OPPONENT MODELING IN POKER USING MACHINE LEARNING TECHNIQUES

Patrick McNally & Zafar Rafii

Northwestern University Evanston, IL, 60201, USA
PatrickMcNally2013@u.northwestern.edu
ZafarRafii2011@u.northwestern.edu

ABSTRACT

This paper details the results of plying several machine learning techniques to the task of predicting an opponent's next action in the poker game *Texas Hold'em*. Hold'em is a game of imperfect information, deception and chance played between multiple competing agents. These complexities make it a rich game upon which to use machine learning models because the emergent statistical patterns are often subtle and difficult for a person to recognize. To this end, we explore a variety of features for predicting action, showing which appear to be the strongest. Furthermore, we show that as more rounds of betting are observed for a particular hand, it becomes easier to predict action. Finally, we propose a feature to differentiate players in terms of their style of play that is easily calculated for new opponents in real time.

1. INTRODUCTION

In recent years, much progress has been made on computer gameplay in games of complete information such as chess, where computers have succeeded in surpassing even the top chess players. On the other hand, games of incomplete information such as poker have been less studied and artificial poker bots are still below the level of that of professional players. Unlike board games such as chess or checkers where the entire game state is completely visible for all the participants, poker is a complex stochastic game involving a number of attributes such as imperfect knowledge (hidden hands), multiple competing agents (two players or more), risk management (betting) and deception (bluffing), which actually make it a difficult but interesting problem for developing an artificial intelligent agent.

One of the major pieces necessary for developing a competitive poker agent is an accurate opponent model. This is because, in poker, simply "playing the cards" is often not enough, and many advanced strategies and expected value computations hinge on how an opponent is expected to play. To this end, an accurate model of an opponent (a good idea of what they are likely to do in a given situation) is incredibly valuable information. A good opponent model informs your strategy and helps you to gain a greater edge to press over and over again. Machine learning techniques are particularly suited to problems of this nature. Given a large enough relevant data set, they can suss out subtle statistical patterns in behavior that human players would be unable to recognize. We propose to bring two machine learning models to bear on the simple problem of opponent modeling: *Artificial Neural Networks* and *Decision Trees*. We hypothesize these techniques can provide accurate predictions of opponent actions.

The paper is organized as follows. Section 2 introduces the *Texas Hold'em* game and the database used for the opponent modeling. The two machine learning models are presented in section 3. Experimental results are then given in section 4. Finally, conclusion and perspectives are drawn in 5.

2. THE GAME

2.1. The Texas Hold'em

Fig. 1. A *Texas Hold'em* game with a player and his 2 hidden private cards and the 5 community cards face up on the board.

Texas Hold'em is perhaps the most popular poker game in the casinos and poker card rooms across North America and Europe, as well as online, especially since the 2000s due to massive exposure on television and Internet. Hold'em is a multiplayer community card game where each player may use any combination of the five community cards and the player's own two hole cards to make a poker hand, in contrast to poker variants like stud or draw where each player holds a separate individual hand. Because each player starts with only two cards and the remaining cards are shared, it is an excellent game for strategic analysis (including mathematical analysis), and also interesting in the perspective of machine learning techniques.

The rules of *Texas Hold'em* are summarized as follows. Play begins with each player being dealt two cards face down. After a "pre-*flop*" betting round, the dealer deals a *flop*, three face-up community cards. The *flop* is followed by a second betting round. Then, a single community card called the *turn* or *fourth street* is dealt, followed by a third betting round. A final single community card called the *river* or *fifth street* is then dealt, followed by a fourth betting round and the showdown, if necessary.

2.2. The Database

For several years, before the advent of real-money online poker servers, there have been IRC channels with poker dealing bots and "fake money" stakes. From 1995 to 2001, 10 million complete hands of poker were collected using an observing bot called *Observer* that sat in on IRC poker channels and logged the details of every game. This hand database has been made publicly available to further the development of Poker Artificial Intelligence research [4].

Although these hands do not represent "real money" play nor are there any guarantees of quality of play, their ease of acquisition and volume make them a very good place to start a machine learning experiment.

Once aquired, this database had to be refined into the input-output vector pairs used for training. The four actions we wished to predict were fold, check, call and bet. The input features we decided on using were the following:

- *hand value* - community cards available to everyone are rated from 0 to 1.0 with a straight flush being 1.0 and a high card being 0.0.
- *number of suited cards* - the highest number of suited cards on the table.
- *number of consecutive cards* - the highest number of cards whose values follow in a row, i.e. the number of straight cards.
- *the player's previous action* - the most recent action taken by the player in the round.
- *the player's position at the table* - position determines betting order. A higher position is often considered stronger as you get to see other player's actions before deciding on your own.
- *amount required to call* - the number of small bets, if any, required to match in order to stay in the round.
- *the player's total number of bets made in the round* - this feature counts bets made only when no amount is required to stay in the hand.
- *the player's total number of raises made in the round* - this feature counts bets made when a smaller amount is required to stay in the hand.
- *the player's total number of calls made in the round* - this feature counts how many times the player has called in the round.
- *player aggressiveness* - this feature is pre-computed for each player in the database. It is simply the ratio of aggressive actions (bet and raise) to total actions over the course of the player's history. This feature can also be computed fairly easily for new opponents on the fly by keeping tallies of their aggressive and total actions.
- *the betting round in the hand* - whether the betting is pre-flop, on the flop, on the turn or on the river.

Most of these features are simply pieces of a given table state. One problem for any feature set based solely on the state of the table is that player's styles are not accounted for. Given the exact same situation, two players may consistently make different choices. How to best characterize the play style of a given player is a tremendous problem. There are notions of "tight" and "loose" players in poker that correspond to overly risky or overly conservative play respectively. It is often very difficult to arbitrarily recognize tight or loose actions much as it is very difficult, even for very experienced players, to estimate the skill of a new opponent without a thorough hand history.

We propose a very simple metric to differentiate players from one another: the ratio of raises and bets to total actions. We call this feature *player aggressiveness* above. This ratio is easy to compute and tends to differ quite a bit from player to player. This value ranges from less than a percentage point for some players to nearly 50%, with most players sitting at around 15% of their actions considered "aggressive". We believe this feature will help greatly in differentiating one player's behavior from another.

3. THE BASELINE

3.1. Decision Trees

Due to their short training time we chose to use a basic pruned decision tree to establish a performance baseline for our models. These decision trees could be trained in a matter of seconds on different sets of our features, providing insight into the value of each.

3.2. Artificial Neural Networks

Given that certain key features in our data were continuous in nature, we expected drastically better results from Artificial Neural Networks when compared to decision trees. Though there were improvements, the gains were not nearly as marked as we would expected. Furthermore Neural Networks take significantly longer to train. Due to this and time constraints, the parameters of our experiment were not as thorough as we would have liked. Specifically, we would have liked to have trained many more networks with varying numbers of hidden nodes in order to determine the proper number for a network structure.

4. RESULTS

Using decision trees we systematically examined the value of each of our features. The data is such that around 36% of all actions were checks (i.e. the player elected to stake no money in the pot). The other three actions occurred in similar frequency to one another, but much less frequently than checks. So, a naive predicter that always guessed check would be correct about 36% of the time. One interesting phenomenon that arises, though, is that one feature, amount required to call, can trivially remove checking as a possible guess because if a bet is required to stay in the hand, checking is no longer a possibility by the rules of the game. In other words, we can use the knowledge that checking is not possible when a player is required to match a bet to help our predictions. We can guess check whenever the amount required to call is 0.0, then we can guess the other most common action when that amount is greater than 0.0. In fact, 'amount required to call' alone is enough to get us to 57% accuracy with a decision tree.

Unfortunately, none of the other features provide such a dramatic gain in accuracy by themselves. In fact only aggressiveness and previous action provide any immediate gain beyond the initial 57% offered by amount required to call at all (61.0% and 62.8% respectively). This suggests that the relationships between the features of the game state are complex and dependent on one another. In fact, when all the features are combined to train a decision tree, the accuracy jumps nearly ten point to 66.7%.

Interestingly, without player aggressiveness, a decision tree only achieves 63.6% accuracy. Having an idea of how often a player makes an aggressive action appears to provide a great deal of information in terms of player behavior.

Another interesting thing to note is that roughly 35% of all actions in the database are coming from the first round of betting (out of four), while only around 13% come from the last round. This is simply a result of players dropping from play as the hand progresses, but it means that a large chunk of the data consists of early-round behavior, which can be very different from late-round behavior. Betting and checking, for example, are the most common actions in the first round by a large margin, while folding and calling become more prominent in subsequent rounds.

This should suggest that the round of betting can signify a lot of information that is pertinent to player behavior. In fact, we found that it is much easier to predict a player's action in a later round than an early round. The following figure shows both a decision tree's and an artificial neural network's accuracy when trained on samples of a specific round only. You can clearly see the trend of increased accuracy in the later two rounds.

round	DT accuracy	ANN accuracy
Round 0	63.0 %	55.8 %
Round 1	65.7 %	65.19 %
Round 2	70.1 %	70.65 %
Round 3	68.8 %	69.0 %

Accuracy of decision tree and artificial neural networks trained on a specific round.

You can also clearly see the differences between neural networks and decision trees. Decision trees actually outperfom neural networks substantially for the first round and fall ever so slightly behind in the later rounds. It should be noted that our networks were trained for a very brief number of epochs and better performance is likely possible.

Finally, we trained an artificial neural network on the features with 32 hidden nodes for 5,000 epochs. Unfortunately training neural networks takes a great deal of time. A proper exploration with neural networks would have systematically performed every experiment with many networks, each consisting of different numbers of hidden nodes to establish the correct network structure. Due to time limitations this was not feasible. Furthermore, we discovered that when we did train neural networks for a short number of epochs, the results were often better than decision trees only by fractions of percents. Nevertheless, poker is all about pressing tiny advantages over the long term so more experimentation should be done to exhaustively determine the best parameters for network structure. Ultimately our network achieved levels of accuracy closely comparable to that of decision trees.

5. CONCLUSION

This paper presented our feature representation for the task of opponent modeling in the popular game of Texas Hold'em. We presented accuracy figures for models trained on our data and discussed which features appeared to provide the most gains. We proposed and proceeded to show that the ratio of aggressive actions to total actions for a player over the course of their game history, a feature that can be easily computed on the fly for a new opponent, is a very valuable feature in predicting action. Furthermore, we have shown that action in later rounds of betting is easier to predict than in earlier rounds of betting. In the future we would like to see more work in terms of expanding the number of features considered and establishing the correct parameters for a neural network trained for this task, as our preliminary models show promise.

6. REFERENCES

[1] Darse Billings, Denis Papp, Jonathan Schaeffer, and Duane Szafron. Opponent modeling in poker, 1998.

[2] Aaron Davidson. Using artifical neural networks to model opponents in texas hold'em, 1999.

[3] Mark Deckert and Karen Glocer. Opponent modeling in poker, 2007.

[4] Michael Maurer. Irc poker database. `http://games.cs.ualberta.ca/poker/IRC/`.

[5] Wikipedia. Texas hold 'em. `http://en.wikipedia.org/wiki/Texas_hold_'em`.

[6] Ian H. Witten and Eibe Frank. Data mining: Practical machine learning tools and techniques (weka), 2005.