# Experiment 2

March 16, 2018

## Simulation Details

Considered $K = 3$, $T = 1001$, $N = 500$. Report statistics at $t = 1000$
**The Bandit priors that were considered**:

- Uniform: Draw the mean rewards for the arms from [0.25, 0.75]
- "HeavyTail": We took the mean rewards to be randomly drawn from Beta($\alpha = 0.6, \beta = 0.6$). With this distribution it was likely to have arms that were at the extremes (close to 1 and close to 0) but also some of the arms with intermediate value means.
- Needle-in-haystack
    1. High - 2 arms with mean 0.50, 1 arm with mean 0.70 (+ 0.20)

**Algorithms considered**:

1. ThompsonSampling with priors of $Beta(1, 1)$ for every arm.
2. DynamicGreedy with priors of $Beta(1, 1)$ for every arm
3. Bayesian Dynamic $\epsilon$-greedy with priors of $Beta(1, 1)$ for every arm and $\epsilon = 0.05$

**Agent Algorithms considered**:

1. HardMax
2. HardMaxWithRandom
3. SoftMax ($\alpha = 10$)

**Memory Sizes**

1. 100

**Simulation Procedure**

```
for Each prior p do
    Generate true distribution from p (except for needle-in-haystack, just use p itself)
    Generate T × K realizations for the arms
    for Each agent algorithm agentalg do
        for Each principal algorithm pair principalalg1, principalalg2 do
            for N simulations do
                Give the agents 5 observations from each principal
                Give principal 2 200 free observations (the agents also get these observations)
                Run simulation for T periods
            end for
        end for
    end for
end for
```

## Results

All results are reported for memory size = 100.

The rows represent Principal 1 and the columns represent Principal 2. Thus, the cell (1, 2) represents principal 1 playing Thompson Sampling and principal 2 playing DynamicEpsilonGreedy. In this experiment remember that principal 2 gets 200 free observations.

Within each cell, the following data are presented:

1. The first row displays the sample mean market share for principal 1 as well as a 95% confidence band of the mean.
2. The second row displays the sample variance of the market share and in parentheses are 95 % confidence bands for the variance.
3. The third row displays the % of simulations that resulted in "extreme" market shares. Extreme market shares are defined as being simulations were one of the principals (either principal 1 or 2) ended up with 90% or more of the market.

Results for HardMax t = 1000 Needle In Haystack High

| | ThompsonSampling | DynamicEpsilonGreedy | DynamicGreedy |
|---|---|---|---|
| ThompsonSampling | **0.1** +/- 0.022<br>0.063 (0.056, 0.072)<br>Extreme Shares: 87 % | **0.13** +/- 0.026<br>0.086 (0.076, 0.097)<br>Extreme Shares: 88 % | **0.28** +/- 0.036<br>0.17 (0.15, 0.19)<br>Extreme Shares: 84 % |
| DynamicEpsilonGreedy | **0.093** +/- 0.022<br>0.064 (0.056, 0.072)<br>Extreme Shares: 91 % | **0.18** +/- 0.031<br>0.12 (0.11, 0.14)<br>Extreme Shares: 89 % | **0.26** +/- 0.035<br>0.16 (0.14, 0.18)<br>Extreme Shares: 86 % |
| DynamicGreedy | **0.11** +/- 0.024<br>0.072 (0.064, 0.081)<br>Extreme Shares: 87 % | **0.18** +/- 0.03<br>0.12 (0.1, 0.13)<br>Extreme Shares: 88 % | **0.23** +/- 0.033<br>0.14 (0.13, 0.16)<br>Extreme Shares: 85 % |

Results for HardMaxWithRandom t = 1000 Needle In Haystack High

| | ThompsonSampling | DynamicEpsilonGreedy | DynamicGreedy |
|---|---|---|---|
| ThompsonSampling | **0.17** +/- 0.023<br>0.067 (0.059, 0.076)<br>Extreme Shares: 73 % | **0.3** +/- 0.031<br>0.13 (0.11, 0.14)<br>Extreme Shares: 66 % | **0.34** +/- 0.034<br>0.15 (0.13, 0.17)<br>Extreme Shares: 71 % |
| DynamicEpsilonGreedy | **0.22** +/- 0.026<br>0.088 (0.078, 0.1)<br>Extreme Shares: 67 % | **0.26** +/- 0.029<br>0.11 (0.094, 0.12)<br>Extreme Shares: 68 % | **0.37** +/- 0.034<br>0.15 (0.13, 0.17)<br>Extreme Shares: 66 % |
| DynamicGreedy | **0.23** +/- 0.026<br>0.086 (0.076, 0.097)<br>Extreme Shares: 65 % | **0.29** +/- 0.03<br>0.12 (0.1, 0.13)<br>Extreme Shares: 65 % | **0.37** +/- 0.033<br>0.14 (0.12, 0.16)<br>Extreme Shares: 63 % |

Results for SoftMax t = 1000 Needle In Haystack High

| | ThompsonSampling | DynamicEpsilonGreedy | DynamicGreedy |
|---|---|---|---|
| ThompsonSampling | **0.43** +/- 0.0082<br>0.0088 (0.0078, 0.01)<br>Extreme Shares: 0 % | **0.46** +/- 0.012<br>0.018 (0.016, 0.02)<br>Extreme Shares: 0 % | **0.52** +/- 0.016<br>0.033 (0.029, 0.037)<br>Extreme Shares: 0 % |
| DynamicEpsilonGreedy | **0.42** +/- 0.011<br>0.016 (0.014, 0.018)<br>Extreme Shares: 0 % | **0.46** +/- 0.013<br>0.023 (0.021, 0.027)<br>Extreme Shares: 0 % | **0.5** +/- 0.017<br>0.036 (0.032, 0.041)<br>Extreme Shares: 0 % |
| DynamicGreedy | **0.39** +/- 0.013<br>0.022 (0.02, 0.025)<br>Extreme Shares: 0 % | **0.44** +/- 0.015<br>0.03 (0.026, 0.034)<br>Extreme Shares: 0 % | **0.49** +/- 0.017<br>0.039 (0.034, 0.044)<br>Extreme Shares: 0 % |

Results for HardMax t = 1000 Heavy Tail

| | ThompsonSampling | DynamicEpsilonGreedy | DynamicGreedy |
|---|---|---|---|
| ThompsonSampling | **0.042** +/- 0.014<br>0.026 (0.023, 0.03)<br>Extreme Shares: 93 % | **0.081** +/- 0.021<br>0.06 (0.053, 0.068)<br>Extreme Shares: 93 % | **0.12** +/- 0.027<br>0.091 (0.081, 0.1)<br>Extreme Shares: 93 % |
| DynamicEpsilonGreedy | **0.11** +/- 0.022<br>0.062 (0.055, 0.071)<br>Extreme Shares: 84 % | **0.15** +/- 0.026<br>0.087 (0.077, 0.099)<br>Extreme Shares: 81 % | **0.16** +/- 0.028<br>0.099 (0.088, 0.11)<br>Extreme Shares: 83 % |
| DynamicGreedy | **0.14** +/- 0.024<br>0.077 (0.068, 0.088)<br>Extreme Shares: 82 % | **0.2** +/- 0.031<br>0.13 (0.11, 0.14)<br>Extreme Shares: 83 % | **0.18** +/- 0.027<br>0.097 (0.086, 0.11)<br>Extreme Shares: 76 % |

Results for HardMaxWithRandom t = 1000 Heavy Tail

| | ThompsonSampling | DynamicEpsilonGreedy | DynamicGreedy |
|---|---|---|---|
| ThompsonSampling | **0.12** +/- 0.018<br>0.042 (0.037, 0.048)<br>Extreme Shares: 80 % | **0.17** +/- 0.024<br>0.073 (0.065, 0.083)<br>Extreme Shares: 75 % | **0.21** +/- 0.028<br>0.1 (0.092, 0.12)<br>Extreme Shares: 76 % |
| DynamicEpsilonGreedy | **0.17** +/- 0.021<br>0.055 (0.049, 0.063)<br>Extreme Shares: 67 % | **0.27** +/- 0.026<br>0.09 (0.08, 0.1)<br>Extreme Shares: 54 % | **0.25** +/- 0.027<br>0.095 (0.084, 0.11)<br>Extreme Shares: 62 % |
| DynamicGreedy | **0.21** +/- 0.024<br>0.077 (0.068, 0.087)<br>Extreme Shares: 63 % | **0.31** +/- 0.029<br>0.11 (0.096, 0.12)<br>Extreme Shares: 55 % | **0.3** +/- 0.028<br>0.1 (0.091, 0.12)<br>Extreme Shares: 54 % |

Results for SoftMax t = 1000 Heavy Tail

| | ThompsonSampling | DynamicEpsilonGreedy | DynamicGreedy |
|---|---|---|---|
| ThompsonSampling | **0.44** +/- 0.0059<br>0.0045 (0.004, 0.0052)<br>Extreme Shares: 0 % | **0.48** +/- 0.0083<br>0.009 (0.008, 0.01)<br>Extreme Shares: 0 % | **0.48** +/- 0.011<br>0.015 (0.013, 0.017)<br>Extreme Shares: 0 % |
| DynamicEpsilonGreedy | **0.43** +/- 0.0084<br>0.0092 (0.0081, 0.01)<br>Extreme Shares: 0 % | **0.46** +/- 0.0087<br>0.0098 (0.0087, 0.011)<br>Extreme Shares: 0 % | **0.47** +/- 0.012<br>0.018 (0.016, 0.021)<br>Extreme Shares: 0 % |
| DynamicGreedy | **0.44** +/- 0.0099<br>0.013 (0.011, 0.014)<br>Extreme Shares: 0.2 % | **0.47** +/- 0.011<br>0.016 (0.014, 0.018)<br>Extreme Shares: 0.2 % | **0.49** +/- 0.011<br>0.015 (0.014, 0.017)<br>Extreme Shares: 0 % |

Results for HardMax t = 1000 Uniform

| | ThompsonSampling | DynamicEpsilonGreedy | DynamicGreedy |
|---|---|---|---|
| ThompsonSampling | **0.092** +/- 0.02<br>0.053 (0.047, 0.06)<br>Extreme Shares: 86 % | **0.14** +/- 0.026<br>0.089 (0.079, 0.1)<br>Extreme Shares: 84 % | **0.19** +/- 0.03<br>0.11 (0.1, 0.13)<br>Extreme Shares: 81 % |
| DynamicEpsilonGreedy | **0.16** +/- 0.026<br>0.089 (0.079, 0.1)<br>Extreme Shares: 80 % | **0.18** +/- 0.028<br>0.1 (0.089, 0.11)<br>Extreme Shares: 76 % | **0.19** +/- 0.03<br>0.12 (0.1, 0.13)<br>Extreme Shares: 82 % |
| DynamicGreedy | **0.18** +/- 0.029<br>0.11 (0.095, 0.12)<br>Extreme Shares: 81 % | **0.19** +/- 0.029<br>0.11 (0.099, 0.13)<br>Extreme Shares: 79 % | **0.23** +/- 0.031<br>0.12 (0.11, 0.14)<br>Extreme Shares: 75 % |

Results for HardMaxWithRandom t = 1000 Uniform

| | ThompsonSampling | DynamicEpsilonGreedy | DynamicGreedy |
|---|---|---|---|
| ThompsonSampling | **0.22** +/- 0.024<br>0.076 (0.067, 0.086)<br>Extreme Shares: 60 % | **0.27** +/- 0.028<br>0.1 (0.091, 0.12)<br>Extreme Shares: 61 % | **0.31** +/- 0.031<br>0.12 (0.11, 0.14)<br>Extreme Shares: 61 % |
| DynamicEpsilonGreedy | **0.27** +/- 0.026<br>0.088 (0.078, 0.1)<br>Extreme Shares: 54 % | **0.3** +/- 0.028<br>0.1 (0.091, 0.12)<br>Extreme Shares: 54 % | **0.35** +/- 0.03<br>0.12 (0.11, 0.14)<br>Extreme Shares: 53 % |
| DynamicGreedy | **0.25** +/- 0.026<br>0.086 (0.076, 0.098)<br>Extreme Shares: 59 % | **0.32** +/- 0.029<br>0.11 (0.099, 0.13)<br>Extreme Shares: 55 % | **0.36** +/- 0.03<br>0.12 (0.11, 0.14)<br>Extreme Shares: 52 % |

Results for SoftMax t = 1000 Uniform

| | ThompsonSampling | DynamicEpsilonGreedy | DynamicGreedy |
|---|---|---|---|
| ThompsonSampling | **0.45** +/- 0.0084<br>0.0091 (0.0081, 0.01)<br>Extreme Shares: 0 % | **0.47** +/- 0.0099<br>0.013 (0.011, 0.014)<br>Extreme Shares: 0 % | **0.51** +/- 0.013<br>0.021 (0.018, 0.024)<br>Extreme Shares: 0 % |
| DynamicEpsilonGreedy | **0.44** +/- 0.01<br>0.013 (0.012, 0.015)<br>Extreme Shares: 0 % | **0.46** +/- 0.011<br>0.016 (0.014, 0.018)<br>Extreme Shares: 0 % | **0.49** +/- 0.014<br>0.024 (0.021, 0.027)<br>Extreme Shares: 0.2 % |
| DynamicGreedy | **0.43** +/- 0.011<br>0.017 (0.015, 0.019)<br>Extreme Shares: 0 % | **0.46** +/- 0.012<br>0.019 (0.017, 0.021)<br>Extreme Shares: 0 % | **0.49** +/- 0.013<br>0.022 (0.02, 0.025)<br>Extreme Shares: 0.2 % |

Comments: The story I seem to get from this is (roughly) that the worse the algorithm that the incumbent plays is, the better the entrant will do. Thus, the incumbent should want to play a better algorithm. However, in order for the entrant to do as well as possible, the entrant should in fact play a worse algorithm! If you think of it from the incumbent point of view, the incumbent is in the market first and so wants to learn as much as possible before the entrant comes in (thus does better with a better algorithm since it should have learned more by the time the entrant enters). From the entrant point of view, since the incumbent has already learned something in order to have any chance to compete you do better by purely exploiting than by engaging in potentially suboptimal exploration. Two follow-up questions: Will this result hold if we run the simulations for more periods? What will happen to this result if we increase memory size?