PLAYING TETRIS USING LEARNING BY IMITATION

Dapeng Zhang, Zhongjie Cai, Bernhard Nebel
Research Group on the Foundations of Artificial Intelligence
University of Freiburg
Georges-Köhler-Allee Geb. 52 Freiburg
Email: zhangd, caiz, nebel@informatik.uni-freiburg.de

KEYWORDS
Tetris, Learning by Imitation

ABSTRACT

Tetris is a stochastic and open-end board game. Several artificial players were developed to automatically play Tetris. These players perform well in single games. In this paper, we developed a platform based on an open source project for game competitions among multiple players. We develop an artificial player employed learning by imitation, which is novel in Tetris. The imitation tasks of playing Tetris were mapped to a standard data classification problem. The experiments showed that the performance of the player can be significantly improved when our player acquires similar game skills as those of the imitated human. Our player can play Tetris in diverse ways by imitating different players, and has chances to defeat the best-known artificial player in the world. The framework supports incremental learning because the artificial player can find stronger players and imitate their skills.

A. INTRODUCTION

Tetris was first invented by Alexey Pajitnov et al. in 1984, and remains one of the most popular video games today. It can be found in many game consoles and several desktop systems in PC, such as KDE and GNOME.

Tetris is a stochastic and open-end board game. A piece of block is dropped from the top of the board. The piece is randomly chosen from seven predefined ones, and it falls down step by step. The player can move and rotate the current piece to place it in a proper position. A new piece appears at the top of the board after the current one touches the ground. A fully-occupied row will be cleared and the blocks above it will automatically fall down one step. The goal of the game is to build as many such rows as possible.

Two players can compete against each other in Tetris. When one player places the current piece to clear n rows, the other player will receive an attack of $n-1$ rows, each of which contains $n-1$ empty cells. The attacks are pushed into the game board from the bottom, raising all the accumulated blocks up $n-1$ steps in the board. The player who has no more space to accommodate the next piece loses the game.

The single Tetris game was used as a test-bed in the research in artificial intelligence. Researchers developed artificial players using different approaches [1]. Fehey created a hand-coded player [2], Böhm et al. employed genetic algorithms [3], and Szita et al. used cross-entropy methods in Tetris [4]. These players can play the single game, clearing hundreds of thousands of rows, which would take several weeks or even months for a human player.

The competition in Tetris is certainly an interesting topic. In theory, the two-player Tetris is much more complex than the single one [5]. Assuming both human and the artificial player handle the piece with the same speed, human players can defeat the best artificial player with ease in the competition mode. To our knowledge, the existing artificial players cannot create many attacks in the competitions. The researchers evaluate their players mainly in single games.

Imitation is essential in social learning [6]. Assuming the similarities between the observations and themselves, humans acquire various skills via imitation. Imitation learning can be applied in robotics and automatic systems in several ways[7]. For instance, Billard et al. built a system according to the structure of the human brain [8]. Atkeson et al. developed a method to explain the actions of a demonstrator, and to use the explanations in an agent [9].

This paper was motivated by building an artificial player for the competitions in Tetris. As a human is superior in the competitions, we employed learning by imitation to clone the game skills of human players. The highlights of this paper can be summarized as follows:

- We developed an open source platform for the competitions.
- To our knowledge, learning by imitation is novel in Tetris.
- Our artificial player can acquire diverse game behaviors by imitating different players.
- Our player has chances to defeat the best-known artificial player in the competitions.
- The framework supports incremental learning.

This paper is structured in the following manner: first, the relation between this work and the literature is addressed in next Section. Then, an open source platform for Tetris competitions is introduced in Section -B. Next, a method is developed to map the imitation to a standard data classification problem in Section -C. After that, the performance of the developed methods is shown in Section -D. Finally, we draw the conclusion and discuss the future works in Section -E.

Related Works

Learning by imitation has been widely applied in robotics,

especially in humanoid robots [8]. The core idea of imitation is to improve the similarity between the imitated system and the imitator, even if certain physical or virtual dissimilarities exist. In this paper, a framework is developed to imitate both human and artificial players. The structure of our approach is certainly different from human brains or the models of the other artificial players. Generally, we follow the idea of learning by imitation. To our knowledge, it is the first time that imitation learning has been applied in Tetris.

The single Tetris games have been used as test-beds in several branches in artificial intelligence [1]. For example, the standard 10×20 Tetris game is still a challenging task for the methods in reinforcement learning [10] [11]. The number of rows that a player can clear is widely accepted as a criteria for the evaluation. So far, several successful artificial players e.g. in [3], [4], and [2], are based on building an evaluation function with linear combinations of the weighted features. These features were listed in [1]. We also employ 19 hand-coded features in our approach, some of which cannot be found in the list. Instead of a linear evaluation function, we use multiple support vector machines in our framework.

Support Vector Machine (SVM) was first proposed by Cortes and Vapnik in 1995[12], and became an important method for data classification. SVM is well-developed. I was implemented in several open source packages which were available in Internet. In this paper, SVM is used as a tool. Our implementation is based on LIBSVM [13]. We modeled the imitation tasks in Tetris as a standard data classification problem which can be finally solved by SVMs.

Incremental learning is mainly about a series of machine learning issues in which the training data is available gradually [14]. It is a special learning method with which a certain evaluation can be improved by the learning process during a fairly long period. In order to do that, we defined a learning paradigm: switching attention learning [15]. In the paradigm, there are multiple learners with their inputs and outputs forming a loop. The performance of one learner generates potential improvement space for the others. Following this idea, Tetris is used as a test-bed. Our artificial player can choose a game played by a stronger player as its target to imitate.

B. AN OPEN SOURCE FRAMEWORK FOR TETRIS

KDE [1] is an advanced desktop platform which provides user-friendly graphic interface. It is an open source project. KBlocks is the Tetris game in KDE. We developed KBlocks to a platform for researches in artificial intelligence. The system components of KBlocks is shown in Figure 1.

KBlocks can be run in two modes: KDE users can use it as a normal desktop game; researchers can choose to start a game engine, a GUI, or a player. The GUIs and the players are connected to the game engine via UDP sockets. The components can be run in one or several computers.

KBlocks can be configured with parameters defined in a text file. The game engine (and the GUI) supports game

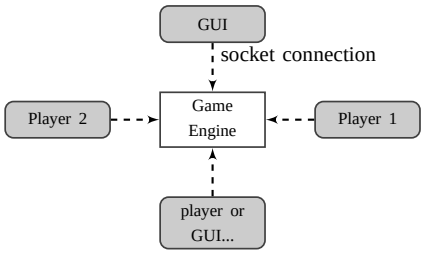

Fig. 1. The System Components of KBlocks

competitions among up to 8 players, in which one player could be a human. A hand-coded artificial player is integrated [16]. It can clear on average $2000-3000$ lines in single games. The competitions can be done in a synchronized mode, in which each player gets the new piece after the slowest player finishes the current placement.

A new artificial player can be integrated into the platform with ease. We provide a source code package in Internet [2], in which the class KBlocksDummyAI is a clean and simple interface for the further development. Graduate students can simply change the source code for their internship or thesis. Researchers can play around with some ideas or organize competitions.

C. LEARNING BY IMITATION

In Section -B, we addressed the functionality of the Tetris platform. In this section, the learning by imitation is discussed in details. First, we give a brief introduction to the system components. Then, the patterns, which are used in the filters and support vector machines (SVMs), are explained. Last, we address how the SVMs are used for data classification in our imitation learning.

The training data of the imitation learning are obtained from the imitated system. In this paper, they are the Tetris games played by the imitated player. We created several models to obtain the skills of the imitated player. The training process receives positive feedback if the models make the same decision as the imitated system. Otherwise, it receives the negative feedback. The imitation learning is successful if the trained models keep the similarity even if the data never appear in the training set.

The learning system consists of several components, as shown in Figure 2. We created three catalogs for these components: the data representation; the algorithms; and the learners. They are illustrated as the gray rectangles, the regular rectangles, and the round-cornered rectangles in the figure.

Figure 2 also shows the relations among the components. We align these components vertically according to the catalogs. A lower algorithm uses the outputs of the upper one as its inputs. The learners computes the models which are used in the algorithms.

The middle column with the dotted arrow lines shows the sequence of the computation in the games. With the current

[1] official cite: http://kde.org

[2] http://www.informatik.uni-freiburg.de/~kiro/KBlocksDummyPlayer.tgz

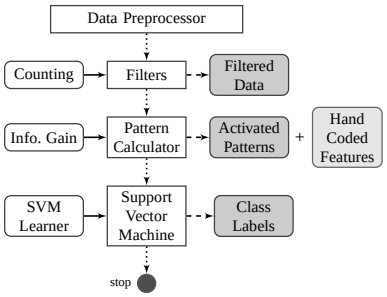

Fig. 2. System Components

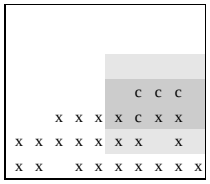

Fig. 3. An example of the Pattern

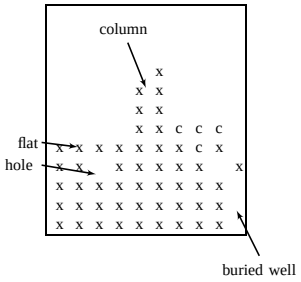

Fig. 4. The Illustrations of the Features

board state and the piece (s, p), the data preprocessor can generate up to 34 candidate placements by enumerating all the rotation and the position of the p. The candidates are filtered because of the heavy computational power required by training the SVMs. The rests of the candidates are passed to the pattern calculator and the hand-coded features. Each candidate is transferred into a vector of the values of the patterns and the features. The vectors are used as the input of the SVM for the prediction. The output of the SVM can be described as how similar a candidate is to the choice of the imitated player. Consequently, the most similar one is labeled as the final choice.

The Learning of the Patterns

Training the SVMs is time consuming. There are 7 different pieces in Tetris: L, J, O, I, T, Z, and S. To place one of L, J, or T, there will be 34 candidates by combining all the possible rotations and positions; O has 9 combinations; I, Z, or S have 17. The candidate chosen by the imitated player is regarded as the positive case. The others are the negative cases. If the size of the training set is 10000, there are about 220000 tuples (cases) in the set. If each tuple is a vector of 39 values, training a SVM from these data would take more than a week using a 2.3GHz PC.

In order to reduce the data set, the types of pieces are used in the data preprocessor to separate the data into 7 subsets. Each subset is used to train its own filter, patterns, and SVM. In other words, seven SVMs work together in the artificial player.

When placing the current piece, human players can first reduce the candidates to a limited number by observing the surface of the accumulated blocks. Then, they choose one from the filtered candidates as their final decision. This idea was used to develop the filters for reducing the amount of data in the learning.

A filter consists of a set of patterns. Figure 3 shows the concept of the patterns. The current piece is denoted by 'c', it is an 'L' in the figure. We use 'x' to denote the already occupied cells. Around the placement, a small field, which is marked in gray, is chosen as the activated area for the patterns. The patterns are smaller than the small field. For example, the deeper gray area in the figure shows a pattern. It contains 5×2 cells. The cells with a 'c' or 'x' inside are occupied.

A pattern can be activated by a placement. As mentioned above, the small field is activated by the placement. All the 5×2 patterns can be enumerated. We move a pattern around the small field. It is activated by the placement, if the occupied cells in the pattern match the occupied cells in the background (the small field).

Filters can thus be learned by counting. If a pattern is never activated by the placements of the imitated player, it can be used to reduce the candidates. Each filter is a set of such patterns. It can be learned by running the activation tests over all the training data.

Support Vector Machines

The patterns are useful not only in the filters but also for modeling the skills of the imitated players. For instance, a pattern was activated 1000 times over the training set, among which 900 were activated by the positive cases. This pattern cannot be used in a filter because there are mixed negative and positive cases. However, activating it apparently indicates that the placement tends to be positive because of the positive to negative rate in the training data. Therefore, the patterns are also used in this section to compute the inputs of the SVMs.

However, the patterns can only get the "local" information. They are checked within the small field around the placement. From another aspect, it is important to consider some "global" parameters in Tetris. For example, a candidate placement can clear 4 rows. This would be important for the game. The patterns, however, cannot express this occurrence.

We designed hand-coded features to acquire "global" information. If the patterns can define the tactics of the games, the features can be used to describe the strategies. In order to define these features, we use Figure 4 to illustrate some phrases: *hole*, *flat*, *column*, and *well*. A well or a hole is buried if it is no deeper than three cells from the surface.

The features are listed in Table I. Items 2 and 3 are for the column. Items 4 – 6 are about the flat. 9 – 11 are for the hole. 14 – 18 are about the well. Our features are compared

TABLE I
LIST OF HAND-CODED FEATURES

1*	How many attacks are possible after the current placement.
2	The number of the columns.
3	The increased height of the column.
4	The increased number of the flat.
5	The decreased number of the flat.
6	The maximum length of the flat
7	The increased height of accumulated blocks.
8	The height difference between the current placement and the highest position of the accumulated blocks.
9	How many holes will be created after the current placement
10	How many holes will be removed after the current placement.
11*	How may occupied cells are added over a hole.
12	The number of removed lines of the current placement.
13*	How well will the next piece be accommodated.
14	If a well is closed by the current placement, how deep is the well.
15	If a well is open by the current placement, how deep is the well.
16*	How may occupied cells are added over a buried well.
17	The number of the open wells.
18	How deep is the well, if it is created by the current placement.
19	Whether a well is removed by current placement.

with the features listed in [1]; the items with * were not mentioned. There are differences in the descriptions of the features because we use them as the inputs of the SVMs. The other researchers developed the evaluation function with the linear combinations of the weighted features.

A large number of patterns can be created by enumeration. For example, an enumeration of 5×2 will create 1024 patterns. It is difficult to consider all these patterns as the inputs of the SVMs because of the required computational power. To our knowledge, there is no trivial way to compute a subset of the patterns which yield to the optimal performance of the SVMs. Therefore, we employ the information gain in decision tree for computing a subset of 20 patterns for each SVM.

SVMs are a popular method in data classification, in which the whole data set are globally classified with a set of the labels. Nevertheless, the data in Tetris are grouped by the current piece. Among the candidate placements of the current piece, the algorithm needs to choose the one which is closest to the choice of the imitated player. LIBSVM [13] provides an API to compute this probability, which is used in our implementation.

The values of the inputs should be within the same range in the SVMs. The patterns always have a value of 0 or 1, which denotes whether or not it is activated by the current placement. The value of features, however, can be much bigger. For example, the maximum length of the flat can be up to 9 in a standard Tetris game. In order to avoid this situation, the values of the features were mapped to 0, 0.5, or 1 in our implementation.

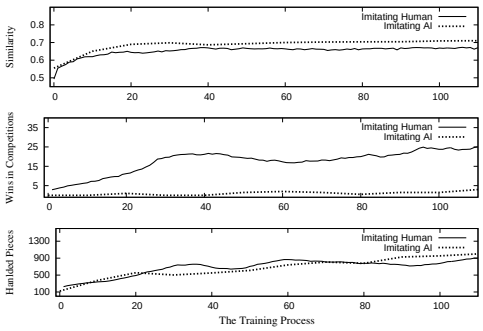

Fig. 5. The Training Process

D. EXPERIMENTS

The experiments were done in a grid system. There are 8 computers in the grid. Each computer has 8 2.3*GHz* AMD CPUs, and 32*G* memory. 64 processes can be run in parallel in the grid.

We recorded 10 games of a human player. Each game lasted more than one hour. The game speed was limited, so that the player had enough time for the game. The player can play Tetris at an amateur level. In total 6720 rows were cleared in these games. The human player was regarded as the first imitated player.

The Fehey's artificial player [2] was run for about 1 hour. It cleared 6774 lines without a restart. The game was recorded as the training set. Fehey's artificial player was the second imitated player.

The two imitated players had very different behaviors in the games. If the human player competes with the artificial player in the synchronized mode, the artificial player has very little chance to win, because it attacks only a few times.

The recorded data were divided into 150 subsets, 120 of them were used as the training set. The rests comprised the testing set, through which the similarity between the trained models and the imitated players can be calculated the rate that the trained model chooses the same placements as the imitated player. The results are shown in the upper plot of Figure 5. The data were averaged over 10 slices.

The solid lines show the performance of the player that imitates the human player. The dotted lines are the player that was imitating Fehey's player. Both imitations achieved a similarity of about 0.7. The curves resemble a typical learning curve because the similarity is regarded as the evaluation in the learning. The similarity cannot be higher because of the differences in the data representation and the models between the imitating system and the imitated systems.

The trained models compete against Fehey's player in the synchronized two-player games. 200 random piece sequences were generated for the 200 games, so that each model was evaluated in the same set of the games. The middle plot in Figure 5 shows the winning rates of the imitating players. The player imitating human finally achieved 0.25 as its rate of wins

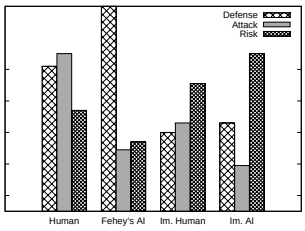

Fig. 6. Behavoiurs of Different Players

in the competitions against Fehey's player. The other imitator did not perform well because the competitions were between the imitating and imitated systems. As the similarity cannot be very high in our implementation, the imitated system should in principle be better than the imitating system.

The trained models also play the single games. The piece sequences used in the games were generated and fixed. The number of handled pieces was used as the evaluation of the player. The results are shown in the lower plot in Figure 5. Fehey's artificial player is better than the human player in the single games, which explains the observation that the imitator of Fehey's player is in the end better than the other imitator.

The training process was designed to search for the maximum rate of the similarity. The rate reached 0.68 at the 30^{th} data slice, and kept this value after that. The performance in the competitions and single games can still be improved after the 30^{th} data slice. This observation indicates that a bigger training set helps to improve the game skills, though it does not improve the similarity in the imitation.

The human player, Fehey's player, and their imitators have different behaviors in the games. In order to show the difference, we designed the evaluations for the attack, defense, and risk. Each player played the same sequences of the pieces in the single games. Attack is the average number of attacks that the player made to clear 100 lines. Defense is evaluated by the average number of cleared lines of each game. Risk is measured by the average height of the placements. The results are shown in Figure 6.

Fehey's player has a defense ability several levels of significance better than the other players. This information was shown as the open-end column in the figure. The other evaluations were mapped to a comparable range. The human player has the best attack ability, which explains how its imitator has chances to defeat Fehey's player in the competitions. The two imitators show quite different behaviors according to the evaluations, which means our imitation learning can generate various artificial players according to the imitated systems.

E. Conclusions

In this paper, we developed a platform for Tetris competitions. The platform is based on an open-source project. The GUIs and players can connect with the game engine via the socket connections. A dummy player was provided as an interface for further development.

We implemented a framework by using learning by imitation. The framework consists of several sets of filters, pattern calculators, and SVMs. The imitation tasks were mapped to a standard data classification problem. The experiments show that our imitators have chances to defeat Fehey's player, which is the best-known artificial player in single Tetris games. And the imitation learning can acquire diverse skills in Tetris games.

There are multiple learners in the framework. The learned artificial player can be used to select an interesting game for further training. The inputs and outputs of the learners form a loop so that each performance of one of the learners create improvement space for the incremental learning.

Discussions

The imitator did not win many games in the competitions. In the next step, we will develop an extra learner for better results in the competitions. The initial experiments showed that the wins in the competitions can be significantly improved by using the rate of wins as the evaluation in the learning.

Tetris was studied mainly in single games. If the sequence of the pieces are known, how can a player win the competitions? AI planning is an interesting direction for further development. We are going to implement the bandit based Monte-Carlo planning in Tetris.

References

[1] S. B. T. Christophe, "Building Controllers for Tetris," *International Computer Games Association Journal*, vol. 32, pp. 3–11, 2009.

[2] C. Fehey, "Tetris AI," http://www.colinfahey.com/tetris/tetris_en.html, 2003, www accessed on 02-August-2010.

[3] G. K. S. M. N. Böhm, "An evolutionary approach to tetris," 2005, in Proceedings of the sixth metaheuristics international conference (MIC2005).

[4] I. A. Lőrincz, "Learning tetris using the noisy cross-entropy method," *Neural Computation*, vol. 18, pp. 2936–2941, 2006.

[5] L. Reyzin, "2 player tetris is pspace hard," 2006, in Proceedings of 16th Fall Workshop on Computational and Combinatorial Geometry.

[6] A. Bandura, "Social learning theory," *New York: General Learning Press*, 1971.

[7] B. S. C. Breazeala, "Robots that imitate humans," *Trends in Cognitive Sciences*, vol. 6-11, pp. 481–487, 2002.

[8] M. M. A. Billard, "Human arm movement by imitation:evaluation of biologically inspired connectionist architecture," *Robotics and Autonomous Systems*, vol. 35, pp. 145–160, 1998.

[9] S. S. C.G. Atkeson, "Learning from demonstration," 1997, pp. 12–20, in Proceedings of 14th International Conference on Machine Learning (ICML97).

[10] K. Driessens, "Relational reinforcement learning," 2004, phD thesis, Catholic University of Leuven.

[11] D. Carr, "Adapting reinforcement learning to tetris," 2005, bachelor Thesis of Rhodes University, Grahamstown 6139, South Africa.

[12] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[13] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[14] Z. S. A. Bouchachia, B. Gabrys, "Overview of some incremental learning algorithms," 2007, pp. 1–6, in Proceedings of Fuzzy Systems Conference, 2007. FUZZ-IEEE.

[15] A. H. D. Zhang, B. Nebel, "Switching attention learning - a paradigm for introspection and incremental learning," 2008, pp. 99–104, in Proceedings of Fifth International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2008).

[16] J. Tang, "Ai agent for tetris," 2009, bachelor Thesis, University of Freiburg, Germany.