Stockholms universitet

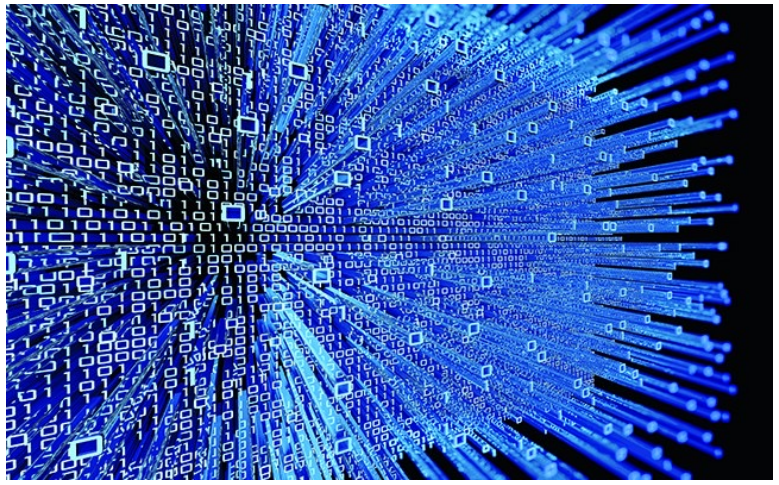
Reinforcement Learning

Data-Driven Decision Making

Machine Learning

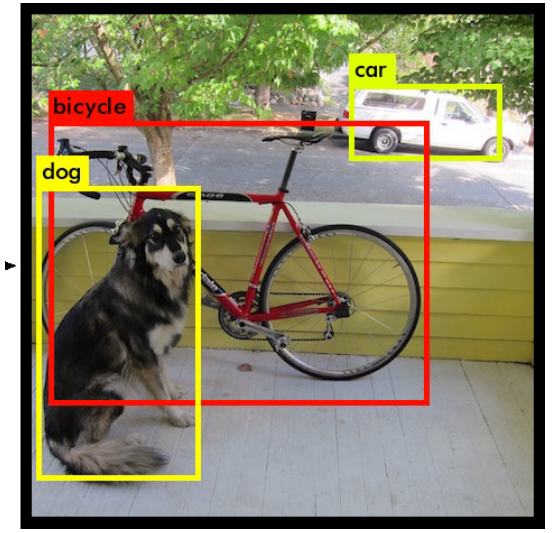
Sindri Magnússon

Machine Learning



Data

Training



Predictive model

The study of how software agents learn to make good **predictions** or **decisions** from **experience** or **data**

Today: AI/Data-Driven Decision Making

How can a software agents learn to make good **decisions** from **experience** or **data**?

Reinforcement Learning

Goal: Learn to make **optimal sequences** of decisions in an unknown enviroment by **Trial and Error**

What does the AI AGENT Learn?

Goal:

$$\underset{\pi(\cdot) \in \mathbf{\Pi}}{\text{maximize}} \quad \sum_{t=1}^{T} R_t$$

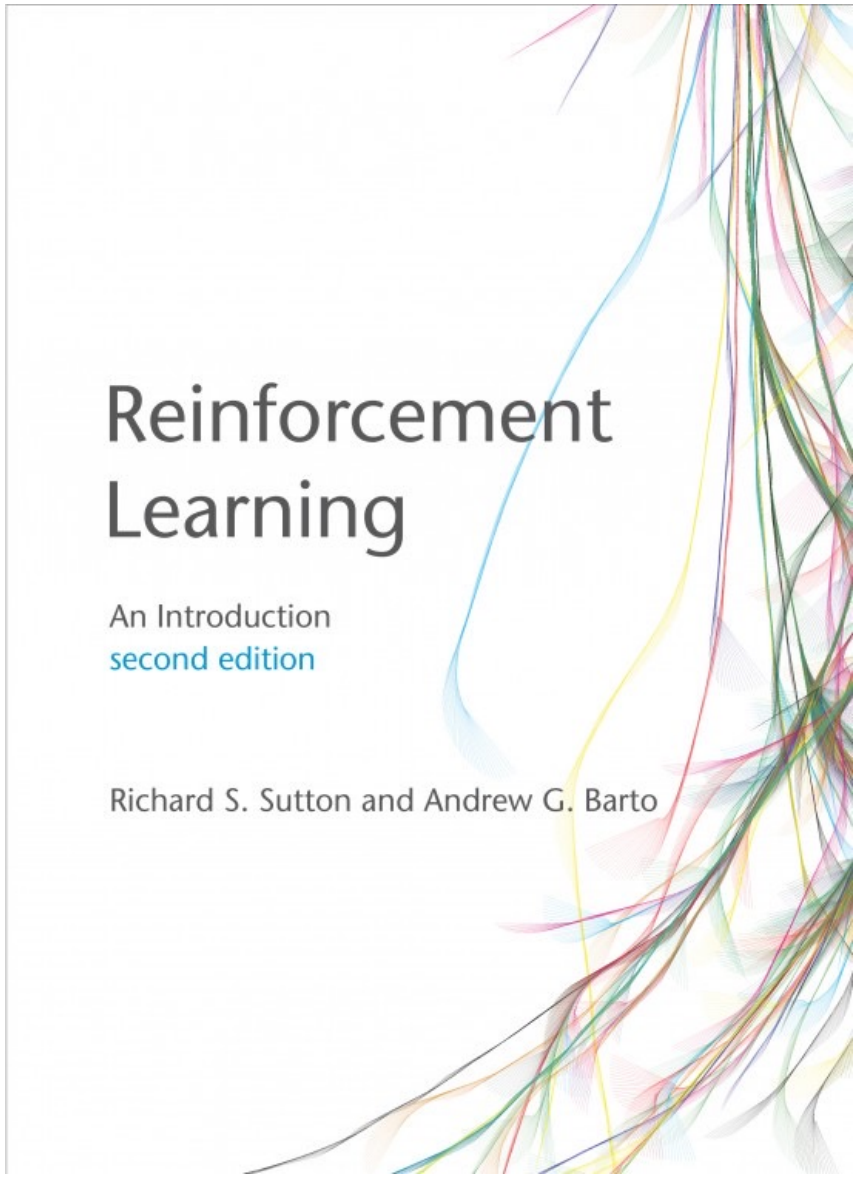
Subject to $a_t = \pi(s_t)$

Policy (function from states to action):

$$a_t = \pi(s_t)$$

Main Characteristics of RL

1. Optimization
2. Exploration-Explotation
3. Delayed Reward
4. Generalization



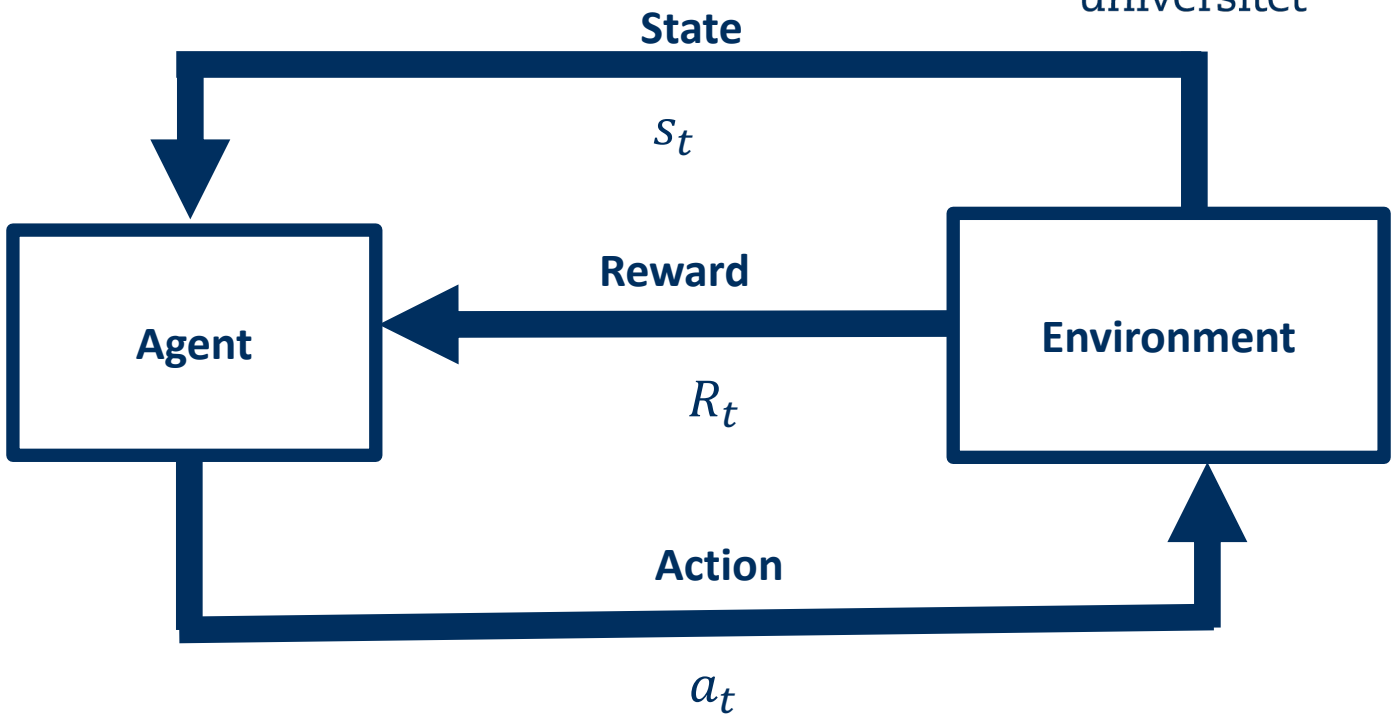
Stockholms universitet

Overview

- **Markov Decicion Process**
- Tabluar Methods
  - Value Evaluation
  - Policy Improvement
  - Q-learning
- Approximtion Methods
  - Deep Q-learning
  - Policy Gradient methods
- Conclusions

Markov Decision Process

- States: $s \in S$
- Actions: a $\in A$
- Rewards: R

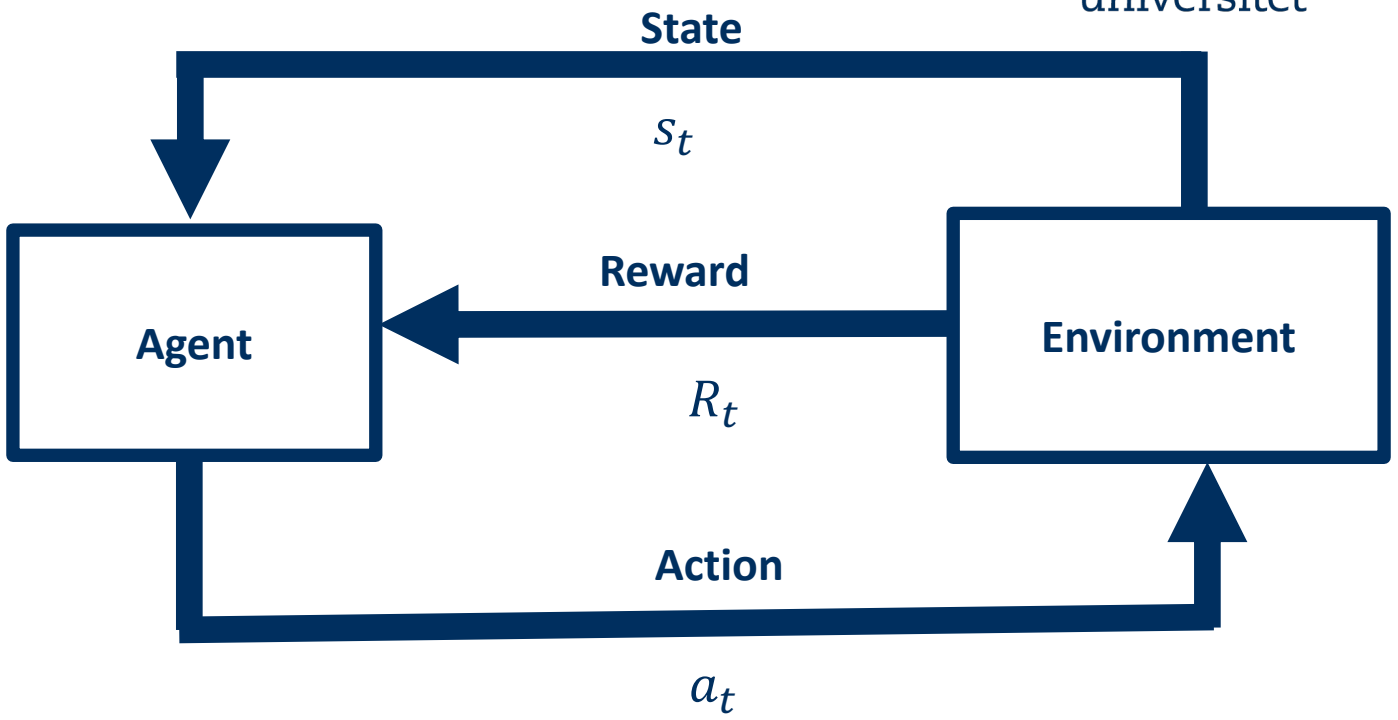


Markov Decision Process

- States: $s \in S$
- Actions: a $\in A$
- Rewards: R

- **Dynamics:**

$$p(s_{t+1}, | \underbrace{s_t, a_t, \ldots, s_0, a_0}_{\text{History}})$$



Markov Decision Process

- States: $s \in S$
- Actions: $a \in A$
- Rewards: R

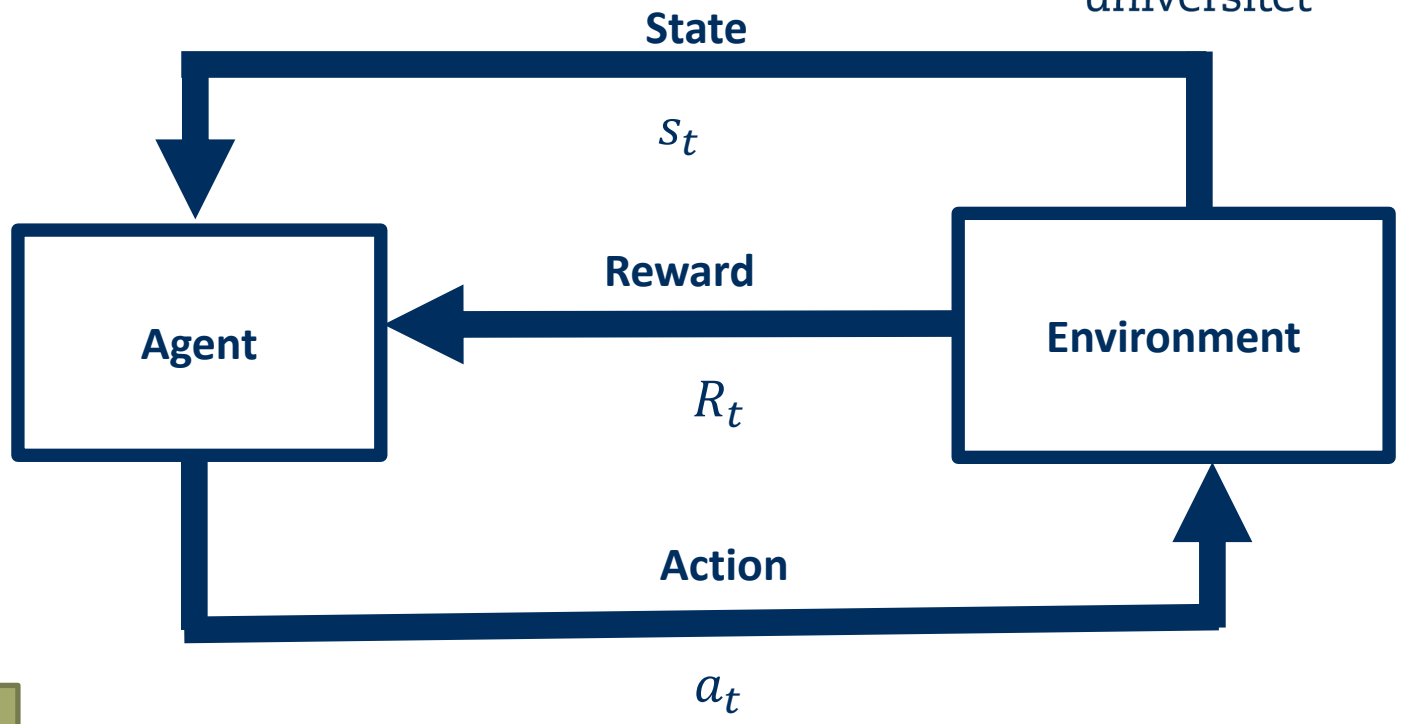
- **Dynamics:**

$$p(s_{t+1}, | \underbrace{s_t, a_t, \dots, s_0, a_0}_{\text{History}})$$

- **Markovian Assumption:**

$$p(s_{t+1} | s_t, a_t, \dots, s_0, a_0) = p(s_{t+1} | s_t, a_t)$$

- **Trajetory:** $s_0, a_0, R_1, s_1, a_1, R_2, \dots$



Markov Decision Process

- States: $s \in S$
- Actions: $a \in A$
- Rewards: R

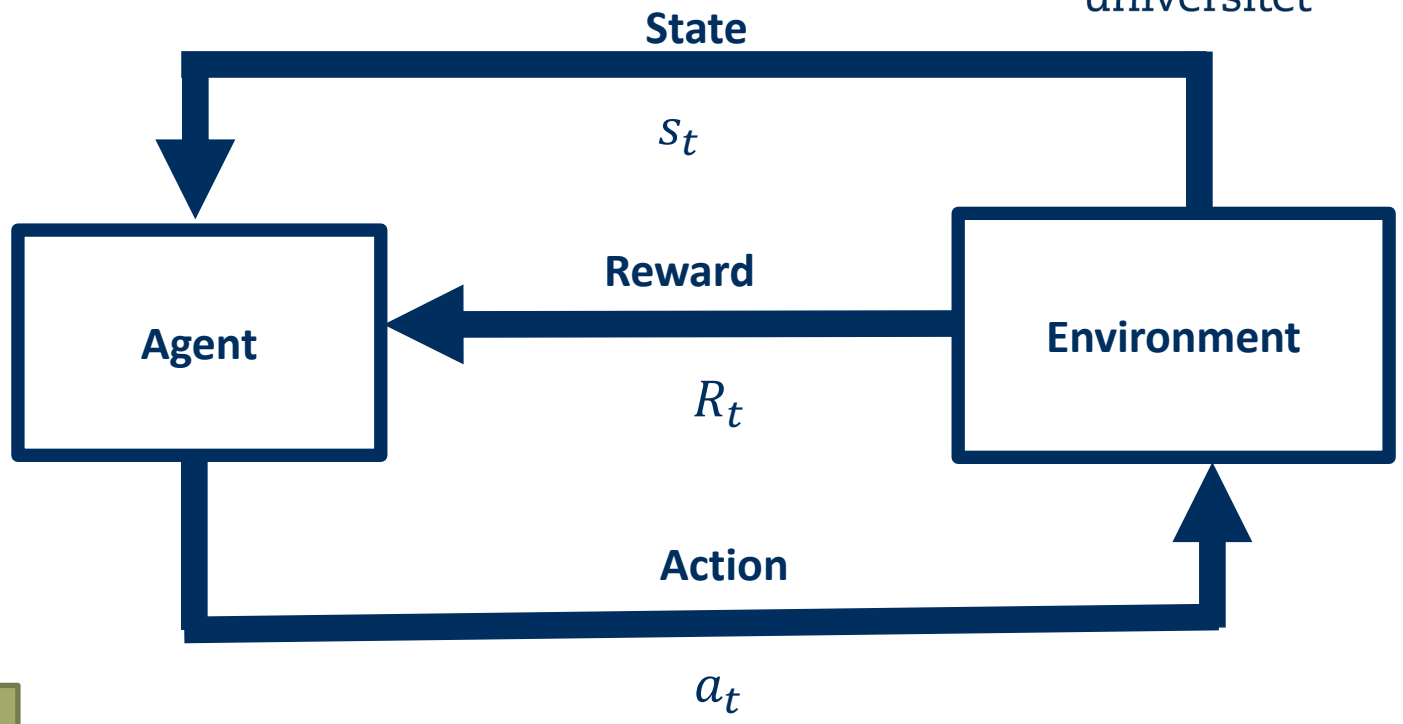
- **Dynamics:**

$$p(s_{t+1}, | \underbrace{s_t, a_t, \dots, s_0, a_0}_{\text{History}})$$

- **Markovian Assumption:**

$$p(s_{t+1} | s_t, a_t, \dots, s_0, a_0) = p(s_{t+1} | s_t, a_t)$$

- **Trajetory:** $s_0, a_0, R_1, s_1, a_1, R_2, \dots$

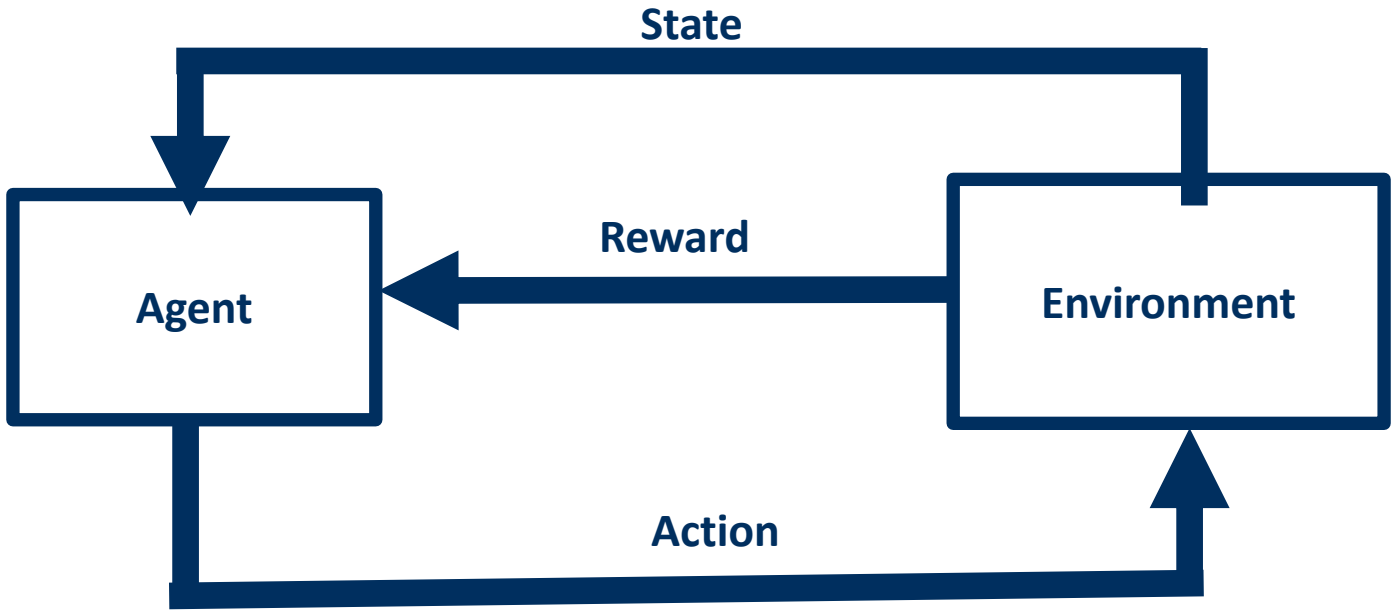


Goal:

$$\underset{\pi(\cdot) \in \mathbf{\Pi}}{\text{maximize}} \quad \sum_{t=1}^{T} R_t$$

Subject to $\quad a_t = \pi(s_t)$

Markov Decision Process



Time	1	2	3	4	5
State	Happy	Happy	Happy	Sad	Sad
Action	Positive	Negative	none	Positive	Postive
Revard	0	0	1	1	1

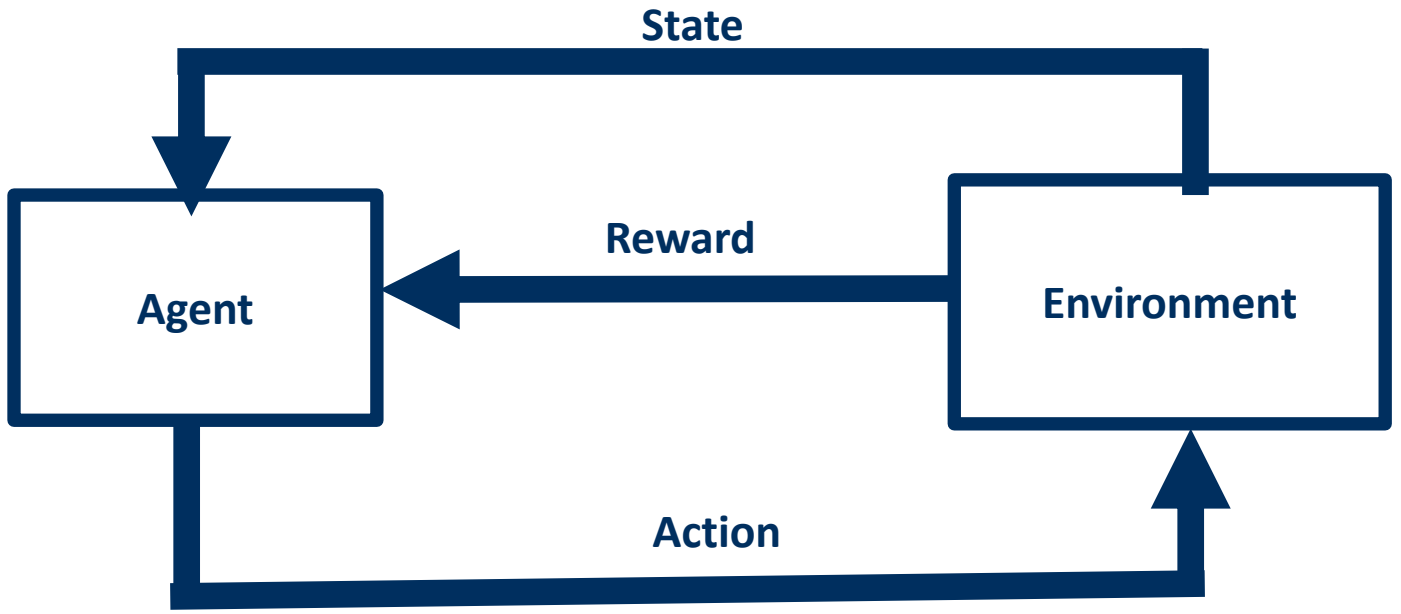
Message 1 Message 2 No Message

Markov Decision Process: Policy

- **Policy:** $\pi(s) = a$ with distribution $p^{\pi}(a|s)$

π(Happy)=
- Positive with probability 0.25
- Negative with probability 0.25
- None with probability 0.5

π(sad)=
- Positive with probability 0.5
- Negative with probability 0.25
- None with probability 0.25

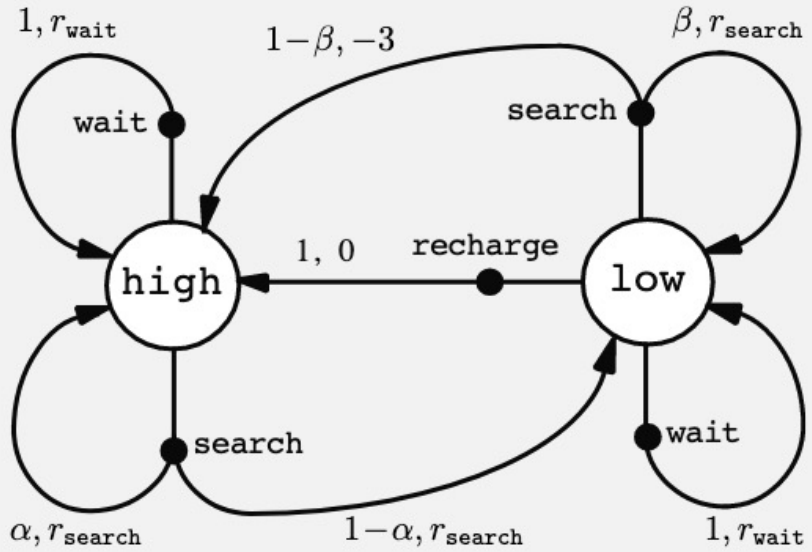


Goal: Find Optimal $\pi(\cdot)$

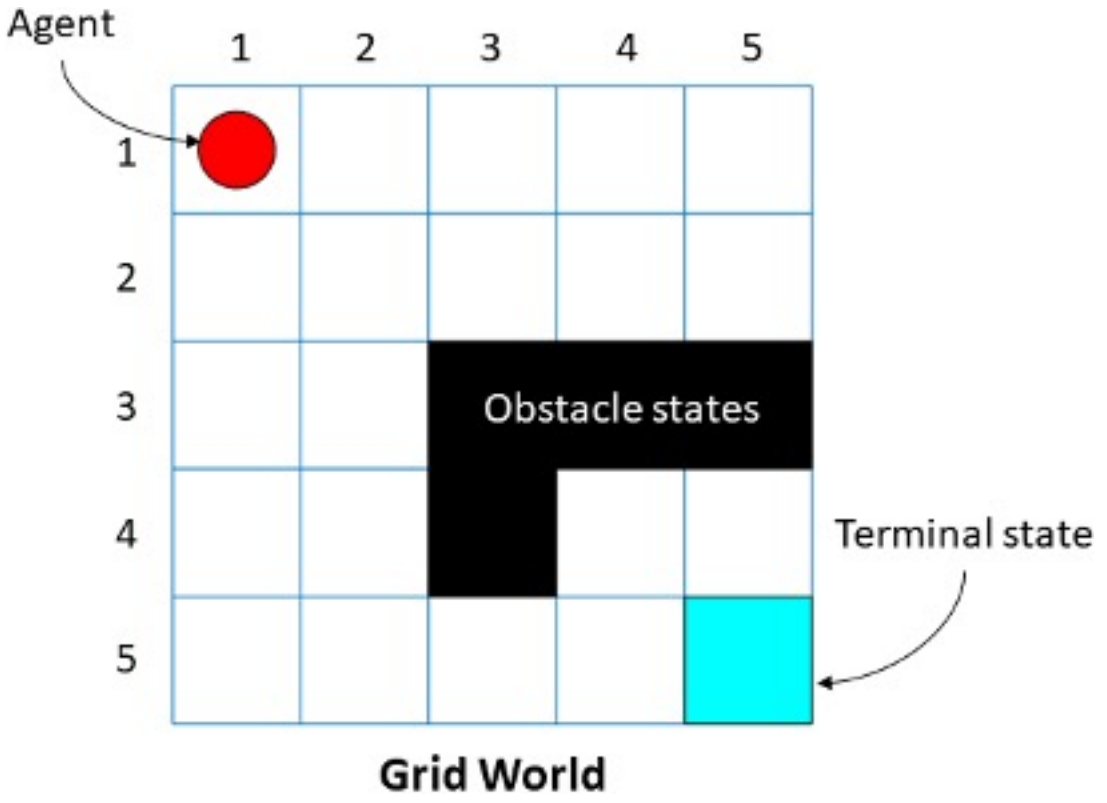
$$\max_{\pi} \sum_{t} R_t$$

Markov Decision Process

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1 - \alpha$	$r_{\texttt{search}}$
low	search	high	$1 - \beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-



Markov Decision Process



Markov Decision Process

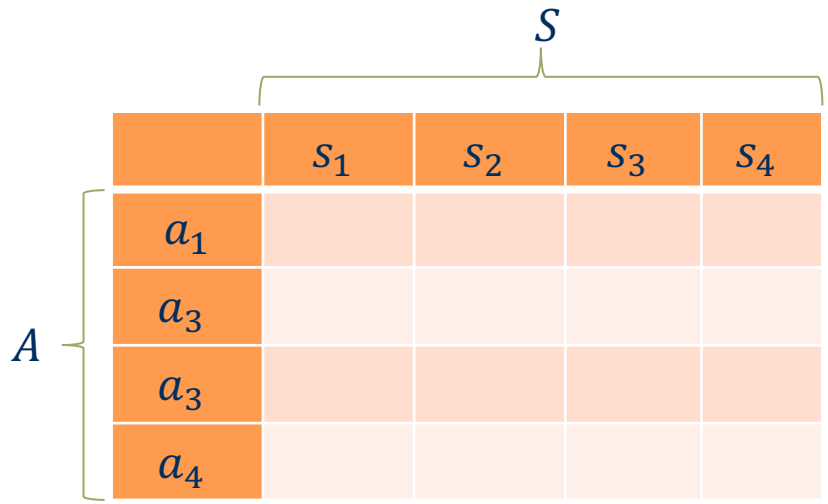
Overview

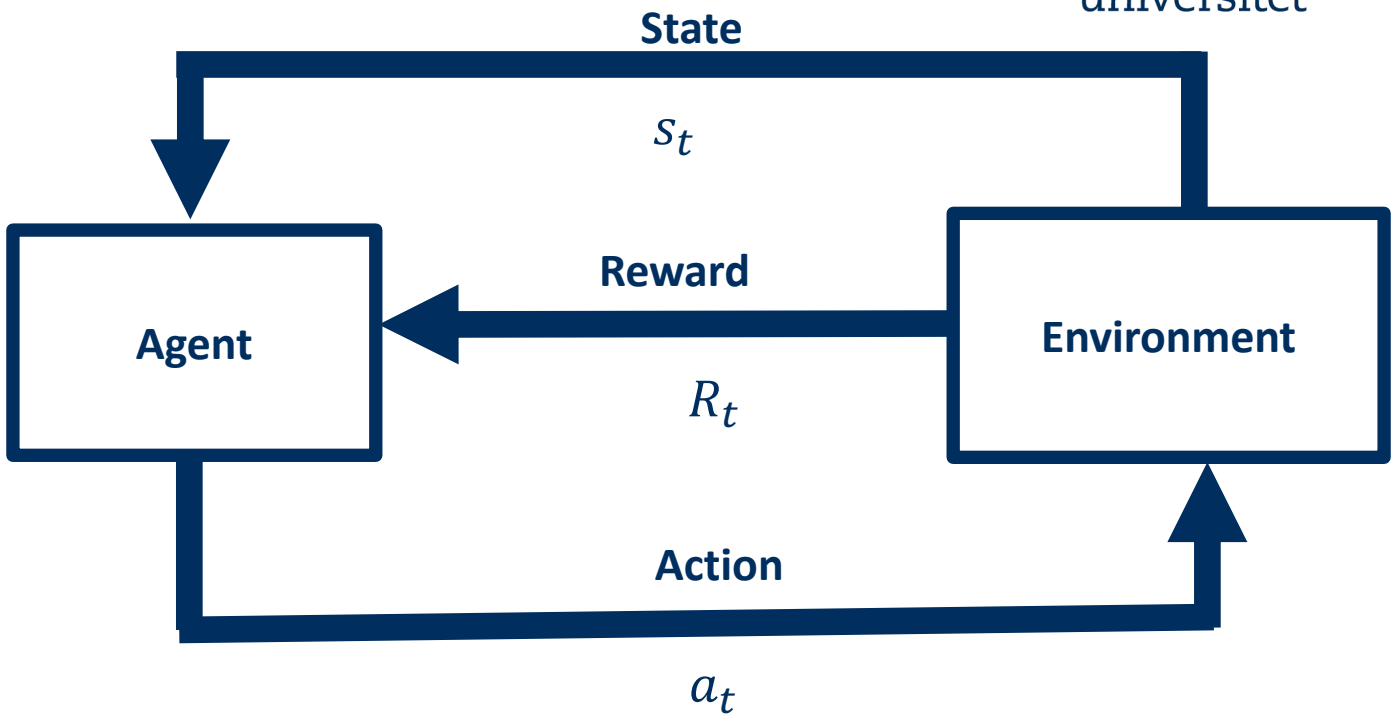
- Markov Decicion Process
- **Tabluar Methods**
  - Value Evaluation
  - Policy Improvement
  - Q-learning
- Approximtion Methods
  - Deep Q-learning
  - Policy Gradient methods
- Conclusions

Tabular Methods

- States: $s \in S$
- Actions: $a \in A$
- Rewards: R

- Assume that S and A are finite



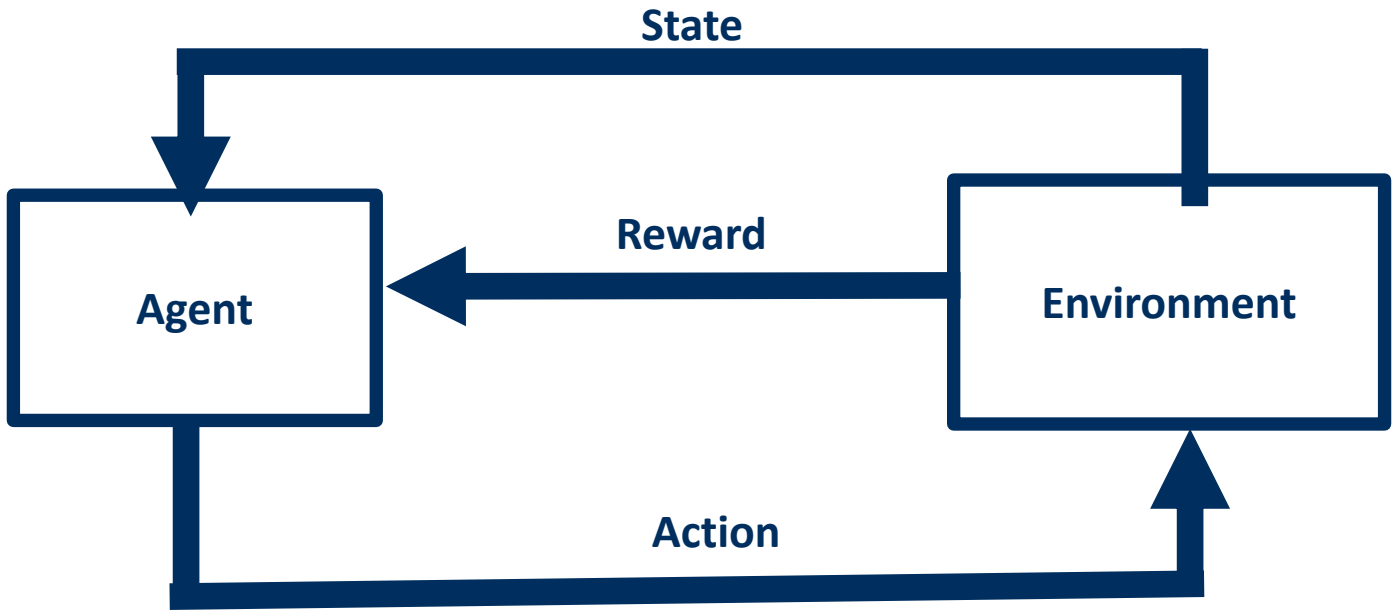


Overview

- Markov Decicion Process
- **Tabluar Methods**
  - **Value Evaluation**
  - Policy Improvement
  - Q-learning
- Approximtion Methods
  - Deep Q-learning
  - Policy Gradient methods
- Conclusions

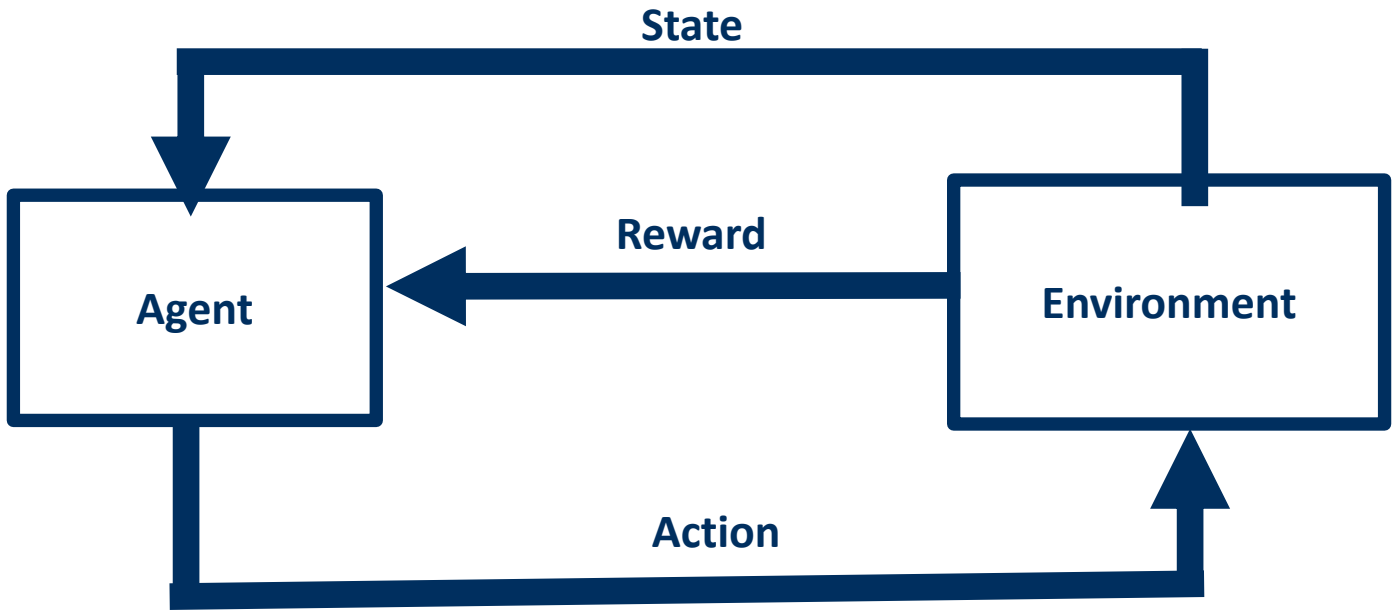
Value function

- **Policy:** $\pi(s) = a$ with distribution $p^{\pi}(a|s)$
- **Value:**
  - $V^{\pi}(s) = \boldsymbol{E}\left[\sum_{t=0}^{\infty} R_t \mid s_0 = s\right]$
- **How to evalute a policy?**



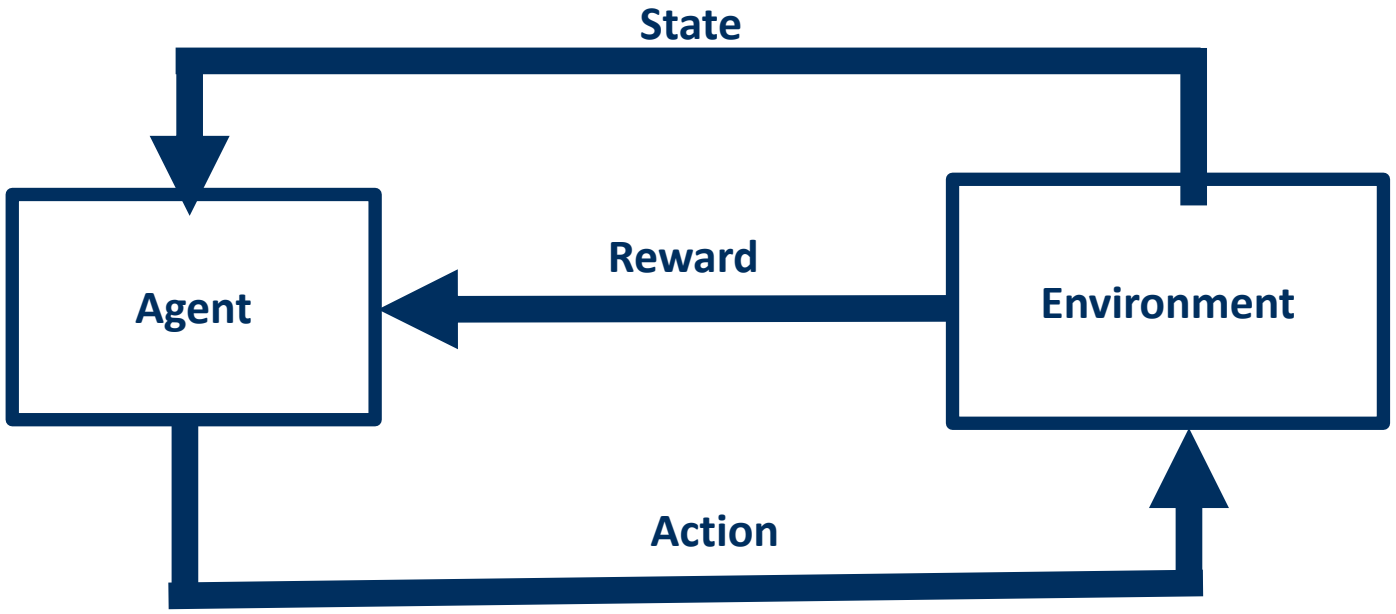
Value function

- **Policy:** $\pi(s) = a$ with distribution $p^{\pi}(a|s)$
- **Value:**
  - $V^{\pi}(s) = \boldsymbol{E}\left[\sum_{t=0}^{\infty} R_t \mid s_0 = s\right]$
- How to evalute a policy?
  - **If dynamics are know:** Dynamic programming
  - **If dynamics are not know:** Approximate dynamic programming



Dynamic Programming: Known Dynamics and Rewards

- **Know:**
  - $p(s_{t+1}|s_t, a_t)$
  - $r(s,a) = \boldsymbol{E}[R_{t+1}|s_t = s, a_t = a]$
- **Goal:** Learn the value
  - $V^{\pi}(s) = \boldsymbol{E}\left[\sum_{t=0}^{\infty} R_t \mid s_0 = s\right]$
- **Bellmann Equation:**
  - $V(s) = r^{\pi}(s) + \sum_{s'} p(s'|s,a)V(s')$
  - $r^{\pi}(s) = \sum_{a \in A} \pi(a|s) r(s,a)$



Dynamic Programming: Known Dynamics and Rewards

- **Know:**
  - $p(s_{t+1}|s_t, a_t)$
  - $r(s,a) = \boldsymbol{E}[R_{t+1}|s_t = s, a_t = a]$
- **Goal:** Learn the value
  - $V^{\pi}(s) = \boldsymbol{E}\left[\sum_{t=0}^{\infty} R_t \mid s_0 = s\right]$
- **Bellmann Equation:**
  - $V(s) = r^{\pi}(s) + \sum_{s'} p(s'|s,a)V(s')$
  - $r^{\pi}(s) = \sum_{a \in A} \pi(a|s) r(s,a)$

- **Dynamic programming:**

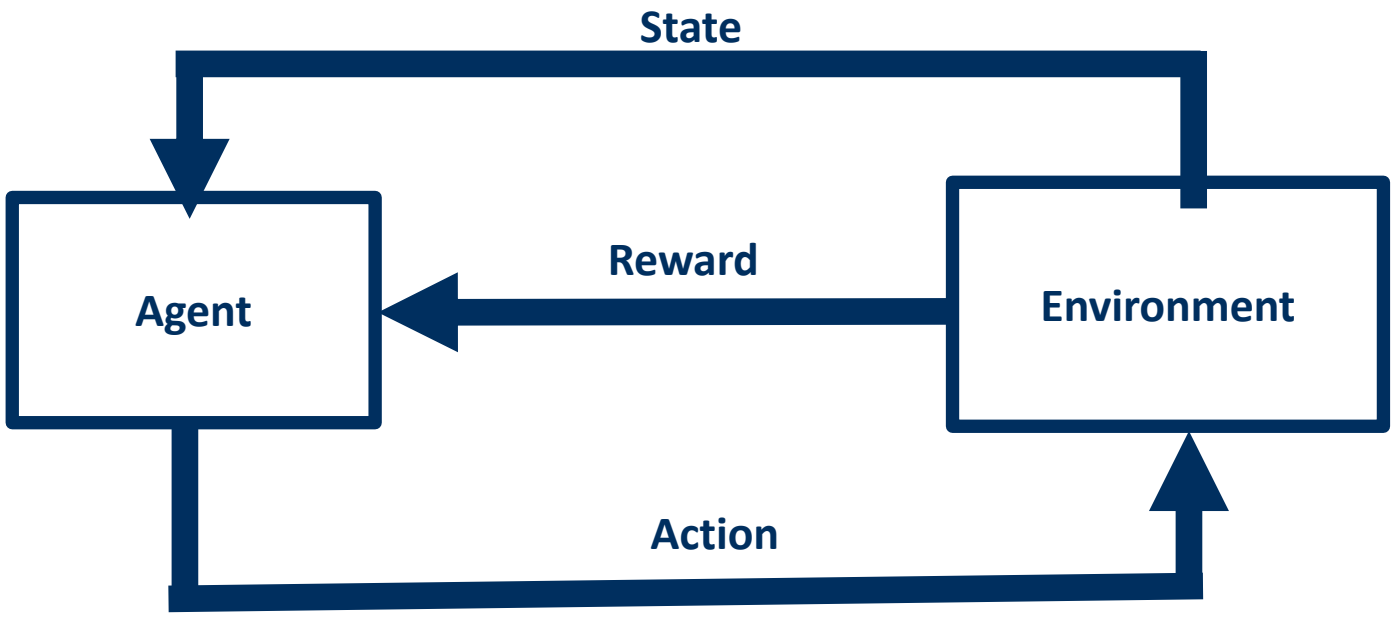
```
Initialize V̂_0(s) for all s
for t= 1,2,3 …
    for s∈ S
        V̂_{t+1}(s) = r^π(s) + Σ_{s'} p(s'|s,a) V̂_t(s')
```

- **Converges to $\mathbf{V}(\cdot)$:**

$$\lim_{i \to \infty} \hat{V}_i(\cdot) = V(\cdot)$$

Approximate Dynamic Programming

- **Know:**
  - $p(s_{t+1}|s_t, a_t)$
  - $r(s,a) = \boldsymbol{E}[R_{t+1}|s_t = s, a_t = a]$
- **Goal:** Learn the value
  - $V^{\pi}(s) = \boldsymbol{E}\left[\sum_{t=0}^{\infty} R_t \mid s_0 = s\right]$
- **Bellmann Equation:**
  - $V(s) = r^{\pi}(s) + \sum_{s'} p(s'|s,a)V(s')$
  - $r^{\pi}(s) = \sum_{a \in A} \pi(a|s) r(s,a)$



Approximate Dynamic Programming

- **Goal:** Learn the value
  - $V^{\pi}(s) = \boldsymbol{E}\left[\sum_{t=0}^{\infty} R_t \mid s_0 = s\right]$
- **Bellmann Equation:**
  - $V(s) = r^{\pi}(s) + \sum_{s'} p(s'|s,a)V(s')$
  - $r^{\pi}(s) = \sum_{a \in A} \pi(a|s) r(s,a)$

- **Policy Evaluation:**

```
Initialize V̂(s) for all s
for t= 1,2,3 …
```

$$a_t \sim \pi(\cdot\,|s^t)$$

$$\hat{V}(s_t) = \hat{V}(s_t) + \alpha[\mathrm{R}_{t+1} + \hat{V}(s_{t+1}) - \hat{V}(s_t)]$$

Learning rate

- **Converges to V(·):**

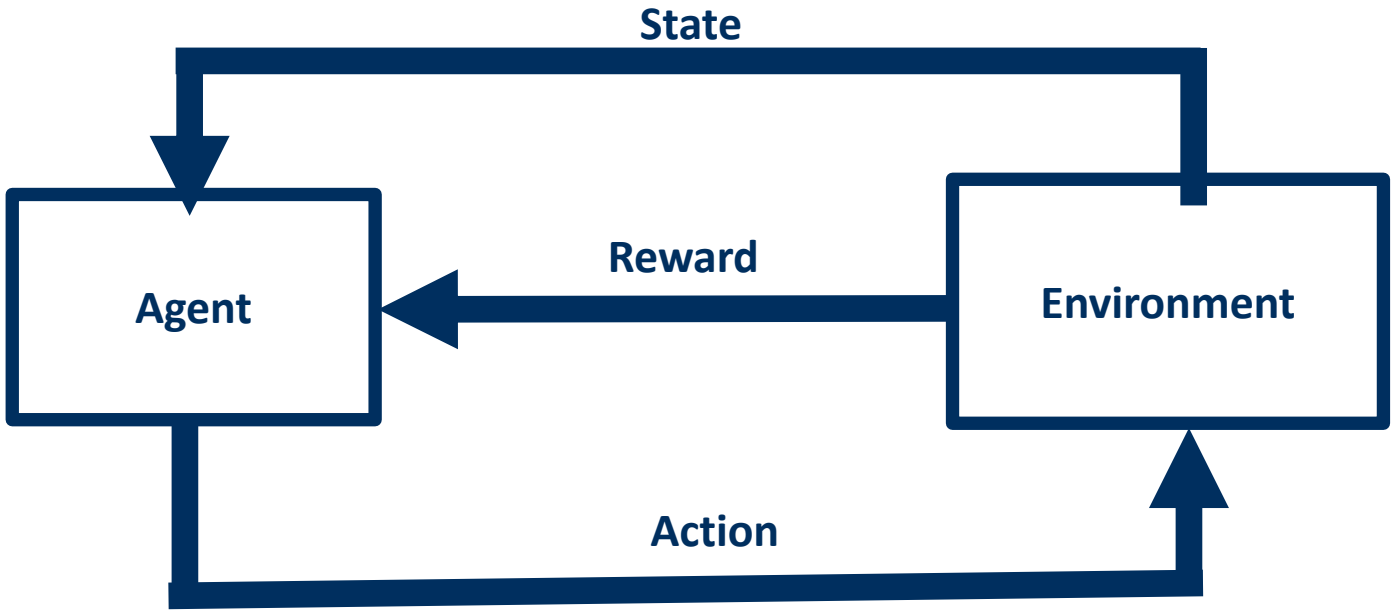
$$\lim_{i \to \infty} \hat{V}_i(\cdot) = V(\cdot)$$

Overview

- Markov Decicion Process
- **Tabluar Methods**
  - Value Evaluation
  - **Policy Improvement**
  - Q-learning
- Approximtion Methods
  - Deep Q-learning
  - Policy Gradient methods
- Conclusions

Dynamic Programming: Known Dynamics and Rewards

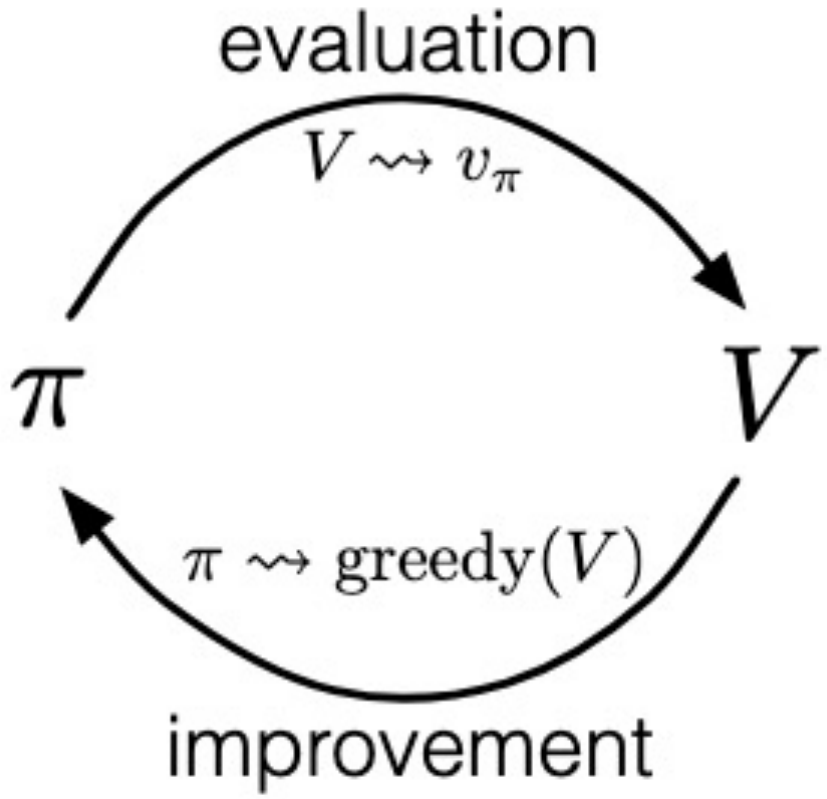
- **Know:**
  - $p(s_{t+1}|s_t, a_t)$
  - $r(s, a)$
- **Goal:** Learn optimal policy
  - $\pi^\star(\cdot)$



Dynamic Programming: Known Dynamics and Rewards

- **Know:**
  - $p(s_{t+1}|s_t, a_t)$
  - $r(s, a)$

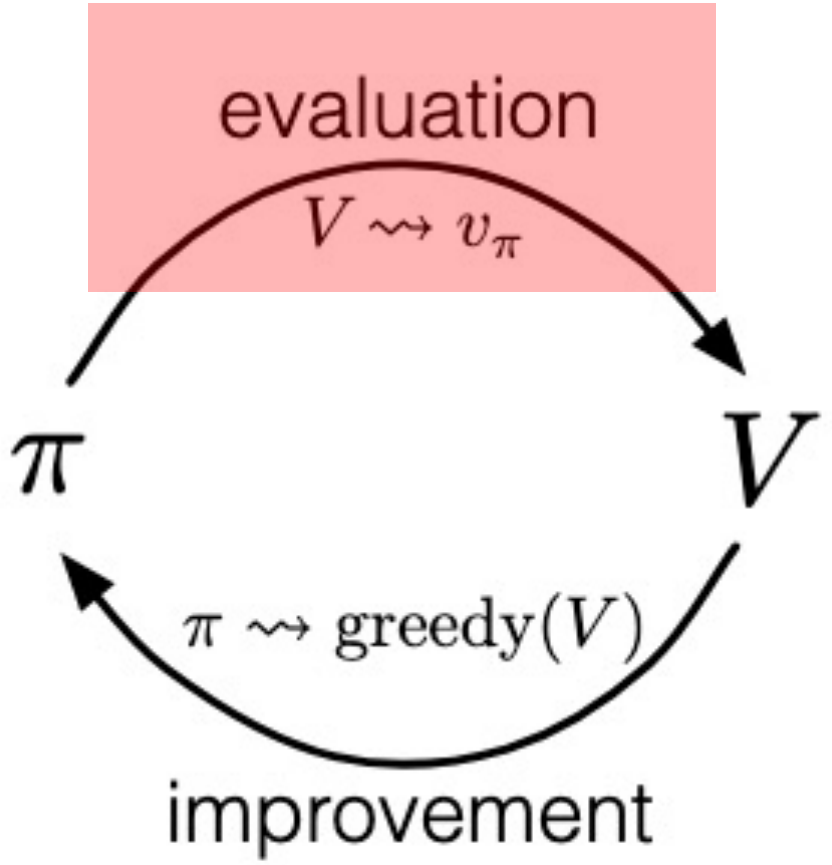
- **Goal:** Learn optimal policy
  - $\pi^\star(\cdot)$



Dynamic Programming: Known Dynamics and Rewards

- **Know:**
  - $p(s_{t+1}|s_t, a_t)$
  - $r(s, a)$

- **Goal:** Learn optimal policy
  - $\pi^\star(\cdot)$



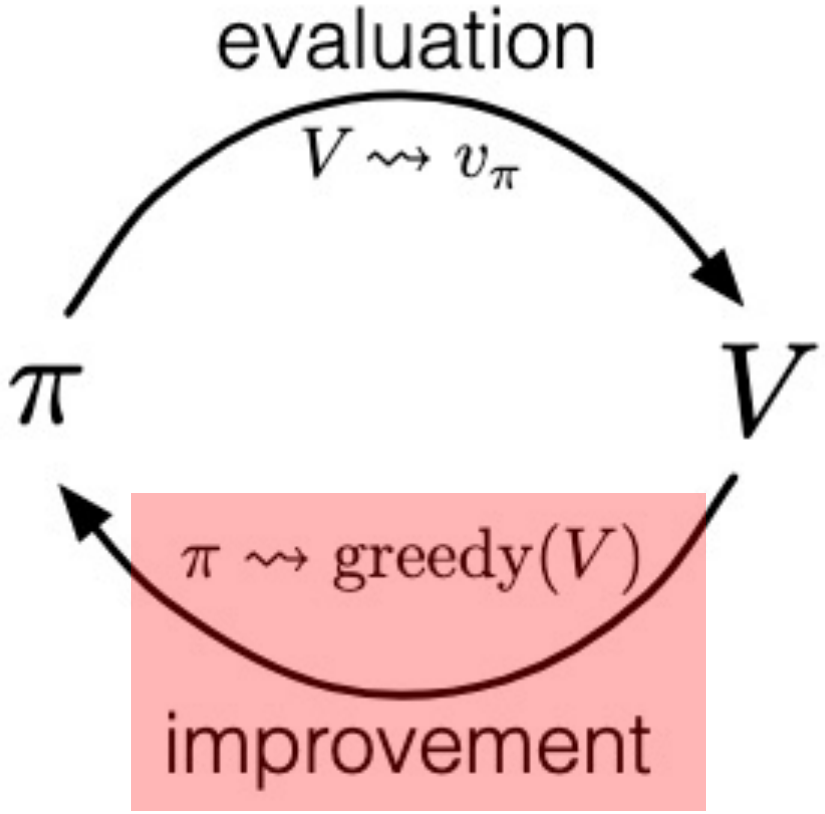
Dynamic Programming: Known Dynamics and Rewards

- **Know:**
  - $p(s_{t+1}|s_t, a_t)$
  - $r(s, a)$

- **Goal:** Learn optimal policy

- **Policy Improvement:**

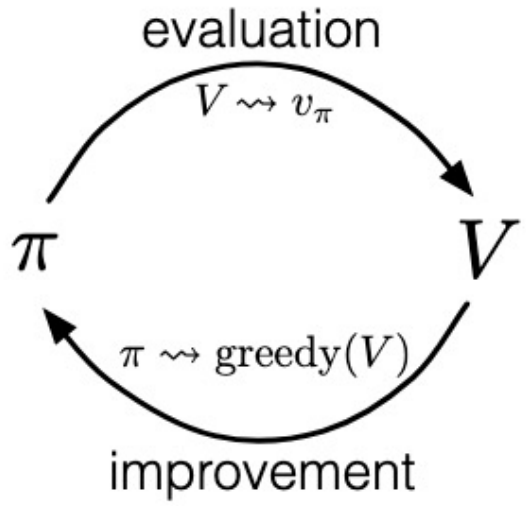
$$\pi(s) = \underset{a}{\text{argmax}}\, r(s, a) + \sum_{s' \in S} p(s'|s, a)V(s')$$



Dynamic Programming: Known Dynamics and Rewards

Stockholms universitet

- **Know:**
  - $p(s_{t+1}, R_{t+1}|s_t, a_t)$
  - $r(s, a)$
- **Goal:** Learn optimal policy
  - $\pi^\star(\cdot)$



- **Policy Evaluation:**

Infinite iterations

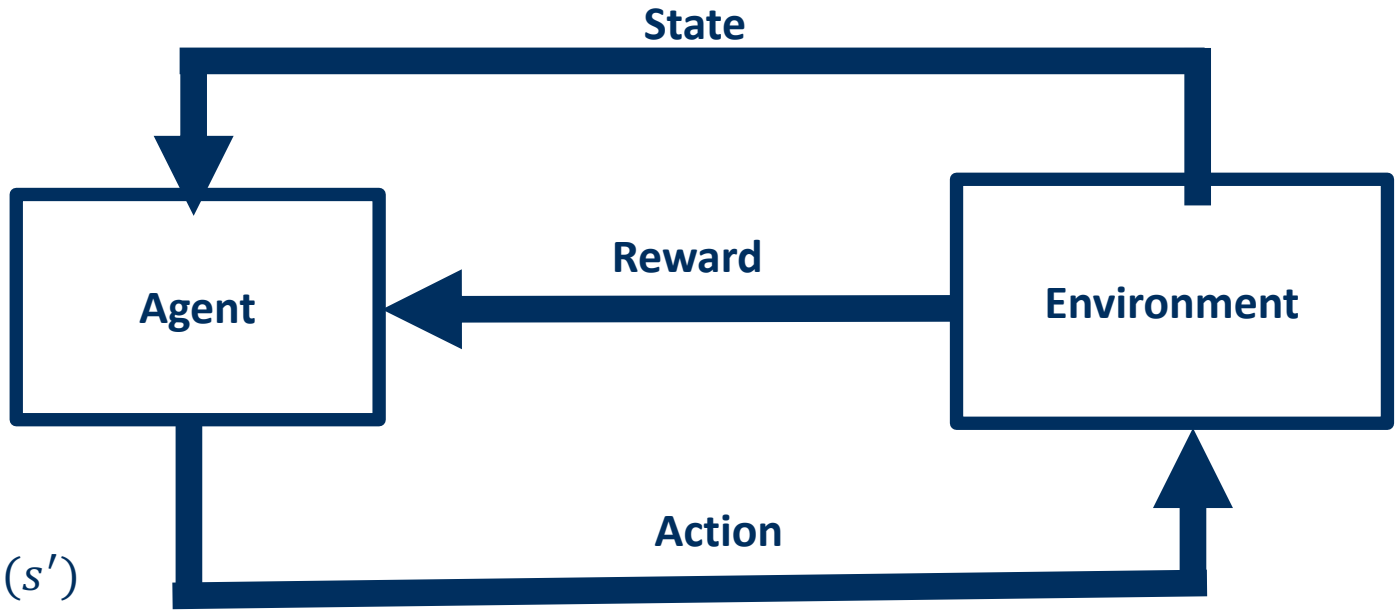
```
Initialize π_0(·)

for t=1,2,3,...
     V^{π_t}  = Policy_Evaluation(π_t)
     π_t(·) = Policy_Improvement(V^{π_t})
```

- **Converges to V(·):**

$$\lim_{i \to \infty} \pi_t(\cdot) = \pi^\star(\cdot)$$

Dynamic Programming: Value Iteration

- **Know:**
  - $p(s_{t+1}|s_t, a_t)$
  - $r(s, a)$
- **Goal:** Learn optimal policy
  - $\pi^\star(\cdot)$
- **Optimal Bellmann Equation:**
  - $V^*(s) = \max_a r(s, a) + \sum_{s'} p(s'|s, a)V^*(s')$



Dynamic Programming: Value Iteration

Stockholms universitet

- **Know:**
  - $p(s_{t+1}, R_{t+1}|s_t, a_t)$
  - $r(s, a)$
- **Goal:** Learn optimal policy
  - $\pi^\star(\cdot)$
- **Optimal Bellmann Equation:**
  - $V^*(s) = \max_a r(s,a) + \sum_{s'} p(s'|s,a)V^*(s')$

Value Iteration:

Initialize: $\hat{V}_0(s)$ for all s

for $i = 1,2,3 \ldots$

for $s \in S$

$$\hat{V}_{i+1}(s) = \max_a r(s,a) + \sum_{s'} p(s'|s,a)\,\hat{V}_i(s')$$

Converges to $V^*(\cdot)$:

$$\lim_{i\to\infty} \hat{V}_i(\cdot) = V^*(\cdot)$$

Optimal Policy:

$$\pi(s) = arg\max_a r(s,a) + \sum_{s'} p(s'|s,a)\,\hat{V}_{end}(s')$$

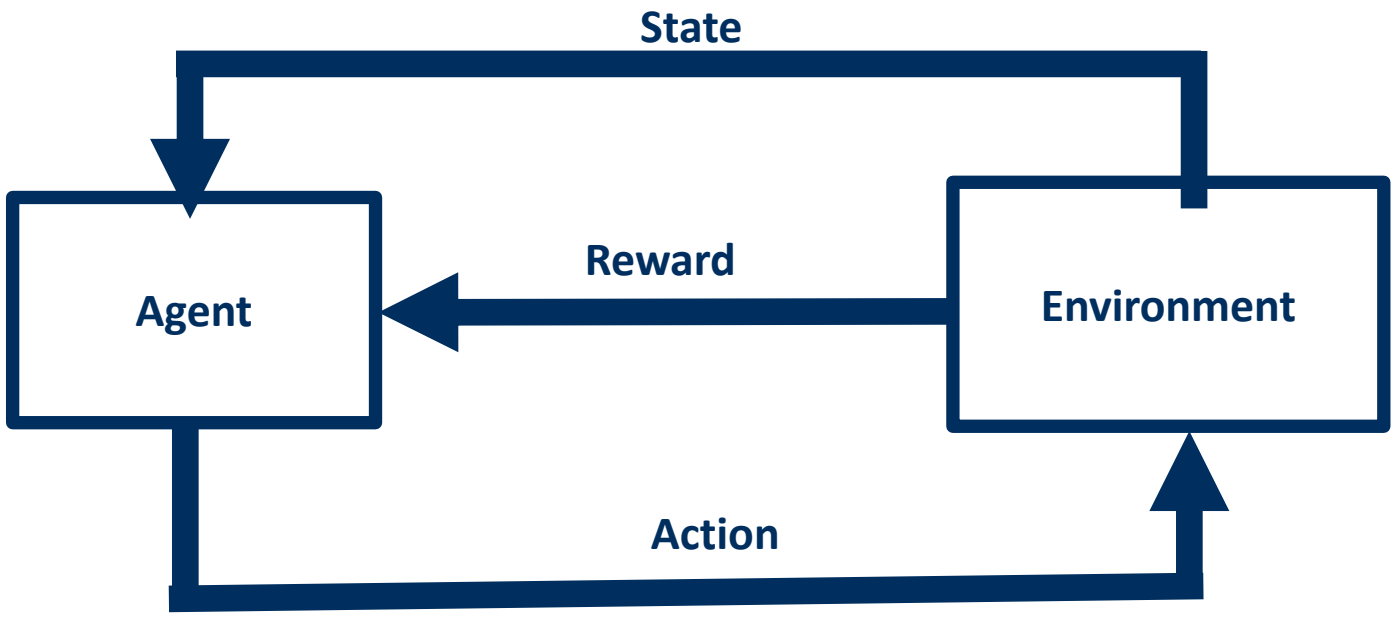
Overview

- Markov Decicion Process
- **Tabluar Methods**
  - Value Evaluation
  - Policy Improvement
  - **Q-learning**
- Approximtion Methods
  - Deep Q-learning
  - Policy Gradient methods
- Conclusions

Q-Learning

- **Goal:**
  - Learn optimal policy $\pi^{\star}(\cdot)$
  - Only have experience (don't know dynamics)
- **Action-value:**
  - $Q^{\pi}(s,a) = \boldsymbol{E}\left[\sum_{t=0}^{\infty} R_t \mid s_0 = s, a_0 = a\right]$
- **Optimal Bellmann Equation:**

$$Q^{*}(s,a) = \quad r(s,a) + \max_{a'} \sum_{s'} p(s'|s,a) Q^{*}(s',a')$$



Q-Learning

Stockholms universitet

Q Learning:

Initialize $\hat{Q}\ (s,a)$ for all s and a
Choose initial state s_0

for t= 0,1,2,3 …
    Coose action a_i from $\hat{Q}$ (e.g. ϵ Greedy)
    Take action a_t and observe R_{t+1} and s_{t+1}
    $\hat{Q}(s_i, a_i) = \hat{Q}(s_i, a_i) + \alpha[R_{t+1} + \max_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t)]$

Converges to $Q^*(\cdot)$:

$$\lim_{t\to\infty} \hat{Q}\ (\cdot) = Q^*(\cdot)$$

Optimal Policy:

$$\pi(s) = arg\max_a\ \hat{Q}(s,a)$$

Q-Learning

Stockholms universitet

- Off-policy Method – can use any exploration policy
- Learning rate
- Optimality

- Other algorithms -almost all are value-based, e.g.
  - Monte-Carlo Learning
  - SARSA
  - n-steep bootstrapping

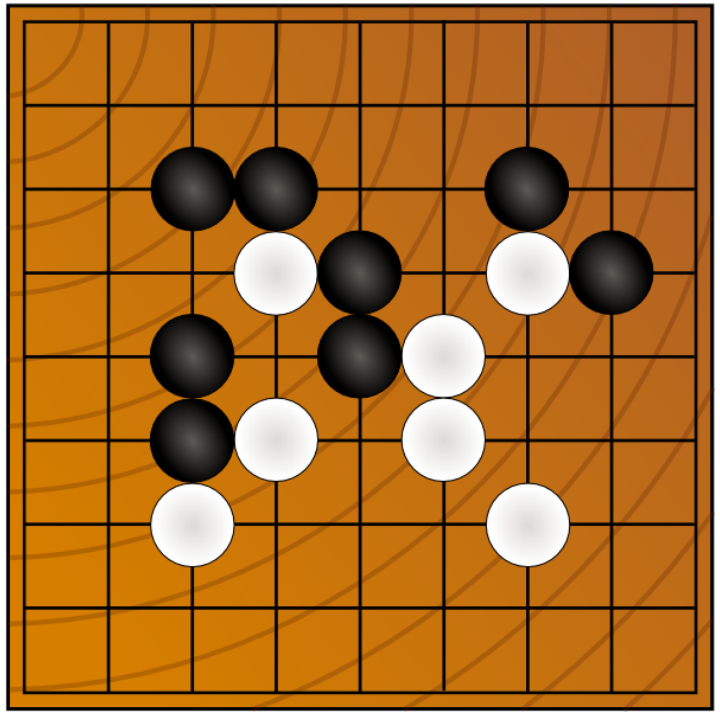
Overview

- Markov Decicion Process
- **Tabluar Methods**
  - Value Evaluation
  - Policy Improvement
  - Q-learning
- **Approximtion Methods**
  - Deep Q-learning
  - Policy Gradient methods
- Conclusions

Deep RL: Generalization

- State-spaces often huge

$(256^{100\times300})^3$ states



3^{361} states

Deep RL: Generalization

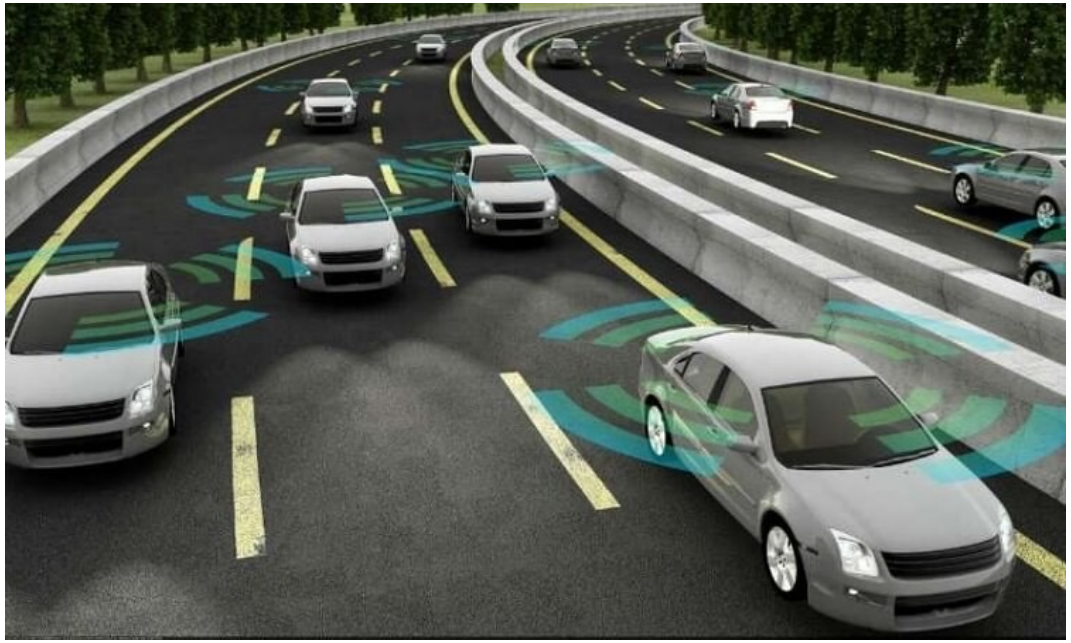
Stockholms universitet

- State-spaces grows exponentially with features

			Number of features	Number of states
Blood-pressure	High	Low	1	2
Gender	Man	Woman	2	$2^2 = 4$
Age	Old	Young	3	$2^3 = 8$
Heart-rate	High	High	4	$2^4 = 16$
-	-	-	5	$2^5 = 32$
⋮	⋮	⋮	⋮	⋮
-	-	-	20	$2^{20} = 1048576$

Deep RL: Generalization

- State/action-spaces grows exponentially with agents



Deep RL: Generalization

Deep Q-learning

Approximate Q

Policy Optimization

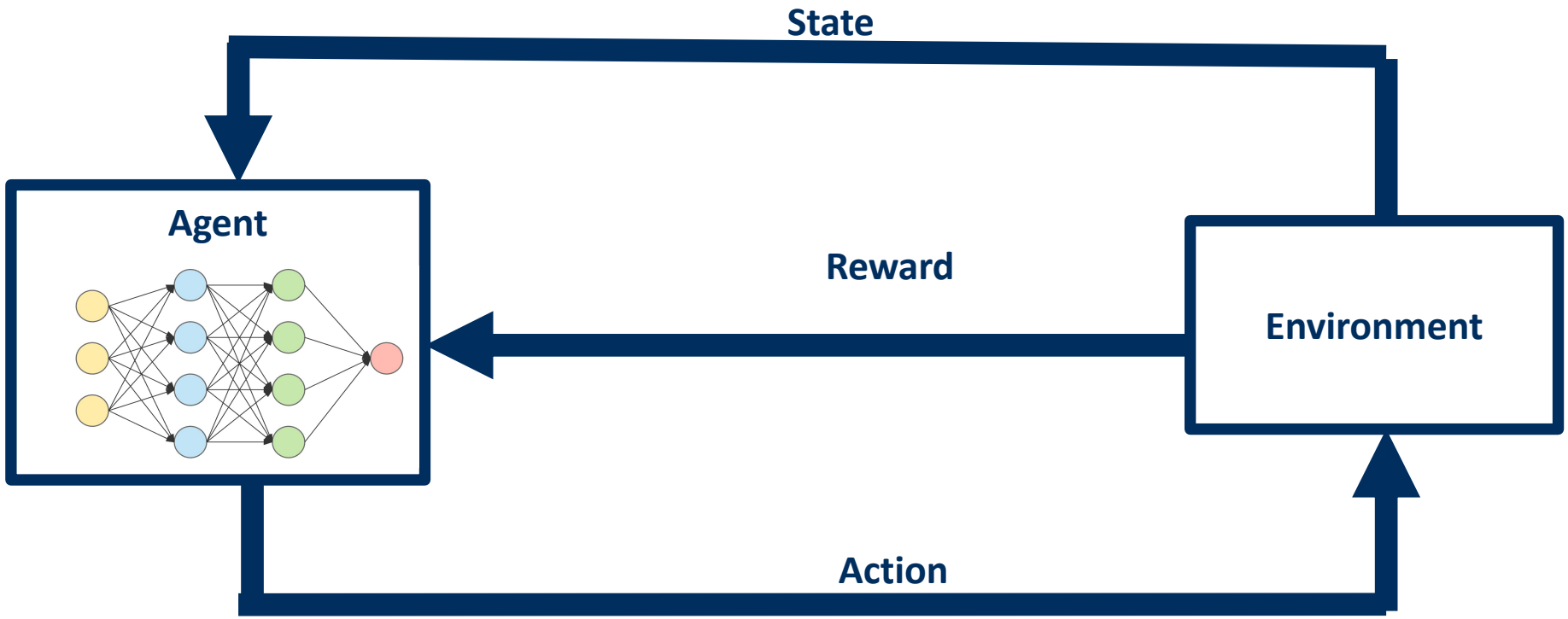
Approximate the policy

Overview

- Markov Decicion Process
- **Tabluar Methods**
  - Value Evaluation
  - Policy Improvement
  - Q-learning
- **Approximtion Methods**
  - **Deep Q-learning**
  - Policy Gradient methods
- Conclusions

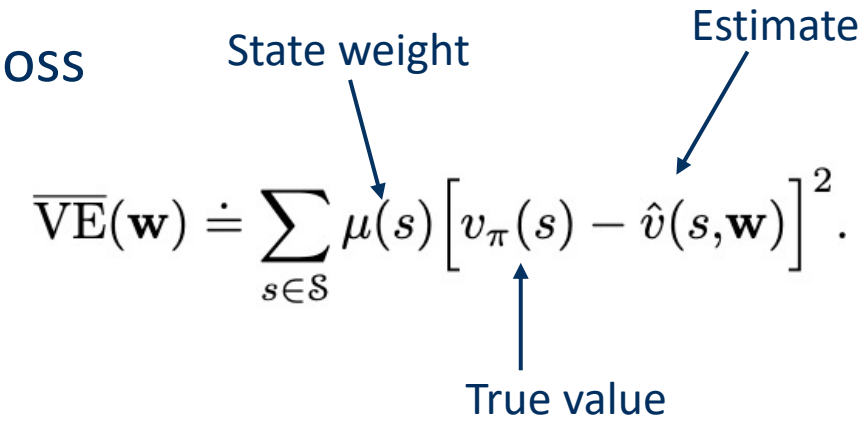
Deep RL: Generalization

- Cannot learn $Q^*(s, a)$ by trying all state action pairs!
- **Solution:** Learn approximate $Q^*(s, a)$ by a deep neural network
- Train $Q_w(s, a) \approx Q^*(s, a)$ where w are the weights of a neural networ



Deep RL: Policy Evaluation

- Minimize the loss



- Gradient descent

$$\begin{aligned} \mathbf{w}_{t+1} &\doteq \mathbf{w}_t - \frac{1}{2}\alpha \nabla \left[v_\pi(S_t) - \hat{v}(S_t,\mathbf{w}_t) \right]^2 \\ &= \mathbf{w}_t + \alpha \left[v_\pi(S_t) - \hat{v}(S_t,\mathbf{w}_t) \right] \nabla \hat{v}(S_t,\mathbf{w}_t), \end{aligned}$$

Not known

Deep RL: Policy Evaluation

Stockholms universitet

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})\big]\nabla\hat{v}(S,\mathbf{w})$
    $S \leftarrow S'$
  until S is terminal

Estimate of the true value

Deep RL: Q-learning

Stockholms universitet

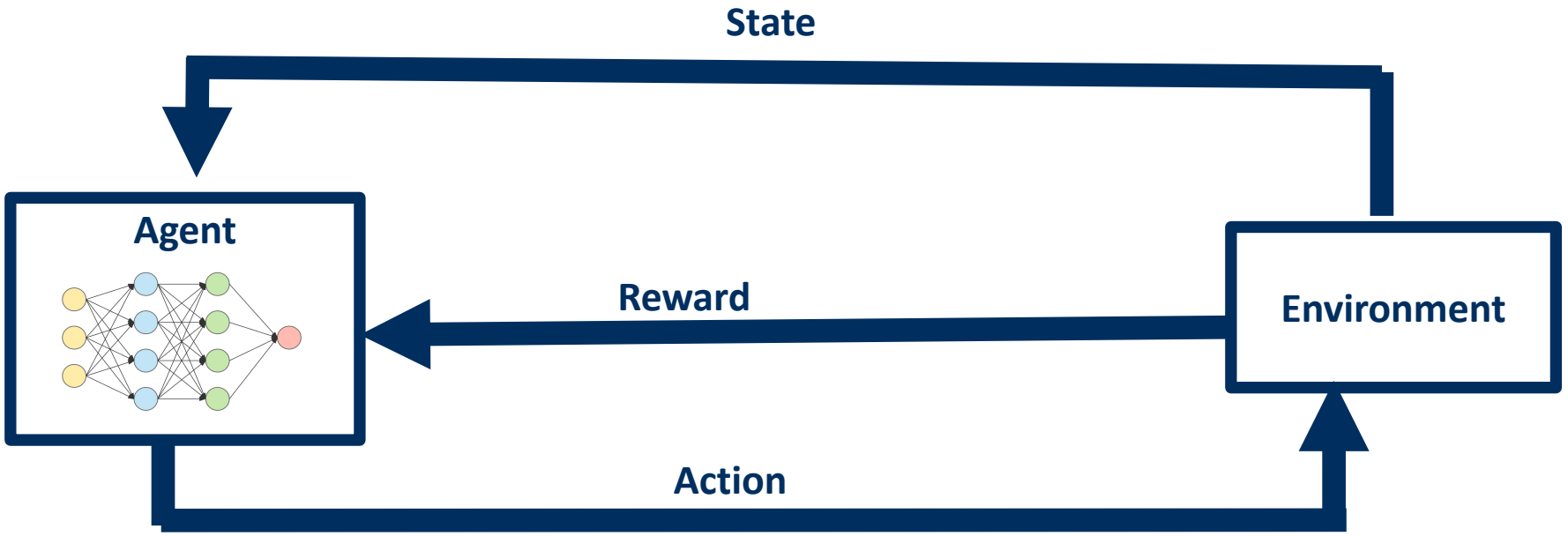
- Can do similar for policy optimization, math will be more complex
- Off-policy methods
- Can deal with continues state-space
- Usually better than tabular methods when assumptions are relaxed

Overview

- Markov Decicion Process
- **Tabluar Methods**
  - Value Evaluation
  - Policy Improvement
  - Q-learning
- **Approximtion Methods**
  - Deep Q-learning
  - **Policy Gradient methods**
- Conclusions

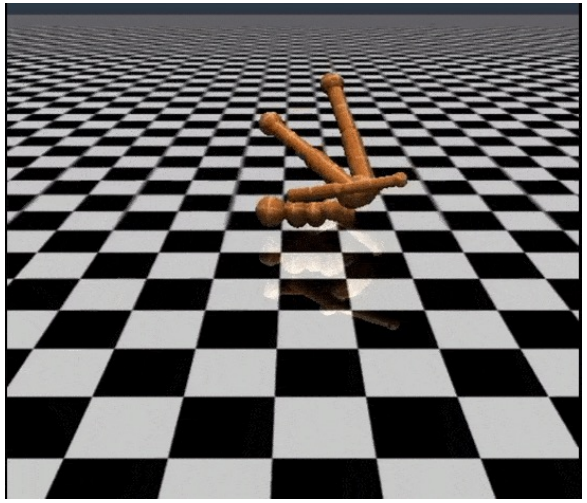
Deep RL: Generalization

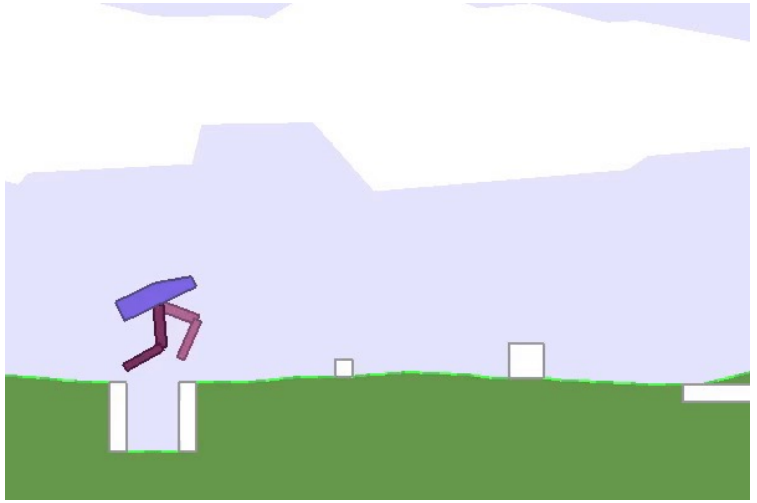
- Directly learn policy (without value function)
- **Solution:** Learn a policy network: $\pi(a|s,\boldsymbol{\theta}) = \Pr\{A_t = a \mid S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$
- Gradient descent: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\widehat{\nabla J(\boldsymbol{\theta}_t)},$

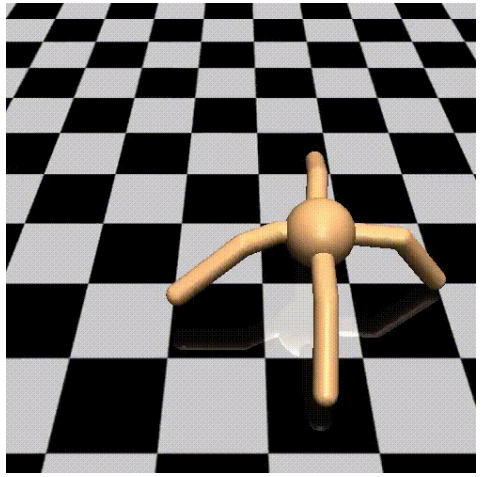


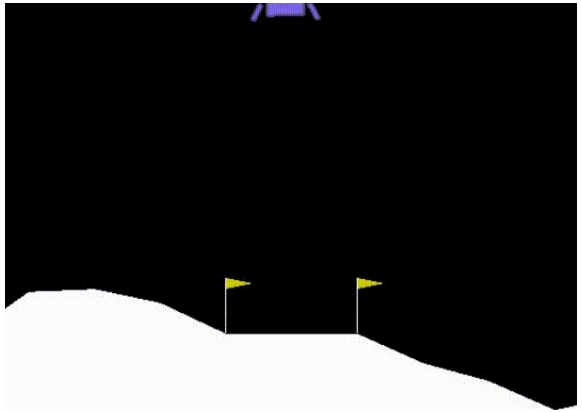
Stockholms universitet

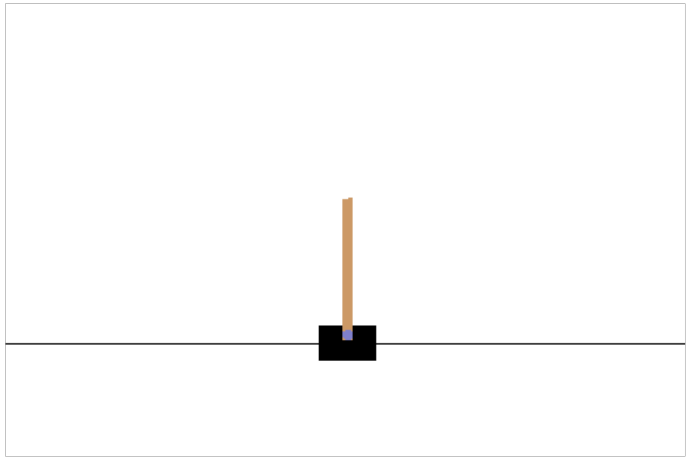
OpenAI GYM







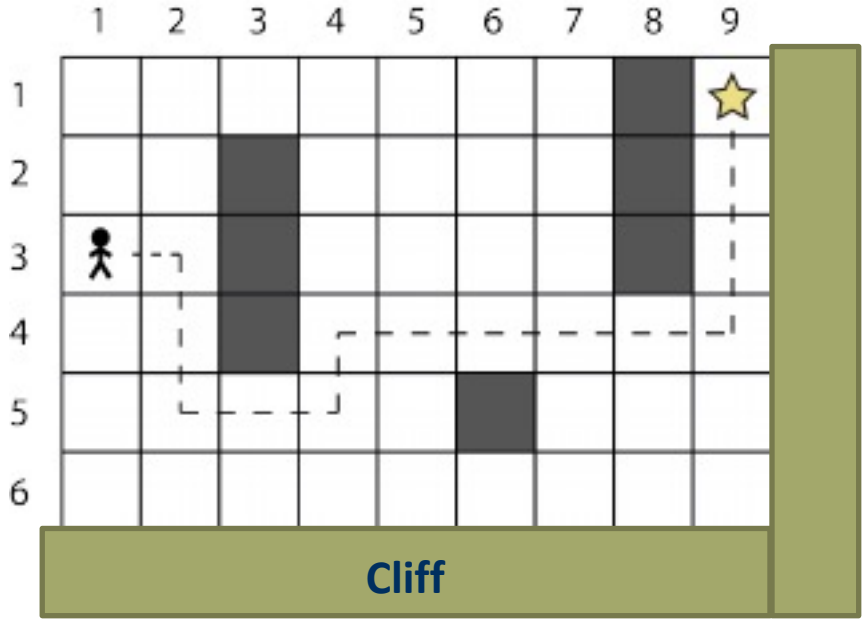




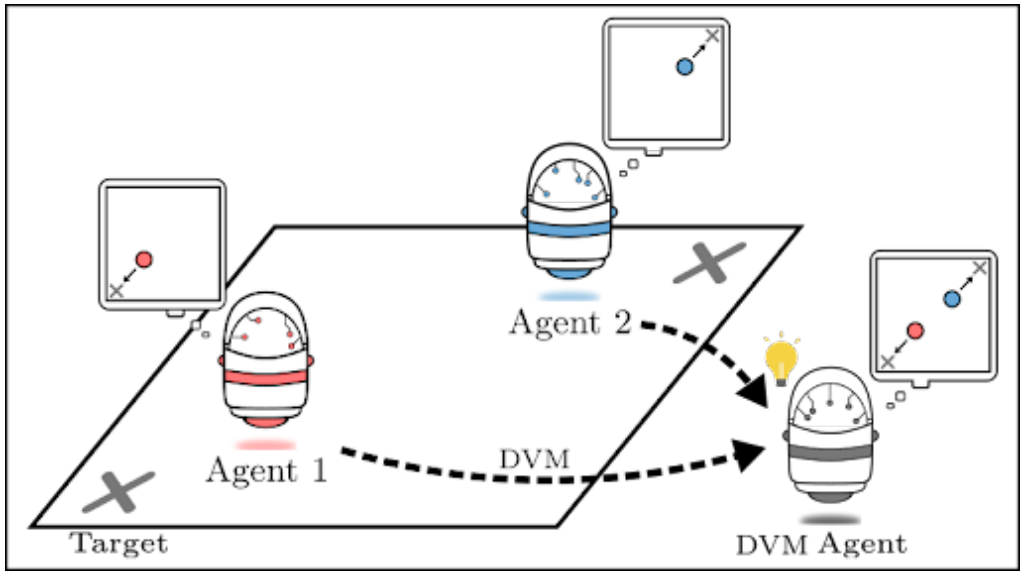
Overview

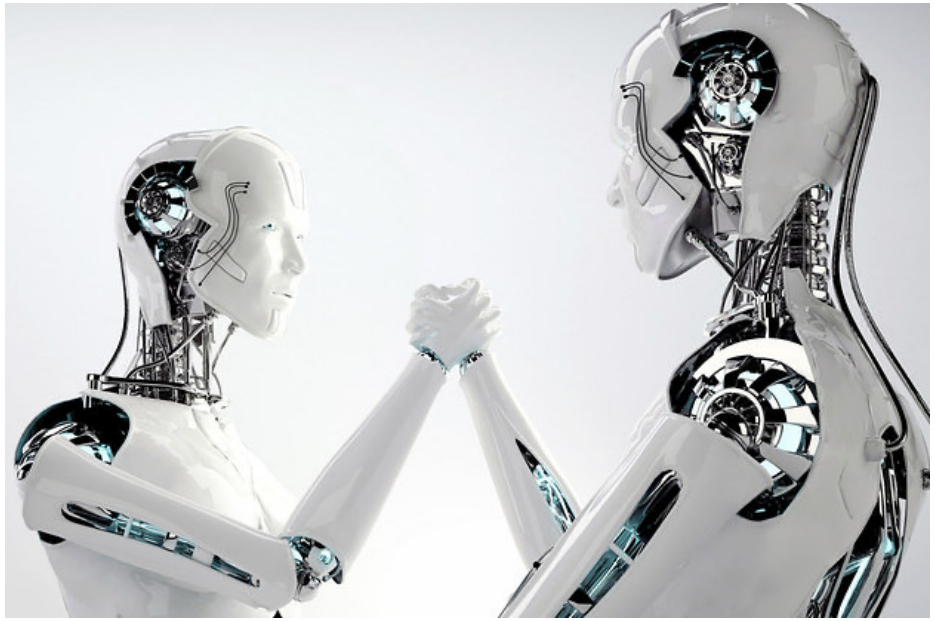
- Markov Decicion Process
- Tabluar Methods
  - Value Evaluation
  - Policy Improvement
  - Q-learning
- Approximtion Methods
  - Deep Q-learning
  - Policy Gradient methods
- Conclusions

Current trends - safe exploration in RL

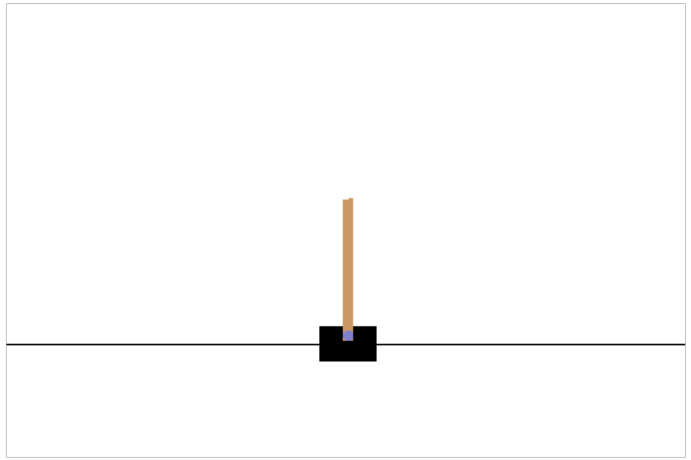


Current trends - Multi-Agent Reinforcement Learning





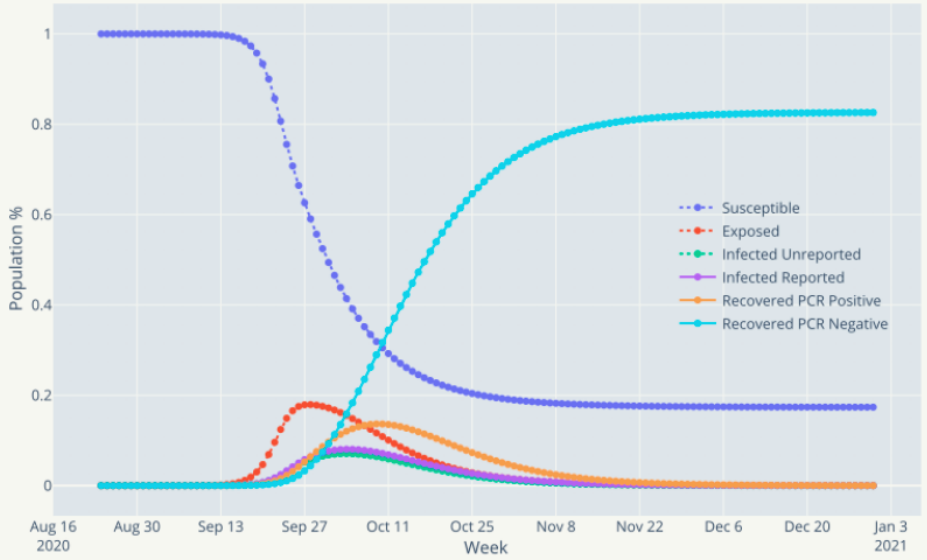
Current trends - Data-Driven CPSs and IoT Operations

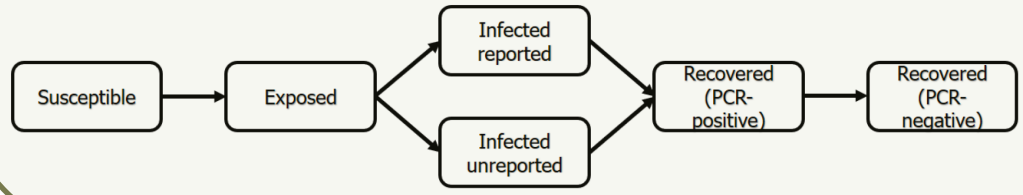


Current trends - Public Health Policy for Covid-19

Stockholms universitet

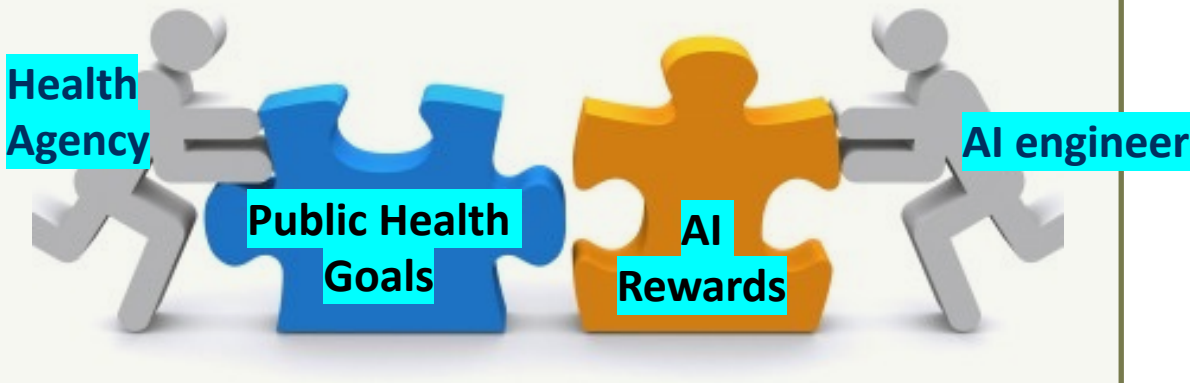
States/Dynamics



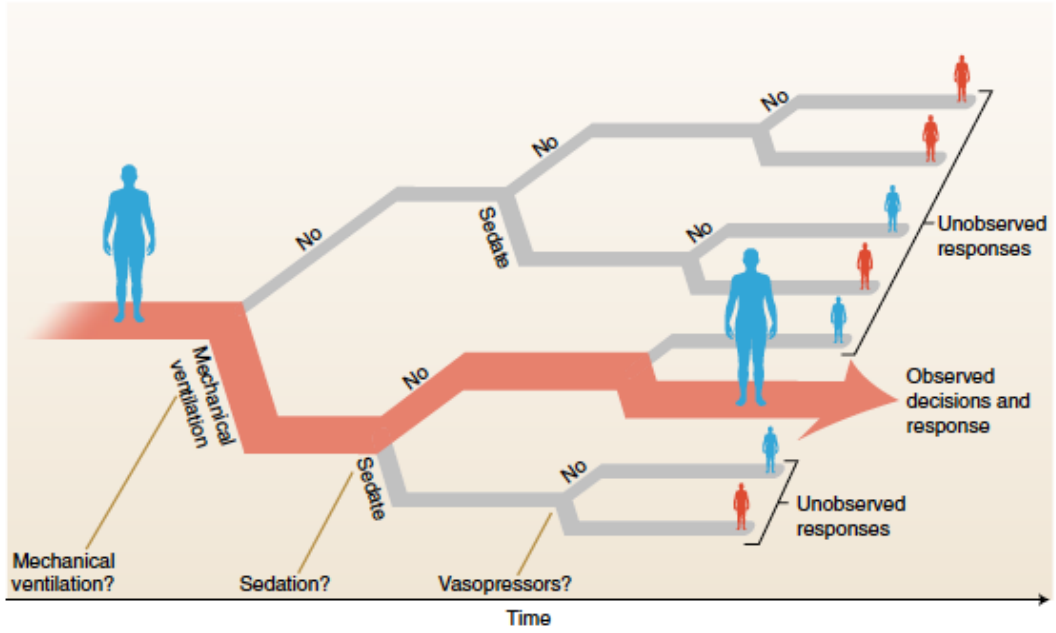


Actions

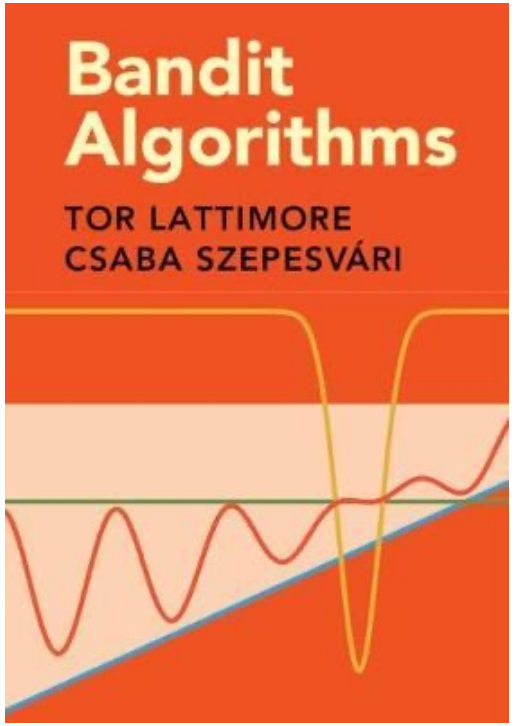
Reward



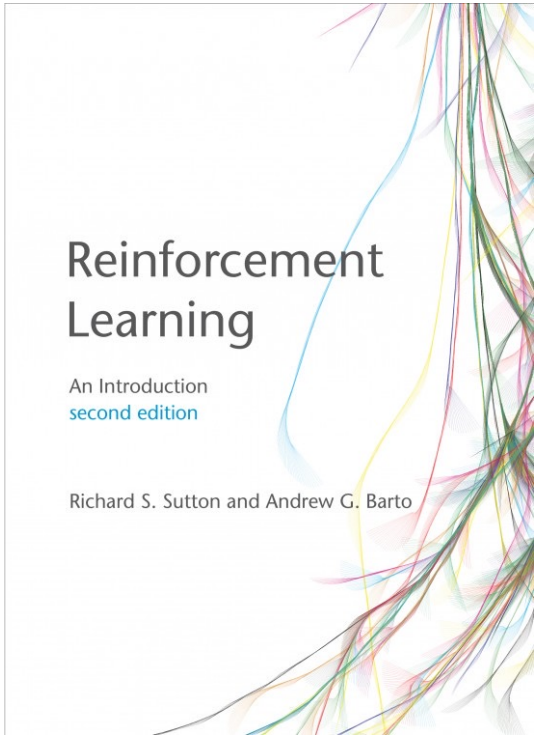
Current trends - Treatment Selection – Personalized Medicine

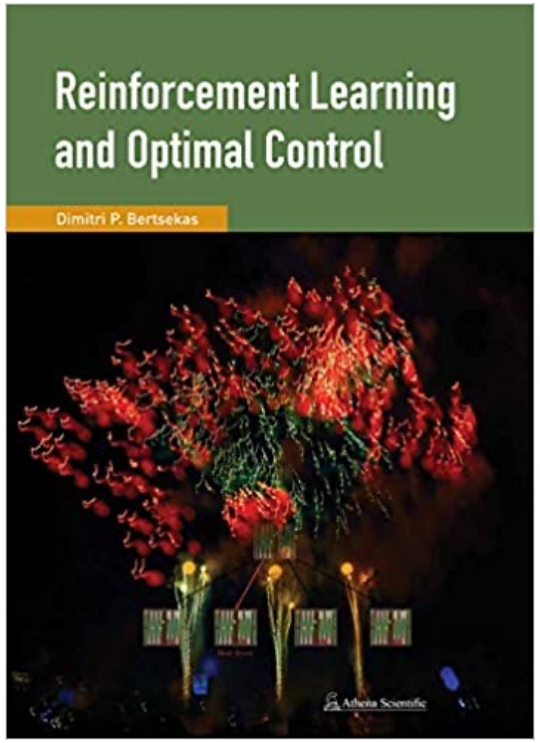


Literature


Bandit Algorithms
TOR LATTIMORE
CSABA SZEPESVÁRI


Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems


Reinforcement Learning
An Introduction
second edition
Richard S. Sutton and Andrew G. Barto


Reinforcement Learning and Optimal Control
Dimitri P. Bertsekas