Learning how to play a game meant for children

Trevor Squires

Last Updated: January 9, 2023

1 Introduction

My wife bought me a relatively simple dice game this past Christmas called "Shut The Box". The game itself could not be easier to describe: you are given a box with tiles numbered 1-9 all in the up position. The player is to roll a pair of six-sided dice and can place down any remaining tiles in the up position as long as their sum is exactly equal to the number shown on the pair of dice. For example, to start the game all nine tiles are up. If I were to roll a 3, then I must put either the tile numbered 3, or the tiles numbered 1 and 2 since those are the only combinations that sum to 3. If I were to then roll another 3, I would be forced to take whichever action I did not choose previously. The goal of the game is to put all the tiles in the down position, or in other words, to shut the box. A player loses if there are no possible actions on a particular roll. For example, if I were to roll my third straight 3, I could not continue since there is no subset of $\{4, 5, 6, 7, 8, 9\}$ that sums to 3.

I have played my fair share of dice games and I'm well familiar with the distribution of the sum of two fair six-sided dice. The most common sum is a 7 with a $\frac{6}{36}$ or $\frac{1}{6}$ chance of occurring. Any other sum has a $\frac{6-t}{36}$ chance of happening where t is the distance of the given sum from 7, e.g. $P(\text{rolling a } 4) = \frac{3}{36}$. Furthermore, from the rules of the game, the odds of us being able to place down a tile go down as the numbers grow larger. There are far more decompositions of the numbers $1 - 12$ that contain smaller numbers than ones that contain a 9. Thus, it stands to reason that a smart way to play this game is to always place down the largest tile possible at every opportunity.

Following this quick thought process, I proceeded to play the game a few times. I came close on a few occasions, but never actually managed to win. The game was not hopeless - on many runs I was only a single dice roll away from winning - it just never seemed to happen for me. That's ok, I thought. I was clearly putting in effort to understand the situation and was happy with how things played out. That was the case, at least until we passed the game around. Everyone else in the family was winning multiple times in the same time span that I had just spent. Even more insulting, some of the strategies employed were simply based on *vibes* rather than attempting to win. Was my strategy not optimal? Or even worse, was it so bad that it was worse than if I

had just chosen actions randomly? Or maybe I was overreacting and it was all just dumb luck.

The root of my issue was that my strategy was by no means rigorous or provably optimal (at least I had not done so). Perhaps there is some sort of parity that I should look to maintain? If my tiles remaining are $\{2, 4, 6, 8\}$ then I lose if my next roll has an odd sum. Maybe there is value in keeping around the tile numbered 1 since it fits in so many decompositions? I initially dismissed these claims, but after a debilitating session, I was ready to hear them out. In response to my poor showing, I set out to properly solve for the best set of actions given any certain scenario. If I lose due to poor luck, that's something I can tolerate. But I would not be comfortable unless I was taking the best actions, in expectation. This document is a survey and case study of Markov Decision Processes and how they can be used to find a desired set actions.

2 Markov Decision Process

2.1 Preliminary Setup

A Markov Decision Process (MDP) is a type of discrete-time stochastic control process. These time control processes are useful mathematical models for making decisions over time. Mathematically, an MDP is a tuple $(\mathcal{S}, \mathcal{A}, P_a, R_a)$ where

- $\mathcal{S}$ is a set of states
- $\mathcal{A}$ is a set of actions, $\mathcal{A}_s \subset \mathcal{A}$ is the set of actions available in state $s \in \mathcal{S}$
- P_a is a probability transition matrix (or function) associated with action a which satisfies the Markov property
- R_a is a reward function for action a.

At each discrete point in time, the decision maker, sometimes called an agent, will observe the state of the environment $s \in \mathcal{S}$. The agent is then to take one of the available actions in state s which we will denote as $a \in \mathcal{A}_s$. The process or environment will then transition to some state $s' \in \mathcal{S}$ according to the probability transition function $P_a(s, s')$. Here, we use the notation

$$P_a(s, s') = P(\text{arriving at state } s' \text{ after taking action a in state s}).$$

Once the environment moves to the next state s', the agent observes a reward $R_a(s, s')$ which represents the value of the action taken. It should also be noted that in order for a tuple to be an MDP, its transitional properties must satisfy the Markov property. In essence, this is simply enforces that the probabilities at a particular state are conditionally independent of how we got to the state. Perhaps one of the reasons that MDPs are so useful is their representative power. Many single player games can be modeled as MDPs and these representations can be used to various effects. Our game falls into this category.

To model Shut The Box as an MDP, we must first properly define all of the elements of our tuple. What are the possible states that we can be in? At first, it may be tempting to say that the set of tiles remaining in the up position can represent our state. However, this alone does not uniquely define all of $\mathcal{A}_s$ since our actions (and consequently states) are a function of our dice roll as well. Instead, let us set our state space to be a tuple itself $\mathcal{S} = (N, T)$ where N denotes a number $2 - 12$ representing our dice roll and $T \subset \{1, \ldots, 9\}$ which represents the number of tiles remaining in the up position.

By defining our states in this way, it is very easy to set the rest of our MDP. Given a specific state (N, T), our action space $\mathcal{A}_s$ is limited to all subsets of T who sum up to N. We can also append a "give up" action in case that given our tiles T and dice roll N, there is no other valid action to take. In other words,

$$\mathcal{A}_s = \{E \mid \sum_{t \in E} = N, E \subset T\} \cup \{\text{give up}\}.$$

The transitional probabilities are easy to compute as well. Given state $s = (N, T)$, the only way that we can reach $s = (N', T')$ is if we were to take action $a = T - T'$ and we were to roll N' on the subsequent roll. In other words, the only way to reach s' from s is if our action in state s is to remove all of the tiles that are in T but not T' and then roll N'. With the introduction of the "give up" action, we also need to properly define how the process changes if the "give up" action is taken. For simplicity, we can say that if "give up" is chosen, then we return the starting state with probability 1. Or more formally,

$$P_a((N, T), (N', T')) = \begin{cases} P(\text{rolling } N'), & a = T - T' \\ P(\text{rolling } N'), & a = \text{give up,T' } = \{1,\ldots,\ 9\} \\ 0, & \text{otherwise} \end{cases}$$

where we previously discussed how to compute the probability of rolling a particular sum on a pair of dice. Also note that these probabilities satisfy the Markov property.

Lastly, we need to define the reward function in order to specify the objective of our games. Many such rewards work, but let us focus on a simple one. The primary objective is to close the box, so let us set

$$R_a((N, T), (N', T')) = \begin{cases} 1 & T' = \emptyset \\ 0 & \text{otherwise} \end{cases},$$

e.g. we receive a reward of 1 for actions that win us the game, and 0 for all other actions.

2.2 Identifying Objectives

Frequently listed alongside the definition of an MDP, we can also define certain functions of importance. Our previously mentioned goal was to find an optimal

way to choose our actions in a given state. In time control processes, a policy π is defined as a function that specifies which action $\pi(s)$ the agent will make when in state s. Our goal is to find π^*, the optimal policy (in terms of the total reward gained).

As I have often found in optimal control, sometimes it is easier to solve certain problems by identifying related, more directly approached problems. Here, we will do so by identifying an auxiliary task: computing the value of each state. Let V be a function of a state s that returns to us how valuable a particular state is. The beauty of the value function is two-fold. One, by computing the value for each state, we can easily obtain the optimal policy via

$$\pi^*(s) := \text{argmax}_a \sum_{s'} P_a(s, s')(R_a(s, s') + \gamma V^*(s')).$$

We can interpret the relation as follows: the best action to take at state s is the action that maximizes both our immediate reward given by $R_a(s, s')$ and our future reward $\gamma V(s')$. In this case, the future reward is *discounted* by discount factor γ so that our model favors obtaining rewards early rather than postponing them, potentially indefinitely. However, since our control process is stochastic (a state + an action does not uniquely define which state the environment will transition to), we will need to maximize the expected value of our immediate and future rewards. In summary, the optimal policy is immediately obtained from the optimal value function.

Two, the optimal value function itself is recursive, i.e,

$$V^*(s) := \max_a \sum_{s'} P_a(s, s')(R_a(s, s') + \gamma V^*(s')).$$

Indeed, the value of a state is equal to the sum of its immediately obtainable reward and its discounted future reward. The power of this recursive equation is that it allows us to iteratively update our value functions from an initial setting. Suppose $V_i(s)$ is our initial guess at what the values of our states are. Then we can repeatedly use the Bellman equation

$$V_{i+1}(s) := \max_a \sum_{s'} P_a(s, s')(R_a(s, s') + \gamma V_i(s'))$$

to improve our value estimate until $V_{i+1} = V_i$. Once we have computed the optimal V^*, we can then use it to obtain our optimal policy, which was the goal.

This iterative process is known as Value Iteration. It has many variants which cleverly exploit the relationship between a value and a policy. Some of these variants are desirable in settings where the number of states is large or when the probability transition function exists but is unknown. Perhaps one day I will encounter a problem worth solving that satisfies some of these conditions, but Value Iteration works incredibly well for our simple problem.

3 Solving The Game

To quote Charles Kettering, "a problem well stated is half solved". With our formulation from the previous section in hand, we can easily go about computing our optimal policy. All that remains to do is manage the storage and design for how we will iterate through various information.

There are a few highly nontrivial computational tasks to do so that we do not develop an incredibly slow solver. One of interest is determining all the valid actions given a particular state. This problem amounts to finding all distinct integer partitions. Simply counting these partitions is a combinatorial problem itself - computing them all is somewhat of an NP-esqe nightmare. Fortunately for us, our state-action space is relatively small and these can be computed in short order. The same goes for managing of the state/value/policy information. Even a poorly thought through solution will likely solve the problem in under a minute.

Interested readers can find my solution on my Github, `https://github.com/trevorsquires/shut_the_box`.

3.1 The Results

After a quick refresher into MDPs, what does our newly required optimal policy tell us about the game? Unfortunately, nothing new. It appears that for any state, the optimal action is to continuously place down the largest possible tile. It seems that my poor performance was simply a matter of bad luck. However, in some partially reassuring news, by some further analysis which looks into how valuable each *action* is, we can see that for any fixed state, the optimal action is not all that much better than one chosen at random. This difference is of somewhat subjective since it depends on the value of the arbitrarily chosen discount factor, initial value setting, and reward function, but the value of the optimal action in many cases is closer to the median action in value than the median action is from the worst action! This makes the game great for families, not so great for highly competitive problem solvers like myself.

At least I will be able to rest assured knowing that I would have done well in expectation.