Visualizing Interactive Gradient Descent

Nova S. Zhang*
MBAn Candidate

Dao Ming Lee†
MBAn Candidate

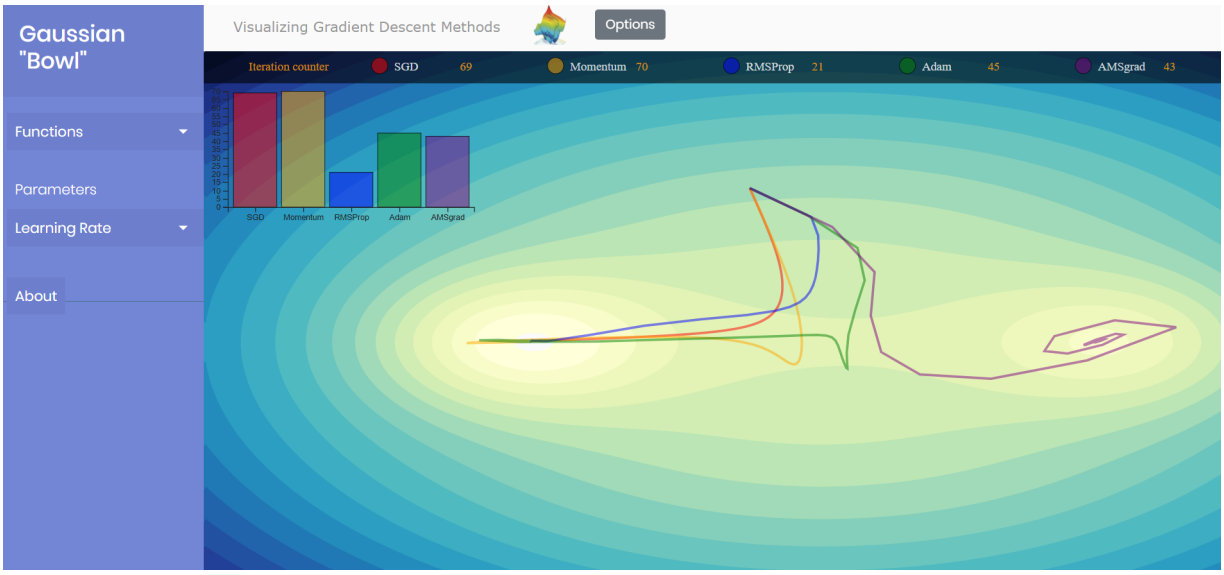

Figure 1: Going downhill fast?: A demonstration of the gradient descent visualization

Abstract

.

1 Introduction

Constrained optimization seeks to minimize a cost function within its feature space. From an initialization point, gradient descent algorithms move iteratively in directions according to the descent given until the algorithm converges to a local minimum.

For instance, in machine learning, this underlies the process in which model parameters are updated. We see this in coefficients for linear regression, and weights and biases in neural networks. The choice of which gradient descent method and its hyperparameter values can determine if cost minimization is likely to be achieved in an appropriate amount of time.

In practice, black-box optimizers are often used arbitrarily or without thorough consideration. Hence, understanding gradient descent methods and how it affects factors such as time to convergence and optimality becomes integral. This visualization seeks to close that gap by providing an interactive medium for users to utilize and self-educate on the basics of gradient descent.

2 Goals

- Apply: Allows the user to perform hyperparameter tuning (e.g. learning rates) and gradient descent method selection on different feature spaces
- Educate: Allows the user to learn about gradient descent in an interactive manner. To non-technical audiences, gradient descent lends itself most aptly to visualization. The ability to follow the respective path taken by each algorithm is integral to understanding related phenomena (e.g. oscillation), and being able to specify different initialization points allows users to experience how functions

*e-mail: novaz@mit.edu
†e-mail: daoming@mit.edu

3 Related Work

Ruder (2019) [3] provides a cogent introduction to gradient descent algorithms, running the gamut from vanilla methods like stochastic gradient descent (SGD) and momentum, to state-of-the-art methods such as Adaptive Moment Estimation (ADAM). He covers the key strengths and weaknesses of each approach, which does help the user develop some intuition for algorithm selection given a particular problem. However, the uninitiated user may experience difficulties in grasping concepts without accompanying visualizations. Central ideas such as how learning rate affects convergence, or the existence of multiple local optima/saddle points that induce trapping appear to be best understood through visualization.

In examining existing gradient descent visualizations, we found that they went some way towards but ultimately fell short of our goals. A large majority of them were non-interactive, relying on the use of charts Rana (2018) [2] Saslow [4]. to demonstrate the the different methods on sample feature spaces. As a self-learning guide, these do not empower the user to experiment with these algorithms.

Of those that allow for user interaction, they featured fewer gradient descent methods, were limited to in-built functions and fixed hyperparameter values [1]. They also lacked the explicit comparisons drawn between the methods that the former had. These quantitative comparisons are a key complement to insights drawn from the visualization by confirming observations with hard data.

4 Detailed Description of System

Our visualization system comprises of 4 nested layers:

- Domain problem characterization

- Data/Operation abstract design
- Methods (Encoding/Interaction techniques & Algorithmic design)

4.1 Domain Problem Characterization

We summarize the central issue as a lack of understanding regarding gradient descent methods in associated applications, leading to sub-optimal choices and outcomes. Our proposed solution is to equip users with requisite knowledge through an interactive visualization approach.

4.2 Data/Operation Abstract Design

We begin by mapping our approach into a series of operations and low-level tasks. The system starts by exposing uncertainty in algorithmic selection. We give a brief introduction that informs users on practical advantages and limitations on each method, allowing target users to realize their knowledge gap. Then, by showing first-hand how different descent methods operate under different parameters, it goes onto facilitate hypotheses formulation by users on the effect of different user-defined parameters (e.g. learning rate, domain of feature space) on the results. This process of adjusting a parameter and seeing the difference allows for learning through an explicit cause-and-effect model. After that, the system retrieves quantitative results and presents them to confirm or invalidate hypotheses. Once users are comfortable with the interface, the system can retrieve custom input for plotting, allowing the user to trial newfound knowledge on relevant problems.

4.3 Methods

4.3.1 Encoding/Interaction Techniques

In carrying out the relevant operations, we exploited the use of encodings and interactive techniques. In reference to Fig. 1, we first used a contour plot to illustrate the feature space of the function. This allows us to translate a 3D function to 2D space, yet preserve the 3rd dimension (cost function value) through the use of color hue and intensity.

On the left, the sidebar allows for simple navigation on our web application. Clicking 'Option' allows the user to toggle its appearance, enabling navigation without distracting from the visualization centerpiece when in use. It allows users to (i) read a brief introduction to gradient descent methods, (ii) select between 3 sample feature spaces - Gaussian Bowl, Rastrigin and Rosenbrock, (iii) build their own custom feature space, (iv) set hyperparameters for the gradient descent methods (i.e. learning rate).

In the top bar, we allowed the user to (i) select relevant methods for visualization and (ii) view the no. of iterations the current method required for convergence. Selection is carried out by clicking the associated colored circles (E.g. Yellow for SGD) to toggle. Mousing over them also elicits a short tooltip description that reminds the user what it is without having to refer back to the introduction.

Next, we traced out the path of gradient descent for each method using a line. This allows the user to follow individual gradient descent paths to understand path and convergence dependency on initialization parameters for a given feature space.

Finally, to quantify the difference in the key metric - no. of iterations to convergence - we used a simple bar chart. The length of each bar reflects individual algorithmic efficacy. Put together, it allows the user to compare and choose the best performing method.

Throughout the process, we consistently associated the same color with its gradient descent method to ease cognitive load. This allows users to easily discern the paths taken by the selected methods and their associated no. of iterations to convergence.

4.3.2 Algorithmic Design

Under the hood, we coded 5 different gradient descent methods into Javascript: Stochastic Gradient Descent (SGD), Momentum, RMSProp, Adam and AMSgrad. We chose a wide variety that spans fundamental methods to more advanced techniques used in optimizers today. This serves as the optimization engine for our visualization and makes use of Math.js and loops to reflect the iterative nature of gradient descent. The system takes in user input (e.g. learning rate, variable domain/range, cost function, selected methods) and traces out the path taken by each method using this backend engine. It then collects the results (no. of iterations to convergence) and reflects them on the rectangular bar and bar chart.

5 Results

The key visualization produced by the system is the traced paths of each gradient descent method on the contour plot. The visualization allows users to not only see the path of each descent method, but also click to pick the initialization point and show the speed (number of iterations it takes to converge) of descent methods. The paths are supplemented by a bar chart to quantify the results of the iterations obtained.

6 Discussion

Through a few self-conducted trials, we find that our system achieves our stated goals of application and education.

For a user acquainted with optimization that seeks to learn gradient descent method selection for an application (e.g. machine learning), we provide a custom function section. This allows the user to input a cost function, specify domain parameters and hyperparameters, to observe how a variety of gradient descent methods perform and choose the most appropriate one.

For a user seeking to gain optimization knowledge, we anticipate a first stop at our introduction to gradient descent to gain rudimentary knowledge of the various methods. Subsequently, the user could explore how they perform on the sample feature spaces by clicking around different initialization points and observing iteration counts on the top bar/bar chart. Additionally, adjusting the default learning rates and observing the behavior examines the significance hyperparameter tuning. In all, the user would understand how (i) initialization points, (ii) learning rates and (iii) gradient descent methods affect the optimization in terms of its optimality and time to convergence.

7 Future Work

Our current visualization system achieves much of what we set out to do, but can definitely be extended and refined.

For instance, to introduce the web application, we could have an optional 'tour' on the home page, which provides a brief description of each aspect of the system. This would enable users to navigate in a more targeted manner in line with their purpose for visiting, rather than having to explore the web application.

It could be useful for users in the future to see descent methods in higher dimensions and select what sub-spaces to view. In this way, it may give users more options for functions they can view.

For the back-end optimization, the plotting of a user-specified function as a contour plot in d3 using Math.js and loops in Javascript to manually calculate the gradient descent path can require high memory usage that results in occasional browser unresponsiveness. As such, we could explore using other packages or coding languages that would more efficiently handle the memory load.

Acknowledgments

The authors wish to thank Professor Arvind Satyanarayan and teaching Assistants Nava Haghighi, Rupayan Neogy and Rishabh Chandra for sharing their knowledge and patient teaching that enabled this paper.

REFERENCES

[1] E. Dupont. Optimization algorithms visualization. July 2019.

[2] A. Rana. Why visualize gradient descent optimization? September 2018.

[3] S. Ruder. An overview of gradient descent optimization algorithms. Sep 2019.

[4] E. Saslow. Visualizing gradient descent. November 2018.