Minimax

This article is about the decision theory concept. For other uses, see Minimax (disambiguation).

Minimax (sometimes **MinMax** or **MM**[1]) is a decision rule used in decision theory, game theory, statistics and philosophy for *mini*mizing the possible loss for a worst case (*max*imum loss) scenario. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision-making in the presence of uncertainty.

1 Game theory

1.1 In general games

The **maximin value** of a player is the largest value that the player can be sure to get without knowing the actions of the other players. Its formal definition is:[2]

$$\underline{v_i} = \max_{a_i} \min_{a_{-i}} v_i(a_i, a_{-i})$$

Where:

- i is the index of the player of interest.
- $-i$ denotes all other players except player i .
- a_i is the action taken by player i .
- a_{-i} denotes the actions taken by all other players.
- v_i is the value function of player i .

Calculating the maximin value of a player is done in a worst-case approach: for each possible action of the player, we check all possible actions of the other players and determine the worst possible combination of actions - the one that gives player i the smallest value. Then, we determine which action player i can take in order to make sure that this smallest value is the largest possible.

For example, consider the following game for two players, where the first player ("row player") may choose any of three moves, labelled T, M, or B, and the second player ("column" player) may choose either of two moves, L or R. The result of the combination of both moves is expressed in a payoff table:

(where the first number in each cell is the pay-out of the row player and the second number is the pay-out of the column player).

For the sake of example, we consider only pure strategies. Check each player in turn:

- The row player can play T, which guarantees him a payoff of at least 2 (playing B is risky since it can lead to payoff -100, and playing M can result in a payoff of -10). Hence: $\underline{v_{row}} = 2$.
- The column player can play L and secure a payoff of at least 0 (playing R puts him in the risk of getting -20). Hence: $\underline{v_{col}} = 0$.

If both players play their maximin strategies (T,L), the payoff vector is (3,1). In contrast, the only Nash equilibrium in this game is (B,R), which leads to a payoff vector of (4,4).

The **minimax value** of a player is the smallest value that the other players can force the player to receive, without knowing his actions. Equivalently, it is the largest value the player can be sure to get when he *knows* the actions of the other players. Its formal definition is:[2]

$$\overline{v_i} = \min_{a_{-i}} \max_{a_i} v_i(a_i, a_{-i})$$

The definition is very similar to that of the maximin value - only the order of the maximum and minimum operators is inverse.

For every player i, the maximin is at most the minimax:

$$\underline{v_i} \leq \overline{v_i}$$

Intuitively, in maximin the maximization comes before the minimization, so player i tries to maximize their value before knowing what the others will do; in minimax the maximization comes after the minimization, so player i is in a much better position - they maximize their value knowing what the others did.

Usually, the maximin is strictly smaller than the minimax. Consider the game in the above example:

- The row player can get a value of 4 (if the other player plays R) or 5 (if the other player plays L), so: $\overline{v_{row}} = 4$.
- The column player can get 1 (if the other player plays T), 1 (if M) or 4 (if B). Hence: $\overline{v_{col}} = 1$.

1.2 In zero-sum games

In zero-sum games, the minimax solution is the same as the Nash equilibrium.

In the context of zero-sum games, the minimax theorem is equivalent to:[3]

> For every two-person, zero-sum game with finitely many strategies, there exists a value V and a mixed strategy for each player, such that
>
> (a) Given player 2's strategy, the best payoff possible for player 1 is V, and
>
> (b) Given player 1's strategy, the best payoff possible for player 2 is −V.

Equivalently, Player 1's strategy guarantees him a payoff of V regardless of Player 2's strategy, and similarly Player 2 can guarantee himself a payoff of −V. The name minimax arises because each player minimizes the maximum payoff possible for the other—since the game is zero-sum, he/she also minimizes his/their own maximum loss (i.e. maximize his/her minimum payoff). See also example of a game without a value.

1.3 Example

The following example of a zero-sum game, where **A** and **B** make simultaneous moves, illustrates *minimax* solutions. Suppose each player has three choices and consider the payoff matrix for **A** displayed on the right. Assume the payoff matrix for **B** is the same matrix with the signs reversed (i.e. if the choices are A1 and B1 then **B** pays 3 to **A**). Then, the minimax choice for **A** is A2 since the worst possible result is then having to pay 1, while the simple minimax choice for **B** is B2 since the worst possible result is then no payment. However, this solution is not stable, since if **B** believes **A** will choose A2 then **B** will choose B1 to gain 1; then if **A** believes **B** will choose B1 then **A** will choose A1 to gain 3; and then **B** will choose B2; and eventually both players will realize the difficulty of making a choice. So a more stable strategy is needed.

Some choices are *dominated* by others and can be eliminated: **A** will not choose A3 since either A1 or A2 will produce a better result, no matter what **B** chooses; **B** will not choose B3 since some mixtures of B1 and B2 will produce a better result, no matter what **A** chooses.

A can avoid having to make an expected payment of more than 1/3 by choosing A1 with probability 1/6 and A2 with probability 5/6: The expected payoff for **A** would be $3 \times (1/6) - 1 \times (5/6) = -1/3$ in case **B** chose B1 and $-2 \times (1/6) + 0 \times (5/6) = -1/3$ in case **B** chose B2. Similarly, **B** can ensure an expected gain of at least 1/3, no matter what **A** chooses, by using a randomized strategy of choosing B1 with probability 1/3 and B2 with probability 2/3. These mixed minimax strategies are now stable and cannot be improved.

1.4 Maximin

Frequently, in game theory, **maximin** is distinct from minimax. Minimax is used in zero-sum games to denote minimizing the opponent's maximum payoff. In a zero-sum game, this is identical to minimizing one's own maximum loss, and to maximizing one's own minimum gain.

"Maximin" is a term commonly used for non-zero-sum games to describe the strategy which maximizes one's own minimum payoff. In non-zero-sum games, this is not generally the same as minimizing the opponent's maximum gain, nor the same as the Nash equilibrium strategy.

1.5 In repeated games

The minimax values are very important in the theory of repeated games. One of the central theorems in this theory, the folk theorem, relies on the minimax values.

2 Combinatorial game theory

In combinatorial game theory, there is a minimax algorithm for game solutions.

A **simple** version of the minimax *algorithm*, stated below, deals with games such as tic-tac-toe, where each player can win, lose, or draw. If player A *can* win in one move, his best move is that winning move. If player B knows that one move will lead to the situation where player A *can* win in one move, while another move will lead to the situation where player A can, at best, draw, then player B's best move is the one leading to a draw. Late in the game, it's easy to see what the "best" move is. The Minimax algorithm helps find the best move, by working backwards from the end of the game. At each step it assumes that player A is trying to **maximize** the chances of A winning, while on the next turn player B is trying to **minimize** the chances of A winning (i.e., to maximize B's own chances of winning).

2.1 Minimax algorithm with alternate moves

A **minimax algorithm**[4] is a recursive algorithm for choosing the next move in an n-player game, usually a two-player game. A value is associated with each position or state of the game. This value is computed by means of a position evaluation function and it indicates how good it would be for a player to reach that position. The player then makes the move that maximizes the minimum value of the position resulting from the opponent's possible following moves. If it is **A**'s turn to move, **A** gives a value to each of his legal moves.

A possible allocation method consists in assigning a certain win for **A** as +1 and for **B** as −1. This leads to combinatorial game theory as developed by John Horton Conway. An alternative is using a rule that if the result of a move is an immediate win for **A** it is assigned positive infinity and, if it is an immediate win for **B**, negative infinity. The value to **A** of any other move is the minimum of the values resulting from each of **B**'s possible replies. For this reason, **A** is called the *maximizing player* and **B** is called the *minimizing player*, hence the name *minimax algorithm*. The above algorithm will assign a value of positive or negative infinity to any position since the value of every position will be the value of some final winning or losing position. Often this is generally only possible at the very end of complicated games such as chess or go, since it is not computationally feasible to look ahead as far as the completion of the game, except towards the end, and instead positions are given finite values as estimates of the degree of belief that they will lead to a win for one player or another.

This can be extended if we can supply a heuristic evaluation function which gives values to non-final game states without considering all possible following complete sequences. We can then limit the minimax algorithm to look only at a certain number of moves ahead. This number is called the “look-ahead”, measured in "plies". For example, the chess computer Deep Blue (the first one to beat a reigning world champion, Garry Kasparov at that time) looked ahead at least 12 plies, then applied a heuristic evaluation function.[5]

The algorithm can be thought of as exploring the nodes of a *game tree*. The *effective branching factor* of the tree is the average number of children of each node (i.e., the average number of legal moves in a position). The number of nodes to be explored usually increases exponentially with the number of plies (it is less than exponential if evaluating forced moves or repeated positions). The number of nodes to be explored for the analysis of a game is therefore approximately the branching factor raised to the power of the number of plies. It is therefore impractical to completely analyze games such as chess using the minimax algorithm.

The performance of the naïve minimax algorithm may be improved dramatically, without affecting the result, by the use of alpha-beta pruning. Other heuristic pruning methods can also be used, but not all of them are guaranteed to give the same result as the un-pruned search.

A naïve minimax algorithm may be trivially modified to additionally return an entire Principal Variation along with a minimax score.

2.2 Pseudocode

The pseudocode for the depth limited minimax algorithm is given below.

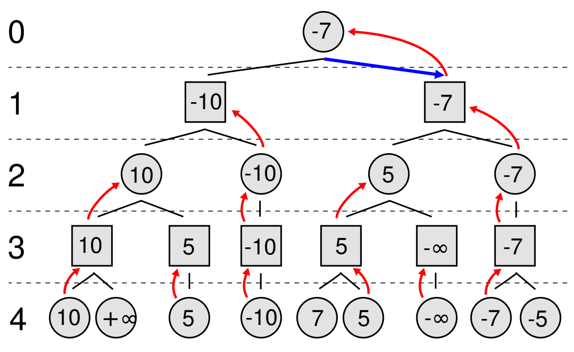
```
01 function minimax(node, depth, maximizingPlayer)
02     if depth = 0 or node is a terminal node
03         return the heuristic value of node
04     if maximizingPlayer
05         bestValue := −∞
06         for each child of node
07             v := minimax(child, depth − 1, FALSE)
08             bestValue := max(bestValue, v)
09         return bestValue
10     else    (* minimizing player *)
11         bestValue := +∞
12         for each child of node
13             v := minimax(child, depth − 1, TRUE)
14             bestValue := min(bestValue, v)
15         return bestValue

(* Initial call for maximizing player *)
minimax(origin, depth, TRUE)
```

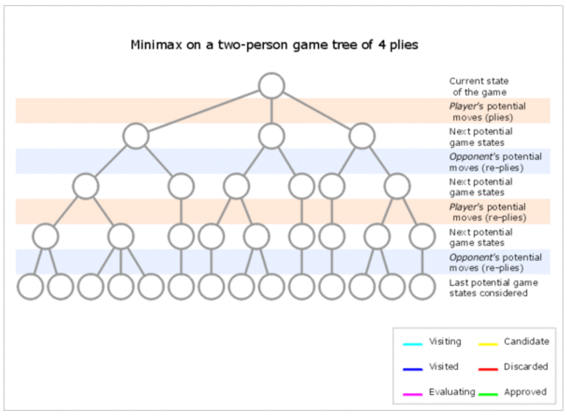
The minimax function returns a heuristic value for leaf nodes (terminal nodes and nodes at the maximum search depth). Non leaf nodes inherit their value, *bestValue*, from a descendant leaf node. The heuristic value is a score measuring the favorability of the node for the maximizing player. Hence nodes resulting in a favorable outcome, such as a win, for the maximizing player have higher scores than nodes more favorable for the minimizing player. The heuristic value for terminal (game ending) leaf nodes are scores corresponding to win, loss, or draw, for the maximizing player. For non terminal leaf nodes at the maximum search depth, an evaluation function estimates a heuristic value for the node. The quality of this estimate and the search depth determine the quality and accuracy of the final minimax result.

Minimax treats the two players (the maximizing player and the minimizing player) separately in its code. Based on the observation that $\max(a, b) = -\min(-a, -b)$, minimax may often be simplified into the negamax algorithm.

2.3 Example



Suppose the game being played only has a maximum of two possible moves per player each turn. The algorithm generates the tree on the right, where the circles represent the moves of the player running the algorithm (*maximizing player*), and squares represent the moves of the opponent (*minimizing player*). Because of the limitation


Minimax on a two-person game tree of 4 plies
Current state of the game
Player's potential moves (plies)
Next potential game states
Opponent's potential moves (re-plies)
Next potential game states
Player's potential moves (re-plies)
Next potential game states
Opponent's potential moves (re-plies)
Last potential game states considered
Visiting
Candidate
Visited
Discarded
Evaluating
Approved

An animated pedagogical example that attempts to be human-friendly by substituting initial infinite (or arbitrarily large) values for emptiness and by avoiding using the negamax coding simplifications.

of computation resources, as explained above, the tree is limited to a *look-ahead* of 4 moves.

The algorithm evaluates each *leaf node* using a heuristic evaluation function, obtaining the values shown. The moves where the *maximizing player* wins are assigned with positive infinity, while the moves that lead to a win of the *minimizing player* are assigned with negative infinity. At level 3, the algorithm will choose, for each node, the **smallest** of the *child node* values, and assign it to that same node (e.g. the node on the left will choose the minimum between “10” and "+∞", therefore assigning the value “10” to itself). The next step, in level 2, consists of choosing for each node the **largest** of the *child node* values. Once again, the values are assigned to each *parent node*. The algorithm continues evaluating the maximum and minimum values of the child nodes alternately until it reaches the *root node*, where it chooses the move with the largest value (represented in the figure with a blue arrow). This is the move that the player should make in order to *minimize* the *maximum* possible loss.

3 Minimax for individual decisions

3.1 Minimax in the face of uncertainty

Minimax theory has been extended to decisions where there is no other player, but where the consequences of decisions depend on unknown facts. For example, deciding to prospect for minerals entails a cost which will be wasted if the minerals are not present, but will bring major rewards if they are. One approach is to treat this as a game against *nature* (see move by nature), and using a similar mindset as Murphy’s law or resistentialism, take an approach which minimizes the maximum expected loss, using the same techniques as in the two-person zero-sum games.

In addition, expectiminimax trees have been developed, for two-player games in which chance (for example, dice) is a factor.

3.2 Minimax criterion in statistical decision theory

Main article: Minimax estimator

In classical statistical decision theory, we have an estimator δ that is used to estimate a parameter $\theta \in \Theta$. We also assume a risk function $R(\theta,\delta)$, usually specified as the integral of a loss function. In this framework, $\tilde{\delta}$ is called **minimax** if it satisfies

$$\sup_\theta R(\theta,\tilde{\delta}) = \inf_\delta \sup_\theta R(\theta,\delta).$$

An alternative criterion in the decision theoretic framework is the Bayes estimator in the presence of a prior distribution Π . An estimator is Bayes if it minimizes the *average* risk

$$\int_\Theta R(\theta,\delta)\, d\Pi(\theta).$$

3.3 Non-probabilistic decision theory

A key feature of minimax decision making is being non-probabilistic: in contrast to decisions using expected value or expected utility, it makes no assumptions about the probabilities of various outcomes, just scenario analysis of what the possible outcomes are. It is thus robust to changes in the assumptions, as these other decision techniques are not. Various extensions of this non-probabilistic approach exist, notably minimax regret and Info-gap decision theory.

Further, minimax only requires ordinal measurement (that outcomes be compared and ranked), not *interval* measurements (that outcomes include “how much better or worse”), and returns ordinal data, using only the modeled outcomes: the conclusion of a minimax analysis is: “this strategy is minimax, as the worst case is (outcome), which is less bad than any other strategy”. Compare to expected value analysis, whose conclusion is of the form: “this strategy yields E(*X*)=*n*." Minimax thus can be used on ordinal data, and can be more transparent.

4 Maximin in philosophy

In philosophy, the term “maximin” is often used in the context of John Rawls's *A Theory of Justice,* where he refers to it (Rawls (1971, p. 152)) in the context of The Difference Principle. Rawls defined this principle as the rule which states that social and economic inequalities should be arranged so that “they are to be of the greatest benefit to the least-advantaged members of society”.[6][7]

5 See also

- Alpha-beta pruning
- Claude Shannon
- Computer chess
- Expectiminimax
- Horizon effect
- Minimax Condorcet
- Monte Carlo tree search
- Minimax regret
- Negamax
- Negascout
- Sion’s minimax theorem
- Transposition table
- Wald’s maximin model

6 Notes

[1] Provincial Healthcare Index 2013 (Bacchus Barua, Fraser Institute, January 2013 -see page 25-)

[2] Michael Maschler, Eilon Solan & Shmuel Zamir (2013). *Game Theory*. Cambridge University Press. pp. 176–180. ISBN 9781107005488.

[3] Osborne, Martin J., and Ariel Rubinstein. *A Course in Game Theory*. Cambridge, MA: MIT, 1994. Print.

[4] Russell, Stuart J.; Norvig, Peter (2003), *Artificial Intelligence: A Modern Approach* (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, pp. 163–171, ISBN 0-13-790395-2

[5] Hsu, Feng-hsiung (1999), “IBM’s Deep Blue Chess Grandmaster Chips”, *IEEE Micro*, Los Alamitos, CA, USA: IEEE Computer Society, **19** (2): 70–81, doi:10.1109/40.755469, During the 1997 match, the software search extended the search to about 40 plies along the forcing lines, even though the nonextended search reached only about 12 plies.

[6] Arrow, “Some Ordinalist-Utilitarian Notes on Rawls’s Theory of Justice, Journal of Philosophy 70, 9 (May 1973), pp. 245-263.

[7] Harsanyi, “Can the Maximin Principle Serve as a Basis for Morality? a Critique of John Rawls’s Theory, American Political Science Review 69, 2 (June 1975), pp. 594-606.

7 External links

- Hazewinkel, Michiel, ed. (2001), “Minimax principle”, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- A visualization applet
- “Maximin principle” from A Dictionary of Philosophical Terms and Names.
- Play a betting-and-bluffing game against a mixed minimax strategy
- The Dictionary of Algorithms and Data Structures entry for minimax.
- Minimax (with or without alpha-beta pruning) algorithm visualization — game tree solving (Java Applet), for balance or off-balance trees.
- Minimax Tutorial with a Numerical Solution Platform
- Java implementation used in a Checkers Game

8 Text and image sources, contributors, and licenses

8.1 Text

8.2 Images

8.3 Content license