Dashboard for Multi Armed Bandit (MAB) Algorithms

Surbhi Gupta, Kishan Patel

November 13, 2013

Supervisor: Aditya Mahajan, Design Project 1

Objective and Purpose

Objective

To build a **dashboard** in order to represent the results of executing a generic class of **Multi Armed Bandit** (MAB) algorithms used for **Website Optimization** (WO)

Purpose

Ease of identification of best performing (most efficient) MAB algorithm for WO as well as

- In-depth visual understanding
- Engaging interactive design

Terminology

Some terms to familiarize with

- **Agent**: Decision maker
- **Arm**: Action
- **Gain**: Measure of success or reward

MAB Problem

Problem

An **agent** chooses 1 **arm**, and receives a **gain** from it.
How can the agent **maximize** his gain?

Algorithm

Look for the most optimal arm by

- Exploiting the highest performing arms
- Exploring other arms to see if they perform even better

Website Optimization

WO as a bandit problem

What do each of these represent?

- Agent: User
- Arm: Website version with unique styling
  - Color scheme
  - Layouts
  - Size of buttons
- Gain: **Effectiveness** of a particular website version
  - Effectiveness can be defined as a metric of success
  - Definition varies across domains
  - Eg. 1 Number of purchases of a particular item on Amazon.com
  - Eg. 2 Number of donors on a fundraising website

Project Execution Overview

1. Introduction to MAB
2. Model WO as a MAB problem
3. Identify the purpose of a creating a dashboard
   - Research to choose a suitable charting library
   - Create graphs using the chosen library
   - Discuss feedback with supervisor
4. Next steps
   - Prioritize requirements and visit backlog
   - Create a tentative timeline for next semester

Charting Library Research

- Options explored: Radian, Cubism.js, NVD3.js, Rickshaw
- Narrowed choices to: Radian, Rickshaw

Radian

Parameter	Radian
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)

Rickshaw

Parameter	**Rickshaw**
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

Charting Library Research: Result

- Final choice: Rickshaw
- Increased reliability- more established framework
- Enhanced resource availability- comprehensive Github repository
- Neutral learning curve
  - Common skill between group members- JavaScript
  - Limited time to learn a new framework (Angular.js)
- Rich feature set
  - Wide range of extensions to customize (suitable for our project)
  - Eg. Time fixture feature for incorporating time series graphs

File Upload

- The results of the simulation is saved in a file
- Most basic file contains information about each clock tick (arm selected and result achieved)
- Multiple formats supported: CSV, JSON, Tabular
- Once file is loaded, charts are generated
- The end objective is to support live data in addition to file upload

File Upload - Bar Chart

- The bar chart simply represents the data at each clock tick - A 1 or 0.
- In this example, for the first three clock ticks, there was no gain for any arm. For the next three clock ticks, arm 1 resulted in a reward and for the last two clock ticks, arm 2 resulted in a reward.

File Upload - Line Chart

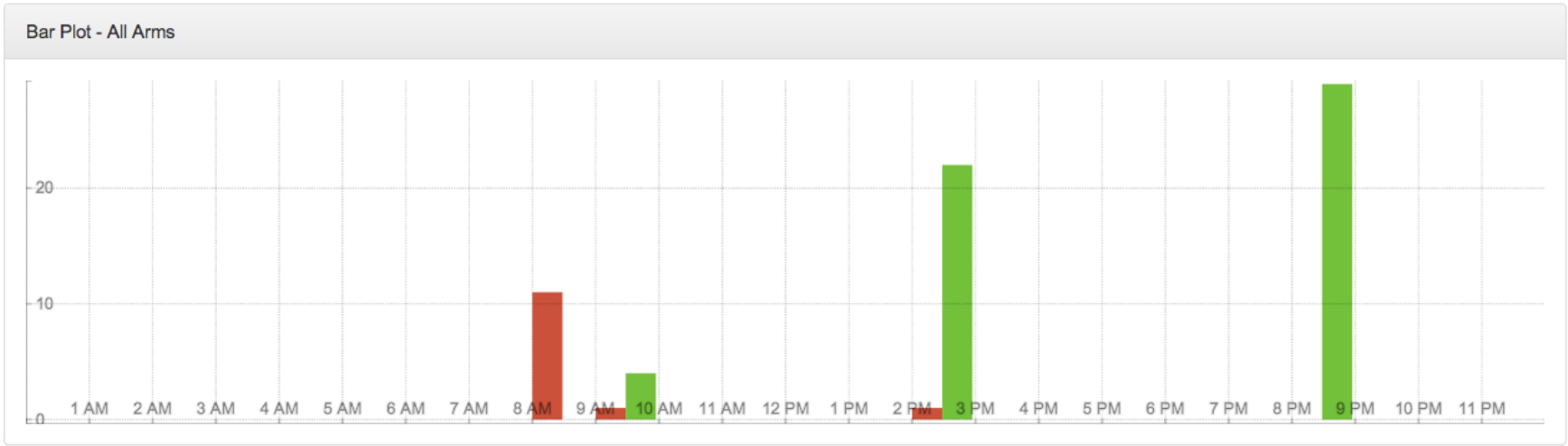
- The line chart represents the average reward received over time
- Each point on the line represents the number of times the arm was played divided by the total time elapsed

Arm Details

- Ability to differentiate between time arm was active/unactive

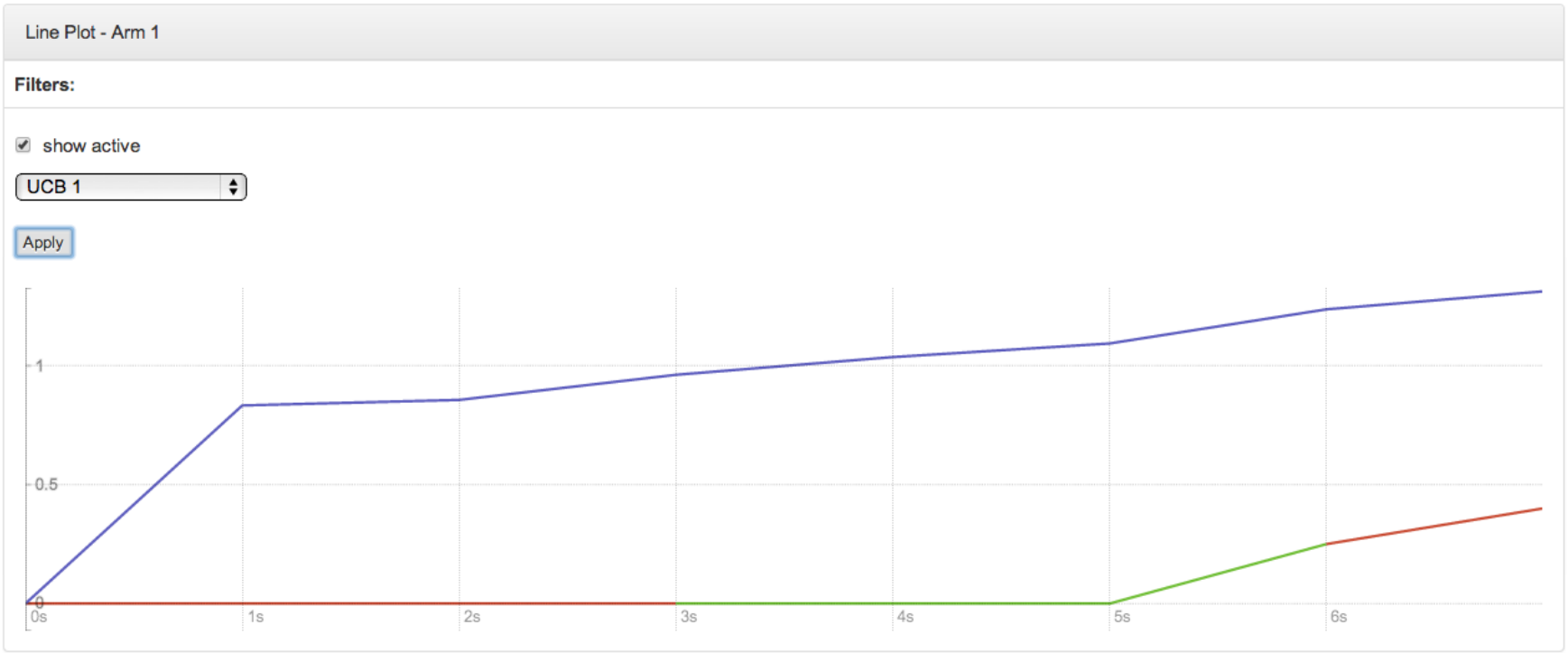
Viewing Results by Time

- For website optimization, traffic varies by time of day
- In order to normalize time zones, bar chart displays results by hour



Running a Simulation

- Function is applied to the static data to generate a simulated line
- Some functions include: UCB, Epsilon Greedy



Support for Live Data and Enhanced Interactivity

- Next major goal is to add support for live data
- The application would listen to events sent by some client and produce graphs in realtime
- The idea is to support multiple clients each having their own graphs
- Increase in interactivity includes: hover effects, adding a slider to accomodate for large data and adding more filtering options
- Define a common interface to communicate with the Web Application

Timeline

DP1, Fall 2013:

- Make sure file upload is robust - all features discussed during meeting are fully implemented and functional
- UI enhancements - filter and hover effects
- Commence live stream implementation planning

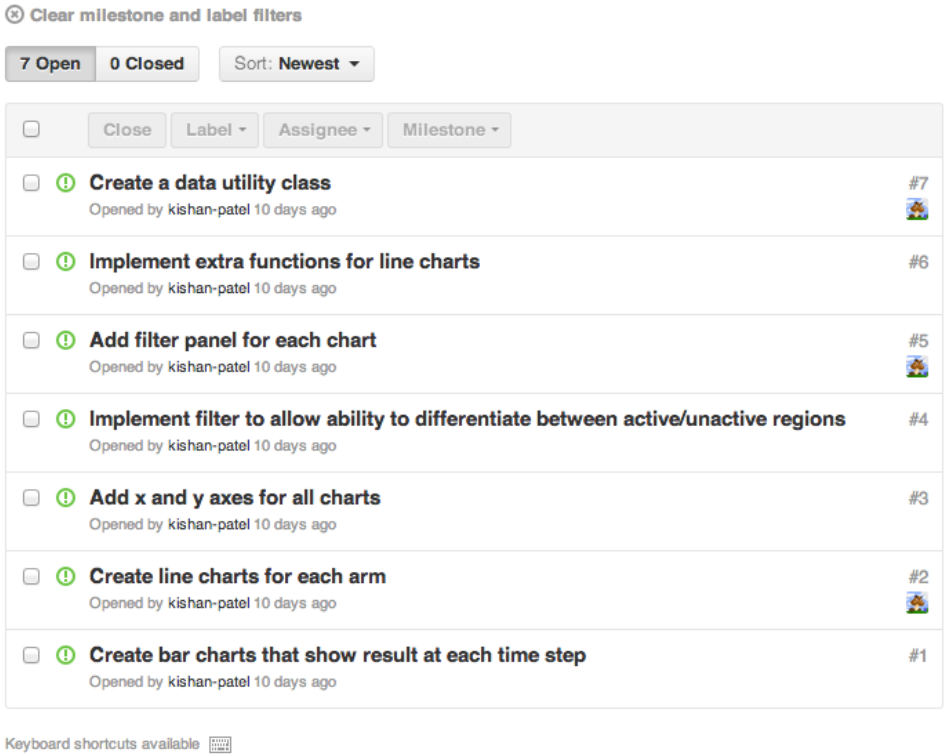
DP2, Winter 2014:

- Complete live stream implementation
- Add additional features (as needed)

Communication

- Primary forms of communication: emails and text messages
- Source code repository: Github. Houses the backlog which consists of list of existing tasks and new tasks are added as progress is made
- Dropbox : documents (meeting notes, resources and submissions)
- Software process models like Waterfall or Agile are not used

Issue Tracker



Challenges

- Higher learning curve in terms of development and testing with limited time resources
- Development: Adding support for multiple clients and live streaming
- Testing: First time testing a web application
  - Familiarize with existing web testing tools
  - Building variety of web test cases
  - Detailed procedure covering functionality, usability, interface and compatibility, performance testing
- Successful implementation of dashboard along with robust testing