Winter Term 21/22

Adversarial Self-Supervised Learning with Digital Twins

Lecture-3: Model-Based Reinforcement Learning

Prof. Dr. Holger Giese (holger.giese@hpi.uni-potsdam.de)

Christian Medeiros Adriano (christian.adriano@hpi.de) - **"Chris"**

He Xu (he.xu@hpi.de)

IT Systems Engineering | Universität Potsdam

Dyna-Q:
learning a model of the environment

Motivation

Algorithms are built with the assumption that we can obtain the true model of the environment.

This might not be true in many situations:

- data is non-i.i.d. (independent and identical distributed) data (auto-correlations)
- environment is non-stationary (change in distributions)
- multiple agents interact within the environment (interference)

Model-Based RL

Advantages:

- Domains where learning a value function is hard, e.g., domains with large action space, like chess. The model is straightforward (2 x 2 table with deterministic outcomes of actions)
- In these cases, a model is a more compact and more useful representation than a value-function or a policy
- Can efficiently learn a model using supervised learning
- Can reason about model uncertain
- Allows the agent to look ahead. To make decisions based on what the agent believes about how the environment works.

Disadvantage:

Two sources of approximation error:

1. learning the model then
2. constructing a value function

Intuition-1

What is the difference between learning a reward function vs a value function (as we did in q-learning)?

Difficult scenarios for learning a value function:

- Board games with an explosion of possible moves and configurations
- Maze that changes after each episode

Solution:

- learn a model of the environment
- use that to learn the value function

Learning a model = learning the transition probabilities

Intuition-2

Models of the environment:

- allow to retrieve state and actions that useful.
- avoid costly interaction with the environment

Conversely:

- Without a model we the experience that we learn is too tightly couple with the experience
- Model allows to keep a level of skepticism about this experience

Example:

- Sometimes we are in an area where I know nothing, so learning is very hard.
- Other times we are in an area where I know everything, so I have to go far to learn something new.

What is a model that we can learn?

- A model is a representation of the MDP <s, a, ρ_η> s=states, a=actions, ρ=transitions parameterized by **η**

- Assuming that states and actions are the same as in the environment

$$R_{t+1}, S_{t+1} \sim \hat{p}_\eta(r, s' \mid S_t, A_t)$$

- Learn the transition function and the reward

How do we learn a model from experience?

Given a stream of experiences: $\{S_1, A_1, R_2, \ldots S_T\}$

It is a supervised learning problem

$$S_1, A_1 \to R_2, S_2$$
$$\vdots$$
$$S_{T-1}, A_{T-1} \to R_T, S_T$$

Learn a function that:

$$f(s,a) = r, \underline{s'}$$

$$s' \approx \mathbb{E}[S_{t+1} \mid s = S_t, a = A_t] \longrightarrow \text{Expectation model}$$

Example:

1. choose a loss function (mean-squared error)
2. find the parameter $\boldsymbol{\eta}$ for the transition function $\boldsymbol{\rho_\eta}$ that minimizes the empirical loss

Expectation Model

Considering Linear Model

- Transition matrix P $\mathbb{E}[\phi_{t+1}] = P\phi_t$
- Value function V $v_\theta(S_t) = \theta^\top \phi_t$

$$
\begin{aligned}
\mathbb{E}[v_\theta(S_{t+n}) \mid S_t = s] &= \mathbb{E}[\theta^\top \phi_{t+n} \mid S_t = s] \\
&= \mathbb{E}[\theta^\top P \phi_{t+n-1} \mid S_t = s] \\
&= \ldots \\
&= \mathbb{E}[\theta^\top P^n \phi_t \mid S_t = s] \\
&= \theta^\top P^n \phi(s) \\
&= v_\theta(P^n \phi(s)) \\
&= v_\theta(\mathbb{E}[\phi_{t+n} \mid S_t = s]) \,.
\end{aligned}
$$

Expectation move to inside

Caveat: Might provide states that are never achievable, which happens in non-linear models

Stochastic Model (Generative Models)

- Instead of querying the real environment, we query our stochastic model.
- Allows to sample trajectories that are plausible, in contrast with expected states.

$$\hat{R}_{t+1}, \hat{S}_{t+1} = \hat{p}(S_t, A_t, \underline{\omega})$$

We add a noise term ω

<u>Caveat:</u> Stochastic models increase variance because of the noise

Full Models (Generative Models)

- Balance bias and variance by learning the transitions and the stochasticity of the environment.
- This involves branching for every state for the same action:

$$\mathbb{E}[v(S_{t+1}) \mid S_t = s] = \sum_a \pi(a \mid s) \sum_{s'} \hat{p}(s,a,s')(\hat{r}(s,a,s') + \gamma v(s'))$$

$$\begin{aligned}\mathbb{E}[v(S_{t+n}) \mid S_t = s] = \sum_a \pi(a \mid s) \sum_{s'} \hat{p}(s,a,s')\Bigg(\hat{r}(s,a,s') + \\ \gamma \sum_{a'} \pi(a' \mid s') \sum_{s''} \hat{p}(s',a',s'')\Bigg(\hat{r}(s',a',s'') + \\ \gamma^2 \sum_{a''} \pi(a'' \mid s'') \sum_{s'''} \hat{p}(s'',a'',s''')\Bigg(\hat{r}(s'',a'',s''') + \dots \Bigg)\Bigg)\Bigg)\end{aligned}$$

"For continuous state spaces, these sums will become integrals. "

Caveat: These trees also might grow very fast.

How to represent these learned models?

- Table lookup Model
- Linear Expectation Model
- Linear Gaussian Model
- Deep Neural Network Model

Table Lookup Model

It is an explicit MDP

It counts the visits N(s,a) to each state action pair

Transition function

$$\hat{p}_t(s' \mid s, a) = \frac{1}{N(s,a)} \sum_{k=0}^{t-1} I(S_k = s, A_k = a, S_{k+1} = s')$$

Reward function

$$\mathbb{E}_{\hat{p}_t}[R_{t+1} \mid S_t = s, A_t = a] = \frac{1}{N(s,a)} \sum_{k=0}^{t-1} I(S_k = s, A_k = a) R_{k+1}$$

Inaccurate models

Given an imperfect model where $\hat{p}_\eta \neq p$

- Performance will be limited to the optimal policy for an approximate MDP $<s, a, \rho_\eta>$
- Hence Model-Based RL will be only as good as the estimated model

Alternative solutions:

1. When model is wrong, use Model-Free RL
2. Reason explicitly about the model uncertainty over η (e.g., Bayesian methods)
3. Combine model-based and model-free in a safe way

Real versus Simulated Experiences

Real experience is sampled from the environment (true MDP)

$$r, s' \sim p$$

Simulated experience is sampled from our model (approximate MDP)

$$r, s' \sim \hat{p}_\eta$$

Road so far

Model-Free RL

- No Model
- Learn Value function of Policy from Experience

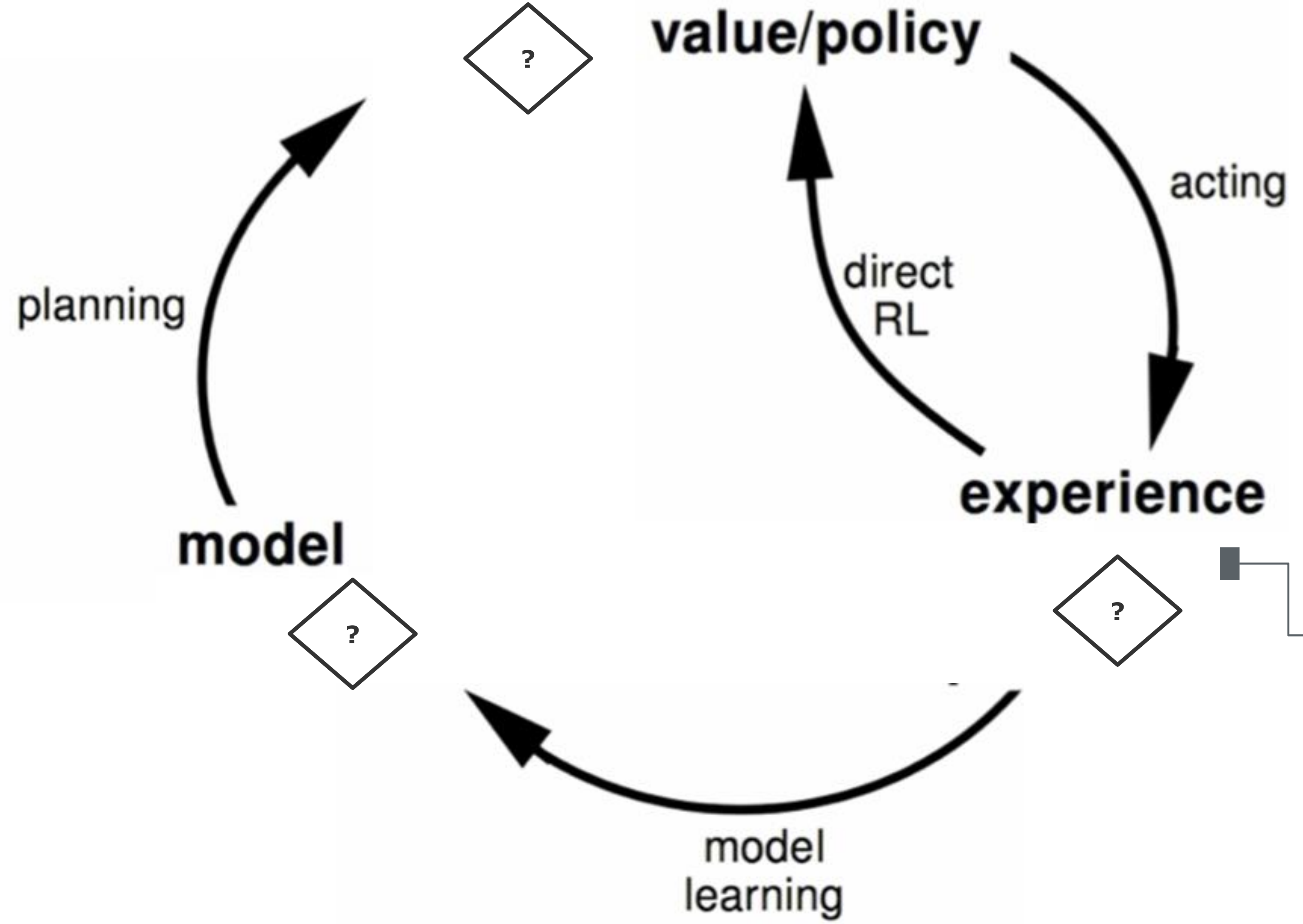
Model-Based RL

- Learn a model from the experience or be given a Model
- Plan Value function or Policy from Model

Dyna Architecture

- Learn a Model from the experience
- Learn and Plan Value function or Policy from real and simulated experience
- Treat real and simulated experiences equivalently
    - Updates from learning and planning are not distinguished

Dyna Architecture



- New observation from the environment

  OR

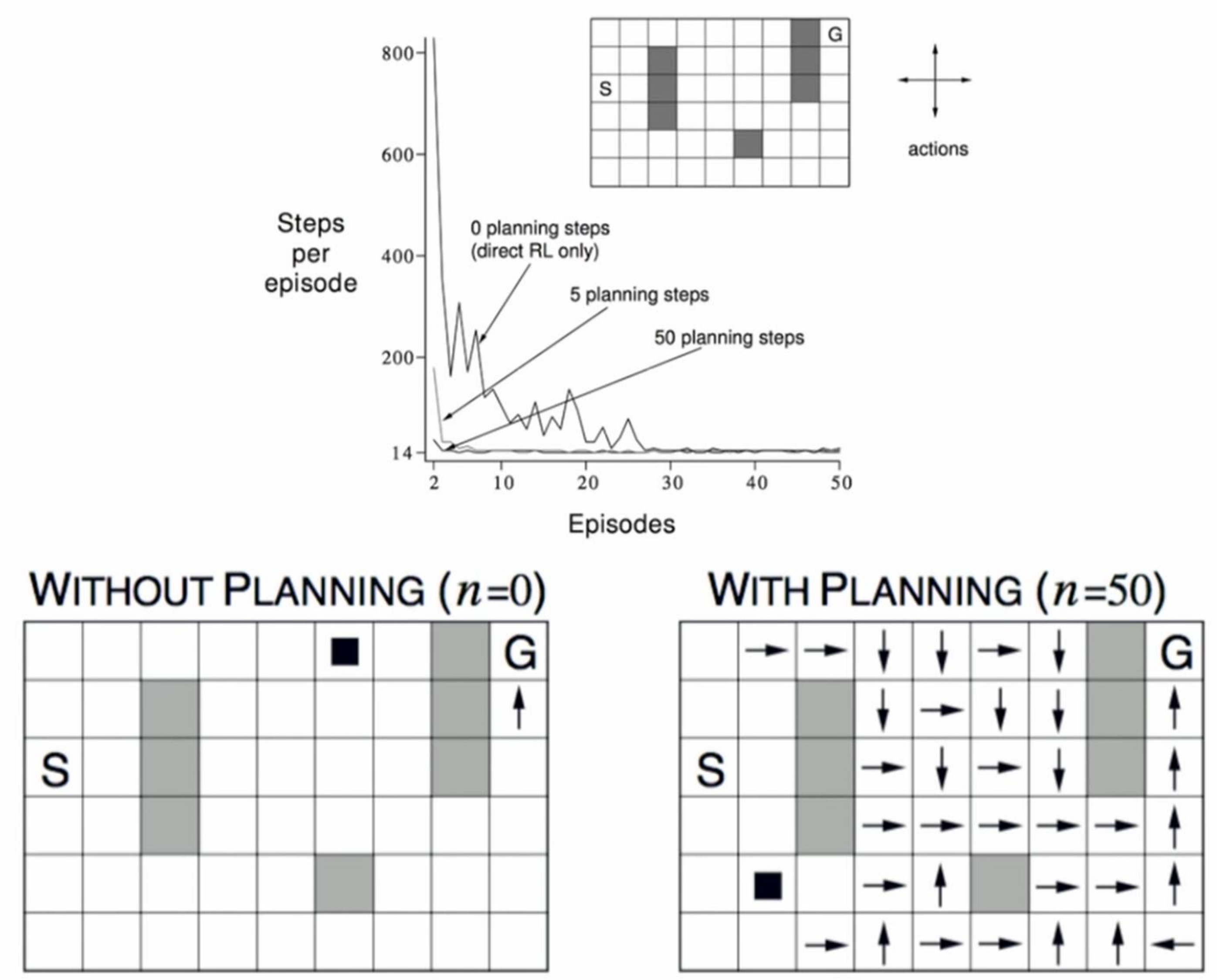
- Reuse previous observations (experience replay)

Dyna pseudo-code

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
 (a) $s \leftarrow$ current (nonterminal) state
 (b) $a \leftarrow \varepsilon$-greedy(s,Q)
 (c) Execute action a; observe resultant state, s', and reward, r
 (d) $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
 (e) $Model(s,a) \leftarrow s', r$ (assuming deterministic environment)
 (f) Repeat N times:
  $s \leftarrow$ random previously observed state
  $a \leftarrow$ random action previously taken in s
  $s', r \leftarrow Model(s,a)$
  $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

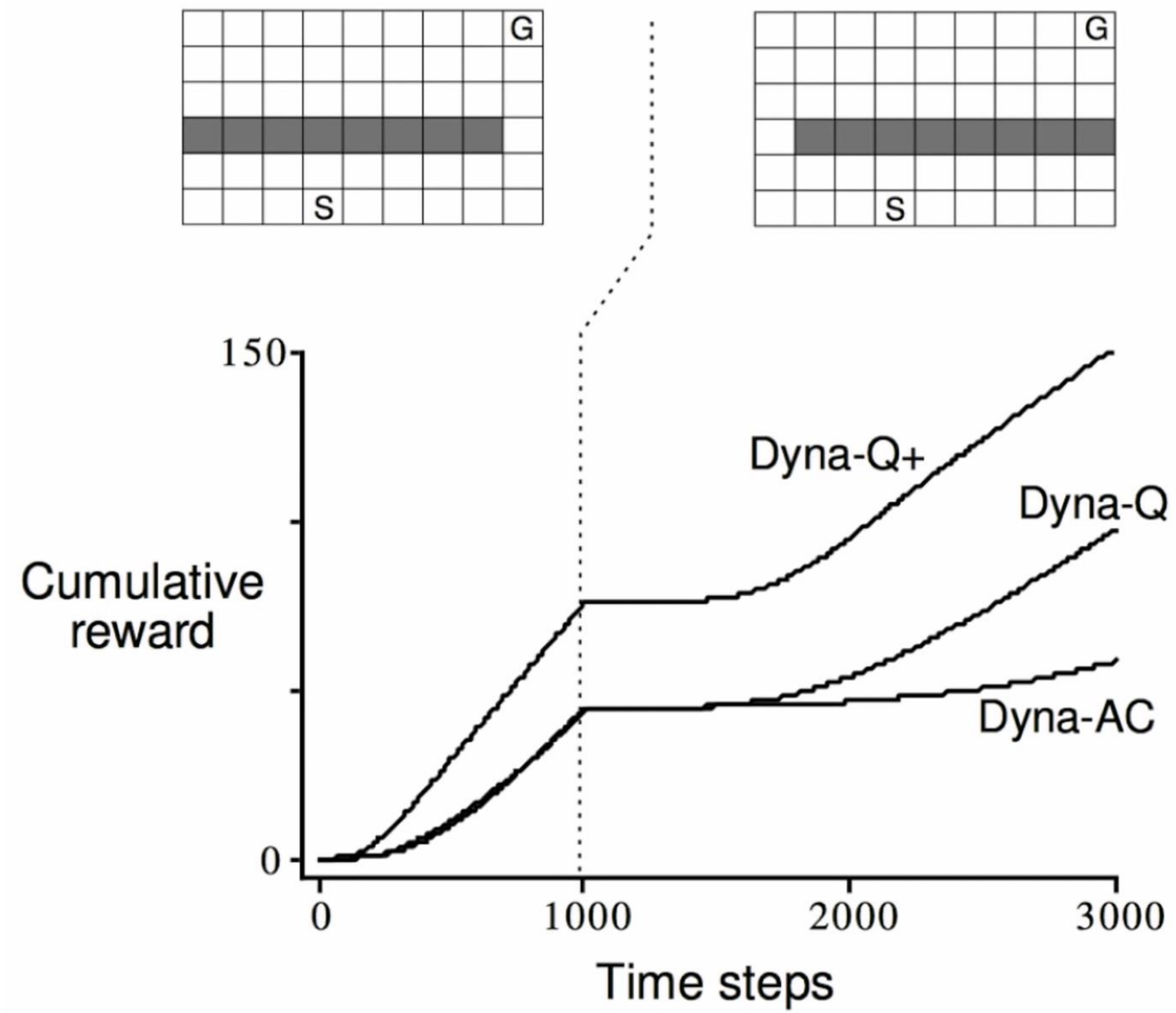
Python code example: https://github.com/andrecianflone/dynaq/blob/master/dynaq.py

Dyna Q with a Maze example



Dyna-Q with inaccurate model

Model changed! Became harder to attain the goal

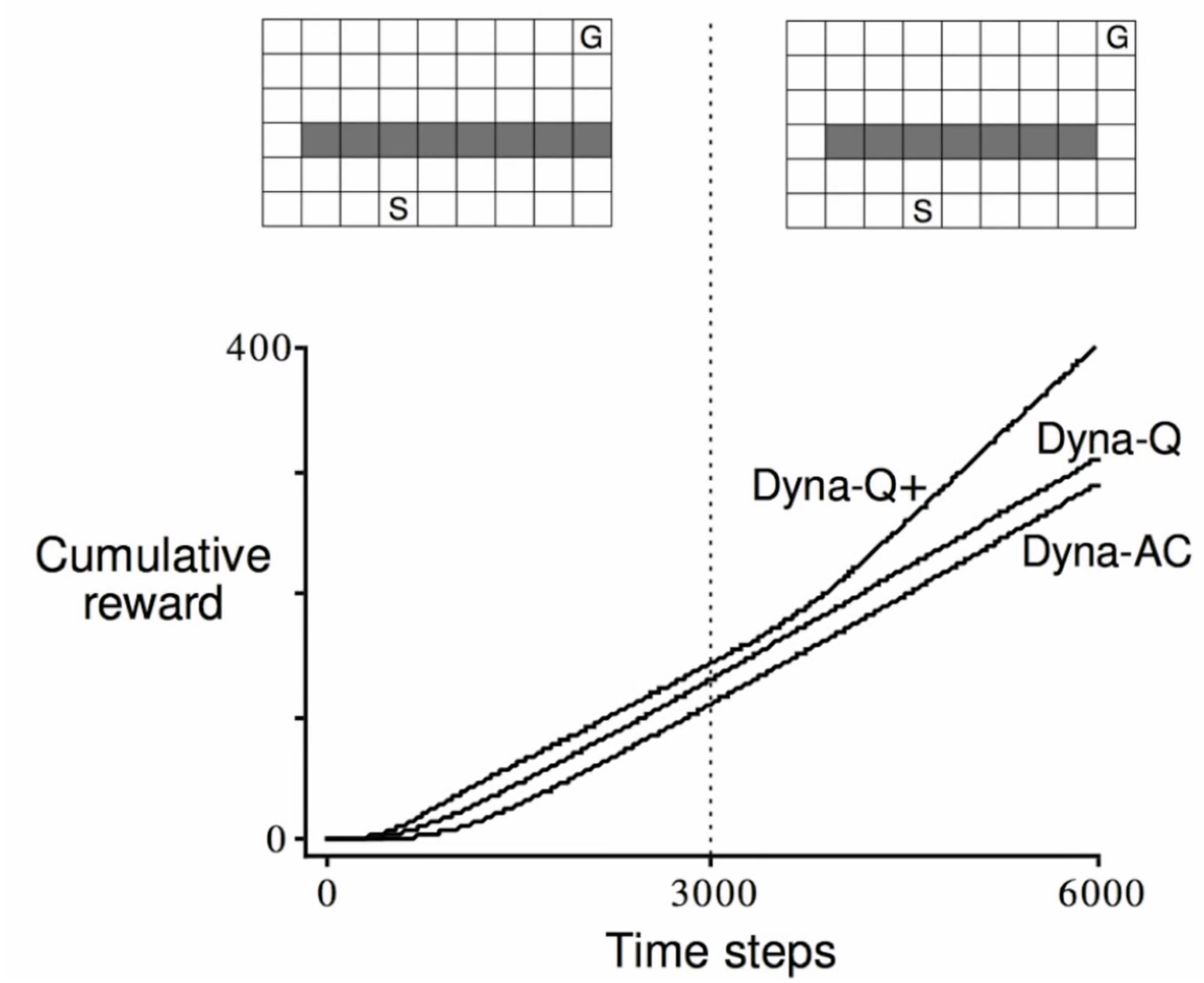


Dyna-Q+ is **Dyna-Q** with an exploration bonus that encourages exploration

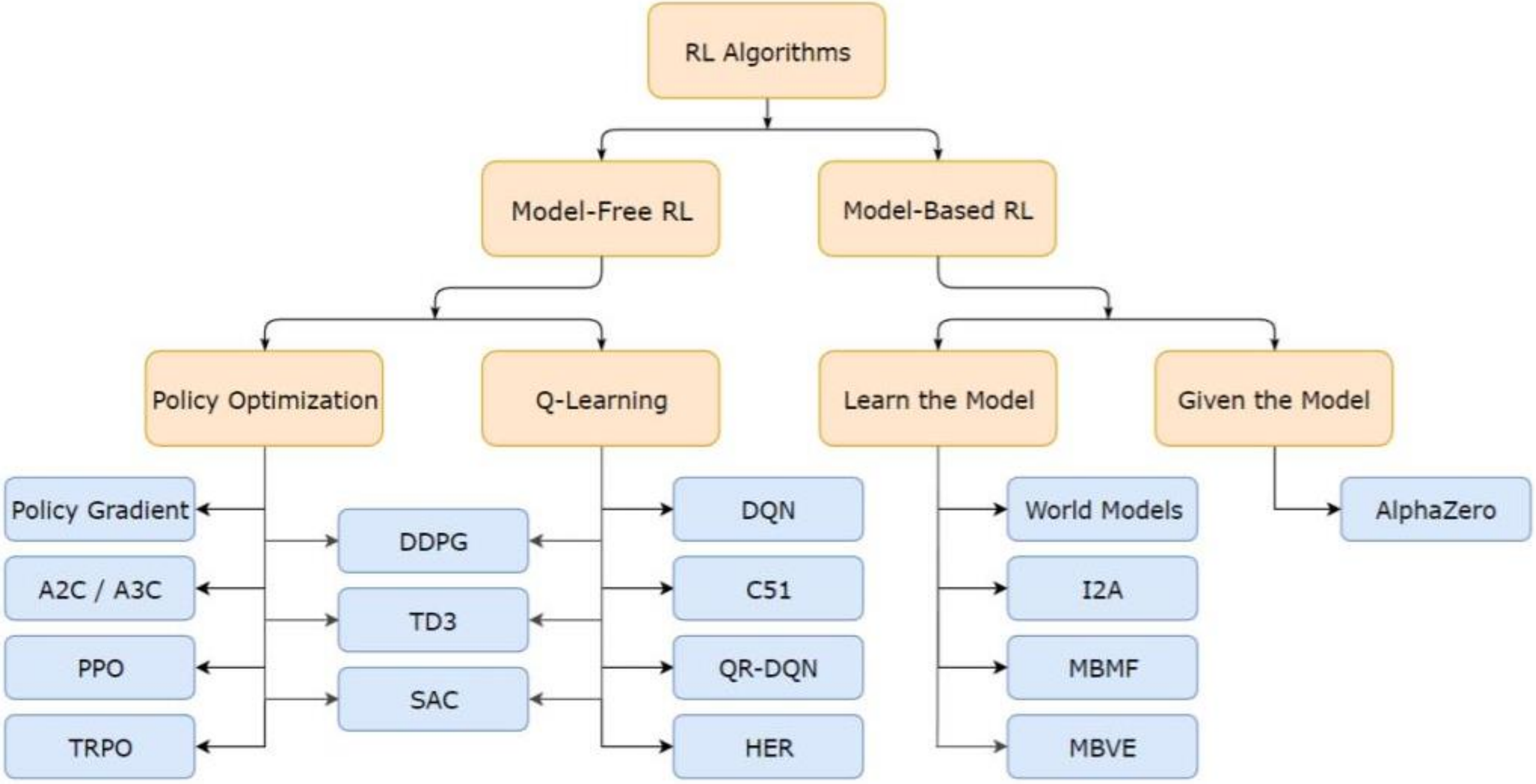
Dyna-AC is a Dyna-Q that uses an actor-critic learning method instead of Q-learning

Dyna-Q with easier inaccurate model

Model changed! Became **easier** to attain the goal



Brief Taxonomy of RL Methods



More interesting topics

Actor-Critic Model

Batch Reinforcement Learning

Hierarchical Reinforcement Learning

Bayesian Reinforcement Learning

Working with non-stationary environments

Reinforcement Learning for Optimizing Hyperparameters

Hidden Markov Models

Partially Observable Markov Decision Processes

….

Interesting lectures

David Silver, 2015, University College London Lecture 1 https://www.youtube.com/watch?v=2pWv7GOvuf0 Lecture 2 https://www.youtube.com/watch?v=lfHX2hHRMVQ&list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ&index=2&pbjreload=10 Lecture 3 https://www.youtube.com/watch?v=Nd1-UUMVfz4&list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ&index=3&pbjreload=10 Course website: https://www.davidsilver.uk/teaching/	Bottom-up explanations based on introducing the simpler models before adding the complexity that justify more complex models. Explains the math behind concepts.
Pascal Poupert, 2018, University of Waterloo, Canada Lecture 1b: https://www.youtube.com/watch?v=yOWBb0mqENw&list=PLdAoL1zKcqTXFJniO3Tqqn6xMBBL07EDc&index=2&pbjreload=10 Lecture 2a: https://www.youtube.com/watch?v=yOWBb0mqENw&list=PLdAoL1zKcqTXFJniO3Tqqn6xMBBL07EDc&index=2&pbjreload=10 Lecture 2b: https://www.youtube.com/watch?v=mjyrRG7RD84&list=PLdAoL1zKcqTXFJniO3Tqqn6xMBBL07EDc&index=5&pbjreload=10 Course website: https://cs.uwaterloo.ca/~ppoupart/teaching/cs885-spring20/schedule.html	More conceptual level with examples of applications.
Charles Isbell & Michael Littman, 2015, Georgia Tech https://www.youtube.com/watch?v=rETmf4NnlPM&list=PL__yccKD1ec_yNMjDl-Lq4-1ZqHcXqgm7&index=128&pbjreload=10	Step-by-step explanations with fun discussions and lot of small examples to illustrate each core MDP principle
Nando Freitas, 2015, Lectures Oxford University, UK Lecture 15 https://www.youtube.com/watch?v=kUiR0RLmGCo Lecture 16 https://www.youtube.com/watch?v=dV80NAlEins Course Website: https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/	Lectures in the context of Deep Learning, but still provide the necessary concepts of RL.