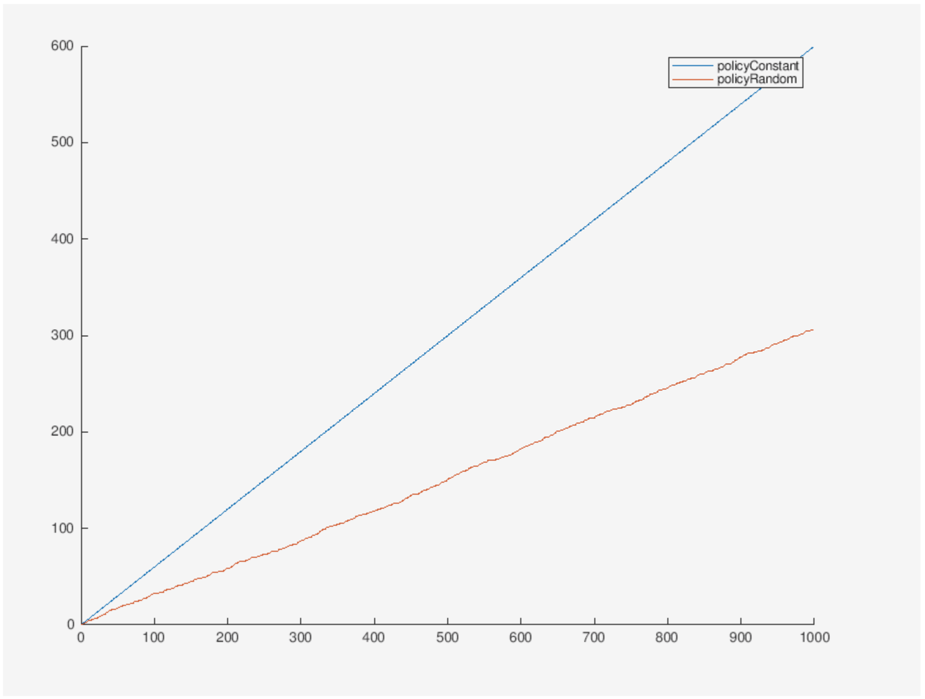
16-831 Homework 3

Jennifer Isaza, jennifei

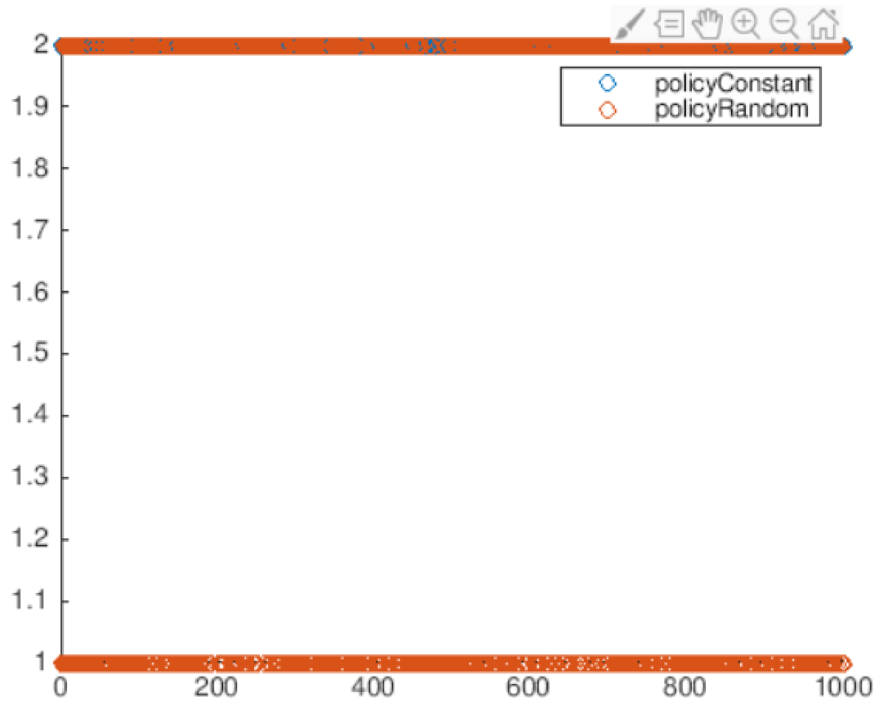
October 2018

2 Framework

2.5 Application



(1) Constant vs Random Regret over Time



(2) Constant and Random Actions Taken over Time

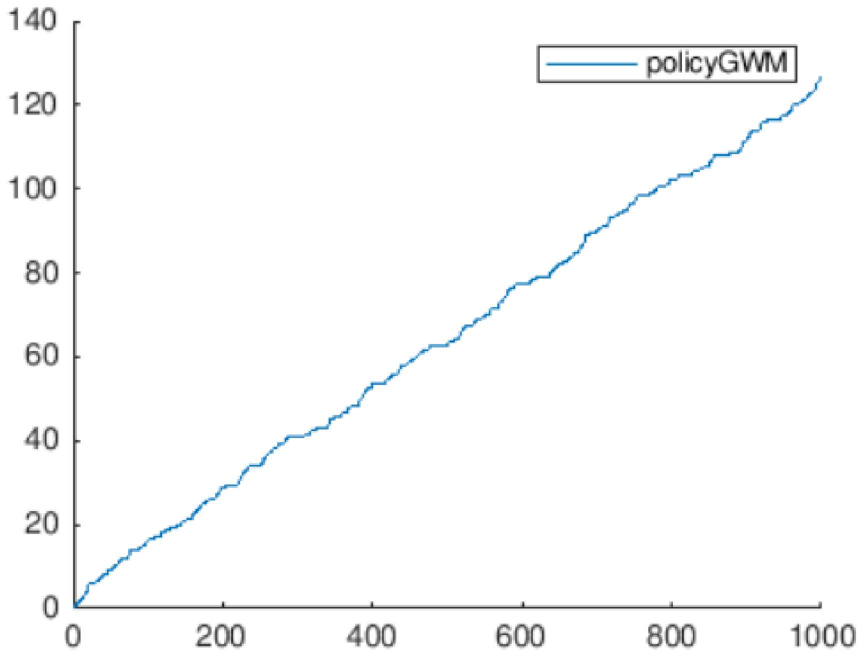
Yes, PolicyRandom regret rises over time. With multiple random restarts, you always get the same results. This is because you are always selecting a random action, so the expected average regret will always be the same.

Yes, PolicyConstant regret rises over time if the best action is chosen. With multiple random restarts, policy constant changes, since the first action chosen is then chosen every time after that. Depending on the action chosen on the first pull, the regret over time changes.

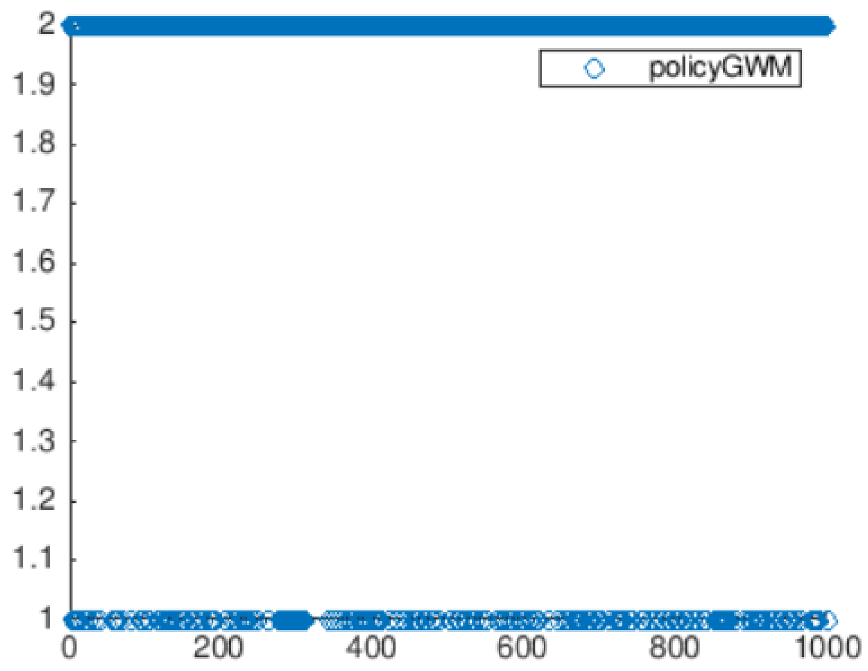
3 EXP3

3.1 Generalized Weighted Majority for Bandits

3.1.1 Implement GWM



(1) GWM Regret over Time



(2) GWM Actions Taken over Time

Yes, the GWM algorithm worked well compared to the previous two policies. GWM had much lower regret than the random polity. GWM is no-regret under the assumption that $l \in [0,1]$ and $\epsilon \in [0, 0.5]$, so that the average regret equation can be sublinear as $t \to \infty$

3.1.3 Unbiased estimator of regret

$$\mathbb{E}[\sum_{t=1}^{T} \bar{l_n^t}] = \sum_{t=1}^{T} l_n^t$$

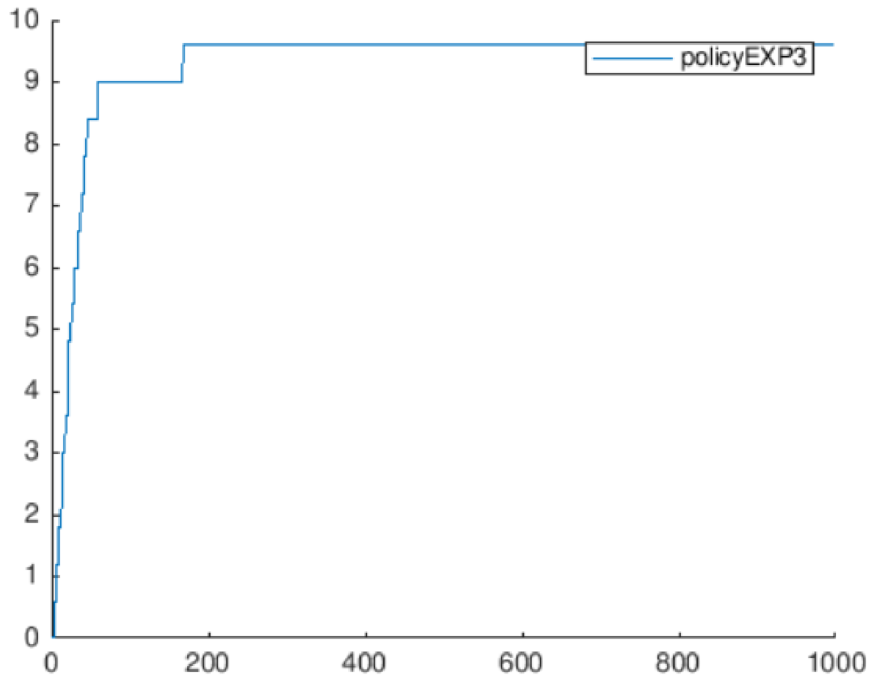
$$p_n^t \sum_{t=1}^{T} \bar{l_n^t} = \sum_{t=1}^{T} l_n^t$$

$$p_n^t \sum_{t=1}^{T} \frac{l_n^t}{p_n^t} \mathbb{1}_{a^t=n} + (1-p_n^t) \sum_{t=1}^{T} \frac{0}{1-p_n^t} \mathbb{1}_{a^t=n} = \sum_{t=1}^{T} l_n^t$$

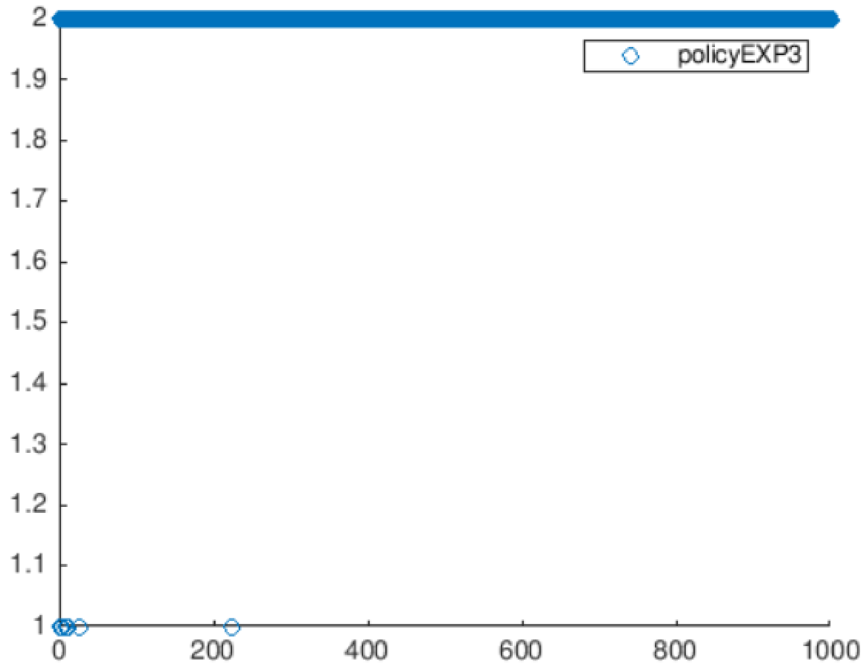
$$\sum_{t=1}^{T} l_n^t = \sum_{t=1}^{T} l_n^t$$

3.2 Implementation

3.2.1 Implement EXP3



(1) EXP3 Regret over Time



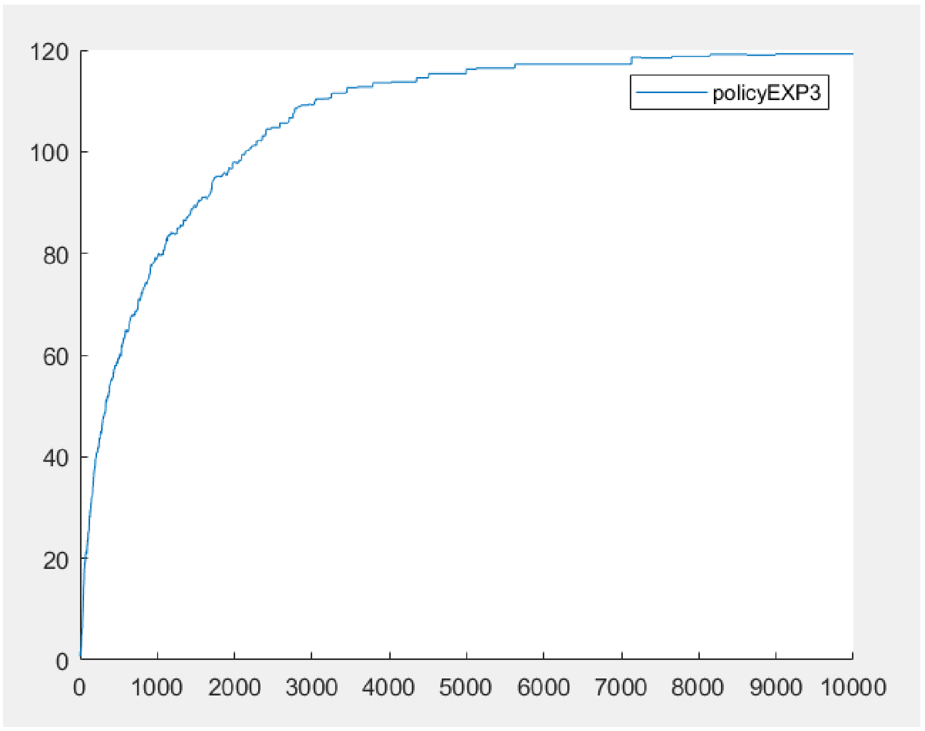
(2) EXP3 Actions Taken over Time

In the beginning of training, EXP3 chooses between different actions while exploring, and the regret increases relatively fast compared to the end of training. At the end of training, the regret stops increasing, as the algorithm picks the best action every time.

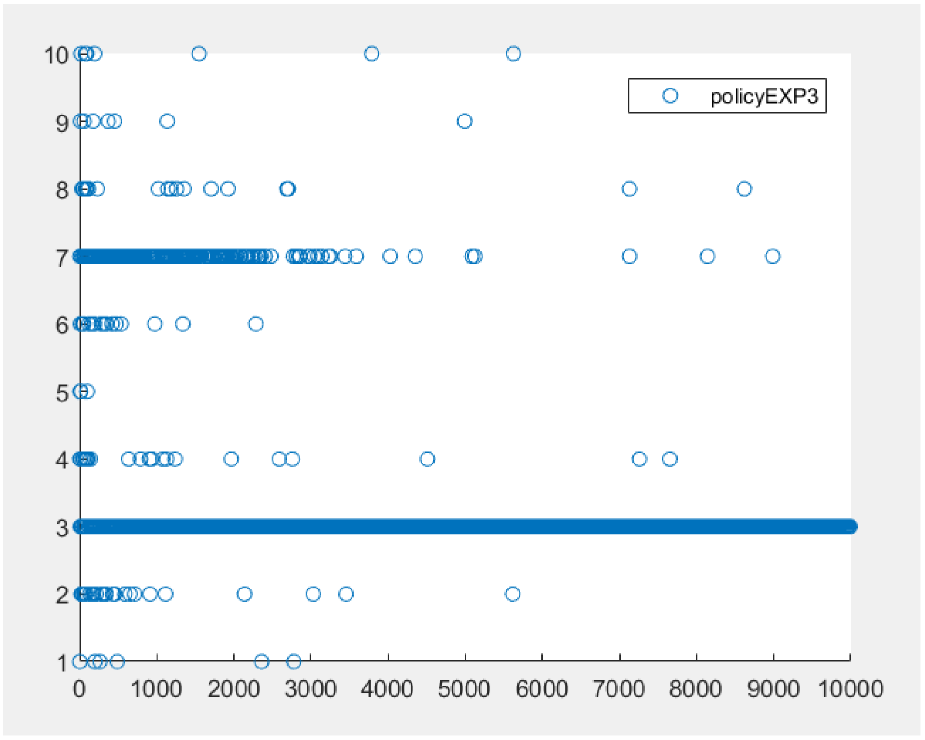
3.3 Gaussian Game

3.3.1 Implement the Gaussian Game

3.3.2 Test EXP3



(1) Gaussian EXP3 Regret over Time



(1) Gaussian EXP3 Actions over Time

4 Upper Confidence Bound (UCB) Algorithm

4.2.1 Upper bound

Given:

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^{m} X_i$$

$$P(\mu - \hat{\mu} \geq \varepsilon) \leq e^{-2m\varepsilon^2}$$

Prove:

$$\mu \leq \frac{1}{m} \sum_{i=1}^{m} X_i + \sqrt{\frac{log(\delta^{-1})}{2m}}$$

Prove upper bound with probability at least 1-δ:

$$1 - \delta \leq P(\mu - \hat{\mu} \geq \varepsilon)$$

$$\delta \geq P(\mu - \hat{\mu} \leq \varepsilon)$$

$$\delta = e^{-2m\varepsilon^2}$$

$$\frac{-log(\delta)}{2m} = \varepsilon^2$$

$$\sqrt{\frac{log(\delta^{-1})}{2m}} = \varepsilon$$

$$\mu - \hat{\mu} \leq \varepsilon$$

$$\mu \leq \frac{1}{m}\sum_{i=1}^{m} X_i + \sqrt{\frac{log(\delta^{-1})}{2m}}$$

4.2.2 Upper bound for the algorithm

Prove:

$$\mu_n^t \leq \hat{\mu_n^t} + \sqrt{\frac{logt}{2C_n^t}}$$

Using:

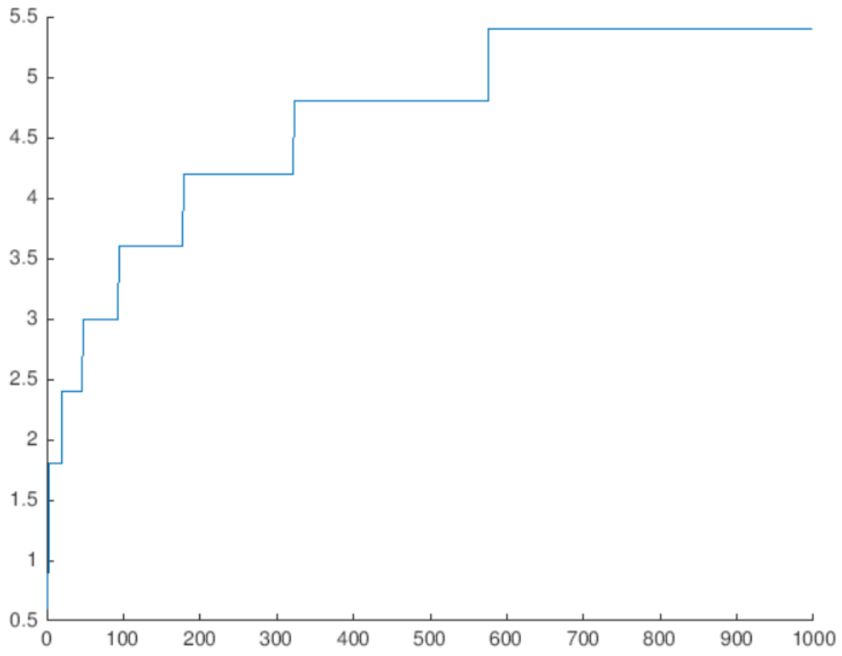
$$\mu \leq \frac{1}{m}\sum_{i=1}^{m} X_i + \sqrt{\frac{-log(\delta)}{2m}}$$

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} X_i$$

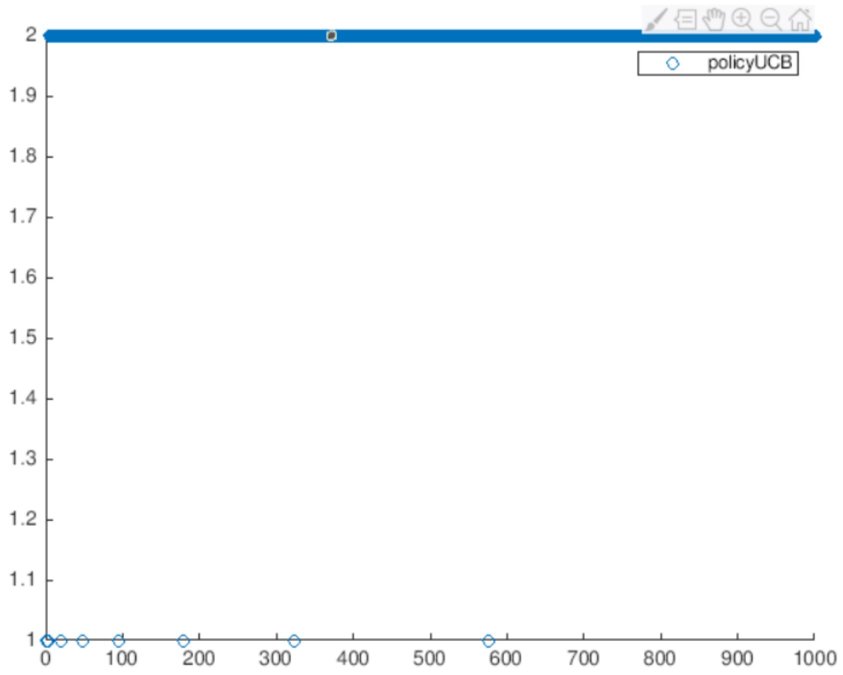
$$C_n^t \leq m$$

$$\mu_n^t \leq \hat{\mu_n^t} + \sqrt{\frac{logt}{2C_n^t}}$$

4.3 Implementation



(1) Constant UCB Regret over Time



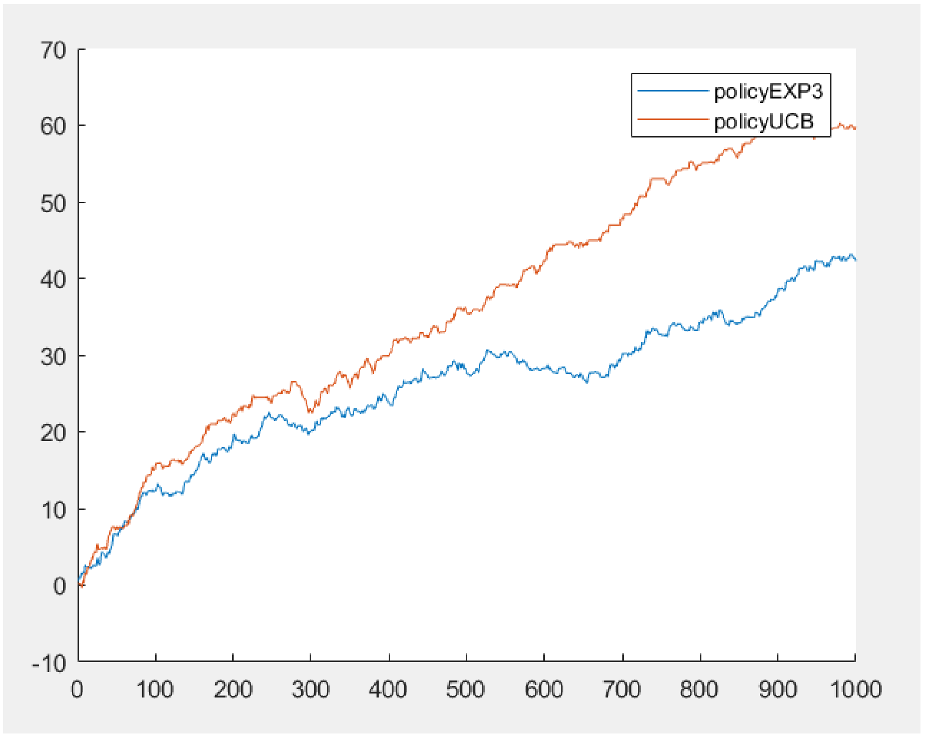
(2) Constant UCB Actions over Time

`change_me.PNG`

(3) Constant UCB Actions Upper Confidence Bounds

Near the beginning of training, UCB exploits the second action after testing both based its optimistic principle. There is minimal switching back to action one after the first rounds, since it loses more and more reward after picking action one, until only action two is chosen at the end and the regret stops increasing. The policy is most confident in action two.

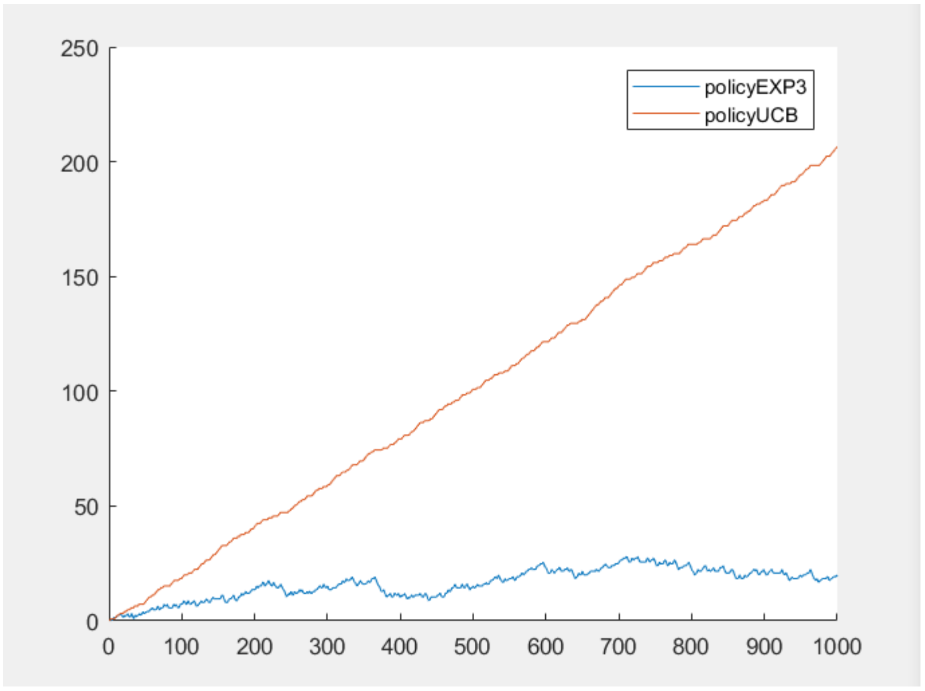
Gaussian Game



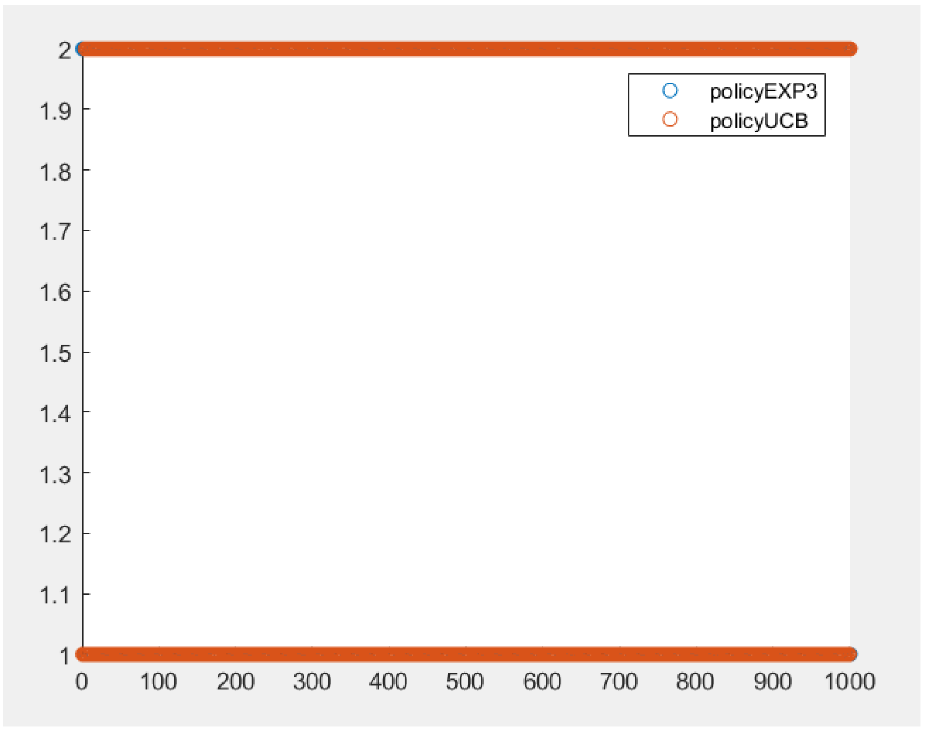
(1) Gaussian UCB vs EXP3 Regret

UCB performs better on average. However, after 10,000 rounds, EXP3 performs better.

Adversarial Game



(1) Adversarial UCB vs EXP3 Regret



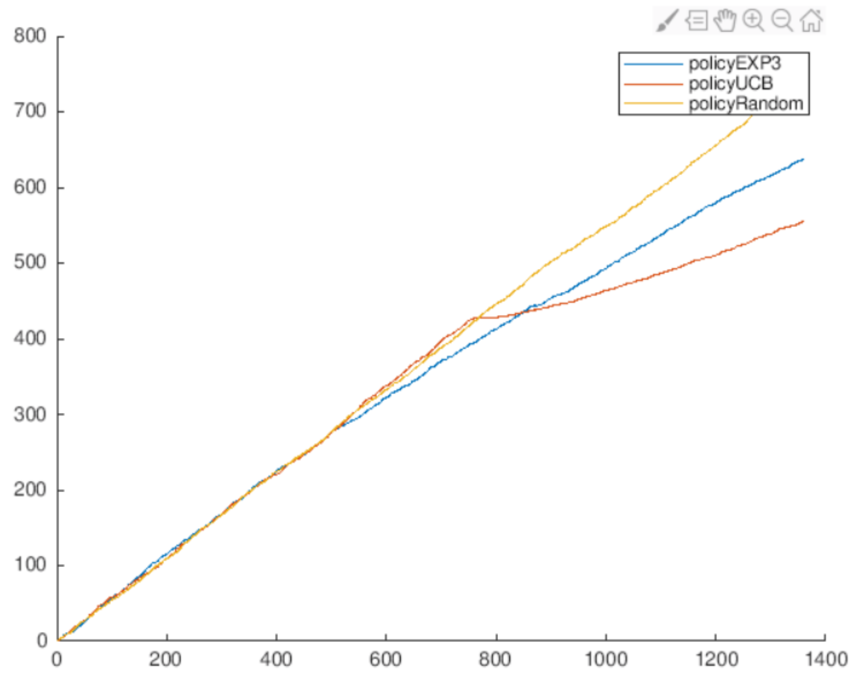
(2) Adversarial UCB vs EXP3 Actions

EXP3 performs better on average due to its multinomial choice of action which cannot be predicted accurately for each round. UCB is deterministic and will always choose the wrong action.

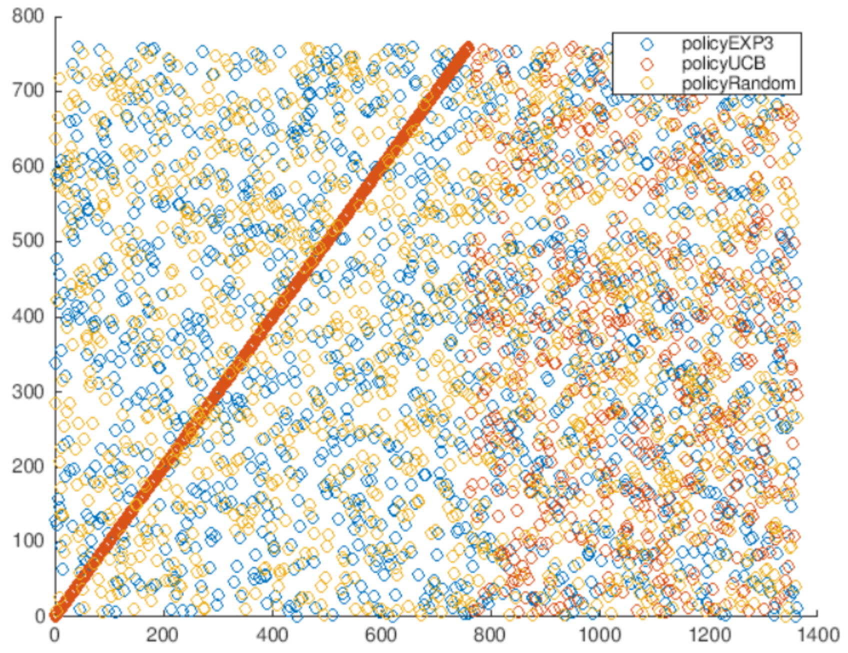
5 Real Datasets

5.1 Lookup Table Game

5.2 University Website Latency Dataset



(1) University Latency UCB, EXP3, Random Regret



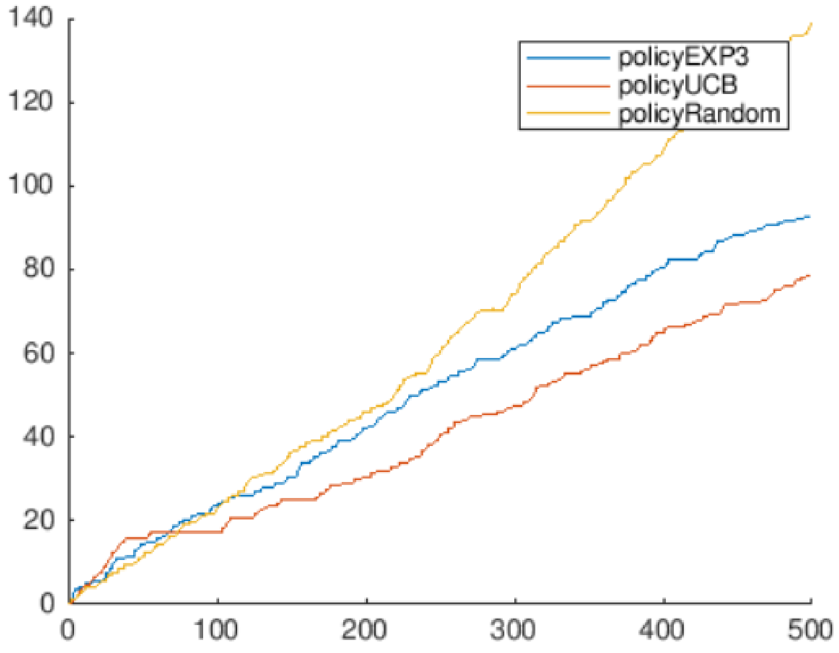
(2) University Latency UCB, EXP3, Random Actions

UCB has the best regret overall, but takes longer to settle on a good action due to exploration.

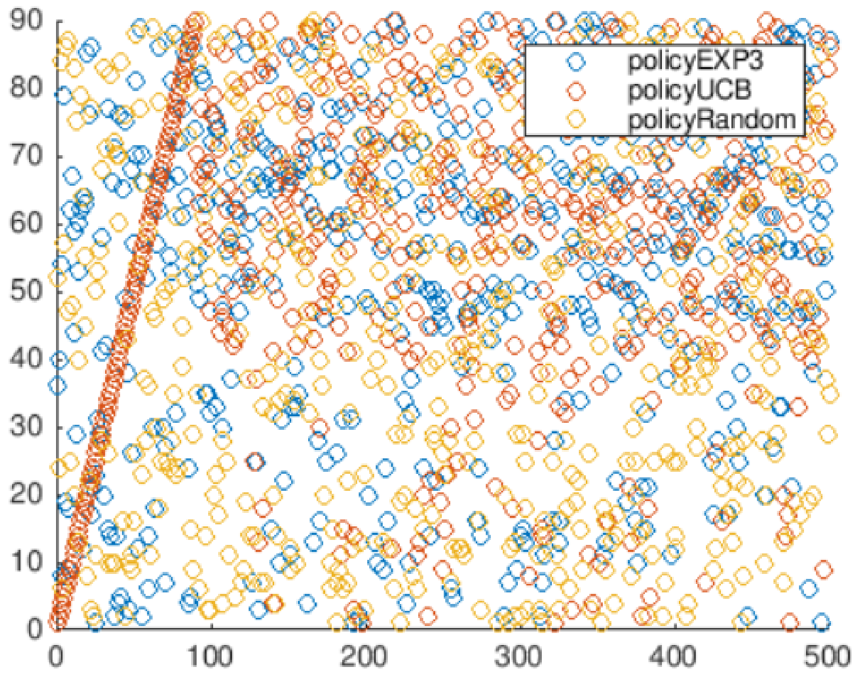
Since there is a large action space compared to the number of rounds, EXP3 and UCB do not perform that much better than Random compared to previous games.

EXP3 takes a lot of rounds to fill up the weight vectors of good actions so that there is a higher probability of choosing them, while UCB has to fill every action before it starts choosing the actions with the best rewards. These can be seen in the weight update step of EXP3 and the initialization for loop of UCB.

5.3 Planner Dataset



(1) Planner Performance UCB, EXP3, Random Regret

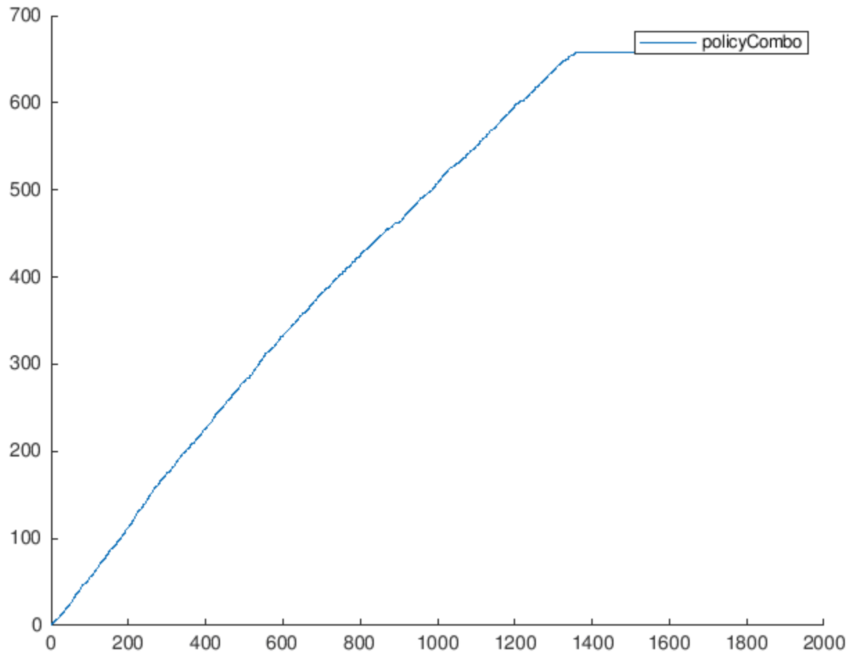


(2) Planner Performance UCB, EXP3, Random Actions

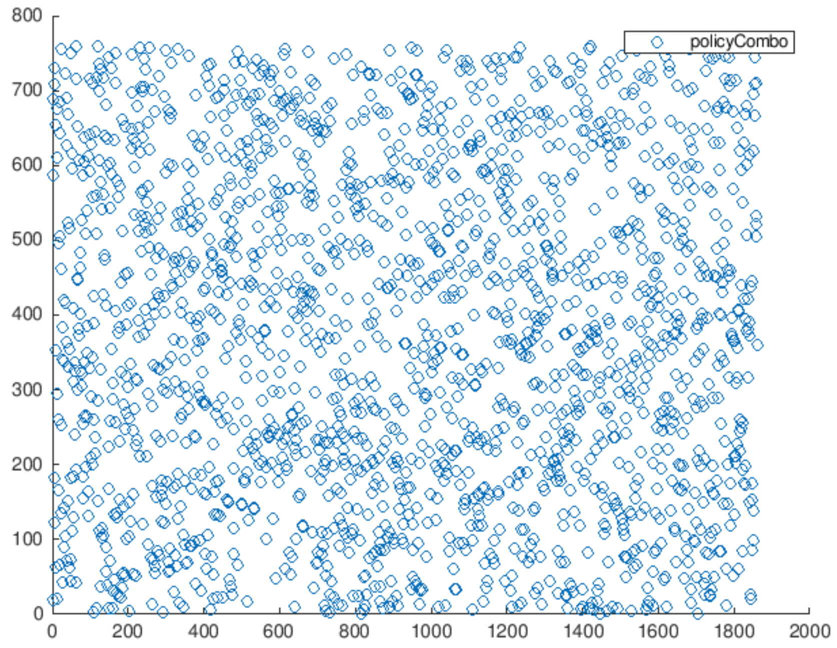
UCB performs better than EXP3 and random.
Since the environment is better for certain planners in the beginning compared to the end, UCB has low regret for the simple environment from training, and then has higher regret later in the cluttered environment when it must switch to different preferred actions.
EXP3 is better than random as well, since it learns the better actions for the simple environment and then learns the better actions for the cluttered environment with each action choice.

6 Incorporating Context

6.2 Context-aware Bot



(1) Regret of combined data using EXP3 with context



(2) Actions of combined data using EXP3 with context

I chose EXP3 to use, to see if being aware of context would bring its performance up to par with UCB. It's performance was about the same as the university latency individual EXP3 run without context, but adding in the planner data made the overall regret improve.