— GUIDE —

Assignment 2: Catch the Wumpus

AI-2 Systems Project (Summer Semester 2024)

Jan Frederik Schaefer

Friedrich-Alexander-Universität Erlangen-Nürnberg, Department Informatik

This document is intended to help you solve the assignment "Assignment 2: Catch the Wumpus" [AS]. You do not have to read it, but we do recommend to at least take a look at the tips and common issues.

1 A few tips

1. Sometimes it is better to wait with the portal creation until you have a better idea of where the Wumpus is.
2. Use a hidden Markov model for this problem. Make sure you understand what your hidden variables and your observations are. Make sure you understand what algorithms you need (in the past, students often started researching and implementing all kinds of unnecessary algorithms).
3. You can use matrices for the transition and sensor model. The advantage is that the algorithms are then very simple. The disadvantage is that for some sensors it gets a bit more complicated (and the matrices may get very large). Alternatively, you can just implement a function that computes the probabilities when needed.
4. Intuitively, the problem is a second order Markov model, but it can also be modeled as a first order Markov model, which might be easier to implement.
5. You do not have to model everything perfectly – you can still get full points if you cut a few corners (e.g. in the sensor model).
6. If your agent decides to wait with the portal creation, you can store the filtering results to re-use them later on and save computation time (in case that becomes a problem).

2 Common issues

1. Not modelling the problem as a hidden Markov model. This may be tempting (at least for the easier environments), but it does not work well, unless you effectively re-invent something equivalent to a hidden Markov model.
2. Trying to learn the probabilities/parameters. This is unnecessary because you can compute all necessary probabilities based on the information provided in the assignment.
3. Implementing unnecessary HMM algorithms. Make sure you understand what the HMM algorithms do and which ones you actually need.
4. Subtle bugs in the sensor model/transition model that are difficult to track down. It helps to check intermediate results where possible. The server also shows you how likely it thinks you were to catch the Wumpus in your run (due to some optimizations, the result is only an approximation). You can compare that to your own result. Additionally, we now have debug environments that you can use to test your agent (see Section 3).

3 Debug environments

For the first time, we have debug environments available for this assignment. This is experimental – please provide feedback if you find it useful or if you have suggestions for improvements. The goal is that these environments can help you debug your agent if it computes the wrong probabilities. The debug environments are similar to the real environments, but there are a few differences:

1. The initial position of the Wumpus is known (`start-position` field in the action request JSON). Additionally, the previous position is also known (`before-start-position`).
2. In the "real" environments, the Wumpus is observed for 10 time steps before a portal can be made. In the debug environments, this is reduced to 1 time step.
3. The server computes for each time step how likely the Wumpus is in each cell and displays this on the website. This can help with debugging. Note that the probabilities are only approximations.

Tips for using the debug environments:

1. Initially, make the portal right away. To see if the algorithms work over multiple time steps, you can wait increasingly longer before making the portal.

2. For optimization, the server has multiple runs in parallel, which can be confusing while debugging. This can be disabled in the client (see next paragraph).
3. You can only view a run on the server once it's finished (i.e. once a portal has been created).

Tips for using the Python client:

1. You can disable parallel runs by setting `parallel_runs=False` when calling `run`.
2. By default, the client keeps running indefinitely. If you use the `client.py` script, you can set `run_limit=1` to have only one full run. Note that previous interrupted runs will still be completed and not counted towards the limit.

References

[AS] *Assignment 2: Catch the Wumpus*. URL: https://kwarc.info/teaching/AISysProj/SS24/assignment-2.2.pdf.