Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Artifictial neural network as an opponent in Quoridor

Bachelor Thesis

2016 Michal Hoľa

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Artificial neural network as an opponent in Quoridor

Bachelor Thesis

Study programme: Applied Informatics
Study subject: 9.2.9 Applied Informatics
Department: Department of Applied Informatics
Advisor: Mgr. Peter Gergeľ

Bratislava, 2016 Michal Hoľa

Declaration of Authorship

I do solemnly declare that I have written the presented thesis by myself under careful supervision of my thesis advisor without undue help from a second person other than that specified.

. . . . . .

Acknowledgement

I would like to thank my advisor Mgr. Peter Gergeľ for suggestions, help, guidance and friendly approach.

Abstract

...absctract text here...

0.1 Introduction

0.1.1 Quoridor Rules

Quoridor is abstract board strategy game for 2 or 4 players with size of 9x9 (81) squares. This thesis covers 2 player version of this game.

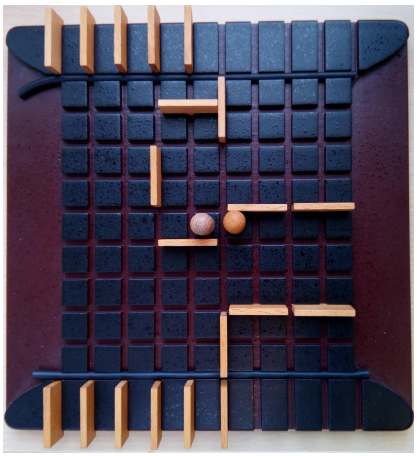

fig. 1: real board

Each player starts with a single pawn in the center of the edge on the opposite side as the opponent. The goal for each player is to reach the opposite edge.

Player also starts with 10 walls (fences) in the stock. Walls are two space wide and can be placed in the groove that runs between the spaces. Placed wall blocks pawns paths forcing them to go arount it. Walls once placed can not be moved nor removed. Wall can not be placed to the position already occupied or crossing by other wall. Also, wall can not cut off the only remaining path of any pawn to his goal.

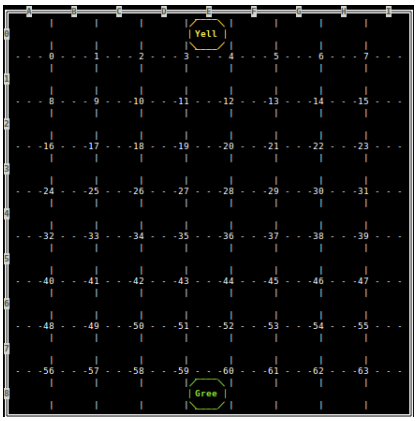

fig. 2: game start

When player is on turn, he must place wall, if he has left some, or move his pawn to adjacent (not diagonal and unoccupied) space. If opponent's pawn stands on an adjacent space, current player can jump with his pawn to all the places where the opponent pawn can move.

0.1.2 Q-Learning

Q-Learning is reinforcement learning method. It learns an action-value (q-value) function for any finite Markov decision process used to find optimal policy for the agent. After it has learned the q-value function, it can follow the optimal strategy by choosing best q-value. ?citation? Algorithm is value iteration update for every observed state ?citation?:

$$Q_{t+1} \leftarrow Q_t(s_t, a_t) + \alpha \cdot (R_{t+1} + \gamma \cdot \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)) \tag{1}$$

where $Q_t(s_t, a_t)$ is old value, α is learning rate, R_{t+1} is reward after performing action a_t in state s_t, γ is discount factor and $\max_a Q_t(s_{t+1}, a)$ is maximum optimal future value.

0.1.3 Quoridor complexity

Estimated game state complexity was $3.9905 \cdot 10^{42}$ [Mertens (2006) (2)], however, this is very rough estimate, since it includes many states multiple times where it counts with permutations instead of combinations of walls.

$$S_p = 81 \cdot 80 = 6480$$
$$S_f = \sum_{i=0}^{20} \prod_{j=0}^{i} (128 - 4i) = 6.1582{\cdot}10^{38} \quad (2)$$
$$S = S_p \cdot S_f = \mathbf{3.9905 \cdot 10^{42}}$$

Average branging factor of the game has been calculated to be 60.4 and average game length to be 91.1 [Glendenning (2005)]. This Mertens used to compute game-tree size $G = 60.4^{91.1} = 1.7884{\cdot}10^{162}$.

My approach with estimating state complexity will be similar. I will estimate maximum states of this game, which will include impossible states such as:

1. pawn in the winning position where it could not end due to walls
2. pawns not having the path to the winning position
3. walls crossing each other
4. pawn in the winning position and on turn

Moreover, this estimate will count as a different state when different player is on the move, and also, different number of walls in players stocks. Both of these could make the game very different in the outcome.

S_p was corrected to not include both pawns in the winning positions. $f(i)$ stands for different wall counts in the stocks. 2 in $2{\cdot}S_p{\cdot}S_f$ counts different players on turn.

$$S_p = 81{\cdot}80 - 9{\cdot}9 = 6399$$
$$f(i) = \begin{cases} i+1 & \text{if } i <= 10 \\ 21 - i & \text{if } i > 10 \end{cases} \quad (3)$$
$$S_f = \sum_{i=0}^{20} f(i) \binom{128}{i} = 1.7796{\cdot}10^{23}$$
$$S = 2{\cdot}S_p{\cdot}S_f = \mathbf{2.2775{\cdot}10^{27}}$$

The result is maximum number of possible states, which is significantly less then former estimate. However, even if I could evaluate 10^6 states in one second, it would take around $7.2172{\cdot}10^{13}$ years to evaluate this many states. Also, lets assume it takes on average 50B of memory per state. Then I would need approximately $1.1387{\cdot}10^{29}$ bytes of memory to store all states in the computer. This is why simple Q-learning is not feasible.

0.1.4 Perceptron

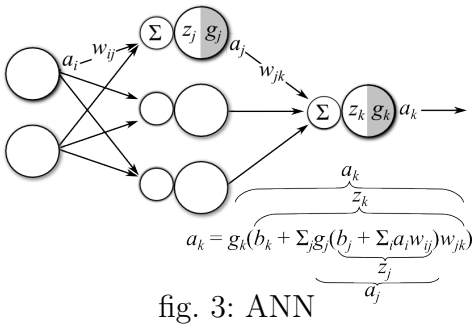

fig. 3: ANN

Artifitial neural network (ANN) is a family of models inspired by biological neural networks used in computer science to approximate functions with large number of inputs. Generally, artifitial neural network is presented as a system of interconnected neurons exchanging messages between each other.

These connections have weights that can be adjusted based on experience which makes the network capable of learning.

Perceptron is an algorithm for supervised learning of binary classifiers where one neuron has multiple weighted inputs and single output. Single perceptron can learn to decide between two linearly separable classes. Multiple perceptrons in multiple layers (MLP) use arbitrary activation function which makes it able to perform classification or regression based on the activation function chosen.

0.1.5 Perceptron estimating Q-learning

Main advantage of ANN is its ability to generalize. This is has been used in game othello [van der Ree and Wiering (2013)] with combining Q-learning so that network has learned to estimate q-values for each state. Iteration update (4) did not include learning rate α since ANN has also learning rate so it can be adjusted there.

$$\hat{Q}^{new} \leftarrow r_t + \gamma \max_a \hat{Q}(s_{t+1}, a) \quad (4)$$

References

Lisa Glendenning. Mastering quoridor, May 2005. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.5204&rep=rep1&type=pdf.

P.J.C. Mertens. A quoridor-playing agent, June 2006. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.5605&rep=rep1&type=pdf.

Michiel van der Ree and Marco Wiering. Reinforcement learning in the game of othello: Learning against a fixed opponent and learning from self-play. April 2013. ISSN 2325-1824. URL http://www.ai.rug.nl/~mwiering/GROUP/ARTICLES/paper-othello.pdf.