Monte-Carlo Planning in the Yahtzee Domain

Michael Anderson, Tyler McClung

March 14, 2011

CS533

Prof. Fern

1 Problem Statement

For our environment, we decided to use the classic dice game Yahtsee. This game consists of 13 rounds played in succession. Each round is divided into three parts. At the beginning of each round, five 6-sided dice are rolled. The agent then decides which of the five dice they wish to "keep" (they may keep as little as none of them or as much as all of them), and then rerolls the rest. The agent then repeats this process and rerolls again, and decides again which dice to keep and which to reroll. After the second reroll, the agent must choose one of 13 categories to score the resulting five dice into. The score that the agent earns during the round depends on the nature of the category chosen and the final values of the dice. Once a category has been used at the end of a round, it cannot be used again in subsequent rounds.

The game ends when all of the categories have been used, and the scores from each round are totalled up. In addition, the total score receives a bonus of 35 if the sums of the scores of the 1s, 2s, 3s, 4s, 5s, and 6s categories are ≥ 63. The goal of the game is to get the best possible total score.

The 13 categories can be divided into three types:

1. The "upper-half" categories 1s, 2s, 3s, 4s, 5s, and 6s. In this type, scorers simply multiply the number of dice in the final roll that have a certain value by the value itself. So if at the end of a round an agent ends up with the roll 1-3-3-6-6, it would be scored as $1 * 1 = 1$ if placed into the 1s category, $2 * 3 = 6$ if placed into the 3s category, and $2 * 6 = 12$ if placed into the 6s category.

2. The categories Yahtzee, 4 of a Kind, 3 of a Kind, Large Straight, Small Straight, and Full House. These categories require that a final roll placed into them possess some specific property, and if it does not then the agent receives a 0 score for that round.

   Yahtzee is a 5 of a kind, all of the five dice must have the same value, for a score of 50. 4 of a Kind requires that 4 of the dice have the same value, for a score of the sum of the values of the dice. A 3 of a Kind requires 3 of the dice have the same value, for a score of the sum of the values of the dice. Large Straight requires that the dice have the values 1-2-3-4-5 or 2-3-4-5-6, forming a sequence or "straight" of 5, for a score of 40. Small Straight requires that 4 of the 5 dice have the values 1-2-3-4 or 2-3-4-5 or 3-4-5-6, forming a straight of 4, for a score of 30. Full house requires that 3 of the dice have the same value, and that the other 2 of the dice have the same value, with the value of the triple not necessary the same as the value of the pair, for a score of 25. For example, 1-1-1-2-2 is a valid Full House.

3. The category Chance. This category is sort of a "throwaway" that can be useful when the agent gets unlucky and does not have a particularly

useful final roll at the end of a round. This is scored by summing the value of the dice.

It should be noted that Yahtzee is usually played with multiple agents who take turns playing rounds, where the winner of the game is the agent with the best total score. However, during this project we viewed Yahtzee as a solitaire game in which an agent tries to employ a strategy that will earn them the best possible final score on average. This is sensible because even in multi-agent games the rounds of each agent are independent and do not affect one another.

2 Problem Analysis

To put this game into an AI context we will divide it into places where the agent needs to make a decision. The types of decisions that can be made is our action space, and the set of all such places that could possibly occur in the game is our state space.

Each of the 13 rounds is broken up into 3 parts, all of which require the agent to take an action. The only exception is the end of the final round of the game, in which the agent only has one category left to "choose" from, because all of the other 12 have already been used. This makes a total of $13 * 3 - 1 = 38$ actions taken per game. In this game there are two distinct types of actions:

1. In the first two parts of the round, the agent specifies which dice to keep. Since there are 5 dice total, and each of them could either be kept or not kept as part of the action, there are $2^5 = 32$ different ways to keep dice. Note that in a roll such as 1-2-3-6-6, keeping the first six and throwing away the second six is indistinct from keeping the second six and throwing away the first six. This means that there are actually ≤ 32 *distinct* ways to keep dice, with the exact number depending on the situation.

2. In the last part of the round, the agent specifies which category to place the roll into. Since there are a maximum of 13 categories remaining in the game at any given moment, there are ≤ 13 different actions of this type at the end of each round.

As for states, at any point when prompted for a action an agent should consider the following:

1. The values of the dice in front of them. Obviously the agent should consider the dice it has in order to decide which of them it should keep, or which category is most suitable. Combinatorics gives the possible number of rolls of five 6-sided dice, with order irrelevant, as $C_5^{10} = 252$.

2. Which categories are left to be used. The agent should not try to keep dice in a way that only helps it to get a Large Straight, when the Large

Straight category has already been used, for example. The agent should also not try to place its final roll into the Large Straight category at the end of the round if it has already been used. Since at any point a given category may or may not remain, there are $2^{13} = 8192$ possibilities for this part of the state.

3. The number of rolls left in the round. If the agent wants to try to get a score for a difficult category such as a Yahtzee, it has a better chance with 2 rolls left than with only 1 roll left. Also, the agent can keep dice with 2 or 1 rolls remaining, but must choose a category with 0 rolls remaining. Since there may either be 2, 1, or 0 rolls remaining when the agent is prompted for an action, there are 3 possibilities for this part of the state.

Each of the 3 parts of the state are orthogonal to one another, so in total there are $252 * 8192 * 3 = 6,193,152$ possible states... a very large state space!

It should also be pointed out that this game is obviously a stochastic domain, as the agent does not know which state it will end up in when it takes an action. The agent may decide to reroll certain dice hoping to maximize its performance in some category at the end of a round, but assuming the dice are fair it is not known at the time the agent chooses which dice to reroll whether they will actually do what the agent hopes for.

Finally, the analysis would not be complete without determining the range of possible scores in a Yahtzee game. It is possible to receive a 0 for each of the categories except for chance, for which the worst possible score receivable is 5 for the roll 1-1-1-1-1. The best possible game of Yahtzee contains the maximum scores for each of the categories, along with the upper-half bonus of 35. These are 5(1s), 10(2s), 15(3s), 20(4s), 25(5s), 30(6s), 50(Yahtzee), 30(4 of a Kind), 25(3 of a Kind), 40(Large Straight), 30(Small Straight), 25(Full House), 30(Chance), and 35(upper-half bonus).

$5 + 10 + 15 + 20 + 25 + 30 + 50 + 40 + 30 + 25 + 40 + 30 + 25 + 30 + 35 = 375$, giving a possible score range between 5 and 375.

3 Approach

In order to play around in this environment, we wrote code to simulate it. Our simulator works by simply running the game (rolling the dice, etc.), providing states to an agent which returns actions, applying the actions to modify the gamestate appropriately, and scoring results. Our simulator is based on one that was provided to us written in Java, but ours is written in Python 2.4.

We know that the best possible score is 375, but that is only achievable if an agent gets extremely lucky rolls. Considering this, we wanted ways to

determine what a good average total score in Yahtzee actually is. Toward this end we used the following agents as benchmarks. They do not use any sort of planning or RL:

1. RandomAgent. This agent simply determines which actions it can take from a given state, and selects one at random. Obviously we expected this agent to perform poorly.

2. GreedyAgent. We developed this agent with a hardcoded coded system for both dice-keeping and category-choosing actions. For dice-keeping, we wanted to appoximate the behavior that a human might use, with simple if statements. It first examines the categories left to choose from to narrow down the possible categories to "roll for". It figures out which value is most prominent in the roll, for example in the roll 1-1-1-3-4 clearly 1 is the most prominent value, and so it will tend to keep 1's and reroll the rest. This is useful for the 1s category, as well as possibly the 3 of a kind, 4 of a kind, Full House, and Yahtzee categories. For straights, it looks to see if it is close to a straight by seeing if there is already three or four values in sequence, e.g. it tends to keep 1-2-3. For category-choosing actions it simply evaluates the score it would receive for each individual category, and greedily chooses the highest scoring category.

3. StrategicAgent. We developed this agent with the same hardcoded system for dice-keeping as GreedyAgent, but with a slightly different system for category-choosing. Instead of greedily choosing the best category, it compares the potential maximum value of each category to the score that would actually be received for that category. It chooses the category with the highest (actual score)/(maximum score) ratio, so that if a category happens to give the highest score of the categories to choose from, but that score is low compared with the category's maximum possible score, that category will be saved for later rounds in the hopes of ending up with a score for that category that is closer to its maximum potential.

4. HumanAgent. This agent simply outputs the state to the user and prompts them to choose an action. We wanted to play some Yahtzee ourselves to see how well a human would perform compared to our agents.

5. OptimalAgent. As it turns out, Yahtzee is already a solved game. We found more than one website/paper that used brute force computation for finding the expectation of all of the state/action pairs possible in the game. The average score of these Yahtzee agents are what we would like to get as close as possible to.

Now for our actual approach to solving this problem. We considered formulating the problem as an MDP and then using RL techniques to solve it.

In order to do this, since the state-space is so large, we would have to use feature vectors for each state. The problem is that it is difficult to find a reasonable number of features that describe a state with any degree of accuracy. For example, in order to determine which action to take, it is very important to consider which categories remain, of which as discussed there are 8192 different possibilities. Some of these categories can almost be lumped together, such as 3 of a Kind/4 of a Kind/Yahtzee, or Small Straight/Large Straight, but this does not reduce the number of features to a managable level. Because of the strong possibility of ending up with thousands of features, we decided to abandon RL as infeasible.

Given that the domain is stochastic, we ended up with Monte Carlo Planning as our best bet. We were not sure how to apply this technique for our category-choosing type actions because these actions are isolated from one another, and it would have required a deep and wide expectimax tree that looked too far down into the game to be computationally feasible with our limited resources. We decided to fix category-choosing to the greedy and strategic hard-coded strategies already implemented in GreedyAgent and StrategicAgent. We focused instead on using stochastic planning only for dice-keeping actions.

For dice-keeping, our algorithm was fairly straightforward, we use sparse sampling to build an expectimax tree. If there is 1 roll remaining in a round, then separately examine all of the possible ≤ 32 combinations of dice to keep. For each combination roll the unkept dice N times and then assign the result to a category using the Greedy or Strategic method. Choose the dice-keeping combination with the best average score at the end of the round. If there are 2 rolls remaining in a round, first sample N times for each combination of dice that can be kept, and then for each of the resulting states in which 1 roll remains sample N times again and choose a category using the Greedy or Strategic method. Again choose based on the best average score at the end of the round.

It should be noted here that course we could make an agent use direct probability calculations to tell us what say, the chance of getting a Yahtzee from a given state is, but then we would be farther away from the general planning techniques given in class and closer to a hardcoded strategy.

4 Results

Here is a table of results for our benchmarking agents:

Agent	Avg Score	Min	Max	StDev
Random	40.15	21	68	13.1
Greedy	122.4	56	168	33.9
Strategic	126.2	71	184	30.8
Human	213.45	173	274	26.3
Optimal	245.0	12	375	59.6

And for our PlanningGreedy and PlanningStrategic agents using Monte Carlo Planning:

Agent	Avg Score	Min	Max	StDev
PlanningGreedy	199.0	164	257	25.1
PlanningStrategic	193.85	147	222	17.2

5 Conclusion

One interesting thing to note is that there was not a large difference between using a greedy or strategic method for category-choosing. This is probably because there are some, but not too many, situations in which the categories they choose are actually different.

All in all I would describe our results as a mixed bag. Our application of Monte Carlo Planning clearly beat both RandomAgent and the hardcoded dice-keeping methods in both GreedyAgent and StrategicAgent. Because of the computational complexity of taking N samples of ≤ 32 dice permutations, and then doing it again, for states with 2 rolls remaining, all of the figures (except for OptimalAgent) are based on only 20 trials with N=10. Even with only 20 trials, since the standard deviations ranged between 17.2 and 33.9, a $\approx$200 average is very clearly superior to a $\approx$125 average. Also, as of course expected, the RandomAgent was blown out of the water.

The disappointing part was that our planning agents lost slightly to our attempts to play Yahtzee ourselves (213 average), and were also considerably inferior to OptimalAgent (245 average). While we are confident in the soundness of our general methodology, the computational resources available to us were not quite adequate to meet our needs. If we were able to take a larger number of samples at each state, we believe that we would be able to get better results, and the agent would have a better sense of what the consequences of keeping one set of dice over another would be.