Chatty Stochastic Multi-Armed Bandits

Akshay Kumar

March 4, 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

1 Introduction **2**
- 1.1 Motivation . . . 2
- 1.2 What is Tigers Anonymous? . . . 2
- 1.3 The Multi-Armed Bandit Problem and POM . . . 2

2 Methods **4**
- 2.1 UCB1-AKSB Algorithm . . . 4
- 2.2 Tigers Anonymous Workflow . . . 5

3 Data **6**

4 Conclusions **7**

A TA Implementation **8**
- A.1 Back-End Functionality . . . 8
  - A.1.1 UCB1-AKSB Implementation . . . 8
  - A.1.2 Princeton IP-Address Filtering Functionality . . . 10
  - A.1.3 Conversation Starter Functionality . . . 11
  - A.1.4 User Matching Functionality . . . 14
  - A.1.5 Back-end Web Server Functionality . . . 17
  - A.1.6 Conversation Metadata Logging Model . . . 19
- A.2 Front-End Code . . . 20

Bibliography **20**

// Chapter 1

Introduction

1.1 Motivation

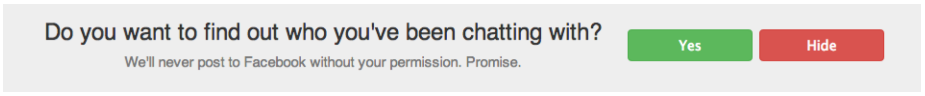
I believe that Tigers Anonymous (TA) satisfies a crucial need at Princeton (and potentially at other college campuses). As students become more settled within their community, it becomes increasingly harder to branch out and meet people outside their immediate social graph. This is where TA comes in. 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 Appendix A.

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 multi-armed bandit algorithm will be my novel variant of the UCB1 algorithm (Auer et al., 2002) as explained below:

Before explaining the algorithm, it will be useful to introduce notation. Let the users be represented as the set U and the bandit arms as the set X. Let the set of arms that have already been played for user $u \in U$ be represented by the set $X^u \subset X$. The goal of the UCB1-AKSB algorithm is to pick some arm $x \in X$ given the pair of users $u, v \in U$. In this specific application, the goal is to pick the optimal conversation starter $x \in X$.

The UCB1-AKSB algorithm proceeds as follows: For each pair of users $u, v \in U$, we pick the conversation starter x such that

$$x = \underset{x \in (X^u \cup X^v)^{\mathsf{c}}}{\operatorname{argmax}} f(x)$$

where

$$f(x) = \begin{cases} \bar{x} + \sqrt{\frac{2 \ln n}{n_x}} & : n_x > 0 \\ \infty & : n_x = 0 \end{cases}$$

In this equation n_x is the number of times that conversation starter x has been played and n is the total number of conversation starters that have been shown. It is useful to note here that ties are broken arbitrarily.

2.2 Tigers Anonymous Workflow

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

1. Two users join the chat server and are matched.
2. Conversation starter is selected based on the UCB1-AKSB algorithm described above.
3. Record the 10-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 10-tuples $(x_i, y_i, t0_i, t1_i, q_i, b_i, c1_i, c2_i, m1_i, m2_i)$ where x_i and y_i represent the pseudonymous user ids of the two participants in the chat, $t0_i$ and $t1_i$ represent the start and end times of the conversation, q_i represents the conversation starter, $b_i \in (0, 1)$ represents whether the drop-down menu was displayed (i.e. both chat participants exchanged more than a predefined number of messages), $c1_i, c2_i \in (0, 1)$ represent whether users x_i and y_i opted to de-anonymize the conversation respectively and $m1_i, m2_i \in (0, 1)$ represent the number of messages that user x_i and y_i sent respectively. The subscript i is unique for each conversation.

The efficacy of the UCB1-AKSB algorithm will be assessed by testing the average value of l and p over time to see whether the optimized conversation starters cause people to chat longer and/or be more likely to opt for de-anonymization.

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 Back-End Functionality

A.1.1 UCB1-AKSB Implementation

```
var questions = require('./questions').list;

// Used in lieu of positive and negative infinity
var largePositiveNumber = 1000000000;
var largeNegativeNumber = -1000000000;

// UCB1 function to pick opening question
exports.getQuestion = function(collection, user1, user2, callback) {
  var questionAsked = {
    $or: [
      {$eq: ["$userID1", user1.id]},
      {$eq: ["$userID2", user1.id]},
      {$eq: ["$userID1", user2.id]},
      {$eq: ["$userID2", user2.id]}
    ]
  };

  var outputFormat = {
    _id: "$question",
    plays: {$sum: 1},
    wins: {$sum: {$cond: [{$and: ["$user1Clicked", "$user2Clicked"]}, 1,
        0]}},
    timesShown: {$sum: {$cond: [questionAsked, 1, 0]}}
  };
  // Aggregate conversation data and call UCB callback
  collection.aggregate().group(outputFormat).exec(function(err, data) {
    if (err) console.log(err);
    UCB1(data, callback);
  });
}

// Helper function to get a random question
var getRandomQuestion = function() {
  var randomIndex = Math.floor(Math.random() * questions.length);
  return questions[randomIndex];
};

// Helper function to invoke callback on the data item with the max UCB
    value
var UCB1 = function(data, callback) {
  var finalData = {};

  // If there's no data, return a random question
  if (data.length === 0) {
    callback(getRandomQuestion());
    return;
  } else {
    // Otherwise, get all the available data for the questions and run
        UCB
    var questionStats = {};
    var totalPlays = 0;

    // For each entry in data, sum the total number of plays and
    // populate the questionStats table with the corresponding question
    for (var i = 0; i < data.length; i++) {
      var entry = data[i];
      questionStats[entry._id] = {
        plays: entry.plays,
        wins: entry.wins,
        shown: (entry.timesShown > 0 ? true : false)
      };
      totalPlays += entry.plays;
    }

    for (var i = 0; i < questions.length; i++) {
      var question = questions[i];
      // If there's no data for this question, then it hasn't been
      // displayed yet, so assign it an arbitrarily large UCB value
      if (!questionStats[question]) {
        finalData[question] = largePositiveNumber;
      } else if (questionStats[question].shown) {
        continue;
      } else {
        // If the question hasn't been shown and there's data for it,
        // compute the UCB value
        var probabilityEstimate =
          questionStats[question].wins / questionStats[question].plays;
        var UCBoundEstimate =
          Math.sqrt(2 * Math.log(totalPlays /
               questionStats[question].plays));
        finalData[question] = probabilityEstimate + UCBoundEstimate;
      }
    }

    if (Object.keys(finalData).length > 0) {
      // Find question with max UCB value
      var bestValue = largeNegativeNumber;
      var bestMatch = null;
      for (var question in finalData) {
        var currentValue = finalData[question];
        if (currentValue >= bestValue) {
          bestMatch = question;
          bestValue = currentValue;
        }
      }
      callback(bestMatch);
    } else {
      callback(getRandomQuestion());
    }
  }
};
```

A.1.2 Princeton IP-Address Filtering Functionality

```
var range_check = require('range_check');

// Pre-defined Princeton IP address blocks
var princetonIPs = [
  "128.112.0.0/16",
  "140.180.0.0/16",
  "204.153.48.0/22",
  "66.180.176.0/24",
  "66.180.177.0/24",
  "66.180.180.0/22"
];

// Check to ensure that the user's IP is a valid Princeton IP
var isValidIP = function (userIP) {
  if (userIP === "127.0.0.1" || // for debugging
      range_check.in_range(userIP, "192.168.0.0/16") ||
      range_check.in_range(userIP, "10.0.0.0/8")) {
```

```
    return true;
  }
  for (var i = 0; i < princetonIPs.length; i++) {
    if (range_check.in_range(userIP, princetonIPs[i])) {
      return true;
    }
  }
  return false;
}

exports.isValidIP = isValidIP;
```

A.1.3 Conversation Starter Functionality

Presented as an array. BLAH.

```
exports.list = [
  "What animal is your Patronus?",
  "If you ruled the world, what laws would you make?",
  "What was your last dream about?",
  "What would you do if you won the lottery?",
  "What does your dream house look like?",
  "What was your favorite vacation?",
  "If you could go back in time to change one thing what would it be?",
  "What's the greatest invention of all time?",
  "Have you ever been admitted to the hospital?",
  "Have you ever had any brushes with the law?",
  "What's the best practical joke you've played on someone?",
  "What's the best practical joke someone's pulled on you?",
  "What is your best achievement to date?",
  "If you could live anywhere, where would it be?",
  "What's your favorite song?",
  "What's your favorite word (inappropriate or otherwise)?",
  "What's the longest period of time you've gone without sleep?",
  "Do you have any scars?",
  "If you could change anything about yourself what would it be?",
  "Would you rather trade some intelligence for looks or looks for
      intelligence?",
  "Have you ever had a secret admirer?",
  "If you could ask your future self one question, what would it be?",
  "Are you a good liar?",
  "What's your favorite joke?",
  "What's the worst present you've ever gotten?",
  "What's your favourite saying?",
  "Have you ever accidentally injured anyone?",
  "What cartoon character would you love to see in 21st century life?",
  "What's the word you use most often?",
  "What's your dream job?",
  "Which song annoys you the most?",
  "What's your first thought when you wake up?",
  "If you could steal one thing in the world, what would it be?",
  "What's your favorite Pokemon character?",
  "When did you stop believing in Santa?",
  "What's your favorite Disney movie?",
  "What's your life motto?",
  "What's the most unusual thing you've ever eaten?",
  "Do you collect anything?",
  "What thing would you like to bring back into fashion?",
  "What makes you nervous?",
  "What's the worst pickup line you've ever heard?",
  "What do you do when you forget someone's name immediately after
      they've introduced themselves?",
  "Have you ever been in a fight?",
  "Have you ever started a rumor?",
  "What's the most memorable rumor you've heard about yourself?",
  "Is there anything about the opposite sex you just don't understand?",
  "If you had a warning label, what would yours say?",
  "Which fictional character do you wish was real?",
  "Who was your first crush?",
  "Do you believe in destiny or free will?",
  "What's the best decision you've made so far?",
  "Who would you want to be with on a desert island?",
  "If you had to pick a new name for a week, what would it be?",
  "What is your first memory?",
  "Where did you go on your first ride on an airplane?",
  "Who was your first best friend?",
  "What was your first detention for?",
  "What would be the name of your debut solo album?",
  "What's something you get compulsive about?",
  "Have you ever stolen anything?",
  "What was the last social faux pas you made?",
  "What makes you nostalgic?",
  "What's the scariest thing you've ever done?",
  "What fairy tale character do you most associate with?",
  "What's the craziest thing you've ever done for someone?",
  "What's the best piece of advice anyone has ever given you?",
  "Do you have a Princeton bucket list?",
  "What's your favorite memory at Princeton?",
  "What building would you donate to Princeton?",
  "What is one thing you always wanted as a kid, but never got?",
  "What is the nicest thing someone else has done for you?",
  "If you could time travel, what would you do?",
  "If you went to a psychiatrist, what would he/she say you suffer
      from?",
  "What one thing annoys you most at a restaurant?",
  "What do Princeton students do too much of today?",
  "What would you like to spend more time doing?",
  "If you could dis-invent one thing, what would it be?",
  "How would you dispose of a dead body?",
  "What's the most recent dream you can remember?",
  "What's something about you that people wouldn't expect?",
  "If you could change one thing about the world, what would it be?",
  "What's your favorite genre of music?",
  "If you could eat lunch with one famous person, who would it be?",
  "How are you feeling right now?",
  "What do you think about the most?",
  "Do you sing in the shower?",
  "Before Princeton, what did you want to be when you grew up?",
  "What is your best childhood memory?",
  "What's something embarrassing that happened to you?",
  "If you could live in any city in the world, where would it be?",
  "Where do you want to travel to?",
  "What's something spontaneous that you've done?",
  "If you could only eat one food for the rest of your life, what would
      it be?",
  "What's your biggest pet peeve?",
  "What was the happiest moment in your life?",
  "What quality about yourself do you value most?",
  "What are you most proud about in your life?",
  "What is your biggest concern about the future?",
  "What is the biggest lesson you've learned in life thus far?",
  "Do you think people can control their own destinies?",
  "How is your relationship with your parents?",
  "If you could go back and relive a day in your life, what would you
      change?",
  "What is the weirdest thing about you?",
  "If you could have any superpower, which one would you pick?",
  "What is the last thing you do before you go to sleep?",
  "Whats the first thing you notice when you meet someone new?",
  "Whats one of your worst habits?",
  "If your house was on fire, what's the one thing you'd want to take
      with you?",
  "If money was no object, what would you be doing with your life?",
  "What does your vision of a utopian society look like?",
  "If you only had one day left to live, what would you do?",
  "What's one thing that you learned this week?",
  "What was the last thing you thought about last night?",
  "What were you like as a kid?",
  "What is one thing you miss about being a kid?",
  "Do you believe in soul mates?",
  "Do you believe in love at first sight?",
  "What's one thing you'd like to change about Princeton?",
  "How was your RCA during your freshman year?"
];
```

A.1.4 User Matching Functionality

```
var mongoose = require('mongoose');
var Conversation = mongoose.model('Conversation');
var ucb = require('./ucb');
var mailer = require('./mailer');

function User(socket, userID) {
  this.socket = socket;
  this.id = userID;
  this.partner = null;
  this.conversation = null;
  this.buttonClicked = false;
  this.messagesSent = 0;
  this.name = null;
  this.fbLink = null;

  var user = this;
  this.socket.on('disconnect', function() {
    if (!user.conversation) return;

    if (!user.conversation.endTime) {
      user.conversation.chatLog.push({
        date: new Date(),
        user: '',
        text: '*** ' + user.pseudonym + ' disconnected ***'
      });

      user.conversation.endTime = new Date();
      user.conversation.save();
      user.partner.socket.emit('finished');
      user.partner.socket.disconnect();
    }
  });

  this.socket.on('chat message', function(data) {
    if (!user.conversation) return;

    user.conversation.chatLog.push({
      date: new Date(),
      user: user.pseudonym,
      text: data.message
    });

    user.messagesSent++;
    user.socket.emit('chat message', {
      name: 'You',
      message: data.message
    });

    var userName = user.conversation.revealed ? user.name : 'Anonymous
        Tiger';
    user.partner.socket.emit('chat message', {
      name: userName,
      message: data.message
    });
  });

  this.socket.on('dropdown displayed', function(data) {
    if (!user.conversation) return;

    user.conversation.buttonDisplayed = true;
  });

  this.socket.on('identity', function(data) {
    if (!user.conversation) return;

    user.conversation.chatLog.push({
      date: new Date(),
      user: '',
      text: '*** ' + user.pseudonym + ' accepted dropdown ***'
    });

    user.name = data.name;
    user.fbLink = data.link;
    user.buttonClicked = true;

    if (user.partner.buttonClicked) {
      user.socket.emit('reveal', {
        name: user.partner.name,
        link: user.partner.fbLink
      });
      user.partner.socket.emit('reveal', {
        name: user.name,
        link: user.fbLink
      });
      user.conversation.revealed = true;

      user.conversation.chatLog.push({
        date: new Date(),
        user: '',
        text: '*** Facebook identities revealed ***'
      });
    }
  });

  this.socket.on('typing', function() {
    if (!user.conversation) return;
```

```
    user.partner.socket.emit('typing');
  });

  this.socket.on('not typing', function() {
    if (!user.conversation) return;

    user.partner.socket.emit('not typing');
  });
}

function ConversationWrapper() {
    this.user1 = null;
    this.user2 = null;
    this.startTime = new Date();
    this.endTime = null;
    this.question = null;
    this.buttonDisplayed = false;
    this.revealed = false;
    this.chatLog = [];

    var self = this;
    this.save = function() {
      new Conversation({
        userID1: self.user1.id,
        userID2: self.user2.id,
        question: self.question,
        startTime: self.startTime,
        endTime: self.endTime,
        buttonDisplayed: self.buttonDisplayed,
        user1Clicked: self.user1.buttonClicked,
        user2Clicked: self.user2.buttonClicked,
        user1MessagesSent: self.user1.messagesSent,
        user2MessagesSent: self.user2.messagesSent
      }).save();

      if (process.env.NODE_ENV === 'production') {
        mailer.sendMail(this);
      }
    };
}

var queue = new Array();
exports.connectChatter = function(socket, userID) {
  var user = new User(socket, userID);

  user.socket.emit('entrance');
  user.socket.emit('waiting');

  if (queue.length === 0) {
    queue.push(user);

    // TODO: remove listener instead of checking index
    user.socket.on('disconnect', function() {
      var index = queue.indexOf(user);
      if (index !== -1) {
        queue.splice(index, 1);
      }
    });
  } else {
    var conversation = new ConversationWrapper();
    conversation.user1 = user;
    user.conversation = conversation;
    user.pseudonym = 'Origin';

    var partner = queue.shift();
    user.partner = partner;
    partner.partner = user;
    conversation.user2 = partner;
    partner.conversation = conversation;
    partner.pseudonym = 'Black';

    ucb.getQuestion(Conversation, user, partner, function(question) {
      user.conversation.question = question;
      user.socket.emit('matched', {
        question: question
      });
      partner.socket.emit('matched', {
        question: question
      });

      conversation.chatLog.push({
        date: new Date(),
        user: '',
        text: question
      });
    });
  }
};
```

A.1.5 Back-end Web Server Functionality

```
var express = require('express'),
    app = express(),
    server = require('http').createServer(app),
    io = require('socket.io').listen(server);
    mongoose = require('mongoose'),
    princeton = require('./server/princeton'),
    conversation = require('./server/conversation'),
    chatter = require('./server/chatter');

var port = process.env.PORT || 5000;
server.listen(port);

var mongoUrl;
io.configure('development', function() {
  mongoUrl = 'mongodb://localhost/test';
});
io.configure('production', function() {
  mongoUrl = process.env.MONGOHQ_URL;
});
mongoose.connect(mongoUrl);

var connectedUsers = {};

app.get('/count', function(req, res) {
  var count = Object.keys(connectedUsers).length;
  res.send(count.toString());
});

io.configure('production', function() {
  io.set('log level', 1);
  io.set('transports', ['websocket']);

  io.set('authorization', function(handshakeData, callback) {
    // Check if Princeton IP
    var ipAddr = getClientIP(handshakeData);
    var isValidIP = princeton.isValidIP(ipAddr);
    if (!isValidIP) {
      callback('Sorry, this site is only for Princeton students!',
          false);
      return;
    }

    // Check if already connected to server
    if (ipAddr in connectedUsers) {
      callback('Sorry, you can only chat with one person at a time!',
          false);
      return;
    }

    callback(null, true);
  });
});

// Needed to get the client's IP on Heroku for socket.io
function getClientIP(handshakeData) {
  var forwardedIps = handshakeData.headers['x-forwarded-for'];
  if (forwardedIps) {
    return forwardedIps.split(', ')[0];
  } else {
    return handshakeData.address.address;
  }
}

function getValueFromCookie(name, cookie) {
  if (cookie) {
    var pairs = cookie.split('; ');
    for (var i = 0; i < pairs.length; i++) {
      var pair = pairs[i].split('=');
      if (pair[0] === name) {
        return pair[1];
      }
    }
  }
}

io.sockets.on('connection', function(socket) {
  var userID = getValueFromCookie('userID',
       socket.handshake.headers.cookie);
  if (userID) {
    // Add user to list of connected users
    var ipAddr = getClientIP(socket.handshake);
    connectedUsers[ipAddr] = true;
    socket.on('disconnect', function() {
      delete connectedUsers[ipAddr];
    });

    chatter.connectChatter(socket, userID);
  } else {
    socket.disconnect();
  }
});
```

A.1.6 Conversation Metadata Logging Model

```
var mongoose = require('mongoose');

var conversationSchema = new mongoose.Schema({
  userID1: String,
  userID2: String,
  startTime: Date,
  endTime: Date,
  question: String,
  buttonDisplayed: Boolean,
  user1Clicked: Boolean,
  user2Clicked: Boolean,
  user1MessagesSent: Number,
  user2MessagesSent: Number
});

mongoose.model('Conversation', conversationSchema);
```

A.2 Front-End Code

```
RAILS CODE GOES HERE!
```

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.