Multiagent Planning and Decision Making

CS4246/CS5446

AI Planning and Decision Making

Optional content.
Non examinable

Topics

- Multiagent Planning and Decision Making (18.1)
  - Properties and environments
- Distributed Decision Making
  - Basic concepts
  - Multiagent reinforcement learning
    - Example: OpenAI Five

Multiagent Planning and Decision Making

Co-ordination and co-operation

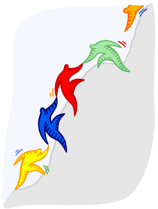
Extended Single Agent Planning

- Extended problems
  - Various degrees of decomposition for single agent
  - Sharing centralized goal and common objective
- Problem types
  - Multieffector planning
  - Multibody planning
  - Decentralized planning

Extended Single Agent Planning

- Multieffector planning
  - Use of different sensors and effectors on same body
- Multibody planning
  - Pooling of sensor information to form common estimate
- Decentralized planning
  - Planning centralized but execution partially decoupled
  - Communication may be needed

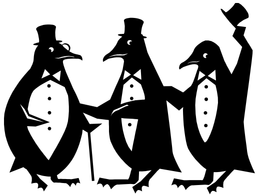
Multiagent Planning



- Multiple agents
  - Each tries to achieve its own goals with help or hindrance of the others
  - May not share the same goal(s)
  - Coordinate to achieve overall goal/objective/behavior
- Mixed problems
  - Mixed centralized and multi-agent planning
    - Need co-ordination and communication
    - May use incentive for reaching the overall goal

Examples

- Which type of planning is involved below?
  - Swarming penguins with central command
  - Tennis players who form doubles team
  - Opposing players and spectators in a football game
  - Teaching staff for a course consists of lecturers, teaching assistants, and lab tutors
  - Courier services such as SpeedPost or Federal Express?
  - Completing homework on Telegram or GoogleDoc?



Main Issues

- Main issues in multiagent planning:
  - Representing and planning for multiple, simultaneous actions
    - From multieffector to multiagent planning settings
  - Cooperation, coordination, and competition
    - In true multiagent settings
- Main computational perspectives:
  - Algorithms, probability and decision theory, game theory, and logic
    - Extended to multieffector, multibody, and multiagent settings

Multi Simultaneous Actions

Multiple Simultaneous Actions

- Multiactor settings
  - Multieffector, multibody, and multiagent settings
- Actors
  - Refer to effectors, bodies, and agents
- Main issues:
  - How to define transition models, correct plans, and efficient planning algorithms?

Some Definitions

- Correct plan
  - A plan that, if executed by the actors, achieves the goal
- Perfect synchronization
  - Each action takes the same amount of time and actions at each point in the joint plan are simultaneous
- Transition model
  - Recall: for deterministic case
  - Transition model: $Result(s, a)$
  - Where s is a state, a is an action

Joint Actions

- In single agent setting:
  - There might be b different choices for the action (b can be large)
  - Use action schemas to provide concise representation
- In multiactor setting:
  - With n actors, define joint action $< a_1, \dots, a_n >$
    where a_i is action taken by i^{th} actor
- Problems
  - How to describe transition model for b^n different joint actions?
  - How to work with joint planning problem with branching factor b^n?

Managing Complexity

- Research on multiactor planning:
  - How to decouple the actors, so that complexity of the problem grows linearly with n rather than exponentially?
- Possible conditions:
  - N actors have no interaction with each other → N separate problems
  - N actors are loosely coupled → Pretend problems are completely decoupled and then fix up interactions
    - For transition model, write independent action schemas

Example: Doubles Tennis Problem

$$
\begin{aligned}
&\mathit{Actors}(A, B)\\
&\mathit{Init}(\mathit{At}(A, \mathit{LeftBaseline}) \wedge \mathit{At}(B, \mathit{RightNet}) \wedge\\
&\qquad \mathit{Approaching}(\mathit{Ball}, \mathit{RightBaseline}) \wedge \mathit{Partner}(A, B) \wedge \mathit{Partner}(B, A)\\
&\mathit{Goal}(\mathit{Returned}(\mathit{Ball}) \wedge (\mathit{At}(x, \mathit{RightNet}) \vee \mathit{At}(x, \mathit{LeftNet}))\\
&\mathit{Action}(\mathit{Hit}(\mathit{actor}, \mathit{Ball}),\\
&\qquad \text{PRECOND:} \mathit{Approaching}(\mathit{Ball}, \mathit{loc}) \wedge \mathit{At}(\mathit{actor}, \mathit{loc})\\
&\qquad \text{EFFECT:} \mathit{Returned}(\mathit{Ball}))\\
&\mathit{Action}(\mathit{Go}(\mathit{actor}, \mathit{to}),\\
&\qquad \text{PRECOND:} \mathit{At}(\mathit{actor}, \mathit{loc}) \wedge \mathit{to} \neq \mathit{loc},\\
&\qquad \text{EFFECT:} \mathit{At}(\mathit{actor}, \mathit{to}) \wedge \neg \mathit{At}(\mathit{actor}, \mathit{loc}))
\end{aligned}
$$

Figure 18.1 The doubles tennis problem. Two actors, A and B, are playing together and can be in one of four locations: $\mathit{LeftBaseline}$, $\mathit{RightBaseline}$, $\mathit{LeftNet}$, and $\mathit{RightNet}$. The ball can be returned only if a player is in the right place. The NoOp action is a dummy, which has no effect. Note that each action must include the actor as an argument.

Source: RN Fig. 18.1

Example: Doubles Tennis Problem

- Goal at one point of game:
  - Returning ball that has been hit to team and ensuring that at least one of the agents is covering the net
- Plan 1:
  - A: $[Go(A, RightBaseline), Hit(A, Ball)]$
  - B: $[NoOp(B), NoOp(B)]$
- Plan 2:
  - A: $[Go(A, RightBaseline), Hit(A, Ball)]$
  - B: $[Go(B, RightBaseline), Hit(B, Ball)]$
- Would both plans work?
  - Not all the time! Why?

Example: Doubles Tennis Problem

- Recall:

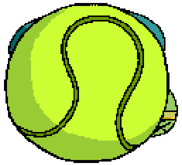
$$
\begin{aligned}
&Action(Hit(a, Ball),\\
&\quad Precond:\ \ Approaching(Ball, loc) \wedge At(a, loc)\\
&\quad Effect:\ \ Returned(Ball)
\end{aligned}
$$

- Plan 2:
  - A: $[Go(A, RightBaseline), Hit(A, Ball)]$
  - B: $[Go(B, RightBaseline), Hit(B, Ball)]$
- Problem:
  - When a plan has both agents hitting ball at the same time
  - Plan 2 won't work in the real world in this case
  - But Action schema for Hit says that the ball will be return successfully!

Concurrent Actions

- Challenge:
  - Preconditions constrain the state in which an action can be executed successfully, but do not constrain other actions that might mess it up
- Concurrent action list
  - Augment action schemas to state which actions must or must not be executed simultaneously

Preventing Intervening Actions



- Changing the Hit action schema:

$$
\begin{aligned}
&Action(Hit(a, Ball),\\
&\quad \boxed{Concurrent:\ b \neq a \rightarrow \neg Hit(b, Ball)}\\
&\quad Precond:\ \ Approaching(Ball, loc) \wedge At(a, loc)\\
&\quad Effect:\ \ Returned(Ball)
\end{aligned}
$$

- Meaning:
  - Hit has the stated effect only if no other Hit action by another agent occurs at the same time

Incorporating Enabling Actions

- Example:

  $Action(Carry(a, cooler, here, there)$

  $Concurrent: b \neq a \wedge Carry(b, cooler, here, there)$

  $Precond: At(a, here) \quad At(cooler, here) \quad Cooler(cooler)$

  $Effect: At(a, there) \quad At(cooler, there) \quad At(a, here) \quad At(cooler, here))$

- Meaning:
  - Desired effect achieved only when another action occurs concurrently

Multiactor Planning

- Loose coupling of subplans
  - Concurrency constraints are rare during plan search
  - Only minor modifications to planning algorithms needed
  - Most single agent planning heuristics are effective
- Extensions
  - Extensions to HTNs, partial observability, conditionals, execution monitoring, and replanning are possible

Multiple Agents

Cooperation and Coordination

- Assumptions
  - Goals and knowledge base are shared
- In multiagent setting
  - Each agent makes its own plan
- Difference from multibody case
  - More than one joint solution exists

Example: Doubles Tennis Problem

- Recall: Plan 1:
  - A: $[Go(A, RightBaseline), Hit(A, Ball)]$
  - B: $[NoOp(B), NoOp(B)]$
- Plan 3 (New!):
  - A: $[Go(A, LeftNet), NoOp(A)]$
  - B: $[Go(B, RightBaseline), Hit(B, Ball)]$
- Problem:
  - If both agents can agree on Plan 1 or Plan 3, goal achieved
  - If A chooses Plan 1 and B chooses Plan 3, nobody will return the ball; conversely, both will try to hit the ball

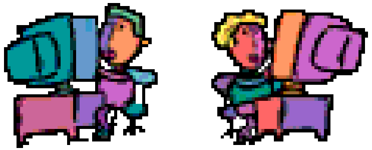
Convention

- Convention
  - Any constraint on the selection of joint plan
  - Adopted by all agents before engaging in joint activity
  - Becomes “social laws” when widespread
  - Can arise through evolutionary processes

Examples

- Establishing conventions in real-life
  - In doubles tennis: “stick to you side of the court”
  - Drivers on the road follows convention of traffic lights
  - Development of human language – adopts convention of “majority rules”
  - Behavior in colony of ants

Communication



- Communication
  - To achieve common knowledge of a feasible joint plan
  - Can work with cooperative or competitive agents
- Examples
  - Shouting “Mine!” or “Yours!” in a doubles tennis game
  - Communication through action (instead of words), as part of a pre-communicated strategy
    - One agent heads for the net, the other stays in Baseline

Co-ordination

- Co-ordination
  - To adjust actions to accommodate other agents' behavior
  - Can be pre-determined through convention or established through communication or plan recognition
- Plan recognition
  - Recognizing strategy through actions of other agents
  - One approach to coordination
  - Works when a single action (or a short sequence of actions) is enough to determine a joint plan unambiguously

Example: Flocking Behavior of Birds

- Simulation of flight for bird agents (boids)
  - Observe positions of nearest neighbors and choose heading and acceleration that maximize weighted sum of:

1. Cohesion: positive score of getting closer to average position of the neighbors
2. Separation: negative score for getting too close to any one neighbor
3. Alignment: positive score of getting closer to the average heading of the neighbors

Example: Flocking Behavior of Birds

- Emergent behavior
  - All boids execute policy fly as a pseudorigid body with roughly constant density that does not disperse over time; occassionally make sudden swooping motions
  - No need for each agent to possess a joint plant that models the actions of the other agents!

Example: Boids

- The Boids Algorithm:
  - Developed by Craig Reynolds (1987)
  - Bats and penguins emergent behavioral movements in Batman Returns (1992)
- Background and (old) updates by Craig Reynolds:
  - http://www.red3d.com/cwr/boids/
  - On youtube: https://youtu.be/xBniZYiyrb4
- Some examples: (Still in practice today)
  - Stampede scene from Disney's "The Lion King" (wait for script to load): https://www.lionking.org/movies/Stampede.mpg
  - 500,000 boids: https://youtu.be/n0PcN0K8EVI
  - Look for some iPhone/iPod/iPad applications

Distributed Decision Making

Multiagent constraint satisfaction and optimization

Distributed Decision Making Techniques

- Distributed constraint satisfaction (Classic approaches)
  - Domain-pruning algorithms
  - Heuristic search algorithms
- Distributed optimization
  - Distributed dynamic programming for path finding
  - Distributed solutions to MDPs
  - Multiagent reinforcement learning (MARL)
  - Economics-inspired optimization algorithms
  - Co-ordination via social laws and conventions

Centralized Planning for Distributed Plans

- Problem definition:
  - Given goal description, set of action schemas, and initial state description
  - Generate a partial order plan.
  - Bias search to find plan with minimal ordering constraints among the steps.
- Steps:
  - Decompose the plan into subplans with least ordering commitment in constraints
  - Insert synchronization actions into subplans.
  - Allocate subplans to agents
  - If failure, return to previous steps
  - If success, insert remaining bindings into subplans
  - Initiate plan execution, and optionally monitor progress

Distributed Planning for Centralized Plans

- Overall planning
  - Decompose overall task and distribute among different specialist agents, each of which generates its own plan
- Tasks Sharing
  - Communication through partial plans
  - Partial plans can be reused and redistributed
  - E.g., manufacturing planning, unman vehicles, logistics
- Results sharing
  - Share partial plans generated in parallel
  - Merge partial plans together and fix inconsistencies
  - Distributed constraint satisfaction search
  - E.g., communication networks

Distributed Planning for Distributed Plans

- Plan merging
  - Use reachability and interaction analysis to detect and fix unsafe situations
  - Plan synchronization and scheduling
  - Distributed constraint satisfaction problem
- Steps
  1. assign goals to agents;
  2. agents formulate local plans;
  3. local plans are exchanged and combined;
  4. messaging and/or timing commitments imposed to resolve negative plan interactions.

Distributed Planning for Distributed Plans

- Iterative Plan Formation
  - Incorporate global constraints
  - Exploit hierarchical structure of plan space to perform distributed hierarchical planning.
- Negotiation, contracts, and auctions
  - Establish laws or conventions

Multiagent Reinforcement Learning

- Cooperative multiagent decision making with decentralized planning
  - Collectively learn, interact, and collaborate with each other
- Examples:
  - OpenAI Five on Dota 2
    - https://openai.com/five/
  - AlphaStar on Starcraft
    - https://deepmind.com/blog/article/AlphaStar-Grandmaster-level-in-StarCraft-II-using-multi-agent-reinforcement-learning
  - OpenAI Hide and Seek
    - https://youtu.be/kopoLzvh5jY
- Main challenges[1]:
  - Exponentially increasing state and action space with respect to no. of agents
  - Non-stationary transitions that depend on joint actions, with possibly changing policies
  - Balancing between centralized and decentralized planning, collaboration and competition, with limited resources

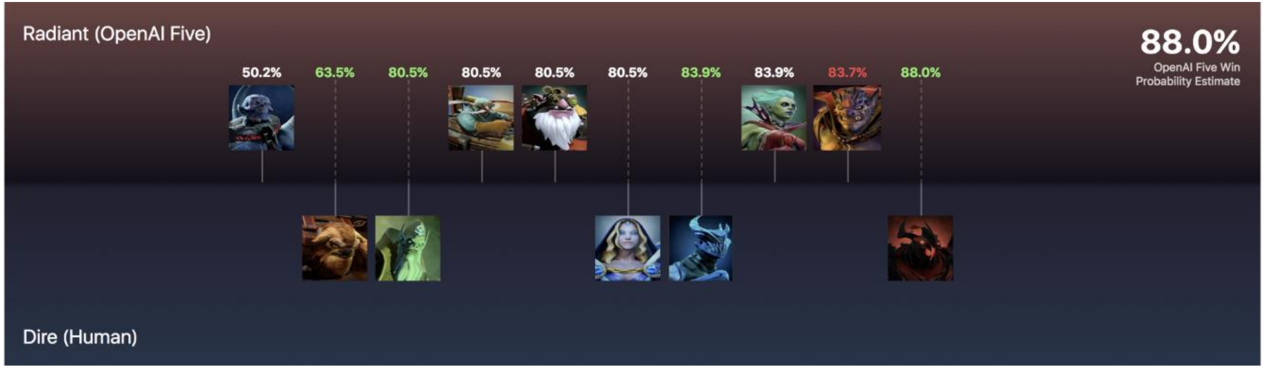
[1]Gronauer, S. and K. Diepold, *Multi-agent deep reinforcement learning: a survey.* Artificial Intelligence Review, 2021.

OpenAI Five

Multi-agent (Deep) Reinforcement Learning in POMDP

OpenAI Five Beat Top Human Players at Dota2[1]

- OpenAI vs human players
  - Policy gradient (Proximal Policy Optimization) with Recurrent neural networks (LSTM)
  - Beat human world champion Dota2 team (April 2019)



Watch: https://www.youtube.com/watch?v=eHipy_j29Xw

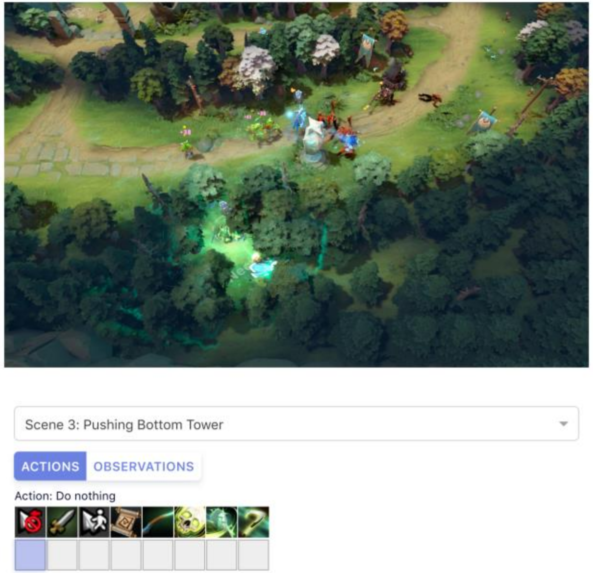
[1] Material from this section mostly from https://openai.com/blog/openai-five/ and https://www.linkedin.com/pulse/science-behind-openai-five-just-produced-one-greatest-jesus-rodriguez

Dota 2

- Multiplayer online battle arena game with 2 teams of 5 players
  - Destroy structure defended by the opposition; defend own structure
  - High stakes!! $40mil prze money
- Average game length: 45 min, 30 fps
  - OpenAI Five observes every 4^{th} frame giving about $20{,}000$ moves
  - Partial observability: Map directly around players visible, rest covered by fog-of-war
  - High dimensional continuous action space: discretized to ~ 1000 valid actions each clock tick. $170{,}000$ possible actions per hero
  - High dimensional continuous observation space: observes $20{,}000$ numbers (all information a human can access)

Very large space to explore!

OpenAI Five in Action

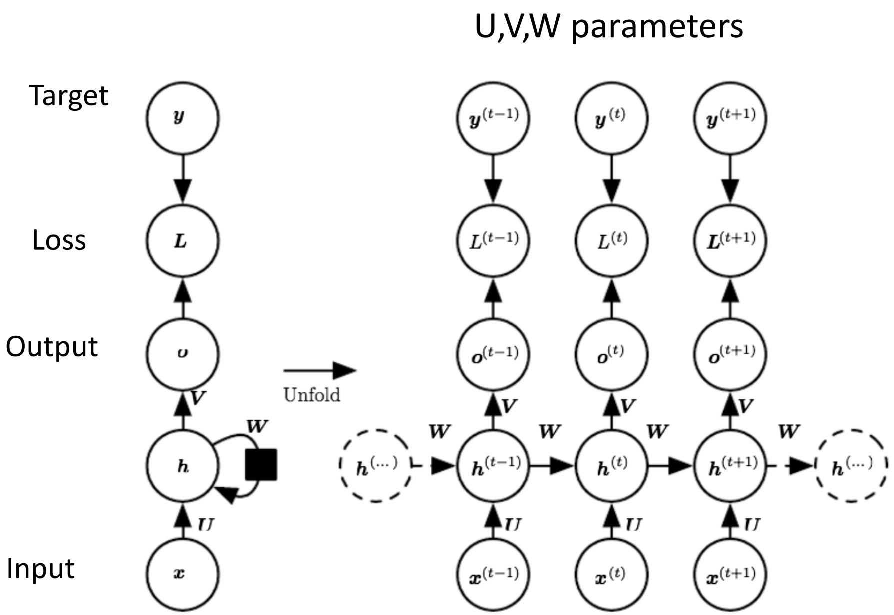


Visualize the action and observation space at https://openai.com/blog/openai-five/.

Handling Partial Observability

- One way to handle partial observability: maintain a belief (conditional distribution of states)
  - Need model { often not easy to learn a model
  - Need to belief tracking, often intractable
- Instead, OpenAI Five use model-free reinforcement learning by using a policy with memory in the form of a LSTM, a type of recurrent neural networks (RNN).
- At time t, a RNN maintains a vector of hidden units h(t) that stores all the information from the before t that is relevant to the future computation.
  - The hidden units can store a fair amount of information, binary vectors of length k can store 2k states.
  - The hidden units are real valued (although discretized to a finite number of a computer).

Recurrent Neural Networks



- Left side of figure shows dependency graph of the computational elements.
  - U, V, and W are weight matrices, L species the loss function for the pair of output o and label y.
  - h is a vector of hidden layer units obtained from applying the weights from W to the value of h at the previous time step and the weights U to the value of input x (typically also composed with activation function).
  - o is the output obtained from applying the weights from V to the hidden units h.

Figure: Image from [1]. Recurrent neural networks. The dependency graph on the left, and unfolded through time on the right.

[1] Goodfellow, I., Y. Bengio, and A. Courville, *Deep Learning*. 2016: MIT Press.

Recurrent Neural Networks

- Execution at each time step depends on values of h at previous time step.
- Computation can be unfolded through time as shown on figure on the right.
  - When unrolled, we get a deep neural network. Size of the deep network depends on length of input.
  - Parameters are shared, i.e. same parameters used in each layer.
  - Hidden layer value (and also the output) at time t depends on input at time t as well as all previous inputs through the hidden layer value at time $t - 1$

$$h^{(t)} = f\,(h^{(t-1)}, x^{(t)}) = g^{(t)}(x^{(t)} \dots, x^{(1)})$$

  - Hidden layer value $h^{(t-1)}$ summarizes all the useful information from the past.

Gated RNN: Long Short Term Memory

- Vanilla RNN tend to have diffculties learning long range dependencies.
- Gated RNNS like LSTM learns much better in practice.

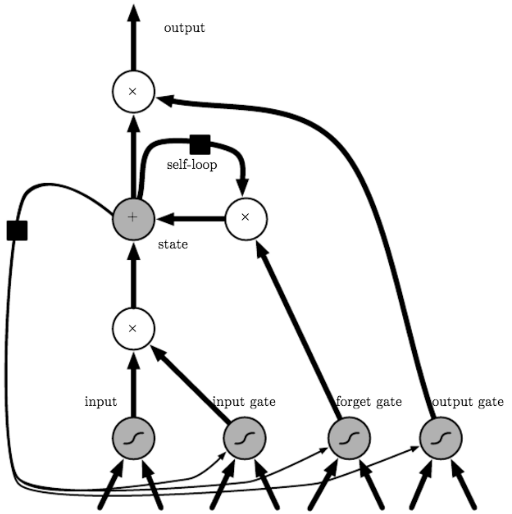


Figure: Image from [1]. LSTM.

[1] Goodfellow, I., Y. Bengio, and A. Courville, *Deep Learning*. 2016: MIT Press.

OpenAI Five Architecture

Players

Inputs

LSTM

Actions

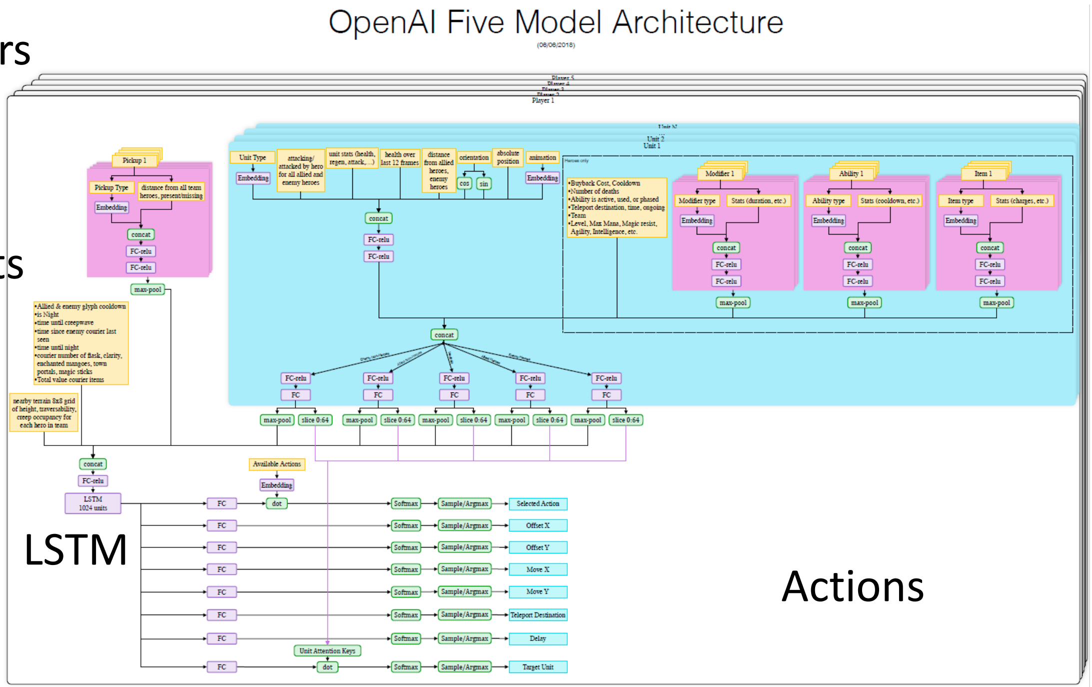
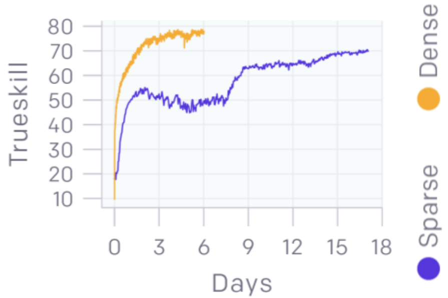


Figure: Image from:
https://www.linkedin.com/pulse/science-behind-openai-five-just-produced-one-greatest-jesus-rodriguez

Challenges – Exploration

- Learn from self-play starting from random weights – a natural curriculum
  - Plays 80% games against itself and 20% against past self
- Initially agent defeated bots, but not humans
  - Randomizing properties (health, speed, start level, etc.,) of units during training helps; started beating humans
- Exploration helped by good reward shaping: reward related to net worth, kills, deaths, assists, etc.,
- Comparing shaped reward with rewarding wins and losses (Image: https://openai.com/blog/openai-five/)



Challenges – Learning to Coordinate

- All agents see the same information
- Collaboration learned by rewarding a combination of individual reward and average team reward
  - Hyperparameter called *team spirit* ranging from 0 to 1 combine individual and average and team reward
  - *Team spirit* slowly changed from 0 to 1 over the training period
- Multi-agent planning and decision making!

Challenges – Compute power

	OPENAI 1V1 BOT	OPENAI FIVE
CPUs	60,000 CPU cores on Azure	128,000 preemptible CPU cores on GCP
GPUs	256 K80 GPUs on Azure	256 P100 GPUs on GCP
Experience collected	~300 years per day	~180 years per day (~900 years per day counting each hero separately)
Size of observation	~3.3 kB	~36.8 kB
Observations per second of gameplay	10	7.5
Batch size	8,388,608 observations	1,048,576 observations
Batches per minute	~20	~60

Source: https://openai.com/blog/openai-five/

OpenAI Five in Action

Watch some of the intelligent plays from the agent at https://www.youtube.com/watch?v=Ub9INopwJ48.

Summary: Main Challenges

- Multiagent planning and decision making
  - an active and emerging research area
  - Multieffector, multibody, multiagent, distributed problem solving
- Most difficult problems
  - Cooperate with own team members and compete against opposing team members, all without centralized control
  - Integrated collaboration, co-ordination, and (friendly) competition
- Applications
  - Economic and war simulations
  - Pandemic response and recovery modeling
  - Corporate strategies
  - Robotic sport and rescue teams
  - Assistive agents for social good
  - Game AI in real-time strategy games, etc.

Multiagent Decision Making Research

- Major conferences:
  - AAAI (www.aaai.org)
  - IJCAI (www.ijcai.org)
  - AAMAS (www.ifaamas.org) - Annual Conference on Autonomous Agents and Multiagent Systems
- Many major journals
  - Journal of Artificial Intelligent Research
  - Artificial Intelligence
  - Autonomous Agents and Multiagent Systems
  - …

Homework

- Readings:
  - RN: 18.1, 18.3
- References:

  (Journal articles publicly available online or through NUS Library e-Resources)

  General references and multiagent systems

  - RN Chapter 18 Historical Notes.
  - (SLB) Shoham, Y. and Leyton-Brown, Kevin. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, 2009
    - eBook download from: http://www.masfoundations.org/
  - Goodfellow, I., Y. Bengio, and A. Courville, *Deep Learning*. 2016: MIT Press.

  Multiagent reinforcement learning

  - Yang, Y., et al., *Mean Field Multi-Agent Reinforcement Learning*, in *Proceedings of the 35th International Conference on Machine Learning*, D. Jennifer and K. Andreas, Editors. 2018, PMLR: Proceedings of Machine Learning Research. p. 5571--5580.
  - Gronauer, S. and K. Diepold, *Multi-agent deep reinforcement learning: a survey.* Artificial Intelligence Review, 2021.