Making the best bet in Liar’s Dice

Danny North, Huy Pham, Bryce Perkins  
CS 478, Winter 2016

Department of Computer Science

Brigham Young University

Abstract

The game Liar’s Dice is one of wits, deception, and educated guessing. Although the game is simple to learn, there is intense strategy and clear patterns that can be used to the advantage of each player.

We present a collection of data that was scraped from an online implementation of Liar’s Dice using a scraping tool called Mozenda. We then discuss the features selected and their importance.

Further, we present results found when the collected dataset is processed using multiple implementations of machine learning algorithms to see if it is feasible to make a good bet in Liar’s Dice, and determine which machine learning approaches are better than the rest for this problem.

Our results concluded that back propagation produced models with the highest average accuracy, giving us an 84% success rate for passing a lie as truth.

1 Introduction

Models of competition between two decision makers, where one decision maker must occasionally lie and the other must attempt to detect the lie, were described in a nonsequential setting in T. S. Ferguson (1970) [Ferguson et al., 1991].

The game “Liar’s Dice” consists of two or more players, each starting with five dice and a cup to conceal them. Each player rolls their dice and hides them under their cup so that they are the only one to see the dice values. Players make “bids” guessing how many dice of a certain value is on the table.

Player continue to the left making bids. A bid must increase either the quantity and/or value of the dice predicted. Bidding continues until a player “challenges” a bid made by the player that preceded them. The dice are then revealed. If there are at least the number of dice predicted by the bidder, then the bidder wins. If the number of dice is less than what the bidder guessed, then the challenger wins. The loser removes one of his dice from the game.

All players then re-roll their dice, and the loser begins the next round of bidding. Players lose when they run out of dice, and those players with less dice have less influence in future rounds.

1.1 Motivation

Because this is a game of deception, we thought it would be interesting to determine which bids had the highest probability of deception, in order to ‘learn’ how to lie better than your opponents. It would be interesting to notice any patterns that come from different types of bids and reduce this problem into a subset of features that give important insight into these bidding patterns.

2 Methods and Data

In this section we discuss the data that was collected for our feature set, and the methods we used for predicting whether or not a certain bid is a good ‘lie’.

**2.1 Data Sources**

The dataset for rounds of Liar’s Dice needed to come from the game actually being played. While it is possible to physically play the game and record it, it was not feasible for the amount of data we needed. For this reason, we decided to use a data scraping tool called Mozenda.

This tool provides the ability to input commands, designate wait periods, and automate information retrieval. With this tool in hand and an online version of Liar’s Dice, six different agents, or automated scrapers, were created (one for each side of a die).

Each agent continually plays the game in the following fashion. It will choose one number to bid (the assigned side of the die) and one quantity higher than the previous bid. The current table information is then collected (see Table 1). The bid is then placed and if the agent starts the next bid phase with the same amount of dice, we conclude that the previous bid was accepted. The agent continues play until

the agent loses or wins the match. The agent then enters a new match and repeats its operations.

This method was chosen for its efficiency and time saving attributes. No existing datasets were found dealing with Liar’s Dice and manual data collection became tediously long. Future considerations include long term data collection including human opponents.

**2.2 Data Instances**

After scraping the game website, we collected about 10,000 rows of usable data. After this initial scrape there was a sanitization of the data that was essential in order to make our data fit into the two learning models we had chosen. Features such as ids and websites needed to be removed. Other features, such as round number, were discussed as whether or not to be important to our feature subset. The final feature subset can be seen at Table 1.

There was also some sanitization involved regarding removal of unwanted characters in order to keep the dataset consistently continuous numbers across the board.

We also allowed unknown values to be in our dataset. In the case where a player had less than five dice to bid on, the remaining dice were given unknown values. Also, in some cases, the outputs themselves were unknown. In this case, we will let the learning models handle what to do with these unknown values.

2.3 Learning Models

Because of the large amounts of data we have access to, thanks to the data-scraping tool, we decided that a multi-layer perceptron with Backpropagation would be a good initial model to choose.

The backpropagation model will be able to take the large amount of data as input and carefully fine-tune its output, hidden and input weights to the two output nodes that we are trying to train. The model we chose consists of one hidden layer with one hidden node per input node (See figure 1). This was provided to us by the Weka data mining and open source machine learning tool [Hall et al., 2009].

Backpropagation is also a good model for this problem because this problem could easily be converted from a classification problem to a regression problem. In this case, the output would not be a binary value, but a value determining the strength of a bid as a lie. This might result in better bidding because one lie might have a regression output value of 0.84 while a different one, which would have also passed a binary test, might have a value of 0.72.

In this case, the first value would be considered a better lie and preferred. Using backpropagation as a regression model would be a good application for future research.

The second model we decided to use was a random forest. While being an interesting model to test our dataset on, it also provided a way to test a broad range of feature combinations. Because the random forest model is an ensemble approach, we hoped that a group of “weak learners” could come together to form a “strong learner” and fit our dataset better than a backpropagation model might.

Also, because we have a lot of unknown data, we hope that a random forest might deal with this better than a backpropagation model might. However, because we include a lot of unknown values in our dataset, we fear that a random forest model might over fit. In this case, the backpropagation model will be the best bet for a reduced error rate.

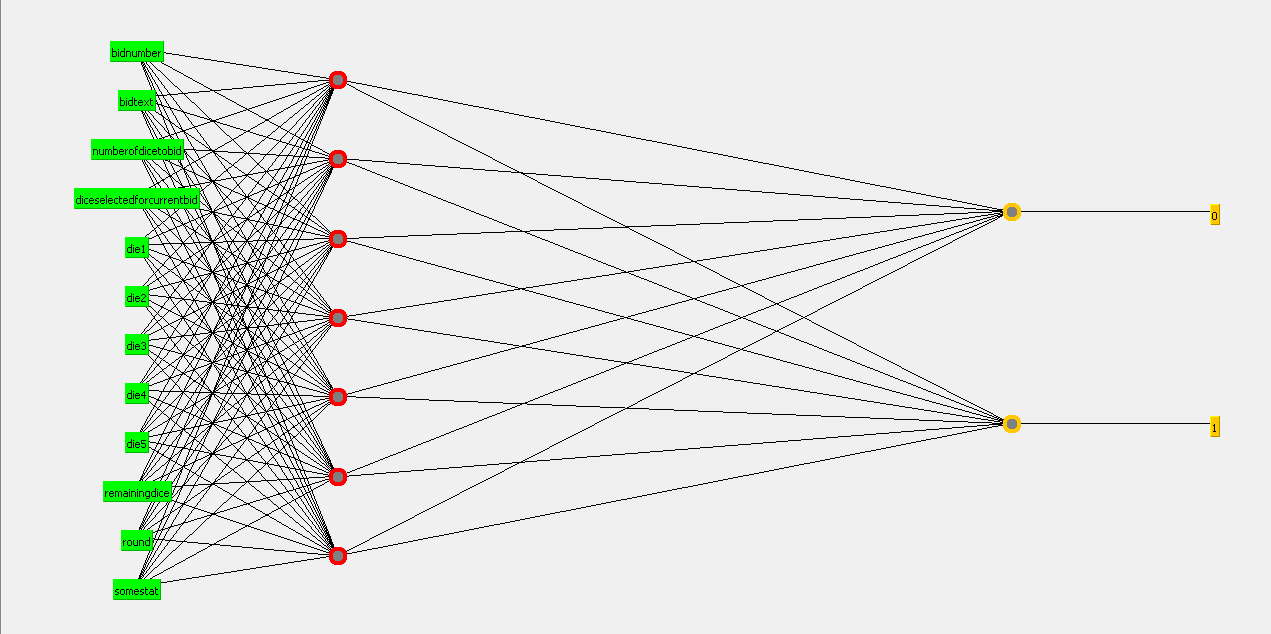


Figure 1: The multi-layered perceptron model. Provided by Weka tools.

|  |  |  |
| --- | --- | --- |
| Attribute | Description | Example |
| previousBidNumberOpponent | Bid number of the previous opponent | 5 |
| previousBidAmountOpponent | Amount of the bid number of the previous opponent | 3 |
| numberOfDiceToBid | The number of dice the current player is going to bet | 4 |
| dieSelectedForBid | The number on the die the current player is going to bet | 5 |
| die1 | The number appearing on the die | 6 |
| die2 | The number appearing on the die | 6 |
| die3 | The number appearing on the die | 2 |
| die4 | The number appearing on the die | 6 |
| die5 | The number appearing on the die | 1 |
| remainingDice | The amount of dice remaining on the playing table | 20 |
| round | The number of the round | 1 |
| statistic | The likelihood that the current prediction is on the table | 0.08 |
| passOrFail | Whether the bid was accepted or caught | 0 |

Table 1: Subset of features from the Mozenda scraping tool that were useful. PassOrFail is the output and says whether or not the bid was accepted. All attributes are continuous, but values can be unknown.

3 Initial Results

Here go the results that Bryce got from his original attempts at using the backpropagation model.

4 Data and Feature Improvements

Here goes what we did to improve the dataset. I think talking about how we sanitized the dataset even more, and just make this a short section.

5 Final Results

Here goes the information that Huy gathered with all the nice charts and data. This should take up a good portion of our report since we tried two different models.

6 Conclusions

While both learning models seemed to give fairly similar results, the backpropagation model performed the best for our dataset. With a final accuracy of about 85%,

References

[Ferguson et al., 1991] Christopher P. Ferguson, Thomas S. Ferguson. *Models for the Game of Liar’s Dice* (PDF). University of California at Los Angeles

[Hall et al., 2009] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.