Visualizing Bayesian Optimization for Teaching

William M. Temple*
University of Colorado Boulder

ABSTRACT

Effectively teaching machine learning, with its high cognitive load, demands exceptional teaching utilities. Often, students find abstract discussions or overly-formal mathematical descriptions either difficult to interpret and understand or insufficiently applicable. In this article, I explore the application of data visualization techniques to Bayesian Optimization (Gaussian Processes) and describe the tasks that students engaged in hypothetical study of this model would likely perform. We further develop some prototype user-interfaces and address technological challenges essential to the development of an interactive system for studying Bayesian Optimization.

1 INTRODUCTION

Educational technology systems present interesting design challenges when analyzed using a data-visualization lens. Students have different goals and desires and separate motivations from data analysts. For better or worse, much of the discourse about data visualization in our coursework and in the literature we have read for class has focused on the consumption of data by analysts—or, at least, by users (even untrained or novice users) *performing* an analytical role, where "analysis" is defined within a narrow precept.

However, as we have a broad interest in the development of educational tools for novice Computer Science and Programming students, we always try to consider the ways in which the data visualization techniques that we have discussed apply in the classroom to enhance the educational experience and provide for a more robust pedagogy.

Towards this goal, in this project we have focused on building a tool to analyze the state of a machine learning algorithm. In particular, the system shows the execution of a Bayesian Optimization using a Gaussian Process Regression. The system allows a curious user to examine the state of the model and shows not merely the *result* of the application of the model (which would be the primary interest of users in analytical roles), but also makes evident the choices that the model makes during execution and empowers the user to understand the mathematical reasoning behind those choices.

1.1 Why Gaussian Processes?

The Gaussian Process is an advanced stochastic process which uses a suite of functions and strategies for optimizing those functions to create estimations of the mean and variance over a distribution of other functions[1]. That is to say, it is relatively complicated.

However, a computer can execute Bayesian Optimization in a small series of abstract steps:

1. Initialize the Algorithm by sampling a small number of random points from a true function f and fitting an initial regression using a *kernel* function.
2. Use an *acquisition* function to choose a point which is likely to improve the estimated mean of the process.
3. Sample that point and add it to the regression.
4. Repeat until the model converges on an accurate estimation of f.

That is to say, the Gaussian Process isn't so complicated that it evades an effective decomposition by way of visualization. It also has certain essential properties that we believe make it particularly suited to a visual approach to teaching its concepts. We discuss these properties further in Section 3.

2 RELATED WORK

In this section, we discuss some existing literature which informs our design choices and our thinking about how visualization and task modeling in particular relate to the goal of education. We also include some discussion of existing education literature which we feel aids our explanation of why visualization provides an augmentation to the learning environment.

2.1 Teaching as Presentation

While many forms of visualization cater to analytical interests, in teaching we wish to present information rather than analyze it. As instructors, we want to integrate knowledge into our courses in such a way that it is "memorable" and "engaging." Robert Kosara describes *Presentation-Oriented Visualization Techniques* in exactly those words, then he further constrains presentation-oriented techniques: "presentation-oriented techniques need to be specific rather than general and compact rather than scalable" [7]. In other words, a presentation does not need to account for the countless ways that a user may want to use it, as its primary purpose is to convey the information that the author (or presenter) desires. In teaching,

As teachers (presenters), we attempt to anchor information into our students' minds. "To get people to actually pay attention to a presentation, they need to find the views interesting and engaging... Learnability", Kosara declares, "is part of this" [7]. If we take Kosara's concerns into account, then we should aim to build a tool that (1) clearly shows the state of our Gaussian Process, (2) does not necessarily generalize to accommodate arbitrary inputs, but (3) sufficiently accommodates student input so that they can explore the visualization an an engaging way. By retaining control over the system that the Gaussian Process is modeling in our examples, but by enabling the student to explore that model at their own pace, I believe that our tool can hold that balance appropriately.

2.2 Task Modeling for Learning

Both instructors and students will engage with our visualization, but both roles have different needs and different tasks they wish to perform. The instructor is a presenter, while the student explores the visualization to uncover the model's inner-workings. We can use a visualization task taxonomy to describe the ways that students will interact with our tool.

We explored two such taxonomies. While the language in *A Multi-Level Typology of Abstract Visualization Tasks* is certainly rich and descriptive for traditional analysis tasks [2], we do not find a student's particular type of learning-driven analysis reflected in the tasks modeled by this system. Instead, the more-granular task representations specified in *A Design Space of Visualization Tasks* seems more suited to our applications [8].

*e-mail: William.Temple@colorado.edu

[1]Prof. Nando de Freitas's video lecture at the University of British Columbia on the subject of Gaussian Processes contains the requisite background knowledge [4]

For the student, specifically, the type of analytic comparison that a student will perform using this visualization is natural to express using the notation that Schulz *et al.* suggest:

*[exploratory, comparison, *, structural, single]*

Or, in other words, students are *exploring* the data, attempting to learn something about the system generating it. They *compare* all of the features (*) at one time-step to what they observed previously, noting change and comparing the *structures* that arise in one view of the model to the structures that arise in another. And, finally, as Kosara suggests, the analysis is focused on a *single* instance of this problem.

2.3 Knowledge Construction

In Constructivist education paradigms, the presentation of examples is essential to the development of accurate knowledge. Mordechai Ben-Ari describes Constructivism as "a theory of learning, which claims that students construct knowledge rather than merely receive and store knowledge transmitted by the teacher" [1]. We agree with this assessment, as such the function of our visualization is to facilitate model-building. Ultimately the student learns of their own accord, and our role as visualization designer is to provide a site for that learning and in which that learning might be enriched.

Several studies have attested to the efficacy of worked-examples practice, in which students observe and attempt to understand the process that leads to the correct answer to a proposed problem [6]. Our visualization bears similarity to worked-example practice, as it shows the process by which an approximate estimation is derived, rather than merely its result. We believe that this form of visualization will guide students to a correct interpretation of Bayesian Optimization based on the effectiveness of similar example styles that have been deployed in algebra classrooms [3].

3 Design

Due to time and resource constraints, as well as considerations about scope listed in the Related Works section above, we decided to restrict the kinds of functions available to the user and focus on creating a compelling visualization for one configuration of Bayesian Optimization. The configuration we ultimately chose utilizes the Upper Confidence Bound (UCB) acquisition function (with a κ parameter of 20), a Matérn kernel function (with a ν parameter of 2.5), and a fixed true function $f(x,y)$ composed of a linear combination of x and y as well as seven Gaussian Distributions (with random means and standard-deviations).

We chose these parameters qualitatively, based on our assessment that this configuration produced the most obvious, evident, and correspondent effects in the display, so that we would have a higher chance of engaging the user with a noticeable effect. Several configurations of the model do *not* result in interesting or memorable effects, so—in line with Kosara's reasoning given above—we fix the parameters to this known-good state.

3.1 User Experience

Early in the design process, we settled on four coordinated views for the interface. One for each of the algorithm's essential features:

- **Top-Left**: a color-mapped graph showing the output of the true function $f(x.y)$
- **Top-Right**: a color-mapped graph showing the estimated mean of the Gaussian Process Regression
- **Bottom-Left**: a color-mapped graph showing the value of the UCB acquisition function
- **Bottom-Right**: a scatter-plot showing the set of points which have already been sampled and incorporated into the regression

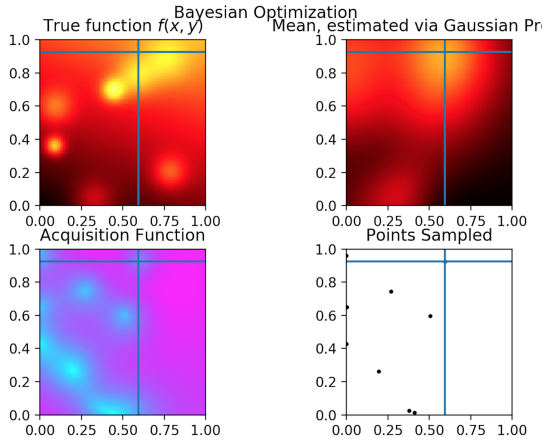

Figure 1: An example of the Python prototype interface after 9 iterations.

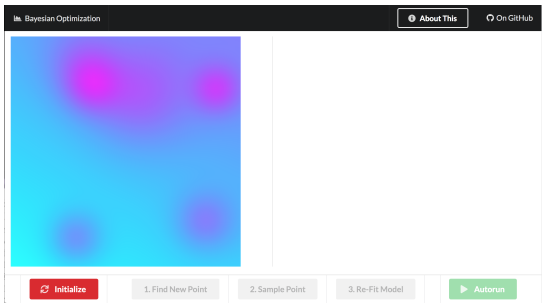

Figure 2: The prototype React UI with an example plot displayed

The plots in our visualization are coordinated. On each iteration of the model, they all update simultaneously to reflect the most immediate change to the model. Each plot displays crosshairs (an intersecting vertical and horizontal line) that denote the location of the most recent point added to the model's regression (Figure 1). We implemented the Bayesian Optimizer engine and prototype interface (which, in these examples, is merely a window into the model state) in Python using `numpy`, `scipy`, and `matplotlib`.

3.1.1 React.js Concept UI or, Why Not Python?

We found it very difficult to attach User-Interface elements to any `matplotlib` elements, and in general we had little success developing a UI in Python. This difficulty (the unsuited-ness of Python to the engineering task of developing a rich User Interface) was a primary motivator in our decision to pursue an alternative web-based UI.

We developed a concept UI in React.js (Figure 2), using the Semantic UI toolkit, that allowed us to rapidly build a rich web interface. However, the web platform comes with its own challenges. While libraries to perform Bayesian Optimization are readily available in Python with copious tutorials and examples, on the web, we would have to implement our own Bayesian Optimization library in JavaScript from the ground up. In addition, due to modifications we made to use the GPU as a co-processor (more on this below), we would additionally require a GPU-friendly implementation of the Gaussian Process Regression. We believe this is possible with a bit more effort. However, due to time and work constraints, our project remains in two pieces: a conceptual UI with no Bayesian

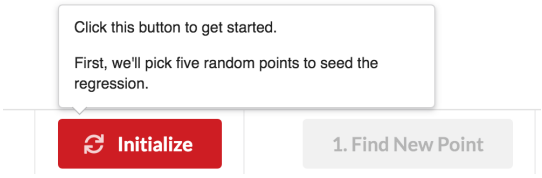

Figure 3: The mouse-hover tooltip which instructs the user how to proceed

Optimization engine, and a Pythonic Bayesian Optimizer with no *true* user interface.

The Concept UI includes an anchored toolbar on the bottom of the screen. Users can use its five buttons to navigate the state of the model. At any time, a user can press "Initialize" to restart the system, and a mouse-over tooltip informs the user that this option is available (Figure 3). The subsequent buttons can only be activated after the model has been initialized successfully. From there, the buttons guide the user through the process of choosing a new point to add to the regression and incorporating it into the model.

3.2 Technical Challenges

Early in the prototyping stages of this project, we realized that computational complexity would pose significant challenges to the interactivity of this software and its usability. The first Python prototypes of the software could only compute and render one iteration of the Optimization algorithm about every 25 seconds. Through some code optimization effort, we managed to reduce the runtime per iteration to 7 seconds, but it became clear during this process that further optimization would require a shift in development paradigm.

At this latency, the interactivity of the system is threatened, as users will easily outpace this latency. As students begin to use the software, it takes considerably less than seven seconds to digest the information divulged by a single iteration of the model. Rudimentary profiling of the code reveals that it spends the vast majority (90%) of its time drawing the four coordinated plots, as each plot iterates over each pixel in its image in a single-threaded loop. Even drawing only one of the plots requires about two seconds using Python, where a single plot consists of ten thousand points (one hundred points in both the X and Y directions).

As a proof of concept, in order to solve this problem, we wrote GPU fragment shader programs using the GPU.js library [5]. This library dynamically recompiles a restricted subset of JavaScript into GLSL shader code, which then runs on the SIMD processors in the host machine's Graphics Processing Unit, resulting in much faster execution. Furthermore, this shader code deposits resulting textures directly into an HTML `<canvas>` element, so that the browser can easily render it. We integrated these fragment shaders into our concept UI application, and measured that the same machine could render a function of the same complexity as those in our Python program at several times the resolution (2048 points in both the X and Y directions, or about four megapixels) in about a tenth of a second [2].

4 Discussion

We believe that this approach to visualizing Bayesian Optimization (described above) is well-suited to the tasks of teaching and learning. By presenting the components of the model as coordinated views into the same Gaussian Process, we force the association between these different components. We *require*, as a matter of visual proximity,

[2]Test machine: 6-CPU Intel i7, 16GB Memory, NVIDIA GeForce GTX 960 Graphics Accelerator

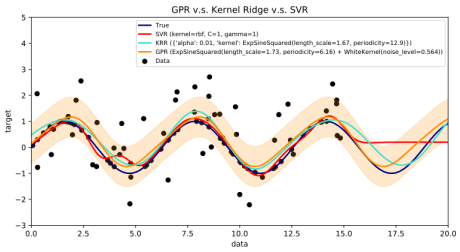

Figure 4: A typical example of a Gaussian Process visualization, showing sample points, and the estimated means of several different kernel functions. Image Shiyu Ji 2017, CC-BY-SA 4.0

that changes to one of the model's properties (or views into its properties, in this case) coincide. The visual coincidence of these phenomena models the actual mathematical coincidence in which the model itself is a mathematical structure, where the many views of the system are all projections of the model state.

There are several visual systems online for Gaussian Processes modeling functions of one variable only. These visualizations tend to overlay, rather than coordinate the multiple views into the model state (Figure 4). While these kinds of overlaid visualizations condense the area required to display this information, and though analysts as well as statisticians may find them useful, we reasoned that the separation of the true function $f(x, y)$ and the estimated mean would result in the most effective and obvious comparisons between the two. The acquisition function and sampled point plots inform both each other and the user of the context on which the decision to choose a new point is based. This facilitates the core aspect of teaching in Constructivist paradigms: effective model building[3].

4.1 Prototype Artifacts

Ultimately our work on this project produced two primary artifacts. First, an engine, capable of performing Bayesian Optimization and displaying its state using a rudimentary UI (a glorified slide-show), albeit slowly. This program is located within the `proto-numpyengine` subdirectory of our project repository.

Second, we have developed the beginnings of a rich and modern User Interface, with considerations made for how its design effectively encourages students to utilize it in the pursuit of learning-driven data-visualization tasks. We made some significant technological improvements to the process by which plot images are rendered in this version of the tool, and, although it is only capable of rendering static functions (like those described in Section 3 above), it is (based on back-of-the-napkin math from several tests) approximately **five-to-eight thousand times** faster than the basic implementation of this routine in Python. This prototype interface is located within the `proto-webui` subdirectory of our project repository.

5 Conclusion

Were we to begin this project again, knowing what we know today, we would not even attempt to rehabilitate, refactor, or otherwise restore the Python implementation. Were time permitting, all further engineering effort would be dedicated to the integration of the subroutines necessary to perform Bayesian Optimization into our web-based Concept UI.We feel that this project has provided a unique opportunity to explore various technologies (such as shader programming) and their applications towards data visualization. As

[3]**Author's Note**: the design and implementation of this tool helped this author understand Bayesian Optimization and Gaussian Processes, which are the subject of some of the author's coursework

well, it has enabled us to explore data visualization through a less-commonly appreciated lens.

REFERENCES

[1] M. Ben-Ari. Constructivism in computer science education. *Journal of Computers in Mathematics & Science Teaching*, 20(1):45 – 73, 2001.

[2] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385, Dec 2013. doi: 10.1109/TVCG.2013.124

[3] W. M. Carroll. Using worked examples as an instructional support in the algebra classroom. *Journal of educational psychology*, 86(3):360–367, 09 1994.

[4] N. de Freitas. Machine learning - gaussian processes.

[5] GPU.js Contributors. GPU.js. `https://github.com/gpujs/gpu.js`, 2018.

[6] P. A. Kirschner, J. Sweller, and R. E. Clark. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, 41(2):75–86, 2006.

[7] R. Kosara. Presentation-oriented visualization techniques. *IEEE Computer Graphics and Applications*, 36(1):80–85, Jan 2016. doi: 10.1109/MCG.2016.2

[8] H. J. Schulz, T. Nocke, M. Heitzler, and H. Schumann. A design space of visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375, Dec 2013. doi: 10.1109/TVCG.2013.120