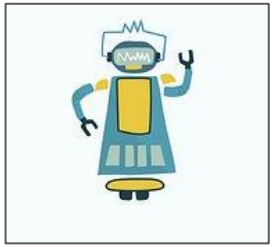
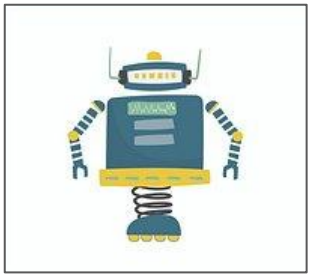
INTRODUCTION TO

Reinforcement Learning





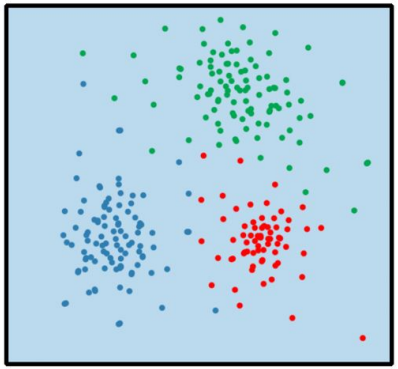
AGENDA

01. What Is Reinforcement Learning?
02. Markov Decision Process MDP
03. Evaluating Policies
04. Aside: Preliminary Machine Learning Background
05. Code Walkthrough: DQN

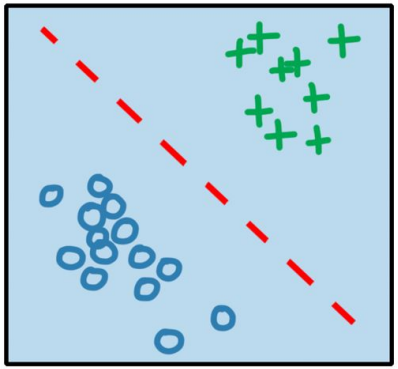
What Is Reinforcement Learning?

Machine Learning

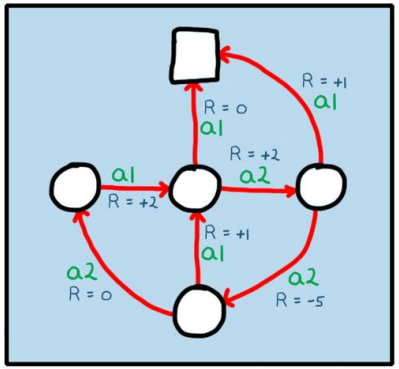
Supervised Learning



Unsupervised Learning

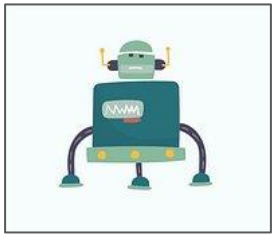


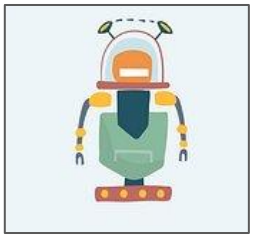
Reinforcement Learning

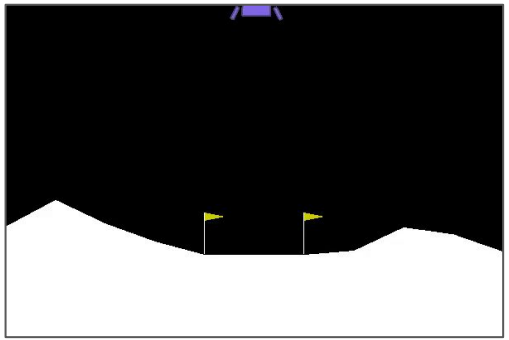


What Is Reinforcement Learning?

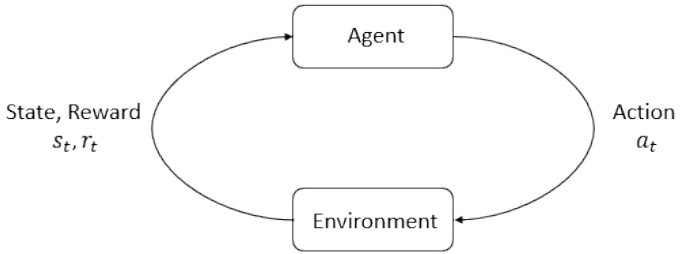
Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment. The agent receives feedback from the environment in the form of a reward or a penalty and learns from this feedback to make better decisions. The goal is to maximize cumulative reward over time.







Markov Decision Process MDP



Trajectory/Episode: a sequence of states and actions representing a single run of the agent interacting with the environment $\tau = (s_0, a_0, s_1, a_1, ...)$.

Policy (π_θ): a mapping from states to actions that defines the behaviour of the agent in the environment

The goal of reinforcement learning is to find the optimal policy π_θ^*.

Evaluating Policies

(Undiscounted) Return: $$R(\tau) = r_0 + r_1 + \cdots + r_T = \sum_{t=0}^{T} r_t$$

Discounted Return: $$R(\tau) = \gamma^T r_0 + \gamma^{T-1} r_1 + \cdots + r_T = \sum_{t=0}^{T} \gamma^t r_t \qquad \gamma \in (0, 1)$$

State-Value Function: $$V^\pi(s) = \mathop{\mathrm{E}}_{\tau \sim \pi} \left[R(\tau) \,|s_0 = s \right]$$

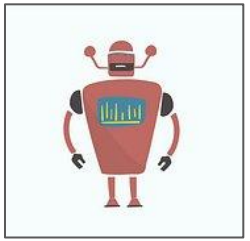
Action-Value Function: $$Q^\pi(s, a) = \mathop{\mathrm{E}}_{\tau \sim \pi} \left[R(\tau) \,|s_0 = s, a_0 = a \right]$$

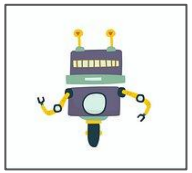
Evaluating Policies

Bellman Equations:

$$V^*(s) = \max_a \mathop{\mathrm{E}}_{s' \sim P} [r(s,a) + \gamma V^*(s')]$$

$$Q^*(s,a) = \mathop{\mathrm{E}}_{s' \sim P} \left[r(s,a) + \gamma \max_{a'} Q^*(s',a') \right]$$

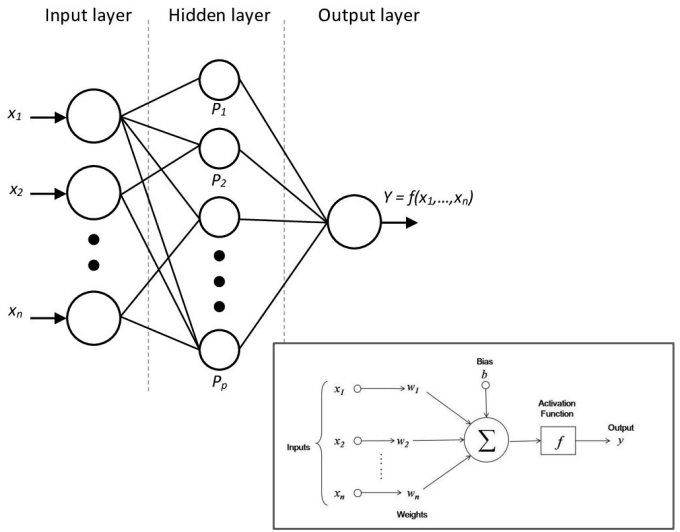




2	1	s_T
3	2	1
4	3	2
s_0	4	3

Aside: Preliminary Machine Learning Background

BGK #1: A neural network is a function approximator



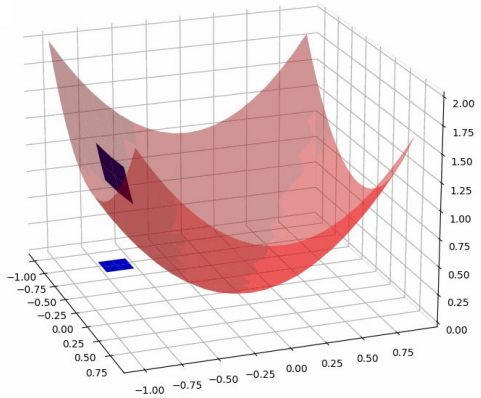

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class PolicyNetwork(nn.Module):
    """ A Parameterized Policy Network π_θ """

    def __init__(self, n_obs_dim: int, n_action_dim: int, n_hidden_dim=128):
        super().__init__()
        self.fc1 = nn.Linear(n_obs_dim, n_hidden_dim)
        self.fc2 = nn.Linear(n_hidden_dim, n_action_dim)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = F.relu(self.fc1(x))
        x = F.softmax(self.fc2(x), dim=1)
        return x
```

Aside: Preliminary Machine Learning Background

BGK #1: A neural network is a function approximation

BGK #2: This function approximation is fine-tuned via gradient descent

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$




```python
# training loop
for epoch in range(epochs):
    model.train()

    y_pred = model(X_train)
    loss = loss_fn(y_pred, y_train)

    optimizer.zero_grad()

    loss.backward()
    optimizer.step()
```

Code Walkthrough: DQN

code: https://github.com/sabrinahirani/reinforcement-learning-w-gym/blob/main/DQN.ipynb