The Wumpus World:
A Demonstration of Probabilistic Decision Making

Bryan Bugyi

Advised by Jonathon Weisbrod of
Rowan College at Burlington County

May 9, 2017

Abstract

Computer chess programs represent what is perhaps the most recognizable example of a machine mastering a game environment. In the game of Chess, the environment is fully observable; that is to say, in the game of Chess, the computer program (or *agent*) has access to the complete state of the environment at each point in time. In many real-world scenarios, however, the environment in question is not nearly so discernible. In an attempt to better understand decision making in the face of uncertainty, we shall consider the wumpus world environment described by Stuart Russell and Peter Norvig in their conjointly authored text, *Artificial Intelligence: A Modern Approach*. This environment, being both partially observable and stochastic, provides a level of uncertainty not encountered in most board games traditionally associated with AI programming.

1 Introduction

In this environment, an agent is given the task of navigating a 4×4 grid of squares. Some of these squares contain danger in the form of pits and a creature we shall refer to as the wumpus (which will gladly terminate the agent on sight). Not only is the agent unaware of which squares contain danger and which do not, but it is also unaware of how many pits can be expected to be found on the board (this number varies per game). The only information awarded to the agent are the contents of the square it currently resides in and sensory information that provides hints as to what dangers the agent can expect to find in the squares adjacent to it; if a pit or wumpus is in an adjacent square, the agent will be provided with a "wind" or "stench" sense, respectively.

The most common methods of programmatically relaying the knowledge required to successfully navigate such an environment generally fall into one of two categories (though not necessarily exclusively): logical or probabilistic. This paper demonstrates a probabilistic approach. Specifically, this paper focuses on demonstrating solutions to the two main challenges faced when designing such an agent. The first of which being: How do we model our understanding of this environment in such a way that we can easily update the model given new information? And the second of which being: How can our agent best utilize this model to make decisions in the face of uncertainty? We shall address both of these challenges with the help of probability theory, a modest amount of graph theory, and the design of a utility function that makes sense of the final model and decides on an appropriate action.

2 Rules of Wumpus World

Description

The wumpus world is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the terrible wumpus, a beast that eats anyone who enters its room. Some rooms contain

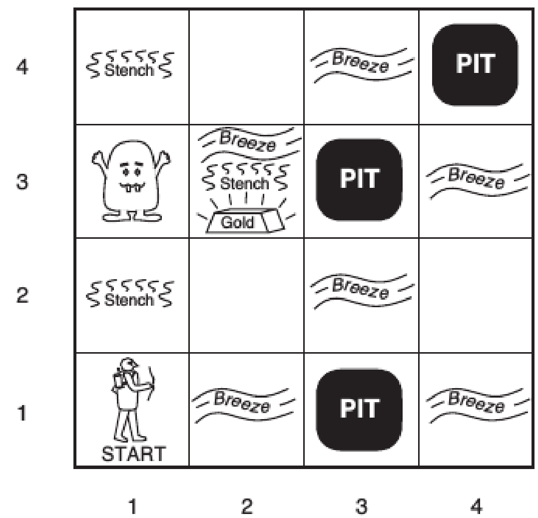

Figure 1: An example wumpus world environment

bottomless pits that will trap anyone who wanders into these rooms. The only mitigating feature of this bleak environment is the possibility of finding a heap of gold.

Environment

The environment is a 4×4 grid of rooms. The agent always starts the game in a corner square. The locations of the gold and the wumpus are chosen randomly, with uniform distribution, from the remaining squares. In addition, each remaining square has a 0.2 probability of being a pit. Figure 1 illustrates the typical features of a wumpus world environment, as we have just described it.

Sensors

If a square is adjacent (not diagonal) to a pit, the agent will sense a "Breeze" when at this square. If the square is adjacent to a wumpus, the agent will sense a "Stench." It is important to emphasize that *the agent is only aware of the contents of the square that it currently resides in.*

3 Probability Derivations

3.1 Wumpus Probability

In this section, we will attempt to uncover a method for deriving the probability of a wumpus being located in square $\boldsymbol{x}$, where $\boldsymbol{x}$ is some square located in the Wumpus World. We shall denote this probability by $P(w_{\boldsymbol{x}})$. First, however, we must establish a few important definitions.

$$E = \{\boldsymbol{x} \mid \boldsymbol{x} \text{ is a square in the environment}\}$$
$$V = \{\boldsymbol{x} \mid \text{The agent has visited } \boldsymbol{x},\ \boldsymbol{x} \in E\}$$

We shall also need familiarity with the concept of a **closed neighborhood** from Graph Theory. Recall that if A is some subset of E, then $\forall \boldsymbol{x} \in E$, $\boldsymbol{x}$ is an element of the closed neighborhood $N[A]$ if and only if $\exists \boldsymbol{a} \in A$ such that $\boldsymbol{x}$ is adjacent to $\boldsymbol{a}$ or $\boldsymbol{x} = \boldsymbol{a}$.

Let $T = \{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m\}$ represent the set of all squares that the agent has visited where a "stench" sense was *not* detected. And, finally, let $S = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$ represent the set of all squares that the agent has visited where a "stench" sense *was* detected (note that S and T partition V).

We can now use S and T to derive the set of all squares where the wumpus could possibly be found (W). If S is nonempty, then W is the difference between the intersection of the closed neighborhoods of all squares in S and the closed neighborhood of T. Otherwise, W is equal to the difference between E and the closed neighborhood of T. That is,

$$W = \begin{cases} N[\boldsymbol{s}_1] \cap \cdots \cap N[\boldsymbol{s}_n] - N[T] & \text{if } S \text{ is non-empty} \\ E - N[T] & \text{otherwise} \end{cases}$$

Therefore, by the axioms of probability, we can deduce that

$$P(w_{\boldsymbol{x}}) = \begin{cases} \frac{1}{|W|} & \text{if } \boldsymbol{x} \in W \\ 0 & \text{if } \boldsymbol{x} \notin W \end{cases}$$

where $P(w_{\boldsymbol{x}})$ represents the probability of a wumpus being located at square $\boldsymbol{x}$.

3.2 Pit Probability

Calculating $P(p_{\boldsymbol{x}})$, the probability that a pit is located at square $\boldsymbol{x}$, is a much more difficult task.

During gameplay, the agent builds a set of models (one for each possible world), and then selectively filters this set as the game goes on. Essentially, it can be shown that $P(p_{\boldsymbol{x}}) = \sum_i^n P(M_{i\boldsymbol{x}})$ for all n models where $\boldsymbol{x}$ contains a pit.

To optimize this process, the agent only models worlds for the squares contained in

$$V' = N[V] - V.$$

This set includes all possible "next move" squares. Attempting to consider all squares in E while modeling possible worlds results in an algorithmic complexity no less than $\mathcal{O}(2^n)$. To understand why this could be problematic, note that a 10×10 environment (like that shown in Figure 4) would require 2^{100} models be built!

4 Utility Function

When deciding on which square to move to next, the agent will consider the square $\boldsymbol{x}$ only if $\boldsymbol{x} \in V'$.

The agent will consider two factors when deciding on the best next square: *death probability* and *distance*, where the former takes precedence over the latter. If all squares considered have a death probability of 0.5 or higher, the agent will forfeit.

Once decided on the next square to travel to, the agent must find the shortest path to said square...

4.1 Graph Structure / Shortest Path

After the next square is chosen, the agent models a portion of the environment using a graph-like structure where the nodes are $\{\boldsymbol{x} \mid \boldsymbol{x} \text{ is the target square or } \boldsymbol{x} \in V\}$.

Figure 2 shows an example of how this structure can be visualized using an undirected graph.

The agent then calculates the shortest path with the help of Dijkstra's algorithm. Figure 2 highlights the shortest path in red.

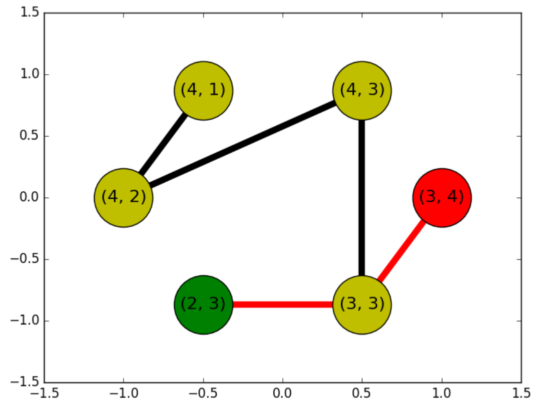

Figure 2: Visualization of the graph structure used to determine the shortest path from $(2, 3)$—the agent's location in figure 3a—to $(3, 4)$—the agent's location in figure 3b.

(a) (b) (c)

Figure 3: A test run showing WW environment along with the agent's perceived death probabilities $[P(w_{\boldsymbol{x}}) + P(p_{\boldsymbol{x}})]$

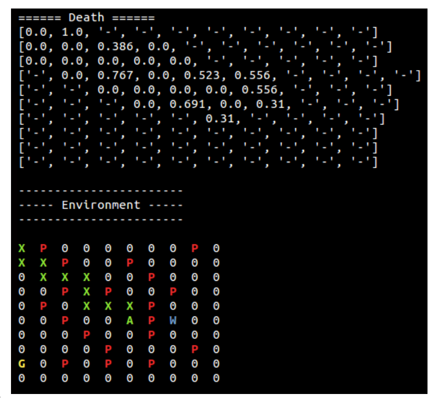

Figure 4: A 10×10 WW environment

5 Conclusion

A 5000 trial test was run on this implementation of the wumpus world agent (using a 4×4 environment). The results were as follows:

- Found Gold: 2568 (51%)
- Flopped: 2013 (40%)
- Death: 237 (4%)
- Forfeited: 182 (3%)

I have defined "flop" in this context to indicate that the agent forfeited on the first move (as it will if it spawns directly adjacent to a pit or wumpus).

Therefore, of the environments that did not result in a flop, the agent found the gold 86% of the time, died 8% of the time, and forfeited 6% of the time.

References

[1] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2009

[2] Sheldon Ross. *A First Course in Probability.* Prentice Hall, 2009

[3] Kenneth H. Rosen. *Discrete Math and Its Applications.* McGraw-Hill, 2012