# Chapter 1

# $n$-Armed bandits

## 1.1 Notes

### 1.1.1 $n$-Armed Bandit Problem

We have $n$ different options (actions) representing $n$ different slot machines. Each action has a given reward, sampled from a stationary probability $q(a)$ only dependent on the chosen action $a$. We want to maximize the (expected) total reward over a given (large) time $T$: $\sum t = 1^T R_t$. To do that, we estimate the value $Q_t(a)$ of each action given what we have seen so far. Let $R_t$ the reward at time $t$ and $N_t(a)$ the number of times the action $a$ has been chosen so far.

### 1.1.2 Estimating value

We estimate the value with:

$$Q_t(a) = \frac{R_1 + ... + R_{N_t(a)}}{N_t(a)}$$

with $Q_t(a) = Q_1(a)$ a default value. With $N_t(a) \rightarrow \infty$ we have $Q_t(a) \rightarrow q(a)$.

Step-by-step, this can be calculated using incremental implementation to save computation time:

$$Q_{k+1} = \frac{1}{k} \sum_{i=1} k R_t$$

$$...$$

$$Q_{k+1} = Q_k + \frac{1}{k} (R_k - Q_k)$$

This looks like $\textit{NewEstimate} \leftarrow \textit{OldEstimate}+\textit{StepSize}\,(\textit{Target} - \textit{OldEstimate})$, with $\textit{StepSize} = \frac{1}{k}$ here.

For tasks that never stop this estimation diverges, plus we may be interested in tracking a nonstationary problem. To achieve this, we can introduce a constant step size, that effectively weights recent rewards more heavily:

$$\begin{aligned} Q_{k+1} &= Q_k + \alpha\,(R_k - Q_k) \\ &\dots \\ Q_{k+1} &= (1-\alpha)^k Q_1 + \alpha \sum_{i=1}^{k} (1-\alpha)^{k-i} R_t \end{aligned}$$

As it turns out, this defines a weighted average with weights $(1-\alpha)^k, \alpha(1-\alpha)^k, ..., \alpha(1-\alpha)^{k-i}$ (they sum to 1).

By denoting $\alpha_k(a)$ the weight (step-size) used for the $k$-th selection of action a, we need to have two conditions:

1. $\sum_{k=1}^{\infty} \alpha_k(a) = \infty$, to guarantee that we overcome initial estimate, and

2. $\sum_{k=1}^{\infty} \alpha_k^2(a) < \infty$, to guarantee convergence.

## Chosing actions

To chose the action, the *greedy* way is to select the one with the highest value: $A_t = \operatorname{argmax}_a Q_t(a)$. Problem: this does not spend any time to sample other actions to refine the estimates $Q_t(a)$.

First solution: $\epsilon$-greedy algorithms, where $A_t = \operatorname{argmax}_a Q_t(a)$ $1-\epsilon$ of the times and $A_t = uniform(a)$ the other $\epsilon$ of the times.

Second solution: optimistic initial values, to preferentially select unsampled actions.

Third solution: *Upper Confidence Bound (UCB)* action selection, with

$$A_t = \operatorname*{argmax}_a \left( Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right)$$

Rationale: the square-root term is a measure of the uncertainty (or variance) in the estimate of the value of $a$. The higher this term (and c), the higher the chance the action will be taken instead of the optimal action. UCB is often hard to transpose outside of the $n$-armed bandit problem.

**Gradient bandits**

Instead of estimating the value of each action, we can estimate the relative preference $H_t(a)$ of one action over others. We then compute the probability of taking action $a$ using the softmax distribution (softmax "normalizes" its inputs so that any constant added to all preferences has no effect on the probability):

$$Pr(A_t = a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{n} e^{H_t(b)}} = \pi_t(a)$$

Initially, $H_1(a) = 0$, so every action has the same probability to be chosen. We then use a variation on stochastic gradient ascent to update the preference after each step ($A_t$ is the action taken at step $t$ and $R_t$ the corresponding reward):

$$\begin{aligned} H_{t+1}(A_t) &= H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t)) && \text{and} \\ H_{t+1}(a) &= H_t(a) + \alpha(R_t - \bar{R}_t)\pi_t(A_t) && \forall a \neq A_t \end{aligned}$$

Explanation: if the reward ($R_t$) is better than the average reward observed thus far ($\bar{R}_t$), we increase the probability to select $A_t$ and decrease the probabilities of all other actions in proportion to that difference ($R_t - \bar{R}_t$). On the contrary, if the reward is lower than the current average reward, we decrease $\pi_t(A_t)$ and increase all others. We increase/decrease more if the selected action had lower probability (thus the $1 - \pi_t(A_t)$ term).

## 1.2 Exercises

*In the comparison in fig 2.1, which method will perform best in the long run in terms of cumulative reward, and cumulative probability of selecting the best action?*

Because of the law of large numbers, in the long run we have $Q_t(a) \approx q(a)$ for $\epsilon$-greedy methods, since each actions will have been sampled a very large number of times. This is of course not true for the greedy method which correctly sample only one action, the one it always choses. For $\epsilon$-greedy method, in the long run we select the action $A^* = \text{argmax}_a\, q(a)$ $(1-\epsilon)$ of the time and a random action $\epsilon$ of the time. There doesn't seem to be any exact function that gives the expectation of the maximum of $n$ iid normal variables, I could only find inequalities for large $n$... I computed the value for $n = 10$ using 1 million series of 10 iid normal variables, and got a value of $\approx 1.54$

The average reward when selecting action $A^*$ is $\mathbb{E}[q(A^*)] \approx 1.54$. Meanwhile, let $Z$ be the reward when selection a random action, and $Y \sim \mathcal{N}(0,1)$ the noise term added to $q(a)$ when computing the reward. We have:

$$\mathbb{E}[Z] = \mathbb{E}[\mathbb{E}[q(a)] + Y] \qquad = \mathbb{E}[q(a)] + \mathbb{E}[Y] = 0,$$

since both random variables $q(a)$ and $Y$ follow a standard normal distribution. All in all, for $\epsilon$-greedy method, the average expected reward is:

$$\mathbb{E}[\bar{R}] = \epsilon\mathbb{E}[Z] + (1-\epsilon)\mathbb{E}[q(A^*)] \approx (1-\epsilon) \times 1.54$$

Which means 1.39 for $\epsilon = 0.1$ and 1.52 for $\epsilon = 0.01$.

For the true greedy method, the first action $A^\dagger$ for which the reward gets over 0 gets chosen everytime (at first approximation). Thus we need to find $\mathbb{E}[q(a)|q(a)+Y > 0]$. I could not manage to find a seemingly correct formula unfortunately (but this one should not be too hard for a statistician), so I once again computed an estimate for this number for $n = 10$, and came up with $\approx 0.56$. This looks a bit weird since in fig 2.1 the average reward of the greedy method is around 1.