CS 747 Programming Assignment 1

Multi-Arm Bandits

Submitted By

Krishna Wadhwani

160010031

Contents

1 Objective

The objective of this assignment was to solve multi-arm bandit problem and analysis of different sampling algorithms such as round-robin, $\epsilon-$greedy, UCB, KL-UCB and Thompson Sampling.

2 Code Structure

The submission contains the following files:

1. python files:
   i. agent.py
   ii. algorithms.py
   iii. main.py
   iv. plot.py
2. bash script files:
   i. bandit.sh
   ii. get_output_data.sh
   iii. check.sh
3. references.txt
4. outputData.txt

To execute the script for a particular bandit instance, sampling algorithm, horizon, epsilon value and seed value, run the following:

```
$ bash bandit.sh --instance <path-to-instance-file> \
--algorithm <one of the five algorithms implemented>  \
--horizon <horizon-value> --epsilon <epsilon-value> --randomSeed <seed-value>
```

All the five things need to be specified at command line in order to run the script (Execution does not depend on the order of the 5 quantities). Specifically, bandit.sh calls main.py which creates an agent object (defined in agent.py) which calculates expected cumulative regret for the given bandit instance and other parameters. Sampling algorithms have been defined in algorithms.py.

To generate the final output data for all the algorithms (7350 cases defined in the problem statement) and store the results in outputData.txt, run:

```
$ bash generate_output_data.sh  > outputData.txt
```

After the output data has been generated, the required graphs can be generated as follows (plot.py will only run after the generation of outputData.txt):

```
$ python plot.py
```

All the references used have been listed in the references.txt file.

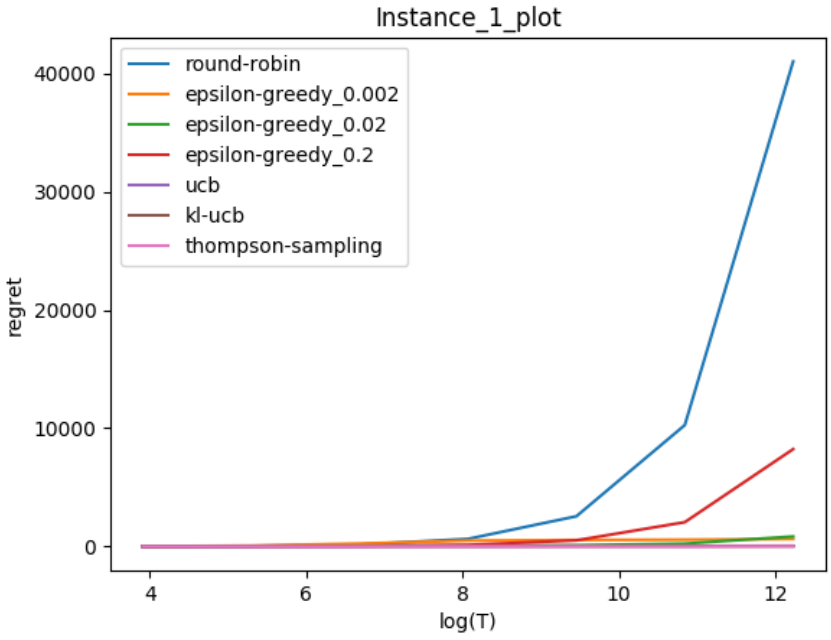
3 Assumptions and Features

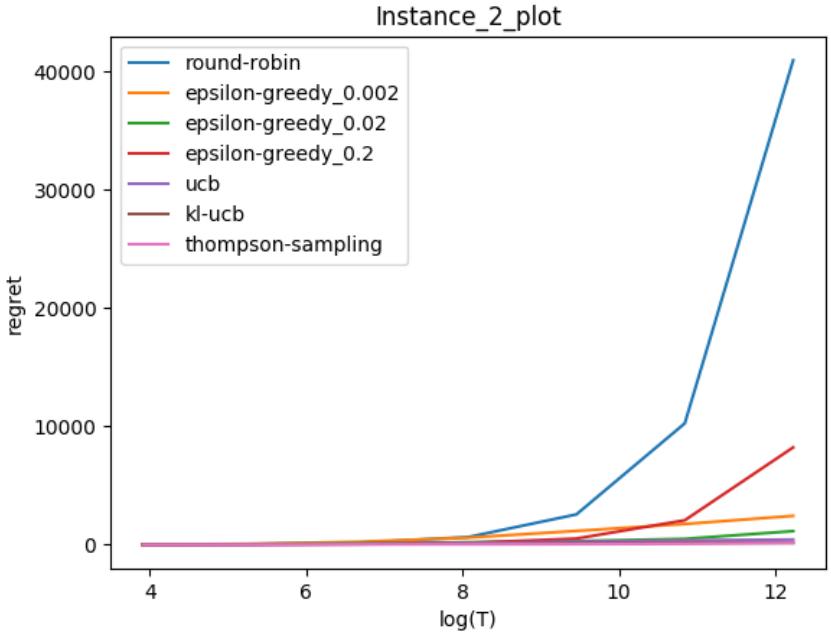
1. All ties are settled by picking the first occurring of the tied value (default behaviour in numpy).
2. Solving kl-ucb:
   (a) As KL divergence between emperical mean and q is monotonically increasing with q in the interval of $[\hat{p}_a^t, 1]$, we can find the required q by solving the inequality (at every timestep) with Bisection method:
   $$KL(\hat{p}_a^t, q) \leq \frac{lnt + 3lnlnt}{\hat{u}_a^t}$$
   (b) Bisection method has been used with Termination condition as the following:
   ```
   abs(rhs_value-kld) > error_precision) and
   q_r - q_l > error_precision
   ```
   where: rhs_value: $(lnt + 3lnlnt)/u_a^t$
   q_r: right estimate of q in bisection method
   q_l: left estimate of q in bisection method
   error_precision: Taken as 1e-5 after some tuning
3. For drawing from beta distribution for Thompson Sampling (np.random.beta), uniform distribution in exploration of $\epsilon-$greedy (np.random.randint), deciding between exploration and exploitation in $\epsilon-$greedy (np.random.choice), numpy library has been used.

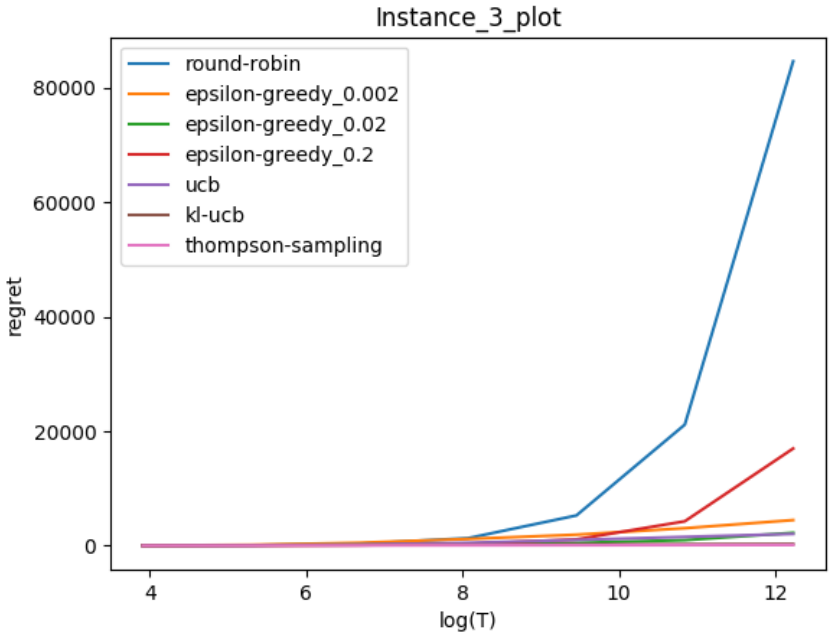
4 Plots

Please note that log represents natural logarithm (base e).

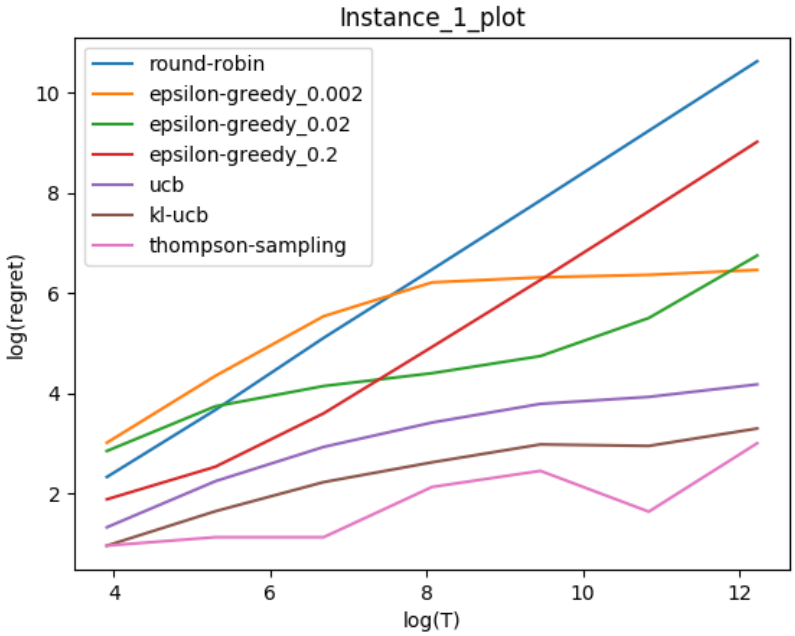
4.1 Expected Cumulative Regret vs Log of Horizon

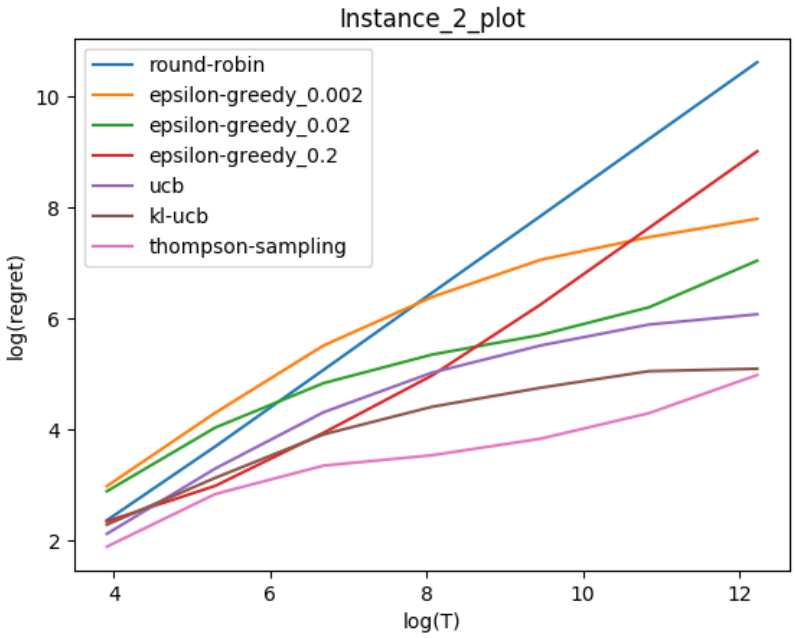


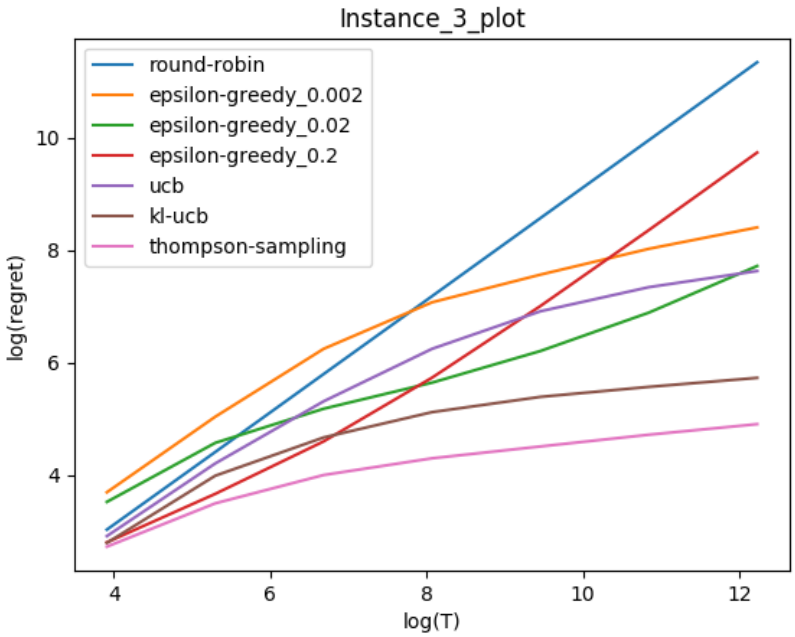

Instance_2_plot
round-robin
epsilon-greedy_0.002
epsilon-greedy_0.02
epsilon-greedy_0.2
ucb
kl-ucb
thompson-sampling
regret
log(T)
0
10000
20000
30000
40000
4
6
8
10
12


Instance_3_plot
round-robin
epsilon-greedy_0.002
epsilon-greedy_0.02
epsilon-greedy_0.2
ucb
kl-ucb
thompson-sampling
regret
log(T)
0
20000
40000
60000
80000
4
6
8
10
12

4.2 Log of Expected Cumulative Reward vs Log of Horizon







5 Conclusions

1. As expected, Thompson Sampling performs the best, followed by kl-ucb.

2. KL-UCB and UCB have regret of the order $O(logT)$, while as per the paper Further Optimal Regret Bounds for Thompson Sampling, regret of Thomson sampling (with beta priors) is of the order $O(\sqrt{logT})$. In the section 4.2, this bound can be empirically verified by the downward bending (downward concavity) of these 2 algorithms.

3. For $\epsilon-$greedy algorithms, $\epsilon = 0.02$ performs the best among other epsilon values, as it seems to achieve the optimum balance of exploration and exploitation. $\epsilon = 0.2$ seems to explore too much while $\epsilon = 0.002$ seems to suffer from premature exploitation

4. The difference between the algorithms is much more evident at larger horizon values.

5. Regret is higher for instance 3, which is expected as instance 3 has more arms and hence, it will be more difficult to find the optimum arm.

6. A very strange observation is that $\epsilon-$greedy ($\epsilon = 0.02$) performs better than UCB for Instance 3 for a considerable number of horizon values. However, the regret grows faster for $\epsilon-$greedy algorithm and for largest horizon value of the experiment, regret for UCB is lower.