Learning of Evaluation Function in Digital Curling Considering the Probability of Scores at Each End

Kotaro Ataka,[1] Wataru Noguchi,[2] Hiroyuki Iizuka,[3] Masahito Yamamoto[3]
[1]Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan
[2]Center for Human Nature, Artificial Intelligence and Neuroscience, Hokkaido University, Sapporo, Japan
[3]Faculty of Information Science and Technology, Hokkaido University, Sapporo, Japan
{ataka, noguchi, iizuka, masahito}@complex.ist.hokudai.ac.jp

Abstract

Curling is a team sport in which two teams, each with four players slide stones alternately on the ice. Curling is a very strategic team sport so that where to deliver stones is crucial to win. To discuss strategies in curling on the computer, the curling playing simulator called "digital curling" has been developed by Ito et al. We have developed a curling AI program based on the game tree search. Following the previous work by Yamamoto et al., we used the expected scores distribution at the end of *end* for the evaluation function of the game tree search, and developed a neural network model for predicting the expected scores distribution from the game states. In this paper, we extended our previous model by adding new features of game states for the inputs. We analyzed our model's prediction ability and found that it is more difficult for our model to accurately predict the expected scores distribution for the state such that many stones exist in the play area and many collisions will occur.

Introduction

Curling is a winter sport that has been designated as an official event of the Winter Olympics since 1998. As it is said to be "chess on ice", it is a highly strategic game. Actually, in curling, it is necessary for the players to not only have a high level of technical ability to deliver accurate shots but also correctly judge the game situations and consider what is the best shot for the winning.

Digital curling has been developed by Ito et al. (Ito and Kitasei 2015) to discuss curling strategies on a computer. Uncertainty is implemented in the state transition to emulate real curling. In digital curling, it is possible to analyze and discuss better strategies by letting curling AIs with different strategies fight each other.

Curling is a team sport, but it can be regarded as a two-player game like chess by considering one team as one player. Curling AI has been developed based on the game search, which is widely used for a two-player game. For example, Ohto and Tanaka applied Monte-Carlo Tree Search (MCTS) to digital curling (Ohto and Tanaka 2017). The MCTS method has succeeded in many games including the backgammon which is also with uncertainty like curling. As another approach, expectimax (Ballard 1983; Hauk 2004) which is a game tree search algorithm for games with uncertainty is applied to digital curling. Yamamoto et al. discretize actions and states space to applying a game tree search algorithm to digital curing. In their method, the expected value of the game states is efficiently calculated by convolving state values with the probability distribution of arrival points of curing stone (Yamamoto, Kato, and Iizuka 2015).

In this study, we aim to make the evaluation function more accurate to analyze curling strategies. Yamamoto et al. propose the evaluation function using the expected scores distribution at the end of *end* and its learning method (Yamamoto, Kato, and Iizuka 2018). We also use a neural network as the model and propose adding new features to the input to the model.

Curling

In curling, players slide stones towards the target circle area called "house". The sliding action is also called "deliver" or "shot". In most cases, the stones used in the game are colored by red or yellow. A team consists of four players. The player of each team delivers stones alternating with the opponent. In each *end*, each player delivers two stones, and thus, each team delivers eight stones in total. A part of the field seen from above is shown as Fig. 1. The stone is supposed to be delivered upward. The stones delivered between hog and back lines without touching the sidelines are regarded as valid plays. Otherwise, the stone will be removed from the game. The player can adjust the velocity of the stone even after the stone is delivered by "sweeping" the surface of the ice with a brush.

The score is determined by the number of stones in house after 16 stones are delivered by both teams at each *end*. The team gains points by the number of stones that exist in or touch the house and are closer to the center of the house than any opponent's stones. For example, the first and second closest stones represented by A and B in Fig. 2 are yellow and the third closest stone represented by C is red, and in that case, the team which has yellow stones gets two points.

Because it is considered advantageous to deliver the last shot, the delivery order is switched: the scoring team deliv-

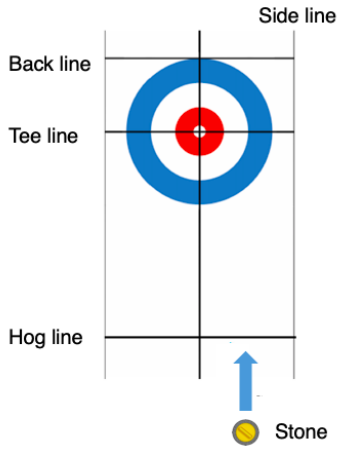

Figure 1: A part of the curling sheet. The direction of the stone is indicated by the blue arrow.

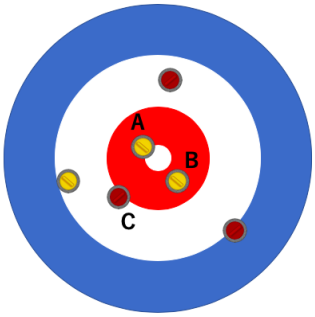

Figure 2: Example of the result at the end of *end*. Yellow team gets two points.

ers the stone first at the next *end*. If the point for each team is zero, the order is not switched. The game is composed of either eight or ten *ends*. The number of *ends* is ten in most national championships, but to save time the number of *ends* is eight in many competitions. The team with the higher overall scores wins.

The team that delivers the last stone can score relatively easily by playing the stone closest to the center of the house or placing the stone inside the stone. Therefore, they usually try to get more than one point to expand the lead. On the other hand, the team which delivers the first stone tries to lose only one point to obtain the advantage to deliver the last stone in the next *end*.

The stone trajectory changes depending on the direction of rotation. When rotated clockwise, the trajectory curls to the right and when turned counterclockwise, it curls to the left. It is known that the stone trajectory does not change greatly depending on the rotation speed. This turning of stones increases the variety of strategies in curling.

Digital Curling

In curling, the strategy is very important and directly leads to victory or defeat. In other words, it is necessary to be able

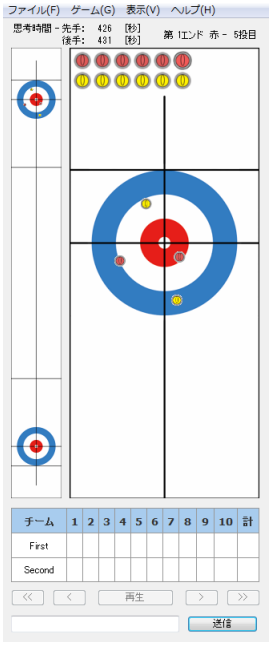

Figure 3: The appearance of the digital curling.

to correctly judge where and how to deliver stones throughout the game. However, the best strategy depends on the situation of the stone arrangement, the number of remaining *ends*, score difference, etc. Since it is necessary to take into account the uncertainty of the shot due to changes in the ice surface condition and the skill of the player, it is further difficult to select the correct strategy.

To discuss and analyze curling strategies on a computer, Ito et al. have developed the curling simulator which is called digital curling (Ito and Kitasei 2015). Figure 3 shows the appearance of digital curling. In the digital curling, two curling AI can play the curling game based on the client-server architecture. When the client passes the velocity and turn of the stone to the server, the server simulates a delivery of the stone with the given velocity and turn. The algorithm described in the paper by Ito et al. (Ito and Kitasei 2015) is used for the trajectory calculation, and the two-dimensional physics engine Box2D is used for the collision calculation.

When the velocity and direction of rotation are obtained, the result is uniquely determined, but the uncertainty because of the player's skill is implemented by adding random numbers (noise) according to a certain distribution to the velocity obtained from the client. In addition to the velocity, the arrival point of the shot is also deviated by noise which follows a constant distribution.

In the case of arrival point, the noise distribution has a larger variance in the y-direction than in the x-direction, which reflects the fact that the adjustment of the weight (delivery speed) is more difficult in real curling. In real curing, the velocity of the stone can be dynamically adjusted by sweeping action, and the variance in the velocity cannot be simply expressed by the noise. However, for simplicity, sweeping action is not implemented in digital curing, and the

variance of the velocity is simply expressed by the noise and the scale of the noise is constant. Although digital curing does not completely simulate real curing, it still simulates the continuous and stochastic aspect of real curing. By developing curing AI that can perform a strong strategy, we can investigate and analyze what strategy is effective for dealing with such continuity and stochasticity.

Curling AI

In curling, the game progresses by two teams delivering stones alternately. By considering one team as one player, it can be regarded as a turn-based two-player game. In such a two-player game, a game tree search is effective. For dealing with uncertainty of digital curing, expectimax (Ballard 1983; Hauk 2004) can be used. However, it is not practical to apply standard game tree search and expectimax to digital curing directly because the number of candidate moves is innumerable as the candidate moves take continuous values of velocity and direction of rotation. To efficiently apply game tree search and expectimax to digital curing, we discretized the candidate moves and made them finite by making the arrival points of the stone correspond to grid points in play area. In the discretized space, the branches of the game tree, which are expanded due to uncertainty, can be shared between different moves. Consequently, the computational complexity is saved and the efficiency is improved by convolving with a known probability density function rather than by repeating random simulation.

Game Tree Search

To apply the game tree search to curling, we first discretize candidate moves. Before the random numbers were added, the combination of the velocity and direction of the stone uniquely corresponded to a certain arrival point. Therefore, candidate moves can be represented by arrival points. The number of arrival points inside the play area is $60 \times 80 = 4,800$. The points behind the play area are also included in candidate moves. These candidate moves behind the play area is for implementing strong shots that are intended to remove stones. The number of those arrival points is $60 \times 70 = 4,200$ whose intervals among points same as those inside the play area. Thus the total number of candidate moves is $18,000 = (4,800 + 4,200) \times 2$. For the same arrival point, there are two ways depending on whether the direction of stone rotation is clockwise or counterclockwise.

In digital curling, each node of the game tree has a stochastic bifurcation due to the uncertainty in state transitions, and these nodes are called chance nodes. A simple way to find the evaluation value of a chance node is repeating random simulations until the evaluation value converges. However, that method takes too much time and results cannot be obtained within a realistic time. To reduce the computational complexity, we adopt the method using the probability distribution by the Billiard AI (Smith 2007; Archibald, Altman, and Shoham 2009). In their method, the expected value of a move can be calculated by simulation without using random numbers.

The expected value of a shot to a certain arrival point is obtained by referring to the value of a nearby point of that arrival point. In digital curling, the error between the assumed arrival point and the actual arrival point follows a predetermined distribution. For each arrival point, the evaluation value is calculated in advance by performing a simulation without adding random numbers. The expected evaluation value of the target point is obtained by convolving the evaluation value of each arrival point with a certain probability distribution. The amount of calculation is reduced by pre-calculating evaluation values that are repeatedly used in convolution.

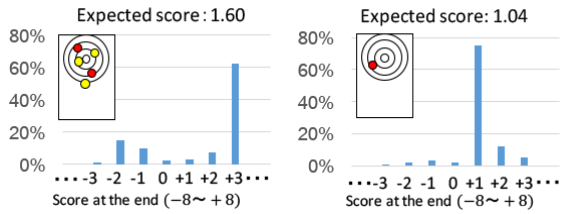

Figure 4: An example of expected scores distribution for two states.

Evaluation Function

To apply the game tree search to find a better strategy in curling, the evaluation function which evaluates the value of the current state is required. The evaluation function maps the current state to a scalar of the state's value. Our evaluation function mainly consists of two elements: the expected scores distribution, and the game equity table.

Expected Scores Distribution In digital curling, as the random values which follow a certain probability distribution are added to the velocity of the stone, the outcome of the shot is not always as intended. Therefore, how many scores the player gets at the end of an *end* also follows some probability distribution. We call this the expected scores distribution.

Game Equity Table If the player can predict the end score appropriately, the decision-making process of the player in each turn will be improved. For example, the state shown left in Fig. 4 is superior in terms of the expected scores to the right state in Fig. 4. However, the right state is better than the left state when the team leads more than one point or ties at the last *end* because at least one point is enough for the team. In this way, the player can increase the winning percentage by avoiding to lose scores instead of aiming high scores.

To realize such decision-making process, the following evaluation function E_{nn} was designed as follows:

$$E_{nn} = \sum_{i=-8}^{+8} y_i \times w(r, d, i), \tag{1}$$

where y_i is the i-th output of the neural network and $w(r, d, i)$ is the winning percentage for the team which gets i points by d points differences with r *ends* remaining. The matrix of the values of $w(r, d, i)$ is called "Game Equity Table" inspired by the technique used in the backgammon AI,

Difference of scores	The number of ends remaining				
	4	3	2	1	0
3	88.7%	91.9%	94.6%	96.2%	100.0%
2	70.7%	77.1%	79.4%	88.1%	100.0%
1	54.5%	60.9%	55.7%	67.7%	100.0%
0	34.2%	34.0%	27.9%	26.0%	22.0%
-1	18.9%	16.2%	12.2%	4.2%	0.0%
-2	6.5%	3.4%	2.1%	1.1%	0.0%
-3	1.5%	0.0%	1.4%	1.1%	0.0%

Figure 5: An example of a games equity table.

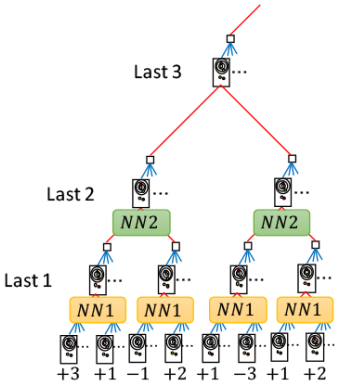

Figure 6: Learning of other neural networks NN_x.

and it is calculated based on a large number of self-plays in advance.

The output of the neural network y_i represents the probability to get i points, and the set of y_i for all i represents the expected scores distribution. By combining the expected scores distribution and the winning percentage, the value of the evaluation function E_{nn} becomes the expected winning percentage.

Learning of Expected Scores Distribution

In this paper, we use a deep neural network as the model for predicting the expected scores distribution from the game states. Generally, the expected scores distribution, which we want to predict by using neural networks, depends on which move is chosen. For simplicity, we fix the policy to select a move. The best policy is to take a move whose winning rate is the largest; however, we generally cannot know which move can provide the largest winning rate in advance. Instead, we decided to use policy such that the expected scores are maximized throughout the experiment.

Input to the Model

The input of the neural network are the values that represent the information of each stone: the x and y coordinates of the stone, the owner of the stone, whether the stone is in the house or not, whether the stone is in play area or not, the distance from the center of the house to the stone, whether the opponent's stone exists inside or not. Totally there are seven values for each stone. The first five input features are also used in the previous model (Yamamoto, Kato, and Iizuka 2018). The last two input features are newly introduced by this paper for providing information about whether the stones are involving scores or not more explicitly. The closest stone to the tee, which is the center of house, is called No. 1 stone and the i-th closest stone is called No. i stone. The input consists of stone information (7 per stone as mentioned above) arranged in order of distance from the center. The outputs of the network are the probability distribution of the score that the team gets at the end of *end*. There are 17 possible scores to get: the integer value from -8 to 8 where the negative values represent the score which the opponent team gets.

Learning Method

We use different models for different situations with respect to how many shots remain. The model for the situation where x shots remain is called NN_x. We train the NN_x in order of $x = 1, x = 2, ..., x = 15$: the closer to the end the situation is, the earlier the NN_x for the situation is trained. First, we train the neural network NN_1 that predicts the expected scores distribution when 15 stones have already delivered and only one shot remains. The score of *end* is determined soon after the last shot is delivered from such states, therefore NN_1 can learn the expected scores distribution more precisely than other NN_x.

Generating states as inputs The states used for training of the models are created randomly. We aim to create states that are more likely to occur in real games, and we control the variation of the number of stones in the play area. In particular, we do not create states that have more than 4 stones from either team because it rarely happens. Because the states in which the difference in the number of stones between teams is large are less likely to happen, we control the state creation in the way that the number of created states is proportional to the difference in the number of stones between teams in the states. Especially, the number of data for the states where there was no stone for both teams is very small as all such states are the same. The symmetric states of the randomly created states as described above are added to the data set.The states for NN_x where x is more than 1 are also generated as above. The number of states for NN_1 is $200,000$ and the number of states for NN_x where $x > 1$ is $20,000$ for each in total.

In generating states, we put a stone in house or out of house in the play area with a probability of 60% and 40% respectively. In the "in house" case, both x coordinate and y coordinate of the stone follow the normal distribution with the mean of the center of house and the standard deviation is the half of house radius. In the "out of house" case, y coordinate of the stone follows the uniform distribution.

Creating teaching signals First, we create the teaching signal for the state where the last one shot remains because soon after taking a shot from these states the score gets determined. We take all discrete candidate moves without random noise, then we calculate the expected scores for each candidate move by convolving with a certain probability distribution. Consequently, we can roughly estimate the best

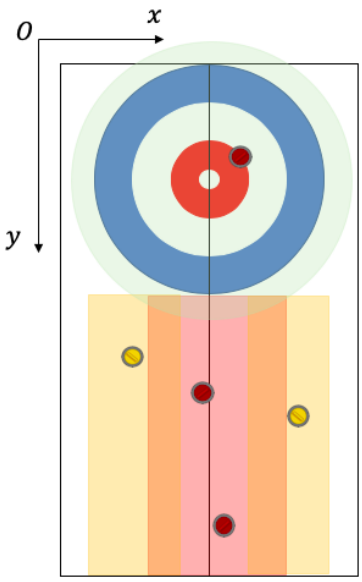

Figure 7: An example of a state generated using random numbers as following. When placing a stone in house, it is included in the green area. When placing a stone outside house if the stone belongs to the first player, x coordinate follows the normal distribution with the mean of $\frac{W}{2}$ (W is the width of the curling sheet) and the standard deviation is $\frac{W}{4}$ (represented by the red area). If the stone belongs to the second player, x coordinate follows the normal distribution with the mean of $\frac{W}{4}$ or $\frac{3W}{4}$ (chosen with a probability 50% respectively) and the standard deviation is $\frac{W}{16}$ (represented by the yellow area).

move to maximize the expected score. Then, we calculate the expected scores distribution by repeating the estimated best move 500 times. Second, as for states where more than one shot remains, the expected scores distribution as a teaching signal can be calculated by searching to the end of *end*. However, it costs a lot of time to search. To reduce the computational costs, we derive the teaching signal for NN_x by using the output of NN_{x-1}. We create states where $x-1$ shots remain by taking all discrete candidate moves without random noise from states where x shots remain. After getting the expected scores distribution as the output of NN_{x-1} for each candidate move, we again convolve the expected scores distribution by NN_{x-1} with a probability distribution over all candidate moves, and we obtain the expected scores distribution for each move as candidates of the teaching signal for NN_x. From these candidates expected scores distribution, we actually selected the distribution with the maximum expected scores as the teaching signal for NN_x.

Result

We conduct the experiment for NN_1 which is the model for the states where the last one shot remaining. The learning curve is shown in Fig. 8. The training and test error decreased, however, there is a gap between them. Actually, we

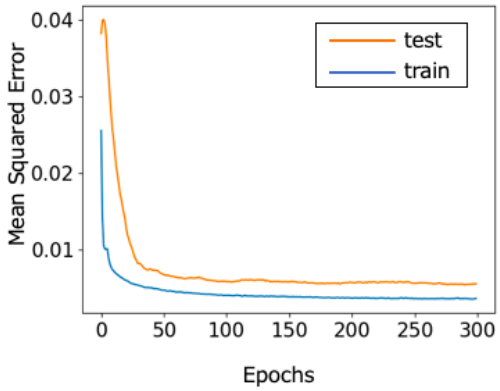

Figure 8: The learning curve: the mean squared error is used for the objective function. The orange curve is for test data and the blue curve is for train data.

found that there are states for which our model's prediction is successful and unsuccessful in test data. Below, we show the examples for these states. In these examples, the red team is the first player and the yellow team is the second player. Therefore, the last stone is delivered by the yellow team.

Figure 9 shows the example state which the model successfully learned. In the situation shown in Fig. 9, the best shot is such that the delivered stone hits and removes the red stone in the center. Since no stone collides with the red stone after the collision, the result is likely to be simple, and our model successfully predicts the expected scores distribution (Fig. 9 right). In such situations where the number of collisions is small, the results are easy to predict because there are few variations in the expected scores distribution as a teaching signal.

Next, the example of the state learned unsuccessfully is shown in Fig. 10. In the state, the predicted distribution by our model is quite different from the distribution of the teaching signal (Fig. 10 right). In this state, the best shot is such that the delivered stone hits and removes two red stones on the left side of the centerline in house at the same time. By taking the best shot, the delivered stone collides more than once. In such a situation, the variation of the result after the collision is diverse because the small difference in the noise added to the initial velocity of the stone greatly changes the collision result. Thus the situation after the collisions is likely to be complicated and the prediction is difficult. These results show that there is room for improvement in our model, especially for the situation where many collisions should be considered.

Conclusion

To analyze curling strategies in digital curling, we created the neural network model to learn the expected scores distribution used for the evaluation function in the game tree search.

In future works, based on the results of this study, we will consider other features as input features to correctly learn aspects that we found difficult to learn. As other ways for im-

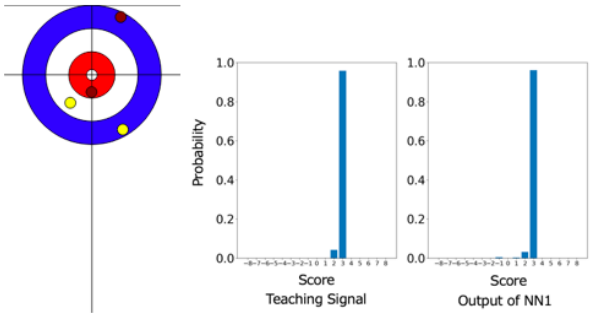

Figure 9: An example of a state successfully learned and the teaching signal and output of NN_1 successfully learned

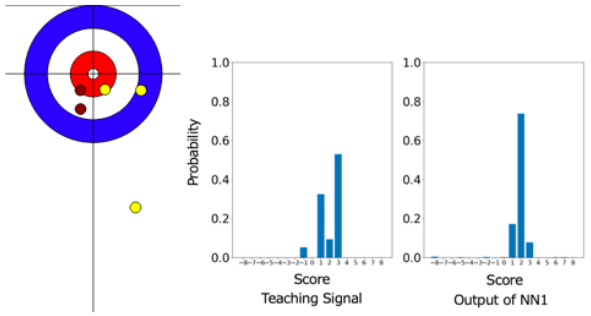

Figure 10: An example of a state unsuccessfully learned and the teaching signal and output of NN_1 unsuccessfully learned

proving the model, we will increase the number of data and change the structure by increasing or decreasing the number of hidden layers in a neural network. After improve the accuracy of NN_1, we will make $\mathrm{NN}_x(x > 1)$.

We will also consider applying our model to analyze strategies in real curing. There have been some team sports analyses using machine learning. For example, Liu and Schulte proposed the original metric called GIM for evaluating the ice hockey players' performance. Their method is based on the learned action values in reinforcement learning and realizes evaluation depending on the game context (Liu and Schulte 2018). On the other hand, in our study, we create the evaluation function for the game tree search. This evaluation function consists of the expected scores distribution and the game equity table. By using the evaluation value calculated as the expected winning rate for the situation, it is possible to evaluate whether the shot aimed by the player was good or bad. In addition, to analyze real curing strategies, it is worth considering to evaluate with consideration of the accuracy of the player's shot.

References

Archibald, C.; Altman, A.; and Shoham, Y. 2009. Analysis of a winning computational billiards player.

Ballard, B. W. 1983. The*-minimax search procedure for trees containing chance nodes.

Hauk, T. G. 2004. Search in trees with chance nodes.

Ito, T., and Kitasei, Y. 2015. Proposal and implementation of" digital curling".

Liu, G., and Schulte, O. 2018. Deep reinforcement learning in ice hockey for context-aware player evaluation.

Ohto, K., and Tanaka, T. 2017. A curling agent based on the monte-carlo tree search considering the similarity of the best action among similar states.

Smith, M. 2007. Pickpocket: A computer billiards shark.

Yamamoto, M.; Kato, S.; and Iizuka, H. 2015. Digital curling strategy based on game tree search.

Yamamoto, M.; Kato, S.; and Iizuka, H. 2018. Learning of expected scores distribution for positions of digital curling.