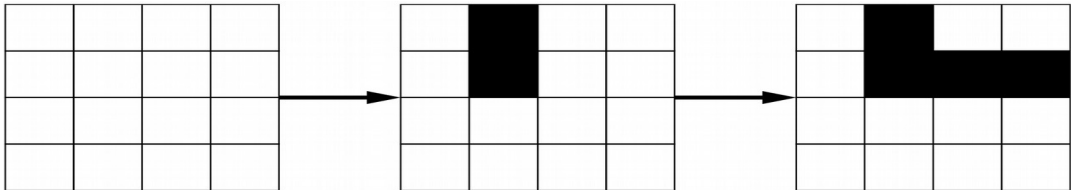
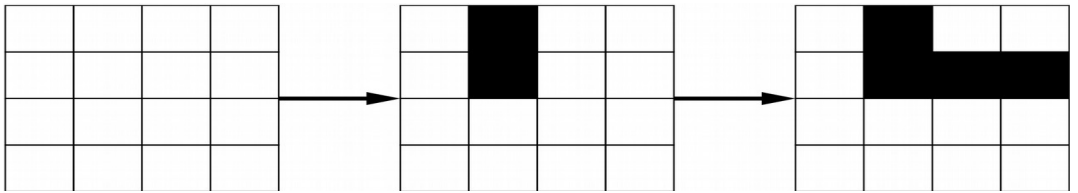
Domineering

- Put dominoes on a board :



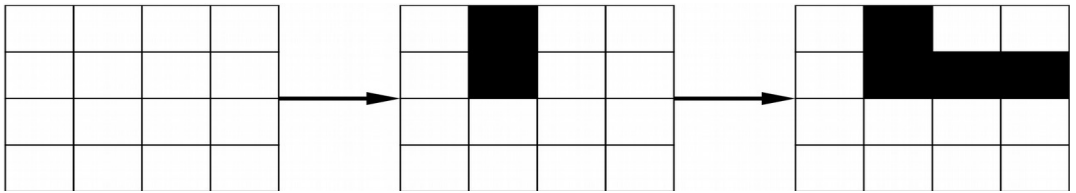
- The first player that cannot move has lost.
- http://www.kongregate.com/games/ginoboby/domineering

Domineering



- Represent a board and a move for 8x8 Domineering.
- Write a function for listing the possible moves.
- Playout : play random moves until the end of the game.

Domineering



- For each possible move play 100 playouts and calculate the average of the playouts.
- Recall the move that has the best average and play it.

Practical Work

- Generate games using a Monte Carlo evaluation to choose moves at Domineering.

- Instead of a flat Monte Carlo, use UCB to select moves to sample.

Practical Work

- Generate csv files for the Domineering positions
- csv line for a 2x2 board :

0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0

board, flipped board, player plane, move to learn

- The goal is to learn a policy network.

- For each position recall the input (i.e. the board, the flipped board, and the turn) and the output of the same size as the board with only the best move found by the Monte Carlo evaluation marked as 1.

csv files

csv line for a 2x2 board :

0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0

Corresponding input tensor :

0 0 1 1 1 1

0 0 1 1 1 1

Corresponding output tensor :

1 0

0 0

Load Data

- www.lamsade.dauphine.fr/~cazenave/domineering.csv.zip
- load csv file into numpy arrays.
- Data augmentation : generate the positions for the four symmetries of a board.
- Shuffle the data so that similar positions are not next to each other.

Policy Network

- Goal: Learn the Domineering Monte Carlo evaluation function.
- Declare a fully convolutional network
- 64 planes
- 5 hidden layers
- Softmax after the last layer

Policy Network

- Use the last examples as a test set.
- Print the percentage of test examples with the correct answer.
- Print the average test and training errors.

Policy Network

- Compare different network architectures.
- Make the network play games.