Chatty Stochastic Multi-Armed Bandits

Akshay Kumar

February 18, 2014

Abstract

This thesis uses a variant of the classic stochastic multi-armed bandit framework to improve the user experience in an anonymous chat application online by selecting good conversation starters. While the traditional algorithm would converge on the "optimal" conversation starter and use it for every conversation, this novel version of the algorithm attempts to provide new conversation starters for each user while still attempting to maximize the conversation quality. This thesis examines the empirical behavior of such an algorithm in a web application deployed at Princeton University.

Contents

Chapter 1

Introduction

1.1 Motivation

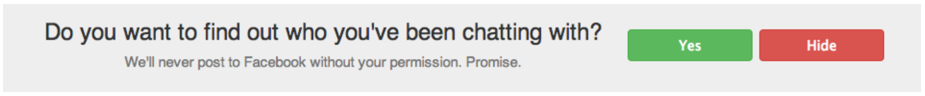
I believe that Tigers Anonymous (TA) satisfies a crucial need at Princeton (and potentially at other college campuses). As students become more settled within the community, it becomes increasingly harder to branch out and meet people outside one's immediate social graph. This is where TA comes in. By providing a way to anonymously be matched with, chat with, and potentially meet fellow classmates, TA allows students to make new connections and shake up their social network. Additionally, I wanted my thesis to be a tangible application of the multi-armed bandit problem and a step towards making Princeton more "hacker-friendly", both of which are satisfied by TA.

1.2 What is Tigers Anonymous?

Tigers Anonymous (TA) is the title of a chat application that allows any Princeton student to be matched with another Princeton student. After being matched, the students will be taken to an anonymous chatroom where they can start a conversation. Once both participants have exchanged a pre-determined number of messages, a drop-down menu appears containing two choices (see Figure 1.1 below). If both users click "Yes", the application will authenticate both users via Facebook and reveal each users' identities to the other to facilitate communication outside of TA. The algorithm used to select the conversation starters is the main focus of this thesis and the success of the algorithm will be measured by whether both users opt to de-anonymize the conversation. For more information on how TA is implemented, see the "TA Implementation" section.

1.3 The Multi-Armed Bandit Problem and POM

This section will contain:


Do you want to find out who you've been chatting with?

We'll never post to Facebook without your permission. Promise.

Figure 1.1: TA Drop-Down Menu

- Description of the multi-armed bandit problem and its common variants, using Bubeck and Cesa-Bianchi (2012) as a starting point
- Description of the bandit problem formulation used for POM and a justification for why it's used
- Description of the algorithms that can be used to solve the POM bandit problem and a justification of the UCB algorithm was chosen

Chapter 2

Methods

2.1 UCB1-AKSB Algorithm

The tentative multi-armed bandit algorithm will be a variant of the UCB1 algorithm proposed by (Auer et al., 2002) as explained below.

The UCB1-AKSB algorithm is initialized by playing each arm once and creating a vector $\bar{X} = [\bar{x}_1, \bar{x}_2, ..., \bar{x}_K]$ where K is the number of conversation starters $\bar{x}_i$ is the empirical estimate of the probability that "Reveal My Identity" will be clicked given the use of conversation starter i. Then, for each subsequent matching, we pick the matching heuristic i using the following rule:

$$\operatorname*{argmax}_{i \in K} \bar{x}_i + \sqrt{\frac{2 \ln n}{n_i}}$$

In this equation n_i is the number of times that machine i has been played and n is the total number of matches that have been completed.

2.2 POM Workflow

The basic workflow of using the upper-confidence bound algorithm for multi-armed stochastic bandits is as follows:

1. New user connects to POM.
2. If the queue size is below the threshold, place the new user into the queue.
3. If the queue size is above the threshold, insert the new user into the queue, remove the user from the head of the queue and use the UCB1 algorithm to pick the matching algorithm to match the selected user with one of the users in the queue.
4. Observe whether or not the "Reveal My Identity" button was clicked and update the $\bar{x}_i$ and n_i for the chosen arm i.

5. Record the 6-tuple representing the data point of this chat session (see Data section below for more details).

Chapter 3

Data

The data that will be collected can be represented by the vector of 6-tuples $(a_i, b_i, k_i, l_i, t_i, p_i)$ where a_i and b_i represent the user ids of the two participants in the chat, k represents the index of the matching heuristic used, l represents the length of the chat conversation, t represents the time that the chat was initiated, $p \in (0, 1)$ is a discrete variable with 0 representing failure to click "Reveal My Identity" button and 1 representing success, and i representing the unique ID of the chat conversation.

The efficacy of the UCB1 algorithm will be assessed by testing the average value of l and p over time to see whether the optimized matching heuristics cause people to chat longer and/or be more likely to click "Reveal My Identity". Additionally, the average values of l and p for each matching heuristic k will be analyzed to see if there are statistically significant differences between the matching heuristics.

Chapter 4

Conclusions

Analysis of data gathered and conclusions drawn go here.

Appendix A

TA Implementation

The following pieces of code implement the back-end and front-end functionality of Tigers Anonymous.

A.1 Chat-room Server Code

This module is used to facilitate multiple anonymous chatrooms, as well as having the code to implement the matching heuristics. The code below is written in a modular way so that multiple heuristics can be implemented using the UCB1 algorithm in the next stage of development (i.e. the getItem method and the SocketWrapper object to manage client-side user data). User history data will be stored as a cookie on the client-side, thus eliminating the need to manage a large database of user history data on the server-side. The only database that needs to be maintained on the server-side is the vector of the historical performance of each of the matching heuristics.

```
CODE!
```

A.2 Authentication Server Code

This module runs the web-server that POM is running on. IP address filtering (to Princeton internal network IP addresses) and Facebook integration will be implemented in this module.

```
CODE!
```

A.3 Front-End Code

This HTML page provides the basic user interface for POM. It will be updated to use Angular.js so that POM looks more visually appealing. The picture below is the extremely bare-bones version of the prototype that is currently being hosted on Heroku for testing.

```
HTML
```

Bibliography

Peter Auer, Nicoló Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

Sébastien Bubeck and Nicoló Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.