



COMP702: Classical Artificial Intelligence

4: Theoretical models for games



Game theory



Game theory

Game theory

- ▶ A branch of mathematics studying **decision making**

Game theory

- ▶ A branch of mathematics studying **decision making**
- ▶ A **game** is a system where one or more **players** choose **actions**; the combination of these choices lead to each agent receiving a **payoff**

Game theory

- ▶ A branch of mathematics studying **decision making**
- ▶ A **game** is a system where one or more **players** choose **actions**; the combination of these choices lead to each agent receiving a **payoff**
- ▶ Important applications in economics, ecology and social sciences as well as AI

The Prisoner's Student's Dilemma

The Prisoner's Student's Dilemma

- ▶ Two students, **Alice** and **Bob**, are suspected of copying from each other

The Prisoner's Student's Dilemma

- ▶ Two students, **Alice** and **Bob**, are suspected of copying from each other
- ▶ Each is offered a deal in exchange for information

The Prisoner's Student's Dilemma

- ▶ Two students, **Alice** and **Bob**, are suspected of copying from each other
- ▶ Each is offered a deal in exchange for information
- ▶ Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do

The Prisoner's Student's Dilemma

- ▶ Two students, **Alice** and **Bob**, are suspected of copying from each other
- ▶ Each is offered a deal in exchange for information
- ▶ Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do
- ▶ If **both stay silent**, both receive a C grade

The Prisoner's Student's Dilemma

- ▶ Two students, **Alice** and **Bob**, are suspected of copying from each other
- ▶ Each is offered a deal in exchange for information
- ▶ Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do
- ▶ If **both stay silent**, both receive a C grade
- ▶ If **Alice betrays Bob**, she receives an A whilst he gets expelled

The Prisoner's Student's Dilemma

- ▶ Two students, **Alice** and **Bob**, are suspected of copying from each other
- ▶ Each is offered a deal in exchange for information
- ▶ Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do
- ▶ If **both stay silent**, both receive a C grade
- ▶ If **Alice betrays Bob**, she receives an A whilst he gets expelled
- ▶ If **Bob betrays Alice**, he receives an A whilst she gets expelled

The Prisoner's Student's Dilemma

- ▶ Two students, **Alice** and **Bob**, are suspected of copying from each other
- ▶ Each is offered a deal in exchange for information
- ▶ Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do
- ▶ If **both stay silent**, both receive a C grade
- ▶ If **Alice betrays Bob**, she receives an A whilst he gets expelled
- ▶ If **Bob betrays Alice**, he receives an A whilst she gets expelled
- ▶ If **both betray each other**, both get an F

Payoff matrix

Payoff matrix

	A silent	A betray
B silent	A: 50 B: 50	A: 70 B: -100
B betray	A: -100 B: 70	A: 0 B: 0

Alice's thought process

Alice's thought process

- The best outcome overall is for both of us to stay silent

Alice's thought process

- ▶ The best outcome overall is for both of us to stay silent
- ▶ However if Bob stays silent, I can get a better mark by betraying him

Alice's thought process

- ▶ The best outcome overall is for both of us to stay silent
- ▶ However if Bob stays silent, I can get a better mark by betraying him
- ▶ And if Bob betrays me, I should betray him to avoid getting expelled

Alice's thought process

- ▶ The best outcome overall is for both of us to stay silent
- ▶ However if Bob stays silent, I can get a better mark by betraying him
- ▶ And if Bob betrays me, I should betray him to avoid getting expelled
- ▶ Therefore the rational choice is to betray

Alice's thought process

- ▶ The best outcome overall is for both of us to stay silent
 - ▶ However if Bob stays silent, I can get a better mark by betraying him
 - ▶ And if Bob betrays me, I should betray him to avoid getting expelled
 - ▶ Therefore the rational choice is to betray
- ... and Bob's thought process is the same!

Nash equilibrium

Nash equilibrium

- ▶ Consider the situation where both have chosen to betray

Nash equilibrium

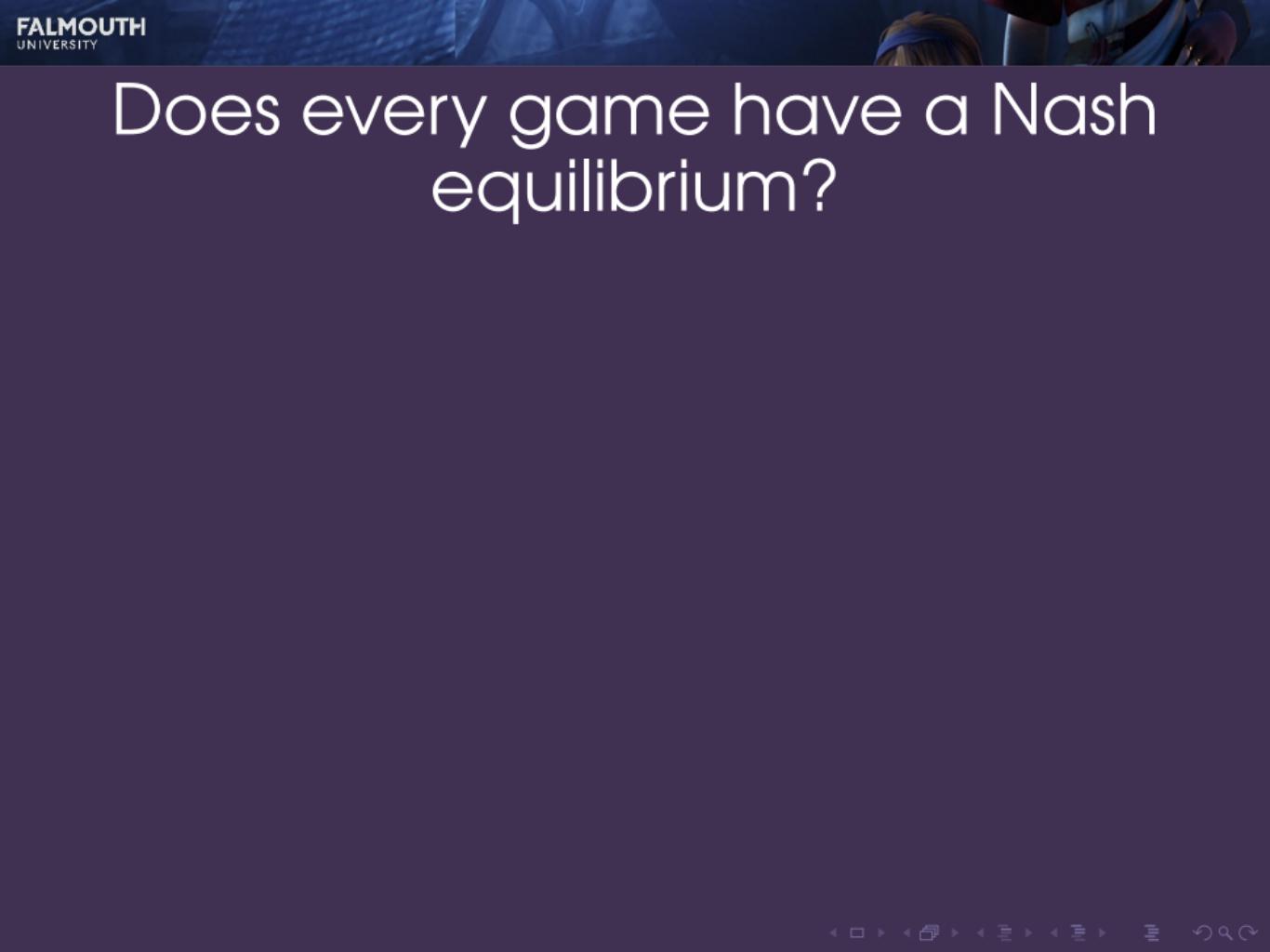
- ▶ Consider the situation where both have chosen to betray
- ▶ Neither person has anything to gain by switching to silence, assuming the other person doesn't also switch

Nash equilibrium

- ▶ Consider the situation where both have chosen to betray
- ▶ Neither person has anything to gain by switching to silence, assuming the other person doesn't also switch
- ▶ Such a situation is called a **Nash equilibrium**

Nash equilibrium

- ▶ Consider the situation where both have chosen to betray
- ▶ Neither person has anything to gain by switching to silence, assuming the other person doesn't also switch
- ▶ Such a situation is called a **Nash equilibrium**
- ▶ If all players are **rational** (in the sense of wanting to maximising payoff), they should converge upon a Nash equilibrium



Does every game have a Nash equilibrium?

Does every game have a Nash equilibrium?

	A rock	A paper	A scissors
B rock	A: 0 B: 0	A: +1 B: -1	A: -1 B: +1
B paper	A: -1 B: +1	A: 0 B: 0	A: +1 B: -1
B scissors	A: +1 B: -1	A: -1 B: +1	A: 0 B: 0

Nash equilibrium for Rock-Paper-Scissors

Nash equilibrium for Rock-Paper-Scissors

- ▶ Committing to any choice of action can be **exploited**

Nash equilibrium for Rock-Paper-Scissors

- ▶ Committing to any choice of action can be **exploited**
- ▶ E.g. if you always choose paper, I choose scissors

Nash equilibrium for Rock-Paper-Scissors

- ▶ Committing to any choice of action can be **exploited**
- ▶ E.g. if you always choose paper, I choose scissors
- ▶ If we try to reason naïvely, we get stuck in a loop

Nash equilibrium for Rock-Paper-Scissors

- ▶ Committing to any choice of action can be **exploited**
- ▶ E.g. if you always choose paper, I choose scissors
- ▶ If we try to reason naïvely, we get stuck in a loop
 - ▶ If I choose paper, you'll choose scissors, so I should choose rock, but then you'll choose paper, so I'll choose scissors, so you'll choose rock, so I choose paper...

Nash equilibrium for Rock-Paper-Scissors

- ▶ Committing to any choice of action can be **exploited**
- ▶ E.g. if you always choose paper, I choose scissors
- ▶ If we try to reason naïvely, we get stuck in a loop
 - ▶ If I choose paper, you'll choose scissors, so I should choose rock, but then you'll choose paper, so I'll choose scissors, so you'll choose rock, so I choose paper...
- ▶ The optimum strategy is to be **unpredictable**

Nash equilibrium for Rock-Paper-Scissors

- ▶ Committing to any choice of action can be **exploited**
- ▶ E.g. if you always choose paper, I choose scissors
- ▶ If we try to reason naïvely, we get stuck in a loop
 - ▶ If I choose paper, you'll choose scissors, so I should choose rock, but then you'll choose paper, so I'll choose scissors, so you'll choose rock, so I choose paper...
- ▶ The optimum strategy is to be **unpredictable**
- ▶ Choose rock with probability $\frac{1}{3}$, paper with probability $\frac{1}{3}$, scissors with probability $\frac{1}{3}$

Mixed strategies

Mixed strategies

- A **mixed strategy** assigns probabilities to actions and chooses one at random

Mixed strategies

- ▶ A **mixed strategy** assigns probabilities to actions and chooses one at random
- ▶ In contrast to a **pure** or **deterministic strategy**, which always chooses the same action

Mixed strategies

- ▶ A **mixed strategy** assigns probabilities to actions and chooses one at random
- ▶ In contrast to a **pure** or **deterministic strategy**, which always chooses the same action
- ▶ If we allow mixed strategies, **every game has at least one Nash equilibrium**



Guess $\frac{2}{3}$ of the average

Guess $\frac{2}{3}$ of the average

- ▶ Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive

Guess $\frac{2}{3}$ of the average

- ▶ Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- ▶ The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses

Guess $\frac{2}{3}$ of the average

- ▶ Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- ▶ The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses
- ▶ Example:

Guess $\frac{2}{3}$ of the average

- ▶ Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- ▶ The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses
- ▶ Example:
 - ▶ If the guesses are 30, 40 and 80...

Guess $\frac{2}{3}$ of the average

- ▶ Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- ▶ The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses
- ▶ Example:
 - ▶ If the guesses are 30, 40 and 80...
 - ▶ ... then the mean is $\frac{30+40+80}{3} = 50...$

Guess $\frac{2}{3}$ of the average

- ▶ Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- ▶ The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses
- ▶ Example:
 - ▶ If the guesses are 30, 40 and 80...
 - ▶ ... then the mean is $\frac{30+40+80}{3} = 50$...
 - ▶ ... so the winning guess is 30, as this is closest to $\frac{2}{3} \times 50 = 33.333$

The rational guess

The rational guess

- The average can't possibly be greater than 100

The rational guess

- ▶ The average can't possibly be greater than 100
- ▶ So no rational player would guess a number greater than 66.666

The rational guess

- ▶ The average can't possibly be greater than 100
- ▶ So no rational player would guess a number greater than 66.666
- ▶ Which means the average can't possibly be greater than 66.666

The rational guess

- ▶ The average can't possibly be greater than 100
- ▶ So no rational player would guess a number greater than 66.666
- ▶ Which means the average can't possibly be greater than 66.666
- ▶ So no rational player would guess greater than 44.444

The rational guess

- ▶ The average can't possibly be greater than 100
- ▶ So no rational player would guess a number greater than 66.666
- ▶ Which means the average can't possibly be greater than 66.666
- ▶ So no rational player would guess greater than 44.444
- ▶ Which means the average can't possibly be greater than 44.444

The rational guess

- ▶ The average can't possibly be greater than 100
- ▶ So no rational player would guess a number greater than 66.666
- ▶ Which means the average can't possibly be greater than 66.666
- ▶ So no rational player would guess greater than 44.444
- ▶ Which means the average can't possibly be greater than 44.444
- ▶ So no rational player would guess greater than 29.629

The rational guess

- ▶ The average can't possibly be greater than 100
- ▶ So no rational player would guess a number greater than 66.666
- ▶ Which means the average can't possibly be greater than 66.666
- ▶ So no rational player would guess greater than 44.444
- ▶ Which means the average can't possibly be greater than 44.444
- ▶ So no rational player would guess greater than 29.629
- ▶ ... and so on ad infinitum

The rational guess

- ▶ The average can't possibly be greater than 100
- ▶ So no rational player would guess a number greater than 66.666
- ▶ Which means the average can't possibly be greater than 66.666
- ▶ So no rational player would guess greater than 44.444
- ▶ Which means the average can't possibly be greater than 44.444
- ▶ So no rational player would guess greater than 29.629
- ▶ ... and so on ad infinitum
- ▶ So the only **rational** guess is 0, as every rational player should guess 0 and $\frac{2}{3}$ of 0 is 0

Rationality

Rationality

- ▶ Rationality is a useful assumption for mathematics and AI programmers

Rationality

- ▶ Rationality is a useful assumption for mathematics and AI programmers
- ▶ However it's important to remember that **humans aren't always rational**

Markov decision processes and games



Markov decision processes

Markov decision processes

- ▶ A **Markov decision process (MDP)** is defined by:

Markov decision processes

- ▶ A **Markov decision process (MDP)** is defined by:
 - ▶ A finite set S of **states**;

Markov decision processes

- ▶ A **Markov decision process (MDP)** is defined by:
 - ▶ A finite set S of **states**;
 - ▶ A finite set A of **actions**;

Markov decision processes

- ▶ A **Markov decision process (MDP)** is defined by:
 - ▶ A finite set S of **states**;
 - ▶ A finite set A of **actions**;
 - ▶ $P(s, a, s')$ is the **probability** that action a in state s leads to state s' ;

Markov decision processes

- ▶ A **Markov decision process (MDP)** is defined by:
 - ▶ A finite set S of **states**;
 - ▶ A finite set A of **actions**;
 - ▶ $P(s, a, s')$ is the **probability** that action a in state s leads to state s' ;
 - ▶ $R(s, a, s')$ is the **reward** received from performing action a in state s and ending up in state s' .

The Markov property

The Markov property

- ▶ Given a state s and an action a , the next state s' is determined by $P(s, a, s')$

The Markov property

- ▶ Given a state s and an action a , the next state s' is determined by $P(s, a, s')$
- ▶ The previous states before s have no effect

The Markov property

- ▶ Given a state s and an action a , the next state s' is determined by $P(s, a, s')$
- ▶ The previous states before s have no effect
- ▶ Hence an MDP is “memoryless”

The Markov property

- ▶ Given a state s and an action a , the next state s' is determined by $P(s, a, s')$
- ▶ The previous states before s have no effect
- ▶ Hence an MDP is “memoryless”
- ▶ (Or rather, any memory has to be contained within the state)

(Non)determinism

(Non)determinism

- MDP defines $P(s, a, s')$

(Non)determinism

- ▶ MDP defines $P(s, a, s')$
- ▶ From state s and action a , there may be several possible states s'

(Non)determinism

- ▶ MDP defines $P(s, a, s')$
- ▶ From state s and action a , there may be several possible states s'
- ▶ MDPs are **nondeterministic** i.e. **stochastic**

(Non)determinism

- ▶ MDP defines $P(s, a, s')$
- ▶ From state s and action a , there may be several possible states s'
- ▶ MDPs are **nondeterministic** i.e. **stochastic**
- ▶ If we have

$$P(s, a, s') = \begin{cases} 1 & \text{for some } s' \\ 0 & \text{for all other } s' \end{cases}$$

then the MDP is **deterministic**

(Non)determinism

- ▶ MDP defines $P(s, a, s')$
- ▶ From state s and action a , there may be several possible states s'
- ▶ MDPs are **nondeterministic** i.e. **stochastic**
- ▶ If we have

$$P(s, a, s') = \begin{cases} 1 & \text{for some } s' \\ 0 & \text{for all other } s' \end{cases}$$

then the MDP is **deterministic**

- ▶ In the deterministic case, the same state s and action a always leads to the same state s'

Markov decision problems

Markov decision problems

- ▶ A **policy** for an MDP is a function $\pi : S \rightarrow A$

Markov decision problems

- ▶ A **policy** for an MDP is a function $\pi : S \rightarrow A$
- ▶ I.e. in state s , choose action $a = \pi(s)$

Markov decision problems

- ▶ A **policy** for an MDP is a function $\pi : S \rightarrow A$
- ▶ I.e. in state s , choose action $a = \pi(s)$
- ▶ Goal: find π which maximises the total reward over time

Multi-agent MDPs

Multi-agent MDPs

- We have assumed a **single agent** choosing a policy π

Multi-agent MDPs

- ▶ We have assumed a **single agent** choosing a policy π
- ▶ We can extend to n agents choosing policies π_1, \dots, π_n

Multi-agent MDPs

- ▶ We have assumed a **single agent** choosing a policy π
- ▶ We can extend to n agents choosing policies π_1, \dots, π_n
- ▶ The “action” is now a combination (a_1, \dots, a_n) of all the agents’ actions

Multi-agent MDPs

- ▶ We have assumed a **single agent** choosing a policy π
- ▶ We can extend to n agents choosing policies π_1, \dots, π_n
- ▶ The “action” is now a combination (a_1, \dots, a_n) of all the agents’ actions
- ▶ Each agent has their own reward function and is trying to maximise it

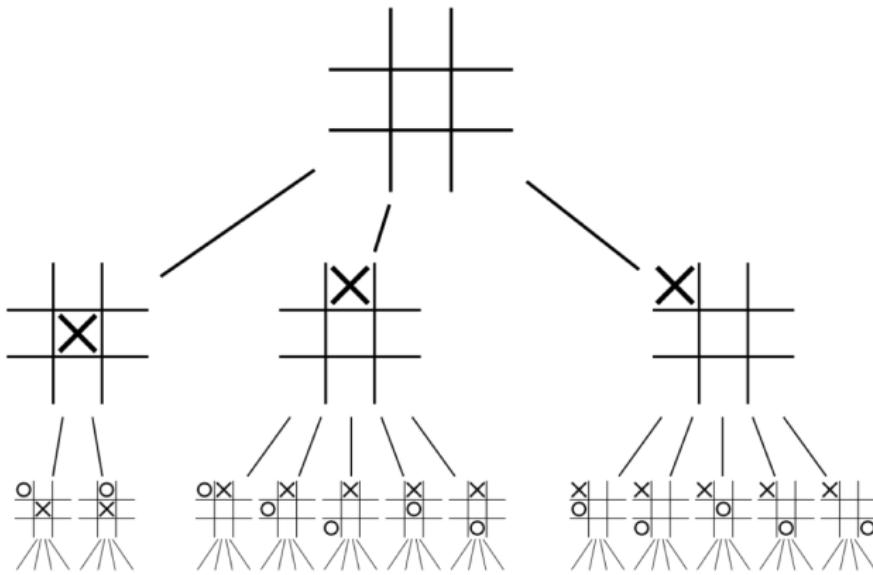
Multi-agent MDPs

- ▶ We have assumed a **single agent** choosing a policy π
- ▶ We can extend to n agents choosing policies π_1, \dots, π_n
- ▶ The “action” is now a combination (a_1, \dots, a_n) of all the agents’ actions
- ▶ Each agent has their own reward function and is trying to maximise it
- ▶ This is a **game!**

Minimax search



Game trees



Minimax

Minimax

- ▶ Consider a 2-agent MDP with the following restrictions:

Minimax

- ▶ Consider a 2-agent MDP with the following restrictions:
 - ▶ Deterministic

Minimax

- ▶ Consider a 2-agent MDP with the following restrictions:
 - ▶ Deterministic
 - ▶ Only one agent chooses an action from a given state

Minimax

- ▶ Consider a 2-agent MDP with the following restrictions:
 - ▶ Deterministic
 - ▶ Only one agent chooses an action from a given state
 - ▶ Some states are **terminal** — these are the only ones with non-zero reward

Minimax

- ▶ Consider a 2-agent MDP with the following restrictions:
 - ▶ Deterministic
 - ▶ Only one agent chooses an action from a given state
 - ▶ Some states are **terminal** — these are the only ones with non-zero reward
 - ▶ The game is **zero sum**: $R_1(s, a, s') + R_2(s, a, s') = 0$

Minimax

- ▶ Consider a 2-agent MDP with the following restrictions:
 - ▶ Deterministic
 - ▶ Only one agent chooses an action from a given state
 - ▶ Some states are **terminal** — these are the only ones with non-zero reward
 - ▶ The game is **zero sum**: $R_1(s, a, s') + R_2(s, a, s') = 0$
- ▶ I want to **maximise** my reward

Minimax

- ▶ Consider a 2-agent MDP with the following restrictions:
 - ▶ Deterministic
 - ▶ Only one agent chooses an action from a given state
 - ▶ Some states are **terminal** — these are the only ones with non-zero reward
 - ▶ The game is **zero sum**: $R_1(s, a, s') + R_2(s, a, s') = 0$
- ▶ I want to **maximise** my reward
- ▶ My opponent wants to maximise their reward, which is the same as **minimise** my reward

Minimax

- ▶ Consider a 2-agent MDP with the following restrictions:
 - ▶ Deterministic
 - ▶ Only one agent chooses an action from a given state
 - ▶ Some states are **terminal** — these are the only ones with non-zero reward
 - ▶ The game is **zero sum**: $R_1(s, a, s') + R_2(s, a, s') = 0$
- ▶ I want to **maximise** my reward
- ▶ My opponent wants to maximise their reward, which is the same as **minimise** my reward
- ▶ Therefore I want to **maximise** the **minimum** value my opponent can achieve

Minimax search

Minimax search

- ▶ Recursively defines a **value** for non-terminal game states

Minimax search

- ▶ Recursively defines a **value** for non-terminal game states
- ▶ Consider each possible “next state”, i.e. each possible move

Minimax search

- ▶ Recursively defines a **value** for non-terminal game states
- ▶ Consider each possible “next state”, i.e. each possible move
- ▶ If it’s my turn, the value is the **maximum** value over next states

Minimax search

- ▶ Recursively defines a **value** for non-terminal game states
- ▶ Consider each possible “next state”, i.e. each possible move
- ▶ If it’s my turn, the value is the **maximum** value over next states
- ▶ If it’s my opponent’s turn, the value is the **minimum** value over next states

Minimax search – example

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer = 1 then
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer = 1 then
        bestValue = -∞
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer = 1 then
        bestValue = -∞
        for each possible nextState do
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer = 1 then
        bestValue = -∞
        for each possible nextState do
            v = MINIMAX(nextState, 3 – currentPlayer)
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer = 1 then
        bestValue = -∞
        for each possible nextState do
            v = MINIMAX(nextState, 3 – currentPlayer)
            bestValue = Max(bestValue, v)
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer = 1 then
        bestValue = -∞
        for each possible nextState do
            v = MINIMAX(nextState, 3 – currentPlayer)
            bestValue = MAX(bestValue, v)
        return bestValue
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer = 1 then
        bestValue = -∞
        for each possible nextState do
            v = MINIMAX(nextState, 3 – currentPlayer)
            bestValue = MAX(bestValue, v)
        return bestValue
    else if currentPlayer = 2 then
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer = 1 then
        bestValue = -∞
        for each possible nextState do
            v = MINIMAX(nextState, 3 – currentPlayer)
            bestValue = MAX(bestValue, v)
        return bestValue
    else if currentPlayer = 2 then
        bestValue = +∞
        for each possible nextState do
            v = MINIMAX(nextState, 3 – currentPlayer)
            bestValue = MIN(bestValue, v)
        return bestValue
```

Minimax search pseudocode

```
procedure MINIMAX(state, currentPlayer)
    if state is terminal then
        return value of state
    else if currentPlayer = 1 then
        bestValue = -∞
        for each possible nextState do
            v = MINIMAX(nextState, 3 – currentPlayer)
            bestValue = MAX(bestValue, v)
        return bestValue
    else if currentPlayer = 2 then
        bestValue = +∞
        for each possible nextState do
            v = MINIMAX(nextState, 3 – currentPlayer)
            bestValue = MIN(bestValue, v)
        return bestValue
```

Stopping early

for each possible nextState **do**

```
v = MINIMAX(nextState, 3 – currentPlayer)  
bestValue = MAX(bestValue, v)
```

Stopping early

for each possible nextState **do**

```
v = MINIMAX(nextState, 3 – currentPlayer)  
bestValue = MAX(bestValue, v)
```

- ▶ State values are always between –1 and +1

Stopping early

for each possible nextState **do**

$v = \text{MINIMAX}(\text{nextState}, 3 - \text{currentPlayer})$
 $\text{bestValue} = \text{MAX}(\text{bestValue}, v)$

- ▶ State values are always between -1 and $+1$
- ▶ So if we ever have $\text{bestValue} = 1$, we can stop early

Stopping early

for each possible nextState **do**

```
v = MINIMAX(nextState, 3 – currentPlayer)  
bestValue = MAX(bestValue, v)
```

- ▶ State values are always between -1 and $+1$
- ▶ So if we ever have $\text{bestValue} = 1$, we can stop early
- ▶ Similarly when minimising if $\text{bestValue} = -1$

Stopping early

for each possible nextState **do**

```
v = MINIMAX(nextState, 3 - currentPlayer)  
bestValue = MAX(bestValue, v)
```

- ▶ State values are always between -1 and $+1$
- ▶ So if we ever have $\text{bestValue} = 1$, we can stop early
- ▶ Similarly when minimising if $\text{bestValue} = -1$
- ▶ There are techniques for smarter early stopping, e.g.
alpha-beta pruning

Using minimax search

Using minimax search

- To decide what move to play next...

Using minimax search

- ▶ To decide what move to play next...
- ▶ Calculate the minimax value for each move

Using minimax search

- ▶ To decide what move to play next...
- ▶ Calculate the minimax value for each move
- ▶ Choose the move with the maximum score

Using minimax search

- ▶ To decide what move to play next...
- ▶ Calculate the minimax value for each move
- ▶ Choose the move with the maximum score
- ▶ If there are several with the same score, choose one at random

Minimax and game theory

Minimax and game theory

- ▶ For a **two-player zero-sum** game with **perfect information** and **sequential moves**

Minimax and game theory

- ▶ For a **two-player zero-sum** game with **perfect information** and **sequential moves**
- ▶ Minimax search will always find a **Nash equilibrium**

Minimax and game theory

- ▶ For a **two-player zero-sum** game with **perfect information** and **sequential moves**
- ▶ Minimax search will always find a **Nash equilibrium**
- ▶ I.e. a minimax player plays **perfectly**

Minimax and game theory

- ▶ For a **two-player zero-sum** game with **perfect information** and **sequential moves**
- ▶ Minimax search will always find a **Nash equilibrium**
- ▶ I.e. a minimax player plays **perfectly**
- ▶ **But...**

Minimax for larger games

Minimax for larger games

- The game tree for noughts and crosses has only a few thousand states

Minimax for larger games

- ▶ The game tree for noughts and crosses has only a few thousand states
- ▶ Most games are too large to search fully

Minimax for larger games

- ▶ The game tree for noughts and crosses has only a few thousand states
- ▶ Most games are too large to search fully
 - ▶ Connect 4 has $\approx 10^{13}$ states

Minimax for larger games

- ▶ The game tree for noughts and crosses has only a few thousand states
- ▶ Most games are too large to search fully
 - ▶ Connect 4 has $\approx 10^{13}$ states
 - ▶ Chess has $\approx 10^{47}$ states

Minimax for larger games

- ▶ The game tree for noughts and crosses has only a few thousand states
- ▶ Most games are too large to search fully
 - ▶ Connect 4 has $\approx 10^{13}$ states
 - ▶ Chess has $\approx 10^{47}$ states
- ▶ Generally need to **cut off** the tree at a certain depth and use **heuristics**

Minimax for larger games

- ▶ The game tree for noughts and crosses has only a few thousand states
- ▶ Most games are too large to search fully
 - ▶ Connect 4 has $\approx 10^{13}$ states
 - ▶ Chess has $\approx 10^{47}$ states
- ▶ Generally need to **cut off** the tree at a certain depth and use **heuristics**
- ▶ Basically, define our own reward function that (hopefully) approximates the real one

Application to games



Application to board games

Application to board games

- The ideas of MDPs and game theory apply readily to board games

Application to board games

- ▶ The ideas of MDPs and game theory apply readily to board games
- ▶ State = state of the board

Application to board games

- ▶ The ideas of MDPs and game theory apply readily to board games
- ▶ State = state of the board
- ▶ Action = a move

Application to board games

- ▶ The ideas of MDPs and game theory apply readily to board games
- ▶ State = state of the board
- ▶ Action = a move
- ▶ Reward = win or loss, score, etc.

Application to video games

Application to video games

- *Technically* a video game is an MDP

Application to video games

- ▶ *Technically* a video game is an MDP
 - ▶ State = entire state of the game (memory?)

Application to video games

- ▶ *Technically* a video game is an MDP
 - ▶ State = entire state of the game (memory?)
 - ▶ Action = controller input on each frame

Application to video games

- ▶ *Technically* a video game is an MDP
 - ▶ State = entire state of the game (memory?)
 - ▶ Action = controller input on each frame
- ▶ However this is intractable

Application to video games

- ▶ *Technically* a video game is an MDP
 - ▶ State = entire state of the game (memory?)
 - ▶ Action = controller input on each frame
- ▶ However this is intractable
- ▶ Generally necessary to **abstract** parts out of the game to treat as MDPs / game theory games