Introduction to Artificial Intelligence

Adversarial Games

Andres Mendez-Vazquez

January 24, 2019

Outline

Outline

Introduction

Observation

Games are interesting because they are too hard to solve.

Introduction

Observation

Games are interesting because they are too hard to solve.

For example:

- Chess has an average branching factor of about 35.
- Chess games often go to 50 moves by each player, so the search tree has about 35^{100} nodes.

Introduction

Observation

Games are interesting because they are too hard to solve.

For example:

- Chess has an average branching factor of about 35.
- Chess games often go to 50 moves by each player, so the search tree has about 35^{100} nodes.

In addition

Games also penalize inefficiency severely.

Introduction

Observation

Games are interesting because they are too hard to solve.

For example:

- Chess has an average branching factor of about 35.
- Chess games often go to 50 moves by each player, so the search tree has about 35^{100} nodes.

In addition

Games also penalize inefficiency severely.

Example Against Using Classic Search

Something Notable

A* is a best-first graph search algorithm that finds the least-cost path from a given initial node to one goal node.

Example Against Using Classic Search

Something Notable

A* is a best-first graph search algorithm that finds the least-cost path from a given initial node to one goal node.

However

- Although this can be half efficient in many problems.
- In adversarial games as chess, a single bad move can be highly penalized.

Example Against Using Classic Search

Something Notable

A* is a best-first graph search algorithm that finds the least-cost path from a given initial node to one goal node.

However

- Although this can be half efficient in many problems.
- In adversarial games as chess, a single bad move can be highly penalized.

Outline

Differences

We have two main differences

- **"Unpredictable" opponents**
  - It makes specifying every move for every reply given by the opponent something quite difficult.
- **Time Limits**
  - Quite different to find a goal, thus you must approximate.

Differences

We have two main differences

- **"Unpredictable" opponents**
  - It makes specifying every move for every reply given by the opponent something quite difficult.
- **Time Limits**
  - Quite different to find a goal, thus you must approximate.

Differences

We have two main differences

- **"Unpredictable" opponents**
  - It makes specifying every move for every reply given by the opponent something quite difficult.
- **Time Limits**
  - Quite different to find a goal, thus you must approximate.

Differences

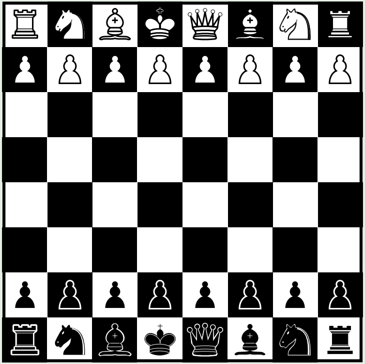
We have two main differences

- **"Unpredictable" opponents**
  - It makes specifying every move for every reply given by the opponent something quite difficult.
- **Time Limits**
  - Quite different to find a goal, thus you must approximate.

Example

Chess Board

According to John McCarthy, Chess is the Drosophila of AI, in an analogy with dominant use of that fruit fly to study inheritance.



Outline

Game Theory

Definition

"Game theory is the study of mathematical models of conflict and cooperation between intelligent rational decision-makers."

- Myerson, Roger B. (1991). Game Theory: Analysis of Conflict, Harvard University Press, p. 1. Chapter-preview links, pp. vii–xi.

Here, yo need to define

1. Players of the game
2. Payoff for each outcome

Game Theory

Definition

“Game theory is the study of mathematical models of conflict and cooperation between intelligent rational decision-makers."

- Myerson, Roger B. (1991). Game Theory: Analysis of Conflict, Harvard University Press, p. 1. Chapter-preview links, pp. vii–xi.

Here, yo need to define

1. Players of the game
2. Payoff for each outcome

Example: Two Player Game

Rock-Scissor-Paper

Consider the two player game "Rock-Scissors-Paper"

Example: Two Player Game

Rock-Scissor-Paper

Consider the two player game "Rock-Scissors-Paper"

Rules

- If both persons play the same action, then it is a draw game.
- You get a matrix of possible payoffs

	P	S	R
P	0	-1	1
S	1	0	-1
R	-1	1	0

or $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$

- It is based on the definition of Nash equilibrium

Example: Two Player Game

Rock-Scissor-Paper

Consider the two player game "Rock-Scissors-Paper"

Rules

- If both persons play the same action, then it is a draw game.
- You get a matrix of possible payoffs

	P	S	R
P	0	-1	1
S	1	0	-1
R	-1	1	0

or $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$

Observations

- In game theory, this is the two player zero-sum game.
- It is based on the definition of Nash equilibrium.

Example: Two Player Game

Rock-Scissor-Paper

Consider the two player game "Rock-Scissors-Paper"

Rules

- If both persons play the same action, then it is a draw game.
- You get a matrix of possible payoffs

	P	S	R
P	0	-1	1
S	1	0	-1
R	-1	1	0

or $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$

Observations

- In game theory, this is the two player zero-sum game.
- It is based on the definition of Nash equilibrium.

Example: Two Player Game

Rock-Scissor-Paper

Consider the two player game "Rock-Scissors-Paper"

Rules

- If both persons play the same action, then it is a draw game.
- You get a matrix of possible payoffs

	P	S	R
P	0	-1	1
S	1	0	-1
R	-1	1	0

or $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$

Observations

- In game theory, this is the two player zero-sum game.
- It is based on the definition of Nash equilibrium.

Example: Two Player Game

Rock-Scissor-Paper

Consider the two player game "Rock-Scissors-Paper"

Rules

- If both persons play the same action, then it is a draw game.
- You get a matrix of possible payoffs

	P	S	R
P	0	-1	1
S	1	0	-1
R	-1	1	0

or $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$

Observations

- In game theory, this is the two player zero-sum game.
- It is based on the definition of Nash equilibrium.

Strategy

Definition

The **strategy** of a player is any of the options it can choose in a setting where the outcome depends not only on his own actions but on the action of others.

Thus

A strategy set of a player defines what strategies are available for them to play.

For example

In the case of the Rock-Scissor-Paper's Game, we could use the following strategy:

Rock	Scissor	Paper
0	0	1

Note: You always choose Paper.

Strategy

Definition

The **strategy** of a player is any of the options it can choose in a setting where the outcome depends not only on his own actions but on the action of others.

Thus

A strategy set of a player defines what strategies are available for them to play.

For example

In the case of the Rock-Scissor-Paper's Game, we could use the following strategy:

Rock	Scissor	Paper
0	0	1

Note: You always choose Paper.

Strategy

Definition

The **strategy** of a player is any of the options it can choose in a setting where the outcome depends not only on his own actions but on the action of others.

Thus

A strategy set of a player defines what strategies are available for them to play.

For example

In the case of the Rock-Scissor-Paper's Game, we could use the following strategy:

Rock	Scissor	Paper
0	0	1

Note: You always choose Paper.

Therefore

Something Notable

We know there is no pure strategy solution for this game.

- Where pure strategies provides a complete definition of how a player will play a game.

Therefore

Something Notable

We know there is no pure strategy solution for this game.

- Where pure strategies provides a complete definition of how a player will play a game.

Solution? Use a probabilistic approach with the pure strategy

$$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$$

Therefore

Something Notable

We know there is no pure strategy solution for this game.

- Where pure strategies provides a complete definition of how a player will play a game.

Solution? Use a probabilistic approach with the pure strategy

$$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$$

Outline

Formally

A game can be seen a search problem with

- The **initial state** – include the board position and initial player.
- A **successor function**, which returns a list of (**move, state**) pairs.
  - This indicate a legal move and the resulting state.

Formally

A game can be seen a search problem with

- The **initial state** – include the board position and initial player.
- A **successor function**, which returns a list of (**move, state**) pairs.
  - This indicate a legal move and the resulting state.

In addition

- A **series of terminal states**, which determines if the game is over.
- A **utility function** (objective function or payoff function), which gives a numeric value for the terminal states.
  - For example Chess – win, loss, or draw +1, -1, or 0.

Formally

A game can be seen a search problem with

- The **initial state** – include the board position and initial player.
- A **successor function**, which returns a list of (**move, state**) pairs.
  - This indicate a legal move and the resulting state.

In addition

- A **series of terminal states,** which determines if the game is over.
- A **utility function** (objective function or payoff function), which gives a numeric value for the terminal states.
  - For example Chess – win, loss, or draw +1, -1, or 0.

Formally

A game can be seen a search problem with

- The **initial state** – include the board position and initial player.
- A **successor function**, which returns a list of (**move, state**) pairs.
  - This indicate a legal move and the resulting state.

In addition

- A **series of terminal states,** which determines if the game is over.
- A **utility function** (objective function or payoff function), which gives a numeric value for the terminal states.
  - For example Chess – win, loss, or draw +1, -1, or 0.

Formally

A game can be seen a search problem with

- The **initial state** – include the board position and initial player.
- A **successor function**, which returns a list of (**move, state**) pairs.
  - This indicate a legal move and the resulting state.

In addition

- A **series of terminal states,** which determines if the game is over.
- A **utility function** (objective function or payoff function), which gives a numeric value for the terminal states.
  - For example Chess – win, loss, or draw +1, -1, or 0.

Formally

A game can be seen a search problem with

- The **initial state** – include the board position and initial player.
- A **successor function**, which returns a list of (**move, state**) pairs.
  - This indicate a legal move and the resulting state.

In addition

- A **series of terminal states,** which determines if the game is over.
- A **utility function** (objective function or payoff function), which gives a numeric value for the terminal states.
  - For example Chess – win, loss, or draw +1, -1, or 0.

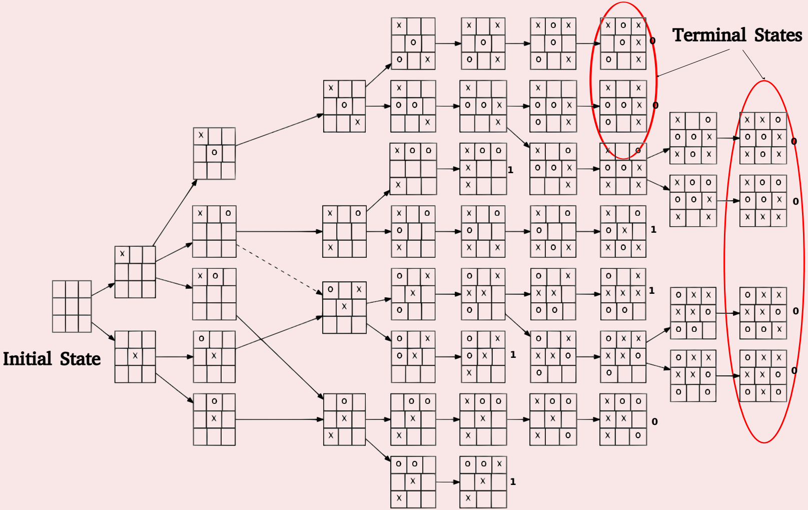
Example

If you are player "X"

- Win is $+1$
- Loss is -1
- Draw is 0

Tic-Tac-Toe

Expansion Tree



Outline

Minimax

Rule

Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario.

Minimax

Rule

Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario.

History

- Originally formulated for two-player zero-sum game theory.
  - Here players can take alternate moves or simultaneous moves.

- In game parlance, this game is one deep move consisting of two half-moves, called ply.

Minimax

Rule

Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario.

History

- Originally formulated for two-player zero-sum game theory.
  - Here players can take alternate moves or simultaneous moves.

In this games each turn

Finishes after each Max and Min make a move:

- In game parlance, this game is one deep move, consisting of two half-moves, called ply.

Minimax

Rule

Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario.

History

- Originally formulated for two-player zero-sum game theory.
  - Here players can take alternate moves or simultaneous moves.

In this games each turn

Finishes after each Max and Min make a move:

- In game parlance, this game is one deep move, consisting of two half-moves, called ply.

Minimax

Rule

Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario.

History

- Originally formulated for two-player zero-sum game theory.
  - Here players can take alternate moves or simultaneous moves.

In this games each turn

Finishes after each Max and Min make a move:

- In game parlance, this game is one deep move, consisting of two half-moves, called ply.

Strategy for the Minimax Game

Consider all moves and select the optimal one

Optimal is the move that results in the most favorable position even if opponent does his/her best.

Little Problem!!!

You do not know what opponent thinks is the best but assume he/she thinks like you.

Strategy for the Minimax Game

Consider all moves and select the optimal one

Optimal is the move that results in the most favorable position even if opponent does his/her best.

Little Problem!!!

You do not know what opponent thinks is the best but assume he/she thinks like you.

Outline

What is known as Nash Equilibrium

Definition

In game theory, the Nash equilibrium is a solution concept of a non-cooperative game involving two or more players:

- In which each player is assumed to know the equilibrium strategies of the other players
- No player has anything to gain by changing only their own strategy.

What is known as Nash Equilibrium

Definition

In game theory, the Nash equilibrium is a solution concept of a non-cooperative game involving two or more players:

- In which each player is assumed to know the equilibrium strategies of the other players
- No player has anything to gain by changing only their own strategy.

What is known as Nash Equilibrium

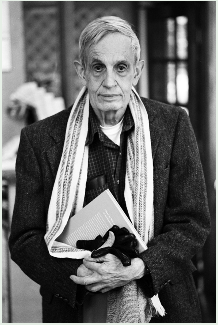
Definition

In game theory, the Nash equilibrium is a solution concept of a non-cooperative game involving two or more players:

- In which each player is assumed to know the equilibrium strategies of the other players
- No player has anything to gain by changing only their own strategy.

John Forbes Nash, Jr

Who is him?



John Forbes Nash, Jr. (born June 13, 1928) is an American mathematician whose works in game theory, differential geometry, and partial differential equations have provided insight into the factors that govern chance and events inside complex systems in daily life.

Outline

John von Neumann

Who is him?

- John von Neumann was a Hungarian and later American pure and applied mathematician, physicist, inventor and polymath.
- He made major contributions to a number of fields, including mathematics (foundations of mathematics, functional analysis, ergodic theory, geometry, topology, and numerical analysis), physics (quantum mechanics, hydrodynamics, and fluid dynamics), economics (game theory), computing (Von Neumann architecture, linear programming, self-replicating machines, stochastic computing), and statistics.

John von Neumann

Who is him?

- John von Neumann was a Hungarian and later American pure and applied mathematician, physicist, inventor and polymath.
- He made major contributions to a number of fields, including mathematics (**foundations of mathematics, functional analysis, ergodic theory, geometry, topology, and numerical analysis**), physics (**quantum mechanics, hydrodynamics, and fluid dynamics**), economics (**game theory**), computing (**Von Neumann architecture, linear programming, self-replicating machines, stochastic computing**), and statistics.

Outline

Setup for the Minimax Theorem

Remember the Payoff Matrix for Paper-Scissor-Rock

	P	S	R
P	0	-1	1
S	1	0	-1
R	-1	1	0

or $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$ or $A = (a_{ij})$

- a_{ij} represents a payoff given by
  - i strategy - row player
  - j strategy - column player

Setup for the Minimax Theorem

Remember the Payoff Matrix for Paper-Scissor-Rock

	P	S	R
P	0	-1	1
S	1	0	-1
R	-1	1	0

or $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$ or $A = (a_{ij})$

- a_{ij} represents a payoff given by
  - i strategy - row player
  - j strategy - column player

Setup for the Minimax Theorem

Remember the Payoff Matrix for Paper-Scissor-Rock

	P	S	R
P	0	-1	1
S	1	0	-1
R	-1	1	0

or $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$ or $A = (a_{ij})$

- a_{ij} represents a payoff given by
  - i strategy - row player
  - j strategy - column player

Setup for the Minimax Theorem

Remember the Payoff Matrix for Paper-Scissor-Rock

	P	S	R
P	0	-1	1
S	1	0	-1
R	-1	1	0

or $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$ or $A = (a_{ij})$

- a_{ij} represents a payoff given by
  - i strategy - row player
  - j strategy - column player

Thus, The Game is Played as Follow

We have then

The payoff a_{ij} is given to the row player when he/she picks strategy i and the column player picks strategy j

Therefore

The payoff to the column player is $-a_{ij}$

Then

Let y_i and x_j represent probabilities of the row player and column player picking their i^{th} and j^{th} strategies respectively

Thus, The Game is Played as Follow

We have then

The payoff a_{ij} is given to the row player when he/she picks strategy i and the column player picks strategy j

Therefore

The payoff to the column player is $-a_{ij}$

Then

Let y_i and x_j represent probabilities of the row player and column player picking their i^{th} and j^{th} strategies respectively

Thus, The Game is Played as Follow

We have then

The payoff a_{ij} is given to the row player when he/she picks strategy i and the column player picks strategy j

Therefore

The payoff to the column player is $-a_{ij}$

Then

Let y_i and x_j represent probabilities of the row player and column player picking their i^{th} and j^{th} strategies respectively

Thus, we have

The vectors for the mixed strategies

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \text{ and } \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{1}$$

With $\sum_i y_i = 1$ and $\sum_j x_j = 1$

Thus, we have

The vectors for the mixed strategies

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \text{ and } \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{1}$$

With $\sum_i y_i = 1$ and $\sum_j x_j = 1$

The resulting expected payoffs

$$\boldsymbol{y}^T A \boldsymbol{x} \tag{2}$$

Thus for this Game

We have the following theorem

Theorem (Von Neumann Minimax Theorem)
For every two-person, zero-sum game given by a payoff matrix $n \times m$ A, there exists a mixed strategy for each player, such that the expected payoff for both is the same value V when the players use these strategies. Furthermore, V is the best payoff each can expect to receive from a play of the game; that is, these mixed strategies are the optimal strategies for the two players.

Proof

We formulate the payoff of row player and column player as below

- Payoff of row player (minmaximizer) $= \min_{\boldsymbol{y} \in \Delta^n} \max_{\boldsymbol{x} \in \Delta^m} \boldsymbol{y}^t A \boldsymbol{x}$
- Payoff of column player (maxminimizer) $= \max_{\boldsymbol{x} \in \Delta^m} \min_{\boldsymbol{y} \in \Delta^n} \boldsymbol{y}^t A \boldsymbol{x}$

Proof

We formulate the payoff of row player and column player as below

- Payoff of row player (minmaximizer) $= \min\limits_{\boldsymbol{y}\in\Delta^n}\max\limits_{\boldsymbol{x}\in\Delta^m} \boldsymbol{y}^t A\boldsymbol{x}$
- Payoff of column player (maxminimizer) $= \max\limits_{\boldsymbol{x}\in\Delta^m}\min\limits_{\boldsymbol{y}\in\Delta^n} \boldsymbol{y}^t A\boldsymbol{x}$

Where

- A is the payoff matrix $n \times m$.
- x and y are probability vectors i.e. $\sum_{i=1}^{n} y_i = 1$ and $\sum_{j=1}^{m} x_j = 1$.
- Δ^m or Δ^n strategy sets.

Proof

We formulate the payoff of row player and column player as below

- Payoff of row player (minmaximizer) $= \min_{\boldsymbol{y}\in\Delta^n} \max_{\boldsymbol{x}\in\Delta^m} \boldsymbol{y}^t A\boldsymbol{x}$
- Payoff of column player (maxminimizer) $= \max_{\boldsymbol{x}\in\Delta^m} \min_{\boldsymbol{y}\in\Delta^n} \boldsymbol{y}^t A\boldsymbol{x}$

Where

- A is the payoff matrix $n \times m$.
- x and y are probability vectors i.e. $\sum_{i=1}^{n} y_i = 1$ and $\sum_{j=1}^{m} x_j = 1$.
- Δ^m or Δ^n strategy sets.

Proof

We formulate the payoff of row player and column player as below

- Payoff of row player (minmaximizer) $= \min_{\boldsymbol{y} \in \Delta^n} \max_{\boldsymbol{x} \in \Delta^m} \boldsymbol{y}^t A \boldsymbol{x}$
- Payoff of column player (maxminimizer) $= \max_{\boldsymbol{x} \in \Delta^m} \min_{\boldsymbol{y} \in \Delta^n} \boldsymbol{y}^t A \boldsymbol{x}$

Where

- A is the payoff matrix $n \times m$.
- x and y are probability vectors i.e. $\sum_{i=1}^{n} y_i = 1$ and $\sum_{j=1}^{m} x_j = 1$.
- Δ^m or Δ^n strategy sets.

Proof

We formulate the payoff of row player and column player as below

- Payoff of row player (minmaximizer) $= \min_{\boldsymbol{y} \in \Delta^n} \max_{\boldsymbol{x} \in \Delta^m} \boldsymbol{y}^t A \boldsymbol{x}$
- Payoff of column player (maxminimizer) $= \max_{\boldsymbol{x} \in \Delta^m} \min_{\boldsymbol{y} \in \Delta^n} \boldsymbol{y}^t A \boldsymbol{x}$

Where

- A is the payoff matrix $n \times m$.
- x and y are probability vectors i.e. $\sum_{i=1}^{n} y_i = 1$ and $\sum_{j=1}^{m} x_j = 1$.
- Δ^m or Δ^n strategy sets.

Formally, the Minimax Theorem looks like

Theorem (Von Neumann Minimax Theorem)

$$\min_{\boldsymbol{y}\in\Delta^n}\max_{\boldsymbol{x}\in\Delta^m}\boldsymbol{y}^t A\boldsymbol{x} = V = \max_{\boldsymbol{x}\in\Delta^m}\min_{\boldsymbol{y}\in\Delta^n}\boldsymbol{y}^t A\boldsymbol{x} \tag{3}$$

Proof

Questions

- What happens if one of the players exposes her mixed strategy and let the other player choose the strategy of his liking?
- if we reverse the order of the players?

Actually

The von Neumann's theorem allow to say the order does not change the valuer of the game

Proof

Questions

- What happens if one of the players exposes her mixed strategy and let the other player choose the strategy of his liking?
- if we reverse the order of the players?

Actually

The von Neumann's theorem allow to say the order does not change the valuer of the game

Proof

Now

If you chose strategy $\boldsymbol{x}$, the payoff $\min_{\boldsymbol{y} \in \Delta^n} \boldsymbol{y}^t A \boldsymbol{x}$ is a simple linear programming problem

With the constraints

$$\boldsymbol{y} \geq 0$$

$$\sum_i y_i = 1$$

Proof

Now

If you chose strategy $\boldsymbol{x}$, the payoff $\min_{\boldsymbol{y}\in\Delta^n} \boldsymbol{y}^t A\boldsymbol{x}$ is a simple linear programming problem

With the constraints

$$\boldsymbol{y} \geq 0$$
$$\sum_i y_i = 1$$

Proof

This problem defines a polytope

- With vectors $\{e_i\}_{i=1}^{n}$, where e_i is a vector with 1 at i^{th} location and 0 otherwise.

In other words

We may always assume the second player always chooses a pure strategy to achieve the best payoff for herself.

Proof

This problem defines a polytope

- With vectors $\{e_i\}_{i=1}^{n}$, where e_i is a vector with 1 at i^{th} location and 0 otherwise.

In other words

We may always assume the second player always chooses a pure strategy to achieve the best payoff for herself.

Proof

Then, we have that

$$\max_{\boldsymbol{x}\in\Delta^m}\min_{\boldsymbol{y}\in\Delta^n}\boldsymbol{y}^t A\boldsymbol{x} = \max_{\boldsymbol{x}\in\Delta^m}\min_{i}\left(A\boldsymbol{x}\right)_i \tag{4}$$

Second

$$\min_{\boldsymbol{y}\in\Delta^n}\max_{\boldsymbol{x}\in\Delta^m}\boldsymbol{y}^t A\boldsymbol{x} = \min_{\boldsymbol{y}\in\Delta^n}\max_{j}\left(\boldsymbol{y}A\right)_j \tag{5}$$

Proof

Then, we have that

$$\max_{\boldsymbol{x}\in\Delta^m}\min_{\boldsymbol{y}\in\Delta^n}\boldsymbol{y}^t A\boldsymbol{x} = \max_{\boldsymbol{x}\in\Delta^m}\min_{i}\,(A\boldsymbol{x})_i \tag{4}$$

Second

$$\min_{\boldsymbol{y}\in\Delta^n}\max_{\boldsymbol{x}\in\Delta^m}\boldsymbol{y}^t A\boldsymbol{x} = \min_{\boldsymbol{y}\in\Delta^n}\max_{j}\,(\boldsymbol{y}A)_j \tag{5}$$

Proof

What we want?

We want to prove that they are equal!!!

$$\max_{\boldsymbol{x}\in\Delta^m}\min_i (A\boldsymbol{x})_i = \min_{\boldsymbol{y}\in\Delta^n}\max_j (\boldsymbol{y}A)_j \tag{6}$$

Proof

Using Linear Programming setup equation $\max_{\boldsymbol{x} \in \Delta^m} \min_i (A\boldsymbol{x})_i$ becomes

$$
\begin{aligned}
& \max t \\
s.t. \ & \sum_j a_{ij} x_j \geq t \\
& \sum_j x_j = 1 \\
& x \geq 0 \\
& t \lesseqgtr 0 \text{ (Unconstrained)}
\end{aligned}
$$

Proof

Similarly the payoff for the column player, $\min_{\boldsymbol{y} \in \Delta^n} \max_j (\boldsymbol{y}A)_j$

$$
\begin{aligned}
& \min w \\
s.t. \; & \sum_j y_i a_{ij} \leq w \\
& \boldsymbol{y} \in \Delta^n \\
& w \lessgtr 0 \text{ (Unconstrained)}
\end{aligned}
$$

Thus

We can use the idea of duality for it

Proof

Similarly the payoff for the column player, $\min\limits_{\boldsymbol{y} \in \Delta^n} \max_j (\boldsymbol{y}A)_j$

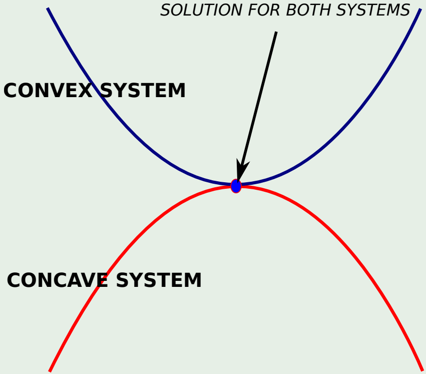
$$\begin{aligned} &\min w \\ s.t.\ &\sum_j y_i a_{ij} \leq w \\ &\boldsymbol{y} \in \Delta^n \\ &w \lessgtr 0 \text{ (Unconstrained)} \end{aligned}$$

Thus

We can use the idea of duality for it

Proof

An intuitive idea about the Duality!!!



A more formal and different view

Primal of linear programming

$$maximize : z = \sum_{j=1}^{n} c_j x_j$$

$$s.t. \sum_{j=1}^{n} a_{ij} x_j \leq b_i \ \ (i = 1, 2, ..., m)$$

$$x_j \geq 0 \ \ (j = 1, 2, ..., n)$$

A more formal view

Symmetric Dual

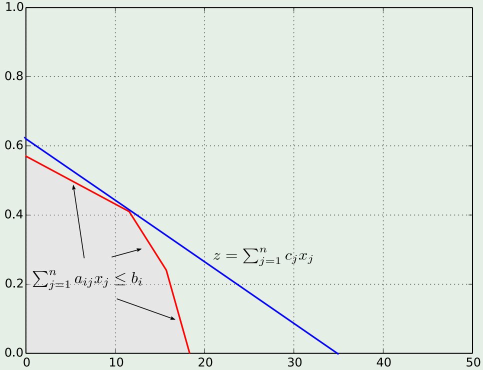
$$minimize :v = \sum_{i=1}^{m} b_i y_i$$

$$s.t. \sum_{i=1}^{m} a_{ij} y_i \geq c_j \ \ (i = 1, 2, ..., m)$$

$$y_i \geq 0 \ \ (j = 1, 2, ..., n)$$

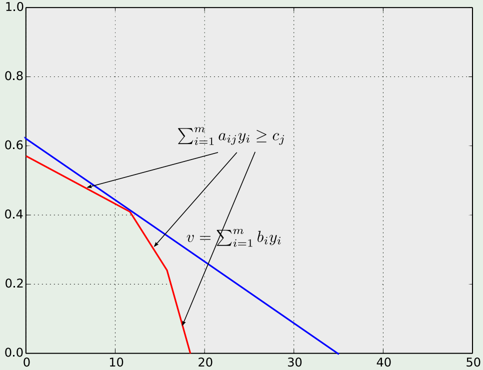
Proof

Primal for LP



Proof

Dual for LP



Re-writing the Primal

Primal

$$
\begin{aligned}
& \max t \\
s.t.\ & t - \sum_j a_{ij} x_j \leq 0 \\
& \sum_j x_j = 1 \\
& x \geq 0 \\
& t \lesseqgtr 0 \text{ (Unconstrained)}
\end{aligned}
$$

The Strong Duality Theorem

If either Primal or Dual has a finite optimal value, then so does the other, the optimal values coincide, and optimal solutions to both Primal and Dual exist.

Re-writing the Primal

Primal

$$\max t$$

$$s.t.\ t - \sum_j a_{ij} x_j \leq 0$$

$$\sum_j x_j = 1$$

$$x \geq 0$$

$$t \lesseqgtr 0 \text{ (Unconstrained)}$$

The Strong Duality Theorem

If either Primal or Dual has a finite optimal value, then so does the other, the optimal values coincide, and optimal solutions to both Primal and Dual exist.

Thus

Applying the definition

$$
\begin{aligned}
\min w \\
s.t.\ w - \sum_i y_i a_{ij} &\geq 0 \\
\sum_i y_i &= 1 \\
\boldsymbol{y} &\geq 0 \\
w &\lesseqgtr 0 \text{ (Unconstrained)}
\end{aligned}
$$

Then

Re-writing the Dual

$$
\begin{aligned}
&\min w\\
s.t.\ \sum_i y_i a_{ij} &\leq w\\
\boldsymbol{y} &\in \Delta^n\\
w &\lesseqgtr 0 \text{ (Unconstrained)}
\end{aligned}
$$

Finally

We have the following

The previous system is equivalent to the problem of

$$\min_{\boldsymbol{y} \in \Delta^n} \max_j \left(\boldsymbol{y}A\right)_j \tag{7}$$

Then

$$\max_{\boldsymbol{x} \in \Delta^m} \min_i \left(A\boldsymbol{x}\right)_i = \min_{\boldsymbol{y} \in \Delta^n} \max_j \left(\boldsymbol{y}A\right)_j \quad \text{Q.E.D.} \tag{8}$$

Finally

We have the following

The previous system is equivalent to the problem of

$$\min_{\boldsymbol{y}\in\Delta^n} \max_j \left(\boldsymbol{y}A\right)_j \tag{7}$$

Then

$$\max_{\boldsymbol{x}\in\Delta^m} \min_i \left(A\boldsymbol{x}\right)_i = \min_{\boldsymbol{y}\in\Delta^n} \max_j \left(\boldsymbol{y}A\right)_j \text{ Q.E.D.} \tag{8}$$

Outline

Minimax Algorithm

It is based in the repeated application d-times of two ply to a game

$$\underbrace{\max_{i}\min_{j}\max_{k}\dots\min_{w}}_{d} f\left(i,j,\dots,w\right)$$

Thus

You have that the MAX player tries to get as many points as possible, and the MIN player tries to minimize its loses.

Minimax Algorithm

It is based in the repeated application d-times of two ply to a game

$$\underbrace{\max_{i}\min_{j}\max_{k}\ldots\min_{w}}_{d} f\left(i,j,\ldots,w\right)$$

Thus

You have that the MAX player tries to get as many points as possible, and the MIN player tries to minimize its loses.

Minimax Algorithm

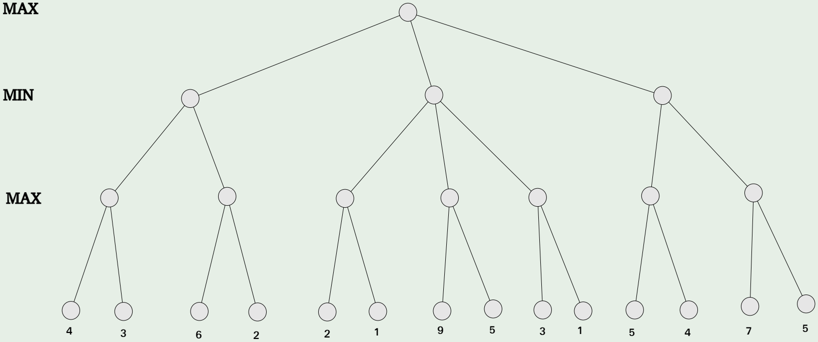
This allows to define the following cost function Minimax function

$$MinMax\left(n\right)=\begin{cases}Utility\left(n\right) & \text{if } n \text{ is a terminal state}\\ max_{s\in Succ(n)}\left\{MinMax\left(s\right)\right\} & \text{if } n \text{ is a MAX state}\\ min_{s\in Succ(n)}\left\{MinMax\left(s\right)\right\} & \text{if } n \text{ is a MIN state}\end{cases}$$

Outline

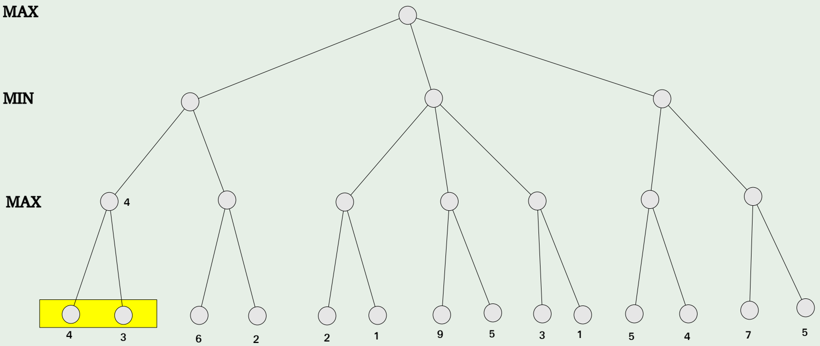
Example

We have



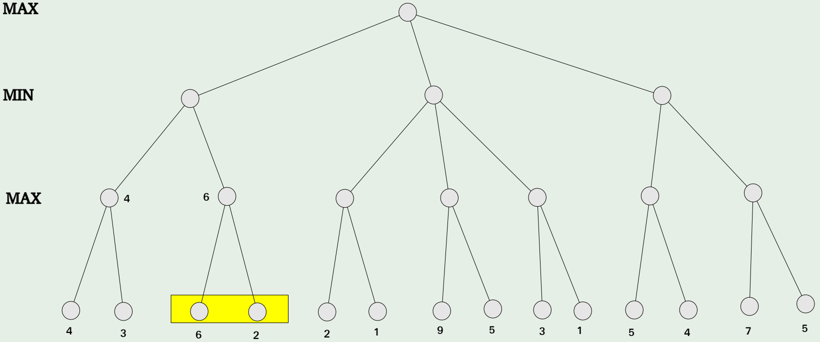
Example

We have



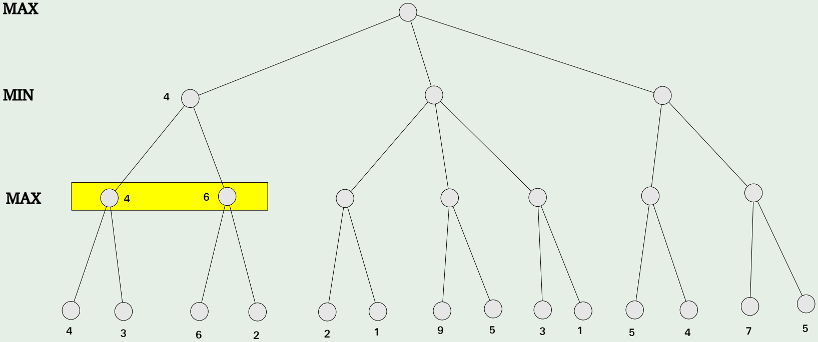
Example

We have



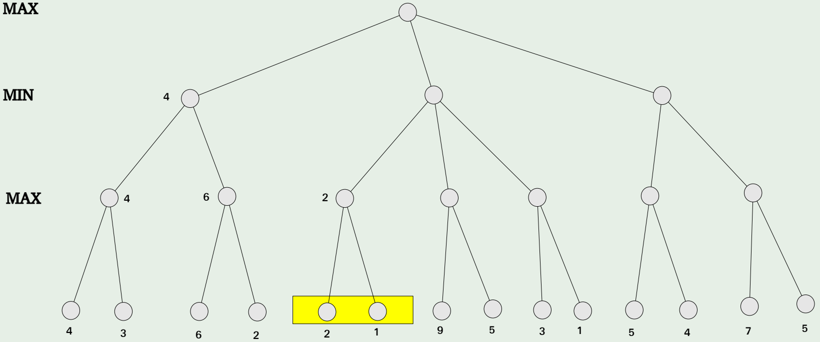
Example

We have



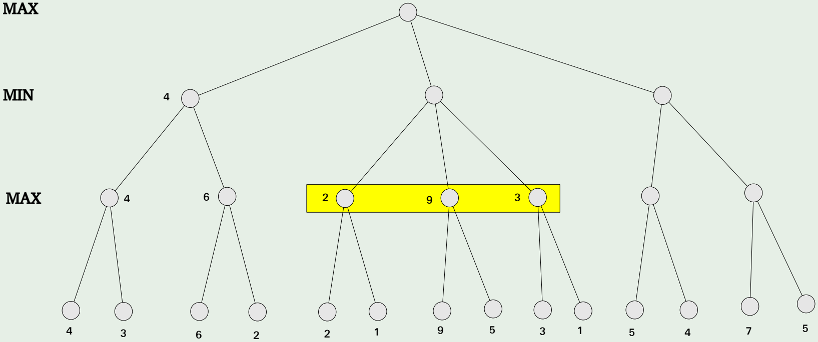
Example

We have



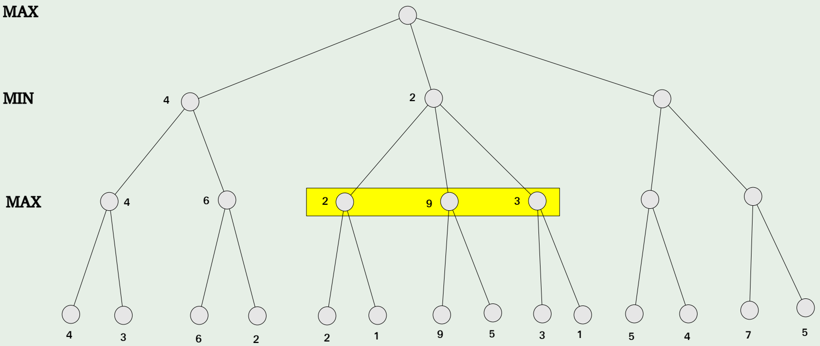
Example

We have



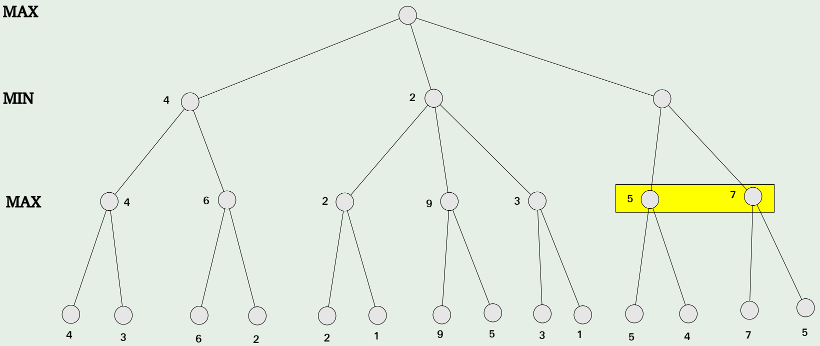
Example

We have



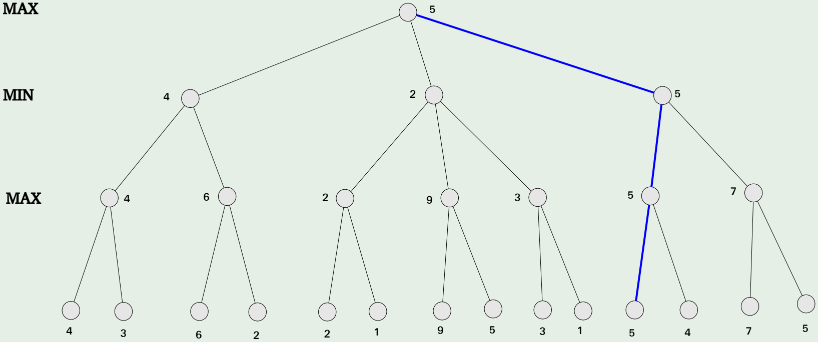
Example

We have



Example

We have



Final Algorithm

Procedure Minimax(u)

Input: Position u

Output: Value at root

```
if (leaf(u))
    return Eval(u)
if (max − node(u))  ◁ Selecting the Max or Min Node
    val = −∞
else
    val = +∞
for each v ∈ Succ(u)
    if (max − node(u))
        val = max{val, Minimax(v)}
    else
        val = min{val, Minimax(v)}
return val
```

Properties of the Minimax

Properties

1. **Complete**? Yes (if tree is finite)
2. **Optimal**? Yes (against an optimal opponent)
3. **Time complexity**? $O\left(b^{\delta}\right)$
4. **Space complexity**? $O(b\delta)$ (depth-first exploration)

Properties of the Minimax

Properties

1. **Complete**? Yes (if tree is finite)
2. **Optimal**? Yes (against an optimal opponent)
3. **Time complexity**? $O\left(b^{\delta}\right)$
4. **Space complexity**? $O(b\delta)$ (depth-first exploration)

Properties of the Minimax

Properties

1. **Complete**? Yes (if tree is finite)
2. **Optimal**? Yes (against an optimal opponent)
3. **Time complexity**? $O\left(b^{\delta}\right)$
4. **Space complexity**? $O(b\delta)$ (depth-first exploration)

Properties of the Minimax

Properties

1. **Complete**? Yes (if tree is finite)
2. **Optimal**? Yes (against an optimal opponent)
3. **Time complexity**? $O\left(b^{\delta}\right)$
4. **Space complexity**? $O(b\delta)$ (depth-first exploration)

However

STILL!!!

For chess, $b \approx 35$, $\delta \approx 100$ for "reasonable" games THUS exact solution completely infeasible

What to do?

Pruning minimax tree

However

STILL!!!

For chess, $b \approx 35$, $\delta \approx 100$ for "reasonable" games THUS exact solution completely infeasible

What to do?

Pruning minimax tree

Outline

Pruning

Are there times when you know you need not explore a particular move?

- When the move is poor?
- Poor compared to what?
- Poor compared to what you have explored so far.

Pruning

Are there times when you know you need not explore a particular move?

- When the move is poor?
- Poor compared to what?
- Poor compared to what you have explored so far.

Pruning

Are there times when you know you need not explore a particular move?

- When the move is poor?
- Poor compared to what?
- Poor compared to what you have explored so far.

Thus

$\alpha - \beta$ Pruning

We can improve on the performance of the minimax algorithm through $\alpha - \beta$ pruning

Basic Idea

"If you have an idea that is surely bad, don't take the time to see how truly awful it is." – Pat Winston

Thus

$\alpha - \beta$ Pruning

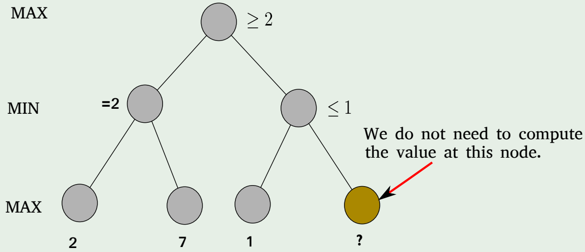
We can improve on the performance of the minimax algorithm through $\alpha - \beta$ pruning

Basic Idea

"If you have an idea that is surely bad, don't take the time to see how truly awful it is." – Pat Winston

Example: If the minimum value in a min node is below the max?

No matter what it is, it cannot affect the value of the root node.



Outline

How is done?

First

Traverse the search tree in depth-first order

Second

At each MAX node n, $\alpha(n)$ = the maximum lower bound of possible solutions.

- The α value is changed to t (Coming from the children) if $\alpha < t$.

Third

At each MIN node n, $\beta(n)$ = the minimum upper bound of possible solutions.

- The β value is changed to t (Coming from the children) if $\beta > t$.

How is done?

First

Traverse the search tree in depth-first order

Second

At each MAX node n, $\alpha(n)$ = the maximum lower bound of possible solutions.

- The α value is changed to t (Coming from the children) if $\alpha < t$.

Third

At each MIN node n, $\beta(n)$ = the minimum upper bound of possible solutions.

- The β value is changed to t (Coming from the children) if $\beta > t$.

How is done?

First

Traverse the search tree in depth-first order

Second

At each MAX node n, $\alpha(n)$ = the maximum lower bound of possible solutions.

- The α value is changed to t (Coming from the children) if $\alpha < t$.

Third

At each MIN node n, $\beta(n)$ = the minimum upper bound of possible solutions.

- The β value is changed to t (Coming from the children) if $\beta > t$.

A more graphical view

IMPORTANT

- The α values start at $-\infty$ and only increase.
- The β values start at $+\infty$ and only decrease.

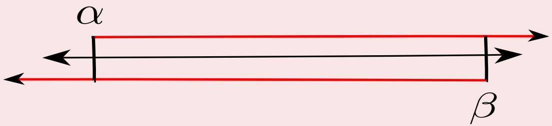
You have intervals where solutions $V(n)$ can happen if

A more graphical view

IMPORTANT

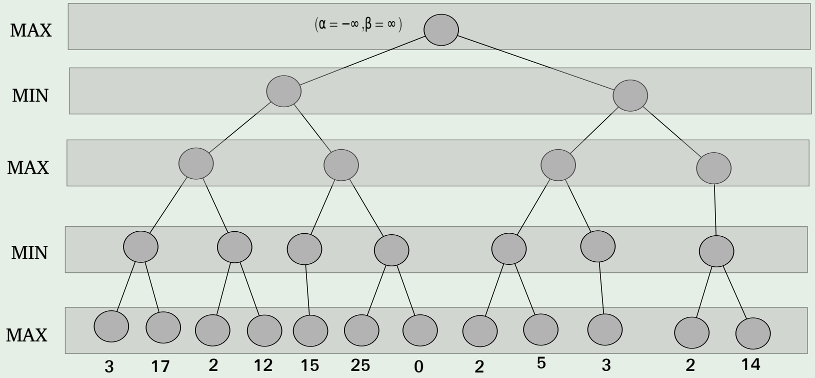
- The α values start at $-\infty$ and only increase.
- The β values start at $+\infty$ and only decrease.

You have intervals where solutions $V(n)$ can happen if



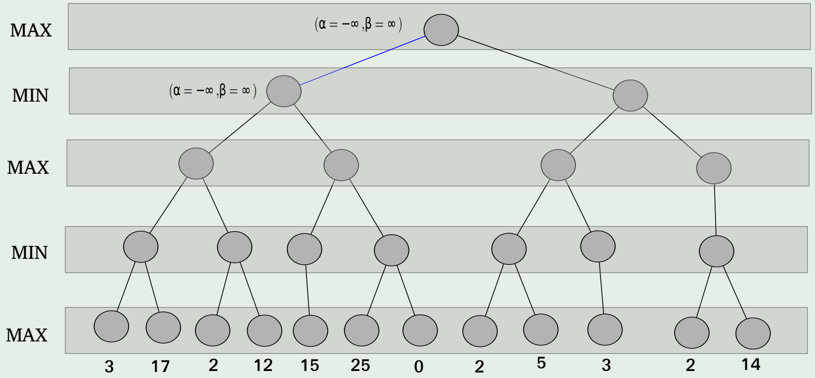
Example

You start with $(\alpha = -\infty, \beta = \infty)$



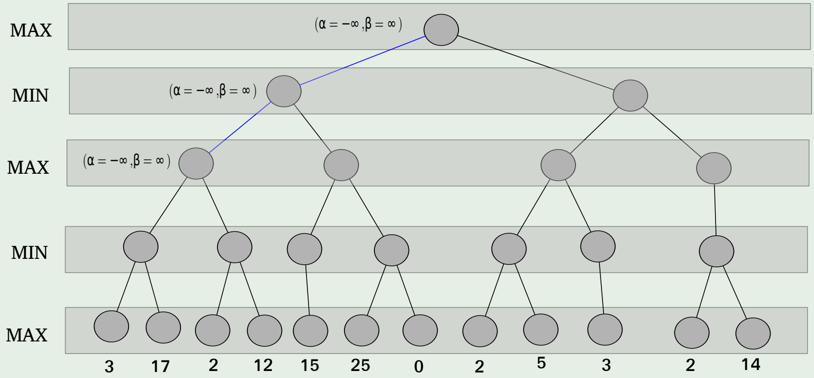
Example

You pass it along the children $(\alpha = -\infty, \beta = \infty)$



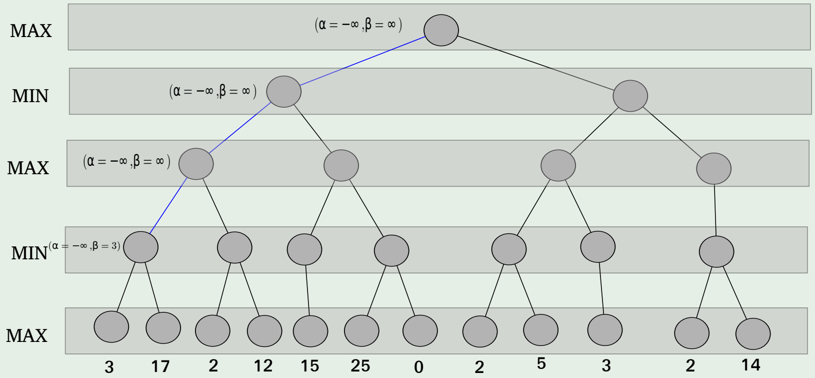
Example

You pass it along the children $(\alpha = -\infty, \beta = \infty)$



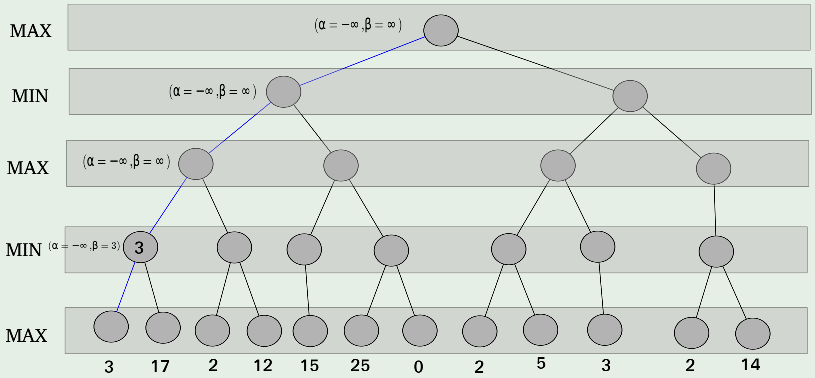
Example

Now you change β because $\infty > 3$



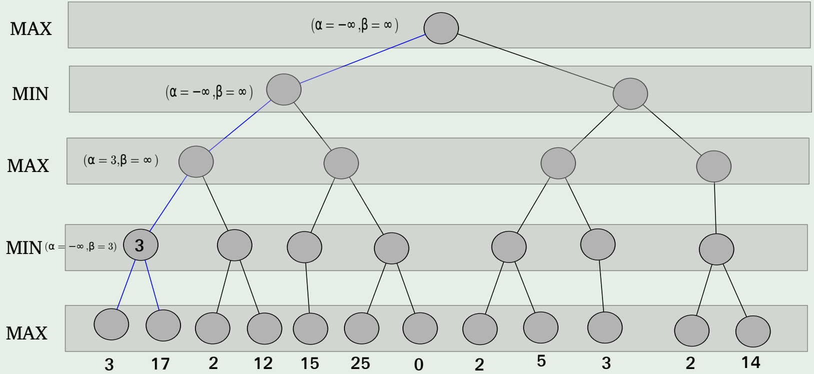
Example

Children with value 17 is ignored because 3<17



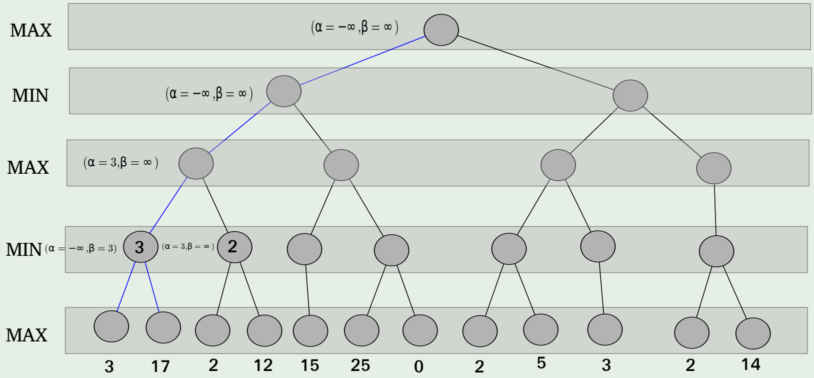
Example

Now, α is changed to 3 because $\alpha < 3$



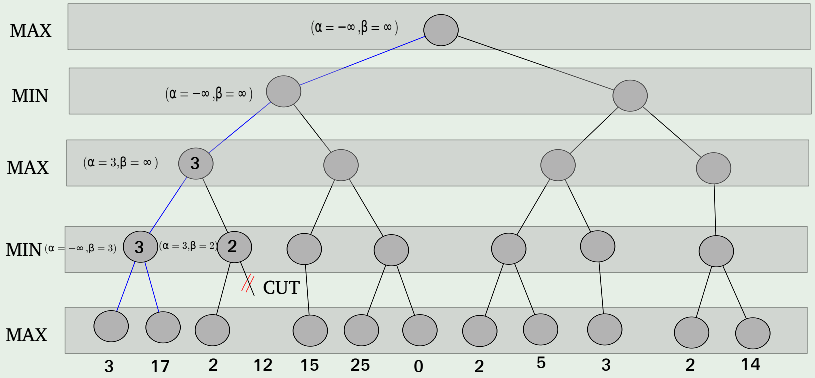
Example

Then $(\alpha = 3, \beta = \infty)$ is moved into the expanded children



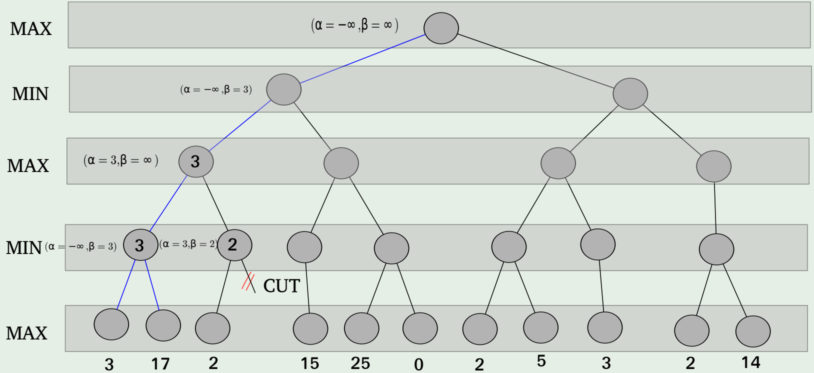
Example

Children with value 12 is removed because $(\alpha = 3, \beta = 2)$



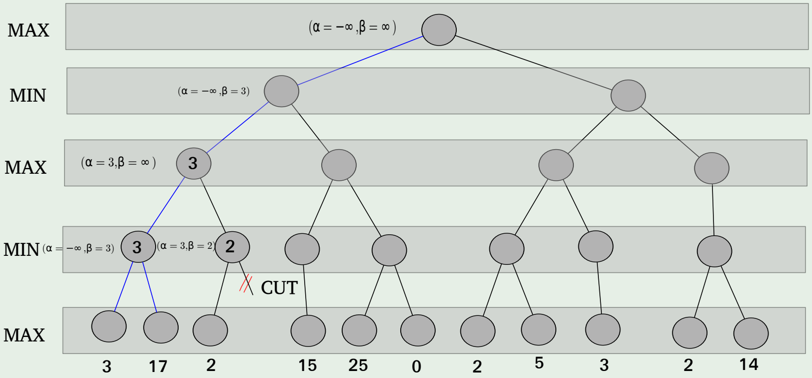
Example

Now, we go up into a parent node



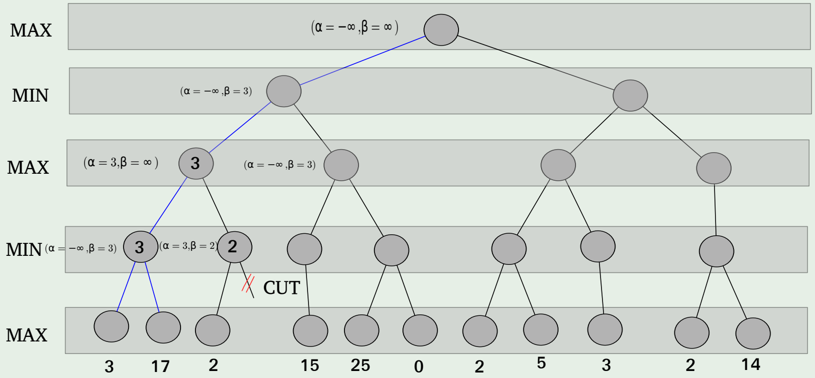
Example

Then $(\alpha = -\infty, \beta = 3)$ is moved down the expanded children



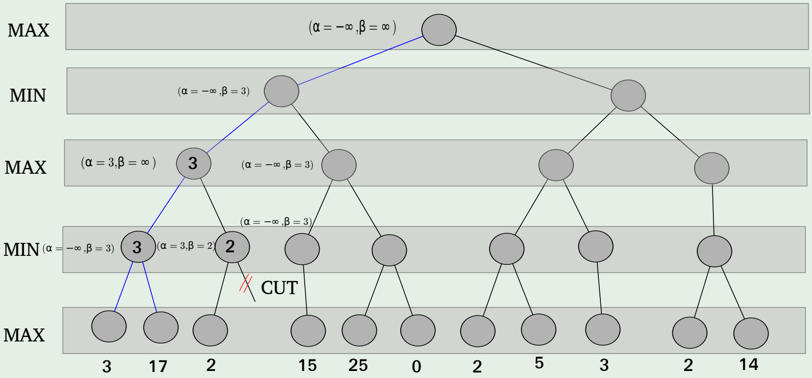
Example

Then $(\alpha = -\infty, \beta = 3)$ is moved down the expanded children



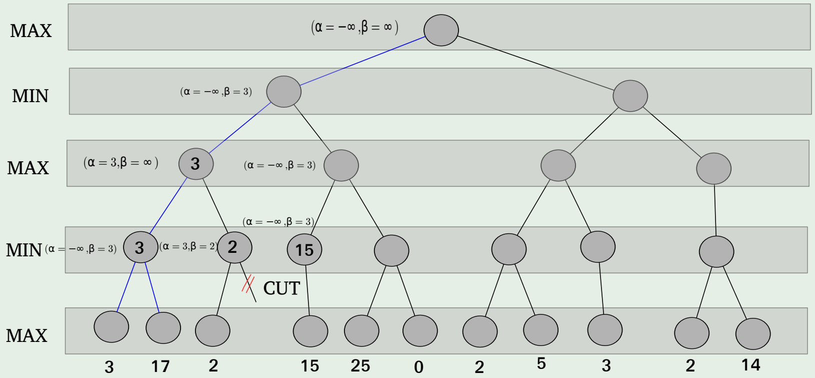
Example

Then $(\alpha = -\infty, \beta = 3)$ is moved down the expanded children



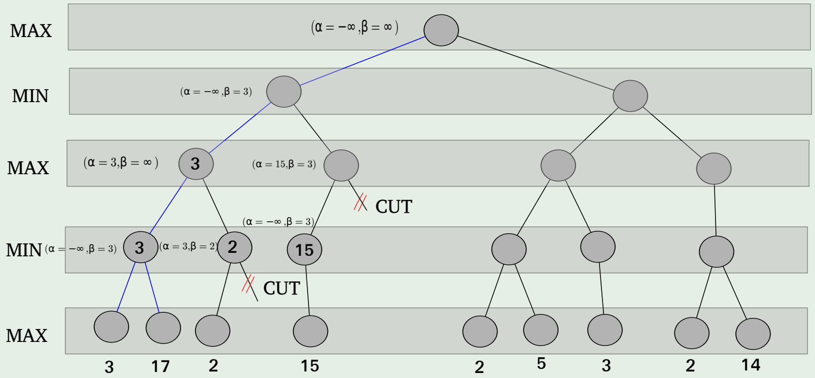
Example

15 goes into the min node



Example

$\alpha$is changed to 15 then cut right children



Algorithm

Procedure MinimaxAlphaBeta(u, α, β)

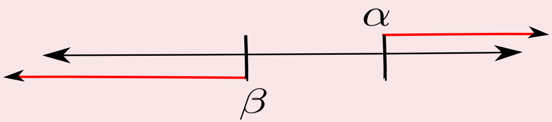
Input: Position u, value α, value β

Output: Value at root

if $(leaf\,(u))$
    return $Eval\,(u)$
if $(max - node\,(u))$
    $res = \alpha$
    for each $v \in Succ\,(u)$
        $val = MinimaxAlphaBeta(v, res, \beta)$
        $res = \max\{res, val\}$
        if $(res \geq \beta)$ **return** res $\Rightarrow$ res **exceeds threshold**
else
    $res = \beta$
    for each $v \in Succ\,(u)$
        $val = MinimaxAlphaBeta(v, \alpha, res)$
        $res = \max\{res, val\}$
        if $(res \leq \alpha)$ **return** res $\Rightarrow$ res **exceeds threshold**

A graphical view of exceeding thresholds

"if $(\alpha \geq \beta)$ return α" or "if $(\beta \leq \alpha)$ return β



Correctness of Minimax Search with $\alpha\beta$-Pruning

Theorem 12.1

Let u be an arbitrary position in a game and $\alpha < \beta$. Then the following three assertions are true.

1. $MinimaxAlphaBeta(u, \alpha, \beta) \leq \alpha$ if and only if $Eval(u) \leq \alpha$.
2. $MinimaxAlphaBeta(u, \alpha, \beta) \geq \beta$ if and only if $Eval(u) \geq \beta$.
3. $\alpha < MinimaxAlphaBeta(u, \alpha, \beta) < \alpha$ if and only if $\alpha < Eval(u) < \beta$.

Effectiveness of $\alpha - \beta$

Something Notable

$\alpha - \beta$ is guaranteed to compute the same value for the root node as computed by minimax, with less or equal computation.

Worst case

No pruning, examining b^{δ} leaf nodes, where each node has b children and a δ-ply search is performed

Best case

It examines only $(2b)^{\delta/2}$ leaf nodes.

- You can search twice as deep as minimax!
- Best case is when each player's best move is the first alternative generated!!!

Effectiveness of $\alpha - \beta$

Something Notable

$\alpha - \beta$ is guaranteed to compute the same value for the root node as computed by minimax, with less or equal computation.

Worst case

No pruning, examining b^{δ} leaf nodes, where each node has b children and a δ-ply search is performed

Best case

It examines only $(2b)^{\delta/2}$ leaf nodes.

- You can search twice as deep as minimax!
- Best case is when each player's best move is the first alternative generated!!!

Effectiveness of $\alpha - \beta$

Something Notable

$\alpha - \beta$ is guaranteed to compute the same value for the root node as computed by minimax, with less or equal computation.

Worst case

No pruning, examining b^{δ} leaf nodes, where each node has b children and a δ-ply search is performed

Best case

It examines only $(2b)^{\delta/2}$ leaf nodes.

- You can search twice as deep as minimax!
- Best case is when each player's best move is the first alternative generated!!!

It is more...

Something Notable

In Deep Blue, they found empirically that alpha-beta pruning meant that the average branching factor at each node was about 6 instead of about 35!

Why is it called $\alpha - \beta$?

First

α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for max.

Then

If v is worse than α, max will avoid it, then prune that branch

Similarly

Define β similarly for min

Why is it called $\alpha - \beta$?

First

α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for max.

Then

If v is worse than α, max will avoid it, then prune that branch

Similarly

Define β similarly for min

Why is it called $\alpha - \beta$?

First

α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for max.

Then

If v is worse than α, max will avoid it, then prune that branch

Similarly

Define β similarly for min

Outline

Problems

Something Notable

Although $\alpha - \beta$ allows to cut the search tree, it still needs to search to a portion of the terminal states.

Thus

Shannon proposed instead (Programming a computer for playing chess 1950) that the programs should cut off early and apply a heuristic evaluation function.

Problems

Something Notable

Although $\alpha - \beta$ allows to cut the search tree, it still needs to search to a portion of the terminal states.

Thus

Shannon proposed instead (Programming a computer for playing chess 1950) that the programs should cut off early and apply a heuristic evaluation function.

Example

Imagine the following

- Suppose we have 100 s
- And we explore 10^4 nodes/second

We need to explore

10^6 nodes per move!!! TOO MUCH

What can we do?

1. Use an **Evaluation Function** = estimated desirability of position
2. Use a Cuttoff test - for example "depth limit"

Example

Imagine the following

- Suppose we have 100 s
- And we explore 10^4 nodes/second

We need to explore

10^6 nodes per move!!! TOO MUCH

What can we do?

1. Use an **Evaluation Function** = estimated desirability of position
2. Use a Cuttoff test - for example "depth limit"

Example

Imagine the following

- Suppose we have 100 s
- And we explore 10^4 nodes/second

We need to explore

10^6 nodes per move!!! TOO MUCH

What can we do?

1. Use an **Evaluation Function** $=$ estimated desirability of position
2. Use a Cuttoff test - for example "depth limit"

Outline

Evaluation functions

Something Notable

Most Evaluations functions work by calculating different features in a equivalence class of states.

For Example

- Experience suggest 72% of the states encountered so far lead to win (utility +1).
- 20% to a loss (utility -1).
- 8% to a draw (utility 0).

Thus

$$f_{eval} = 0.72 \times 1 + 0.2 \times -1 + 0.08 \times 0 = 0.52$$

Evaluation functions

Something Notable

Most Evaluations functions work by calculating different features in a equivalence class of states.

For Example

- Experience suggest 72% of the states encountered so far lead to win (utility $+1$).
- 20% to a loss (utility -1).
- 8% to a draw (utility 0).

Thus

$$f_{eval} = 0.72 \times 1 + 0.2 \times -1 + 0.08 \times 0 = 0.52$$

Evaluation functions

Something Notable

Most Evaluations functions work by calculating different features in a equivalence class of states.

For Example

- Experience suggest 72% of the states encountered so far lead to win (utility $+1$).
- 20% to a loss (utility -1).
- 8% to a draw (utility 0).

Thus

$$f_{eval} = 0.72 \times 1 + 0.2 \times -1 + 0.08 \times 0 = 0.52$$

However...

Something Notable

This type of evaluation functions will force you to know all the categories.

Better

It is better to compute separate numerical functions and combine them

Example Chess

$$f_{eval}(s) = \sum_{i=1}^{n} w_i f_i(s)$$

However...

Something Notable

This type of evaluation functions will force you to know all the categories.

Better

It is better to compute separate numerical functions and combine them

Example Chess

$$f_{eval}(s) = \sum_{i=1}^{n} w_i f_i(s)$$

However...

Something Notable

This type of evaluation functions will force you to know all the categories.

Better

It is better to compute separate numerical functions and combine them

Example Chess

$$f_{eval}(s) = \sum_{i=1}^{n} w_i f_i(s)$$

Example

You could have something like this

$$f_1(s) = \text{(number of white queens)-(number of black queens)}$$

Observations

- Still Eval functions should be applied only to positions unlikely to exhibit wild swings in value.
- Many techniques from Machine Learning can be used when no experience about the problem exist.

Example

You could have something like this

$$f_1(s) = \text{(number of white queens)-(number of black queens)}$$

Observations

- Still Eval functions should be applied only to positions unlikely to exhibit wild swings in value.
- Many techniques from Machine Learning can be used when no experience about the problem exist.

Outline

Example

Backgammon

- Backgammon is a two-player game with uncertainty.
- Players roll dice to determine what moves to make.

Something Notable

Such games are good for exploring decision making in adversarial problems involving skill and luck.

Example

Backgammon

- Backgammon is a two-player game with uncertainty.
- Players roll dice to determine what moves to make.

Something Notable

Such games are good for exploring decision making in adversarial problems involving skill and luck.

How can we handle them?

Chance Nodes

They are used to represent random events

Something Notable

For a random event with N outcomes, each chance node has N distinct children; a probability is associated with each

How can we handle them?

Chance Nodes

They are used to represent random events

Something Notable

For a random event with N outcomes, each chance node has N distinct children; a probability is associated with each

How can we handle them?

Then, it is possible to use a Expected Minimax function

$$ExMinMax(n) = \begin{cases} Utility(n) & \text{if } n \text{ is a terminal state} \\ max_{s \in Succ(n)} \{ExMinMax(s)\} & \text{if } n \text{ is a MAX state} \\ min_{s \in Succ(n)} \{ExMinMax(s)\} & \text{if } n \text{ is a MIN state} \\ \sum_{s \in Succ(n)} Pr(s) ExMinMax(s) & \text{if } n \text{ is a chance state} \end{cases}$$

Outline

Example

We have