Competing Bandits

Guy Aridor, Kevin Liu

December 4, 2017

Introduction / Setup

In this project we look at a game played between two competing bandit algorithms:

1. Agents (who can be thought of as customers or users) choose between two competing principals (who can be thought of as firms). Each agent does so based on running an algorithm and some beliefs about the rewards they will receive from each principal. Agents seek to maximize their reward. In our implementation, an agent's beliefs are based on the realized rewards that prior agents received, which differs from the paper (in which each agent has no information on what happened before them but has beliefs about what has happened). There are T agents in each simulation, so the simulation lasts for T rounds.

2. Before the beginning of the game, each principal selects a bandit algorithm to use, and starts with some prior beliefs on the distributions of the K arms. Each time the principal is chosen by an agent, the principal runs its bandit algorithm and selects an arm. The reward (drawn from a Bernoulli distribution in our simulations) is given to the agent, and the $(arm, reward)$ data point is used to Bayesian update the principal's posterior (which is a Beta distribution) on that arm. Each principal seeks to maximize its market share (the fraction of agents that chooses them), and can do so by choosing arms with high rewards so that agents will choose them.

3. At each $t = 1, ..., T$, a single agent enters (and lives only for one period) and selects a principal according to the Agent algorithm (throughout the simulation, every agent uses the same algorithm) and the agent's beliefs (which, in our implementation, varies between agents according to past information).

4. The chosen principal runs its bandit algorithm to select an arm. The reward generated by the arm is given to the agent. The principal not chosen is not aware of the $(arm, reward)$ pair.

This is based off the setup of the model in Mansour, Slivkins, and Wu (2017). For more specific details on the model, please refer to their paper. We note any meaningful departures we took from their paper in this paper. We wrote a variety of simulations to look to see if we could reproduce the original results in the paper as well as a few experiments testing certain pertubations of the model.

The first step was to take the model in the paper and make it amenable to simulation as the methodology in the theoretical analysis of the paper made it hard to simulate the beliefs of the agent. To tackle this we assumed that the agent had access to a "reputation" score that came from a sliding window average of the past n agents that had picked this principal.

Then, we implemented the following four behavioral algorithms that describe the decision rule of the agent given the reputation score:

1. HardMax: The agent chooses the firm with a larger score, breaking ties uniformly

2. HardMaxRandom: Each agent uses HardMax with probability $1 - \epsilon$, and chooses between the principals uniformly at random with probability ϵ

3. SoftMax: The principals are chosen randomly according to a logistic function, whose steepness is controlled by α, and each principal has a minimum probability ϵ of being chosen. Further detail is given in a section below.

4. Uniform: Each agent chooses between the two principals uniformly at random. Though we implemented this algorithm, we did not end up using it.

In the simulations we maintained the assumption in the paper that the principals commit to a learning algorithm in the first period and thus we implemented standard bandit learning algorithms for the principals. In general, we were interested in thinking about the differences that would arise when the principals were running greedy algorithms, non-adaptive exploration algorithms, or adaptive exploration algorithms. Bandit algorithms that we implemented include:

1. ExploreThenExploit: The algorithm takes X exploration steps divided evenly among the K arms, according to a modulus function. After the X exploration steps, it always selects the arm it believes has the highest mean.

2. StaticGreedy: Always select the arm whose mean is highest according to the prior distributions.

3. DynamicGreedy: Each round it's run, it selects the arm whose mean is highest according to the posterior distributions at that moment in time.

4. DynamicEpsilonGreedy: With probability ϵ, picks an arm at random. With probability $1 - \epsilon$, runs DyanmicGreedy according to the above.

5. ThompsonSampling: For each of the K posterior distributions, sample a point from each distribution. Take the highest point, and choose the arm corresponding to the distribution that the point came from.

6. UCB: Select the arm whose confidence interval has the highest upper bound.

Summary of Results

In this project, we looked at a competing bandits framework and explored several experiments using it. In our first experiment we looked at the effects on exploration from imposing finite memory on the agents so that they quickly forget the past. In this experiment we showed that with sufficiently low memory we can generate exploration even when agents are perfect expected utility maximizers. In our second experiment we ran an exhaustive experiment comparing the results of various learning algorithms competing against each other. Our results show that Thompson Sampling wins against the other algorithms. As well, the results of which algorithm wins seem to be consistent across all the behavioral models we considered. In our third experiment we explored the consequences of different parameterizations of the SoftMax behavioral model. We found that the results are sensitive to the α that we pick, with $\alpha = 10$ generating results we expected. We found that once we set $\alpha = 10$, the value of ϵ does not make a big difference. In our fourth experiment we tried to answer the question of if the learning algorithm or correctness of the initial information is more important in determining which principal gets higher market share. Our simulations indicated that with enough time, adaptive exploration with poor initial information will catch up with a static greedy algorithm with better initial information. Non-adaptive exploration however will catch up at a slower pace. Our fifth experiment attempted to explore what the consequences of giving the principals free information was if the agents were expected utility maximizers and we found that this has virtually no effect.

A few notes on the simulations / reported results:

1. MS in the figures means market share (% of agents that picked the principal)

2. Unless otherwise noted, the reported results are an average over 25 simulations

3. The distribution of the arms in all of the simulations is a Bernoulli distribution and the principals have Beta priors.

Experiment 1 - Finite Memory

In our simulations we encoded agents' beliefs about the principals as coming from a sliding window average of the rewards previous agents had experienced. One set of simulations that we ran considered the consequences of changing the size of this window. This can be interpreted as agents having finite memory so that "bad" actions by the principals in the past will be forgotten at some point. Intuitively one would expect this to increase exploration even if we fixed agents as playing $HardMax$ and thus being expected utility maximizers. In the paper the main result around $HardMax$ was that ANY deviation from dynamic greedy would cause a principal to lose all remaining agents. However, if there is finite memory we don't necessarily expect this to be the case.

There are two reasonable ways to define finite memory in the context of this model, but we report results from only one. The results we report are both the market share that the principals get as well as the regret that they incur since we want to determine both if there is more exploration and if there are differences in the resulting share of agents they get (i.e. if they can recover from "bad"' choices).

The two definitions of finite memory are:

1. Agents remember only the last n periods, no matter who was picked. If a principal wasn't picked in the last n periods, then we default back to the original prior.

2. Agents form their score for a principal based on the last n observations they have from that principal.

We only simulate definition 2 and the results are as follows:

Agent: HardMax
Principals: Thompson Sampling vs Thompson Sampling

Memory Size	Principal 1 MS	Principal 2 MS	Principal 1 Avg Regret	Principal 2 Avg Regret
1	0.498	0.502	0.021	0.021
5	0.566	0.433	0.041	0.068
10	0.540	0.460	0.062	0.080
50	0.480	0.520	0.109	0.103
100	0.320	0.680	0.130	0.067

Agent: HardMax
Principals: Thompson Sampling vs Dynamic ϵ-Greedy

Memory Size	Dynamic ϵ-Greedy MS	Thompson Sampling MS	D-ϵ Avg Regret	TS Avg Regret
1	0.495	0.505	0.035	0.032
5	0.607	0.3926	0.052	0.079
10	0.531	0.469	0.061	0.065
50	0.402	0.598	0.131	0.0801
100	0.520	0.480	0.111	0.109

These results show that regret roughly increases as the size of memory increases. One of the results from the paper our setting is based off is that in HardMax the principals cannot do any exploration otherwise they'll lose for the remaining periods. In this simulation, when memory is large, we see that in a particular simulation (as opposed to on aggregate) one of the principals always ends up getting almost every agent that comes into the market and there is little exploration, which is in line with the results from the paper. However, when memory is small, there is a more even split of the market in a given simulation and we have that average regret is lower (and thus there is more exploration on average in a given simulation). This makes sense given the fact that agents have a short-term memory and thus principals may not necessarily be punished for too long for sub-optimal decisions. Thus we see that we can get "innovation" even without any behavioral deviations by assuming that agents forget the past eventually.

Experiment 2 - Which Algorithm Wins?

This experiment mainly looks at which learning algorithm "wins" when the two firms do not play the same algorithm. In this experiment we fix the priors of the principals to be identical over the arms ($Beta(0.5, 0.5)$ prior distributions over all arms) and the agent priors to be identical ($Beta(0.5, 0.5)$) over the principals at the beginning of the simulation.

We looked at each behavioral assumption and asked, fixing that behavioral assumption, is there any case where playing a better learning algorithm helps the principal get a higher market share? In all of the simulations below, $K = 10$ and $T = 5000$. 8 arms have distributions $Beta(0.45, 0.55)$, one arm has a distribution $Beta(0.55, 0.45)$, and the last arm has a distribution of $Beta(0.65, 0.35)$.

Agent: HardMax

Principal 1 Alg	Principal 2 Alg	Principal 1 MS	Principal 2 MS	P1 Avg Regret	P2 Avg Regret
StaticGreedy	ExploreThenExploit	0.352	0.648	0.200	0.084
StaticGreedy	Dynamic ϵ-greedy	0.510	0.490	0.200	0.098
DynamicGreedy	ExploreThenExploit	0.444	0.556	0.129	0.093
DynamicGreedy	Dynamic ϵ-greedy	0.486	0.514	0.124	0.111
UCB1	ExploreThenExploit	0.360	0.640	0.156	0.080
UCB1	Dynamic ϵ-greedy	0.335	0.664	0.138	0.100
ThompsonSampling	ExploreThenExploit	0.480	0.512	0.094	0.102
ThompsonSampling	Dynamic ϵ-greedy	0.559	0.441	0.086	0.118
ThompsonSampling	StaticGreedy	0.517	0.482	0.101	0.200
ThompsonSampling	DynamicGreedy	0.508	0.492	0.085	0.137
UCB1	StaticGreedy	0.640	0.360	0.119	0.20
UCB1	DynamicGreedy	0.713	0.287	0.111	0.180
UCB1	ThompsonSampling	0.40	0.600	0.148	0.086

Agent: HardMaxWithRandom, $\epsilon = 0.1$

Principal 1 Alg	Principal 2 Alg	Principal 1 MS	Principal 2 MS	P1 Avg Regret	P2 Avg Regret
StaticGreedy	ExploreThenExploit	0.168	0.832	0.200	0.040
StaticGreedy	Dynamic ϵ-greedy	0.154	0.846	0.200	0.045
DynamicGreedy	ExploreThenExploit	0.408	0.592	0.093	0.053
DynamicGreedy	Dynamic ϵ-greedy	0.485	0.515	0.100	0.083
UCB1	ExploreThenExploit	0.248	0.752	0.130	0.042
UCB1	Dynamic ϵ-greedy	0.265	0.735	0.132	0.045
ThompsonSampling	ExploreThenExploit	0.538	0.462	0.038	0.036
ThompsonSampling	Dynamic ϵ-greedy	0.557	0.443	0.042	0.064
ThompsonSampling	StaticGreedy	0.903	0.097	0.021	0.200
ThompsonSampling	DynamicGreedy	0.603	0.397	0.035	0.089
UCB1	StaticGreedy	0.828	0.172	0.088	0.200
UCB1	DynamicGreedy	0.388	0.612	0.125	0.082
UCB1	ThompsonSampling	0.242	0.758	0.132	0.039

Agent: SoftMax, $\alpha = 10$, $\epsilon = 0.1$

Principal 1 Alg	Principal 2 Alg	Principal 1 MS	Principal 2 MS	P1 Avg Regret	P2 Avg Regret
StaticGreedy	ExploreThenExploit	0.247	0.753	0.200	0.027
StaticGreedy	Dynamic ϵ-greedy	0.250	0.750	0.200	0.032
DynamicGreedy	ExploreThenExploit	0.485	0.515	0.055	0.043
DynamicGreedy	Dynamic ϵ-greedy	0.422	0.578	0.110	0.061
UCB1	ExploreThenExploit	0.344	0.656	0.122	0.029
UCB1	Dynamic ϵ-greedy	0.392	0.608	0.116	0.053
ThompsonSampling	ExploreThenExploit	0.521	0.479	0.034	0.049
ThompsonSampling	Dynamic ϵ-greedy	0.496	0.504	0.041	0.047
ThompsonSampling	StaticGreedy	0.752	0.248	0.026	0.200
ThompsonSampling	DynamicGreedy	0.585	0.415	0.030	0.093
UCB1	StaticGreedy	0.658	0.342	0.010	0.200
UCB1	DynamicGreedy	0.442	0.558	0.114	0.079
UCB1	ThompsonSampling	0.345	0.655	0.120	0.027

We note several learnings from this experiment:

1. Regardless of the behavioral assumption, Thompson Sampling seems to get a larger market share regardless of what the competing algorithm is.
2. In general, non-adaptive exploration algorithms seem to "beat" greedy algorithms.
3. UCB does surprisingly poorly against every algorithm except for StaticGreedy
4. There does not appear to be a large advantage to using an adaptive exploration algorithm compared to a non-adaptive exploration algorithm (as previously noted, Thompson Sampling does slightly better but UCB does worse).
5. The results do not seem to qualitatively vary across different behavioral assumptions.

Experiment 3 - Tuning SoftMax

We implemented SoftMax using a logistic function. There are two parameters that we need to tune: *alpha*, which controls the steepness of the curve, and ϵ, which controls the minimum and maximum values that the logistic function can take. A typical logistic function takes the form

$$f(x) = \frac{1}{1 + e^{-\alpha(x)}}$$

In our case, x corresponds in the difference between principal 1's score and principal 2's score (we calculated score as the mean of the past 50 rewards each principal attained). The range of this function lies in $(0, 1)$, but we want to compress the range to $(\epsilon, 1 - \epsilon)$. That way, when we draw uniformly a z from the range $[0, 1]$, we can compare it to $f(x)$ and pick principal 1 if $z < f(x)$, and pick principal 2 otherwise. So, we instead use the following $f(x)$, which achieves the desired logistic function:

$$f(x) = \epsilon + \frac{1 - 2\epsilon}{1 + e^{-\alpha(x)}}$$

To tune α, we construct the following situation: $K = 10$, $T = 5000$. 8 arms have distributions $Beta(0.45, 0.55)$, 1 arm has distribution $Beta(0.55, 0.45)$, and 1 arm has distribution $Beta(0.8, 0.2)$. Both principals start with prior distributions over all 10 arms of $Beta(0.5, 0.5)$, and agent priors over the two principals start as $Beta(0.5, 0.5)$, so the agents do not initially favor either principal over the other. Then, we run $[StaticGreedy, DynamicGreedy]$ pairwise against $[UCB, ThompsonSampling]$, for $\alpha \in (1, 5, 10, 15)$, and $\epsilon = 0.1$. A reasonable value for α would be one in which the adaptive algorithms take a reasonable chunk of the market. The results are below:

Alpha	Principal 1 Alg	Principal 2 Alg	Principal 1 MS	Principal 2 MS
1	StaticGreedy	ThompsonSampling	0.435	0.565
1	StaticGreedy	UCB1	0.451	0.549
1	DynamicGreedy	ThompsonSampling	0.472	0.528
1	DynamicGreedy	UCB1	0.495	0.505
5	StaticGreedy	ThompsonSampling	0.234	0.766
5	StaticGreedy	UCB1	0.279	0.721
5	DynamicGreedy	ThompsonSampling	0.443	0.557
5	DynamicGreedy	UCB1	0.548	0.452
10	StaticGreedy	ThompsonSampling	0.145	0.855
10	StaticGreedy	UCB1	0.180	0.820
10	DynamicGreedy	ThompsonSampling	0.392	0.608
10	DynamicGreedy	UCB1	0.472	0.528
15	StaticGreedy	ThompsonSampling	0.115	0.885
15	StaticGreedy	UCB1	0.145	0.855
15	DynamicGreedy	ThompsonSampling	0.310	0.690
15	DynamicGreedy	UCB1	0.603	0.397

In the $\alpha = 1$ case, the market is split roughly evenly. This makes sense because if we graph the logistic function, it is very flat, especially in the $[-1, 1]$ domain that x lies in. For α values 10 and 15, the adaptive algorithms reliably beat the greedy algorithms, but there wasn't a big difference between 10 and 15, so we decided to go with $\alpha = 10$.

To tune ϵ, we want to construct a situation where the epsilon baseline allows a "smarter" algorithm to "catch up" and beat a "dumber" algorithm that starts with an advantage. We run simulations pitting StaticGreedy against DynamicGreedy, and construct the following circumstance: $K = 10$, $T = 5000$. 8 arms are distributed according to $Beta(0.45, 0.55)$, 1 arm is distributed according to $Beta(0.55, 0.45)$, and 1 arm is distributed according to $Beta(0.8, 0.2)$. StaticGreedy starts with Beta priors of mean 0.5 over all arms except the arm that has mean 0.55. StaticGreedy's prior on that is $Beta(0.55, 045)$, so StaticGreedy will always choose the arm whose true mean is 0.55 (in other words, it always chooses the second-best arm). DynamicGreedy, on the other hand, starts at a disadvantage: it initially chooses one of the worst arms, because it starts with Beta priors of mean 0.5 over all arms except for one arm (whose real mean is 0.45), for which its prior is $Beta(0.7, 0.3)$.

When we run HardMax, StaticGreedy beats DynamicGreedy, taking, on average over our simulations, 81.4% of the market share. But when we switch to SoftMax (with $\alpha = 10$), even an epsilon of 0.01 allows DynamicGreedy to win.

Epsilon	Principal 1 Alg	Principal 2 Alg	Principal 1 MS	Principal 2 MS
0.01	StaticGreedy	DynamicGreedy	0.331	0.669
0.05	StaticGreedy	DynamicGreedy	0.349	0.651
0.10	StaticGreedy	DynamicGreedy	0.344	0.656

Based on this, we concluded that the particular value of epsilon didn't matter all that much, as long as it is positive so that a "smart" algorithm has a chance to get back in the game and prove itself to the agents. Thus, we considered 0.05 or 0.10 to both be ok.

Experiment 4 - Prior or Algorithm

A natural question to ask in this context is whether the correctness of the initial information that the principals have matters more than the learning algorithm they employ. Specifically, we want to test what happens if we fix one principal as having a better prior than the other but that principal plays *StaticGreedy* and thus only uses their original priors to decide on their actions. Suppose the other principal has a more "incorrect" set of initial beliefs but employs a more sophisticated, adaptive exploration algorithm such as Thompson Sampling. Will the principal playing a smarter learning algorithm catch up eventually (for any of the behavioral assumptions) or is the original prior more important?

We simulate this in the following scenario. We set $K = 10$ where 8 of the arms have approximately identical means ($Bernoulli(0.45)$), one of the arms (denote it as arm B) has a higher mean ($Bernoulli(0.55)$), and we vary the mean of the "best" arm (denote it as arm A) in increments of 0.1. Thus, we consider a simulation where arm A has true distributions of $Bernoulli(0.55), Bernoulli(0.65), Bernoulli(0.75)$.

We suppose that the "dumb" principal 2 has relatively correct beliefs in that she has a prior such that arm B is the best arm. We suppose that the "smart" principal 1 has perverse beliefs such that she has a prior that the 8 worst arms have the best mean reward ($Beta(0.45, 0.55)$), arm B has slightly worse mean reward ($Beta(0.4, 0.6)$, and arm A has substantially worse mean reward ($Beta(0.1, 0.9)$).

We run simulations under this setup to see if the "smart" principal's learning algorithm can let her overcome the bad beliefs.

Agent: HardMax, T = 5000

Mean of Arm A	Principal 1 Alg	Principal 1 MS	Principal 2 MS	Principal 1 Avg Regret	Principal 2 Avg
0.55	ThompsonSampling	0.440	0.560	0.049	0.000
0.65	ThompsonSampling	0.647	0.353	0.057	0.100
0.75	ThompsonSampling	0.710	0.291	0.059	0.200
0.55	Dynamic ϵ-greedy	0.298	0.656	0.057	0.000
0.65	Dynamic ϵ-greedy	0.335	0.664	0.138	0.100
0.75	Dynamic ϵ-greedy	0.442	0.558	0.154	0.200

Agent: HardMaxWithRandom, T = 5000

Mean of Arm A	Principal 1 Alg	Principal 1 MS	Principal 2 MS	Principal 1 Avg Regret	Principal 2 Avg
0.55	ThompsonSampling	0.317	0.683	0.031	0.000
0.65	ThompsonSampling	0.630	0.369	0.028	0.100
0.75	ThompsonSampling	0.787	0.291	0.015	0.200
0.55	Dynamic ϵ-greedy	0.343	0.702	0.038	0.000
0.65	Dynamic ϵ-greedy	0.400	0.600	0.076	0.100
0.75	Dynamic ϵ-greedy	0.615	0.385	0.074	0.200

Agent: SoftMax, T = 5000

Mean of Arm A	Principal 1 Alg	Principal 1 MS	Principal 2 MS	Principal 1 Avg Regret	Principal 2 Avg
0.55	ThompsonSampling	0.442	0.558	0.027	0.000
0.65	ThompsonSampling	0.600	0.400	0.025	0.100
0.75	ThompsonSampling	0.721	0.280	0.026	0.200
0.55	Dynamic ϵ-greedy	0.442	0.556	0.038	0.000
0.65	Dynamic ϵ-greedy	0.400	0.600	0.021	0.100
0.75	Dynamic ϵ-greedy	0.666	0.334	0.050	0.200

Regardless of the behavioral model of the agent, we see that the "dumb" principal 2 wins a higher share of the market than principal 1 when she happens to have correct prior beliefs about the best arm (when the distribution of arm A is $Bernoulli(0.55)$). This happens whether principal 1 plays an adaptive or a non-adaptive exploration algorithm. Intuitively this makes sense as the "dumb" principal will end up getting regret 0 simply because of her correct beliefs and it is incredibly hard for a smarter learning algorithm to make up for this.

However, as we increase the reward of arm A such that principal 2 no longer has completely correct beliefs (she believes the second-best arm is the best arm), we see that when principal 1 plays an adaptive exploration algorithm she catches up very quickly and takes most of the market. However, when principal 1 plays a non-adaptive exploration algorithm we see that the "dumb" algorithm with better initial information still wins out, especially when the agent is HardMax. As we make the agent "more behavioral", the non-adaptive exploration algorithm becomes progressively better.

The previous results show that, especially when playing an adaptive exploration algorithm, it is possible that a better learning algorithm helps overcome initial bad information even if the opponent has almost correct information. However, one expects this takes time and that perhaps if we set T to be lower, the "dumb" principal would still win. To test this, we re-run the same simulations as above but set $T = 500$

Agent: HardMax, T = 500

Mean of Arm A	Principal 1 Alg	Principal 1 MS	Principal 2 MS	Principal 1 Avg Regret	Principal 2 Avg
0.55	ThompsonSampling	0.181	0.812	0.088	0.0
0.65	ThompsonSampling	0.287	0.712	0.178	0.100
0.75	ThompsonSampling	0.274	0.723	0.244	0.200
0.55	Dynamic ϵ-greedy	0.183	0.817	0.100	0.000
0.65	Dynamic ϵ-greedy	0.169	0.831	0.190	0.100
0.75	Dynamic ϵ-greedy	0.297	0.703	0.251	0.200

This demonstrates that part of the results from above were driven by the large T and that with a low T of 500, the "smart" learning algorithm does not have enough time to overcome the bad initial information. Thus we conclude that the "smart" learning algorithm will matter if the number of rounds is sufficiently high but for a low number of rounds, better initial information may be more important than a better learning algorithm.

Experiment 5 - Warm Start and HardMax

What is the effect of a "warm start" on the performance of HardMax? A "warm start" is defined as giving principals free observations without including it in the information set of the agents. Our previous simulations show that with HardMax the initial rounds matter a lot in a particular simulation and whoever wins in the first few rounds takes the entire market. We explore if having a warm start makes it so that the market is more evenly split in a given simulation.

We experimented with giving a warm start of 0, 5, 25, 50, and 100 observations to each principal. We ran the following competing algorithms: UCB vs UCB, UCB vs DynamicGreedy, StaticGreedy vs UCB and observed little effect. Regardless of the algorithm that was played or the number of free observations we gave to the principals, the sequence of simulations lead to market shares in each simulation looking roughly the same. Namely, in a given simulation one of the principals would take the entire market but since the principals had the same priors over the arms and the agents had the same priors over the principals, it was random who would take the entire market. Thus, it seems that free observations for the principal do not impact the resulting market share.

References

1. Mansour Y., Slivkins A., Wu S. (2017) Competing bandits: Learning under Competition, The 9th Innovations in Theoretical Computer Science (ITCS'18)

Appendix

Code for the simulations can be found at https://github.com/rawls238/bandits-rl-project