JOURNAL OF APPLIED
COMPUTER SCIENCE
Vol. 20 No. 1 (2012), pp. 119-127

Neural Network Application for Automatic Decisions in Poker

Bartosz Ziółko[1], **Daniel Bochniak**[2], **Grzegorz Jankowski**[2]

AGH University of Science and Technology
[1]*Department of Electronics*
[2]*Department of Computer Science*
al. Mickiewicza 30, Kraków
bziolko@agh.edu.pl

Abstract. *The paper describes a system taking automatic decisions in Poker Texas Hold'em. It can play the game autonomously, so we will refer to it as a bot. Apart from various poker evaluations and decision taking methods, the bot uses a neural network to asses future movements of opponents. A few latest solutions in a field of automatic decisions in conflict situations are also summarised.*
Keywords: *artificial intelligence in games, neural networks.*

1. Introduction

Computer games are a rapidly growing software industry in Poland. Computer games can be also seen as a field to test and develop several types of AI (artificial intelligence) algorithms [1, 2, 3, 4]. One of the tasks of applying AI in games is designing a system for automatic decisions in poker [5, 6, 7, 8, 9].

Our bot for a Texas hold'em poker game was developed and tested using an open source poker platform (OOPoker) which provides a game background like shuffling and dealing cards, making sure if the move is valid considering proper

game rules and provides some basic statistics. Texas hold'em is a variation of the standard poker. The game consists of two cards being dealt face down to each player and then five community cards being placed by the host of the game. A series of three ("the flop") then two additional single cards ("the turn" and "the river"), with players having the option to call, raise or fold after each deal, i.e. betting may occur prior to the flop – "on the flop" or later – "on the turn" and "on the river." The game can be seen as a sequence of states described by values of cards and players decisions. Texas hold'em poker is a symmetric, zero sum, non-cooperative and imperfect information game. Frequently, the poker decision systems are not published because they are used commercially and kept as secrets. This is why, in the next section, we will approach describing the decision taking algorithms more widely.

2. Decision taking algorithms in conflict situations

Game theory describes how to take automatic decisions in various conflict situations in an optimal way and is a well developed field of science. Even though, there are numerous new approaches and applications. Let us start with summarising a few latest achievements in the field.

A conflict resolution system for production cycle decisions was presented [10]. It considers multiple attributes and a set of possible strategies. It can model points of view of a single user or a few users (for example, separately a department responsible for limiting costs and one for keeping proper quality). The presented multi-objective linear goal programming method was applied to the information-oriented service and resources planning of a company which was kept anonymous.

Granular computing was tested in research on conflicts [11]. The idea is based on a presumption that if a subject has accessed an object x, then it is not allowed to access any object in conflict with x. Every conflict objects is implicated in a conflict with particular objects. These two facts result in associating further inherited conflicts between objects.

Decision making is also considered in a risk analysis regarding terrorism [12]. In this case the decision to be taken, is for example, which possible targets A or B should be defended. It leads to relatively simple attacker-defender games, but according to the author, game theory is surprisingly rarely applied to counterterrorism. The author criticises attempts of estimating and using conditional probabilities of threats based on undefined and ambiguous events.

For example, the phrase "given that an attack is attempted" is commonly used in defining vulnerability. This phrase is very ambiguous, because it will mean something different for attackers who decide to act only at specified conditions, like when they find out that vulnerability is high, and for attackers who take decisions randomly with some specified frequency independent of vulnerability.

Constrained rationality approach [13] brings back the decision taking and conflict situations to its roots, namely analysis of goals and plans to achieve the most important goals. It is conducted by formal qualitative goal-reasoning process. Goals, internal and external constraints can be modeled. Constraints are evaluated and realities are reasoned to limit or open ways to goals based on interrelations among these goals and constraints. The model is kept in a tree structure with nodes being goals and constraints connected if there are direct relations between them.

Goals have *achievement* values providing a measure of the achievement level of the goal. Similarly constraints have *achievement* value reflecting the strength of an enforcer. The elements of the tree have also various *prevention* values which describes how much goals can impact negatively on other goals. Constraints also can have *prevention* values to reflect the prevention constraints suffer from stopping them fully from having their effect on the goals attached to them. Finally goals (and only goals) can have *operationalisation* values describing a level with which the agent has committed itself to a set of plans ensuring a degree of operationalisation for the goal.

Necessity of taking decisions can also happen in social conflicts when different activities are expected from one agent at the time [14]. This type of conflicts can be solved often by negotiation and arguing. The presented argumentation framework consists of four main elements. The first one is a schema that captures how agents reason about social influences. The framework contains also possible social arguments of the mentioned schema. The next element is a language with a protocol for facilitating dialogues about the social conflicts. Finally, a set of decision functions that agents may use to generate dialogues within the protocol.

Applying fuzzy logic in decision making was considered as well [15]. Describing preferences and constraints with probability-like values rather then binary ones can make system more flexible and decisions can be taken more carefully.

3. Algorithm of automatic decisions in Poker

Our bot uses several strategies to predict opponent moves and take the best possible decisions.

1. David Sklansky and Mason Malmuth cards group assignment [16] is used on preflop.
2. Do-or-die strategy is used on preflop when being on a short stack.
3. Bot counts a win chance after preflop.
4. Bot takes into consideration win chance calculated in point 3, when (if) raising.
5. Bot is aware of pot odds and it will call almost every time when pot odds are good.
6. Bot attempts to predict opponent's answer for own move (whether he will fold, call or raise) using a neural network (NN) [17].

Let us present more details of these features.

1. David Sklansky and Mason Malmuth assigned poker hands into groups of similarity. There are 9 groups. The first group (with AA, AK, KK, QQ, JJ) is considered the best, where A stands for an ace, K for a king, Q for a queen and J for a jack. The 9th group contains hands without value, so called check-fold hands.
2. Do-or-die strategy means taking a high risk and change the state of the game, as much as possible, in situations were probability of winning tends to 0. In this way, one can either collect blinds or to be lucky and win. Statistically, it is more efficient, because there will be some successes, while with strategy of keeping a weak hand very rarely. In this tactics, it does not matter, what hand do we have, because we assume it is worth nothing.
3. Win chance is counted by assigning random cards to the opponent and filling remaining cards on a table randomly. This procedure is repeated many times. Then chance to win is calculated, as a mean of win chances after all analysed assignments.

4. Raise amount is counted as a random number and win chance, like 1:3 or 1:1, having chance of winning over 80% or only over 60% respectively.

5. Pot odds are well known poker issue. Pot odds are the ratio of the current size of the pot to the cost of a contemplated call

$$pot\_odds = \frac{amount\_to\_call}{pot\_size + amount\_to\_call} . \quad (1)$$

   In other words, if the pot contains 100, and a player must call 10 to stay in the hand, then the player has 10:1 pot odds. Pot odds are often compared to the probability of winning a hand with a future card in order to estimate the call's expected value. Pot odds are mathematically simple, but effective scheme and commonly used by human players. It can be easily applied in an AI algorithm for poker.

6. Neural network was designed in an attempt to answer the question about opponent reactions on particular movements in particular situations. This feature can avoid situations when the bot raises and opponent re-raises. In such situations, if the bot does not have a really strong win chances, it will probably fold and lose money. Also it can help to outbluff the opponent.

   The implemented neural network takes five input parameters: the stage of the game (flop, turn or river) as binary values, for example 100, the pot odds percentage value and the knowledge whether the opponent has already taken part in this round or not (binary value). The output of the neural network is the prediction of opponent's future decision – fold, call or raise, known for training data. Except for the inputs and outputs, there are two additional, hidden layers, each with 6 neurons. The network has three bias neurons as well. The neural network learns using a simple back propagation algorithm. The structure of the network is presented in Fig. 1.

4. Evaluation

The effectiveness of the bot was tested in games by counting number of wins/loses and earnt money. The neural network was trained by 100 games against the same bot without using neural networks, but only five other decision taking routines, described above. This bot will be referred as a test bot. When training of the neural

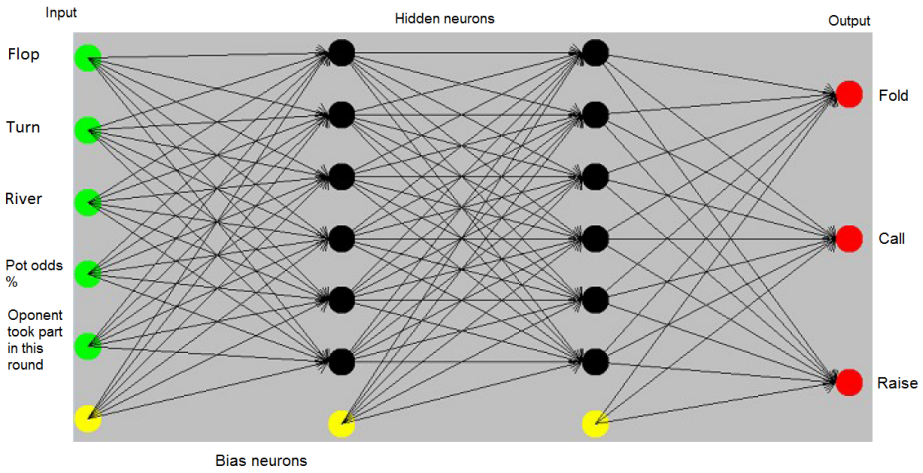

Figure 1. Neural network used in our poker bot. Its architecture was chosen as a compromise between a complicate structure and predictability power, without strict, empirical comparison with other architectures, because it was found very efficient in playing against all available bots

network was finished, the bot was tested by playing 50 games against the test bot and a few other bot with different strategies. The effectiveness of predicting the opponent's next move occurred to be quite accurate. 97% of the predicted calls really occurred to be a call, and 79% of the predicted raises really occurred to be a raise. This knowledge can be used to modify own moves and play less or more aggressively.

Prediction of other bot's moves occurred to be either very easy or nearly impossible. Bots named *call* (which always calls/checks) and *raise* (which always raises) are very predictable because they always make the same decision. The neural network had no problems with foreseeing their moves. On the other hand, *random bot* (which makes totally random decisions) is obviously impossible to predict. The last bot, *smart* (which has a more complicated decision model), is also very difficult to predict because it makes its decisions only upon the strength of its cards, which for obvious reasons is secret and cannot be used as an input to the neural network.

Table 1. Results of games against several types of opponents. Each tournament consists of several games. A game ends when one of the sides loose all money

opponent	Tournaments played	Won	Lost	k$
test bot	50	45	5	40
random bot	50	41	9	32
call bot	50	48	2	46
raise bot	50	41	9	32
smart bot	50	39	11	28

5. Conclusions

The tests showed that our bot is very successful playing against other poker bots. It is very efficient against the *test bot*, mainly because the neural network, which can predict opponent's moves, was trained using games against it.

Our neural network has about 50% efficiency of predicting against random bot which acts randomly, so not taking into consideration any game-situations. This reasult was very easy to be predicted. However, because of other features of our bot (weighting card power, win chances, pot odds) and random decisions being a bad startegy, our bot typically wins against the random bot.

The *call bot* was easy to defeat because it will call every time when we have a strong hand. Quite similarly with the *raise bot* as it is sufficient to have a strong hand against him.

Finally, against the best bot – *smart bot*, our bot is better mainly because of improved raises amounts (smart bot simply does a random amount raise, while our bot raise amount depends on winning chances).

Our implementation is GNU licensed and publicly available on http://code.google.com/p/oopokerbot/.

6. Acknowledgments

Thanks to open source OO Poker platform http://sourceforge.net/projects/ oopoker/ where our bot was developed and tested.

References

[1] Baba, N., Jain, L. C., and (eds.), H. H., *Advanced Intelligent Paradigms in Computer Games*, Springer, 2007.

[2] (Ed.), S. C. R., *AI game programming wisdom*, Charles River Media, Inc., 2002.

[3] Funge, J., *Artificial Intelligence for Computer Games: An Introduction*, A K Peters, Wellesley, MA., 2004.

[4] Millington, I. and Funge, J., *Artificial intelligence for games*, Elsevier, 2009.

[5] Billings, D., Davidson, A., Schaeffer, J., and Szafron, D., *The Challenge of Poker*, Artificial Intelligence Journal, 2001.

[6] Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., and Szafron, D., *Approximating Game-Theoretic Optimal Strategies for Full-scale Poker*, Proceedings of the 2003 International Joint Conference on Artificial Intelligence (IJCAI-03), 2003.

[7] Billings, D., Davidson, A., Schauenberg, T., Burch, N., Bowling, M., Holte, R., Schaeffer, J., and Szafron, D., *Game-Tree Search with Adaptation in Stochastic Imperfect-Information Games*, Lecture Notes in Computer Science, Vol. 3846, 2006, pp. 21 – 34.

[8] Davidson, A., Billings, D., Schaeffer, J., and Szafron, D., *Improved Opponent Modeling in Poker*, Proceedings of International Conference on Artificial Intelligence (ICAI'2000), pp. 1467–1473.

[9] Davidson, A., *Opponent Modeling in Poker: Learning and Acting in a Hostile Environment*, MSc. Thesis, University of Alberta.

[10] Ahn, B. and Choi, S., *Conflict resolution in a knowledge-based system using multiple attribute decision-making*, Expert Systems with Applications, Vol. 36, 2009, pp. 11552–11558.

[11] Liu, S., Wang, J., and Lin, H., *Associated-Conflict Analysis Using Covering Based on Granular Computing*, Lecture Notes in Computer Science, Vol. 6328, 2010, pp. 297–303.

[12] Cox, L., *Game Theory and Risk Analysis*, Risk Analysis, Vol. 29, No. 8, 2009, pp. 1062–1068.

[13] Al-Shawa, M. and Basir, O., *Constrained Rationality: Formal Goals-Reasoning Approach to Strategic Decision & Conflict Analysis*, Proceedings of 2009 IEEE International Conference on Systems, Man and Cybernetics, 2009, pp. 1439–1445.

[14] Karunatillake, N. C., Jennings, N. R., Rahwan, I., and McBurney, P., *Dialogue Games that Agents Play within a Society*, Artificial Intelligence, Vol. 173, 2009, pp. 935–981.

[15] Bashar, M., Hipel, K., and Kilgour, D., *Fuzzy Preferences in Conflict Resolution*, Proceedings of 2009 IEEE International Conference on Systems, Man and Cybernetics, 2009, pp. 2332–2337.

[16] Sklansky, D. and Malmuth, M., *Texas Hold'em for the Advanced Player*, Two Plus Two Publishing, 1994.

[17] Tadeusiewicz, R., *Sieci neuronowe*, Akademicka Oficyna Wydawnicza, Warszawa, 1993.