Multi-Armed Bandit

Deciding which of two slot machines (arms) to play when you don’t know the distributions they are drawing from is an example of a Multi-Arm Bandit (MAB) problem. If the objective is win as much as possible, people often make suboptimal decisions in this task. What if a machine learner (ML) was able to help? Would the collaboration together perform better than the human alone?

Problem Setup

The Setting:
A person performs a set of pulls on 2 arms \{A, B\} with unknown distributions. On each iteration, ML calculates the optimal pull and gives a suggestion. The person performs a pull and both the person and ML see the reward. This process repeats many times.

The Knowns:
- \(j_i\): Arm pulled on pull \(i\), \(i = 1, \ldots, n\)
- \(n_j\): Number of pulls made so far on arm \(j\)
- \(n\): Number of pulls made so far total \((n_A + n_B)\)
- \(R_i\): Reward received at pull \(i, r \in [1, 100]\)

The Unknowns:
- \(P_A\): Distribution of \(A\) with mean \(\mu_A\)
- \(P_B\): Distribution of \(B\) with mean \(\mu_B\)
- \(\mu_i\): True best expected value

The Players:
- (HUMAN): Person playing the arms
- (MACHINE): ML learning the optimal arm

The Constraints:
- Only (HUMAN) can pull the arms
- Both (HUMAN) and (MACHINE) can see the reward

The Goal: To maximize the reward received

Machine Learner

The UCB1 algorithm \cite{auer2002finite} solves this optimally, where optimal is defined as choosing the arm corresponding to \(\max(\mu_A, \mu_B)\).

Per-Trial Regret: \(\mu_i = \frac{1}{n} \sum_{i=1}^{n} R_i\)

At each pull, UCB1 finds the arm with the highest expected upper bound on its reward.

UCB1: \(j_i^* = \arg\max_j r_j + \sqrt{\frac{2 \ln n}{n_j}}\)

Instead of using UCB1 we use UCB1-Tuned (UCB1t) which was found to work better empirically. UCB1t uses a different method to calculate the upper bound.

UCB1t: \(j_i^* = \arg\max_j r_j + \sqrt{\frac{\ln n}{n_j} \min\left(\frac{1}{4}, V_j(n_j)\right)}\)

\(V_j(n_j) = \frac{1}{n_j} \sum_{i=1}^{n_j} r_j^i - r_j n_j + \sqrt{\frac{2 \ln i}{s}}\)

The Experiment

112 participants played two arms for 150 pulls.

Participants were told:
- The range of the reward
- One arm might give better results than the other
- They might receive suggestions
- That their goal was to maximize their reward

Arm distributions were designed to confuse the human learner into choosing the suboptimal arm.

- \(A (\mu_A = 35)\): consistently mediocre reward
- \(B (\mu_B = 50)\): high and low rewards (optimal)

An iteration consists of an actual pull, reward presentation, calculation of the next optimal pull and conversion into a suggestion.

Results

Results were measured using Per-Trial Regret and Best Arm Percentage, the percentage of iterations where the optimal arm was pulled.

As a comparison, UCB1t was run on its own for 5000 sessions, each session 29 trials of 150 iterations.

Differences were not found to be statistically significant, however the trend is surprising in that suggestions do not seem to have helped and, in fact, have hurt in \(S\).

Hybrid system performance was found to be at best equivalent to human performance without suggestions.

Future Work

- Expand suggestions available
- Learn the best suggestion type for a participant
- Create a more complex model of participant agreement taking into account the full history
- Rerun using Gittins’ Dynamic Allocation Process as the ML algorithm (direct reward maximization)

References


With a Little Help from the Computer: Hybrid Human-Machine Systems on Bandit Problems

Bryan R. Gibson, Kwang-Sung Jun, Xiaojin Zhu
University of Wisconsin-Madison, USA