Strategy Acquisition for the Game Othello Based on Reinforcement Learning

Taku Yoshioka, Shin Ishii and Minoru Ito
IEICE Transactions on Information and Systems, 1999

Speaker : Sameer Agarwal
Course : Learning Algorithms Seminar
Date : May 7, 2001

1

Task

To produce a computer program that plays the game of Othello

Why play games ?

- Rich problem domain
- There is money to be made
- Pit man against machine
- It is fun

2

Classic Computer Game Playing

Basic Assumption

Rational opponent who will try to beat you

Classical Approach

- MinMax Search – extremely expensive O(b^d)
- Some improvement using alpha-beta pruning O($b^{d/2}$)
- Full MinMax even with alpha-beta is not feasible
- Use a **payoff function**

Problem

The payoff function is only a **heuristic**.
How to construct a good **payoff function** ?

Good Payoff Function = Good Player

3

Payoff Function Construction

- First attempts at game playing used hand crafted payoff functions
- Typically linear

Research on Machine-learned payoff functions

- Arthur Samuel build a checkers program which learned its own payoff function.(1954)
- Gerry Tesauro built TD-Gammon based on TD-Reinforcement learning. Top ranked player in the world today. (1995)

Is it possible to construct an algorithm that can learn a good payoff function for Othello?

4

Rules of Othello

- 2 player game
- Each player starts with 2 discs on the board and 30 discs in reserve
- Player with more disks on the board at the end of the game wins

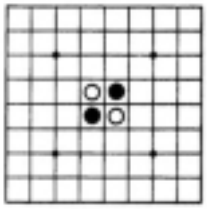


FIGURE 1

Start position

5

Making a Move

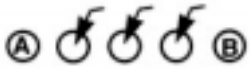
A move consists of "outflanking" your opponent's disc(s), then flipping the outflanked disc(s) to your color.

To **outflank** means to place a disc on the board so that your opponent's row (or rows) of disc(s) is bordered at each end by a disc of your color. (A "row" may be made up of one or more discs.)

Here's one example:

White disc **A** was already in place on the board.
The placement of white disc **B** outflanks the row of three black discs.

Before : 

After : 

6

Another Example

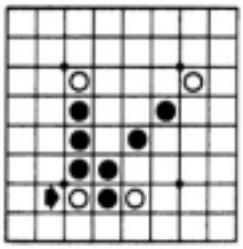


FIGURE 2

Before

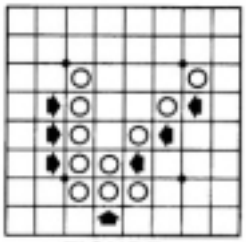


FIGURE 3

After

7

Contributions of this Paper

- Radial Basis Function (called NGnets) based payoff function.
- MinMax Reinforcement Learning.
- Demonstration that it is possible to learn a payoff function autonomously using MinMax Learning.
- A comparison with an evaluation function based on a Multilayer Perceptron (MLP)
- A comparison with an evaluation function based on TD(0) Reinforcement Learning.

8

Organization of the rest of the talk

- Algorithm Design
  - Strategy – How do we search for a good move ?
  - Payoff Function – How do we know what is a good move ?
  - Learning Method – How to construct a good payoff function ?
- Experimental Setup
- Experiment Results
- Conclusion

9

Strategy

Optimal Evaluation Function

$$V_b^* = \begin{cases} B(s) - W(s) & (\text{if } s \text{ is the final state}) \\ \max_b \{\min_w \{V_b^*(S(S(s,b),a))\}\} & (\text{otherwise}) \end{cases}$$

Optimal Action

$$b^* = \arg\max_b \{\min_w \{V_b^*(S(S(s,b)w))\}\}$$

10

Back in the real world (strategy)

Since we **do not** know the **optimal** V^* we make do with an estimate

$$\hat{b} = \arg\max_b \{\min_w \{\hat{V}_b(S(S(s,b)w))\}\}$$

Here $\hat{V}_b$ is the estimated payoff function that we construct.

Since Othello is completely deterministic we include a random term to increase exploration. Hence

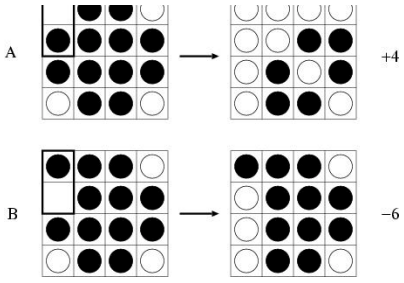
$$\hat{b} = \arg\max_b \{\min_w \{\hat{V}_b(S(S(s,b)w)) + N(0,\sigma_L)\}\}$$

11

Payoff Function

Highly nonlinear game play

A single move can change the board drastically



MLP are useful for approximating “smooth” functions like Backgammon evaluation functions. Othello requires something different..

12

Payoff Function : Algebraic Form

Normalized Gaussian Networks(NGNet)

$$V'(s) = \sum_{i=1}^{N} w_i N_i(s)$$ Weighted sum of normalized gaussians

$$N_i(s) = \frac{g_i(s)}{\sum_{j=1}^{N} g_j(s)}$$

$$g_i(s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-|s-\mu_i|^2}{2\sigma^2}\right)$$

13

Learning Method: MMRL

Goal : To minimize the prediction error over the length of the game

$$E = \frac{1}{2} \sum_{t=1}^{T} \{\hat{V}_b^*(s_t) - \hat{V}_b(s_t)\}^2$$

T is the length of the game
S_t is the state at the t-th black turn
V^* is the reinforcement signal

$$\hat{V}_b^* = \begin{cases} B(s) - W(s) & (\text{if } s \text{ is the final state}) \\ \max_b \left\{ \min_w \left\{ \hat{V}_b(S(S(s,b),a)) \right\} \right\} & (\text{otherwise}) \end{cases}$$

Perform Gradient Descent using the above error formula to get update rules for the parameters of the NGNet

14

Putting it all Together

1. Initialize the parameters of the NGNet.
2. Start a game.
3. For each step t in the game:
   a. select a move based on the MinMax decision
   b. compute the weight corrections.
4. At the end of the game update the NGNet parameters.
5. Go to step 2.

15

Experimental Setup

Heuristic Player 1 (non-adaptive)

If there are 10 or less positions left empty on the board,
then perform exhaustive search

else, choose the action a that maximizes the following function

$$V_w^1(s) = \frac{W(s') - B(s')}{W(s') + B(s')} + C(a) + N(0, \sigma_{HP1}) \quad \text{where } s' = S(s,a)$$

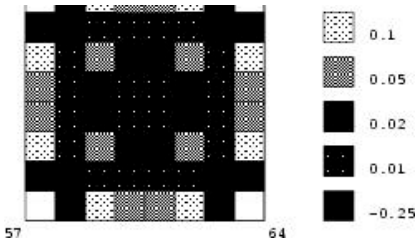
C(a) is 1 if a is a corner on the board 0 otherwise.

16

Heuristic Player 2(non-adaptive)

Chooses a move based on a 2-ply MinMax search using

$$V_w^2(s) = \sum_{i=1}^{64} f_i s_i + N(0, \sigma_{HP2})$$



The weights f_i are based on the above map

17

Parameter settings for our player

1. No. of units in the NGNet = 100
2. Initially $w_i = 0$ and μ_i is set at random.
3. Each player plays 50,000 games for training against one opponents
4. Each training epoch is 500 games, with a 100 game test epoch
5. Testing is done by playing against one of the heuristic players

18

Different Players Trained

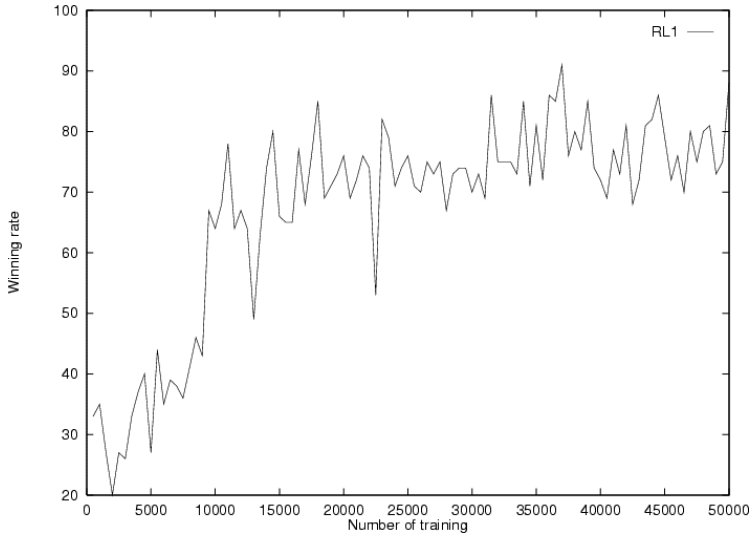
Two Learning Players - Black and white sides **employ their own** NGNets
Train by playing against each other (RL1(B),RL1(W)

Training with HP1 - A black player is trained by playing **against a white** HP1 (RL2(B))

Self-Play - Black and white sides **share the same NGNet** SL(W), SL(B)

19

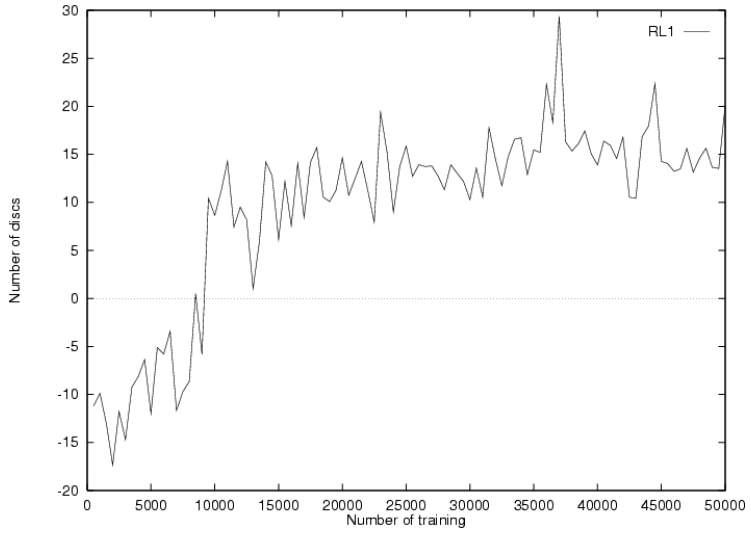
Experimental Results



Learning Behavior [RL1(B) / HP1(W)]

20

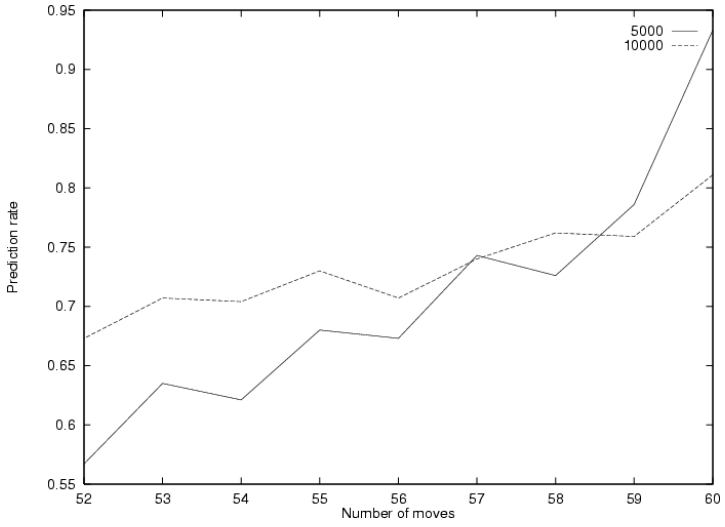
Experimental Results



Average number of discs won [RL1(B)]

21

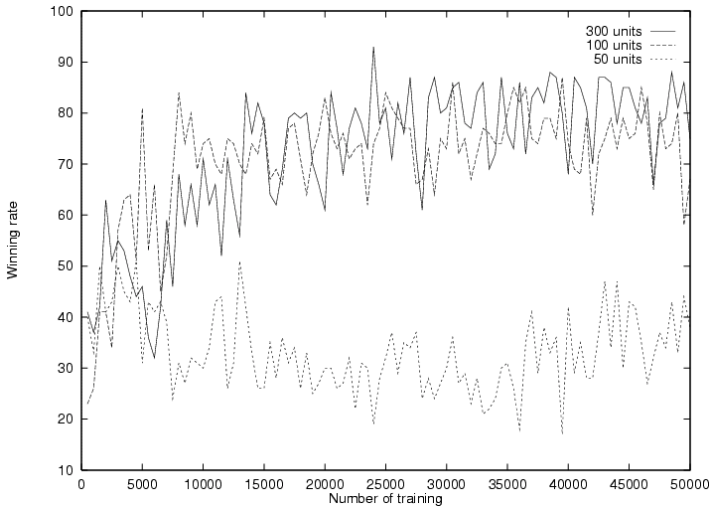
Experimental Results



Predictive Behavior RL1(B)

22

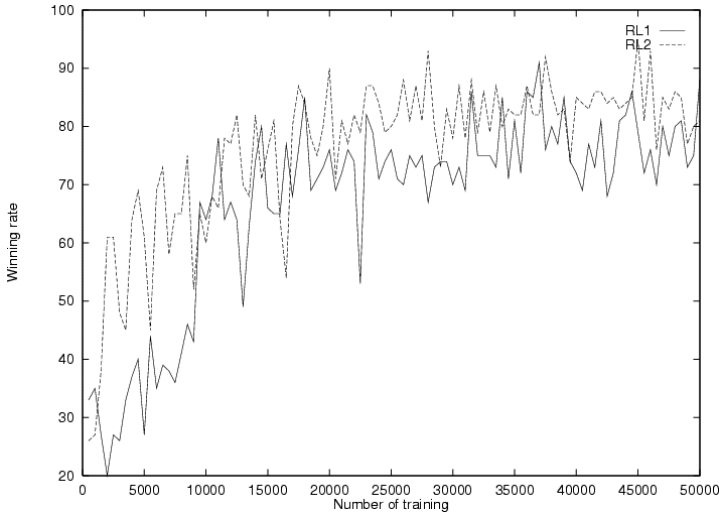
Experimental Results



Varying Number of Centers [RL1(B) / HP1(W)]

23

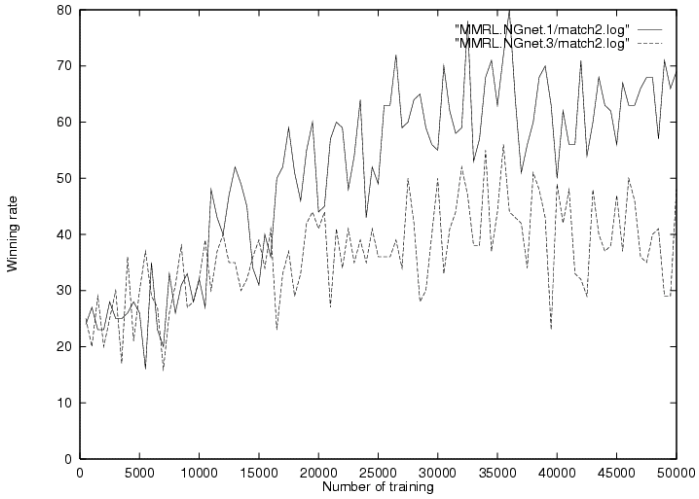
Experimental Results



Winning Rate
RL1(W) and RL2(W) against HP1(B)

24

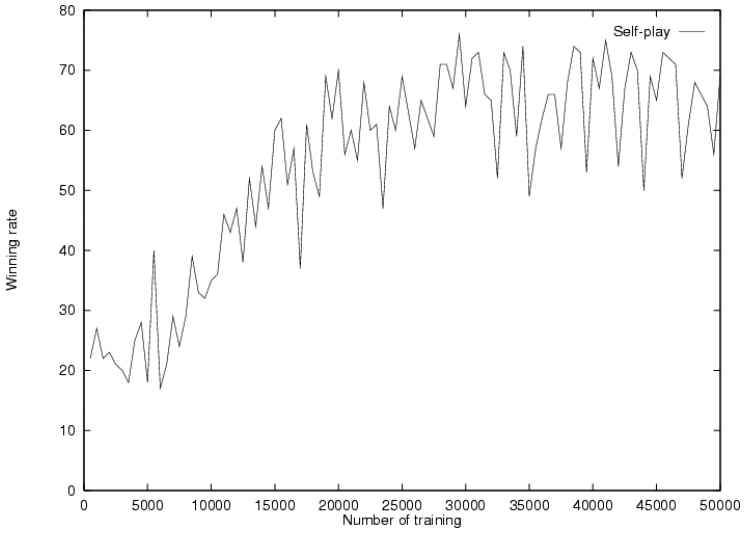
Experimental Results



Winning Rate
RL1(W) and RL2(W) against HP2(B)

25

Experimental Results



Winning Rate [SL(B) / HP(W)]

26

Experimental Results

MinMax Learning tries to predict the end result of the game
TD Learning tries to predict the board state on the immediate future

$$z - P_t = \sum_{k=t}^{m} (P_{k+1} - P_k) \quad \text{where } P_{m+1} = z$$

Revised Payoff functions

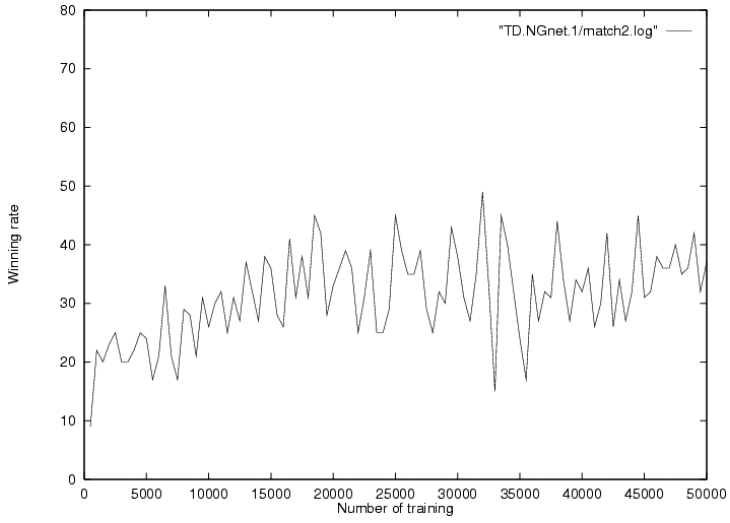
$$\hat{V}_b^* = \begin{cases} B(s) - W(s) & (\text{if s is the final state}) \\ \hat{V}_b(S(S(s,b),w)) & (\text{otherwise}) \end{cases}$$

Revised Action

$$\hat{b} = \hat{V}_b(S(S(s,b)w)) + N(0,\sigma_T)$$

27

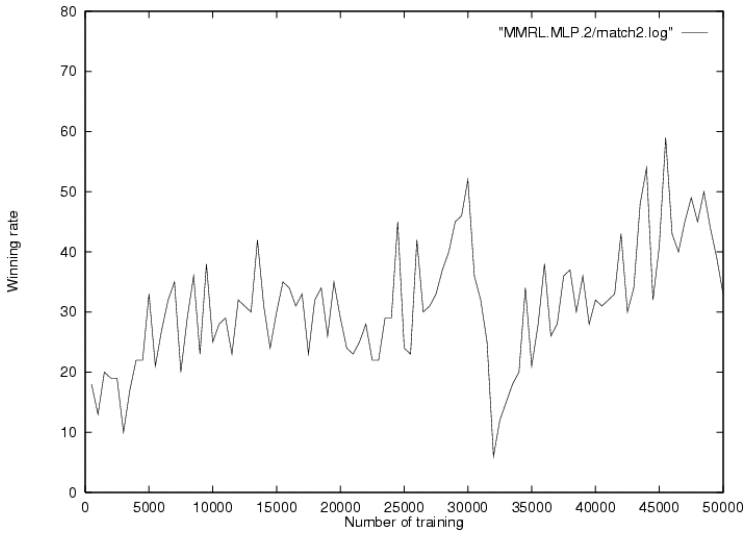
Experimental Results



Winning Rate [TD(B) / HP2(W)]

28

Experimental Results



Winning Rate
MLP(100 hidden units)(B) / HP2(B)

29

Conclusions

1. A general purpose new game playing algorithm (MinMax RL) is proposed. It is shown to be successful against heuristic players.
2. A Radial Basis Function/NGNet based payoff function is shown to be effective for evaluating Othello board configurations.
3. RBFs are shown to be better than MLP based payoff functions.
4. MinMax-RL is shown to be better than TD(0).

30