FAKULTÄT FÜR WIRTSCHAFTSINFORMATIK
UND WIRTSCHAFTSMATHEMATIK

UNIVERSITÄT MANNHEIM

Prof. Dr. Leif Döring
Sara Klein, Benedikt Wille

Reinforcement Learning

2. Exercise Sheet

1. Sub-Gaussian random variables

Recall Definition 1.3.3. of a σ-sub-Gaussian random variable X.

a) Show that every σ-sub-Gaussian random variable satisfies $\mathbb{E}[X] = 0$ and $\mathbb{V}[X] \leq \sigma^2$.

b) Suppose X is σ-sub-Gaussian. Prove that cX is $|c|\sigma$-sub-Gaussian.

c) Show that $X_1 + X_2$ is $\sqrt{\sigma_1^2 + \sigma_2^2}$-sub-Gaussian if X_1 and X_2 are independent σ_1-sub-Gaussian and σ_2-sub-Gaussian random variables.

d) Show that a Bernoulli-variable is $\frac{1}{2}$-sub-Gaussian.

e) Show that every centered bounded random variable, say bounded below by a and above by b is $\frac{(b-a)}{2}$-sub-Gaussian.

2. Regret Bound

Recall the upper bound on the regret for ETC in the case of two arms from the first exercise sheet. Show that

$$R_n(\pi) \leq \Delta + C\sqrt{n}$$

for some model-free constant C so that, in particular, $R_n(\pi) \leq 1 + C\sqrt{n}$ for all bandit models with regret bound $\Delta \leq 1$ (for instance for Bernoulli bandits).
Hint: First show that Equation (1.2) in the lecture notes is true and then use the same trick as in the proof of Theorem 1.3.9.

3. Advanced: ϵ-greedy Regret

Let π the learning strategy that first explores each arm once and then continuous according to ϵ-greedy for some $\epsilon \in (0,1)$ fixed. Furthermore, assume that ν is a 1-sub-gaussian bandit model. Show that the regret grows linearly:

$$\lim_{n\to\infty} \frac{R_n(\pi)}{n} = \frac{\epsilon}{K} \sum_{a\in\mathcal{A}} \Delta_a$$

4. Programming task*: ε-greedy and UCB

In this task, we want to implement the ε-greedy algorithm and the UCB algorithm of the lecture for the multi-armed bandit problem. As a reminder, we have written the two algorithms again on the exercise sheet.

(a) Suppose we have a Gaussian bandit with 10 arms that walks for 1000 steps. Implement the ε-greedy algorithm and plot the regret and percentage of optimal actions for different ε-configurations. Perform the same experiment with the Bernoulli Bandit. Are there any differences?

(b) Implement the UCB algorithm. Compare your results for different Gaussian Bandits (especially different variances) and use different δ. Especially compare different prefactors for the term $\log(1/\delta)$ with $\delta = \frac{1}{n^2}$ as discussed in the lecture. As a reminder, the UCB algorithm is given by

$$\mathrm{UCB}_a(t,\delta) = \begin{cases} \hat{Q}_a(t) + \sqrt{\frac{2\log(1/\delta)}{T_a(t)}} & , T_a(t) \neq 0 \\ \infty & , T_a(t) = 0 \end{cases}.$$

In addition, for the Bernoulli Bandit use the modified UCB for σ-subgaussian Bandits from the skript. Compare again different values for σ.

(c) In the lecture, the regret bound

$$R_n \leq 3 \sum_{a \in \mathcal{A}} \triangle_a + 16 \log(n) \sum_{a : a \neq a*} \frac{1}{\triangle_a}$$

for $\delta = \frac{1}{n^2}$ was derived. Add this plot to the experiment from the previous experiment.

Input : Parameter δ, number of total timesteps n and number of arms k
Output: Trajectory of Rewards $(X_t)_{t \in \{1,\dots,n\}}$ and actions $(A_t)_{t \in \{1,\dots,n\}}$
begin
  for $t \leftarrow 1$ **to** n **do**
    Choose action $A_t = \arg\max_i \mathrm{UCB}_i(t-1,\delta)$;
    Observe reward X_t and update the upper confidence bounds;
  end
  return $(X_t)_{t \in \{1,\dots,n\}}, (A_t)_{t \in \{1,\dots,n\}}$;
end

Algorithm 1: UCB(δ) algorithm

Input : Parameter ε, number of arms n
Output: Trajectory of Rewards $(X_t)_{t\in\{1,\ldots,n\}}$ and actions $(A_t)_{t\in\{1,\ldots,n\}}$

```
begin
    for t ← 1 to n do
        Choose U ~ 𝒰([0,1]);
        if U < ε then
            Choose A_t uniformly;
            Obtain reward X_t by playing arm A_t;
            Update the estimated action value function Q̂(t);
        end
        else
            Set A_t = arg max_a Q̃_a(t − 1);
            Obtain reward X_t by playing arm A_t;
            Update the estimated action value function Q̃(t);
        end
    end
    return (X_t)_{t∈{1,...,n}}, (A_t)_{t∈{1,...,n}};
end
```

Algorithm 2: ε-greedy algorithm