ETH zürich

Introduction to Deep Reinforcement Learning and Multi-Agent Modelling

Presented

by Thomas Asikis, asikist@ethz.ch

Agent-Based Modeling and Social System Simulation

Fall Semester 2019

Usual Setting

- Optimization Objective: $\min(O(a,s)) \vee \max(O(a,s))$.
- Time t: Usually discrete, (non-)uniform time intervals.
- State $s_t \in \mathbb{S} \subset \mathbb{R}^{n \times m \times \cdots}$
- Actions $a_{t,} \in \mathbb{A} \subset \mathbb{R}^{n \times m \times \cdots}$
- Reward: $r_t = h(O(a,s))$, usually noise estimate of $(O(a,s))$
- Environment → State transition: $s_{t+1} = g(s_t, a_t)$
- Code for slides of the 2nd will be uploaded to: https://github.com/asikist-ethz/reinforcement_learning

ϵ-Greedy Policy

ϵ-Greedy Policy Algorithm

- **Inputs:** s, $Q(s, a)$
- **Parameters:** small $\epsilon > 0$

```
#Determine Optimal action
```

$a^* \leftarrow \underset{a}{\operatorname{argmax}} Q(s_t, a)$

For a in $\mathcal{A}_t$: #$\mathcal{A}_t \leftarrow$ environment.all_possible_actions(s_t)

$$\pi(a|s_t) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}_t|}, a = a^* \\ \frac{\epsilon}{|\mathcal{A}_t|}, a \neq a^* \end{cases}$$

Return $a \leftarrow \pi(a|s_t)$

Reward Design

- Rewards on irrelevant tasks: e.g. capture the flag vs disabling an enemy player, over-maximizing a single asset instead of average portfolio.
- Sparse Rewards: Some tasks may be difficult to model as dense rewards, so learning may suffer.
- Reward and goal monotonicity: Rewards needs to increase as we approach the goal.
- Imitation vs Optimization!
- Inverse RL: Look at optimal behaviors and infer the reward.
- Some board examples:

On-policy Reinforcement Learning

Update a policy based on actions taken by that policy. Exploration is included in the policy learning model.

The policy learned is soft greedy but not greedy (always near optimal)

Example:

1. Chose action a based on π (e.g. ϵ-greedy*)
2. State transition
3. Update π based on outcome of a

nice discussion: https://datascience.stackexchange.com/questions/26471/is-my-understanding-of-on-policy-and-off-policy-td-algorithms-correct

*choose an action greedily or randomly based on a probability, e.g. $\epsilon = 0.1$ for random choice, and $1 - \epsilon = 0.9$ for deterministic. When choosing randomly, usually the actions are chosen uniformly. Then the probability of choosing any action becomes: $\frac{\epsilon}{}$

Off-policy Reinforcement Learning

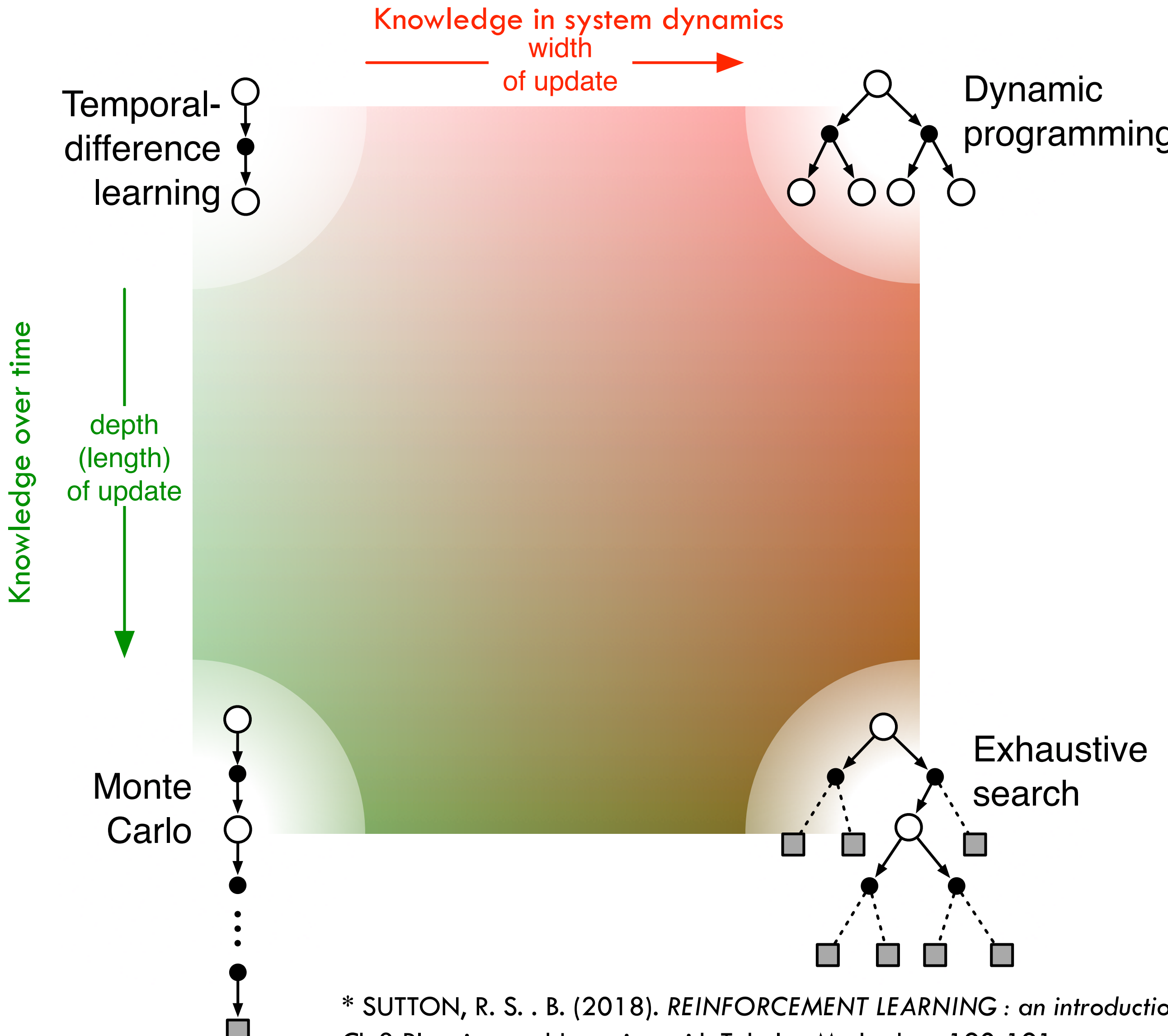
Update an optimal/greedy policy based on action taken by another exploratory policy

Example:

1. Chose action a based on $\pi' \neq \pi$
2. State transition
3. Update π based on outcome of a

* π': target policy that is evaluated or learned (e.g. $\max(Q_{t+1})$)
  π : behavior policy that affects choice of actions (e.g. ϵ-greedy)

Dimensions of the Reinforcement Learning Problem



* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS. Ch.8 Planning and Learning with Tabular Methods, p.190-191

Temporal Difference

- Combination of Dynamic Programming and Monte-Carlo methods
- On-line application: step by step updates.
- Can be extended to n-step updates (e.g. eligibility traces)
- System dynamics can be unknown.
- A simple state value TD Method:

$$\underbrace{V(s_t)}_{\text{New value estimate}} \leftarrow \overbrace{V(s_t)}^{\text{Current value estimate}} + \underbrace{\alpha}_{\text{Learning Rate}}[\overbrace{r_t}^{\text{Reward}} + \underbrace{\gamma V(S_{t+1})}_{\text{Discounted value estimate for next possible state}} - \overbrace{V(S_t)}^{\text{Current value estimate}}]$$

Bootstraping

- Updating estimates based on other estimates:

Current value estimate

Current value estimate

$$V(s_t) \leftarrow V(s_t) + \alpha[r_t + \gamma V(S_{t+1}) - V(S_t)]$$

New value estimate

Discounted value estimate for next possible state

Q-learning

Learning rate

Discount factor

Reward

$$Q(s_t, a_t) \leftarrow (1-\alpha)\ Q(s_t, a_t)\ +\ \alpha \cdot \left(r_t + \gamma \cdot \max_a [Q(s_{t+1}, a)]\right)$$

Value from previous calculation

π':greedy policy over π

Estimate of optimal future value, using ϵ-greedy.

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction.* MIT PRESS. Ch.6 Temporal Difference Learning, p.131

Q-Learning

Off Policy TD Control (Q-Learning) Algorithm

- **Inputs:** ϵ-greedy $\pi(a|s, Q(s,a))$, environment
- **Parameters:** step-size $\boldsymbol{\alpha}$, small $\epsilon > 0$, $0 \leq \gamma \leq 1$
- **Variables:** $\mathrm{Q}(s,a)$, the expected return after taking and action a on state s. Arbitrary initiation. $\mathrm{Q}(s_{terminal}, a) = 0 \; \forall \, a$

  total_episodes

For episode **in** total_episodes:

$t \leftarrow 0$

$s_t \leftarrow$ environment.reset() # reset and get initial state

While $t < T$ **and** $\boldsymbol{s_t \neq s_{terminal}}$:

$a_t \leftarrow \pi(a|s, Q(s,a))$

$s_{t+1}, r_t \leftarrow$ environment.step(a)

$$Q(s_t, a_t) \leftarrow (1-\alpha) \; Q(s_t, a_t) \; + \; \alpha \cdot \left(r_t + \gamma \cdot \max_a [Q(s_{t+1}, a)]\right)$$

$s_t \leftarrow s_{t+1}$

Return $\pi(a|s, Q(s,a)), Q(s,a)$

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction.* MIT PRESS.
Ch.6 Temporal Difference Learning, p.131

Challenges for classic RL

- Maximum operator can be biased. E.g. stochastic policies, maximization bias etc.
- Decision and state may be non-linearly dependent… More complex estimators than "mean" are required.
- States might be continuous, too many. e.g. if state is a permutation of values… Table methods are inefficient or don't work.
- Decision may need to be done in continuous time – Continuous MDP.
- Partially Observable MDP?

Pluging a good value and policy estimator can tackle most of the above "efficiently".

Partially Observable Markov Decision Process

- MDPs are rare in real world scenarios
- State aliasing is often. Same state-actions → completely different rewards.
- Terminal states difficult to identify.
- State transformation or learning can help in this cases. Such Transformation involve a Representation Learning Task.

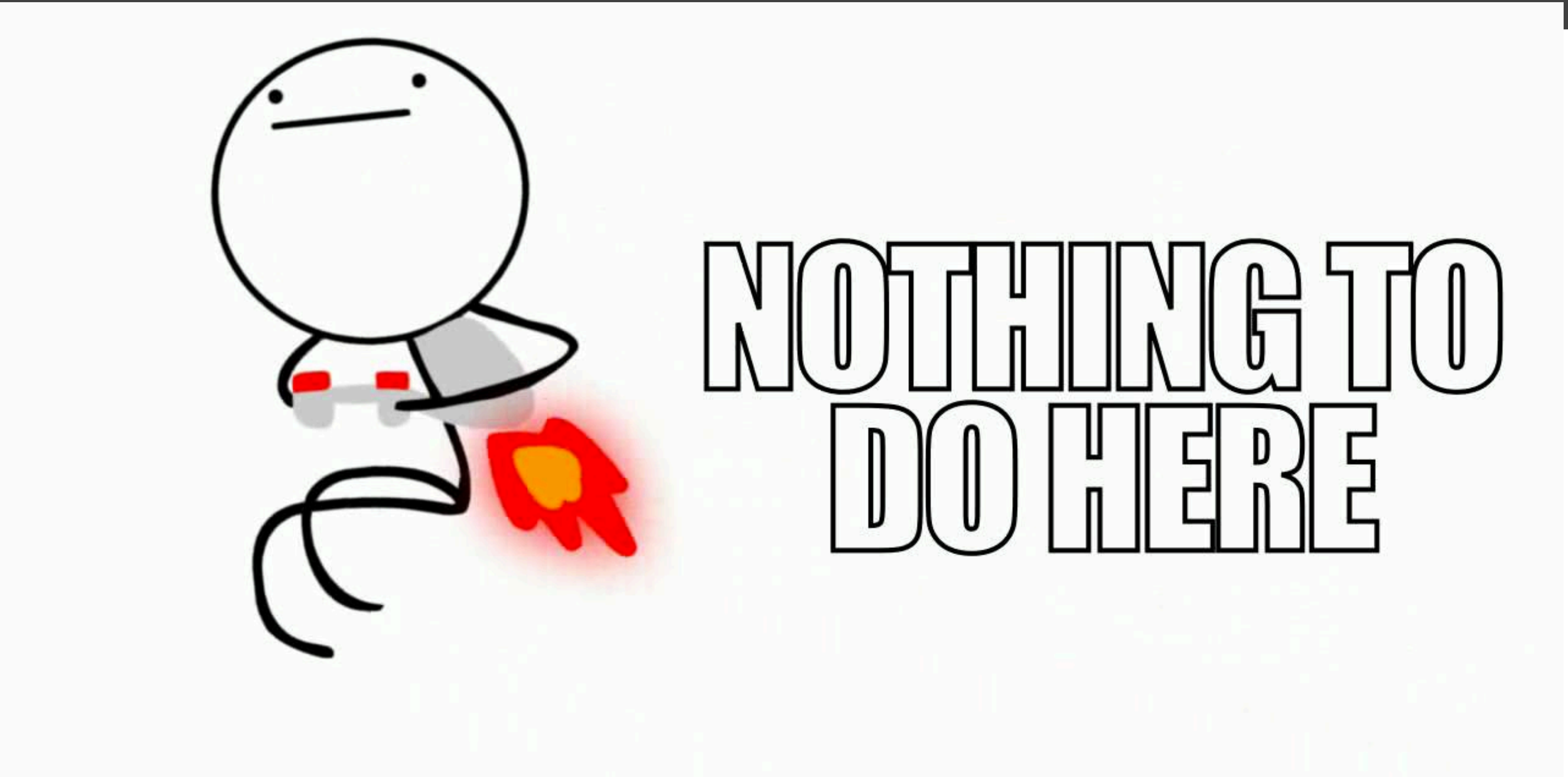
On-policy Distribution

$\mu(s)$:

- The number of timesteps spent in the state s under a policy π.
- Also called on-policy distribution.
- Can also be parametrized or estimated as well

Intuition: Finding the optimal policy and value may strongly depend from this distribution, which is usually unknown.

Worst case: it changes when policy changes…



Too many challenges

Thanks for your attention!

The Prediction Problem: Precise estimation of values

- Parametrize the value function: $V(s, \boldsymbol{w})$
- Given a policy π, update parameters w s.t.:

$$\min_{w}\left(\overline{VE(w)}\right)$$

Where $\overline{VE}$ is a prediction objective (e.g. approximation error metric):

$$\overline{VE(w)} = \sum_{s\in\mathbb{S}} \mu(s) \cdot [V_\pi(s) - V(s, \boldsymbol{w})]^2$$

Iteratively update the parameters with gradient update:

$$w' = w - \eta \nabla_{\mathrm{w}} \overline{VE(w)}$$

Empirical Estimation of $\mu(s)$ can be used here, as the true value.

η: as in machine learning, it is the learning rate. Usually a constant in $(0,1)$.

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS.
Ch.9 On Policy Prediction with Approximation, p.199

Policy Gradient

- A policy can be parametrized $\pi(a|s, \boldsymbol{\theta})$.
- And also approximated by maximizing some policy performance metric.
- Approximate optimal policy by learning parameters

$$\pi(\boldsymbol{\theta}) \rightarrow \pi^*$$

- Update $\theta' = \theta + \eta \nabla_\theta J(\theta)$, where $\boldsymbol{J} = \boldsymbol{v_{\pi,\theta}(s_o)}$ a learning performance metric to maximize.
- Recall from previous lecture:

Each policy has its own true state value function:

$$\boldsymbol{v_{\pi,\theta}(s)} = \mathbb{E}_{\boldsymbol{\pi}}(\boldsymbol{G_t}|\boldsymbol{S_t} = \boldsymbol{s}, \boldsymbol{\theta})$$

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS.
Ch.13 Policy Gradient Methods, p.325

Policy Gradient

- But $v_{\pi,\theta}(s_o)$ depends on action selection π and state distribution $\boldsymbol{\mu(s)}$. Changing the policy parameter θ affects both, and thus performance as well:

  Assumption: $\nabla_\theta J(\theta)$ depends on $\nabla_\theta \mu(s|\theta)$
- The rate of change of policy distribution $\nabla_\theta \mu(s|\theta)$ is very difficult to estimate.

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS.
Ch.13 Policy Gradient Methods, p.325

Policy Gradient Theorem

Policy gradient theorem: The learning performance gradient is proportional to $\mu, Q, \nabla\pi$

$$\nabla_{\theta} J(\theta) \propto \sum_{s \in \mathbb{S}} \mu(s) \sum_{a \in \mathbb{A}} Q_{\pi}(s, a) \nabla_{\theta} \pi(a|s, \boldsymbol{\theta})$$

- And so it doesn't involve a derivative on the state distribution!!!
- So we can calculate the policy gradient.

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS. Ch.13 Policy Gradient Methods, p.325

Actor Critic

- Combine policy gradient with value estimation
- Use one set of parameters $\boldsymbol{w}$ to calculate accurate estimates with V
- Another set of parameters $\boldsymbol{\theta}$ to estimate π given the output of V.
- Update both using gradient update with different learning rates

Actor Critic

Actor Critic Algorithm

- **Inputs:** differentiable $\pi(a|s,\boldsymbol{\theta})$, differentiable $V(s,\boldsymbol{w})$
- **Parameters:** learning rates $\eta_w, \eta_\theta > 0$, $0 \le \gamma \le 1$
- **Variables:** $\boldsymbol{\theta}, \boldsymbol{w}$ e.g. uniform initialization

For episode **in** total_episodes:

$t \leftarrow 0$, $I \leftarrow 1$

$s_t \leftarrow$ environment.reset() # reset and get initial state

While $t < T$ **and** $\boldsymbol{s_t \neq s_{terminal}}$:

$a_t \leftarrow \pi(a|s,\theta)$

$s_{t+1}, r_t \leftarrow$ environment.step(a)

$\delta \leftarrow r_t + \gamma V(s_{t+1}, w) - V(s_t, w)$

$w \leftarrow w + \eta_w \delta \nabla_w V(s_t, w)$

$\theta \leftarrow \theta + \eta_\theta \delta I \nabla_\theta \ln[\pi(a|s_t,\theta)]$

$I \leftarrow \gamma I$, $s_t \leftarrow s_{t+1}$

Return $\pi(a|s, Q(s,a))$, $Q(s,a)$

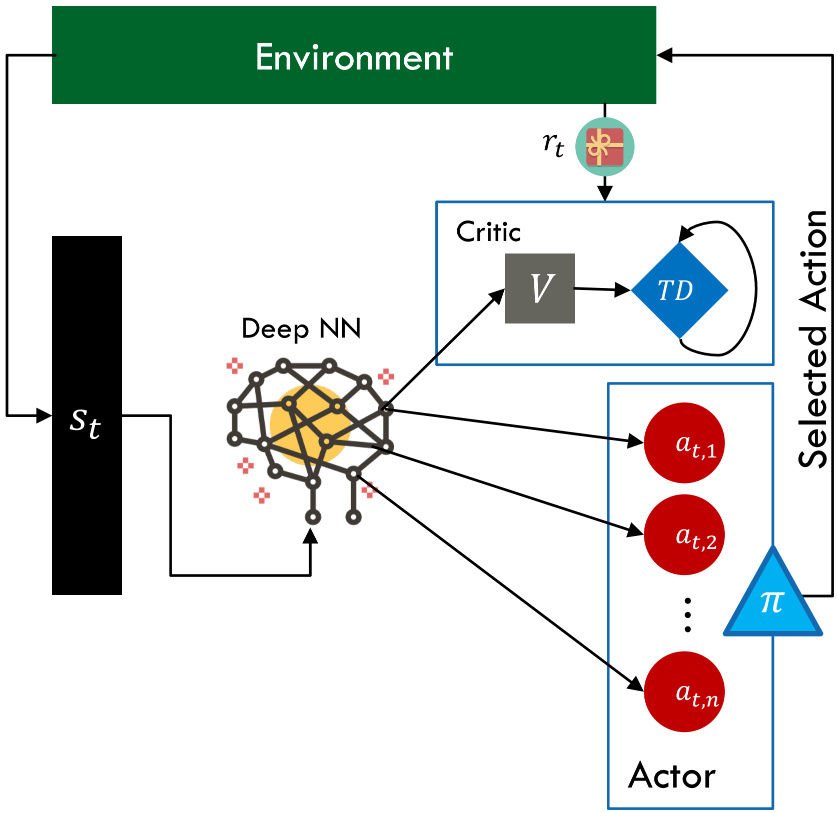
* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS. Ch.13 Policy Gradient Methods, p.332

Deep Learning

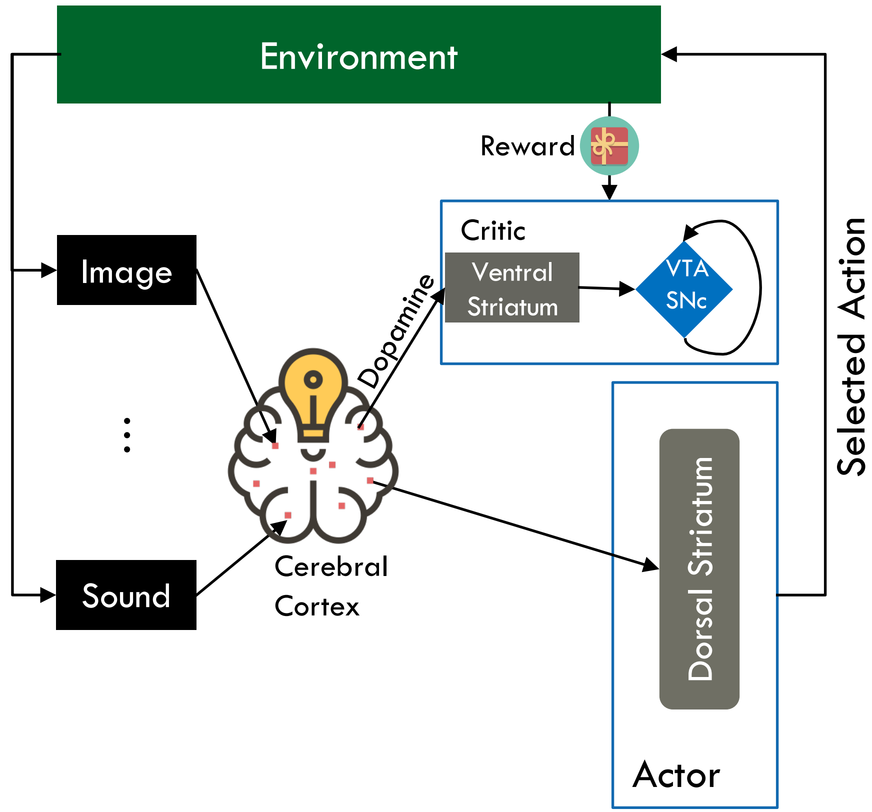
- Nice differentiable functions.
- High performance in learning tasks. Provides very generic and expressive approximators.
- Requires high data volumes ~ easily acquired in combinatorial problems, big data systems and real world applications.
- Efficient function approximation using non-linearity
- A variety of learning loss functions to be used as value $\overline{VE}$ or policy J estimators

Neural Actor-Critic

Deep Learning Implementation



Human Brain



Multi-Agent Reinforcement Learning

Independent

- A multi agent approach, where each agent learns to optimize their objectives independently of the others.
- An agent can interact with another agent by changing their environment and observing their actions via state changes.
- No extra interaction dynamics are modelled in this case.
- Usually it underperforms cooperative agents in cooperative tasks.

Dependent:

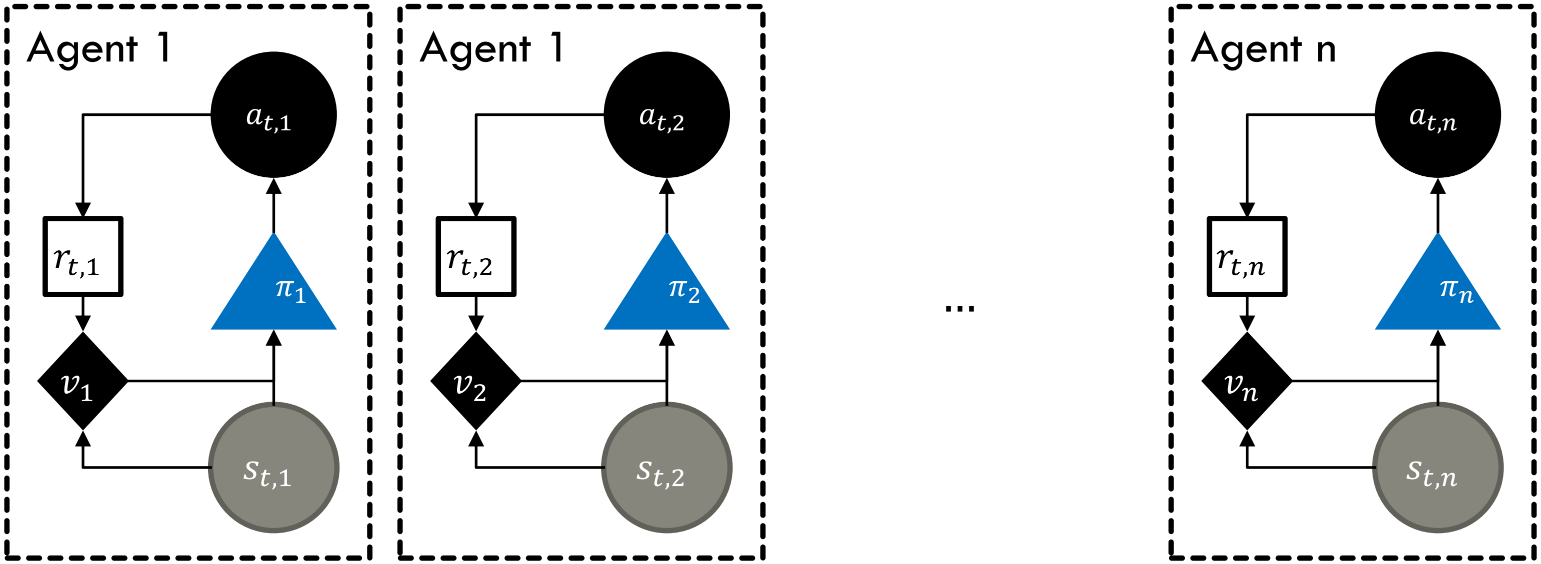
- Shared Observations, e.g. a ”super” agent that manages joint actions/observations.
- Aggregate over rewards
- Competitive rewards
- Latent Features Sharing

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning* (pp. 330-337).

Challenges of Multi-Agent Systems

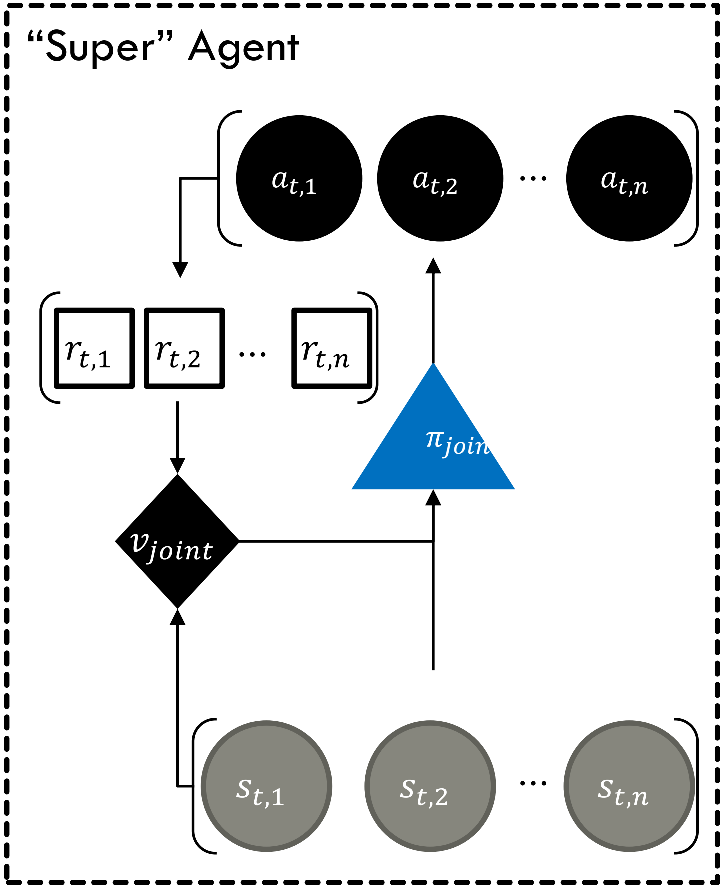
- Challenging Reward Design, e.g. average vs maxmin over all rewards?
- Individual Local observations → POMDP
- Heterogeneous agents → Difficult to model shared observations/rewards
- Scalability
- Small divergence in initial setting of agents usually diverges to much different states! (chaotic)

Independent RL



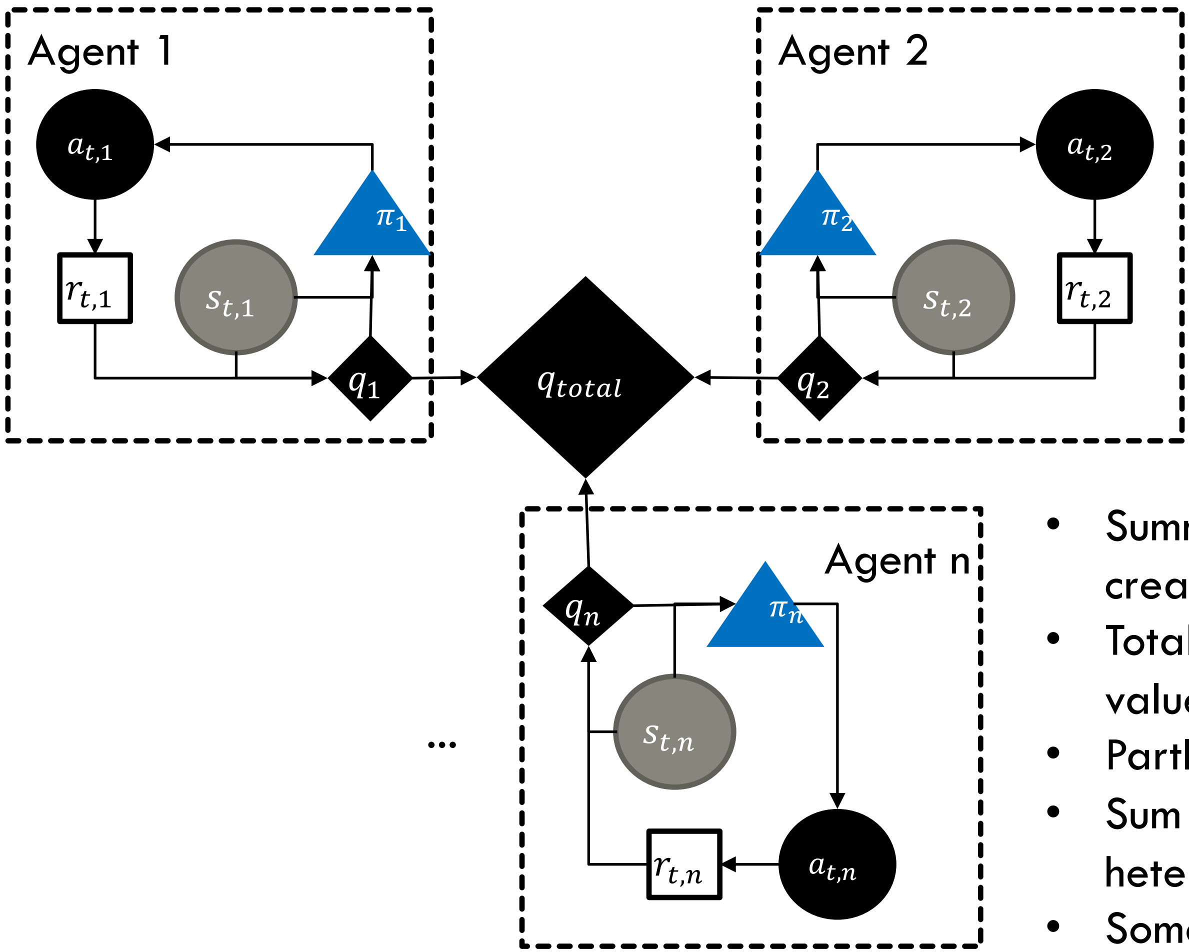
- Easiest Approach
- Expect that an agent infers behaviors of others by observation
- Scalable
- Works for heterogeneous agents
- Model Free (no communication modelling)
- In reality: doesn't learn agent interactions

"Super" Agent



- Agent interactions learning within network
- Not scalable
- Centralized
- May be difficult to work for heterogeneous agents
- In reality: Rarely scales over 100 agents

Value Decomposition Networks



- Summing individual values to create an aggregate one.
- Total q-value is a utility not a value
- Partly centralized
- Sum may not work for heterogeneous rewards
- Some Recent Architectures extend it: e.g. Q-Mix

Summary

- Reinforcement Learning can be used for uncertain and dynamic environments
- Exploitation vs Exploration
- Control Problem: optimal policy,
- Prediction Problem: optimal estimation
- Deep learning can be used to tackle the prediction problem and state transformation for POMDP.
- Actor-Critic modelling tackles both prediction and control problems
- A simple approach to MARL can be done via Independent Reinforcement Learning.

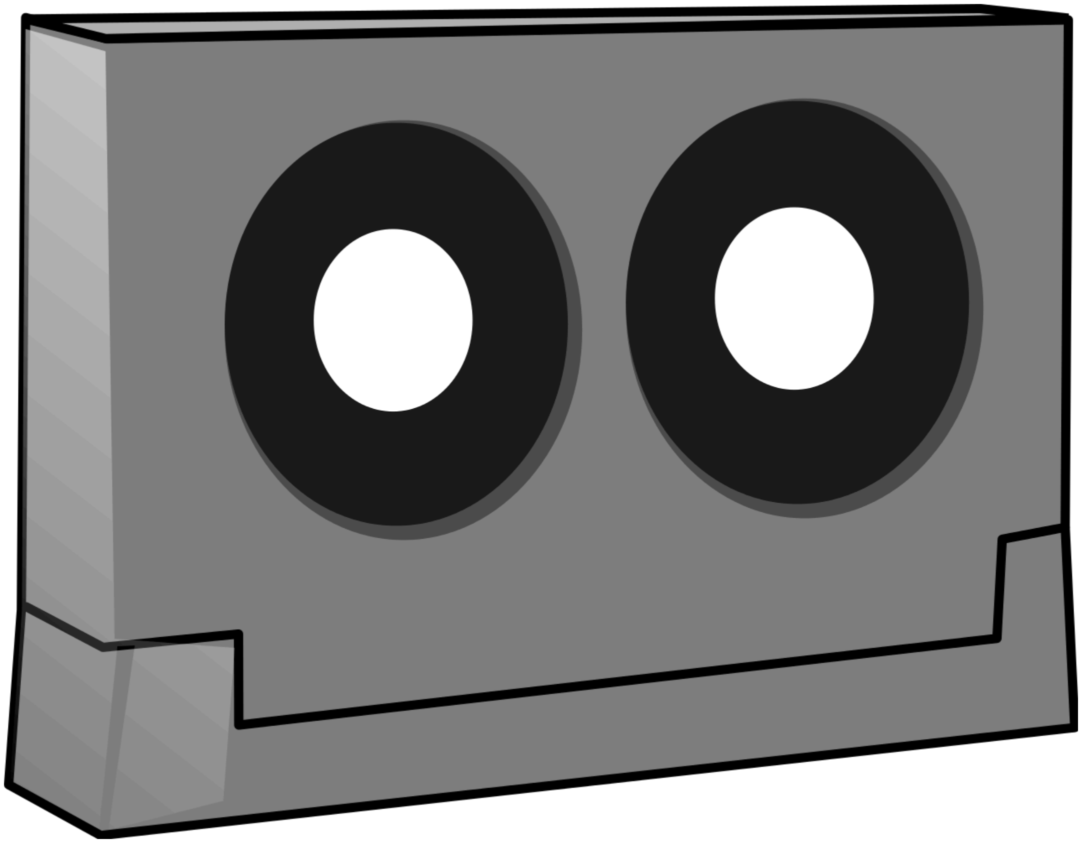
Questions

?

Some References

- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning : an introduction*. MIT Press. Retrieved from https://drive.google.com/file/d/1opPSz5AZ_kVa1uWOdOiveNiBFiEOHjkG/view

  Interesting Chapters:
  - 1 Introduction: 1.1 Reinforcement Learning, 1.2 Examples, 1.3 Elements of Reinforcement Learning
  - 3 Finite Markov Processes: 3.1 The Agent-Environment Interface, 3.2 Goals and Rewards, 3.3 Returns and Episodes
  - 4 Dynamic Programming: All
  - 5 Monte Carlo Methods: All
  - 6 Temporal-Difference Learning: 6.1 TD Prediction, 6.2 Advantages of TD Prediction Methods, 6.4 Sarsa: On-policy TD Control, 6.5 Q-learning Off-policy TD Control
  - 8 Planning and Learning with Tabular Methods: 8.1 Models and Planning, 8.2 Dyna: Integrated Planning, Acting, and Learning, 8.13 Summary of Part I: Dimensions
  - 9 On-policy Prediction with Approximation: 9.1 Value-function Approximation, 9.2 The Prediction Objective $(\overline{VE})$, 9.3 Stochastic-gradient and Semi-gradient Methods
  - 13 Policy Gradient Methods: 13.1 Policy Approximation and its Advantages, 13.2 The Policy Gradient Theorem, 13.5 Actor-Critic Methods
  - 14 Psychology: 14.1 Prediction and Control
  - 15 Neuroscience: 15.4 Dopamine, 15.7 Neural Actor-Critic, 15.9 Hedonistic Neurons, 15.10 Collective Reinforcement Learning
  - 17 Frontiers: 17.3 Observations and State
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., … Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. https://doi.org/10.1038/nature24270
- David Silver's slides: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html



Extra Slides

Just in case

SARSA: On Policy Temporal Difference

Learning rate

Discount factor

Reward

$$Q(s_t, a_t) \leftarrow (1 - \alpha)\ Q(s_t, a_t)\ +\ \alpha \cdot (r_t + \gamma \cdot Q(s_{t+1}, a))$$

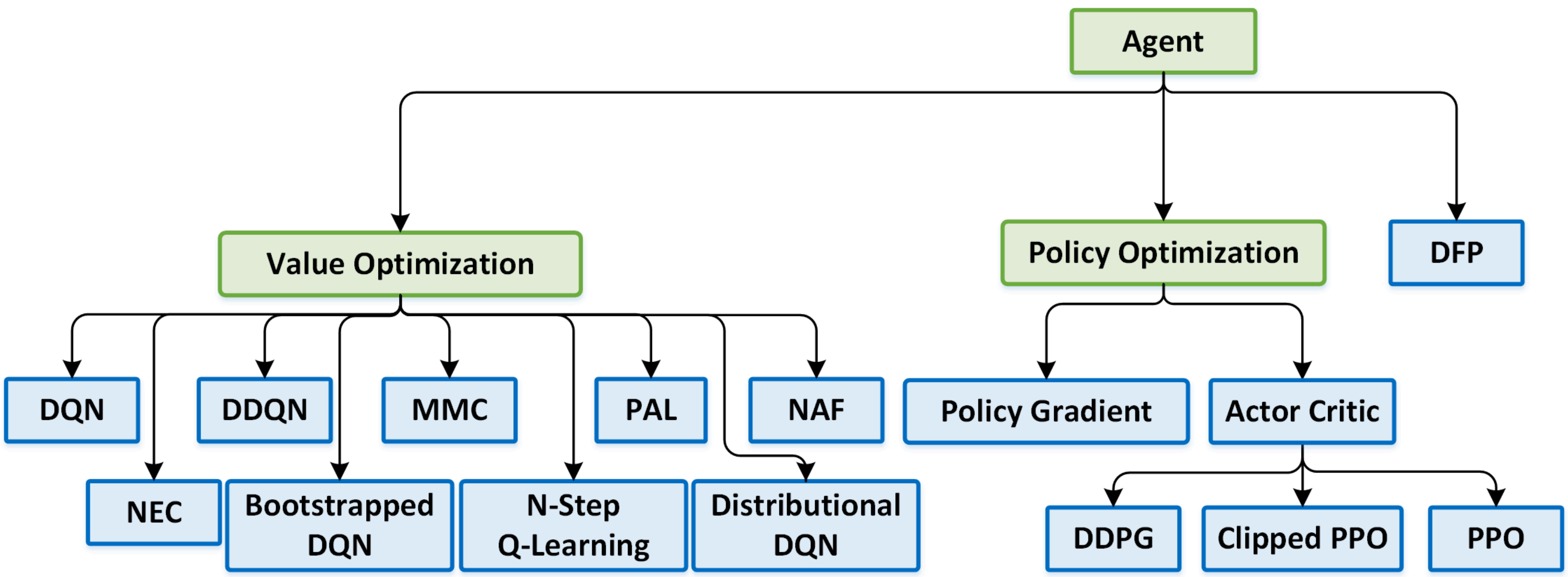
Value from previous calculation

Estimate of optimal future value

Training via Experience Buffer

- Allow a policy to act for some episodes or steps and then save each quadruplet (s_t, a_t, r_t, s_{t+1}) in a buffer.
- Once enough buffers/samples are generated, make a training pass, by using s_t as input.
  - Select the target policy outputs $\pi(a|\theta)$ corresponding to a_t.
  - Calculate V_π and Q_π based on r_t, s_{t+1} for all the steps in the buffer following s_t.
  - Calculate the gradients of value and policy estimators: $\nabla_\theta J(\theta)$, $\nabla_{\mathrm{w}} \overline{VE(w)}$
  - Update parameters θ, w

An RL Taxonomy



https://ai.intel.com/reinforcement-learning-coach-intel/

Learning more

Some useful terms to check for learning more:

- Dueling networks
- Prioritized experience replay
- Advantage
- A2C
- Q-MIX
- Differentiable Inter-Agent Learning and Reinforced Inter-Agent Learning.

Notation Table I

Symbol	Explanation
i, j	Agent indeces
$O(x)$	An objective function that operates on input x
t	A timestep
$a_{t,i}$	An action taken by agent i at time t
$s_{t,i}$	The agent state of agent i at time t
$r_{t,i}$	The reward received by an agent i at time t
$g(s_{t,i}a_{t,i})$	The state transition that happens from time t to $t+1$ given agent i state and selected action
$V(s_{t,i})$	The value function, that provides the agent i estimates about how optimal is its state at time t
$Q(s_{t,i}, a_{t,i})$	The action-value function, that provides the agent estimates about how optimal is its state $s_{t,i}$ and the action it selected $a_{t,i}$ at time t

Notation Table II

Symbol	Explanation
$v(s_{t,i})$ $q(s_{t,i}, a_{t,i})$	The true state and action-state value functions that provided the actual value of how optimal the state $s_{t,i}$ and selected action $a_{t,i}$ are for agent i at time t. Usually, they are not known.
$\pi(a_{t,i}\|s_{t,i})$	The policy that selects the action $a_{t,i}$ given the state $s_{t,i}$ for the agent i at time t.
γ	The discount factor, usually $0 \leq \gamma \leq 1$, which discounts future rewards and values
$R_{t,i}$	The cumulative reward from time t and on, for agent i
$G_{t,i}$	The return, which is the cumulative reward from time t until the end of an episode (e.g.. when a goal is met or failed for an agent i).
$\pi_*(a_{t,i}\|s_{t,i})$	The optimal policy that maximizes cumulative reward and return
$o_{t,i}$	The environmental observation of an agent i at time t. Usually modelled along with state.

Notation Table III

Symbol	Explanation
α	The learning rate in temporal difference models, usually $0 \leq \alpha \leq 1$
$\max_x f(x)$	The maximization of a function $f(x)$ in regards to x
$\min_x f(x)$	The minimization of a function $f(x)$ in regards to x
w	A tensor of learnable parameters for value estimation
$\overline{VE(w)}$	Value estimator of via parameters w
η	The learning rate for value and policy estimation via parameter learning
$\nabla_w f(w)$	The gradient of function $f(w)$ in regards to elements of w
$\mu(s)$	On-policy distribution. The amount of timesteps spent (or expected to be spent) on the state s
θ	A tensor of learnable parameters for policy estimation
$J(\theta)$	Policy estimator via parameter learning