Jeopardy! Wagering Under Uncertainty

Kartik Chandra
Stanford University
CS 238, Fall 2019

Abstract

The popular TV game show *Jeopardy!* gives players occasional opportunities to bet some or all of their score on their ability to correctly respond to trivia questions. This simple game mechanic leads to complex wagering strategies. Most existing work on optimal wagering in *Jeopardy!* studies the problem of optimal wagering in the final round, which can be treated as a single-shot game. In this paper we develop a model for *mid-game* wagering in Jeopardy and evaluate it on real game data.

1 Introduction

Jeopardy! is a popular American TV game show based on answering trivia questions. As part of the game, players may occasionally make "wagers," betting up to 100% of their current score on the correctness of their response to certain clues.

How much should a player wager? There are many factors at play. Players may wish to maximize their cash winnings, but are also interested in maintaining a lead if they are in first place — these two goals lead to a tradeoff between aggressive and defensive wagering. There are also several factors that introduce uncertainty into the decision: players need to be aware of not only their own confidence in providing correct response, but also the relative skill level of the other players. They also need to be aware of how other players might wager in the future.

We would like to have a "wagering advisor" which, given only data about a game so far, advises a player on how much to wager. The wagering advisor should use the data from the game so far to inform its probabilistic model of the game's dynamics, and then suggest a wager that maximizes the player's expected reward, assuming *other* players wager optimally as well (from *their* perspective).

The rest of this paper is structured as follows: below, we give a brief description of the game *Jeopardy!* for those unfamiliar with it. In Section 2, we give a survey of prior work, and discuss our specific problem domain and the simplifications we make. Section 3 explains our proposed algorithm, and Section 4 evaluates that algorithm on real game data.

1.1 Brief Description of the game *Jeopardy!*

Jeopardy! is played by three players. The players are shown a series of "clues," and they "buzz in" on a signaling device to be the first to give the correct response. Each clue is associated with a known dollar value; correct responses increase the player's score by that dollar value, whereas incorrect responses decrease the player's score by that dollar value (and create the opportunity for others to buzz in). There are 30 clues in the first round (Single Jeopardy or SJ) and 30 clues in the second round (Double Jeopardy or DJ); they are placed on a grid among six "categories" and five dollar amounts. DJ clues have higher dollar amounts than SJ clues.

There are two notable exceptions to scoring. First, there are three "Daily Double" (DD) clues, which are hidden randomly among the other 57 clues (one is in SJ, the other two are in DJ). During a DD, the nominal dollar value is irrelevant: instead, the player to give the previous correct response must "wager" any or all of their current score and be posed the clue individually (others cannot buzz in, and the wagerer must provide a response). Note that a player with a score below $1,000 can wager up to $1,000.

Second, there is a Final Jeopardy (FJ) round, with a single clue for which all players individually make a wager and provide a response. You must have a non-negative score before FJ to be invited to join the FJ round (though it is rare for a player to be dismissed for this reason).

The goal is to accumulate the highest score at the end of the match.

2 Prior Work and Problem Domain

Prior work, such as Gilbert and Hatcher [1994], Ferguson and Melolidakis [1997], discusses FJ wagering at length from a game-theoretic perspective, but does not consider the rest of the game. On the other extreme, Tesauro et al. [2014] discusses nearly every aspect of *Jeopardy!* game strategy, including clue selection, buzz/no-buzz decisions, etc. from the perspective of designing IBM's "Watson" agent, which went on to defeat several extremely strong *Jeopardy!* champions in an exhibition tournament.

In this paper we stay focused on wagering strategy, but unlike Gilbert and Hatcher [1994], Ferguson and Melolidakis [1997] we consider DD wagering instead of FJ wagering. Tesauro et al. [2014] limit their analysis of DD wagering strategy to the *last* DD, because of the complexities associated with the way past wagers might influence future wagers. In this paper we look deeper into this complexity by explicitly simulating all three players, and thus being able to compute DD wagers for all three DDs.

Specifically, the problem is as follows: given a transcript of a *Jeopardy!* game up until a DD is selected, output a legal wager that optimizes the expected score going into FJ. A "transcript" is defined as a sequence of clue IDs (category and value) as well as all (correct and incorrect) responses, and wagers in the case of DDs.

Notice that the agent has only as much information as a player would every time it attempts to make a wager.

There are some important simplifications we made in this model:

Most importantly, we ignore the complex reward model that comes into play during FJ. The first-place winner ("champion") is awarded their score in cash (minus taxes); second-place and third-place are clamped to $2,000 and $1,000 respectively. Furthermore, the champion is invited to play again the next day, allowing for more potential winnings. Saunders [2017] calculate that a champion wins $20,000 on average and returns with probability roughly 50% — thus, summing the geometric series, we conclude that becoming the champion is worth a $40,000 bonus on top of the final score. There is one more complication: at some point these figures become dependent on how many games have been played already, owing to the possibility of being invited to the Tournament of Champions, which offers additional winnings. Of course, there are also various real-world complications such as nonlinear taxation on the monetary benefits, and the utility of the pride of being a *Jeopardy!* Champion.

Rather than model these dynamics all at once, we treat FJ as a black-box and instead consider the game *going into* FJ. We consider two regimes separately: in the first, we simply aim for each player to individually try to maximize their winnings going into FJ. This is a good heuristic, but it could conceivably backfire in cases where one might want to (for example) bet slightly less than optimal by our standard, in order to preserve a runaway lead going into FJ (a "runaway" is defined as when the first-place player has more than twice the second-place player's score going into FJ, guaranteeing their win if they wager $0 in FJ). So, we also separately consider a reward model where players strive to have the security of a runaway lead, regardless of the actual score.

We also ignore any information given by categories of the clues. Importantly, this means we don't have the ability to compute a player's confidence within a given category, even though in reality this is an important factor. The reason for this is that player's confidence within categories is difficult to estimate from the noisy signal of at most 5 clues; Tesauro et al. [2014] observe this constraint as well.

We treat DD placement as uniformly random across the 30 clues in each round, whereas in reality DDs are with high probability placed among the second-highest dollar-amount clues, and rarely in the second column. Incorporating a DD placement model is left as future work.

We ignore any effect current scores might have on square-selection, treating it instead as a pre-ordained process — under the hypothesis that for most games, all clues are eventually visited, and if DDs are indeed random, then the order in which they are revealed should not significantly influence the outcome.

Finally, we ignore effects of timing: the game is time-limited, and occasionally not all clues are revealed. In these circumstances, players tend to opt for higher-value clues, and thus lower-value clues are likelier not to be revealed. However, this is a rare circumstance and we do not model its probability (though we do take it into account when *evaluating* our agent on historical game data).

3 Method

We approach this problem using sparse sampling. We build a generative model of possible playthroughs of a game given the transcript so far, and for each possible wager approximate the expected winnings by sampling from this generative model.

More specifically, the generative model expects a transcript that is truncated at a DD, and advances the transcript to the next wagering event (DD or FJ) by drawing from the remaining clues (at random) and simulating response outcomes based on the observed history.

The set of possible response outcomes is the set of 3-tuples $\{\mathsf{Correct}, \mathsf{Wrong}, \mathsf{No\ response}\}^3$. Note that tuples with multiple "Correct" entries must be included to account for the (very rare) case of when a player successfully contests an "Incorrect" ruling, which retroactively awards that player the points without affecting other players. We model the distribution over these outcomes as a Dirichlet distribution whose counts are given by the information in the transcript so far.

The sparse sampler also keeps track of what player it is currently representing. When it samples a new continuation of the transcript, it makes the next wager on behalf of the player simulated to be making that wager. For example, in a situation with two remaining DDs, we might consider what happens if Alice wagers $1,000. The sampler might return a hypothetical transcript where the next DD is found by Bob. We would then try to approximate Bob's optimal wager from *his* perspective at that point, and then backtrack to approximate Alice's utility at the first DD based on that computation.

More formally, the algorithm can be defined as follows:

```
function ComputeWager(transcript T, player P) {
  // OUTPUT: (wager, expected score)
  if transcript is at FJ
    return reward for P

  bestWager, bestAverageScore = null, -infinity
  for w <- sample N legal wagers {
    averageScore = 0
    for T', P' <- sample k extensions of T if P wagers w {
      ignore, expectedScore = ComputeWager(T', P')
      averageScore <- averageScore + expectedScore / k
    }
    if averageScore > bestAverageScore {
      bestWager <- w
      bestAverageScore <- averageScore
    }
  }
  return (bestWager, bestAverageScore)
}
```

We can control the run-time performance of this algorithm by tuning N (number of wagers to sample) and k (number of transcript-extensions to sample). Increasing N gives us finer control over the wagers explored, whereas increasing k decreases the variance in our estimates. Because the number of recursive calls is bounded by the 3 DDs, the run-time of the algorithm is bounded by $(Nk)^3$.

As mentioned above, we will consider two separate reward models. In the "Pre-FJ Score" heuristic, each player individually aims to maximize their score before FJ; the reward is simply their pre-FJ score. In the "Runaway" heuristic, players aim to simultaneously (a) be the first-place player entering FJ, and (b) have more than twice the second-place player's score. The reward model is +1 if this condition is satisfied, -1 if some *other* player has a runaway, and 0 otherwise. Notice that the latter heuristic is in some sense "more difficult" because selecting an optimal wager now depends not only on past performance, but also on potential future wagers made by *other* players (the "Pre-FJ Score" heuristic is agnostic to what other players wager). However, the algorithm above handles both cases.

4 Evaluation

To evaluate our approach, we downloaded and parsed transcripts of 110 non-sequential *Jeopardy!* games from `j-archive.com`, spanning a few decades. We then simulated those games, substituting our agent in place of the players for each DD wagering event using $(N = 5, k = 5)$. In particular, at each DD event we spawn a fresh instance of the algorithm, so all wagers are computed independently (but with knowledge of the full past transcript, which of course includes past synthetic wagers).

For the "Pre-FJ Score" heuristic, because our goal is to maximize scores going into FJ, our metric for success is simply the average delta in pre-FJ scores from the historical wagers provided by `j-archive.org` and the synthetic wagers recommended by our agent. For the "Runaway" heuristic, our metric for success is the percentage of games that entered FJ with a runaway for the first-place player.

Figure 1 and Figure 3 show scatterplots[1] representing historic scores and scores resulting from synthetic wagers for the two strategies. Notice that for the vast majority of players, in the case of the "Pre-FJ Score" heuristic their score is significantly higher using our agent's synthetic wagers than by using the historical human-selected wagers. On the other hand, using the "Runaway" heuristic, their score is lower using our agent's synthetic wagers. This is as expected: the algorithm succeeds in increasing scores overall for the "Pre-FJ Score" heuristic; however, for the "Runaway" heuristic it tends to "play it safe."

We might also examine the wagers themselves. Figure 2 and Figure 4 plot human wagers against corresponding synthetic wagers. The synthetic wagers are usually, but not always, higher than the human wagers for the "Pre-FJ Score" heuristic and lower than the human wagers for the "Runaway" heuristic. This is also as expected: for the "Pre-FJ Score" heuristic it is beneficial to bet aggressively to maximize winnings, whereas for the "Runaway" heuristic it is better to play it safe.

The aggressive DD wagering tactics employed by recent champion James Holzhauer (and pioneered by Alex Jacob) provide a good case study: these champions operate in a regime where they know they are very likely to win, and therefore their goal is to maximize winnings (i.e. the "Pre-FJ Score" heuristic is more valuable because the "Runaway" heuristic is pretty much guaranteed). Hence, we see much more aggressive wagering from such players than from typical players.

Finally, Table 1 categorizes our 110 sample games into four categories, based on whether or not they ended in a runaway for the lead player using historical and synthetic wagers. Notice that the vast majority of games reached the same runaway state using historical and synthetic wagers, suggesting that our wagering is not too far from what humans manage. However, there were 12 games where the synthetic wagering strategies could force a runaway where humans could not, and only 3 games where the synthetic wagering strategies failed to reach a runaway where humans could. This suggests that it may be easier for an optimally-playing lead player to "force" a runaway than for an optimally-playing second-place player to "fight" a runaway.

In terms of efficiency, our algorithm averages 7.1 seconds to evaluate a full game, making it practically real-time (contestants on the game show are typically allowed around 10 seconds to calculate and declare their wager for each DD).

[1] Figures appear in an appendix. The orange line represents the identity function $y = x$ in all scatterplots; the x-axis and y-axis scales are not equalized.

5 Conclusion

In this paper we formalized the problem of mid-game DD wagering for *Jeopardy!*. We proposed an agent that uses sparse sampling to suggest wagers, under two different reward models representing different goal heuristics. Finally, we evaluated our agent on historical game data using the two heuristics.

The simplifications described in Section 2 lend themselves quite naturally to future work. For example, modeling DD placement, square-selection, and clue-category information into the generative model may yield a more robust algorithm. Finally, combining our DD agent with known FJ agents would yield an end-to-end wagering strategy, the evaluation of which is left to future work.

References

T. S. Ferguson and C. Melolidakis. Last round betting. *Journal of Applied Probability*, 34(4):974–987, 1997. ISSN 00219002. URL `http://www.jstor.org/stable/3215011`.

G. T. Gilbert and R. L. Hatcher. Wagering in final jeopardy! *Mathematics Magazine*, 67(4):268–277, 1994. ISSN 0025570X, 19300980. URL `http://www.jstor.org/stable/2690846`.

A. Saunders. What are the average winnings of a contestant on jeopardy? `https://thejeopardyfan.com/2017/05/average-winnings.html`, 2017.

G. Tesauro, D. Gondek, J. Lenchner, J. Fan, and J. M. Prager. Analysis of watson's strategies for playing jeopardy! *CoRR*, abs/1402.0571, 2014. URL `http://arxiv.org/abs/1402.0571`.

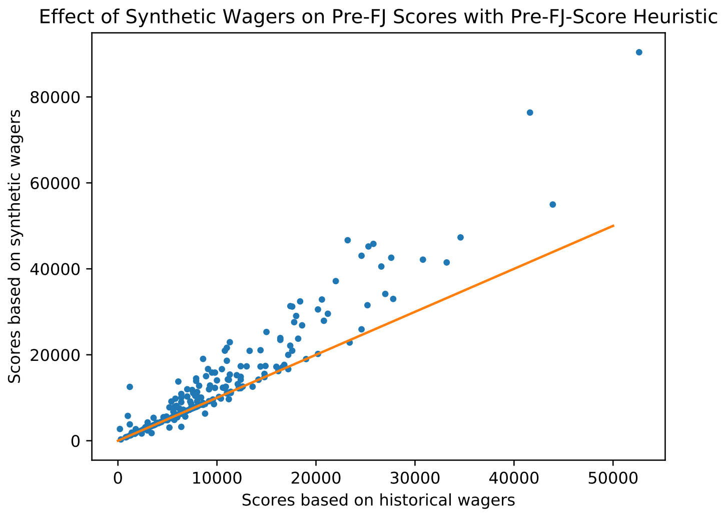

Figure 1: The synthetic wagers provided by our agent for the “Pre-FJ Score” heuristic typically lead to better pre-FJ scores. The orange line represents identity; scatter points above the line therefore represent an improvement.

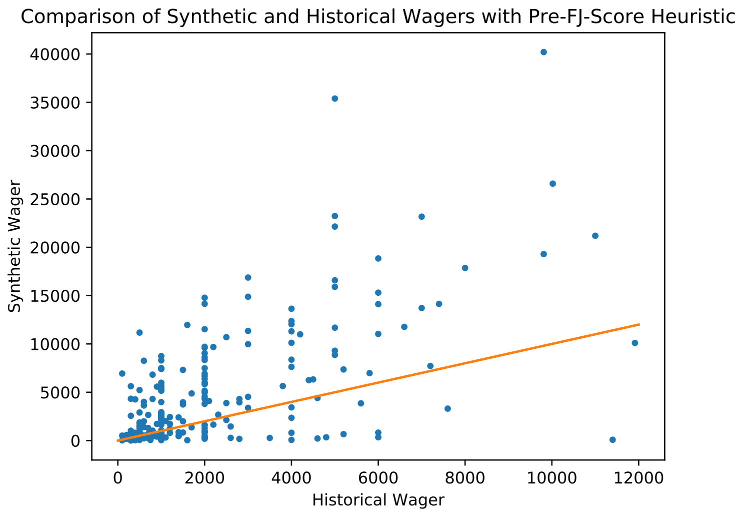

Figure 2: The synthetic wagers provided by our agent for the “Pre-FJ Score” vary from lower than to higher than the human wagers, though there is a definite preference towards higher wagers, suggesting more aggressive bets.

	Historical runaway	Historical non-runaway
Synthetic runaway	32	12
Synthetic non-runaway	3	63

Table 1: This table breaks down the games in terms of whether they ended up in runaways or not based on historical and synthetic wagers. Synthetic wagers are able to force runaways more frequently than historical wagers; there are only 3 cases where synthetic wagers are unable to force a runaway that historical wagers were able to force.

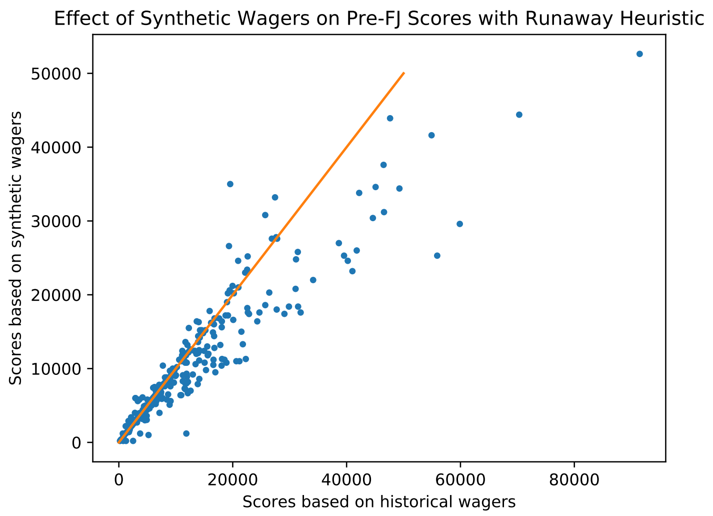

Figure 3: The synthetic wagers provided by our agent for the “Runaway” heuristic lead to slightly lower pre-FJ scores.

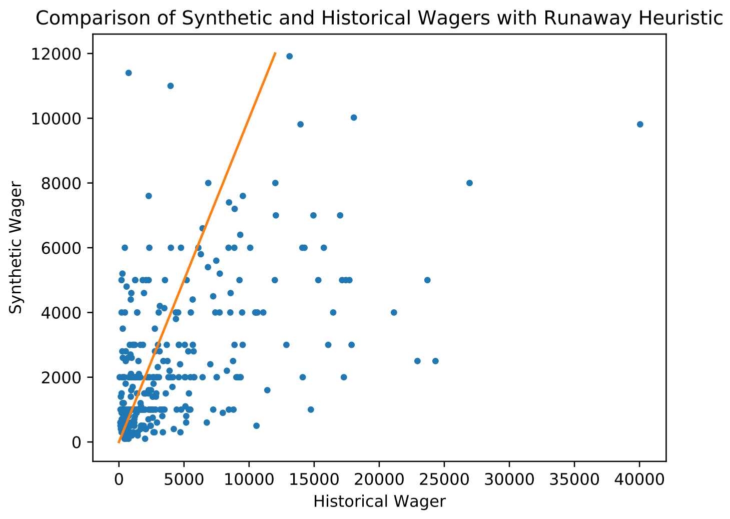

Figure 4: The synthetic wagers provided by our agent for the “Runaway” heuristic vary from lower than to higher than the human wagers, though there is a definite preference towards lower wagers, suggesting safer bets.