Predicting Poker Bets from Observed Players

Author One, Author Two, and Author Three

Department of Computer Science, Anonymous University,
{`authors@university.biz`}

Abstract. The game of Poker has received much attention recently. We consider the popular variant *Texas Hold'em* that is played online frequently. Based on a huge data base of games played exclusively by humans we apply several versions of feed forward neural networks and support vector machines to predict the betting behavior of a player in each state of the game. Neither of these variants outperform all others but a so-called 'meta-decider' that makes a prediction based on the predictions of the ensemble of predictor variants increases the quality significantly.

1 Introduction

The poker game has become more and more the focus of interest in many scientific papers [1]. Poker is a game with imperfect information as you do not know the cards your opponents hold and also the community cards are not known in advance. Hence it is different from e.g. chess where the information about the whole game state is always identical for every player. You need a little luck but also a good strategy to win a poker game. An important factor for a good strategy is not only to regard your own cards and the community cards but also to observe the opponents betting behavior. If you knew which move an opponent would do in the current game state you could adjust and improve your own strategy. Therefore it is important to create a model of your opponent which predicts his future action. For human players it is difficult to discover regularities in the betting behavior of their opponents. Therefore we address this challenge by using knowledge discovery techniques in order to explore the opponents strategies.

2 The Game of Poker

We consider the poker variant *Texas Hold'em* [2], which is currently the most popular one. Before any cards are dealt the two players to the left of the dealer (the button) put a predetermined amount of money (*blinds*) into the pot. Two so called *hole cards* are dealt to every player which are hidden from the opponents. During the game five *community cards* are put on the table. There are four different betting rounds where the player can act – *pre-flop* (no community cards are on the table), *flop* (three community cards are revealed), *turn* (four community cards) and *river* (all five community cards). A player has different possibilities to act. He can *fold* which means that he looses all the money invested yet and quits the game. A player *checks* by just passing the action to the next player and not putting any money into the pot. If the amount of money he puts into the pot is just enough to remain in the game, the action is named *call*. Check and call are mostly considered as the same action because the player neither quits the game nor ups the ante. *Raise* is the term for the action if a player puts long odds into the pot and hence all other players have to react to stay in the game. If there are at least two players who remain in the game after the river round the *showdown* occurs. Now the player chooses five cards from his hole cards and the community cards to compose a so-called hand. The player who shows the hand with the highest value wins the game and gets all the money from the pot.

3 Techniques for Predicting Poker Bets

To classify poker game states into classes which represent the players' actions *fold*, *check/call* and *raise*, different supervised learning techniques are used.

3.1 Artifical Neural Networks

Artifical Neural Networks (ANN) have been applied successfully in many research fields to solve classification problems [3]. For opponent modelling they have already been used by Davidson [4, 5]. Inspired by the human central nervous system, an ANN consists of artificial neurons which are connected with weighted links. If the sum of the weighted links of a neuron reaches a defined bias the neuron is active and 'fires'. So the ANN produces an overall output which is representative for a class. Classification with ANN is divided in a learning and a testing phase. In the learning phase input vectors, which represent a game state, are applied to the net and the produced output is compared with the real output. When a wrong output occures, a learning algorithm adjusts the weights in the ANN to produce the correct output. After several iterations the training stops and the classification is tested by new input vectors to check the generalization of the ANN. In our experiments we used multilayer perceptrons, a special kind of ANN. We applied the backpropagation and resilient-propagation algorithm [6] to adjust the weights in the ANN.

3.2 Support Vector Machines

A Support Vector Machine (SVM) [7] is a classification method which maps the input data into a high dimensional space. In that space the SVM constructs a hyperplane which separates the data points in two classes. The margin of this hyperplane is maximized so that the distance between the data points from different classes is as large as possible. Since mapping the input data into a high dimensional space and constructing the hyperplane is very complex and timeconsuming the so-called kernel trick is applied. With the kernel trick the separation of the classes is performed without explicitly mapping in that space. Hence the classication can be applied much faster. So-called C-SVM use slack variables to allow classification errors. The C determines a trade-off between the classification error and maximizing the separating hyperplane. As the SVM is a binary classifier but we need a separation in more than two classes we applied the One-Versus-All-Multiclass-SVM. Therefore several single SVM are constructed which separate one class from the rest. The single SVM with the highest output assigns the class.

3.3 Meta-Decider

The output of the ANN and SVM can be considered as a distribution of the classes. Normalized you get a probability triple PT $= (f, c, r)$ which shows the probability $P(\text{fold})$, $P(\text{check/call})$ and $P(\text{raise})$ for the particular action [5]. The one with the highest value determines the class. Since we constructed many different deciders, it is hard to decide which prediction should be taken finally. So we constructed two meta-deciders which combine the probability triple of all single deciders. The first one determines the highest value of the particular class, the second one calculates the average of all values for a particular class from the different deciders and a new probability triple is constructed.

4 Database and Experimental Setup

As mentioned before we classify the current game state in the classes fold, check/call and raise. The state is represented by several attributes describing the boardcards and the opponent as well as other generic information. All attributes are shown in table 1 and they are scaled in the interval [0, 1]. The poker data stems from Michael Maurer's IRC Poker Database [8]. There are approximately 10^7 of played poker games available which were logged by an observer program from 1995 to 2001. All players whose betting behavior we tried to explore played in a poker room (*nobots*) where poker bots were not allowed so that games are played human players only. The blinds are 10 and 20 Chips and there is no limit of the betting amount. We have only generated and classified input vectors for the rounds flop, turn and river but not for pre-flop. The betting behavior before the flop differs a lot from the behavior after the flop [4] and we have expected better results omitting the pre-flop round. For every player we have generated eleven training and test sets, the first in chronological, the other in uniformly distributed order.

In pre-experiments we determined parameters for the learning algorithms Backpropagation and Resilient-Propagation to adjust the weights in the ANN as well as parameters of the kernels for the SVM.

4.1 ANN Setup

The ANN experiments have been conducted with the Stuttgart Neural Network Simulator [9]. It provides two modules for backpropagation (BackpropMomen-

Table 1: *Input attributes*

attribute	description	type
queen	queen on board	boolean
king	king on board	boolean
ace	ace on board	boolean
pairOnBoard	pair on board	boolean
color	(max. # of identically colored cards -1) / 4	real
straight	straight possible	boolean
aggressivePlayerBehind	# of aggressive players behind / active players	real
numberBets	# of bets before the player (0: 0, 1: 0.5, $\geq$2: 1)	real
position	relative position to the other players	real
bankroll	bankroll of player relatively to the chipleaders bankroll	real
blind	player is in big or small blind	boolean
betLastRound	player bet last round	boolean
playerChipleader	player is chipleader	boolean
bankrollActiveChipleader	relative bankroll to active player with most chipstack	real
bankrollActiveLowstack	relative bankroll of active player with lowest chipstack to the observed player	real
playfrequency	frequency of playing the flop	real
betfrequency	frequency of betting	real
stage	current stage (flop: 0, turn: 0.5, river: 1)	real
playersDealtCards	# players dealt cards / 10	real
pActive	# active players / 10	real
chipleaderInGame	chipleader is one of active players	boolean
potsize	current potsize / all available chips at the table	real

tum) and resilient-propagation (RProp). Table 2 shows the parameters applied. Here, η is the learning rate and β is the momentum term which decreases the learning rate in rough and increases it in smooth error planes. γ is the flat spot elimination term and d_{max} the maximum error which is ignored without changing the weights in the ANN. The latter is able to avoid overfitting. Resilient-Propagation uses alterable learning rates. They are limited by η^- and η^+. $\varDelta_0$ and $\varDelta_{max}$ are the start and the maximum values for the amount of the weight change. With the term α you can control the maximum sum of all weights in the ANN. With Backpropagation the minimum error of the test set has appeared at 20 iterations, with Resilient-Propagation already at 15 iterations. We have applied a logistic activation function and the identity output function for all neurons. The ANNs consist of 22 input neurons and 25 neurons on one hidden layer. The number of output neurons corresponds to the number of possible player actions. So, we use four neurons to distinguish between fold, check, call, and raise. In another ANN we unite check and call resulting in three output neurons.

Table 2: *Parameters of Backpropagation and Resilient-Propagation Learning.*

Output	Backpropagation				Resilient-Propagation				
neurons	η	β	γ	d_{max}	α	η^-	η^+	$\varDelta_0$	$\varDelta_{max}$
3	0.2	0.37	0.1	0.1	3.4	0.5	1.2	0.1	50
4	0.14	0.35	0.1	0.1	4	0.5	1.2	0.1	50

4.2 SVM Setup

The data mining tool RapidMiner (former YALE) [10], [11] provides the implementation LIBSVM for the SVM from Chih-Chung Chang and Chih-Jen Lin [12]. We applied two different kernels. The first one is a linear kernel $\langle \mathbf{x}, \mathbf{x}' \rangle$ (dot product) with the parameter C = 5. The other one is a polynomial kernel $(\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^2$ with C = 100.

4.3 Different enhancements

In order to improve the classification, we experimented to distinguish between the action check and call. Hence we have the four classes fold, check, call and raise. After the classification process, the correct classified samples of the classes check and call are merged again to be comparable to the results with just three classes.

Another method we tried to enhance the classification ratio is to apply classificators for every particular round, flop, turn and river. Theses classificators learn only with samples of that particular round and are tested with samples of that round. For comparison we used an all-classifier which classified all rounds at once based on the complete data sets.

5 Experimental Results

Table 3 shows the best results the particular classification methods achieved. On average a classification ratio of approximately 72% is obtained. The turn round has been classified best and the river round worst. Player 6 is one which is not well predictable, in contrast to player 5. The best classification ratio is achieved in the flop round for player 2 with 84%.

Table 3: *Best classification ratio of particular rounds and players*

Round	Pl. 0	Pl. 1	Pl. 2	Pl. 3	Pl. 4	Pl. 5	Pl. 6	Average
All	0.7046	0.7166	0.7645	0.7168	0.6998	**0.7870**	*0.6092*	0.7141
Flop	0.6747	0.7351	**0.8428**	0.7179	0.7334	0.7804	*0.6010*	0.7265
Turn	0.7566	0.7250	**0.8046**	0.7317	0.6842	0.7974	*0.6457*	0.7350
River	0.7091	0.6978	0.6179	0.7017	0.6656	**0.8077**	*0.6125*	0.6875
Average	0.7113	0.7186	0.7574	0.7170	0.6957	**0.7931**	*0.6171*	0.7158

5.1 Separation of Check from Call

Table 5.1 (left) is an examplary so-called confusion matrix for four classes. The columns show the classes which the classification method predicted and the rows show the real class. The values in the diagonal represent the correctly classified samples and the last cell shows the sum of them. In this example a real fold is predicted 30 times correctly as a fold, never as a check but 4 times incorrectly as a call and 2 times as a raise. As you can see the classification method has classified 232 of 345 samples correctly which corresponds to a classification ratio of 67.2%. By "merging" the results of check and call as seen in table 5.1 (right) the classification ratio is enhanced to 67.5%.

Table 4: *Left: confusion matrix with CR = 67.2% (232/345).*
Right: "merged" confusion matrix with CR = 67.5% (233/345).

	pred. fold	pred. check	pred. call	pred. raise	$\sum$
fold	30	0	4	2	36
check	0	109	0	10	119
call	20	1	7	15	43
raise	7	51	3	86	147
$\sum$	57	161	14	113	232

	pred. fold	pred. call	pred. raise	$\sum$
fold	30	4	2	36
call	20	117	25	162
raise	7	54	86	147
$\sum$	57	175	113	233

Both enhancements and deterioration occur by using four classes as shown in table 5. The best enhancement is achieved for player 4 with 0.0228 in the flop round and the ANN with resilient propagation. For player 5 deteriorations up to 0.0416 (river round and ANN with backpropagation) are actually gained but just enhancements up to 0.016 (turn round and ANN with backpropagation). Hence in some cases it is worthwhile to add a fourth class but in many other cases it is not. But there is another interesting observation. As you have already seen in table 5.1 (left), a real fold has never been predicted as a check and vice versa. There are hardly any wrong classifications between check and call. This behavior occured in almost all other confusion matrices of every player and round. This means that first the classification methods are able to seperate between fold and check very well and second that with the very good separating ability between check and call a specific poker rule has been learned which the classification method does not know explicitly. If there is a game state where a check is not a valid action because a player has raised before then neither the ANN nor the SVM predict a check (less than 0.5%).

Table 5: *Maximum absolute deviation of the classification ratio by using 4 instead of 3 classes*

player	0	1	2	3	4	5	6
max. enhancement	0.0207	0.0122	0.0123	0.0208	0.0288	0.0160	0.0278
max. deterioration	0.0140	0.0159	0.0292	0.0143	0.0163	0.0416	0.0133

Figure 5.1 shows the average classification ratios for all players by the ANN with backpropagation (BP) and resilient propagation (RE) and by the SVM with the linear and the polynomial kernel. The reference line you see in the figures is the result of a simple classificator. This simple classificator always predicts the action a player has chosen most in that particular round. Thus all results above the reference show that the application of classification methods are useful. The lines between the points are just for clarity purposes. Some points are covered by other points but along the lines you can identify the hidden points. You can see in the figures that for both ANN and SVM the differences of the average classification ratios between 3 and 4 classes are marginal. Also the results in comparison betwenn ANN and SVM are nearly identical with some small deviation. But in comparison to the reference almost all methods classify better, e.g. for player the classification by the ANN and SVM is approximately 0.2 above the results of the simple classificator.

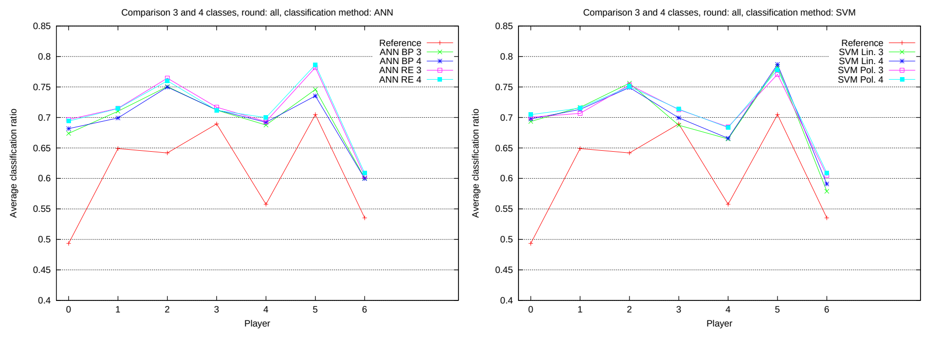

Fig. 1: *Comparison between 3 and 4 classes, round all: ANN (left) vs. SVN (right).*

5.2 Round-specific Classificators

The application of classificators which only classify samples of a particular round has not lead to the expected results. Analogical to the distinction between check and call there are enhancements as well as deviations as you can see in table 6. In some cases, e.g. the turn round of player 0 and player 6, the specific classificator works better than the all-classificator. By contrast for player 4 and player 5 the turn-classifier only achieves worse results.

In figure 2 you can see boxplots of the all-classifier (f.A.) next the round-specific classifier for the flop round. For comparison reasons the reference is plotted again. For player 0, player 4 and player 5 the classification ratios are much better than the reference. For player 1, player 2, player 3 and player 6 the archieved results are in average as good as the reference. In some cases the reference performs better, e.g. the SVM with linear kernel for player 6. Strange outlier occur for the ANN with backpropagation in case of player 5.

Table 6: *Maximum absolute deviation of the classification ratio by using round-specific classificators.*

		pl. 0	pl. 1	pl. 2	pl. 3	pl. 4	pl. 5	pl. 6
max. enhancement	F	0.0042	0.0001	0.0112	0.0034	0.0573	0.0383	0.0381
	T	0.0385	0.0020	0.0128	0.0154	-0.0122	-0.0021	0.0285
	R	0.0256	0.0051	0.0402	0.0143	0.0033	0.0296	0.0310
max. deviation	F	0.0421	0.0180	0.0148	0.0115	0.0139	0.0016	0.0033
	T	-0.0045	0.0075	0.0138	0.0280	0.0472	0.0215	-0.0030
	R	0.0217	0.0242	0.0374	0.0273	0.0315	0.0073	0.0097

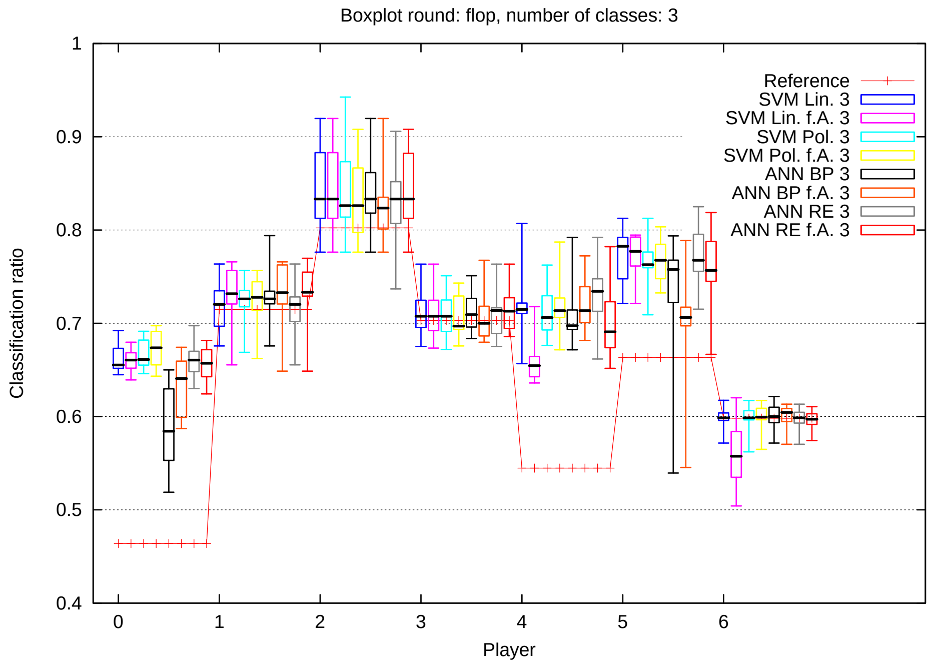

Fig. 2: *Boxplots of every method with all-classifier (f.A), flop-classifier and simple classifier*

5.3 Meta-Decider

We now have many results of the different classification methods. The meta-decider combines all these results as described in section 3.3. Figure 3 displays boxplots for the ANN and SVM using the all-classifier in comparison to the meta-deciders. Mostly the meta-deciders perform as well as the best single decider. In many cases it gains even better results, e.g. for player 0, player 3 and player 6. By the combination of the results the system is more robust and not that vulnerable against outliers. The evolved probability triples can be used by poker bots as e.g. *Poki* from the UACPRG. This bot uses the probability triple for their betting strategy and the update of weight matrices which represent the conditional probability for the hole cards of the opponents [1].

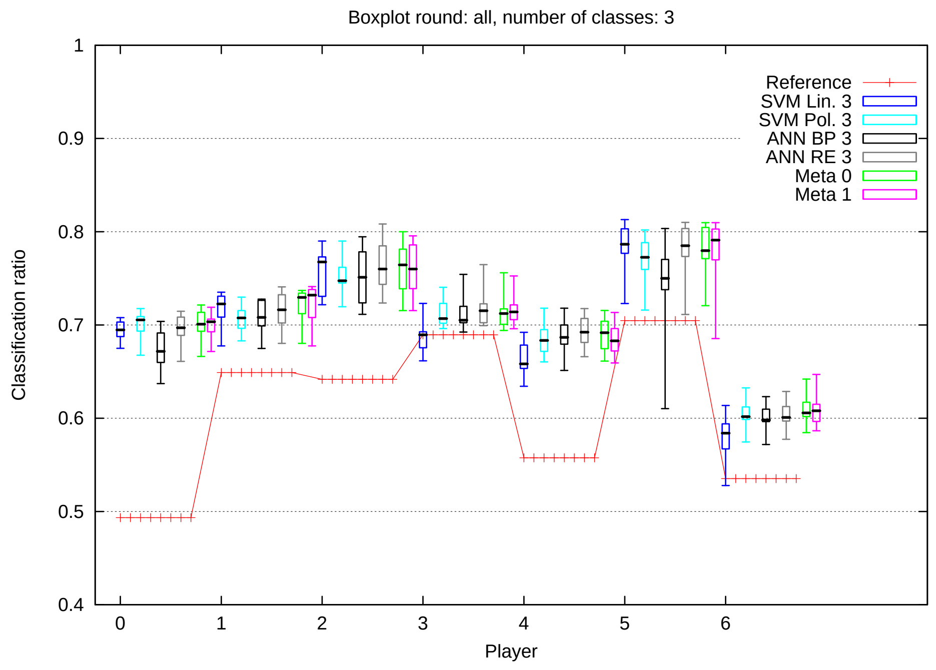

Fig. 3: *Meta-Decider*

6 Conclusions

Summing up we can say both, ANN and SVM are appropriate methods for opponent modeling. No method performs always better than the other one. But the gained results are always of higher quality than those from the simple classifier. The separation between the actions check and call as well as the application of round-specific classificators achieved enhancements but also deteriorations. However the combination of the decisions by the meta-decider lead to very good results. Both methods have learned the poker specific rule that a check is not possible if a player has already raised before. Potentially there are further enhancements possible by applying other classification methods or using more or less attributes for describing the current game state.

References

1. Billings, D., Davidson, A., Szafron, D.: The challenge of poker. Artificial Intelligence **134**(1–2) (2002) 201–240
2. Sklansky, D.: The Theory of Poker: A Professional Poker Player Teaches You How to Think Like One. 3 edn. Two Plus Two Pub (2002)
3. Zhang, G.P.: Neural networks for classification: A survey. IEEE Transactions on Systems, Man, and Cybernetics — Part C **30**(4) (2002) 451–462
4. Davidson, A.: Cmput 499 – research project review: Using artifical neural networks to model opponents in texas hold'em. Technical report, University of Alberta (1999)
5. Davidson, A.: Opponent modeling in poker: Learning and acting in a hostile environment (2002) Masterthesis.
6. Riedmiller, M.: Advanced supervised learning in multi-layer perceptrons - from backpropagation to adaptive learning algorithms. Computer Standards and Interfaces **16**(5) (1994) 265–278
7. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press (2001)
8. Maurer, M.: Michael Maurer's IRC Poker Database http://games.cs.ualberta.ca/poker/IRC/.
9. University of Tübingen: SNNS – Stuttgart Neural Network Simulator, software available at: http://www.ra.cs.uni-tuebingen.de/SNNS/.
10. Rapid-I GmbH: RapidMiner, software available at: http://rapid-i.com/.
11. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In Ungar, L., Craven, M., Gunopulos, D., Eliassi-Rad, T., eds.: KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (August 2006) 935–940
12. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001) software available at: http://www.csie.ntu.edu.tw/˜cjlin/libsvm.