Reinforcement Learning

All-Cloud Bootcamp – Day 4

Sandra Pico Oristrell
Professional Services – Associate Data Science
oristrel@amazon.com

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Goals

- Understand Reinforcement Learning Basic Concepts
- How to formulate a business problem into a RL Framework.
- Understand the different types of RL agents.
- Familiarize yourself with SageMaker RL and RL Toolkits.
- Run some example Notebooks to get the feeling of training problems with SageMaker RL

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Schedule

Total time: 4 hours (2:30pm-6:30pm)

Part 1: 2:30pm-3:30pm 

Reinforcement Learning Theory

3:30pm – 3:40pm Break

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Schedule

Total time: 4 hours (2:30pm-6:30pm)

Part 2: 3:45pm-4:45pm 

SageMaker RL, RL Toolkits and RL Environments

- **CartPole Toy-problem**
  - DQN
  - Q-Learning

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Schedule

Total time: 4 hours (2:30pm-6:30pm)

Part 3: 4:45pm-6:00pm

4:45pm – 5:00pm Break

HVAC Notebook:
- PPO

© 2019, Amazon Web Services, Inc. or its Affiliates.

Schedule

Total time: 4 hours (2:30pm-6:30pm)

Part 4: 6:00-6:30pm 

DeepRacer

Summary, Materials and Resources

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Part 1

Reinforcement Learning Theory

2:30pm – 3:30pm

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

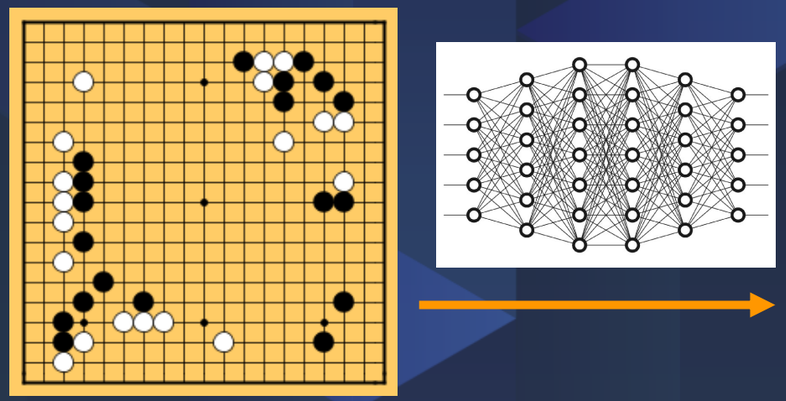
Why RL, Definition and Use cases

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Why Reinforcement Learning?

Do it by supervised learning



19×19
positions

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Why Reinforcement Learning?

- Supervised Learning
  - Supervisor
  - Golden truth
  - Mimic

- Reinforcement Learning
  - Maximize a Goal
  - By itself
  - Goal-driven

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning definition

- Learn by interacting with the environment, i.e. make decisions based on interaction.
- Differs from other type of learnings:
  - It is active. It is actively gathering experience.
  - Interactions are often sequential. Future interactions can depend on earlier ones.
- Goal-directed
- RL allows to learn without examples of optimal behavior.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

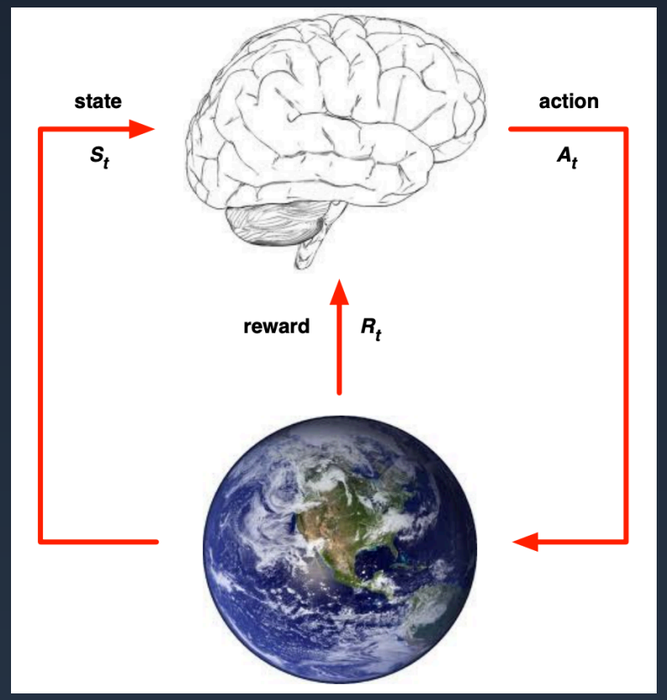
How does differ from other ML learnings?

- It does not require supervision, only a reward signal.
- Feedback can be delayed, it is not instantaneous.
- Time matters.
- Huge potential scope.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

RL elements



1. Agent

   Policy $\pi(s) = a$

   State-Value: $v_\pi(s) = \mathbb{E}_\pi(\sum_{k=0}^{\infty} R_{t+k} \,|S_t = s)$

   Action-Value: $q_\pi(s, a) = \mathbb{E}_\pi(\sum_{k=0}^{\infty} R_{t+k} \,|S_t = s, A_t = a)$

2. Goal: Maximizing the total rewards $\sum_t R_t$

3. Environment: everything else.

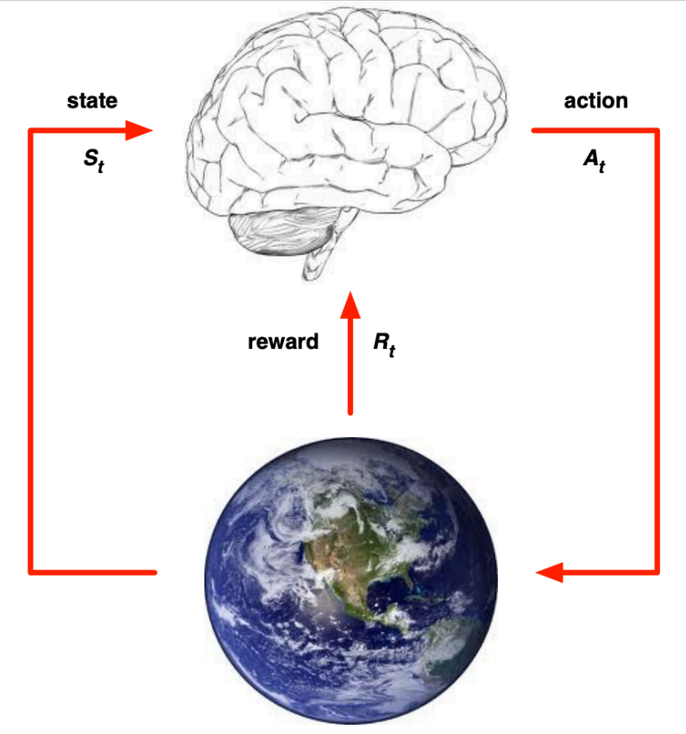
4. Actions: The actions that agent can take

5. State: State of environment.
   All the information for making decision

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

RL optimal policy



$$\pi^*(s) = a_{best} = \underset{a \in A}{\operatorname{argmax}}\, q_{\pi^*}(s, a)$$

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

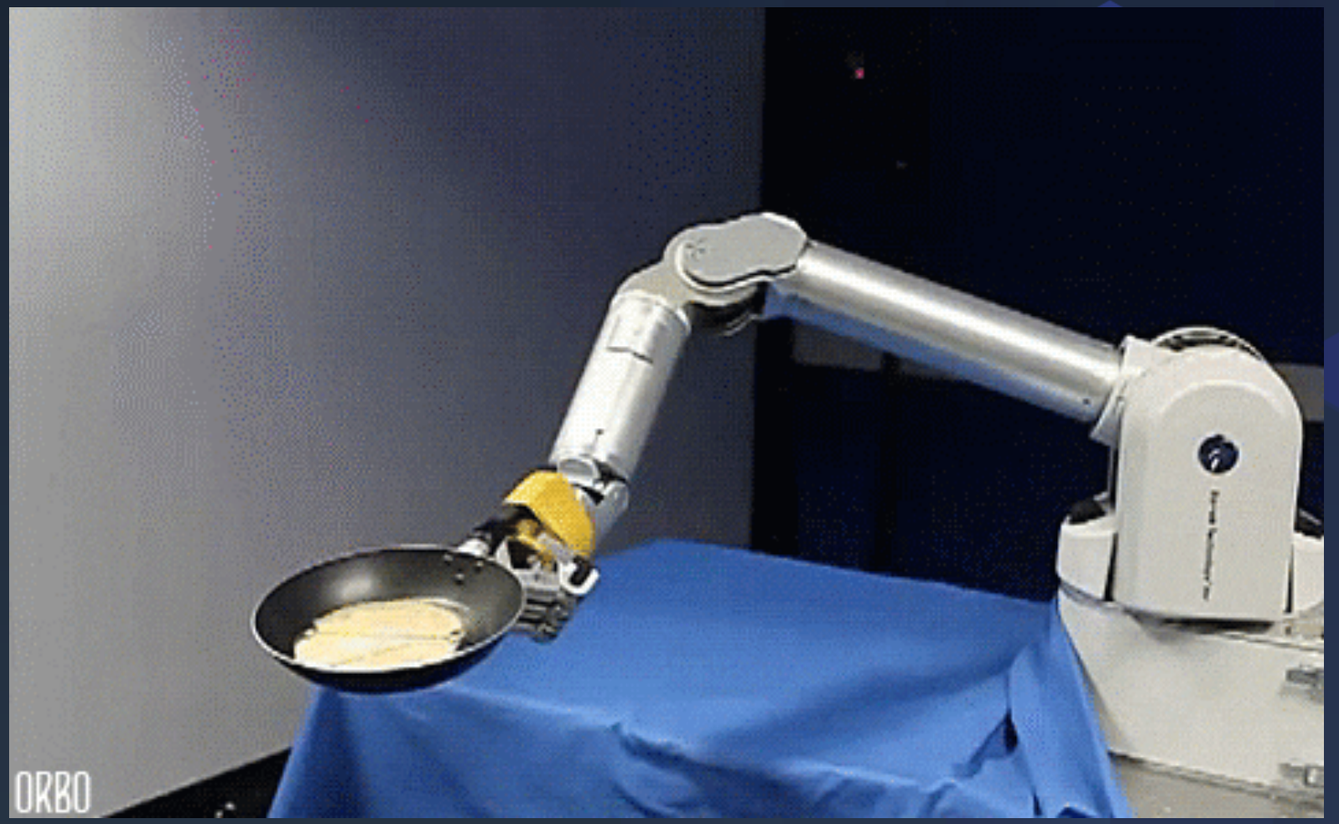
RL applicable in many domains

Robotics	Industrial Control	HVAC	Autonomous vehicles	Advertising
Dialog Systems	Operations	Finance	Resource Allocation	Online content delivery

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

RL use case: Dynamic Control

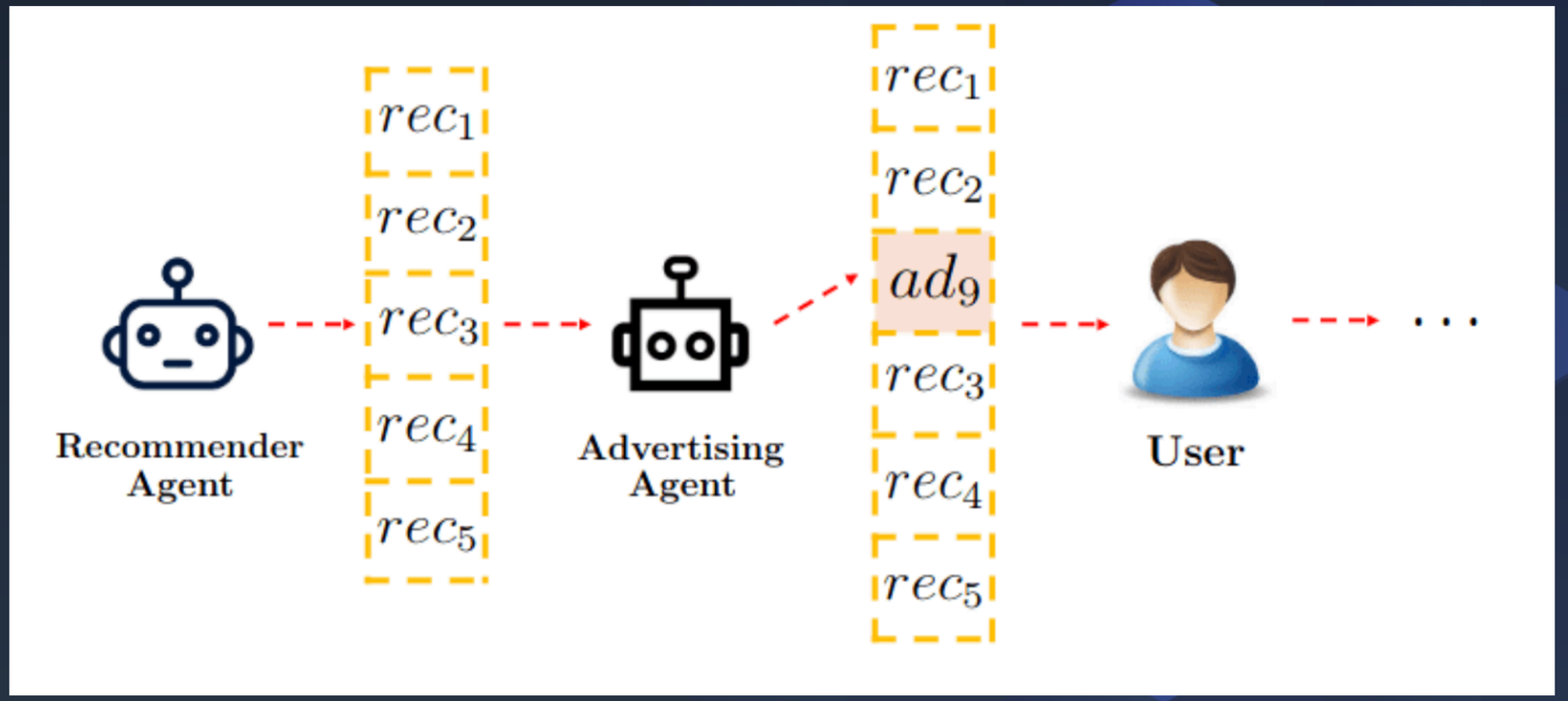


Kormushev, P., Calinon, S. and Caldwell, D.G., 2013. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, *2*(3), pp.122-148.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

RL use case: Recommendation/Advertisement/Chatbot

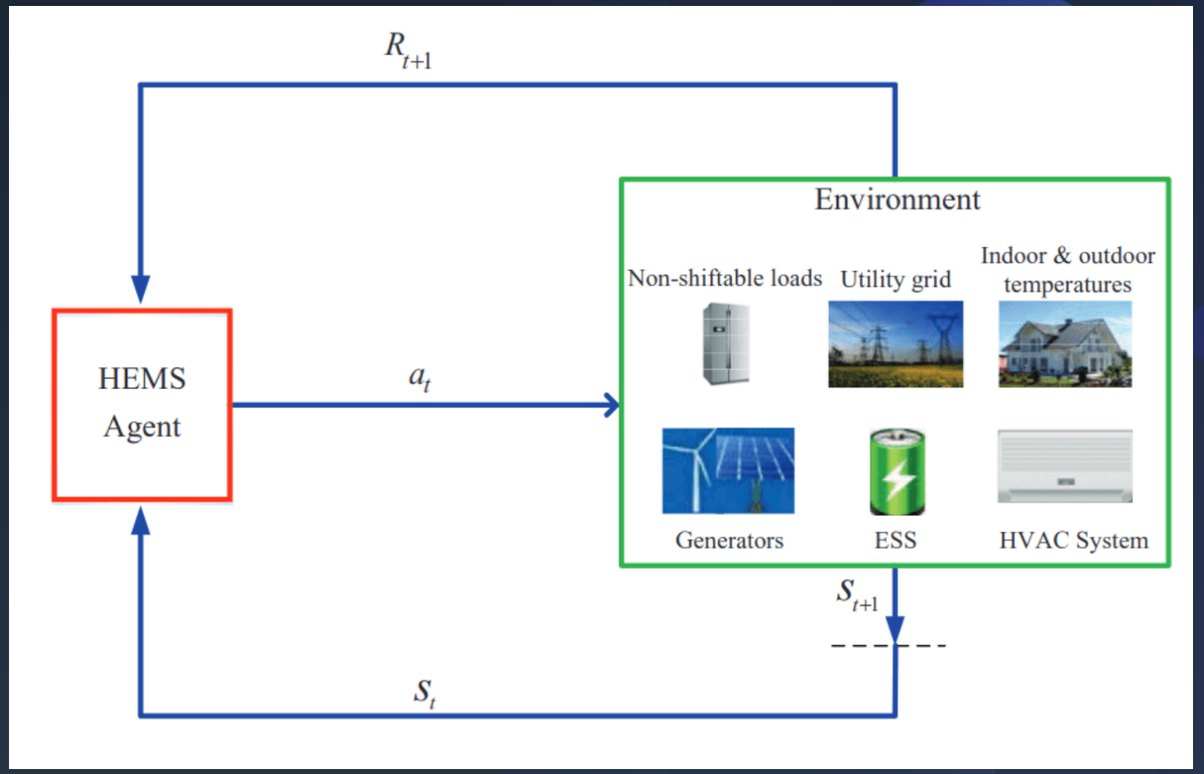


Theocharous, G., Thomas, P.S. and Ghavamzadeh, M., 2015, June. Personalized ad recommendation systems for life-time value optimization with guarantees. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

RL use case: Energy Management



Yu, L., Xie, W., Xie, D., Zou, Y., Zhang, D., Sun, Z., Zhang, L., Zhang, Y. and Jiang, T., 2019. Deep Reinforcement Learning for Smart Home Energy Management. *IEEE Internet of Things Journal*.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning: Bridging the gap from games to industry

- It is important to formulate the problem properly.
  - What are you trying to teach?

- Simulation environments really matters. You do not want to train in a real-world, since some of the problems might be difficult/dangerous.

  - Main bottleneck for Reinforcement Learning is the simulation itself.
  - Fast simulations and fidelity.
  - Scalability and Performance

- Once the algorithm have achieved a good performance in the simulation environment. You can then Tune in the real-world environment.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning: Bridging the gap from games to industry

- Once the algorithm have achieved a good performance in the simulation environment. You can then Tune in the real-world environment.

  Make sure training is physically meaningful.

  Tolerances early.

  Is online learning suitable for tunning in the physical world?

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning: Bridging the gap from games to industry

- Simulators that model a lot of business problems
  - Discrete Event
  - Fly, Drive, Sail
  - Transportation and Logistics
  - Chemestry and Pharma
  - Medical simulators
  - Security and Networking
  - Robots and Drones

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning: Bridging the gap from games to industry

- Reward function design is key.
  - Is it really what you want to learn?
  - Shaping the reward function
    - Sparse vs gradually teaching.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning: Bridging the gap from games to industry

- Reward function design is key.
  - Structure the reward function.

Positive rewards encourage:

- Keep going to accumulate reward
- Avoid terminal states unless yield to a very high reward.

Negative rewards encourage:

- Reach a terminal state as quickly as possible to avoid Accumulative penalties.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning: Bridging the gap from games to industry

- Reward function design is key.
  - Is my reward function really complex?

    If ___, then Reward_x

  - Subject matter expertise

    Sometimes you can break the problem

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning: Bridging the gap from games to industry

- Apprenticeship Learning
  - Learning via imitation.
  - Use expert guided examples to constrain the state space to explore.

© 2019, Amazon Web Services, Inc. or its Affiliates.

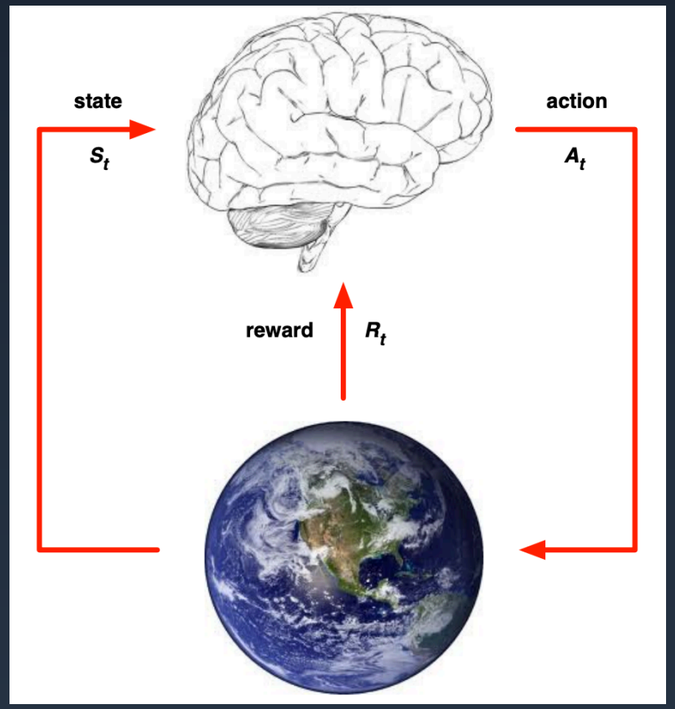
aws

From Business Problem to RL Framework

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Core Reinforcement Learning elements



1. Agent

   Policy $\pi(s) = a$

   State-Value: $v_\pi(s) = \mathbb{E}_\pi(\sum_{k=0}^{\infty} R_{t+k} | S_t = s)$
   Action-Value: $q_\pi(s, a) = \mathbb{E}_\pi(\sum_{k=0}^{\infty} R_{t+k} | S_t = s, A_t = a)$

2. Goal: Maximizing the long term reward $\sum_t R_t$

3. Environment: everything else.

4. Actions: The actions that agent can take

5. State: State of environment.
   All the information for making decision

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Core Reinforcement Learning elements

Reward and Return

Any goal can be formalized as the outcome of maximizing a cumulative reward.
(Reward hypothesis)

Reward R_t is a scalar feedback signal. Indicates how well the agent is doing at step t. This signal can be dense or sparse.

The agent's job is to maximize the cumulative reward, the Return.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots$$

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Core Reinforcement Learning elements

Value-state function

$V(s)$ = expected cumulative reward, from state s.

$$V(s) = \mathbb{E}[G_t | S_t = s] = \mathbb{E}[\, R_{t+1} + R_{t+2} + \ \ldots \,|\, S_t = s \,]$$

Goal: maximize the value, by picking the optimal actions (Policy).

Optimal state-value function $v^*(s)$ is the maximum value function over all policies.

$$v^*(s) = \max_\pi v^\pi(s)$$

Optimal action-value function $q^*(s,a)$ is the maximum action-value function over all policies.

$$q^*(s,a) = \max_\pi q^\pi(s,a)$$

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Core Reinforcement Learning elements

Actions

- The agent needs to select actions to maximize the value.
- Actions may have long term consequences.
- The agent may need to sacrifice immediate reward to gain more long-term reward.
- We need to define what actions the agent can do:
    - Discrete actions
    - Continuous action-space

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Core Reinforcement Learning elements

Agent State

- Both Agent and environment may have an internal state.
- The state of the environment might differ with the agent state.
    - Not usually visible to the agent.
    - Event if it is visible, it may contain lots of irrelevant information.

- History is a sequence of observations, actions and rewards…

$$H_t = O_0, A_0, R_1, O_1, \ldots, O_{t-1}, A_{t-1}, R_t, O_t$$

- This history can be used to construct an agent state S_t
- Simple case, the agent state can be represented by the current observation.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Markov Decision Processes

- Markov decision processes (MDPs) provide a useful mathematical framework for RL problems. Each time-step considers of:
  - Environment
  - State
  - Action
  - Reward
  - Observation
- A decision process is Markov if verifies Markov Assumption:

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

$$p(r, s \mid S_t, A_t) = p(r, s \mid H_t, A_t)$$

- Once the state is known, the history may be thrown away.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Partially Observability

- In a lot of problems, the agent gets partial information.
  - A robot with camera vision does not know the absolute location.
  - A poker playing agent only observes public cards.

Image from DeepMind lectures

- These two states are not Markov.
- How could you construct a Markov agent state in this maze (for any reward signal)?

- Now the observation is not Markov.
- This is formally call Partially Observable Markov Decision Processes (POMDP)

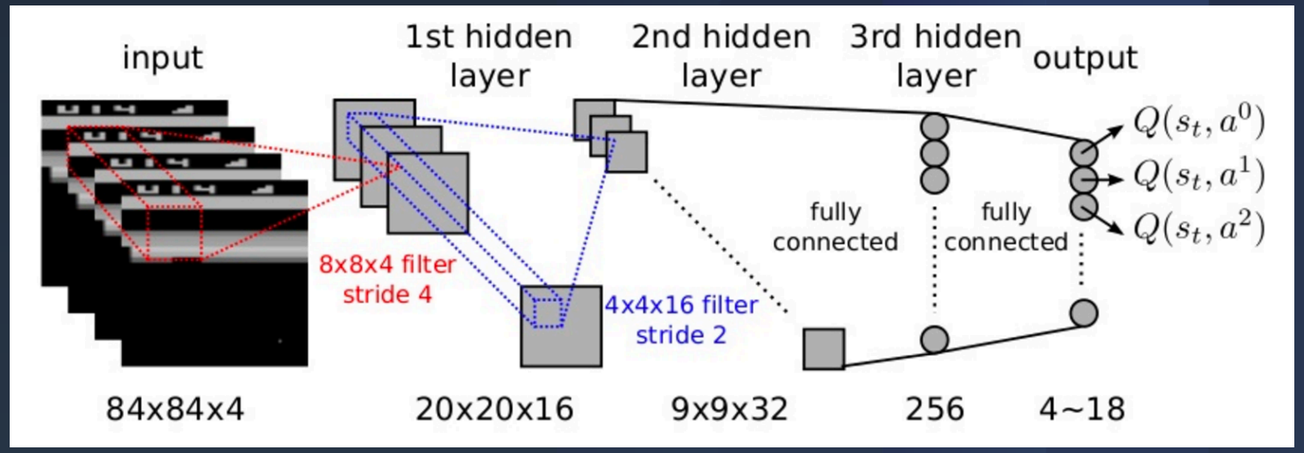
© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Partially Observability

- Agent can construct suitable state representations to deal with Partial Observability.

  - If it is possible, the agent stat should be Markov.
  - But sometimes, might not be feasible.
    - However, the state should contain enough information for good policies and/or good value predictions.



© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Map a Business problem into RL elements

Business Problem definition

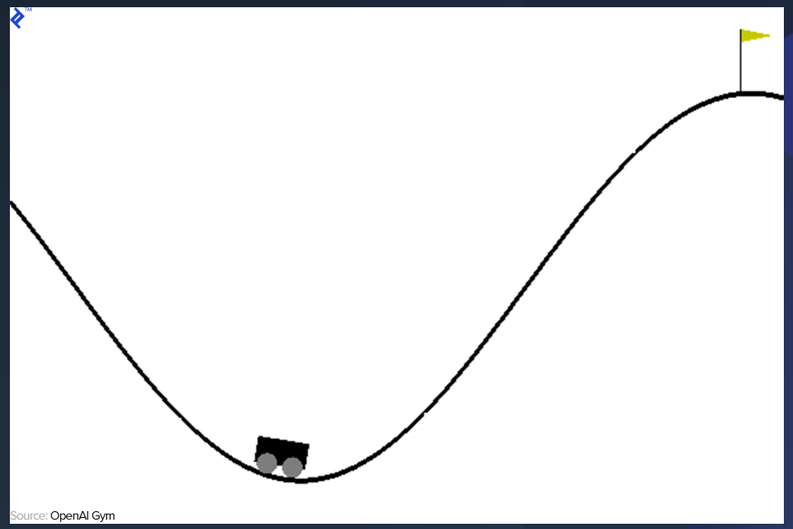
1. Objective
2. Environment
   - Observation
3. Agent State
4. Action Set
5. Reward Function

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Map a Business problem into RL elements

Example



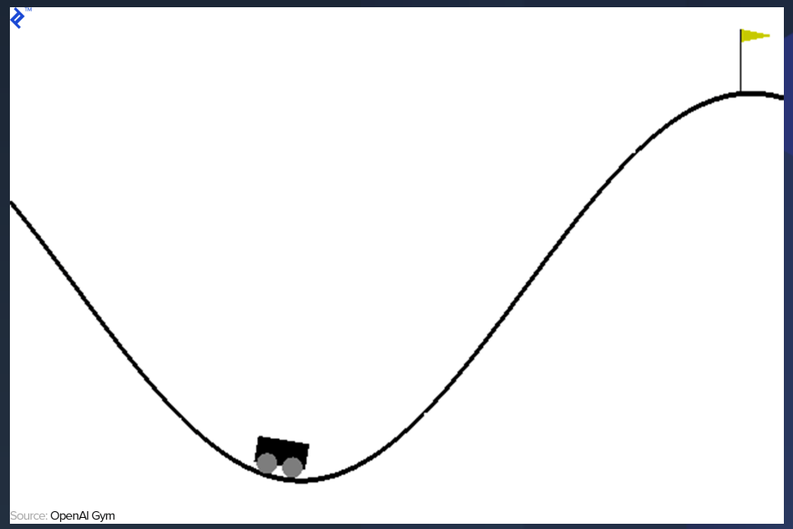
Mountain Car is a classic control Reinforcement Learning problem that was first introduced by A. Moore in 1991 [1]. An under-powered car is tasked with climbing a steep mountain, and is only successful when it reaches the top. Luckily there's another mountain on the opposite side which can be used to gain momentum, and launch the car to the peak. It can be tricky to find this optimal solution due to the sparsity of the reward.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Map a Business problem into RL elements

Example



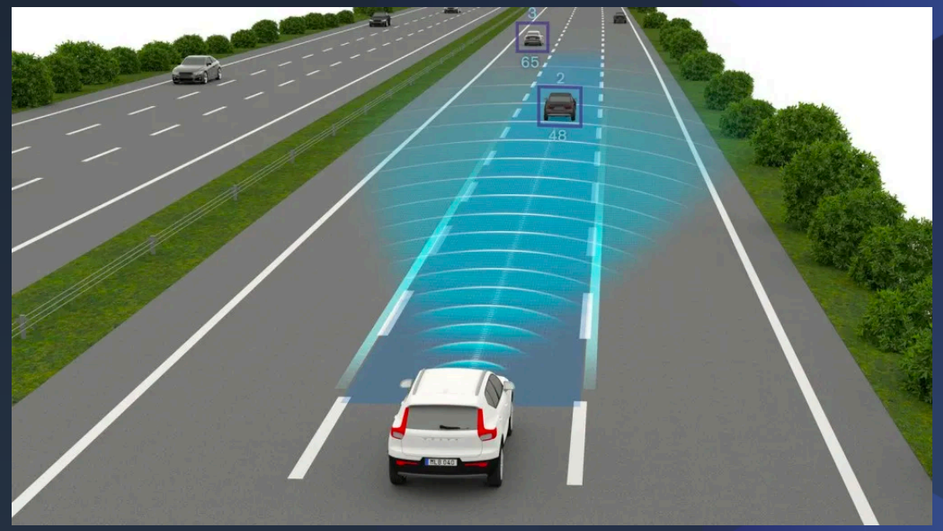
1. Objective: Get the car to the top of the right hand side of the mountain.
2. Environment: Open AI Gym's MontainCar-v0.
   - Observation: The car's horizontal position and velocity.
3. Agent State: Car's horizontal position and velocity (can be negative).
4. Action Set: Discrete actions - direction of push (left, nothing or right).
5. Reward Function: -1 for every step until success, incentivises quick solutions.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Map a Business problem into RL elements

Business Problem: Lane centering / Auto steer



A system is designed to keep a car centered in the lane, relieving the driver of the task of steering.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Map a Business problem into RL elements

Business Problem: Lane centering / Auto steer

Example:

1. Objective: Learn to drive autonomously by staying inside the lane.
2. Environment: A 3D driving simulator hosted on AWS.
3. Agent State: The driving POV image captured by the car's head camera
4. Action Set: Steering wheel angular position, car velocity
5. Reward Function:
    - Positive reward for staying inside the lane
    - High penalty for going off-track.
    - Add penalty for steering
    - Add penalty for high changes of angular position and velocity

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning Agents

Given the RL formulation. How do I solve the problem?

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning Agents

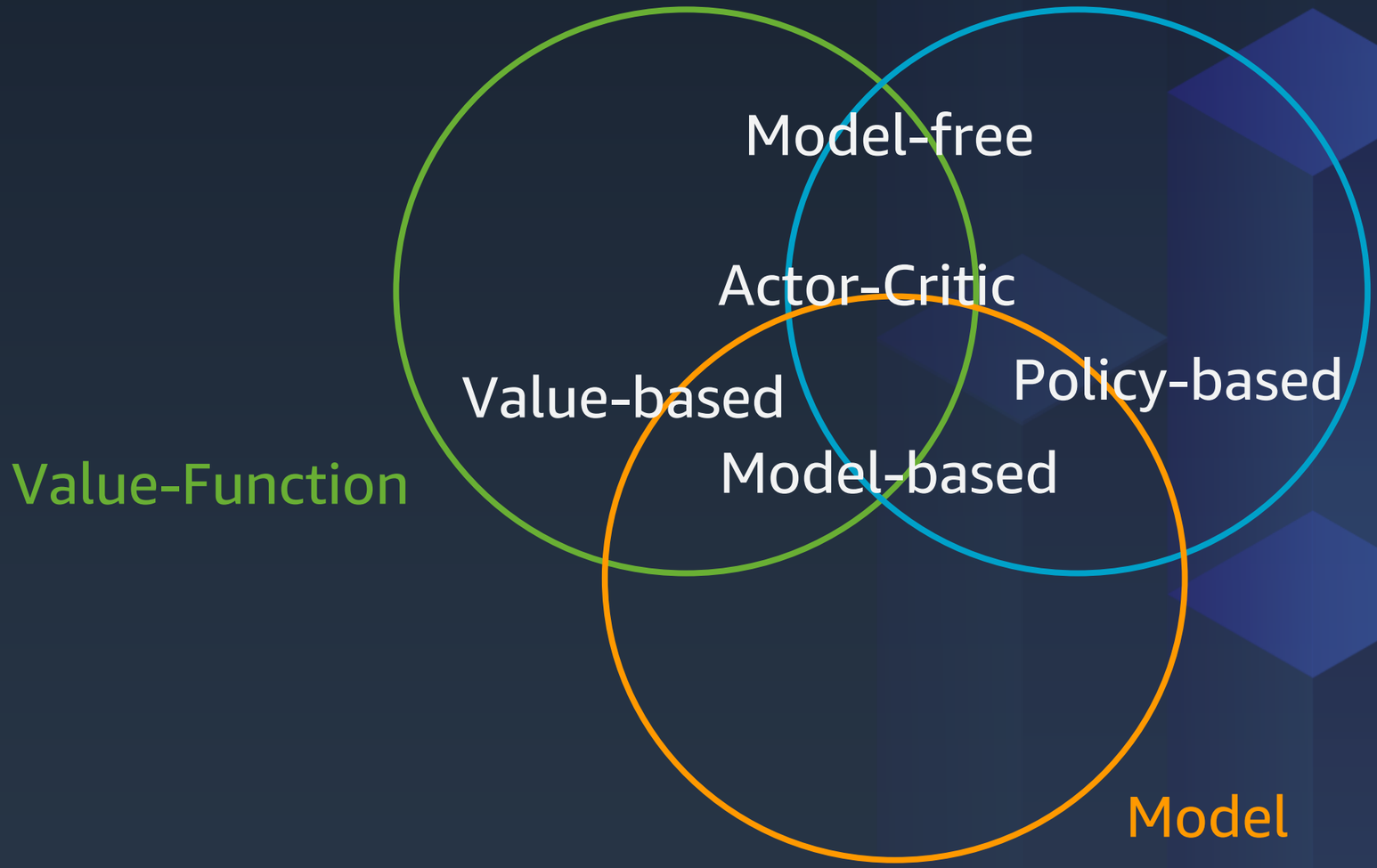
- **Value-based**
  - Value function
- **Policy-based**
  - Policy
- **Actor-Critic**
  - Policy
  - Value Function

- **Model Free**
  - Policy and/or Value Function
  - No Model
- **Model Based**
  - Optionally Policy and/or Value Function
  - Model

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning Agents

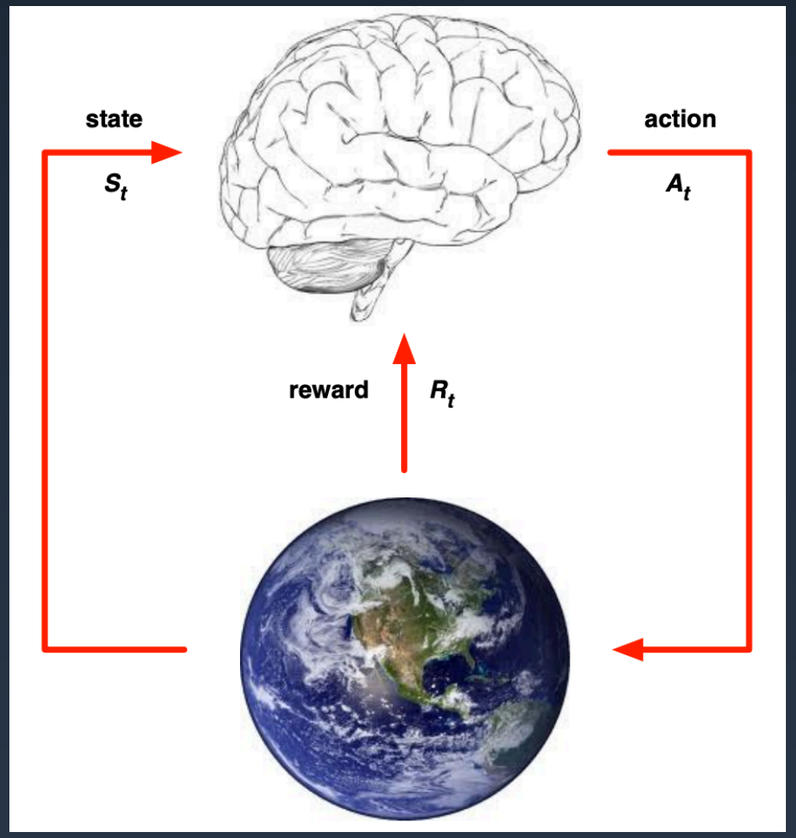


© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Generalized Policy Iteration(GPI)

Start with an arbitrary stochastic $\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \ldots \xrightarrow{I} \pi_* \xrightarrow{E} q_*$



π_k

Experience data

Policy Evaluation

$Q_{\pi_k}(s_i, a_{left}) > Q_{\pi_k}(s_i, a_{right})$

Policy Improvement

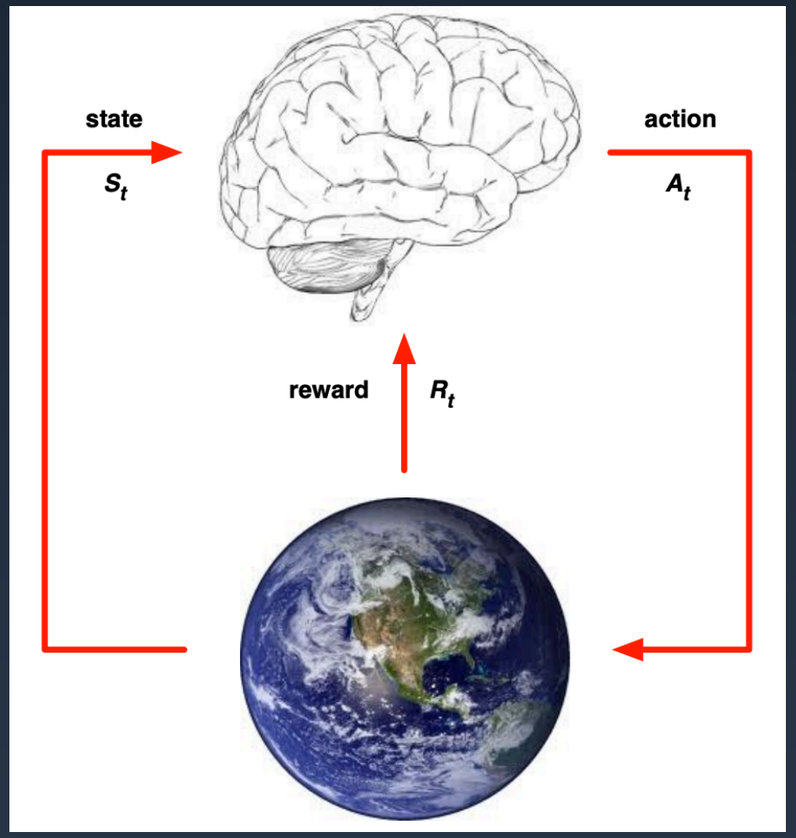
$\pi_{k+1} \leftarrow \varepsilon\text{−greedy } Q_{\pi_k}$

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Generalized Policy Iteration(GPI)

Start with an arbitrary stochastic $\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \ldots \xrightarrow{I} \pi_* \xrightarrow{E} q_*$



π_k

Experience data

The prediction problem

Policy Evaluation

$Q_{\pi_k}(s_i, a_{left}) > Q_{\pi_k}(s_i, a_{right})$

Policy Improvement

$\pi_{k+1} \leftarrow \varepsilon\text{-greedy } Q_{\pi_k}$

The control problem

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Prediction and Control

- Prediction Problem: Evaluate the future (Value-Function) for a given policy.
- Control Problem: Optimize the future, find the best policy.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Exploration and Exploitation

- Exploration fins more information
- Exploitation exploits known information to maximise reward

But, how can we reason about the exploration trade off?
Can we trade off exploration and exploitation optimally?

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Exploration and Exploitation

Algorithms

ε-Greedy algorithm

With probability $1 - \varepsilon$, $a = argmax_{a \in A} Q_t(a)$
With probability ε, select random action.

Upper Confidence Bounds

$$a_t = argmax_{a \in A}\, Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$$

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Learning the components of an agent

- All components are functions:
  - Policies map states to actions
  - Value functions map states to values
  - Models map states to states and/or rewards
  - State updates map states and observations to new states

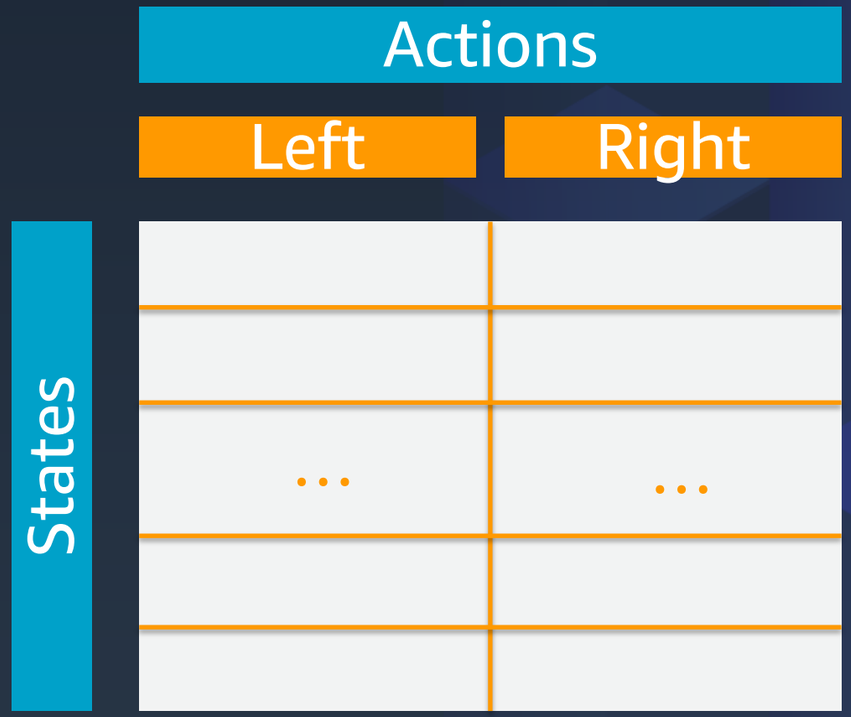
- We could represent these functions as:
  - Tabular methods
  - Approximation methods such as using Neural Networks

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Tabular Methods

- State and action spaces are small enough to approximate value functions to be represented as arrays/tables.



© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Tabular Methods

- State and action spaces are small enough to approximate value functions to be represented as arrays/tables.

  - Dynamic Programming

    - Model-based Finite MDP.
    - Operate in sweeps through the state set, performing an expected update on each state.
    - Each operation updates the value of one state based on the values of the successor states.
    - Asynchronous DP methods: Not sweep in all states, but in arbitrary order.

    Bootstrapping: Update estimates basis on other estimates.

    - Value iteration, Policy iteration algorithms

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Tabular Methods

- State and action spaces are small enough to approximate value functions to be represented as arrays/tables.
    - MonteCarlo methods
        - Model-free: no knowledge of MDP transitions.
        - Sample episodes
        - Simple idea: value = mean return
        - Episodic tasks
        - Efficient to focus in small subset of states
        - Do not bootstrap

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Tabular Methods

- State and action spaces are small enough to approximate value functions to be represented as arrays/tables.
    - Temporal Difference Learning
        - Model-free
        - Sample episodes
        - Works in continuing environments.
        - Can learn online every step
        - Can learn from incomplete sequences
        - Bootstrap

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning with Function Approximation

- Most of the real-world problems have to deal with large MDPs
  - Too many states and/or actions to store in memory.
  - Too slow to learn the value of each state individually.
  - Need to Generalise from seen states to unseen states.

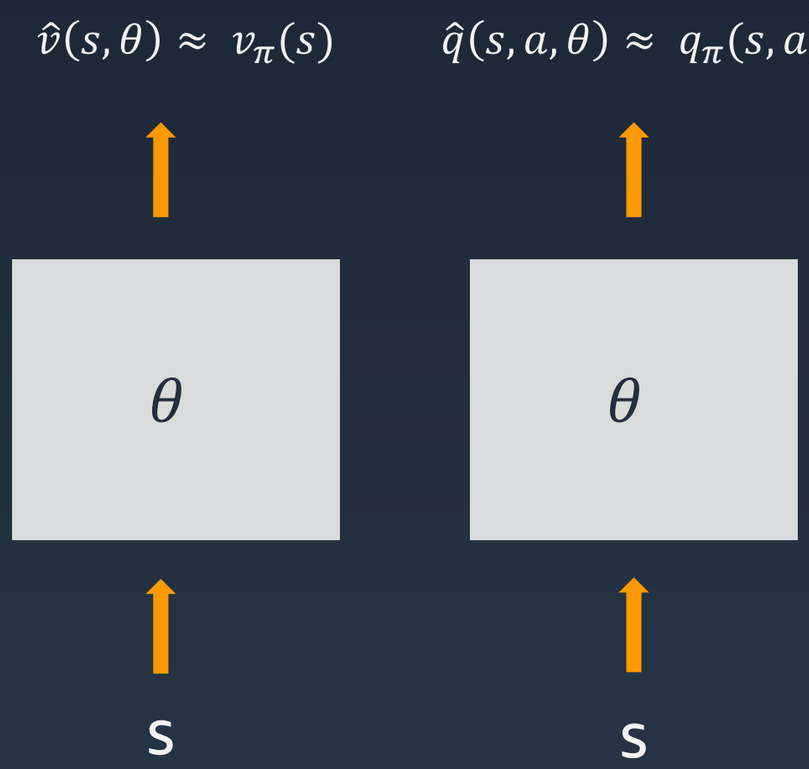
- Value-based RL Agents
  - Value function Approximation
- Policy-Based RL Agents
  - Policy function Approximation
- Actor-Critic RL Agents
  - Policy function Approximation and Value-Function Approximation

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning with Function Approximation

Value Function Approximation



- Multiple type of Function Approximators.
  - When we use Deep Neural Networks, we talk about Deep Reinforcement Learning.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning with Function Approximation

Value Function Approximators

- Represent state by a feature vector
    - Example, if state is represented through pixels: CNN.

$$x(S) = \begin{pmatrix} x_1(S) \\ . \\ . \\ . \\ x_n(S) \end{pmatrix}$$

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Reinforcement Learning with Function Approximation

Function Approximation

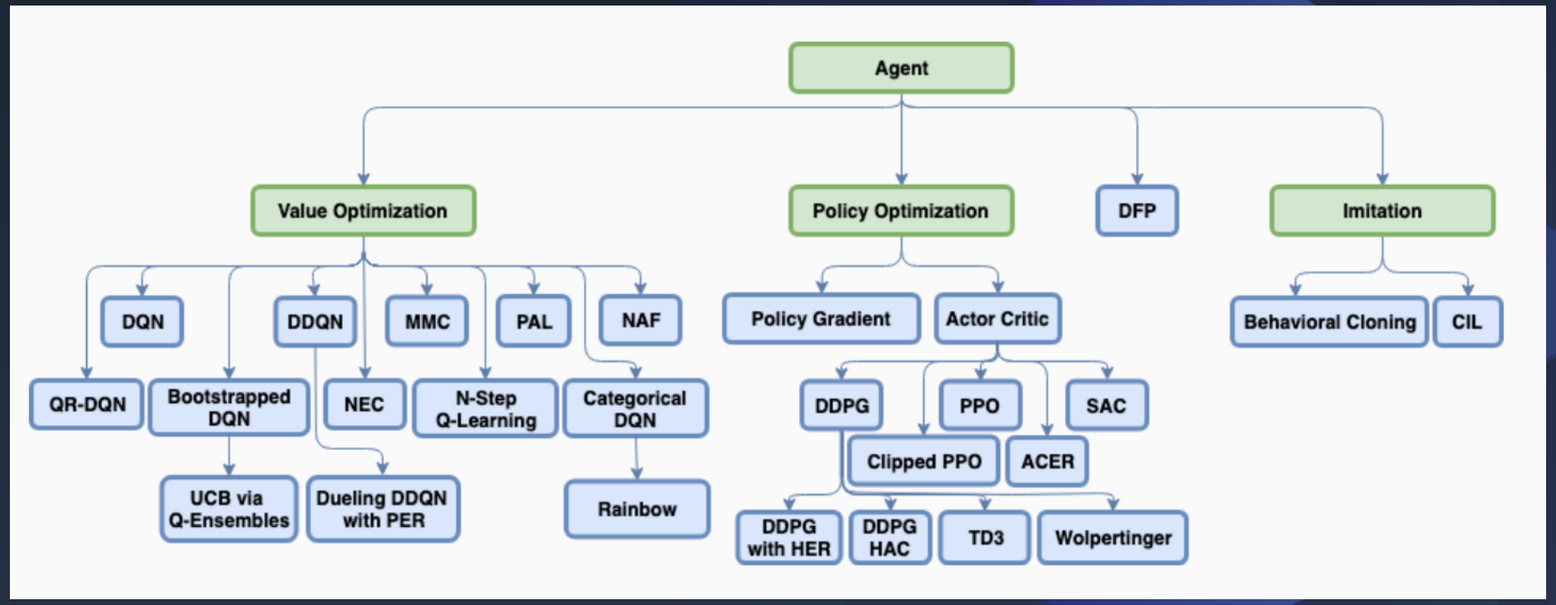
- Incremental Learning
  - Update the weights every time step.
- Batch Learning
  - Using Agent’s experience as Training data.
    - Sample state value from experience.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Selecting an Algorithm

- Lot of different Reinforcement Learning algorithms.



How do I select the right algorithm?

https://nervanasystems.github.io/coach/selecting_an_algorithm.html

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Break (10 minutes)

© 2019, Amazon Web Services, Inc. or its Affiliates.

Recap Part 1

- Why Reinforcement Learning
- Use cases and Industries
- Challenges to move to real-world problems
- Main elements in Reinforcement Learning
- Formulate a business problem into the RL framework
- Exploration and Exploitation
- Reinforcement Learning and Deep Reinforcement Learning Agents

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Part 2

3:45pm – 4:45pm

How and where do I train?

- SageMaker RL, RL Toolkits and RL Environments

CartPole Problem

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

SageMaker RL, RL Toolkits and RL Environments

© 2019, Amazon Web Services, Inc. or its Affiliates.

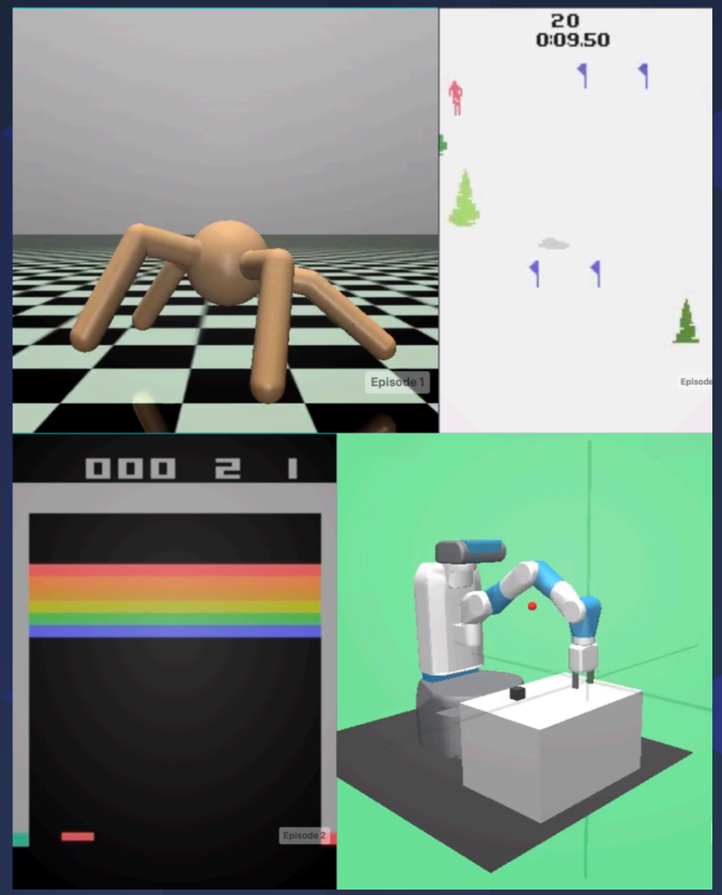
aws

RL Toolkits & Frameworks

- Intel Coach
- Facebook's ReAgent: RL Toolkit
- Keras-RL
- Pytorch
- RLCard
- Tensorflow: TF-Agents
- …

RL Environments

- OpenAI Gym
- Gym Trading
- ViZDoom
- OpenSpiel
- Ns3-gym
- RecoGym
- OpenSim-RL-Biomechanics
- …



Some environments from OpenAI Gym. Images taken from the official website.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

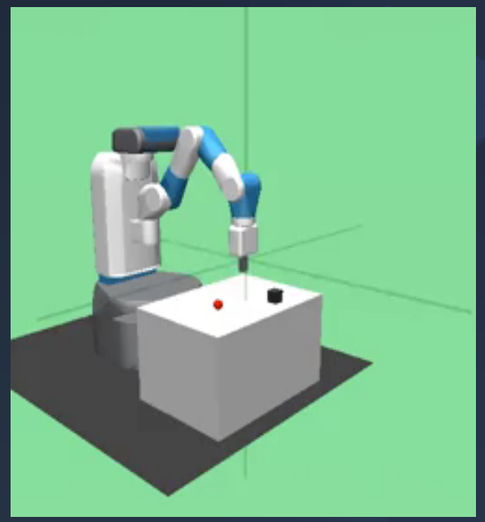
RL Environments

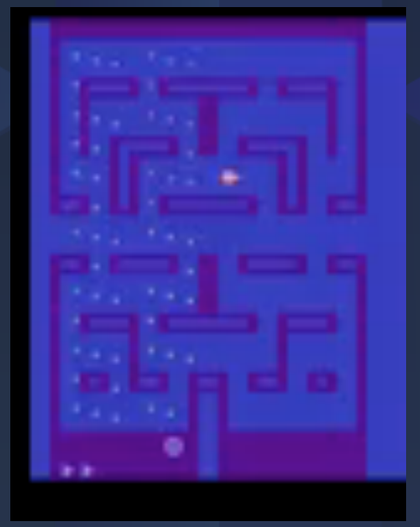
OpenAI-Gym

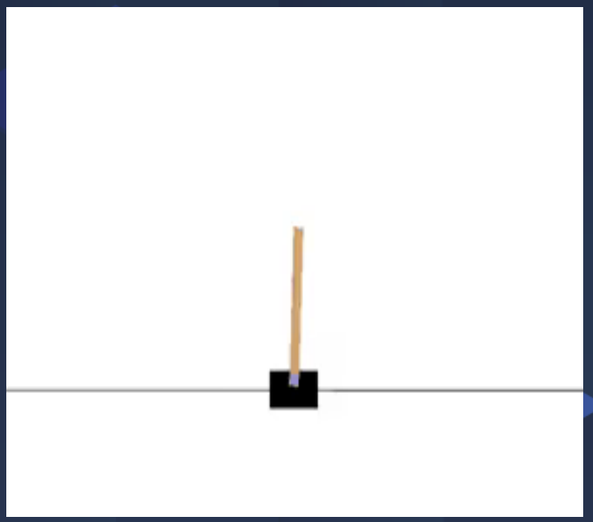
- *Collection of test problems – environments – that you can use to work out your RL algorithms. These environments have a shared interface, allowing you to write general algorithms.*

- Interface

```
env.action_space
env.observation_space
env.reset()
step()
env.render()
```







http://gym.openai.com/envs/#atari

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

RL Toolkits

Intel Coach, Ray/RLlib, VW and others

- *Toolkits are libraries that provide specific algorithms to train a RL Model.*

RLlib

- RLlib is easier to use. You can directly execute multiple clusters and train in parallel.
- It is more modern, so it is integrated with latest versions of TensorFlow and PyTorch.
- Newest version of RLlib (0.8.5) is really easy-to-use.

Coach

- Coach is older, so it has more state-of-the-art RL algorithms implemented.
- There is more support (examples and documentation).
- SageMaker RL is container-based, so you can run different toolkits there.
- There is more examples in Coach.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

SageMaker RL

Key features of Amazon SageMaker RL

- To train RL models in Amazon SageMaker RL, use the following components:
  - A DL framework. SageMaker RL supports RL in TensorFlow ,MXNet, PyTorch.
  - RL Toolkit. Manages the interaction betweeen the agent and the environment, and provides a wide selection of state of the art RL algorithms. SageMaker RL supports Intel Coach and Ray RLlib.
  - RL Environment. You can use custom environments, open-source environments, or commercial environments.
- Supports Distributed Training and Hyperparameter Tunning jobs

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

SageMaker RL

RL Environments

- OpenAI-Gym is installed in the SageMaker RL containers. To use OpenAI Gym environments, you have to use the OpenAI Gym Interface.
- Open Source Environments: you can use open source environments (EnergyPlus and RoboSchool), by building your own container.
- You can use commercial environments, such as MATLAB, by building your own container. You will then need to manage your won licenses.
- You can also connect SageMaker RL to AWS Simulation environments such as RoboMaker.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

SageMaker RL

General Diagram for SageMaker

Amazon Sagemaker

Managed notebook instances

```
estimator = RLEstimator(entry_point="train-coach.py",
                        source_dir='src',
                        dependencies=["common/sagemaker_rl"],
                        toolkit=RLToolkit.COACH,
                        toolkit_version='0.11.0',
                        framework=RLFramework.MXNET,
                        role=role,
                        train_instance_type=instance_type,
                        train_instance_count=1,
                        output_path=s3_output_path,
                        base_job_name=job_name_prefix,
                        hyperparameters = {
                          "RLCOACH_PRESET": "preset-cartpole-clippedppo",
                          "rl.agent_params.algorithm.discount": 0.9,
                          "rl.evaluation_steps:EnvironmentEpisodes": 8,
                          "improve_steps": 10000,
                          "save_model": 1
                        }
                    )
```

------- OR -------

Laptop with SageMaker SDK

./train-coach.py
./src

Deep Learning Container

Fully-managed
SageMaker Cluster

S3

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

SageMaker RL

Amazon SageMaker RL Containers

- Set of Dockerfiles that enables RL solutions to be used in SageMaker.
- RL Toolkits: Ray and Coach.
- DL Frameworks:
    - Tensorflow (used for Ray and Coach)
    - MXNet (used for Coach)

https://github.com/aws/sagemaker-rl-container/#rl-images-provided-by-sagemaker

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

SageMaker RL

Python SDK for Reinforcement Learning

RL Training

- With RL Estimators, you can train reinforcement learning models on SageMaker.

```
from sagemaker.rl import RLEstimator, RLToolkit, RLFramework

rl_estimator = RLEstimator(entry_point='coach-train.py',
                           toolkit=RLToolkit.COACH,
                           toolkit_version='0.11.1',
                           framework=RLFramework.TENSORFLOW,
                           role='SageMakerRole',
                           train_instance_type='ml.p3.2xlarge',
                           train_instance_count=1)
```

```
rl_estimator.fit()
```

- We will cover how to prepare the training script in the practical examples.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

SageMaker RL

Python SDK for Reinforcement Learning

Deploying RL Models

- After calling fit, you can call deploy on an RLEstimator to create a SageMaker endpoint.

- .deploy returns:
  - a sagemaker.mxnet.MXNetPredictor for MXNet
  - or sagemaker.tensorflow.serving.Predictor for TensorFlow

- .predict returns the result of inference against your model

```
predictor = rl_estimator.deploy(instance_type='ml.m4.xlarge',
                                initial_instance_count=1)

response = predictor.predict(data)
```

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

SageMaker RL

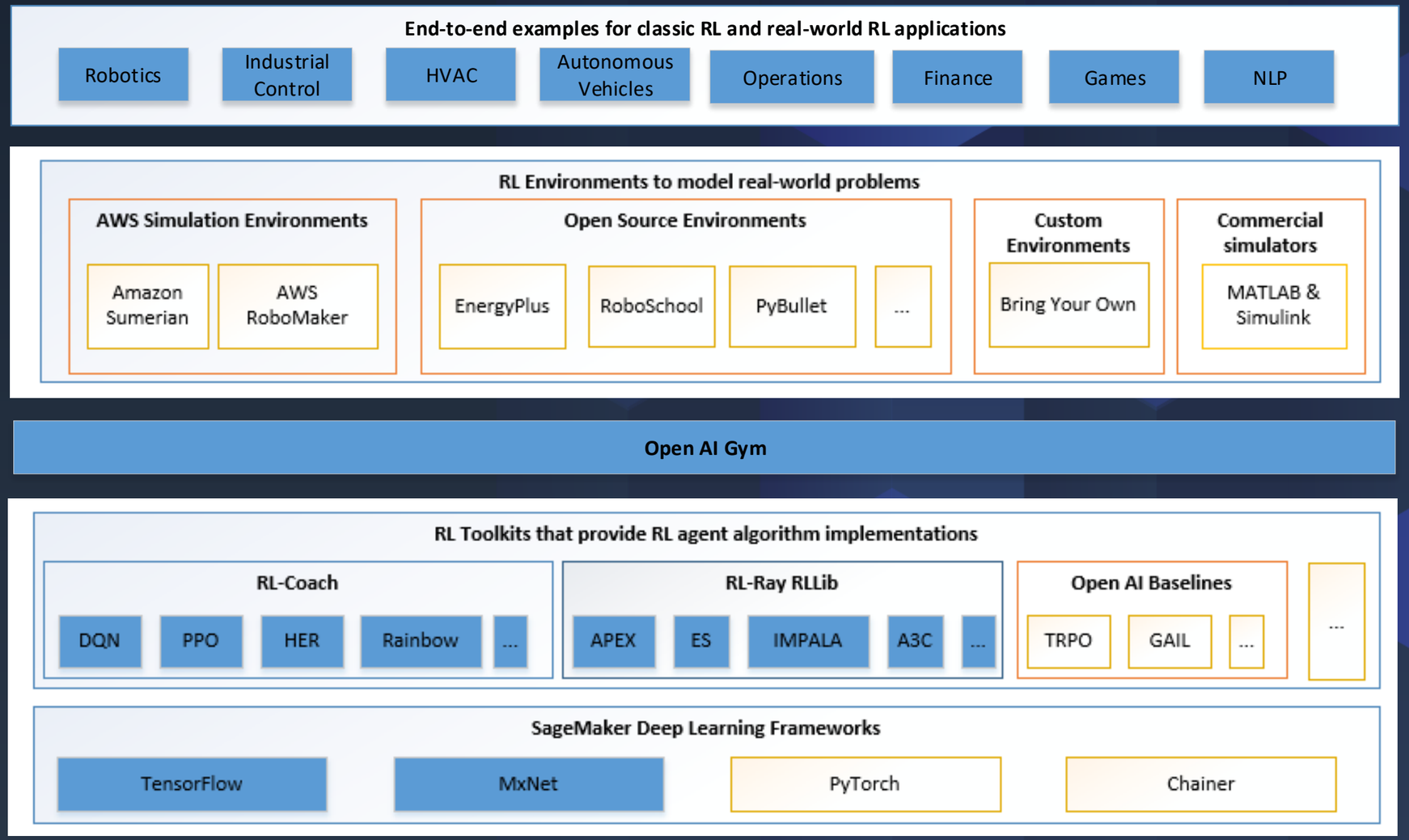
Main Diagram

Sample Notebooks

RL environments

OpenAI Gym

RL Toolkits and algorithm support



© 2019, Amazon Web Services, Inc. or its Affiliates.

SageMaker RL

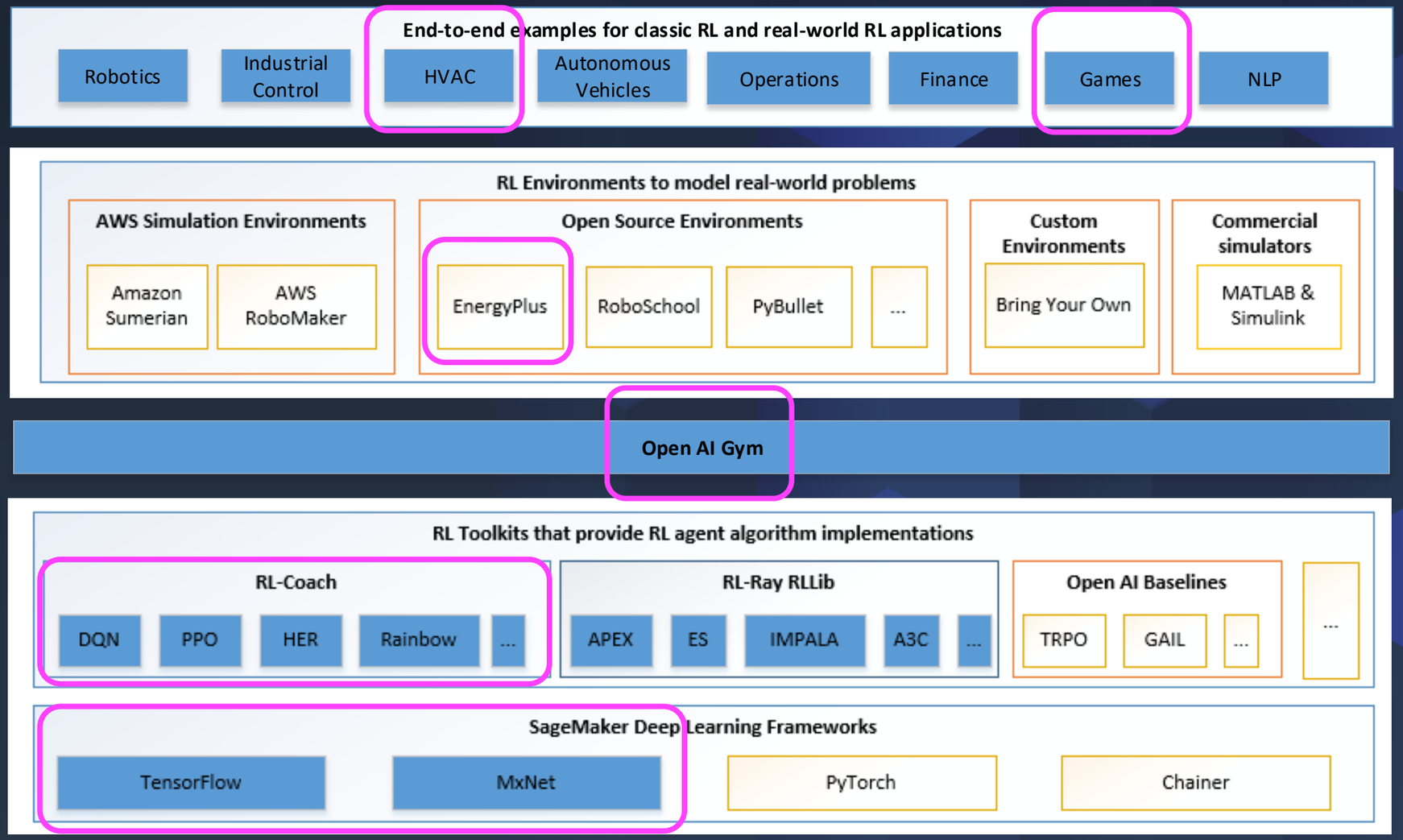
Main Diagram

Sample Notebooks

RL environments

OpenAI Gym

RL Toolkits and algorithm support



© 2019, Amazon Web Services, Inc. or its Affiliates.

Sample Workflow Using Amazon SageMaker RL

1. **Formulate the business problem into an RL problem defining the components of the MDP:**
   - Goal
   - Environment
   - State
   - Action
   - Reward
2. Define the **RL environment**
3. Define the **Presets**
4. Write the **training code** as a Python Script.
5. **Train the RL Model.** Use the RLEstimator from SageMaker Python SDK.
6. **Visualize** the training metrics and output
7. **Evaluate** the model
8. **Deploy** the RL Models

© 2019, Amazon Web Services, Inc. or its Affiliates.

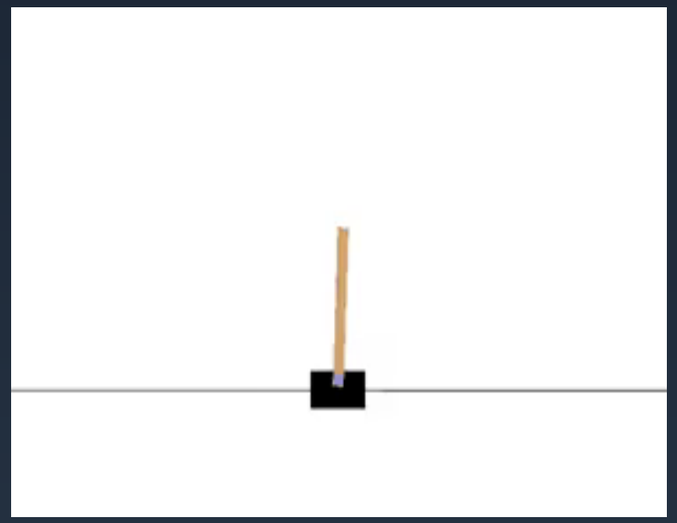
aws

The CartPole problem

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

The Cartpole problem



Cartpole OpenAI-Gym

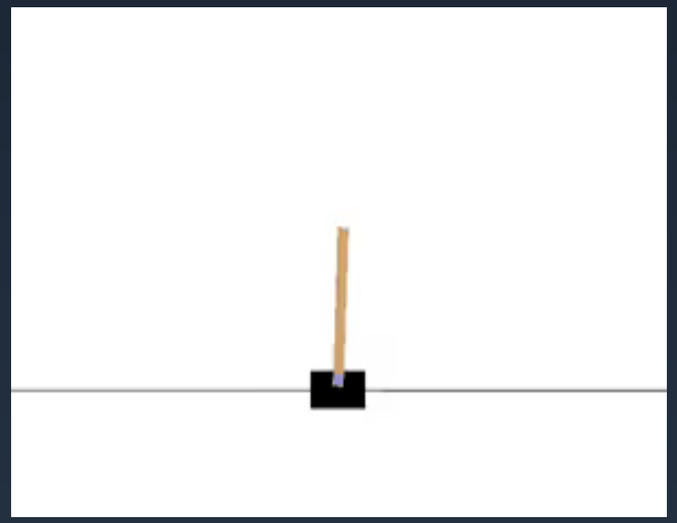
A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the centre.

Cartpole-v0 defines "solving" as getting average reward of 195.0 over 100 consecutive trials. The environment corresponds to the version of the cart-pole problem described by Barto, Sutton and Anderson.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

The Cartpole problem



Cartpole OpenAI-Gym

1. Objective: Prevent the pole from falling over.
2. Environment: Cartpole Environment OpenAI Gym
3. Agent State: Cart position, Cart velocity, Pole angle, Pole velocity.
4. Action Set: Push cart left, push cart right.
5. Reward: +1 for every time step, including termination.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

The Cartpole problem

- Observation space: Box(2)

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

- Action space: Discrete(2)

Num	Action
0	Push cart to the left
1	Push cart to the right

- Episode Termination: Box(2)

1. Pole Angle is more than ±12°
2. Cart Position is more than ±2.4 (center of the cart reaches the edge of the display)
3. Episode length is greater than 200 (500 for v1).

https://github.com/openai/gym/wiki/CartPole-v0

© 2019, Amazon Web Services, Inc. or its Affiliates.

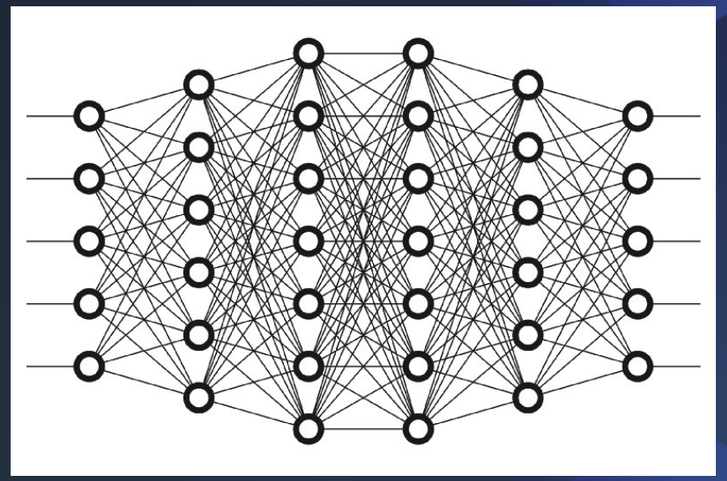
aws

Solving the CartPole problem with : DQN Agent

Definition

Input: State

$$\begin{bmatrix} Cart\ Velocity \\ Cart\ Position \\ Pole\ Angle \\ Pole\ Angular\ Velocity \end{bmatrix}$$



Output: Action Values

$$Q\begin{bmatrix} Left \\ Right \end{bmatrix}$$

$$Q^*(s,a) \approx Q(s,a,\theta)$$

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

DQN Agent

Training the Network

DQN uses experience replay and fixed Q-targets

- Take action a_t according to the $\varepsilon - greedy$ policy.
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory D.
- Sample random mini-batch of transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ from D.
- Compute Q-learning targets by using the fixed parameters $\overline{w}$
- Optimize the MSE between Q-Network and Q-learning targets.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

DQN Agent

Training the Network

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

DQN Preset Definition

Coach API

https://nervanasystems.github.io/coach/_modules/rl_coach/agents/dqn_agent.html#DQNAlgorithmParameters

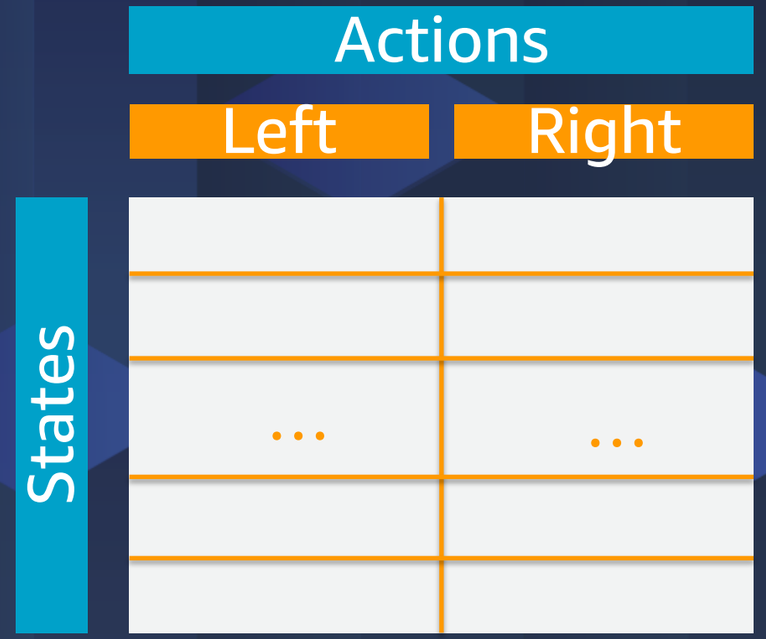
© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Solving the CartPole Problem with: Q-Learning

- Divide the state space into discrete buckets.
  - Per each of the state variables, you define X bins.

Initialize Q(s,a)
Repeat for each episode:
    Initialize s
    Repeat for each time-step:
        Choose a from s using policy derived from Q
            (e.g. epsilon-greedy)
        Take action a, observe r, s′
        $Q(s,a) = Q(s,a) + \alpha[r + \gamma max_{\alpha'} Q(s',a') - Q(s,a)]$
        s = s′
    until s is terminal



© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Solving the CartPole Problem

Batch Reinforcement Learning with Offline data

https://github.com/NervanaSystems/coach/blob/master/tutorials/4.%20Batch%20Reinforcement%20Learning.ipynb

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Break (10 minutes)

© 2019, Amazon Web Services, Inc. or its Affiliates.

Part 3

HVAC (Heating, Ventilation and Air Conditioning)

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

HVAC

- HVAC takes up a whooping 50% of the energy in a building and accounts for 40% of energy use in the US.
- Reinforcement Learning is a good fit because it can learn how to interarct with the environment and identify strategies to limit wasted energy.
- In the Notebook, we will use a simulator to train the agent: EnergyPlus.
- EnergyPlus is an open source, state of the art HVAC simulator from the US Department of Energy.

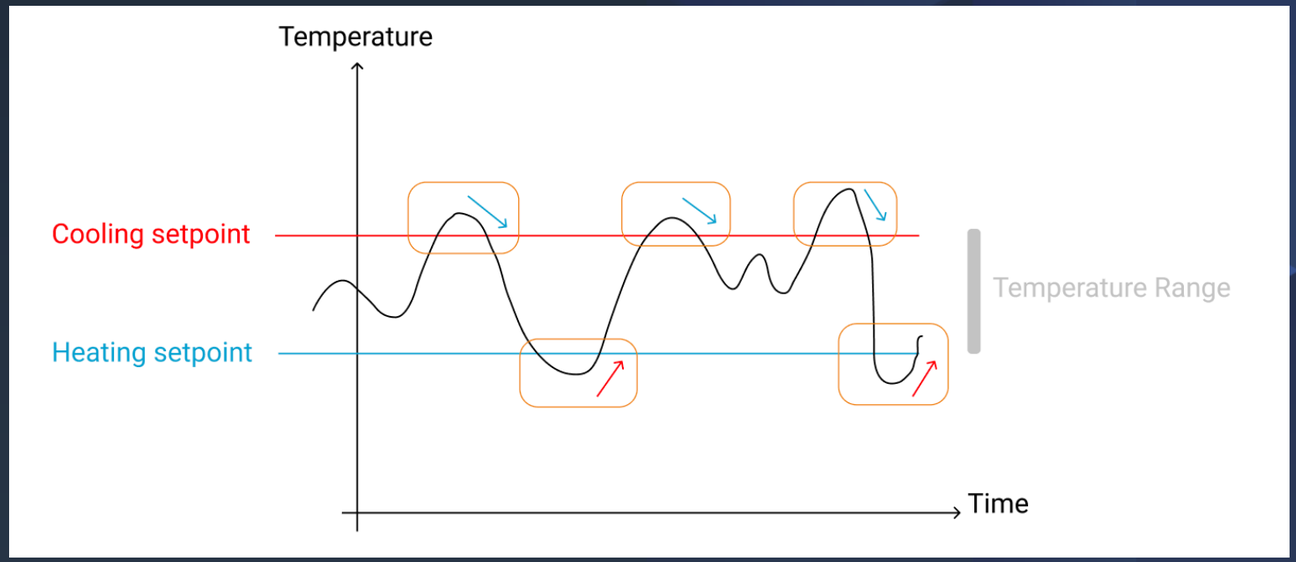
© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

HVAC

- HVAC Systems help to keep temperature of a room within a specific temperature range.

- The range is defined by two setpoints:
  - The lower limit, which is called the heating setpoint.
  - The upper limit, which is called the cooling setpoint.

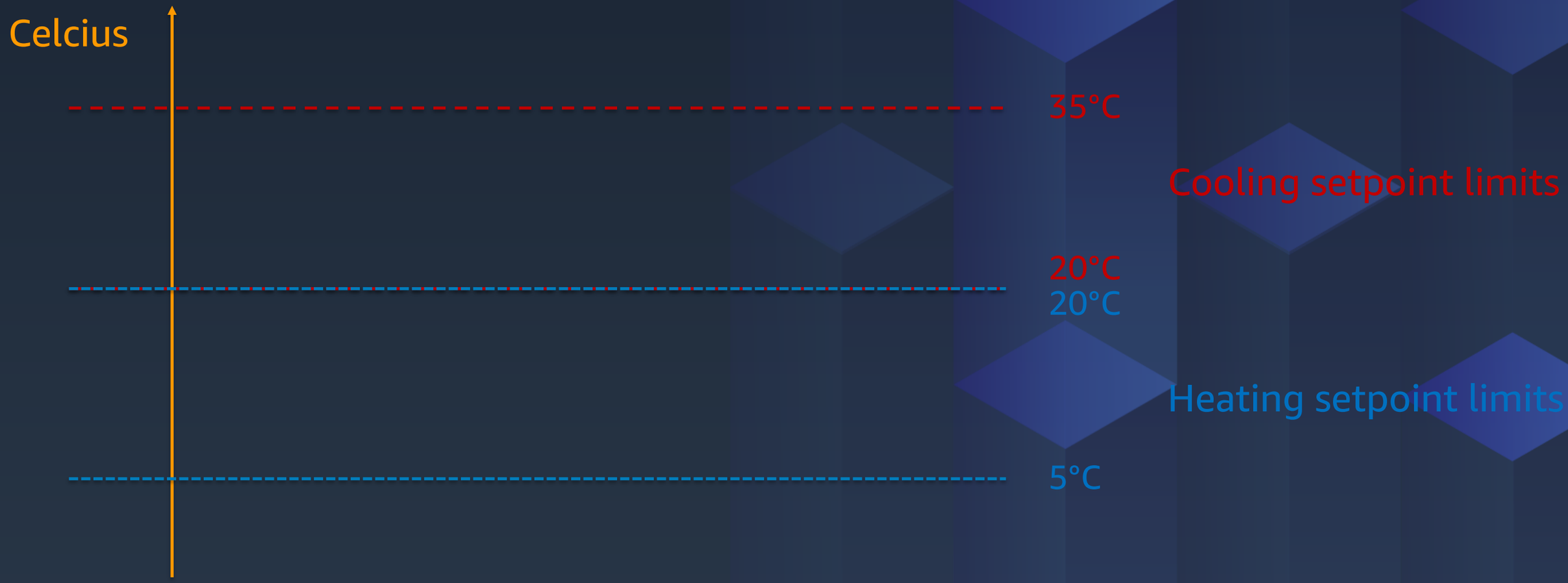


© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

HVAC Use case

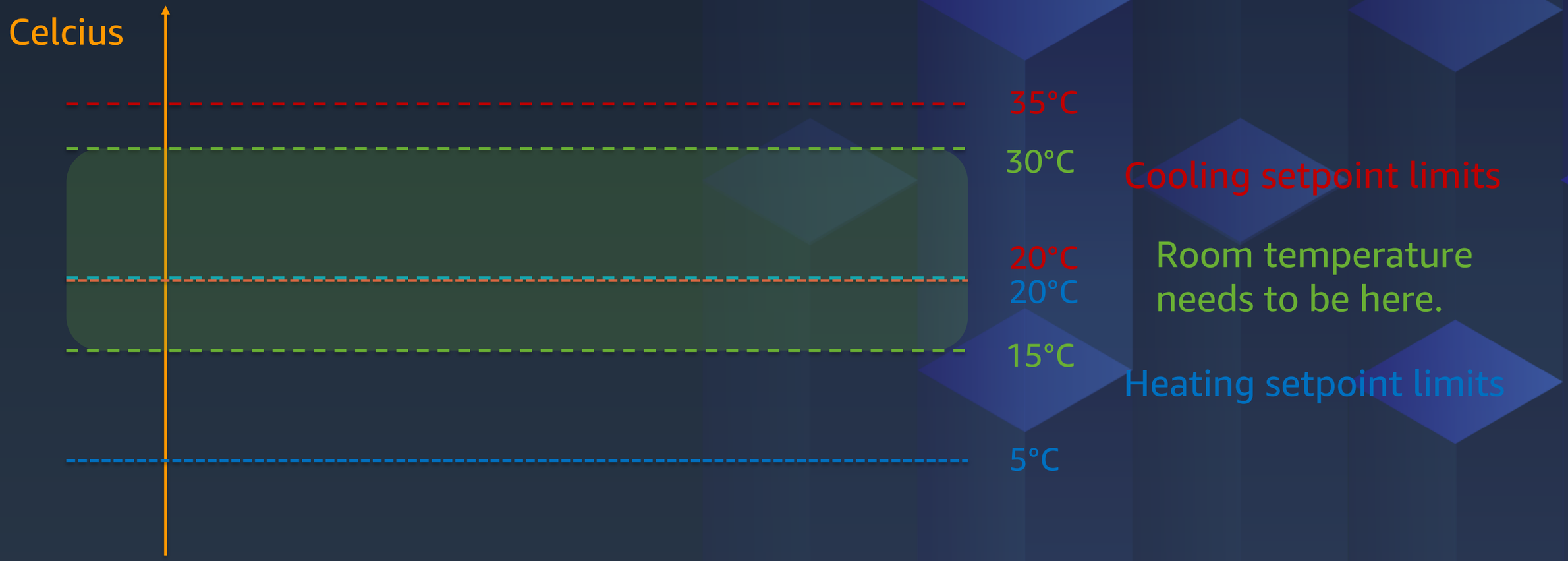
Our HVAC machine has its cooling and heating setpoint limits.



© 2019, Amazon Web Services, Inc. or its Affiliates.

HVAC Use case

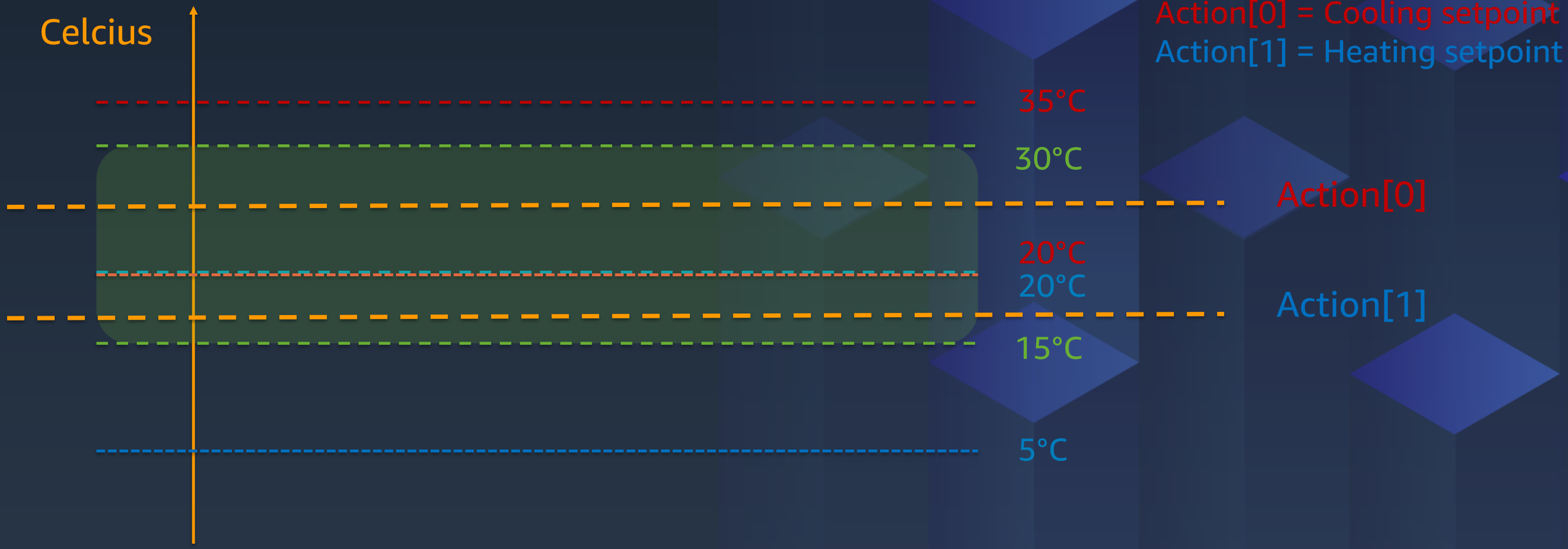
On the other hand, our Data Center only accepts temperatures between 15-30°C



© 2019, Amazon Web Services, Inc. or its Affiliates.

HVAC Use case

Our system needs to predict the two setpoints within the specifications.



© 2019, Amazon Web Services, Inc. or its Affiliates.

HVAC

1. Objective: Control the data center HVAC system to reduce the energy consumption while ensuring the room temperature stays within specified limits.

2. Environment: We have a small single room datacenter that the HVAC system is cooling to ensure the compute equipment works properly. We will train our RL agent to control this HVAC system for one day.

3. State: The outdoor temperature, outdoor humidity and indoor room temperature.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

HVAC

4. Action: The agent can set the heating and cooling setpoints. The cooling setpoint tells the HVAC system that it should start cooling the room if the room temperature goes above this setpoint. Likewise, the HVAC systems start heating if the room temperature goes below the heating setpoint.

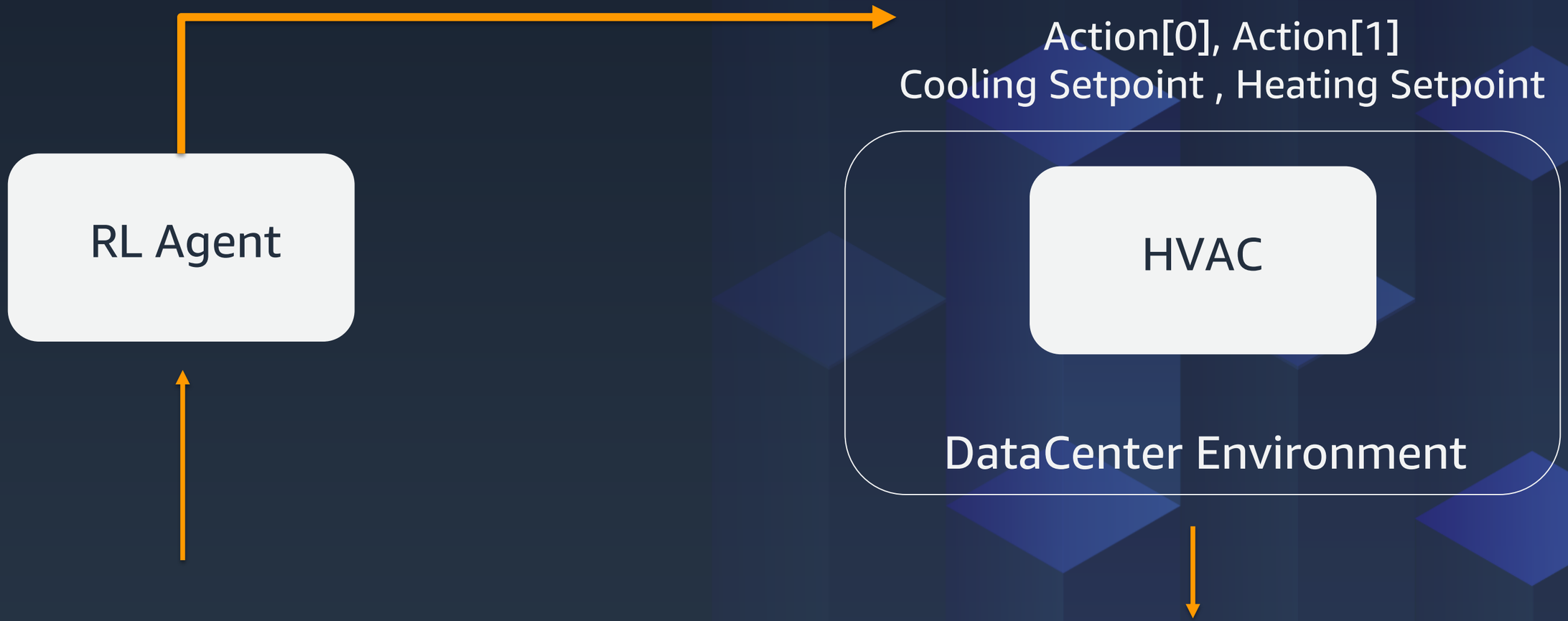
5. Reward:

reward = -Energy Consumed – Penalty of exceeding temperature limits

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

HVAC



$$O_t = \{room\ temperature, outdoor\ temperature, outdoor\ humidity\}$$

$$R_t = reward\ signal$$

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

HVAC

Reward Signal ./eplus/envs/data_center_env.py

Reward = energy_coeff * energy + heating_coeff * heating_penalty + cooling_coeff * cooling_penalty + action_penalty

Energy Penalty: Based on energy consumption.
Heating Penalty: If heating setpoint < 15°C
Cooling Penalty: If cooling setpoint > 30°C
Action Penalty: If action is out of limits (35°C- 20°C, 20°C- 5°C)

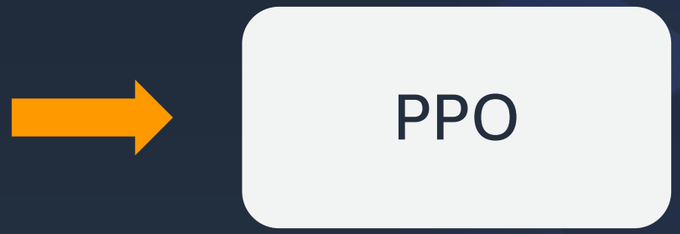
© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Proximal Policy Optimization (PPO)

$S_t = \{room\ temperature, outdoor\ temperature, outdoor\ humidity\}$

$R_t = reward\ signal$



$a_t = \{\text{Heating Setpoint, Cooling setpoint}\}$

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

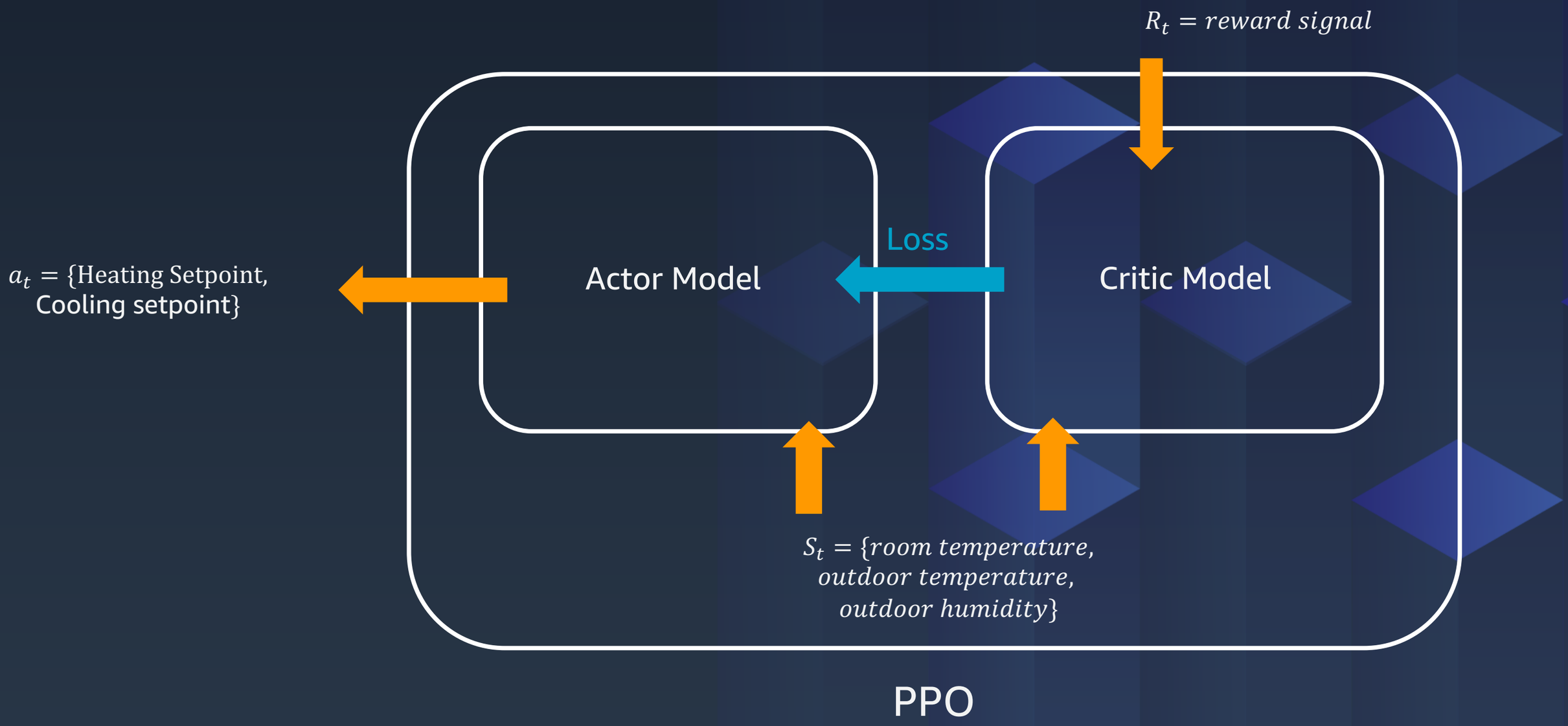
Proximal Policy Optimization (PPO)

- Introduced by OpenAI team in 2017.
- Collects a small batch of experiences interacting with the environment and using that batch to update its decision-making policy.
- On-policy learning: Once the policy is updated with this batch, the experiences are thrown away and a newer batch is collected with the newly updated policy. Experience samples collected are only useful for updating the current policy once.
- Key contribution: New update policy does not change it too much from the previous policy.

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Proximal Policy Optimization (PPO)



© 2019, Amazon Web Services, Inc. or its Affiliates.

Proximal Policy Optimization (PPO)

- Run through this entire loop with the two models for a fixed number of steps known as PPO Steps. The PPO steps is basically the batch of samples experience.

- Basically, we interact with our environment for certain number of steps and we collect states, actions,rewards, which we will use for training.

- The experience will be used to update our models after we have seen a large enough batch of such samples.

{States, actions, rewards, output actor, output critic}

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Proximal Policy Optimization (PPO)

- PPO Upgrades:

1. Generalized Advantage Estimation

$$\delta = r + \gamma V(s') * mask - V(s)$$
$$gae = \delta + \gamma V(s') * mask - V(s)$$
$$return(s, a) = gae + V(s)$$

1. Surrogate Policy Loss

3. Mini-batch Updates

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Part 4

6:00pm-6:30pm

AWS DeepRacer and Summary

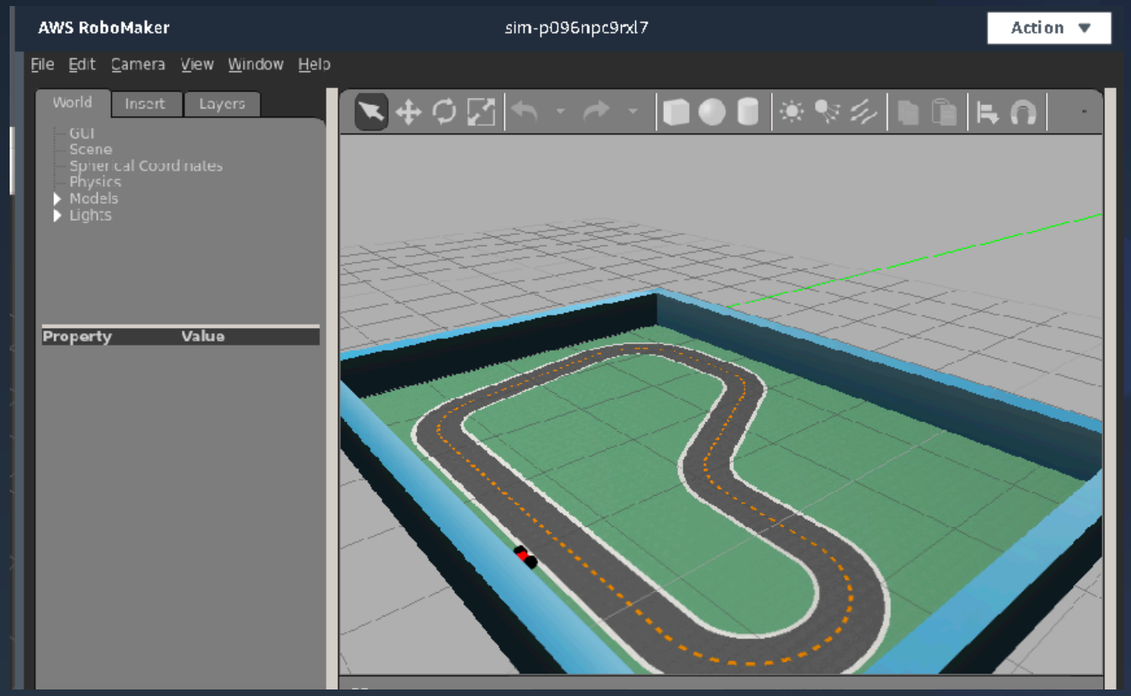
© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

AWS DeepRacer

© 2019, Amazon Web Services, Inc. or its Affiliates.

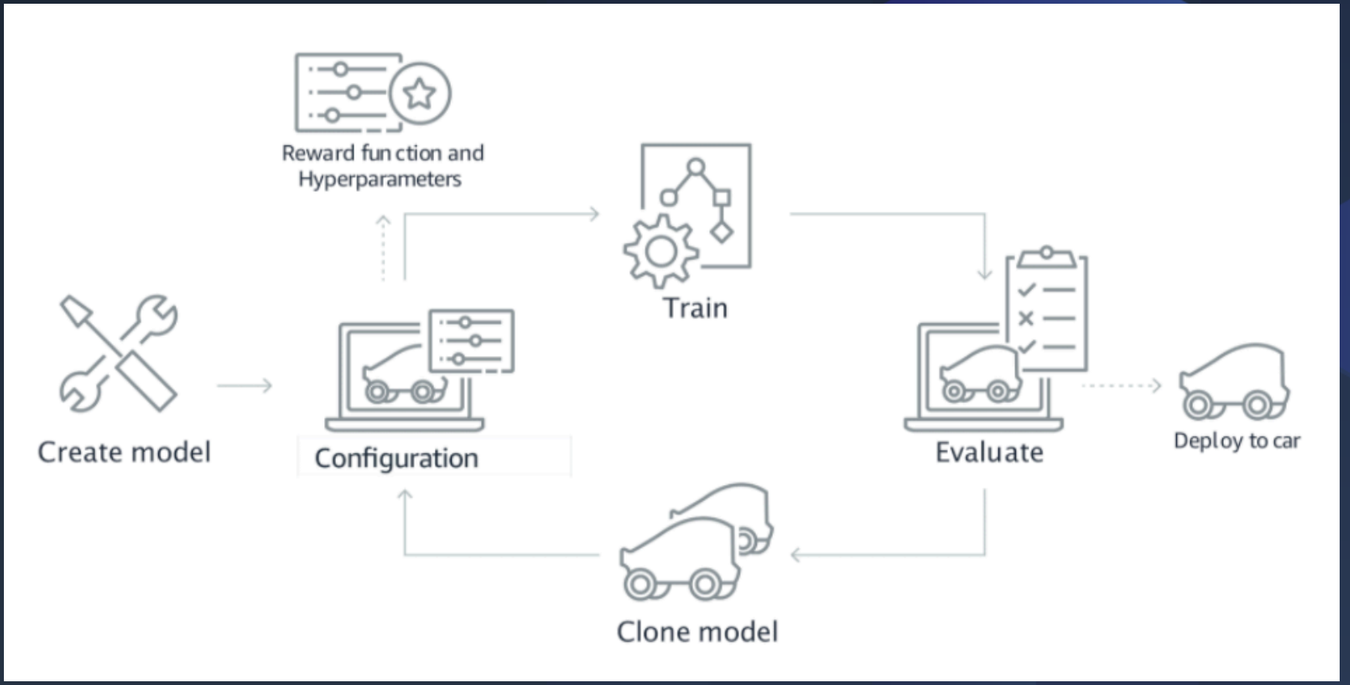
DeepRacer



© 2019, Amazon Web Services, Inc. or its Affiliates.

DeepRacer

Workflow



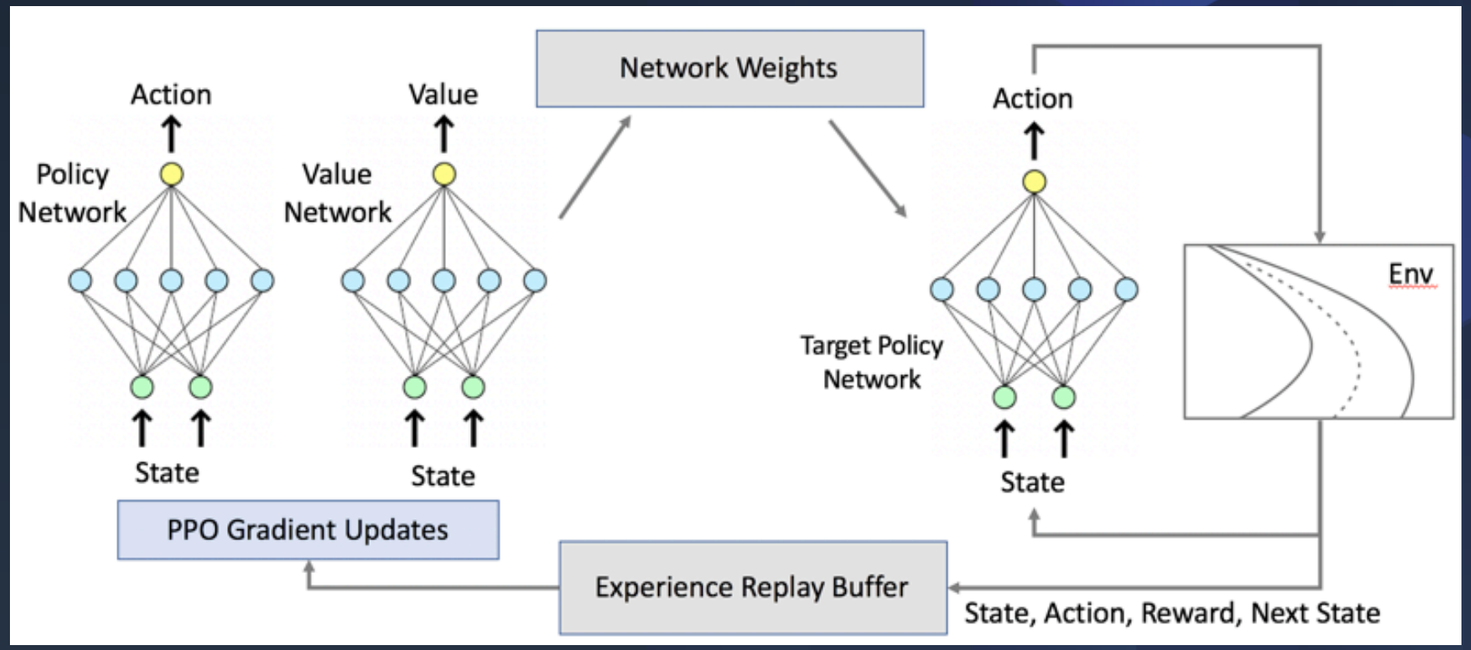
DeepRacer Console, Notebook example, Workshop

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

DeepRacer

Proximal Policy Optimization (PPO)



AWS DeepRacer Training Algorithm

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Resources and Summary

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Summary

- Understand Reinforcement Learning Basic Concepts
- How to formulate a business problem into a RL Framework.
- Understand the different types of RL agents.
- Familiarize yourself with SageMaker RL and RL Toolkits.
- Run some example Notebooks to get the feeling of training problems with SageMaker RL.
    - CartPole problem
    - HVAC use case

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Resources

Reinforcement Learning Theory

- Book

  Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning: An introduction*. MIT press.

- Online course

  RL Course by David Silver
  http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
  https://www.youtube.com/playlist?list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ

  Advanced Deep Learning and Reinforcement Learning by DeepMind and UCL
  https://www.youtube.com/watch?v=iOh7QUZGyiU&list=PLqYmG7hTraZDNJre23vqCGIVpfZ_K2RZs&index=1

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Resources

SageMaker RL

- SageMaker Developer Guide
  https://docs.aws.amazon.com/sagemaker/latest/dg/reinforcement-learning.html
- SageMaker Python SDK for Reinforcement Learning
  https://sagemaker.readthedocs.io/en/stable/frameworks/rl/using_rl.html
- RL Images Provided by SageMaker
  https://github.com/aws/sagemaker-rl-container/#rl-images-provided-by-sagemaker
- Reinforcement Learning Notebook Examples
  https://github.com/awslabs/amazon-sagemaker-examples/tree/master/reinforcement_learning

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Resources

RL Toolkits and Frameworks

- Open-AI Gym
  https://gym.openai.com/envs/
- Reinforcement Learning Coach Toolkit
  https://github.com/NervanaSystems/coach
- Ray/RLlib
  https://docs.ray.io/en/master/rllib.html
- How to select the right RL algorithm
  https://nervanasystems.github.io/coach/selecting_an_algorithm.html

© 2019, Amazon Web Services, Inc. or its Affiliates.

aws

Thank you

Sandra Pico Oristrell
Professional Services – Associate Data Science
oristrel@amazon.com

© 2019, Amazon Web Services, Inc. or its Affiliates.