

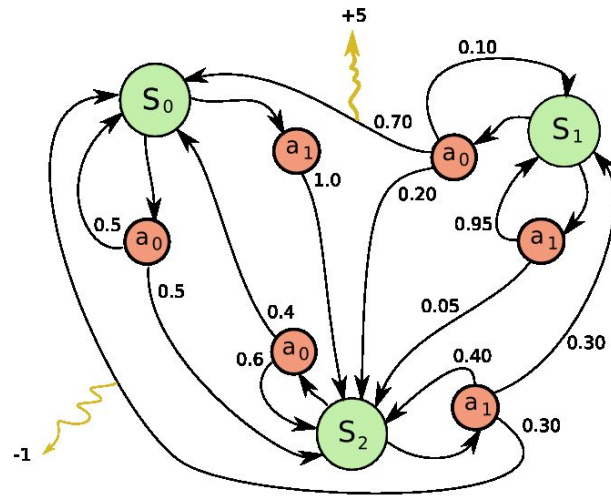
# An Actor Critic Algorithm for Mortgage Refinancing

Algorithm by V. L. Raju Chinthalapati and S. Bhatnagar

*Kaivalya Rawal and Samarth Mehrotra*

# Reinforcement Learning

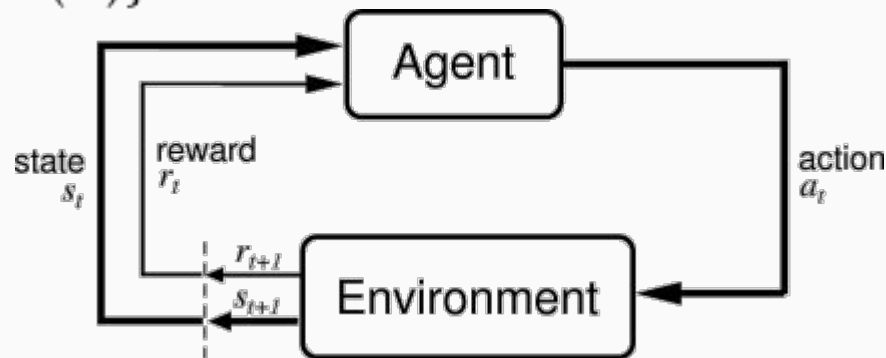
- Different from Supervised learning and Unsupervised learning
- State - Action - Reward scenario : A markov decision process
- Modern algorithms include Q-Learning, and Policy Gradients



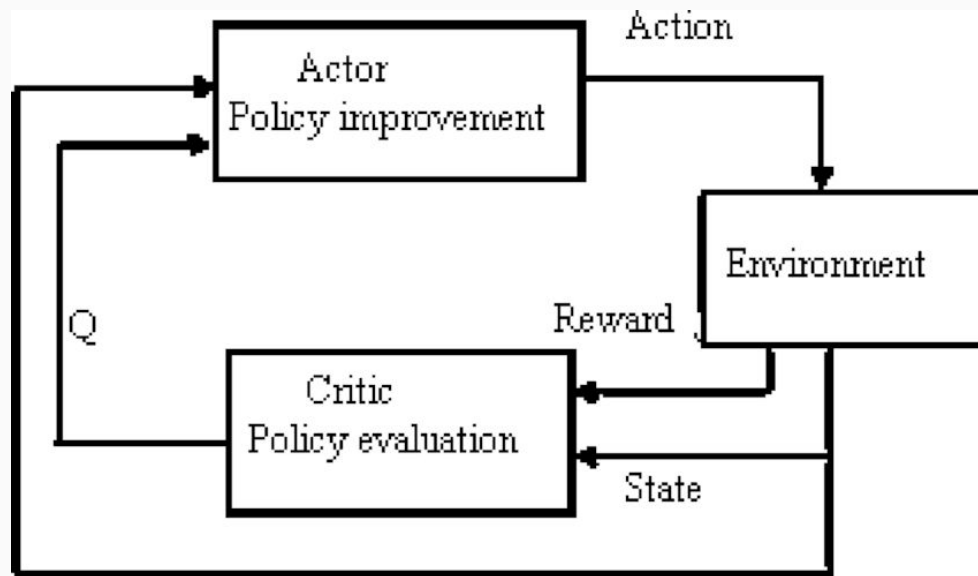
# Reinforcement Learning

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^\pi(s').$$

$$V^*(s) = \max_a \{R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s')\}.$$



# Actor Critic Algorithms



# Mortgage Refinancing Scenario

An agent has taken a loan with certain characteristics

Opportunities to refinance the loan arrive

Agent can choose whether to refinance

**Objective: Minimise Cash Flow**

# Optimal Policies

In each state, it may or may not make sense to refinance - these decisions are made according to an (initially arbitrary) **policy**

The particular means of refinancing also need to be decided

The agent needs to learn how to make these decisions by learning an **optimal policy** over time

# Difficulty of the Problem

The state space size is  $\sim 800,000$

There is a lookahead involved with each value estimate, from an action set of size 11

Problem is intractable using typical Dynamic Programming, etc

# The Algorithm

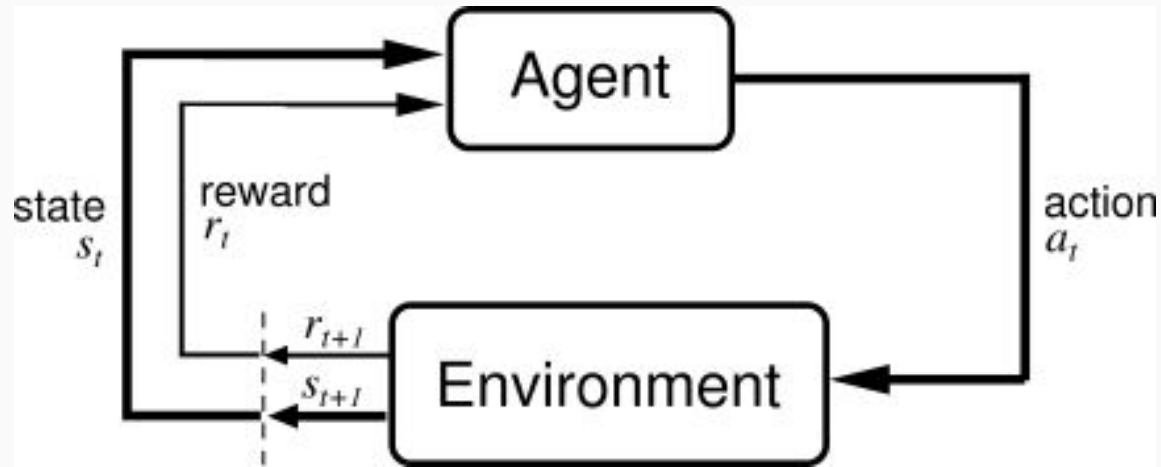
Initialise arbitrary policies

Modify them slightly (perturbations) based upon random updates using a hadamard matrix

Use the wealth function to decide the direction of the updates, ultimately converging to optimal policies



# Implementation Overview



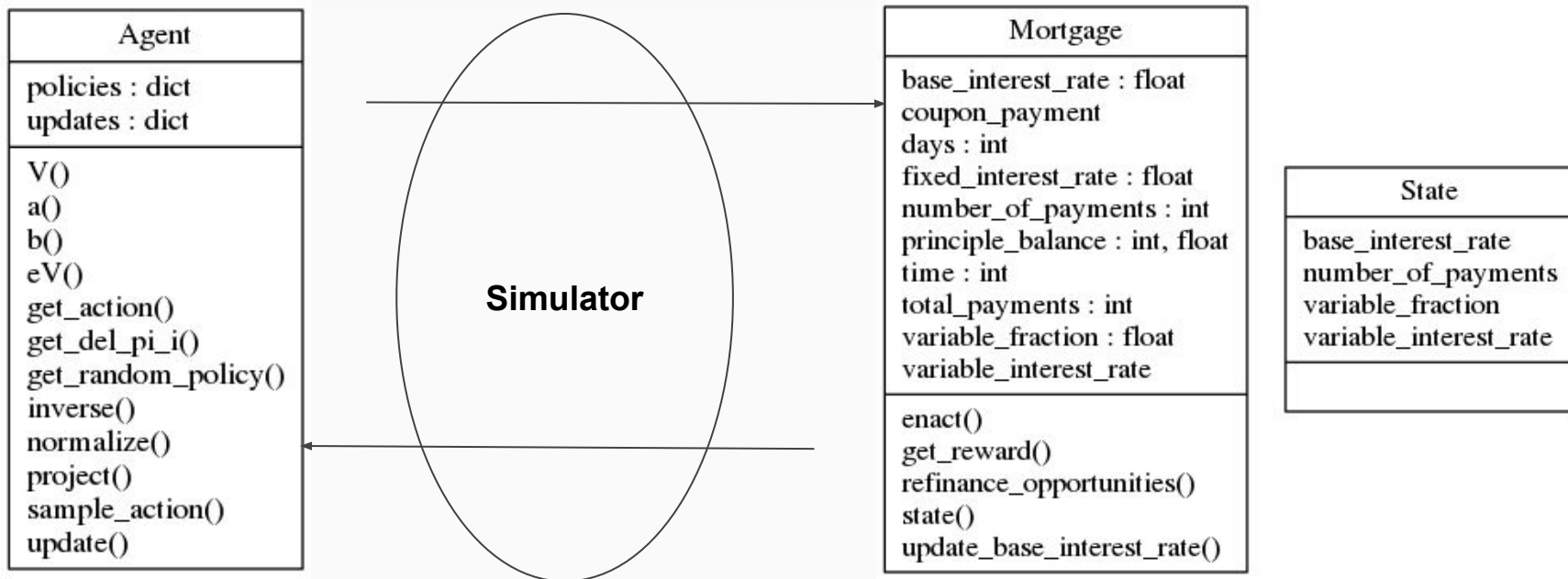
# Implementation Overview

Agent
policies : dict updates : dict
V() a() b() eV() get_action() get_del_pi_i() get_random_policy() inverse() normalize() project() sample_action() update()

Mortgage
base_interest_rate : float coupon_payment days : int fixed_interest_rate : float number_of_payments : int principle_balance : int, float time : int total_payments : int variable_fraction : float variable_interest_rate
enact() get_reward() refinance_opportunities() state() update_base_interest_rate()

State
base_interest_rate number_of_payments variable_fraction variable_interest_rate

# Implementation Overview



# Demo

Code Demonstration

Expected runtime (1000 updates per state) until convergence  $> 20 \times 11$  days

# Future Work

Comparison of different algorithms by using drop-in replacements for *agent.py*.

Comparison of same algorithm in different settings by using drop in replacements for environment like the OpenAI Gym.

Thank You