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 the following:

   1. a Deep Neural Network to predict the action selected by the player (e.g. bet), as a classification
   2. a Deep Neural Network to predict the amount of the bet or raise, as a regression

4. Evaluate and tune each model using a validation set
5. Obtain an overall score over all models to test against the benchmark

The result is a set of 14 networks: one for actions, and one for amounts, for each of the 7 situations.

Metrics

The variety in the structures of these 14 learning problems necessitates 3 different scoring metrics: for the binary classification of non-bet-facing actions, F1 score; for the multiclass classification of bet-facing actions, multiclass-adjusted F1 score; and for the regression of bet and raise amounts, Mean-Absolute-Error.

The choice of F1 score for binary classification was due to the natural interpretation of the problem as an identification of bets. If the problem is stated as, "will this person make a bet?", it becomes obvious what the positive and negative labels are. The classes aren't horribly unbalanced, but they aren't 50/50, so for these reasons F1 score is a good fit.

For multiclass, the typical metric is accuracy score. However, because these labels are particularly weighted towards folds (especially in the Preflop case), accuracy score is inadequate for measuring the true predictive value; it considers all labels equal, when really, a model that only predicted folds would do reasonably well, better than random guessing. The best response to this situation is to adapt the F1 score, which is designed for imbalanced classes, to the multiclass setting. This is done by taking the F1 scores of each one-vs-all model (of which we have 3) and taking a weighted average according to the distribution of labels.

The main decision for the choice of regression metric is between RMSE and MAE (mean absolute error). When the desired effect is for large errors to be punished linearly proportional to their size, MAE is appropriate. When large errors should be punished with greater effect, RMSE is better. For this problem, making the mistake of assuming a very large raise or bet when it was simply a large raise or bet is not necessarily terrible; the response from the majority of opponents will be fairly robust to this, and so the agent's view of the game and ability to roll forward and plan would not be greatly effected by this mistake. For this reason, I will use MAE instead of RMSE.

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”
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)”

Field	Data Type	Description
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

Field	Data Type	Description
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
Preflop-False	77289	78
Preflop-True	14240503	84
Preflop-True-sample	0	0
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	fold	check	call	bet	raise
Preflop-True	0.672	0.027	0.133	0.0	0.168
Flop-False	0.0	0.621	0.0	0.379	0.0
Flop-True	0.546	0.0	0.335	0.0	0.12
Turn-False	0.0	0.633	0.0	0.367	0.0
Turn-True	0.498	0.0	0.405	0.0	0.097
River-False	0.001	0.634	0.0	0.366	0.0
River-True	0.555	0.0	0.362	0.0	0.083

Exploratory Visualization

My exploratory visualizations will examine the distributions of actions and amounts, as this is the most important information in the dataset (the label). For the amounts, I will then split the data over rounds. This information was previously presented in table form, but it is always a good idea to view information of this structure in a visual format. Figures 1-3 are, in order: the distribution of actions, the distribution of amounts for bets, and the distribution of amounts for raises. I will discuss these in order.

Preflop, it is clear that the majority of actions will be folds. Across future rounds, the distribution between check/bet and fold/call/raise is remarkably similar, with checks consistently making up roughly 60% of actions in non-bet-facing situations, and folds, calls, and raises taking a 50/40/10 split in bet-facing situations. A model that just predicted folds would get about 55% accuracy overall, which is a good benchmark to be aware of.

Bet amounts peaked for the most part just over 1/2 of the pot size, which follows from standard poker theory. The distributions are roughly (very roughly) normal, except for the River where there is a secondary mode directly at pot-size. Flop bets had a sharper peak than Turn and River, which tended to be more spread out around the mean.

Raise amounts had a distnictly positive skew, as they peaked around 3/4 pot size and decayed from there. The exception was preflop raises. An incredibly distinct pattern emerges in preflop raises, which is that the great majority of players would make 2-pot raises, with strong secondary amounts of 4/3- and 7/3-pot. I would anticipate that this pattern should ease the learning process for this particular network.

Algorithms and Techniques

In this section, you will need to discuss the algorithms and techniques you intend to use for solving the problem. You should justify the use of each one based on

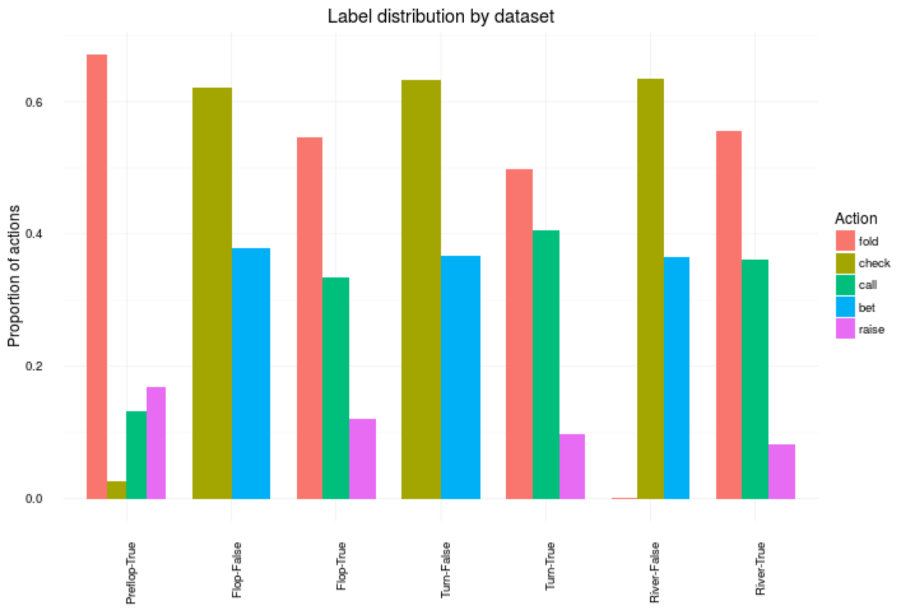

Figure 1: Action distribution

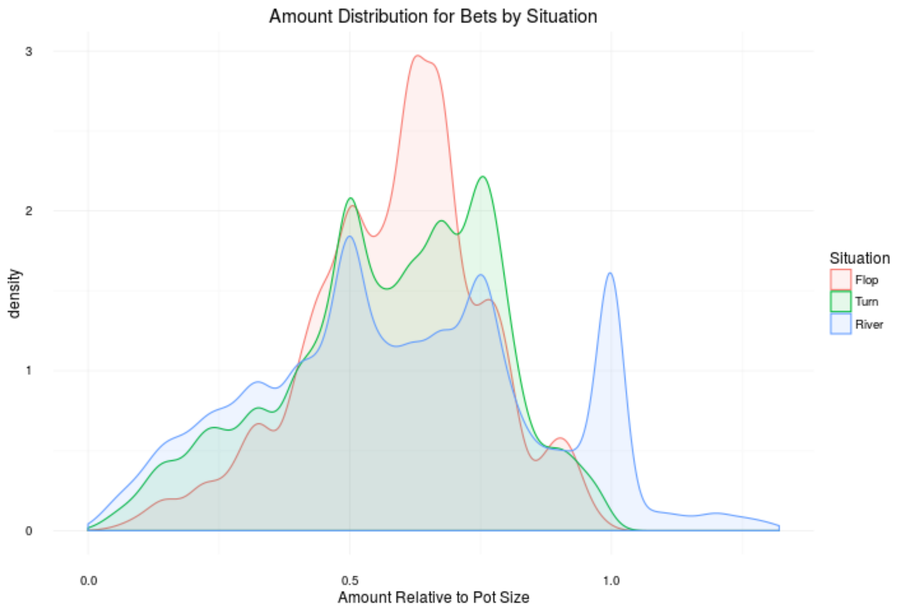

Figure 2: Bet Amount distribution

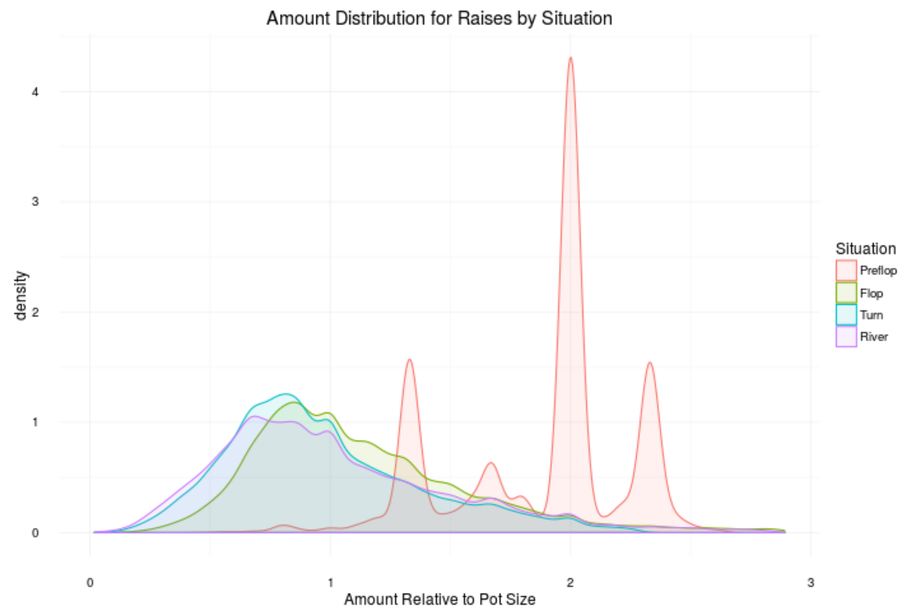

Figure 3: Raise Amount distribution

the characteristics of the problem and the problem domain. Questions to ask yourself when writing this section:

- Are the algorithms you will use, including any default variables/parameters in the project clearly defined?
- Are the techniques to be used thoroughly discussed and justified?
- Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?

Benchmark

In this section, you will need to provide a clearly defined benchmark result or threshold for comparing across performances obtained by your solution. The reasoning behind the benchmark (in the case where it is not an established result) should be discussed. Questions to ask yourself when writing this section:

- Has some result or value been provided that acts as a benchmark for measuring performance?
- Is it clear how this result or value was obtained (whether by data or by hypothesis)?

Methodology

Data Pre-Processing

The original data for this project was a corpus of game logs from various online poker sites, such as PokerStars and PartyPoker. These game logs had errors, missing information, names were obfuscated, and because it was from many sources, it had many different formats. The first and most time consuming task of this project was to parse this text into a database of structured data, and then build features from that data.

The parsing from text to tables was done in `data/lib/fileReaders.py`, where I wrote 5 separate parsing functions for the 5 data sources, then using Bash and MySQL stored them in a relational database. This process, for all ~33,000 files, took approximately 5 hours on my local machine. Once the data was parsed, the next step was to construct the feature set, which was done in `data/lib/buildDataset.py`. The result was 114 features covering various aspects of the environment (discussed above in Problem Statement), most of which were specific to certain subsets of the data. Because of the specific nature of these features (which fell out of the inherently different underlying structure of each situation), 7 subsets of the data needed to be separated. This brought the problem from a task of learning one model to learning many, though their structure would remain the same.

There were few abnormalities in the data in the sense of illegal or improbable actions. The sites are run with restrictions in place to prevent actions that would result in bad data, and because the stakes of these games were quite high in some cases (as these were games played with real money), the decision-making of the players can be trusted somewhat more than a fake-money site. That said, there were numerous instances in the raw text where information had been overwritten and lost, or errors had been made in recording that resulted in strange values, but for the most part these were cleaned in the orginal processing. Thankfully, the obfuscating of names was consistent, so there was no confusion as to which player was which, even though their real names had been removed.

Overall, the hand-crafting of this feature set straight from raw text was the most intensive of all tasks involved in this project.

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?
- Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?
- Was there any part of the coding process (e.g., writing complicated functions) that should be documented?

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

- Has an initial solution been found and clearly reported?
- Is the process of improvement clearly documented, such as what techniques were used?
- Are intermediate and final solutions clearly reported as the process is improved?

Results

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

- Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?
- Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?
- Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?
- Can results found from the model be trusted?

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- Are the final results found stronger than the benchmark result reported earlier?
- Have you thoroughly analyzed and discussed the final solution?
- Is the final solution significant enough to have solved the problem?

Conclusion

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

- Have you visualized a relevant or important quality about the problem, dataset, input data, or results?
- Is the visualization thoroughly analyzed and discussed?
- If a plot is provided, are the axes, title, and datum clearly defined?

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

- Have you thoroughly summarized the entire process you used for this project?
- Were there any interesting aspects of the project?
- Were there any difficult aspects of the project?
- Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

- Are there further improvements that could be made on the algorithms or techniques you used in this project?
- Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?
- If you used your final solution as the new benchmark, do you think an even better solution exists?

Applications

Talk about how this fits into the Poker AI project.