Date created: 2025-07-21-Mon

## 2.10 Summary

---

We have considered several ways of balancing exploration and exploitation:

- $\epsilon$-greedy methods choose randomly a small fraction of the time,
- UCB methods choose deterministically but subtly favor actions that are underexplored,
- gradient algorithms estimate not action values but action preferences, and favor preferred actions in a graded, probabilistic manner using a softmax distribution, and
- simply initializing optimistically causes even greedy methods to explore significantly.

To compair these methods, we can run a *parameter study*, i.e. running each method over a range of parameter values to produce *learning curves*, then averaging and compairing these learning curves.

In the parameter study, we not only need to pay attention to how well each algorithm does at its best parameter value, but also how sensitive it is to its parameter value. Overall, it seems that UCB performs best on this task.

Despite their simplicity, methods studied in this chapter could arguably be considered state-of-the-art for bandit problems. More sophisticated methods have complexity or assumptions that make them impractical. But better methods do exist for balancing exploration with exploitation. These *Bayesian* methods (e.g. Gitten index methods) assume knowledge of prior distributions which get updated at every step. In general, these update conputations get very complex, but for certain special distributions (called *conjugate priors*) they can be easy. One possibility then is to select actions at each step according to their posterior probability of being the best action (*posterior sampling* or *Thompson sampling*). In the Bayesian setting it is even conceivable to compute the *optimal* balance between exploration and exploitation, where one computes for each possible action the probability of each possible immediate reward and resultant posterior distributions. This evolving distribution becomes the information state of the problem, and might require one to consider all possible actions, all possible rewards, and all possible next actions, and so on for all steps. Although exact computation is not feasible for most problems, efficient approximation is possible. This would effectively turn the bandit problem into the full RL problem.

### Exercises

#### 2.11: Programming

Make a figure analogous to Figure 2.6 for the nonstationtry case outlined in Exercise 2.5. Include the constant-step-size $\epsilon$-greedy algorithm with $\alpha = 0.1$. use runs of 200,000 steps and, as a performance measure for each algorithm and parameter setting, use the average reward over the last 100,000 steps.

**Solution** As shown, constant-step-size $\epsilon$-greedy algorithm performs best.

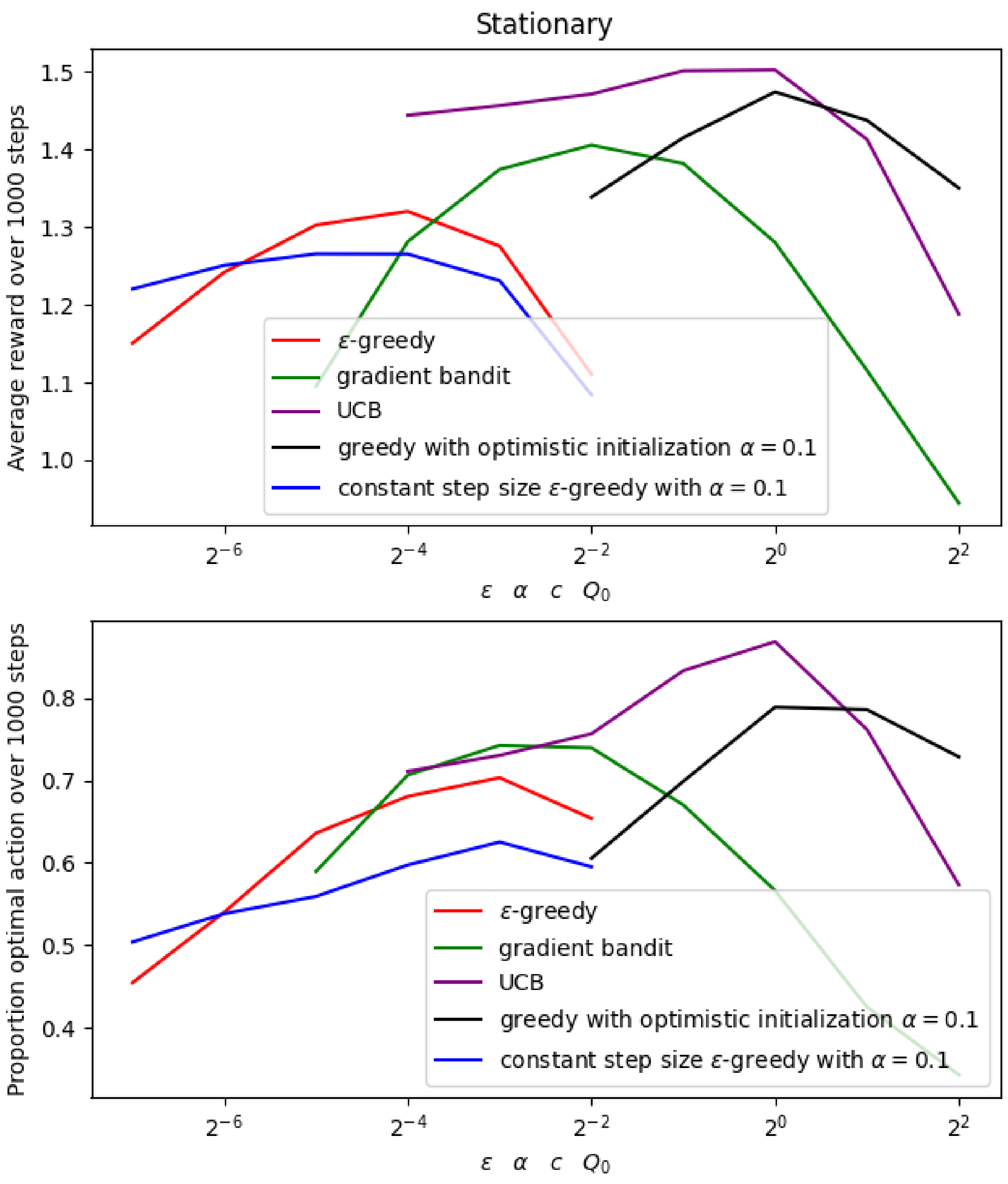

Figure 1: Parameter study, stationary bandit, averaged over 1000 steps

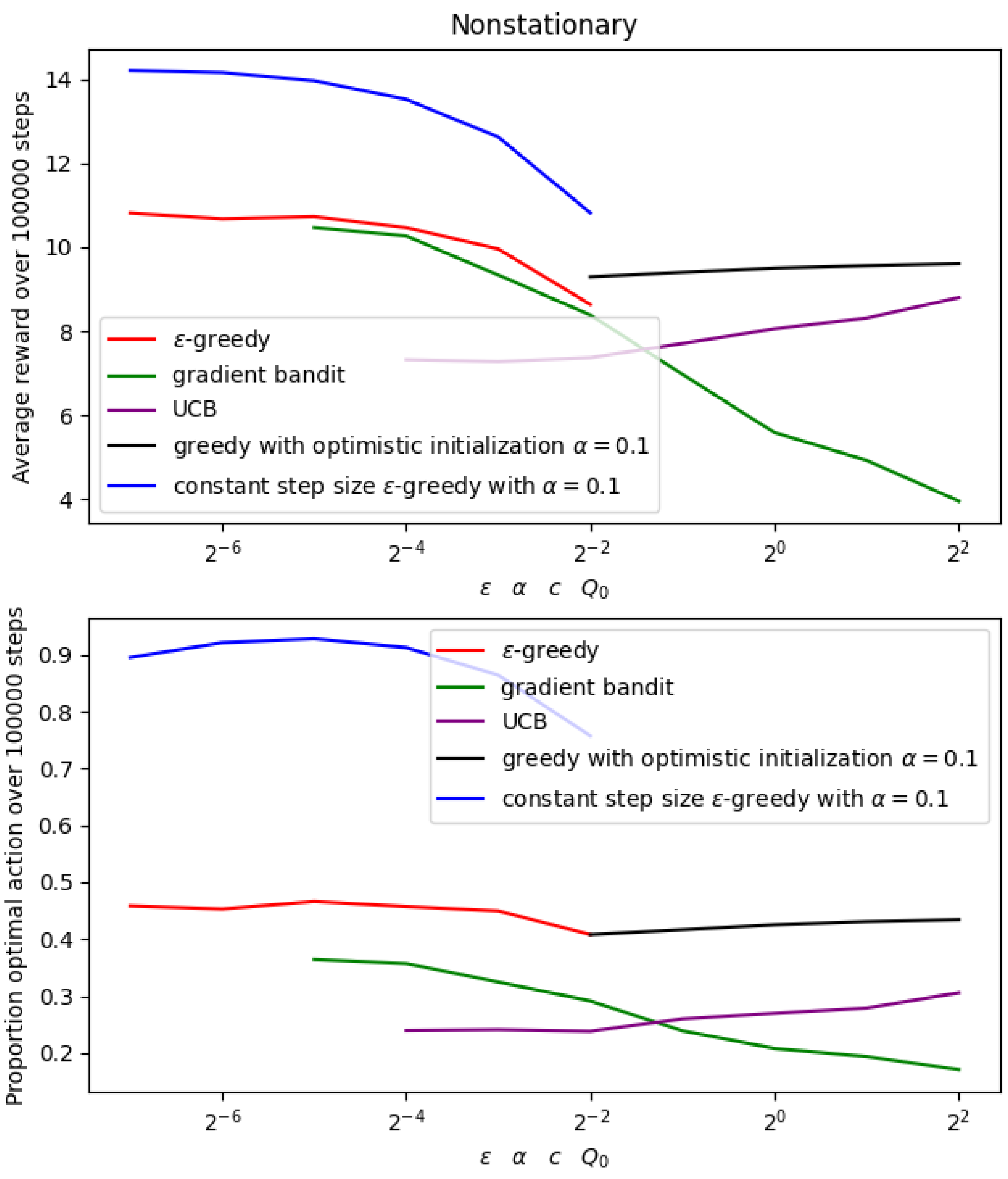

Figure 2: Solution to exercise 2.11, 2,000 runs of 200,000 steps, averaged over the final 100,000 steps.