Adaptive Agents in the Context of Connect Four

Olana Missura
Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS
Schloss Birlinghoven, D-53754 Sankt Augustin, Germany
olana.missura@iais.fraunhofer.de

Artificial intelligence (AI) is a part of most computer games and it plays the main role in the players' satisfaction. Games where from the point of view of the player the computer is "dumb" or "too smart" quickly become boring or frustrating.

One of the aspects involved in games' AI is difficulty scaling. For a game to be interesting for a human player, it should be neither too easy, nor too difficult. The conventional method to implement this property is a setting called "level of difficulty" that a player can set up for herself. This doesn't satisfy most people. The skills of a player are changing continuously, not discretely; hence there will be times, when not one of the proposed values for the level of difficulty will match her level. Additionally, level of difficulty setting does not necessarily control the strategy or decision making abilities of a computer opponent, but rather environmental variables. Consider also the fact that generally games require several different skills to play them. The situation when computer can adjust all these skill levels automatically is more user-friendly than offering several settings for a user to set. It is also possible that developers of the game would prefer to keep the information about the specific skills hidden from players so as not to disclose too much data about the mechanics of the game.

In this work we attempt to investigate the two following questions:

1. *To what extent can methods of machine learning be used to develop an online adaptive agent?*
2. *How well does such an adaptive agent perform against humans players?*

Most research on developing online adaptive agents comes from the game developers community. One of the examples is the work of [Hunicke and Chapman, 2004]. In their paper the authors place the task of adaptation on the game engine. It traces and evaluates the player's performance and attempts to adapt the game world in such a way that the game keeps being challenging, but not too difficult for the player. The adaptation is done by adjusting the characteristics of the player's opponents, their numbers and locations, etc.

While this approach certainly has its place in solving the problem of developing games that adapt themselves to the players, modifying the game world is not the answer to the question of creating an online adaptive agent.

An example of an online adaptive agent based on a modified reinforcement learning (RL) approach is presented in the work of [Danzi *et al.*, 2003]. Here the authors use the Q-learning together with a challenge function to build intelligent agents that automatically control the game difficulty level. Q-learning produces a ranking on a set of actions available to the agent in any given state. While the ordinary Q-learning agent chooses the best possible action in every state, the agent designed by the authors uses the challenge function to evaluate the performance of its opponent. Informally, the challenge function tells the agent if the game is too difficult or too easy for the opponent. If it is too difficult (easy), the agent chooses the action that is worse (better) than the one it made before. This approach was evaluated empirically in the context of a fighting game, Knock 'em, against the non-adaptive agents developed by the authors.

The disadvantage of the proposed method is that the agent needs to be trained offline to produce the Q-matrix of the states and evaluations. Depending on the complexity of the game this matrix can be huge. There is no guarantee that while training offline the agent will encounter all possible states, which can lead to it losing the adaptive qualities. While this can be overcome with Q-regression, i.e. replacing the Q-matrix with the Q-function, it does not eliminate the fact that to learn and adapt, the agent needs to play many games.

To answer the questions stated above we needed to address the problem of evaluating a given adaptive agent. For that purpose first we decided on the game that our agents and human players would play: Connect Four. Then we constructed a test environment consisting of two parts. The first part contains four preprogrammed algorithms having four distinct skill levels in Connect Four. The second part provides an environment where human players can play against developed agents and the statistics necessary for evaluation are gathered.

We looked at the problem of creating an online adaptive agent from two different viewpoints: Can the agent estimate the skill level of its opponent and if yes, how can it adapt itself? What can the agent do if such information is not available? As the result of these considerations two distinctly different agents were created and evaluated.

To sum up the main contributions of this work are:

- Design and implementation of a test environment that allows to evaluate the adaptive qualities of a given agent in the context of Connect Four.
- Design and implementation of an adaptive agent for Connect Four based on the MiniMax algorithm. Demonstration of its good adaptive qualities based on the empirical evaluation.
- Design and implementation of an adaptive agent for Connect Four based on the SVM algorithm. Demonstration that its adaptive qualities are not yet satisfactory.

The first of the developed agents, AdaptiveMiniMax, uses a quantative approach to select an appropriate strategy. It evaluates each move made by its opponent and each move available to itself using the same heuristic, builds the ranking on the moves and chooses an appropriate action. To function, AdaptiveMiniMax requires domain knowledge in the form of a heuristic. Without the appropriate heuristic that evaluates the moves or the strategies in the game, the agent would not construct the correct ranking. An additional disadvantage of this method is that AdaptiveMiniMax can play only as good as the underlying MiniMax algorithm. Therefore, it is bound to lose its adaptive properties when playing against opponents who are stronger than MiniMax.

AdaptiveMiniMax showed good performance when playing against the preprogrammed algorithms, adapting well to their respective skill levels, with exception of the optimal algorithm [Allis, 1988], to which it was losing steadily due to the disadvantage mentioned above. In the experiments with the human players data confirming the adaptive qualities of this agent was obtained. There is no correlation between the skill level of a specific player and the percent of games this player won against AdaptiveMiniMax. From the same data it seems that for the majority of players AdaptiveMiniMax chose a strategy that was weaker than the corresponding player's skill level, i.e. the percentage of the games won by humans is mostly greater than 50%. It would be interesting to see how these statistics would change if AdaptiveMiniMax was equipped with memory, that is if it was provided with a way to incorporate the data about the win-loss proportion into the strategy choosing mechanism.

At the moment AdaptiveMiniMax's decision about which move to make is based on the average of the ranking scores of all moves made by its opponent. It is possible that human players make a lot of far from optimal moves in the beginning of the game, when situation on the board is hard to foresee. In this case the average ranking is influenced by these weak moves and it may lead to the apparent weakness of AdaptiveMiniMax when playing against humans. In the future work we would like to investigate how this behaviour can be changed, for example by introducing some kind of discounting scheme, so that the moves made recently have bigger influence on the resulting ranking score than the moves made (relatively) long time ago.

The second agent, SVM Agent, was developed in an attempt to overcome both disadvantages of AdaptiveMiniMax, the need for the good heuristic and the limit on its playing skill. The problem of choosing an appropriate strategy in the game was presented in the context of supervised learning as a binary classification problem, where training data is built from the moves that the agent's opponent made and the agent is making a prediction about which move the opponent would make. An existing implementation of the SVM algorithm [Chang and Lin, 2001] was used to solve this problem. The kernel function was designed to represent the similarities between the pairs of the board states in Connect Four. The detailed description of the problem's formulation and the kernel function can be found in [Missura, 2007].

The experiments were designed to evaluate the SVM Agent's performance using the games played by the preprogrammed algorithms against themselves. As a measure of performance the cross-validation accuracy and the comparison between the SVM Agent's predictions and the moves made in the recorded games were used. The results are dissatisfying. It was to be expected that in the beginning of a game the training set is too small to allow for any good prediction, but there was also hope that as the game progresses and the size of the training set grows, the agent's predictions are going to get better. In reality even though the cross-validation accuracy shows acceptable values (generally around 80%) the predictions that SVM Agent makes are almost always off the mark, and when it does get it right it seems more the case of a random guess succeeding.

Despite these results, we feel that more experimenting can be done with the SVM approach. Adapting the cross-validation procedure to the specifics of our training sets, replacing binary labels with continuous ones, trying out different kernel functions or different types of SVM, or equipping the agent with memory can potentially lead to the improvements. Another way to improve its performance, especially in the beginning of the game, is to provide it with additional domain knowledge, for example of the same kind that AdaptiveMiniMax uses.

References

[Allis, 1988] Victor Allis. A knowledge-based approach of connect-four. The game is solved. Master's thesis, Free University of Amsterdam, October 1988.

[Chang and Lin, 2001] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[Danzi *et al.*, 2003] G. Danzi, A. H. P. Santana, A. W. B. Furtado, A. R. Gouveia, A. Leitão, and G. L. Ramalho. Online adaptation of computer games agents: A reinforcement learning approach. *II Workshop de Jogos e Entretenimento Digital*, pages 105–112, 2003.

[Hunicke and Chapman, 2004] R. Hunicke and V. Chapman. AI for dynamic difficulty adjustment in games. *Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence*, 2004.

[Missura, 2007] Olana Missura. Adaptive agents in the context of connect four. Master's thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, August 2007.