greedy versions of our algorithm with much lower runtime asymptotically, and near optimal performance. Finally our algorithm was tested on a COVID-19 dataset to predict future Coronavirus deaths from current Coronavirus cases, and it still performed optimally.

Keywords: Gaussian Processes, Additive Manufacturing, Coronavirus, Time Series Optimization, Delayed Rewards

1. Introduction

When performing real world experiments, one first chooses a set of experimental parameters, then performs the experiment, then observes the results. We call this three step process an experimental "episode". When trying to acheive a desired experimental result, one often goes through several episodes. In automating the episodes, the conceptually difficult step is the first one: which experimental parameters do we choose next given the data from past episodes? Our goal is to be as data efficient as possible: we want to achieve the best results in the fewest episodes.

There is a special case of experimental episode that has not been explored before, which is the focus of this paper. In this setup, the experimental parameters are a time sequence of actions to be performed by a robot. The experimental result is a corresponding time sequence of rewards observed by the robot after the actions have been made. There is a relationship between the time sequence of actions and the time sequence of rewards. Each reward depends on the corresponding action performed at that time step and the previous k actions performed, for some k.

This type of experimental episode is intuitive. If the robot makes a certain action at a given time step, it should affect the rewards for the next k time steps, nothing before or after that. This type of episode assumes the robot has no sensors that can be used while it is performing the experiment. This is often the case in practice, for example when observing the result can change the result itself. Also, there are often separate machines for performing the experiment and observing the results.

There are many problems in the real world that follow this framework. In an additive manufacturing setting, the robot extrudes a sequence of amounts of a certain material. Once the object has been created, a sensor scans the resulting object and determines how well each section was made. Using the time sequence of extrusion amounts and the time series of rewards, the robot decides the next sequence of extrusion amounts to pick.

In this paper, we explore this episodic problem formulation and approach it using Gaussian Processes. We give an optimal algorithm, compare it to non-optimal algorithms, and give a time efficient approximation to the optimal algorithm. We show experiments performed on synthetic data and a COVID-19 dataset. Our algorithm is used to predict future Coronavirus deaths based on current Coronavirus cases.

Related Work. There are several papers using Gaussian Processes that are similar to our problem formulation but differ in major ways. In the batch GP method described by Kathuria et al. (2016), there is a function we want to optimize. We can pass several actions to that function at once, in a batch. Then we receive the rewards corresponding to each action. In batch GP, the rewards only depend on the corresponding action. But in our definition of an episode, each reward depends on the previous k actions as well.

Bogunovic et al. (2016) describes a method of using Gaussian processes to optimize a time varying function. They optimize this function continually as it changes and have no opportunity to go back and try again from the beginning. By contrast, our formulation has multiple opportunities to optimize the same time varying function. Our formulation optimizes the time varying function at every episode and learns from past episodes as well.

Our problem formulation is the combination of both of these papers. The actions are sent in a batch, but the actions taken can have a delayed effect on the subsequent k rewards as well.

The area of Reinforcement Learning (RL) as outlined in Sutton and Barto (2018) is different from our setting as well. In RL there is a state announced at each round. An action is made for the state and then a reward is returned for that state-action pair. The state in RL is similar to the context in an episode. But in our case, the rewards are not announced instanteneously after each action is made. They are only announced after T rounds. One could have the sum of all the rewards announced every T rounds. But this would be suboptimal as it would not have the time dependencies included.

2. Background

2.1 GP-UCB

Say we want to maximize a function $f : \mathbb{R}^n \to \mathbb{R}$ by iteratively making guesses. For this we use the Gaussian Process upper confidence bound (GP-UCB) algorithm from Srinivas et al. (2009). First, a Gaussian Process will input training points for f, and output two functions. $f_\mu : \mathbb{R}^n \to \mathbb{R}$ is the approximation of f, and $f_\sigma : \mathbb{R}^n \to \mathbb{R}$ is the uncertainty of its prediction. Using these two functions, the GP-UCB algorithm will choose this as its next point:

$$\hat{x} = \arg\max_x f_\mu(x) + \beta f_\sigma(x)$$

This method picks a point with a high expected label, but also one with a high variance, as it might yield a high label. β is the exploration-exploitation tradeoff parameter.

2.2 CGP-UCB

There is a modification of GP-UCB described in Krause and Ong (2011). At every round, a context $c \in \mathcal{C}$ is announced, and the algorithm needs to optimize $f(c, \cdot)$. The proposed solution is very similar to GP-UCB but incorporates the context as well. We use past training data to approximate $f : \mathcal{C} \times \mathbb{R}^n \to \mathbb{R}$ with $f_\mu : \mathcal{C} \times \mathbb{R}^n \to \mathbb{R}$ as the mean and $f_\sigma : \mathcal{C} \times \mathbb{R}^n \to \mathbb{R}$ as the variance of our prediction. Then our best guess to maximize $f(c, \cdot)$ is

$$\hat{x} = \arg\max_x f_\mu(c, x) + \beta f_\sigma(c, x)$$

3. Problem Statement

Here we formalize our episodic problem setup. At each episode,

1. A series of contexts $c_0, \ldots, c_T \in \mathcal{C}$ are announced.

2. A series of actions $a_0, \ldots, a_T \in \mathcal{A}$ are decided upon based on the contexts.

3. A series of rewards $r_k, \ldots, r_T \in \mathbb{R}$ are measured.

Each of these steps happens one after another, i.e. there is no overlap between steps 2 and 3. k is the time dependency parameter. Our assumption is that r_i only depends on $a_i, \ldots, a_{i-k}$ and $c_i, \ldots, c_{i-k}$.

4. Algorithms

We propose several algorithms that solve the problem statement from section 3. The first one, Episodic CGP-UCB is the most intuitive, and the optimal solution from our problem statement.

E-CGP-UCB We assume there is a function $f : \mathcal{C}^{k+1} \times \mathcal{A}^{k+1} \to \mathbb{R}$ and that $r_i = f(c_i, \ldots, c_{i-k}, a_i, \ldots, a_{i-k})$. We approximate f using a GP on past data with f_μ and

f_σ. Then our update rule is

$$\hat{a}_0, \ldots, \hat{a}_T = \underset{a_0,\ldots,a_T \in \mathcal{A}}{\arg\max} \sum_{i=k}^{T} f_\mu(c_i, \ldots, c_{i-k}, a_i, \ldots, a_{i-k}) + \beta f_\sigma(c_i, \ldots, c_{i-k}, a_i, \ldots, a_{i-k})$$

Next, there are several algorithms that make weaker assumptions than our problem formulation. These are described in **CGP-UCB-ONE**, **CGP-UCB-ALL**, and **E-GP-UCB** .

CGP-UCB-ONE We assume there is a function $f : \mathcal{C} \times \mathcal{A} \to \mathbb{R}$ and that $r_i = f(c_i, a_i)$. We approximate f using a GP on past data with f_μ and f_σ. Then our update rule is

$$\hat{a}_i = \underset{a \in \mathcal{A}}{\arg\max} \, f_\mu(c_i, , a_i) + \beta f_\sigma(c_i, a_i)$$

CGP-UCB-ALL We assume there is a function $f : \mathcal{C}^{T+1} \times \mathcal{A}^{T+1} \to \mathbb{R}$ and that $\sum_{i=k}^{T} r_i = f(c_0, \ldots, c_T, a_0, \ldots, a_T)$. We approximate f using a GP on past data with f_μ and f_σ. Then our update rule is

$$\hat{a}_0, \ldots, \hat{a}_T = \underset{a_0,\ldots,a_T \in \mathcal{A}}{\arg\max} \, f_\mu(c_0, \ldots, c_T, a_0, \ldots, a_T) + \beta f_\sigma(c_0, \ldots, c_T, a_0, \ldots, a_T)$$

E-GP-UCB This algorithm is the same as **E-CGP-UCB**, but it ignores context. Therefore We assume there is a function $f : \mathcal{A}^{k+1} \to \mathbb{R}$ and that $r_i = f(a_i, \ldots, a_{i-k})$. We approximate f using a GP on past data with f_μ and f_σ. Then our update rule is

$$\hat{a}_0, \ldots, \hat{a}_T = \underset{a_0,\ldots,a_T \in \mathcal{A}}{\arg\max} \sum_{i=k}^{T} f_\mu(a_i, \ldots, a_{i-k}) + \beta f_\sigma(a_i, \ldots, a_{i-k})$$

An issue with **E-CGP-UCB** is that it has a $O\left(|\mathcal{A}|^T\right)$ runtime. Therefore we present greedy approximation algorithms **E-CGP-UCB-G1** and **E-CGP-UCB-G2**.

E-CGP-UCB-G1 In order to reduce the runtime of **E-CGP-UCB**, we greedily optimize the first element of the summation, and then each subsequent element. That is we first find

$$\hat{a}_0, \ldots, \hat{a}_k = \underset{a_0,\ldots,a_k \in \mathcal{A}}{\arg\max} \, f_\mu(c_k, \ldots, c_0, a_k, \ldots, a_0) + \beta f_\sigma(c_k, \ldots, c_0, a_k, \ldots, a_0)$$

Then we find

$$\hat{a}_{k+1} = \underset{a_{k+1} \in \mathcal{A}}{\arg\max} \, f_\mu(c_{k+1}, \ldots, c_1, a_{k+1}, \hat{a}_k, \ldots, \hat{a}_1) + \beta f_\sigma(c_{k+1}, \ldots, c_1, a_{k+1}, \hat{a}_k, \ldots, \hat{a}_1)$$

Then so on for $\hat{a}_{k+2}, \ldots$. This reduces the runtime to $O\left(|\mathcal{A}|^k\right)$.

E-CGP-UCB-G2 We first run **E-CGP-UCB-G1** and then we try changing each individual action to increase the overall UCB. We check if we can replace $\hat{a}_0$ with a different value and increase the overall UCB. Then we check $\hat{a}_1$, $\hat{a}_2$, and so on. We run multiple passes from $\hat{a}_0$ to $\hat{a}_T$ to increase the overall UCB. Overall, we still have a runtime of $O\left(|\mathcal{A}|^k\right)$.

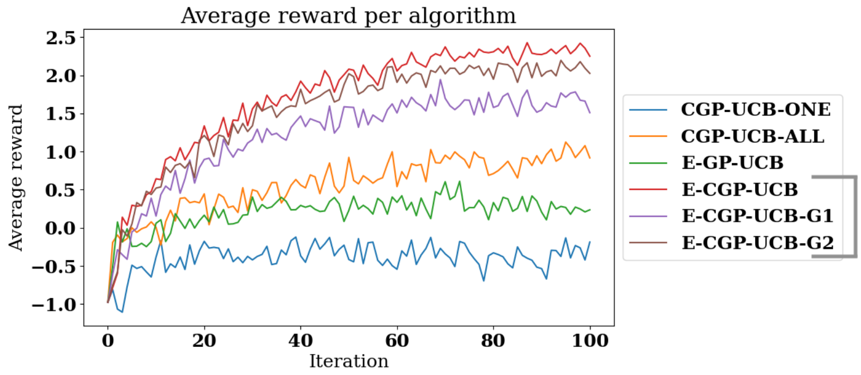

Figure 1: Comparing all algorithms on synthetic data. Algorithms designated by the grey bracket are new algorithms we have designed.

5. Synthetic Evaluation

We ran all the algorithms on synthetic data to compare their performance. We created our target function by sampling from a GP which follows our problem statement. For our experimental parameters we had $T = 4$, $k = 2$, $\mathcal{A} = [5]$, $\mathcal{C} = \{0, 5\}$. For our GP we used an RBF kernel with $\sigma = 2$. At each episode, we pick a random context. We run each of the algorithms on 100 random initialization points and plot the average reward per iteration. In Figure 1, we plot our results. The performance of the algorithms is explained below.

E-CGP-UCB This algorithm has optimal performance.

CGP-UCB-ONE has low performance because it only uses data for an individual day rather than the past k days. It will never find the optimal solution.

CGP-UCB-ALL has low performance because it is trying to optimize over a very large function. This makes it slow to find the optimal solution.

E-GP-UCB This algorithm has very low performance because it doesn't take context into account. Because knowledge of the context is necessary to solve the problem, it will never find the optimal solution.

E-CGP-UCB-G1 This algorithm approximates the optimal algorithm. It approaches optimal performance but doesn't match it since it is a greedy version.

E-CGP-UCB-G2 This algorithm is even closer to the optimal solution. It is very close to optimal and has much smaller runtime asymptotically.

6. COVID-19 Evaluation

Next, we tested our algorithms on a dataset of Coronavirus cases and deaths over time. We chose this dataset because it has a time dependency associated with it. The deaths lag behind cases by 8.053 ± 4.116 days as shown by Jin (2021). We used official Coronavirus data from the CDC. Let $p_0, p_1, \ldots$ be the cases and $q_0, q_1, \ldots$ be the deaths. At each episode,

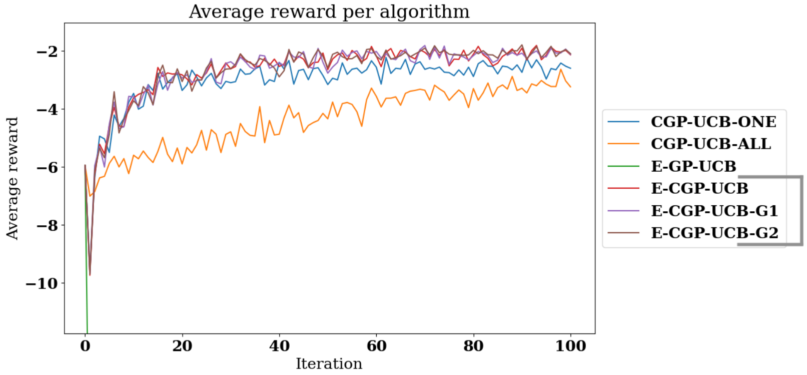

Figure 2: Comparing all algorithms on COVID-19 data. Algorithms designated by the grey bracket are new algorithms we have designed.

a random day d is chosen. The context announced is $c_0 = p_d, \dots, c_T = p_{d+T}$. Then the algorithm chooses $a_0, \dots, a_T$. Then the rewards $r_k, \dots, r_T$ are calculated as follows:

$$r_i = -\left| \sum_{j=i-k}^{i} a_j - q_{j+8} \right|$$

This encourages the algorithm to pick the correct deaths $q_{d+8}, \dots, q_{d+8+T}$. Therefore the algorithm ends up predicting future deaths. The results are shown in Figure 2.

For this experiment, **E-GP-UCB** is off of the graph. It has very bad performance because it ignores context, in this case ignoring the number of COVID cases. As is the case with the synthetic experiments, **CGP-UCB-ALL** is very slow as it is optimizing over a large function. **CGP-UCB-ONE** performed suboptimally because it only takes one case into account when predicting one death. The greedy approximations perform optimally.

7. Conclusion

We have presented a method of optimizing a series of actions over a series of rewards, where the actions have an unknown, delayed effect on the rewards. We showed our optimal algorithm has much higher performance on synthetic and real world data when compared with suboptimal algorithms with weaker assumptions. The optimal algorithm has a large runtime asymptotically, so we created two greedy approximations. These approximations perform close to optimal and are much faster to run. For future work, we would like to compare the algorithms in an active learning scenario.

Acknowledgments

This research is supported in part by the AFRL Data-Driven Discovery of Optimized Multifunctional Material Systems Center of Excellence (D3OM2S CoE) grant.

Appendix A.

The full implementation is available here: https://github.com/arifiorino/Modified-GP

References

Ilija Bogunovic, Jonathan Scarlett, and Volkan Cevher. Time-varying gaussian process bandit optimization. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 314–323, Cadiz, Spain, 09–11 May 2016. PMLR. URL https://proceedings.mlr.press/v51/bogunovic16.html.

Raymond Jin. The lag between daily reported covid-19 cases and deaths and its relationship to age. *Journal of Public Health Research*, 10(3):jphr.2021.2049, 2021. doi: 10.4081/jphr.2021.2049. URL https://doi.org/10.4081/jphr.2021.2049. PMID: 33709641.

Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/a1d7311f2a312426d710e1c617fcbc8c-Paper.pdf.

Andreas Krause and Cheng Ong. Contextual gaussian process bandit optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/f3f1b7fc5a8779a9e618e1f23a7b7860-Paper.pdf.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Gaussian process bandits without regret: An experimental design approach. *CoRR*, abs/0912.3995, 2009. URL http://arxiv.org/abs/0912.3995.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.