CS 229 Project
Opponent Modelling in Heads-up NL Hold'em

Jon Harris
Vianney Hocquet
Pierre Thomas

December 11, 2009

Introduction

"Heads-Up No Limit Hold'em" refers to the most successful poker game, with only two players and no limit in the bets. Without limit, computer bots are not able to beat a good human player. We focused our work on the opponent modelling, which is a promising field to improve the results of poker bots. We had on average 51 hands per player. Our algorithms must be able to identify quickly our opponent strategy and predict from this inference what the opponent will play. This suggests that we train generic-specific models that we will use when we identify the style of play of our opponent. So we must work on two different parts: classification algorithms and generic predictive algorithms. Among the many ways to deal with the second part of this problem are Bayesian algorithms and neural networks. We used a database of hands played for real money on various websites ad at various limits.

1 Probabilistic Model

In this approach action probabilities are calculated based on the previous two actions in the hand as well as the current pot odds (the size of the pot of bets already made relative to the required bet size). A naive Bayes type of assumption of the independence of the features is used. We use the following notation: A=action, H=hand history, PO=pot odds R=raise (or bet), C=call, Ch=check, F=fold

We first calculate the conditional probability of a particular hand history H given action A based on the raw counts of these events in the data:

$$p(H|A) = \frac{n(H|A)}{n(A)}$$

Our model for the action probabilities based on the pot odds is basically a two-step logit model and has form:

$$\begin{aligned} p(R|PO) &= \frac{1}{1+\exp(-r_0 - r_1 * PO)} \\ p(C|PO) &= \frac{1}{1+\exp(-c_0 - c_1 * PO)} * (1 - p(R|PO)) \\ p(F|PO) &= 1 - p(R|PO) - p(C|PO) \end{aligned}$$

With the above two models we can now calculate the probability of action A given H and PO as

$$p(A|H, PO) = \frac{p(H|A)\, p(A|PO)}{\sum_i p(H|A_i)\, p(A_i|PO)}$$

At any time the predicted action is the action with the maximum probability.

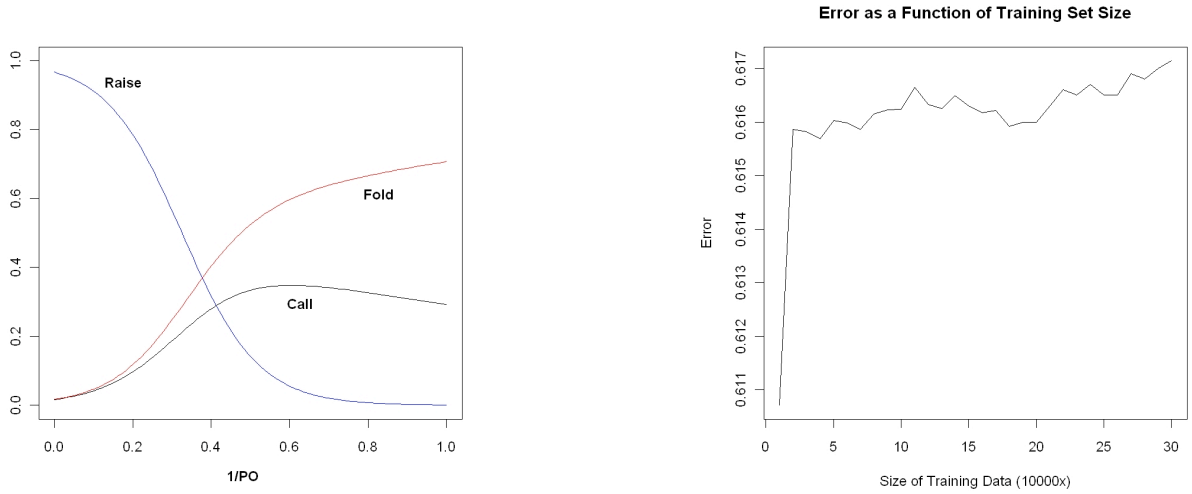

Figure 1: *left*:The figure shows how the different action probabilities depend on PO.
right: This figure shows how the "Error" (actually the percent of correctly predicted actions) depends on the size of the training set used.

2 Identifying Strategies

In Poker a first distinction is whether a player is loose or tight. Does she/he play a lot of hands or not ? Then, depending on her/his play after the flop, is she/he aggressive or weak? One can imagine more classes but for the time being we will focus on the way to differentiate between those. We chose to use as a training set 75% of the hands available. From the start we faced some problems. Most classification algorithms necessitate that the training examples are labeled. But even if we went through all the data "by hand" we could not certify that one player belongs to this specific type for sure. So we tried K-means clustering (used package stats, clust and fpc in R) with 4 clusters hoping that it would result in the above mentioned classification. But in Heads-Up, because of the blinds paid at each game, players have to be much looser. So the numbers significant for a game of 6 players or more don't have any meaning in this case. So we used the classification supplied by clustering. We used the kmeans package.

Won $ when seeing flop

Betting frequency at each point (PreFlop and Post Flop)

Aggressiveness

But this classification is based on statistics which need quite a few hands played to converge. When we will play they won't be available or at least significant. But a betting pattern will often tell us more.Consequently we used other algorithms such as LDA (Linear Discriminant Analysis, library MASS in R) or K-Nearest Neighbors (library class) or SVM (library e1701) to attribute to each player his type of play while playing. The rates for the prediction were quite good. Our error function was built as such : we compared the labeling given to the training set after 10 and 20 hands played with the labeling given 5 hands before the end of the game to avoid that the statistics are skewed by excessive risk taking. And we got those results :

LDA	5%
Knn	3%
SVM	2%

3 Tailored predictions

We used the classification to train four different SVM on four training sets corresponding to the different types of players. For each SVM, the type of inputs were the same. We used a combination of those classifiers on the basis that ensemble prediction will yield better results, which in our case

was marginally true. Then we used those classification not as a direct input for our SVM prediction algorithm but to choose the one which correspond to the type of player identified. That yielded the best results among the three techniques used. However, we trained the SVM on much larger datasets than the other algorithms, so the comparison is not entirely valid. We used the package svm in R.

Error rate SVm average player	27%
Error rate SVM tailored	23%

(with features used).

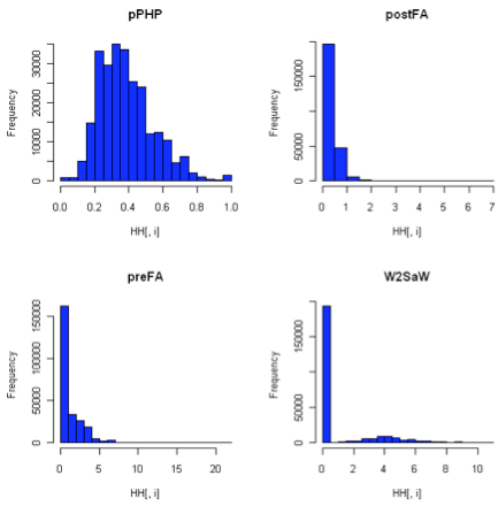

Figure 2: Clustering results

4 Neural Network

Neural Networks are a common tool in machine learning, and have already been applied to opponent modelling (see [2]). We try to prove again the efficiency of neural networks in opponent modelling, to study the important inputs and parameters, and to adapt the opponent classifiaction that we made. However, we used R packages to compute neral networks and did not sepdn time on arificial neural network theory.

4.1 Neural Network modelisation

We used R and the package neural to train, test and setup the parameters of our neural networks. The neural networks we used are MLP neural networks, which are setuped by a back propagation algorithm. The initialisation of the nodes is made randomly. The outputs in our situation were the three boolean values : Did the player fold ? Did he raise/bet ? Did he call/check ? We chose to limit our study to very simple neural network, made of one hidden layer.(this choice was motivated by [2]). Therefore, we had to optimise our results with respect to the number of neurons, the number of iterations to setup the neural network, the choice of the features, and the size of the training set.

4.2 parameters of the neural network

We tried to optmise over the number of neurons (*NbNeurons*), the number of iterations to setup the neural network (*It*), and the size of the training set (*TrainSize*). After some blind trials, we decided to take as standard values $NbNeurons = 5$, $It = 200$, and $TrainSize = 200$. We trained our neural networks and then found the error on a testing set for several values of these parameters (see figure [**?**]).

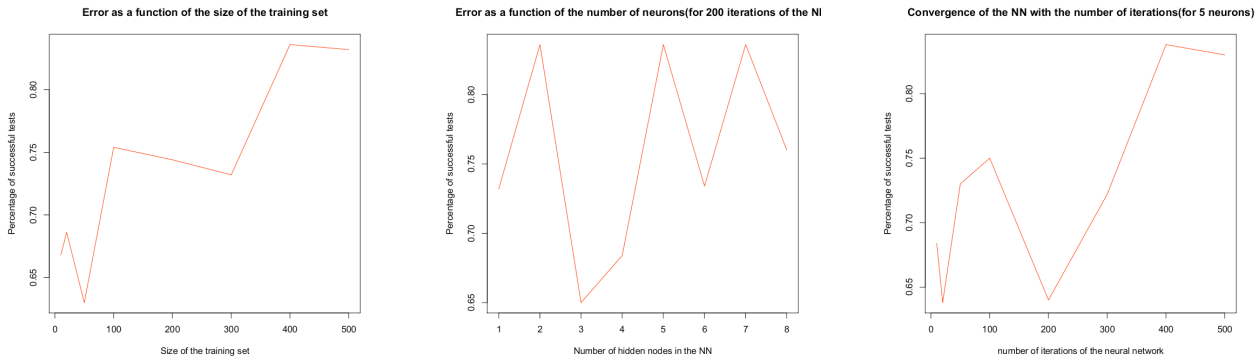

Figure 3: Parameters selection in the neural network

We can conclude that the best parameters are $NbNeurons = 6$, $It = 100$, and $TrainSize = 100$. As expected, the more iteration and the bigger training set, the better results we get, but we need to choose reasonable values to limit the computation time. Error with the number of neurons is harder to interpret.

5 Action prediction

5.1 Action prediction using standard features

Neural networks output the conditional probability of the outputs given the inputs. So, the ouputs of a neural network are the same as the quantity computed by our bayesian algorithm. Our main concern in action prediction was to choose the best features to predict the opponent's next move. From the experience we had from previous experiments, we tried the following features : *1:Action Player, 2:Immed. Pot Odds, 3:Stage=Flop, 4:Stage=Turn, 5:Stage=River, 6:Last act.=bet/raise 7:Flush poss. 8:A on Board 9:K on board 10:(AKQ on board)/(Cards), 11:Previous(P) Action Player 12:P Action 13:P P Action Player 14:Previous Previous Action 15:Pot Total Before Action, 16:calls/Bets*.

With these features, the mean error of the algorithm was 0.25. We compared it to the error the algorithm made if one of the features was taken off. From figure 4, we can see that the most usefull parameter is the pot total before action. It is always interesting to know that most of this value carry a lot of informations on the game. The rest of the parameters contribute to approximatively 5 percent of the error and are equivament to our neural network.

5.2 Action prediction using enhanced features

Finally, we used new indicators to try to improve our results. The training of a neural network is long enough to discourage us from using four differents networks for the four categories of player as done in part *Tailored predictions*. In addition, results were not much better with this technic. However, we added four features to our neural network : those which had been developped to enhance the strategy identification process : *1:the pre-flop odds, 2:a picture of hands played, 3:the pre-flop aggression, 4:the post-flop aggression*. The error of the neural network with these four new features was : 0.166 which is a significant improvment. More precisely, we can see in the following tabular that the pre-flop odds and the post-flop aggression are the key parameters to reduce error. The tabular presents what the errors would be if we added three instead of four features.

Feature deleted	1	2	3	4
New Error	0.312	0.162	0.164	0.230

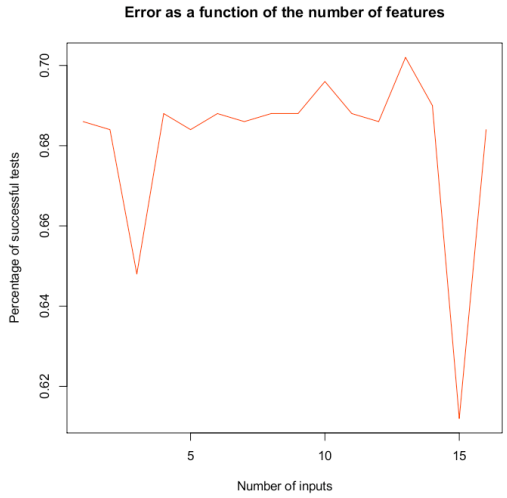

Figure 4: Using the neural network without one of the feature.

Conclusion

The standard bayesian approach is part of a very interesting framework and could be improved by mixing two distributions of action : one trained on a set of hands played by a lot of players, the other one trained on a set of hands played by a lot of players ([1]). However, the algorithm is complicated to adapt to every situation and does not predict actions as well as the neural networks. Another way to approach opponent modeling is to classify the other player. We have seen that these results can be used to improve action prediction using neural networks or standard machine learning algorithm. But the main direction for our work and challenge for potential improvements is how to adapt the algorithms for a real time utilisation as well as how to handle the vast amount of data that we have (more than 600 millions of hands). We would like to thank Stanislas Marion and the website pokerai.org for providing the data.

References

[1] M. Ponsen1, J. Ramon, T. Croonenborghs, K. Driessens and K. Tuyls, *Bayes-Relational Learning of Opponent Models from Incomplete Information in No-Limit Poker*, Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, 2008

[2] A. Davidson, *Using Artificial Neural Networks to Model Opponents in Texas Hold'em*, Alberta University, 1999

[3] D. Billings, A. Davidson, J. Schaeffer , D.Szafron, *The challenge of poker*, Artificial Intelligence 134 (2002) 201-240