Reinforcement Learning

Jonathan Somer

April 29, 2018

# Reinforcement Learning In Python Course

## 1 Introduction

### SAR:

- Start in state $S_t$
- Apply action $A_t$
- Get reward $R_{t+1}$

### SAS:

- Start in state $S_t$
- Apply action $A_t$
- Move to state $S_{t+1}$

## 2 Return of the Multi-Armed Bandit

### Explore-Exploit Strategies:

#### Epsilon-Greedy:

Constant exploration ratio throughout entire game. Thus, choosing to learn quickly comes at a cost in long games and vice versa.

**Algorithm 1** Epsilon-Greedy Explore-Exploit Strategy

```
for turn do
    draw a random p ∈ [0,1]
    if p < ϵ then
        Explore: play some random bandit and update its predicted p
    else
        Exploit: play the best bandit and update its predicted p
    end if
end for
```

**Efficient Mean Update:**

$$\bar{X}_N = \frac{N-1}{N}\bar{X}_{N-1} + \frac{1}{N}X_N$$

**Optimistic Initial Value :**

By initially setting all predicted means to the upper limit, and then playing exploits-only, this strategy explores the least explored bandits first as they will have the highest predicted probabilities. We will stop exploring once the best bandit is discovered so we will not pay for unnecessary exploration late in the game.

**Algorithm 2** Optimistic Initial Value Explore-Exploit Strategy

```
Set all initial predicted probabilities to the max possible value
for turn do
    Exploit: play the best bandit and update its predicted p
end for
```

**Chernoff-Hoeffding Bound:**

$$P[|\bar{X} - \mu| \geq \epsilon] \leq 2(e^{-2\epsilon^2 N})$$

**UCB1:**

Play exploits only with respect to:

$$X_{UCB-j} = \bar{X}_j + \sqrt{2\frac{ln(N)}{N_j}}$$

Initially we tend to explore the least explored bandits, but as the game goes on we rely more highly on the predicted mean we have arrived at.

**Thompson Sampling:**