Assignment 0 (warm-up): Find a hut

AI2SysProj 2022

Topic:	Basic probabilities
Due on:	May 31, 2022
Version from:	May 19, 2022

1 Task summary

Your agent got lost on a hike. Given a map, plan a way that maximizes the chances of reaching a hut before nightfall. The assignment repository[A0] contains example problems for you to solve.

Objectives

1. Apply basic probabilistic reasoning to a problem,
2. develop an algorithm to solve a non-trivial problem,
3. gain experience in improving the efficiency of an algorithm,
4. implement a small software project from scratch,
5. get to know the AISysProj setup and workflows.

Prerequisites and useful methods

1. Basic probabilistic reasoning: Conditional probabilities, marginalization, Bayes' rule, etc. (Sections 20.2 – 20.6 in the lecture notes).
2. Admin requirements: Follow the onboarding guide to get access to the assignment repository and to get your personal repository.

2 Detailed problem description

Your goal is to plan a way that maximizes the probability of reaching a hut with the help of a map. In this case, a **map** is made up of squares which are annotated with a letter describing the local vegetation: C indicates coniferous trees, D indicates deciduous trees, and M indicates a meadow. Some squares are instead annotated with an H to mark the location of a hut.

```
4x3,2,C
MMCC
CMHM
DDMH
```

Listing 1: An example problem file.

A **problem file** starts with a line describing the map size, the time until nightfall, and the observed vegetation at your location. The remaining file lists the map data. Figure 1 shows an example problem file. The map size is 4×3 and the time until nightfall is 2, which means that the agent still has time to travel 2 squares on the map. Furthermore, the agent believes to be on a square with coniferous trees (C). Unfortunately, the agent confuses coniferous and deciduous trees 20% of the time (but never confuses meadows and trees).

A **path plan** is a sequence of the directions north (N), east (E), south (S), and west (W). For the example problem, the plan should be of length 2 (the time until nightfall). An optimal plan would be SW, which would lead the agent to a hut with a probability of ≈ 0.57.

More problem details:

1. The agent is guaranteed to be somewhere on the map and all locations are a priori equally likely.
2. There are no huts outside the map area.
3. It is perfectly acceptable if the agent reaches a hut before "finishing" the path plan.
4. To keep the problem simple, we do not allow the agent to change the plan along the path (which would be reasonable as new observations could clarify the agent's location).

3 Solution format

The solution to the problem files should be stored in a .csv file consisting of three columns: the file name, an optimal path plan, and the probability of reaching a hut with that plan. The solution to the example problem file would be the entry

```
exampleproblem.txt,SW,0.57142857
```

The assignment repository contains more example problems along with an example solution file that you can use for comparison.

4 What to submit

Your solution should be in a folder assignment0 in your personal repository on GitLab. It should contain:

1. all your code,
2. a README file explaining how to run your code to solve other problem files (including how to install dependencies),
3. a brief summary of how you solved the problem either as a PDF file ($\approx \frac{1}{2}$ page) or as part of your README,
4. a file solutions.csv with your solutions to the problem files

5 Random tips

1. A useful intermediate goal is to compute the probability that you are in a particular square (the probabilities should add up to one).
2. It might be instructive to solve a small problem by hand (for example the problem from Listing 1).
3. The problem files are increasing in difficulty – start with the first ones. In particular, it might be easier if you initially only support the path length 1.
4. For the more difficult problems, performance matters. However, rather than implementing everything in C, you should think about efficient algorithms (my Python+numpy implementation solves all 80 problems in < 10 seconds).

6 Points

Let $n \leq 80$ be the number of solved problem files. Then

$$P = \begin{cases} 0 & n < 20 \\ 40 & 20 \leq n < 40 \\ 60 + \left\lfloor \frac{n-40}{2} \right\rfloor & 40 \leq n \leq 80 \end{cases}$$

is the number of points you get for solving the problems. The reason is that the first 20 problems are simplified (path length of 1), the next 20 problems are "proper" (path length > 1), and the remaining 40 problems are incrementally bigger, requiring more efficient algorithms.

Assuming you have at least a partial solution, you can additionally get up to 20 points for the quality of the submission (README, explanation, ...). The maximum number of points is therefore 100. If the grading scheme doesn't seem to work well, we might adjust it later on (likely in your favor).

References

[A0] *Assignment 0*. URL: `https://gitlab.rrze.fau.de/wrv/AISysProj/ss22/a0-basic-probabilities/assignment` (visited on 05/10/2022).