Multiagent Learning

Foundations and Recent Trends

Stefano Albrecht and Peter Stone

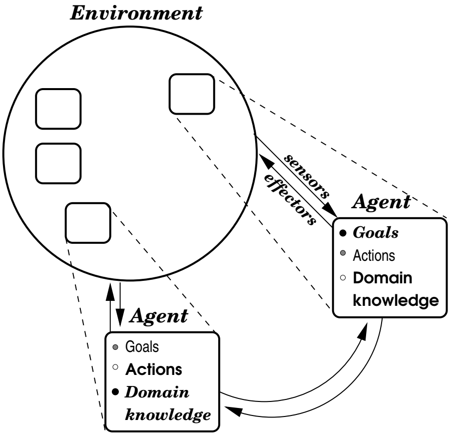
The University of Texas at Austin
Computer Science

Tutorial at IJCAI 2017 conference:
http://www.cs.utexas.edu/~larg/ijcai17_tutorial

Overview

Multiagent Systems

- Multiple agents interact in common environment
- Each agent with own sensors, effectors, goals, ...
- Agents have to coordinate actions to achieve goals

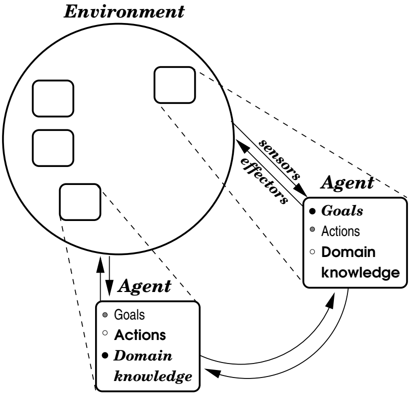


Environment defined by:

- state space
- available actions
- effects of actions on states
- what agents can observe

Agents defined by:

- domain knowledge
- goal specification
- policies for selecting actions

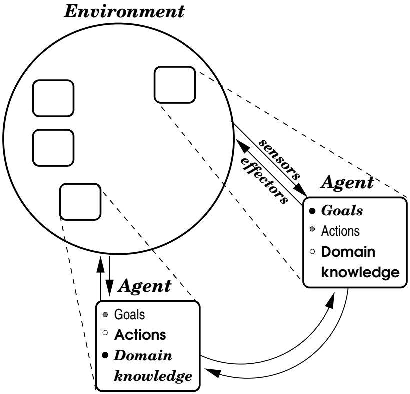


Environment defined by:

- state space
- available actions
- effects of actions on states
- what agents can observe

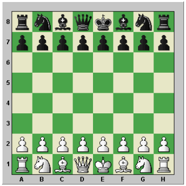
Agents defined by:

- domain knowledge
- goal specification
- policies for selecting actions



Many problems can be modelled as multiagent systems!

Chess

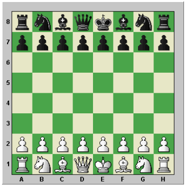


Poker

Starcraft

Multiagent Systems – Applications

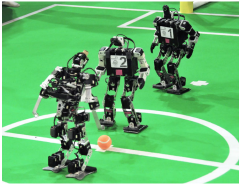
Chess



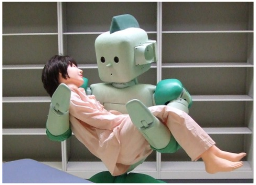
Poker

Starcraft

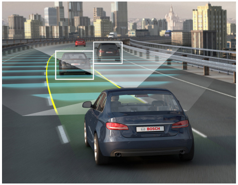
Robot soccer



Home assistance



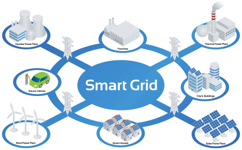
Autonomous cars



Negotiation

Wireless networks

Smart grid

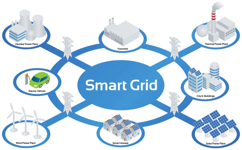


Multiagent Systems – Applications

Negotiation

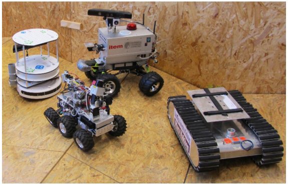
Wireless networks

Smart grid



User interfaces

Multi-robot rescue



Multiagent learning

- Learning is process of improving performance via experience
- Can agents learn to coordinate actions with other agents?
- What to learn?

Multiagent learning

- Learning is process of improving performance via experience
- Can agents learn to coordinate actions with other agents?
- What to learn?

  $\Rightarrow$ How to select own actions

Multiagent learning

- Learning is process of improving performance via experience
- Can agents learn to coordinate actions with other agents?
- What to learn?

  $\Rightarrow$ How to select own actions

  $\Rightarrow$ How other agents select actions

Multiagent Learning

Multiagent learning

- Learning is process of improving performance via experience
- Can agents learn to coordinate actions with other agents?
- What to learn?

  $\Rightarrow$ How to select own actions

  $\Rightarrow$ How other agents select actions

  $\Rightarrow$ Other agents' goals, plans, beliefs, ...

Why learning?

- Domain too complex to solve by hand or with multiagent planning
  e.g. computing equilibrium solutions in games

Why learning?

- Domain too complex to solve by hand or with multiagent planning
  e.g. computing equilibrium solutions in games

- Elements of domain unknown
  e.g. observation probabilities, behaviours of other agents, ...
  $\Rightarrow$ Multiagent planning requires complete model

Multiagent Learning

Why learning?

- Domain too complex to solve by hand or with multiagent planning
  e.g. computing equilibrium solutions in games

- Elements of domain unknown
  e.g. observation probabilities, behaviours of other agents, ...
  $\Rightarrow$ Multiagent planning requires complete model

- Other agents may learn too
  $\Rightarrow$ Have to adapt continually!
  "*Moving target problem*" central issue in multiagent learning

Research in Multiagent Learning

Multiagent learning studied in different communities

- AI, game theory, robotics, psychology, ...

Research in Multiagent Learning

Multiagent learning studied in different communities

- AI, game theory, robotics, psychology, ...
- Some conferences & journals: AAMAS, AAAI, IJCAI, NIPS, UAI, ICML, ICRA, IROS, RSS, PRIMA, JAAMAS, AIJ, JAIR, MLJ, JMLR, ...

  $\Rightarrow$ Very large + growing body of work!

Research in Multiagent Learning

Multiagent learning studied in different communities

- AI, game theory, robotics, psychology, ...
- Some conferences & journals: AAMAS, AAAI, IJCAI, NIPS, UAI, ICML, ICRA, IROS, RSS, PRIMA, JAAMAS, AIJ, JAIR, MLJ, JMLR, ...

  $\Rightarrow$ Very large + growing body of work!
- Many algorithms proposed to address different assumptions (constraints), learning goals, performance criteria, ...

Tutorial

This tutorial:

- Introduction to basics of multiagent learning:
    - Interaction models & assumptions
    - Learning goals
    - Selection of learning algorithms
- Plus some recent trends

Tutorial

This tutorial:

- Introduction to basics of multiagent learning:
  - Interaction models & assumptions
  - Learning goals
  - Selection of learning algorithms
- Plus some recent trends

Further reading:

- AIJ Special Issue *"Foundations of Multi-Agent Learning"*
  Rakesh Vohra, Michael Wellman (eds.), 2007
- Surveys: Tuyls and Weiss (2012); Busoniu et al. (2008); Panait and Luke (2005); Shoham et al. (2003); Alonso et al. (2001); Stone and Veloso (2000); Sen and Weiss (1999)
- Our own upcoming survey on agents modelling other agents!

Overview

- Introduction
- Multiagent Models & Assumptions
- Learning Goals
- Learning Algorithms
- Recent Trends

Overview

Multiagent Models & Assumptions

Standard multiagent models:

- Normal-form game
- Repeated game
- Stochastic game

Assumptions and other models

Normal-Form Game

Normal-form game consists of:

- Finite set of agents $N = \{1, ..., n\}$
- For each agent $i \in N$:
  - Finite set of actions A_i
  - Utility function $u_i : A \to \mathbb{R}$, where $A = A_1 \times ... \times A_n$

Normal-Form Game

Normal-form game consists of:

- Finite set of agents $N = \{1, ..., n\}$
- For each agent $i \in N$:
  - Finite set of actions A_i
  - Utility function $u_i : A \rightarrow \mathbb{R}$, where $A = A_1 \times ... \times A_n$

Each agent i selects policy $\pi_i : A_i \rightarrow [0, 1]$, takes action $a_i \in A_i$ with probability $\pi_i(a_i)$, and receives utility $u_i(a_1, ..., a_n)$

Given policy profile $(\pi_1, ..., \pi_n)$, expected utility to i is

$$U_i(\pi_1, ..., \pi_n) = \sum_{a \in A} \pi_1(a_1) * ... * \pi_n(a_n) * u_i(a)$$

$\Rightarrow$ Agents want to maximise their expected utilities

Normal-Form Game: Prisoner's Dilemma

Example: Prisoner's Dilemma

- Two prisoners questioned in isolated cells
- Each prisoner can Cooperate or Defect
- Utilities (row = agent 1, column = agent 2):

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Example: Chicken

- Two opposite drivers on same lane
- Each driver can **Stay** on lane or **Leave** lane
- Utilities:

	S	L
S	0,0	7,2
L	2,7	6,6

Normal-Form Game: Rock-Paper-Scissors

Example: Rock-Paper-Scissors

- Two players, three actions
- **Rock** beats **Scissors** beats **Paper** beats **Rock**
- Utilities:

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

Repeated Game

Learning requires experience

- Normal-form game is single interaction

  $\Rightarrow$ No experience!

- Experience comes from repeated interactions

Learning requires experience

- Normal-form game is single interaction

  $\Rightarrow$ No experience!
- Experience comes from repeated interactions

Repeated game:

- Repeat same normal-form game: at each time t, each agent i chooses action a_i^t and gets utility $u_i(a_1^t, ..., a_n^t)$
- Policy $\pi_i : \mathbb{H} \times A_i \to [0,1]$ assigns action probabilities based on history of interaction

$$\mathbb{H} = \cup_{t \in \mathbb{N}^0} \mathbb{H}^t, \quad \mathbb{H}^t = \left\{ H^t = (a^0, a^1, ..., a^{t-1}) \mid a^\tau \in A \right\}$$

Repeated Game

What is expected utility to i for policy profile $(\pi_1, \ldots, \pi_n)$?

- Repeating game $t \in \mathbb{N}$ times:

$$U_i(\pi_1, \ldots, \pi_n) \;=\; \sum_{H^t \in \mathbb{H}^t} P(H^t | \pi_1, \ldots, \pi_n) \sum_{\tau=0}^{t-1} u_i(a^\tau)$$

$$P(H^t | \pi_1, \ldots, \pi_n) \;=\; \prod_{\tau=0}^{t-1} \prod_{j \in N} \pi_j(H^\tau, a_j^\tau)$$

Repeated Game

What is expected utility to i for policy profile $(\pi_1, ..., \pi_n)$?

- Repeating game ∞ times:

$$U_i(\pi_1, ..., \pi_n) = \lim_{t \to \infty} \sum_{H^t} P(H^t | \pi_1, ..., \pi_n) \sum_{\tau} \gamma^\tau u_i(a^\tau)$$

Discount factor $0 \leq \gamma < 1$ makes expectation finite

Interpretation: low γ is "myopic", high γ is "farsighted"
(Or: probability that game will end at each time is $1 - \gamma$)

Repeated Game

What is expected utility to i for policy profile $(\pi_1, ..., \pi_n)$?

- Repeating game ∞ times:

$$U_i(\pi_1, ..., \pi_n) = \lim_{t \to \infty} \sum_{H^t} P(H^t | \pi_1, ..., \pi_n) \sum_{\tau} \gamma^\tau u_i(a^\tau)$$

  Discount factor $0 \leq \gamma < 1$ makes expectation finite

  Interpretation: low γ is "myopic", high γ is "farsighted"
  (Or: probability that game will end at each time is $1 - \gamma$)

  Can also define expected utility as limit average

Repeated Game: Prisoner's Dilemma

Example: Repeated Prisoner's Dilemma

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Example policies:

- At time t, choose C with probability $(t+1)^{-1}$
- Grim: chose C until opponent's first D, then choose D forever
- Tit-for-Tat: begin C, then repeat opponent's last action

Example: Repeated 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

Example policy:

- Compute empirical frequency of opponent actions over past 5 moves

$$P(a_j) = \frac{1}{5} \sum_{\tau=t-5}^{t-1} [a_j^\tau = a_j]_1$$

and take best-response action $\max_{a_i} \sum_{a_j} P(a_j)\, u_i(a_i, a_j)$

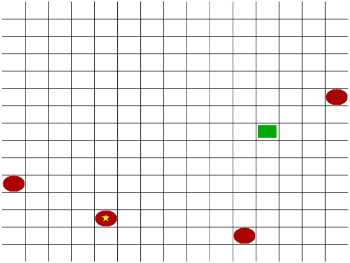
Stochastic Game

Agents interact in common environment

- Environment has states, actions have effect on state
- Agents choose actions based on state-action history

Example: Pursuit (e.g. Barrett et al., 2011)

- Predator agents must capture prey
- State: agent positions
- Actions: move to neighbouring cell



Stochastic Game

Stochastic game consists of:

- Finite set of agents $N = \{1, ..., n\}$
- Finite set of states S
- For each agent $i \in N$:
  - Finite set of actions A_i
  - Utility function $u_i : S \times A \to \mathbb{R}$, where $A = A_1 \times ... \times A_n$
- State transition function $T : S \times A \times S \to [0, 1]$

Stochastic Game

Stochastic game consists of:

- Finite set of agents $N = \{1, ..., n\}$
- Finite set of states S
- For each agent $i \in N$:
  - Finite set of actions A_i
  - Utility function $u_i : S \times A \rightarrow \mathbb{R}$, where $A = A_1 \times ... \times A_n$
- State transition function $T : S \times A \times S \rightarrow [0, 1]$

Generalises **Markov decision process (MDP)** to multiple agents

Stochastic Game

Game starts in initial state $s^0 \in S$

At each time t:

- Each agent i...

Stochastic Game

Game starts in initial state $s^0 \in S$

At each time t:

- Each agent i...
  - observes current state s^t and past joint action a^{t-1} (if $t > 0$)

Stochastic Game

Game starts in initial state $s^0 \in S$

At each time t:

- Each agent i...
  - observes current state s^t and past joint action a^{t-1} (if $t > 0$)
  - chooses action $a_i^t \in A_i$ with probability $\pi_i(H^t, a_i^t)$ where $H^t = (s^0, a^0, s^1, a^1, ..., s^{t-1})$ is state-action history

Stochastic Game

Game starts in initial state $s^0 \in S$

At each time t:

- Each agent i...
  - observes current state s^t and past joint action a^{t-1} (if $t > 0$)
  - chooses action $a_i^t \in A_i$ with probability $\pi_i(H^t, a_i^t)$ where $H^t = (s^0, a^0, s^1, a^1, ..., s^{t-1})$ is state-action history
  - receives utility $u_i(a_1^t, ..., a_n^t)$

Stochastic Game

Game starts in initial state $s^0 \in S$

At each time t:

- Each agent i...
  - observes current state s^t and past joint action a^{t-1} (if $t > 0$)
  - chooses action $a_i^t \in A_i$ with probability $\pi_i(H^t, a_i^t)$ where $H^t = (s^0, a^0, s^1, a^1, ..., s^{t-1})$ is state-action history
  - receives utility $u_i(a_1^t, ..., a_n^t)$
- Game transitions into next state $s^{t+1} \in S$ with probability $T(s^t, a^t, s^{t+1})$

Stochastic Game

Game starts in initial state $s^0 \in S$

At each time t:

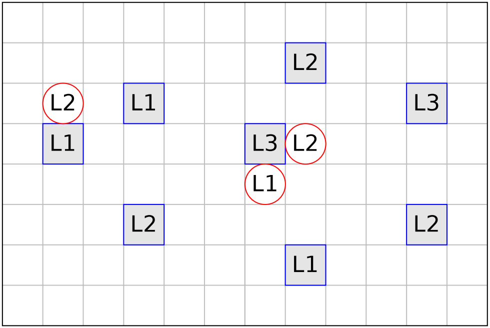
- Each agent i...
  - observes current state s^t and past joint action a^{t-1} (if $t > 0$)
  - chooses action $a_i^t \in A_i$ with probability $\pi_i(H^t, a_i^t)$ where $H^t = (s^0, a^0, s^1, a^1, ..., s^{t-1})$ is state-action history
  - receives utility $u_i(a_1^t, ..., a_n^t)$
- Game transitions into next state $s^{t+1} \in S$ with probability $T(s^t, a^t, s^{t+1})$

Process repeated finite or infinite number of times, or until terminal state is reached (e.g. prey captured).

Stochastic Game: Level-Based Foraging

Example: Level-Based Foraging (Albrecht and Ramamoorthy, 2013)

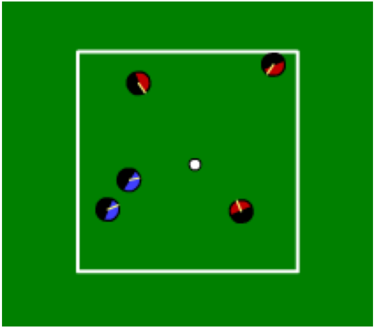
- Agents (circles) must collect all items (squares)
- State: agent positions, item positions, which items collected
- Actions: move to neighbouring cell, try to collect item



Stochastic Game: Soccer Keepaway

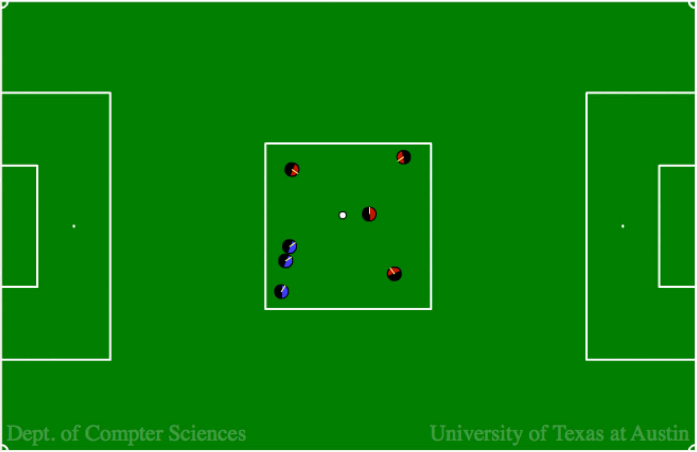
Example: Soccer Keepaway (Stone et al., 2005)

- "Keeper" agents must keep ball away from "Taker" agents
- State: player positions & orientations, ball position, ...
- Actions: go to ball, pass ball to player, ...



Stochastic Game: Soccer Keepaway

Video: 4 vs 3 Keepaway



Source: `http://www.cs.utexas.edu/~AustinVilla/sim/keepaway`

Assumptions

Models and algorithms make assumptions, e.g.

- What do agents know about the game?
- What can agents observe during a game?

Assumptions

Models and algorithms make assumptions, e.g.

- What do agents know about the game?
- What can agents observe during a game?

Usual assumptions:

- Game elements known (state/action space, utility function, ...)
- Game states & chosen actions are commonly observed

  $\Rightarrow$ *"full observability"* or *"perfect information"*

Assumptions

Models and algorithms make assumptions, e.g.

- What do agents know about the game?
- What can agents observe during a game?

Usual assumptions:

- Game elements known (state/action space, utility function, ...)
- Game states & chosen actions are commonly observed

  $\Rightarrow$ *"full observability"* or *"perfect information"*

Many learning algorithms designed for repeated/stochastic game with full observability

- But assumptions may vary and other models exist!

Other Models

Other assumptions & models:

- Assumption: elements of game unknown
  - Bayesian game, stochastic Bayesian game

    AAAI'16 tutorial "Type-based Methods for Interaction in Multiagent Systems"

    `http://thinc.cs.uga.edu/tutorials/aaai-16.html`

- Assumption: partial observability of states and actions
  - Extensive-form game with imperfect information
  - Partially observable stochastic game (POSG)
  - Multiagent POMDPs: Dec-POMDP, I-POMDP, ...

Overview

Learning Goals

Learning is to improve performance via experience

- But what is goal (end-result) of learning process?
- How to measure success of learning?

Learning Goals

Learning is to improve performance via experience

- But what is goal (end-result) of learning process?
- How to measure success of learning?

Many learning goals proposed:

- Minimax/Nash/correlated equilibrium
- Pareto-optimality
- Social welfare & fairness
- No-regret
- Targeted optimality & safety

… plus combinations & approximations

Maximin/Minimax

Two-player zero-sum game: $u_i = -u_j$

- e.g. Rock-Paper-Scissors, Chess

Maximin/Minimax

Two-player zero-sum game: $u_i = -u_j$

- e.g. Rock-Paper-Scissors, Chess

Policy profile (π_i, π_j) is maximin/minimax profile if

$$U_i(\pi_i, \pi_j) = \max_{\pi_i'} \min_{\pi_j'} U_i(\pi_i', \pi_j') = \min_{\pi_j'} \max_{\pi_i'} U_i(\pi_i', \pi_j') = -U_j(\pi_i, \pi_j)$$

Utility that can be guaranteed against worst-case opponent

Maximin/Minimax

Two-player zero-sum game: $u_i = -u_j$

- e.g. Rock-Paper-Scissors, Chess

Policy profile (π_i, π_j) is maximin/minimax profile if

$$U_i(\pi_i, \pi_j) = \max_{\pi_i'} \min_{\pi_j'} U_i(\pi_i', \pi_j') = \min_{\pi_j'} \max_{\pi_j'} U_i(\pi_i', \pi_j') = -U_j(\pi_i, \pi_j)$$

Utility that can be guaranteed against worst-case opponent

- Every two-player zero-sum normal-form game has minimax profile (von Neumann and Morgenstern, 1944)
- Every finite or infinite+discounted zero-sum stochastic game has minimax profile (Shapley, 1953)

Nash Equilibrium

Policy profile $\pi = (\pi_1, ..., \pi_n)$ is Nash equilibrium (NE) if

$$\forall i \; \forall \pi_i' : U_i(\pi_i', \pi_{-i}) \leq U_i(\pi)$$

No agent can improve utility by unilaterally deviating from profile (every agent plays best-response to other agents)

Nash Equilibrium

Policy profile $\pi = (\pi_1, ..., \pi_n)$ is Nash equilibrium (NE) if

$$\forall i \; \forall \pi_i' : U_i(\pi_i', \pi_{-i}) \leq U_i(\pi)$$

No agent can improve utility by unilaterally deviating from profile (every agent plays best-response to other agents)

Every finite normal-form game has at least one NE (Nash, 1950) (also stochastic games, e.g. Fink (1964))

- Standard solution in game theory
- In two-player zero-sum game, minimax is same as NE

Nash Equilibrium – Example

Example: Prisoner's Dilemma

- Only NE in normal-form game is (D,D)
- Normal-form NE are also NE in infinite repeated game
- Infinite repeated game has many more NE ⇒ Folk theorem

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Nash Equilibrium – Example

Example: Prisoner's Dilemma

- Only NE in normal-form game is (D,D)
- Normal-form NE are also NE in infinite repeated game
- Infinite repeated game has many more NE $\Rightarrow$ Folk theorem

	C	D
C	-1,-1	-5,0
D	0,-5	-3,-3

Example: Rock-Paper-Scissors

- Only NE in normal-form game is $\pi_i = \pi_j = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$

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

Correlated Equilibrium

Each agent i observes signal x_i with joint distribution $\xi(x_1, ..., x_n)$

- E.g. x_i is action recommendation to agent i

Correlated Equilibrium

Each agent i observes signal x_i with joint distribution $\xi(x_1, ..., x_n)$

- E.g. x_i is action recommendation to agent i

$(\pi_1, ..., \pi_n)$ is correlated equilibrium (CE) (Aumann, 1974) if no agent can individually improve its expected utility by deviating from recommended actions

- NE is subset of CE $\rightarrow$ no correlation
- CE easier to compute than NE $\rightarrow$ linear program

Correlated Equilibrium – Example

Example: Chicken

Correlated equilibrium:

- $\xi(L, L) = \xi(S, L) = \xi(L, S) = \frac{1}{3}$
- $\xi(S, S) = 0$

Expected utility to both:

$7 * \frac{1}{3} + 2 * \frac{1}{3} + 6 * \frac{1}{3} = 5$

	S	L
S	0,0	7,2
L	2,7	6,6

Example: Chicken

Correlated equilibrium:

- $\xi(L,L) = \xi(S,L) = \xi(L,S) = \frac{1}{3}$
- $\xi(S,S) = 0$

	S	L
S	0,0	7,2
L	2,7	6,6

Expected utility to both:

$7 * \frac{1}{3} + 2 * \frac{1}{3} + 6 * \frac{1}{3} = 5$

Nash equilibrium utilities:

- $\pi_i(S) = 1,\ \pi_j(S) = 0 \ \rightarrow \ (7,2)$
- $\pi_i(S) = 0,\ \pi_j(S) = 1 \ \rightarrow \ (2,7)$
- $\pi_i(S) = \frac{1}{3},\ \pi_j(S) = \frac{1}{3} \ \rightarrow \ \approx 4.66$

The Equilibrium Legacy

The "Equilibrium Legacy" in multiagent learning:

- Quickly adopted equilibrium as standard goal of learning
- But equilibrium (e.g. NE) has many limitations...

The Equilibrium Legacy

The "Equilibrium Legacy" in multiagent learning:

- Quickly adopted equilibrium as standard goal of learning
- But equilibrium (e.g. NE) has many limitations...

1. Non-uniqueness
   Often multiple NE exist, how should agents choose same one?

The Equilibrium Legacy

The "Equilibrium Legacy" in multiagent learning:

- Quickly adopted equilibrium as standard goal of learning
- But equilibrium (e.g. NE) has many limitations...

1. Non-uniqueness
   Often multiple NE exist, how should agents choose same one?
2. Incompleteness
   NE does not specify behaviours for off-equilibrium paths

The Equilibrium Legacy

The "Equilibrium Legacy" in multiagent learning:

- Quickly adopted equilibrium as standard goal of learning
- But equilibrium (e.g. NE) has many limitations...

1. Non-uniqueness
   Often multiple NE exist, how should agents choose same one?
2. Incompleteness
   NE does not specify behaviours for off-equilibrium paths
3. Sup-optimality
   NE not generally same as utility maximisation

The Equilibrium Legacy

The “Equilibrium Legacy” in multiagent learning:

- Quickly adopted equilibrium as standard goal of learning
- But equilibrium (e.g. NE) has many limitations…

1. Non-uniqueness
   Often multiple NE exist, how should agents choose same one?
2. Incompleteness
   NE does not specify behaviours for off-equilibrium paths
3. Sup-optimality
   NE not generally same as utility maximisation
4. Rationality
   NE assumes all agents are rational (= perfect utility maximisers)

Pareto Optimum

Policy profile $\pi = (\pi_1, ..., \pi_n)$ is Pareto-optimal if there is no other profile π' such that

$$\forall i : U_i(\pi') \geq U_i(\pi) \quad \textbf{and} \quad \exists_i : U_i(\pi') > U_i(\pi)$$

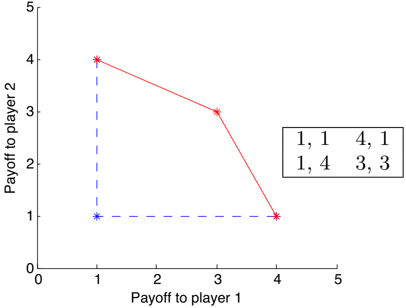
Can't improve one agent without making other agent worse off

Pareto Optimum

Policy profile $\pi = (\pi_1, ..., \pi_n)$ is Pareto-optimal if there is no other profile π' such that

$$\forall i : U_i(\pi') \geq U_i(\pi) \quad \textbf{and} \quad \exists_i : U_i(\pi') > U_i(\pi)$$

Can't improve one agent without making other agent worse off



Pareto-front is set of all Pareto-optimal utilities (red line)

Social Welfare & Fairness

Pareto-optimality says nothing about social welfare and fairness

Social Welfare & Fairness

Pareto-optimality says nothing about social welfare and fairness

Welfare and fairness of profile $\pi = (\pi_1, ..., \pi_n)$ often defined as

$$Welfare(\pi) = \sum_i U_i(\pi) \qquad Fairness(\pi) = \prod_i U_i(\pi)$$

π welfare/fairness-optimal if maximum $Welfare(\pi)$/$Fairness(\pi)$

Social Welfare & Fairness

Pareto-optimality says nothing about social welfare and fairness

Welfare and fairness of profile $\pi = (\pi_1, ..., \pi_n)$ often defined as

$$Welfare(\pi) = \sum_i U_i(\pi) \qquad Fairness(\pi) = \prod_i U_i(\pi)$$

π welfare/fairness-optimal if maximum $Welfare(\pi)$/$Fairness(\pi)$

Any welfare/fairness-optimal π is also Pareto-optimal! (Why?)

No-Regret

Given history $H^t = (a^0, a^1, ..., a^{t-1})$, agent i's regret for not having taken action a_i is

$$R_i(a_i | H^t) = \sum_{\tau=0}^{t-1} u_i(a_i, a_{-i}^\tau) - u_i(a_i^\tau, a_{-i}^\tau)$$

No-Regret

Given history $H^t = (a^0, a^1, ..., a^{t-1})$, agent i's regret for not having taken action a_i is

$$R_i(a_i|H^t) = \sum_{\tau=0}^{t-1} u_i(a_i, a_{-i}^\tau) - u_i(a_i^\tau, a_{-i}^\tau)$$

Policy π_i achieves no-regret if

$$\forall a_i : \lim_{t\to\infty} \frac{1}{t} R_i(a_i|H^t) \leq 0$$

(Other variants exist)

No-Regret

Like Nash equilibrium, no-regret widely used in multiagent learning

But, like NE, definition of regret has conceptual issues

No-Regret

Like Nash equilibrium, no-regret widely used in multiagent learning

But, like NE, definition of regret has conceptual issues

- Regret definition assumes other agents don't change actions

$$R_i(a_i|H^t) = \sum_{\tau=0}^{t-1} u_i(a_i, a_{-i}^{\tau}) - u_i(a_i^{\tau}, a_{-i}^{\tau})$$

$\Rightarrow$ But: entire history may change if different actions taken!

No-Regret

Like Nash equilibrium, no-regret widely used in multiagent learning

But, like NE, definition of regret has conceptual issues

- Regret definition assumes other agents don't change actions

$$R_i(a_i|H^t) = \sum_{\tau=0}^{t-1} u_i(a_i, a_{-i}^\tau) - u_i(a_i^\tau, a_{-i}^\tau)$$

  $\Rightarrow$ But: entire history may change if different actions taken!

- Thus, minimising regret not generally same as maximising utility (e.g. Crandall, 2014)

Targeted Optimality & Safety

Many algorithms designed to achieve some version of targeted optimality and safety:

Targeted Optimality & Safety

Many algorithms designed to achieve some version of targeted optimality and safety:

- If other agent's policy π_j in certain class, agent i's learning should converge to best-response

$$U_i(\pi_i, \pi_j) \approx \max_{\pi_i'} U_i(\pi_i', \pi_j)$$

Targeted Optimality & Safety

Many algorithms designed to achieve some version of targeted optimality and safety:

- If other agent's policy π_j in certain class, agent i's learning should converge to best-response

$$U_i(\pi_i, \pi_j) \approx \max_{\pi_i'} U_i(\pi_i', \pi_j)$$

- If not in class, learning should at least achieve safety (maximin) utility

$$U_i(\pi_i, \pi_j) \approx \max_{\pi_i'} \min_{\pi_j'} U_i(\pi_i', \pi_j')$$

Targeted Optimality & Safety

Many algorithms designed to achieve some version of targeted optimality and safety:

- If other agent's policy π_j in certain class, agent i's learning should converge to best-response

$$U_i(\pi_i, \pi_j) \approx \max_{\pi_i'} U_i(\pi_i', \pi_j)$$

- If not in class, learning should at least achieve safety (maximin) utility

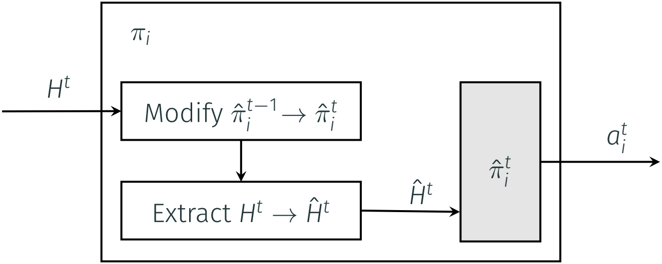
$$U_i(\pi_i, \pi_j) \approx \max_{\pi_i'} \min_{\pi_j'} U_i(\pi_i', \pi_j')$$

Policy classes: non-learning, memory-bounded, finite automata, ...

How does learning take place in policy π_i?

The internal view:

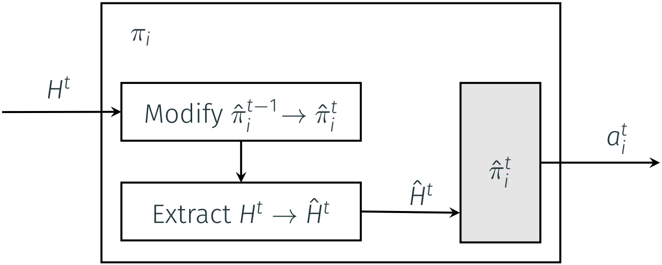
- π_i continually modifies internal policy $\hat{\pi}_i^t$ based on H^t
- $\hat{\pi}_i^t$ has own representation and input format $\hat{H}^t$



Learning Algorithms – The Internal View

Internal policy $\hat{\pi}_i^t$:

- **Representation:** Q-learning, MCTS planner, neural network, ...
- **Parameters:** Q-table, opponent model, connection weights, ...
- **Input format:** most recent state/action, abstract feature vector, ...



Fictitious Play

Simple example: Fictitious Play (FP) (Brown, 1951)

At each time t:

Simple example: Fictitious Play (FP) (Brown, 1951)

At each time t:

1. Compute opponent's action frequencies:

$$P(a_j) = \frac{1}{t+1} \sum_{\tau=0}^{t} [a_j^\tau = a_j]_1$$

Fictitious Play

Simple example: Fictitious Play (FP) (Brown, 1951)

At each time t:

1. Compute opponent's action frequencies:

$$P(a_j) = \frac{1}{t+1} \sum_{\tau=0}^{t} [a_j^\tau = a_j]_1$$

2. Compute best-response action:

$$a_i^t \in \arg\max_{a_i} \sum_{a_j} P(a_j)\, u_i(a_i, a_j)$$

Fictitious Play

Simple example: **Fictitious Play (FP)** (Brown, 1951)

At each time t:

1. Compute opponent's action frequencies:

$$P(a_j) = \frac{1}{t+1} \sum_{\tau=0}^{t} [a_j^\tau = a_j]_1$$

2. Compute best-response action:

$$a_i^t \in \arg\max_{a_i} \sum_{a_j} P(a_j)\, u_i(a_i, a_j)$$

Self-play: all agents use fictitious play

- If policies converge, policy profile is Nash equilibrium

Learning Algorithms

Many multiagent learning algorithms exist, e.g.

- Minimax-Q (Littman, 1994)
- JAL (Claus and Boutilier, 1998)
- Regret Matching (Hart and Mas-Colell, 2001, 2000)
- FFQ (Littman, 2001)
- WoLF-PHC (Bowling and Veloso, 2002)
- Nash-Q (Hu and Wellman, 2003)
- CE-Q (Greenwald and Hall, 2003)
- OAL (Wang and Sandholm, 2003)
- ReDVaLeR (Banerjee and Peng, 2004)
- GIGA-WoLF (Bowling, 2005)
- CJAL (Banerjee and Sen, 2007)
- AWESOME (Conitzer and Sandholm, 2007)
- CMLeS (Chakraborty and Stone, 2014)
- HBA (Albrecht, Crandall, and Ramamoorthy, 2016)

(Conditional) Joint Action Learning

Joint Action Learning (JAL) (Claus and Boutilier, 1998) and
Conditional Joint Action Learning (CJAL) (Banerjee and Sen, 2007)
learn Q-values for joint actions $a \in A$:

$$Q^{t+1}(a^t) = (1-\alpha)Q^t(a^t) + \alpha\, u_i^t$$

- u_i^t is utility received after joint action a^t
- $\alpha \in [0, 1]$ is *learning rate*

(Conditional) Joint Action Learning

Joint Action Learning (JAL) (Claus and Boutilier, 1998) and
Conditional Joint Action Learning (CJAL) (Banerjee and Sen, 2007)
learn Q-values for joint actions $a \in A$:

$$Q^{t+1}(a^t) = (1 - \alpha) Q^t(a^t) + \alpha \, u_i^t$$

- u_i^t is utility received after joint action a^t
- $\alpha \in [0, 1]$ is *learning rate*

Use opponent model to compute expected utilities of actions:

$$\text{JAL:} \quad E(a_i) = \sum_{a_j} P(a_j) \, Q^{t+1}(a_i, a_j)$$

$$\text{CJAL:} \quad E(a_i) = \sum_{a_j} P(a_j | a_i) \, Q^{t+1}(a_i, a_j)$$

(Conditional) Joint Action Learning

Opponent models estimated from history H^t:

- JAL:

$$P(a_j) = \frac{1}{t+1} \sum_{\tau=0}^{t} [a_j^\tau = a_j]_1$$

- CJAL:

$$P(a_j | a_i) = \frac{\sum_{\tau=0}^{t} [a_j^\tau = a_j, a_i^\tau = a_i]_1}{\sum_{\tau=0}^{t} [a_j^\tau = a_j]_1}$$

(Conditional) Joint Action Learning

Opponent models estimated from history H^t:

- JAL:

$$P(a_j) = \frac{1}{t+1} \sum_{\tau=0}^{t} [a_j^\tau = a_j]_1$$

- CJAL:

$$P(a_j | a_i) = \frac{\sum_{\tau=0}^{t} [a_j^\tau = a_j, a_i^\tau = a_i]_1}{\sum_{\tau=0}^{t} [a_j^\tau = a_j]_1}$$

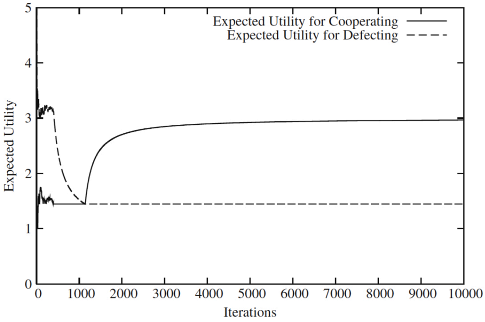
Given expected utilities $E(a_i)$, use some action exploration scheme:

- E.g. ϵ-greedy: choose $\arg\max_{a_i} E(a_i)$ with probability $1 - \epsilon$, else choose random action

(Conditional) Joint Action Learning

JAL and CJAL can converge to Nash equilibrium in self-play

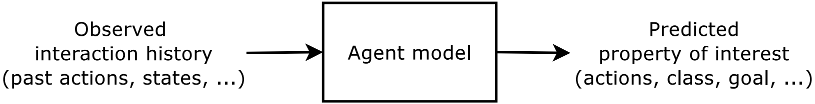
CJAL in variant of Prisoner's Dilemma (from Banerjee and Sen, 2007):



Converging to Pareto-optimal NE (C,C)

FP, JAL, CJAL are simple examples of opponent modelling:

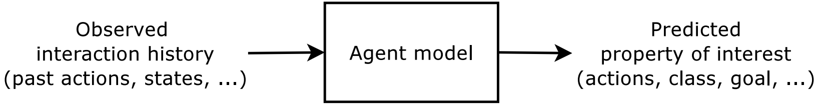
- Use model of other agent to predict its actions, goals, beliefs, ...



Opponent Modelling

FP, JAL, CJAL are simple examples of opponent modelling:

- Use model of other agent to predict its actions, goals, beliefs, ...



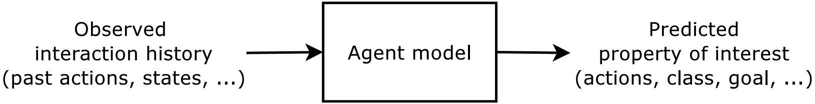
Many forms of opponent modelling exist:

- Policy reconstruction
- Type-based methods
- Classification
- Plan recognition
- Recursive reasoning
- Graphical methods
- Group modelling
- ...

Opponent Modelling

FP, JAL, CJAL are simple examples of opponent modelling:

- Use model of other agent to predict its actions, goals, beliefs, ...



Many forms of opponent modelling exist:

- Policy reconstruction
- Type-based methods
- Classification
- Plan recognition
- Recursive reasoning
- Graphical methods
- Group modelling
- ...

Upcoming survey by S. Albrecht & P. Stone!

Minimax/Nash/Correlated Q-Learning

Minimax Q-Learning (Minimax-Q) (Littman, 1994) and
Nash Q-Learning (Nash-Q) (Hu and Wellman, 2003) and
Correlated Q-Learning (CE-Q) (Greenwald and Hall, 2003)
learn joint-action Q-values for each agent $j \in N$:

$$Q_j^{t+1}(s^t, a^t) = (1 - \alpha) Q_j^t(s^t, a^t) + \alpha \left[u_j^t + \gamma E Q_j(s^{t+1}) \right]$$

Minimax/Nash/Correlated Q-Learning

Minimax Q-Learning (Minimax-Q) (Littman, 1994) and
Nash Q-Learning (Nash-Q) (Hu and Wellman, 2003) and
Correlated Q-Learning (CE-Q) (Greenwald and Hall, 2003)
learn joint-action Q-values for each agent $j \in N$:

$$Q_j^{t+1}(s^t, a^t) = (1 - \alpha) Q_j^t(s^t, a^t) + \alpha \left[u_j^t + \gamma E Q_j(s^{t+1}) \right]$$

- Assumes utilities u_j^t are commonly observed

Minimax/Nash/Correlated Q-Learning

Minimax Q-Learning (Minimax-Q) (Littman, 1994) and
Nash Q-Learning (Nash-Q) (Hu and Wellman, 2003) and
Correlated Q-Learning (CE-Q) (Greenwald and Hall, 2003)
learn joint-action Q-values for each agent $j \in N$:

$$Q_j^{t+1}(s^t, a^t) = (1-\alpha)Q_j^t(s^t, a^t) + \alpha \left[u_j^t + \gamma EQ_j(s^{t+1})\right]$$

- Assumes utilities u_j^t are commonly observed
- $EQ(s^{t+1})$ is expected utility to agent j under equilibrium profile for normal-form game with utility functions $u_j(a) = Q_j^t(s^{t+1}, a)$

  $\Rightarrow$ **Minimax-Q:** use minimax profile (assumes zero-sum game)

  $\Rightarrow$ **Nash-Q:** use Nash equilibrium

  $\Rightarrow$ **CE-Q:** use correlated equilibrium

Minimax/Nash/Correlated Q-Learning

Minimax-Q, Nash-Q, CE-Q can converge to equilibrium in self-play

- E.g. Nash-Q formal proof of convergence to NE
- But based on strong restrictions on Q_j^t!

Assumption 3 *One of the following conditions holds during learning.*[3]

Condition A. *Every stage game* $(Q_t^1(s),\ldots,Q_t^n(s))$*, for all* t *and* s*, has a global optimal point, and agents' payoffs in this equilibrium are used to update their Q-functions.*

Condition B. *Every stage game* $(Q_t^1(s),\ldots,Q_t^n(s))$*, for all* t *and* s*, has a saddle point, and agents' payoffs in this equilibrium are used to update their Q-functions.*

(Hu and Wellman, 2003)

Model-Free Learning

JAL, CJAL, Nash-Q, … learn models of other agents

- Model-based learning

Model-Free Learning

JAL, CJAL, Nash-Q, ... learn models of other agents

- Model-based learning

Can also learn without modelling other agents

- Model-free learning
- e.g. WoLF-PHC, Regret Matching

Win or Learn Fast Policy Hill Climbing

Win or Learn Fast Policy Hill Climbing (WoLF-PHC) (Bowling and Veloso, 2002) uses policy hill climbing in policy space:

$$\hat{\pi}_i^{t+1}(s^t, a_i^t) = \hat{\pi}_i^t(s^t, a_i^t) + \begin{cases} \delta & \text{if } a_i^t = \arg\max_{a_i'} Q(s^t, a_i') \\ -\frac{\delta}{|A_i|-1} & \text{else} \end{cases}$$

- Q is standard Q-learning

Win or Learn Fast Policy Hill Climbing

Win or Learn Fast Policy Hill Climbing (WoLF-PHC) (Bowling and Veloso, 2002) uses policy hill climbing in policy space:

$$\hat{\pi}_i^{t+1}(s^t, a_i^t) = \hat{\pi}_i^t(s^t, a_i^t) + \begin{cases} \delta & \text{if } a_i^t = \arg\max_{a_i'} Q(s^t, a_i') \\ -\frac{\delta}{|A_i|-1} & \text{else} \end{cases}$$

- Q is standard Q-learning

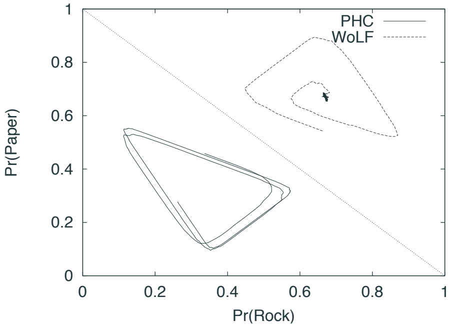
Variable learning rate δ:

$$\delta = \begin{cases} \delta_w & \text{if } \sum_{a_i} \hat{\pi}_i^t(s^t, a_i)\, Q(s^t, a_i) > \sum_{a_i} \bar{\pi}_i(s^t, a_i)\, Q(s^t, a_i) \\ \delta_l & \text{else} \end{cases}$$

- adapt slowly when "winning", fast when "losing" ($\delta_w < \delta_l$)
- $\bar{\pi}_i$ is average policy over past policies $\hat{\pi}_i$

Win or Learn Fast Policy Hill Climbing

WoLF gradient ascent in self-play converges to Nash equilibrium in two-player, two-action repeated game (Bowling and Veloso, 2002)



(b) Rock-Paper-Scissors Game

Targeted optimality: if opponent policy converges, WoLF-PHC converges to best-response against opponent

Regret Matching

Regret Matching (RegMat) (Hart and Mas-Colell, 2000) computes conditional regret for not choosing a_i' whenever a_i was chosen:

$$R(a_i, a_i') = \frac{1}{t+1} \sum_{\tau : a_i^\tau = a_i} u_i(a_i', a_j^\tau) - u_i(a^\tau)$$

Regret Matching

Regret Matching (RegMat) (Hart and Mas-Colell, 2000) computes conditional regret for not choosing a'_i whenever a_i was chosen:

$$R(a_i, a'_i) = \frac{1}{t+1} \sum_{\tau : a_i^\tau = a_i} u_i(a'_i, a_j^\tau) - u_i(a^\tau)$$

Used to modify policy:

$$\hat{\pi}_i^{t+1}(a_i) = \begin{cases} \frac{1}{\mu} \max[R(a_i^\tau, a_i), 0] & a_i \neq a_i^t \\ 1 - \sum_{a'_i \neq a_i^\tau} \hat{\pi}_i^{t+1}(a'_i) & a_i = a_i^t \end{cases}$$

- $\mu > 0$ is "inertia" parameter

Regret Matching

RegMat converges to correlated equilibrium in self-play

Assumes actions commonly observed and utility functions known

Regret Matching

RegMat converges to correlated equilibrium in self-play

Assumes actions commonly observed and utility functions known

- **Modified RegMat** (Hart and Mas-Colell, 2001) removes assumptions — only observe own action and utilities

$$R(a_i, a_i') = \frac{1}{t+1} \sum_{\tau : a_i^\tau = a_i'} \frac{\hat{\pi}_i^\tau(a_i)}{\hat{\pi}_i^\tau(a_i')} u_i^\tau - \frac{1}{t+1} \sum_{\tau : a_i^\tau = a_i} u_i^\tau$$

(plus modified policy normalisation)

Regret Matching

RegMat converges to correlated equilibrium in self-play

Assumes actions commonly observed and utility functions known

- **Modified RegMat** (Hart and Mas-Colell, 2001) removes assumptions — only observe own action and utilities

$$R(a_i, a_i') = \frac{1}{t+1} \sum_{\tau : a_i^\tau = a_i'} \frac{\hat{\pi}_i^\tau(a_i)}{\hat{\pi}_i^\tau(a_i')} u_i^\tau - \frac{1}{t+1} \sum_{\tau : a_i^\tau = a_i} u_i^\tau$$

(plus modified policy normalisation)

- Also converges to correlated equilibrium in self-play!

Bonus question: How do algorithms perform in mixed groups?

Empirical study by Albrecht and Ramamoorthy (2012):

- Tested 5 algorithms in mixed groups:
  JAL, CJAL, Nash-Q, WoLF-PHC, Modified RegMat

Learning in Mixed Groups

Bonus question: How do algorithms perform in mixed groups?

Empirical study by Albrecht and Ramamoorthy (2012):

- Tested 5 algorithms in mixed groups:
  JAL, CJAL, Nash-Q, WoLF-PHC, Modified RegMat
- Tested in all (78) structurally distinct, strictly ordinal 2×2 repeated games (Rapoport and Guyer, 1966), e.g.

1,2	2,4
4,1	3,3

Learning in Mixed Groups

Bonus question: How do algorithms perform in mixed groups?

Empirical study by Albrecht and Ramamoorthy (2012):

- Tested 5 algorithms in mixed groups:
  JAL, CJAL, Nash-Q, WoLF-PHC, Modified RegMat
- Tested in all (78) structurally distinct, strictly ordinal 2×2 repeated games (Rapoport and Guyer, 1966), e.g.

1,2	2,4
4,1	3,3

- Also tested in 500 random strictly ordinal $2 \times 2 \times 2$ (3 agents) repeated games

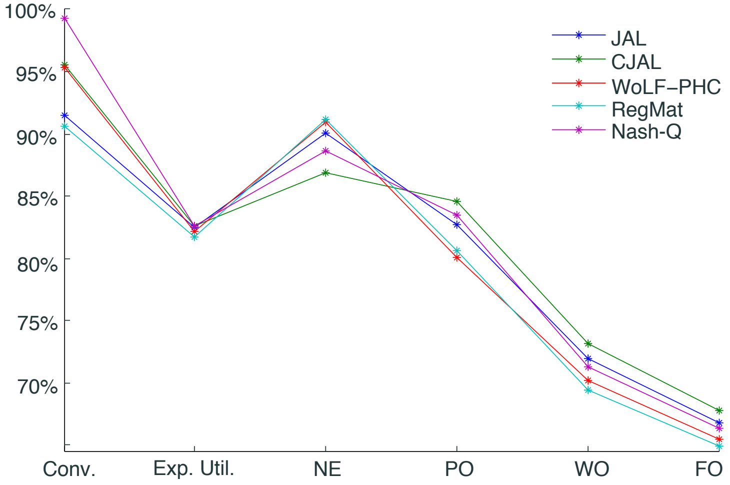
Test criteria:

- Convergence rate
- Final expected utilities
- Social welfare/fairness
- Solution rates:
  - Nash equilibrium (NE)
  - Pareto-optimality (PO)
  - Welfare-optimality (WO)
  - Fairness-optimality (FO)

Test criteria:

- Convergence rate
- Final expected utilities
- Social welfare/fairness
- Solution rates:
  - Nash equilibrium (NE)
  - Pareto-optimality (PO)
  - Welfare-optimality (WO)
  - Fairness-optimality (FO)

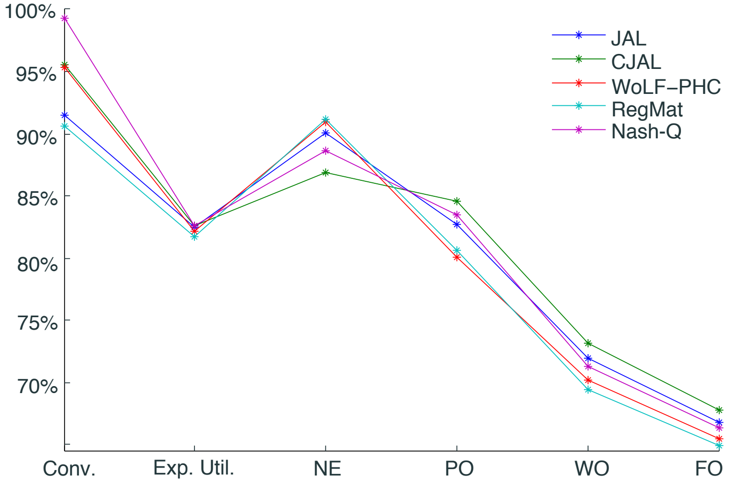
Which algorithm is best?



100% is maximum possible utility/rate

Difficulty: NE < PO < WO < FO

Learning in Mixed Groups



Answer:

No clear winner!

Overview

Introduction

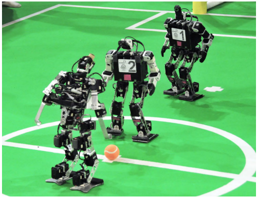
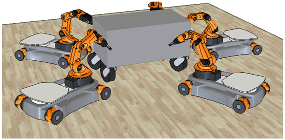
Multiagent Models & Assumptions

Learning Goals

Learning Algorithms

Recent Trends

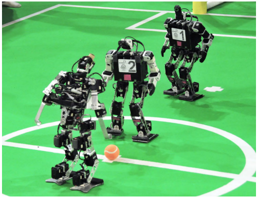
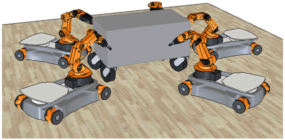
Teamwork

Typical approach:

- Whole team designed and trained by single organisation
- Agents share coordination protocols, communication languages, domain knowledge, algorithms, ...

Teamwork

Typical approach:

- Whole team designed and trained by single organisation
- Agents share coordination protocols, communication languages, domain knowledge, algorithms, ...

  $\Rightarrow$ Pre-coordination!

What if pre-coordination not possible?

What if pre-coordination not possible?

- Forming temporary teams “on the fly”

Ad Hoc Teamwork

What if pre-coordination not possible?

- Forming temporary teams “on the fly”
- Agents designed by different organisations

Ad Hoc Teamwork

What if pre-coordination not possible?

- Forming temporary teams “on the fly”
- Agents designed by different organisations
- Don’t speak same language, no knowledge of other agents’ capabilities, different beliefs, …

Ad Hoc Teamwork

What if pre-coordination not possible?

- Forming temporary teams "on the fly"
- Agents designed by different organisations
- Don't speak same language, no knowledge of other agents' capabilities, different beliefs, ...

Challenge: Ad Hoc Teamwork (Stone et al., 2010)

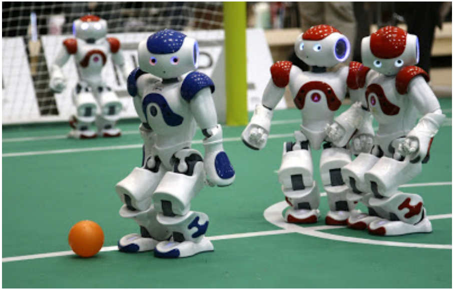
"Create an autonomous agent that is able to efficiently and robustly collaborate with previously unknown teammates on tasks to which they are all individually capable of contributing as team members."

RoboCup Drop-In Competition

RoboCup SPL Drop-In Competition '13, '14, '15 (Genter et al., 2017)

- Mixed players from different teams
- No prior coordination between players

Video: Drop-In Competition



Many learning algorithms not suitable for ad hoc teamwork:

Learning in Ad Hoc Teamwork

Many learning algorithms not suitable for ad hoc teamwork:

- RL-based algorithms (JAL, CJAL, Nash-Q, ...) need 1000's of iterations in simple games

  Ad hoc teamwork: not much time for learning, trial & error, ...

Many learning algorithms not suitable for ad hoc teamwork:

- RL-based algorithms (JAL, CJAL, Nash-Q, ...) need 1000's of iterations in simple games

  Ad hoc teamwork: not much time for learning, trial & error, ...

- Many algorithms designed for self-play (all agents use same algorithm)

  Ad hoc teamwork: no control over other agents

Learning in Ad Hoc Teamwork

Many learning algorithms not suitable for ad hoc teamwork:

- RL-based algorithms (JAL, CJAL, Nash-Q, ...) need 1000's of iterations in simple games

  Ad hoc teamwork: not much time for learning, trial & error, ...

- Many algorithms designed for self-play (all agents use same algorithm)

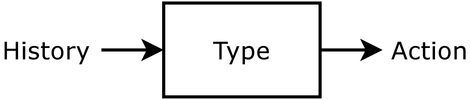
  Ad hoc teamwork: no control over other agents

Need method which can learn quickly to interact effectively with unknown other agents!

Type-Based Method

Hypothesise possible types of other agents:

- Each type θ_j is blackbox behaviour specification:

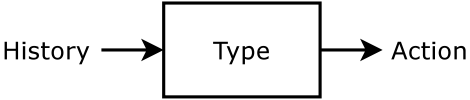


$P(a_j | H^t, \theta_j)$

Type-Based Method

Hypothesise possible types of other agents:

- Each type θ_j is blackbox behaviour specification:



$P(a_j | H^t, \theta_j)$

- Generate types from e.g.
  - experience from past interactions
  - domain and task knowledge
  - learn new types online (opponent modelling)

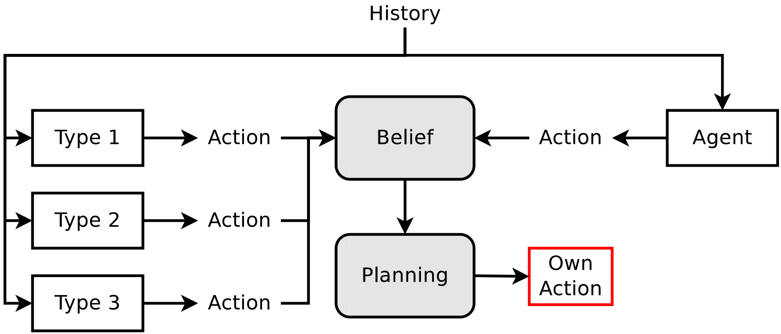
Type-Based Method

During the interaction:

- Compute belief over types based on interaction history H^t:

$$P(\theta_j|H^t) \propto P(H^t|\theta_j)\, P(\theta_j)$$

- Plan own action based on beliefs



Harsanyi-Bellman Ad Hoc Coordination (HBA) (Albrecht et al., 2016)

$$\pi_i(H^t, a_i) \sim \arg\max_{a_i} E_{s^t}^{a_i}(H^t)$$

$$E_s^{a_i}(\hat{H}) = \sum_{\theta_j} P(\theta_j|\hat{H}) \sum_{a_j} P(a_j|\hat{H}, \theta_j)\, Q_s^{(a_i,a_j)}(\hat{H})$$

$$Q_s^a(\hat{H}) = \sum_{s'} T(s, a, s') \left[u_i(s, a) + \gamma \max_{a_i} E_{s'}^{a_i} \left(\langle \hat{H}, a, s' \rangle \right) \right]$$

Harsanyi-Bellman Ad Hoc Coordination (HBA) (Albrecht et al., 2016)

$$\pi_i(H^t, a_i) \sim \arg\max_{a_i} E_{s^t}^{a_i}(H^t)$$

$$E_s^{a_i}(\hat{H}) = \sum_{\theta_j} P(\theta_j|\hat{H}) \sum_{a_j} P(a_j|\hat{H}, \theta_j)\, Q_s^{(a_i,a_j)}(\hat{H})$$

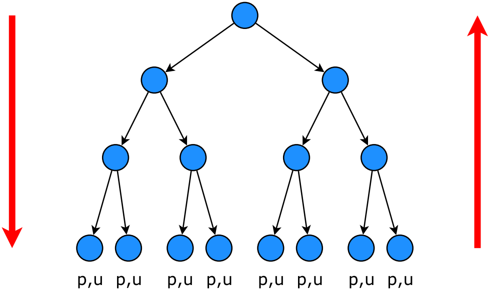
$$Q_s^{a}(\hat{H}) = \sum_{s'} T(s,a,s') \left[u_i(s,a) + \gamma \max_{a_i} E_{s'}^{a_i} \left(\langle \hat{H}, a, s' \rangle \right) \right]$$

Optimal planning with built-in exploration: Value of Information

Type-Based Method – Planning

Can compute $E_s^{a_i}$ with finite tree-expansion:

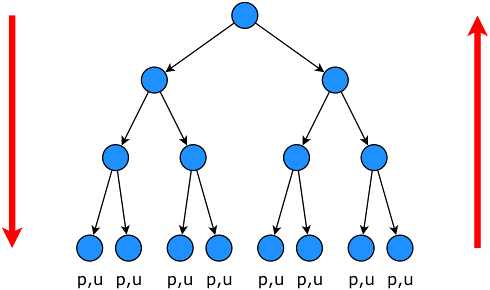
- Unfold tree of future trajectories with fixed depth
- Associate each trajectory with **p**robability and **u**tility
- Calculate expected utility of action by traversing to root



Type-Based Method – Planning

Can compute $E_s^{a_i}$ with finite tree-expansion:

- Unfold tree of future trajectories with fixed depth
- Associate each trajectory with **p**robability and **u**tility
- Calculate expected utility of action by traversing to root

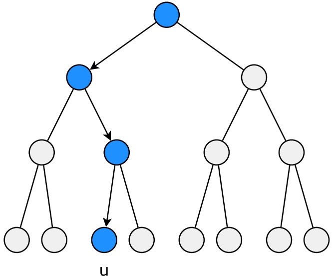


Inefficient: exponential in states, actions, agents

Use Monte-Carlo Tree Search (MCTS) for efficient approximation:

Repeat x times:

1. Sample type $\theta_j \in \Theta_j$ with probabilities $P(\theta_j | H^t)$
2. Sample interaction trajectory using θ_j and domain model T
3. Update utility estimates via backprop on trajectory



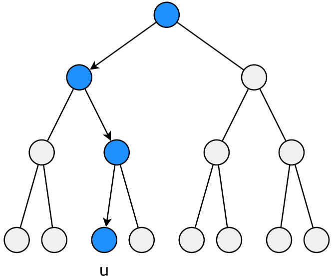
E.g. Albrecht and Stone (2017), Barrett et al. (2011)

Type-Based Method – Planning

Use Monte-Carlo Tree Search (MCTS) for efficient approximation:

Repeat x times:

1. Sample type $\theta_j \in \Theta_j$ with probabilities $P(\theta_j | H^t)$
2. Sample interaction trajectory using θ_j and domain model T
3. Update utility estimates via backprop on trajectory



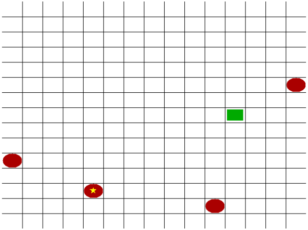
E.g. Albrecht and Stone (2017), Barrett et al. (2011)

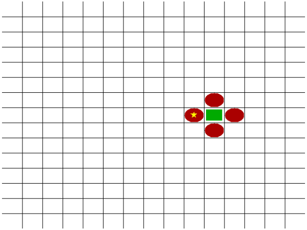
But: loses value of information! (no belief change during planning)

Ad Hoc Teamwork: Predator Pursuit

4 predators must capture 1 prey in grid world (Barrett et al., 2011)

- We control one agent in predator team
- Policies of other predators unknown (prey moves randomly)
- 4 types provided to our agent; online planning using MCTS





Video: 4 types, true type inside

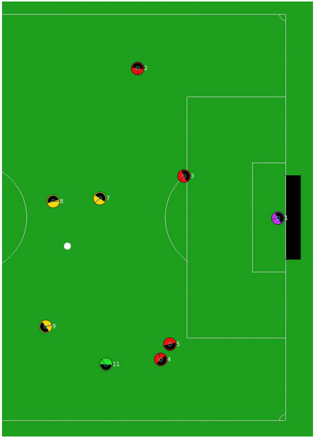
Video: 4 types, true type outside (students)

Source: `http://www.cs.utexas.edu/~larg/index.php/Ad_Hoc_Teamwork:_Pursuit`

Ad Hoc Teamwork: Half Field Offense

4 offense players vs. 5 defense players (Barrett and Stone, 2015)

- We control one agent (green) in offensive team (yellow)
- Policies of teammates unknown (defense uses fixed policies)
- 7 team types provided to our agent; for each team type, plan own policy offline using RL



Video: 4v5 Half Field Offense

Source:
`http://www.cs.utexas.edu/~larg/index.php/Ad_Hoc_Teamwork:_HFO`

Learning Parameters in Types

We can learn more: parameters in types! (Albrecht and Stone, 2017)

$$\text{History} \longrightarrow \boxed{\text{Type + Parameters}} \longrightarrow \text{Action}$$

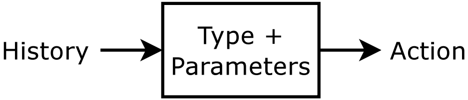
$$P(a_i | H^t, \theta_j, p)$$

- $p = (p_1, ..., p_k)$ continuous parameter vector
- Complex types can have several parameters

  $\Rightarrow$ learning rate, exploration rate, discount factor, ...

Learning Parameters in Types

We can learn more: parameters in types! (Albrecht and Stone, 2017)



$P(a_i | H^t, \theta_j, p)$

- $p = (p_1, ..., p_k)$ continuous parameter vector
- Complex types can have several parameters

  $\Rightarrow$ learning rate, exploration rate, discount factor, ...

Goal: simultaneously learn type and parameters in type

For each type $\theta_j \in \Theta_j$, maintain parameter estimate $p \in [p^{\min}, p^{\max}]$

Observe action a_j^t of agent j

Select types $\Phi \subset \Theta_j$ for updating

For each $\theta_j \in \Phi$, update estimate $p^t \rightarrow p^{t+1}$

Update beliefs:

$$P(\theta_j | H^{t+1}) \propto P(a_j^t \mid H^t, \theta_j, p^{t+1}) \, P(\theta_j | H^t)$$

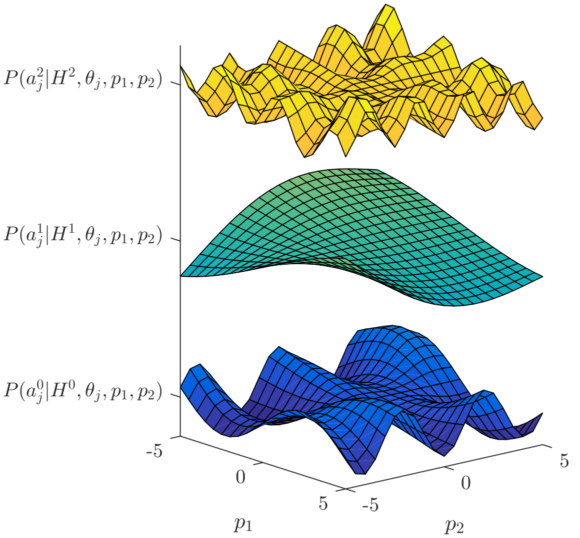
Plan own action

Updating Parameter Estimates

Given type θ_j, update parameter estimate $p^t \rightarrow p^{t+1}$

Type defines action likelihoods

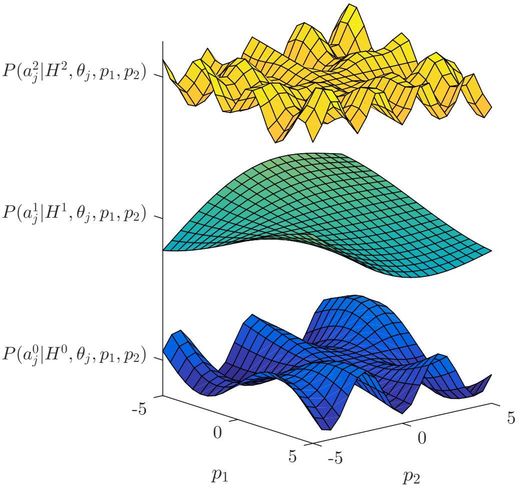
$$P(a_j^t \mid H^t, \theta_j, p)$$



Updating Parameter Estimates

Bayesian updating:

- Approximate $P(a_j^t \mid H^t, \theta_j, p)$ as polynomial with variables p
- Perform conjugate updates through successive layers



Bayesian updating:

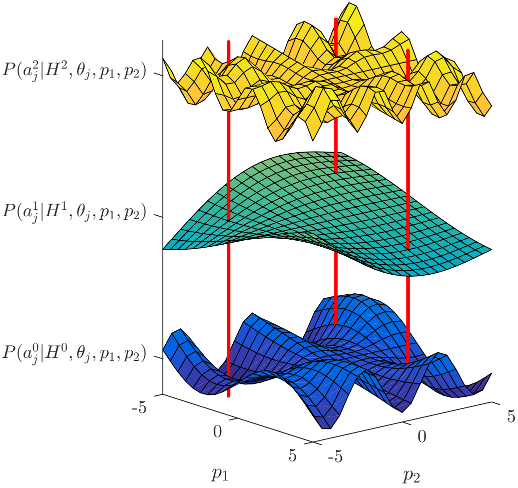
- Approximate $P(a_j^t \mid H^t, \theta_j, p)$ as polynomial with variables p
- Perform conjugate updates through successive layers

Global optimisation:

$$\arg\max_p \prod_{\tau=1}^{t+1} P(a_j^{\tau-1} | H^{\tau-1}, \theta_j, p)$$

Solve with Bayesian optimisation

(Albrecht and Stone, 2017)

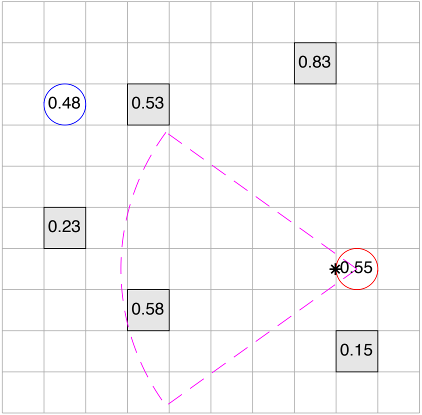


Ad Hoc Teamwork: Level-Based Foraging

Blue = our agent, red = other agents

Goal: collect all items in minimal time

Agents can collect item if
sum of agent levels $\geq$ item level



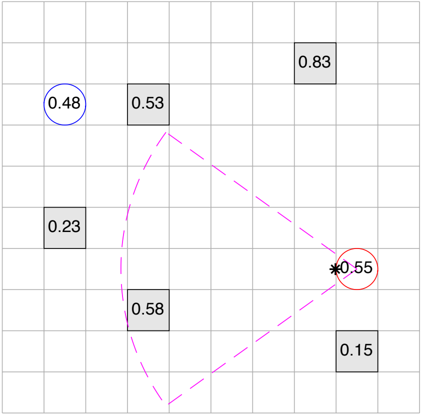
Blue = our agent, red = other agents

Goal: collect all items in minimal time

Agents can collect item if
sum of agent levels $\geq$ item level

4 possible types for red, e.g.

- search for item, try to load
- search for agent, load item closest to agent



Ad Hoc Teamwork: Level-Based Foraging

Blue = our agent, red = other agents

Goal: collect all items in minimal time

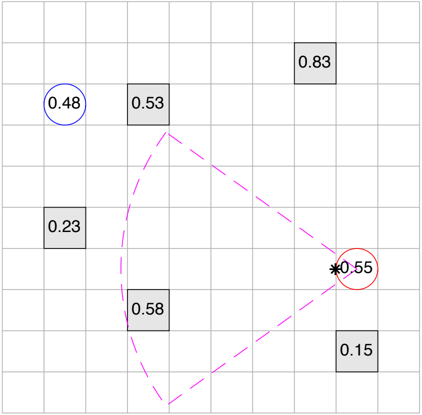
Agents can collect item if
sum of agent levels $\geq$ item level

4 possible types for red, e.g.

- search for item, try to load
- search for agent, load item closest to agent

Each type uses 3 parameters:

- skill level, view radius, view angle



Ad Hoc Teamwork: Level-Based Foraging

Blue = our agent, red = other agents

Goal: collect all items in minimal time

Agents can collect item if
sum of agent levels $\geq$ item level

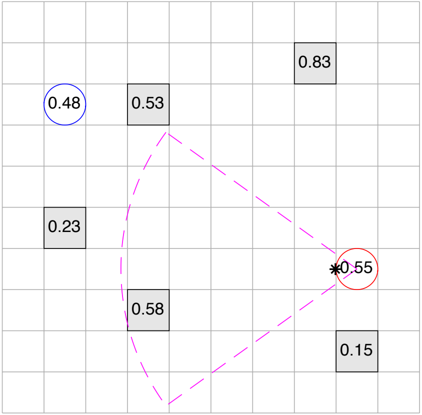
4 possible types for red, e.g.

- search for item, try to load
- search for agent, load item closest to agent

Each type uses 3 parameters:

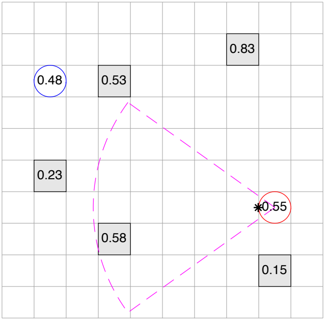
- skill level, view radius, view angle

Blue doesn't know true type of red nor parameter values of type

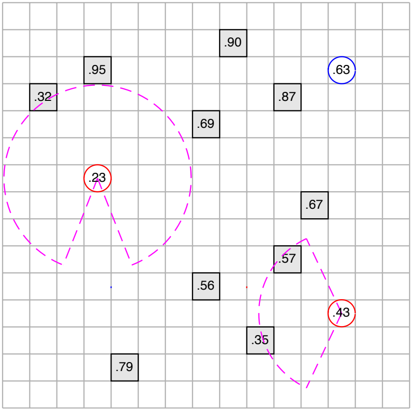


Ad Hoc Teamwork: Level-Based Foraging

Video: 10x10 world, 2 agents



Video: 15x15 world, 3 agents



- AAAI'16 Tutorial on Type-Based Methods:
  `http://thinc.cs.uga.edu/tutorials/aaai-16.html`
- Special Issue on Multiagent Interaction without Prior Coordination (MIPC): `http://mipc.inf.ed.ac.uk/journal`
- MIPC Workshops:
  - AAMAS'17, Sao Paulo, Brazil
  - AAAI'16, Phoenix, Arizona, USA
  - AAAI'15, Austin, Texas, USA
  - AAAI'14, Quebec City, Canada

  `http://mipc.inf.ed.ac.uk`

Standard Q-learning assumes tabular representation:

- One entry in Q for each (s, a)

Deep Reinforcement Learning

Standard Q-learning assumes tabular representation:

- One entry in Q for each (s, a)

  $\Rightarrow$ Does not scale to complex domains!

Standard Q-learning assumes tabular representation:

- One entry in Q for each (s, a)

  $\Rightarrow$ Does not scale to complex domains!

  $\Rightarrow$ Does not generalise values!

Standard Q-learning assumes tabular representation:

- One entry in Q for each (s, a)

  $\Rightarrow$ Does not scale to complex domains!

  $\Rightarrow$ Does not generalise values!

Needs extra engineering to work, including:

- **State abstraction** to reduce state space (usually hand-coded & domain-specific)
- **Function approximation** to store and generalise Q (e.g. linear function approximation in state features)

New problem: extra engineering may limit performance!

- State abstraction may be wrong (e.g. too coarse)
- Function approximator may be inaccurate

New problem: extra engineering may limit performance!

- State abstraction may be wrong (e.g. too coarse)
- Function approximator may be inaccurate

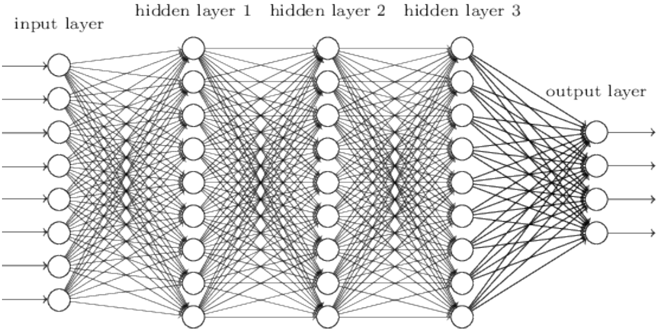
Idea: deep reinforcement learning

- Use "deep" neural network to represent Q
- Learn on raw data (no state abstraction)

$\Rightarrow$ Let network learn good abstraction on its own!

Deep learning: neural network with many layers

- Input layer takes raw data $\rightarrow s$
- Hidden layers transform data
- Output layer returns target scalars $\rightarrow Q(s, \cdot)$
- Train network with back-propagation on labelled data



Deep Q-Learning

Deep Q-Learning (Mnih et al., 2013)

Initialise network parameters Ψ with random weights

Deep Q-Learning

Deep Q-Learning (Mnih et al., 2013)

Initialise network parameters Ψ with random weights

1. Observe current state s^t

Deep Q-Learning (Mnih et al., 2013)

Initialise network parameters Ψ with random weights

1. Observe current state s^t
2. With probability ϵ, select random action a^t
   Else, select action $a^t \in \arg\max_a Q(s^t, a; \Psi)$

Deep Q-Learning

Deep Q-Learning (Mnih et al., 2013)

Initialise network parameters Ψ with random weights

1. Observe current state s^t
2. With probability ϵ, select random action a^t
   Else, select action $a^t \in \arg\max_a Q(s^t, a; \Psi)$
3. Get reward r^t and new state s^{t+1}

Deep Q-Learning

Deep Q-Learning (Mnih et al., 2013)

Initialise network parameters Ψ with random weights

1. Observe current state s^t
2. With probability ϵ, select random action a^t
   Else, select action $a^t \in \arg\max_a Q(s^t, a; \Psi)$
3. Get reward r^t and new state s^{t+1}
4. Store experience (s^t, a^t, r^t, s^{t+1}) in D

Deep Q-Learning

Deep Q-Learning (Mnih et al., 2013)

Initialise network parameters Ψ with random weights

1. Observe current state s^t
2. With probability ϵ, select random action a^t
   Else, select action $a^t \in \arg\max_a Q(s^t, a; \Psi)$
3. Get reward r^t and new state s^{t+1}
4. Store experience (s^t, a^t, r^t, s^{t+1}) in D
5. Sample random minibatch $D^+ \subset D$

Deep Q-Learning (Mnih et al., 2013)

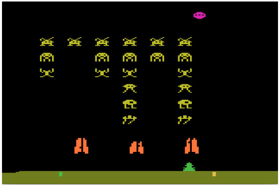
Initialise network parameters Ψ with random weights

1. Observe current state s^t
2. With probability ϵ, select random action a^t
   Else, select action $a^t \in \arg\max_a Q(s^t, a; \Psi)$
3. Get reward r^t and new state s^{t+1}
4. Store experience (s^t, a^t, r^t, s^{t+1}) in D
5. Sample random minibatch $D^+ \subset D$
6. For each $(s^\tau, a^\tau, r^\tau, s^{\tau+1}) \in D^+$, perform gradient descent step on

$$(y^\tau - Q(s^\tau, a^\tau; \Psi))^2$$

$$y^\tau = r^\tau + \gamma \max_{a'} Q(s^\tau, a'; \Psi^{fixed})$$

Multiagent Deep Reinforcement Learning



Deep RL very successful at many singe-agent games

- e.g. Atari games, Go, 3D maze navigation, ...

Can we use Deep RL for multiagent learning?

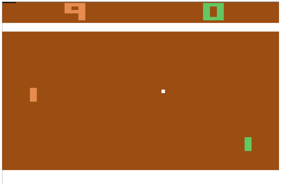
- **Problem:** learning of other agents makes environment non-stationary (breaks Markov property)

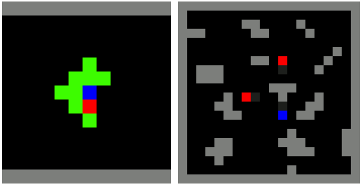
Independent Deep Q-Learners

Video: Cooperative Pong
Video: Competitive Pong

(Tampuu et al., 2017)

`https://www.youtube.com/watch?v=Gb9DprIgdGw`
`https://www.youtube.com/watch?v=nn6_GUVDnVw`





Video: Gathering game
Video: Wolfpack game

(Leibo et al., 2017)

`https://www.youtube.com/watch?v=F97lqqpcqsM`
`https://www.youtube.com/watch?v=kXudpMfecs4`

Video: Starcraft

(Foerster et al., 2017)

`https://www.youtube.com/watch?v=RK7y_uQmwhw`

Multiagent Deep Reinforcement Learning

Some recent works on multiagent deep RL:

- Emergence of cooperative/competitive behaviours
  (Tampuu et al., 2017; Leibo et al., 2017)
- Learning communication protocols
  (Sukhbaatar et al., 2016; Foerster et al., 2016)
- Opponent modelling
  (He et al., 2016)
- Improved minibatch selection
  (Palmer et al., 2017; Foerster et al., 2017)
- Multi-task learning
  (Omidshafiei et al., 2017)
- Learning value decomposition
  (Sunehag et al., 2017)

Concluding Remarks

We covered...

- Multiagent models: normal-form games, repeated games, stochastic games, ...
- Learning goals: equilibria, no-regret, targeted optimality, ...
- Learning algorithms: internal view, model-based, model-free
- Recent trends: ad hoc teamwork, deep RL

Download tutorial slides at:
`http://www.cs.utexas.edu/~larg/ijcai17_tutorial`

Watch out for our upcoming survey on agents modelling other agents!

S. Albrecht and S. Ramamoorthy. Comparative evaluation of MAL algorithms in a diverse set of ad hoc team problems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 349–356, 2012.

S. Albrecht and S. Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. Technical report, https://arxiv.org/abs/1506.01170, 2013.

S. Albrecht and P. Stone. Reasoning about hypothetical agent behaviours and their parameters. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*, pages 547–555, 2017.

S. Albrecht, J. Crandall, and S. Ramamoorthy. Belief and truth in hypothesised behaviours. *Artificial Intelligence*, 235:63–94, 2016.

E. Alonso, M. D'Inverno, D. Kudenko, M. Luck, and J. Noble. Learning in multi-agent systems. *Knowledge Engineering Review*, 16(3):277–284, 2001.

References ii

R. Aumann. Subjectivity and correlation in randomized strategies. *Journal of mathematical Economics*, 1:67–96, 1974.

B. Banerjee and J. Peng. Performance bounded reinforcement learning in strategic interactions. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence*, pages 2–7, 2004.

D. Banerjee and S. Sen. Reaching pareto-optimality in prisoner's dilemma using conditional joint action learning. *Autonomous Agents and Multi-Agent Systems*, 15(1):91–108, 2007.

S. Barrett and P. Stone. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2010–2016, 2015.

S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 567–574, 2011.

M. Bowling. Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems*, pages 209–216, 2005.

M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

G. Brown. Iterative solution of games by fictitious play. In *Proceedings of the Conference on Activity Analysis of Production and Allocation, Cowles Commission Monograph 13*, pages 374–376, 1951.

L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(2), 2008.

D. Chakraborty and P. Stone. Multiagent learning in the presence of memory-bounded agents. *Autonomous Agents and Multi-Agent Systems*, 28(2):182–213, 2014.

References iv

C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 746–752, 1998.

V. Conitzer and T. Sandholm. AWESOME: a general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, 2007.

J. Crandall. Towards minimizing disappointment in repeated games. *Journal of Artificial Intelligence Research*, 49:111–142, 2014.

A. Fink. Equilibrium in a stochastic n-person game. *Journal of Science of the Hiroshima University*, 28(1):89–93, 1964.

J. Foerster, Y. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pages 2137–2145, 2016.

References v

J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1146–1155, 2017.

K. Genter, T. Laue, and P. Stone. Three years of the RoboCup standard platform league drop-in player competition: Creating and maintaining a large scale ad hoc teamwork robotics competition. *Autonomous Agents and Multi-Agent Systems*, 31(4):790–820, 2017.

A. Greenwald and K. Hall. Correlated Q-learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 242–249, 2003.

S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

S. Hart and A. Mas-Colell. A reinforcement procedure leading to correlated equilibrium. *Economic Essays: A Festschrift for Werner Hildenbrand*, pages 181–200, 2001.

H. He, J. Boyd-Graber, K. Kwok, and H. Daumé III. Opponent modeling in deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1804–1813, 2016.

J. Hu and M. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 464–473, 2017.

M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

M. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the 18th International Conference on Machine Learning*, pages 322–328, 2001.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning*, pages 2681–2690, 2017.

G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani. Lenient multi-agent deep reinforcement learning. *arXiv preprint arXiv:1707.04402*, 2017.

L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

A. Rapoport and M. Guyer. A taxonomy of 2×2 games. *General Systems: Yearbook of the Society for General Systems Research*, 11:203–214, 1966.

S. Sen and G. Weiss. Learning in multiagent systems. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 6, pages 259–298. MIT Press, 1999.

L. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.

Y. Shoham, R. Powers, and T. Grenager. Multi-agent reinforcement learning: A critical survey. Unpublished survey, 2003.

P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad hoc autonomous agent teams: collaboration without pre-coordination. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1504–1509, 2010.

S. Sukhbaatar, A. Szlam, and R. Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems 29*, pages 2244–2252, 2016.

P. Sunehag, G. Lever, A. Gruslys, W. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Leibo, K. Tuyls, and T. Graepel. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS ONE*, 12(4):e0172395, 2017.

K. Tuyls and G. Weiss. Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, 33(3):41, 2012.

J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

X. Wang and T. Sandholm. Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In *Advances in Neural Information Processing Systems 15*, pages 1603–1610, 2003.