Homework 5

CS 4710

April 24, 2018

Instructions

- Submit one file per group. In that file, define the following constants relative to your group:

```
TEAM_NAME = "mighty_ducks" #Pick a team name
MEMBERS = ["mrf9n","jr5cf","kc4bf"] #Include a list of your members' UVA IDs
```

- Be absolutely sure your code works against the interfaces described. Your single file submission must be importable as a library in a fresh Python3 interpreter without issue.
- Like last time, you will have free access to the load_data and save_data interfaces. **If using load_info or save_info, please standardize to load_data and save_data!**

Trial: Bandit Problem and Auction

This trial consists of two phases. Each player starts with **1 million units** of utility, which will be spent throughout play. At the end of both phases, players will be ranked according to the amount of utility they end up with.

All players will play phase 1 independently, and then phase 2 together.

Phase 1: Bandit Problem

In phase 1, you will be presented with 100 slot machines. Slot machines are the same for all players (they will be auctioned in the second phase). Each slot machine i will be referred to by index (0-99). Each machine costs some amount of utility to play, U_i, which is only observed once the machine is played. In other words, each machine has a fixed yet initially unknown cost to play.

Each machine has a Beta distribution B_i with parameters α_i and β_i and a scaling factor S_i that determines payoffs. When using the slot machine, a player pays U_i utility, and then receives a payoff that equals a draw from B_i multiplied by S_i. Additionally, each machine has an 8-bit number associated with it (e.g. 01010011) which will serve as a set of 8 binary metadata features for that machine. These features can be used to help predict S_i, α_i, and β_i. Slot parameters are drawn from a predictable pattern, so there's a tradeoff between learning about one particular slot to high-degree, and between learning about the mechanism behind the selection of parameters.

In phase 1, each player will be given 10,000 opportunities to play (or not play) the slot machine of

their choice. Players can choose to use their 10,000 selections to explore the slot machines (observing pulls to see which give better expected payoffs), to sit out some proportion of the rounds, or to try and exploit the slots they've already discovered (taking the opportunity to increase their utility).

When playing phase 1, my program will provide yours with a state similar to the one below:

```
{
   "team-code": "eef8976e",
   "game": "phase_1",
   "pulls-left": 99999,
   "last-cost": 0.75,
   "last-payoff": 20.7,
   "last-metadata": 00110101
}
```

You will reply with something similar to the following:

```
{
   "team-code": "eef8976e",
   "game": "phase_1",
   "pull": 17,
}
```

which would pull slot machine 17. If you wish to pull nothing at all, provide the following:

```
{
   "team-code": "eef8976e",
   "game": "phase_1",
   "pull": None,
}
```

At the end of phase 1, all players will have some amount of utility left (which might be more than they started with if they chose to exploit the slots well), and some set of information about the machines. Then phase 2 begins.

Phase 2: Auction

In phase 2, each slot machine will be auctioned to the entire player population. Some may not sell at all, and players can purchase as many as they please. Slot machines are auction sequentially, starting at slot 0, and going to slot 99. At the end of phase 2, each player is forced to pull the slots they won at auction 10,000 times a piece. Utilities for each player will be summed, and determine the player rankings.

The auction phase has two simple components. First, I will send you the following state:

```
{
   "team-code": "eef8976e",
   "game": "phase_2_a",
}
```

Your program should return a list of **no more than 10** slot machines for which you will enter the auction. In other words, you are pre-committing to which auctions (indexed by the slot machine being auctiond) you will attend.

```
{
   "team-code": "eef8976e",
   "game": "phase_2_a",
   "auctions": [20,30,45,60,0,1,8,5,2,95]
}
```

Everybody's "auctions" lists will be published before the auctions are actually run. This may give you a bit of insight into what other players have learned in phase 1. Your program will be entered into each auction. Auctions are **second price, sealed bid**. So playing in an auction is as simple as submitting your bid. Note the "auction-lists" field will return a list of team-codes declared for each auction. It is a list of 100 lists, some of which may be empty, containing the team-codes of the teams which have declared for that auction.

The auction will then proceed in order, index by index. You must bid on the auctions you committed to bid on, but you may also bid on **five additional** auctions of your choice, which will not be broadcast to the other players ahead of time. If you wish to pass on an auction you are involved in, submit a bid of 0.

```
{
   "team-code": "eef8976e",
   "game": "phase_2_b",
   "auction-number": 20,
   "your-slots": [],
   "auction-lists": [
      ["eef8976e", ...]
      ...
   ]
}
```

with valid move

```
{
   "team-code": "eef8976e",
   "game": "phase_2_b",
   "bid": 30.0
}
```

The "your-slots" key in the state will tell you which slots you currently own. If you enter a bid that's higher than your store of utility, you will still spend into negative utility – you cannot run out of "money" this way. However, you will put yourself so far in the hole that your utility is unlikely to recover by the end of the game. **If a player fails to win an auction, they are randomly assigned one of the remaining, unsold slots.**

After all auctions, my program will automatically force your bot to pull each of the slots it owns 10,000 times. Note some slots may have negative expectation!