

Value Iteration and Application on Tic-Tac-Toe

Presenters: LI ZITIAN

Final Presentation of OTH5427

Aug 5th 2022



Overview

1 Solution of First 6 Questions

- Elementary Proof, Q1, Q2, Q3
- Numeric Result, Q4, Q5, Q6

2 Application on Tic-Tac-Toe

- Best First Step on 3-3-3 Game
- Best First Step on 4-4-4 Game

Questions of Course Project III

Project III Consists of 3 parts

- Q1, Q2, Q3: Preliminary result of value iteration
- Q4, Q5, Q6: Numeric experiment to validate variants of Value Iteration
- Q7: How to solve Tic-Tac-Toe

○
●○○○○○○○○○○○○○
○○○○○○○

○○○
○○○○○○○
○○○○○

Elementary Proof, Q1, Q2, Q3

Q1

$$\begin{aligned} \min & \sum_{i=1}^m \sum_{j \in \mathcal{A}_i} c_{i,j} x_{i,j} \\ \text{s.t.} & \sum_{i=1}^m \sum_{j \in \mathcal{A}_i} (e_i - \gamma p_{i,j}) x_{i,j} = e, x_{i,j} \geq 0, \forall i, j \end{aligned}$$

where

- $x_{i,j}$, $j \in \mathcal{A}_i$ is the expected present value of the number of times in which the process visits state i and take state-action $j \in \mathcal{A}_i$
- $p_{i,j}$ is the state transition probabilities from state i to all states, if action j is taken
- $c_{i,j}$ is the immediate cost when action j is taken at state i

Q1

footnotesize

$$\begin{aligned} \min \quad & \sum_{i=1}^m \sum_{j \in \mathcal{A}_i} c_{i,j} x_{i,j} \\ \text{s.t.} \quad & \sum_{i=1}^m \sum_{j \in \mathcal{A}_i} (e_i - \gamma p_{i,j}) x_{i,j} = e, x_{i,j} \geq 0, \forall i, j \end{aligned}$$

Prove following statement

- Every basic feasible solution represent a policy, the basic variables have exactly one variable from each state i
- Each basic variable value is no less than 1
- The sum of all basic variable values is $\frac{m}{1-\gamma}$

Q1-Solution

- Every basic feasible solution represent a policy, the basic variables have exactly one variable from each state i

Proof.

There are $m + \sum_{i=1}^m \#|\mathcal{A}_i|$ constraints, and $\sum_{i=1}^m \#|\mathcal{A}_i|$ decision variables

If $\exists i$, such that $x_{i,j} = 0, \forall j \in \mathbb{A}_i$, then the i^{th} entry of $\sum_{i=1}^m \sum_{j \in \mathcal{A}_i} (e_i - \gamma p_{i,j}) x_{i,j}$ would be no greater than 0, which leads to contradiction.


Thus if there are only m entries are non-zero, then for $\forall i \in [m]$, $\exists j \in \mathcal{A}_i$, such that $x_{i,j} \geq 0$, which represent a policy. □

Q1-Solution

- Each basic variable value is no less than 1

Proof.

Without loss of generality, we can focus on $i = 1$.

If $x_{1,j'} < 1$ and $x_{1,j} = 0, \forall j \in \mathcal{A}_i, j \neq j'$, then the the first entry of $\sum_{i=1}^m \sum_{j \in \mathcal{A}_i} (e_i - \gamma p_{i,j}) x_{i,j}$ would be strictly smaller than 1, which leads to contradiction 

Q1-Solution

- The sum of all basic variable values is $\frac{m}{1-\gamma}$

Proof.

Assume for index i , $x_{i,j_i} > 0, \forall j \in \mathcal{A}_i, j \neq i$

Sum up the m constraints of $\sum_{i=1}^m \sum_{j \in \mathcal{A}_i} (e_i - \gamma p_{i,j}) x_{i,j} = e$, we can derive

$$\sum_{i=1}^m x_{i,j_i} - \gamma \sum_{i=1}^m x_{i,j_i} = m$$

$$\Leftrightarrow \sum_{i=1}^m x_{i,j_i} = \frac{m}{1-\gamma}$$

Q2

Starting with any vector y^0 , we iteratively update it with following formula

$$y_i^{k+1} = \arg \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}\}$$

Prove the contraction result

$$\|y^{k+1} - y^*\|_{+\infty} \leq \gamma \|y^k - y^*\|_{+\infty}$$

Where y^* is the fixed-point or optimal value vector, that is

$$y_i^* = \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^*\}$$

Q2-Solution

Proof.

Denote $j_1 = \arg \min_j \{c_{ij} + \gamma p_{i,j}^T y^k\}$, $j_2 = \arg \min_j \{c_{ij} + \gamma p_{i,j}^T y^*\}$

On the one hand

$$\begin{aligned}
 y_i^{k+1} - y_i^* &= \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^k\} - \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^*\} \\
 &= c_{i,j_1} + \gamma p_{i,j_1}^T y^k - c_{i,j_2} - \gamma p_{i,j_2}^T y^* \\
 &\geq c_{i,j_1} + \gamma p_{i,j_1}^T y^k - c_{i,j_1} - \gamma p_{i,j_1}^T y^* \\
 &\geq \gamma p_{i,j_1}^T (y^k - y^*) \\
 &\geq -\gamma \|y^k - y^*\|_\infty
 \end{aligned}$$



Q2-Solution

Proof.

On the other hand

$$\begin{aligned}
 y_i^{k+1} - y_i^* &= \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^k\} - \min_{j \in \mathcal{A}_i} \{c_{i,j} + p_{i,j}^T y^*\} \\
 &= c_{i,j_1} + \gamma p_{i,j_1}^T y^k - c_{i,j_2} - \gamma p_{i,j_2}^T y^* \\
 &\leq c_{i,j_2} + \gamma p_{i,j_2}^T y^k - c_{i,j_2} - \gamma p_{i,j_2}^T y^* \\
 &\leq \gamma p_{i,j_2}^T (y^k - y^*) \\
 &\leq \gamma \|y^k - y^*\|_\infty
 \end{aligned}$$

which is $\forall i, -\gamma \|y^k - y^*\|_\infty \leq y_i^{k+1} - y_i^* \leq \gamma \|y^k - y^*\|_\infty$,
 further $\|y^{k+1} - y^*\|_\infty \leq \gamma \|y^k - y^*\|_\infty$



Q3

Derive the dual problem of above linear programming

$$\begin{aligned}
 & \max_y \sum_{i=1}^m y_i \\
 & s.t. y_1 - \gamma p_{1,j}^T y \leq c_{1,j}, j \in \mathcal{A}_1 \\
 & \quad y_2 - \gamma p_{2,j}^T y \leq c_{2,j}, j \in \mathcal{A}_2 \\
 & \quad \dots \\
 & \quad y_m - \gamma p_{m,j}^T y \leq c_{m,j}, j \in \mathcal{A}_m
 \end{aligned}$$

Q3

If starting with any vector $y^0 \geq y^*$ and assuming $y^1 \leq y^0$, then prove the following entry-wise monotone property:

$$y^* \leq y^{k+1} \leq y^k, \forall k$$

On the other hand, if we start from a vector such that

$$y_i^0 < \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^0\}$$

(y^0 in the interior of the feasible region), then prove the entry-wise monotone property

$$y^* \geq y^{k+1} \geq y^k, \forall k$$

Q3-Solution

Proof.

If $y^0 \geq y^*$, and assuming $y^1 \leq y^0$, then

$$y_i^1 = \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^0\} \geq \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^*\} = y^*$$

With induction, we can conclude $y^k \geq y^*$

$$\text{While } y_i^2 = \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^0\} \leq \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^0\} = y^1$$

with induction, we can conclude $y^k \geq y^{k+1}$ □

Q3-Solution

Proof.

If y^0 is in the interior of the feasible region, We take $S' = \{i : y_i^0 > y_i^*\}$, $S'' = \{i : y_i^0 \leq y_i^*\}$, if $S' \neq \emptyset$, we can take $i = \arg \min_{i' \in S'} y_{i'}^* - y_{i'}^0$. For this i , we can find a $j \in \mathcal{A}_i$, such that

$$c_{ij} = y_i^* - \sum_{s \in S'} \gamma p_{i,j,s} y_s^* - \sum_{s \in S''} \gamma p_{i,j,s} y_s^*$$

On the other hand

- $c_{ij} > y_i^0 - \sum_{s \in S'} \gamma p_{i,j,s} y_s^0 - \sum_{s \in S''} \gamma p_{i,j,s} y_s^0$
- $-\sum_{s \in S''} \gamma p_{i,j,s} (y_s^* - y_s^0) \leq 0$
- $(y_i^* - y_i^0) - \sum_{s \in S'} \gamma p_{i,j,s} (y_s^* - y_s^0) \leq (1 - \gamma)(y_i^* - y_i^0) < 0$




Q3-Solution

Proof.

Thus we can conclude

$$c_{ij} > y_i^* - \sum_{s \in S'} \gamma p_{i,j,s} y_s^* - \sum_{s \in S''} \gamma p_{i,j,s} y_s^*$$

which leads to a contradiction. Thus $y^0 \leq y^*$. The remaining part is similar with above. 

○
○○○○○○○○○○○○○○○
●○○○○○

○○○
○○○○○○○
○○○○○

Numeric Result, Q4, Q5, Q6

Q4,Q5,Q6

Instead of using $y_i^{k+1} = \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^k\}, \forall i \in [m]$

■ Q4: Random VI

In the k^{th} iteration, randomly select a subset $B^k \subset [m]$,

$$y_i^{k+1} = \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T y^k\}, \forall i \in B_k$$

■ Q5: Cyclic VI

Initialize $\tilde{y}^k = y^k$, For $i = 1$ to m

$$\tilde{y}_i^k = \min_{j \in \mathcal{A}_i} \{c_{i,j} + \gamma p_{i,j}^T \tilde{y}^k\}$$

■ Q6: permuted Cyclic VI, permute sequence $1, 2, \dots, m$ in Q5

○
○○○○○○○○○○○○○
○●○○○○○

○○○
○○○○○○○
○○○○○

Numeric Result, Q4, Q5, Q6

Q4 - Impact of size of B_k

We set $m = 10, 15, 20$, Action size $\propto m$ and randomly generate

- γ
- Cost to go
- Transition probability

Stopping criterion

- Initialize y^0 as zero vector
- When the difference between solution from linear programming and iteration is close.

Statistic

- Use 10 different random seeds to repeatedly solve same problems, and then record the mean value of required epochs

Q4 - Impact of size of B_k

Table: Required epochs of Value iteration

m	Required Epochs
10	151
15	145
20	141

Q4 - Impact of size of B_k

Table: Required epochs of Random VI

m	$B_{k-m}/2$	$B_{k-m}/3$	$B_{k-m}/4$	$B_{k-m}/5$	B_{k-1}
10	309.8	519.5	777.4	777.4	1569.5
15	324.4	453.5	768.1	768.1	2303.7
20	289.7	488.5	586.3	730.4	2926.2

Q4 - Impact of size of B_k

Table: Required epochs / $\frac{m}{\#B_k}$ of Random VI

m	$B_{k-m}/2$	$B_{k-m}/3$	$B_{k-m}/4$	$B_{k-m}/5$	B_{k-1}
10	154.9	173.17	194.35	155.48	156.95
15	162.2	151.17	192.025	153.62	153.58
20	144.85	162.83	146.575	146.08	146.31

Randomization cannot significantly help

Q5 and Q6 - RPCyclicVI, CyclicVI and VI

We set $m = 10, 15, 20, 25, 30, 25, 30$, Action size $\propto m$ and randomly generate

- γ
- Cost to go
- Transition probability

Stopping criterion

- Initialize y^0 as zero vector
- When the difference between solution from linear programming and iteration is close.

Q5 and Q6 - RPCyclicVI, CyclicVI and VI

Table: Comparison among VI, CyclicVI, RPCyclicVI

m	VI	CyclicVI	RPCyclicVI
20	142	76	87.6
40	129	68	78.0
60	124	64	74.8
80	118	61	71.1
100	118	61	71.0

Conclusion

- CyclicVI can significantly expedite the convergence
- RPCyclicVI is slightly worse than CyclicVI, but still better than VI

Definition of m-n-k Game

(Wikipedia) m,n,k-game is an abstract board game in which

- **Two players** take turns in placing a stone of their color
- **m-by-n** board
- Winner Condition: **k stones** of their own color in a row, horizontally, vertically, or diagonally

Example

- 3-3-3 game: Tic-Tac-Toe
 - 19-19-5 game: Gomoku
 - 19-19-5 game with restriction on first player: Renju
- Reference for invincible policy [1, 3]

m-n-k game is a sequential game with finite actions and status
Pure Nash Equilibrium exists

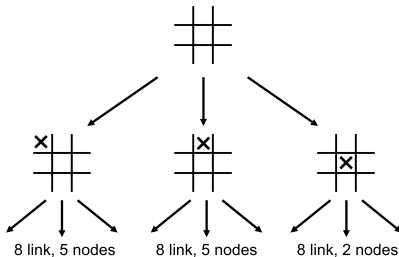
The diagram illustrates the minimax algorithm for a 3x3 tic-tac-toe game. It shows a sequence of nodes representing the game state, with arrows indicating the path from the root node to the final state.

- Root Node:** An empty 3x3 grid. Label: "X win".
- Level 1 (Child Nodes):** Three possible moves for X (top-left, top-middle, top-right). Each node is labeled "X win".
- Level 2 (Grandchild Nodes):** Six possible moves for O, branching from the Level 1 nodes. Each node is labeled "X win".
- Level 3 (Great-grandchild Nodes):** Three possible moves for X, branching from the Level 2 nodes. Each node is labeled "X win".

The path highlighted by the arrows starts from the root node, moves to the top-middle child node, then to the top-right grandchild node, and finally to the top-right great-grandchild node.

If O is a random player and consider the best policy of x player

[illegible]



Use Backward Induction to find the best policy

Several tricks to accelerate the process

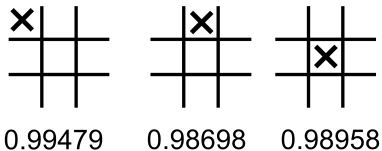
- Adopt Breadth-First-Search to avoid missing and replicate status
 - There are only 765 possible status, not a big number
- Eliminate status that can be transformed to another existing status with combination of rotation, transpose, flip

We adopt following setting

- O player is a random player. He will uniformly randomly drop piece on vacant position
- 1 denotes X win, 0 denotes tie, -1 denotes O win
- We aim to maximize reward

Use Backward Induction to find the best policy

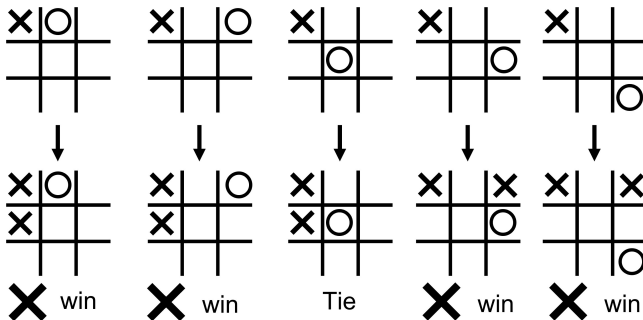
Figure: Reward-of-Different-Positions



Then the best first step would be corner

Use Backward Induction to find the best policy

Figure: Possible Policy



I guess we are all quite familiar with such result

Use Value Iteration to find the best policy

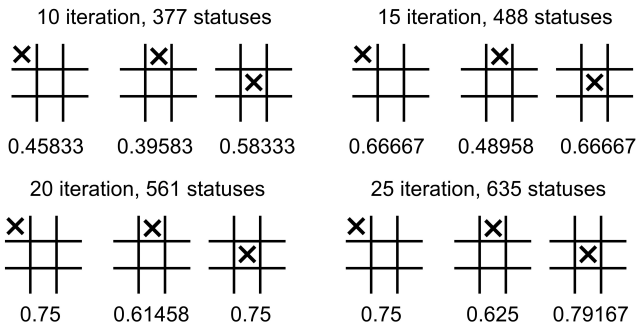
In fact we can have other faster methods to derive best first step, for example

■ Random Value Iteration

- Each time, we randomly extend at most 20 existing statuses for one step
- We immediately update all the reward value after adding more status
- When update reward, we always begin with the "bottom nodes"

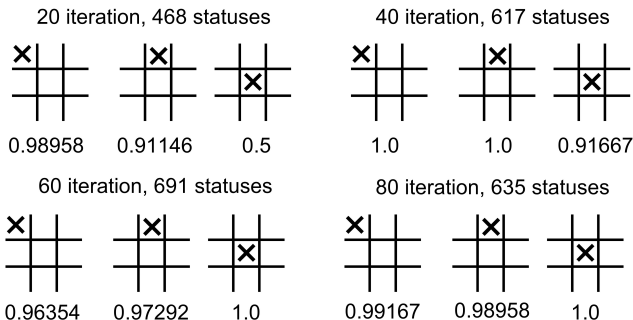
Use Value Iteration to find the best policy

Figure: Random Value Iteration



Use Value Iteration to find the best policy

Figure: Random Value Iteration - DFS



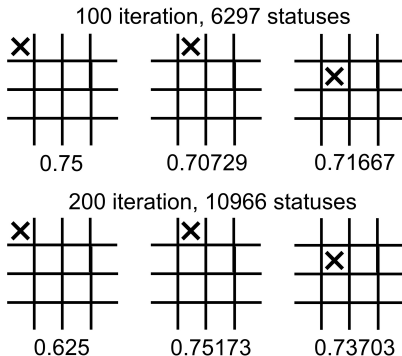
4-4-4 Game

Difference between 4-4-4 game and 3-3-3 game

- The amount of statuses in 4-4-4 game is much larger, quite hard to build the full acyclic directed graph
- "Tie" is common, hard to converge

Use Value Iteration to find the best policy

Figure: Random Value Iteration - DFS



Summary

In fact, Tic-Tac-Toe, Chess, and Go have some properties

- We cannot assure our choice is the best, unless we traverse all the possible statues
- In most of the cases, cost-to-go is zero, customized reward/punishment would be a challenging work
- Suffering from **curse of dimension**
- We don't care about the exact present value, we just want to **identify the best position/policy**

That implies value iteration might not be the best choice.
 (Monte Carlo Tree Search supports the third choice.)

Further Extension

- Other methods to set up reward or punishment
 - Prior knowledge
 - Deep Neural Network, [2]
- Monte Carlo Tree Search method to identify the best arm, instead of estimating
- Several tricks to accelerate
 - Zobrist Hash
 - Doubly linked list

End

Thanks for watching!

Reference

- [1] Victor Allis. “Searching for solutions in games and artificial intelligence”. ISBN: 9789090074887 OCLC: 905509528. PhD thesis. S.l.: s.n.], 1994.
- [2] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (Jan. 2016). Number: 7587 Publisher: Nature Publishing Group, pp. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961. URL: <https://www.nature.com/articles/nature16961> (visited on 08/04/2022).
- [3] János Wágner and István Virág. “Solving Renju”. In: *ICGA Journal* 24 (Mar. 1, 2001), pp. 30–35. DOI: 10.3233/ICG-2001-24104.