Building a Computer Mahjong Player Based on Monte Carlo Simulation and Opponent Models

Naoki Mizukami[1] and Yoshimasa Tsuruoka[1]

[1] The University of Tokyo

Introduction

- Imperfect information games are challenging research
  - Contract bridge [Ginsberg 2001]
  - Skat [Buro et al 2009]
  - Texas Hold'em [Bowling et al 2015]

- We focus on Japanese Mahjong
  - Multiplayer
  - Imperfect information
  - Enormous number of information sets
    - Mahjong : 10^{60}
    - Texas Hold'em: 10^{18}

Related work

- Computer poker
  - Nash equilibrium strategy
    - CFR+ method has solved Heads-up limit hold’ em poker [Bowling et al 2015]
  - Opponent modeling
    - Opponent modeling and Monte Carlo tree search for exploitation [Van der Kleij 2010]
    - The program updates a hand rank distribution in the current game state when the showdown occurs [Aaron 2002]

Japanese Mahjong

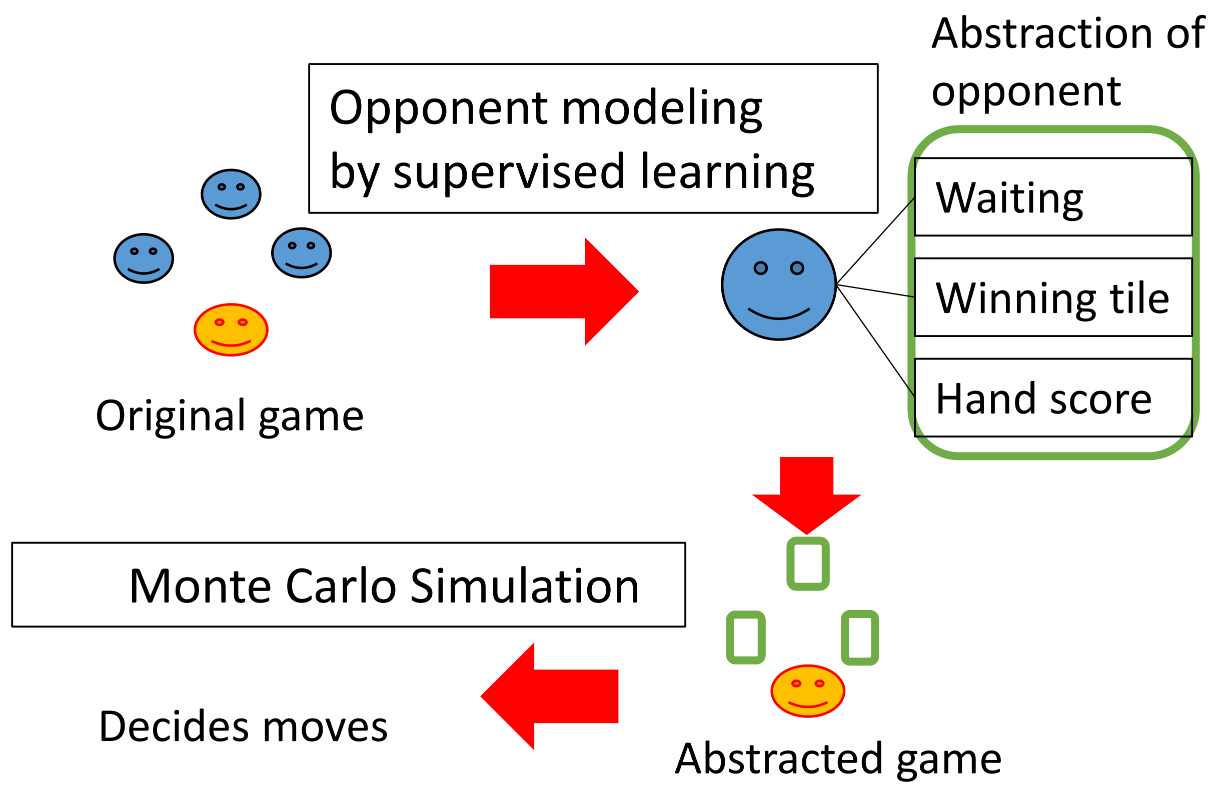
- Rules
  - It play with four players
  - A player can win round by completing a winning hand consisting of 13 *tiles*
  - One game of mahjong consists of 4 or 8 rounds
- Terms
  - *Waiting*
    - A player's hand needs only one tile to win
  - *Folding*
    - A player gives up to win and only tries to avoid discarding a winning tile for opponents
    - Is not action but strategy

One-player mahjong [Mizukami et al 2014]

- Implement folding system
- One-player Mahjong
  - A One-player Mahjong player only tries to win
  - It is trained by supervised learning using game records
  - It plays an important role in our Monte Carlo simulation
- Recognizing Folding situations
  - Folding system is realized by supervised learning
  - Positions in game records are annotated manually
- Result: Beyond average human players
- Problem: It is difficult to annotate required data

Proposed method

- Overview



- Advantage
  - It is not necessary to predict opponents' specific hands
  - Can be trained models only using game records

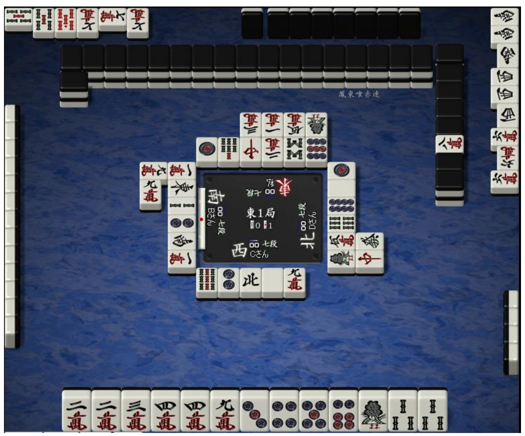
Training setting

- Game records
  - Internet Mahjong site called ``Tenhou''
- Dataset
  - Training data 1.7×10^7
  - Test data 100
- Models
  - Waiting: logistic regression model
  - Winning tile: logistic regression model
  - Hand score: Linear regression model

Waiting

- The model predicts whether an opponent is waiting or not

Input



Discarded tiles

Opponent's hand

revealed melds

Label: waiting

Output $P(opponent = waiting) = 0.8$

Evaluation and result

- Evaluation
  - Area Under the Curve

Player	AUC
Expert player	0.778
Prediction model	0.777
-Discarded tiles	0.772
-Number of revealed melds	0.770

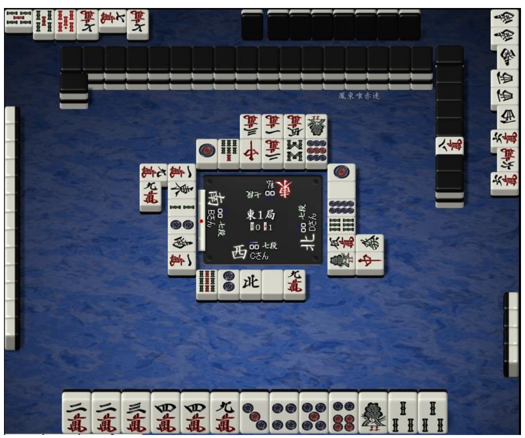
- Same prediction ability as the expert player
- Expert player: Top 0.1% of the players

Winning tiles

- Model predicts opponents’ winning tiles
- In general, there are one or more winning tiles

  → Build prediction models for all kinds of tiles

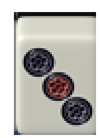
Input



Discarded tiles

Opponent’s hand

revealed melds

Winning tile  or 

Output  : 0.0  0.10  0.15

Evaluation method

1: Input opponents' information e.g winning tiles 

2: Tiles that a player has are arranged in ascending order of probability of being a winning tile for opponent

Ranking about winning tiles for opponent

Evaluation value = 6 / (14-2)=0.5

Result

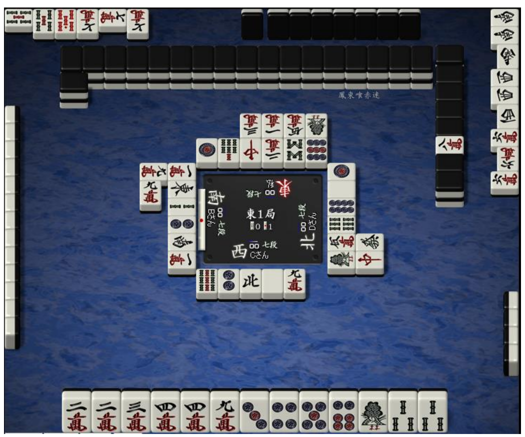
- *Random*: Tiles are arranged randomly

Player	Evaluation value
Expert player	0.744
Prediction model	0.676
-Revealed melds	0.675
-Discarded tiles	0.673
Random	0.502

Hand Score (HS)

- The model predicts the score that the player has to pay

Input



Discarded tiles

Opponent's hand

revealed melds

Hand Score 2,600

Output 2,000

Evaluation method and result

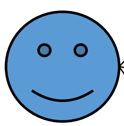
- Evaluation method
  - Mean Squared Error (MSE)

Player	MSE
Prediction model	0.37
-Revealed Melds	0.38
-Revealed fan value	0.38
Expert player	0.40

Performance of prediction model is higher than that of an expert player

Overview of proposed method

Abstraction of opponent



Waiting

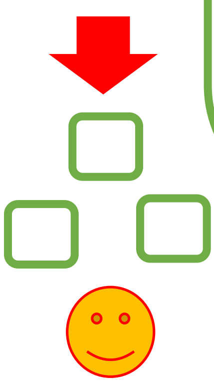
Winning tile

Hand score

$$P(p = waiting) = \frac{1}{1 + \exp(-\boldsymbol{w}^T \boldsymbol{x}_p)}$$

$$P(Tile = winning) = \frac{1}{1 + \exp(-\boldsymbol{w}^T \boldsymbol{x}_p)}$$

$$HS = \boldsymbol{w}^T \boldsymbol{x}$$



Abstracted game

Application of opponent models

- Using three prediction models to estimate an expected value

- *LP* (*Losing probability*)

$$\mathrm{L}P(p, Tile) = P(p = waiting) \times P(Tile = winning)$$

- *EL* (*Expected Loss*)

$$\mathrm{E}L(p, Tile) = LP(p, Tile) \times HS(p, Tile)$$

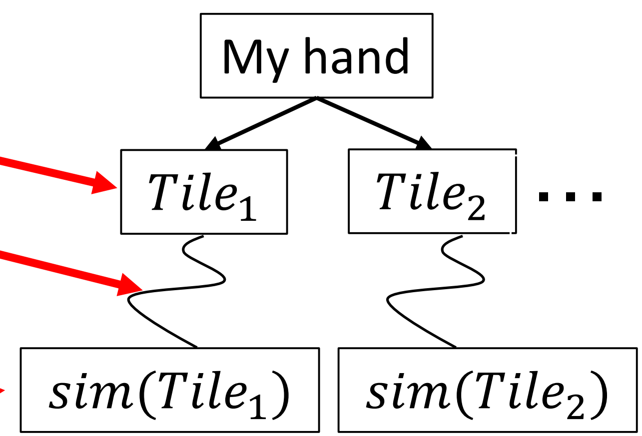
Monte Carlo simulation

- The program calculates Score$(Tile)$ for each tile
  - Program selects the tile that has the highest Score$(Tile)$

$$\text{Score}(Tile) = sim(Tile) \times \prod_{p \in opponents} \Big(1 - \big(LP(p, Tile)\big)\Big) - \sum_{p \in opponents} EL(p, Tile)$$

- Procedure of $sim(Tile)$

  1: Discard a tile
  2: Opponent's turn
  3: Program's turn
  4: Repeat 2,3
  5: Get reward



Evaluation setting

- Compared to our previous work
  - Moves are computed in a second
  - Length of a game is four rounds
- VS state-of-the-art program
  - Mattari Mahjong
  - Duplicate mode
    - can generate same tile sequences
    - can compare the result
- VS human players
  - Internet Mahjong site ``Tenhou’’

Result

- VS Mattari Mahjong

	1st (%)	2nd(%)	3rd(%)	4th(%)	Average rank	Games
Proposed method	25.2	25.6	24.7	24.5	2.48$\pm$0.07	1000
Mattari Mahjong	24.8	24.7	25.0	25.5	2.51$\pm$0.07	1000
[Mizukami+ 2014]	24.3	22.6	22.2	30.9	2.59$\pm$0.07	1000

- VS Human players

	1st (%)	2nd(%)	3rd(%)	4th(%)	Average rank	games
Proposed method	24.1	28.1	24.8	23.0	2.46$\pm$0.04	2634
[Mizukami + 2014]	25.3	24.8	25.1	24.8	2.49$\pm$0.07	1441

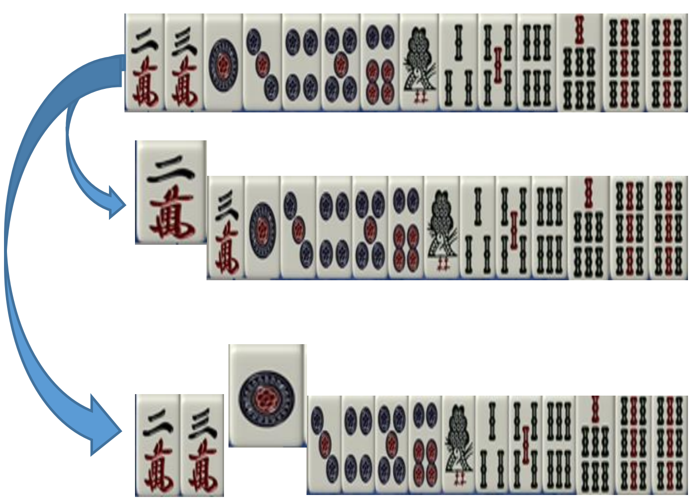
Conclusion and Future work

- Conclusion
  - Performance of the three prediction models is high
  - Our program outperforms state-of-the-art program by Monte Carlo simulation

- Future work
  - Consider final rank
  - Improve players' actions in simulation

Training of 1-player mahjong players

- A weight vector is updated so that the player can make moves as expert players.
- We used the averaged perceptron



Evaluation value

3 1

-2 -1

Record of a game's move 

Update weight vector

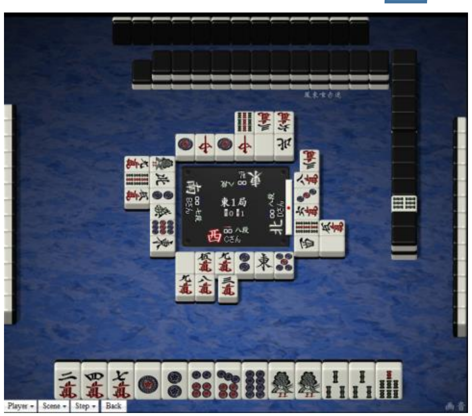
$W' = W + X$ $- X$

$\mathrm{x}: feaure\ vector\ W: weight\ vector$

Recognizing folding situations

- We train a classifier for folding situations using a machine learning approach
- This approach requires training data.

➔ Positions in game records are annotated manually



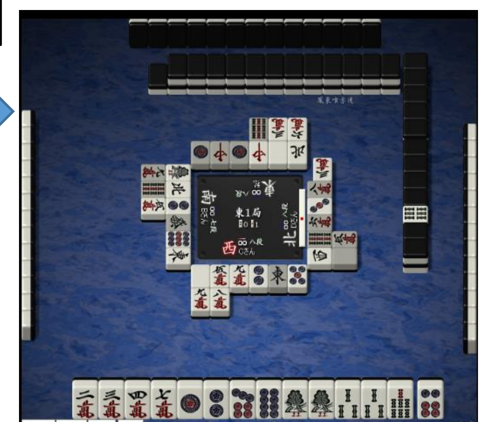
Human players

The player folded. Because

were discarded

Before discarding

Folding situations



Setting

- Dataset
  - Training data 1.77×10^7
  - Test data 100
- Features
  - Discarded tiles, number of revealed melds, and so on
  - 6,888 dimension
- logistic regression model

$$P(p = waiting) = \frac{1}{1 + \exp(-\boldsymbol{w}^T \boldsymbol{x}_p)}$$

Setting

- Dataset
  - Training data 1.77×10^7
  - Test data 100
- Features
  - Discarded tiles, number of revealed melds, and so on
  - 31,416 dimension
- logistic regression model

$$P(Tile = winning) = \frac{1}{1 + \exp(-\boldsymbol{w}^T \boldsymbol{x}_p)}$$

Setting

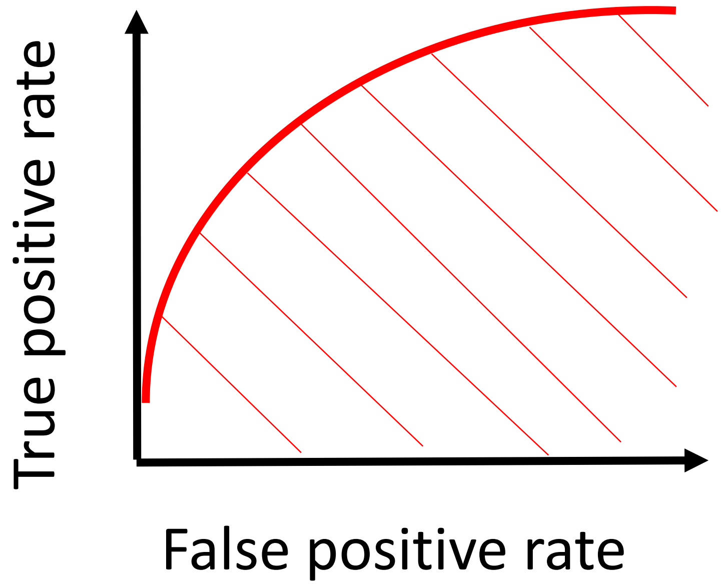
- Dataset
  - Training data 5.92×10^7
  - Test data 100
- Features
  - Revealed Melds, Revealed fan value and so on
  - 26,889 dimension
- Linear regression model

$$HS = \boldsymbol{w}^T \boldsymbol{x}$$

Evaluation and result

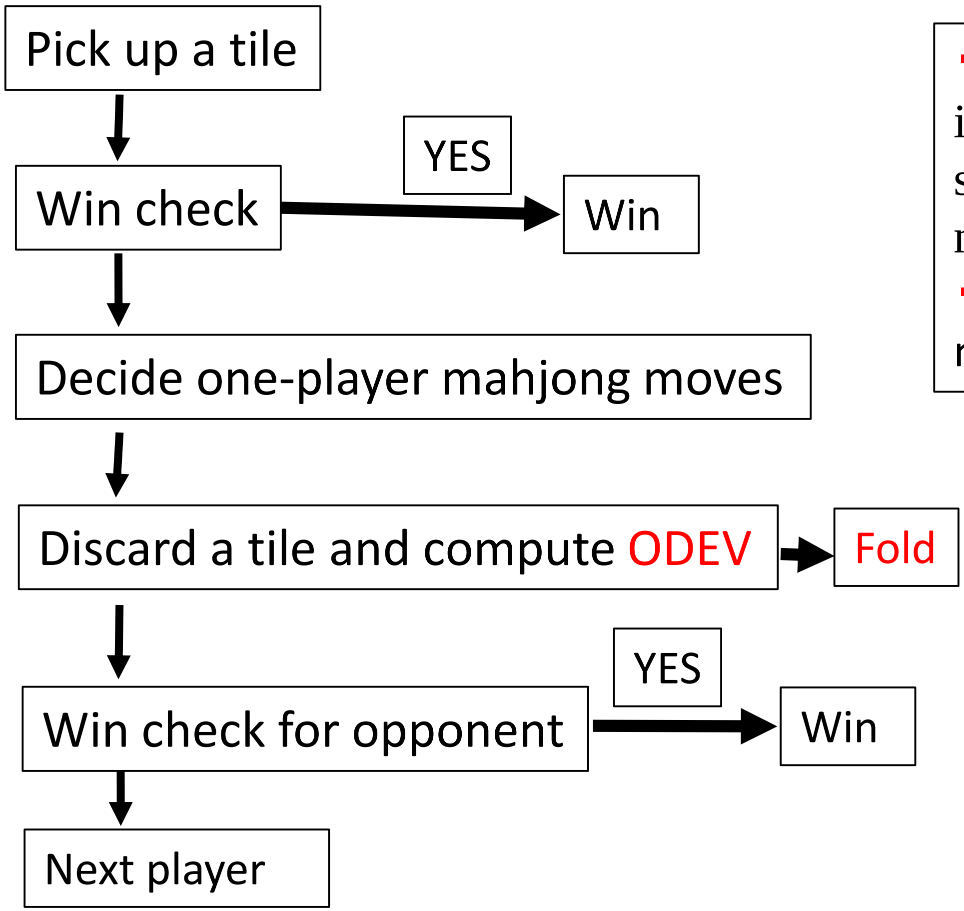
- Evaluation
  - Area Under the Curve

Player	AUC
Expert player	0.778
Prediction model	0.777
-Discarded tiles	0.772
-Number of revealed melds	0.770



- Same prediction ability as the expert player
- Expert player: Top 0.1% of the players

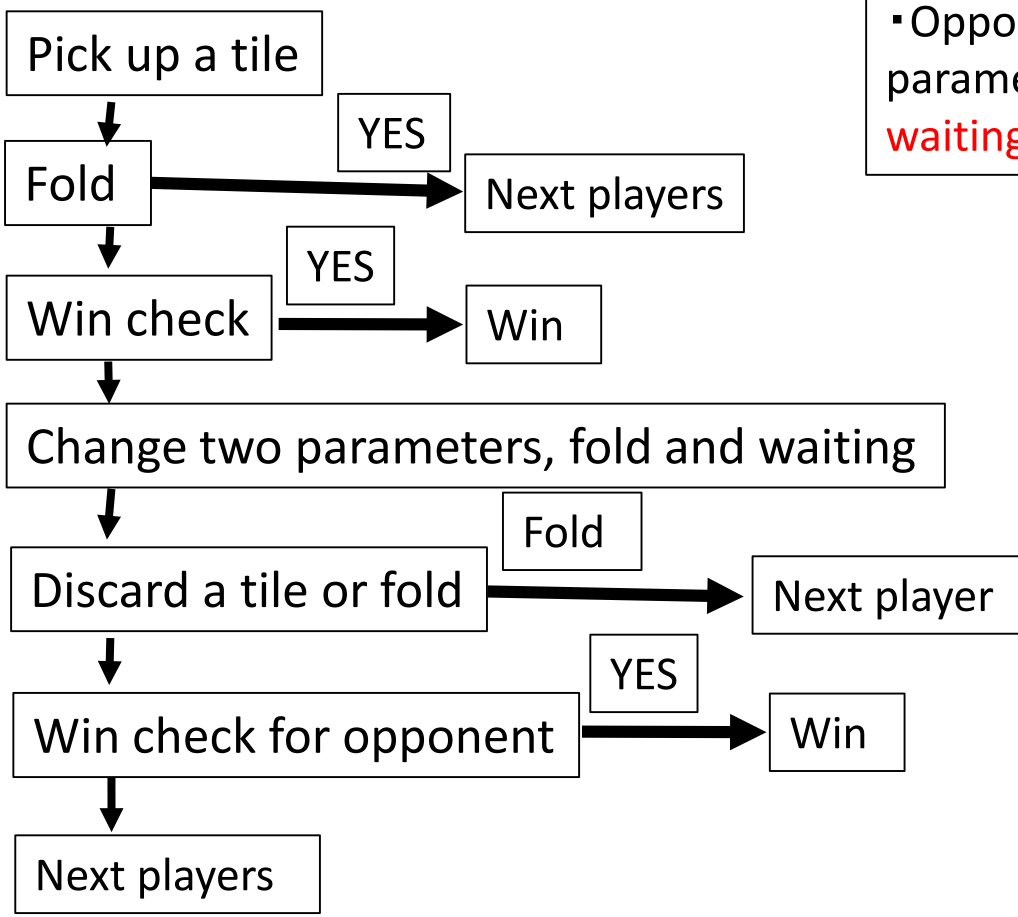
Flowchart of program’s turn



▪*ODEV* (*One-Depth Expected Value*)
is an expected value that is calculated by searching game trees until the program’s next turn.

▪Fold: a player picks up a tile and discards no tiles

Flowchart of opponent’s turn



・Opponent player has two binary parameters indicating whether he is waiting or folding