Introduction to Reinforcement Learning

Lecture 4: POMDPs & Multi-Agent RL

Shimon Whiteson
Dept. of Computer Science
University of Oxford

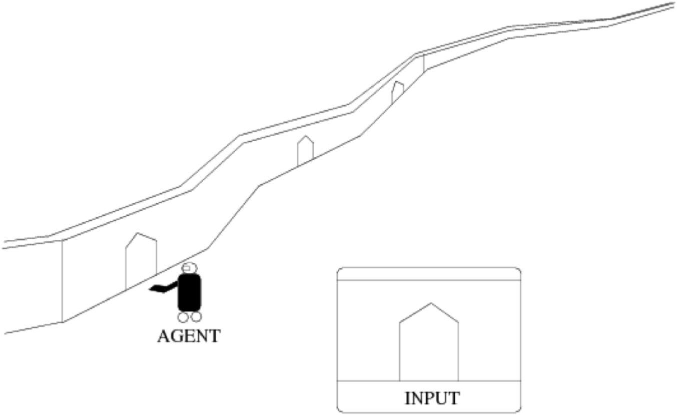
(based on material from Frans Oliehoek, Tony Cassandra, Michael Littman, and Leslie Kaelbling)

joint work with Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, Tabish Rashid, Mikayel Samvelyan, and Christian Schroeder de Witt

Partial observability

- In a *partially observable* decision problem, the agent does not have access to the true state of the environment
- Instead agent receives only *observations* correlated with the state
- There are two possible causes of partial observability:
  1. Noisy sensors: many-to-many function mapping states to observations
  2. Perceptual aliasing: many-to-one mapping

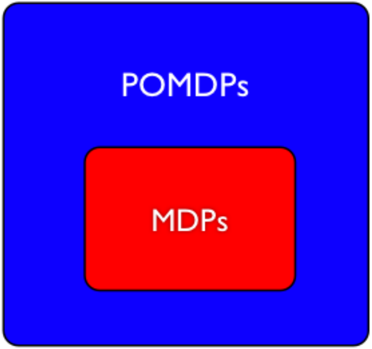
Hallway example

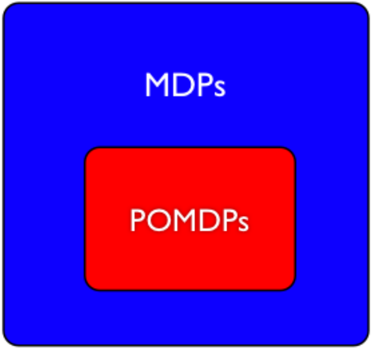


Partially observable Markov decision processes

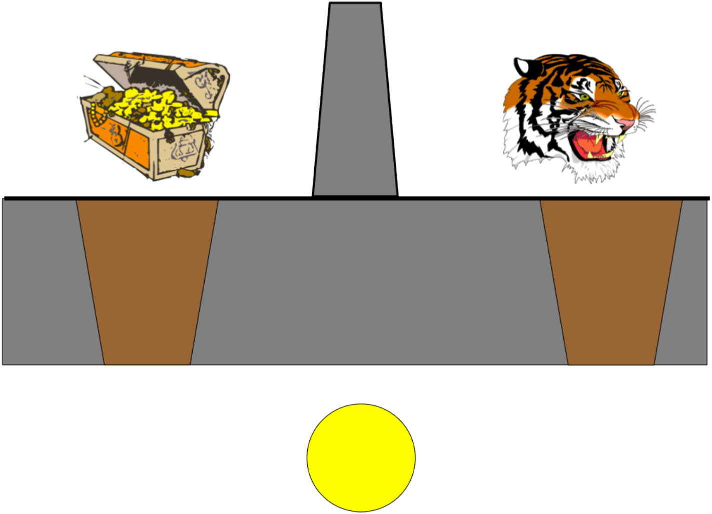
- POMDPs extend MDPs to model partial observability
- Environment is stationary and possibly stochastic environment
- A finite POMDP consists of:
  - Discrete time $t = 0, 1, 2, \dots$
  - A discrete set of states $s \in S$
  - A discrete set of observations $o \in O$
  - A discrete set of actions $a \in A$
  - A *transition model* $p(s'|s,a)$: the probability of transitioning to state s' when the agent takes action a at state s
  - An *observation model* $p(o|s',a)$: the probability of receiving an observation o after taking action a and landing in state s'
  - A *reward* function $R : S \times A \mapsto \mathbb{R}$, so that the agent receives reward $R(s,a)$ when it takes action a at state s
  - A planning horizon, which can be infinite

MDPs $\subset$ POMDPs or POMDPs $\subset$ MDPs?





Tiger example (1)



Tiger example (2)

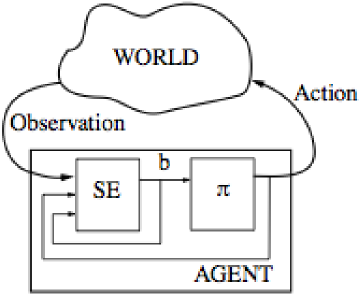
- $S = \{L, R\}$
- $A = \{OL, OR, Li\}$
- $O = \{HL, HR\}$
- Transitions: state is static but opening resets
- Rewards:
    - Correct door: $+10$
    - Wrong door: -100
    - Listen: -1
- Observations: correct 85% of the time:
    - $p(HL|L, Li) = 0.85$
    - $p(HR|L, Li) = 0.15$
    - $p(HL|R, Li) = 0.15$
    - $p(HR|R, Li) = 0.85$

Heuristic approaches

- Without Markov property, reactive policies are suboptimal
- Can sometimes settle for them anyway
- Or condition actions on:
    - Entire history
    - A fixed window of history
    - An engineered subset of history
    - An engineered higher-level observation

Beliefs (1)

- Principled approaches formalise uncertainty about the state
- A *belief* is a probability distribution over states, conditioned on what the agent has observed: $b(s) = p(s)$



Beliefs (2)

- Updating the belief requires knowledge of b, T, and O
- Start from Bayes rule:

$$p(A|B) = \frac{p(A,B)}{p(B)} = \frac{p(B|A)p(A)}{p(B)}$$

- In our case:

$$p(s'|o) = \frac{p(o|s')p(s')}{p(o)}$$

- Adding other givens:

$$p(s'|o,a,b) = \frac{p(o|s',a,b)p(s'|a,b)}{p(o|a,b)}$$

Beliefs (3)

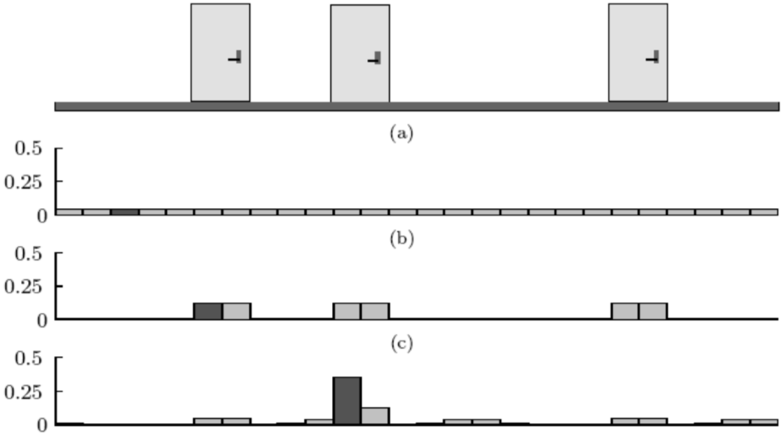
- Expanding $p(s'|a, b)$:

$$b'(s') = \frac{p(o|a, s') \sum_s p(s'|s, a) b(s)}{p(o|a, b)}$$

- Where:

$$p(o|a, b) = \sum_{s'} p(o|a, s') \sum_s p(s'|s, a) b(s)$$

Beliefs (4)



Most likely state

- Solve underlying MDP
- Condition actions on *most likely state*

$$\pi_{MLS} = \arg\max_a Q(s_{ML}, a)$$

where:

$$s_{ML} = \arg\max_s b(s)$$

Q_{MDP}

- Solve underlying MDP, select action with best expected value:

$$\pi_{QMDP} = \arg\max_a Q(b, a)$$

where:

$$Q(b, a) = \sum_s Q(s, a) b(s)$$

- Suppose $b(s_1) = 0.75$, $b(s_2) = 0.25$, and $Q(s, a)$ is:

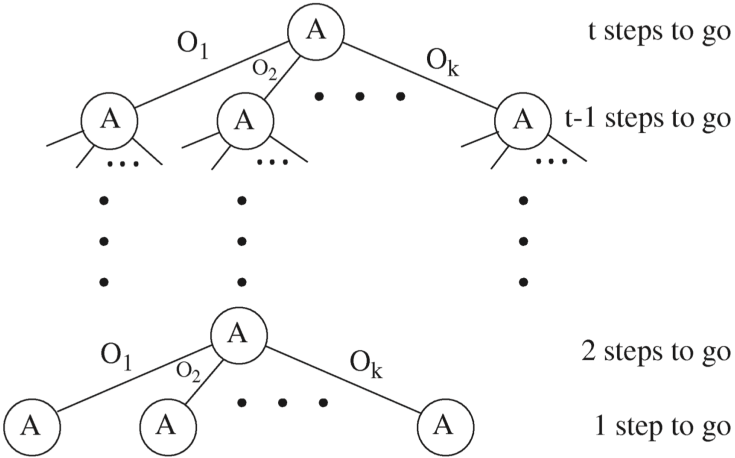
	s_1	s_2
a_1	100	100
a_2	101	0

- Will most likely state or Q_{MDP} yield a higher expected return?

Belief MDPs

- Belief is a *sufficient statistic* for history
- Therefore, we can define a *belief MDP*:
  - States are beliefs in the POMDP: $s_{BMDP} = b(s_{POMDP})$
  - Rewards are expectations wrt b: $R(b, a) = \sum_s b(s)R(s, a)$
  - Belief update happens in environment: $p(b'|b_t, a_t, o_t) = 1$ iff $b' = b_{t+1}$
- Automatically balance reward and information gathering
- Belief MDP has continuous state: belief vector has length $|S_{POMDP}|$
- Fortunately, the value function is *piecewise-linear and convex*
- What prevents *convenient delusions*?

Policy trees



POMDP value functions

- Value function of t-step policy tree π:

$$V^{\pi}(s) = R(s,a) + \gamma \sum_{s'} p(s'|s,a_{\pi}) \sum_{o} p(o|s',\pi_a) V^{\pi_o}(s')$$

  where π_o is the $(t-1)$-step policy subtree of π associated with o

- But we need value functions over beliefs, not states:

$$V^{\pi}(b) = \sum_{s \in S} b(s) V^{\pi}(s)$$

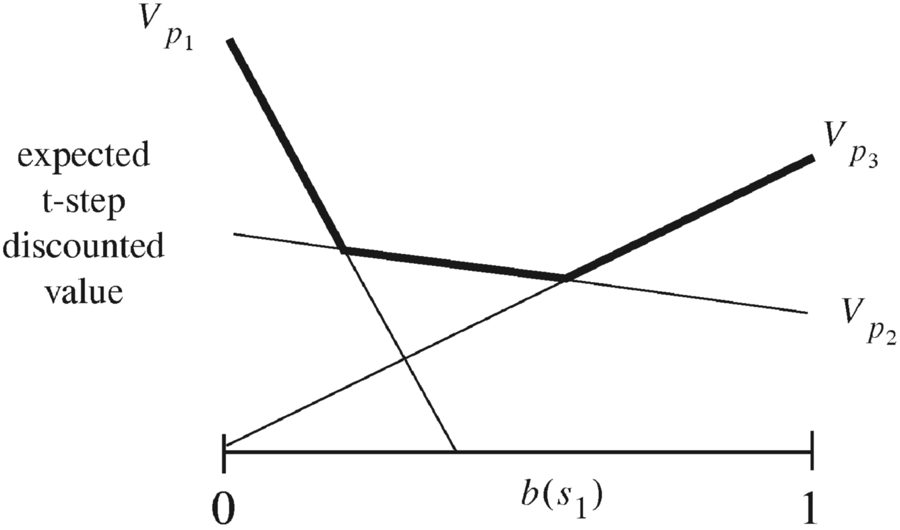
- For compactness, we write the state-value function as an *α-vector* $\alpha_{\pi} = \langle V_{\pi}(s_1), \ldots, V_{\pi}(s_{|S|}) \rangle$ such that:

$$V^{\pi}(b) = b \cdot \alpha_{\pi}$$

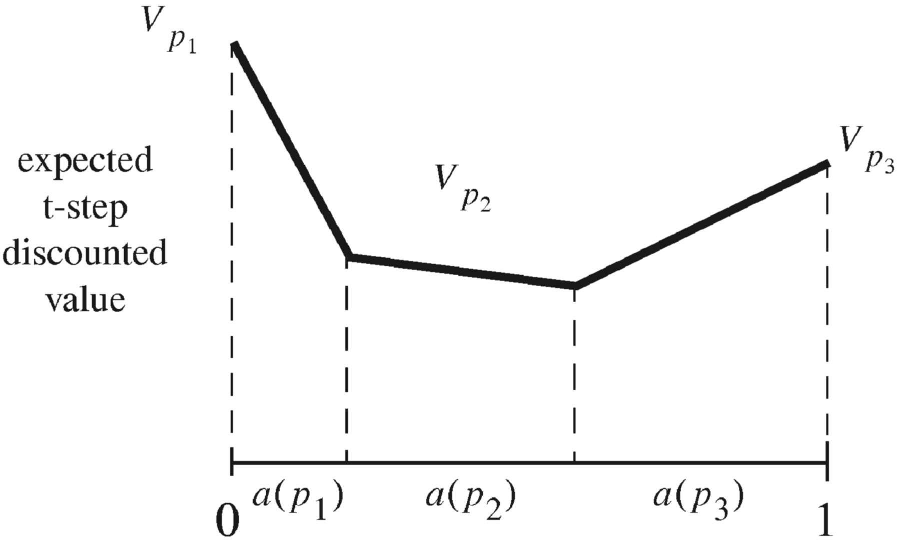
- The optimal value function is thus:

$$V^{*}(b) = \max_{\pi} b \cdot \alpha_{\pi}$$

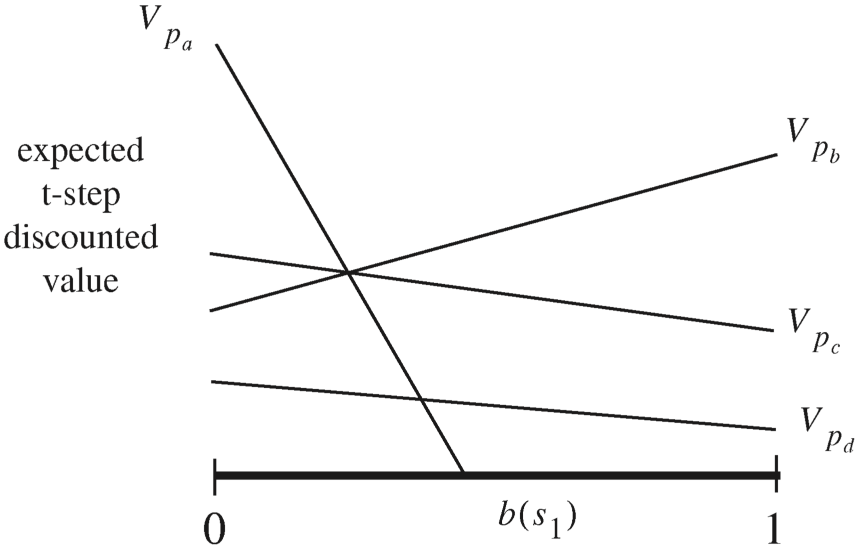
Piecewise-linear convex value functions



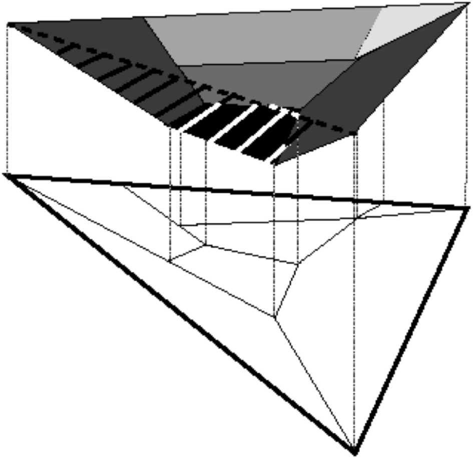
POMDP action selection



Dominated policy trees



POMDP value functions in 3D

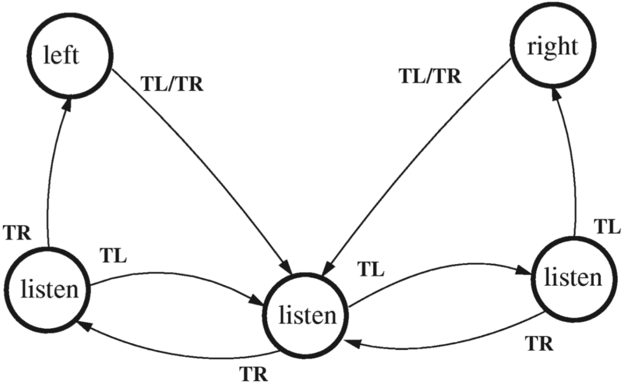


POMDP planning

- Value iteration [Sondik 1971, Monahan 1982]
  - Given set Π_{t-1} of undominated $(t-1)$-step policy trees
  - Construct all $|A||\Pi_{t-1}|^{|O|}$ t-step policy trees by extension
  - Prune dominated policy trees to form Π_t
- Faster: linear support [Chang 1988] & Witness [Cassandra et al. 1997]
- Approximate: point-based value iteration [Pineau et al. 2003]
- More scalable: on-line POMDP planning [Ross et al. 2008]

Infinite horizon planning

- Infinite horizon POMDP planning is undecidable!
- Optimal value function may have infinite facets
- Finite horizon planning may still converge for large t
- Yields finite state machine, e.g., infinite horizon tiger for $b_0 = 0.5, 0.5$:



Bayes-optimal reinforcement learning

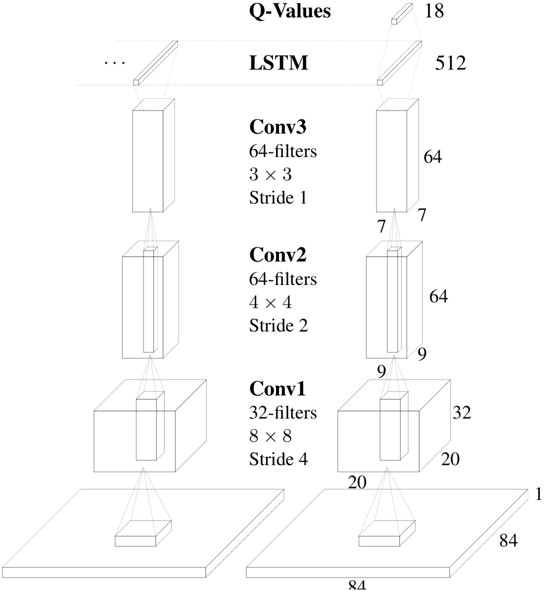
- Problem of learning in an MDP is cast as one of planning in a POMDP where the hidden state corresponds to the unknown model parameters: $s_{POMDP} = (s_{MDP}, T, R)$

- Like any other POMDP, this POMDP can be treated like a belief MDP: $s_{BMDP} = b(s_{POMDP})$

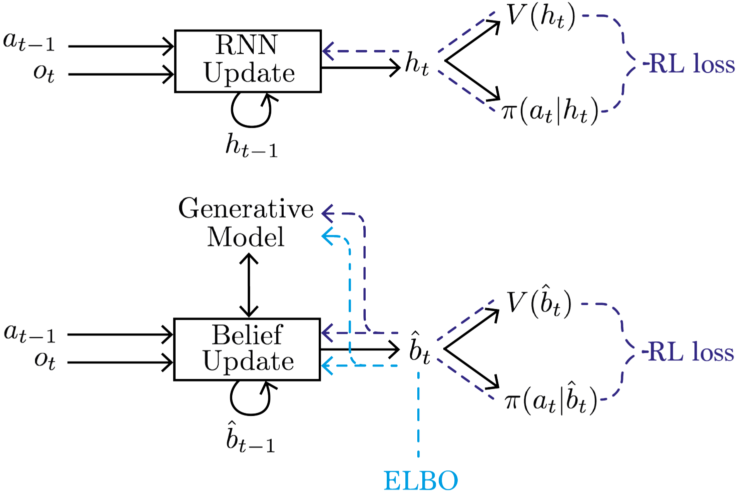
- However, since s_{MDP} is directly observed, only a belief over T and R is necessary, thus:

$$s_{BMDP} = b(s_{POMDP}) = b(s_{MDP}, T, R) = (s_{MDP}, b(T, R))$$

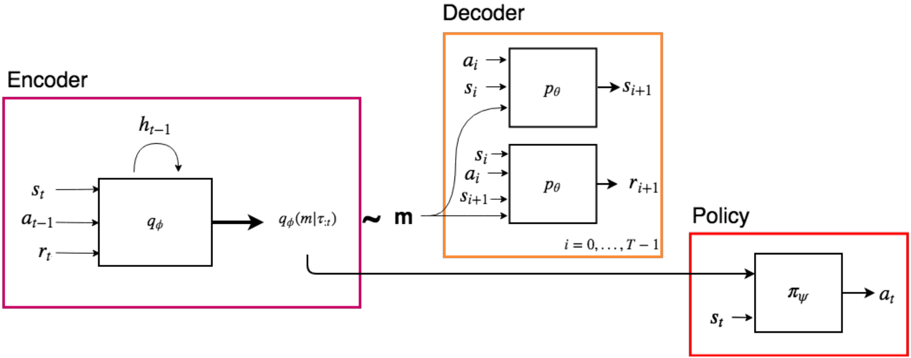
DRQN [Hausknecht & Stone 2015]



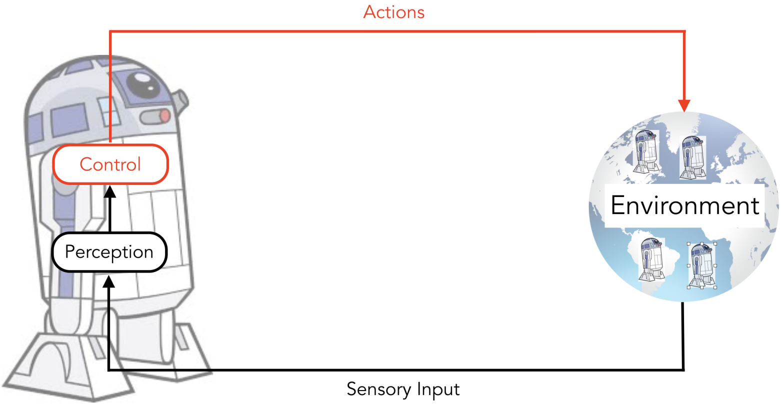
Deep Variational RL [Igl et al. 2018]



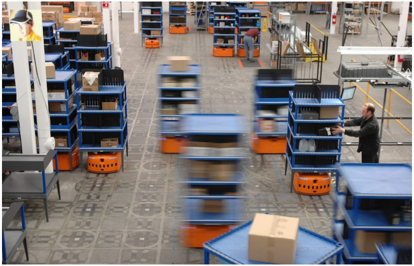
VariBAD [Zintgraf et al. 2019]



Multi-Agent Paradigm



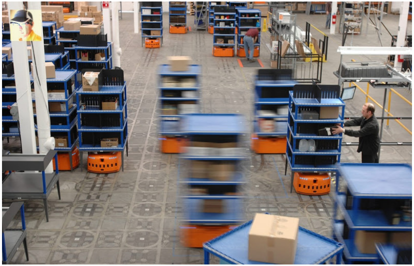
Multi-Agent Systems are Everywhere



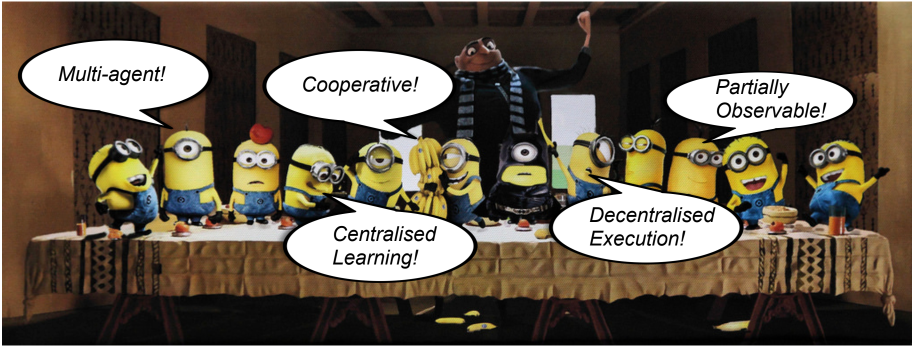
Types of Multi-Agent Systems

- *Cooperative:*
  - Shared team reward
  - Coordination problem
- *Competitive:*
  - Zero-sum games
  - Individual opposing rewards
  - Minimax equilibria
- *Mixed:*
  - General-sum games
  - Nash equilibria
  - What is the question? [Shoham et al. 2007]

Coordination Problems are Everywhere



Setting



(Figure by Jakob Foerster)

Multi-Agent MDP

- All agents see the global state s
- Individual actions: $u^a \in U$
- State transitions: $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \rightarrow [0, 1]$
- Shared team reward: $r(s, \mathbf{u}) : S \times \mathbf{U} \rightarrow \mathbb{R}$
- Equivalent to an MDP with a factored action space

Dec-POMDP

- Observation function: $O(s, a) : S \times A \to Z$
- Action-observation history: $\tau^a \in T \equiv (Z \times U)^*$
- Decentralised policies: $\pi^a(u^a|\tau^a) : T \times U \to [0, 1]$
- Natural decentralisation: communication and sensory constraints
- Artificial decentralisation: improve tractability
- Centralised learning of decentralised policies

The Predictability / Exploitation Dilemma

- Exploitation:
  - Maximising performance requires collecting reward
  - In a single-agent setting, this requires *exploiting* observations
- Predictability:
  - Dec-POMDP agents cannot explicitly communicate
  - Coordination requires *predictability*: "stick to the plan!"
  - Predictability can require ignoring private information

When does the benefit of exploiting private observations outweigh the cost in predictability?

Independent Learning

- Independent Q-learning [Tan 1993]
  - Each agent learns independently with its own Q-function
  - Treats other agents as part of the environment

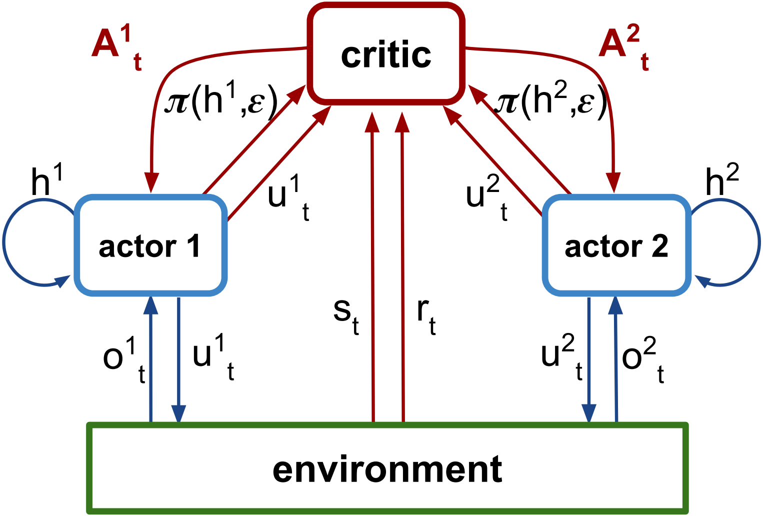
- Independent actor-critic [Foerster et al. 2018]
  - Each agent learns independently with its own actor-critic
  - Treats other agents as part of the environment

- Speed learning with *parameter sharing*
  - Different inputs, including a, induce different behaviour
  - Still independent: value functions condition only on τ^a and u^a

- Limitations:
  - Nonstationary learning
  - Hard to learn to coordinate

Centralised Critics [Lowe et al. 2017; Foerster et al. 2018]

Centralised $V(s, \boldsymbol{\tau})$ or $Q(s, \boldsymbol{\tau}, \mathbf{u})$ → hard greedification → actor-critic

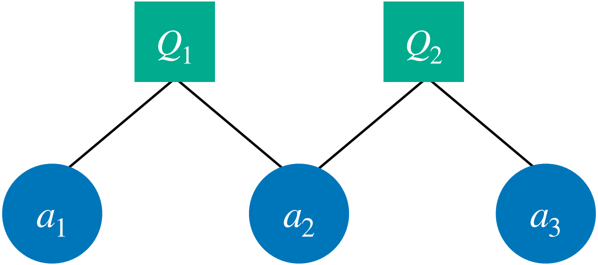


Factored Joint Value Functions

- *Factored value functions* [Guestrin et al. 2003] can improve scalability:

$$Q_{tot}(\boldsymbol{\tau}, \mathbf{u}; \boldsymbol{\theta}) = \sum_{e=1}^{E} Q_e(\tau^e, \mathbf{u}^e; \theta^e)$$

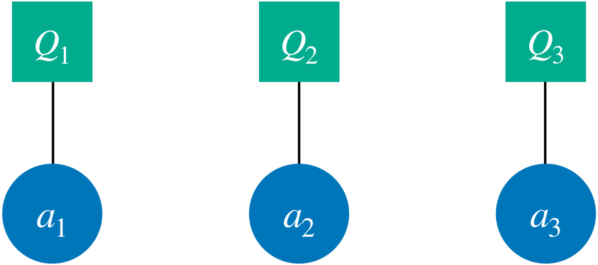
where each e indicates a subset of the agents



Value Decomposition Networks [Sunehag et al., 2017]

- Most extreme factorisation: one per agent:

$$Q_{tot}(\boldsymbol{\tau}, \mathbf{u}; \boldsymbol{\theta}) = \sum_{a=1}^{N} Q_a(\tau^a, u^a; \theta^a)$$



Decentralisability

- Added benefit of decentralising the max and arg max:

$$\max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}; \boldsymbol{\theta}) = \sum \max_{u^a} Q_a(\tau^a, u^a; \theta^a)$$

$$\arg\max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}; \boldsymbol{\theta}) = \begin{pmatrix} \arg\max_{u^1} Q_1(\tau^1, u^1; \theta^1) \\ \vdots \\ \arg\max_{u^n} Q_n(\tau^n, u^n; \theta^n) \end{pmatrix}$$

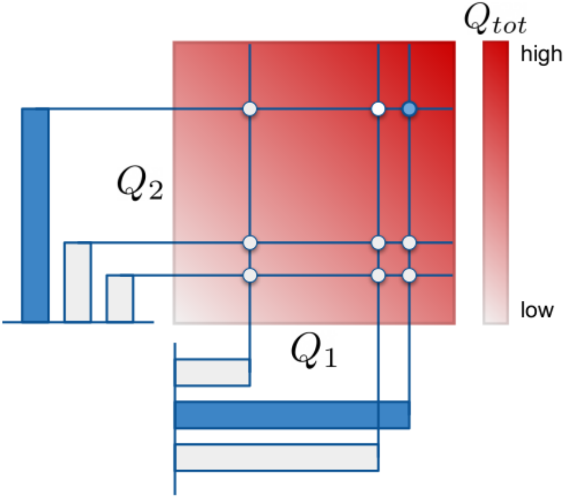
- No more hard greedification $\implies$ Q-learning, not actor-critic:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{b} \left[\left(y_i^{\text{tot}} - Q_{tot}(\boldsymbol{\tau}, \mathbf{u}; \boldsymbol{\theta}) \right)^2 \right],$$

$$y_i^{\text{tot}} = r_i + \gamma \max_{\mathbf{u}'} Q_{tot}(\boldsymbol{\tau}'_i, \mathbf{u}'; \boldsymbol{\theta}^-)$$

QMIX's Monotonicity Constraint

To decentralise max / arg max, it suffices to enforce: $\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \ \forall a \in A$



Representational Capacity

It'll never work: monotonic mixing still can't capture the benefit of coordination

		Agent 2	
		A	*B*
Agent 1	*A*	0	1
	B	1	2

linear & monotonic

VDN & QMIX

		Agent 2	
		A	*B*
Agent 1	*A*	0	1
	B	1	8

nonlinear & monotonic

Just QMIX

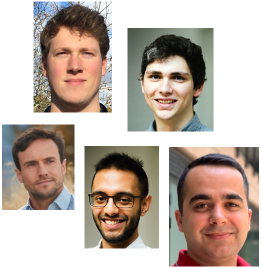
		Agent 2	
		A	*B*
Agent 1	*A*	2	1
	B	1	8

nonlinear & nonmonotonic

Neither

Bootstrapping

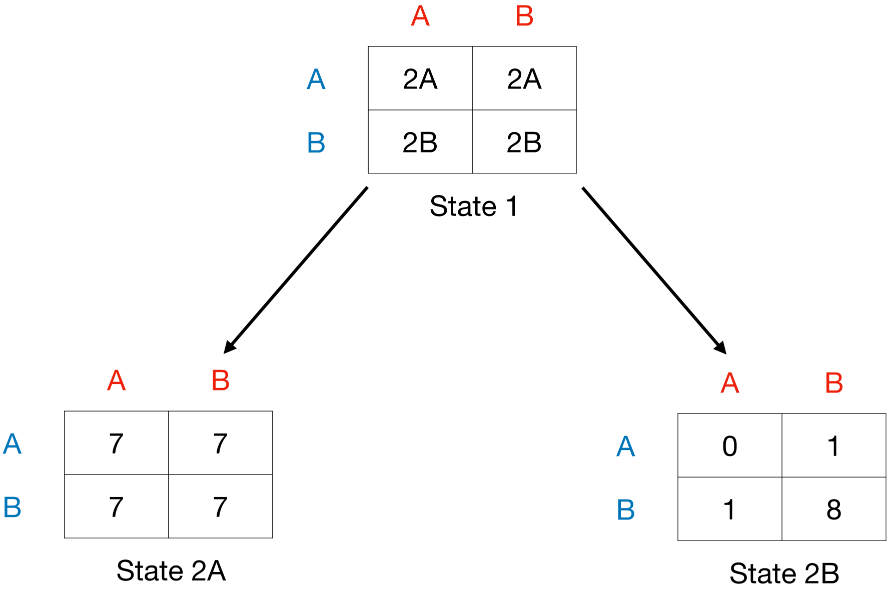
It matters because of *bootstrapping*



$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{b} \left[\left(y_i^{\text{tot}} - Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s; \boldsymbol{\theta}) \right)^2 \right],$$

$$y_i^{\text{tot}} = r_i + \gamma \max_{\mathbf{u}'} Q_{tot}(\boldsymbol{\tau}'_i, \mathbf{u}', s'; \boldsymbol{\theta}^-)$$

Two-Step Game



	A	B
A	2A	2A
B	2B	2B

State 1

	A	B
A	7	7
B	7	7

State 2A

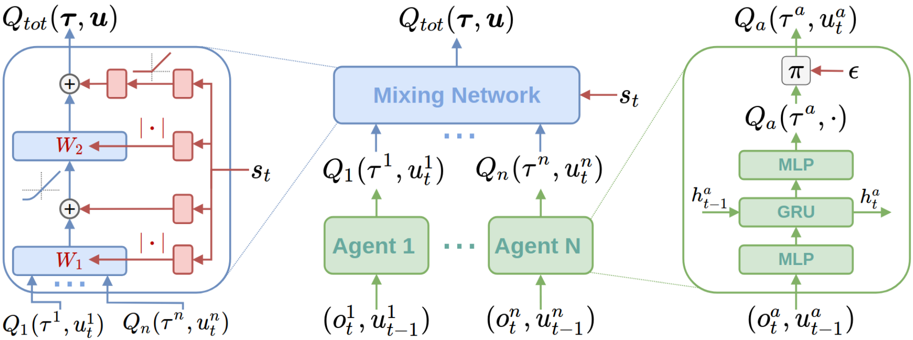
	A	B
A	0	1
B	1	8

State 2B

Two-Step Game Results

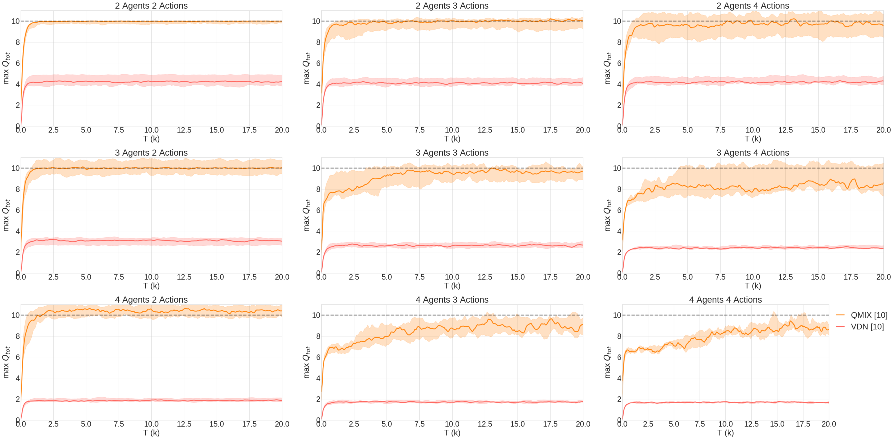
		A	B		A	B		A	B
Ground Truth	A	7	7	A	7	7	A	0	1
	B	8	8	B	7	7	B	1	8
		A	B		A	B		A	B
VDN	A	6.94	6.94	A	6.99	7.02	A	-1.87	2.31
	B	6.35	6.36	B	6.99	7.02	B	2.33	6.51
		A	B		A	B		A	B
QMIX	A	6.93	6.93	A	7.00	7.00	A	0.00	1.00
	B	7.92	7.92	B	7.00	7.00	B	1.00	8.00
		State 1			State 2A			State 2B	

QMIX [Rashid et al. 2018]



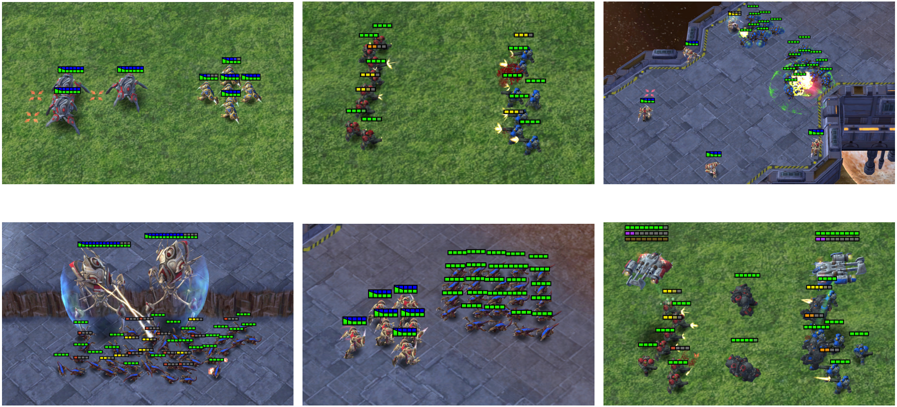
- Agent network: represents $Q_i(\tau^a, u^a; \theta^a)$
- Mixing network: represents $Q_{tot}(\boldsymbol{\tau})$ using nonnegative weights
- Hypernetwork: generates weights of hypernetwork based on global s

Random Matrix Games (The Students Were Right)



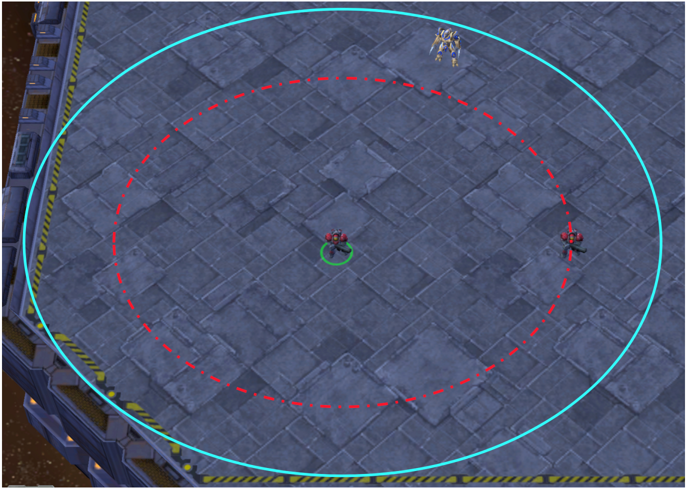
StarCraft Multi-Agent Challenge (SMAC)

[Samvelyan et al. 2019]



https://github.com/oxwhirl/smac
https://github.com/oxwhirl/pymarl

Partial Observability in SMAC

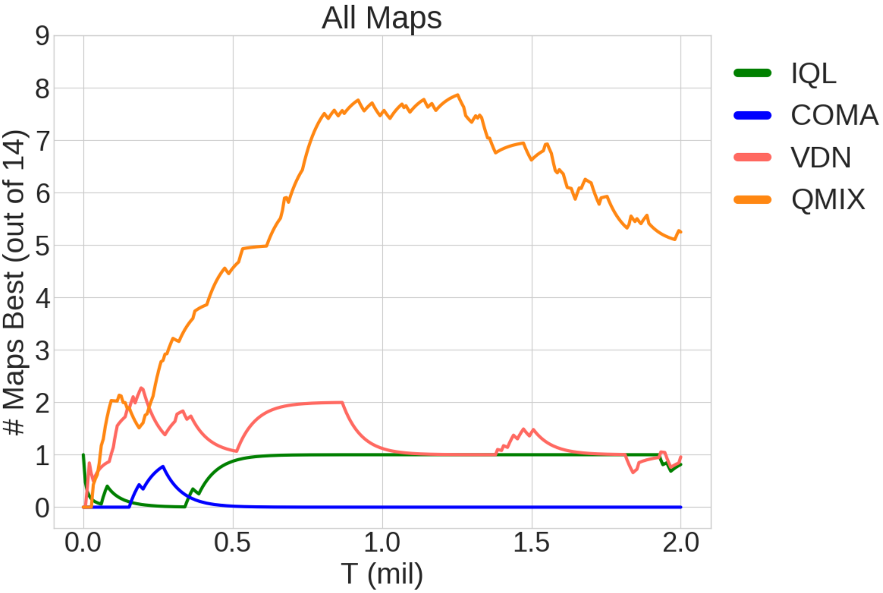


Cyan = sight range   Red = shooting range

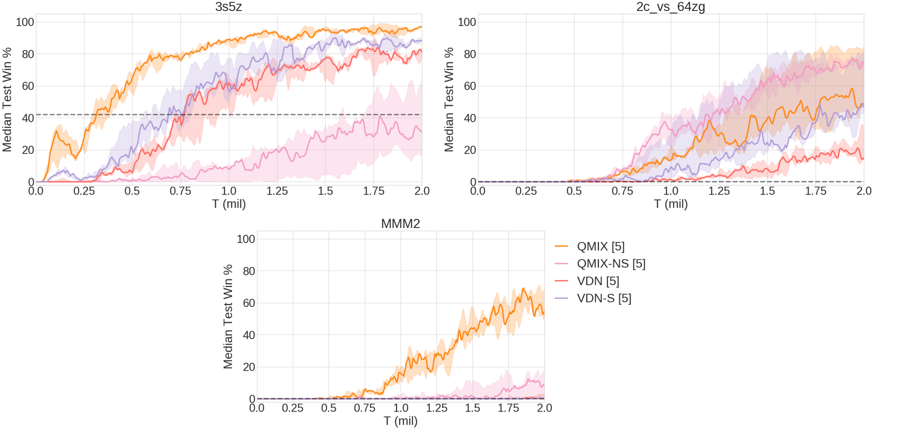
SMAC Maps

Name	Ally Units	Enemy Units
2s3z	2 Stalkers & 3 Zealots	2 Stalkers & 3 Zealots
3s5z	3 Stalkers & 5 Zealots	3 Stalkers & 5 Zealots
1c3s5z	1 Colossus, 3 Stalkers & 5 Zealots	1 Colossus, 3 Stalkers & 5 Zealots
5m_vs_6m	5 Marines	6 Marines
10m_vs_11m	10 Marines	11 Marines
27m_vs_30m	27 Marines	30 Marines
3s5z_vs_3s6z	3 Stalkers & 5 Zealots	3 Stalkers & 6 Zealots
MMM2	1 Medivac, 2 Marauders & 7 Marines	1 Medivac, 3 Marauders & 8 Marines
2s_vs_1sc	2 Stalkers	1 Spine Crawler
3s_vs_5z	3 Stalkers	5 Zealots
6h_vs_8z	6 Hydralisks	8 Zealots
bane_vs_bane	20 Zerglings & 4 Banelings	20 Zerglings & 4 Banelings
2c_vs_64zg	2 Colossi	64 Zerglings
corridor	6 Zealots	24 Zerglings

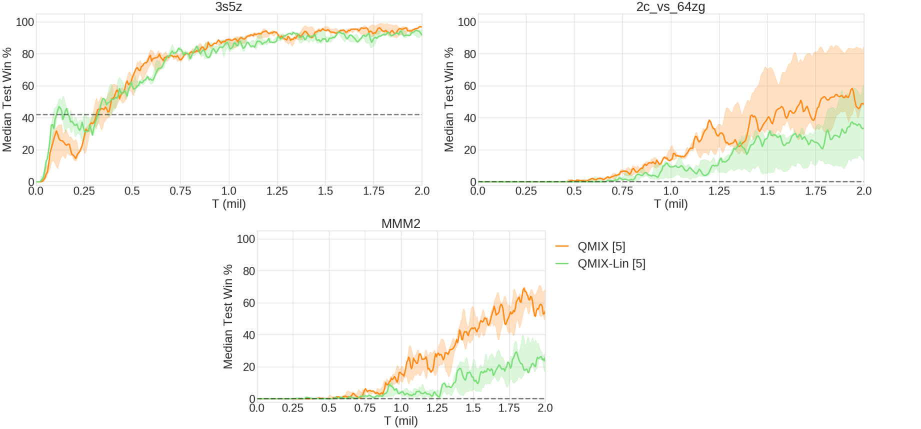
Overall Results (The Students Were Right)



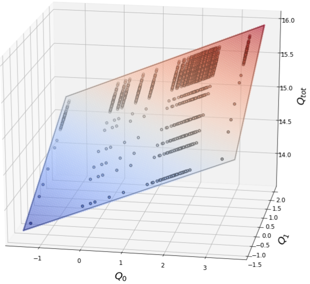
State Ablations



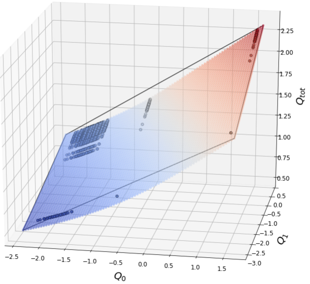
Linear Ablations



Learned Mixing Functions (2c_vs_64zg)

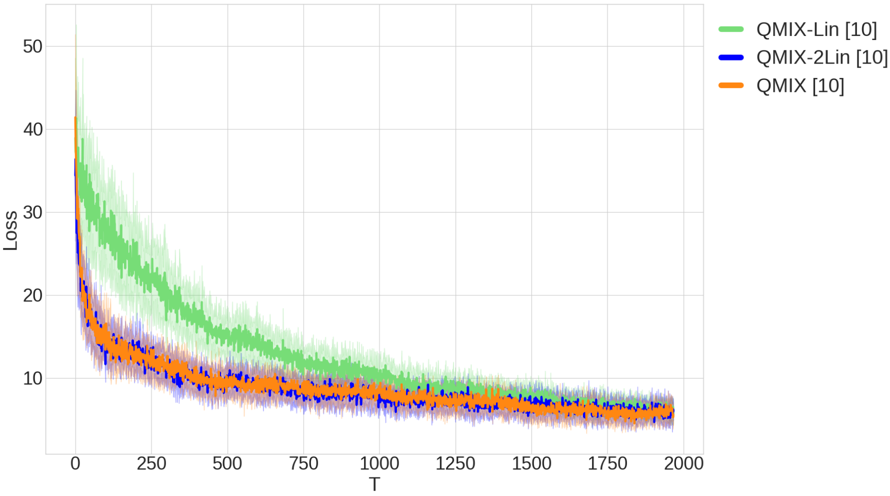


$t = 0$

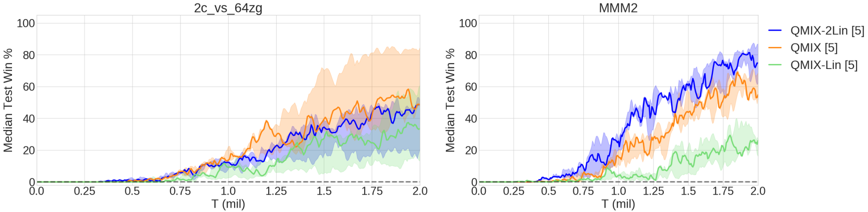


$t = 50$

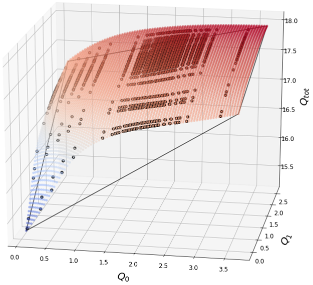
Multi-Layer Linear Mixing (Regression)



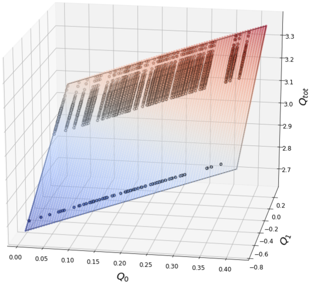
Multi-Layer Linear Mixing (SMAC)



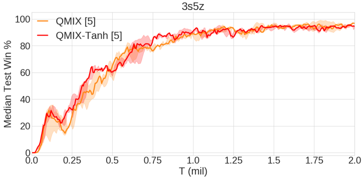
Tanh Activation

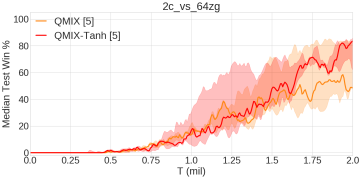


$t = 0$



$t = 50$





QMIX Takeaways

- Value function factorisation is crucial
- Flexible conditioning on central state is crucial
- Richly parameterised mixing is crucial
- Nonlinear mixing is not crucial (on SMAC)

Whiteson Research Lab

Using Reflexive Eye Movements for Fast Challenge-Response Authentication
A
#
Z
!
Q
&
F2
Ctrl
@
Shift
FIRE ACTION