Advanced Topics in Machine Learning 2015-2016

Yevgeny Seldin Christian Igel Brian Brost

Home Assignment 5

Deadline: Sunday, 18 October, 2015, 23:59

The assignments must be answered individually - each student must write and submit his/her own solution. We encourage you to work on the assignments on your own, but we do not prevent you from discussing the questions in small groups. If you do so, you are requested to list your group partners in your individual submission.

Submission format: *Please, upload your answers in a single* `.pdf` *file and additional* `.zip` *file with all the code that you used to solve the assignment. (The* `.pdf` *should* ***not*** *be part of the* `.zip` *file.)*

IMPORTANT: *We are interested in how you solve the problems, not in the final answers. Please, write down all your calculations.*

Question 1 (Value function - 25 points)**.** The floor plan in Figure 1 represents a 3×4 grid-world. Each of the 12 rooms corresponds to one state. The agent can go from one room to another if the rooms are connected by a door. The actions are the movements $\{\text{up}, \text{down}, \text{right}, \text{left}\}$. Every movement (except in the terminal state) gives a negative reward of -1 (i.e., every movements costs 1). If you bump into a wall without a door, you stay in the same room, but the movement has still to be paid. There are three rooms that give you an *additional* negative reward of -5 and -10, respectively, *when you enter them*, see figure. The room in the bottom right is a terminal state. An episode ends when this room is reached, which can be modelled by every action in this state leaving the state unchanged and having no cost.

This exercise requires an implementation of the MDP. Note that all rewards and transitions are deterministic. They could be described by simple mappings $S \times A \to \mathbb{R}$ and $S \times A \to S$, respectively, which can be encoded by simple tables.

Use an algorithm presented in the lecture to compute the value function V^{rand} of the random policy, that is, the policy that chooses an action uniformly at random in every state. Report the 12 V^{rand} values. Provide the implementation of the algorithm.

Use an algorithm presented in the lecture to compute the optimal value function V^*. Report the 12 V^* values. Provide the implementation of the algorithm.

Question 2 (Gaussian distribution - 25 points)**.** In the upcoming lectures, we will work with multi-variate normal distributions quite a bit. This exercise shall prepare you for that. Solving it will most likely require that you consult books or the Internet for information about matrices and Gaussian distributions.

Let $\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$ denote multi-variate normal distribution with mean $\boldsymbol{m} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$. Accordingly, $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is the distribution of a standard normally distributed random vector with mean $\boldsymbol{0} \in \mathbb{R}^n$ and covariance matrix equal to the identity matrix $\boldsymbol{I} = \text{diag}(1, \dots 1) \in \mathbb{R}^{n\times n}$.

1. Consider $\boldsymbol{a} \in \mathbb{R}^n$.
   (a) Prove that the rank of the matrix $\boldsymbol{C} = \boldsymbol{a}\boldsymbol{a}^{\mathrm{T}}$ equals one.
   (b) Prove that $\boldsymbol{a}$ is an eigenvector of matrix $\boldsymbol{C} = \boldsymbol{a}\boldsymbol{a}^{\mathrm{T}}$. What is the corresponding eigenvalue?
   (c) *Bonus (+5 points):* Which normal distribution with zero mean $\mathcal{N}(\boldsymbol{0}, \boldsymbol{C})$ produces a fixed random vector $\boldsymbol{b} \in \mathbb{R}^n$ with highest probability (i.e., how do we have to choose the covariance matrix $\boldsymbol{C}$ to maximize the probability that a given fixed $\boldsymbol{b}$ is sampled)? (If you cannot figure out which covariance matrix gives you the proper length, that's fine; important is the direction.)

2. Consider m random vectors $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

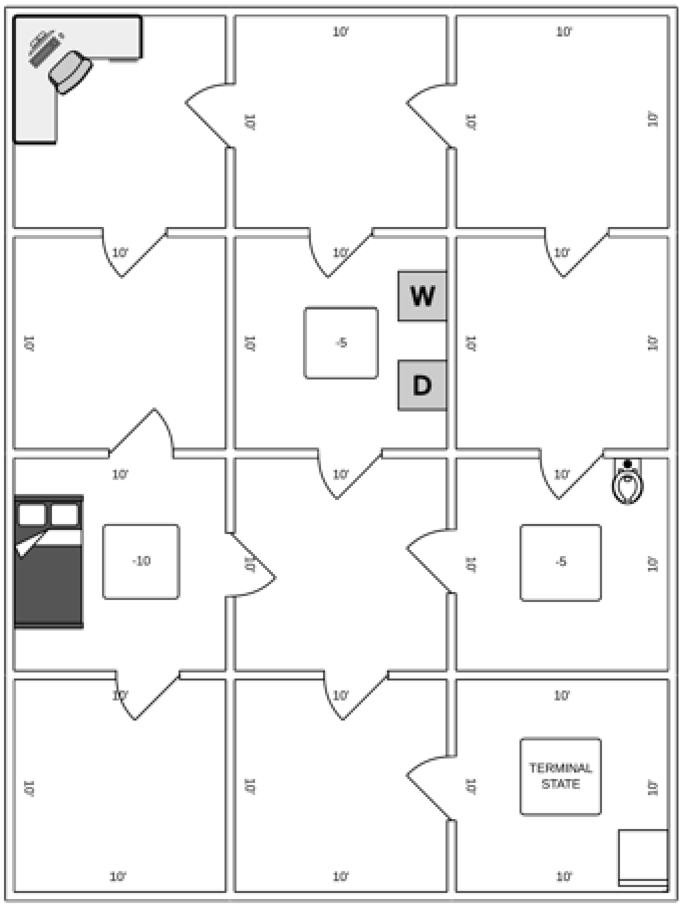

Figure 1: Illustration for Question 1.

(a) How is the random vector

$$\boldsymbol{z} = \sum_{i=1}^{m} \boldsymbol{x}_i$$

distributed?

(b) Now additionally consider m positive real-valued weights $w_1, \ldots, w_m \in \mathbb{R}^+$. How is the random vector

$$\boldsymbol{z}_w = \sum_{i=1}^{m} w_i \boldsymbol{x}_i$$

distributed?

(c) What is (with probability one) the rank of

$$\boldsymbol{C} = \sum_{i=1}^{m} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} \ ?$$

Consider the cases $m < n$ and $m \geq n$.

3. Let $\boldsymbol{a} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\boldsymbol{b} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $c \in [0, 1]$. Show that $(1-c)\boldsymbol{a} + \sqrt{c(2-c)}\boldsymbol{b} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

Question 3 (Empirical evaluation of algorithms for adversarial environments - 25 points)**.** Is it possible to evaluate experimentally the quality of algorithms for adversarial environments? If yes, how would you design such experiment? If no, explain why it is not possible.

Hint: Think what kind of experiments can certify that an algorithm for adversarial environment is good and what kind of experiments can certify that the algorithm is bad? How easy or hard is it to construct the corresponding experiments?

Question 4 (Importance-weighted sampling for Label-efficient prediction and label-efficient bandits - 25 points)**.** In many applications getting labels is expensive. In this question we investigate what can be done in adversarial settings when we are restricted in the number of labels we are allowed to request.

We start with adversarial full information setting. Assume that the game lasts T rounds and you are allowed to observe on average εT columns in the matrix of losses for $0 < \varepsilon \leq 1$ (which means that we can evaluate the quality of experts we are working with on εT out of T rounds). Consider the following playing strategy:

```
Label-Efficient Forecaster
$\forall a : \tilde{L}_0(a) = 0$
for $t = 1, \dots, T$ do
    $\forall a : \ p_t(a) = \frac{e^{-\eta \tilde{L}_{t-1}(a)}}{\sum_{a'} e^{-\eta \tilde{L}_{t-1}(a')}}$
    Sample $A_t$ according to $p_t$ and play it
    Draw Bernoulli random variable $Z_t$ with bias $\varepsilon$
    if $Z_t = 1$ then
        Request to observe $\ell_t^1, \dots, \ell_t^K$
    else
        Make no observations
    end if
    $\forall a : \ \tilde{\ell}_t^a = \frac{\ell_t^a \mathbb{1}_{\{Z_t=1\}}}{\varepsilon} = \begin{cases} \frac{\ell_t^a}{\varepsilon}, & \text{if } Z_t = 1 \\ 0, & \text{otherwise} \end{cases}$
    $\forall a : \ \tilde{L}_t(a) = \tilde{L}_{t-1}(a) + \tilde{\ell}_t^a$
end for
```

1. Show that in expectation the above algorithm will request to observe εT columns.
2. Show that the expected regret of the above algorithm in adversarial setting satisfies $\mathbb{E}[R_T] \leq \sqrt{2\frac{1}{\varepsilon}T \ln K}$. Note that for $\varepsilon = 1$ we are allowed to observe all columns and we recover the Hedge algorithm, whereas for $\varepsilon < 1$ the performance gradually degrades with $\sqrt{\frac{1}{\varepsilon}}$.
3. Show that with high probability the exact number of observed columns will not be significantly larger than εT.
4. How would you modify the algorithm if you had a hard restriction εT on the number of columns that you are allowed to observe? How would you analyze the modified algorithm? (*Hint: you know that if you have run over the budget before the game finished your regret is still bounded by T. So all you have to do is to make sure that you do not run over the budget too often. The exact parameter tuning in this case is a bit nasty, so if you clearly explain your approach at a high-level you will get full points. If in addition you propose a good setting for the parameters of the algorithm you will get up to 5 extra points.*)
5. Now consider adversarial *bandit* setting and assume that you are allowed to observe your own loss on average on εT out of T rounds of the game. Propose an algorithm for this setting and analyze its expected regret.
6. *Bonus (+10 points):* You are mostly welcome to evaluate the algorithms empirically. I.i.d. experiments will be the easiest to construct, but you are also welcome to try some adversarial settings. Please, explain your experimental settings clearly.

Question 5 (*Bonus question: Rewards vs. Losses +25 points*)**.** The original EXP3 algorithm for multiarmed bandits was designed for the game with rewards rather than losses (see Auer et al. (2002, Page 6)). In the game with rewards we have an infinite matrix of rewards $\{r_t^a\}_{a\in\{1,\dots,K\},t\geq 1}$, where $r_t^a \in [0,1]$.

On each round of the game the algorithm plays an action A_t and accumulates and observed reward $r_t^{A_t}$. The remaining rewards r_t^a for $a \neq A_t$ remain unobserved.

On the one hand, we can easily convert rewards into losses by taking $\ell_t^a = 1 - r_t^a$ and apply the EXP3 algorithm we saw in class. On the other hand, a bit surprisingly, working directly with rewards (as Auer et al. did) turns to be more cumbersome and less efficient. The high-level reason is that the games with rewards and losses have a different dynamics. In the rewards game when an action is played its relative quality (expressed by the cumulative reward) increases. Therefore, we need explicit exploration to make sure that we do not get locked on a suboptimal action. In the losses game when an action is played its relative quality (expressed by the cumulative loss) decreases. Therefore, we never get locked on any particular action and exploration happens automatically without the need to add it explicitly (sometimes this is called implicit exploration). The low-level reason when it comes down to the analysis of the algorithm is that it is easier to upper bound the exponent of x for negative x as opposed to positive x.

The original EXP3 algorithm for the rewards game looks as follows, where the most important difference with the algorithm for the losses game is highlighted in red and two additional minor differences are highlighted in blue (we explicitly emphasize that the sign in the exponent changes from "-" to "+"). $\tilde{R}_t$ is used to denote cumulative importance-weighted rewards.

The EXP3 Algorithm for the game with rewards and fixed time horizon

```
1: $\forall a : \tilde{R}_0(a) = 0$
2: for $t = 1, \dots, T$ do
3:     $\forall a : \; p_t(a) = (1 - \eta) \frac{e^{+\eta \tilde{R}_{t-1}(a)}}{\sum_{a'} e^{+\eta \tilde{R}_{t-1}(a')}} + \frac{\eta}{K}$
4:     Sample $A_t$ according to $p_t$ and play it
5:     Observe $r_t^{A_t}$
6:     $\forall a : \; \tilde{r}_t^a = \frac{r_t^a \mathbb{1}_{\{A_t = a\}}}{p_t(a)} = \begin{cases} \frac{r_t^a}{p_t(a)}, & \text{if } A_t = a \\ 0, & \text{otherwise} \end{cases}$
7:     $\forall a : \; \tilde{R}_t(a) = \tilde{R}_{t-1}(a) + \tilde{r}_t^a$
8: end for
```

1. Explain why the analysis of the EXP3 algorithm for the rewards game without the addition of explicit exploration $\frac{\eta}{K}$ in Line 3 of the algorithm would not work. More specifically - if you would try to follow the lines of the analysis of EXP3 with losses, at which specific point you would get stuck and why?

2. How the addition of explicit exploration term $\frac{\eta}{K}$ in Line 3 of the algorithm allows the analysis to go through? (You can check the analysis of the algorithm in Auer et al. (2002, Page 7).)

3. By how much the expected regret guarantee for EXP3 with rewards is weaker than the expected regret guarantee for EXP3 with losses? (Check Auer et al. (2002, Corollary 3.2) and assume that g takes its worst-case value, which is T.)

4. You are mostly welcome to experiment and see whether theoretical analysis reflects the performance in practice. I.i.d. experiments will be the easiest to construct, but you are also welcome to try some adversarial settings.

References

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal of Computing*, 32(1), 2002.

Good luck!
Yevgeny, Christian, & Brian