Reinforcement Learning - TP2
Q-Learning
Stochastic Bandit Algorithms

Chia-Man Hung

November 27, 2016

1 Q-LEARNING

The Q-Learning algorithm simulates trajectories using

$$Q_{t+1}(x_t, a_t) = (1 - \alpha_{N(x_t,a_t)}(x_t, a_t))Q_t(x_t, a_t) + \alpha_{N(x_t,a_t)}(x_t, a_t)[r_t + \gamma \max_{b \in A} Q_t(x_{t+1}, b)] \quad (1.1)$$

where

- N(x, a) is the number of visits of the state-action (x, a)
- for each (x, a), $\alpha_i(x, a)$ is a sequence of *stepsizes*
- in state x_t, a_t is chosen based on some *exploration policy*

For the algorithm to converge, the stepsizes and exploration policy should be chosen such that all state-action pairs are visited infinitely often and $\alpha_i(x, a)$ satisfies the usual stochastic approximation requirements:

$$\sum_i \alpha_i(x, a) = +\infty \text{ and } \sum_i \alpha_i^2(x, a) < +\infty$$

Q1. Describe the parameters of the exploration policy and learning rate chosen, and illustrate the convergence of Q-Learning using some of the following performance metrics.

- Performance in the initial state $|V^*(I) - V^{\pi_n}(I)|$, where π_n is the greedy policy w.r.t. Q_n at the end of the n-th episode
- Performance over all the other states $||V^* - V^{\pi_n}||_\infty$
- Reward accumulated over the episode

A1. We choose the length of episodes $T_{max} = 1000$, $\epsilon = 0.01$, $\alpha = \frac{1}{N}$.

Pseudocode:
for $i = 1, ..., n$
    $x_0 = 1$
    for $t = 1, ..., T_{max}$
        $a_t = \underset{a}{argmax} Q(x_t, a)$ with probability $1 - \epsilon$, $a_t = a \sim U[A]$ with probability ϵ
        $N(x_t, a_t) += 1$
        simulate x_{t+1} based on x_t and a_t and compute r_t
        $\delta_t = r_t + \gamma \underset{a'}{max} Q(x_{t+1}, a') - Q(x_t, a_t)$
        $Q(x_t, a_t) += \frac{\delta_t}{N(x_t, a_t)}$
    endfor
endfor

This is implemented in *q-learning.m*.

Figure 1.1: Performance in the initial state $|V^*(I) - V^{\pi_n}(I)|$.

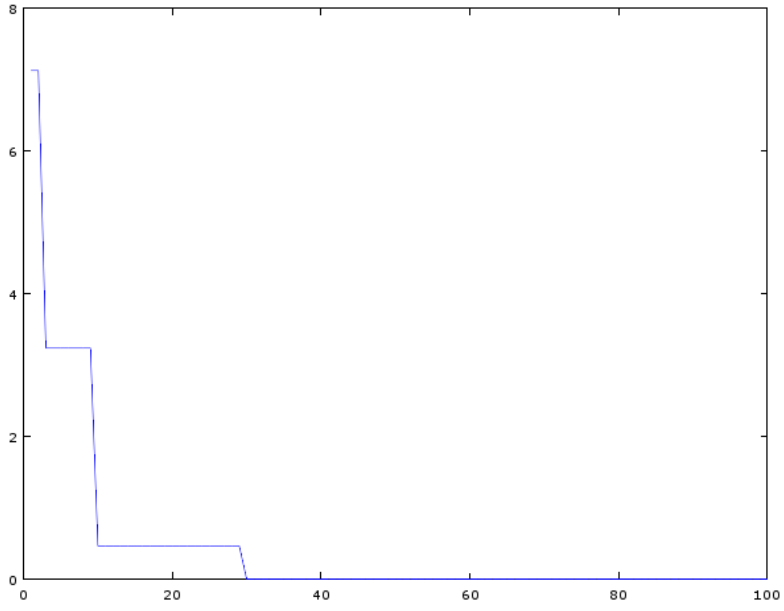


Figure 1.2: Performance over all the other states $||V^* - V^{\pi_n}||_\infty$.

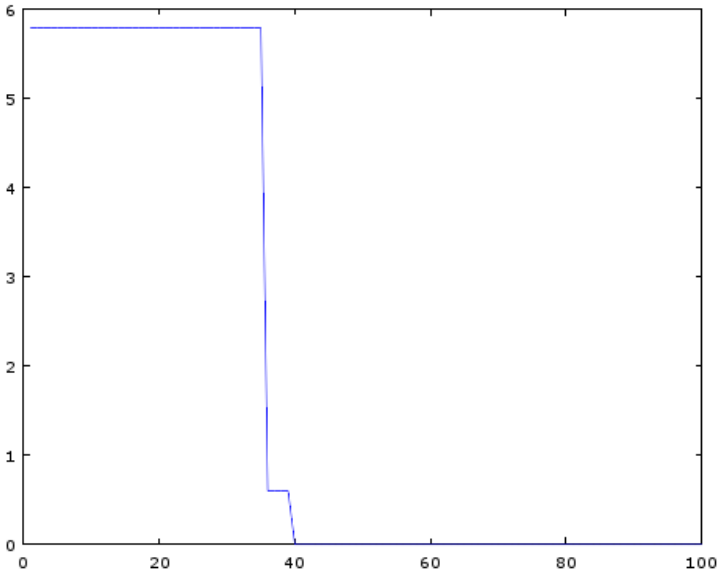
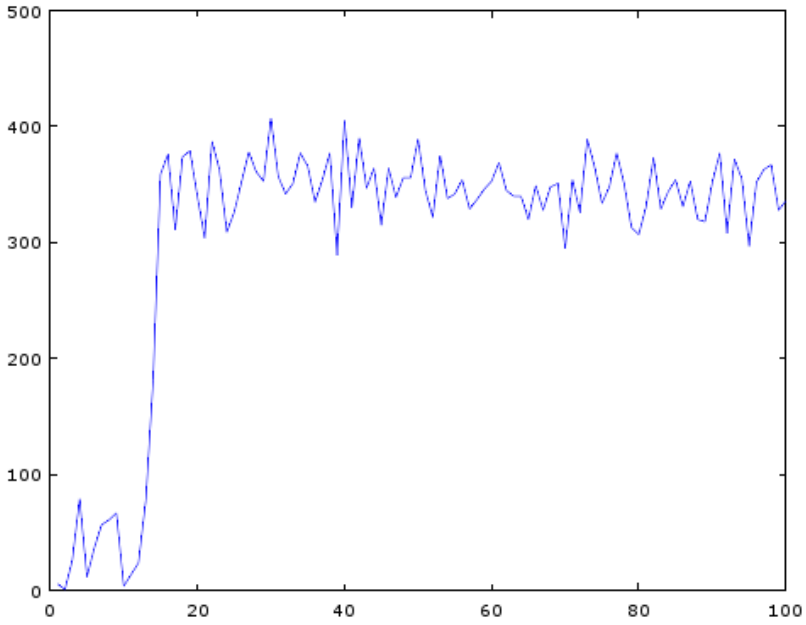


Figure 1.3: Reward accumulated over the episode.



2 Stochastic Multi-Armed Bandits on simulated data

2.1 Bernoulli bandit models

Q2. For two different Bernoulli bandit problems, with different complexity, compare the regret of Thompson Sampling with that of UCB1. Add Lai and Robbins' lower bound on your plots.

Figure 2.1: Regret of one run with UCB1 (blue) and TS (green). MAB = {Ber(0.3), Ber(0.25), Ber(0.2), Ber(0.1)}.

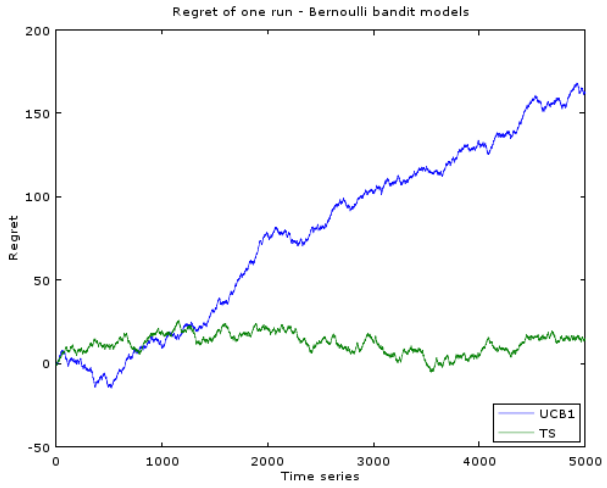


Figure 2.2: Expectation of the regret with UCB1 (blue) and TS (green). Oracle in red. MAB = {Ber(0.3), Ber(0.25), Ber(0.2), Ber(0.1)}.

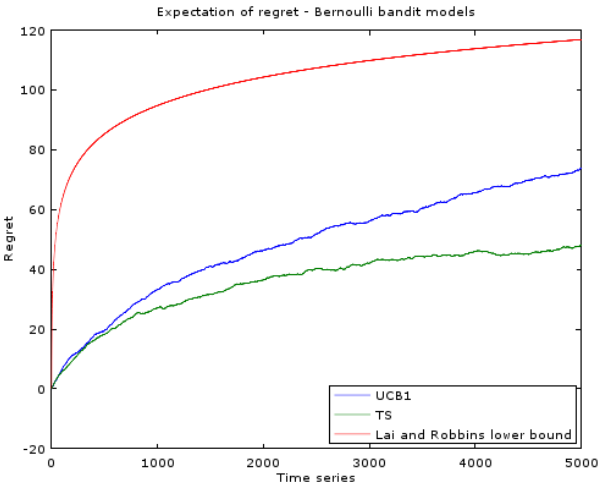


Figure 2.3: Regret of one run with UCB1 (blue) and TS (green). MAB = {Ber(0.6), Ber(0.5), Ber(0.4), Ber(0.2)}.

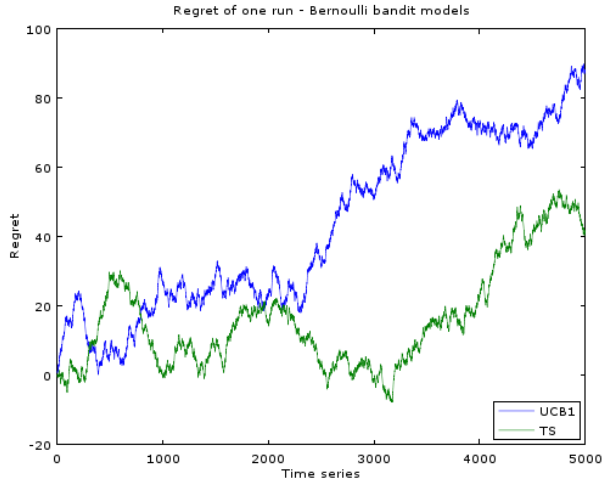
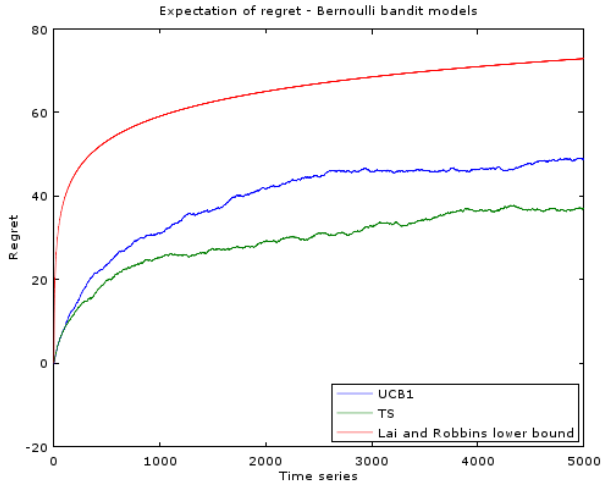


Figure 2.4: Expectation of the regret with UCB1 (blue) and TS (green). Oracle in red. MAB = {Ber(0.6), Ber(0.5), Ber(0.4), Ber(0.2)}.



2.2 Non-parametric bandits (bounded rewards)

Q3. Describe the proposed implementation of Thompson Sampling, and present a regret curve in a bandit model that you specify. Does the notion of complexity still make sense?

A3. The UCB1 algorithm can be used in any bandit model such that each arm is bounded in [0, 1], without modification. In the following, we propose an adaptation of Thompson sampling to handle non-binary rewards. This is implemented in *TS_adaptation.m* and tested

with a multi-armed bandit model of exponential arms. Note that the exponential arms are bounded in [0, 1], so we recomputed its mean, variance, and sample in *armExp.m*.

Adaptation of Thompson sampling:
initialize N and S (arrays of size nbArms) to 0
for t = 1, ..., T
  for arm = 1, ..., nbArms
    sample θ_{arm} from Beta($S_{arm}+1, N_{arm}-S_{arm}+1$)
    play $arm\_best = \underset{arm}{argmax}\,\theta_{arm}$ and observe reward r_temp
    perform a Bernoulli trial with success rate r_temp and observe output r
    $N_{arm\_best} \leftarrow N_{arm\_best}+1$
    $S_{arm\_best} \leftarrow S_{arm\_best}+r$
  endfor
endfor

This is taken from *Analysis of Thompson Sampling for the Multi-armed Bandit Problem, Shipra Agrawal & Navin Goyal (2012)*. The Lai and Robbins lower bound still holds and the notion of complexity still makes sense.

Figure 2.5: Regret of one run with UCB1 (blue) and an adaptation of TS (green). MAB = {Exp(0.3), Exp(0.25), Exp(0.2), Exp(0.1)}.

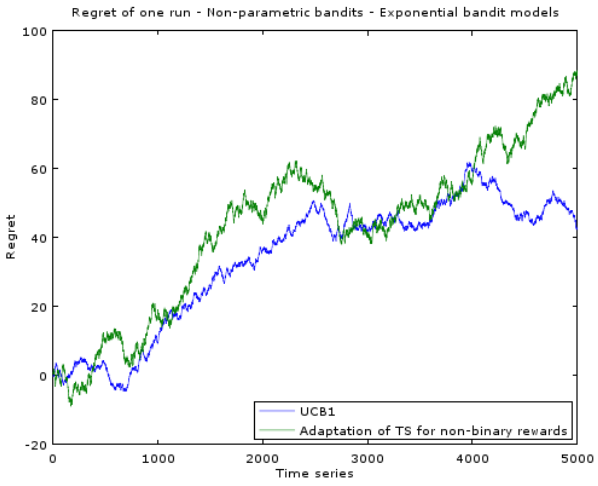


Figure 2.6: Expectation of the regret with UCB1 (blue) and an adaptation of TS (green). Oracle in red. MAB = {Exp(0.3), Exp(0.25), Exp(0.2), Exp(0.1)}.

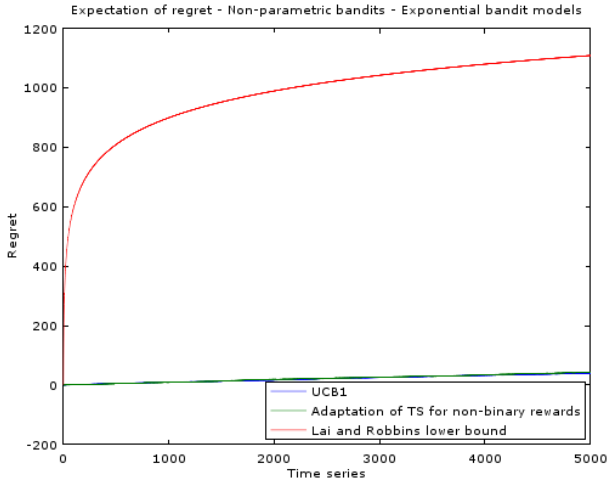


Figure 2.7: Regret of one run with UCB1 (blue) and an adaptation of TS (green). MAB = {Exp(0.6), Exp(0.5), Exp(0.4), Exp(0.2)}.

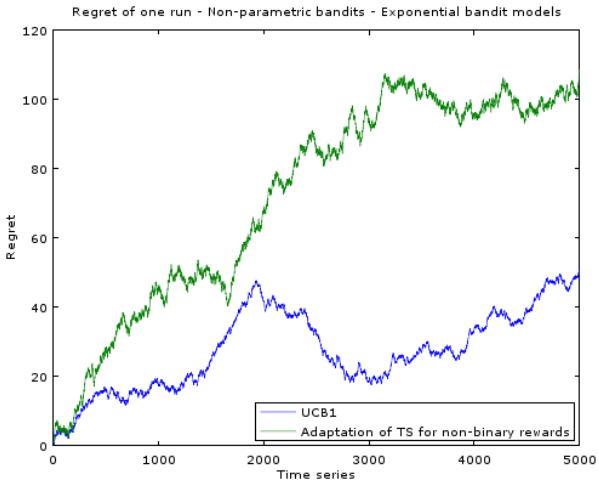


Figure 2.8: Expectation of the regret with UCB1 (blue) and an adaptation of TS (green). Oracle in red. MAB = {Exp(0.6), Exp(0.5), Exp(0.4), Exp(0.2)}.