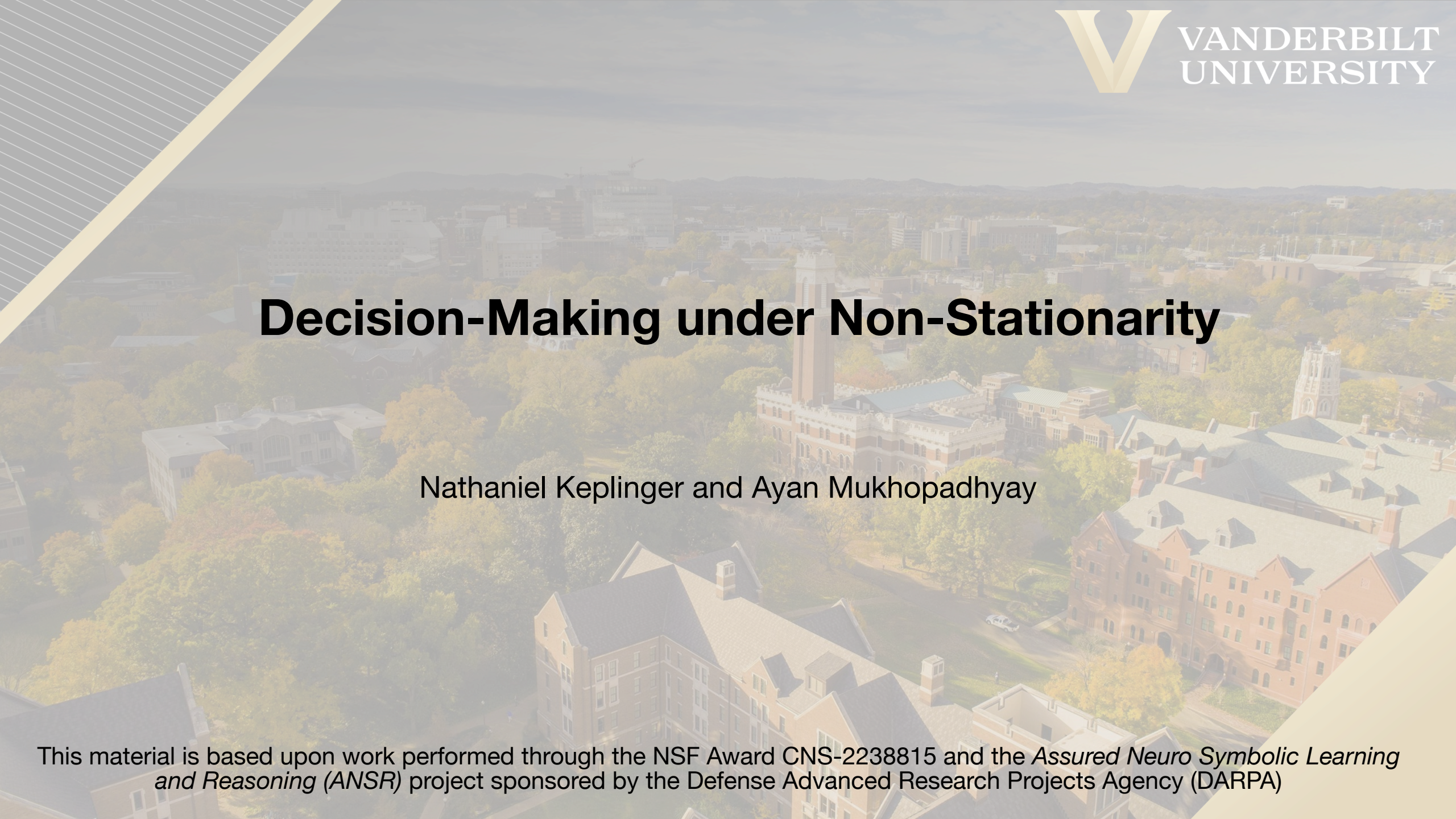


Decision-Making under Non-Stationarity

Nathaniel Keplinger and Ayan Mukhopadhyay

This material is based upon work performed through the NSF Award CNS-2238815 and the *Assured Neuro Symbolic Learning and Reasoning (ANSR)* project sponsored by the Defense Advanced Research Projects Agency (DARPA)

Why study stochastic control processes?

Many real-world problems involve making sequential decisions over time under uncertainty

- Autonomous driving (Kiran et al. 2021)
- Medical diagnosis and treatment (Yu et al. 2021)
- Emergency response (Mukhopadhyay et al. 2022)
- Vehicle Routing (Li, Yan, and Wu 2021)
- Financial portfolio optimization (Pendharkar and Cusatis 2018)

Modeling Autonomous Agents for Sequential Decision-Making

An *agent* in an entity capable of computation that acts or makes decisions based on observations from the environment.

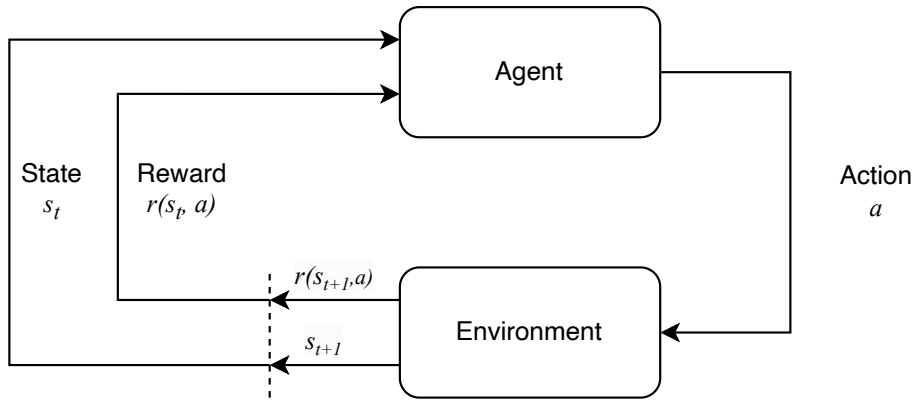
We can model how agents make decision as a Markov Decision Process (MDP).

Markov Decision Processes: A Primer

A general way to model a sequential decision making problem is a **Markov Decision process (MDP).**

An **MDP** consists of the following components:

- **State ($\mathcal{S}$)**: This element summarizes our current "position" in the world or the current conditions of the system that we are interested in.
- **Actions ($\mathcal{A}$)**: The set of choices that the agent has access to. Note that all actions might not be available in all states.
- **Transition**: A probability distribution over the future state, given the current state and action, i.e., $P(s' \mid s, a)$.
- **Rewards**: A reward function, $R(s, a, s')$, that quantifies the instantaneous reward the agent gets by being in state s, taking action a and transitioning to state s'.



Markov Decision Processes: A Primer

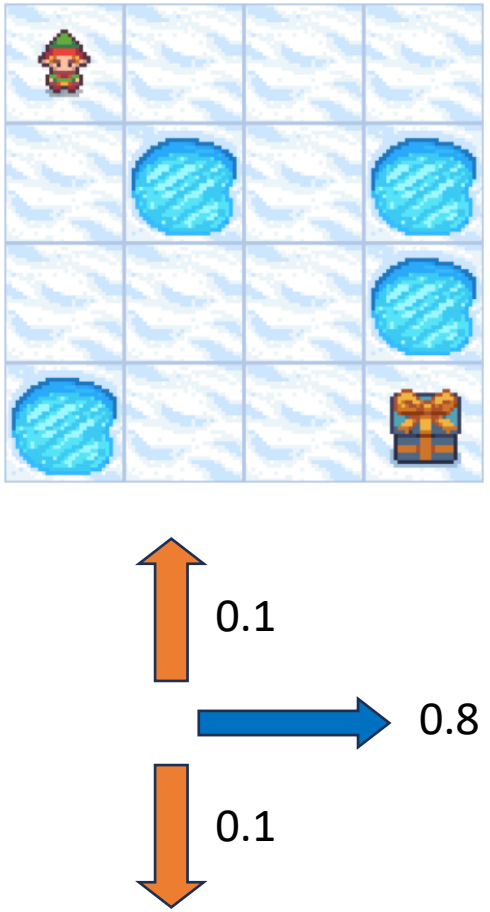
Let us look at a simple example. Consider the Frozen Lake environment from OpenAI Gym.

State: The location of the agent, holes, and goal.

Actions: Go up, right, left, or down.

Transitions: The agent can transition to a new cell corresponding to an action with some chance of transitioning to a perpendicular cell instead.

Rewards: There is a positive reward if we reach the goal.



What is missing?

Unfortunately, the real world is not stationary.

What is a stationary stochastic process?

- A stochastic process whose unconditional joint probability distribution does not change when shifted in time.
- Formally, let $\{X_t\}$ be a stochastic process and let $F_X(x_{t_1+\tau}, \ldots, x_{t_n+\tau})$ represent the cumulative distribution function of the marginal distribution (i.e., with no reference to any particular starting value).
- Then, $\{X_t\}$ is said to be "strictly stationary", if

$$F_X(x_{t_1+\tau}, \ldots, x_{t_n+\tau}) = F_X(x_{t_1}, \ldots, x_{t_n}) \quad \text{for all } \tau, t_1, \ldots, t_n \in \mathbb{R} \text{ and for all } n \in \mathbb{N}_{>0}$$

Since τ does not affect $F_X(\,\cdot\,)$, F_X is independent of time.

Endogenous vs Exogenous Uncertainty

Endogenous Uncertainty:

This uncertainty arises from the decision-making process itself and is influenced by the actions and strategies of the decision-maker.

For a stochastic control process, this uncertainty could be a stochastic policy or uncertainty modeled by the transition or reward functions.

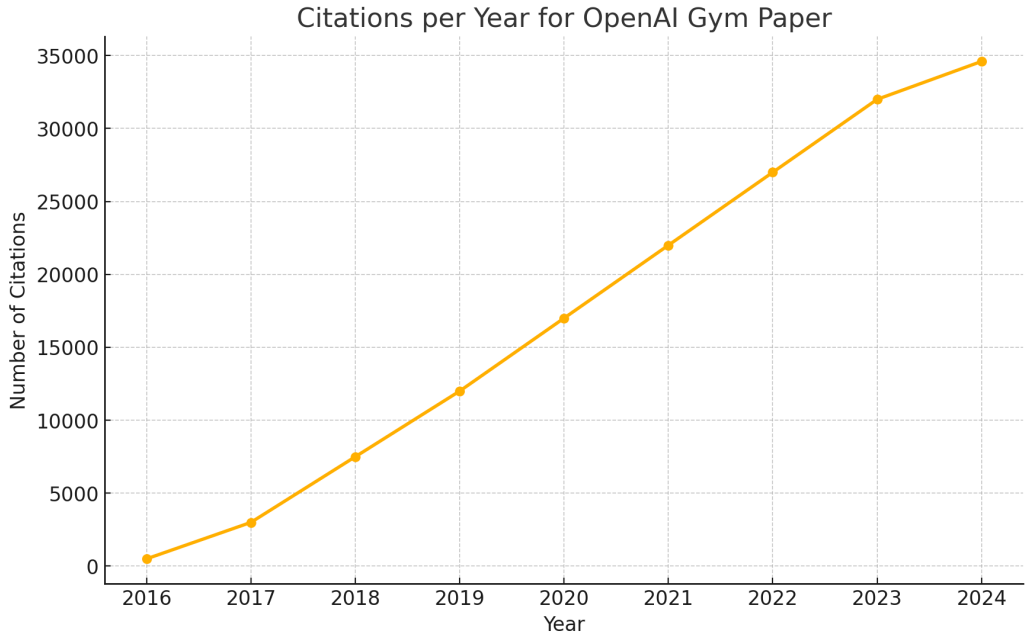
Exogenous Uncertainty:

This type of uncertainty originates from external factors outside the decision-maker's control.

For a stochastic control process, think about what happens if the environment itself changes.

Exploring Non-Stationarity in Decision-Theory

- Non-stationarity has been well explored from the decision-making and control perspectives.
- Unfortunately, decision-theoretic methods have not adopted any unifying theme.
- Is a unifying *theme* critical?
  - Perhaps not, heterogeneous definitions are critical as application domains have diverse requirements.
  - However, a general mathematical model is important for propelling the field.
  - General mathematical models can also be used for develop standardized benchmarks.



Historical Treatment of Non-Stationarity

Early Conceptualization:

- Switching Environments (Ackerson & Fu, 1970):
  - Mean and covariance change over time.

Formalizing Non-Stationarity:

- Sojourn-Time-Dependent Markov Chains (Campo, Mookerjee, & Bar-Shalom, 1991):
  - Semi-Markovian processes with parameter changes after random intervals.

Recent Developments:

- Adapting to a Single Change:
  - Observed changes (Pettet and Zhang et al., 2024) or unobserved changes (Luo et al., 2024).
- Continuous Parameter Changes:
  - Modeled by Lecarpentier & Rachelson (2019).
- Forecast-Based Optimization (Chandak et al., 2020):
  - Maximize future policy performance instead of directly modeling non-stationarity.

What does it come down to?

There are four questions of importance:

1. **What** changes?
2. **How** does it change?
3. Does the agent **detect** the change?
4. Does the agent **know** the change?

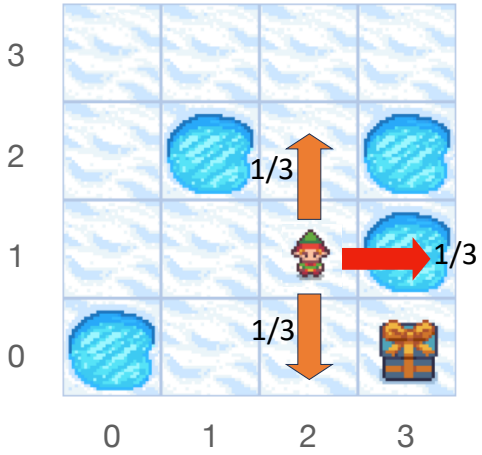
Prior work on non-stationary Markov decision processes

Model	Reference	Is the change notified?	Is the change known?	What changes?	Nature of the change	Is the change bounded?
Piecewise Stationary MAB	Garivier and Moulines (2011)	No	No	Reward	The reward distribution is fixed over certain time periods, and then changes at unknown time steps.	No
Non-stationary MAB	Besbes, Gur, and Zeevi (2014)	No	No	Reward	The reward can change at arbitrary time points.	Yes
Piecewise Stationary MDP	Auer, Jaksch, and Ortner (2008)	No	No	Transition, Reward	Bounded change analyzed as part of the UCRL2 algorithm	Yes
Non-Stationary MDP	Cheung, Simchi-Levi, and Zhu (2020)	N/A	No	Transition, Reward	The reward and transition can change at every time step	Yes
Non-Stationary MDP	Chandak et al. (2020b)	Yes	No	Transition, Reward	Transition and reward can change after each episode, but remain fixed within an episode	No
Non-Stationary MDP	Chandak et al. (2020a)	Yes	No	Transition, Reward	Transition and reward can change after each episode, but remain fixed within an episode	Yes
Non-Stationary MDP	Lecarpentier and Rachelson (2019)	Yes	Yes	Transition	The agent knows the current parameters, but not the future evolution.	Yes
Non-Stationary MDP	Pettet et al. (2024)	Yes	Yes	Transition	A single discrete change	Yes
Non-Stationary MDP	Luo et al. (2024)	Yes	No	Transition	A single discrete change	No
Non-Stationary Bandits with Periodic Variation	Chakraborty and Shettiwar (2024)	No	No	Reward	Periodic Variation	Yes

Example: Transition-bounded MDPs

- Consider the transition probability function $P_t(s' \mid s, a)$, where the subscript t denotes the time step under consideration. Now, consider that the environment undergoes *some* change between time steps 0 and t. We introduce **transition-bounded non-stationary Markov decision processes (T-NSMDP)**, where the measure of the non-stationarity is:

$$\forall s, a : \sum_{s' \in S} |P_t(s' \mid a, s) - P_0(s' \mid a, s)| \leq \eta$$



$Q_t^{\pi}(s, a)$ The expected discounted sum of all future rewards when starting from state s, taking action a, and following policy π thereafter.

Theorem 1: If $\forall s, a : \sum_{s' \in S} \left| P_t(s' \mid a, s) - P_0(s' \mid a, s) \right| \leq \eta$, and $\forall s, a : |r(s, a)| \leq R$, and the discount factor $\gamma < 1$, then

$$|Q_0^{\pi_0^*}(s, a) - Q_t^{\pi_t^*}(s, a)| \leq \frac{\gamma \cdot \eta \cdot R}{(1-\gamma)^2}, \forall s, a \quad \epsilon$$

Expected Long-Term Rewards by taking action a and following the optimal policy at time 0

Expected Long-Term Rewards by taking action a and following the optimal policy at time t

Now, how do we make decisions?

Now, how do we make decisions?

- Our goal is to find optimal actions at execution time t under the following conditions:

Now, how do we make decisions?

- Our goal is to find optimal actions at execution time t under the following conditions:
    - We are given an optimal action-value function $Q_0^{\pi_0^*}(s, a)$ for time step 0 but not at execution time t.

Now, how do we make decisions?

- Our goal is to find optimal actions at execution time t under the following conditions:
  - We are given an optimal action-value function $Q_0^{\pi_0^*}(s, a)$ for time step 0 but not at execution time t.
  - The environmental change is bounded by η.

Now, how do we make decisions?

- Our goal is to find optimal actions at execution time t under the following conditions:
  - We are given an optimal action-value function $Q_0^{\pi_0^*}(s, a)$ for time step 0 but not at execution time t.
  - The environmental change is bounded by η.
  - For now, we assume that the agent “knows” the environmental model at time t.

Now, how do we make decisions?

- Our goal is to find optimal actions at execution time t under the following conditions:
  - We are given an optimal action-value function $Q_0^{\pi_0^*}(s, a)$ for time step 0 but not at execution time t.
  - The environmental change is bounded by η.
  - For now, we assume that the agent “knows” the environmental model at time t.
  - There is a limited computational budget at execution time that prevents re-learning an optimal policy.

Now, how do we make decisions?

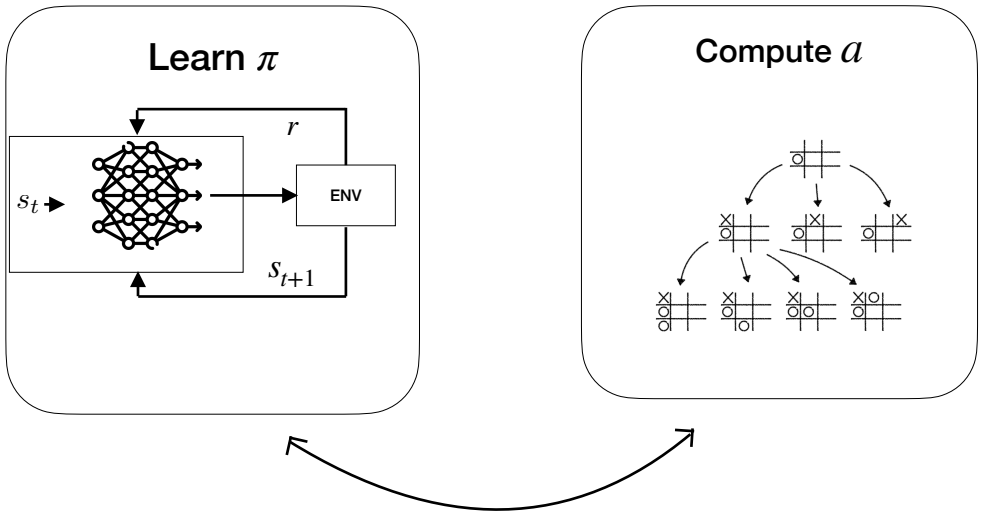
- Our goal is to find optimal actions at execution time t under the following conditions:
  - We are given an optimal action-value function $Q_0^{\pi_0^*}(s, a)$ for time step 0 but not at execution time t.
  - The environmental change is bounded by η.
  - For now, we assume that the agent "knows" the environmental model at time t.
  - There is a limited computational budget at execution time that prevents re-learning an optimal policy.

Key Insight: If the world has not changed much, should we throw away the knowledge about how to act?

Now, how do we make decisions?

- Our goal is to find optimal actions at execution time t under the following conditions:
  - We are given an optimal action-value function $Q_0^{\pi_0^*}(s, a)$ for time step 0 but not at execution time t.
  - The environmental change is bounded by η.
  - For now, we assume that the agent "knows" the environmental model at time t.
  - There is a limited computational budget at execution time that prevents re-learning an optimal policy.

Policy-Augmented Monte Carlo Tree Search (PA-MCTS)



Key Insight: If the world has not changed much, should we throw away the knowledge about how to act?

$$\text{argmax}_{a \in \mathcal{A}_s} \alpha Q_0^{\pi_0^*}(s, a) + (1 - \alpha)\overline{G}_t(s, a)$$

Existing knowledge, optimal in the original environment

Should the agent trust the old knowledge or the new computation?

Online computation in the new environment

Déjà vu

Déjà vu

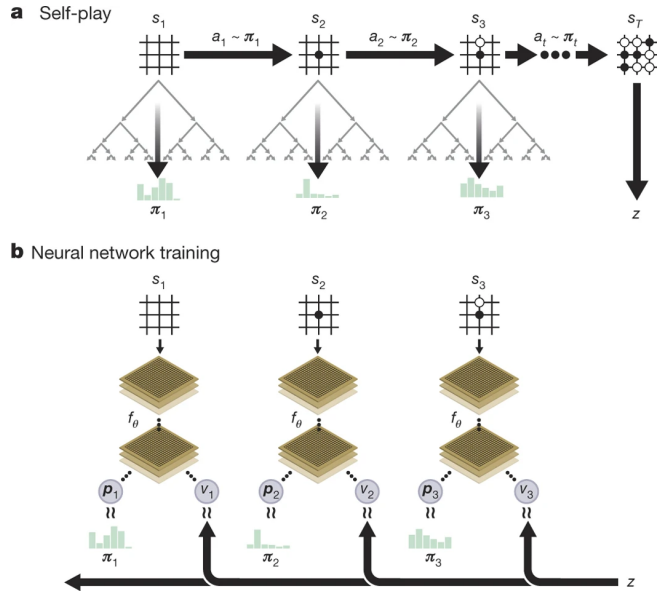
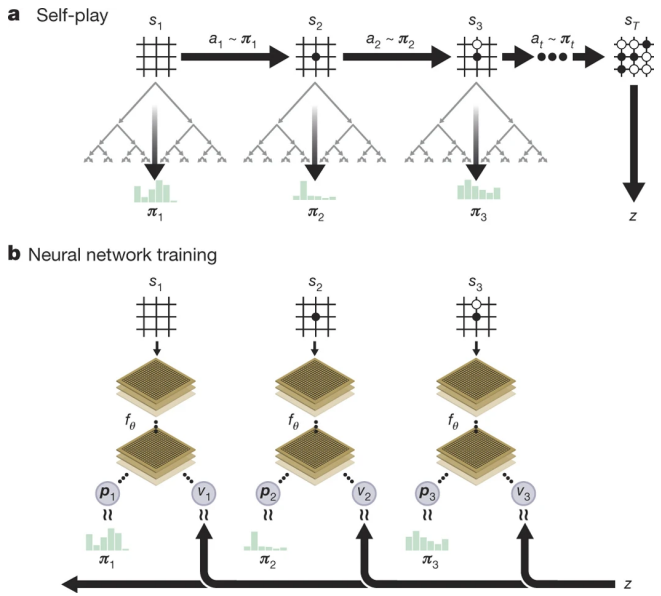


Image Source: Silver et al., Mastering the game of Go without human knowledge. *Nature* (2017)

Déjà vu



- Key Differences:
  - Combining an existing policy and online search “outside the tree” stabilizes search under non-stationarity.
  - This stabilization also allows significantly faster convergence than AlphaZero.
  - We show experimentally that this combination comprehensively outperforms AlphaZero in multiple domains.
  - Unlike AlphaZero, we present strong theoretical guarantees!!

Image Source: Silver et al., Mastering the game of Go without human knowledge. *Nature* (2017)

Can we guarantee the quality of decisions?

Can we guarantee the quality of decisions?

- Let $a_t^* := \text{argmax}_{a \in \mathcal{A}_s} Q^{\pi_t^*}(s, a)$ be the optimal action at an arbitrary time step t.

Can we guarantee the quality of decisions?

- Let $a_t^* := \text{argmax}_{a \in \mathcal{A}_s} Q^{\pi_t^*}(s, a)$ be the optimal action at an arbitrary time step t.
- Let $a_t' := \text{argmax}_{a \in \mathcal{A}_s \setminus \{a_t^*\}} Q_t^{\pi_t^*}(s, a)$ denote the second best action at an arbitrary time step t.

Can we guarantee the quality of decisions?

- Let $a_t^* := \text{argmax}_{a \in \mathcal{A}_s} Q^{\pi_t^*}(s, a)$ be the optimal action at an arbitrary time step t.
- Let $a_t' := \text{argmax}_{a \in \mathcal{A}_s \setminus \{a_t^*\}} Q_t^{\pi_t^*}(s, a)$ denote the second best action at an arbitrary time step t.
- Let $\psi_t(s) := Q_t^{\pi_t^*}(s, a_t^*) - Q_t^{\pi_t^*}(s, a_t')$, which, for a state s, denotes the difference in the Q values when taking actions a_t^* and a_t' at time t and following the optimal policy π_t^* thereafter.

Can we guarantee the quality of decisions?

- Let $a_t^* := \text{argmax}_{a \in \mathcal{A}_s} Q^{\pi_t^*}(s, a)$ be the optimal action at an arbitrary time step t.
- Let $a_t' := \text{argmax}_{a \in \mathcal{A}_s \setminus \{a_t^*\}} Q_t^{\pi_t^*}(s, a)$ denote the second best action at an arbitrary time step t.
- Let $\psi_t(s) := Q_t^{\pi_t^*}(s, a_t^*) - Q_t^{\pi_t^*}(s, a_t')$, which, for a state s, denotes the difference in the Q values when taking actions a_t^* and a_t' at time t and following the optimal policy π_t^* thereafter.
- Let δ denote the bound on the error of the values estimated by MCTS when it is stopped, i.e., $\left| Q_t^{\pi_t^*}(s, a) - \overline{G}_t(s, a) \right|_\infty \leq \delta$

Can we guarantee the quality of decisions?

- Let $a_t^* := \text{argmax}_{a \in \mathcal{A}_s} Q^{\pi_t^*}(s, a)$ be the optimal action at an arbitrary time step t.
- Let $a_t' := \text{argmax}_{a \in \mathcal{A}_s \setminus \{a_t^*\}} Q_t^{\pi_t^*}(s, a)$ denote the second best action at an arbitrary time step t.
- Let $\psi_t(s) := Q_t^{\pi_t^*}(s, a_t^*) - Q_t^{\pi_t^*}(s, a_t')$, which, for a state s, denotes the difference in the Q values when taking actions a_t^* and a_t' at time t and following the optimal policy π_t^* thereafter.
- Let δ denote the bound on the error of the values estimated by MCTS when it is stopped, i.e., $\left| Q_t^{\pi_t^*}(s, a) - \overline{G}_t(s, a) \right|_\infty \leq \delta$

Theorem 2: If $\alpha\epsilon + (1 - \alpha)\delta \leq \frac{\psi_t(s)}{2}$, PA-MCTS is guaranteed to select the optimal action at time step t in state s.

How much has the change affected Q?

What is the quality of my online computation?

How different are the top actions?

Can we guarantee the quality of decisions?

- Let $a_t^* := \text{argmax}_{a \in \mathcal{A}_s} Q^{\pi_t^*}(s, a)$ be the optimal action at an arbitrary time step t.
- Let $a_t' := \text{argmax}_{a \in \mathcal{A}_s \setminus \{a_t^*\}} Q_t^{\pi_t^*}(s, a)$ denote the second best action at an arbitrary time step t.
- Let $\psi_t(s) := Q_t^{\pi_t^*}(s, a_t^*) - Q_t^{\pi_t^*}(s, a_t')$, which, for a state s, denotes the difference in the Q values when taking actions a_t^* and a_t' at time t and following the optimal policy π_t^* thereafter.
- Let δ denote the bound on the error of the values estimated by MCTS when it is stopped, i.e., $\left| Q_t^{\pi_t^*}(s, a) - \overline{G}_t(s, a) \right|_\infty \leq \delta$

Theorem 2: If $\alpha\epsilon + (1 - \alpha)\delta \leq \frac{\psi_t(s)}{2}$, PA-MCTS is guaranteed to select the optimal action at time step t in state s.

How much has the change affected Q?

What is the quality of my online computation?

How different are the top actions?

Are we done?

Can we guarantee the quality of decisions?

- Let $a_t^* := \text{argmax}_{a \in \mathcal{A}_s} Q^{\pi_t^*}(s, a)$ be the optimal action at an arbitrary time step t.
- Let $a_t' := \text{argmax}_{a \in \mathcal{A}_s \setminus \{a_t^*\}} Q_t^{\pi_t^*}(s, a)$ denote the second best action at an arbitrary time step t.
- Let $\psi_t(s) := Q_t^{\pi_t^*}(s, a_t^*) - Q_t^{\pi_t^*}(s, a_t')$, which, for a state s, denotes the difference in the Q values when taking actions a_t^* and a_t' at time t and following the optimal policy π_t^* thereafter.
- Let δ denote the bound on the error of the values estimated by MCTS when it is stopped, i.e., $\left| Q_t^{\pi_t^*}(s, a) - \overline{G}_t(s, a) \right|_\infty \leq \delta$

Theorem 2: If $\alpha\epsilon + (1 - \alpha)\delta \leq \frac{\psi_t(s)}{2}$, PA-MCTS is guaranteed to select the optimal action at time step t in state s.

How much has the change affected Q?

What is the quality of my online computation?

How different are the top actions?

Are we done? Unfortunately not, the agent does not observe $\psi_t(s)$

Can we guarantee the quality of decisions?

- Let $a_t^* := \text{argmax}_{a \in \mathcal{A}_s} Q^{\pi_t^*}(s, a)$ be the optimal action at an arbitrary time step t.
- Let $a_t' := \text{argmax}_{a \in \mathcal{A}_s \setminus \{a_t^*\}} Q_t^{\pi_t^*}(s, a)$ denote the second best action at an arbitrary time step t.
- Let $\psi_t(s) := Q_t^{\pi_t^*}(s, a_t^*) - Q_t^{\pi_t^*}(s, a_t')$, which, for a state s, denotes the difference in the Q values when taking actions a_t^* and a_t' at time t and following the optimal policy π_t^* thereafter.
- Let δ denote the bound on the error of the values estimated by MCTS when it is stopped, i.e., $\left| Q_t^{\pi_t^*}(s, a) - \overline{G}_t(s, a) \right|_\infty \leq \delta$

Theorem 2: If $\alpha\epsilon + (1 - \alpha)\delta \leq \frac{\psi_t(s)}{2}$, PA-MCTS is guaranteed to select the optimal action at time step t in state s.

How much has the change affected Q?

What is the quality of my online computation?

How different are the top actions?

Are we done? Unfortunately not, the agent does not observe $\psi_t(s)$

Corollary 2.1: $\psi_t(s) \leq \psi_0(s) + 2\epsilon$

If the agent knows how to act in the original environment, we can easily compute this quantity.

It turns out that we can connect $\psi_t(s)$ and $\psi_0(s)$.

How well does this work?

DDQN

Current State: [0, 0]
Next State: [0, 0]
Decision: down
Decision Q Value Estimate: 0.0

Success Rate: 0.12 ± 0.01

AlphaZero

Current State: [0, 0]
Next State: [0, 0]
Decision: down
Decision Q Value Estimate: 0.0

Success Rate: 0.114 ± 0.011

MCTS

Current State: [0, 0]
Next State: [0, 0]
Decision: down
Decision Q Value Estimate: 0.0

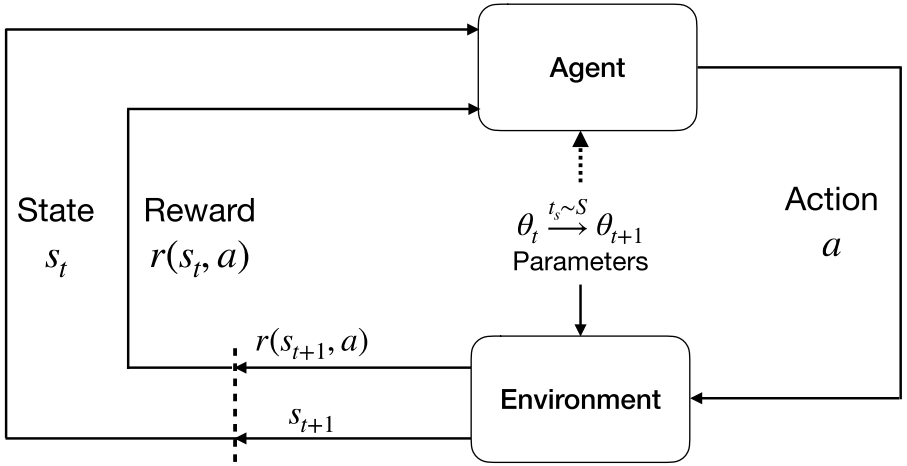
Success Rate: 0.866 ± 0.01

Policy-Augmented Search

Current State: [0, 0]
Next State: [0, 0]
Decision: down
Decision Q Value Estimate: 0.0

Success Rate: 0.936 ± 0.009

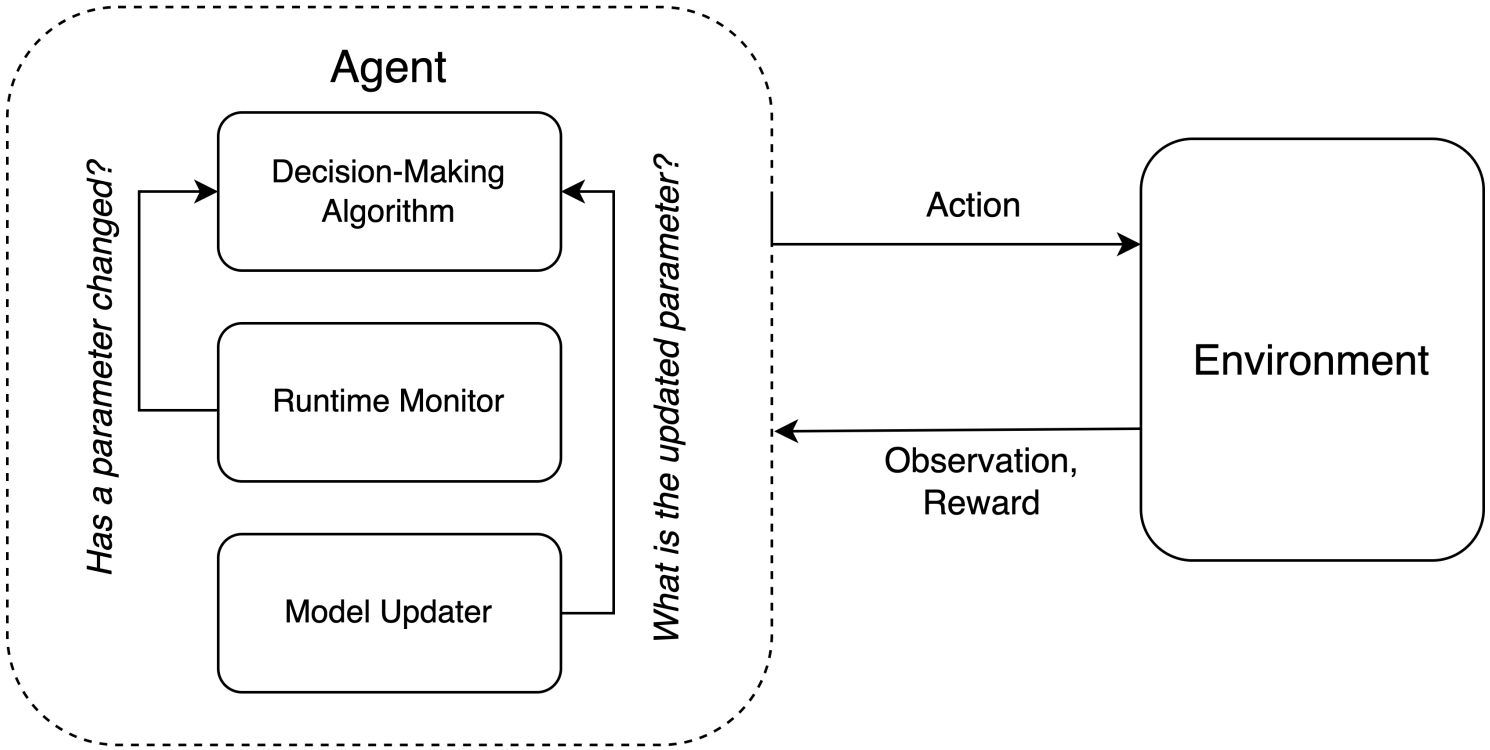
A General Framework for Non-Stationary MDPs



Standing on the shoulders of seminal work

- The Campo et al. paper from 1991 presents a very elegant conceptualization—the Markovian model for the endogenous and the exogenous uncertainties can (and should) be disentangled.
- We adopt their conceptualization.
- A “base MDP” can be used for the underlying control process.
- A semi-Markov process can be used for environmental parameters that change.

Orthogonal Question: What does an “agent” need to operate in non-stationary environments?



So why NS-Gym?

NS-Gym is a simulation toolkit for NS-MDPs that provides a tailored, standardized, and principled set of interfaces for non-stationary environments.

1. To our knowledge, there are no standard problems or benchmarks for NS-MDPs.
2. A toolkit is needed that captures key problem instances of decision making in NS-MDPs.
3. We need a simulation framework that offers broad NS-MDP specifications while maintaining an easy-to-use, familiar interface for researchers.

What can you do with NS-Gym?

- Built on top of the Gymnasium, NS-Gym lets users create custom NS-MDP to meet practical use case assumptions.
- Specify the frequency and how the environment parameters may change.
- Manage the interaction between agent and environment on two levels:
  - Can a decision making **detect** change in the environment?
  - Whether the agent **knows** the change in the environment?

Key Problem Types

Consider the Frozen Lake problem where at sometime point the environment becomes mores slippery, $\theta_1 \rightarrow \theta_2$.

1. Agent knows the environment has changed (θ_1 no longer applies) but does not know to what extent (does not know θ_2).
2. Agent knows the exact change (knows θ_2), but non-stationary environments make may make retraining an agent inefficient.
3. Agent is unaware that the environment has changed (does not know θ_3 no longer applies).

Frequency of change in parameters

An orthogonal problem types models frequency of change in θ in the following manner:

1. Single change within an episode: $\theta_1 \rightarrow \theta_2$

2. Multiple changes within and episode: $\theta_1 \rightarrow \theta_2 \rightarrow \ldots \rightarrow \theta_n \rightarrow \ldots$

3. Changes within multiple episodes.

FrozenLake

Summary

The FrozenLake environment is a stochastic, discrete action, and discrete state space grid-world environment. The agent navigates from a starting cell in the top left corner of the map to a goal cell in the bottom right corner while avoiding holes in the “frozen lake.” The agent can move in an intended direction, with some probability that it will move in a perpendicular direction instead. The agent will get a reward of +1 if it reaches the goal and 0 otherwise.

What changes?

NS-Gym induces non-stationarity by modifying the probability distribution over actions.

Try it on Google Collab

https://tinyurl.com/TheNSGymTutorial

Google Collab

Download from our Github page

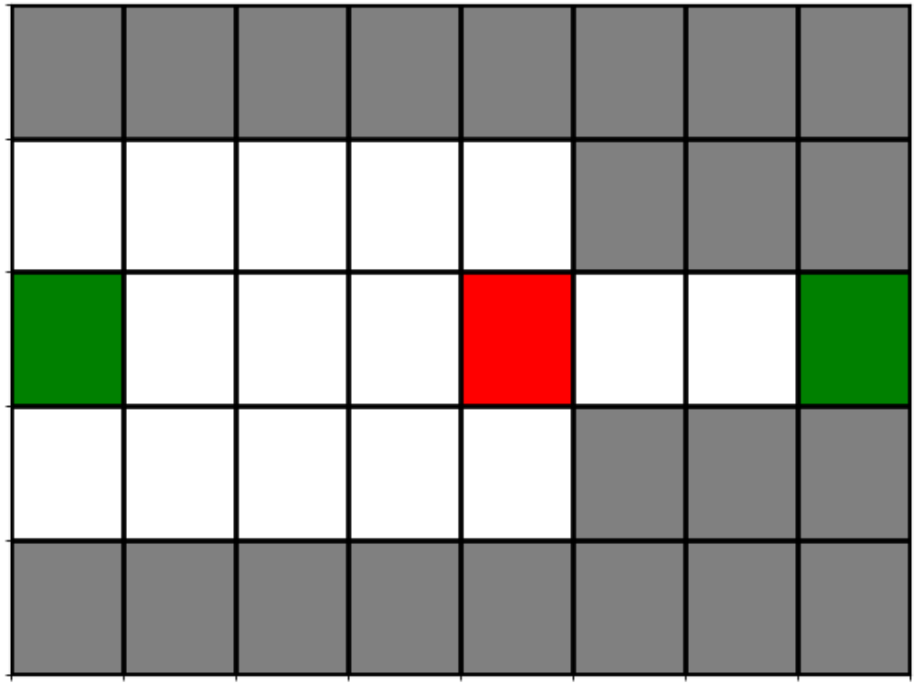
https://github.com/scope-lab-vu/ns_gym

Download from Github

Algorithm Implementations Included in NS-Gym

- **Monte Carlo Tree Search** (Kocsis and Szepesvari, 2006)
- **AlphaZero** (Silver et al., 2017)
- **Adaptive Monte Carlo Tree Search** (ADA-MCTS) (Luo et al., 2024)
- **Policy-Augmented Monte Carlo Tree Search** (PAMCTS) (Pettet et al. 2024)
- We also provide wrappers for interfacing with the **StableBaselines3** RL library (Raffin et al. 2021)

Non-stationary Bridge



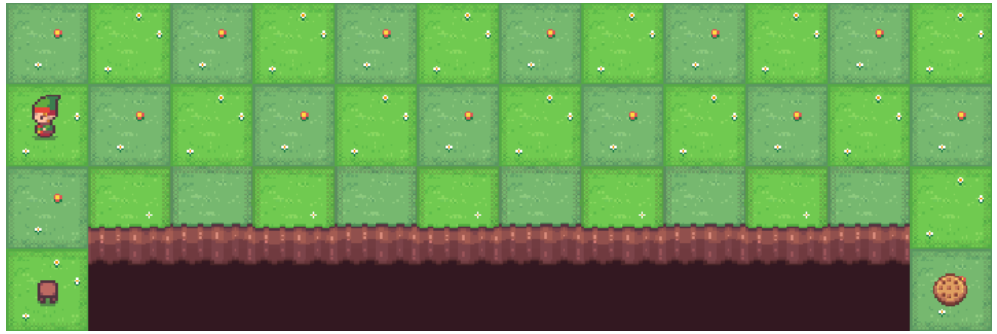
Summary

The non-stationary bridge environment is a grid-world setting where the agent must navigate from the starting cell to one of two goal cells. The environment was originally introduced by Lecarpentier and Rachelson (2019).

What changes?

Bridge environment has two probability distributions for the left and right halves of the grid world. NS-Gym at each decision epoch can make either or both halves of the map more or less slippery.

CliffWalking



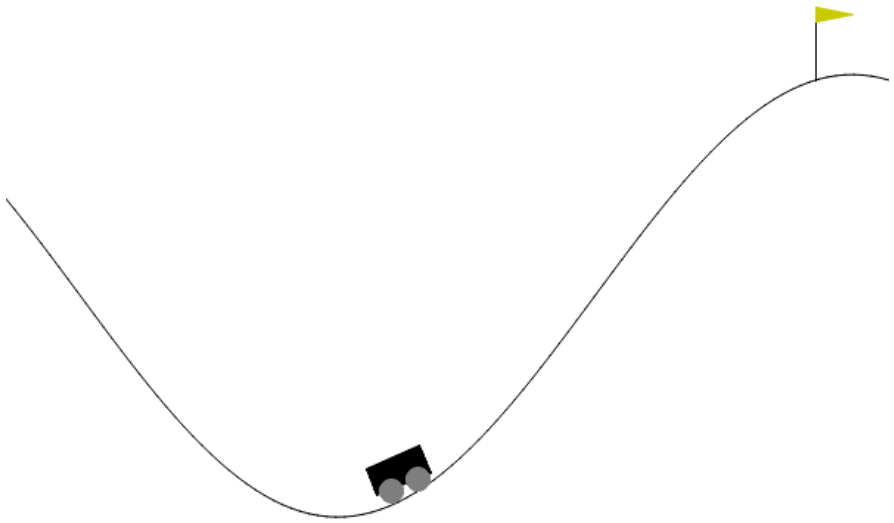
Summary

The agent must navigate from the start to the goal cell in the fewest steps. If the agent falls off a “cliff,” it accrues a reward of -100 and resets at the start cell without ending the episode. The agent accrues -1 reward for each cell that is not a cliff or a goal state. The goal cell is the only terminal state. The agent can move up, down, left, and right

What changes?

NS-Gym induces non-stationary by introducing stochastic transitions that vary with time.

MountainCar



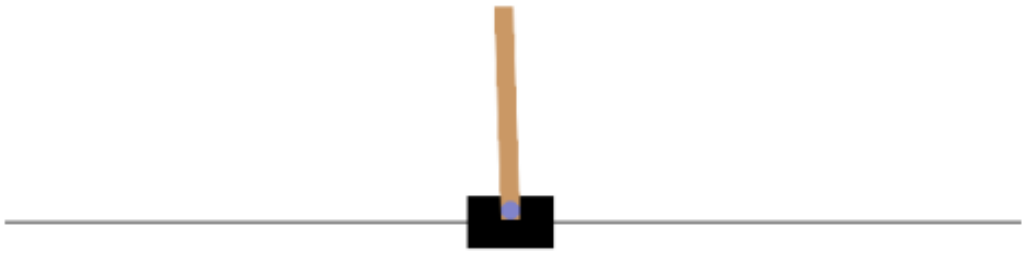
Summary

In this environment, a car is stuck in a valley, and the agent must apply force to the cart to build momentum so that the car can escape. By default, the agent receives a zero reward for escaping the valley and a -1 reward otherwise. The agent can either push the car to the left, right, or not at all. The continuous Mountain Car environment is similar to the standard Mountain Car environment but with a continuous action space. In the continuous analog, the agent chooses the direction in which to apply the force to the car.

What changes?

NS-Gym induces non-stationary effects by changing the gravity and force applied to the car.

CartPole



Summary

The CartPole environment has a discrete action space and a continuous state space. The agent's objective is to keep the pole balanced on top of the cart for as long as possible. The agent receives a reward of +1 for each time step that the pole remains balanced. The state is represented by a four-dimensional vector, which includes the cart's position, cart's velocity, pole's angle, and pole's angular velocity. At each time step, the agent can apply a fixed force to push the cart either left or right.

What changes?

Changes in gravity, the mass of the cart, the mass of the pole, the length of the pole, or the magnitude of the force applied to the cart can be made to create a non-stationary MDP.

Pendulum

Summary

The Pendulum environment is a continuous state and action space environment. The agent aims to keep the pendulum inverted for as long as possible. The agent receives a reward proportional to the pendulum's angle. At each time step, the agent applies some torque magnitude to the pendulum's free end. Figure 6 shows the pendulum environment.

What changes?

NS-Gym induces non-stationarity by increasing the link mass or length.

Acrobot

Summary

The Acrobot environment is a double pendulum. The agent can apply torque to the joint connecting the two links of the double pendulum to move the free end above a threshold height. At each time step, the agent can either apply +1, 0, or -1 units of torque.

What changes?

NS-Gym induces non-stationarity by altering the link lengths, link masses, center of mass position, and link moment of inertia.

Try it on Google Collab

https://tinyurl.com/TheNSGymTutorial

Google Collab

Download from our Github page

https://github.com/scope-lab-vu/ns_gym

Download from Github