ETH zürich

Reinforcement Learning for ABM

Presented

by Thomas Asikis, asikist@ethz.ch

Agent-Based Modeling and Social System Simulation

Fall Semester 2019

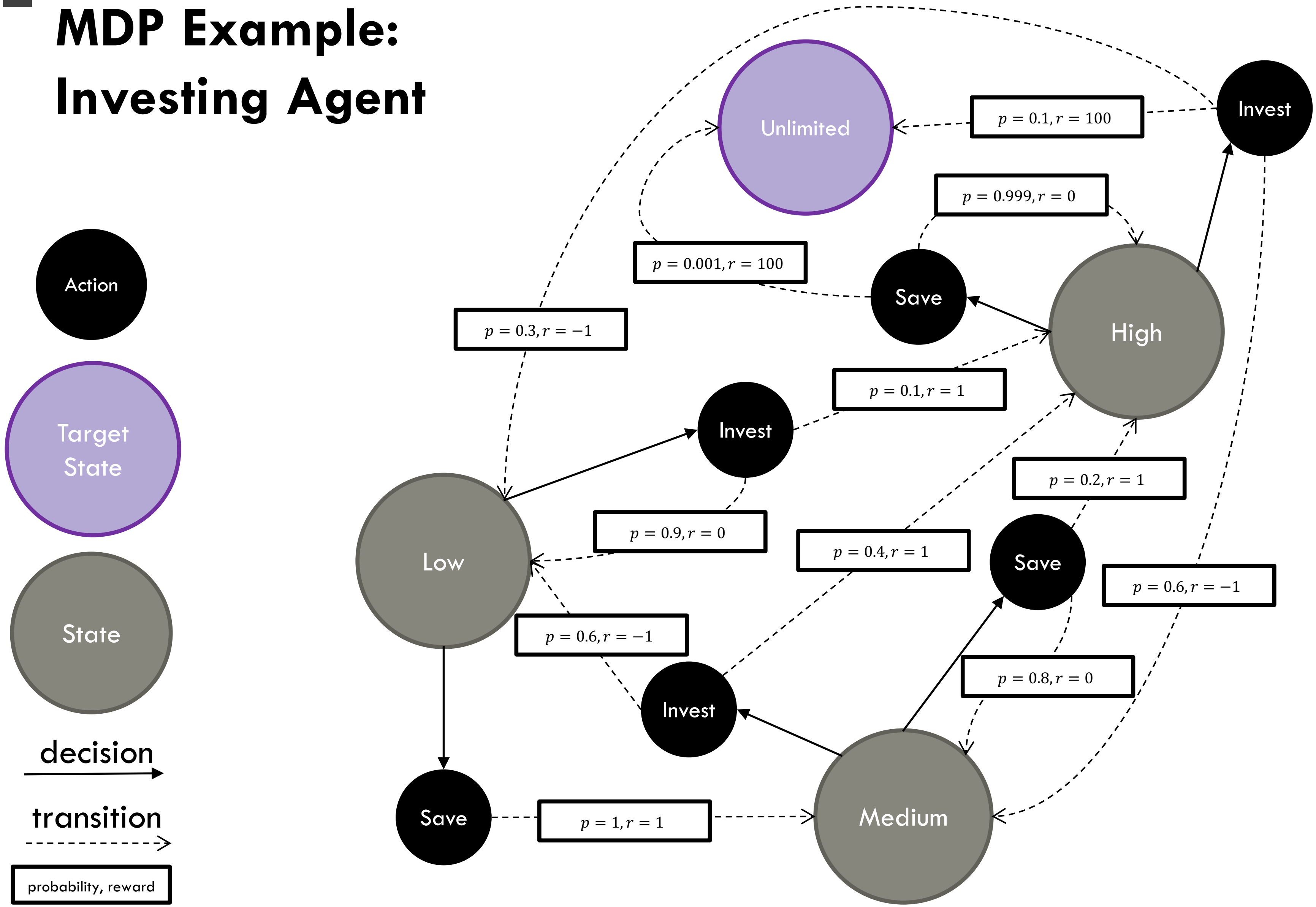
In Today's Course

- ~~An agent learns to become billionaire…~~

- How to design agents that learn to optimize!
- Relevant code is found in:

  https://github.com/asikist-ethz/reinforcement_learning

MDP Example: Investing Agent



* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction.* MIT PRESS. Ch.3 Finite Markov Decision Processes, p.52

Intro

Reinforcement Learning Agents

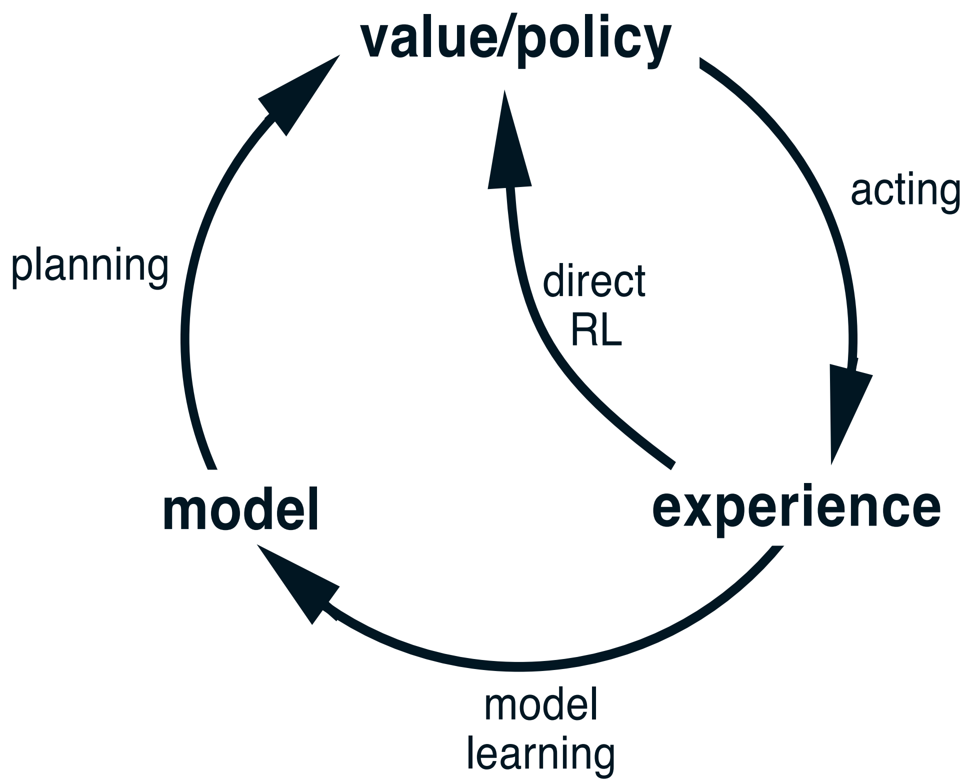
- Main elements: an *agent* and an *environment*.
- ”*A goal directed agent in an uncertain environment*”.
- Learn a behavior that achieves a goal by interacting with environment:
  - ↳ Behavior: choice of actions.
- Maximization of a reward (or minimization of cost).
- Multi Agent simulations can be done via Independent Reinforcement Learning, and then extended by using communication

Reinforcement Learning and Machine Learning

Machine Learning	Supervised	Learning from labeled set - External supervisor
		External supervisor
		Certainty: for given input always the same correct output
	Unsupervised	Finding structure/patterns in data.
		No goal
		Descriptive
	Reinforcement	Goal Driven agents
		Learn to optimize: Exploration vs Exploitation
		Uncertainty: same input does not always result to same output
		Sequential: The current model output affects future received inputs
		Delayed feedback, as the evaluation of the current output may be included in a later sample.

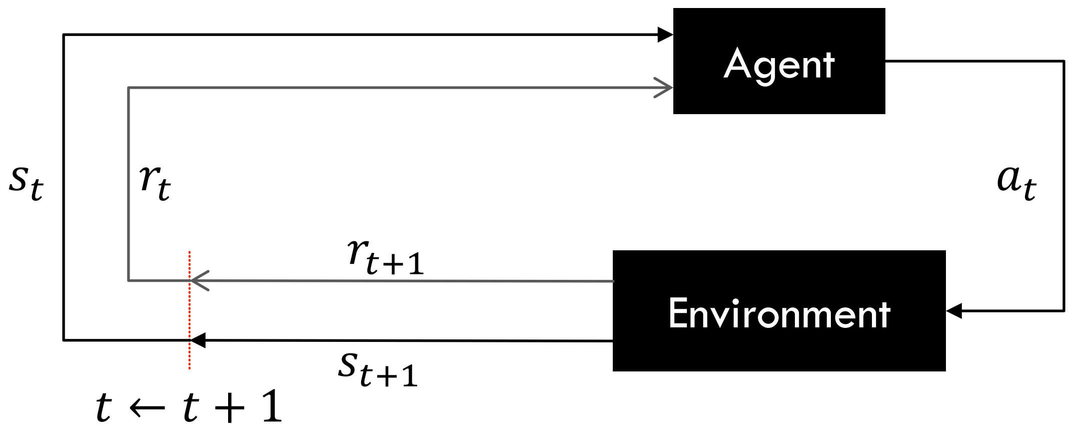
* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS. Ch.1Introduction, p.2

Reinforcement Learning and Planning



* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS.
Ch.8 Planning and Learning with Tabular Methods, p.162-163

Markov Decision Process

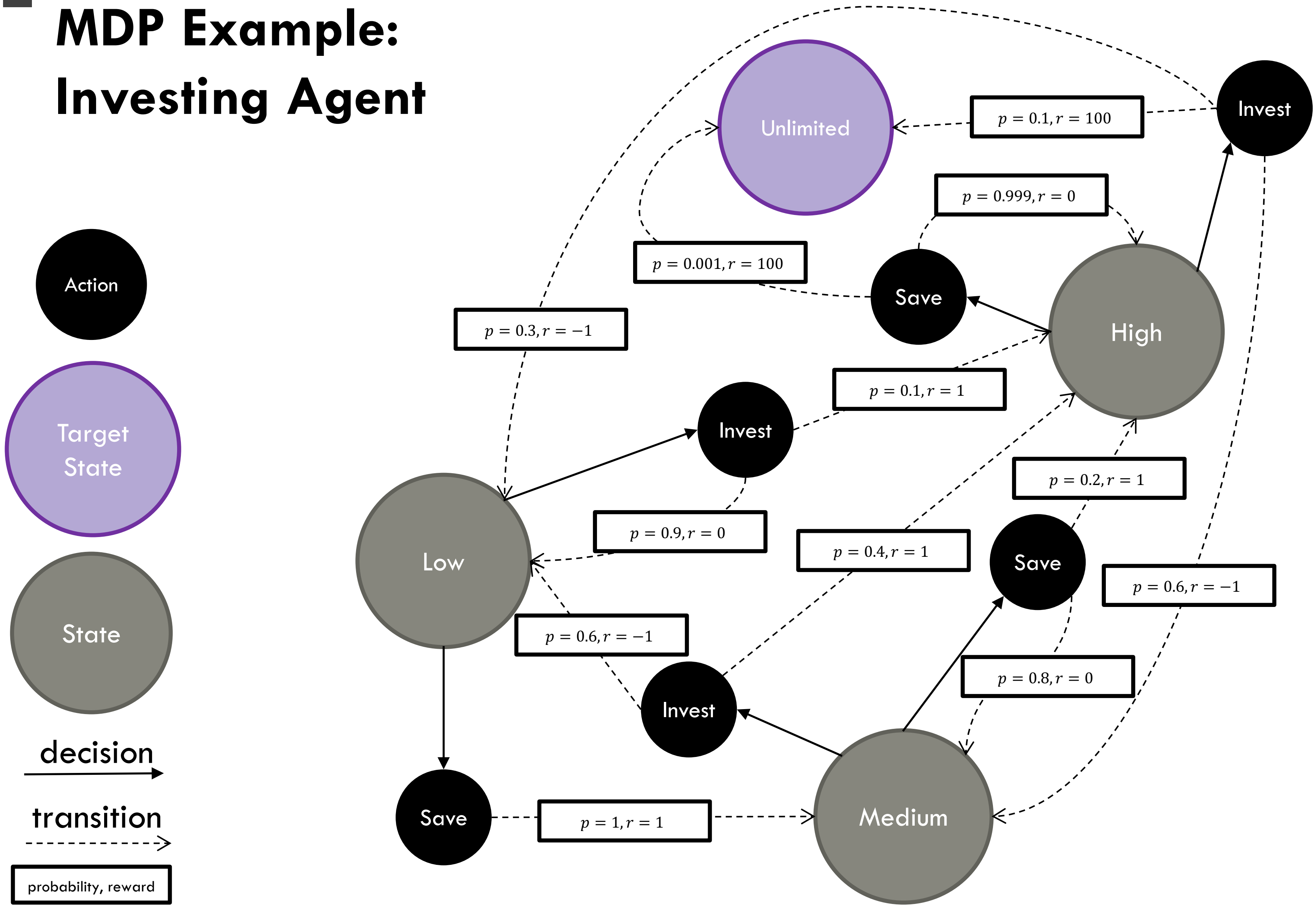


s_1
s_2
.
.
.
$\boldsymbol{s_T}$ Terminal or Goal State

Agent:		The learner and decision maker
Environment:		The *agent* interacts with the environment, which comprises everything outside the agent.
Time	t:	Discrete timesteps (Discrete time).
State	s:	The agent's perception about the environment.
Action	a:	An action the agent takes based on observing S.
Reward	r:	Consequence of action.

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS.
Ch.3 Finite Markov Decision Processes, p.47

MDP Example: Investing Agent



* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction.* MIT PRESS. Ch.3 Finite Markov Decision Processes, p.52

MDP Example ~ Accumulating Experience

s	a	s'	$r(s, a, s')$
Low	Invest	Low	0
Low	Save	Medium	1
Medium	Save	Medium	0
Medium	Invest	High	0
High	Invest	High	0
High	Invest	Low	0

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: $s_{t+1} = g(s_t, a_t)$

State transition $g(s_t, a_t)$

Usually it is used to model the environment, and the true function is unknown.

- Deterministic
- Stochastic
  - ↳ Same action-state may lead to different states. This is often the case for a single agent in a single agent environment.
- Affected by:
  - Current action and state
  - and past actions and states

Policy

The strategy to select an action:

$$\pi(a_t|s_t)$$
$$\pi: \mathbb{S} \rightarrow \mathbb{A}$$

- Usually deterministic. Probabilistic mostly to explore new actions.

s	a
Low	Save
Medium	Save
Medium	Invest
High	Buy

Reward

- Skirner, Behavioral Theory
- Positive is encouragement
- Negative is punishment
- Reward ≠ Objective:
  e.g. difference of objective values in consecutive timesteps:

$$\mathrm{r_t} = O_t - O_{t-1}$$

*Skinner, Burrhus F. "Operant behavior." *American psychologist*18.8 (1963): 503.

Cumulative Reward & Return

- Reward in the long term
- Greedy selection of current maximum reward, or …
- Non-optimal selection now, long term optimization later
- Episode: interval of timesteps until goal or failure ($t = T$)
- Usually a discount factor is used (discounted reward):

$$R_t = \sum_{k=0} \gamma^k r_{t+k} \;, 0 < \gamma < 1$$

- The return is the cumulative reward at the end of an episode*:

$$G_t = \sum_{k=0}^{T} \gamma^k r_{t+k}$$

State Value Function

- Estimate how good is to for an agent to be in a state
  ➥ in terms of acquiring high future rewards
- True state value function $v(s)$
- Each policy has its own true state value function:

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

- Usually $v_\pi(s)$ is unknown, so we need to approximate it:

$$V_\pi(s) \to v_\pi(s)$$

- For fixed policy, the Bellman equation can be derived:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(s', r|a, s)\,[r + \gamma v_\pi(s')]\,, \forall s \in \mathbb{S}$$

Action-State Value Function

- Estimate how good is to for an agent to be in a state

  ➥ in terms of acquiring high future rewards
- True state value function $q(s)$
- Each policy has its own true state value function:

$$q_\pi(a|s) = \mathbb{E}_\pi(G_t|S_t = s, A_t = a)$$

- Usually $q_\pi(a|s)$ is unknown, so we need to approximate it:

$$Q_\pi(a|s) \to q_\pi(s)$$

- For fixed policy, the Bellman equation can be derived:

$$q_\pi(a|s) = \sum_{r,s'} p(s', r|a, s)\left[r + \gamma \sum_{a'} \pi(a'|s') Q_\pi(a|s))\right]$$

The Prediction Problem: Estimating values

- Estimating a true value function given a policy (Policy Evaluation).
- Several parts of the Bellman equation can be estimated (if unknown) to predict the actual value value function.
- Different methods, e.g. Monte-Carlo, Dynamic Programming, Temporal Differences have different approaches to prediction.
- There are methods without value estimation for policy optimization: genetic algorithms, simulated annealing.

The Control Problem: Finding an Optimal Policy

Optimal policy: The policy that maximizes return by selecting optimal actions:

$$\pi_*(a_t|s_t, o_t)$$

The optimal policy has the optimal value functions:

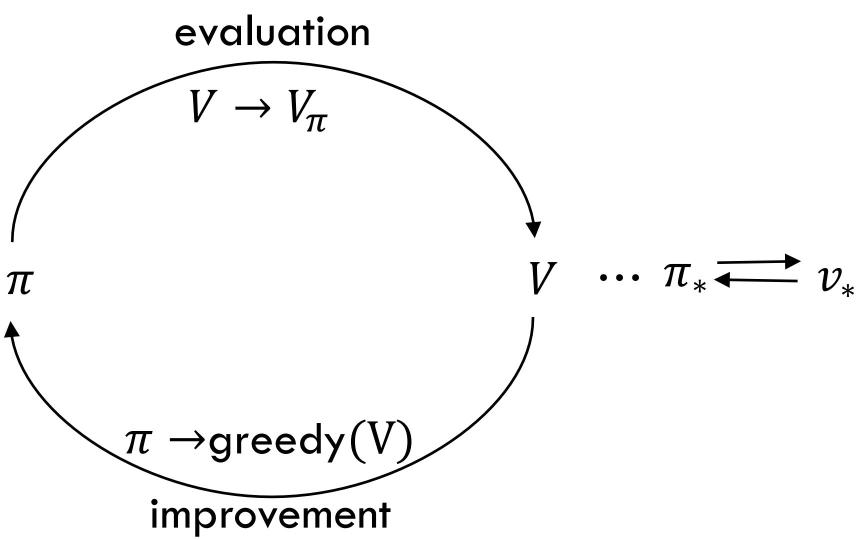
$$v_*, q_*$$

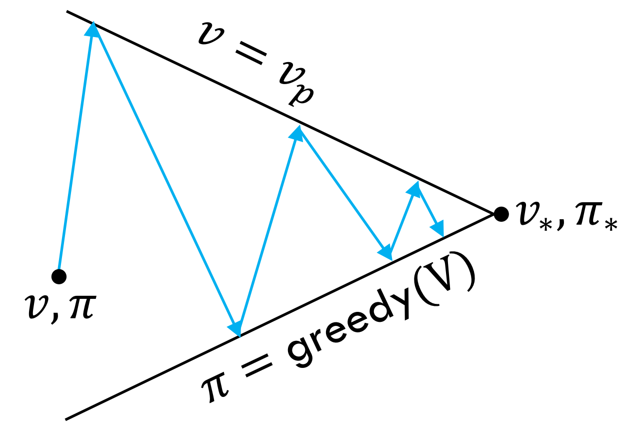
To optimize the policy, many iterations are repeated over all states updating the policy, until the policy is stable:

i.e. for all states, it yields the same action for the same state in consecutive iterations.

Generalized Policy Iteration

Solving both problems simultaneously:





* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction.* MIT PRESS.
Ch.4 Dynamic Programming, p.86

Dynamic Programming

- The dynamics of the system are known, e.g. transition probabilities and rewards in MDP.
- We can then derive a policy iteration algorithm that solves the prediction and control problems efficiently.
- Polynomial time with states and actions, less than direct search which takes exponential time.
- Asynchronous dynamic programming can be used to avoid systematic sweeps of the whole state space. Still, all states need to be visited at least once.
- Next slides contain pseudocode relevant to the prediction and control problem in DP!

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : an introduction*. MIT PRESS.
Ch.4 Dynamic Programming, p.80

Policy Evaluation

Policy Evaluation Algorithm

- **Input:** $\pi(s)$, policy to evaluate
- **Parameters:** $\theta > 0$, small threshold for accuracy of estimation
- **Variable:** $V(s)\ \forall\, s \in S$: a dictionary/map such that:
  $V(s) \sim \mathcal{U}\ \forall\, s \in S^{+}$ and $V(s_{terminal}) = 0$

Do:

$\Delta \leftarrow 0$

For each $s \in S$:

$v \leftarrow V(s)$

$v \leftarrow \sum_a \pi(a|s)\left[\sum_{r,s'} p(s', r|s, a)[r + \gamma V(s')]\right]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

While $\Delta < \theta$

Return $V(s)$

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : An introduction.* MIT PRESS.
Ch.4 Dynamic Programming, p.75

Complementary Notation:

$\mathcal{U}$:	A uniform distribution
$s_{terminal}$:	The terminal state
S^{+}:	The set of all states except terminal

Policy Improvement

Policy Improvement Algorithm

- **Input:** $V(s)$, a value function for states
- **Variables:** $\pi(s)$ a randomly generated policy

  policy-stable ← true

For each $s \in S$:

$a_{old} \leftarrow \pi(s)$

$\pi(s) \leftarrow \underset{a}{\operatorname{argmax}}\left[\sum_{r,s'} p(s', r|s, a)[r + \gamma \mathrm{V}(s')]\right]$

policy-stable ← $a_{old} = \pi(s)$

Return $\boldsymbol{\pi(s)}$, policy-stable

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : An introduction.* MIT PRESS.
Ch.4 Dynamic Programming, p.80

Policy Iteration

Policy Iteration Algorithm

- **Variables** : $V(s)$, any value function for states
  $\pi(s)$, any policy to decide actions
  policy-stable ← true

Do:

$V(s)$ ← Policy Evaluation Algorithm($\pi(s)$)

$\boldsymbol{\pi}(\boldsymbol{s})$, policy-stable ← Policy Improvement Algorithm($V(s)$)

While not policy-stable

Return $\boldsymbol{\pi}(\boldsymbol{s}), \boldsymbol{V}(\boldsymbol{s})$

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : An introduction.* MIT PRESS.
Ch.4 Dynamic Programming, p.80

Value Iteration

Value Iteration Algorithm

- **Variables:** $V(s)$, any value function for states
  $\pi(s)$, any policy to decide actions

Do:

$\Delta \leftarrow 0$

For each $s \in S$**:**

$v \leftarrow V(s)$

$v \leftarrow \max_a \left[\sum_{r,s'} p(s', r|s, a)[r + \gamma V(s')]\right]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

While $\Delta < \theta$

For each $s \in S$**:**

$\pi(s) \leftarrow \operatorname{argmax}_a \left[\sum_{r,s'} p(s', r|s, a)[r + \gamma V(s')]\right]$

Return $V(s)$, $\pi(s)$

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : An introduction.* MIT PRESS. Ch.4 Dynamic Programming, p.83

Monte Carlo Control

- The dynamics of the system are unknown.
- We can sample action-state pairs and estimate their returns for entire.
- Monte Carlo methods will estimate values asymptotically.
- Assumptions:
  - exploring starts (starting from all possible states)
  - Infinite samples of episodes
- Assumptions can be lifted via:
  - ϵ-soft policies (always non-zero probability for taking an action, exploring via "random" actions)
  - Importance sampling (sampling episodes that contain more information about value estimation and policy optimization)
- Next slides contain pseudocode relevant to the prediction and control problem in MC!

Sample experiences

```
Play Episode Algorithm
Input:   π(s), policy to evaluate
         environment

experiences ← list
s ← environment.initial_state()
While s ≠ s_terminal :
   a ← π(s)
   s′,r ← environment.apply(a)
   experiences.add(s,a,s′,r)

Return experiences
```

On Policy Iteration First Visit Monte Carlo

* SUTTON, R. S. . B. (2018). *REINFORCEMENT LEARNING : An introduction.* MIT PRESS. Ch.4 Dynamic Programming, p.101

```
Policy First Visit Monte Carlo Algorithm
▪ Inputs: ϵ-soft π(s), environment
▪ Parameters: γ,ϵ
▪ Variables: V(s) , total_episodes ,
    Q(s, a), the expected return after taking and action a on state s
    state_returns: {a, s, list}, Structure for all observed returns for each pair (a,s)
For episode in total_episodes:
  experiences ← Play Episode Algorithm(π(s), environment)
   G ← 0,  T ← experinces.length
   For t in [T,T − 1, … ,0] :
    s_t, a_t, s_t+1 , r_t ←  experiences.get(t)
    G ← γG + r_t
     If (a_t, s_t) not in experiences.get(t − 1, t − 2, … , 0):
          state_returns.get((a_t, s_t)).append(G)
          Q(s, a) ← mean(state_returns.get((a_t, s_t)))
         a* ← argmax_a Q(s_t, a)
          For a in 𝒜_t: #𝒜_t ← 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 $\pi(a|s_t)$

Summary

- Reinforcement Learning can be used for uncertain and dynamic environments
- Exploitation vs Exploration
- Control Problem: optimal policy ~ policy iteration
- Prediction Problem: optimal estimation ~ policy evaluation
- Dynamic Programming: System dynamics are known, guaranteed convergence & optimality
- Monte Carlo: Sample whole episodes, asymptotically optimal

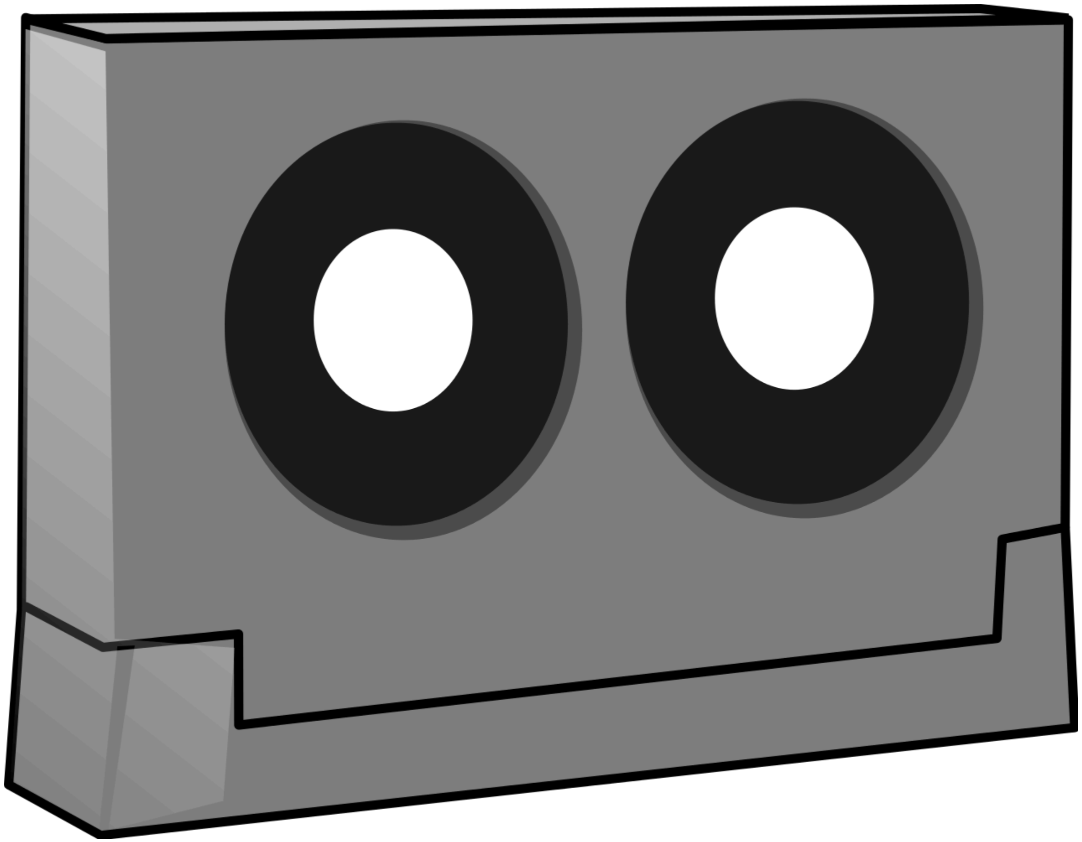
Questions

?

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

Notation Table I

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

Notation Table II

Symbol	Explanation
$v(s_t)$ $q(s_t, a_t)$	The true state and action-state value functions that provided the actual value of how optimal the state s_t and selected action a_t are for agent at time t. Usually, they are not known.
$\pi(a_t \| s_t)$	The policy that selects the action a_t given the state s_t for the agent at time t.
γ	The discount factor, usually $0 \leq \gamma \leq 1$, which discounts future rewards and values
R_t	The cumulative reward from time t and on, for agent.
G_t	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).
$\pi_*(a_t \| s_t)$	The optimal policy that maximizes cumulative reward and return.
o_t	The environmental observation of an agent at time t. Usually modelled along with state.