Part 3 Assignment: Intelligent Decision Making in Online Advertising

CS4210-A – Algorithms for Intelligent Decision Making

March 27, 2020

Introduction

In this assignment we consider an advertiser which presents advertisements to online users while they are browsing the internet. From an advertising point of view it is important that users get exposed to relevant advertisements which increase the likelihood that they buy a product in an online store. There are two aspects which make online advertising a challenging problem. First, users have different interests, and they respond differently to the advertisements they see. Second, typically there is a finite budget available for advertising, which means that online advertising becomes a constrained decision making problem. The main goal of the advertiser is to choose the right ads and an appropriate advertising intensity for each user, such that the number of purchases in an online store is maximized without violating the advertising budget. In this assignment you will explore several algorithms that can be used to solve this decision making problem.

Description of the planning problem

The planning problem corresponding to a user can be formulated using a Markov Decision Process. Below we provide a high-level description of the model.

States The states reflect the levels of customer interest in the advertiser's product, or in the products of competitors. There are 15 states in total, for which a description can be found in the table below. There are three stages that can be identified. In the first stage the user is interested in products in general (states $2, \ldots, 5$). In the second stage the user is interested in products for which advertisements are shown (states $6, \ldots, 9$). In the third stage the user is interested in products of a competitor (states $10, \ldots, 13$). The decision making process always ends in terminal state 14.

State ID	**Description**
0	begin state
1	uninterested
2	general interest
3	searching for products 1
4	searching for products 2
5	searching for products 3
6	interested in products for which ads are shown 1
7	interested in products for which ads are shown 2
8	interested in products for which ads are shown 3
9	buys product for which ads are shown
10	interested in products of competitor 1
11	interested in products of competitor 2
12	interested in products of competitor 3
13	buys product of competitor
14	end state

Actions There are 5 actions representing the levels of advertising intensity. The actions range from 0 (no ads) to 4 (maximum intensity).

Reward There is a positive reward defined for transitioning from purchase state 9 to the end state. This represents a reward for successfully buying an advertiser product.

	$t = 0$	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$	$t = 8$
state	0	3	4	5	6	7	8	9	14
action	1	3	2	4	4	3	4	1	
cost	2	6	4	8	8	6	8	2	
	start		search			interested and conversion			end

Figure 1: Example execution with states, executed actions and associated cost

Cost There is cost associated with presenting advertisements. More intense ads require more money, and increase the probability that the user becomes interested in the advertiser products and progresses in that stage of the decision making process.

Budget There is a global budget L available for advertising. This means that the expected discounted advertising cost should not exceed L.

In the code that we provided there are two models for online users. The child user responds quickly to advertisements but its purchase is typically cheap, which means that reward is low. The adult user is more conservative but tends to buy more expensive products, meaning that expected reward is higher. MDP models for both types of users can be obtained using the methods `UserGenerator.getCMDPChild()` and `UserGenerator.getCMDPAdult()`. These models provide an MDP for the online user, which also includes the state transition probabilities and the rewards. More detailed information about the decision making problem can be found in a paper by Boutilier and Lu (2016)[1]. This paper should be considered as background material.

An example execution of the MDP is provided in Figure 1, which shows a sequence of states, the executed actions and the associated cost. Actions and cost have been shifted to indicate that the action is executed after reaching the state. The agent starts in state 0, and after executing action 1 the process transitions to the search stage in which the user searches for products. After this stage the user becomes interested in the specific product for which advertisements are shown, and it reaches the purchase state. The process ends in the terminal state.

Getting started

We will use an Eclipse project which contains java code to get started.

1. Download `project.zip` from Brightspace.
2. Start Eclipse.
3. Import the project: File → Import → General → Existing Projects into Workspace → Select archive file (`project.zip`) → Click finish.
4. Open Homework.java from the default package, where you can find the main function.
5. If you encounter compilation issues, then you need to check whether the compiler compliance level has been set correctly (Right click on project → Java Compiler). Furthermore, `lib/commons-math3-3.5.jar` needs to be added to the java build path (Right click → Build Path). If you still encounter problems: please check in the project properties that the src folder is added as source folder in the Java Build Path settings, and there should be a JRE configured under Libraries.

Run the example

You can use the example code in `Homework.task0` to see how a model is initialized, and how a policy is used in a simulation. The simulator prints the state for each time step, as well as the action that it executes and the reward the agent gets. The executed policy is a dummy policy which always executes the same action.

[1] Available at `https://research.google.com/pubs/archive/45291.pdf`

Questions

1. **Value iteration for single user**
   (a) (4 points) Implement the value iteration algorithm in `PlanningAlgorithm.solveVI`, such that it returns an optimal unconstrained policy. The returned array `policy` should be defined in such a way that `policy[s][a]` corresponds to the probability to choose action a in state s. The value iteration algorithm that you implement should ignore the costs of the actions. The code to execute value iteration can be found in `Homework.task1`.
   (b) (1 point) What is the optimal expected value (i.e., expected reward) corresponding to the initial state if the discount factor is 0.95?
   (c) (4 points) Investigate the influence of the discount factor to the expected value of the initial state for both the child (`UserGenerator.getCMDPChild()`) and adult (`UserGenerator.getCMDPAdult()`). Visualize this through a plot with the expected value of the initial state as a function of the discount factor.
   (d) (2 points) Explain the difference between the two graphs.
   (e) (2 points) Explain the general trend which can be seen in the graphs, in particular provide reasoning on the expected value approaching 0 for low discount factors.

2. **Constrained planning for single user**

   The value iteration algorithm that you have implemented in the previous task solves an unconstrained planning problem. The algorithm is conceptually easy, but unfortunately it cannot be augmented with additional constraints. In `PlanningAlgorithm.solve` we have implemented the linear programming formulation for Constrained MDPs, which can be used to obtain an optimal stochastic policy for constrained problems.
   (a) (2 points) The provided model does not define costs associated with the actions. Modify the `Homework.task2` method such that costs are assigned to the actions. For this purpose you can use the function `cmdp.assignCost`, as illustrated in the example code. Modify the code in such a way that executing action a has cost $2a$. It is important that you also make this modification for all remaining tasks!
   (b) (1 point) What is the expected discounted cost of the optimal unconstrained policy?
   (c) (2 points) Investigate the influence of the available budget L on the expected value. Visualize this through a plot with the expected value as a function of the available budget L. Make sure to choose an appropriate range for L. `Homework.task3` provides an example on how to solve with a budget limit.
   (d) (2 points) Explain the trend of the graph, and in particular reason about the expected reward approaching the reward given in (1.b).

3. **Constrained planning for multiple users**

   Until now we considered unconstrained and constrained planning for a single child user. In this task we will consider multi-agent planning for a child and adult user, and we consider the following scenarios with $L = 20$:

   - Planning for both users, which does not consider the budget constraint L.
   - Separate planning for both users, based on the budget limit $L/2$ for each user.
   - Multi-agent planning for both users, based on the global budget limit L.

   While answering the following questions, it is important to consider the difference between child users and adult users, as described in the model description above. Example code is provided in `Homework.task4`.
   (a) (2 points) For separate planning, modify the code in such a way that it assigns budget $L/2$ to both users.
   (b) (1 point) What is the expected reward and expected cost of both users? What is the total expected reward?
   (c) (1 point) Does the total expected cost exceed the budget limit L? Why (not)?
   (d) (2 points) For multi-agent planning, what is the expected reward of both users? What is the total expected reward?
   (e) (1 point) Does the total expected cost exceed the budget limit L? Why (not)?
   (f) (2 points) Compare the separate planning and multi-agent planning and explain the differences between the two.

(g) (3 points) Which approach is better to use in practice? Discuss the tradeoffs that need to be made, and the implications on the budget that the advertiser spends on childs and adults.

4. **Scalability of the multi-agent constrained planning**

   In the previous question we have observed the difference between planning for users separately and multi-agent planning. In this task we will further investigate the scalability of the multi-agent approach and explore a practical implication of the way we tackle large-sized instances. Within this question, we assume that a budget of 10 is made available for each individual user (e.g. when a scenario of 5 child and 7 adult users is considered, the available budget is $L = 120$)

   (a) (6 points) Investigate the scalability of the constrained planning in terms of runtime. Do this by considering a different number of users, ranging from 2 to 50 users. Also investigate the influence of the composition of the group (e.g. the number of child versus the number of adult users within the group) on the runtime. Visualize this through plots with the runtime as a function of the number of users and a metric depicting the group composition.

   (b) (4 points) What would be an efficient method to obtain a planning for 1388 child and 2429 adult users, given that solving this directly is infeasible? Explain your method and provide the expected budget that should be allocated to the user and the adult group respectively.

   (c) (4 points) For a more general case where there is a planning to be made for x child and y adult users where a direct calculation of the planning is infeasible, what is a method which can be used to efficiently estimate the budgets to be allocated to each of the user groups in order to obtain the highest expected reward?

Deliverables

Please submit on Brightspace: a report containing the answers to the questions (max 4 pages), and a zip file containing **only** java source files.

References

[1] C. Boutilier and T. Lu. “Budget Allocation using Weakly Coupled, Constrained Markov Decision Processes”. In: *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence.* 2016, pp. 52–61.