POLITECNICO
MILANO 1863

Reinforcement Learning

Evolution of Policy Search methods

3 November 2021

Marcello Restelli

marcello.restelli@polimi.it

Who am I?

- Associate Professor at Politenico di Milano
- Courses
  - Machine Learning
  - Information Retrieval and Data Mining
  - Robotics
- Research interests
  - Reinforcement learning
  - Multi-agent learning
  - Online learning
- Industrial collaborations
  - E-commerce
  - Finance
  - Automotive
  - Industry 4.0
- Co-founder of ML cube s.r.l.

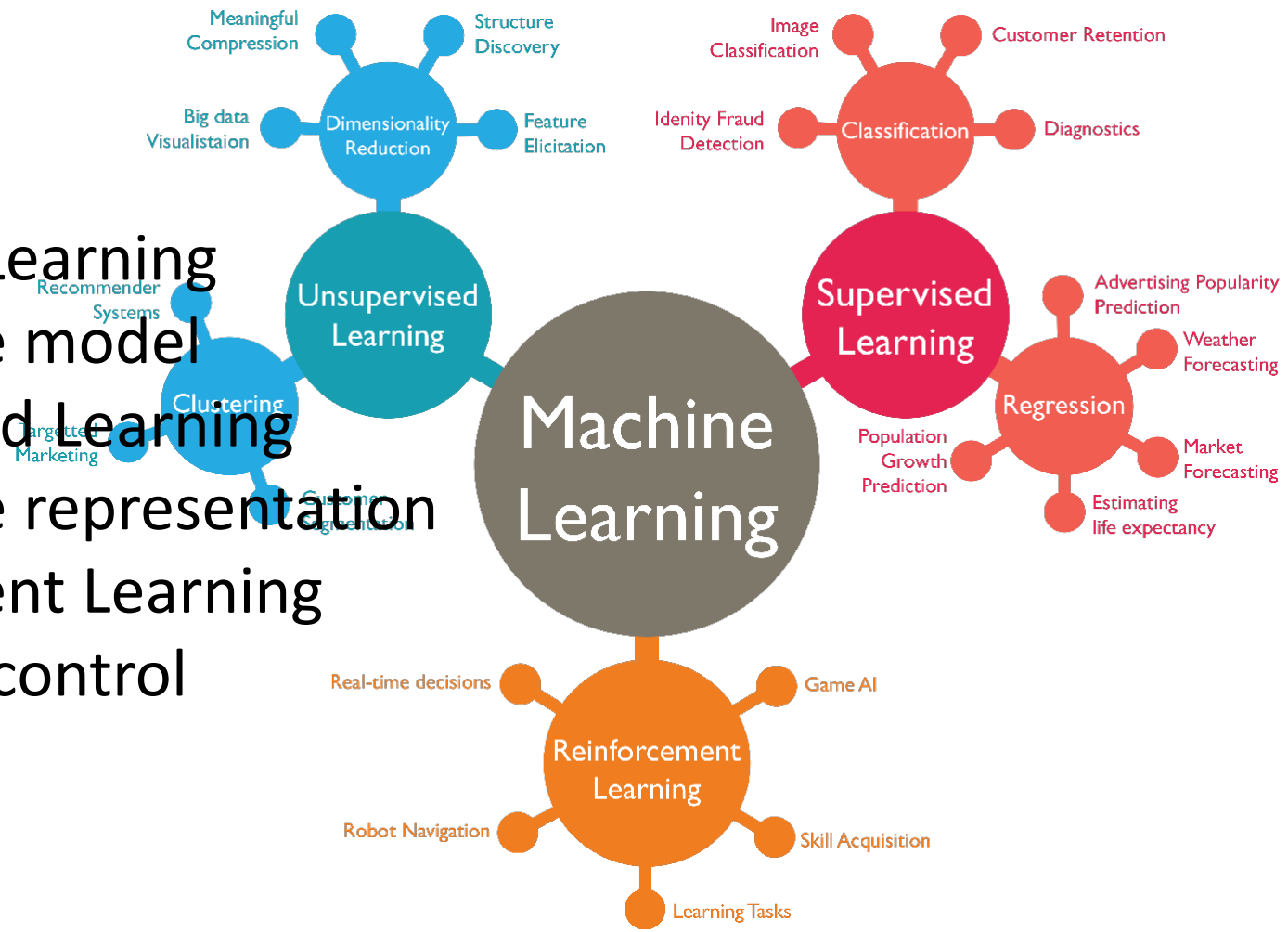
Outline

1. Overview of Reinforcement Learning
2. Policy Search methods
3. Applications

Overview of RL

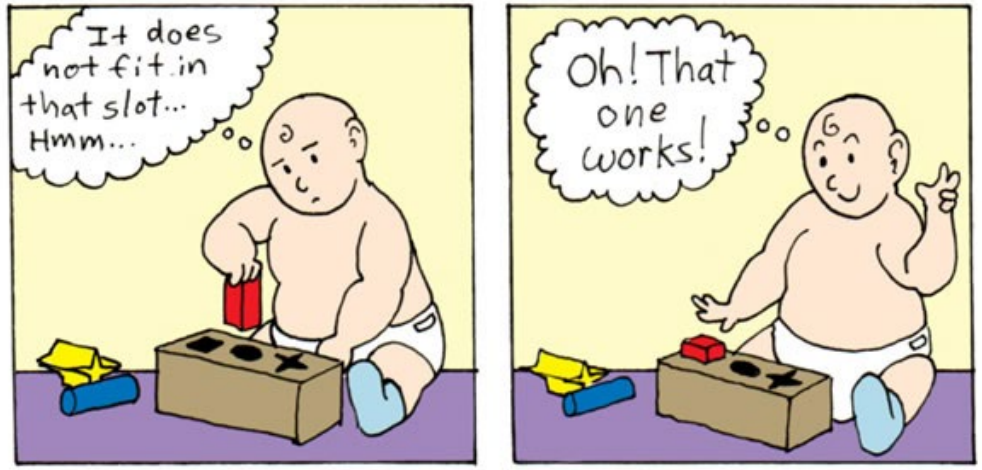
Machine Learning

- Supervised Learning
  - Learn the model
- Unsupervised Learning
  - Learn the representation
- Reinforcement Learning
  - Learn to control



What is Reinforcement Lerning (RL)?

- A set of techniques for solving **Sequential Decision Making** problems
- RL is learning form **interaction**, by **trial-and-error**



Reinforcement Learning

new observations

reward

action

But who’s counting?

But who's counting?

- First game
  - Best possible value : 75421
  - Value following the optimal policy: 75142
- Second game
  - Best possible value: 76530
  - Value following the optimal policy: 75630

History of Reinforcement Learning

Pavlov (1903)
Classical conditioning

Thorndike (1905)
Law of effect

Minsky (1961)
Credit–assignment problem

Psychology, Trial and error

Watkins (1989)
Q–learning

Littman (1994)
minimax–Q

Samuel (1956)
Checkers

Reinforcement Learning

Optimal Control

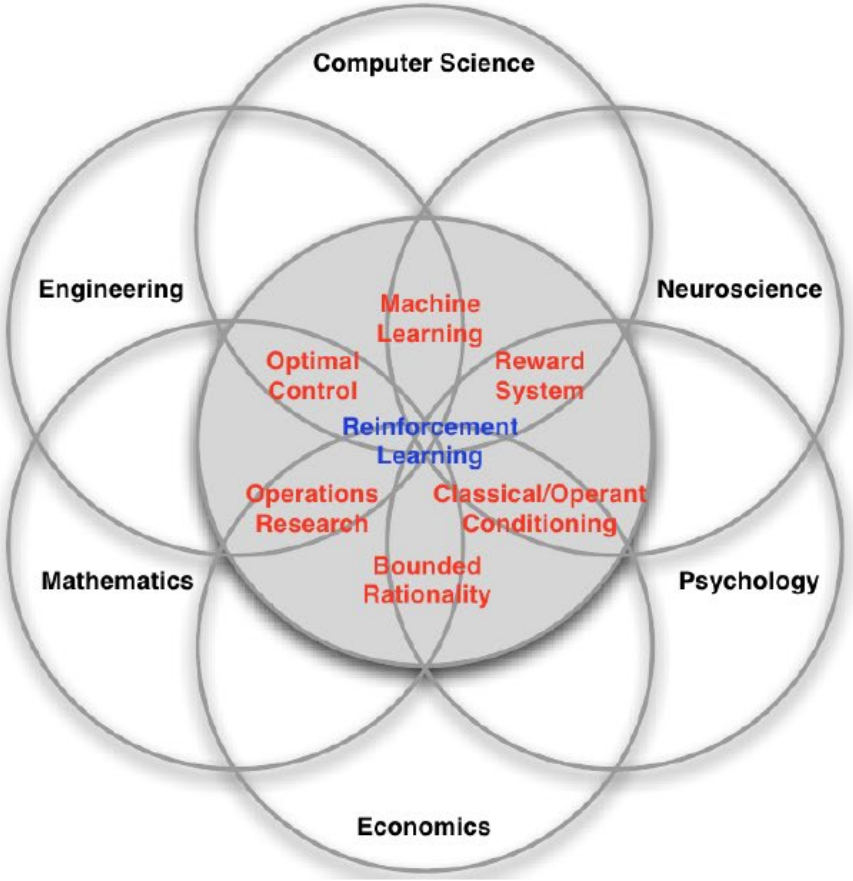
Sutton & Barto (1984)
Temporal Difference

Tesauro (1992)
TD–Gammon

Bellman (1957)
Dynamic Programming

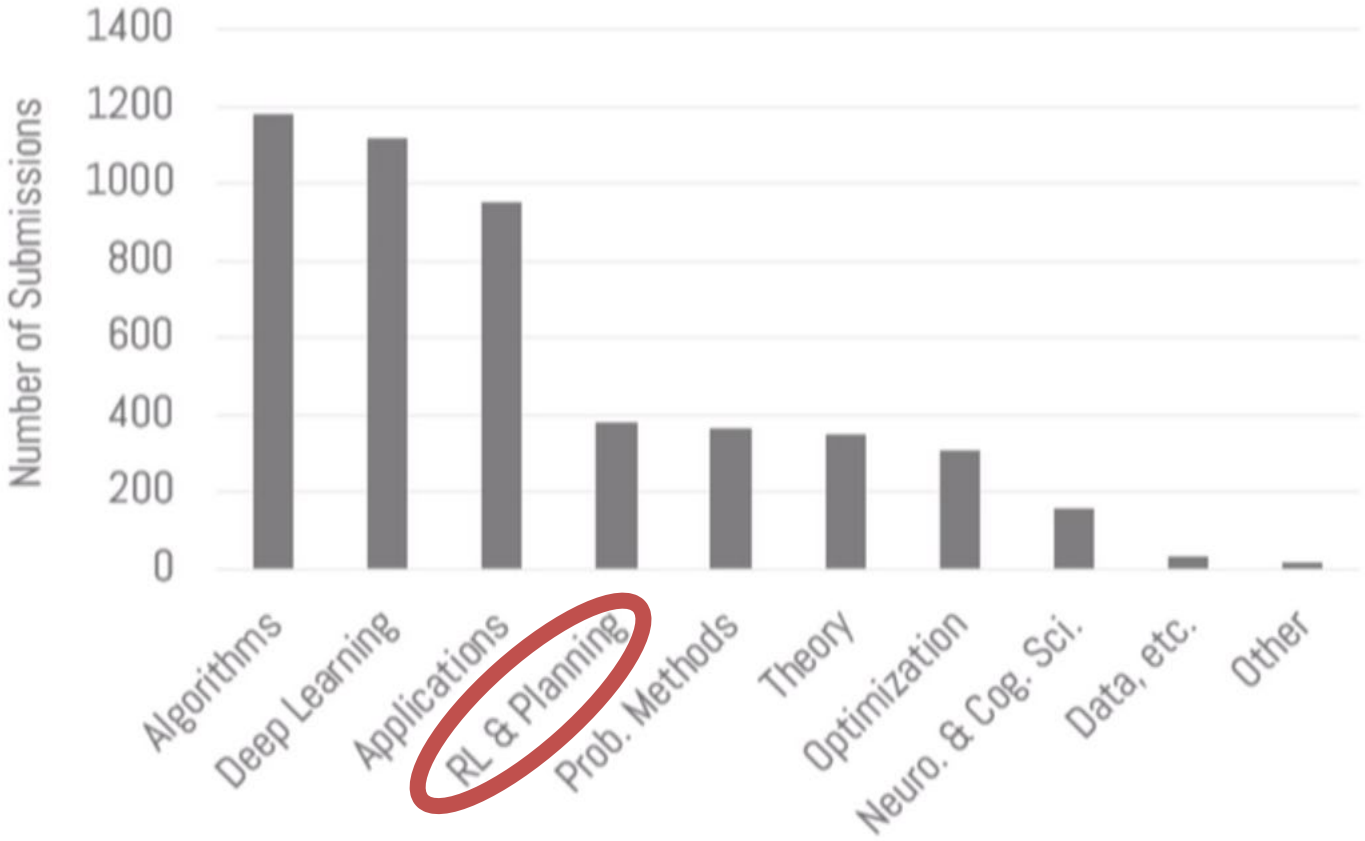
Howard (1960)
Policy Iteration

Where RL comes from?

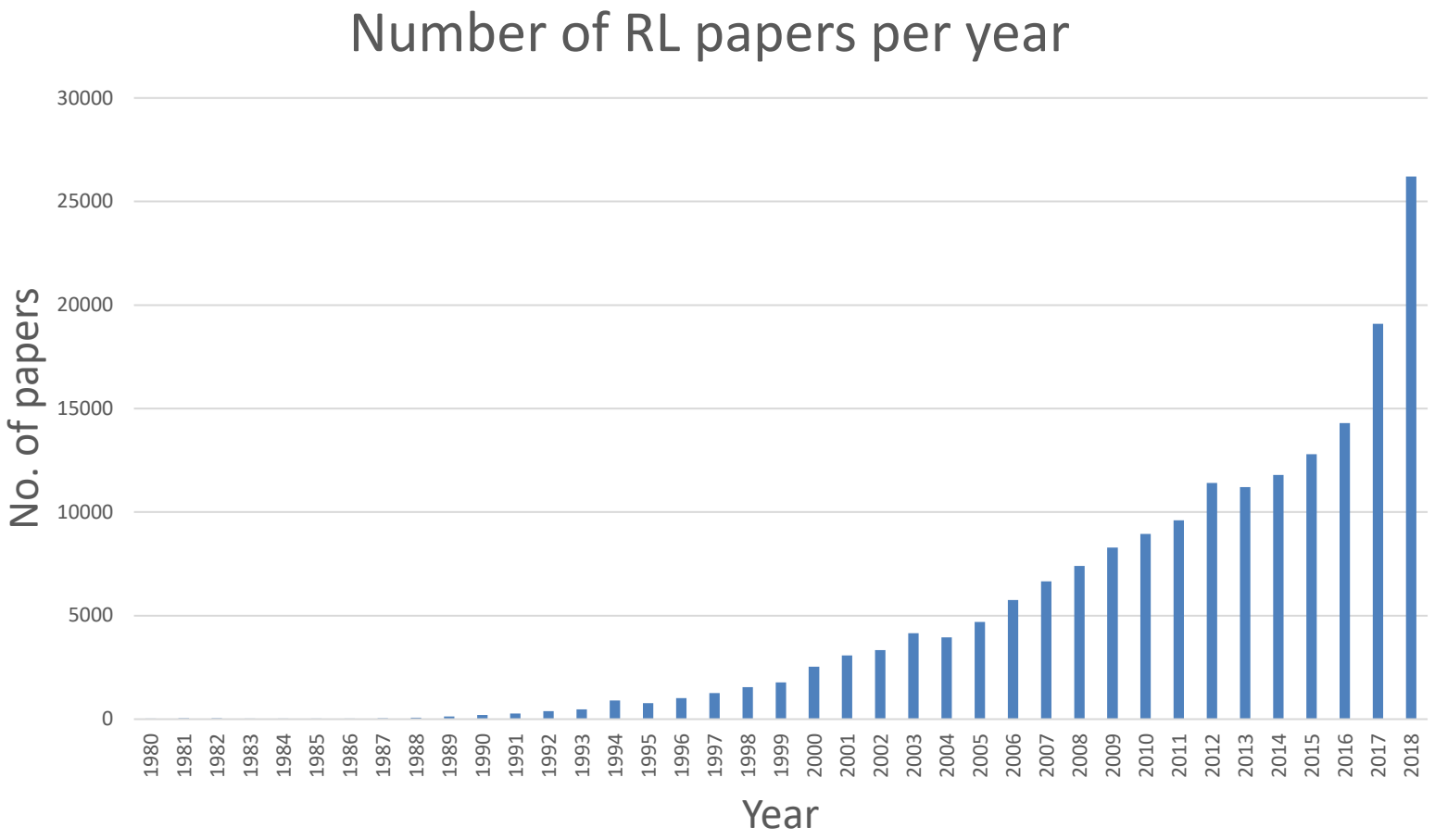


How Large is RL?

NeurIPS 2018 Submissions per area



RL Research Trend



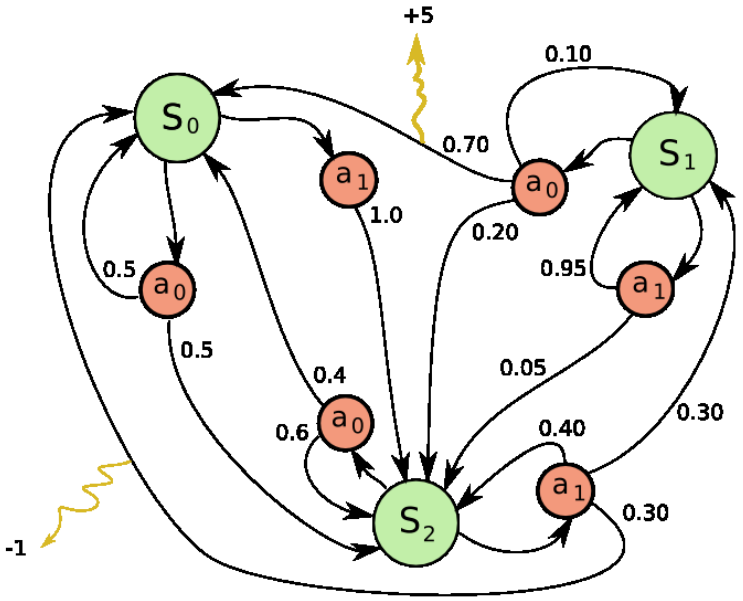
What is the Trend for RL?

Gartner, 2019

Sequential Decision Making

Markov Decision Process (MDP)

- State space S
- Action space A
- Transition model $P(s'|s,a)$
- Reward function $R(s,a,s')$
- Discount factor γ
- Initial state distribution $\mu(s)$



Markov Assumption

The future is **independent** of the past given the present

- **Definition**: a stochastic process X_t is said to be **Markovian** if and only if

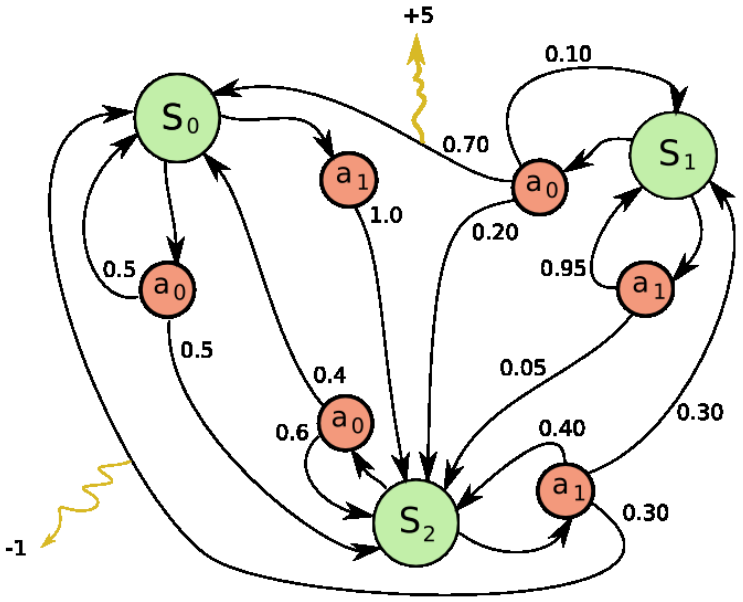
$$P(X_{t+1} = j | X_t = i, X_{t-1} = k_{t-1}, \dots, X_0 = k_0) = P(X_{t+1} = j | X_t = i)$$

  - The state **captures all the information** from history
  - Once the state is known, the history may be **thrown away**
  - The state is a **sufficient statistic** for the future
  - The conditional probabilities are **tranistion probabilities**
  - If the probabilities are **stationary** (time invariant), we can write:

$$p_{ij} = P(X_{t+1} = j | X_t = i) = P(X_1 = j | X_0 = i)$$

Classification of Environments

- Discrete vs Continuous State
- Discrete vs Continuous Action
- Deterministic vs Stochastic
- Fully vs Partially Observable
- Stationary vs Nonstationary
- Single Agent vs Multi Agent

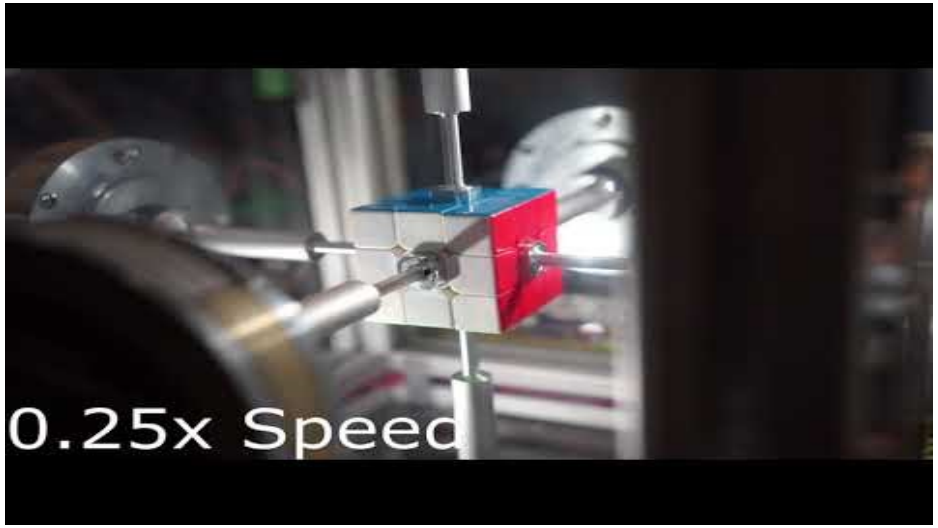


Example 1: Rubik's Cube

- Discrete vs Continuous State
- Discrete vs Continuous Action
- Deterministic vs Stochastic
- Fully vs Partially Observable
- Stationary vs Nonstationary
- Single Agent vs Multi Agent

Example 1: Rubik's Cube

- **Discrete** vs Continuous State
- **Discrete** vs Continuous Action
- **Deterministic** vs Stochastic
- **Fully** vs Partially Observable
- **Stationary** vs Nonstationary
- **Single Agent** vs Multi Agent

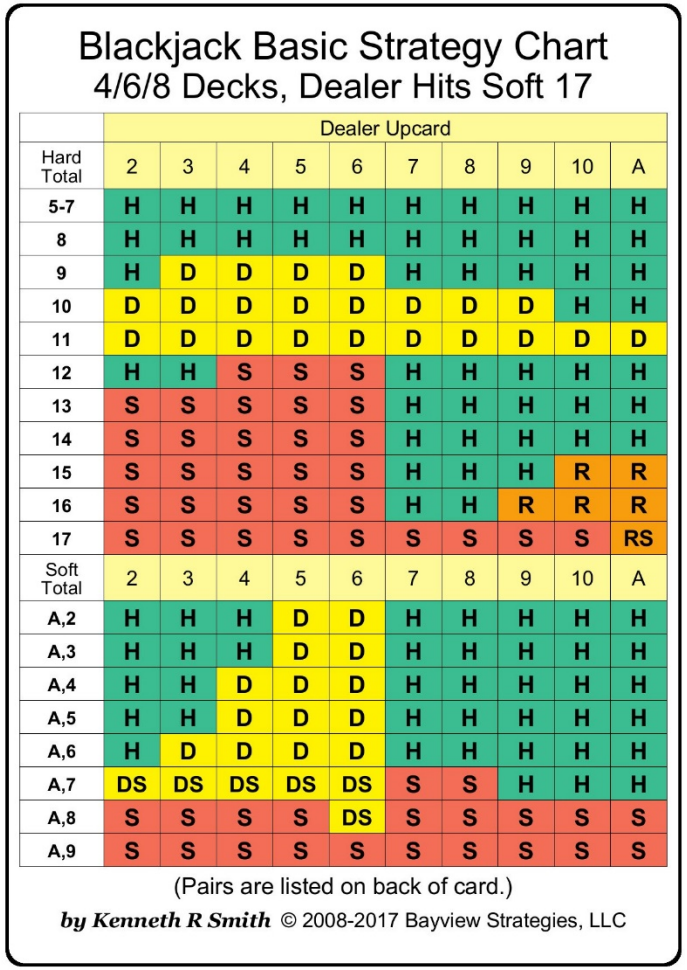


Example 2: Blackjack

- Discrete vs Continuous State
- Discrete vs Continuous Action
- Deterministic vs Stochastic
- Fully vs Partially Observable
- Stationary vs Nonstationary
- Single Agent vs Multi Agent

Example 2: Blackjack

- **Discrete** vs Continuous State
- **Discrete** vs Continuous Action
- Deterministic vs **Stochastic**
- **Fully** vs Partially Observable
- **Stationary** vs Nonstationary
- **Single Agent** vs Multi Agent



Blackjack Basic Strategy Chart
4/6/8 Decks, Dealer Hits Soft 17

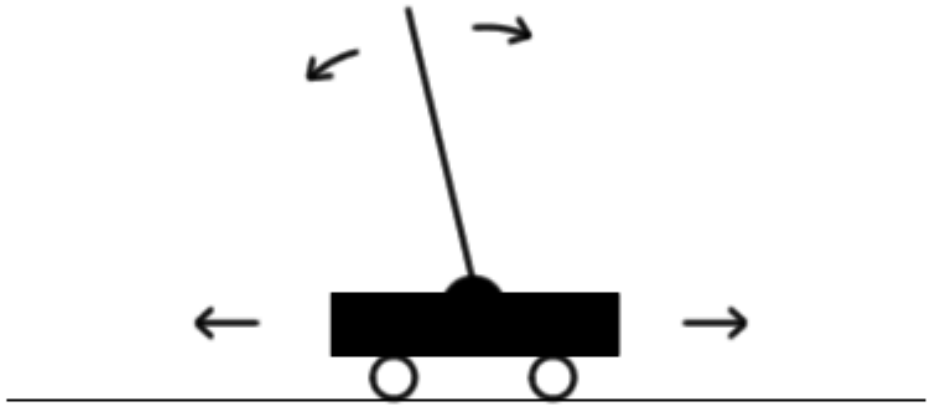
	Dealer Upcard									
Hard Total	2	3	4	5	6	7	8	9	10	A
5-7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	D	D	D	D	H	H	H	H	H
10	D	D	D	D	D	D	D	D	H	H
11	D	D	D	D	D	D	D	D	D	D
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	R	R
16	S	S	S	S	S	H	H	R	R	R
17	S	S	S	S	S	S	S	S	S	RS
Soft Total	2	3	4	5	6	7	8	9	10	A
A,2	H	H	H	D	D	H	H	H	H	H
A,3	H	H	H	D	D	H	H	H	H	H
A,4	H	H	D	D	D	H	H	H	H	H
A,5	H	H	D	D	D	H	H	H	H	H
A,6	H	D	D	D	D	H	H	H	H	H
A,7	DS	DS	DS	DS	DS	S	S	H	H	H
A,8	S	S	S	S	DS	S	S	S	S	S
A,9	S	S	S	S	S	S	S	S	S	S

(Pairs are listed on back of card.)

by Kenneth R Smith © 2008-2017 Bayview Strategies, LLC

Example 3: Pole Balancing

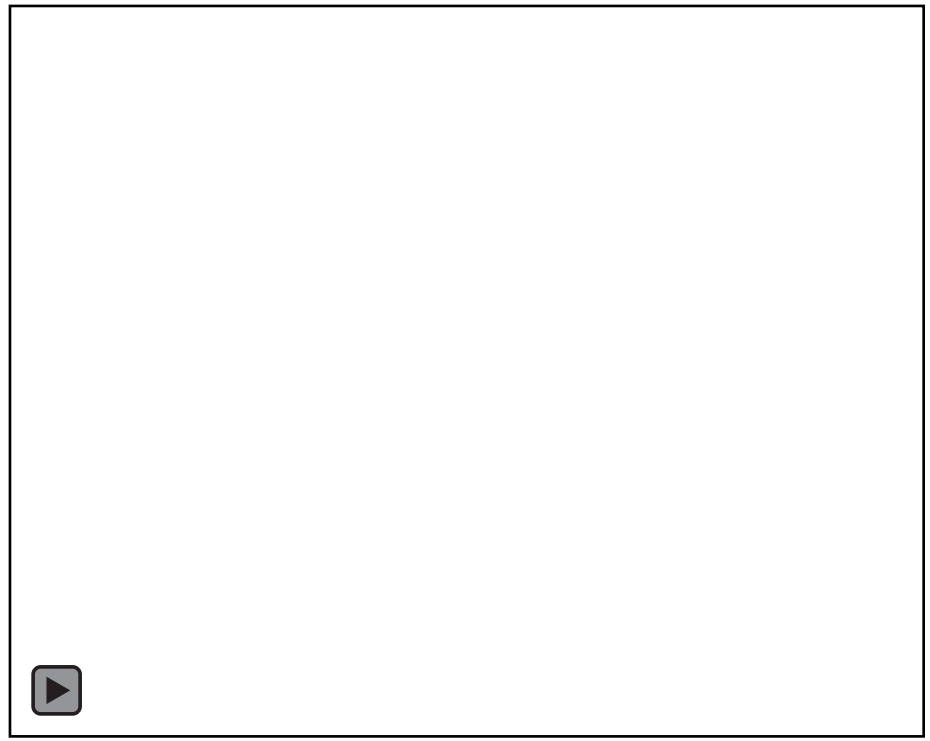
- Discrete vs Continuous State
- Discrete vs Continuous Action
- Deterministic vs Stochastic
- Fully vs Partially Observable
- Stationary vs Nonstationary
- Single Agent vs Multi Agent



Single Pole Balancing

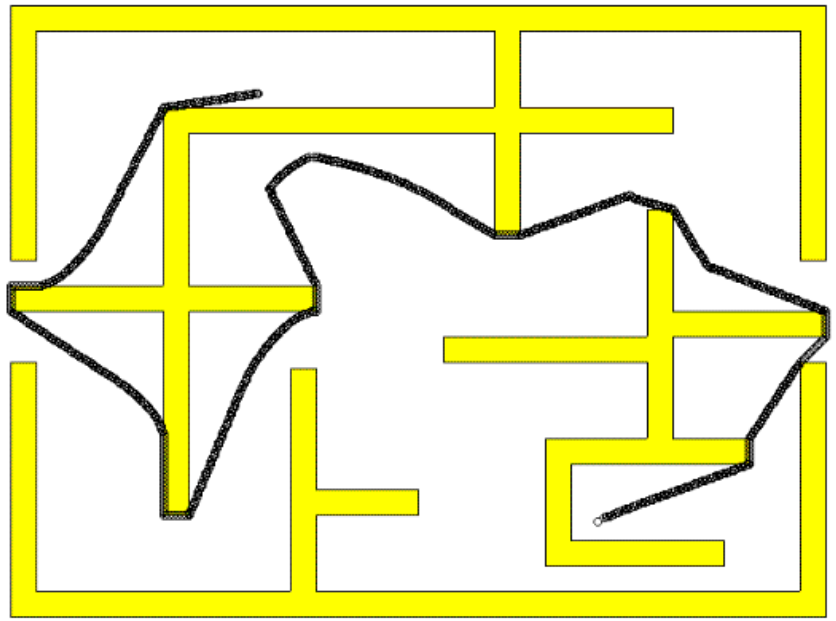
Example 3: Pole Balancing

- Discrete vs **Continuous** State
- Discrete vs **Continuous** Action
- **Deterministic** vs *Stochastic*
- **Fully** vs Partially Observable
- **Stationary** vs *Nonstationary*
- **Single Agent** vs Multi Agent



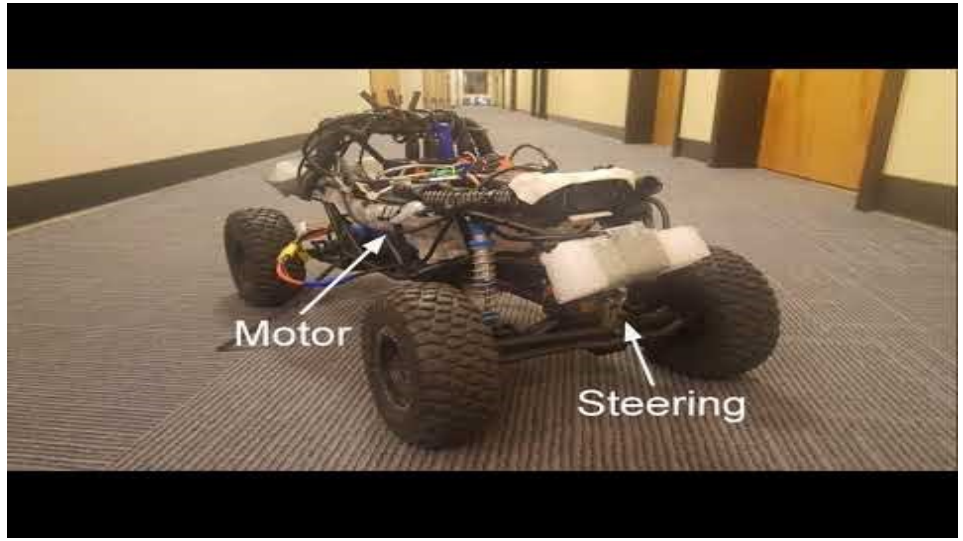
Example 4: Robot Navigation

- Discrete vs Continuous State
- Discrete vs Continuous Action
- Deterministic vs Stochastic
- Fully vs Partially Observable
- Stationary vs Nonstationary
- Single Agent vs Multi Agent



Example 4: Robot Navigation

- Discrete vs **Continuous** State
- Discrete vs **Continuous** Action
- **Deterministic** vs *Stochastic*
- Fully vs **Partially** Observable
- **Stationary** vs *Nonstationary*
- **Single Agent** vs *Multi Agent*



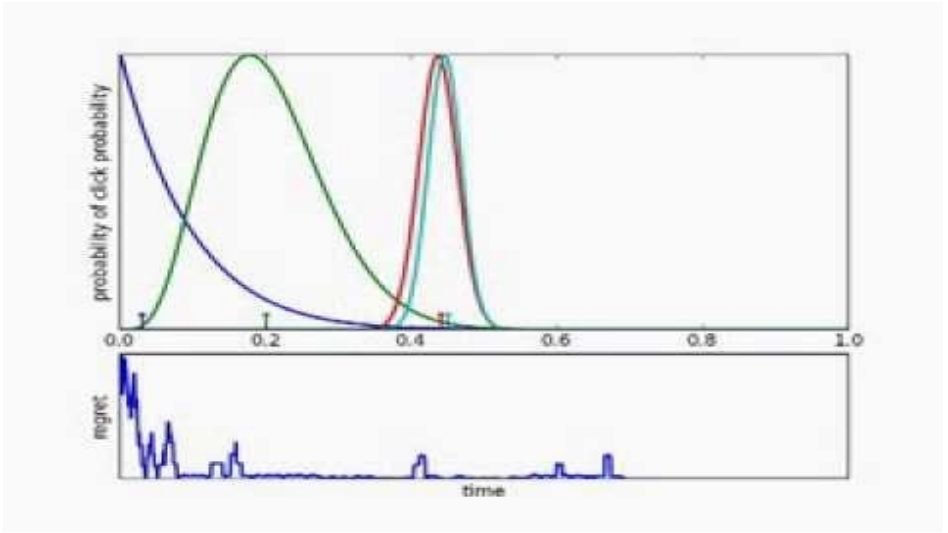
Example 5: Web Banner Advertising

- Discrete vs Continuous State
- Discrete vs Continuous Action
- Deterministic vs Stochastic
- Fully vs Partially Observable
- Stationary vs Nonstationary
- Single Agent vs Multi Agent

Example 5: Web Banner Advertising

Single

- ~~Discrete~~ vs Continuous State
- **Discrete** vs Continuous Action
- Deterministic vs **Stochastic**
- **Fully** vs Partially Observable
- **Stationary** vs *Nonstationary*
- **Single Agent** vs *Multi Agent*

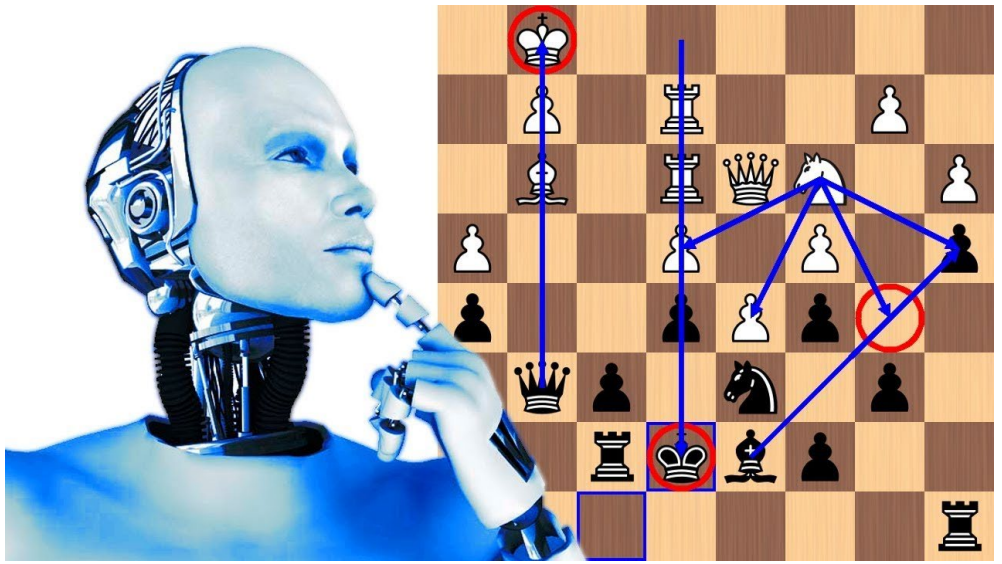


Example 6: Chess

- Discrete vs Continuous State
- Discrete vs Continuous Action
- Deterministic vs Stochastic
- Fully vs Partially Observable
- Stationary vs Nonstationary
- Single Agent vs Multi Agent

Example 6: Chess

- **Discrete** vs Continuous State
- **Discrete** vs Continuous Action
- **Deterministic** vs *Stochastic*
- **Fully** vs *Partially* Observable
- *Stationary* vs **Nonstationary**
- *Single Agent* vs **Multi Agent**



Example 7: Texas Hold'em

- Discrete vs Continuous State
- Discrete vs Continuous Action
- Deterministic vs Stochastic
- Fully vs Partially Observable
- Stationary vs Nonstationary
- Single Agent vs Multi Agent

Example 7: Texas Hold'em

- **Discrete** vs Continuous State
- **Discrete** vs Continuous Action
- Deterministic vs **Stochastic**
- *Fully* vs **Partially** Observable
- *Stationary* vs **Nonstationary**
- *Single Agent* vs **Multi Agent**

Policies and Value Functions

Policies

- A policy, at any given point in time, **decides** which action the agent selects
- A policy fully defines the **behavior** of an agent
- Policies can be:
  - Markovian $\subseteq$ History-dependent
  - Deterministic $\subseteq$ Stochastic
  - Stationary $\subseteq$ Non-stationary

Policies

- A policy, at any given point in time, **decides** which action the agent selects
- A policy fully defines the **behavior** of an agent
- Policies can be:
  - **Markovian** $\subseteq$ History-dependent
  - **Deterministic** $\subseteq$ Stochastic
  - **Stationary** $\subseteq$ Non-stationary

Value functions

- Given a policy π, it is possible to define the **utility** of each state: **Policy Evaluation**
- Value function
  - $V^{\pi}(s) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^{t} r_{t+1} | s_t = s]$
- For **control purposes**, rather than the value of each state, it is easier to consider **the value of each action** in each state
  - $Q^{\pi}(s, a) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^{t} r_{t+1} | s_t = s\,, a_t = a]$

Bellman Expectation Equation

- The value function can again be **decomposed** into immediate reward plus discounted value of successor state

$$V^{\pi}(s) = E_{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1})|s_t = s]$$

$$= \sum_{a} \sum_{s'} \pi(a|s) * P(s'|s,a) * (R(s,a,s') + \gamma V^{\pi}(s'))$$

- The action-value function can similarly be decomposed:

$$Q^{\pi}(s,a) = E_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1})|s_t = s, a_t = a] =$$
$$\sum_{s'} P(s'|s,a) * (R(s,a,s') + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s',a'))$$

Matrix Form

- The Bellman expectatio equation can be expressed **concisely** using the induced MRP

$$V^{\pi} = R^{\pi} + \gamma P^{\pi} V^{\pi}$$

- with **direct solution**

$$V^{\pi} = (I - \gamma P^{\pi})^{-1} R^{\pi}$$

Bellman Operators

- The Bellman operator T^π **maps** value functions to value functions:
  - $(T^\pi V^\pi)(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V^\pi(s')\right)$
  - Using Bellman operator, Bellman expectation equation can be **compactly** written as:

$$T^\pi V^\pi = V^\pi$$

  - V^π is a **fixed point** of the Bellman operator T^π
  - This is a **linear equation** in V^π and T^π
  - If $0 < \gamma < 1$ then T^π is a **contraction** w.r.t. the maximum norm

Policy performance

- The performance of a policy π is the expected discounted sum of the rewards collected by π

$$J^{\pi} = \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R_t(s_t, a_t, s_{t+1}) \,|\pi\right]$$

$$= \sum_{s} \mu(s) V^{\pi}(s)$$

Stationary Distribution

- Given a policy π, the sequence of states comes from a **Markov chain**
- P^{π} is the **state transition probability matrix**
- If P^{π} is regular, then the chain converges to the **stationary distribution**

$$d_{\mu}^{\pi}(s') = \sum_{s} \gamma d_{\mu}^{\pi}(s) P^{\pi}(s'|s), \forall s' \in S$$

- Policy performance:

$$J^{\pi} = \sum_{s} \mu(s) V^{\pi}(s) = \sum_{s} d_{\mu}^{\pi}(s) R^{\pi}(s)$$

Optimal Value Function

- The **optimal state-value function** $V^*(s)$ is the maximum value function over all policies

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

  - The optimal value function specifies the **best** possible performance in the MDP
  - An MDP is «**solved**» when we know the optimal value function

Optimal Policy

- Value functions define a pratial ordering over policies

$$\pi \geq \pi' \text{ if } V^{\pi}(s) \geq V^{\pi'}(s), \forall s \in S$$

- For any MDP
  - There exists **an optimal policy** π^* that is better than or equal to all the other policies: $\pi^* \geq \pi, \forall \pi$
  - All optimal policies achieve the **optimal value function**
  - There is always a **deterministic optimal policy** for any MDP

Bellman Optimality Equation

- Bellman optimality equation for V^*

$$V^*(s) = \max_{\mathrm{a}} \left(\sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V^*(s')) \right)$$

- Bellman optimality equation for Q^*

$$Q^*(s,a) = \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right)$$

- Bellman optimality equation is **non-linear**
- **No closed-form** solution for the general case

Algorithms

Solving MDPs

- Brute force
  - Unfeasible even for simple problems
- Dynamic Programming
  - Requires knowledge of the transition model
- Linear Programming
  - Better worst-case complexity, but worse in average
- Reinforcement Learning
  - Without knowledge of the model

When is RL useful?

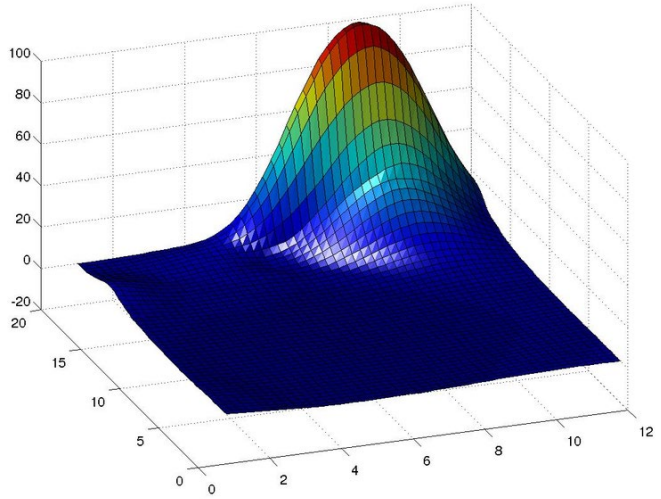
- When the dynamics of the environment are **unknown** or **difficult to be modeled**
  - E.g., trading, betting
- When the model of the environment is too **complex** to be solved exactly, so that **approximate** solutions are searched for
  - E.g., humanoid robot control, group elevator dispatching

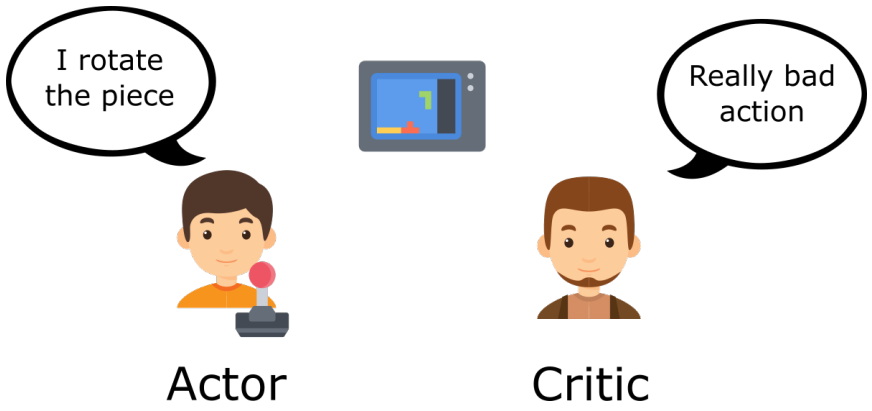
Classification of RL Techniques

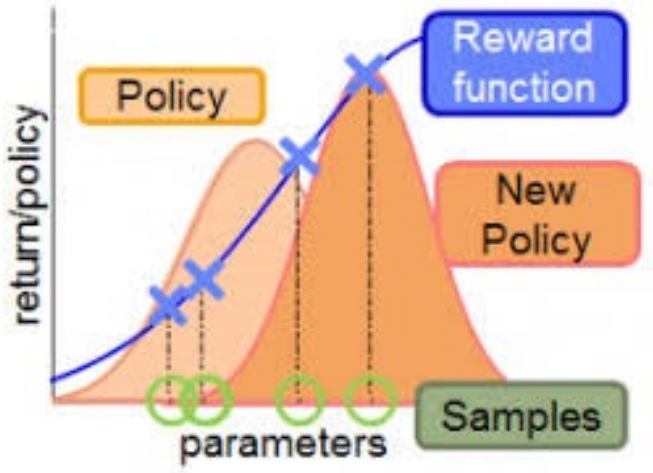
- Model-free vs Model-based
- On-policy vs Off-policy
- Online vs Offline
- Tabular vs Function Approximation
- Value-based vs Policy-based vs Actor-Critic

Reinforcement Learning Approaches

- Value-based
  - Estimate the utility of state-action pairs
- Policy-based
  - Search for the best parametric policy
- Actor-critic
  - Combine the two previous approaches







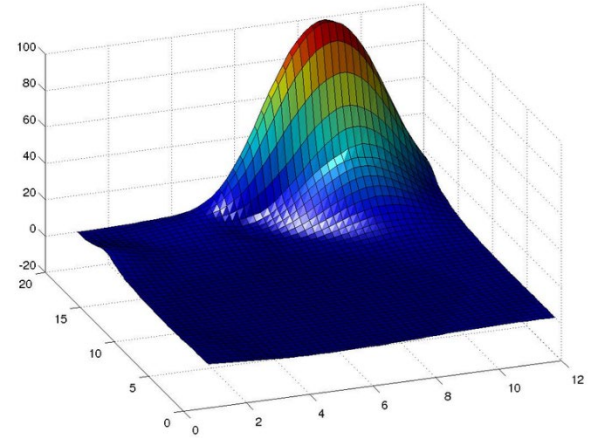
Value-Based Techniques

- **Learn** value function and **implicit** policy
- Monte Carlo vs Temporal Difference
- Some algorithms
  - SARSA (on-policy)

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

  - Q-learning (off-policy)

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$



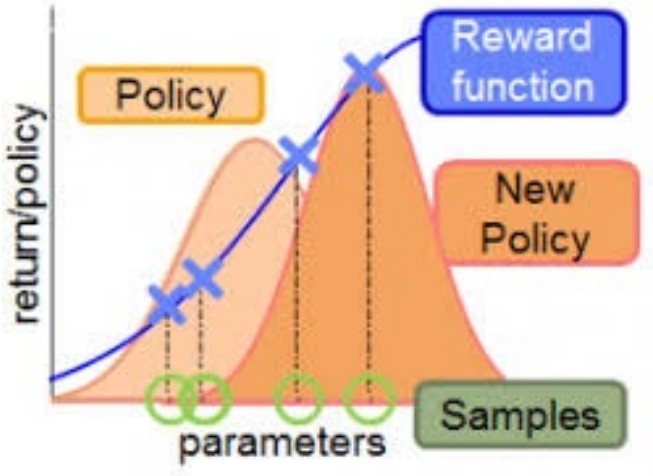
- Continuous states => **Function approximation,** but **no** supervised learning
- When to use them
  - Discrete actions
  - Full Observability

Policy-Based Techniques

- **No** value function and **learn** policy
- Some algorithms
  - Policy gradient: REINFORCE, NPG, DDPG
  - Trust region: REPS, TRPO, PPO, POIS
  - Parameter-based: PGPE, NES, CEM
- Advantages
  - Continuous actions
  - Partial Observability
  - Can benefit from **demonstrations**
  - **Stochastic** policies
- Disadvantages
  - May converge to **local** optima
  - **Sample inefficient**

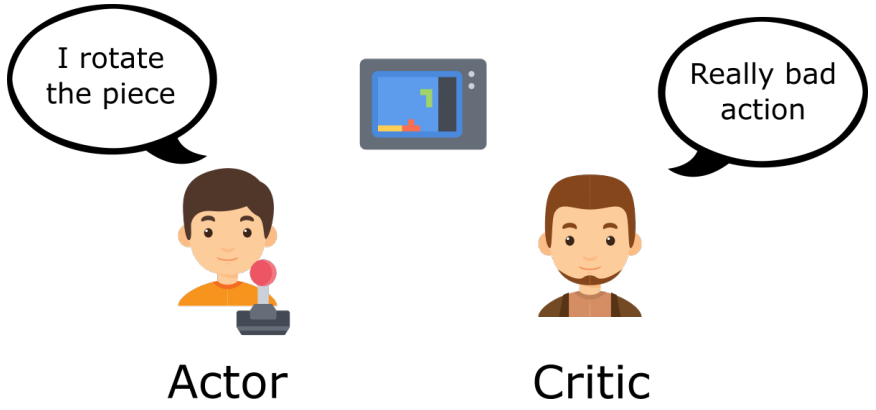
$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\sum_{t=1}^{T} G_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3}$$



Actor-Critic Techniques

- Exploit the best of the previous approaches
- Algorithms
  - A2C, A3C, SAC, …
- When to use them
  - Continuous actions
  - Full observability

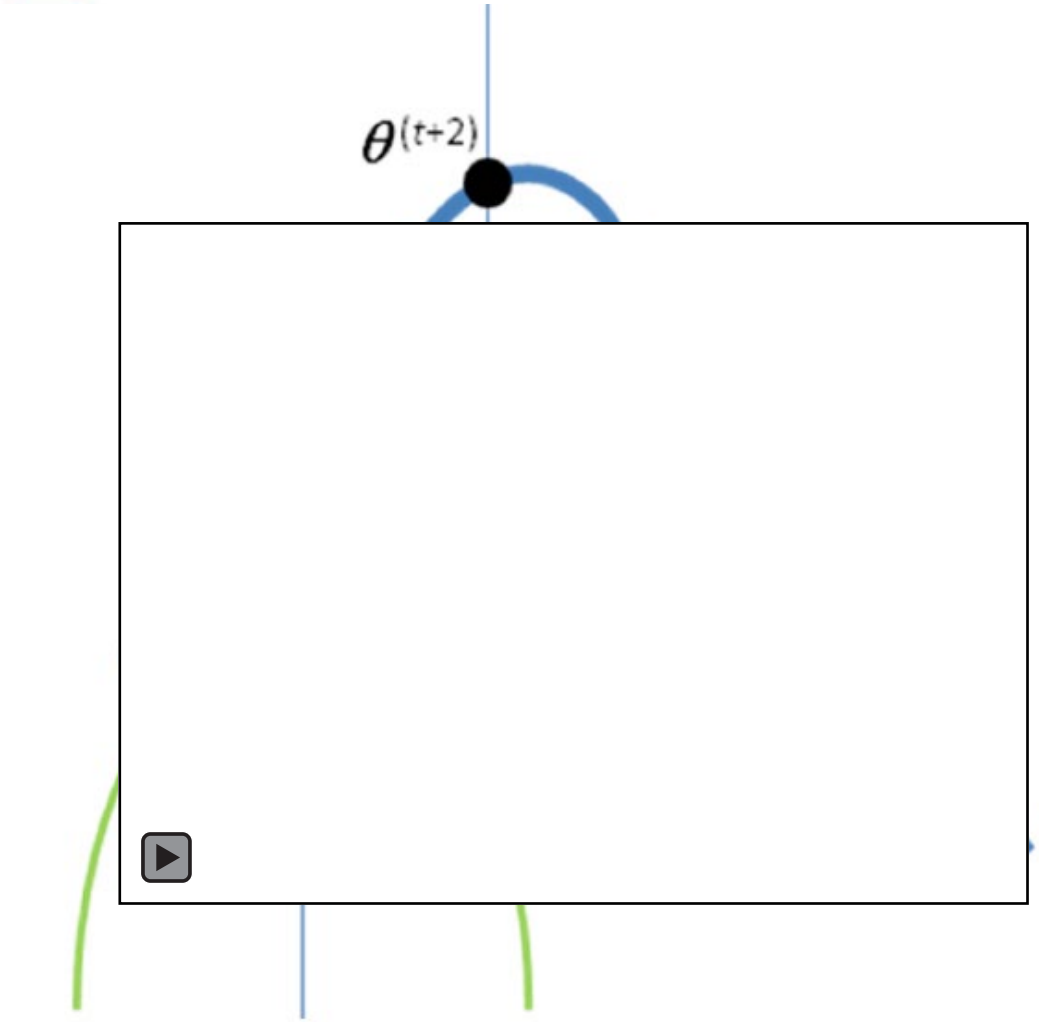


$$Q_w(s,a) \approx Q_w^{\pi_\theta}(\mathrm{s},\mathrm{a})$$

$$\nabla_\theta J(\theta) \approx E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_w(s,a)]$$

Other RL Techniques

- Multi-Objective RL
- Inverse RL
- Transfer RL
- Safe RL
- Multi-Agent RL



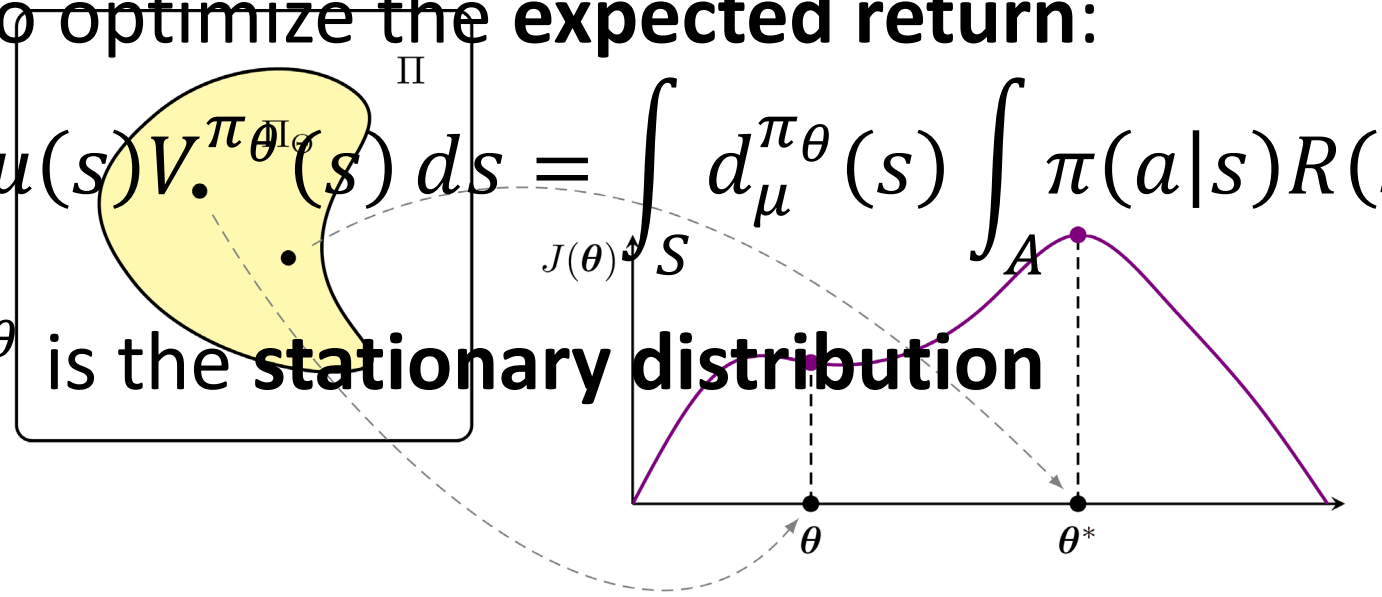
Policy Search

Policy Objective Function

- **Goal:** given a policy $\pi_\theta(a|s)$ with parameters θ, find best θ
- We want to optimize the **expected return**:

$$J(\theta) = \int_S \mu(s) V^{\pi_\theta}(s)\, ds = \int_S d_\mu^{\pi_\theta}(s) \int_A \pi(a|s) R(s,a) da ds$$

- Where $d_\mu^{\pi_\theta}$ is the **stationary distribution**



Greedy vs Incremental

- **Greedy** updates

$$\theta_{t+1} = argmax_a \mathbb{E}_{\pi_{\theta_t}}[Q^{\pi}(s,a)]$$

$$V^{\pi_0} \xrightarrow{\text{large change}} \pi_1 \xrightarrow{\text{large change}} V^{\pi_1} \xrightarrow{\text{large change}} \pi_2 \xrightarrow{\text{large change}}$$

- Potentially **unstable** learning process with **large policy jumps**
- **Policy gradient** updates

$$\theta_{t+1} = \theta_t + \alpha \left. \frac{dJ(\theta)}{d\theta} \right|_{\theta=\theta_t}$$

- **Stable** learning process with **smooth policy improvement**

Policy Gradient Theorem

- For any differentiable policy $\pi(a|s)$, the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)]$$
$$= \int_S d_\mu^{\pi_\theta}(s) \int_A \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a) da ds$$

REINFORCE

- REINFORCE use a Monte-Carlo estimation
- The return G_t is used as an unbiased sample of $Q^{\pi_\theta}(s_t, a_t)$:

$$\Delta_t = \alpha \nabla_\theta \log \pi_\theta(a|s)\, G_t$$

function REINFORCE()
  Initialize θ arbitrarily
  for all episodes $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
    for $t = 1$ to $T - 1$ **do**
      $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t, s_t) G_t$
    end for
  end for
  return θ
end function

Reducing Variance using a Baseline

- We subtract a **baseline function** $B(s)$ from the policy gradient
- This can **reduce variance**, without changing expectation

$$\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) B(s)] = \int_S d_\mu^{\pi_\theta}(s) \int_A \nabla_\theta \pi_\theta(a|s) B(s) da ds$$
$$= \int_S d_\mu^{\pi_\theta}(s) B(s) \nabla_\theta \int_A \pi_\theta(a|s) da ds$$
$$= 0$$

- The **optimal baseline** for REINFORCE is the value function $V^{\pi_\theta}(s)$
- We can rewrite the policy gradient using the **advantage function**

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$
$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s,a)]$$

- This leads to **actor-critic** approaches

Natural Policy Gradient

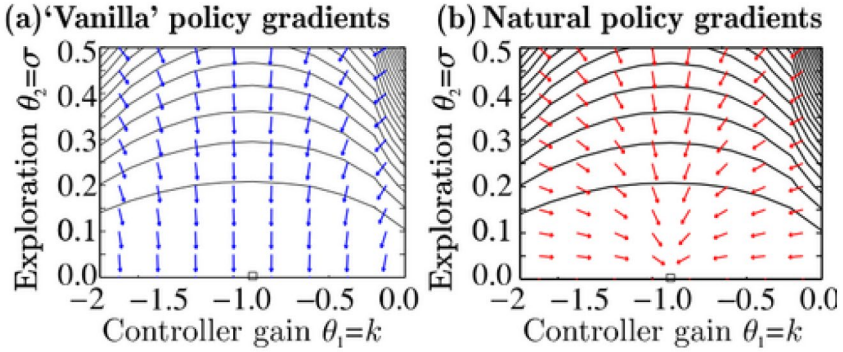
- Vanilla gradient is **sensitive to parametrization** θ
- A more efficient direction id the **natural gradient**

$$\widetilde{\nabla}_\theta J(\theta) = G^{-1}(\theta)\nabla_\theta J(\theta)$$

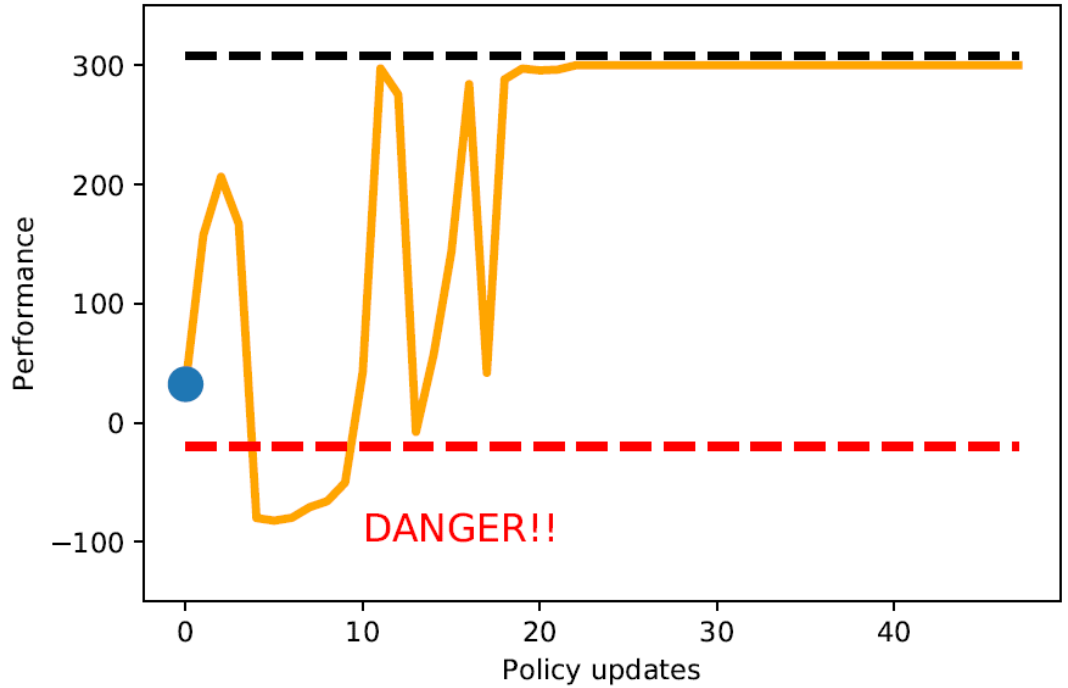
- Where $G(\theta)$ is the **Fisher information matrix**

$$G(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, \nabla_\theta \log \pi_\theta(a|s)^T]$$

- They correspond to the **steepest ascent in policy space** and not in the parameter space



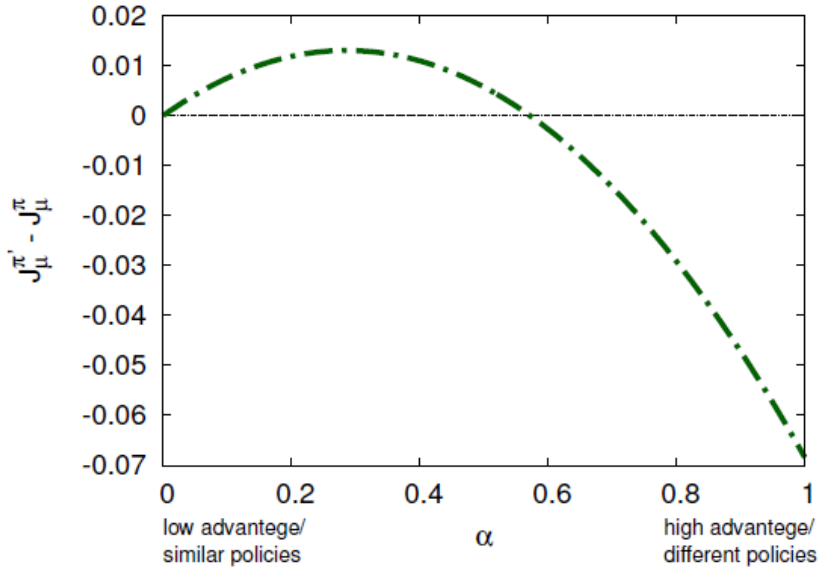
Issues with Policy Search



Is it possible to have a **monotonically improving** learning process?

Conservative Policy Iteration (CPI)

- Consider the **policy improvement**
  - $J^{\pi'} - J^{\pi} = \boldsymbol{d_{\mu}^{\pi'}}^{T} \boldsymbol{A_{\pi}^{\pi'}}$ Unknown!
  - $A_{\pi}^{\pi'}(s) = \mathbb{E}_{a \sim \pi'(\cdot|s)}[Q^{\pi}(s,a)] - V^{\pi}(s)$
- **Conservative** policy update
  - $\pi' = \alpha\pi^{+} + (1-\alpha)\pi$
- **Lower bound** the policy improvement
  - $J^{\pi'} - J^{\pi} \geq \alpha \boldsymbol{d_{\mu}^{\pi}}^{T} \boldsymbol{A_{\pi}^{\pi'}} - \alpha^2 \frac{2\gamma}{(1-\gamma)^2} \|\boldsymbol{A_{\pi}^{\pi'}}\|_{\infty}$
- **Maximize** the lower bound
  - $\alpha = \frac{(1-\gamma)^2 \boldsymbol{d_{\mu}^{\pi}}^{T} \boldsymbol{A_{\pi}^{\pi'}}}{4\gamma \|\boldsymbol{A_{\pi}^{\pi'}}\|_{\infty}}$



Kakade and Langford, ICML 2002

Safe Policy Iteration (SPI)

- **New lower bound** to the policy improvement

$$J^{\pi'} - J^{\pi} \geq \alpha \boldsymbol{d}_{\mu}^{\boldsymbol{\pi}^T} \boldsymbol{A}_{\boldsymbol{\pi}}^{\boldsymbol{\pi}'} - \alpha^2 \frac{\gamma}{(1-\gamma)^2} \|\boldsymbol{\pi}^+ - \boldsymbol{\pi}\|_\infty \|\boldsymbol{A}_{\boldsymbol{\pi}}^{\boldsymbol{\pi}'}\|_\infty$$

New term!

- **Maximize** the lower bound

$$\alpha = \frac{(1-\gamma)^2 \boldsymbol{d}_{\mu}^{\boldsymbol{\pi}^T} \boldsymbol{A}_{\boldsymbol{\pi}}^{\boldsymbol{\pi}'}}{2\gamma \|\boldsymbol{\pi}^+ - \boldsymbol{\pi}\|_\infty \|\boldsymbol{A}_{\boldsymbol{\pi}}^{\boldsymbol{\pi}'}\|_\infty}$$

- **Advantage** w.r.t. CPI
  - Convergence to optimum in **finite time**

Pirotta, Pecorino, Calandriello and **MR**, ICML 2013

SPI Variants

- **Alternative (looser)** lower bound

$$J^{\pi'} - J^{\pi} \geq \alpha \boldsymbol{d}_{\mu}^{\boldsymbol{\pi}^T} \boldsymbol{A}_{\boldsymbol{\pi}}^{\boldsymbol{\pi}'} - \alpha^2 \frac{\gamma}{(1-\gamma)^2} \|\boldsymbol{\pi}^+ - \boldsymbol{\pi}\|_{\infty}^2 \|\boldsymbol{Q}^{\boldsymbol{\pi}}\|_{\infty}$$

This term can be bounded with the maximum KL

- This bounds allows for **state-dependent** α

$$\pi'(a|s) = \alpha(s)\pi^+(a|s) + \big(1 - \alpha(s)\big)\pi(a|s)$$

- **No closed-form** solution
  - **Iterative** solution with complexity $\mathrm{O}(S \log S)$

Pirotta, Pecorino, Calandriello and **MR**, ICML 2013

Safe Policy Gradient (SPG)

- SPI works only for **finite MDPs**
- For continuous MDPs, **policy gradient** are a standard solution
- We generalized the SPI approach to policy gradient methods using **Gaussian policies** $\pi_{\boldsymbol{\theta}} \sim N\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(s), \sigma^2\right)$ and $\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
  - $J(\theta') - J(\theta) \geq \alpha \|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})\|_2^2 -$

    $\alpha^2 \left(\frac{1}{\sqrt{2\pi}\sigma^3} \int_S d_\mu^{\boldsymbol{\theta}}(s) \left(|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})|^T |\boldsymbol{\phi}(s)| \right)^2 \int_A Q^{\boldsymbol{\theta}}(s,a)\, da\, ds + \frac{\gamma M_\phi^2}{2(1-\gamma)^2 \sigma^2} \|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})\|_1^2 \left\| \boldsymbol{Q}^{\boldsymbol{\theta}} \right\|_\infty \right)$
- We derived algorithms both for the **exact** and the **approximate** case

Pirotta, **MR,** and Bascetta, NeurIPS 2013

SPG Extensions

- Lipschitz MDPs
  - Pirotta, **MR** and Bascetta, MLJ 2015
- Adaptive Batch size
  - Papini, Pirotta and **MR**, NeurIPS 2017
- Non-Gaussian policies
  - Papini, Pirotta and **MR**, submitted to JMLR

Pirotta, **MR,** and Bascetta, NeurIPS 2013

Trust Region Policy Optimization (TRPO)

- TRPO is the **state of the art** in policy optimization
- TRPO considers the bound used in SPI/SPG with $\alpha = 1$ and **simplifies** it:

Should be max

$$-J^{\pi'} - J^{\pi} \geq \boldsymbol{d_{\mu}^{\pi}}^{T} \boldsymbol{A_{\pi}^{\pi'}} - \frac{\gamma}{(1-\gamma)^2} KL_{avg}(\pi' || \pi) \| \boldsymbol{Q^{\pi}} \|_{\infty}$$

- **Constrained** optimization problem

$$\max_{\pi'} \quad \boldsymbol{d_{\mu}^{\pi}}^{T} \boldsymbol{A_{\pi}^{\pi'}}$$

$$\text{s. t.} \quad KL_{avg}(\pi' || \pi) < \delta$$

Schulman, Levine, Abbeel and Jordan, ICML 2015

Trust Region Policy Optimization (TRPO)

- This heuristic algorithm has been recently (AAAI'20) proved to **converge to global optima**
  - It can be seen as an instance of a **mirror descent** method
  - **Rate** of convergence:
    - Unregularized: $\tilde{o}\left(\frac{1}{\sqrt{N}}\right)$
    - Regularized: $\tilde{o}\left(\frac{1}{N}\right)$
- Many algorithms have been recently derived from it

Shani, Efroni and Mannor, AAAI 2020

Proximal Policy Optimization (PPO)

- PPO is a **first-order optimization** that simplifies the implementation of TRPO
- It rewrites TRPO objective function:

$$\int_S d_\mu^{\pi_\theta}(s) \int_A \pi_{\theta'}(a|s) A^{\pi_\theta}(s,a) da ds = \int_S d_\mu^{\pi_\theta}(s) \int_A \pi_\theta(a|s) \frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s,a) da ds = \mathbb{E}_{\pi_\theta}[\rho(\theta') A^{\pi_\theta}(s,a)]$$

- It replaces the constraint on the KL, with a **clipping** of $\rho(\theta')$ within a small interval around 1

$$\mathbb{E}_{\pi_\theta}\left[\min\left(\rho(\theta')A^{\pi_\theta}(s,a), clip(\rho(\theta'), 1-\epsilon, 1+\epsilon)A^{\pi_\theta}(s,a)\right)\right]$$

Schulman, Wolski, Dhariwal, Radford and Klimov, arXiv 2017

Policy Optimization via Importance Sampling (POIS)

- Optimize a **statistical lower bound** on the estimated J
- **Off-policy** estimation via **Importance Sampling**
  - Cantelli's Inequality (with probability $1-\delta$):

$$J(\theta') \geq \frac{1}{N}\sum_{i=1}^{N} \frac{w_{\theta'}}{\theta}(\tau_i) R(\tau_i) - \frac{R_{max}}{1-\gamma}\sqrt{\frac{1-\delta}{\delta N} d_2(p_{\theta'} \,||p_\theta)}$$

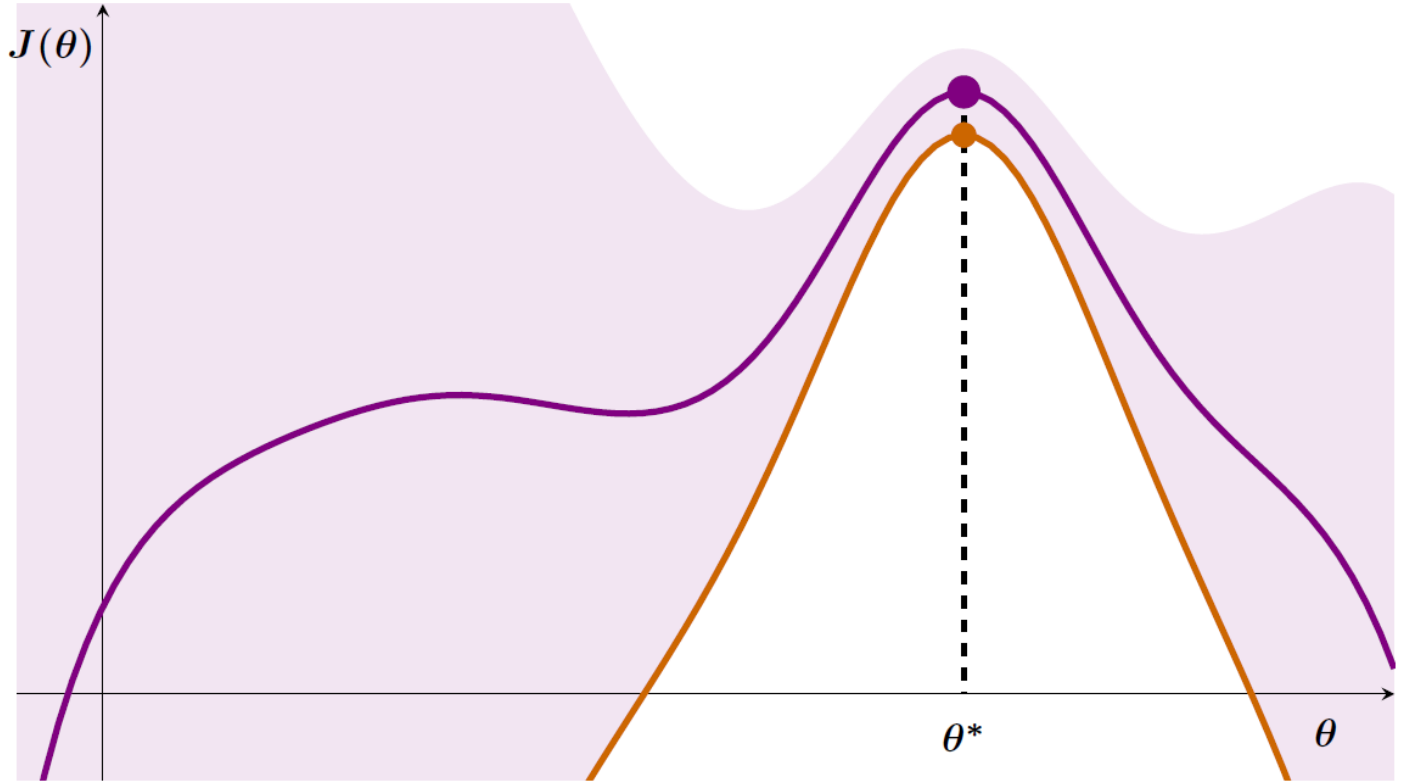
Importance weight

Exponentiated Rényi divergence

- The algorithm alternates **online** and **offline** steps

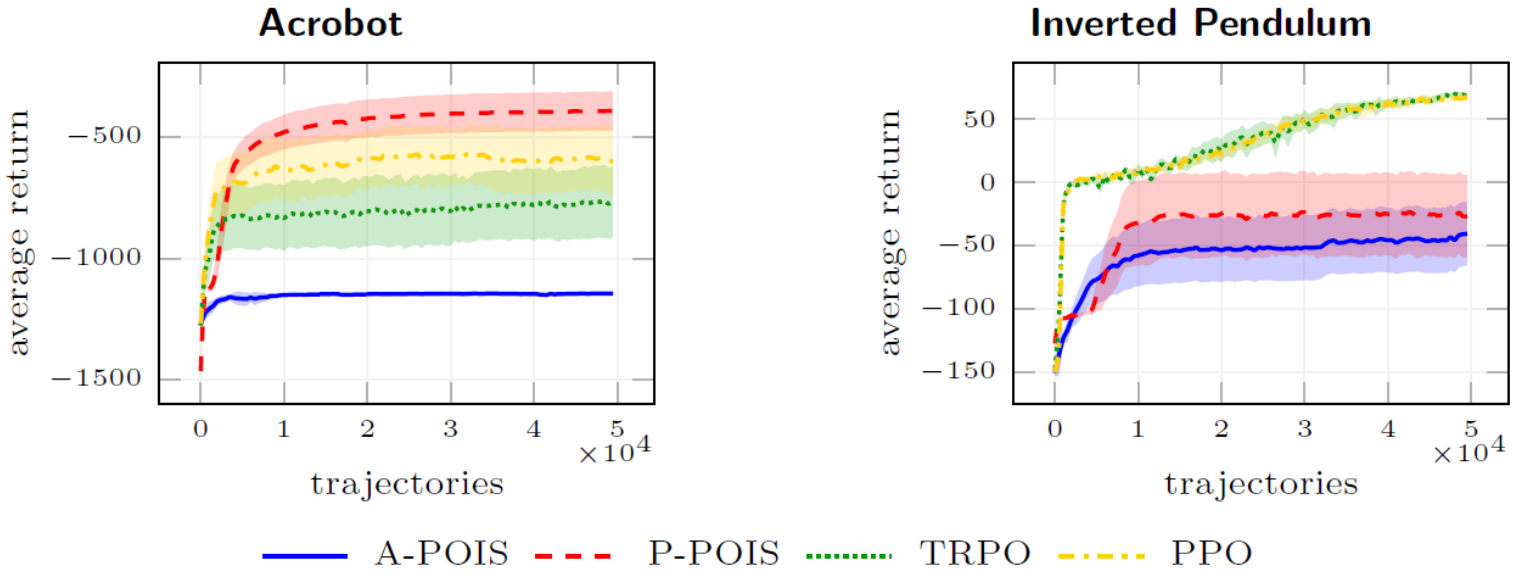
Metelli, Papini, Faccio and **MR**, NeurIPS 2018

POIS Algorithm



Experiments

- Comparison with TRPO (Schulman et al., 2015) and PPO (Schulman et al. 2017)



Applications

Reinforcement Learning Applications

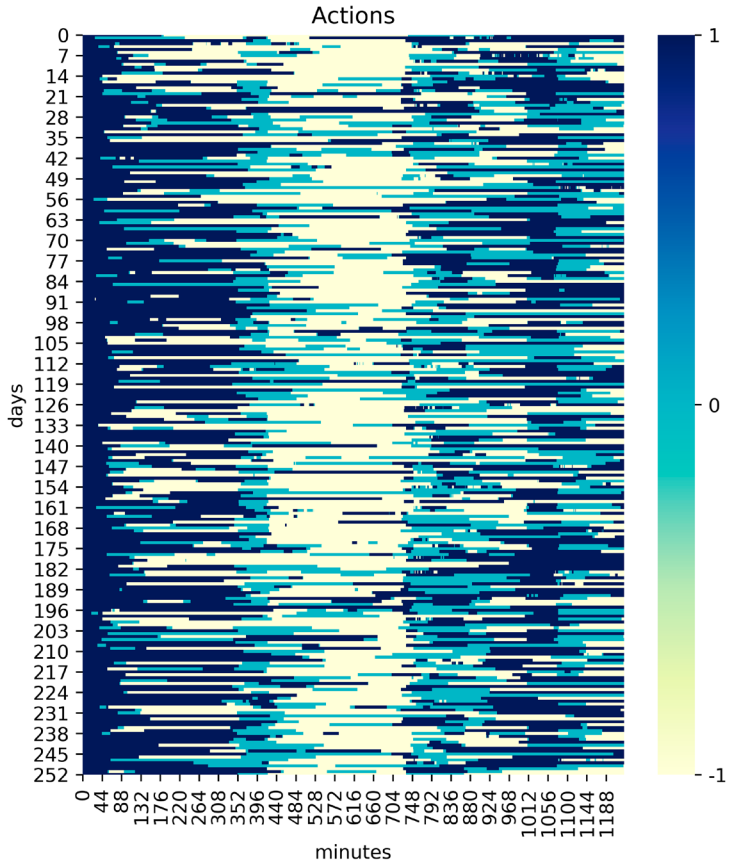
- Robotic Control
- Water Resource Management
- Internet Commerce
- Gaming
- Finance
- Power Systems
- Autonomous Vehicles
- Dialogue Management

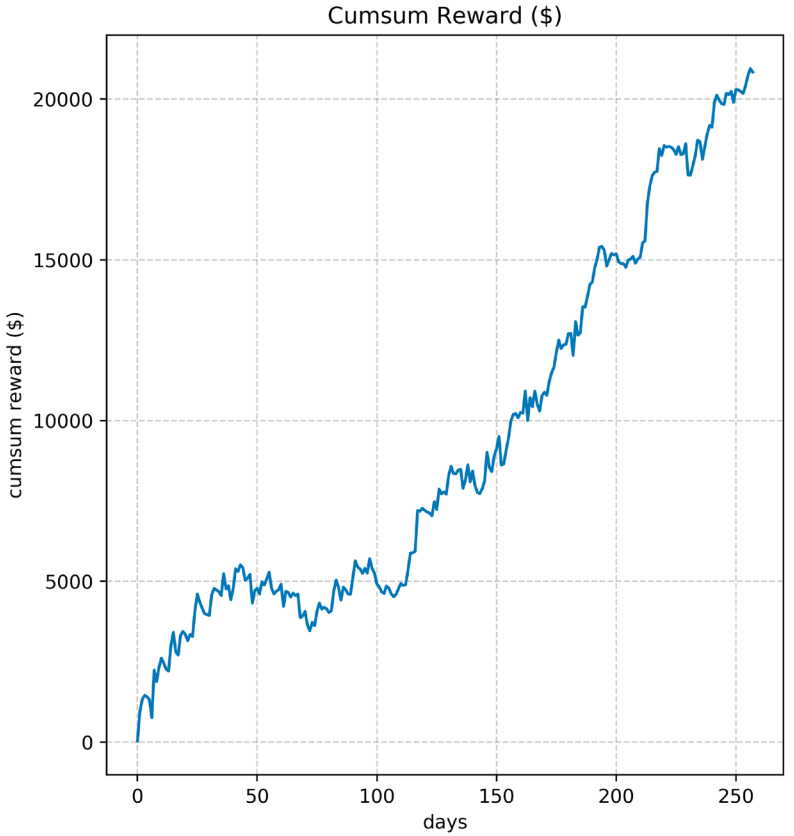
Forex Trading €/$: the problem

- State
  - Position (long/flat/short)
  - Time
  - Trades per minute
  - Price history (60)
- Actions
  - Long/flat/short (100K€)
- Reward
  - P&L – fees (2€)
- Dataset
  - 1-minute data from 2014 to 2018

Forex Trading €/$: the performance

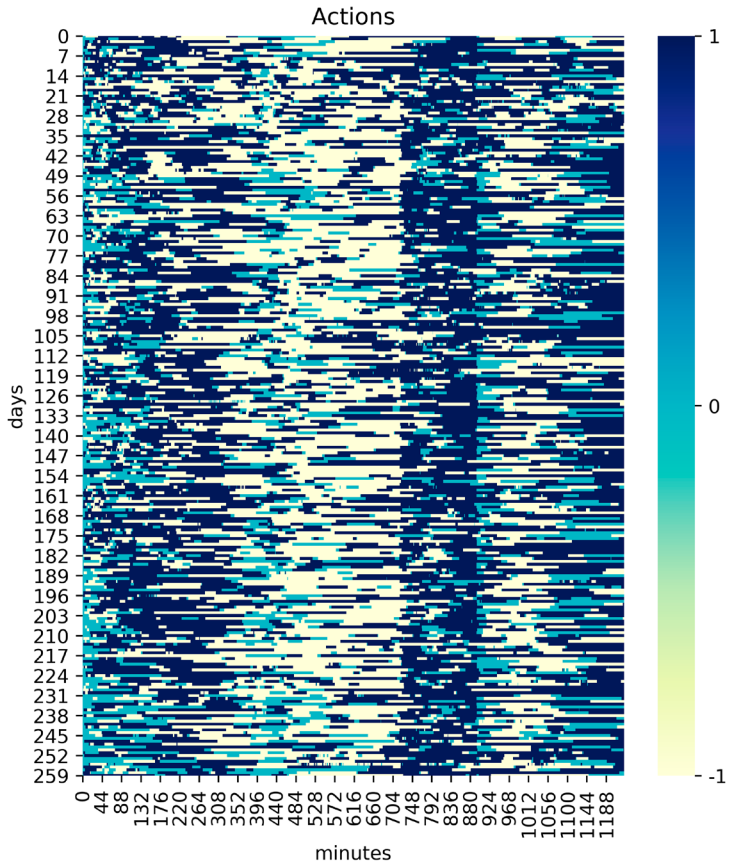
Train: 2015-2016 | Validation: 2014 | Test: 2017

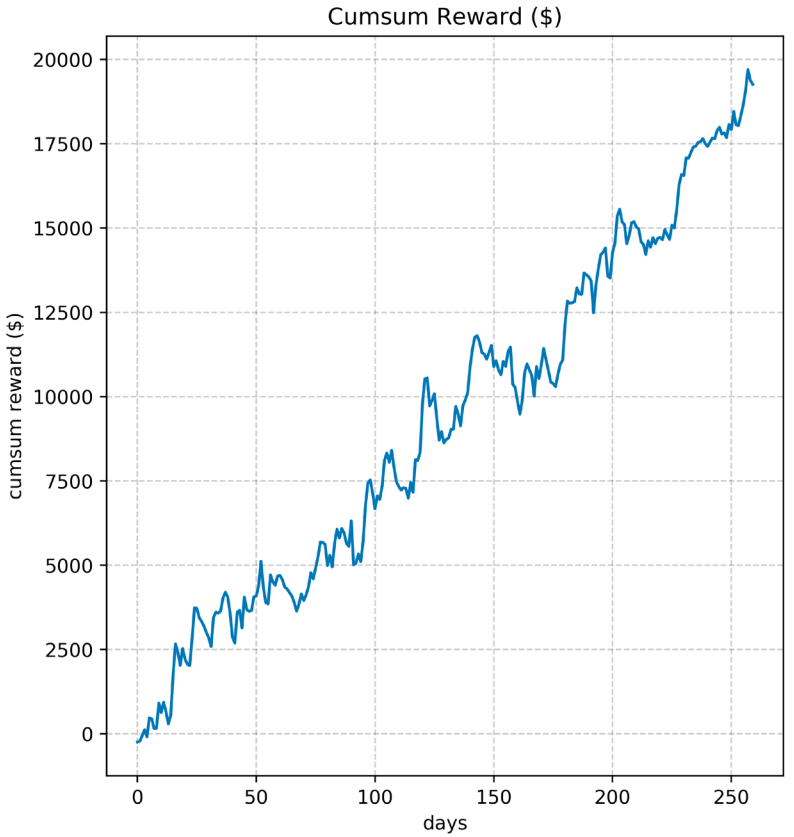




Forex Trading €/$: the performance

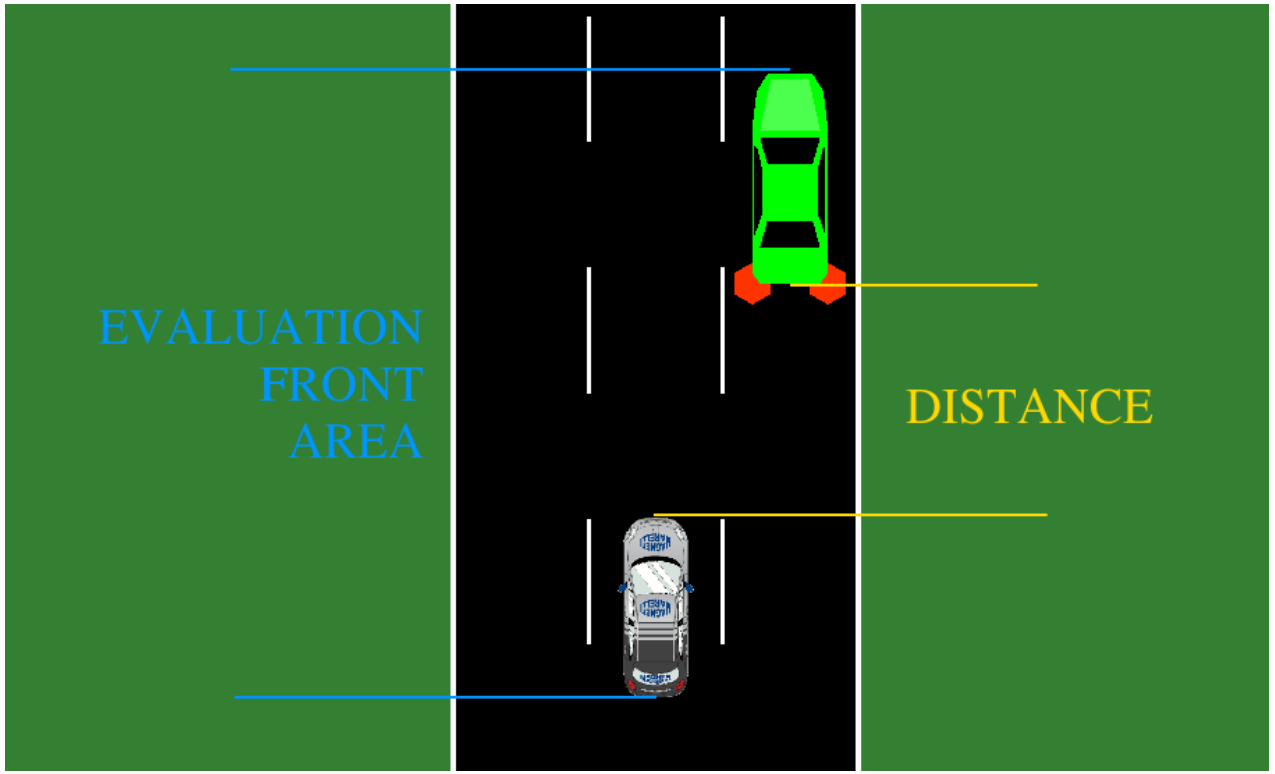
Train: 2016-2017 | Validation: 2015 | Test: 2018





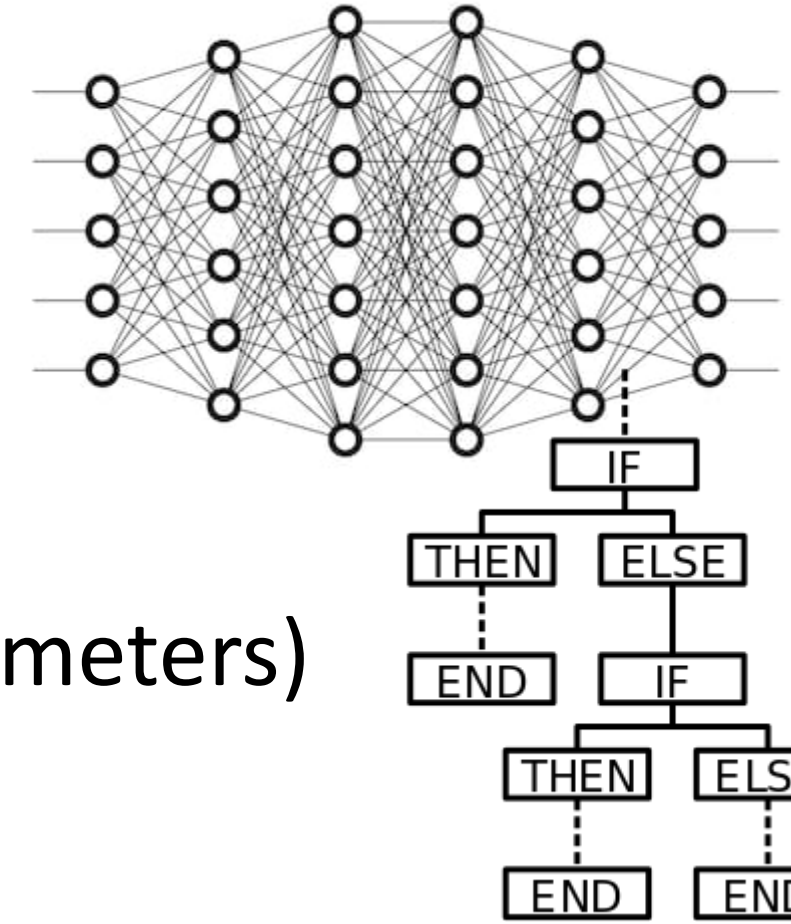
Autonomous Driving: Highway

- State (25 variables)
  - Ego lane occupancy
  - Ego speed
  - Distance front / Distance back
  - Speed front / Speed back
  - Occupancy front / Occupancy back
  - Free left / Free right
  - During lane change
- Actions
  - Car follower
  - Lane change left
  - Lane change right
- Rewards: multi-objective
  - Target speed
  - Safety violation
  - Occupy the rightmost lane
  - Avoid useless lane changes
- Issues
  - Delay (1s)
  - Partial observability

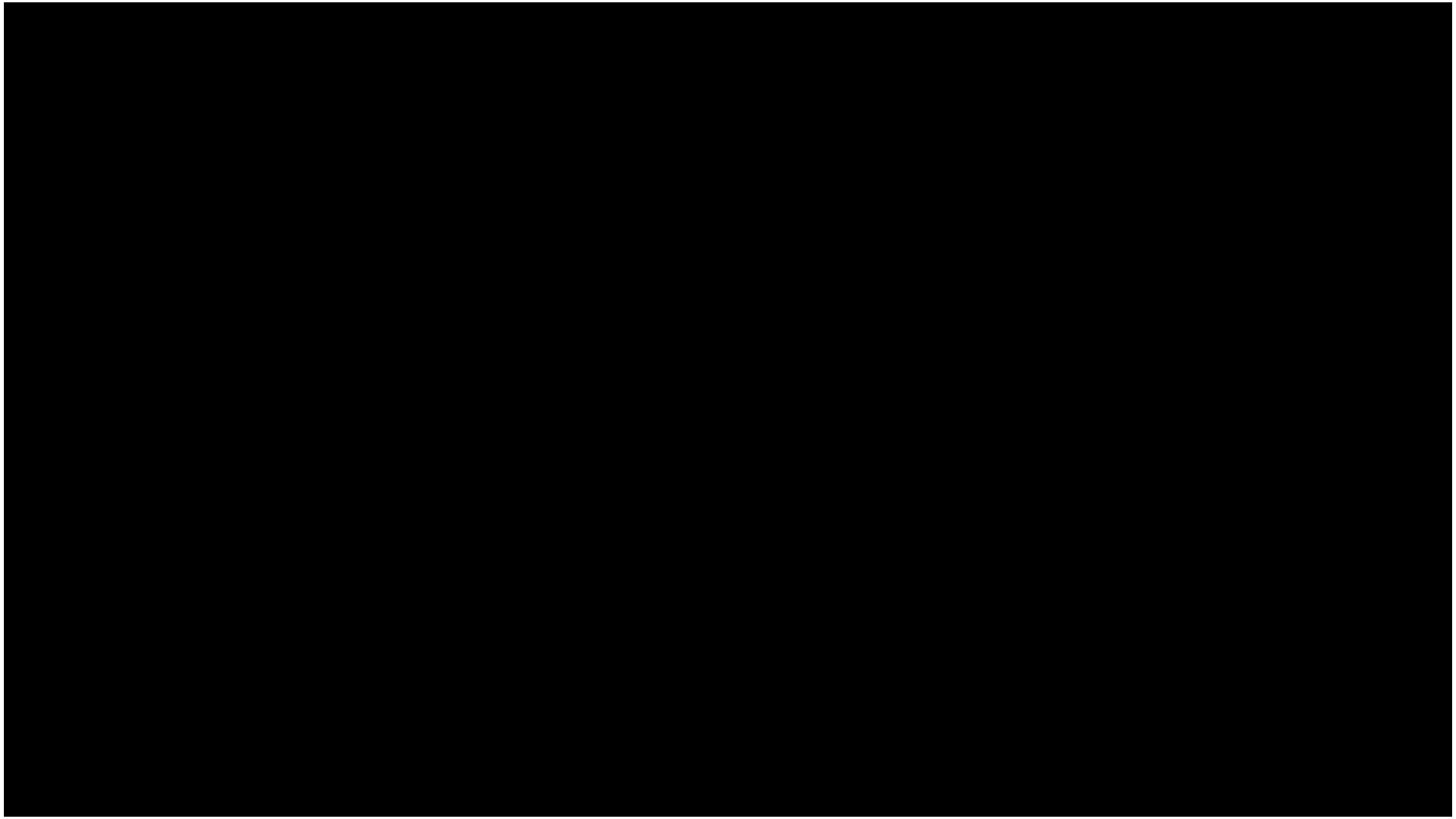


Autonomous Driving: algorithms

- Several deep algorithms
  - DQN
  - TRPO
- They are **not interpretable!**
  - PGPE + parametric rules (6 parameters)

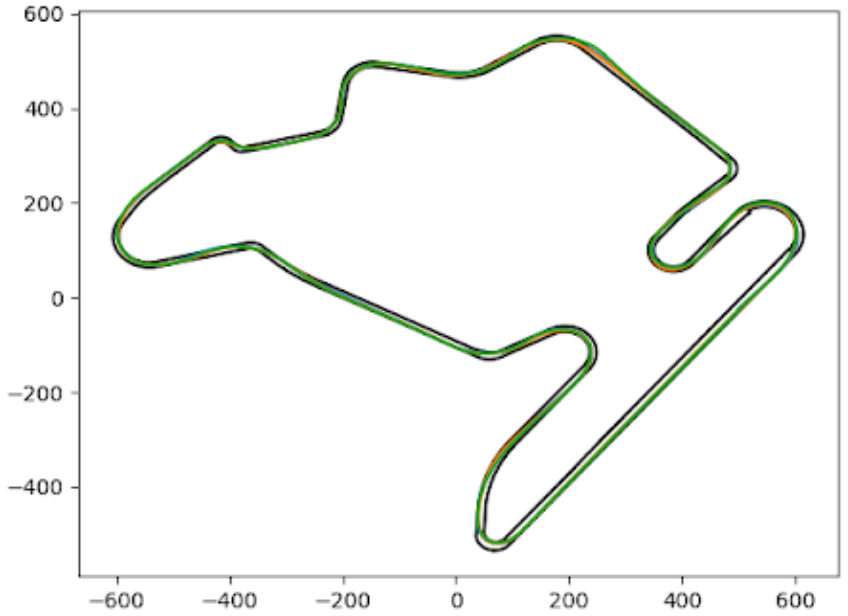


Autonomous Driving: performance



Car Racing: the problem

- Understand the potential of a new car
- Fast learning to drive a new car
  - Demonstrations from drivers
  - Different cars
- Learning from demonstrations
  - Stay close to the best trajectories
- Transfer learning
  - Reuse of samples
  - Importance weighting



Car Racing: the performance

Questions?