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 $N_t(a) \to \infty$ we have $Q_t(a) \to q(a)$.

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

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

$$...$$

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

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

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

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

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

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

Exercise 2.1 *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 ϵ-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 ϵ-greedy method, in the long run we select the action $A^* = \operatorname{argmax}_a q(a)$ $(1-\epsilon)$ of the time and a random action ϵ 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 ≈ 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:

$$\begin{aligned}\mathbb{E}[Z] &= \mathbb{E}[\mathbb{E}[q(a)] + Y] \\ &= \mathbb{E}[q(a)] + \mathbb{E}[Y] \\ &= 0\end{aligned}$$

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

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

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 $\mathbb{E}[q(A^\dagger)] = \mathbb{E}[q(a)|q(a) + Y > 0]$. 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 $\mathbb{E}[q(A^\dagger)] \approx 0.56$. This looks a bit weird since in fig 2.1 the average reward of the greedy method is around 1.

For ϵ-greedy method, the probability of selecting the best action when we decided to chose a random action is obviously $\frac{1}{n}$. As for when we decide to chose the action with the highest estimated value, I get stuck. We need to find, given $(Y, Z) \sim \mathcal{N}(0,1)^2$, the probability $Pr\{q(a) + Y > q(A^*) + Z\}$. That honestly seems daunting to me and I would not even know where to begin anyway... Graphically, the curve seems to plateau at around 80% for $\epsilon = 0.1$, and should be plateauing over this value for $\epsilon = 0.01$. Same for the greedy method, which seems to plateau at around 35%.

Exercise 2.2 *Give pseudocode for a complete algorithm for the n-armed bandit problem. Use greedy action selection and incremental computation of action values with* $\alpha = \frac{1}{k}$ *step-size parameter. Assume a function bandit(a) that takes an action and returns a reward. Use arrays and variables; do not subscript anything by the time index t. Indicate how the action values are initialized and updates after each reward. Indicate how the step-size parameters are set for each action as a function of how many times it has been tried.*

1. Initialization
   $Q(a) \leftarrow 0$ for all $a \in \mathcal{A}$
   $k(a) \leftarrow 1$ for all $a \in \mathcal{A}$

2. Action selection and value update
   Repeat
   $A \leftarrow \text{argmax}_a Q(a)$ (resolve ties randomly)
   $R \leftarrow bandit(A)$
   $Q(A) \leftarrow Q(A) + \frac{1}{k}(R - Q(A))$
   $k(A) \leftarrow k(A) + 1$
   until end of epoch

Exercise 2.3 *If the step-size parameters, α_k, are not constant, then the estimate Q_k is a weighted average of previously received rewards with a weighting different from that given by (2.6). What is the weighting on each prior reward for the general case, analogous to (2.6), in terms of α_k?*

$$\begin{aligned}
Q_{k+1} &= Q_k + \alpha_k(R_k - Q_k) \\
&= \alpha_k R_k + (1 - \alpha_k)Q_k \\
&= \alpha_k R_k + (1 - \alpha_k)[\alpha_{k-1}R_{k-1} + (1 - \alpha_{k-1})Q_{k-1}] \\
&= \alpha_k R_k + \alpha_{k-1}(1 - \alpha_k)R_{k-1} + (1 - \alpha_k)(1 - \alpha_{k-1})Q_{k-1} \\
&= Q_1 \prod_{i=1}^{k}(1 - \alpha_i) + \sum_{i=1}^{k} \left(\alpha_i R_i \prod_{j=i+1}^{k} (1 - \alpha_j) \right)
\end{aligned}$$

The weights are then $\prod_{i=1}^{k}(1-\alpha_i)$ for Q_1 and $\alpha_i R_i \prod_{j=i+1}^{k}(1-\alpha_j)$ for R_i.

Exercise 2.4 (programming experiment) *Design and conduct an experiment to demonstrate the difficulties that sample-average methods have for nonstationary problems. Use a modified version of the 10-armed testbed in which all the q(a) start out equal and then take independent random walks. Prepare plots like Figure 2.1 for an action-value method using sample averages, incrementally computed by $\alpha = \frac{1}{k}$, and another action-value method using a constant step-size parameter, $\alpha = 0.1$. Use $\epsilon = 0.1$ and, if necessary, runs longer than 1000 plays.*

See file *exercise_2_4.py* for the Python code. The code is composed with 3 classes (2 for bandit problem definition, Bandit and DriftBandit, and 1 for a general action-value method, EpsGreedyLearner) and 2 functions (run_drift and experiments) to conduct the experiment.

Figure 1.1.left shows that a fixed step-size consistently outperforms the "average" step-size. Figure 1.1.right clearly shows the difficulty of the action-value method to track a non-stationnary problem. Indeed, the Figure 2.1 in

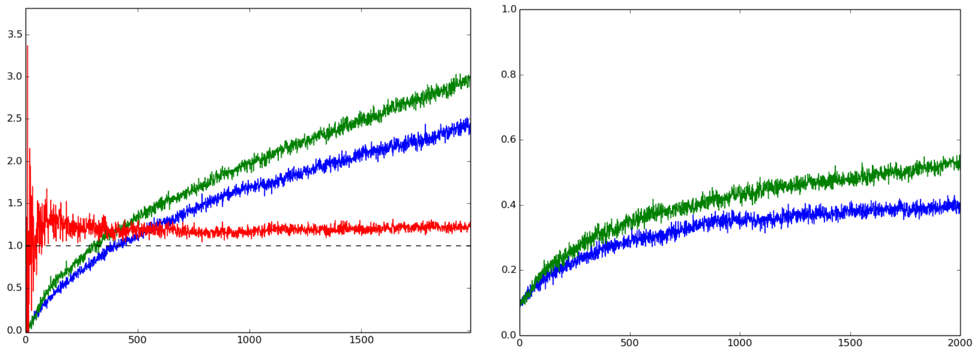

Figure 1.1: Left: average rewards (blue: $\alpha = \frac{1}{k}$, green: $\alpha = 0.1$, red: green/blue). Right: probability of selecting the best action (blue: $\alpha = \frac{1}{k}$, green: $\alpha = 0.1$).

the original manuscript reports probabilities of selecting the best action to be around 80% in the long-run for $\epsilon = 0.1$. For the same ϵ value, this method only reports probabilities of 50-60% – although this number could still increase with a higher number of time steps.