Dashboard for Multi-Armed Bandit Algorithms

By:

Surbhi Gupta

ECSE 474

Kishan Patel

ECSE 476

Supervised by:

Aditya Mahajan

McGill University

# Abstract:

There are many goals of website optimization: providing a good user experience, increasing conversion rates and increasing the rank in search engines are a few. Thus, as web site optimization is gaining importance, it becomes necessary to have a good algorithm in place and be able to evaluate the performance of this algorithm. The algorithm under consideration here is the Multi-Armed Bandit (MAB) algorithm, and the goal of this project is to create a generic dashboard that will display the results of the MAB algorithm in a visually appealing manner while providing the user with a rich set of features. The dashboard is a very useful component since raw data by itself can be very hard to interpret and make conclusions on; it is easier for the end user to observe results and judge the performance of the algorithm through charts, simulators, filters and other options. The aim is to make the dashboard as rich as possible – it should contain a lot of graphs, provide the user the ability to manipulate the data and allow the user to observe what is going on in real time. Till date, the following features have been implemented: ability to see what happens at each clock tick (what is the result of executing the MAB algorithm), observing the progress of each arm, viewing the results of each arm based on the hour of day and the ability to apply a simulator on each arm in the dataset. This is accomplished by uploading a file containing static data. In order to fully achieve the goals that were initially set, the application will be extended to support live data stream; this feature will be added next.

# Acknowledgments:

We would like to express our special thanks to Professor Aditya Mahajan, our research supervisor for his advice and encouragement during the course of this project.

Table of Contents

Abstract: 2

Acknowledgments: 3

Tables 5

Figures 5

List of abbreviations used in the report. 5

Introduction/Motivation and Goals: 5

Introduction/Motiviation 5

Goals 6

Background 6

Introduction to MAB 6

WO as a MAB Problem 7

Design and Implementation 7

Technologies 7

Currently Implemented Features 8

Future Implementations 9

Results and tests: 9

Initial Tests and Experimentation 9

Impact on Society and the environment: 10

Teamwork: 11

Workload 11

Collaboration 11

Conclusion: 11

References: 12

# Tables

Table 1: Describes commonly used terms in Multi-Armed Bandit algorithms

Table 2: Describes the MAB parameters in relation to Website Optimization

Table 3: Provides a summary of Radian.js

Table 4: Provides a summary of Rickshaw.js

# Figures

Figure 1: Describes which arm is selected at each time step and the result obtained

Figure 2: Described the rewards obtained over time

# List of abbreviations used in the report.

* CSV Comma Separated Values
* JS JavaScript
* JSON JavaScript Object Notation
* MAB Multi-Armed Bandit
* UCB Upper Confidence Bound
* WO Website Optimization

# Introduction/Motivation and Goals:

## Introduction/Motivation

With the increasing widespread of e-commerce and e-marketing, the need for optimized websites is also increasing. WOrefers to the process of affecting the visibility of a [website](http://en.wikipedia.org/wiki/Website) in order to provide it with a good rank in [search engines](http://www.webopedia.com/TERM/s/search_engine.html) as well as to make it load faster[1]. Our project arises from the need to find a more efficient way[2] to optimize websites than the currently used statistical hypothesis based classical *A/B testing*\*. A more time, cost and memory efficient algorithm will lead us to designing better web layouts and in turn: websites, leading to increased user engagement and hence higher business prospects. We make use of MAB algorithms to find the most optimal website layout and thereby optimize a website.

\*”A/B testing (sometimes called split testing) is comparing two versions of a web page to see which one performs better” [3].

There are two primary deliverables for this project that have been divided amongst two teams as follows:

1. Framework

* Implements the multi-armed bandit algorithm
* Determines which version of a particular website to display
* Keeps track of the effectiveness of each version, logging the flow of user actions and any other information that may be of interest

1. Dashboard

* Represents the results of executing a set of multi-armed bandit algorithms used for website optimization

Our group is working on deliverable number 2, i.e. building the dashboard.

## Goals

Below is a description of the goals that we are trying to achieve by building this dashboard. We want to provide the user with an interface with the following abilities

1. Uploading a file to generate bar or line charts
   * The file is generated by the framework (deliverable number 1) and it contains results of a MAB simulation
   * Most basic file contains information at each clock tick (website version or arm selected and the resulting success achieved)
   * Multiple formats supported: CSV, JSON or Tabular
2. Providing live data to generate bar or line charts
   * Automatically updating graphs of real time data
3. Viewing statistics for each arm from the bar or line charts generated by file upload and or live data, such as
   * Success at any particular instant (instantaneous)- Bar chart
   * Overall success over time (average)- Line chart
4. Viewing arm details from a line chart
   * Time when a particular arm was active or inactive
5. Viewing results by time from a bar chart
   * Performance of arms at an hourly basis
6. Applying MAB functions on a particular arm and viewing the simulation result in the form of a graph
   * Egg. of functions- UCB, Epsilon Greedy\*\*

# Background

## Introduction to MAB

In order to understand MAB algorithms, it is crucial to first familiarize with some terms commonly used in their reference and then the MAB problem itself. Table 1 describes the terms used in every bandit problem.

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Agent | User |
| Arm | Decision maker |
| Gain | Measure of success or reward |

**Table 1**

A MAB problem can be simply stated as follows[4]:

*An agent chooses (plays) an arm and receives a gain from it. How can the agent maximize his gain? Which arms should the agent play, how many times to play each arm and in what order to receive maximum gain?*

A characteristic typical of a MAB problem is the conflict between making decisions that

* Provide high current gains by *exploiting* high performing arms

OR

* Sacrifice current gains by *exploring* other arms in hopes of finding an even-better arm?

This is also known as the *exploration vs. exploitation* dilemma[5].

To summarize, the objective of the agent is to maximize the sum of rewards earned through a sequence of arms. A MAB algorithm or strategy attempts to solve this problem by looking for *optimal* or *approximately-optimal* solution. Solution in this case refers to a sequence of arms an agent can play in order to receive the maximum gain. It first exploits the highest performing arms. Next, it explores other arms to see if they perform better than the current highest performing ones.

## WO as a MAB Problem

In order to model WO as a MAB problem, the essential parameters (agent, arm and gain) must be defined. Table 2 describes the MAB parameters in relation to WO.

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Agent | User |
| Arm | Website version with unique styling  E.g. Each version has a different .css stylesheet demonstrating different   * Color scheme * Button sizes * Font sizes * Layouts |
| Gain | Effectiveness of a particular version  It can be defined as a metric of success whose definition varies across domains. E.g. Effectiveness can be equal to the   * Number of purchases of a particular item on an e-commerce website such as Amazon.com * Number of donations received on a fundraising website |

**Table 2**

# Design and Implementation

## Technologies

The MAB dashboard was implemented from scratch. As this is a web application, some of the technologies being used include: HTML, CSS, and JavaScript. Some of the JavaScript libraries used thus far include: Twitter Bootstrap, Radian and Angular.js. The biggest challenge while implementing the file upload feature was deciding on which JavaScript charting library to use. As this feature was fairly straight forward, there hasn’t been a firm decision made on which framework to use or which web server to deploy the application on. Nonetheless, we feel that while implementing the live stream feature, it would be advantageous to implement the server in Node.js as this will allow for more control of how the web server works. The advantage of this is that the node npm registry contains several web socket modules, one of which is socket.io, which supports asynchronous communication between server and client. A persistent connection between client and server is something that would be needed as it is necessary for the server to communicate with the client each time some event occurs. Some of the alternatives that we have considered include Apache. It would be inefficient to implement web sockets using Apache as several long running persistent connections would be maintained at a time which would reduce the ability to handle several concurrent connections. Node.js is a more suitable solution in this situation.

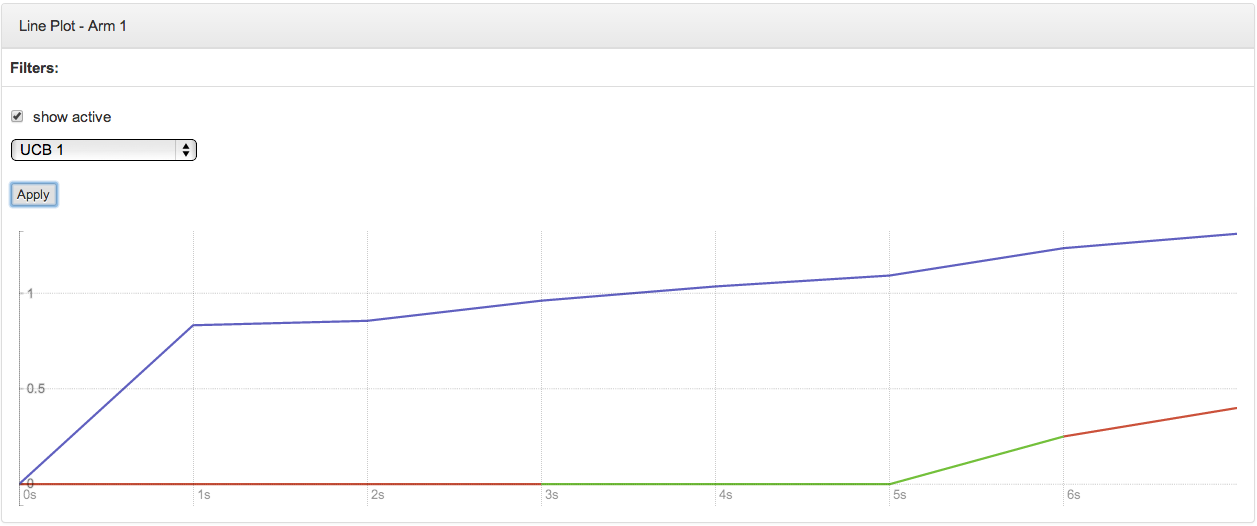
## Currently Implemented Features

The primary feature implemented till date includes being able to upload a set of data from a static file and generate informative charts from this data. The generated charts convey a variety of information. The exact information each chart attempts to bring forward as well as the additional features that are available for each chart is described in detail in the figures shown below.

**Figure 1**: This bar chart demonstrates what is happening at each clock tick. In particular, it shows the arm that was selected by the MAB algorithm as well as whether that arm resulted in a success or failure. Each arm is denoted by a different color to give the user the ability to easily distinguish between the selected arms at each clock tick.



**Figure 2**: This line chart demonstrates the average of rewards earned over time. Thus, each point in time is determined by the number of times the given arm resulted in a positive reward over the number of times that arm was played. There is one such chart for each arm that is present. In addition to this, the chart also displays which arm was selected at each time step. This was done to allow the user to know when a particular arm was active or inactive. Furthermore, the line chart shows a projection of what the rewards for a particular arm have been if the UCB formula was utilized. The latter two features are additional options that have been added for line charts.



## Future Implementations

The next major goal is to implement the live stream feature. This is an alternative to the file upload feature, where, instead of submitting static data to the application, data will be fed in real time and the charts would also be updated in real time once the data is received.

# Results and tests:

## Initial Tests and Experimentation

In order to build the dashboard, a suitable charting library had to be chosen. The research began as an exploration[6] of various JS charting libraries such as Radian, Cubism.js, NVD3.js and Rickshaw.js. Due to the limited time resources at hand, the options were reduced Radian[7][8] and Rickshaw[9]. Sample charts were created in each of these libraries to carefully examine and distinguish the features each had to offer. Table 3 and Table 4 describe the parameters on which the libraries were examined.

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **Reliability** | In development phase  Released in 2013 (very new) |
| **Resource**  **Availability** | Well organized tutorial documentation  External resources for Angular.js directives  Untidy and non-intuitive GitHub repository |
| **Learning**  **Curve** | Knowledge of HTML  Custom HTML elements can represent functional and data plots  Angular.js knowledge for interactive plots |
| **Features and**  **Extensibility** | Limited basic features (covered by Rickshaw) |
| **Size** | radian.min.js = 96.56 KB  No .min version for .css |

**Table 3**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **Reliability** | Established framework  Released in 2011 |
| **Resource**  **Availability** | Limited and concise tutorial documentation  Comprehensive '/examples' section in GitHub repository |
| **Learning**  **Curve** | Knowledge of JavaScript for functional, data and interactive plots |
| **Features and**  **Extensibility** | Feature rich  Vast range of extensions to build on and extend existing functionality |
| **Size** | rickshaw.min.js = 63.592 KB  rickshaw.min.css = 5.975 KB |

**Table 4**

Based on the comparison of the tables above, Rickshaw was our final choice of charting library. The reasons are summarized below:

* Increased reliability- more established framework
* Enhanced resource availability- comprehensive GitHub repository with well explained examples
* Neutral learning curve as JS is the common skill between both group members
* Rich feature set along with a wide range of extensions to customize (suitable for our project)
  + E.g. Time fixture feature for incorporating time series graphs
* .min files available both for JS and CSS
  + rickshaw.min.js is ~30 KB smaller than radian.min.js
  + Smaller file will load faster hence consuming less bandwidth to process

# Impact on Society and the environment:

This section is not applicable to our project.

# Teamwork:

## Workload

The workload has been even thus far. Both members play an active role in writing reports and documentation. Surbhi played a larger role in researching the advantages and disadvantages of the various JavaScript charting libraries that are available; Kishan played a larger role in the implementation.

## Collaboration

We were able to collaborate fairly well as a team. Although this is the first time we’ve worked together, there is a good understanding between the two of us. The most frequent forms of communication are email and text messages. We have status updates meeting once a week in which we discuss what we’ve worked, what we intend to do next as well as any roadblocks that we’ve encountered. In addition to this, after each meeting with our supervisor, we discuss what needs to be done next as well as how tasks should be broken. We feel that using GitHub (for source code) and Dropbox (for documents) greatly alleviates the need to meet frequently in person. Furthermore, we’ve created issues for each feature that needs to be implemented. This helps tracks the progress of each feature and can be used later as reference.

# Conclusion:

This semester allowed us an opportunity to learn more about what MAB algorithms are and the role they play in website optimization. Furthermore, it enabled us to create a user interface to represent the data that is logged from executing a MAB algorithm. The application converts the static data that is supplied to it and generates graphs that allows the user to better interpret the data and make conclusions accordingly. An additional set of options was added for each chart, which enables the user to manipulate the data even further. The goal in the coming semester is to add to the existing feature set of the application. Some of the tasks that will be completed include: enhanced interactivity, improved user interface design and support for live stream.

# References:

[1] Kyle James. (2010, June 15). Why Web Optimization is Important For End User and Business Reasons [Online]. Available: <http://doteduguru.com/id5279-why-website-optimization-is-important.html>

[2] A B Testing The Complete Guide [Online]. Available: https://visualwebsiteoptimizer.com/ab-testing/

[3] Chris Stucchio. (2012, June 3).Why Multi-armed Bandit algorithms are superior to A B testing [Online].Available: <http://www.chrisstucchio.com/blog/2012/bandit_algorithms_vs_ab.html>

[4] Cam Davidson Pilon. (2013, April 06). Multi-Armed Bandits [Online]. Available: <http://camdp.com/blogs/multi-armed-bandits>

[5] [Jean-Yves Audibert](http://imagine.enpc.fr/~audibert), [Rémi Munos](http://researchers.lille.inria.fr/~munos/). ICML Tutorial on bandits [Online]. Available: <https://sites.google.com/site/banditstutorial/>

[6] (2013, Nov 11). Javascript Graphs and Charts libraries Comparison tables [Online]. Available: <http://socialcompare.com/en/comparison/javascript-graphs-and-charts-libraries>

[7] (2013, Nov). openbrainsrc Radian [Online] Available: <https://github.com/openbrainsrc/Radian>

[8] (2011). Radian - Home [Online]. Available: <http://openbrainsrc.github.io/Radian/index.html>

[9] (2013, Nov). Rickshaw A JavaScript toolkit for creating interactive time-series graphs [Online]. Available: http://code.shutterstock.com/rickshaw/