Date created: 2025-07-16-Wed

2.4 Incremental Implementation

We estimate action values using sample averages, but as the number of samples grow, storing these sample values and computing these averages can become expensive. How can we make this computationally efficient? We are looking for constant memory and constant per-time-step computation.

Notation

For simplicity we consider the case with only one action.

- R_i: reward received after the ith iteration of *this* action
- Q_n: estimate of its action value after it has been selected $n-1$ times.

It is easy to see that

$$Q_n := \frac{R_1 + R_2 + ... + R_{n-1}}{n-1}$$

Incremental formula

Given Q_n and the nth reward, R_n, the new average of all n rewards can be computed by

$$\begin{aligned} Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\ &= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}\left(R_n + \frac{n-1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}\left(R_n + (n-1)Q_n\right) \\ &= \frac{1}{n}\left(R_n + nQ_n - Q_n\right) \\ &= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big] \end{aligned}$$

General form

This update rule is of a form that occurs frequently in RL:

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize}\big[\text{Taget} - \text{OldEstimate}\big]$$

Where $[Target - OldEstimate]$ is an *error* in the estimate, which we reduce by taking a step towards the Target. The Target is a desirable direction in which to move, which in our example above is the nth reward. Obviously, the Target may be noisy.

Note that StepSize can be variable. In this book we denote this by $\alpha_t(a)$.

A simple bandit algorithm

```
Initialize, for $a = 1, ..., k$ :
    $Q(a) \leftarrow 0$
    $N(a) \leftarrow 0$
$Loop$ :
    $A \leftarrow \begin{cases} \text{argmax}_a Q(a) & \text{with probability } 1-\epsilon, \text{(ties broken randomly)} \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$
    $R \leftarrow bandit(A)$
    $N(A) \leftarrow N(A) + 1$
    $Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$
```