Could a FIS choose the right Tetrimino?

Santiago Hincapié-Potes

***Abstract*— In this article is presented a model of fuzzy inference for a Tetris game, with three variables of support: holes in the board, piles and wells. The model is tested in various scenarios with different methods for logical operations and defuzzification.**

I. INTRODUCTION

The Tetris is one of the most sellers games in history. It was created in 1984 by Alexey Pijitnov in the URSS. It has been published by several companies, most prominently during a dispute over the appropriation of the game's rights in the late 1980s.

In Tetris, players must complete lines by moving differently shaped pieces, called *tetriminos*, which descend onto the playing field. The completed lines disappear and grant the player points, and the player can proceed to fill the vacated spaces. The game ends when the playing field is filled. Each game of Tetris always ends because there are sequences of Tetriminos that terminate the game, no matter how well they are placed [4].

Despite its simple mechanics, Tetris is a very complex game. Even in a full information scenario, maximizing the number of rows cleared is NP-complete [5], for this reason, It has been used for more than 20 years as a domain to study sequential decision making under uncertainty. So far, various algorithms have yielded good strategies of gameplay but they have not approached the level of performance reached by expert players playing without time pressure. While at first glance the game of tetris does not seem like a particularly relevant problem, further work on the game has the potential to contribute to important topics in artificial intelligence, specially on real-time strategy scenarios [10].

In this work a tetris agent powered by a fuzzy inference system was designed and implemented.

II. BACKGROUND

Tetris can be modeled as a Markov decision process. In the most typical formulation, a state includes the current configuration of the board as well as the identity of the falling Tetrimino. The available actions are the possible legal placements that can be achieved by rotating and translating the Tetrimino before dropping it.

This formulation ignores some pieces of information provided in some implementations of the game, for example, the identity of the next Tetrimino to fall after the current one is placed. The original version of the game used a scoring function that awards one point for each cleared line. More modern versions of the game, use more complex scoring functions, however most Tetris implementations used by researchers use the original scoring function.

With the standard board size of 20×10 tiles, the upper bound of possible game states is in the order of 7×2^{200} states [5]. Given this large number of states, the general approach has been to approximate a value function or learn a policy using a set of features that describe either the current state or the current state–action pair.

Tetris posses a number of difficulties for research:

- Small changes in the implementation of the game cause very large differences in scores. This makes comparison of scores from different research articles difficult.
- The score obtained in the game has a very large variance. Therefore, a large number of games need to be completed to accurately assess the average performance.
- Games can take a very long time to complete. Researchers who have developed algorithms that can play Tetris reasonably well have found themselves waiting for days for a single game to be over [6].

The most common approach to Tetris has been to develop a evaluation function, where each possible placement of the tetrimino is evaluated to select the placement with the highest value, this design decision makes the human task completly incomparable, since it eliminates the factor that introduces the greatest difficulty: the need for a fast response speed. Next, the features used in these evaluation functions will be described.

a) Tsitsiklis and Van Roy [11]: Used two simple features: the number of holes, and the height of the highest column.

b) Bertsekas and Tsitsiklis [1]: Added two sets of features: height of each column, and the difference in height between consecutive columns. Many authors like [8], [7] used the same set of features

c) Lagoudakis et al. [9]: Added further features, including mean column height and the sum of the differences in consecutive column heights.

d) Fahey [6]: Used six simple features: number of holes, landing height of the piece, number of row transitions, number of colum transitions, cummulative number of wells, and eroded cells.

e) Böhm et al [2]: Introduced new features such as the number of connected holes, number of occupied cells, and the number of occupied cells weighted by its height. These additional features were not picked up in subsequent research.

f) Thiery and Scherrer [10]: Added a couple of features such as holes depth and rows with holes

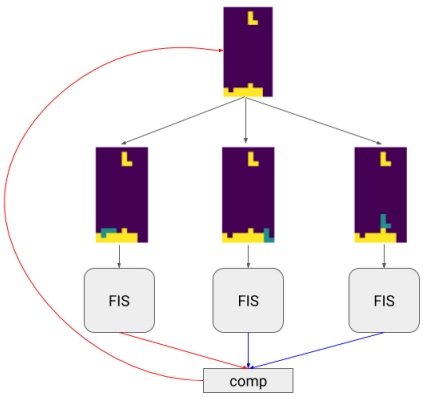


Fig. 1: Diagram of tetris controller

III. Methods

In order to approach this challenging task, a fuzzy inference system (FIS) was proposed. Following the literature described above, the proposed FIS has the task of evaluate the correctness of each possible placement for a given tetromino, in the Figure 1, it is possible to visualize the operation of this agent. The design of this agent is focused on emulating human behavior when playing tetris, reason why it was decided to use the expected stress that a human would have when choosing that position as our measure of correctness.

A. Enviroment

The Tetris enviroment used follows the standard configuration in the literature, i.e. a 20×10 grid and the original scoring funtion. It was developed using OpenAI gym toolkit [3]

B. Linguistic variables

- h: Holes situation
- p: Piles situation
- w: Wells situation
- m: Mood

C. Universe of Discourse

- $h \in \mathbb{Z} \cap [-2, 57]$: quantified with the number of new holes created. Hole refers to any empty block that has some occupied blomck above.
- $p \in \mathbb{Z} \cap [-4, 4]$: quantified with the difference between the high of the largest column between boards
- $w \in \mathbb{Z} \cap [-4, 4]$: quantified with the difference between the largest absolute differences between columns between boards
- $m \in \mathbb{R} \cap [0, 15]$: Being 0 the best mood and 15 the worst

D. Linguistic categories

- h: none, medium, high.
- p: nice, could be better, bad.
- w: remove, accumulate more.
- m: good, half-hearted, bad, crisis.

E. Membership functions

Figure 2 shows the respective membership functions for each one of the linguistic categories described above. They are defined as:

- h:
  - **none**: $dsigmoid(x, 0, 2, 0.5, 4)$
  - **medium:** $dsigmoid(x, 1, 1, 2, 4)$
  - **high**: $sigmoid(x, 2, 2)$
- p:
  - **nice**: $sigmoid(x, 0.5, -4)$
  - **could be better**: $gaussian(x, 1, 0.5)$
  - **bad:** $sigmoid(x, 1, 3)$
- w:
  - **removed:** $sigmoid(x, 0, -3)$
  - **accumulate more:**$sigmoid((x, 1, 3)$
- m:
  - **good:** $gaussian((x, 2, 1)$
  - **half-hearted:** $bell(x, 2, 3, 5)$
  - **bad:** $gaussian(x, 8, 1)$
  - **crisis:** $sigmoid(x, 12, 2)$

F. Rules

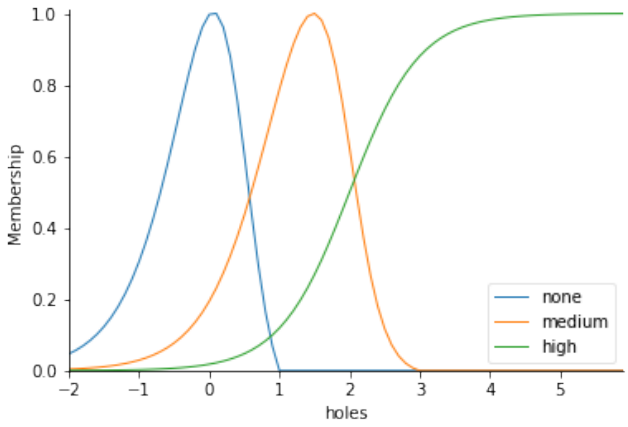
For each rule, it is defined its t-norm and s-norm as the conventional minimum and maximum functions, excluding the first one, that uses as t-norm the drastic norm.

- if [h is **not** none] and [h is **not** medium] and [h is **not** high] then [m is crisis]
- if [h is none] and [p is not that bad] or [w is accumulate more] then [m is good]
- if [p is nice] and [w is removed] then [m is good]
- if [h is mid] and [w is accumulate more] then [m is half-hearted]
- if ([h is **not** none] and [h is **not** medium]) and [w is accumulate more] then [m is bad]
- if [p is nice] and [w is accumulate more] then [m is good]
- if [p is could be better] and [w is accumulate more] then [m is half hearted]
- if [p is bad] and ([h is **not** none] and [h is **not** medium]) then [m is bad]
- if [h is medium] or [p is could be better] then [m is half hearted]
- if [p is bad] and [w is accumulate more] and [h is high] then [m is bad]

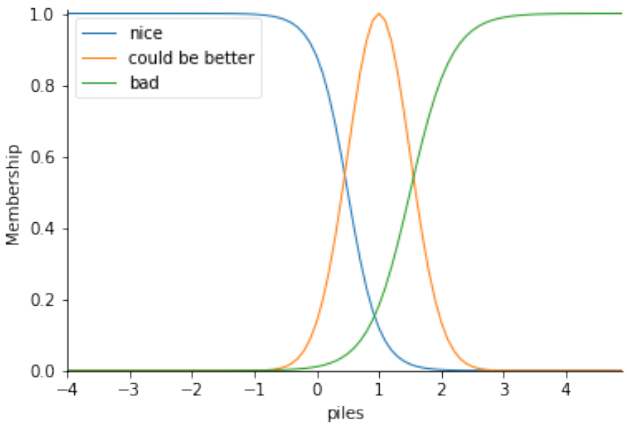
To facilitate the reading of the rules, in the Figure 3 the response surface of the different rules was showed.

IV. Results

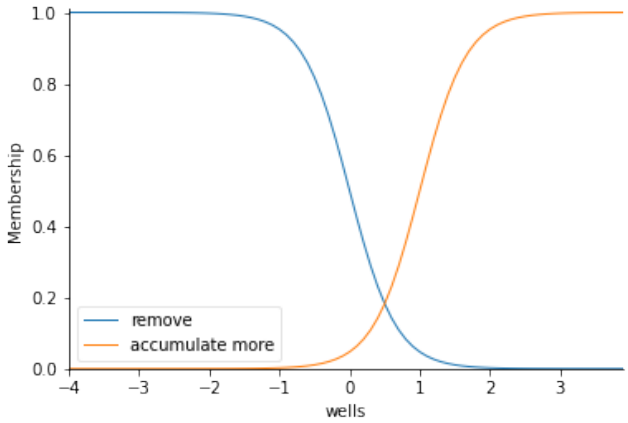
Tetris just like any other sequential decision problem presents a large variance in their results. In order to make a correct comparison between the different parameters, the distributional results of 1000 independent runs was be considered.



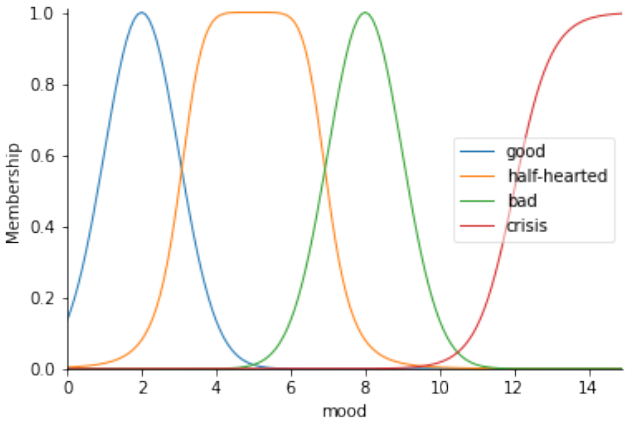
(a) Holes categories



(b) Piles categories

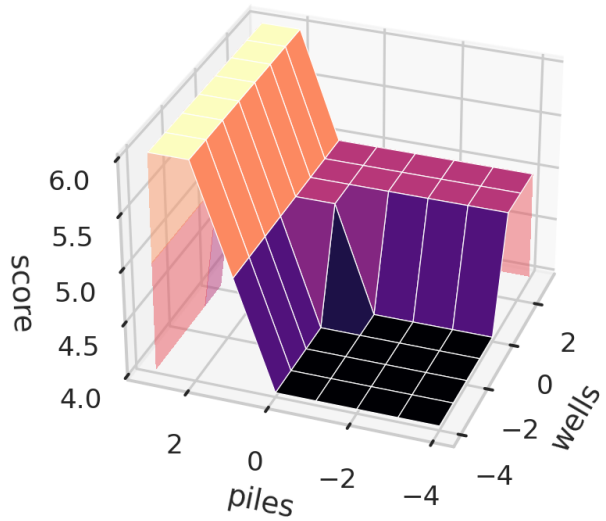


(c) Wells categories

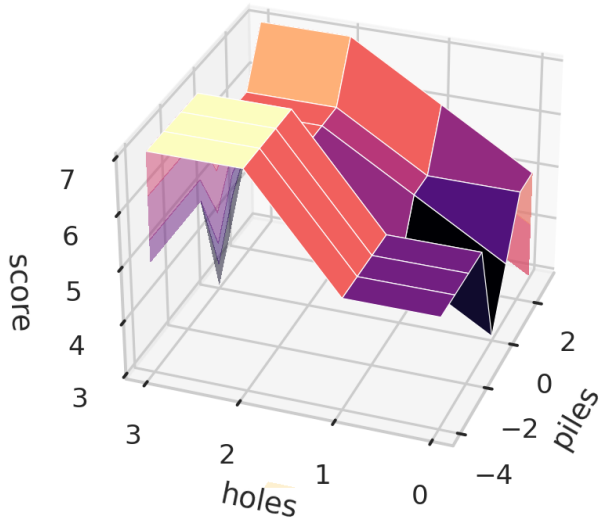


(d) Mood categories

Fig. 2: Membership functions of the different linguistic variables



(a) Response surface piles vs wells



(b) Response surface holes vs piles

Fig. 3: Response Surface for the score

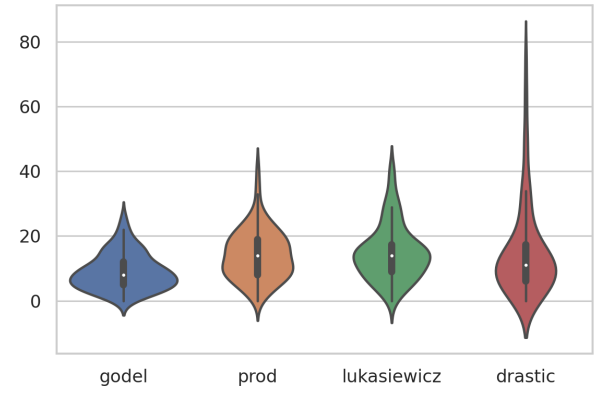


Fig. 4: Comparison of different norms

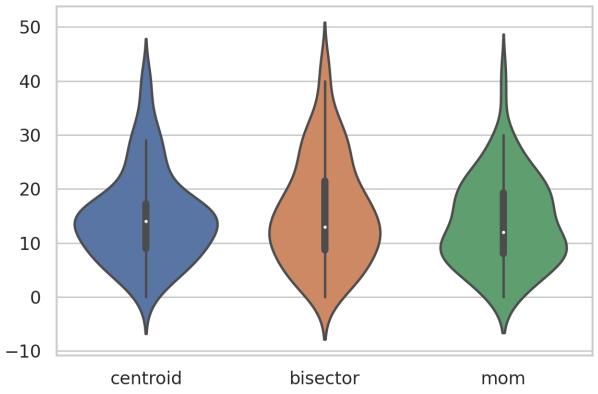


Fig. 5: Comparison of different defuzzification methods

A. T-norm

Different norms were tested, the Figure 4 shows the results of each one of them in a violin plot. It is evident that the more drastic norms have a heavier tail in their distribution, this can be due to an strategy that maximizes the score taking advantage of some edge cases. However, on average, all the result are quite similar.

B. Defuzzification method

As done in theprevous section, different defuzzification methods were tested, and the Figure 5 shows the results of each one of them in a violin plot. The centroid method has the best average score over all the tested methods, and it also condenses more mass around the mean, which may be a sign of a more stable and reliable strategy.

V. CONCLUSIONS

Even though FIS can be used in a large range of problems, in the matter of choosing the right position to place a Tetrimino does not seem to fit well. There are too many variables to be considered, probably more than the ones that can be covered by writing rules one by one rather than letting the system learn by itself. Even when the proposed model is quite small, similar models in different frameworks yield better results with less effort. It would be interesting to combine some evolutionary strategies with a FIS, so the system has the capability to adapt and get information of the previous states of the game, and probably thinking ahead. In this article is presented a model of fuzzy inference in cancer risk, with three variables of support: habits, heritage and age. The model is tested in various scenarios with different methods for logical operations and defuzzification.

REFERENCES

[1] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. 1st. Athena Scientific, 1996. ISBN: 1886529108.

[2] Niko Böhm, Gabriella Kókai, and Stefan Mandl. "An Evolutionary Approach to Tetris". In: 2005.

[3] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[4] Heidi Burgiel. "How to Lose at Tetris". In: *The Mathematical Gazette* 81.491 (July 1997), p. 194. DOI: 10.2307/3619195.

[5] Erik D. Demaine, Susan Hohenberger, and David Liben-Nowell. "Tetris is Hard, Even to Approximate". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 351–363. DOI: 10.1007/3-540-45071-8_36.

[6] Colin Fahey. *Tetris AI*. 2003. URL: https://www.colinfahey.com/tetris/tetris.html.

[7] Vivek F. Farias and Benjamin Van Roy. "Tetris: A Study of Randomized Constraint Sampling". In: *Probabilistic and Randomized Methods for Design under Uncertainty*. Springer London, 2006, pp. 189–201. DOI: 10.1007/1-84628-095-8_6.

[8] Sham Kakade. "A Natural Policy Gradient". In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. NIPS'01. Vancouver, British Columbia, Canada: MIT Press, 2001, pp. 1531–1538.

[9] Michail G. Lagoudakis, Ronald Parr, and Michael L. Littman. "Least-Squares Methods in Reinforcement Learning for Control". In: *Methods and Applications of Artificial Intelligence*. Springer Berlin Heidelberg, 2002, pp. 249–260. DOI: 10.1007/3-540-46014-4_23. URL: https://doi.org/10.1007/3-540-46014-4_23.

[10] Christophe Thiery and Bruno Scherrer. "Building controllers for tetris". In: 2009.

[11] John N. Tsitsiklis and Benjamin van Roy. "Feature-based methods for large scale dynamic programming". In: *Machine Learning* 22.1-3 (Mar. 1996), pp. 59–94. DOI: 10.1007/bf00114724.