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
  - 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 different domains
  - Eg. 1 Number of purchases of a particular item on Amazon.com
  - Eg. 2 Number of donors on a fundraising website

Implementation 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 suitable for our project
  - Eg. Time fixture feature for incorporating time series graphs

Viewing File Data

TODO: Explain what this is about and put a screenshot

Viewing When a Particular Arm is Played

TODO: Explain what this is about and put a screenshot

Viewing Results by Time

TODO: Explain what this is about and put a screenshot

Running a Particular Simulation on Known Data

TODO: Explain what this is about and put a screenshot

Support for Live Data

TODO: Explain what this is about

Enhance Interactivity

TODO: Explain what this is about

Organization and Challenges

TODO: Organization: Talk about how the group organizes itself (meetings breakdown, how we communicate)
Challenges: Talk about the challenges faced eg. in picking charting library, learning curve