A Python 3 Bandit Visualization Library

Alex Rutar (arutar@uwaterloo.ca) with Tor Lattimore and Csaba Szepesvári

Reinforcement Learning and Artificial Intelligence at the University of Alberta

Introduction

The multi-armed bandit problem assumes an agent with a single state and a selection of actions, each yielding a reward. We attempt to minimize the regret. However, to reduce stochastic noise, it can be more meaningful to track the pseudo-regret

$$\overline{R}_n = \max_i \mathbb{E}\left[\sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t}\right]$$

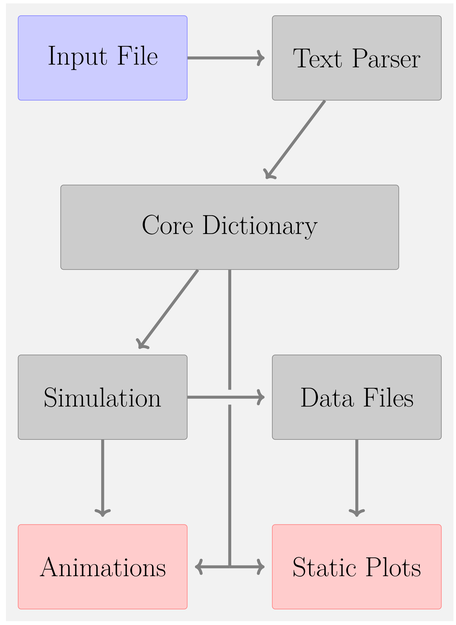
where n is the horizon, $X_{i,t}$ is the reward from the best arm i, and I_t is the arm chosen on timestep t. This library simulates bandit algorithms by

- visualizing the behaviour of bandit algorithms and environments
- using minilanguage syntax to control simulations

Outline of Implemented Features

- support for a wide variety of algorithms, stochastic environments, and linear bandits
- entirely in python with few dependencies: easy to modify, but not very fast
- create regret histograms and line plots to compare algorithm performance
- run animations to visualize properties of the bandit algorithms

Structure of the Library



Example Input

```
init: Visualize
visual: ellipse

horizon: 5000

Simulation:
    Algorithm:
        algtype: Lin_UCB
    Bandit:
        ArmList:
        - [Linear, [1., 1.]]
        - [Linear, [1., 0.]]
        - [Linear, [1., -1.]]
        - [Linear, [-1., 1.]]
        - [Linear, [-1., 0.]]
        - [Linear, [-1., -1.]]
        - [Linear, [0., 1.]]
        - [Linear, [0., -1.]]
        MeanVector: [0.3, 0.4]
    Normalized: True

NoAxesTick: True
HelpLines: True
```

Example Plots and Screenshots

Below are four examples of output created by this library. The two examples on the left are completed plots, and the two examples on the right are screenshots of animations. The *Example Input* corresponds to the image on the far right.

Additional Figure Information

- Figure 1: The pseudo-regret is recorded over a large number of cycles. The TS algorithm has lower regret on average but pays the price of a larger 'tail'.
- Figure 2: Rewards are drawn from a normal distribution, and the average regret is plotted. The $UCB-B7$ algorithm has the best performance.
- Figure 3: The upper confidence estimation reflects confident the algorithm is with a certain arm, based on average past reward and number of pulls.
- Figure 4: Linear Bandits draw reward as a linear function of the arm vector and mean vector, plus some noise. The algorithm tries to estimate the true location of the mean.

Final Comments

- open source code can be found at `www.github.com/alexrutar/banditvis`
- library is still in progress; suggestions for modifications and additions are welcome!

Acknowledgements

I am grateful to Tor Lattimore for giving teaching me the theory and advising me during this project. I would also like to thank Csaba Szepesvári and the Ross and Verna Tate foundation for providing me with the opportunity to work in Reinforcement Learning at the U of A.

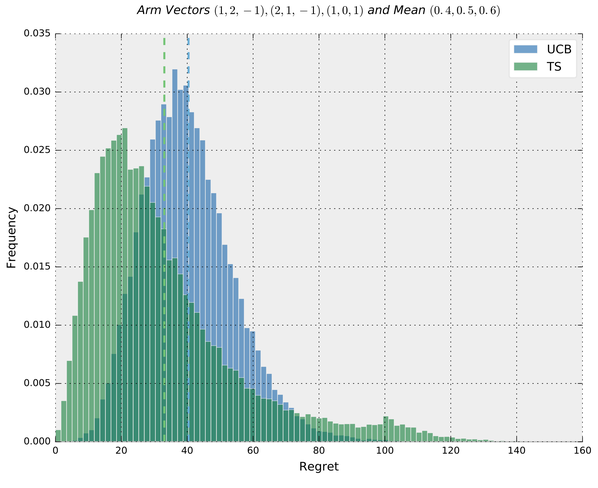


Figure 1: Histogram comparing two Linear Bandit algorithms.

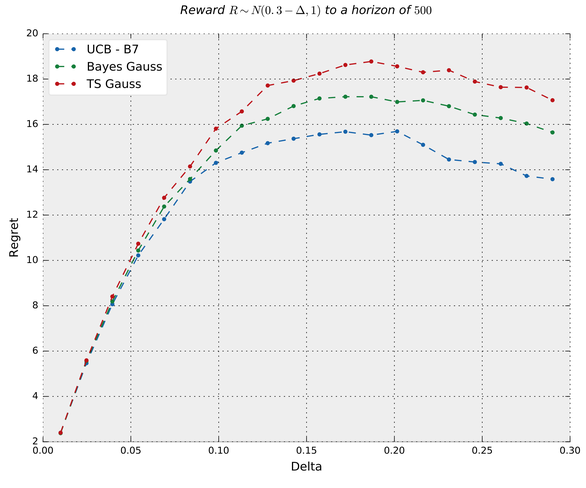


Figure 2: Line plot of regret against delta.

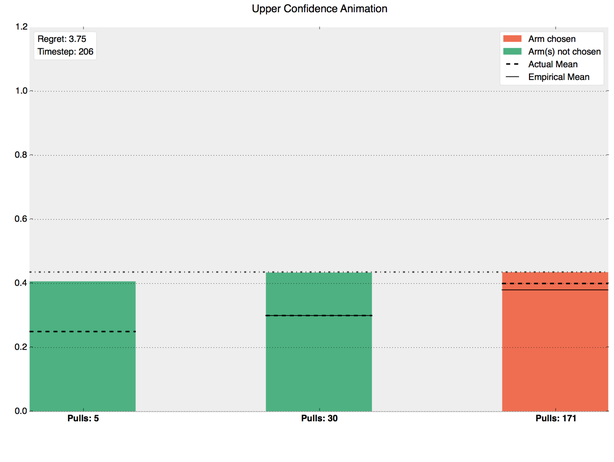


Figure 3: A screenshot from a confidence bound animation.

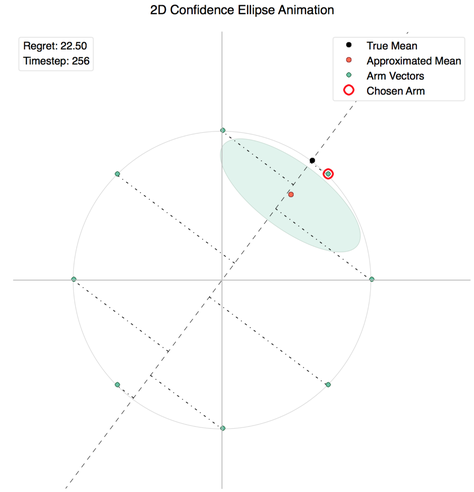


Figure 4: The confidence ellipsoid used by Linear UCB.