Opponent Modelling in No-Limit Texas Hold'em

Machine Learning Engineer Nanodegree Capstone Project

Nash Taylor, September 2016

Definition

Project Overview

The problem of creating an artificially-intelligent poker-playing agent is well studied. In fact, some variants of the game such as Limit Hold'em are already considered solved[1]. However, in more complex variants such as No-Limit Hold'em, there are many factors that limit the effectiveness of such game theoretic models as have been proposed thus far. One of the leading contributors to the difficulty of the game of poker from an AI perspective is its imperfect information; because the hole cards of one's opponents are not known, predicting their response to different possible actions can prove difficult for even the most experienced of human players. This is the specific sub-problem I will attempt to solve.

Predicting a player's action in a given situation is a perfect example of supervised learning. Given features of the state of the game, the player's tendencies, the player's view of his opponents, and the characteristics of the community cards, a supervised model should be able to guess what the player will do next. Note that these features leave out one critical component, which was mentioned earlier: the player's hole cards. However, as we will see, the combination of a well-engineered feature set and a complex function approximator like a deep neural network can make up for this imperfect information state.

All supervised models constructed as part of this project utilized data from HandHQ.com[2].

Problem Statement

The goal of this project is to produce a supervised learning model using Deep Neural Networks trained in TensorFlow to take in features of a poker game and predict the action of whichever player is to act next. The decision of whether to make this a regression or classification problem was tricky, and I have settled on a somewhat 'middling' approach: I will do multiclass classification for the actions, but for bets and raises (which have continuous amounts), I will bin the values. This results in the following set of labels:

[1] http://ai.cs.unibas.ch/_files/teaching/fs15/ki/material/ki02-poker.pdf

[2] http://web.archive.org/web/20110205042259/http://www.outflopped.com/questions/286/obfuscated-datamined-hand-histories

Fold, Check, Call, Bet-min, Bet-Q Bet-Half, Bet-3Q, Bet-Pot, Bet-3Half, Bet-2, Raise-min, Raise-Q Raise-Half, Raise-3Q, Raise-Pot, Raise-3Half, Raise-2+

The amounts associated with bets and raises are relative to the size of the pot. "min" is the minimum legal bet/raise; "Q" is one quarter of the size of the pot; "Half" is half of the size of the pot; "3Q" is three quarters of the size of the pot; "Pot" is a pot-sized bet; "3Half" is 1.5 times the size of the pot; "2+" is 2 or more times the size of the pot, which are all grouped together. Any sizes in between are mapped to their lower bound in the list.

Because the rules of the game allow for certain actions in certain situations, and because the state of the board is different after each "street" (dealing of community cards), I have decided to break up the model into 7 different models, one applied to each of the following situations:

- Pre-flop, facing a bet
- Post-flop, not facing a bet
- Post-flop, facing a bet
- Post-turn, not facing a bet
- Post-turn, facing a bet
- Post-river, not facing a bet
- Post-river, facing a bet

The reason there is no model for "Pre-flop, not facing a bet" is because there are 'blinds', which are mandatory bets that begin every game; therefore, there is only one relatively rare situation in which a player would not be facing a bet before the flop, which is in the Big Blind if no bets are made leading up to that player. This is not an interesting prediction problem (they typically check), and there is not enough data to learn this over.

The result is 7 models, each taking a slightly different subset of the full feature set (see Data Preprocessing) and each predicting an action from a subset of the actions described above. In "facing bet" situations, the actions are: fold, call, raise[amount]. In "not facing bet" situations, the actions are: check, bet[amount].

The overall procedure for learning these models is as follows:

1. Parse the raw text of game logs into a feature set providing information on, for each action taken:
    1. the play style of the player (e.g. their preflop raise percentage)
    2. the player's opponents at the table (e.g. the average stack size)
    3. the community cards (e.g. the number of pairs on the board)
    4. the state of the game (e.g. the number of players remaining)
2. Split the feature set into 7 subsets, separated by the Round and FacingBet fields. For each resulting dataset, take only the corresponding subset of columns (see Data Preprocessing for full lists)
3. For each dataset, train a Deep Neural Network to classify actions, assigning more weight to the correct prediction of fold/check/call than to the exact

amounts of the bets and raises (see Metrics for a discussion of how to weight these accordingly)
4. Evaluate and tune each model using a validation set
5. Obtain an overall score over all models to test against the benchmark

Metrics

In this section, you will need to clearly define the metrics or calculations you will use to measure performance of a model or result in your project. These calculations and metrics should be justified based on the characteristics of the problem and problem domain. Questions to ask yourself when writing this section:

Are the metrics you've chosen to measure the performance of your models clearly discussed and defined?

Have you provided reasonable justification for the metrics chosen based on the problem and solution?

Analysis

Data Exploration

The original data collected for this project was a corpus of text files containing logs of online games from 5 different online poker sites. After parsing relevant information from this text, 3 tables were created and a relational database was formed. From these, a large feature set was constructed.

Boards

Field	Data Type	Description
GameNum	String	Primary key; identifies which game the board is associated with
LenBoard	Int	"Number of cards on the board (represents round of the game; e.g. Flop)"
Board1	Int	"Integer representation of one of the 52 cards in the deck; e.g. 2c = 1"
Board2	Int	"Integer representation of one of the 52 cards in the deck; e.g. 2c = 1"
Board3	Int	"Integer representation of one of the 52 cards in the deck; e.g. 2c = 1"
Board4	Int	"Integer representation of one of the 52 cards in the deck; e.g. 2c = 1"

Field	Data Type	Description
Board5	Int	“Integer representation of one of the 52 cards in the deck; e.g. 2c = 1”

Actions

Field	Data Type	Description
GameNum	String	Primary key; identifies which game the board is associated with
Player	String	Obfuscated name of the player
Action	String	Action without amount
SeatNum	Int	Seat number starting from the top right of the table
RelSeatNum	Int	Seat number starting from the dealer button
Round	String	Round of the game; e.g. Pre-flop
RoundActionNum	Int	“Numbered actions; reset at the start of each new round (e.g. Flop)”
StartStack	Float	Amount of chips for Player at the start of the game
CurrentStack	Float	Amount of chips for Player at current moment (before action)
Amount	Float	Amount of chips associated with action
AllIn	Boolean	Whether the action has put the player all-in
CurrentBet	Float	The amount of the bet that Player must respond to
CurrentPot	Float	The amount of chips currently at stake
InvestedThisRound	Float	The amount of chips Player has invested thus far in the round
NumPlayersLeft	Int	The number of players remaining in the hand
Winnings	Float	The amount that Player received at the end of the hand
HoleCard1	Int	Integer representation of Player’s first hole card
HoleCard2	Int	Integer representation of Player’s second hole card
SeatRelDealer	Int	Player’s seat number relative to the dealer button
isFold	Boolean	Dummy representation of Action
isCheck	Boolean	Dummy representation of Action
isCall	Boolean	Dummy representation of Action
isBet	Boolean	Dummy representation of Action
isRaise	Boolean	Dummy representation of Action

Games

Field	Data Type	Description
GameNum	String	Primary key; identifies which game the board is associated with
Source	String	The online poker site from which the game was scraped
Date	DateObj	The date the game was played
Time	DateObj	The time the game was played
SmallBlind	Float	The size of the small blind for that game
BigBlind	Float	The size of the big blind for that game (should be 2*SmallBlind)
TableName	String	Obfuscated name of the table at which the game was played
Dealer	Int	Number representing seat number of the dealer button
NumPlayers	Int	Number of players active at the beginning of the hand

Features

From these 3 tables, roughly 120 features were produced. To save space, I won't list them here, but they can be found in the data sample. These features can approximately be broken up into 4 categories: - Features of the play style of Player, e.g. Preflop Raise % - Features of the player's opponents, e.g. Average Table Stack - Features of the community cards, e.g. Number Of Pairs - Features of the state of the game, e.g. Is Last To Act The majority of these features are Boolean, and the breakdown of datatypes is as follows:

Data Type	Count
Categorical	1
Boolean	23
Numeric	90

The data being learned over is only a subset of the full data available, due to my own computational limits. This is discussed further in Data Preprocessing. The final shape of each dataset is:

Filename	NumRows	NumCols
Flop-False	2445083	87
Flop-True	1354231	93

Filename	NumRows	NumCols
Preflop-False	77289	78
Preflop-True	14240503	84
River-False	823553	108
River-True	348928	114
Turn-False	1318557	97
Turn-True	626449	103

There is a clear decay in the size of the data as we approach later rounds, which makes logical sense; fewer hands get all the way to the river than start at all. The Preflop section is by far the largest and should theoretically be the most effectively trained model.

The breakdown of labels for each dataset is:

Table	bet	call	check	fold	raise
Preflop-True	0.0	0.133	0.027	0.672	0.168
Flop-False	0.379	0.0	0.621	0.0	0.0
Flop-True	0.0	0.335	0.0	0.546	0.12
Turn-False	0.367	0.0	0.633	0.0	0.0
Turn-True	0.0	0.405	0.0	0.498	0.097
River-False	0.366	0.0	0.634	0.001	0.0
River-True	0.0	0.362	0.0	0.555	0.083

Exploratory Visualization

Algorithms and Techniques

Benchmark

Methodology

Data Pre-Processing

Implementation

Refinement

Results

Model Evaluation and Validation

Justification

Conclusion

Free-Form Visualization

Reflection

Improvement

Applications