COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

GOAL-BASED WEALTH MANAGEMENT WITH REINFORCEMENT LEARNING

ML3 –BOWEN FANG, BOTAO ZHANG, CHONGYI CHIE, AND YICHEN YAO

MEET THE TEAM

Chongyi Chie
MSOR

Bowen Fang
MSOR

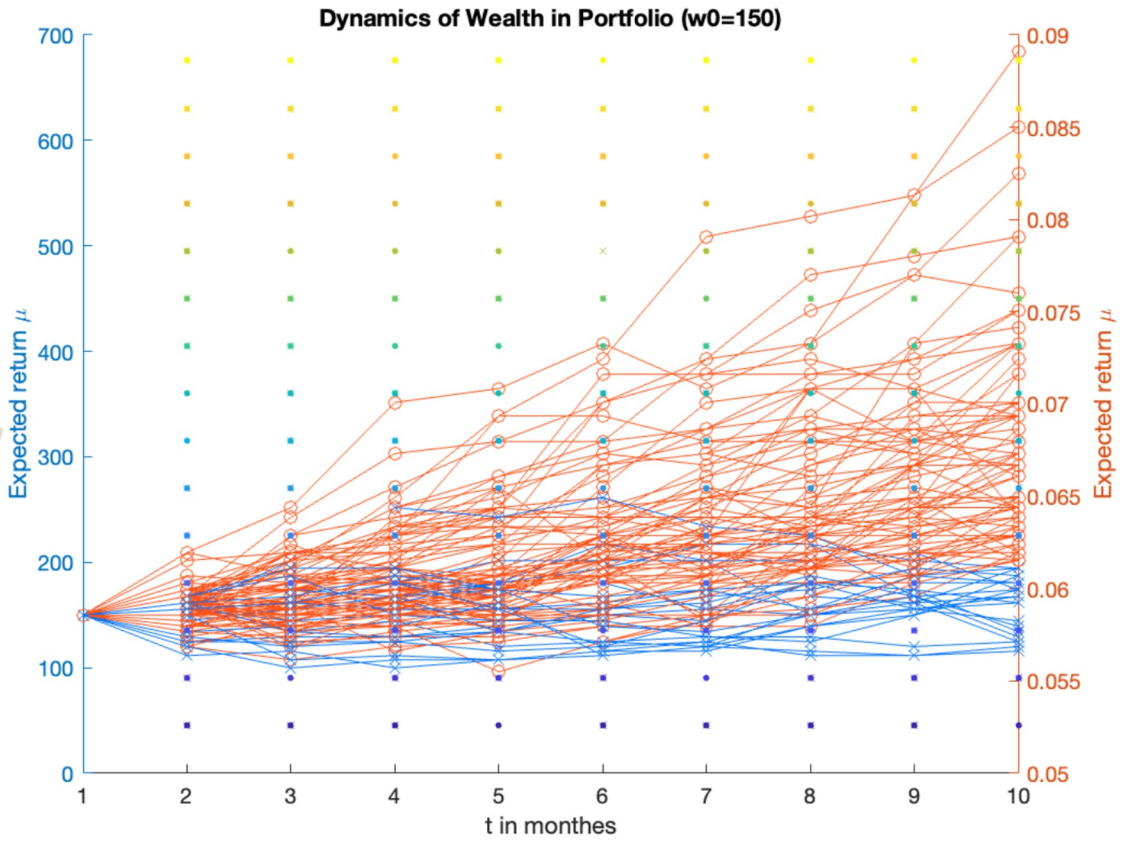
Botao Zhang
MSOR

Yichen Yao
MSOR

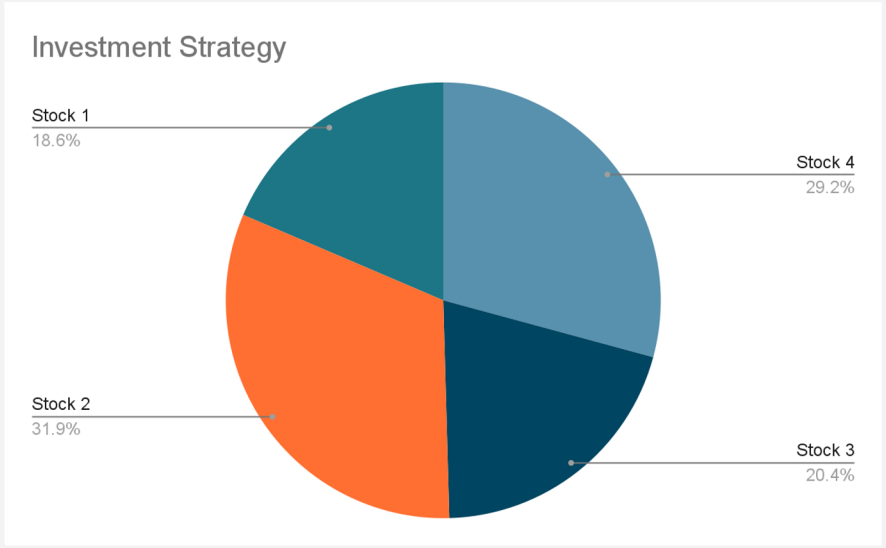
AGENDA

- Application of the Proposed RL Algorithm
- Why Choose RL over Dynamic Programming?
- Motivation
- Problem Statement
- Key Idea
  - Approximation:
    - Agents
    - Environments
- System Overview

Application of the Proposed RL Algorithm



Application of the Proposed RL Algorithm



Defect of the Still Modern Portfolio Theory

This does not work:

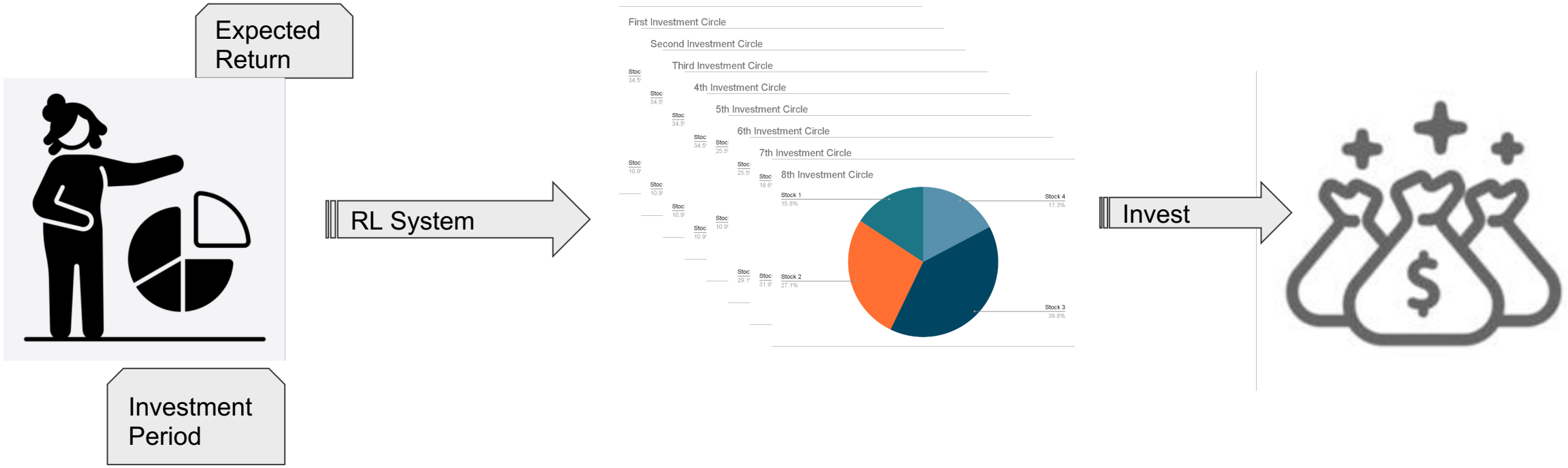
- 1) stock price changes are complex and often impossible to precisely estimate; and
- 2) choices should be made online with noisy inputs and perform effectively under various situations.

Requirement for the System

To maximize the Hit Ratio to the expected cumulative return during an investment cycle:

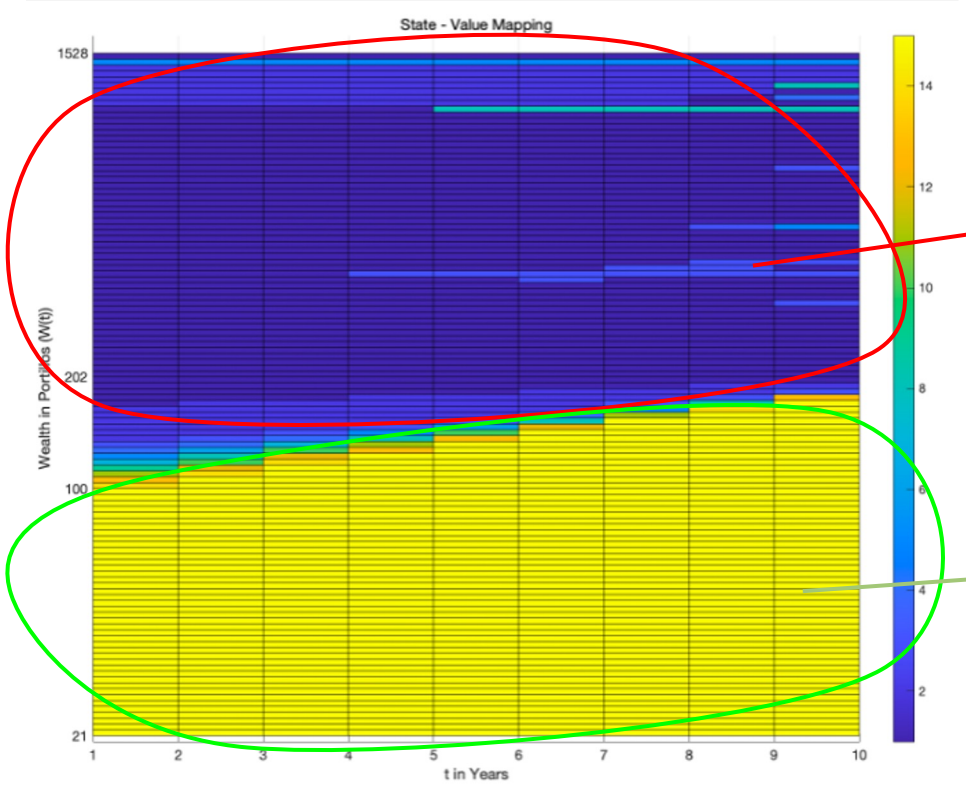
- Reallocate the funds we placed into portoflo in order to dynamically balance risk and return
- Control the risk based on the horizon, i.e. the amount of money we have and the time it will take to reach the goal.

What Investors Should Perform



Why Choose RL over Dynamic Programming?

- Transition probability of securities' price is unknown in real-world. To estimate the transition probability is a harder task. RL could learn without knowledge of trans. prob.
- However, DP provides theoretical optimal solution. We regard DP as a benchmark to RL solutions.



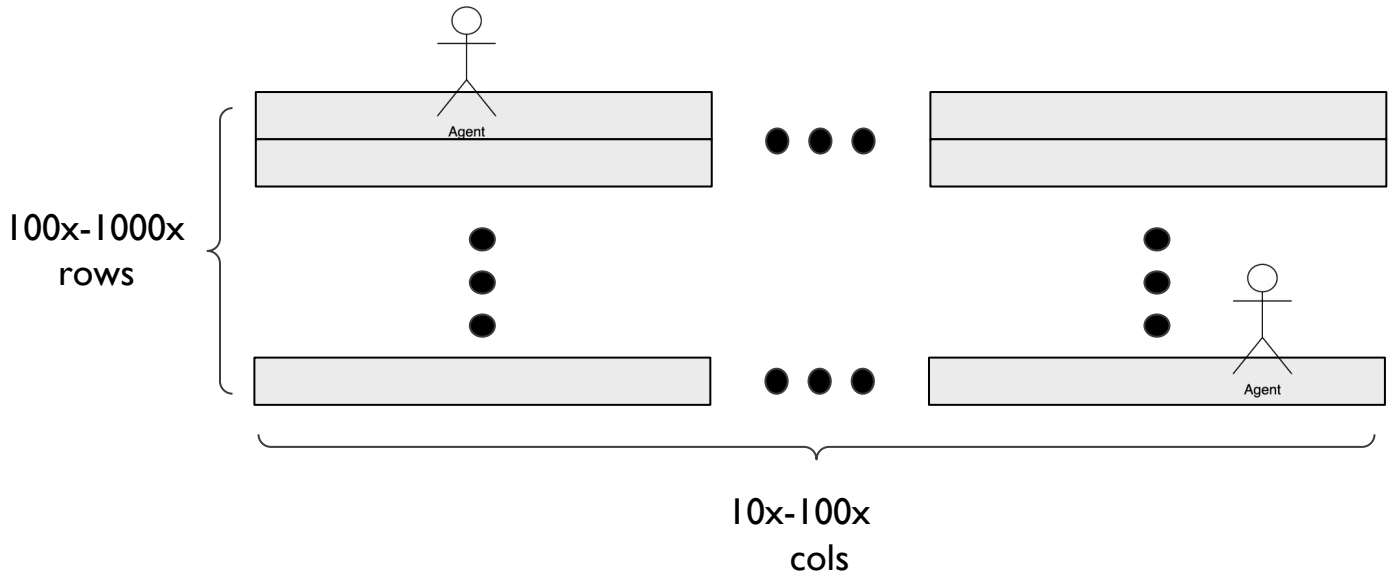
Defensive:
Lower expected return, lower volatility

The optimal success probability is ~66%

Aggressive:
Higher expected return, higher volatility

Motivation

- Developing a RL algorithm which solves GBWM problem with limited observation over the large entire state space is "mission-impossible".
- We developed a RL algorithm that approximates the optimal solution.



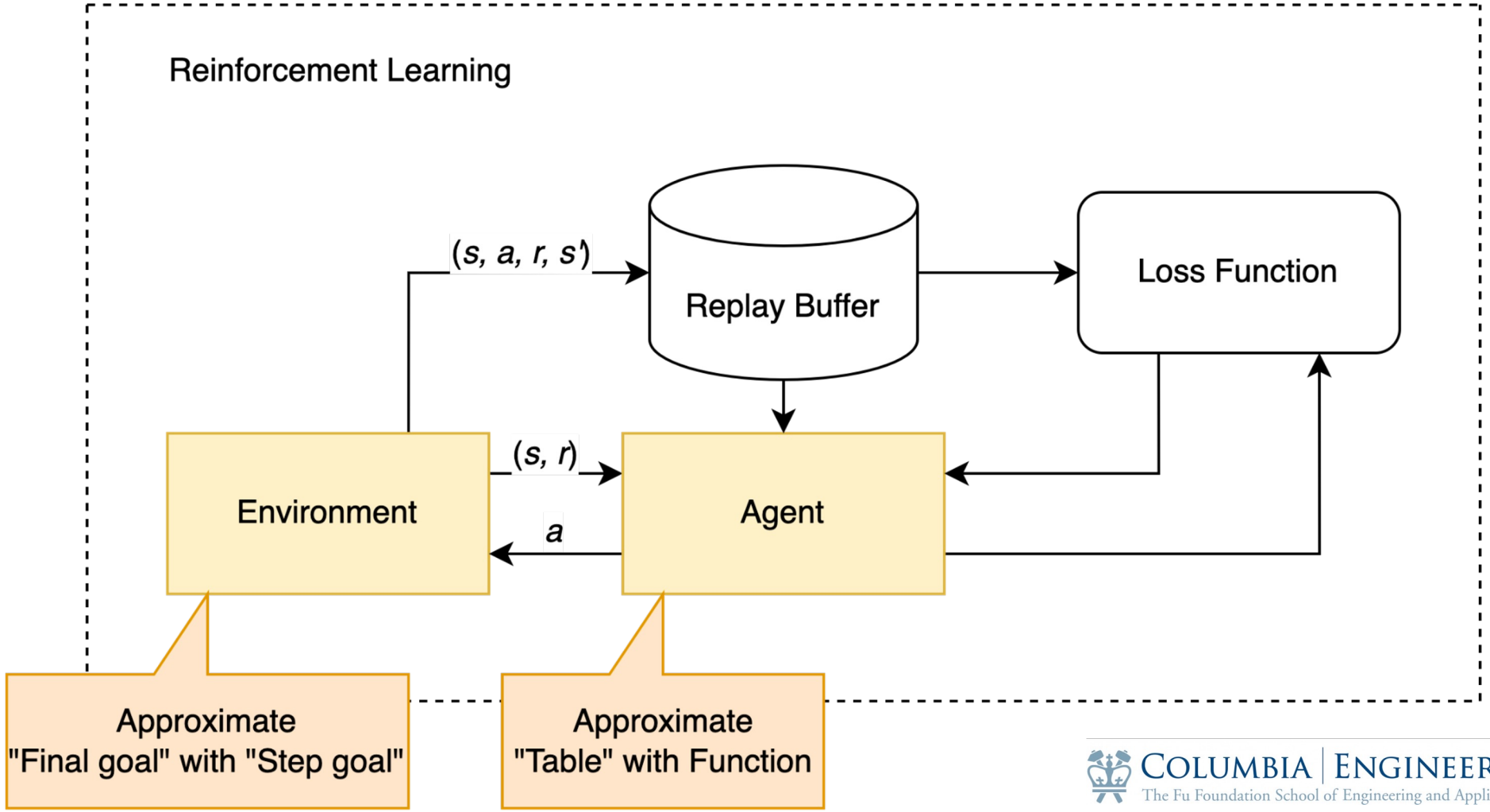
GBWM state space

Agents could visit 10% of rows (others are extremely rare, but important!)

Since the learning of a proper strategy over all possible states is difficult in tabular RL, we have turned to approximation

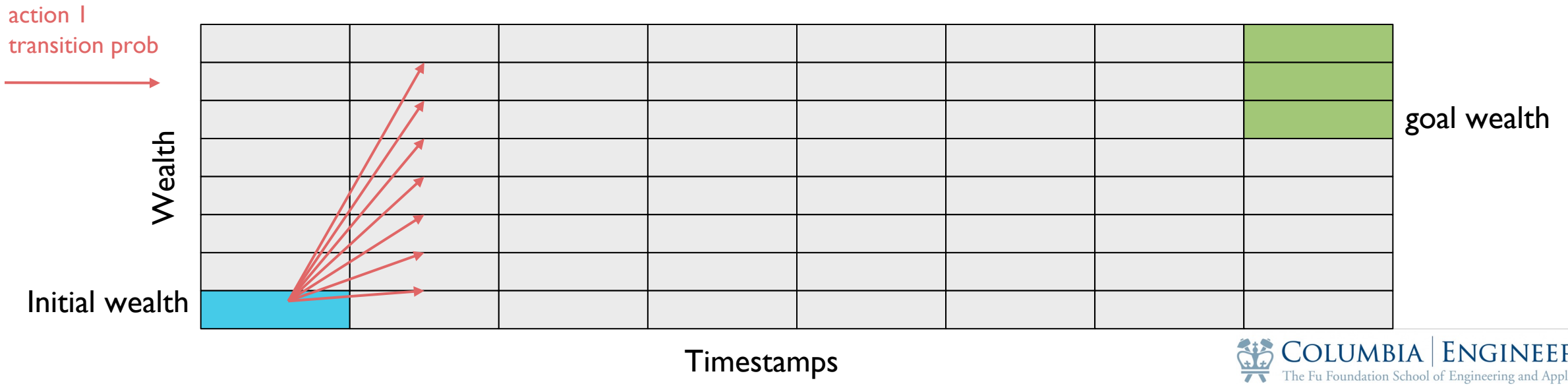
Key Idea

- **Approximation guides RL agent acts properly in real-world environment**
- We approximate Table with Deep Learning network
- We approximate Sparse reward with stepwise reward



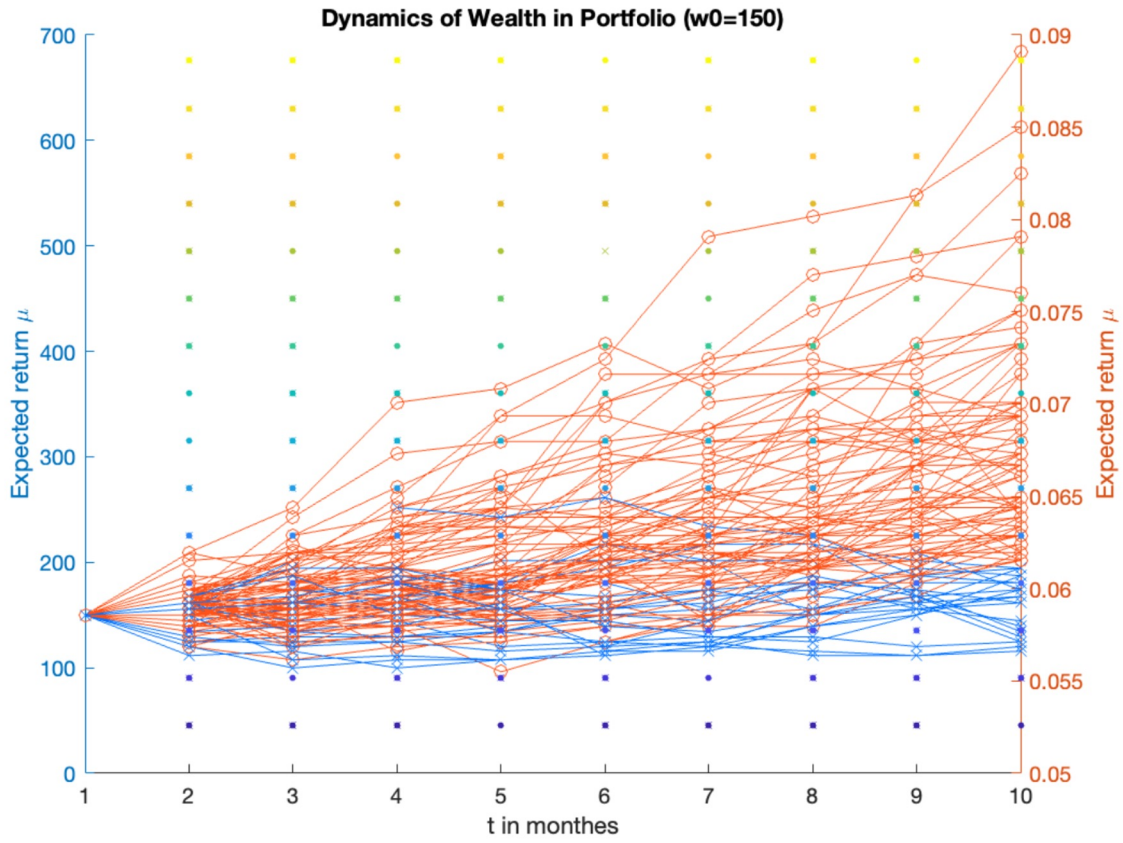
Basic RL Problem Statement

- The agent chooses a portfolio from efficient frontier at each rebalancing period, to maximize the probability that the goal wealth region is reached.
  - State: 1x2 vector [wealth, timestamp]
  - Action: Int [1, 15]
  - Transition: GBM
  - Reward: 1 if timestamp=T and wealth >= goal, 0 otherwise



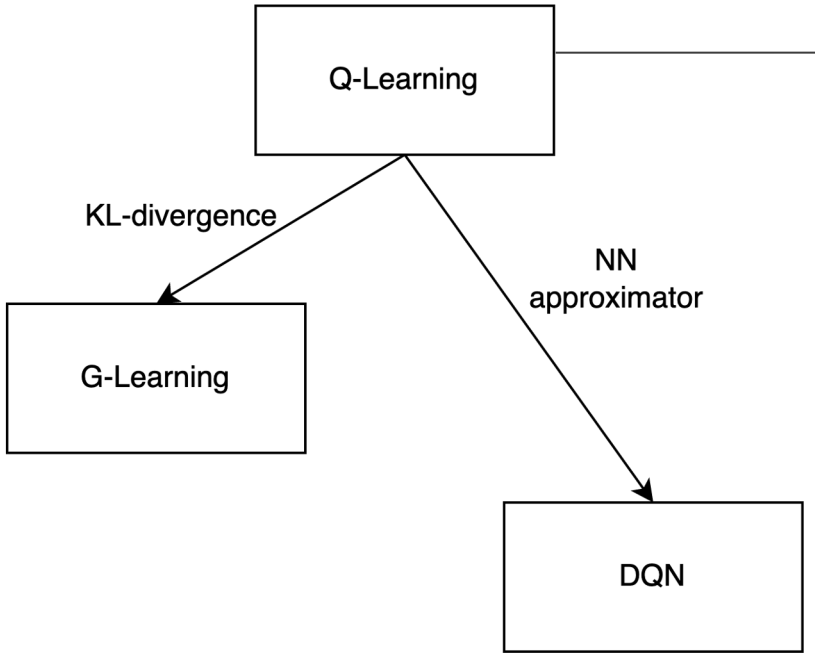
Basic RL Problem Statement

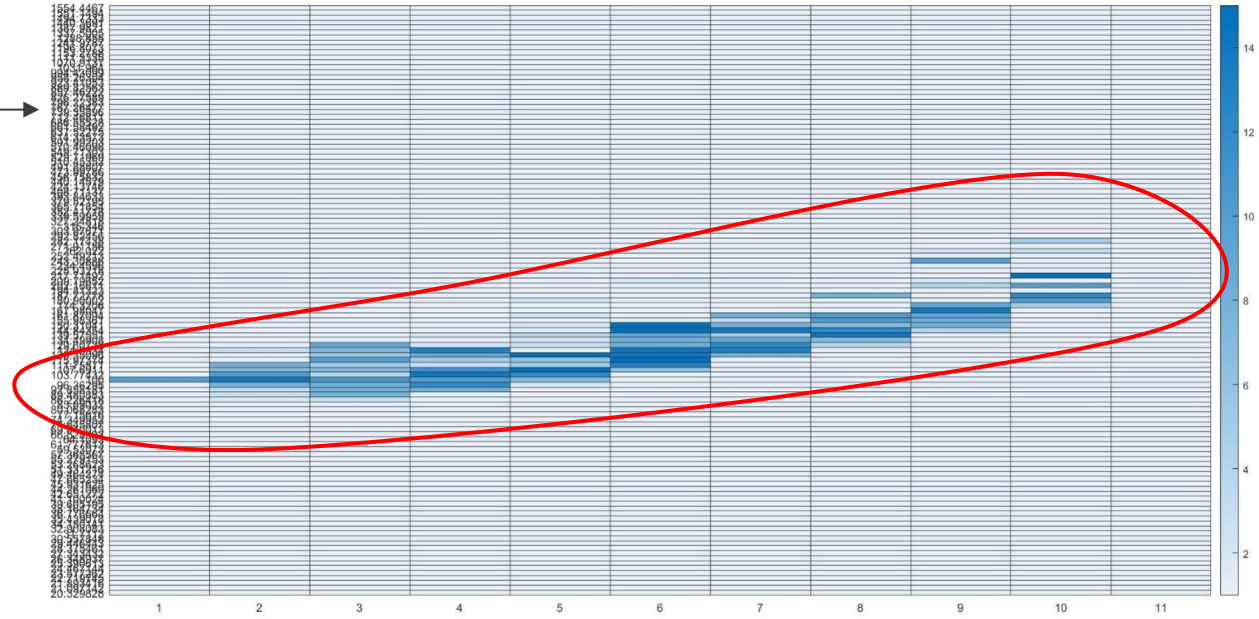
- The agent chooses a portfolio from efficient frontier a
  that the goal wealth region is reached.
  - State: 1x2 vector [wealth, timestamp]
  - Action: Int [1, 15]
  - Transition: GBM
  - Reward: 1 if timestamp=T and wealth >= goal, 0



Approximation for Agent

- Motivation: Agent has little probability to visit then learn policy for all states (Fig).
  We can either force the agent to explore more, or infer the action for unvisited states.
  - We add KL-divergence against uniform distribution to force exploration (G-Learning)
  - We use Neural Network (NN) to approximate the Table. (DQN)

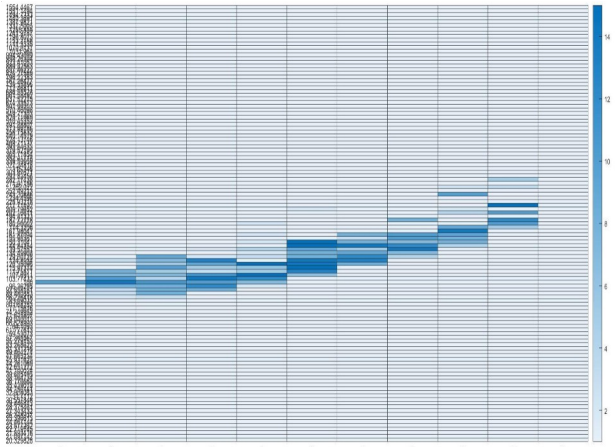




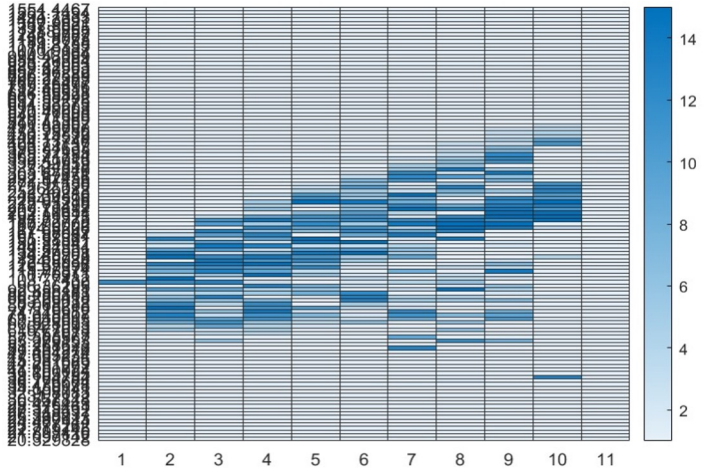
Comparison of Different Agents

- Comparison of action grid and success probability

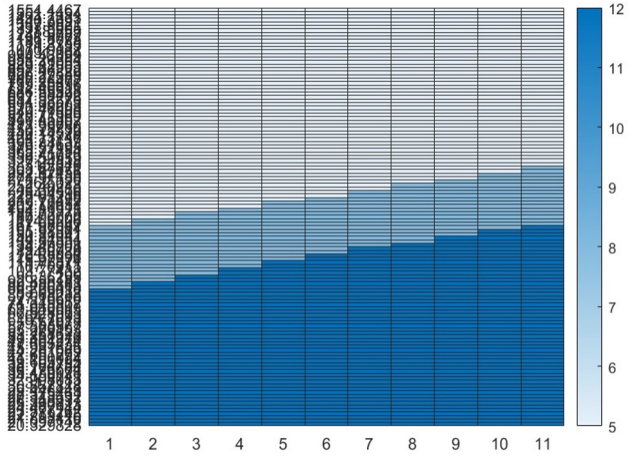
		Agent	
Metrics	**Q-Learning**	**G-Learning**	**DQN**
Success probability	40.50%	33%	**48%**



Q-Learning



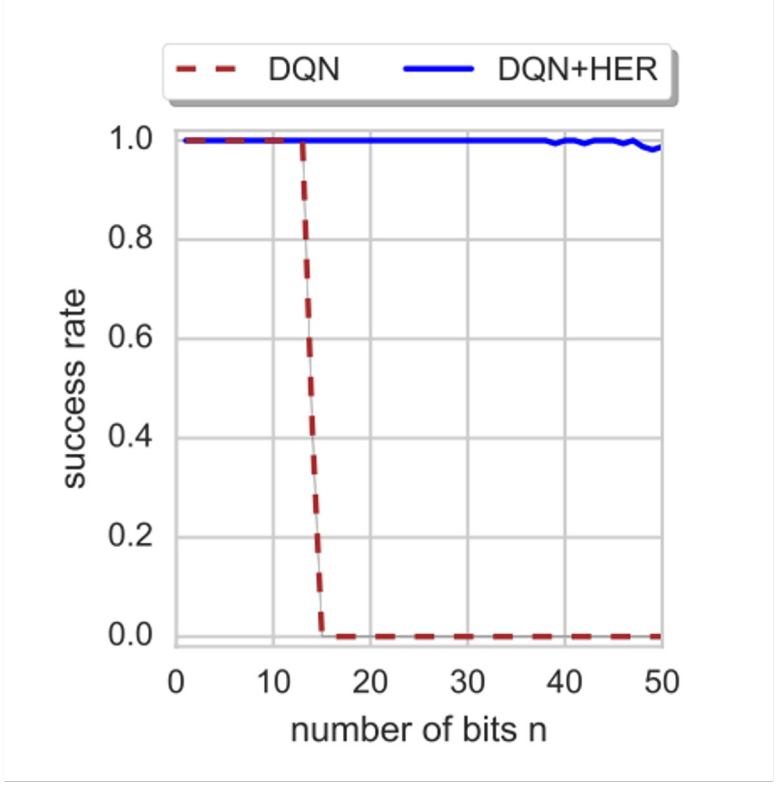
G-Learning



DQN

Approximation for Environment

- Motivation:
  Sparse reward fails to guide RL agent with large action space and environment with long episode. Therefore it is important to approximate the sparse reward with more informative reward.



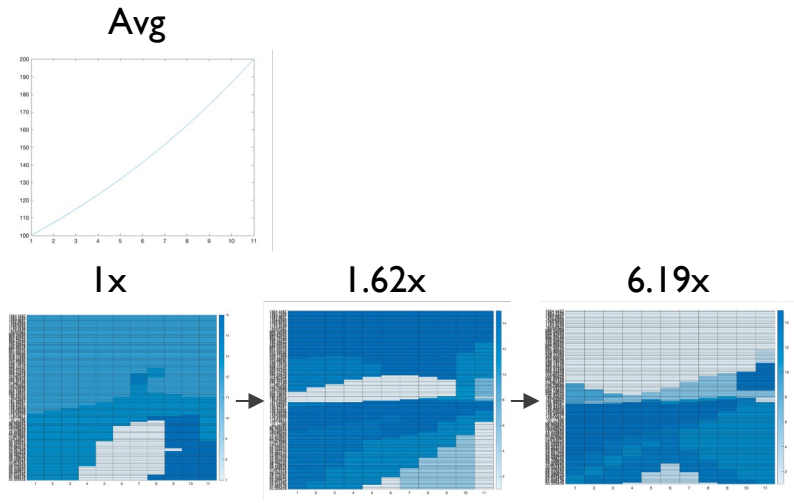
For example, consider a bit-flipping game:

The action is to flip each bit or not. Reward is 1 if all bits are 1, otherwise the reward is 0.

We observe that agent fails to learn when N>20. Because it is almost impossible (2^N) to receive positive reward with random exploration. **The same applies for our case.**

Comparison of Different Environments

- Motivation and definition of three modified environments:
  - Line: Gives 1 reward if above a predefined curve at each step.
  - Scale: Gives scaled reward if goal is fulfilled at each step, the latter the larger.
  - Multi factors
- Comparison Table

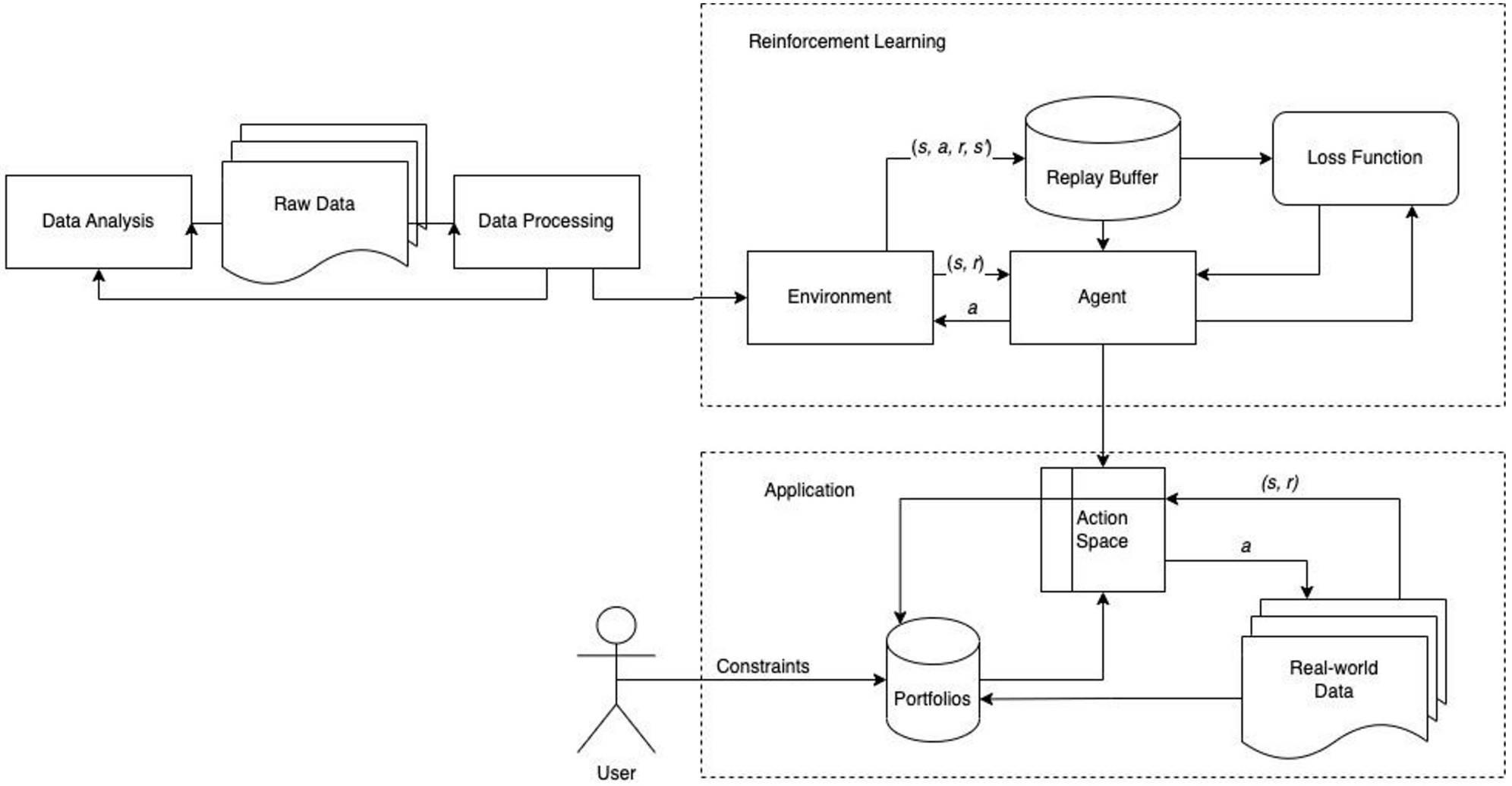


	Train Environment		
DQN	Sparse	Line	Multifactor
Hidden units	**Success probability on Sparse Env.**		
8	38.30%	49.60%	48.20%
16	48.00%	**61.00%**	47.40%
64	57.00%	53.00%	50.40%

Performance of the System

Only -0.07 away from theoretically highest success rate!!!

Overview of the System



THANK YOU

Quick Review of Paper and DP model based on it

efficient frontier:

$$\sigma = \sqrt{a\mu^2 + b\mu + c}.$$

$$\begin{aligned} a &= \mathbf{h}^\top \boldsymbol{\Sigma} \mathbf{h} \\ b &= 2\mathbf{g}^\top \boldsymbol{\Sigma} \mathbf{h} \\ c &= \mathbf{g}^\top \boldsymbol{\Sigma} \mathbf{g}, \end{aligned} \qquad \begin{aligned} \mathbf{g} &= \frac{l\boldsymbol{\Sigma}^{-1}\mathbf{o} - k\boldsymbol{\Sigma}^{-1}\mathbf{m}}{lp - k^2} \\ \mathbf{h} &= \frac{p\boldsymbol{\Sigma}^{-1}\mathbf{m} - k\boldsymbol{\Sigma}^{-1}\mathbf{o}}{lp - k^2}, \end{aligned} \qquad \begin{aligned} k &= \mathbf{m}^\top \boldsymbol{\Sigma}^{-1} \mathbf{o} \\ l &= \mathbf{m}^\top \boldsymbol{\Sigma}^{-1} \mathbf{m} \\ p &= \mathbf{o}^\top \boldsymbol{\Sigma}^{-1} \mathbf{o}. \end{aligned}$$

smallest and largest wealth grid point

$$\hat{W}_{\min} = \min_{\tau \in \{0,1,2,\ldots,T\}} \left[W(0) e^{\left(\mu_{\min} - \frac{\sigma_{\max}^2}{2}\right)\tau - 3\sigma_{\max}\sqrt{\tau}} + \sum_{t=0}^{\tau} C(t) e^{\left(\mu_{\min} - \frac{\sigma_{\max}^2}{2}\right)(\tau - t) - 3\sigma_{\max}\sqrt{\tau - t}} \right]$$

$$\hat{W}_{\max} = \max_{\tau \in \{0,1,2,\ldots,T\}} \left[W(0) e^{\left(\mu_{\max} - \frac{\sigma_{\max}^2}{2}\right)\tau + 3\sigma_{\max}\sqrt{\tau}} + \sum_{t=0}^{\tau} C(t) e^{\left(\mu_{\max} - \frac{\sigma_{\max}^2}{2}\right)(\tau - t) + 3\sigma_{\max}\sqrt{\tau - t}} \right]$$

probability density function:

$$\tilde{p}(W_j(t+1)|W_i(t), \mu) = \phi\left(\frac{1}{\sigma}\left(\ln\left(\frac{W_j}{W_i + C(t)}\right) - \left(\mu - \frac{\sigma^2}{2}\right)\right)\right)$$

transition probabilities:

$$p(W_j(t+1)|W_i(t), \mu) = \frac{\tilde{p}(W_j(t+1)|W_i(t), \mu)}{\sum_{k=0}^{i_{\max}} \tilde{p}(W_k(t+1)|W_i(t), \mu)}$$

$$V(W_i(t)) = \max_{\mu \in [\mu_{\min}, \mu_{\max}]} \left[\sum_{j=0}^{i_{\max}} V(W_j(t+1)) p(W_j(t+1)|W_i(t), \mu) \right]$$

probability distribution:

$$p(W_j(t+1)) = \sum_{i=0}^{i_{\max}} p(W_j(t+1)|W_i(t), \mu_{i,t}) \cdot p(W_i(t))$$

Quick Review of Paper and DP model based on it

Inputs: Initial wealth W0, Target wealth G, Time horizon T
Goal: max Pr[W(T) ≥ G]
Algorithm:

- Step 1: calculate efficient frontier for stock portfolios
- Step 2: set state space grid points
  - smallest wealth grid point = smallest possible wealth, largest wealth grid point = largest possible wealth
- Step 3: dynamic programming for optimizing the chance of obtaining the investor's goal
  - get transition probabilities from probability density function
  - determine Bellman recursion equation to know V (W(0)), optimal probability
- Step 4: determine the probability distribution for the investor's wealth

Something need improve

- The original model has a limited number of observable states
  - But wealth is a continuous variable
- The number of portfolio is fixed and determined before action
- We need to know transition probabilities function,
  - Requires a lot of mathematical calculations
  - RL can learn in the case of transitions unknown

★ Our experiments still use DP's transition to facilitate comparison