Convolution-Based Gomoku Game State Evaluation Algorithm

Peizhi Yan
(Lakehead University, Thunder Bay, Ontario, Canada
pyan@lakeheadu.ca)

Abstract: Convolution is frequently used in pattern recognition tasks, such as face recognition and image classification. As same as many other board games, the traditional approach to evaluate a given Gomoku game state is by using a lot of conditional statements to recognize specific patterns and score the game state regarding the patterns. In this algorithm, the simple 2-dimensional convolution operations replace many conditional statements. This algorithm tremendously reduced the number of codes in a Gomoku game evaluation function, therefore, convenient for coding and debugging. Furthermore, this approach also has enlightening significance in designing other board game evaluation algorithms.

Keywords: Convolution, Gomoku, Evaluation Function
Categories: D.2.0, G.1.3, I.2.1, J.0

1 Introduction

Gomoku game is an ancient board game played by two players in each game. It is traditionally played on a game board with 15×15 grid intersections. The first player who forms an unbroken chain of five stones in the same color in a row is the winner. The unbroken chain of five stones could be vertical, horizontal, or diagonal (left-diagonal, right-diagonal). So there are four basic winning patterns. Not like many other board games such as chess and Shogi, the patterns on a Gomoku game board are intuitive. In other words, the winning patterns can be directly perceived through the visual sense. However, in the traditional approach of designing a Gomoku game evaluation function, the designers need to use many codes to recognize the specific visual patterns on a game board (or game state). Therefore, in a traditional Gomoku game evaluation function, there are many redundant conditional statements and loop statements. Some recent studies attempt to train a convolutional neural network as a value network (like an evaluation function) or a policy network (to make move predictions). However, the performance of those approaches is pretty weak. Moreover, training a convolutional neural network to play Gomoku is much harder than training it as an image classifier. Motivated by the convolution operation in a convolutional neural network, we designed a convolutional approach to recognize specific visual patterns on a given Gomoku game state.

Our algorithm uses 2-dimensional convolution operation to process the encoded Gomoku game state with some pre-defined filters to recognize and count the number of specific visual patterns on the given game state. The designers of Gomoku game evaluation functions can define the specific patterns as filters. This approach

significantly reduces the complexity of development and maintenance of the Gomoku game evaluation function.

The rest of this paper is structured as follows. Section 2 gives the algorithm in full details, including the encoding of Gomoku game state as well as the detailed steps of applying this algorithm to evaluate a Gomoku game state.

2 Algorithm

One of the prerequisites of this algorithm is the pre-defined filters. Those filters are all 2-dimensional matrices. Each filter contains a specific visual pattern. We also need to assign weights to each of the filters. The weight represents how important the pattern it associates with is in the game state (a more considerable weight represents a higher importance level). In our experiment, we use only 16 filters to detect 16 different patterns (see Figure 1). We use a 15×15 2-dimensional matrix of integers to encode the Gomoku game state: 0 represents the empty position; 1 represents the white stone (or the stone of the human player); -1 represents the black stone (or the stone of the artificial intelligence player). We also need an integer matrix that has the same size of the encoded Gomoku game state as the value gradient matrix. Each value in the value gradient represents the value to make a move at that position (this equivalent to a game search tree only has depth 1).

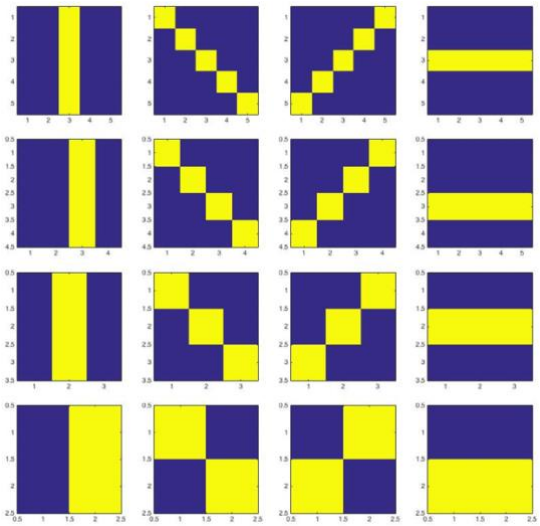

Figure 1: An example set of filters. The first row of filters are 5×5 matrices, the second row of filters are 4×4 matrices, the third row of matrices are 3×3 matrices, the last row of filters are 2×2 matrices. In this figure, blue represents the value is zero, yellow represents the value is one.

We use integer x and y to represent row index and column index. B is the encoded game state matrix, V is the value gradient matrix. We also need a set of filters $\psi = \{F_1, F_2, F_3, \dots, F_n\}$, a set of weights $\omega = \{w_1, w_2, w_3, \dots, w_n\}$. The overall procedure of this algorithm is described as follows:

Step 1: Initialize B and V with all zeros. Create an inverse version of B, name it as B' ($B'_{x,y} = -B_{x,y}$).

Step 2: Loop through all the positions in B (totally 15×15=225 positions). For each non-empty (occupied) position (x,y), set the value of $V_{x,y}$ to 0. For each vacant position, do step 3. Once step 2 finishes, stop the algorithm. V is the final value gradient matrix.

Step 3: Temporarily set the value of both $B_{x,y}$ and $B'_{x,y}$ to 1, do step 4 to get the value: ν. $V_{x,y} = \nu + f(x,y)$, where f is the constraint formula (see 2.2).

Step 4: For each filter $F_i \in \psi$, do the 2-dimensional convolution operation (see 2.1) with B, make the output matrix as M. Count the number of the values in M that equal to the number of non-zero values γ in F_i, make it as c. Multiply c by w_i, make the result as ν_a. Do the same thing to B', make the result as ν_b. Return the value $\nu = (\nu_a + \nu_b)/2$ to step 3.

2.1 2-Dimensional Convolution Operation

We apply full 2-dimensional convolution operation to the game state matrix (with zero padding) with the filter (size is a×a). The size of the output matrix is: b×b, where b = (15+2×(a-1))-(a-1) = 15+(a-1). When the filter detects the same visual pattern in the game state, it will generate a value γ which is the number of non-zero values in the filter. By counting the number of values that equal to γ in the output, we will get the number of the visual pattern in the filter that appears in the game state.

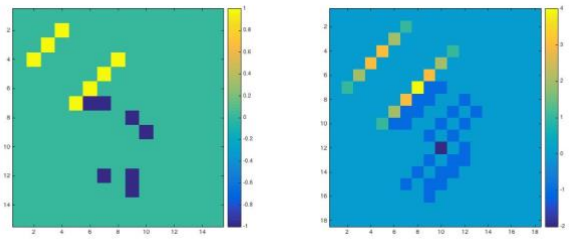

Figure 2: Left sub-figure is the encoded Gomoku game state. Right sub-graph is the output after a full 2-dimensional convolution operation with the filter in Figure 1 (the third filter in the second row).

2.2 Constraint Formula

Without consideration of the stones on a Gomoku game board, the more a position near the center of the game board, the higher the value is. This skill plays a significant role in the first several moves in each Gomoku game. To constraint the Gomoku artificial intelligence make a move near the center of the board in the first several moves, we come up with a constraint formula $f(x,y) = \beta/\sqrt{\{[x-(15/2)]^2+[y-(15/2)]^2\}}$, where β is a tuneable parameter (we use $\beta=10$ in our experiment).

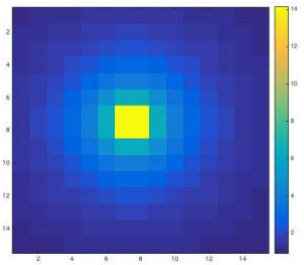

Figure 3: The output of f(x,y) when $\beta=10$, and both x and y $\in$ [0,14].