Games

Multi-agent interaction

Up to this point:

This week:

Multi-agent interaction

Up to this point:

- Stochastic Uncertainty

This week:

Multi-agent interaction

Up to this point:

- Stochastic Uncertainty
- Optimization

This week:

Multi-agent interaction

Up to this point:

- Stochastic Uncertainty
- Optimization

This week:

- Interaction Uncertainty

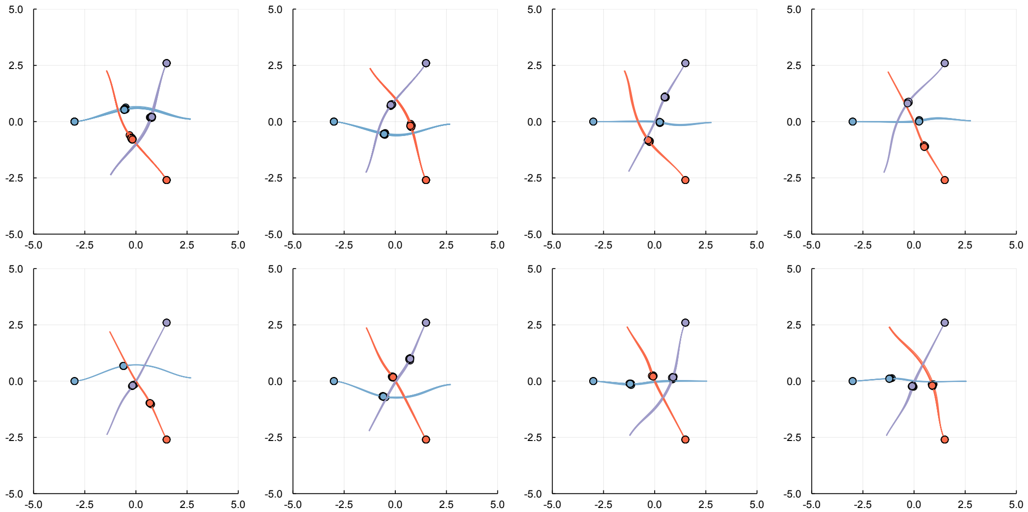
Multi-agent interaction

Up to this point:

- Stochastic Uncertainty
- Optimization

This week:

- Interaction Uncertainty
- Game Solution Concepts



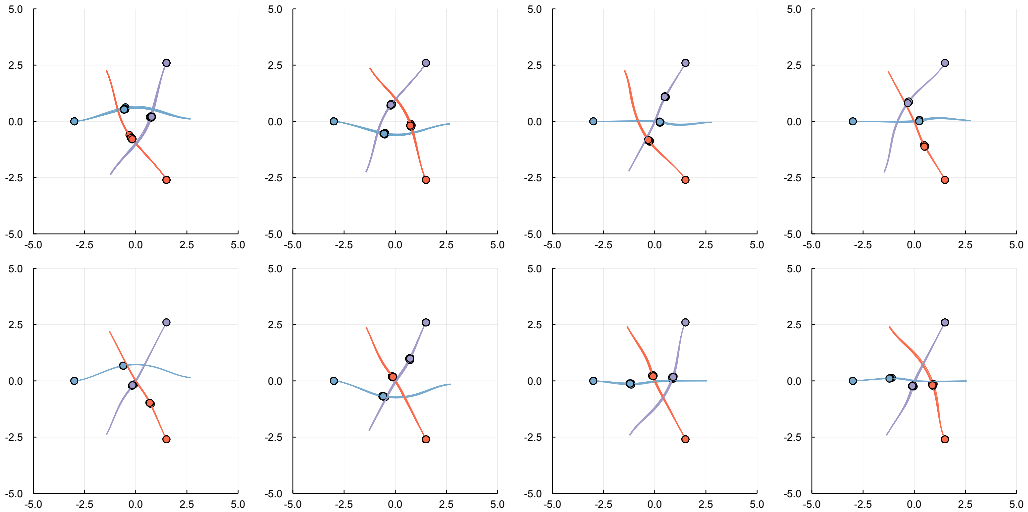
Multi-agent interaction

Up to this point:

- Stochastic Uncertainty
- Optimization

This week:

- Interaction Uncertainty
- Game Solution Concepts



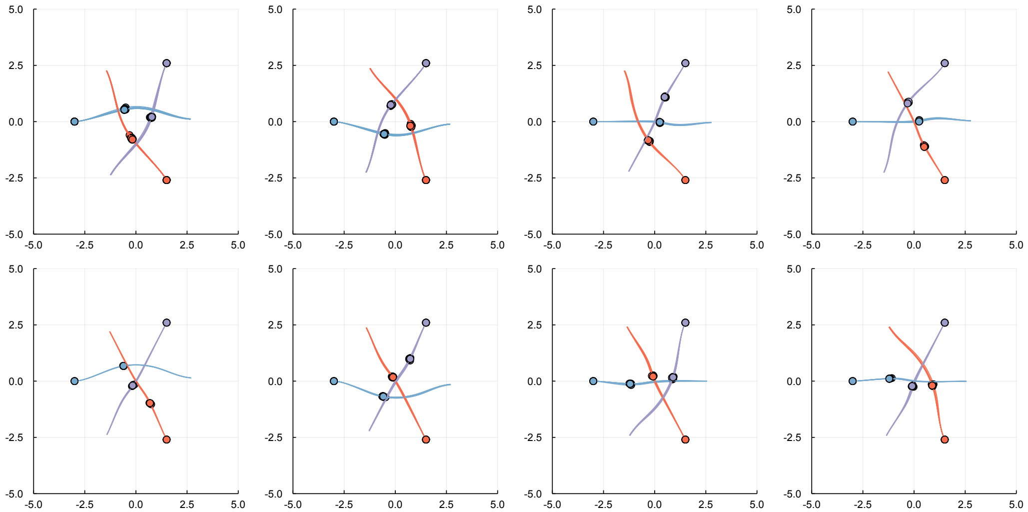
Multi-agent interaction

Up to this point:

- Stochastic Uncertainty
- Optimization

This week:

- Interaction Uncertainty
- Game Solution Concepts



Kriegspiel

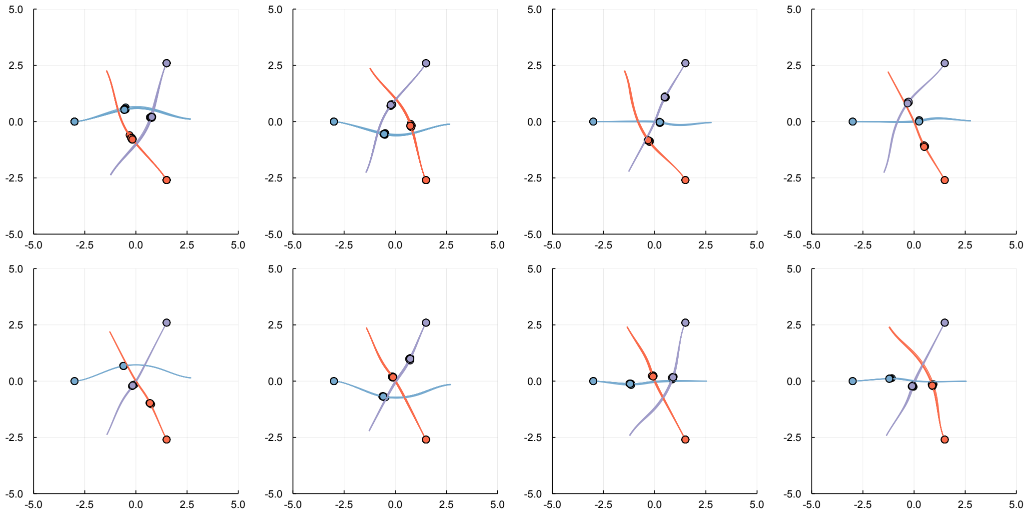
Multi-agent interaction

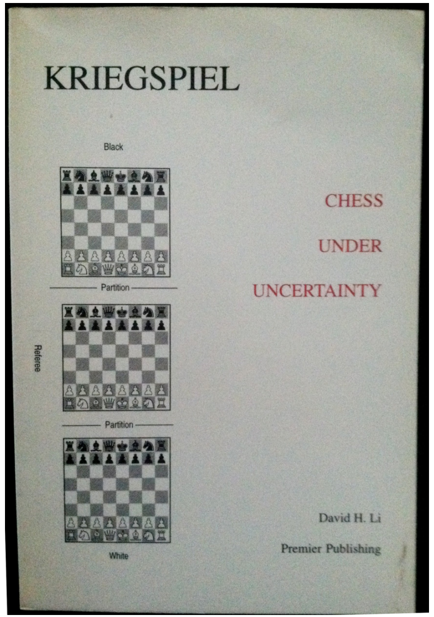
Up to this point:

- Stochastic Uncertainty
- Optimization

This week:

- Interaction Uncertainty
- Game Solution Concepts





Kriegspiel

Outline

Outline

1. Matrix Games
    1. Linear Programming
    2. Regret Matching

Outline

1. Matrix Games
    1. Linear Programming
    2. Regret Matching
2. Turn-taking Games
    1. Minimax Trees
    2. α-β Pruning
    3. Monte Carlo Tree Search

Outline

1. Matrix Games
    1. Linear Programming
    2. Regret Matching
2. Turn-taking Games
    1. Minimax Trees
    2. α-β Pruning
    3. Monte Carlo Tree Search
3. Imperfect Information Games
    1. Extensive-form Games
    2. Counterfactual Regret Minimization

Outline

1. Matrix Games
    1. Linear Programming
    2. Regret Matching
2. Turn-taking Games
    1. Minimax Trees
    2. α-β Pruning
    3. Monte Carlo Tree Search
3. Imperfect Information Games
    1. Extensive-form Games
    2. Counterfactual Regret Minimization

Sources:

- *AI: A Modern Approach* by Russel and Norvig
- "An Introduction to Counterfactual Regret Minimization in Games" by Neller and Lanctot

Game Solution Concepts: Nash and Correlated Equilibria

Game Solution Concepts: Nash and Correlated Equilibria

Rock Paper Scissors

Game Solution Concepts: Nash and Correlated Equilibria

Rock Paper Scissors

	R	P	S
R	$0, 0$	$-1, 1$	$1, -1$
P	$1, -1$	$0, 0$	$-1, 1$
S	$-1, 1$	$1, -1$	$0, 0$

Game Solution Concepts: Nash and Correlated Equilibria

Rock Paper Scissors

	R	P	S
R	$0,0$	$-1,1$	$1,-1$
P	$1,-1$	$0,0$	$-1,1$
S	$-1,1$	$1,-1$	$0,0$

Battle of the Sexes

Gary

Monica	M	G
M	$2,1$	$0,0$
G	$0,0$	$1,2$

Game Solution Concepts: Nash and Correlated Equilibria

Rock Paper Scissors

	R	P	S
R	$0, 0$	$-1, 1$	$1, -1$
P	$1, -1$	$0, 0$	$-1, 1$
S	$-1, 1$	$1, -1$	$0, 0$

Battle of the Sexes

Gary

Monica

	M	G
M	$2, 1$	$0, 0$
G	$0, 0$	$1, 2$

Definitions:

Game Solution Concepts: Nash and Correlated Equilibria

Rock Paper Scissors

	R	P	S
R	$0, 0$	$-1, 1$	$1, -1$
P	$1, -1$	$0, 0$	$-1, 1$
S	$-1, 1$	$1, -1$	$0, 0$

Battle of the Sexes

Gary

Monica		M	G
	M	$2, 1$	$0, 0$
	G	$0, 0$	$1, 2$

Definitions:

- Strategy/Policy: Probability distribution of actions for a player

Game Solution Concepts: Nash and Correlated Equilibria

Rock Paper Scissors

	R	P	S
R	$0,0$	$-1,1$	$1,-1$
P	$1,-1$	$0,0$	$-1,1$
S	$-1,1$	$1,-1$	$0,0$

Battle of the Sexes

Gary

Monica	M	G
M	$2,1$	$0,0$
G	$0,0$	$1,2$

Definitions:

- Strategy/Policy: Probability distribution of actions for a player
- Strategy Profile: Set of strategies for all players

Game Solution Concepts: Nash and Correlated Equilibria

Rock Paper Scissors

	R	P	S
R	$0,0$	$-1,1$	$1,-1$
P	$1,-1$	$0,0$	$-1,1$
S	$-1,1$	$1,-1$	$0,0$

Battle of the Sexes

Gary

Monica	M	G
M	$2,1$	$0,0$
G	$0,0$	$1,2$

Definitions:

- Strategy/Policy: Probability distribution of actions for a player
- Strategy Profile: Set of strategies for all players
- Nash Equilibrium: A strategy where there is no incentive for any player to unilaterally change strategy

Game Solution Concepts: Nash and Correlated Equilibria

Rock Paper Scissors

	R	P	S
R	$0, 0$	$-1, 1$	$1, -1$
P	$1, -1$	$0, 0$	$-1, 1$
S	$-1, 1$	$1, -1$	$0, 0$

Battle of the Sexes

Gary

Monica	M	G
M	$2, 1$	$0, 0$
G	$0, 0$	$1, 2$

Definitions:

- Strategy/Policy: Probability distribution of actions for a player
- Strategy Profile: Set of strategies for all players
- Nash Equilibrium: A strategy where there is no incentive for any player to unilaterally change strategy
- Correlated Equilibrium: Strategy profile with a shared random source

Calculating Mixed-Strategy Equilibria

Calculating Mixed-Strategy Equilibria

Gary

Monica

	M	G
M	$2,1$	$0,0$
G	$0,0$	$1,2$

Calculating Mixed-Strategy Equilibria

Gary

Monica		M	G
	M	$2,1$	$0,0$
	G	$0,0$	$1,2$

By hand

Calculating Mixed-Strategy Equilibria

Gary

		M	G
Monica	M	$2, 1$	$0, 0$
	G	$0, 0$	$1, 2$

By hand

$$\sigma_{\text{Gary}}(M) = x$$

Calculating Mixed-Strategy Equilibria

Gary

		M	G
Monica	M	$2, 1$	$0, 0$
	G	$0, 0$	$1, 2$

<u>By hand</u>

$$\sigma_{\text{Gary}}(M) = x$$

$$U_{\text{Monica}}(\sigma_{\text{Gary}}, M) = 2x,$$

$$U_{\text{Monica}}(\sigma_{\text{Gary}}, G) = 1 - x$$

Calculating Mixed-Strategy Equilibria

Gary

		M	G
Monica	M	$2, 1$	$0, 0$
	G	$0, 0$	$1, 2$

<u>By hand</u>

$$\sigma_{\text{Gary}}(M) = x$$

$$U_{\text{Monica}}(\sigma_{\text{Gary}}, M) = 2x,$$

$$U_{\text{Monica}}(\sigma_{\text{Gary}}, G) = 1 - x$$

If $x = 1/3$, these are equal, so Monica is indifferent. By symmetry, $\sigma = \{(\frac{2}{3}, \frac{1}{3}), (\frac{1}{3}, \frac{2}{3}\}$ is a Nash Equilibrium.

Calculating Mixed-Strategy Equilibria

		Gary	
		M	G
Monica	M	$2, 1$	$0, 0$
	G	$0, 0$	$1, 2$

$$\begin{aligned} \underset{\boldsymbol{\pi}, U}{\text{minimize}} \quad & \sum_i \left(U^i - U^i(\boldsymbol{\pi}) \right) \\ \text{subject to} \quad & U^i \geq U^i(a^i, \boldsymbol{\pi}^{-i}) \text{ for all } i, a^i \\ & \sum_{a^i} \pi^i(a^i) = 1 \text{ for all } i \\ & \pi^i(a^i) \geq 0 \text{ for all } i, a^i \end{aligned}$$

<u>By hand</u>

$$\sigma_{\text{Gary}}(M) = x$$

$$U_{\text{Monica}}(\sigma_{\text{Gary}}, M) = 2x,$$

$$U_{\text{Monica}}(\sigma_{\text{Gary}}, G) = 1 - x$$

If $x = 1/3$, these are equal, so Monica is indifferent. By symmetry, $\sigma = \{(\frac{2}{3}, \frac{1}{3}), (\frac{1}{3}, \frac{2}{3}\}$ is a Nash Equilibrium.

Regret Matching

Regret Matching

For action profile $\mathbf{a}$, player i's *regret* for not playing action a' is:

$$R(a_i') = \ U(a_i', a_{-i}) - U(\mathbf{a})$$

that is, the utility missed out on by not playing a'.

Positive regret means that action would have been better.

Regret Matching

For action profile $\mathbf{a}$, player i's *regret* for not playing action a' is:

$$R(a_i') = \ U(a_i', a_{-i}) - U(\mathbf{a})$$

that is, the utility missed out on by not playing a'.

Positive regret means that action would have been better.

Regret Matching:

For some number of iterations:

Regret Matching

For action profile $\mathbf{a}$, player i's *regret* for not playing action a' is:

$$R(a'_i) = \ U(a'_i, a_{-i}) - U(\mathbf{a})$$

that is, the utility missed out on by not playing a'.

Positive regret means that action would have been better.

Regret Matching:

For some number of iterations:

1. Compute a regret matching profile ($\sigma \propto \text{normalized cumulative regret}$)

Regret Matching

For action profile $\mathbf{a}$, player i's *regret* for not playing action a' is:

$$R(a'_i) = \ U(a'_i, a_{-i}) - U(\mathbf{a})$$

that is, the utility missed out on by not playing a'.

Positive regret means that action would have been better.

Regret Matching:

For some number of iterations:

1. Compute a regret matching profile ($\sigma \propto \text{normalized cumulative regret}$)
2. Add strategy profile to profile sum

Regret Matching

For action profile $\mathbf{a}$, player i's *regret* for not playing action a' is:

$$R(a_i') = \ U(a_i', a_{-i}) - U(\mathbf{a})$$

that is, the utility missed out on by not playing a'.

Positive regret means that action would have been better.

Regret Matching:

For some number of iterations:

1. Compute a regret matching profile ($\sigma \propto \text{normalized cumulative regret}$)
2. Add strategy profile to profile sum
3. Select an action

Regret Matching

For action profile $\mathbf{a}$, player i's *regret* for not playing action a' is:

$$R(a_i') = \ U(a_i', a_{-i}) - U(\mathbf{a})$$

that is, the utility missed out on by not playing a'.

Positive regret means that action would have been better.

Regret Matching:

For some number of iterations:

1. Compute a regret matching profile ($\sigma \propto \text{normalized cumulative regret}$)
2. Add strategy profile to profile sum
3. Select an action
4. Compute regrets

Regret Matching

For action profile $\mathbf{a}$, player i's *regret* for not playing action a' is:

$$R(a'_i) = U(a'_i, a_{-i}) - U(\mathbf{a})$$

that is, the utility missed out on by not playing a'.

Positive regret means that action would have been better.

Regret Matching:

For some number of iterations:

1. Compute a regret matching profile ($\sigma \propto \text{normalized cumulative regret}$)
2. Add strategy profile to profile sum
3. Select an action
4. Compute regrets
5. Add regrets to cumulative regrets

Turn-taking Games

Turn-taking Games

Terminology

Turn-taking Games

Terminology

- Max and Min players

Turn-taking Games

Terminology

- Max and Min players
- deterministic

Turn-taking Games

Terminology

- Max and Min players
- deterministic
- two player

Turn-taking Games

Terminology

- Max and Min players
- deterministic
- two player
- zero-sum

Turn-taking Games

Terminology

- Max and Min players
- deterministic
- two player
- zero-sum
- perfect information

Minimax Trees

Minimax Trees

MDP Expectimax Tree

Minimax Trees

MDP Expectimax Tree

Minimax Tree

Minimax Trees

MDP Expectimax Tree

$$V(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \mathrm{E}[V(s')] \right)$$

Minimax Tree

Minimax Trees

MDP Expectimax Tree

Minimax Tree

$$V(s) = \max_{a\in\mathcal{A}} \left(R(s,a) + \mathrm{E}[V(s')]\right)$$

$$V_1(s) = \max_{a\in\mathcal{A}_1} \left(R(s,a) + \min_{a'\in A_2}(R(s',a') + V(s''))\right)$$

Tree Backup Example

Tree Backup Example

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))
```

```
function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v
```

```
function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

Tree Backup Example

```
function MINIMAX-DECISION(state) returns an action
   return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))
```

```
function MAX-VALUE(state) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for each a in ACTIONS(state) do
      v ← MAX(v, MIN-VALUE(RESULT(s, a)))
   return v
```

```
function MIN-VALUE(state) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← ∞
   for each a in ACTIONS(state) do
      v ← MIN(v, MAX-VALUE(RESULT(s, a)))
   return v
```

3 12 8 2 4 6 14 5 2

Tree Backup Example

function MINIMAX-DECISION(*state*) **returns** *an action*
  return $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT(*state*, *a*))

---

function MAX-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

---

function MIN-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow \infty$
  for each a **in** ACTIONS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a)))
  return v

3 12 8 2 4 6 14 5 2

Why is this harder than an MDP? (think back to sparse sampling)

Alpha-Beta Pruning

```
function ALPHA-BETA-SEARCH(state) returns an action
   v ← MAX-VALUE(state, −∞, +∞)
   return the action in ACTIONS(state) with value v
```

```
function MAX-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for each a in ACTIONS(state) do
      v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
      if v ≥ β then return v
      α ← MAX(α, v)
   return v
```

```
function MIN-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← +∞
   for each a in ACTIONS(state) do
      v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
      if v ≤ α then return v
      β ← MIN(β, v)
   return v
```

Evaluation Functions

Evaluation Functions

Examples of evaluation functions:

Evaluation Functions

Examples of evaluation functions:

- Rollout using default policy

Evaluation Functions

Examples of evaluation functions:

- Rollout using default policy
- Neural Network

Evaluation Functions

Examples of evaluation functions:

- Rollout using default policy
- Neural Network
- Weighted combination of features (e.g. number of Pawns, Knights, etc.)

Deep Blue

Deep Blue

- Defeated world champion Gary Kasparov in may 1997

Deep Blue

- Defeated world champion Gary Kasparov in may 1997
- 30 IBM RS/600 processors for alpha-beta search

Deep Blue

- Defeated world champion Gary Kasparov in may 1997
- 30 IBM RS/600 processors for alpha-beta search
- Routinely reached depth 14, sometimes up to 40

Deep Blue

- Defeated world champion Gary Kasparov in may 1997
- 30 IBM RS/600 processors for alpha-beta search
- Routinely reached depth 14, sometimes up to 40
- Evaluation function with 8000 features

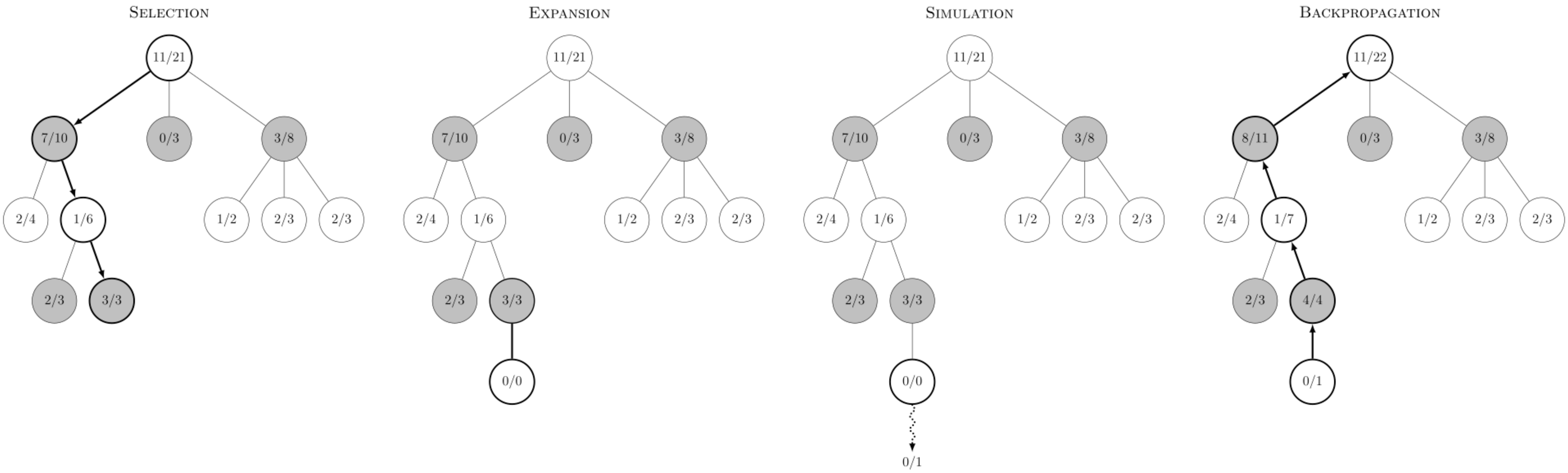
Deep Blue

- Defeated world champion Gary Kasparov in may 1997
- 30 IBM RS/600 processors for alpha-beta search
- Routinely reached depth 14, sometimes up to 40
- Evaluation function with 8000 features
- Opening book of 4000 positions

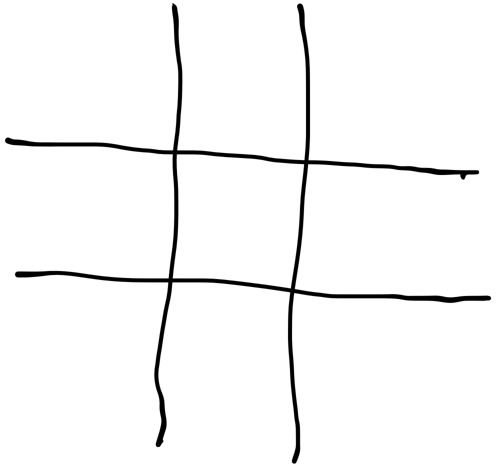
Deep Blue

- Defeated world champion Gary Kasparov in may 1997
- 30 IBM RS/600 processors for alpha-beta search
- Routinely reached depth 14, sometimes up to 40
- Evaluation function with 8000 features
- Opening book of 4000 positions
- Database of 700,000 grandmaster games

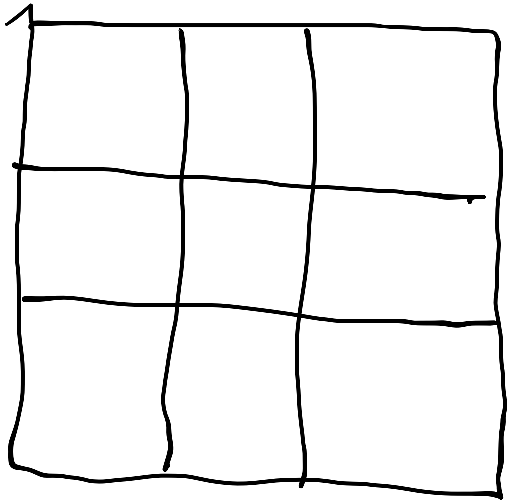
MCTS for Games



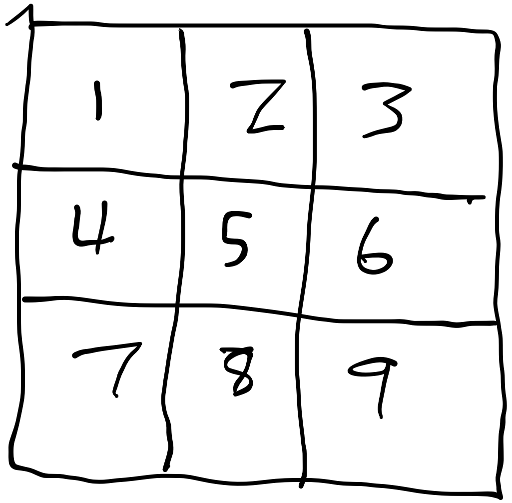
Break Exercise



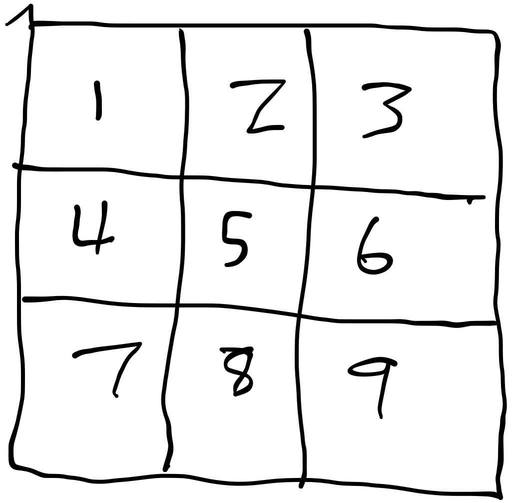
Break Exercise



Break Exercise

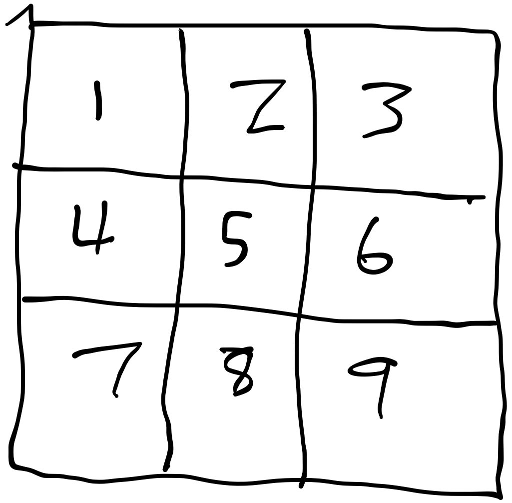


Break Exercise



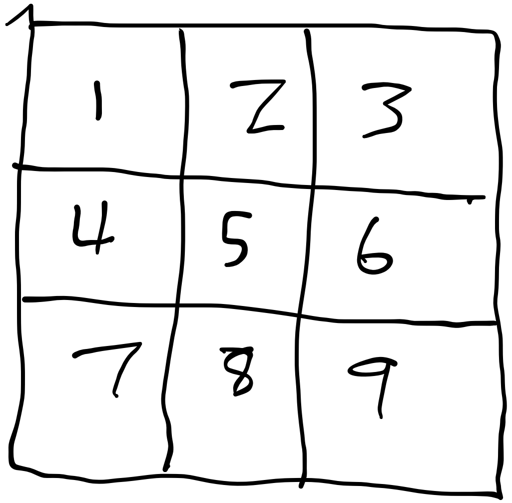
- Start at 5

Break Exercise



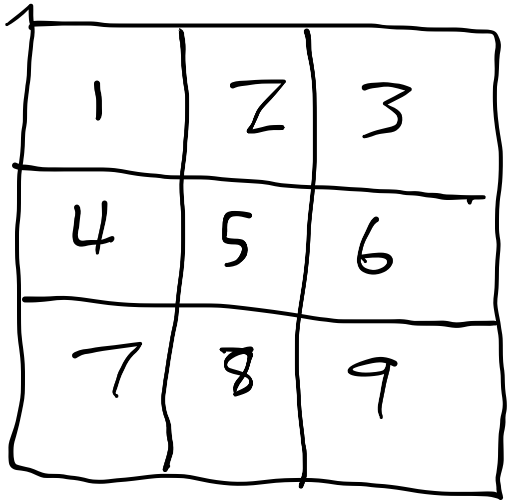
- Start at 5
- Max can move up/down

Break Exercise



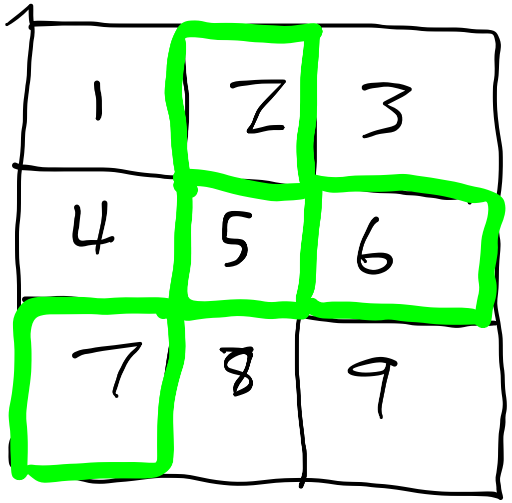
- Start at 5
- Max can move up/down
- Min moves left/right

Break Exercise



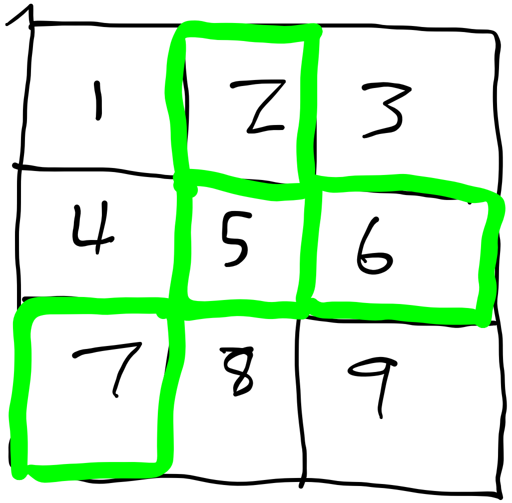
- Start at 5
- Max can move up/down
- Min moves left/right
- Each player gets 2 turns

Break Exercise



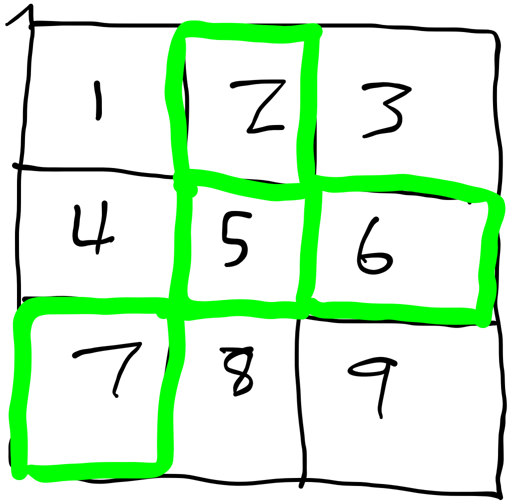
- Start at 5
- Max can move up/down
- Min moves left/right
- Each player gets 2 turns
- On green squares, Max wins

Break Exercise



- Start at 5
- Max can move up/down
- Min moves left/right
- Each player gets 2 turns
- On green squares, Max wins
- Who has advantage and what is first move?

Break Exercise



- Start at 5
- Max can move up/down
- Min moves left/right
- Each player gets 2 turns
- On green squares, Max wins
- Who has advantage and what is first move?

Bonus: what if 5 was not green?

Incomplete Information

Incomplete Information

Incomplete Information

Extensive-form game definition (h is a sequence of actions called a "history"):

Incomplete Information

Extensive-form game definition (h is a sequence of actions called a "history"):

- Finite set of n players, plus the "chance" player

Incomplete Information

Extensive-form game definition (h is a sequence of actions called a "history"):

- Finite set of n players, plus the "chance" player
- $P(h)$ (player at each history)

Incomplete Information

Extensive-form game definition (h is a sequence of actions called a "history"):

- Finite set of n players, plus the "chance" player
- $P(h)$ (player at each history)
- $A(h)$ (set of actions at each history)

Incomplete Information

Extensive-form game definition (h is a sequence of actions called a "history"):

- Finite set of n players, plus the "chance" player
- $P(h)$ (player at each history)
- $A(h)$ (set of actions at each history)
- $I(h)$ (information set that each history maps to)

Incomplete Information

Extensive-form game definition (h is a sequence of actions called a "history"):

- Finite set of n players, plus the "chance" player
- $P(h)$ (player at each history)
- $A(h)$ (set of actions at each history)
- $I(h)$ (information set that each history maps to)
- $U(h)$ (payoff for each leaf node in the game tree)

King-Ace Poker Example

King-Ace Poker Example

- 4 Cards: 2 Aces, 2 Kings

King-Ace Poker Example

- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card

King-Ace Poker Example

- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* (r) the payoff to 2 points or *check* (k) the payoff at 1 point

King-Ace Poker Example

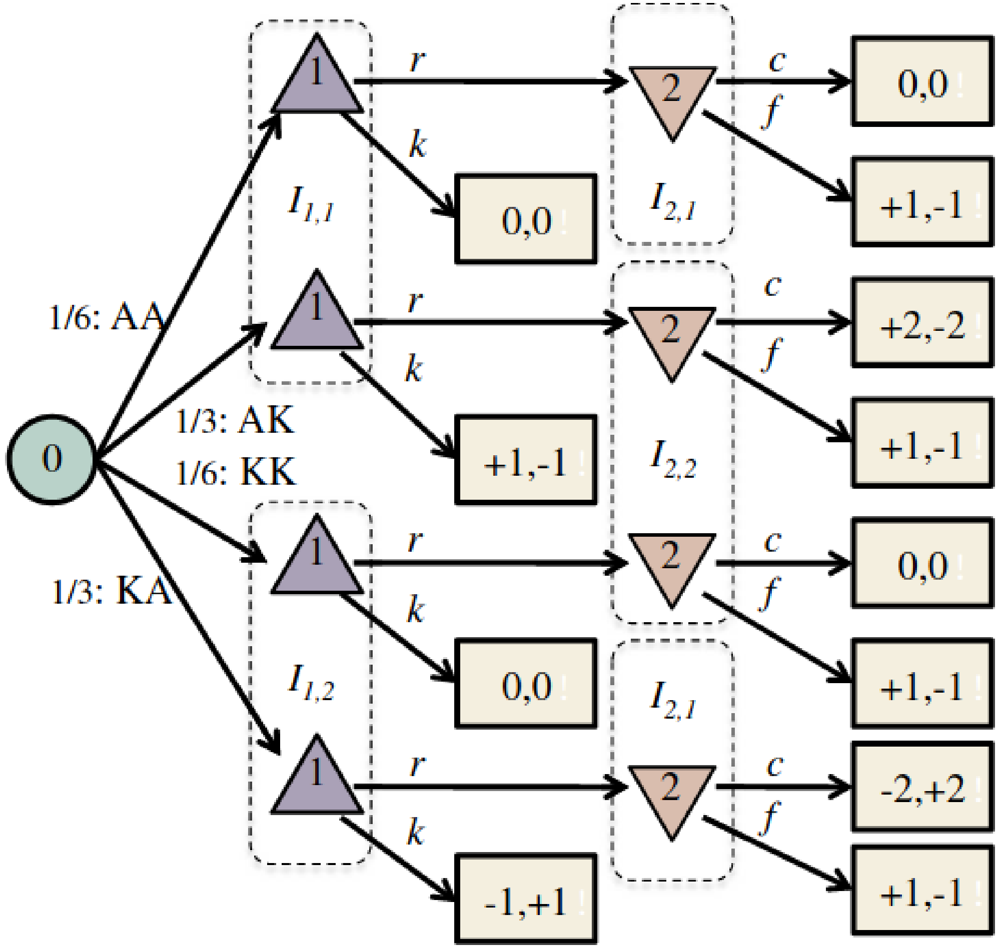
- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* (r) the payoff to 2 points or *check* (k) the payoff at 1 point
- If P1 raises, P2 can either *call* (c) Player 1's bet, or *fold* (f) the payoff back to 1 point

King-Ace Poker Example

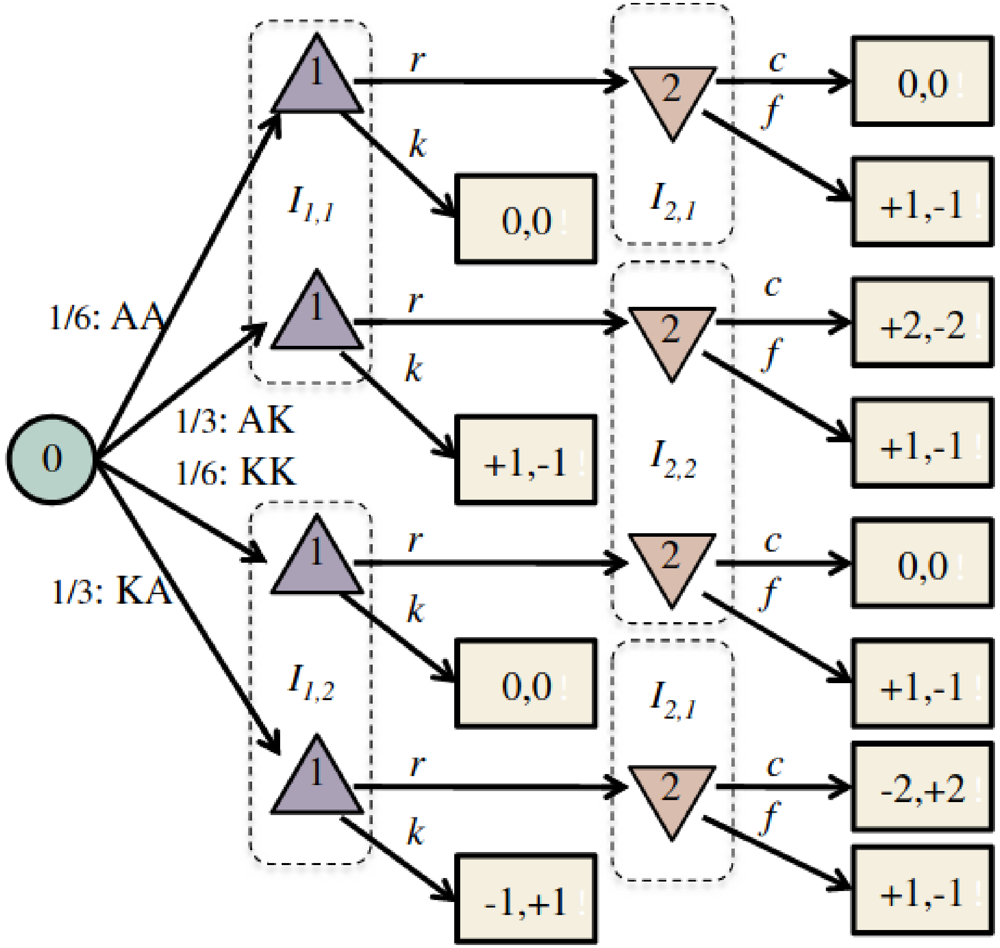
- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* (r) the payoff to 2 points or *check* (k) the payoff at 1 point
- If P1 raises, P2 can either *call* (c) Player 1's bet, or *fold* (f) the payoff back to 1 point
- The highest card wins

King-Ace Poker Example

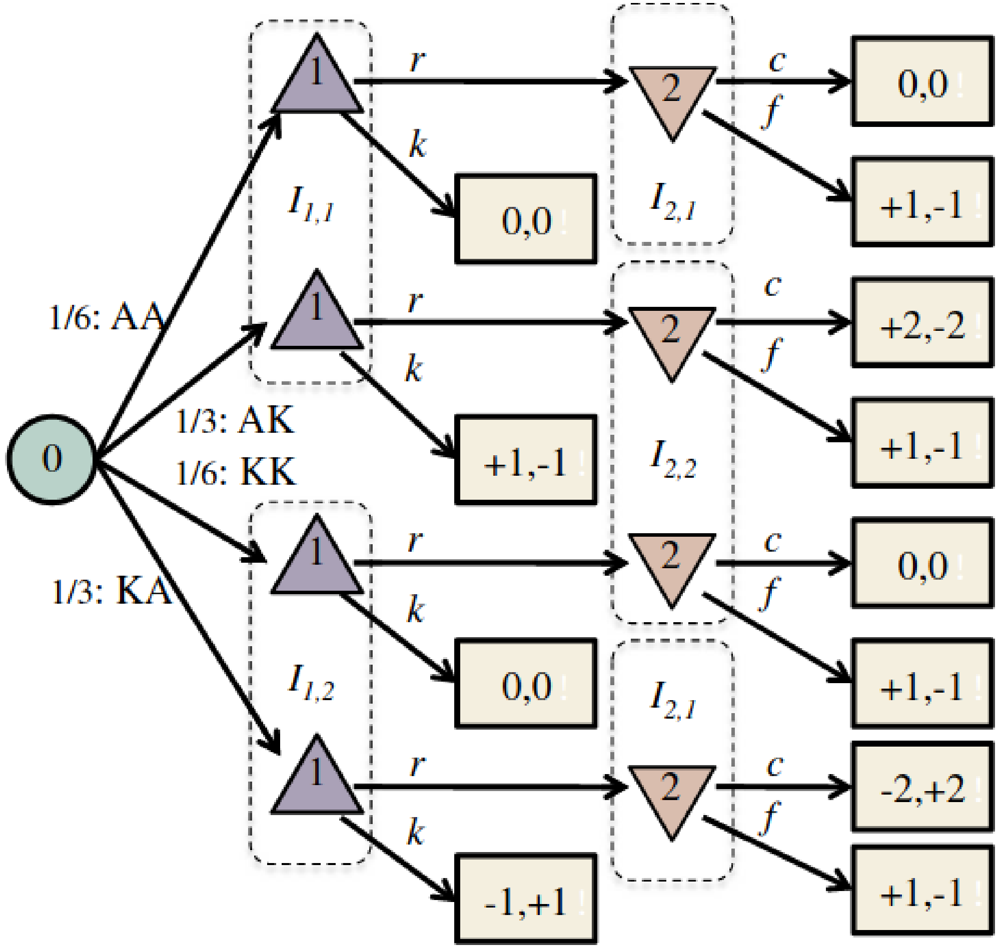
- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* (r) the payoff to 2 points or *check* (k) the payoff at 1 point
- If P1 raises, P2 can either *call* (c) Player 1's bet, or *fold* (f) the payoff back to 1 point
- The highest card wins



Extensive to Matrix Form

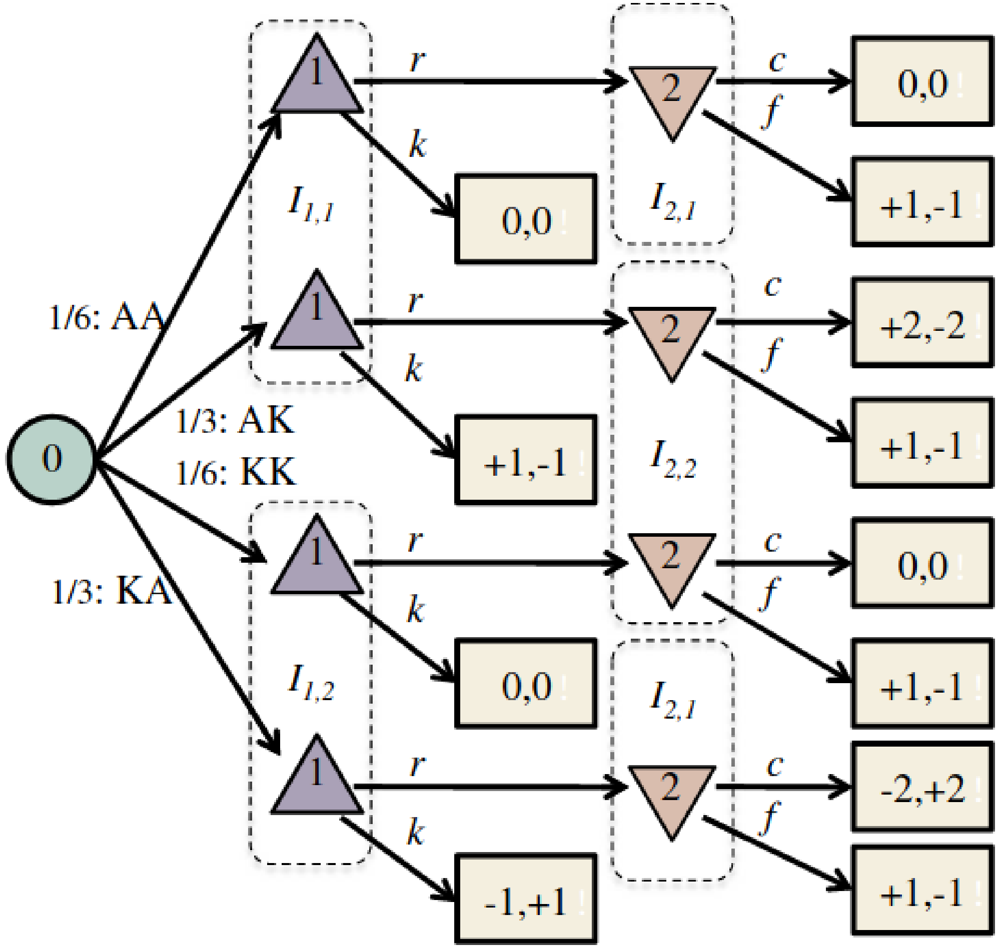


Extensive to Matrix Form



	2:*cc*	2:*cf*	2:*ff*	2:*fc*
1:*rr*	0	-1/6	1	7/6
1:*kr*	-1/3	-1/6	5/6	2/3
1:*rk*	1/3	0	1/6	1/2
1:*kk*	0	0	0	0

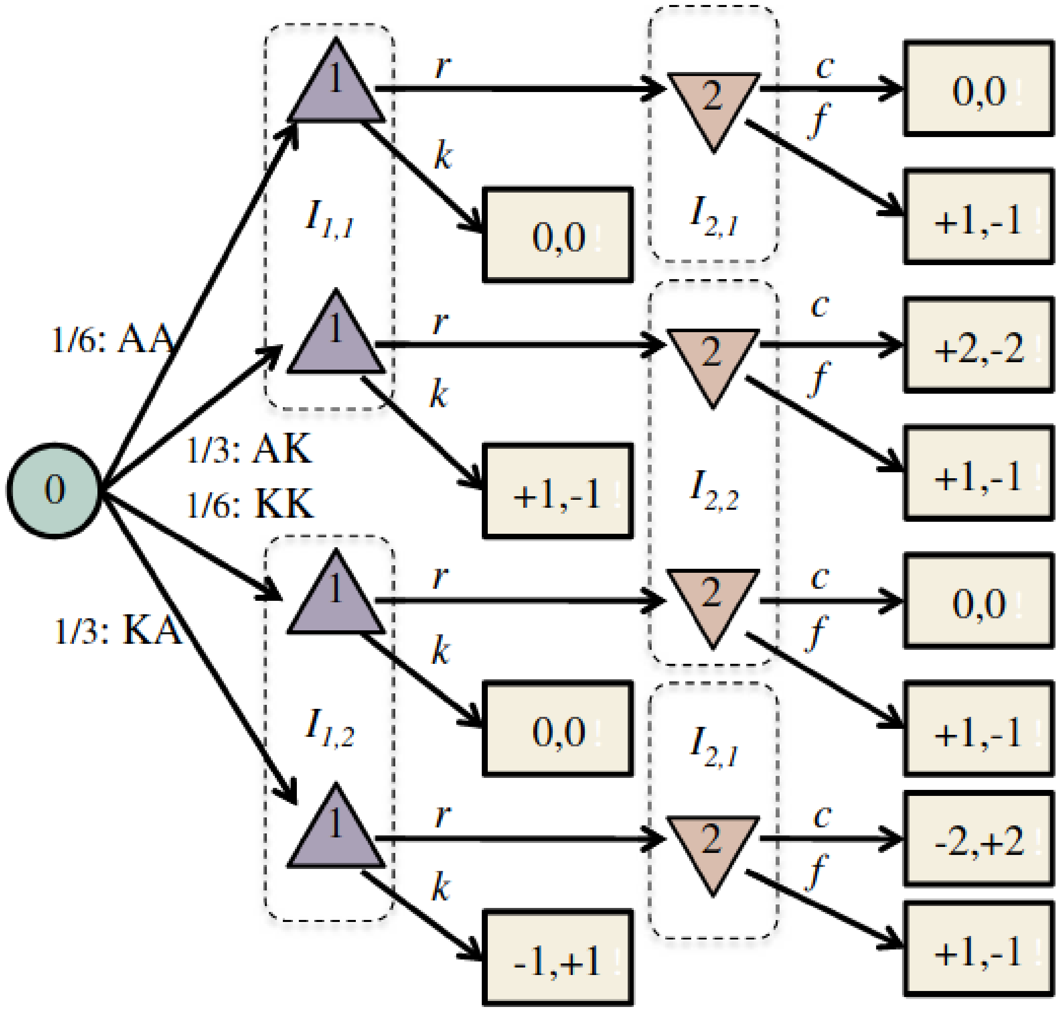
Extensive to Matrix Form



	2:*cc*	2:*cf*	2:*ff*	2:*fc*
1:*rr*	0	-1/6	1	7/6
1:*kr*	-1/3	-1/6	5/6	2/3
1:*rk*	1/3	0	1/6	1/2
1:*kk*	0	0	0	0

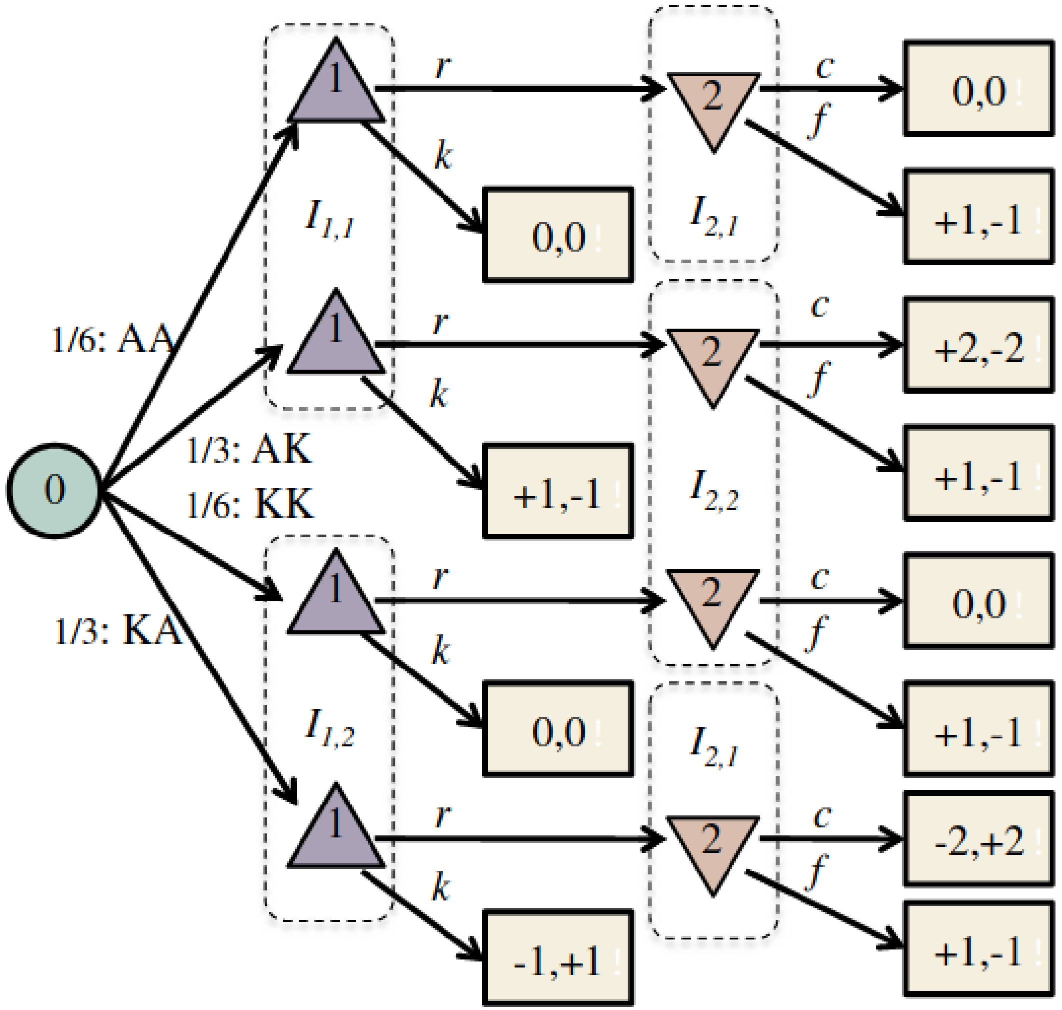
Exponential in number of info states!

Counterfactual Regret Minimization



Counterfactual Regret Minimization

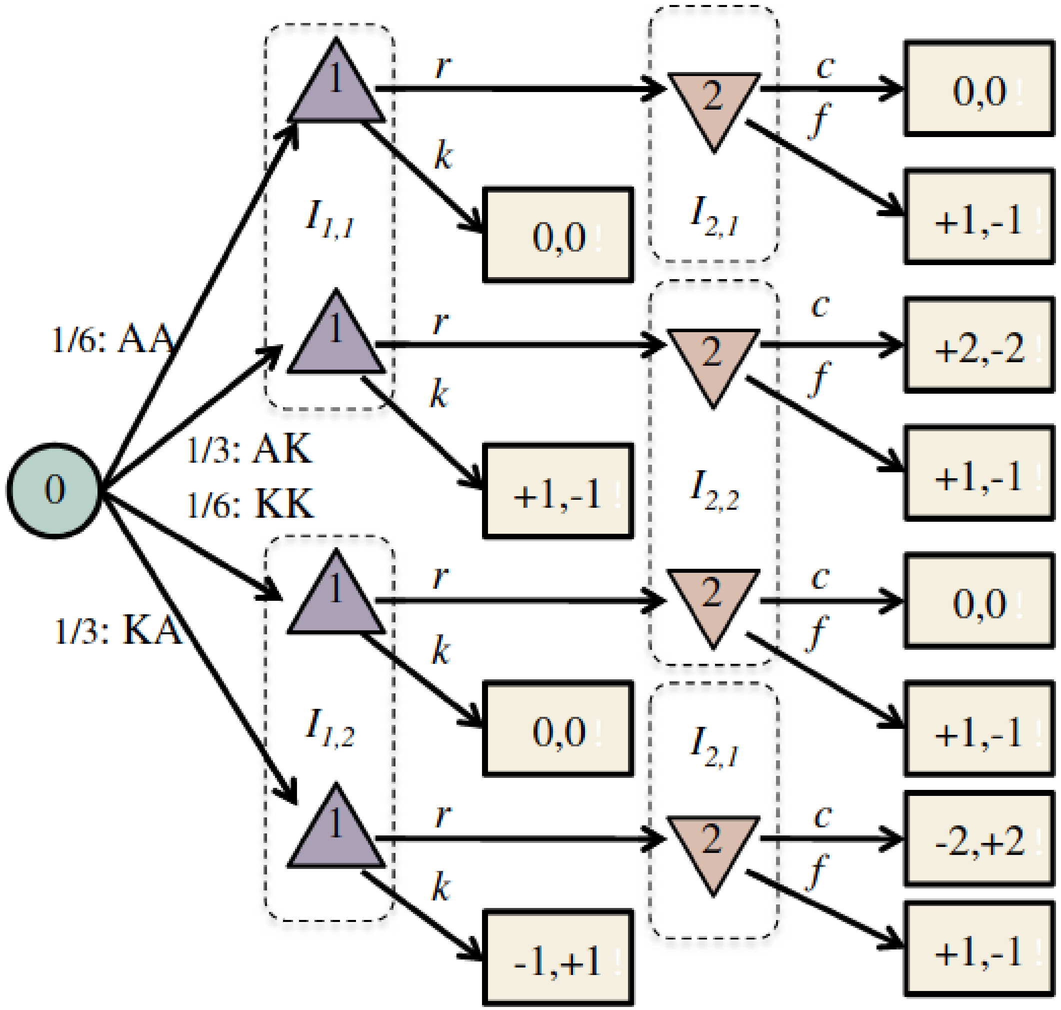
1: Initialize cumulative regret tables: $\forall I, r_I[a] \leftarrow 0$.
2: Initialize cumulative strategy tables: $\forall I, s_I[a] \leftarrow 0$.
3: Initialize initial profile: $\sigma^1(I,a) \leftarrow 1/|A(I)|$



Counterfactual Regret Minimization

```
Initialize cumulative regret tables: ∀I, r_I[a] ← 0.
Initialize cumulative strategy tables: ∀I, s_I[a] ← 0.
Initialize initial profile: σ^1(I, a) ← 1/|A(I)|
```

```
function Solve():
for t = {1, 2, 3, ..., T} do
    for i ∈ {1, 2} do
        CFR(∅, i, t, 1, 1)
    end for
end for
```

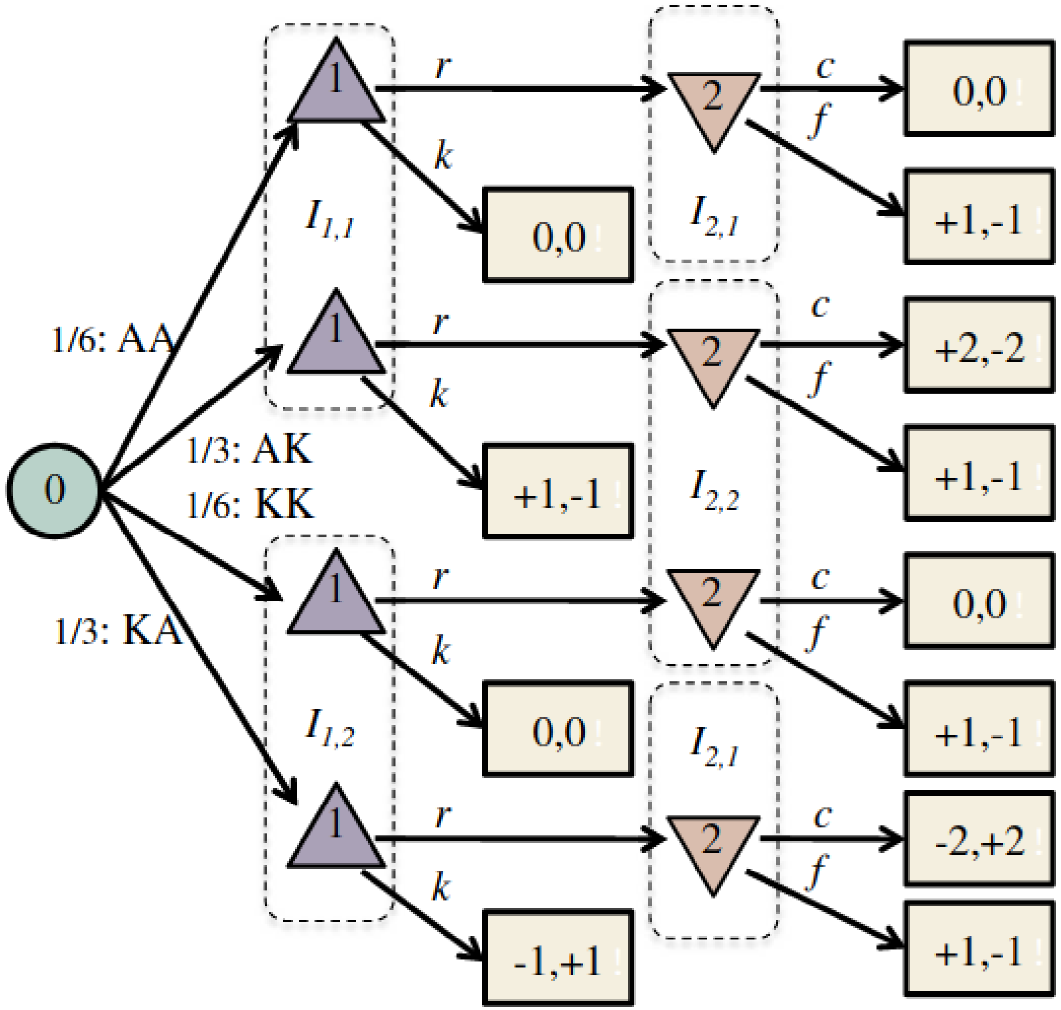


Counterfactual Regret Minimization

Initialize cumulative regret tables: $\forall I, r_I[a] \leftarrow 0$.
Initialize cumulative strategy tables: $\forall I, s_I[a] \leftarrow 0$.
Initialize initial profile: $\sigma^1(I,a) \leftarrow 1/|A(I)|$

function Solve():
for $t = \{1,2,3,\ldots,T\}$ **do**
  for $i \in \{1,2\}$ **do**
    CFR($\emptyset$, i, t, 1, 1)
  end for
end for

function CFR(h, i, t, π_1, π_2):
if h is terminal **then**
  return $u_i(h)$
else if h is a chance node **then**
  Sample a single outcome $a \sim \sigma_c(h,a)$
  return CFR(ha, i, t, π_1, π_2)
end if
Let I be the information set containing h.
$v_\sigma \leftarrow 0$
$v_{\sigma_{I\rightarrow a}}[a] \leftarrow 0$ for all $a \in A(I)$
for $a \in A(I)$ **do**
  if $P(h) = 1$ **then**
    $v_{\sigma_{I\rightarrow a}}[a] \leftarrow$ CFR(ha, i, t, $\sigma^t(I,a)\cdot\pi_1$, π_2)
  else if $P(h) = 2$ **then**
    $v_{\sigma_{I\rightarrow a}}[a] \leftarrow$ CFR(ha, i, t, π_1, $\sigma^t(I,a)\cdot\pi_2$)
  end if
  $v_\sigma \leftarrow v_\sigma + \sigma^t(I,a)\cdot v_{\sigma_{I\rightarrow a}}[a]$
end for
if $P(h) = i$ **then**
  for $a \in A(I)$ **do**
    $r_I[a] \leftarrow r_I[a] + \pi_{-i}\cdot(v_{\sigma_{I\rightarrow a}}[a] - v_\sigma)$
    $s_I[a] \leftarrow s_I[a] + \pi_i\cdot\sigma^t(I,a)$
  end for
  $\sigma^{t+1}(I) \leftarrow$ regret-matching values
end if
return v_σ



Kuhn Poker Example

3 Cards: 1, 2, and 3

Sequential Actions			Payoff
Player 1	Player 2	Player 1	
pass	pass		+1 to player with higher card
pass	bet	pass	+1 to player 2
pass	bet	bet	+2 to player with higher card
bet	pass		+1 to player 1
bet	bet		+2 to player with higher card

Kuhn Poker Example

3 Cards: 1, 2, and 3

Sequential Actions			Payoff
Player 1	Player 2	Player 1	
pass	pass		+1 to player with higher card
pass	bet	pass	+1 to player 2
pass	bet	bet	+2 to player with higher card
bet	pass		+1 to player 1
bet	bet		+2 to player with higher card

Solution (found by hand in 1950s):

- Player 1 may pass on a 3 with arbitrary probability γ, but then they must bet on a 1 w.p. $\gamma/3$ and bet on a 2 in the second round w.p. $\gamma/3 + 1/3$
- Player 2 must bet 1/3 of the time when holding a 1 after a pass and bet 1/3 of the time when holding a 2 with a bet

Matrix to Extensive Form

Matrix to Extensive Form

	R	P	S
R	$0, 0$	$-1, 1$	$1, -1$
P	$1, -1$	$0, 0$	$-1, 1$
S	$-1, 1$	$1, -1$	$0, 0$

Takeaways

Takeaways

- Games do not have a single optimal solution; Instead they have equilibria

Takeaways

- Games do not have a single optimal solution; Instead they have equilibria
- Regret matching is a tool for finding Nash equilibria in Matrix games

Takeaways

- Games do not have a single optimal solution; Instead they have equilibria
- Regret matching is a tool for finding Nash equilibria in Matrix games
- Turn-taking zero-sum games of perfect information can be solved using minimax trees with Alpha-beta pruning or MCTS

Takeaways

- Games do not have a single optimal solution; Instead they have equilibria
- Regret matching is a tool for finding Nash equilibria in Matrix games
- Turn-taking zero-sum games of perfect information can be solved using minimax trees with Alpha-beta pruning or MCTS
- Imperfect information can be represented as an extensive-form game with strategies conditioned on information sets.

Takeaways

- Games do not have a single optimal solution; Instead they have equilibria
- Regret matching is a tool for finding Nash equilibria in Matrix games
- Turn-taking zero-sum games of perfect information can be solved using minimax trees with Alpha-beta pruning or MCTS
- Imperfect information can be represented as an extensive-form game with strategies conditioned on information sets.
- Strategies for imperfect information games often involve stochastic "bluffing"

Takeaways

- Games do not have a single optimal solution; Instead they have equilibria
- Regret matching is a tool for finding Nash equilibria in Matrix games
- Turn-taking zero-sum games of perfect information can be solved using minimax trees with Alpha-beta pruning or MCTS
- Imperfect information can be represented as an extensive-form game with strategies conditioned on information sets.
- Strategies for imperfect information games often involve stochastic "bluffing"
- Counterfactual regret minimization can be used to solve extensive-form games