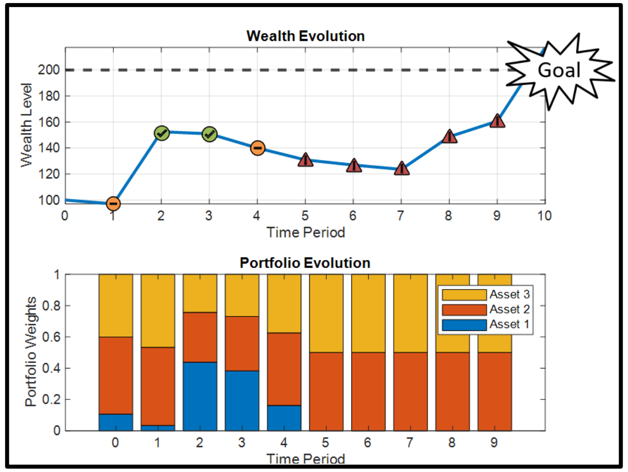
MathWorks
FINANCE CONFERENCE 2023

Multiperiod Goal-Based Wealth Management using Reinforcement Learning

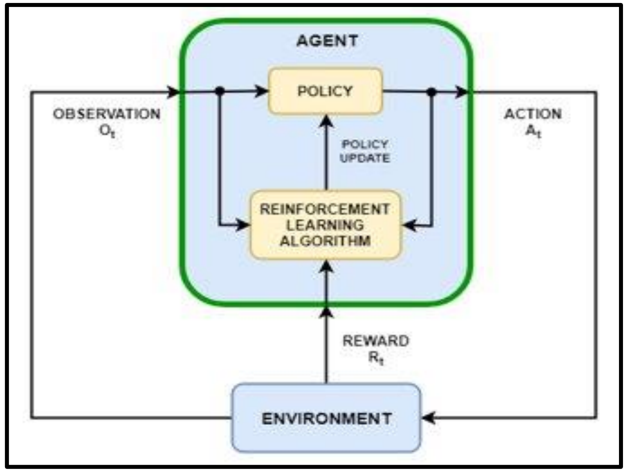
October 11-12 | Online

Valerio Sperandeo, MathWorks

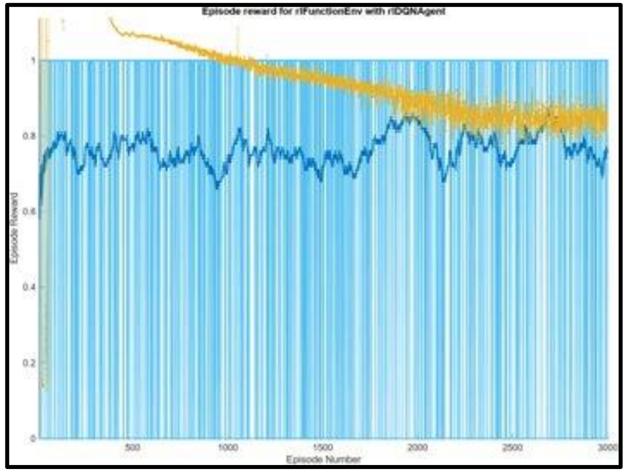
In this presentation we will see..



A Goal-based Wealth Management Problem

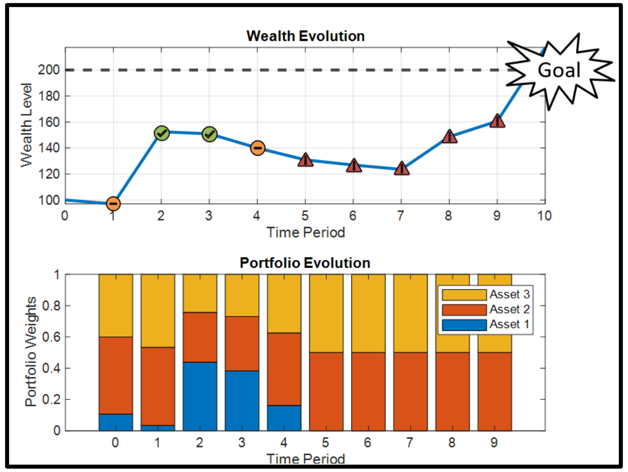


Solving the Problem using Reinforcement Learning

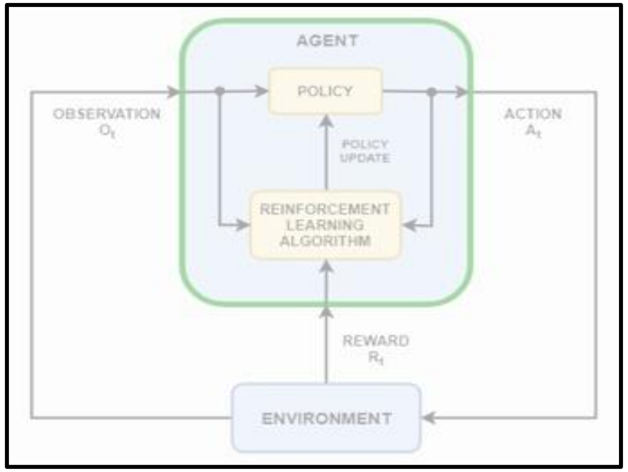


Why use Reinforcement Learning

In this presentation we will see..



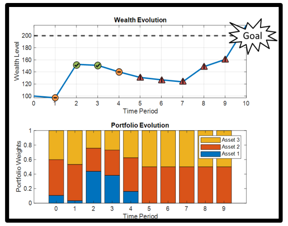
A Goal-based Wealth Management Problem



Solving the Problem using Reinforcement Learning

Why use Reinforcement Learning

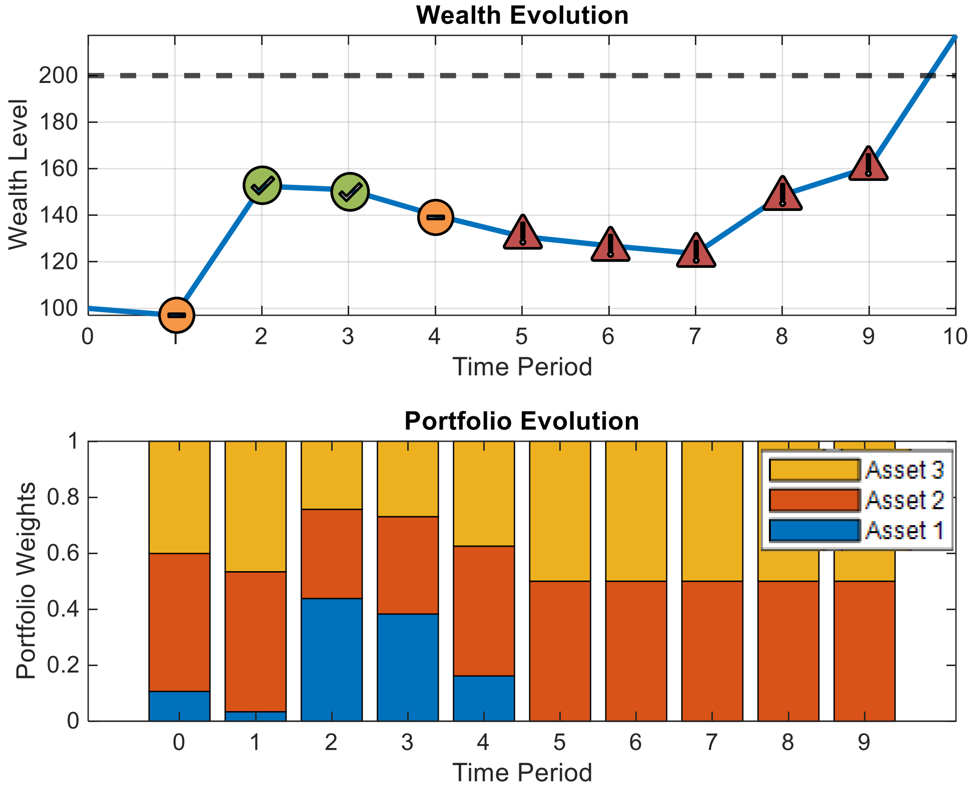
In Goal-Based Wealth Management the asset allocation aims at achieving a specific objective



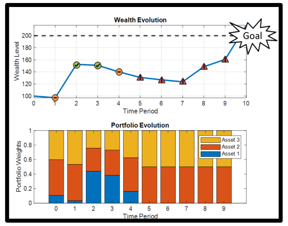
$$\max \ \mathbb{P}(W_{\boxed{T}} \geq \boxed{G})$$

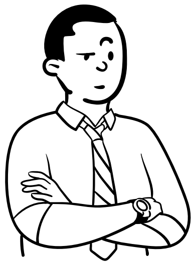
Investment Horizon

Wealth Goal



A classic example of Goal-Based Wealth Management is the retirement problem





Initial Capital W

Start Investment Period

Periodic contribution 1

Periodic contribution 2

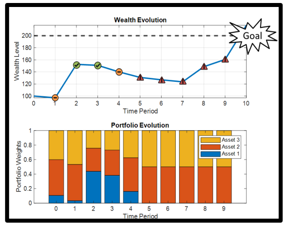
{…}

Periodic contribution *N - 1*

Periodic contribution *N*

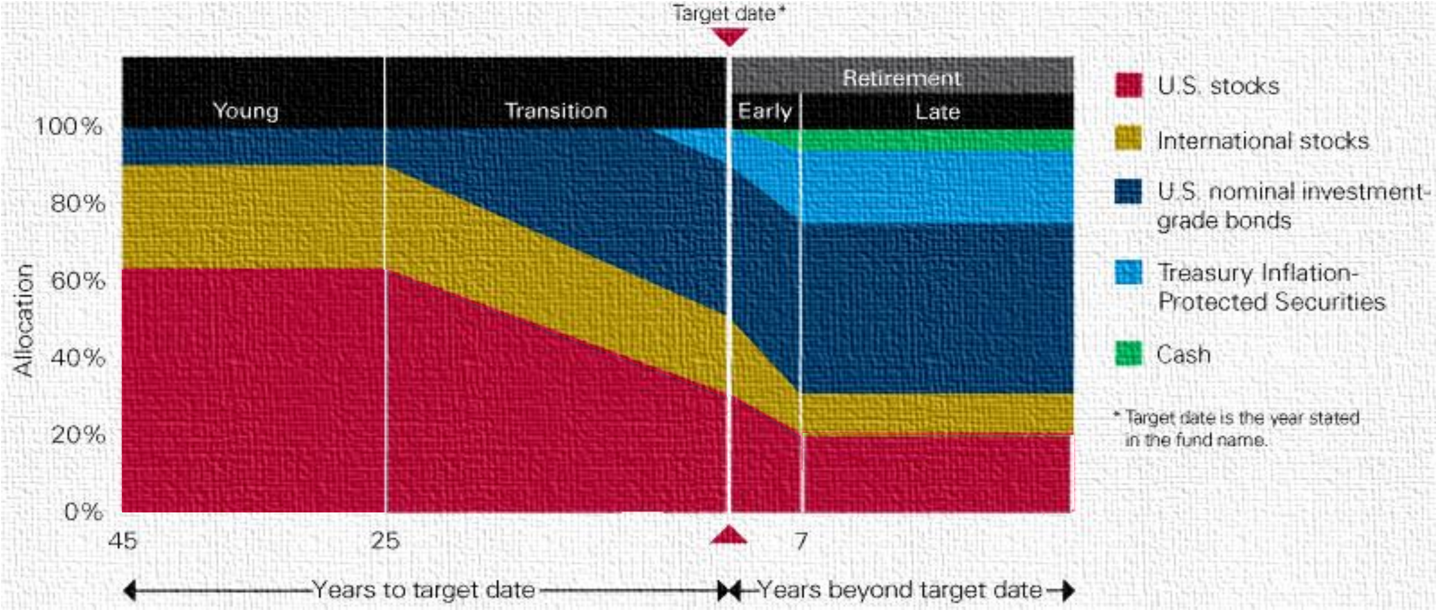
Target Capital G

End Investment Period



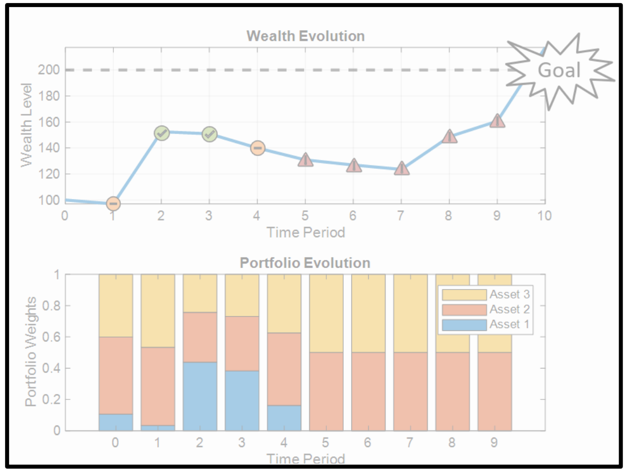
In the retirement problem the asset allocation of the portfolio depends on the proximity to the retirement date

60% Equity → **20% Equity**

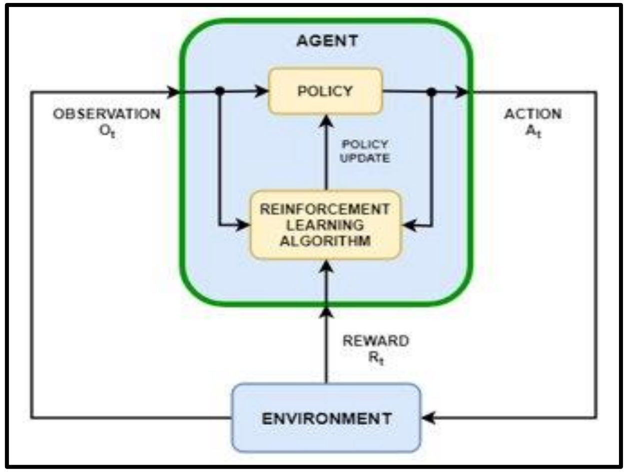


Source: [Vanguard]

In this presentation we will see..



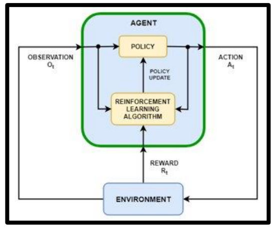
A Goal-based Wealth Management Problem



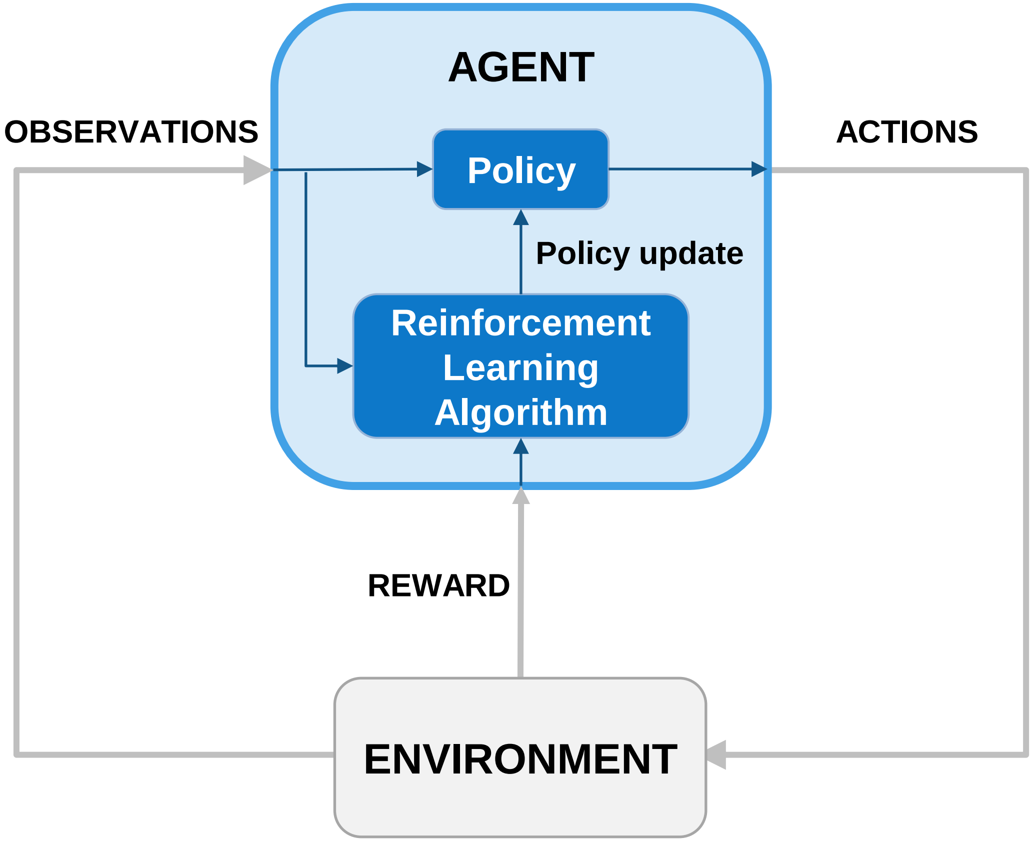
Solving the Problem using Reinforcement Learning

Why use Reinforcement Learning

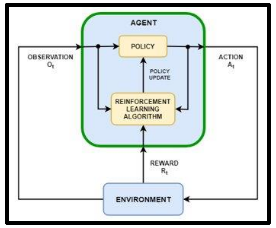
What is Reinforcement Learning?



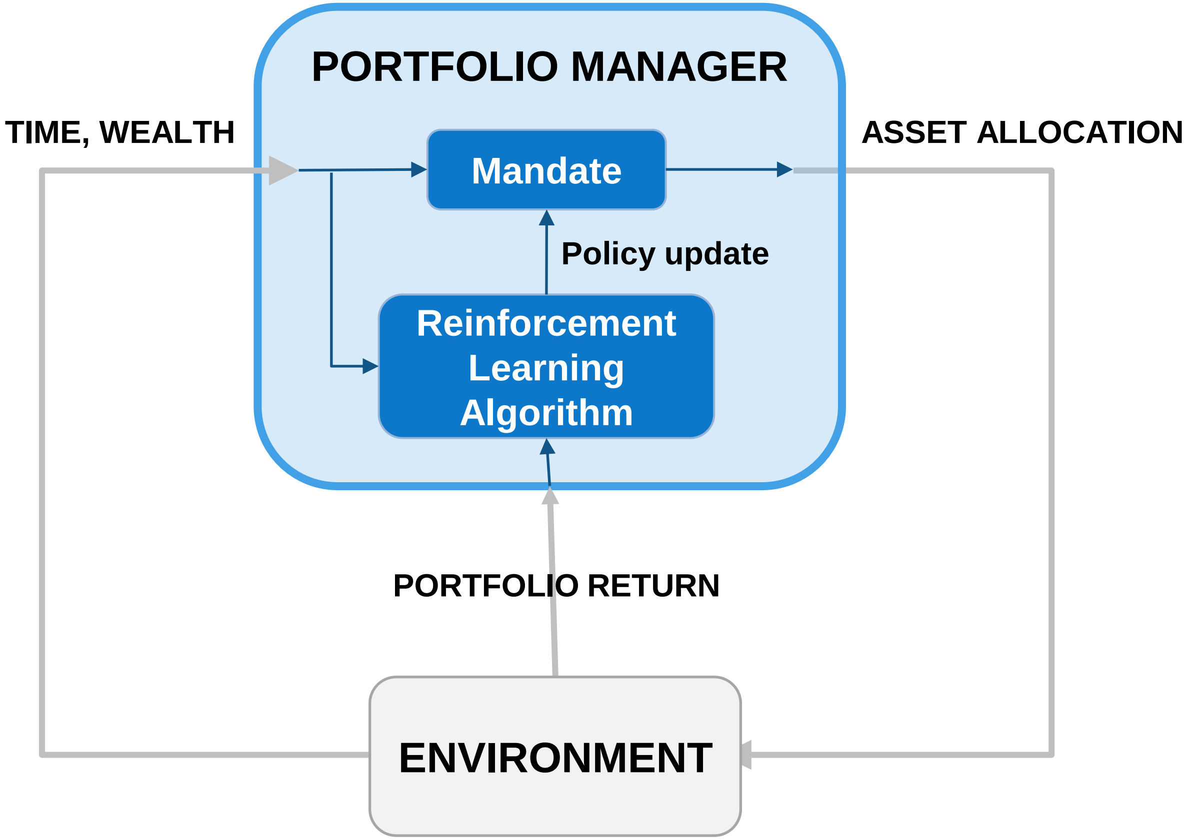
Type of machine learning that trains an **‘agent’** through trial & error interactions with an **environment**



What is Reinforcement Learning?



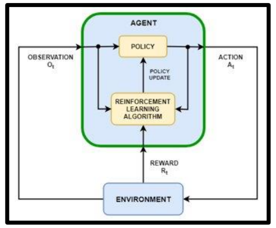
Type of machine learning that trains an **‘agent’** through trial & error interactions with an **environment**

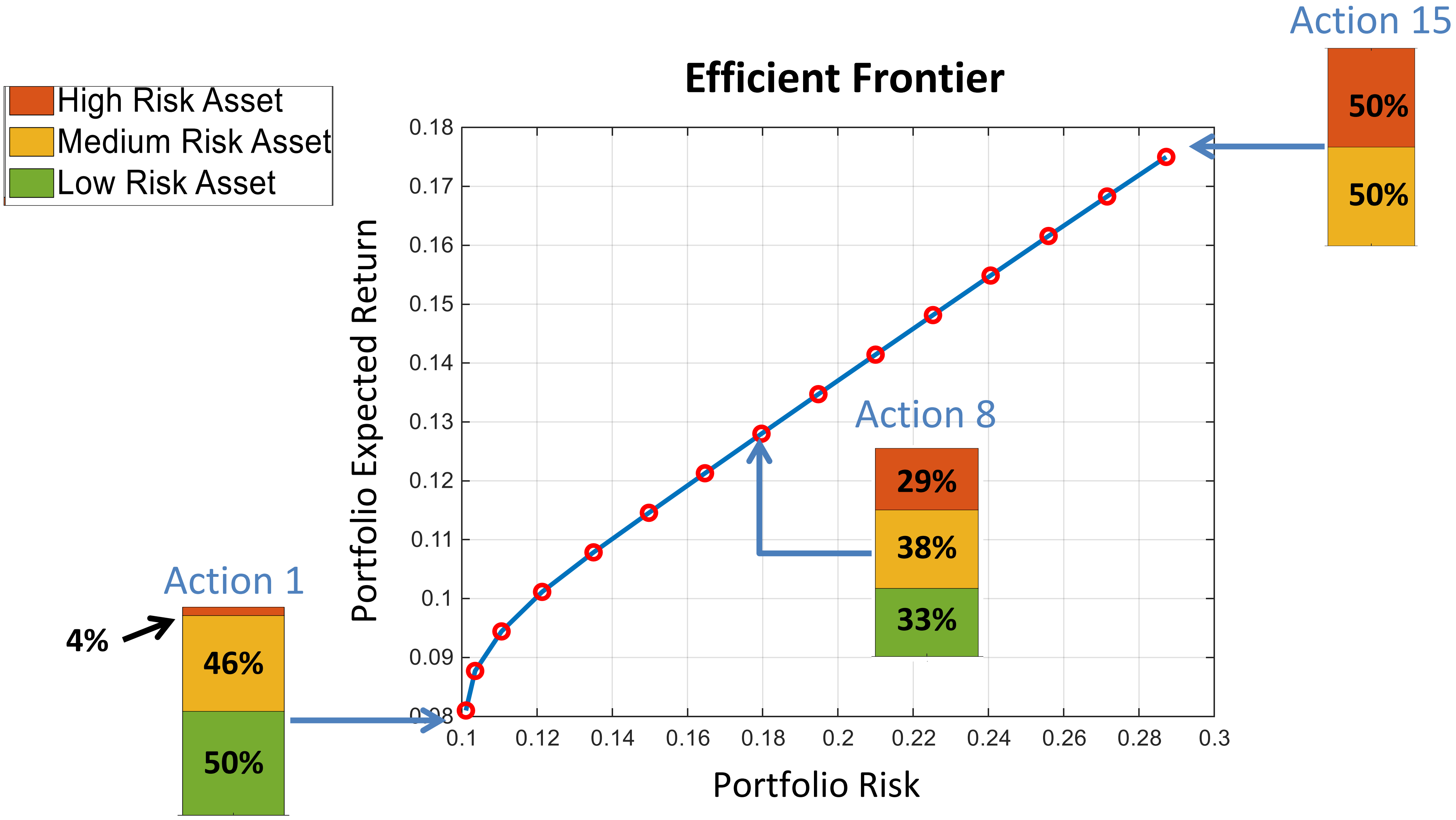


What we need to define:

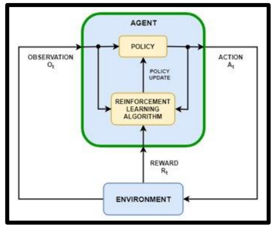
1. *Actions*
2. *Observations*
3. *Environment*
4. *Reward*
5. *Agent*

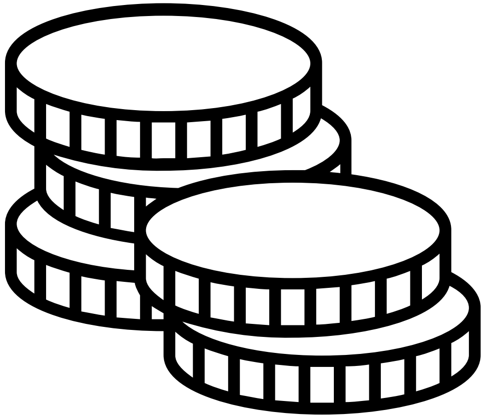
The actions at each rebalancing period are a finite set of portfolio weights on the efficient frontier

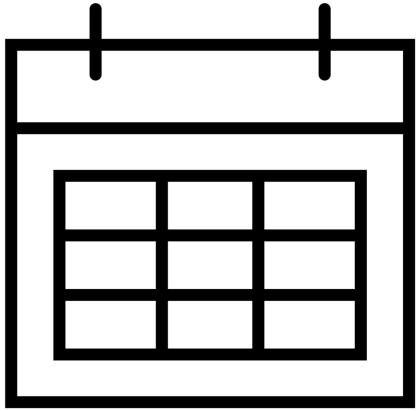




The observations from the environment are the wealth level and the time period







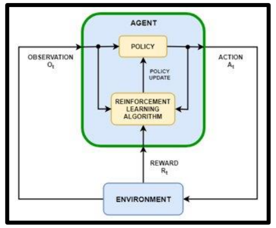
Wealth Level

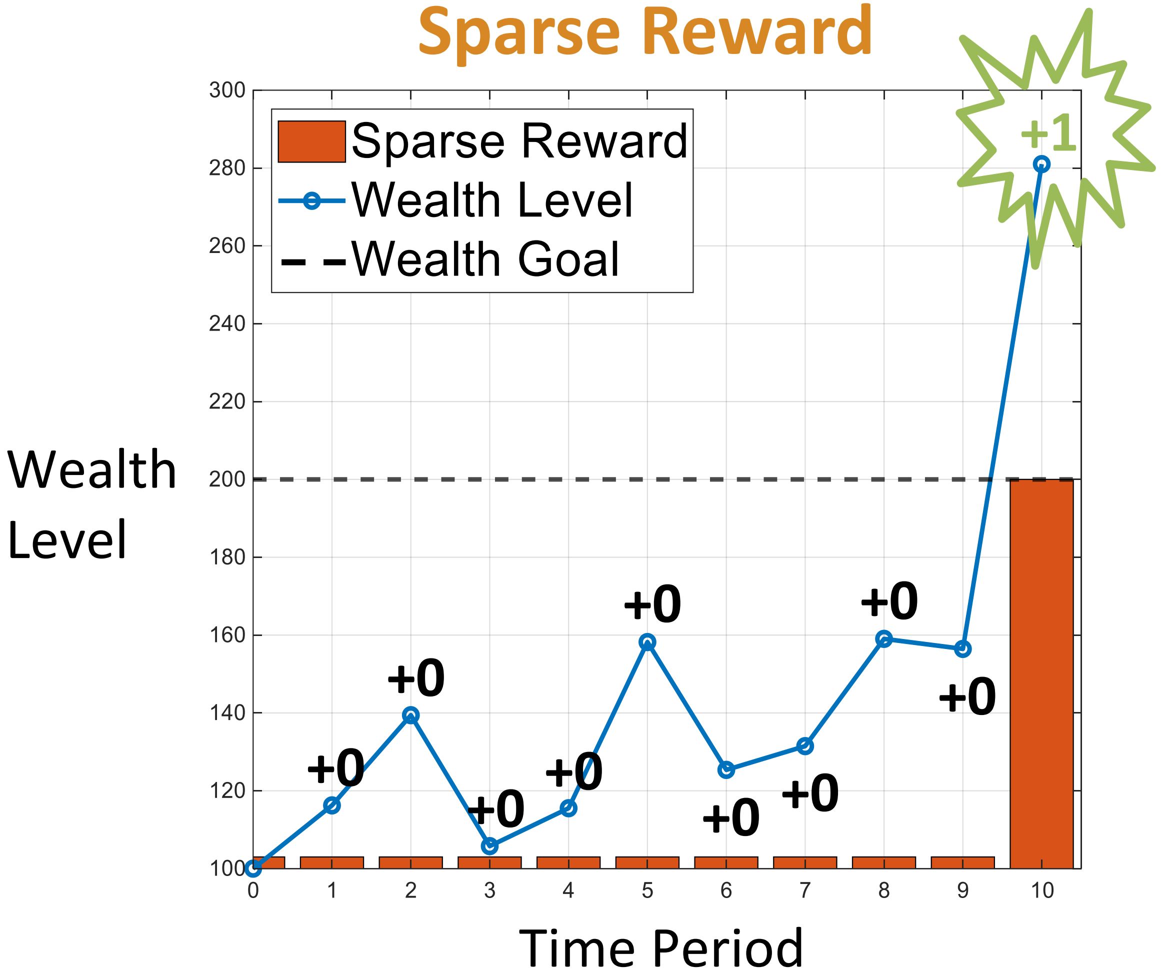
Continuous Variable

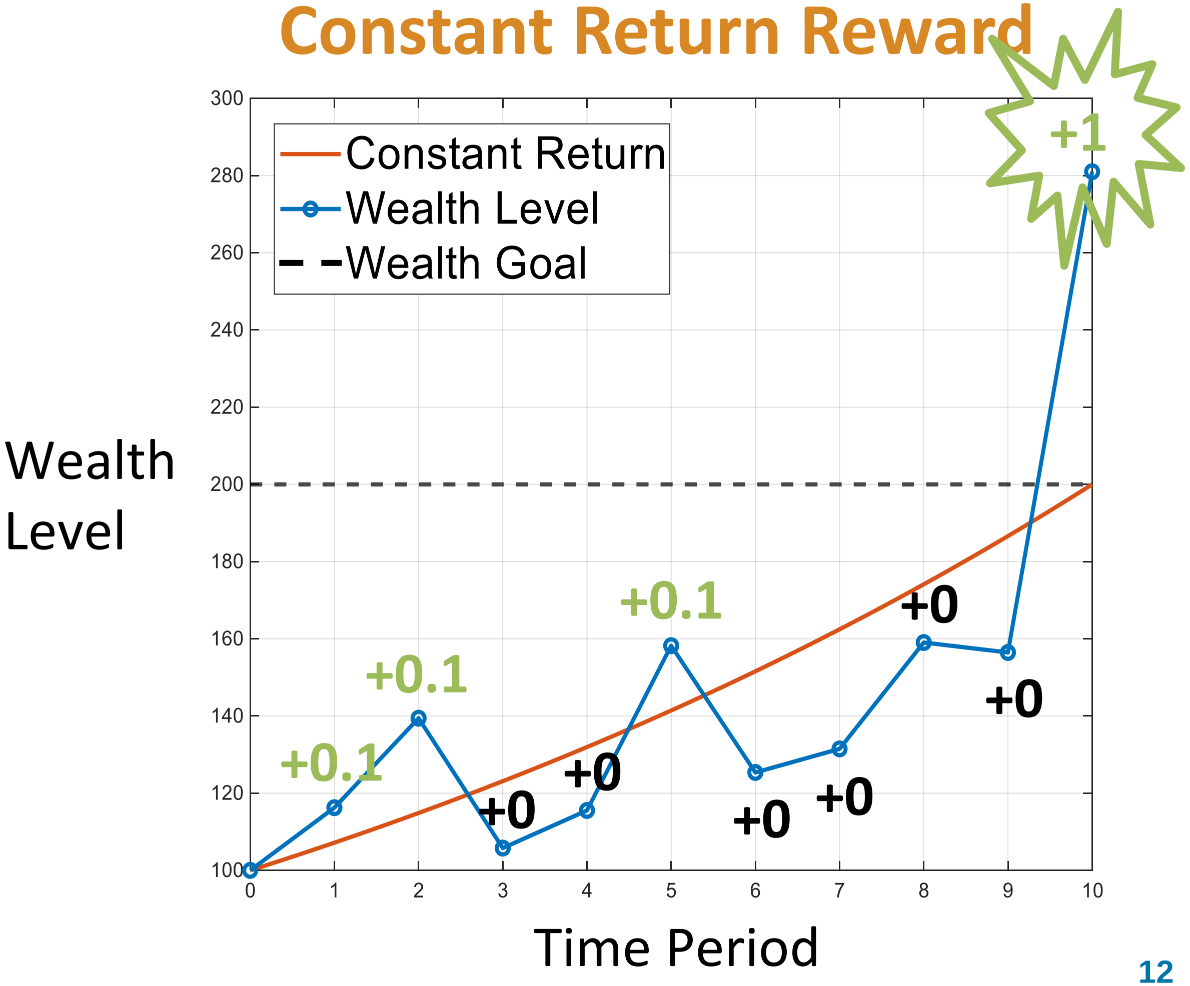
Time Period

Discrete Variable

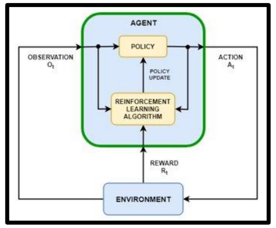
The reward functions provide a compensation for achieving a goal or for following a path

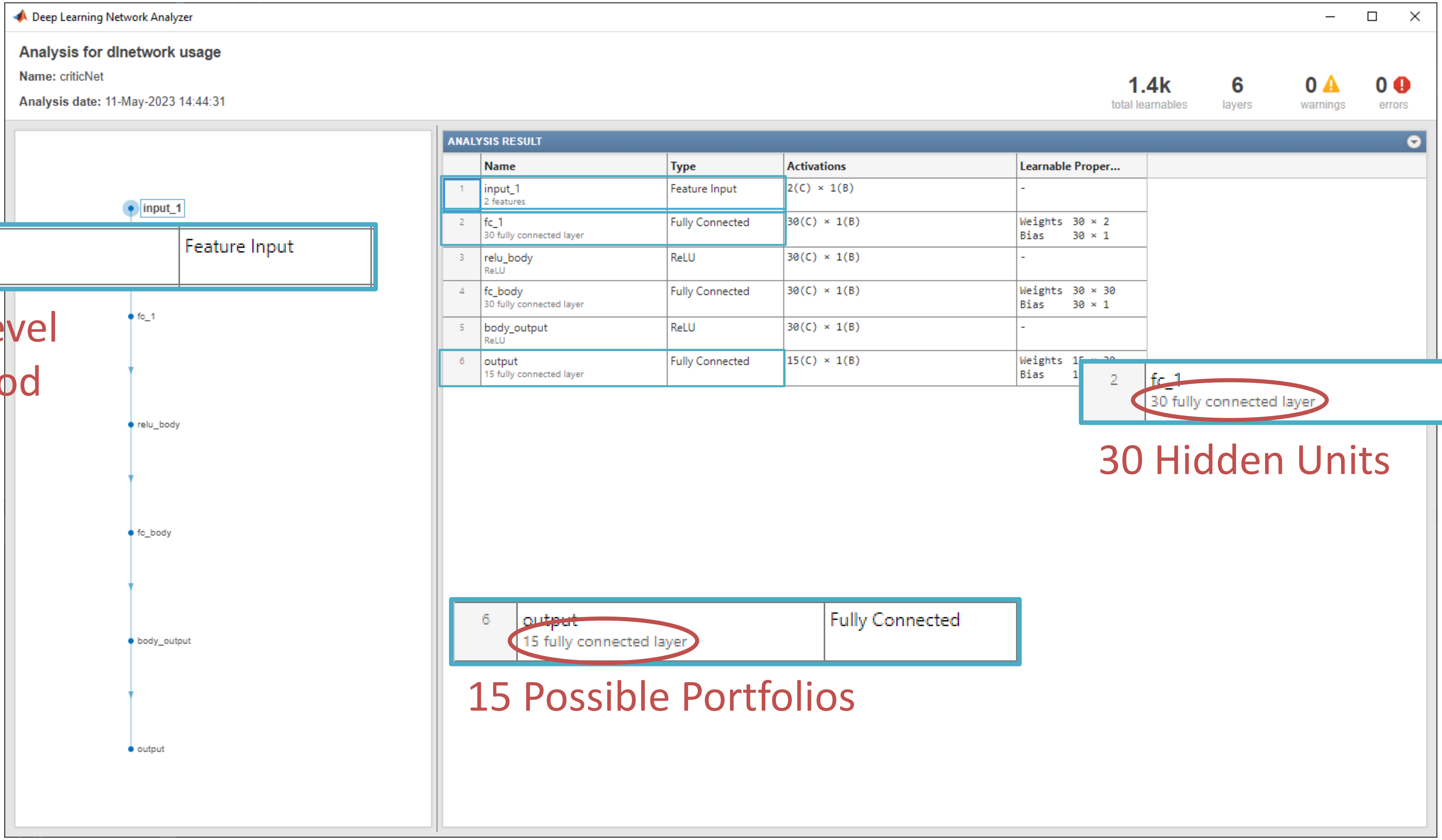






Deep Q-Network (DQN) agent based on the action and observation spaces





	Name	Type	Activations	Learnable Proper...
1	input_1 2 features	Feature Input	2(C) × 1(B)	-
2	fc_1 30 fully connected layer	Fully Connected	30(C) × 1(B)	Weights 30 × 2 Bias 30 × 1
3	relu_body ReLU	ReLU	30(C) × 1(B)	-
4	fc_body 30 fully connected layer	Fully Connected	30(C) × 1(B)	Weights 30 × 30 Bias 30 × 1
5	body_output ReLU	ReLU	30(C) × 1(B)	-
6	output 15 fully connected layer	Fully Connected	15(C) × 1(B)	Weights 15 × 30 Bias 1

1	input_1 2 features	Feature Input

Wealth Level
Time Period

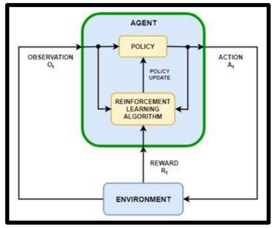
2	fc_1 30 fully connected layer	Fully Connected

30 Hidden Units

6	output 15 fully connected layer	Fully Connected

15 Possible Portfolios

Reinforcement Learning Toolbox provides a rich set of Built-in agents

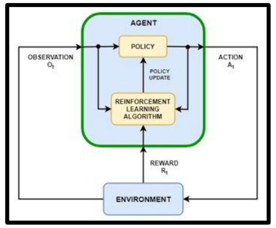


Agent	Type	Action Space
SARSA Agents	Value-Based	Discrete
Policy Gradient (PG) Agents (PG)	Policy-Based	Discrete or continuous
Actor-Critic (AC) Agents (AC)	Actor-Critic	Discrete or continuous
Trust Region Policy Optimization (TRPO) Agents (TRPO)	Actor-Critic	Discrete or continuous
Proximal Policy Optimization (PPO) Agents (PPO)	Actor-Critic	Discrete or continuous
Q-Learning Agents (Q)	Value-Based	Discrete
Deep Q-Network (DQN) Agents	Value-Based	Discrete
Deep Deterministic Policy Gradient (DDPG) Agents	Actor-Critic	Continuous
Twin-Delayed Deep Deterministic (TD3) Policy Gradient Agents (TD3)	Actor-Critic	Continuous
Soft Actor-Critic (SAC) Agents (SAC)	Actor-Critic	Continuous
Model-Based Policy Optimization (MBPO) Agents (MBPO)	Actor-Critic	Discrete or continuous

On-Policy Built-In Agents

Off-Policy Built-In Agents

Any custom agent can be defined as a subclass of the rl.agent.CustomAgent class



1. Create a subclass from the class rl.agent.CustomAgent
2. Define the appropriate Agent properties
3. Define a constructor function
4. Add a critic and an actor (if needed)
5. Define required agent methods
   - getActionImpl
   - getActionWithExplorationImpl
   - learnImpl

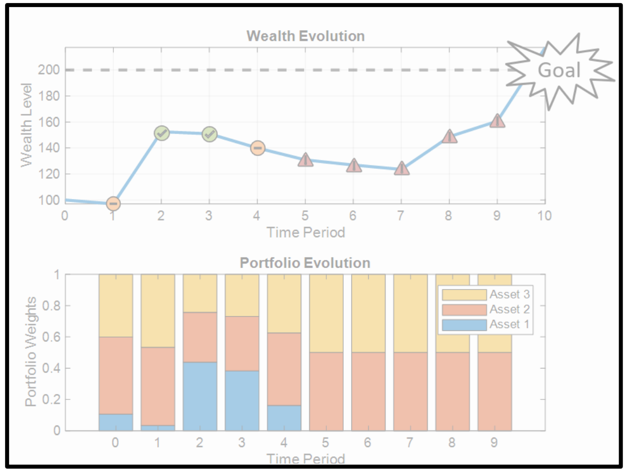
```
classdef CustomGAgent < rl.agent.CustomAgent
```

```
properties
    % G[state, action]
    G = [0, 0]
    % N[state, action]
    N = [0, 0]
    % actions
    actions
    % rho the prior policy
    rho
    % k: param for adjusting beta.
    k
    % total T steps
    T
    % epsilon for exploration
    epsilon
    % discount
    discount
end
```

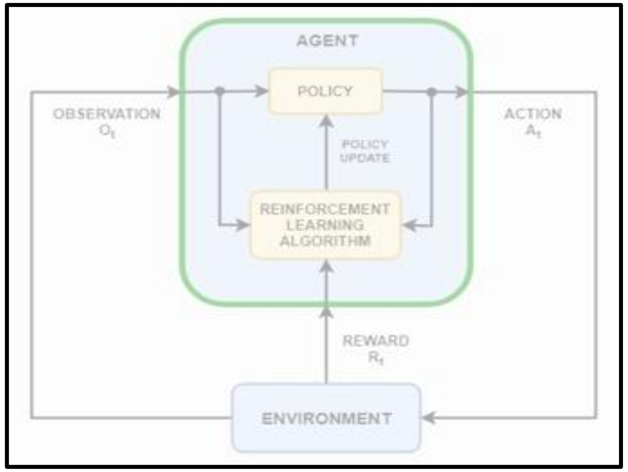
```
methods
    % Constructor
    function obj = CustomGAgent(num_state, num_action,...
            k, epsilon, obs_info, act_info, T, discount)

        % Call the abstract class constructor
        obj = obj@rl.agent.CustomAgent();

        % Set the G and N matrices
        obj.G = zeros(num_state, num_action);
        obj.N = zeros(num_state, num_action);
        obj.rho = ones(num_state, num_action) ./ num_action;
        obj.actions = 1:num_action;

        obj.k = k;
        obj.epsilon = epsilon;
        obj.T = T;
        obj.discount = discount;

        % Define the observation and action spaces
        obj.ObservationInfo_ = obs_info;
        obj.ActionInfo_ = act_info;

    end
```

Doc Example: [Create Custom Reinforcement Learning Agents]()

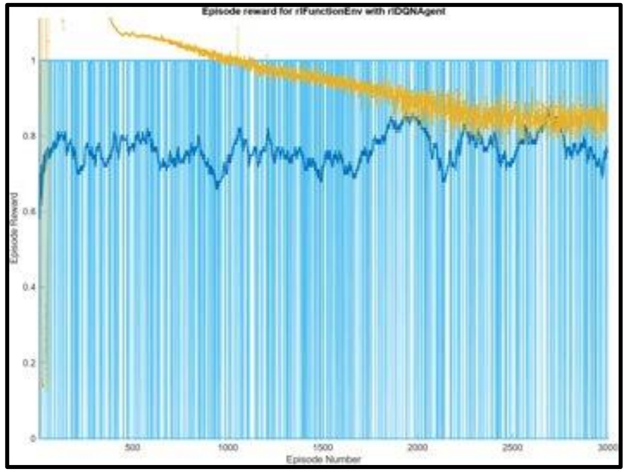
In this presentation we will see..



A Goal-based Wealth Management Problem

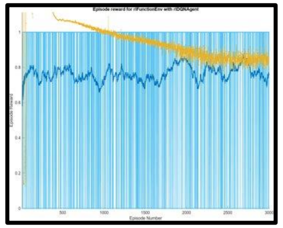


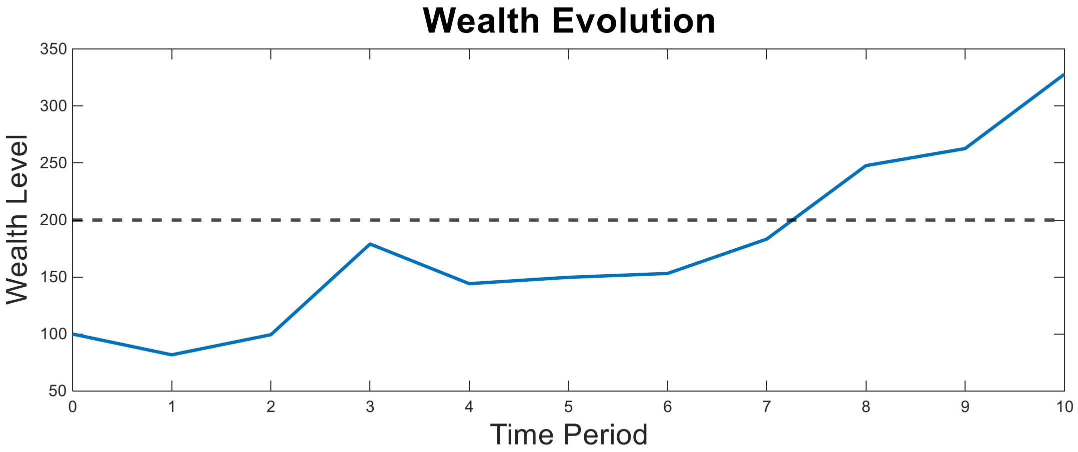
Solving the Problem using Reinforcement Learning

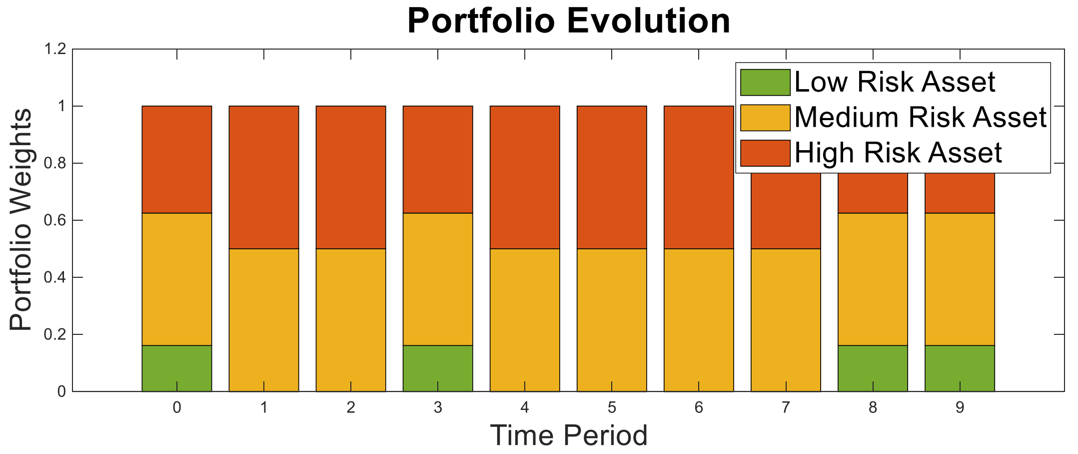


Why use Reinforcement Learning

The optimal strategy is to get more aggressive when the wealth is low, and as we are closer to the end of the investment horizon



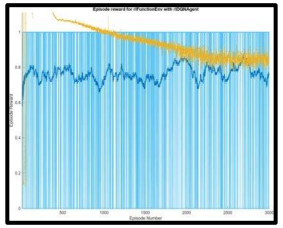


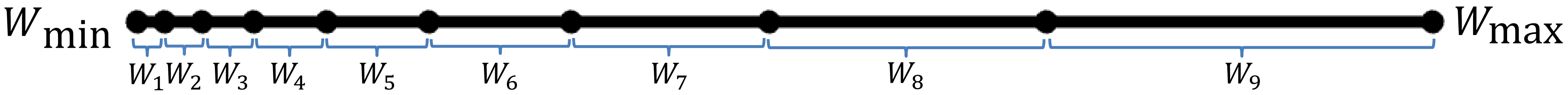


Action

WealthLevel \ TimePeriod	0	1	2	3	4	5	6	7	8	9
23.281	15	15	13	13	13	13	13	13	13	13
28.273	15	15	13	13	13	13	13	13	13	13
34.336	15	15	13	13	13	13	13	13	13	13
41.7	15	15	15	13	13	13	13	13	13	13
50.642	15	15	15	15	13	13	13	13	13	13
61.502	15	15	15	15	13	13	13	13	13	13
74.692	15	15	15	15	15	13	13	13	13	13
90.709	15	15	15	15	15	15	15	13	13	13
110.16	11	15	15	15	15	15	15	15	13	13
133.79	11	11	11	15	15	15	15	15	15	15
162.48	11	11	11	11	11	15	15	15	15	15
197.32	11	11	11	11	11	11	11	11	15	15
239.63	11	11	11	11	11	11	11	11	11	11
291.02	11	11	11	11	11	11	11	11	11	11
353.43	11	11	11	11	11	11	11	11	11	11
429.23	11	11	11	11	11	11	11	11	11	11
521.28	11	11	11	11	11	11	11	11	11	11
633.07	11	11	11	11	11	11	11	11	11	11
768.83	11	11	11	11	11	11	11	11	11	11
933.7	11	11	11	11	11	11	11	11	11	11
1133.9	11	11	11	11	11	11	11	11	11	11
1377.1	11	11	11	11	11	11	11	11	11	11
1672.4	11	11	11	11	11	11	11	11	11	11
2031.1	11	11	11	11	11	11	11	11	11	11
2466.6	11	11	11	11	11	11	11	11	11	11

Reinforcement Learning relaxes model assumptions





Finite Wealth Level States

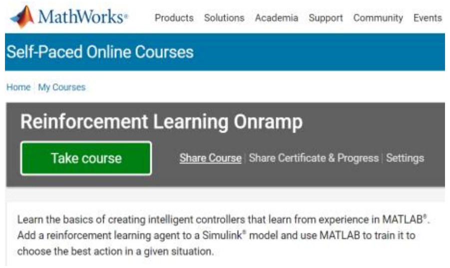
Known Transition Probabilities

$$p_{\mathcal{A}}\left(W_j \mid W_i\right) = \mathbb{P}\left(W_j(t+1) \mid W_i(t), \mathcal{A}\right)$$

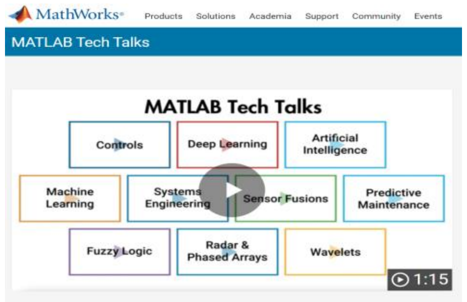
Cash flows must be known at the beginning of the investment horizon

MathWorks lowers the Reinforcement Learning barrier to entry

Learning Resources



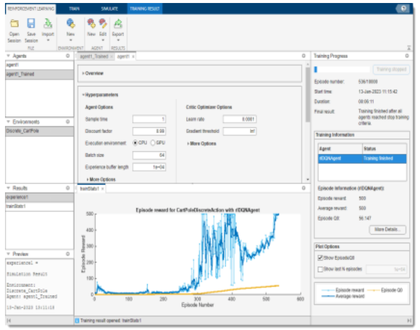
Reinforcement Learning Onramp



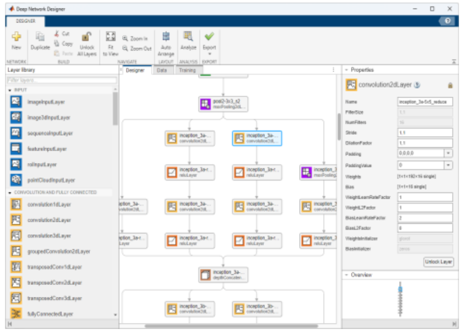
Tech Talks

RL with MATLAB eBook

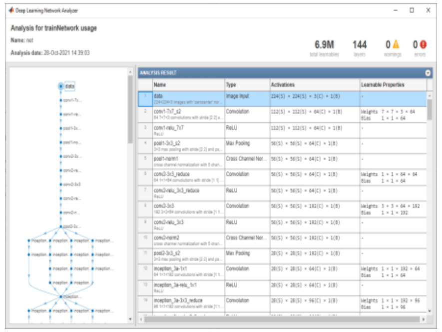
Low-code Workflows



Reinforcement Learning Designer

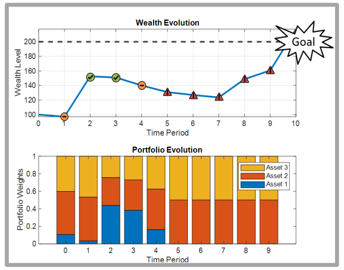


Deep Network Designer

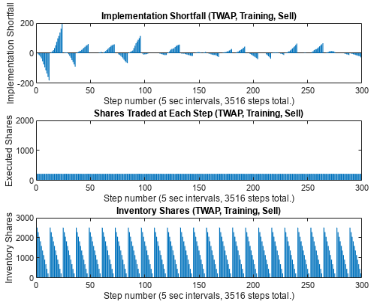


Network Analyzer App

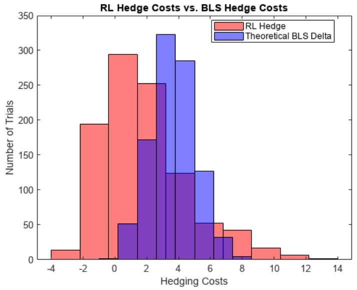
Examples to get started



Multiperiod GBWM Using Reinforcement Learning



Deep RL for Optimal Trade Execution



Hedging an Option Using Reinforcement Learning Toolbox