Advanced concepts of Reinforcement learning

a practical guide

Simon Hirlaender
IDA LAB
simon.hirlaender@plus.ac.at
Artificial intelligence and Human Interfaces
Digital and Analytical Sciences
University of Salzburg

What's the goal

Learn how to make **good** decisions under uncertainty

Goals of today

- We don't necessarily develop new algorithms!
- Learn how to:
  - ➡ identify and set up RL problems
  - ➡ apply RL appropriately
  - ➡ deal with common problems
- RL is not easy, don't expect to understand or solve things immediately

Stop,Think, Apply!

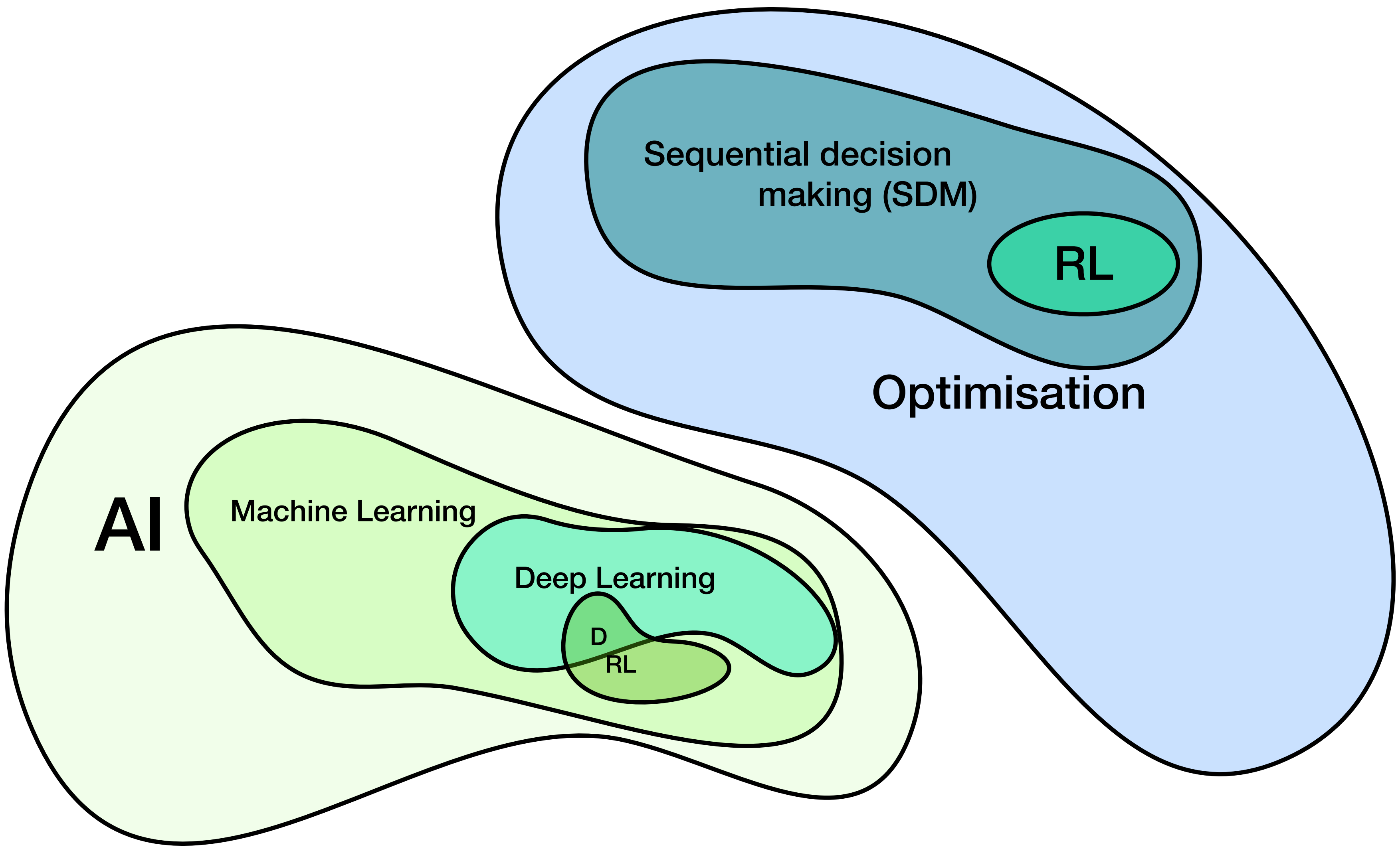
- Pick the right problems!
    - ➡ Ask: does this have a chance of solving an important problem? Does my optimization problem have a chance to be solved? Do I solve the right problem?

RL - what is it today?

- What is addressed by RL: position in AI, community of researchers applying tools, data-driven dynamic programming
- Little knowledge of probabilistic mechanism how data and rewards change over time
- Probe and learn dynamics to find control

B. Recht, 2018

AI and optimization viewpoint

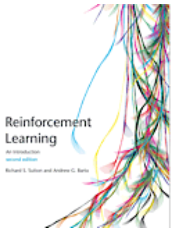


How to approach RL

- Pick the right problems (important and solvable)!
    - ➡ What is my goal?
    - ➡ What are my observables and my actions?
- Model the problem appropriately
- Training and evaluation of RL
- Are there better alternatives?

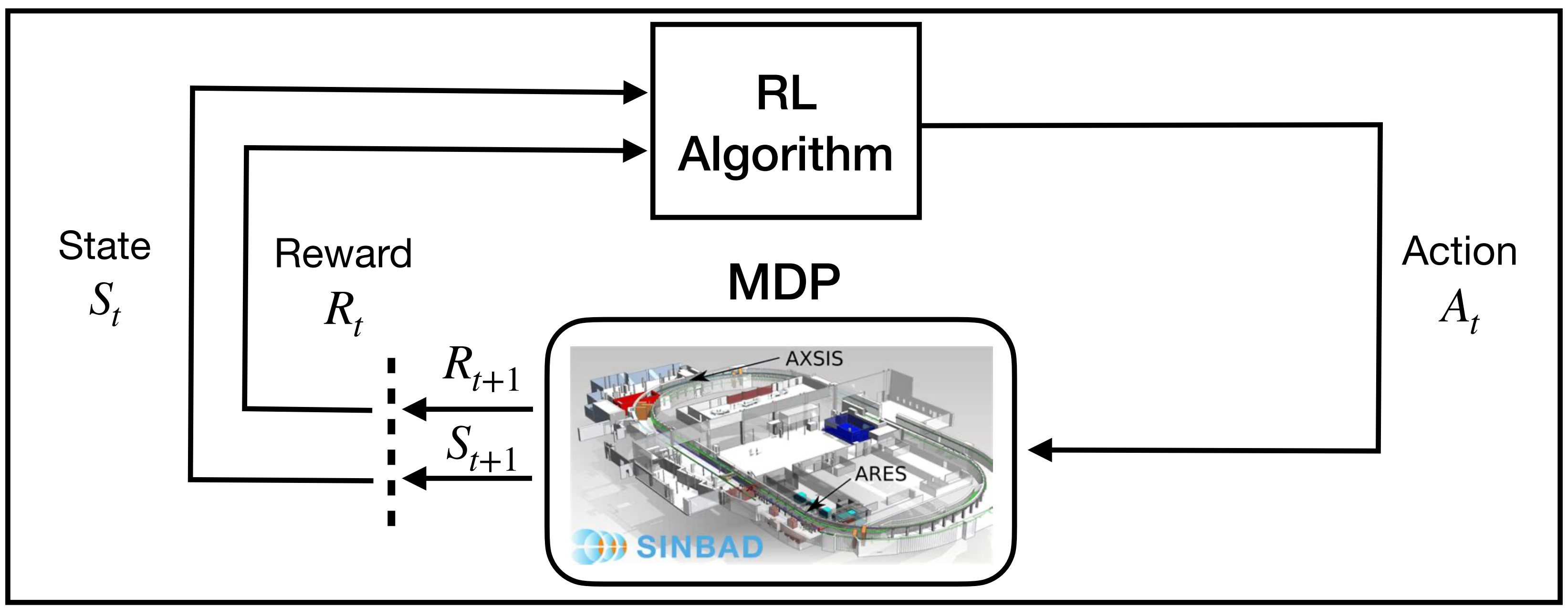
The entire problem

SB: 1, 3, 10, 17



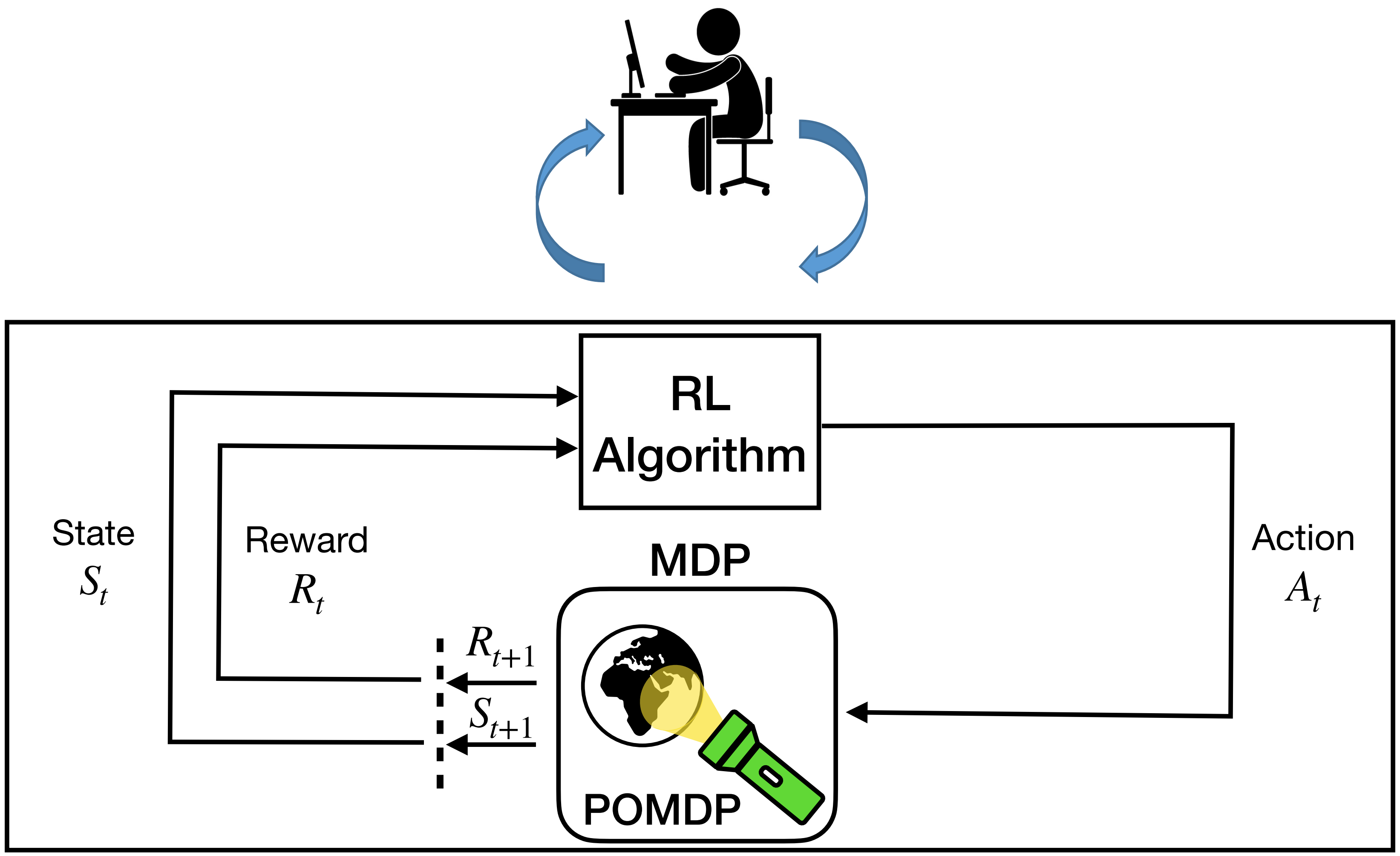
The entire problem

Markov decision process - MDP

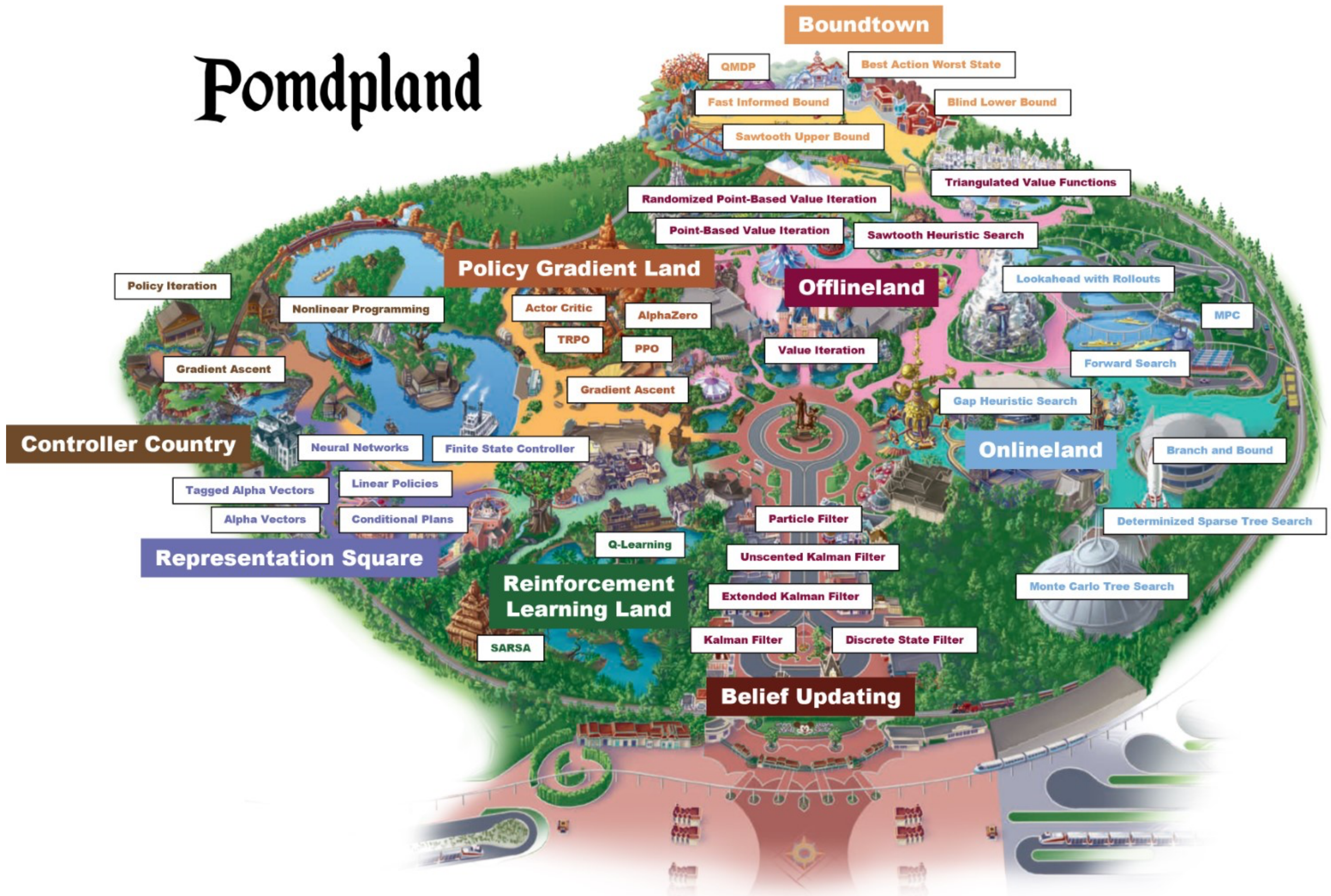


Common set-up

Partially observable Markov decision process - POMDP



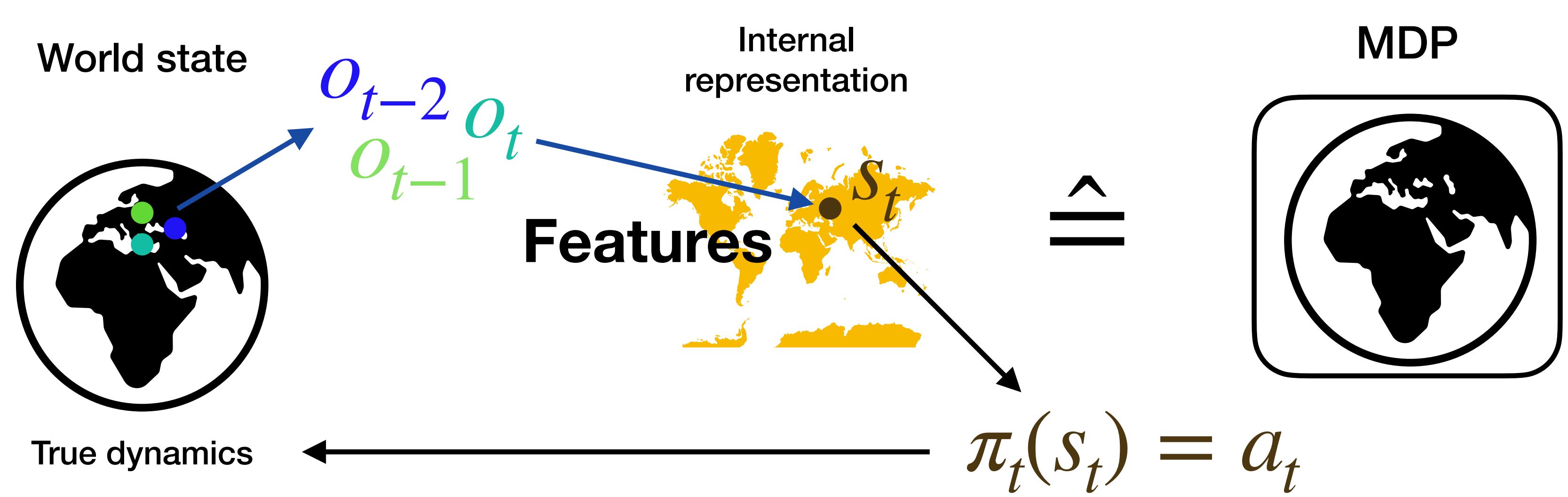
Wellcome to POMDPs



From Mykel Kochenderfer

Problem design - capture the right thing

- We want to solve an SDM problem: Information→Decision→Information→Decision→…
- Such problems are generally be stochastic!
- Consequently we build a feedback system not planing too far in the future:
    - We define a **state** $s_t = h_t(o_t, a_{t-1}, o_{t-1}, a_{t-2}, o_{t-2}\ldots)$, as a function holding **sufficient statistics** until time step t for a decision - (example pong)
    - We look for a decision based on s_t via: $a_t = \pi_t(s_t)$ - the policy - optimise an expected aggregate of future rewards



- Rarely the observation o is the state s, the world state is, but often we assume it is!
- Generally POMDP - today MDPs!

POMDPs and non stationarity

- POMDP generally P-Space hard (not on average)
- To find a proper state we have to solve the additional prediction problem $s_t = h_t(o_t, a_{t-1}, o_{t-1}, a_{t-2}, o_{t-2} \ldots)$
- In the non-stationary, finite horizon formulation the MDP has the form $(S, A, \{P\}_h, \{r\}_h, H, \rho_0) \Rightarrow$ Value-functions $Q_h(s, a)$ get time depended $\Rightarrow$ similar form of Bellman equations
- We can incorporate time into state e.g. $\tilde{s} = (s, h) \Rightarrow$ standard MDP
- Generally Bellman equation nice in discounted, stationary formulation $\Rightarrow$ this is what we usually see and most libraries build on this formulation

Remarks on MDPs

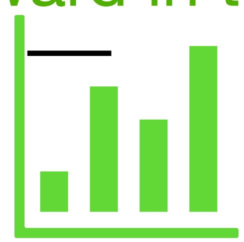
- Mainly we have POMDPs
  - ➡ Try to find a state which provides sufficient statistics to solve your problem (internal representation of the agent) - not world state
- Why is MDP so popular?
  - ➡ It always possible to make an MDP - by including sufficient information
  - ➡ What if not Markov?
    - What happens? Q-learning example
    - Montecarlo - No need for Markov assumption
    - History inclusion - RNNs, LSTMs
- Extreme: Bandits → no states (little knowledge about state)

The problem modelling

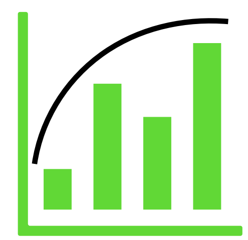
MDP - the ingredients

- MDP - discounted version: $(S, A, R, P, \rho_0, \gamma)$
  - ➡ $P(s', r \mid s, a) = \mathbb{P}(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a)$ - dynamics function
  - ➡ Mainly defined by system (and my state and reward definition)
- What do I want to solve? What's the objective function? How to reach the goal? The expectation of reward in the future!

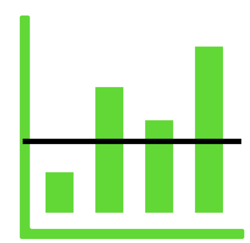
1. Finite horizon: $\max_{\pi} \mathbb{E}_{\pi}[\sum_{t=0}^{T} R_t]$



2. Infinite horizon: $\max_{\pi} \mathbb{E}_{\pi} \lim_{T\to\infty} [\sum_{t=0}^{T} \gamma^t R_t]$



3. Average reward: $\max_{\pi} \mathbb{E}_{\pi} \lim_{T\to\infty} \frac{1}{T}[\sum_{t=0}^{T} R_t]$



- Mostly: $\max_{\pi} \mathbb{E}_{\pi} \lim_{T\to\infty} [\sum_{t=0}^{T} \gamma^t R_t]$ - solution: stationary policy $\pi(s)$

RL and optimization

- All said falls into the domain of optimization:
  - ➡ An optimiser tries to find the arguments of a function to maximise the function value (optimization is greedy!)
  - ➡ RL algorithms look to find a mapping (the policy) from states to actions maximising the expected cumulative reward rather than just a single optimal function value
    - If parametric function approximation is used, we try to find the values of the parameters of the approximated function (either a value function or the policy directly) to obtain this mapping (this a classical optimisation problem).
- RL is comparable to calculus of variation (its origin is in classical mechanics - HJB equation) instead of function optimization

Optimization

maximise$_{\{A_i\}} \sum_{t=0}^{T} R(S_t, A_t, W_t)$

subject to: $S_{t+1} = f(S_t, A_t, W_t)$

RL

maximise$_{\pi_t} \mathbb{E}_{W_t}[\sum_{t=0}^{T} R_t(S_t, A_t, W_t)]$

subject to: $S_{t+1} = f(S_t, A_t, W_t)$

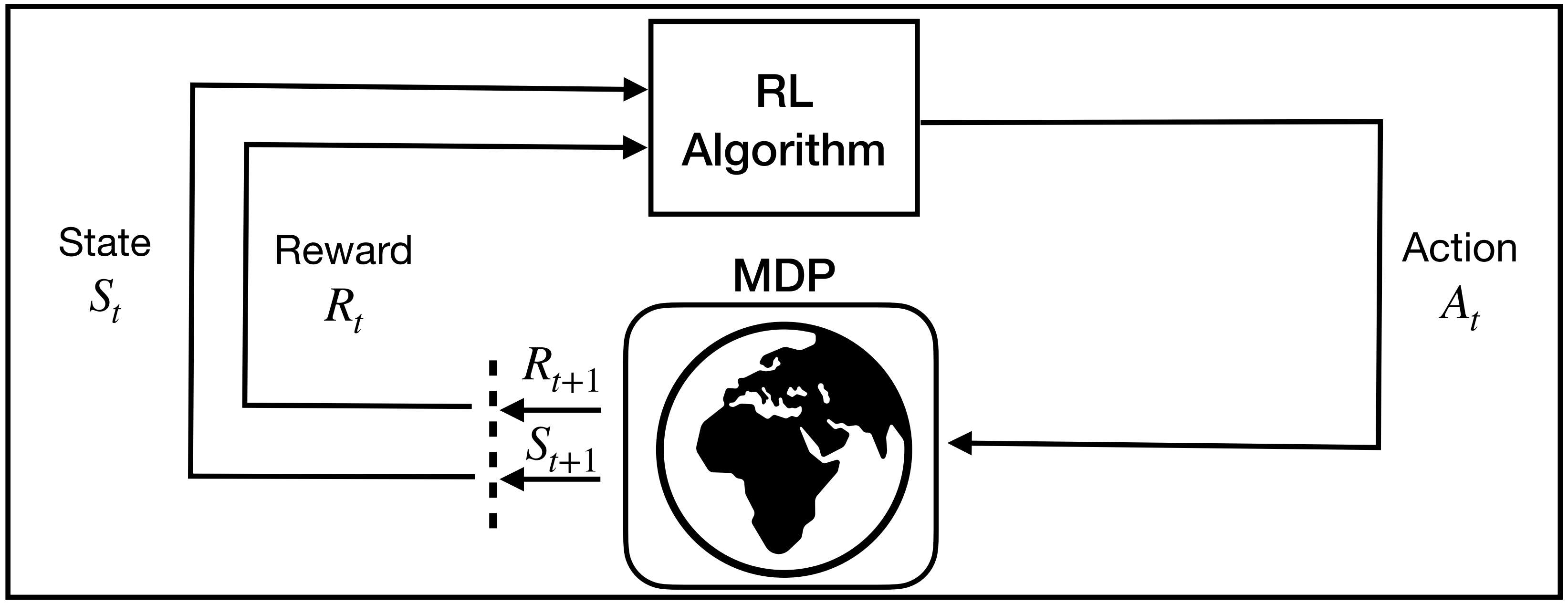
$\boxed{A_t = \pi_t(S_t, S_{t-1}, \ldots)}$

Feedback structure takes noise into account

Often optimization is performed only for one step horizon:

maximise$_a R(\,\cdot\,, a, W_t)$

Episodic training: we probe the system

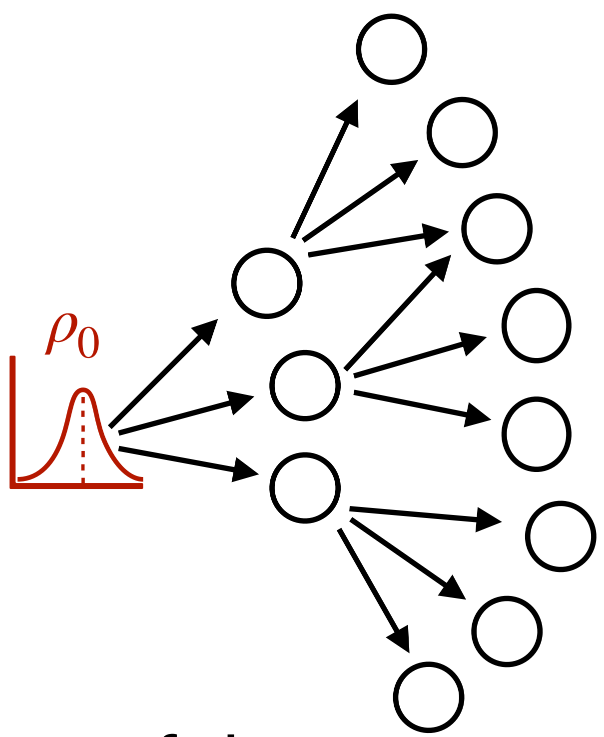


- The system generates noisy trajectories:
  $\tau_i = o_{0,i}, a_{0,i}, o_{1,i}, r_{1,i}, a_{1,i}, o_{2,i}, r_{2,i}, \ldots$
- From these probes we learn, but how?

Learning from episodes

- $(S, A, R, P, \rho_0, \gamma)$

- Rarely we have a full online learning problem
- The problem either is naturally episodic or we train in an episodic manner and are reset (adding absorbing state):
$\tau_i = o_{0,i} \sim \boxed{\rho_0}, a_{0,i}, o_{1,i}, r_{1,i}, a_{1,i}, o_{2,i}, r_{2,i}, \ldots$
- Design the episodic training:
  - ➡ Is it an infinite horizon problem → stability forever?
  - ➡ Is it a finite horizon problem with stationary dynamics?
- What is the role of ρ_0?
- What is the role of a finite maximum length?
- Exploration (finite time in infinite problem) - not part of the problem



Reward design and shaping

- $(S, A, R, P, \rho_0, \gamma)$

- **The reward includes the goal and how to reach the goal!**
- There is an equivalence class of problem formulations leading to the same goal → differences in algorithmic efficiency
  - ➡ Example: Negative/Positive/Normalised Reward
- **Generally probabilistic:**
  $$P(r \mid s, a, s') = \sum_{s'} P(r, s' \mid S_t = s, A_t = a)$$
- Can be formulated in dependence of s and a or given as a direct feedback signal
- To improve exploration or to solve sparse reward problems - reward shaping during training!

The discounting γ - a hyperparameter?

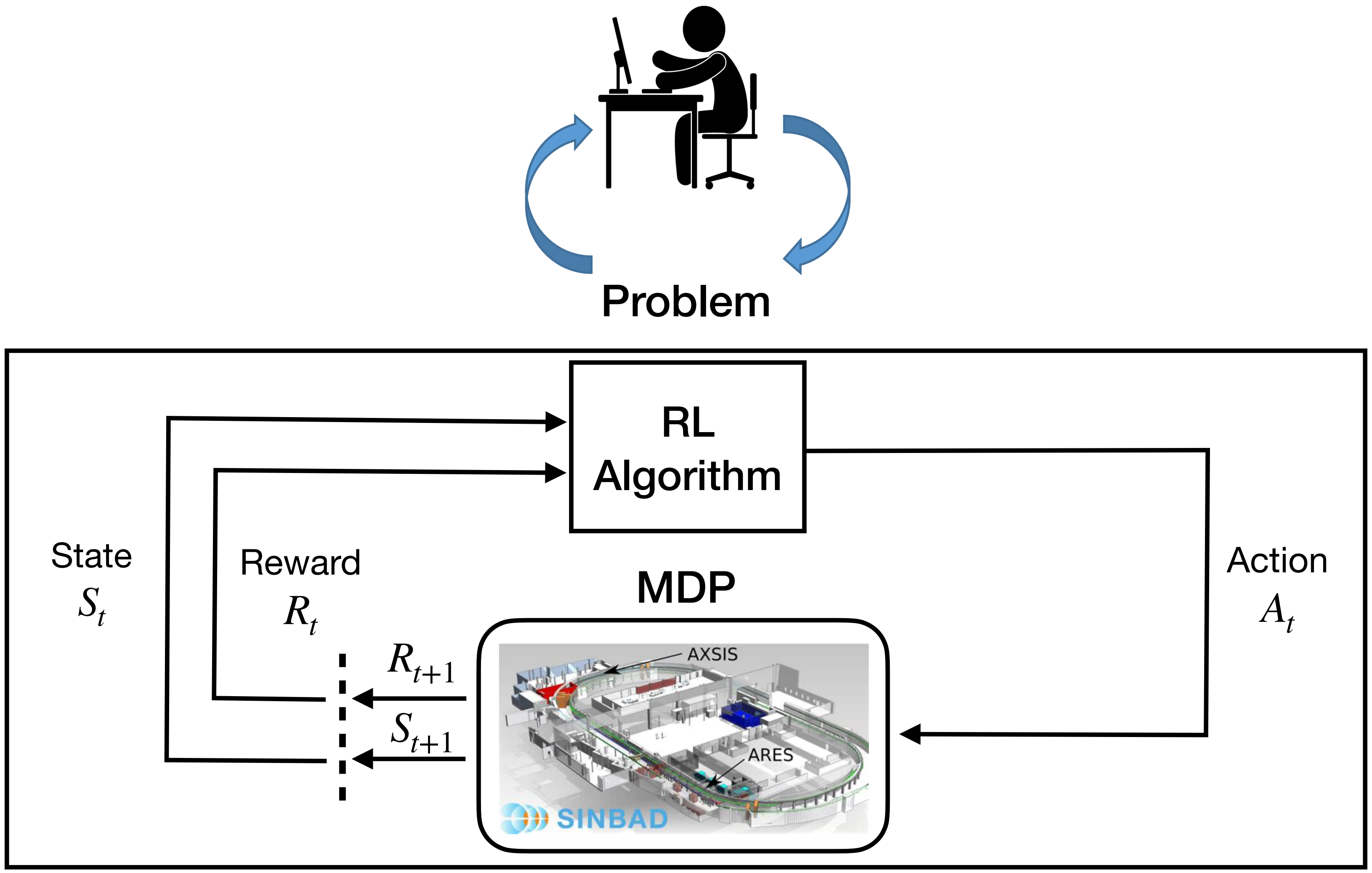
- $(S, A, R, P, \rho_0, \gamma)$

- Generally we don't need discounting
- Introduced due to mathematical convenience: convergence of cumulative sum of rewards as alternative to mean reward: $\sum_t^N \gamma^t R_t \leq \frac{1}{1-\gamma}$, when R_t is bounded in [0,1]
- Can be used influence training performance
- <u>Not needed in naturally episodic problems!</u>

Summary

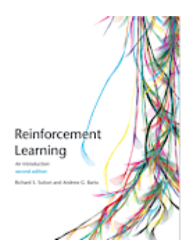
- Capture the right problem
- Formulate the MDP appropriately
- (Problem equivalent) Design has impact on RL algorithm
- Problem I solve = Designed MDP + Reward objective

The entire problem



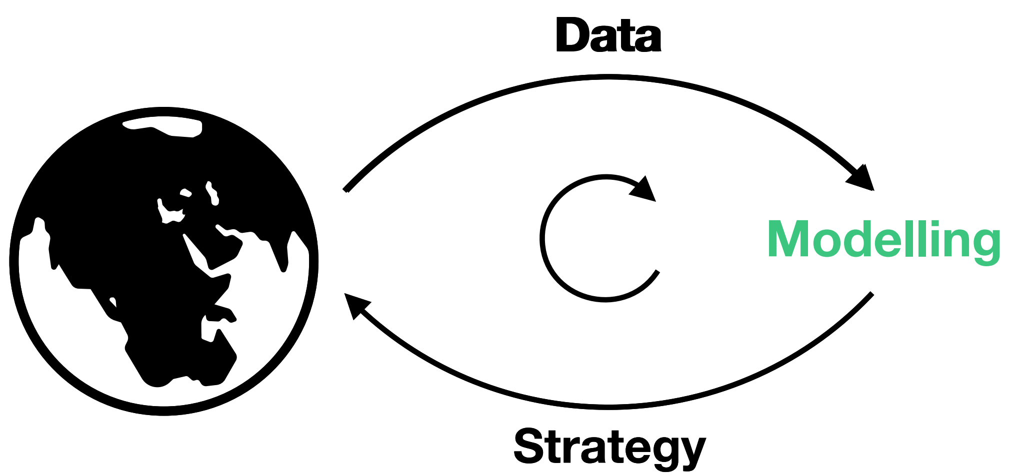
About Machine Learning

SB: 9



RL and decision theory

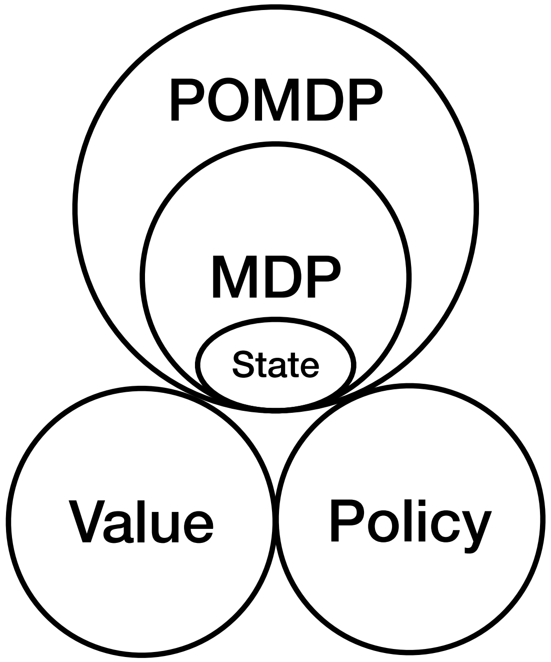
Information → decision → Information → decision → Information → ...



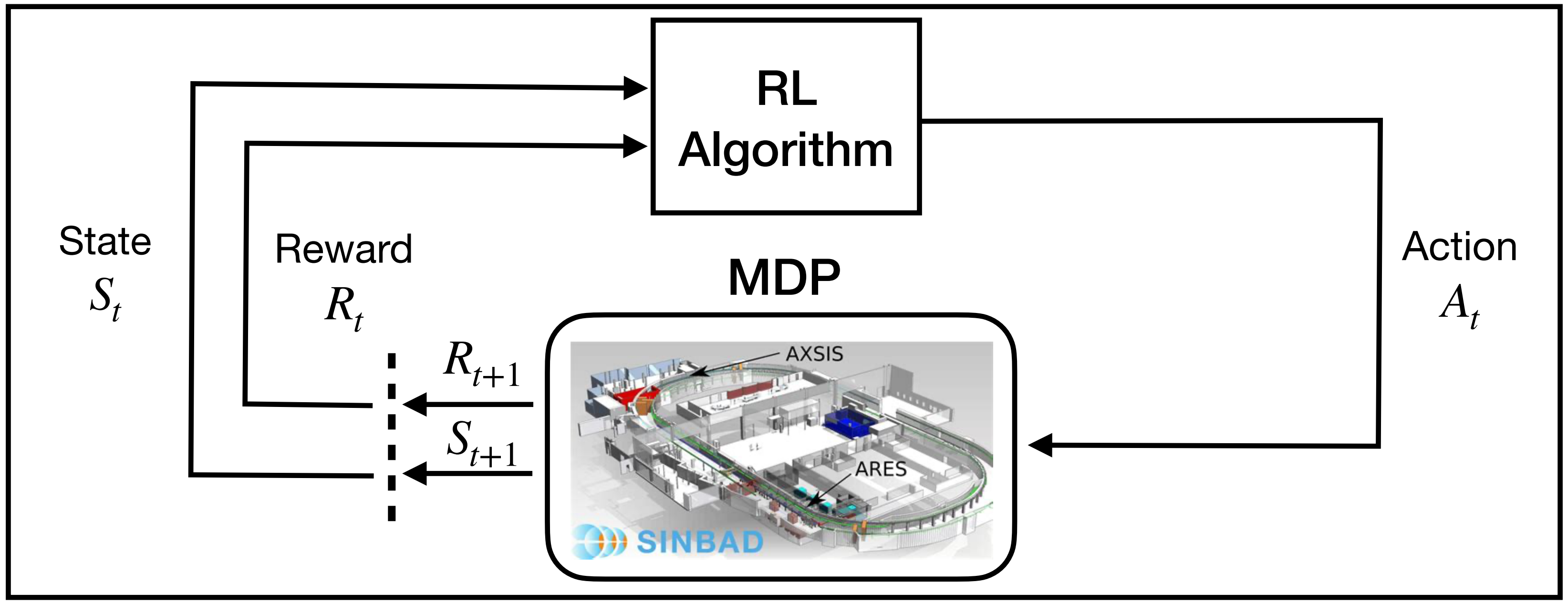
- One step horizon offline RL ⇒ Prediction $\mathbb{P}(Y_i \,|\, X_i)$ - pattern recognition or supervised learning (SL)
- One step horizon RL ⇒ active Learning - e.g. system identification
- RL is a multi step **optimization** problem - we learn about the dynamics of the world

Prediction

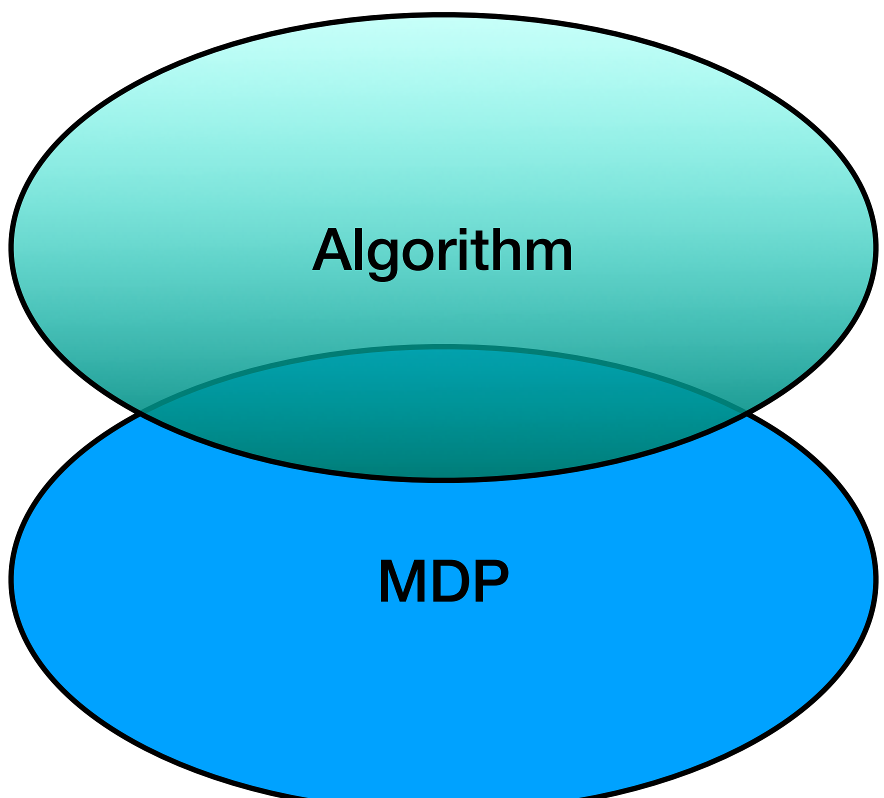
- Function approximation (FA):
  - ➡ Parametric - compact approximation of a function using a parametrised representation $f(x) \approx \hat{f}(x, \bar{\theta})$, where $\bar{\theta}$ are parameters to be adapted
    - Fixed representational power
    - Constant computationally complexity - fixed set of parameters
    - Example: Artificial neural networks (ANNs), linear approximations…
  - ➡ Non-parametric - memory based:
    - $f(x) \approx \hat{f}(x, \underbrace{\mathscr{D}}_{\text{Data}}) = \sum_{x' \in \mathscr{D}} \underbrace{k(x, x')}_{\text{Kernel}} \underbrace{g(x')}_{\text{Weight}}$
    - No fixed representational power
    - Parameters are not learned directly
    - Computationally complexity grows with data
    - Example: Gaussian processes, Kernel-based methods,…



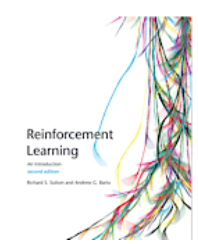
Problem



Solving the problem



SB: 6, 8, 9,10,11,13



Most common issues RL - algorithmic side

- Sample efficiency
- Stability
- Run time
- Hyperparameter tuning
- Exploration

Remarks on RL

- Trial and error
- Only rewards (labels of visited state, actions)
- Policy decides what we learn - usually censored
- Only valid estimates of things sufficiently learned

Special cases

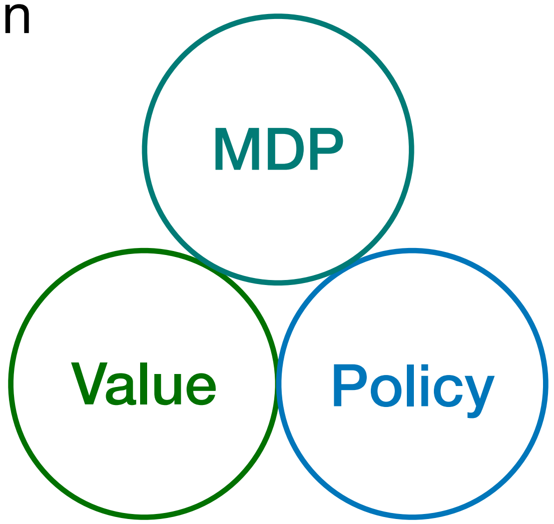
- We know $P(s', r \mid s, a)$ and the MDP is finite - frame work of dynamic problem can be used (Bellman update is exact):
  - ➡ Good theoretic properties
  - ➡ Playground of classical control theory
  - ➡ If large, we use sampling: approximative dynamic programming
- If $P(s', r \mid s, a)$ is known, the dynamics is linear and we design a quadratic dependence of s and a of the reward: analytical solutions (the popular Linear Quadratic Regulator -LQR). Stationary dynamics → Bellman → Ricatti equation - static state feedback.
- Alternative solution methods?
  - ➡ E.g. linear programming

Hidden Markov Models - POMDPs

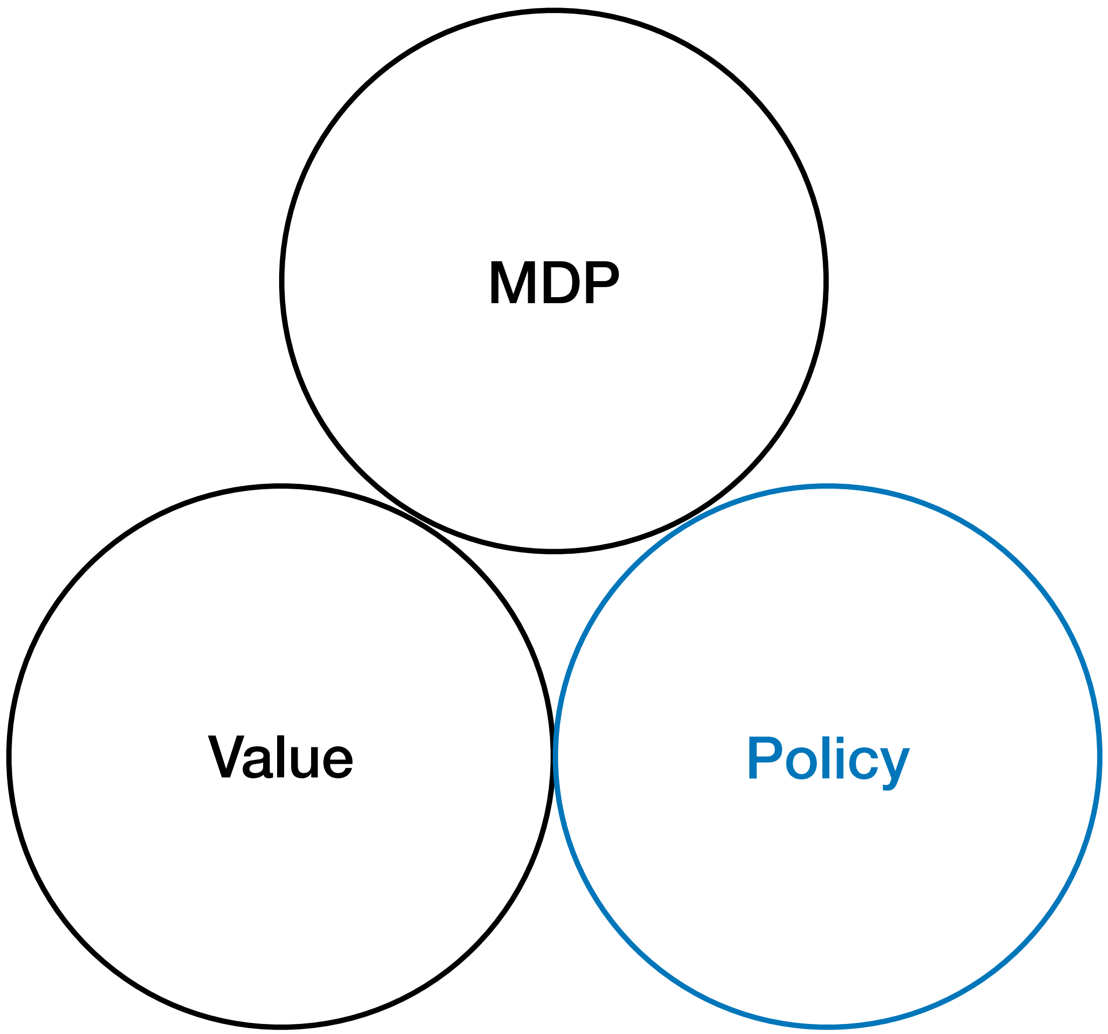
- Linear POMDP: believe state - $O_t = h_t(S_t, A_t, W_t)$
  - ➡ Static output feedback is NP hard (linear in O and dynamics)
  - ➡ General POMDPs are PSPACE hard
- There are ways out - separation principle:
  - ➡ Filtering $\hat{s}_t = f(\{o_t\})$ - prediction problem
  - ➡ Action based on certainty equivalence
  - ➡ Optimal filtering - if dynamics are linear and noise is Gaussian - Kalman filtering - general belief propagation - LQG
  - ➡ Kalman filtered state - duality between estimation and control
  - ➡ Estimate state with prediction $S_t = h(\tau_t)$, τ_t are time lags

General cases

- $P(s', r \mid s, a)$ unknown → approximative methods
- MDP finite: approximative dynamic programming
- General MDPs: Continuous A, S spaces
    - ➡ Value function approximated with FA
    - ➡ Policy approximated with FA
    - ➡ Model of the MDP (learn from data or from simulator) use certainty equivalence
    - ➡ Trained in a stochastic fashion



Policy based



Direct policy search

- RL as derivative free optimization:
  - ➡ maximise$_{z \in \mathbb{R}^d} R(z) \Rightarrow$ maximise$_{p(z)} \mathbb{E}_p[R(z)]$
  - ➡ Parametrise a distribution $p(z;\theta) \Rightarrow$ maximise$_{p(\theta)} \mathbb{E}_{p(z;\theta)}[R(z)]$
  - ➡ Likelihood trick - estimate the derivative:

$$\nabla_\theta J(\theta) = \int R(z)\,\nabla_\theta p(z;\theta)dz = \int R(z)\frac{\nabla_\theta p(z;\theta)}{p(z;\theta)}p(z\ \theta)dz$$

- $$= \int R(z)\,\nabla_\theta \log p(z;\theta)p(z\ \theta)dz = \mathbb{E}_{p(z;\theta)}[R(z)\,\overset{\text{Score function}}{\boxed{\nabla_\theta \log p(z;\theta)}}]$$

- Unbiased gradient estimate of J, if sample efficiently from $p(z;\theta)$ and $\log p(z;\theta)$
- High variance

Probabilistic trajectories

- Objective if episodic: $J(\theta) = V^{\pi_\theta}(s_0) := V(\theta)$
  - ➡ Stochastic search: pure random search, Simplex, Bayesian optimization
- Using the gradient:

$$V^\pi(s_0) = \mathbb{E}_\pi[\sum_t \gamma^t R_{t+1} \,|\, S_t = s_0]$$

➡ $$V(\theta) = \sum_\tau \underbrace{\boxed{P(\tau;\theta)}}_{\text{Trajectory probability}} \underbrace{\boxed{R(\tau)}}_{\text{Trajectory reward}}$$

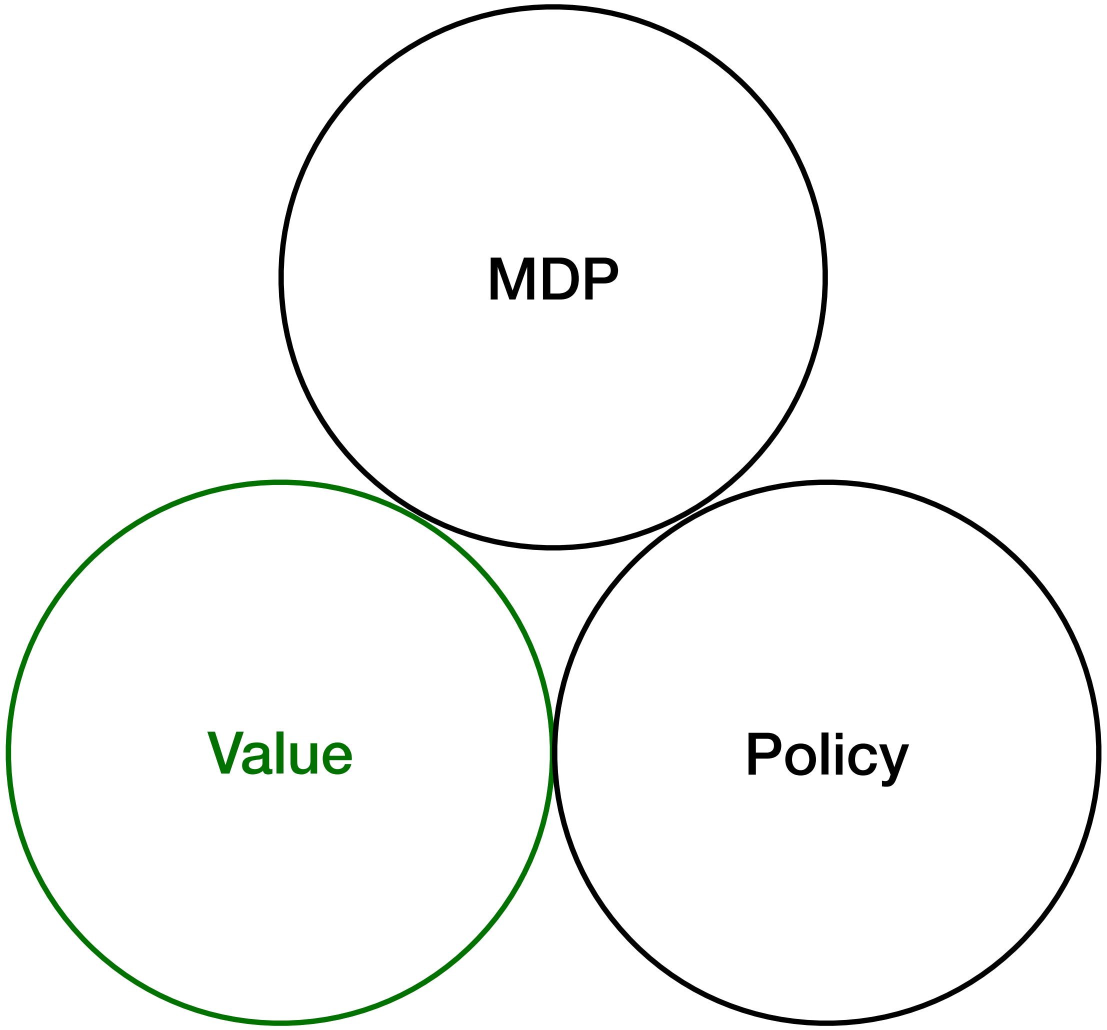
➡ $$\nabla_\theta V(\theta) = \sum_\tau P(\tau;\theta) R(\tau) \underbrace{\boxed{\nabla_\theta \log P(\tau;\theta)}}_{\text{Log likelihood trick}} = \underbrace{\mathbb{E}[R(\tau)\nabla_\theta \log P(\tau;\theta)]}_{\text{Stochastic gradient}}$$

➡ Sampling of $A_t \sim p(\cdot \,|\, \tau_t;\theta)$

- Handle probabilistic policies (example)
- High dimensional and continuous action spaces
- Reinforce algorithm considers temporal structure

➡ Finite difference approximation $\hat{=}$ Reinforce algorithm

Value based



The value-function is introduced to compare policies

Basis of Q-Learning - Temporal difference (TD) learning

- If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference learning. (Sutton and Barto, p.113)
- $V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t} \gamma^{t} R_{t+1} \,|\, S_t = s] = \mathbb{E}_{\pi}[R_{t+1} + \gamma V(s') \,|\, S_t = s]$
- Estimated via sampling:
- $\hat{V}(S_t) \leftarrow \hat{V}(S_t) - \underbrace{\alpha}_{\text{Learn rate}} [\overbrace{\boxed{R_{t+1} + \gamma \hat{V}(S_{t+1}) - \hat{V}(S_t)}}^{\text{TD(0) error}}]$

Bootstrapping = estimating from estimator

- Can be used in episodic an non-episodic scenarios
- Immediate update of the estimator
- If probabilities known - exact update

Q-learning - issues

Bellman equation:

$$Q^*(s,a) = \mathbb{E}[R_t + \max_{a'} Q^*(s',a')]]$$

Bellman-operator is a contraction operator (L^2norm) - converges to a fixed point

Here - stochastic approximation:

$$\hat{Q}(s,a) \approx \hat{Q}(s,a) + \alpha[R_t - \max_{a'} \hat{Q}(s',a')]$$

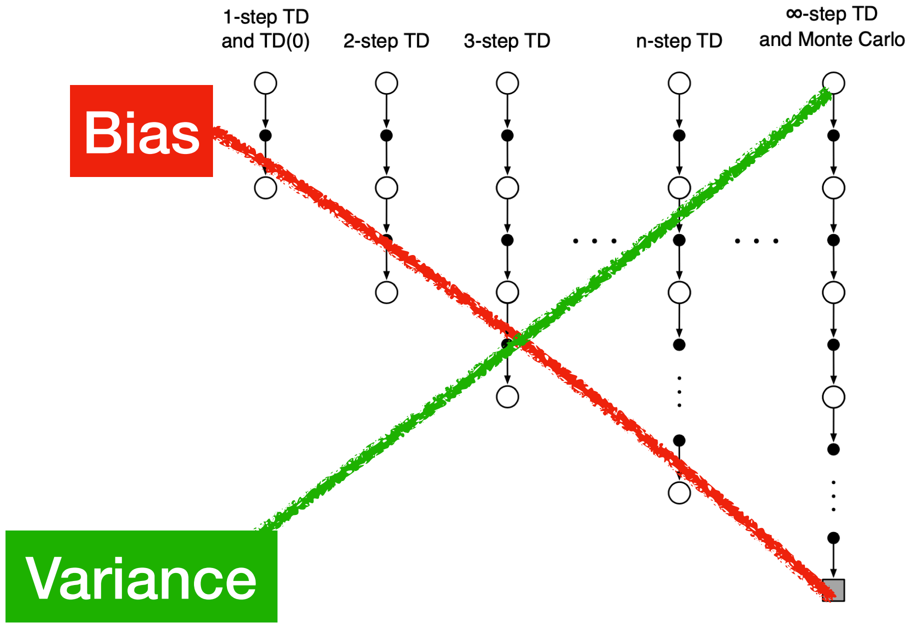
- $\pi(s) =$ argmax$_a Q(s,a)$
- Two immediate consequences:
  - ➡ Maximisation bias (expectation and maximisation don't commute)
  - ➡ Bias through bootstrapping
- Inefficient update (compare to e.g. SL)
- If FA is used contraction property might lost
- Might diverge - **deadly triad**:
  - ➡ FA
  - ➡ Bootstrapping
  - ➡ Off-policy

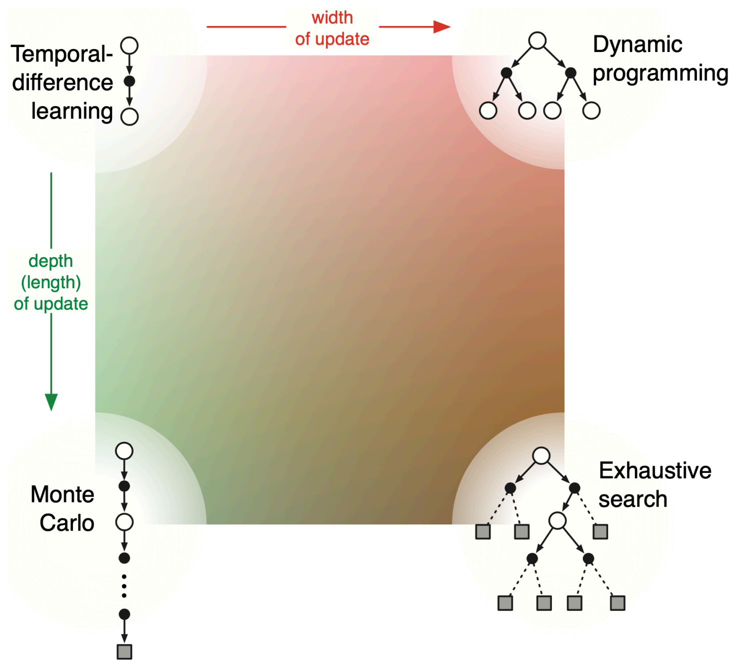
Deep approximate dynamic programming

- Value dominated
- Tries to mitigate maximisation bias (double networks)
- Stabilises training of networks trough tricks (random batches from replay buffer, target network, action noise) - (DDPG, TD3)
- More recent: distributional learning (D4PG), truncating trajectories (TQC)
- …

Modern algorithms

- Interplay of Bias and Variance
- Policy dominated: add baseline - the critic!
- DDPG: maximisation operator in Bellman equation is approximated - the actor!





From Sutton and Barto

Modern landscape

- Bias-Variance trade-off
- Regulated via Policy based and Value based methods
- Policy gradient regulated via update-length of value function

Value vs policy dominated

Value based relies on Bellman updates

Policy based - stochastic optimisation

Actor/Critic

Make more stable!

- Biased
- Can be offline
- Brittle
- No guarantees
- Deterministic ($\pi(s) = \text{argmax} Q(s, a)$)
- Faster
- DDPG, TD3, DQN and derivatives…

Reduce variance!

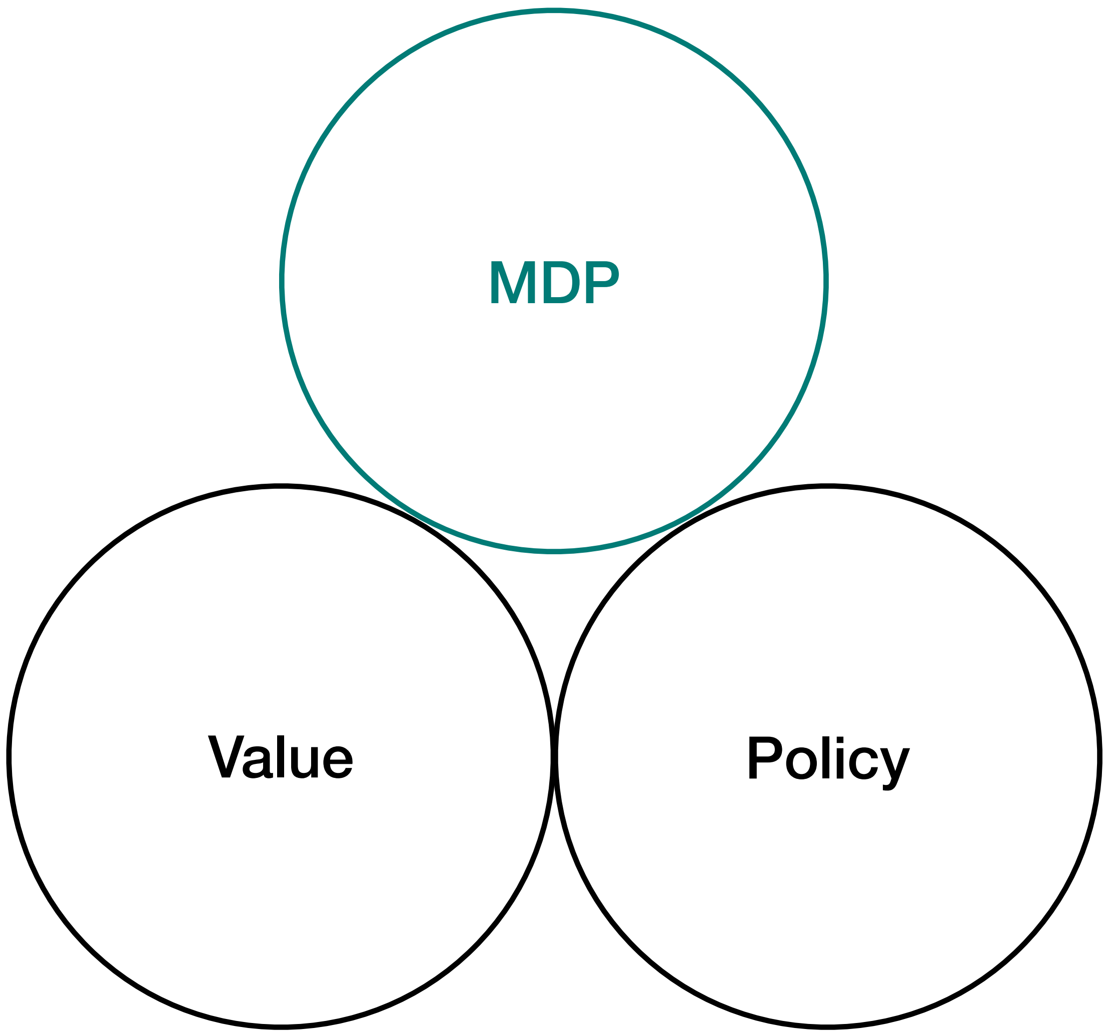
- High Variance
- Generally online
- Performance guarantees, stable
- Local optima
- Stable
- Allows for stochastic policies
- TRPO, PPO

- SAC - Soft Q learning - stochastic
- D4PG - Distributional Q-Learning
- MPO

Summary

- Value based: Approximate bellman updates - biased
- Policy based: high variance but stable
- Modern actor critic algorithms: bias-variance trade-off

Model of the MDP



Separation heuristics

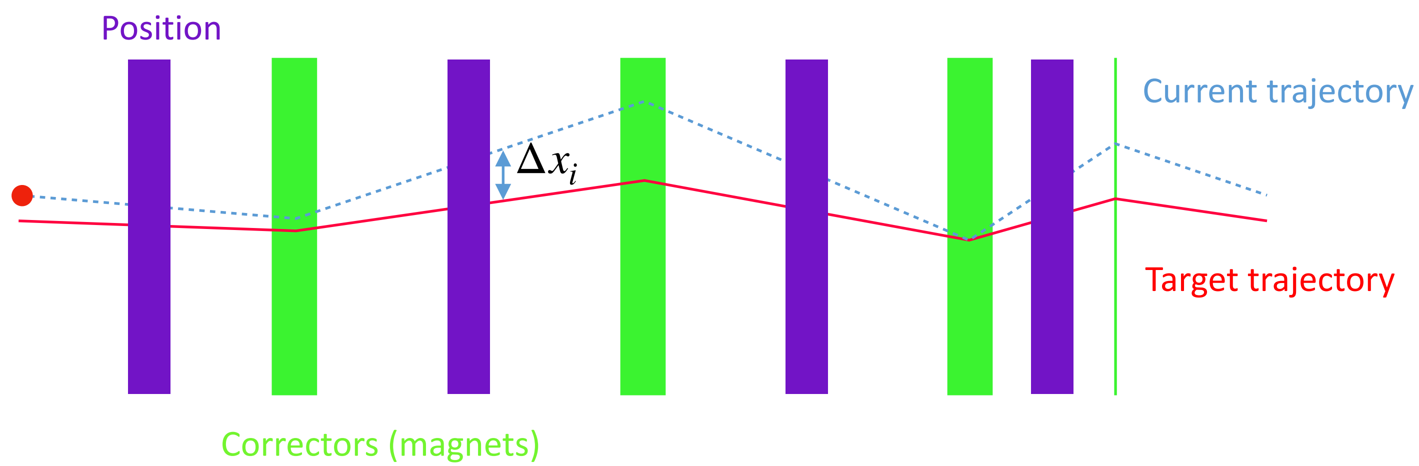
Modelling the MDP

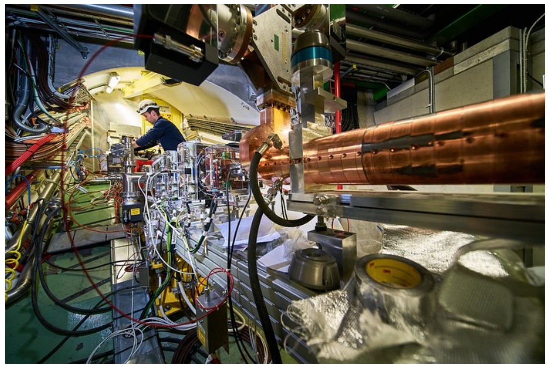
- We can have a **simulation** $\hat{P}(r, s' | s, a)$ which models the real (true) MPD $P(r, s' | s, a)$
- We can learn the MDP from real data using FA
- During training the RL algorithm we use $\hat{P}(r, s' | s, a)$ as it would be $P(r, s' | s, a)$ and hope to solve the problem (certainty equivalence)
- We can use a mixture of both
- Deal with consequences, try to learn $\hat{P}(r, s' | s, a)$ as accurate as possible (system identification)

Train a good MDP

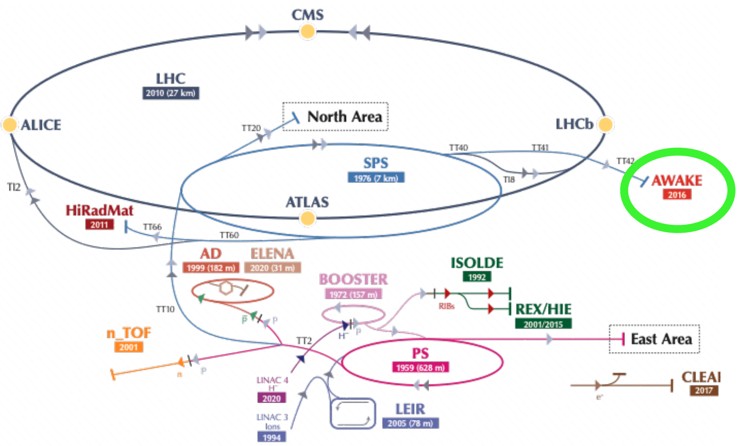
- What degrees of freedom to excite to learn the control problem?
- Model the uncertainties appropriately
    - ➡ Learn a model capturing the epistemic and aleatoric uncertainty (the noise)
    - ➡ Take the epistemic uncertainty appropriately into account
    - ➡ Examples: ensembles of ANNs, Bayesian ANNs, GPs…
- Having a model allows for planning
    - ➡ Horizon is critical!
    - ➡ Safety constrains can be taken into account

AWAKE Simulation Benchmark





- $R = -\left(\sum_i \Delta x_i^2\right)^{\frac{1}{2}}$
- 10 magnets $= \{k_1, k_2, \ldots k_{10}\}$
- 10 positions $= \{\Delta x_1, \Delta x_2 \ldots \Delta x_{10}\}$



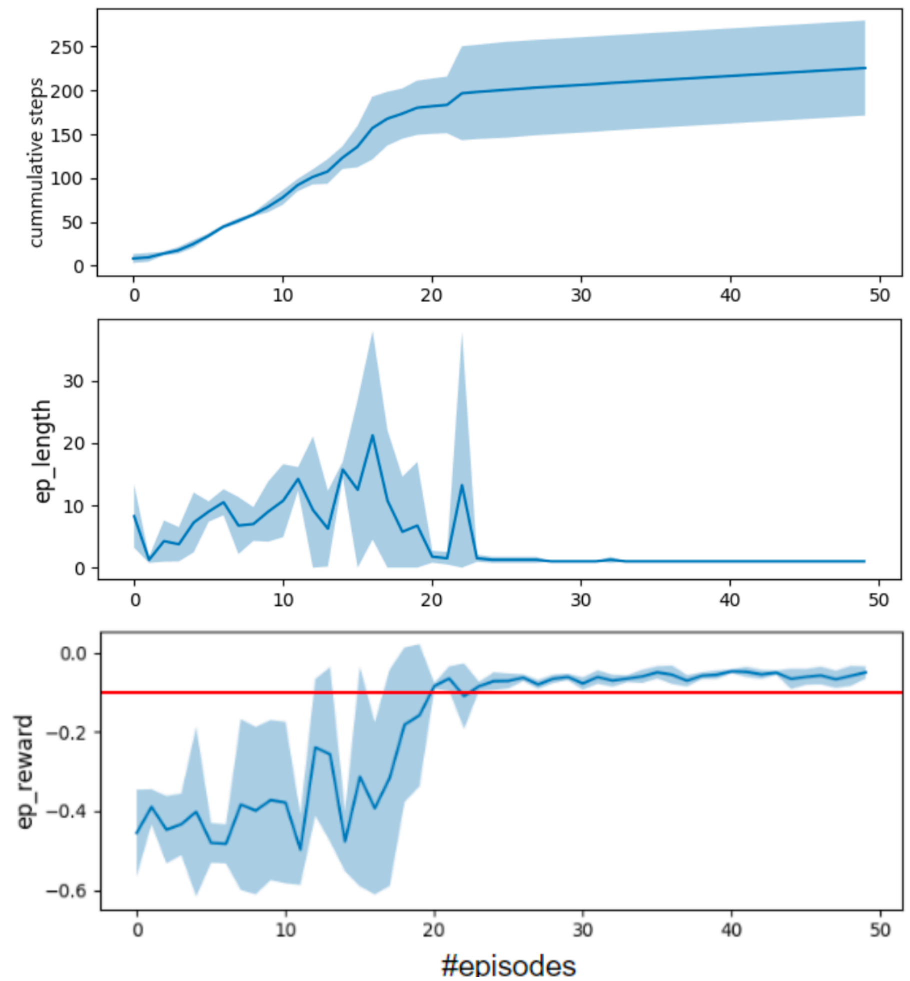
Target: trajectory steering - correct the trajectory in as little steps as possible.

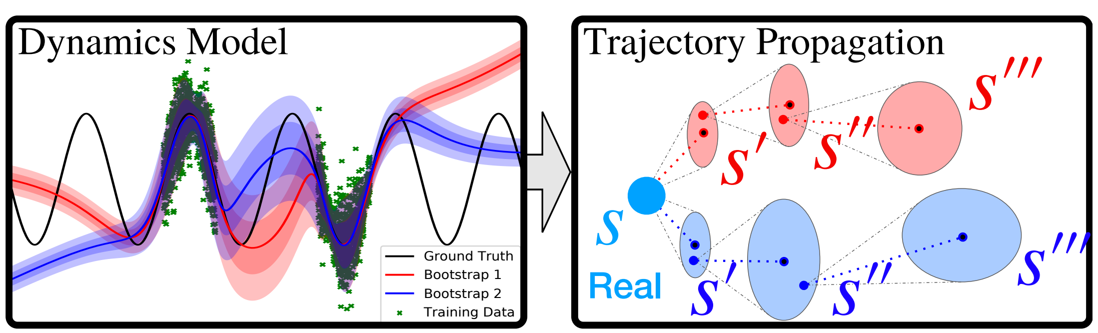
Example: Model based policy optimization

- Dyna style algorithm
- Model: Bayesian ANN bootstrapped ensemble
- RL algorithm: SAC
- Short roll-outs from real interactions
- Monotonic behaviour
- Many hyper-parameters

https://arxiv.org/abs/1906.08253

AWAKE simulation

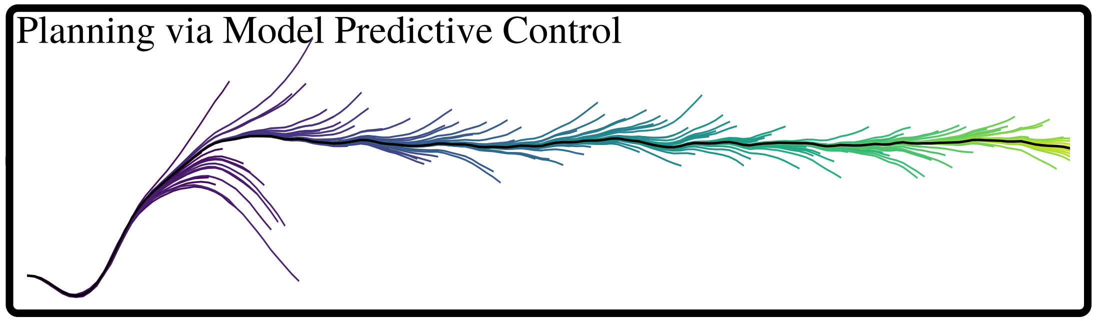




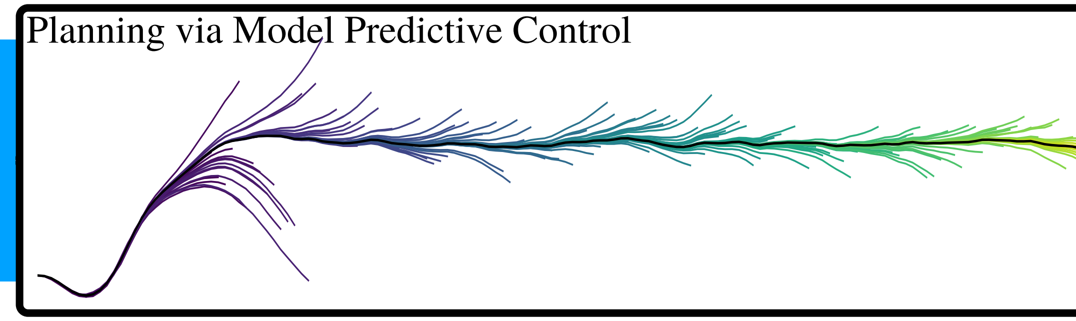
Adapted from https://arxiv.org/abs/1805.12114

Example: Model predictive control (MPC)

- If convergence quickly - no long term planning needed - solve infinite horizon problem with short term planning (open loop)
- Plans action sequence (optimisation) and takes first action only then replan
- Might retrain MDP model each step



Example: MPC



- Solves the infinite optimization problem (W_t=random variable):

$$\text{maximise}_{\pi_t} \mathbb{E}[\sum_{t=0}^{\infty} R_t(S_t, A_t, W_t)]$$

Optimise finite horizon

Final cost performance and robustness

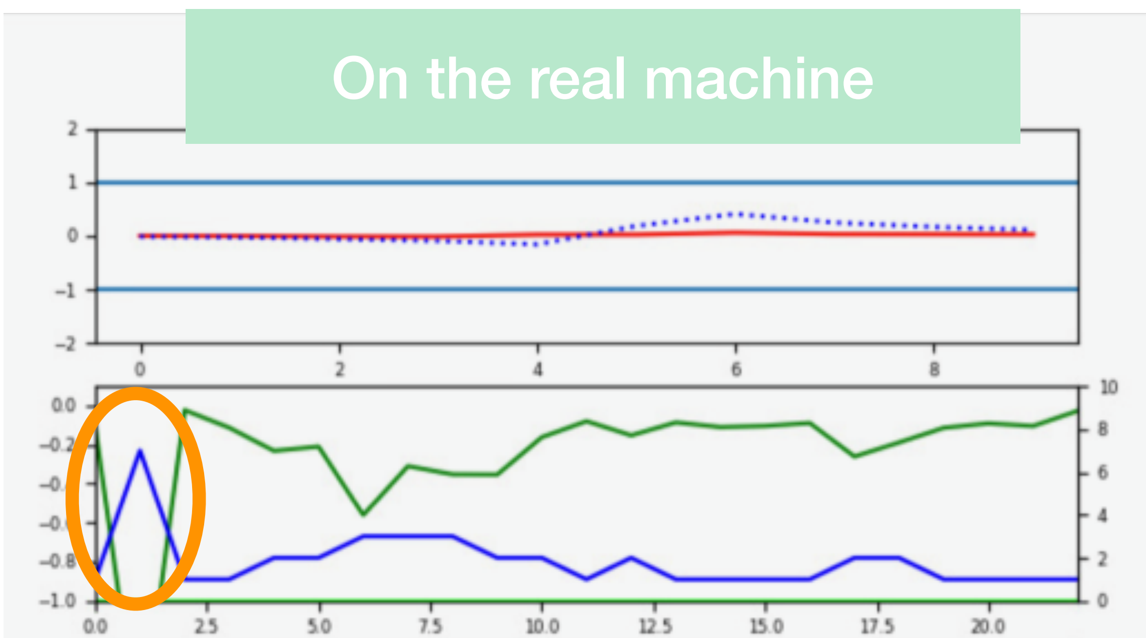
$$\approx \text{maximise}_{\pi_t} \mathbb{E}_{W_t}[\boxed{\sum_{t=0}^{T} R_t(S_t, A_t, W_t)} + \boxed{V(S_{T+1})}]$$

- subject to: $S_{t+1} = f(S_t, A_t, W_t)$

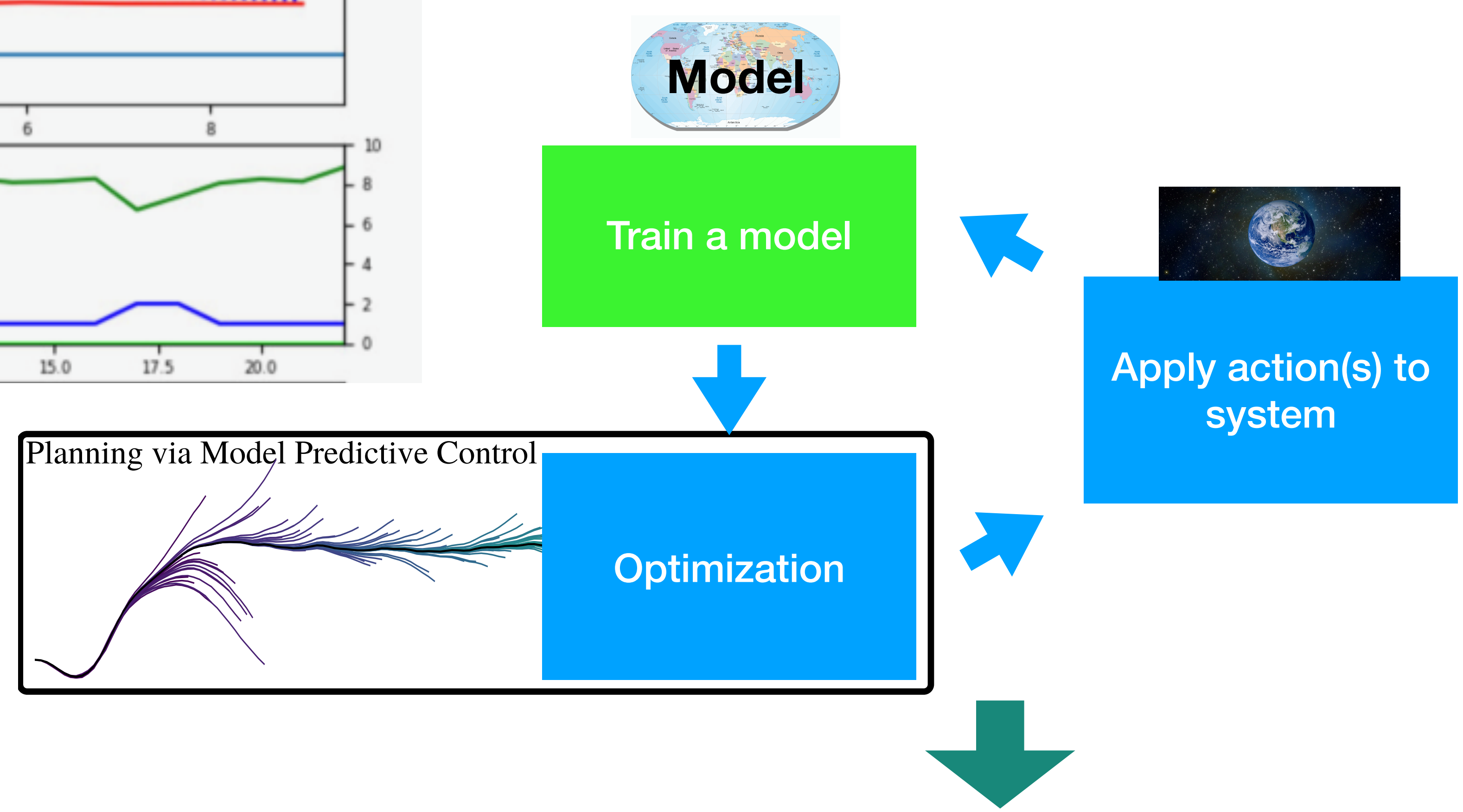
$$A_t = \pi_t(S_t, S_{t-1}, \ldots), \qquad S_0 = s$$

MPC with Gaussian Processes

- Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control
- Few shot RL learning on AWAKE



Shown at IPAC 2023 - Ultrafast Reinforcement learning demonstrated on AWAKE



Model (GP)

Based on: https://arxiv.org/abs/1706.06491v2

Summary

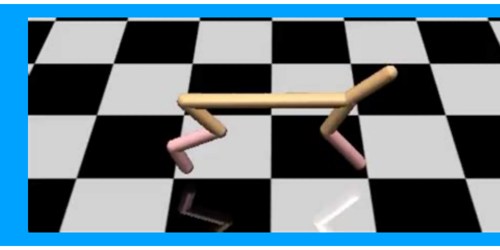
- Modelling the MDP - certainty equivalence
- Modelling epistemic and and aleatoric error
- Short horizons and feedbacks avoid model bias
- With a model safety concerns can be considered

General comments on RL

Exploration/Exploitation

- You only can estimate things you have sufficiently learned!
- Finite MDP: ε - greedy, what does this mean?
- Gaussian noise - continuous actions
- Boltzmann exploration
- Theory: Bandit algorithms to study trade-offs (Book)

Sample-Complexity



Evolutionary/ gradient free

Derivative free methods:

(NES, CMA,..)

10 x Online methods

On-policy Policy gradient

(A3C)

10 x Policy-gradient methods:

Actor-critic

(TRPO)

10 x Replay-Buffer Value function estimation

Off-policy Q-learning

(Q-Learning, DDPG, NAF, SAC,…)

10 x Model-based RL methods

Model-based

(MPO, Guided Policy Search, Dyna)

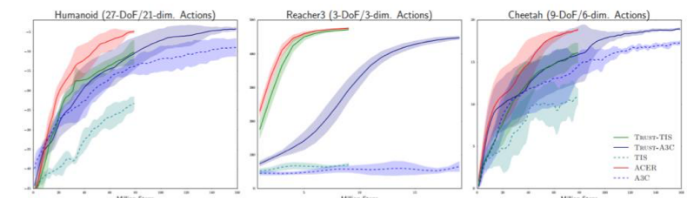
10 x Model-based shallow methods (no NNs)

Few shot GPs…

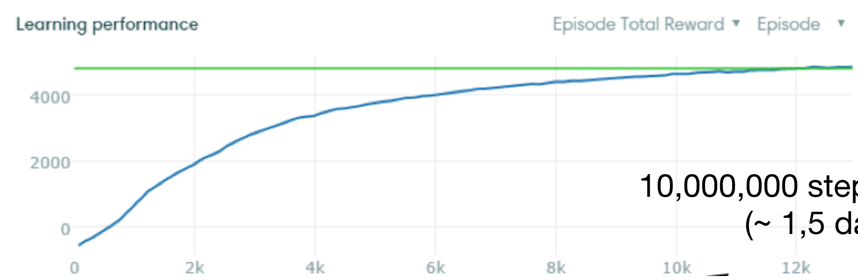
https://arxiv.org/abs/1703.03864

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

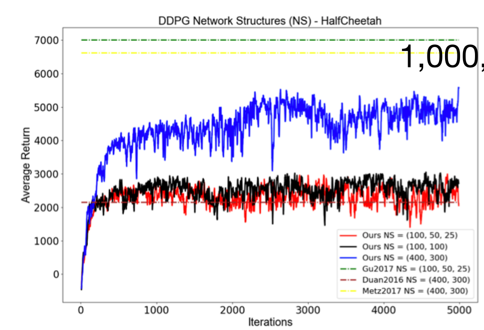
(~ 180 days real time)



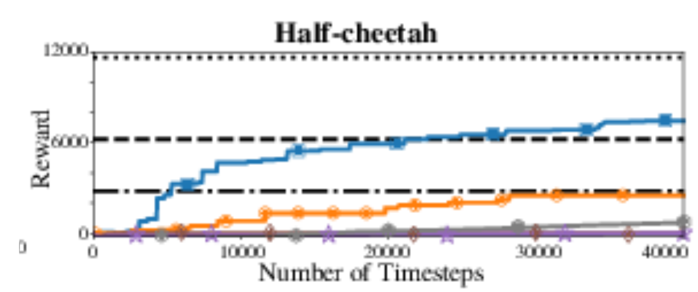
100,000,000 steps (100,000 episodes)
(~ 15 days real time)



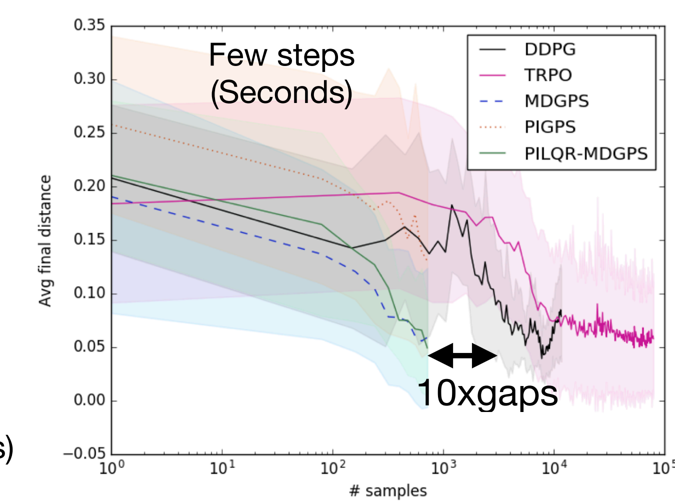
10,000,000 steps (10,000 episodes)
(~ 1,5 days real time)



1,000,000 steps (1,000 episodes)
(~ 3 h real time)



30,000 steps (30 episodes)
(~ 5 min real time)

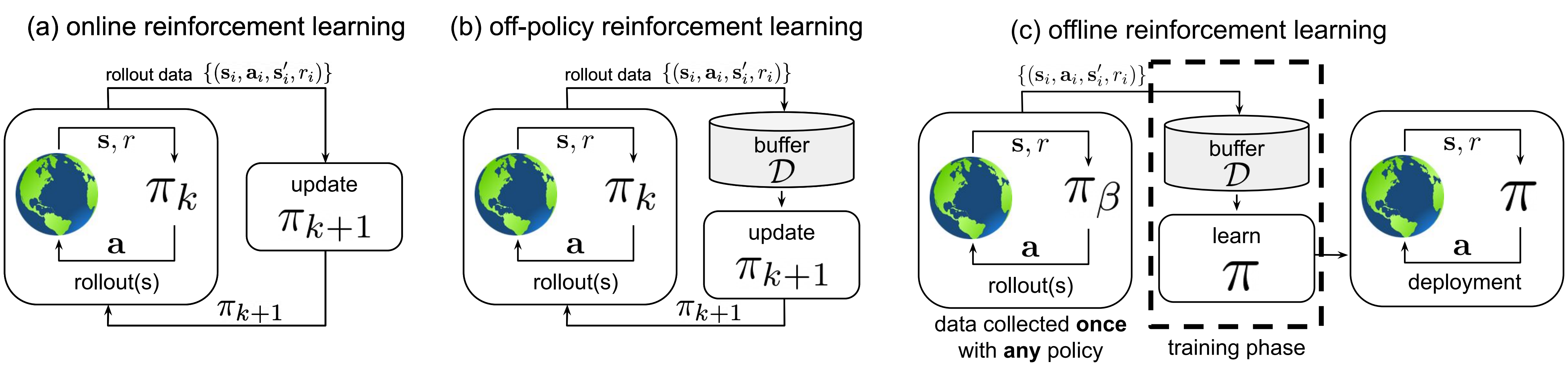


https://arxiv.org/pdf/1703.03078.pdf

Adapted from Sergey Levine

Use existing data?

- On-policy, off-policy, offline training



https://arxiv.org/format/2005.01643

Some training tips

- How to measure the performance of RL algorithms?
  - ➡ Take different seeds!
  - ➡ Return per episode on validation during training
  - ➡ Episode length
- How to set up your algorithmic environment?
  - ➡ Only start to implement, if there is no established library
  - ➡ Debugging is tricky in Monte-Carlo experiments, start simple and slowly increase complexity
  - ➡ More in our tutorial
- The solution is on <u>MDP + objective function</u>

Limitations

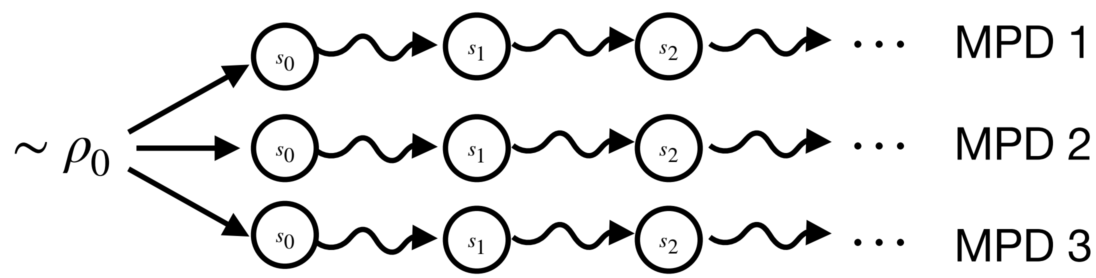
- Stability
- Safety
- Sample-efficiency
- Sufficient observability
- Interpretability
- ...

Summary

- Exploration/Exploitation
- Sample-efficiency
- On-policy, off-policy, offline RL
- Training
- Limitations

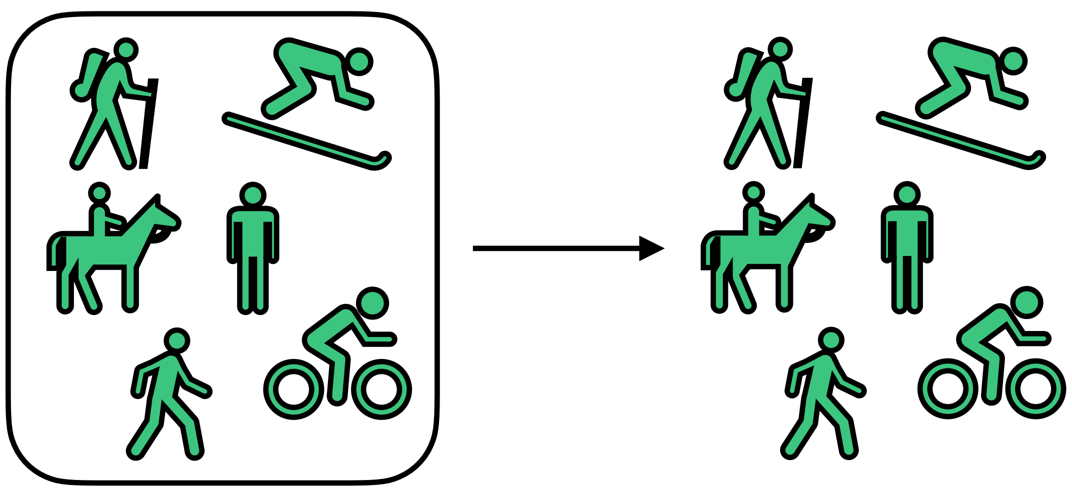
Advanced topics - outlook

Beyond the classical MDP

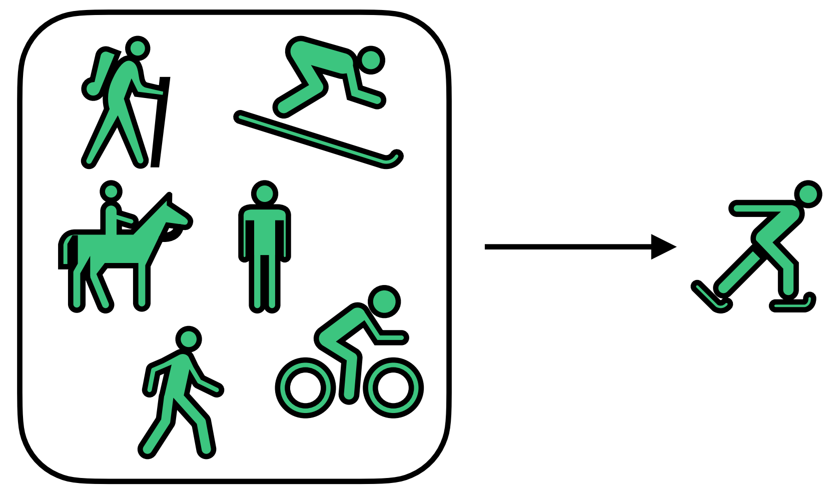


Contextual MDPs

Multi task RL



Meta RL

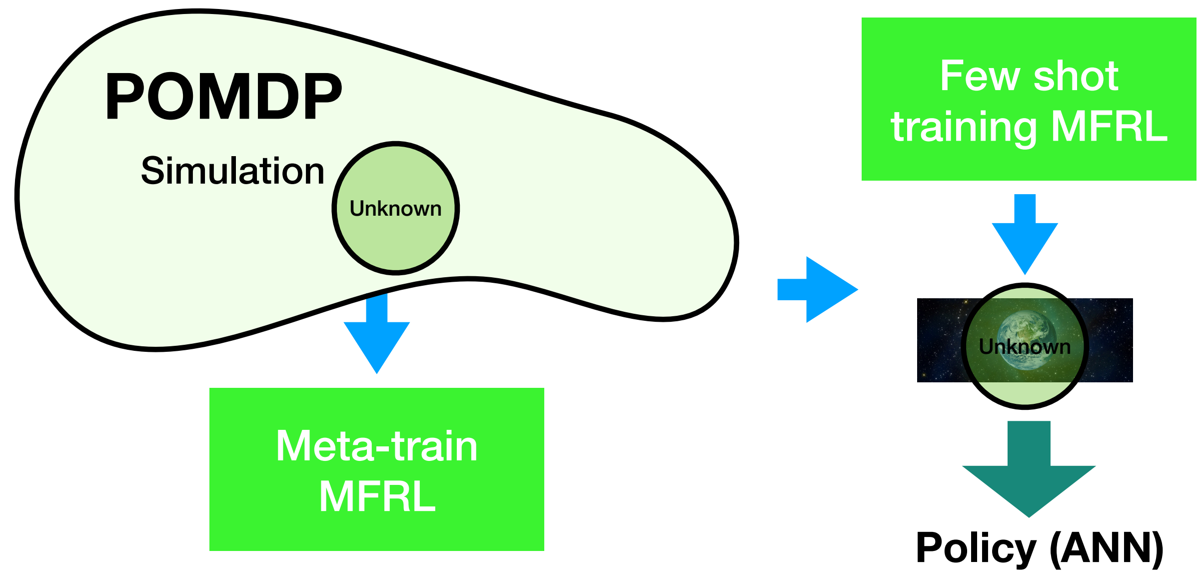


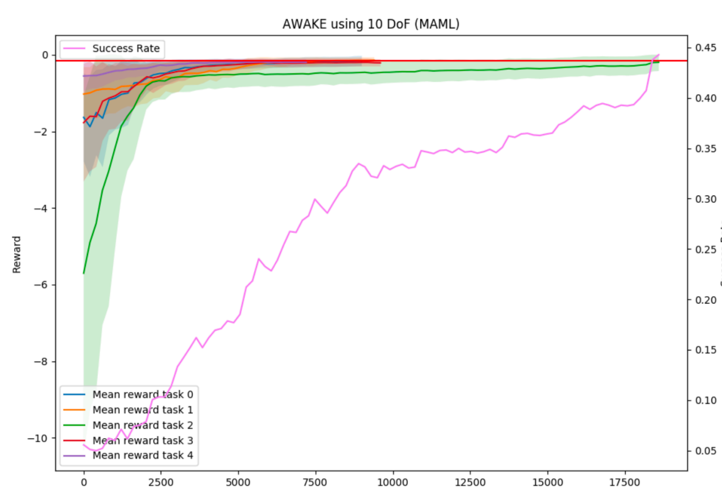
Common learning can improve sample efficiency

Accelerates learning enormously

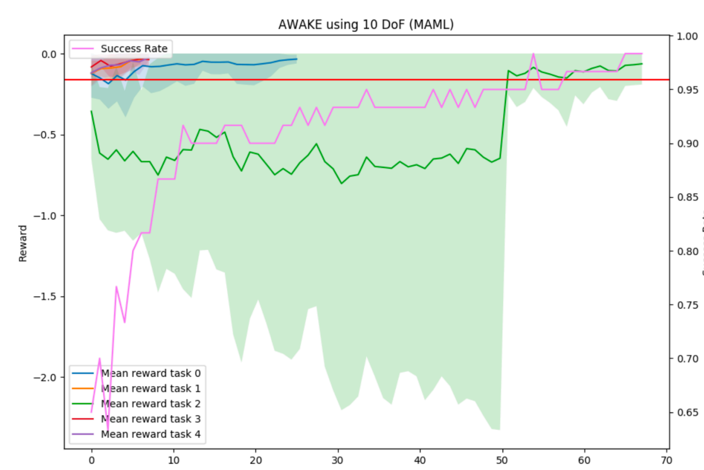
Example: Meta-RL

- Awake simulation, varying quads
- Few shot stable adaption
- TRPO with some guarantees





Untrained ~ 18000 samples 40% success



Meta-trained ~80 samples 100% success ~ **few steps on the machine**

Demonstrated in experiment AWAKE - to be published

Work with Lukas Lamminger Kain Verena

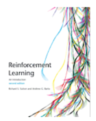
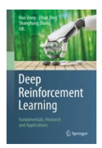
Further advanced topics

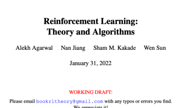
- **Meta RL**
- **Multi task RL**
- Contextual RL
- Multi-agent RL
- Hierarchical RL
- Distributional RL
- Inverse RL/Imitation learning
- …

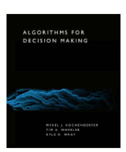
Summary

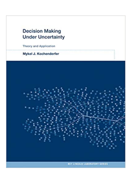
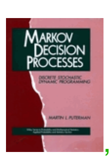
- Overview of designing correct problem
- Function approximation
- Different RL solution approaches: Value, Policy, MDP-model
- RL algorithmic challenges
- Beyond classical RL

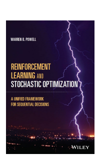
Selected Literature

- General:
    - ➡  Reinforcement Learning: An Introduction http://incompleteideas.net/book/the-book-2nd.html
    - ➡ 

 Deep Reinforcement Learning: Fundamentals, Research and Applications https://link.springer.com/book/10.1007/978-981-15-4095-0
    - ➡ 

 https://rltheorybook.github.io/
- POMDPs:
    - ➡ 

 Algorithms for Decision Making https://algorithmsbook.com/
    - ➡  Decision Making Under Uncertainty: Theory and Application: http://web.stanford.edu/group/sisl/public/dmu.pdf
    - ➡ 

,Markov Decision Processes: Discrete Stochastic Dynamic Programming: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470316887
    - ➡  Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119815068