Reinforcement Learning: Model-Based RL (ESE 3990)

Instructor:
Antonio Loquercio

Fall 2026

University of Pennsylvania

Categorizing RL agents

Axis 1: How to make decisions

- Value Based
  - No Policy (Implicit)
  - Value/Q Function
- Policy Based
  - Policy
  - No Value/Q Function
- Actor Critic
  - Policy
  - Value Function

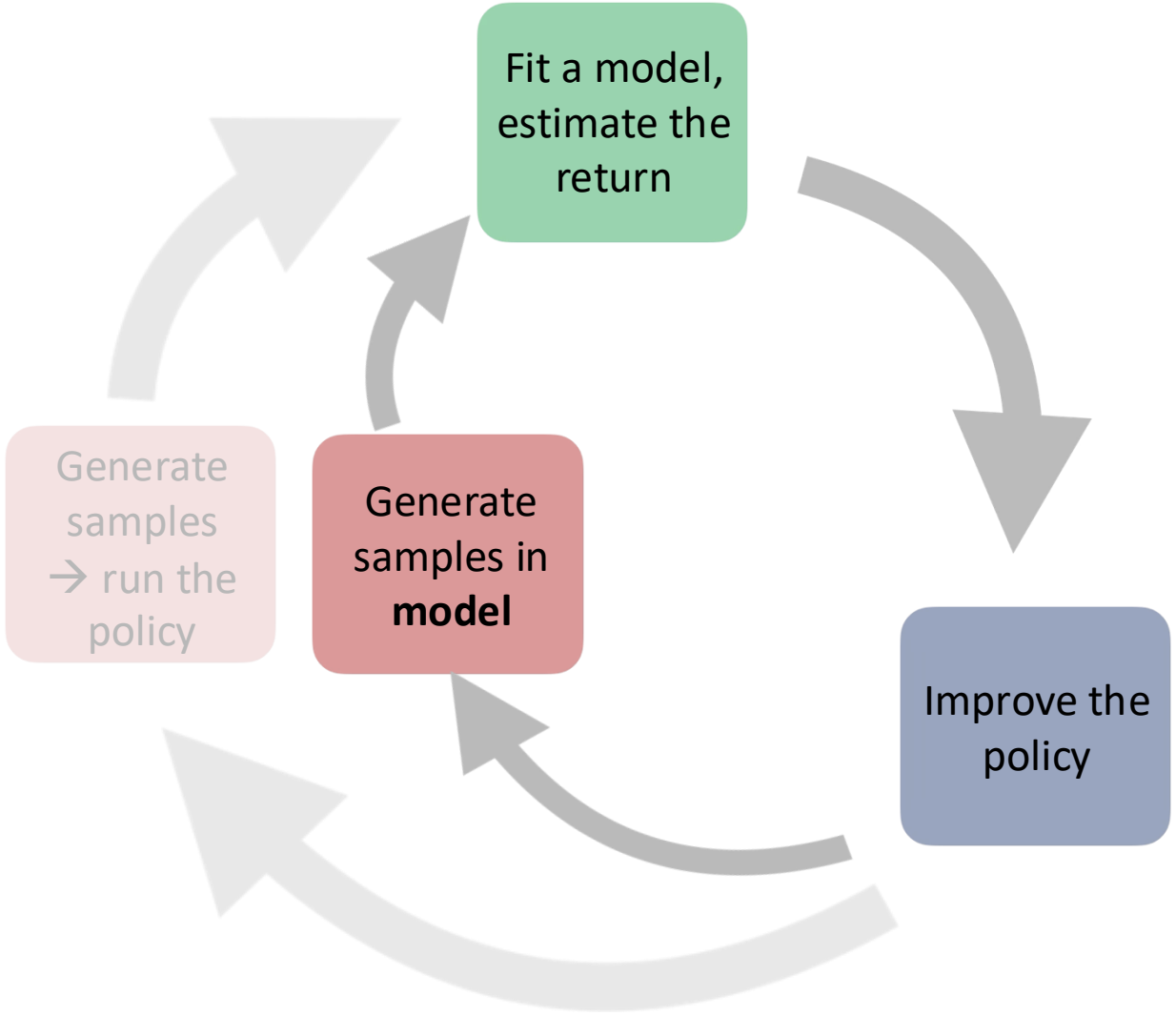
Axis 2: Environment Representation

- Model-based — Today
  - Policy and/or Value/Q Function
  - Model
- Model-free
  - Policy and/or Value/Q Function
  - No Model

Why Model-Based?

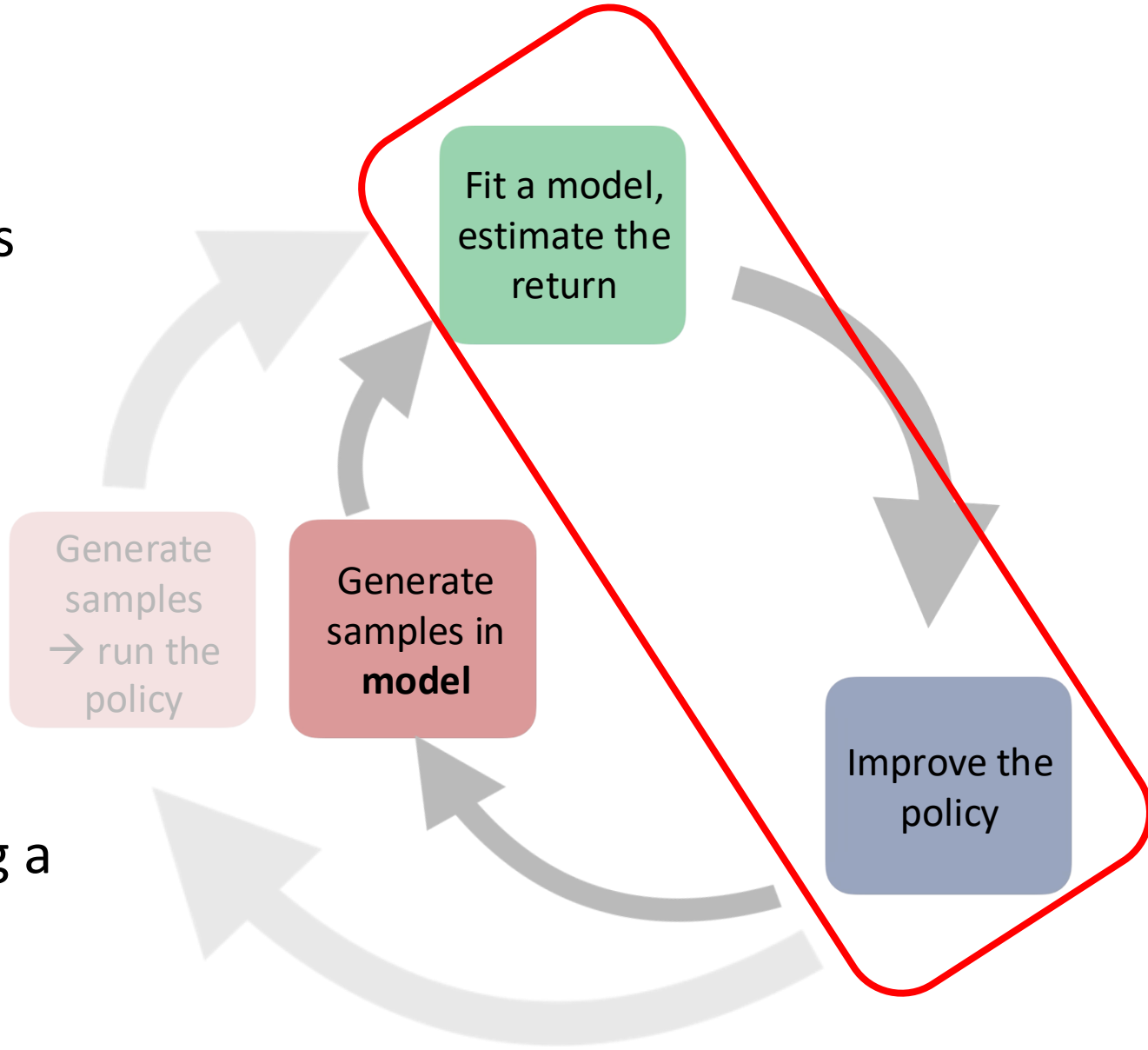
- Generate samples in a “model” of the world.

- The model is a “simulator” of the environment:
  - Predicts the next state given the current state.
  - Predicts the reward given a state.



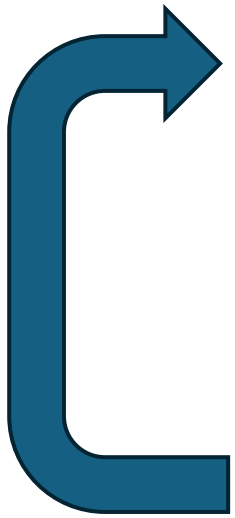
Why Model-Based?

- I can use any of the prior algorithms for policy optimization.
  - Dynamic Programming
  - Model-Free Control
    - With Q-functions
    - Policy-based
    - Actor Critic
- We define the process of evaluating a policy or finding V^*/π^* only using a model *planning*.



Example: Planning with Q-Learning

- Until Converged

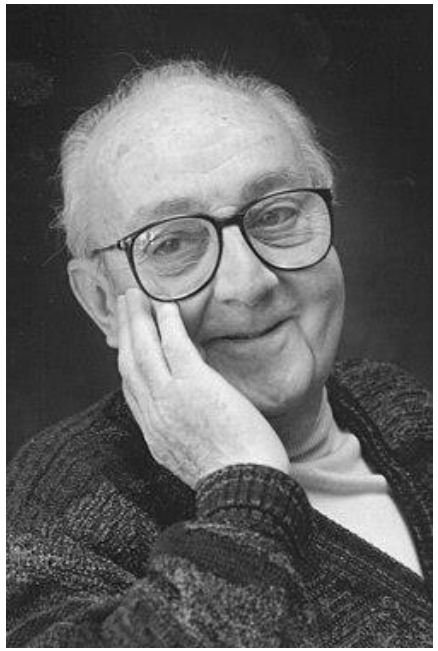


  - Select a state $s \in S$ and an action $a \in A$, at random
  - Send s, a to a model. Obtain: (1) next state s', (2) current reward r
  - Apply one-step tabular Q-learning to (s, a, r, s')

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_a Q(s',a) - Q(s,a) \right]$$

Models are always wrong!

- The environment can change over time.
- The environment is too complex to be completely captured in a model.
- Errors in the model can bias the policy/value function.

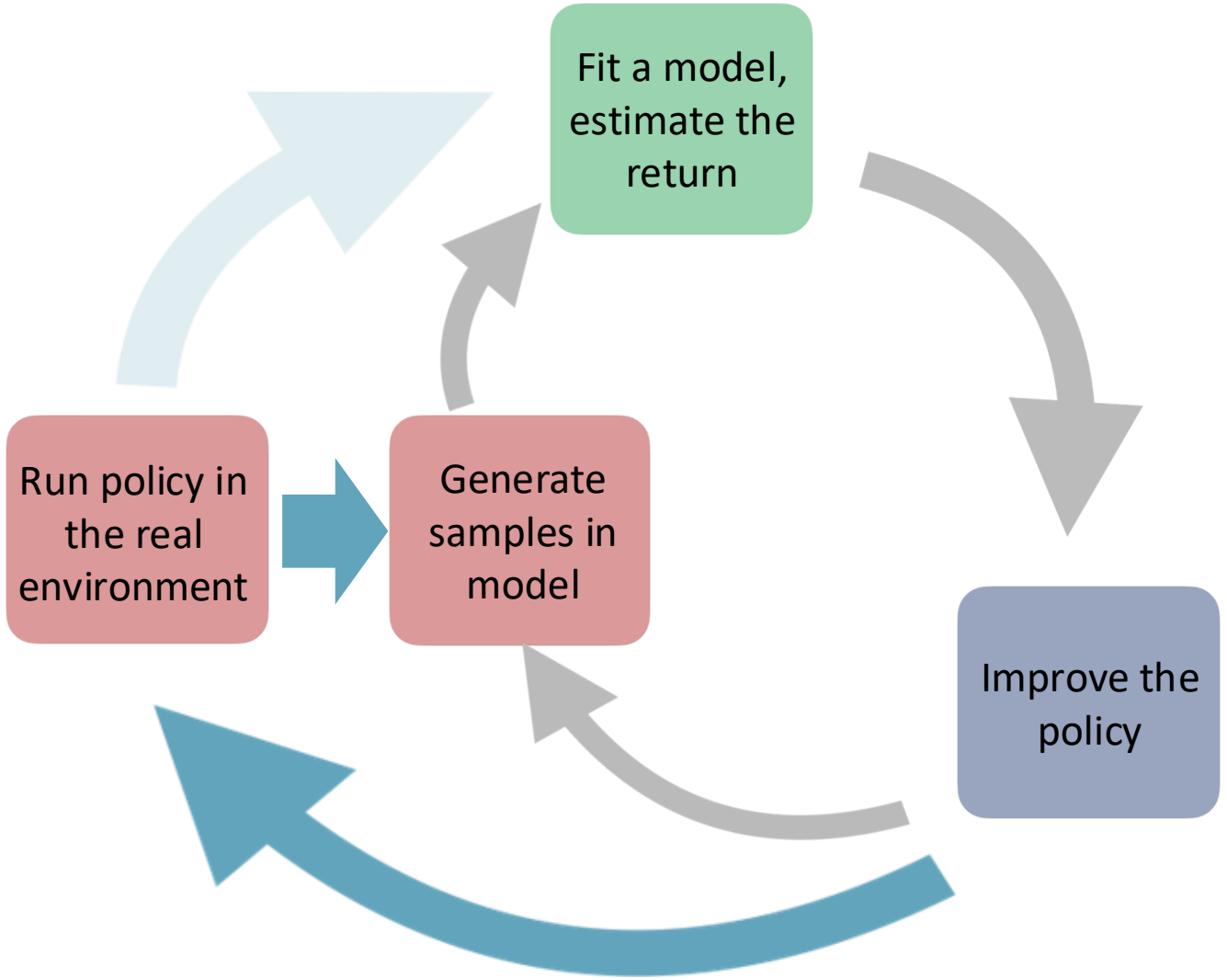


All models are wrong, but some are useful.

George Box

Improve Them With Real Samples!

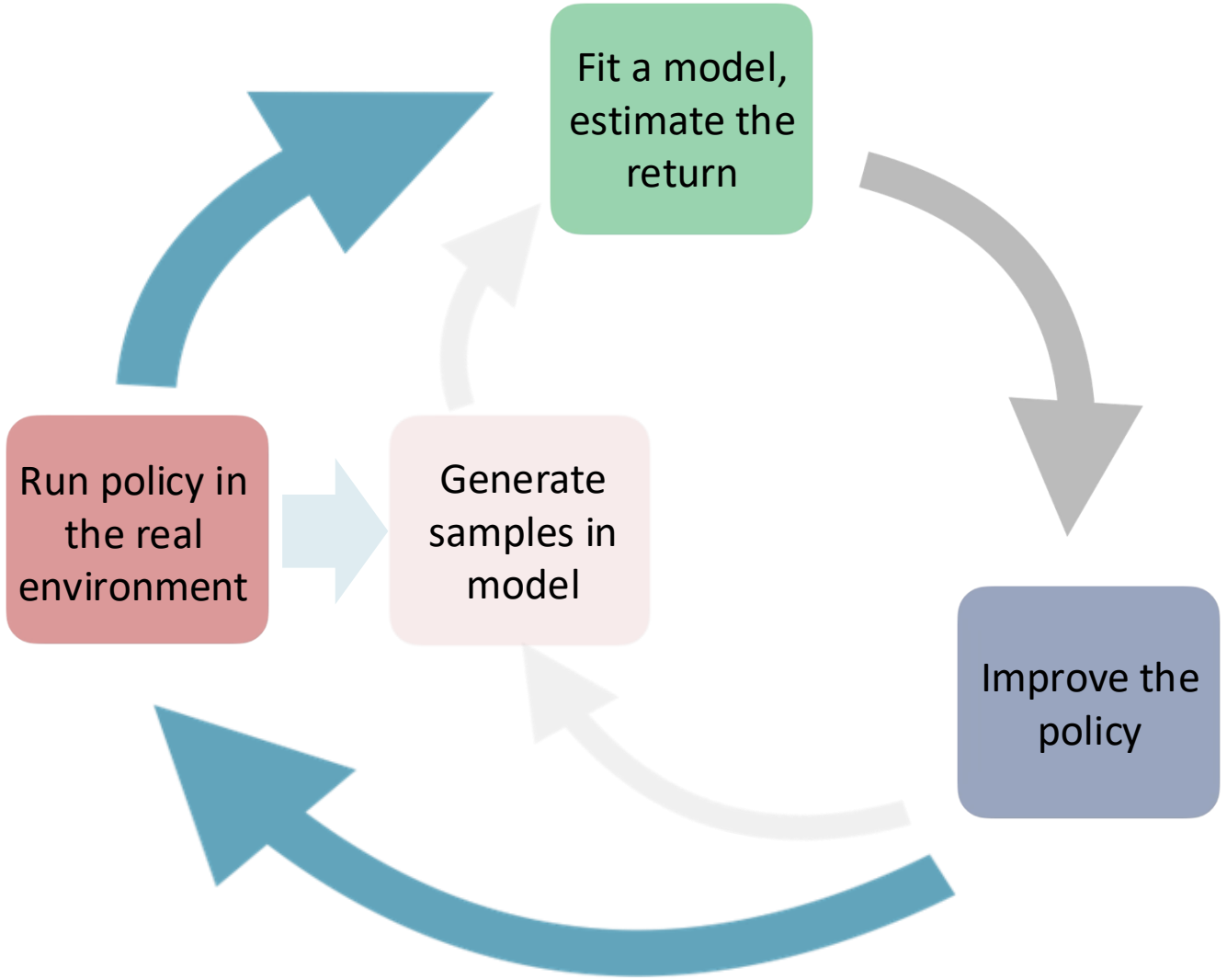
- *Model-learning*
  - Current focus of research on world models.
  - Often make “fuller” use of data.
  - Can achieve a good policy with few environment interactions.
- Some people’s opinion:
  - Always superior to directly learning values/policies.
  - Responsible for learning in humans and animals.



Get Rid of them Completely!

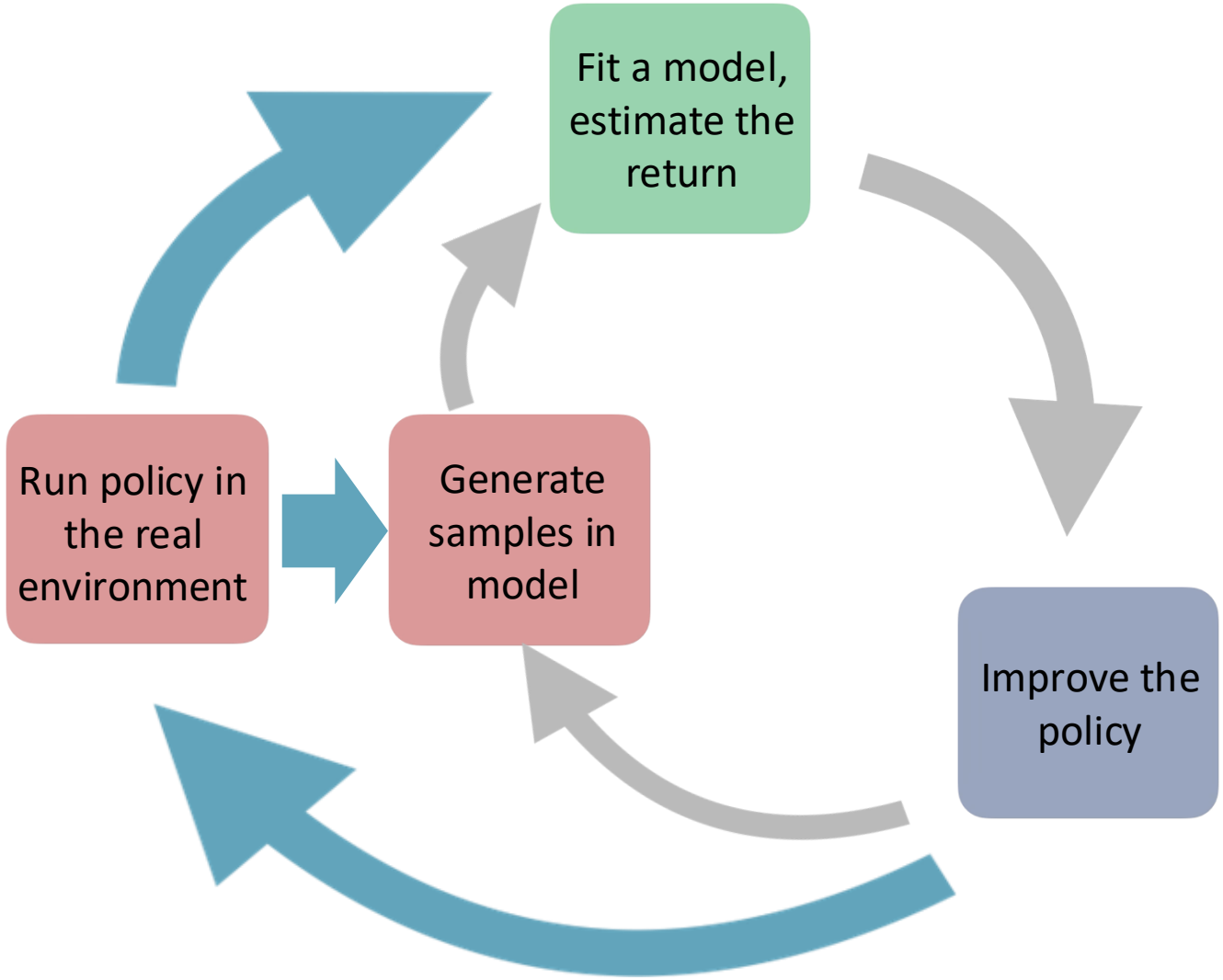
- *Direct learning*
  - Not biased by the model design.
  - Harder to design a model than learning a policy.

- Some people's opinion:
  - Always superior to model-based learning.
  - Responsible for learning in humans and animals.



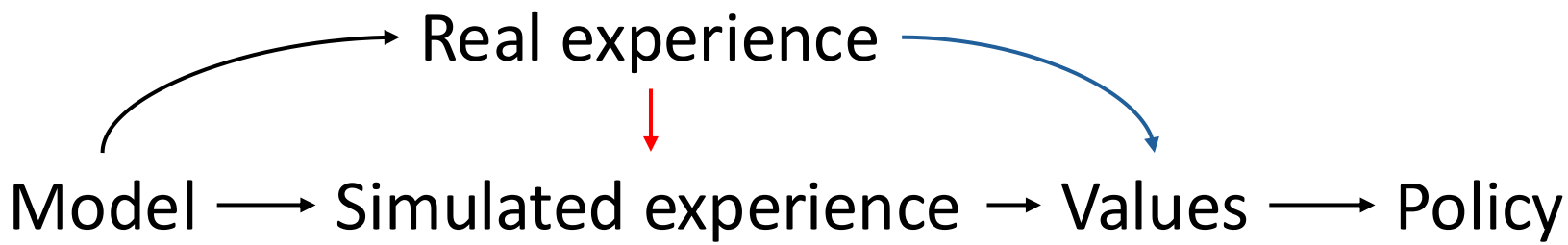
Integrated Planning and Learning

- Both opinions are exaggerated.
- We occasionally run the policy in the real environment:
  - To improve the model.
  - To directly improve the Q function and/or the policy.
- Often called *Dyna (Sutton, '91)*



A Unifying Perspective on Model-Based RL

- **Unification key:** Estimation of value functions.



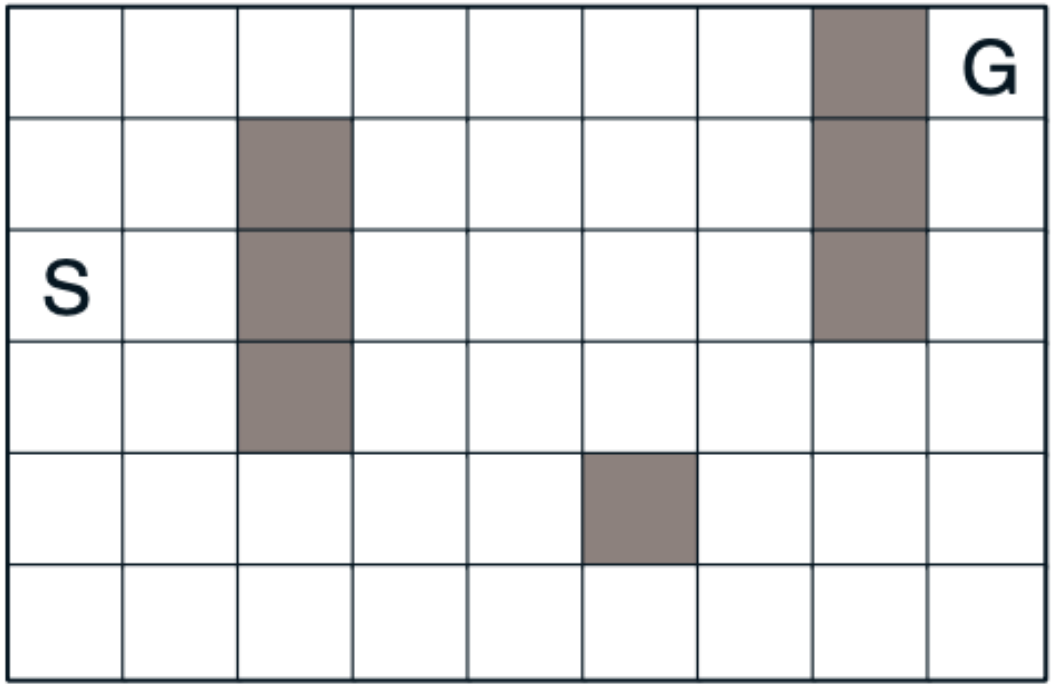
- Smoothly move from one to the other depending on:
  - How easy is it to learn/design a model?
  - How many samples can you afford to collect in the real environment?

Example: Dyna-Q

- Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in S$ and $a \in A$
- Until Converged
  - $s \leftarrow$ current (non-terminal) *real* state
  - $a = \epsilon$-greedy(Q)
  - Take action a in the real environment. Observe r and next state s'
  - Update Q function: $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_a Q(s',a) - Q(s,a)\right]$
  - Update the model (assume deterministic env.): $Model(s,a) \leftarrow r, s'$
  - Plan in the model for n times:
    - $s, a \leftarrow$ randomly *observed* prior states and actions
    - $r, s' \leftarrow Model(s,a)$
    - Update Q function with the predicted transition:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_a Q(s',a) - Q(s,a)\right]$$

Dyna-Q in a Maze

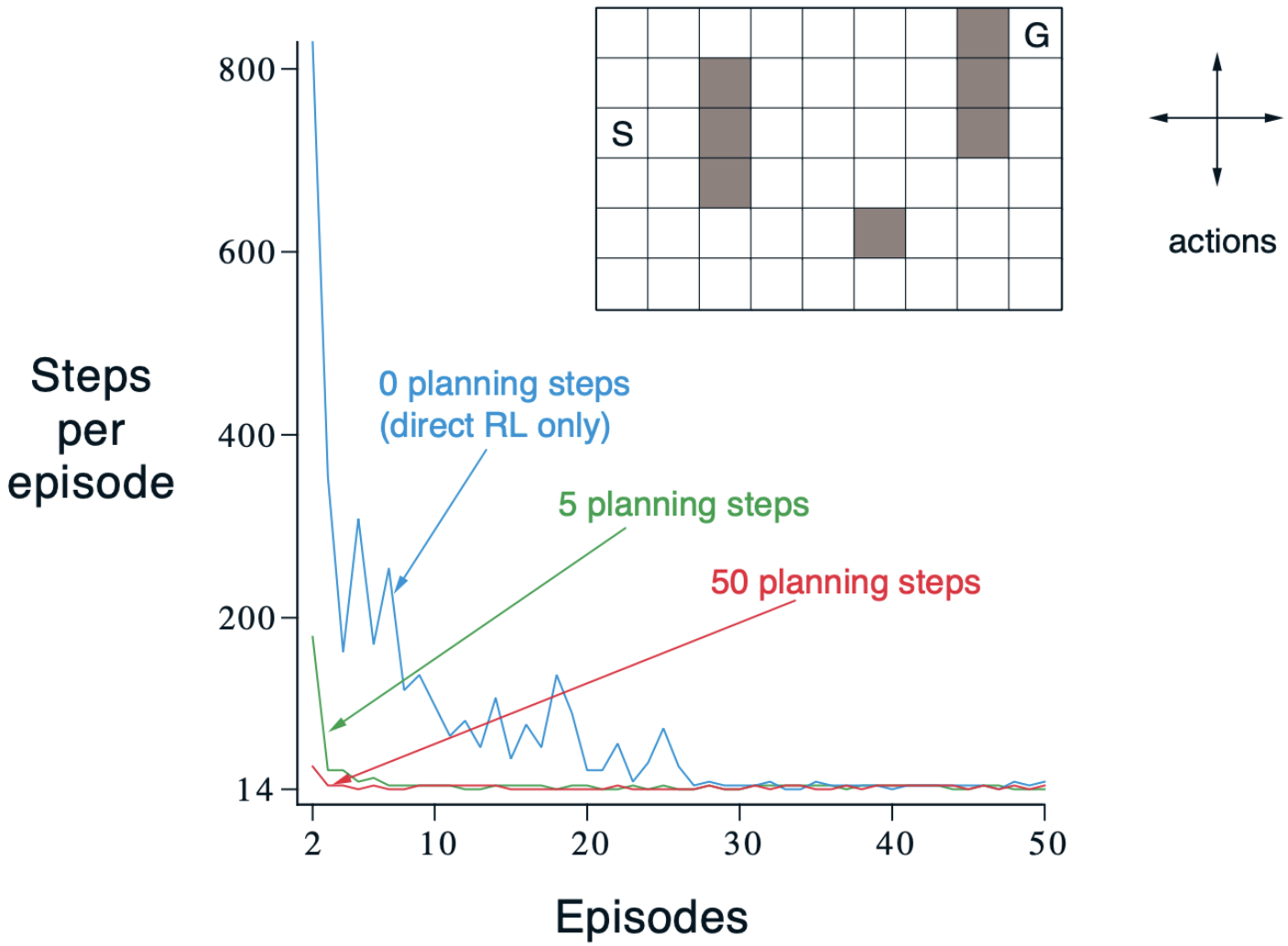




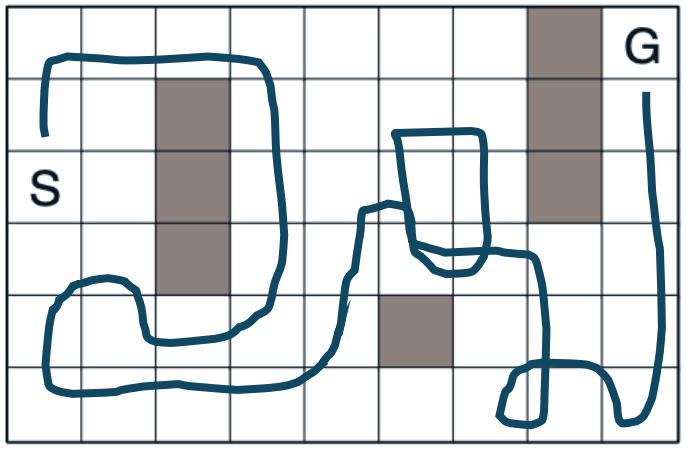
actions

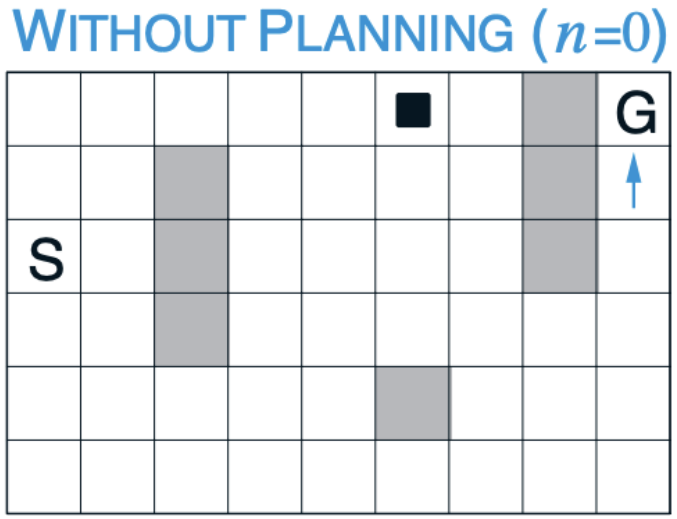
- Sparse rewards: 100 at G, 0 otherwise.
- G is terminal.

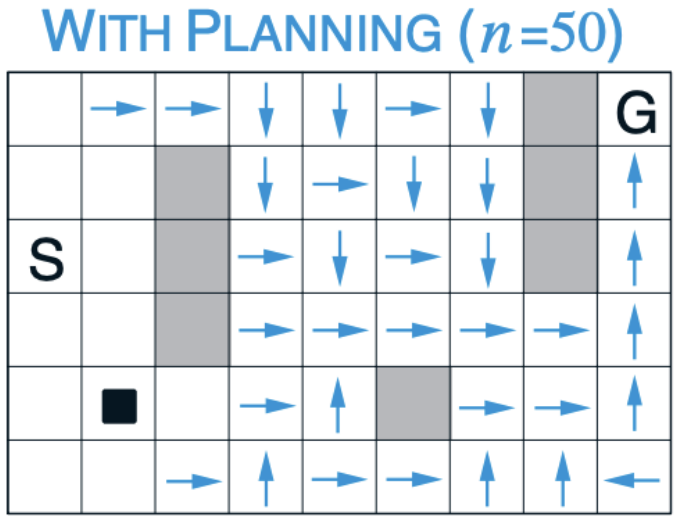
Dyna-Q in a Maze



Why Was Planning So Much Faster?





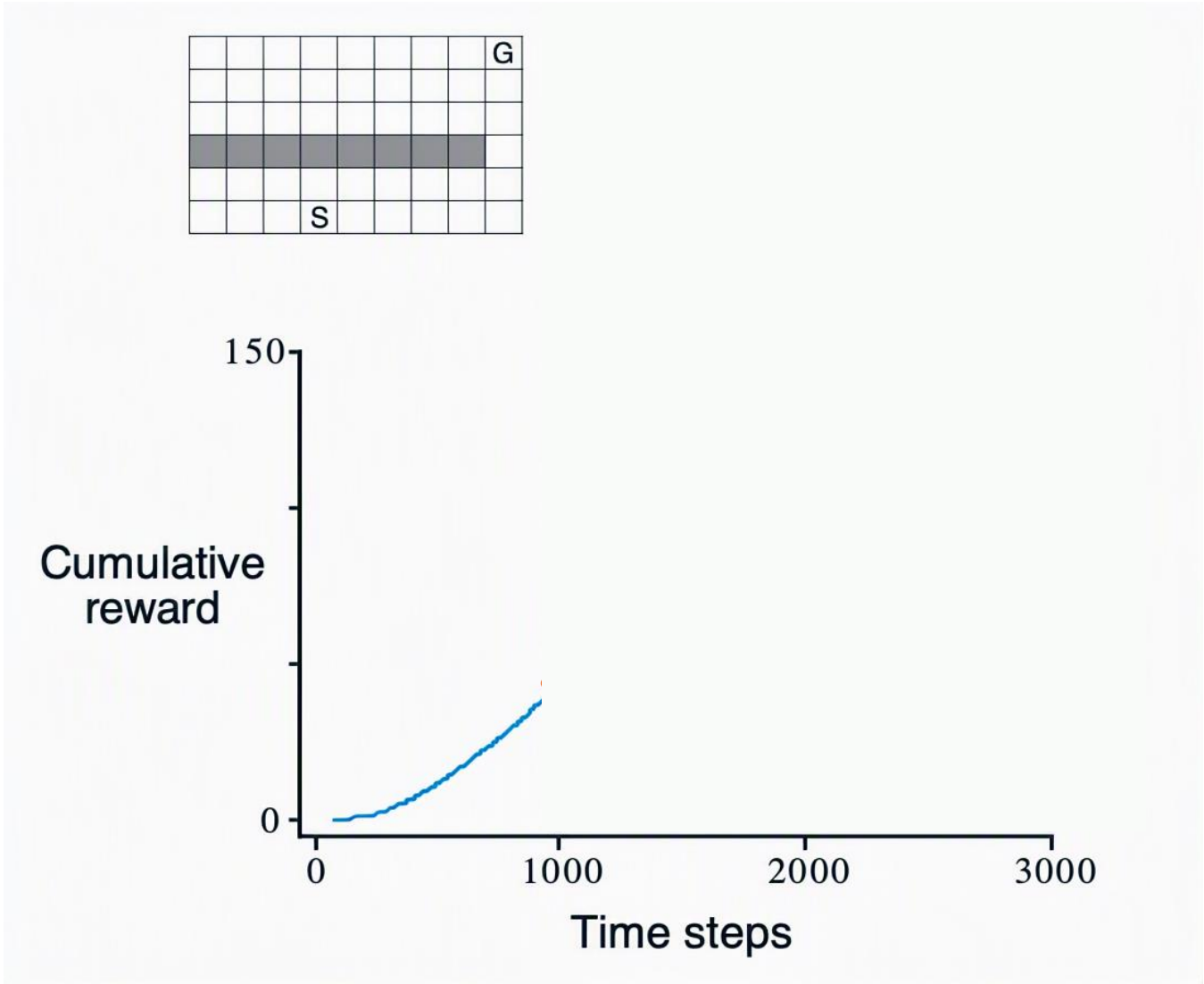


- Assume the first episode is done by an “exploration” policy.
- Policy found by the two agents halfway through the second episode
- Without planning, we learn only from the final transition.
- With planning, the signal propagates through the entire maze.

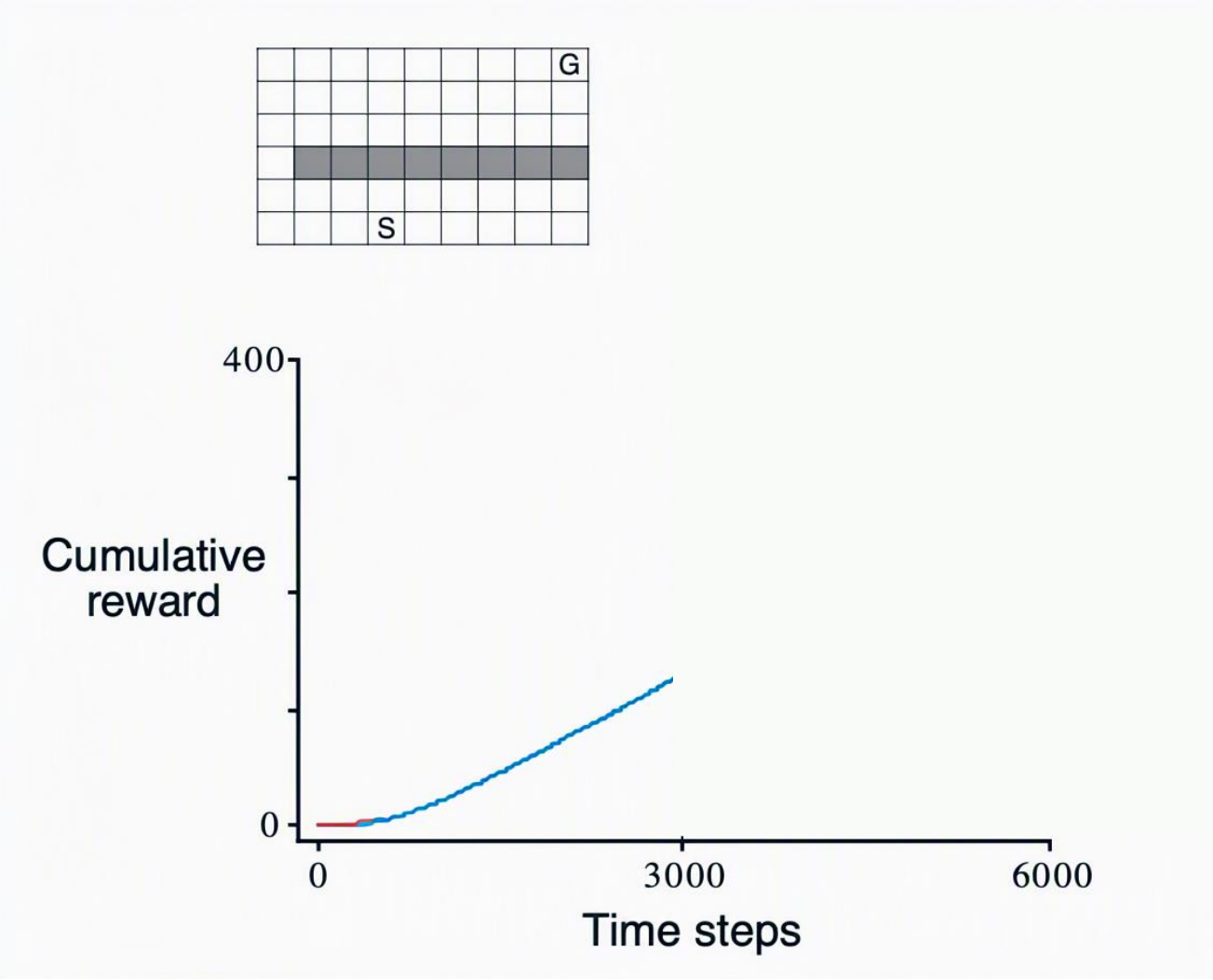
What are the effects of a wrong model?

- Possible causes of wrong models:
  - The environment is stochastic, and we only observe samples. Our estimates could be off.
  - Our model might not be expressive enough.
  - The environment is changing over time.

Example: Blocking Maze



Example: Shortcut Maze



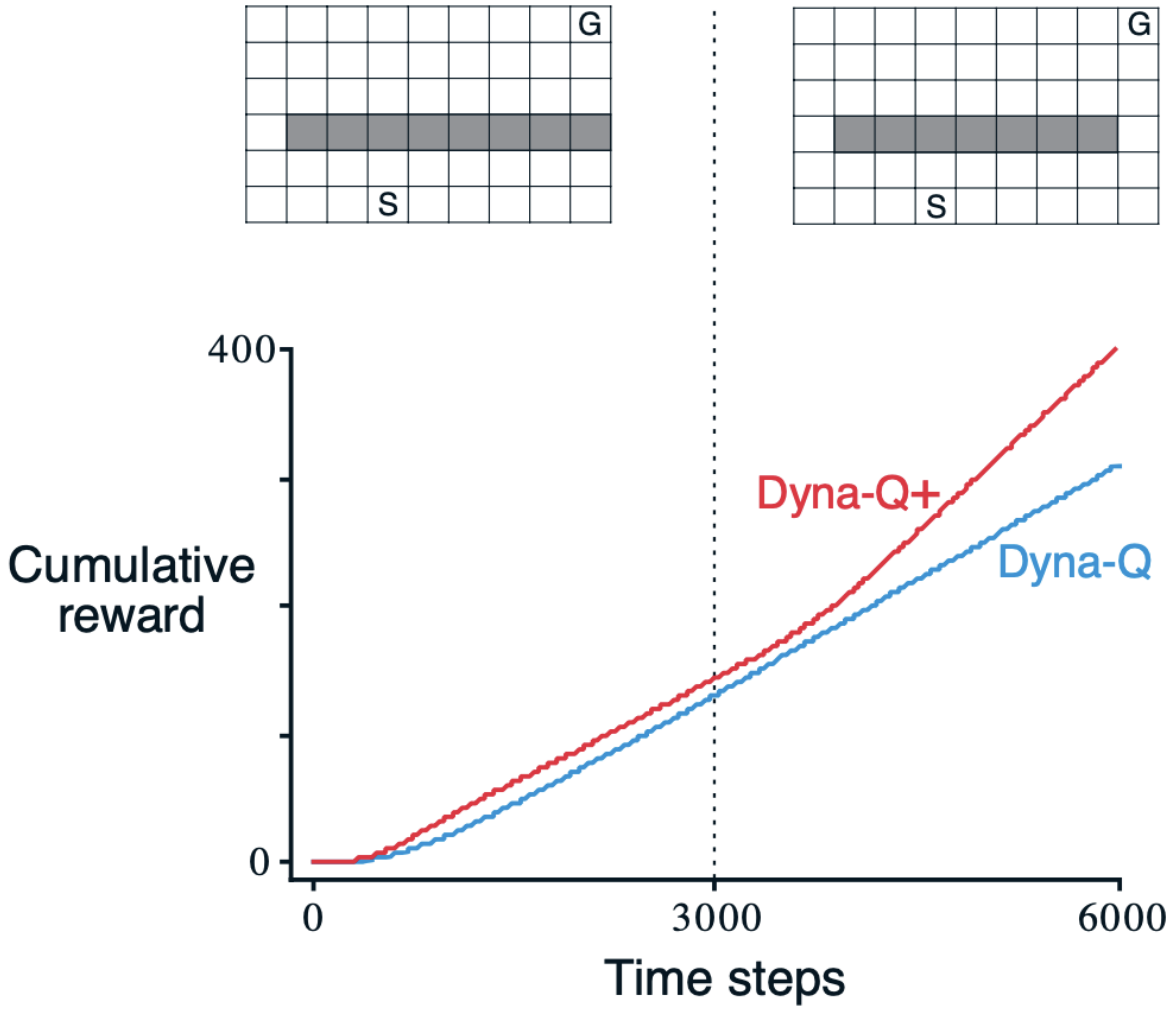
- The model stops you from exploring!

Model-Based RL: Exploration vs Exploitation

- You want to use real-world samples to learn a good model.
- But the actions that lead to a good model might not be the ones that maximize the reward (given your current knowledge).
- Inescapable problem of model-based RL.
  - No solution exists that is both practical and effective.
  - Existing algorithms use different heuristics to address this problem.

Example: Dyna-Q+

- Not a “real” algorithm, just used here as a didactic example.
- Dina-Q+:
  - Bias the model towards states that were not visited for a long time or never visited.



Summary up to now

- Model-based RL integrates learning with real samples and planning with a model.
- We can use the algorithms presented so far with data coming from either the simulated or real environment.
- A wrong model leads to an exploration/exploitation trade-off
- Next, we'll get a bit more technical.

What is a Model?

- A model $\mathcal{M}$ is a representation of an MDP $\langle S, A, P, R \rangle$
- The model has parameters η
- *Assuming* that the state space S and action space A are known
- A model $\mathcal{M} = \langle P_\eta, R_\eta \rangle$ aims to match the state transitions and rewards of the original MDP:

$$s_{t+1} \sim P_\eta(s_{t+1}|s_t, a_t) \approx P(s_{t+1}|s_t, a_t)$$
$$r_t \sim R_\eta(r_t|s_t, a_t) \approx R(r_t|s_t, a_t)$$

- Note: We often know R (but not always)

Example: Autonomous Driving

- What is a_t, s_t?
- What is $P_\eta(s_{t+1}|s_t, a_t)$?
- What is $R_\eta(r_t|s_t, a_t)$?



Model Learning

- **Goal**: Find the model parameters η^* that best fit the environment

$$\eta^* = \arg\min_{\eta} D_p\left(P\ , P_\eta\right) + D_r(R, R_\eta)$$

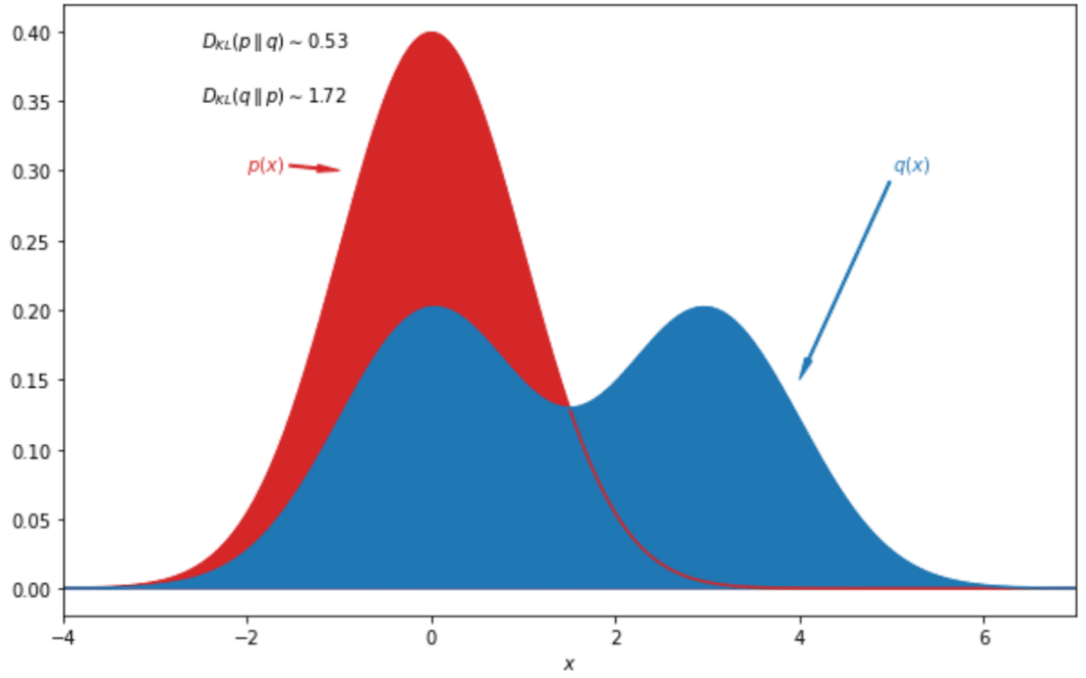
- For some appropriate definition of distances D_p, D_r.

- We typically assume *conditional independence* between state transitions and rewards:

$$\mathbb{P}[s_{t+1}, r_t | s_t, a_t] = P(s_{t+1} | s_t, a_t) R(r_t | s_t, a_t)$$

Distance between State Transitions D_p

- $P(s_{t+1}|s_t, a_t)$ is a probability distribution function.
- We need distance functions D_p that are appropriate for probabilities.
- Examples:
  - KL Divergence (not really a distance)
  - Wasserstein Distance
  - Total Variation
  - …
- Learning P_η is a distribution fitting problem.

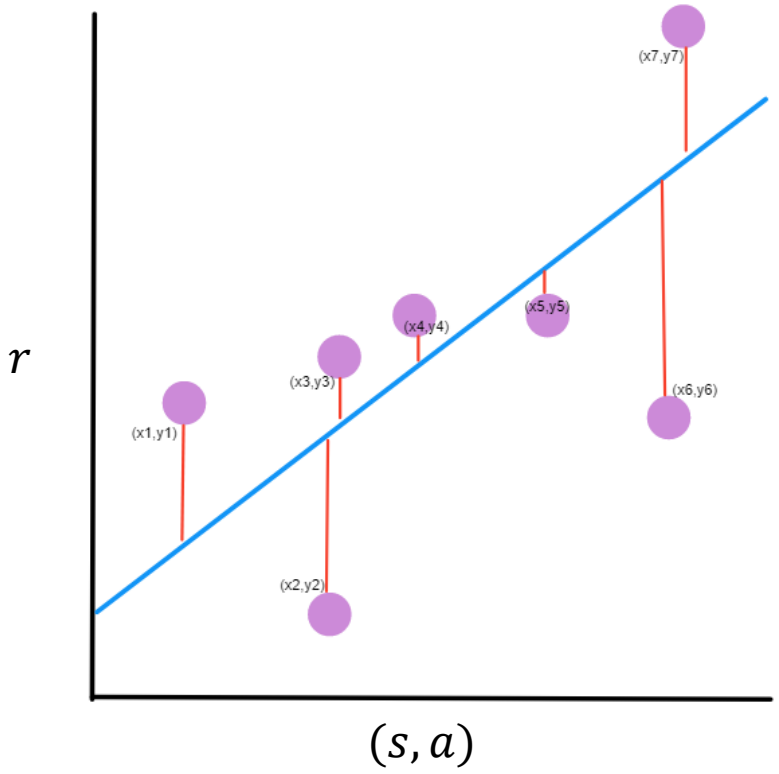


Distance between Reward Functions D_r

- $R(r_t|s_t, a_t)$ is generally a deterministic function.
  - We often write it as $r(s, a)$

- Therefore, we can use distance metrics such as:
  - Mean-Squared Error
  - L_p distance $|R - R_\eta|^p$
  - …

- Learning R_η is a *regression* problem



Model Learning

- **Goal**: Find the model parameters η^* that best fit the environment

$$\eta^* = \arg\min_{\eta} D_p\left(P\ , P_\eta\right) + D_r(R, R_\eta)$$

- There are generally many more states than free parameters η
- Therefore, we can't fit every state perfectly.
- We need to prioritize some states with respect to others:

$$\eta^* = \arg\min_{\eta} \mathbb{E}_{s\sim\rho(s)}\left[D_p\left(P\ , P_\eta\right) + D_r\left(R, R_\eta\right)\right]$$

- Which states would you prioritize?

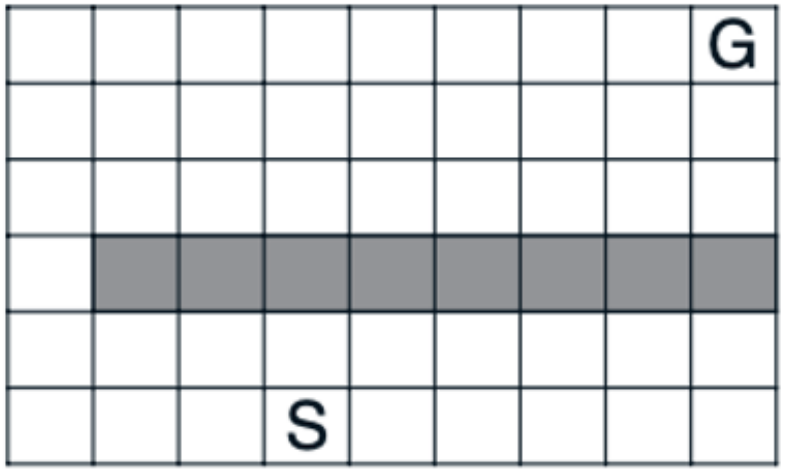
State Selection

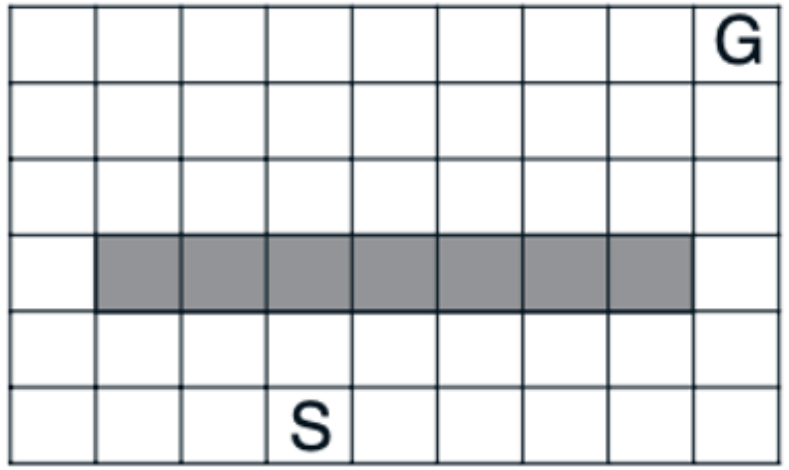
- We are mainly interested in the states observed by our policy π_θ
- And less interested in anything else

$$\eta^* = \arg\min_{\eta} \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[D_p\left(P\ , P_\eta\right) + D_r\left(R, R_\eta\right) \right]$$

- What are the issues with this choice?

Exploration vs Exploitation





State Selection

- We are mainly interested in the states observed by our policy π_θ
- But we also want to know about other states.

$$\eta^* = \arg\min_{\eta} \mathbb{E}_{\tau \sim p_\mu(\tau;\theta)} \left[D_p\left(P\ , P_\eta\right) + D_r\left(R, R_\eta\right) \right]$$

- Where p_μ represents a combination of trajectories from the current policy π_θ and an exploration policy μ
- The exact definition of p_μ varies between algorithms

Model Learning by Sampling

$$\eta^* = \arg\min_{\eta} \mathbb{E}_{\tau \sim p_\mu(\tau;\theta)} \left[D_p\left(P\ , P_\eta\right) + D_r\left(R, R_\eta\right) \right]$$

- In practice, we optimize η by sampling K trajectories $\tau_k = (s_0^k, a_0^k, r_0^k, \dots, s_T^k)$

$$\eta_P^* = \arg\min_{\eta} \sum_k \sum_t D_p\left(P(s_{t+1}^k | s_t^k, a_t^k)\ , P_\eta(s_{t+1}^k | s_t^k, a_t^k)\right)$$

$$\eta_R^* = \arg\min_{\eta} \sum_k \sum_t D_r\left(R(r_t^k | s_t^k, a_t^k)\ , R_\eta(r_t^k | s_t^k, a_t^k)\right)$$

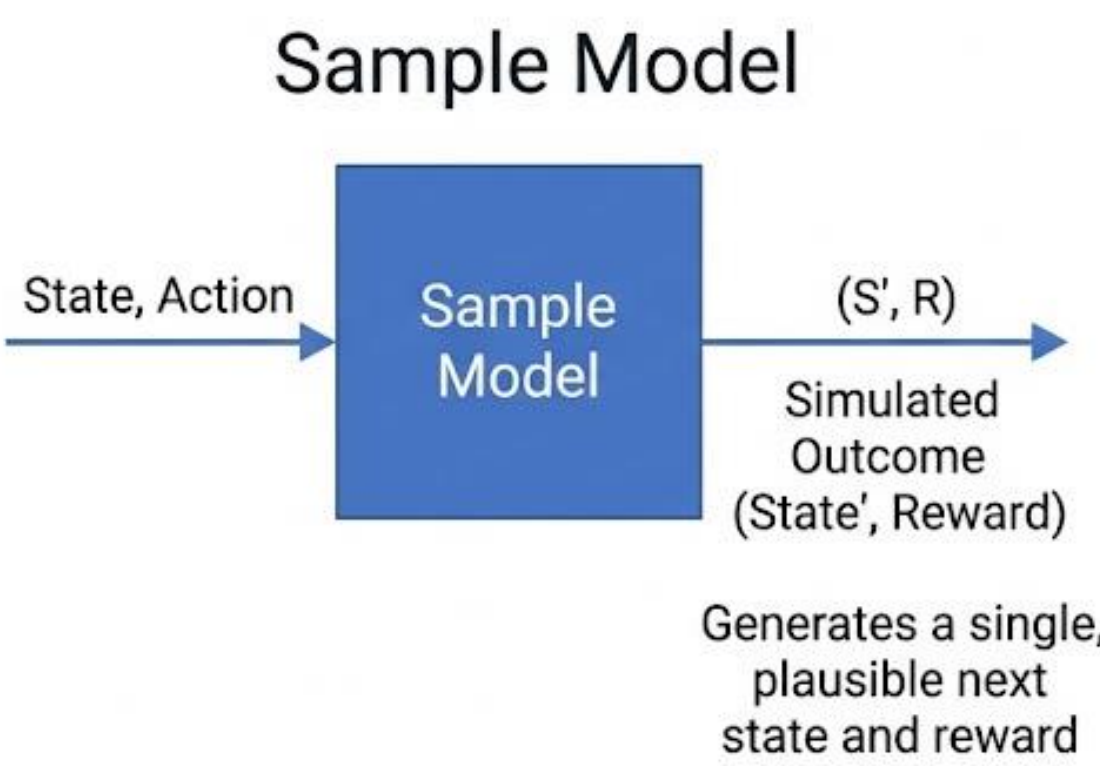
Classes of Models

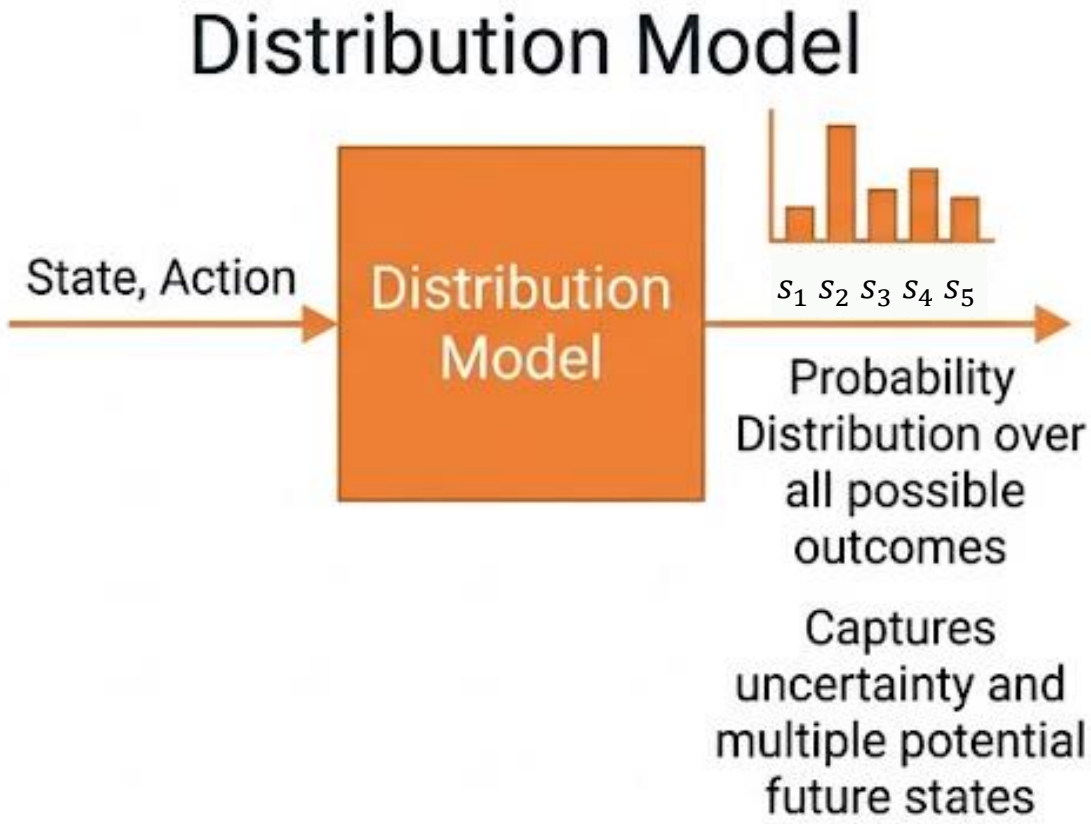
- We distinguish models according to two main categories

- What they output
  - Distributions or samples

- How they are built
  - From first principles
  - From data
  - Hybrid

Classes of Models: Output





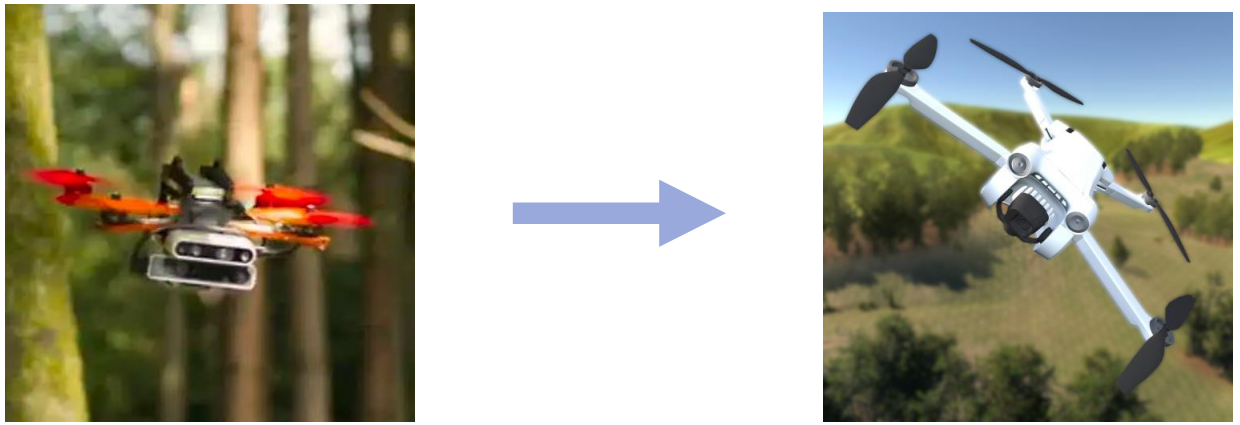
Classes of Models: Learning Signal

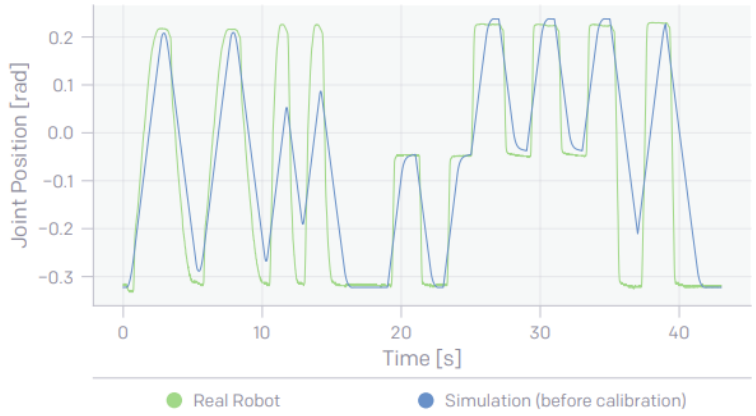
- Based on first principles or prior knowledge of the environment/task
  - Physics simulators for robots or testing sandboxes for LLMs
  - The parameters η represent features of the model (e.g., robot mass)

- Represented by generative models capable of producing *stochastic transitions* (why?)
  - Diffusion models, VAEs, etc.
  - The parameters η are the model weights.

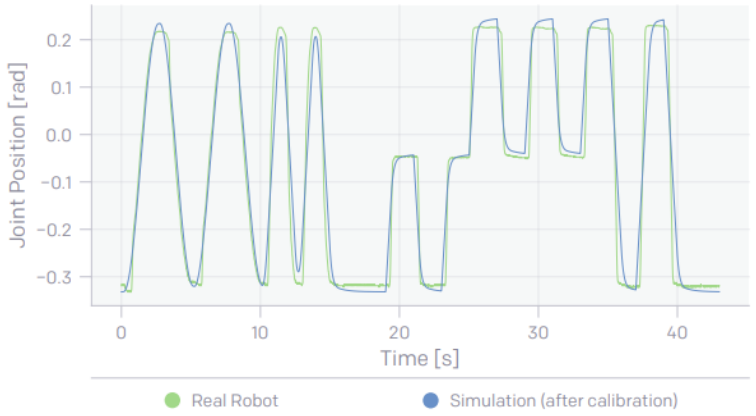
- A combination of the two (hybrid models)

First-principles Simulators





(a) Comparison against original simulation.



(b) Comparison against new simulation.

Source

Differentiable Model Learning

- Sample K trajectories $\tau_k = (s_0^k, a_0^k, r_0^k, \dots, s_T^k)$
- Define the losses

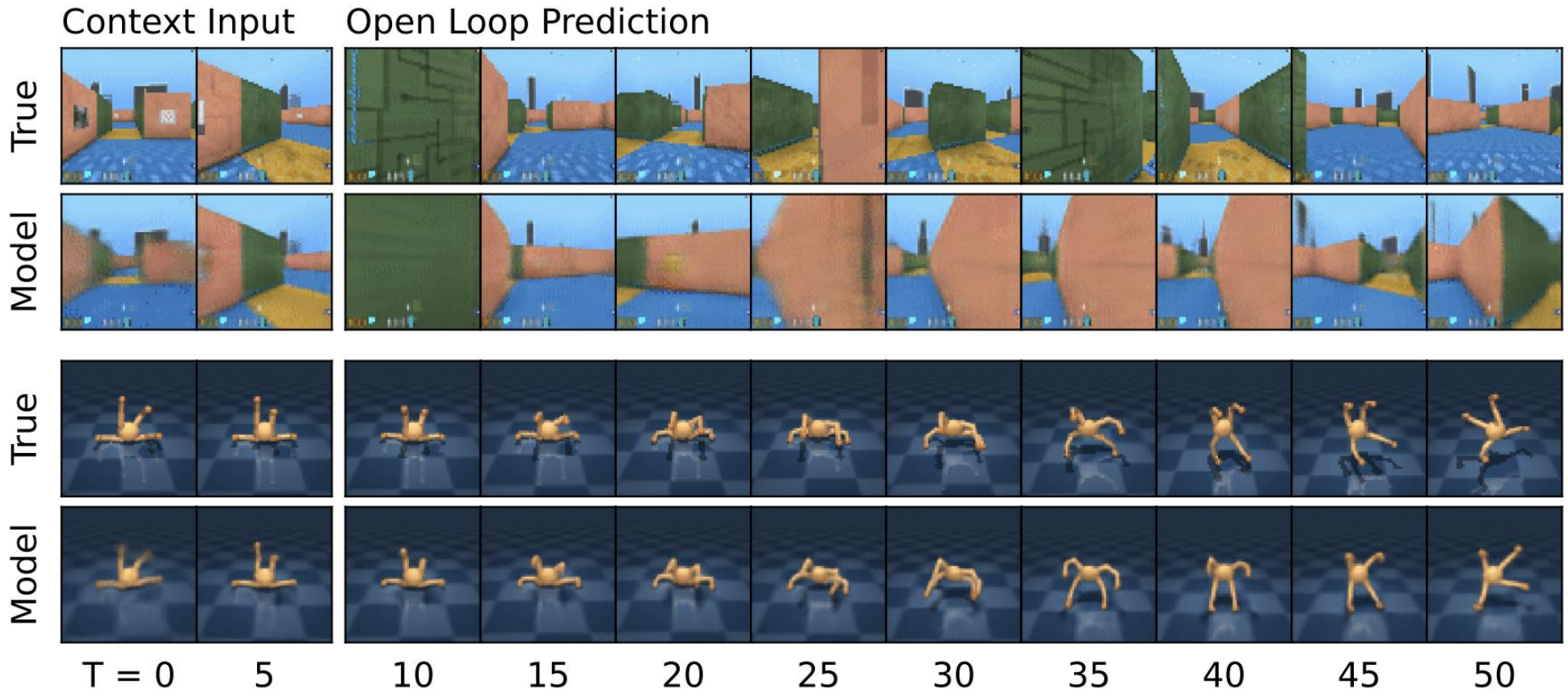
$$L_{\eta_P} = \sum_k \sum_t D_p\left(P(s_{t+1}^k | s_t^k, a_t^k), P_\eta(s_{t+1}^k | s_t^k, a_t^k)\right)$$

$$L_{\eta_R} = \sum_k \sum_t D_r\left(R(r_t^k | s_t^k, a_t^k), R_\eta(r_t^k | s_t^k, a_t^k)\right)$$

- Update the parameters with gradient descent

$$\eta_P \leftarrow \eta_P - \alpha \nabla L_{\eta_P}$$
$$\eta_R \leftarrow \eta_R - \alpha \nabla L_{\eta_R}$$

Example: Generative Models



Source

Model Learning: Summary

- Define a parametric model $\mathcal{M}(\eta)$.
- Optimize η to match real-world environmental interactions as closely as possible.
  - Using gradient descent if the model is differentiable

- Focus on the states that are most closely related to the current π
  - But also wander around

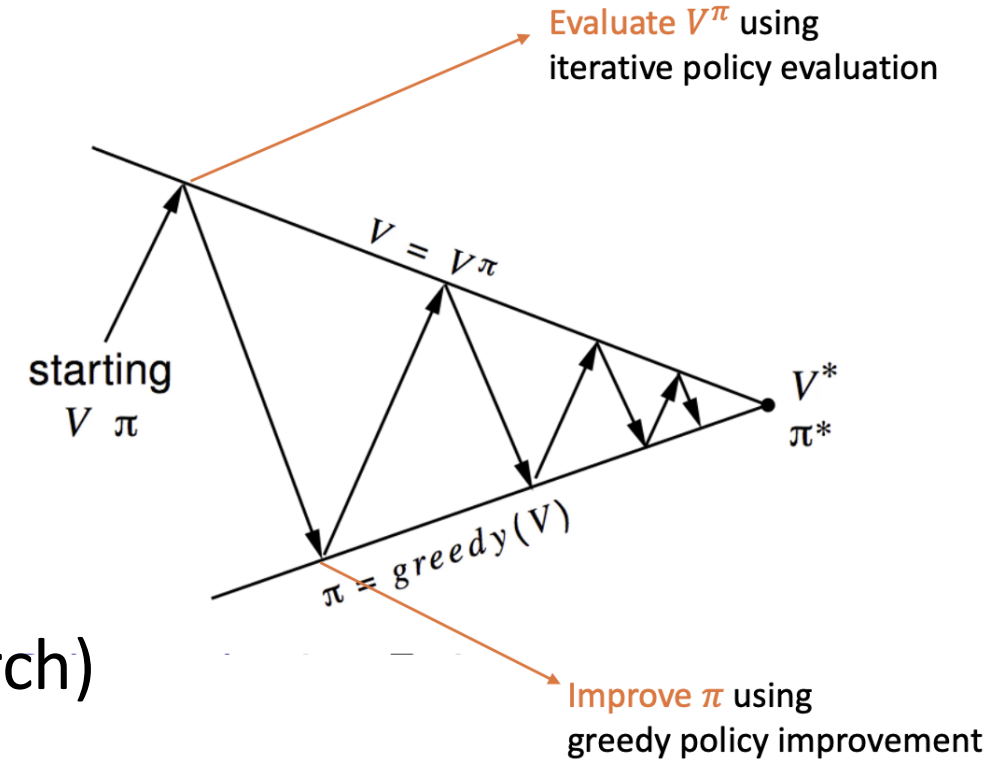
- A model is *always* inaccurate!

Planning in a Model with Model-Based Control

- Once we have a model $\mathcal{M}_\eta$ of the environment
- We can use dynamic programming algorithms to solve the MDP

- Examples:
  - Value Iteration
  - Policy Iteration
  - …

- Or any other model-based optimization approach (e.g., tree search)



Planning in a Model with Model-Free Control

- Use the model $\mathcal{M}_\eta$ only to generate samples (s, a, r, s')
- Where:
  - $s' = P_\eta(s'|a, s)$ $\quad\quad$ $r = R_\eta(r|a, s)$

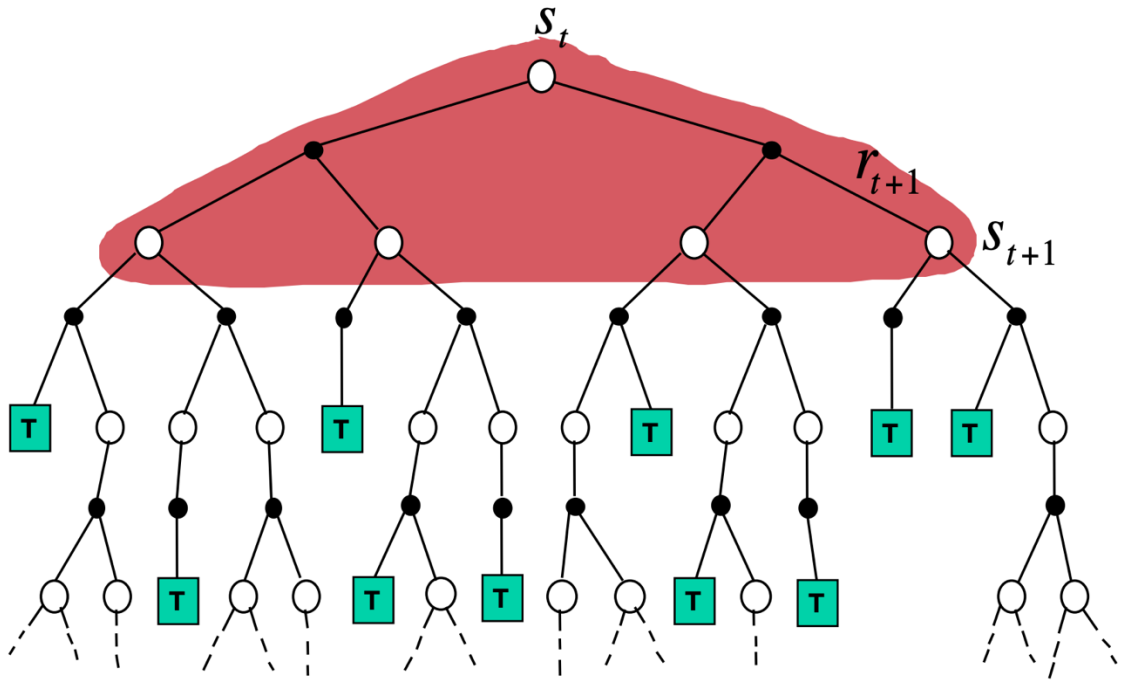
- Then, use model-free RL on these samples
  - Monte-Carlo Control
  - Sarsa
  - Q-Learning
  - …

- This is often called *sample-based planning*

Why sampling-based planning in a *known* model?

- If the model has a *large branching factor*, evaluating $Q(s, a)$ at every state could be very computationally expensive.

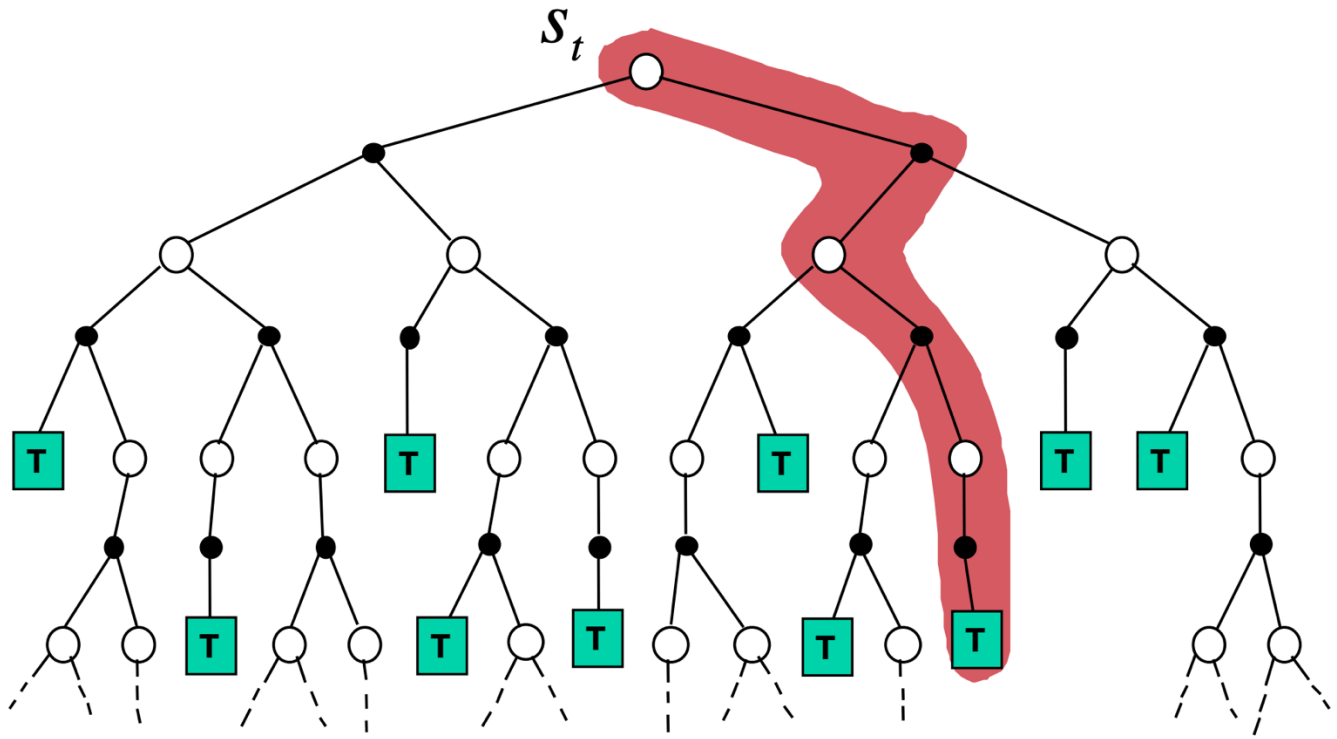
$$Q(s_t, a_t) \leftarrow \mathbb{E}_\pi[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)]$$



Why sampling-based planning in a *known* model?

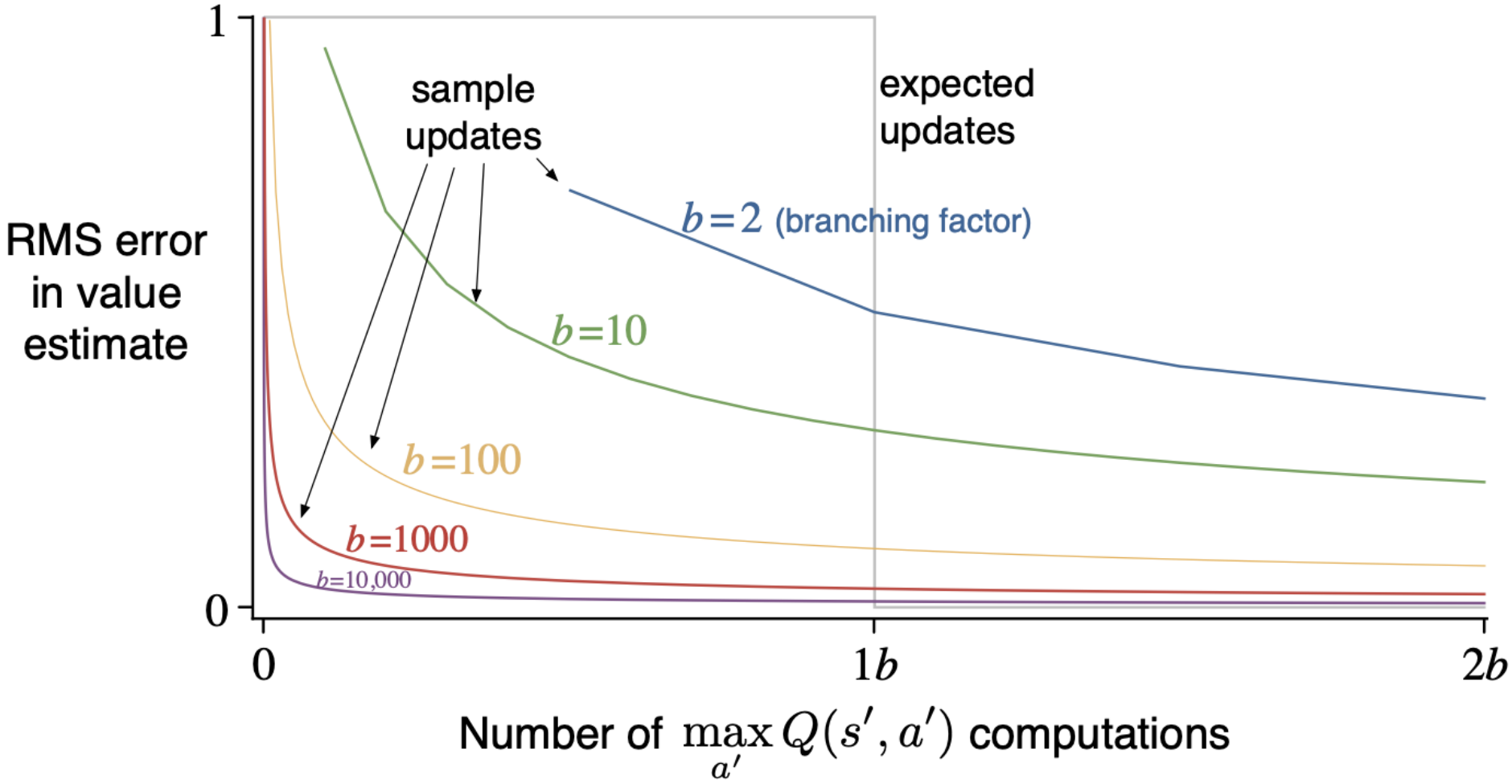
- With the same computational budget, one could sample many states and get a better estimate of $Q(s, a)$ *faster*.

- Focus on the states that matter to the policy (on-policy distribution)



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right)$$

Example: Planning in a Model

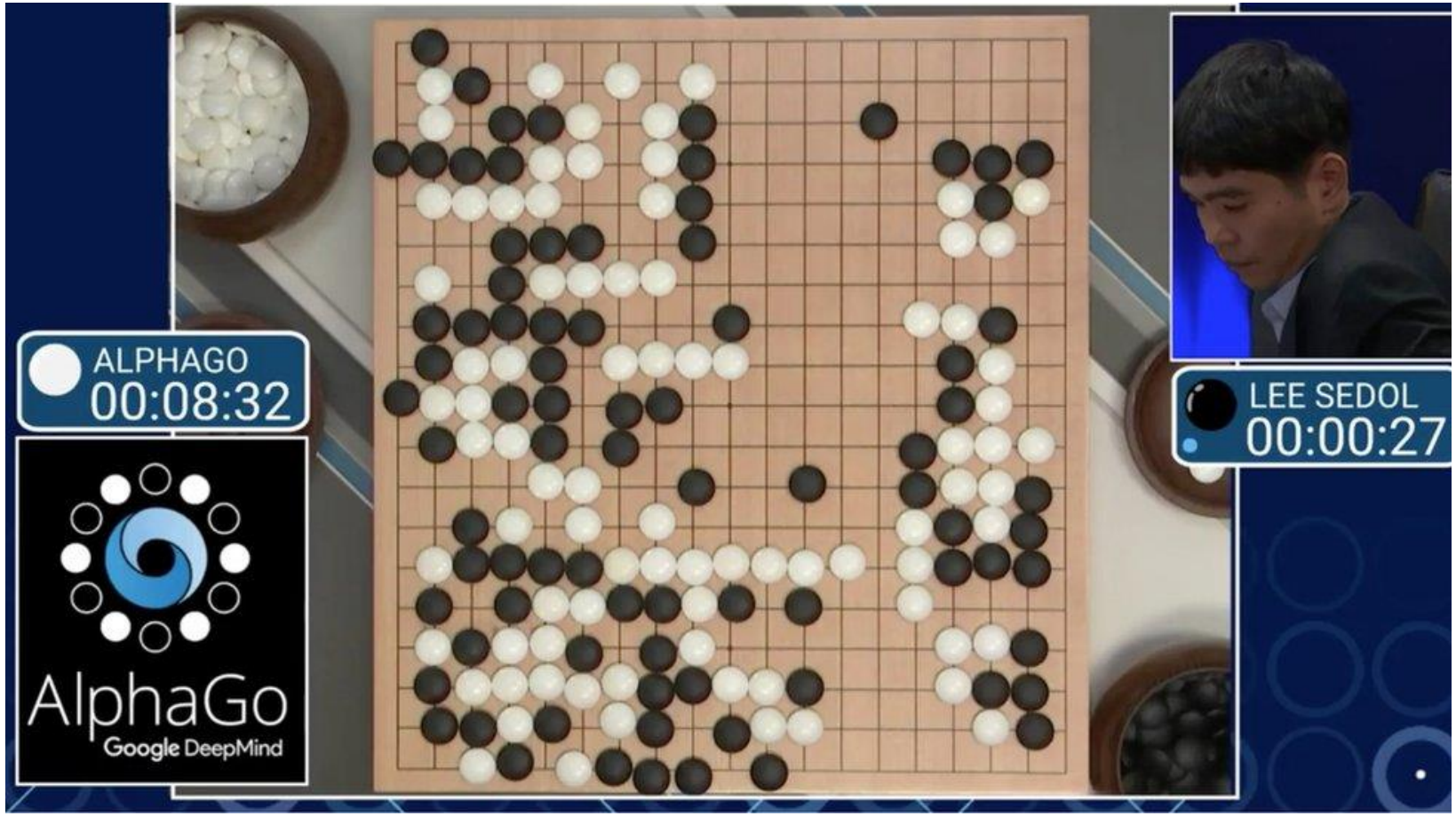


Background vs Decision-Time Planning

- We can use a model to find the optimal value function/policy
- And then just use the policy in the real environment
- This is called *background* planning.

- Another option is to plan at every real state s_t the policy finds itself in.
- This is called *decision-time* planning.
- The latter can be done only if latency is not a problem (e.g., chess).

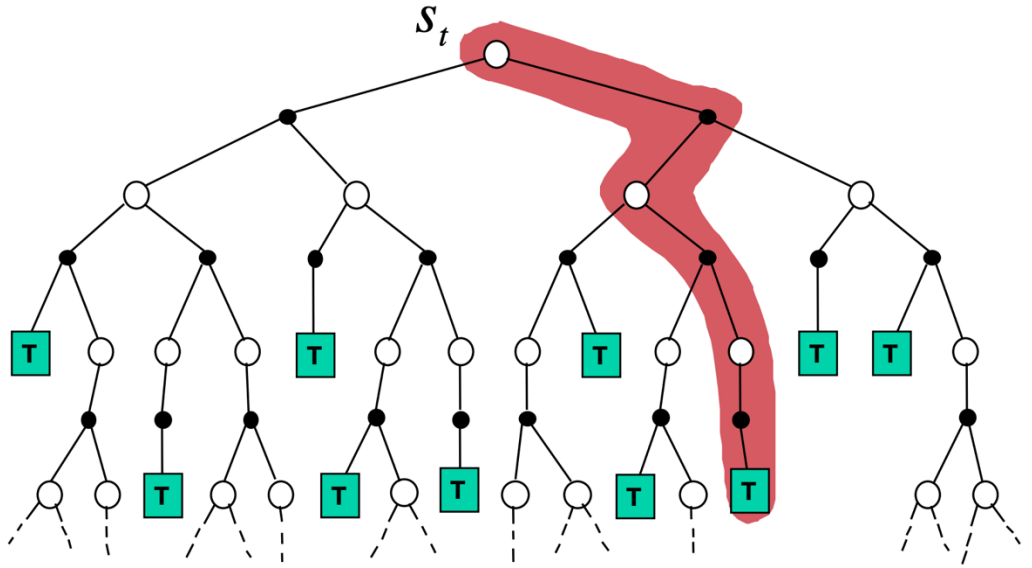
Example: AlphaGo



Model-Based RL: AlphaGo

- Very easy to build a perfect Go model (just use any emulator).
- The branching factor is very high (and depends on the opponent's actions).
- Too expensive to compute the optimal action given the model.

- Sampling to the rescue!



Monte-Carlo Tree Search (simplified)

- Algorithm:
  - For every state s_t you find yourself in.
  - For every action $a \in A$, simulate K episodes of experience from s_t by sampling through the model

$$\{s_t, a, s_{t+1}^k, \dots, s_T^k\}_{k=1}^K \sim \mathcal{M}_\eta$$

  - Evaluate actions by their return (Monte-Carlo Evaluation)

$$Q(s_t, a) = \frac{1}{K} \sum_{k=1}^{K} \left(r(s_t, a) + \sum_{l=1}^{T-1} \gamma^l r(s_{t+l}^k, a_{t+l}^k) \right)$$

  - Select the current real action a_t as the one with the maximum value

$$a_t = \arg\max_{a \in A} Q(s_t, a)$$

- Avoids the problem of globally approximating a Q function!

Summary: Model-based RL

- Planning requires a model of the environment
  - Either we build it from prior data/knowledge
  - Or we learn it on the go with experience
  - The more we know in advance, the less we need to learn.

- Planning with sampling is often more efficient than planning with expected updates (as in dynamic programming)
  - In particular, for models with large branching factors

- Planning and learning a policy are surprisingly close concepts
  - Both estimate the value function. The main difference is where the experience is from.
  - They can naturally be integrated

A Unified Perspective on RL

