

# A Deep Reinforcement Learning Framework for Financial Portfolio Management

Deutsche Bank AI Research

---

Stewart Young

Graduate Technology Analyst, London

# Table of Contents

1. Introduction to Portfolio Management
2. Introduction to Reinforcement Learning
3. Project Overview
4. Results
5. Q&A

# Introduction to Portfolio Management

---

# Portfolio Management

Portfolio Management is the selection and overseeing of a group of investments that meet the financial objectives of the investor e.g. Hedge Funds, Pension Funds, Governments, Institutions etc.

Investors typically seek to maximise returns for a given level of risk, also known as the Markowitz efficient frontier. [1]

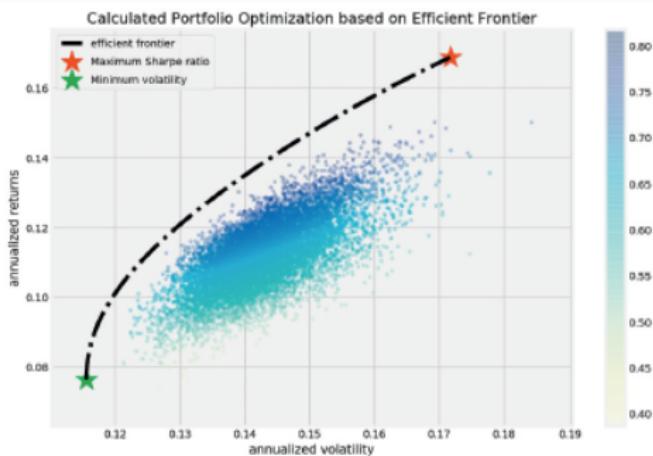


Figure 1: Demonstration of Efficient Frontier [2]

# Advances in Portfolio Management

## Baseline Investment priorities

- Strong return on investments
- Optimised for risk appetite
- Tailored to hedge downside risk

## Additional modern day investment priorities

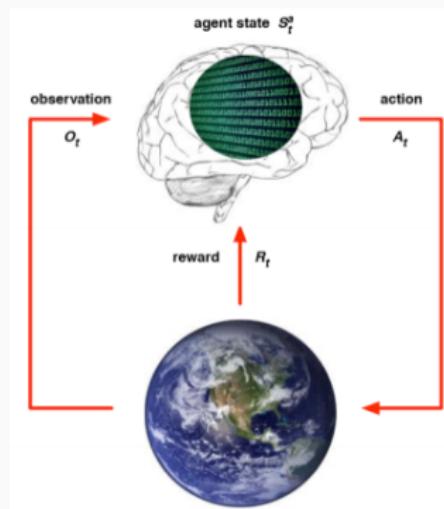
- Environmental concerns of business e.g. climate change, sustainability
- Social effects of business e.g. diversity, human rights, consumer protection, animal welfare
- Corporate Governance e.g. management structure, employee compensation, executive compensation.
- AI can unlock additional returns and remove emotional human decision making

This has lead to the rise of ESG (Environmental, Social, & Corporate Governance) data, and AI for electronic trading.

# Introduction to Reinforcement Learning

---

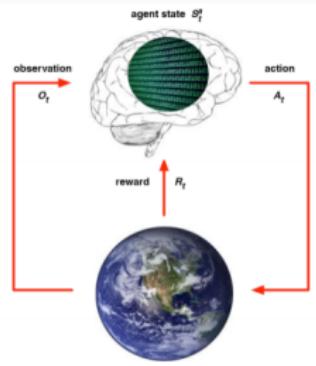
# Reinforcement Learning



**Figure 2:** Reinforcement Learning Decision Space.  
Source-David Silver (UCL/DeepMind) [3]

# Reinforcement Learning

Formally,

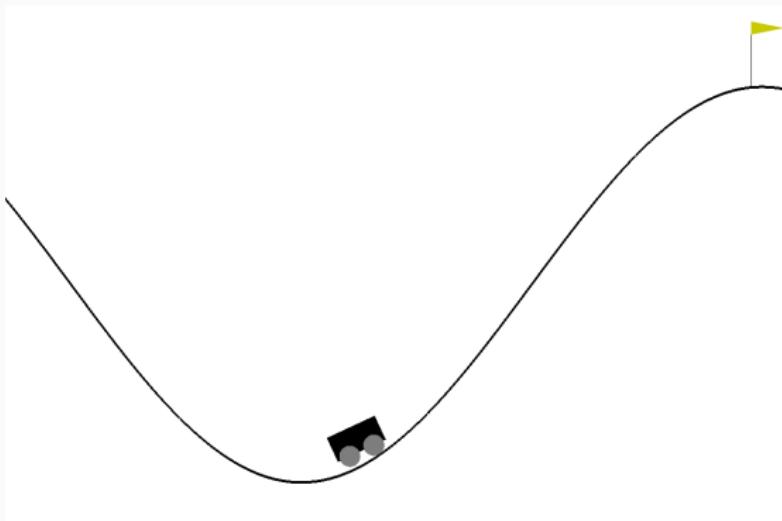


- $O_t$ : Observation
- $A_t$ : Action
- $R_t$ : Reward
- $H_t$ : History
- $S_t$ : State

$$H_t = O_t, R_t, A_{t-1}, O_{t-1}, R_{t-1}, \dots, O_1, R_1, A_1$$

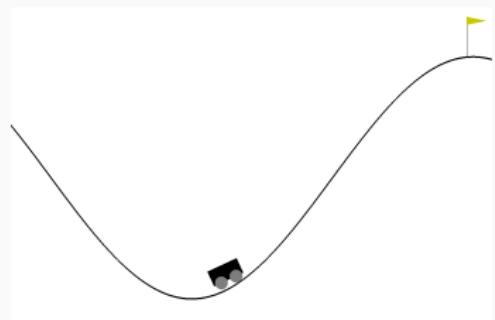
$$S_t = f(H_t) = f(O_t, R_t, A_{t-1}, \dots, O_1, R_1, A_1)$$

# Reinforcement Learning Example



**Figure 3:** Classic example task for reinforcement learning,  
Mountain Car by OpenAI

# Reinforcement Learning Example



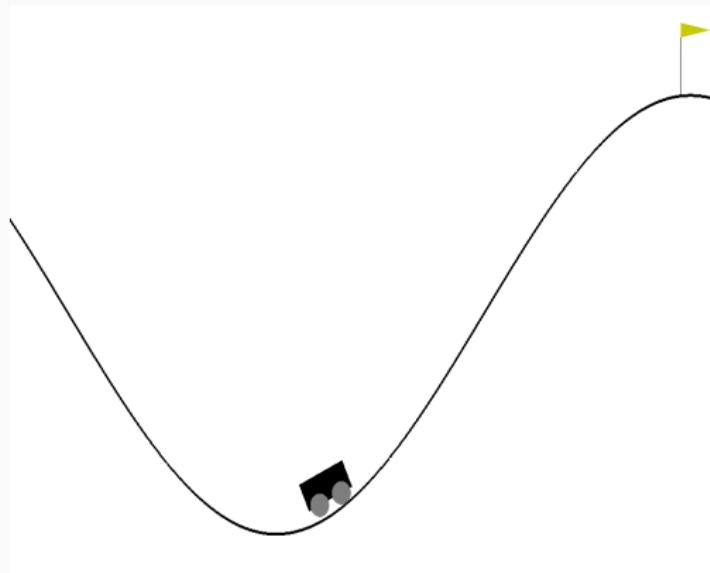
- $S_t$ : ( $Position_t, Velocity_t$ )
- $A_t$ :  $(-1, 0, 1) = (left, none, right)$
- $R_t$ : -1 if not in goal state, else 0.

Can we teach the car to go up the hill?

A logical (rule based) agent may just decide to always go right, good move?

# Reinforcement Learning Demonstration

Always go right?



# Reinforcement Learning Demonstration

To teach the agent, introduce:

- $Q(S_t, A_t)$ : Action Value Function

We use the Bellman Equation to update Q values:

$$Q^{new}(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \underbrace{\alpha}_{\text{learning rate}} \cdot (r_t + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{Q value of optimal action in next state}})$$

# Reinforcement Learning Demonstration

Reinforcement Learning pseudo algorithm:

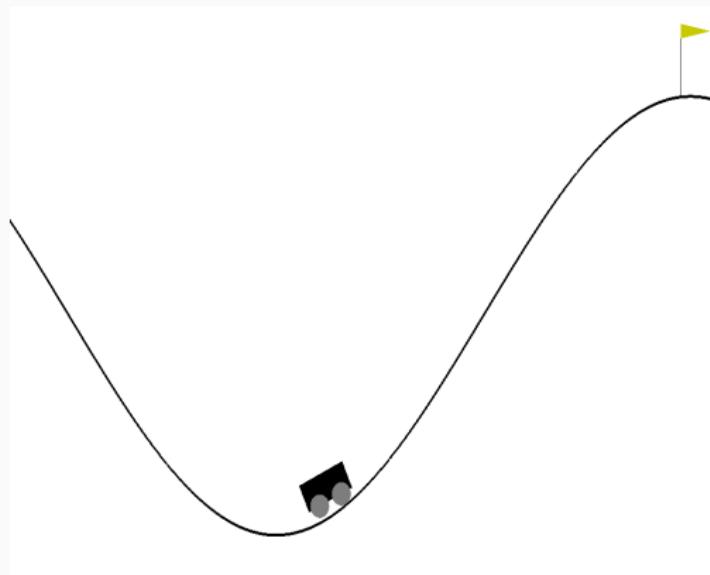
1. Discretise state space. Initialise parameters  $\alpha, \gamma$
2. Assign Q values uniformly random between 0-1
3. For each episode until goal
  - 3.1 Observe state  $s_t$ , make action  $a_t$
  - 3.2 Observe reward  $r_t$ , new state  $s_{t+1}$ .
  - 3.1 If new state=goal, finished
  - 3.2 Else,
    - Estimate action with highest Q value in new state
    - Update Q table via Bellman eqn.

Which creates a Q table, something like this:

		Velocity		
		-0.07	...	0.07
Position	-1.2	(0,0.1,1.33)	...	(0.4,0.7,3.6)
	...	...	...	...
	0.5	(10,10,10)	...	(10,10,10)

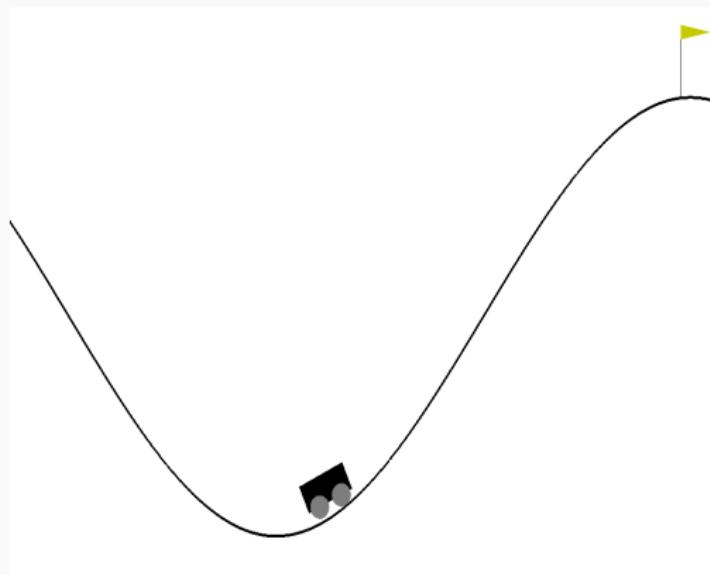
# Reinforcement Learning Demonstration

Bellman Q Learning Episode 0



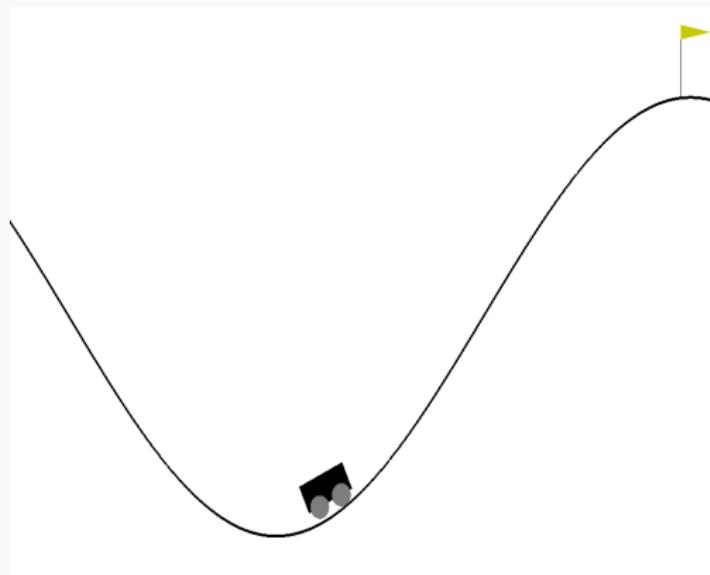
# Reinforcement Learning Demonstration

Bellman Q Learning Episode 4000



# Reinforcement Learning Demonstration

Bellman Q Learning Episode 12000



# Reinforcement Learning Demonstration

Not bad! But some problems:

- A lot of iterations (12000), around 3 minutes to train. Brute force computation.
- Discrete state space - Only have a "lookup" table for Q values
  - What happens if the hill becomes slightly longer? We've no space in the Q value for the unseen points.
- Q table could be terabytes/petabytes of memory for even somewhat difficult games e.g. Atari Pong.

# Reinforcement Learning Demonstration

Solution = Deep Q Learning!

- Allows approximation of much larger state spaces than Mountain Climb.
- Helps learn much more complex decision environments e.g. Go.

Method

1. Use a neural network to get Q values based on state,  $Q = f(S_t)$ .  
Expanded to continuous state space, so no more table lookup.
2. Update weights in neural network as we learn more about the valuable actions in different states.
3. For each time step, and each state, take actions that maximise Q value.

# Reinforcement Learning Example

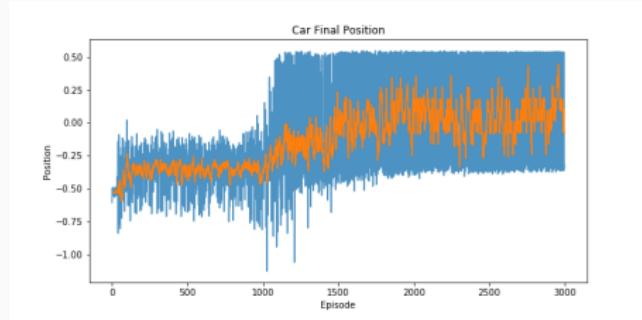


Figure 4: Performance of Deep Q Learning Agent on Mountain Climb

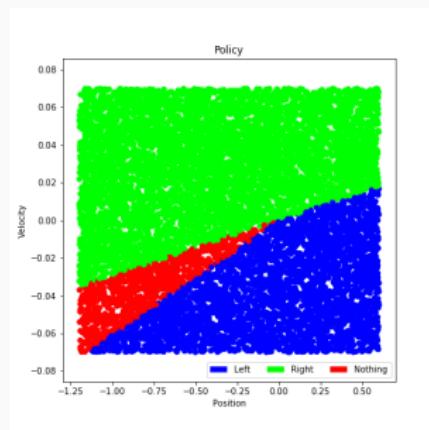


Figure 5: Learned policy function (action function) on Mountain Climb

## Project Overview

---

# Paper Overview

## A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem [4]

- Proposed framework based on Ensemble of Identical Independent Evaluators (EIIE) topology.
- EIIE inspects the history of each asset and evaluates its potential growth for the future. The output becomes the new portfolio weights for the coming trading period.
- Uses experience replay to train EIIE to output a set of optimal portfolio weights. Achieved exceptional ten fold returns in 3 different cryptocurrency portfolio testing environments.

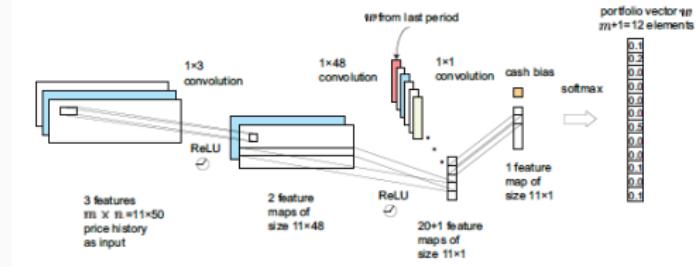


Figure 6: CNN Implementation of EIIE

# The Data

---

- Time frame: 28/09/2004 - 28/09/2018
- Number of trading days: 3654
- Frequency: Daily
- Indices: CAC40, DAX, FTSE100, Nikkei 225, S&P500, TSX
- Financial Metrics: Open, High, Low, Close

# The Algorithm

1. Portfolio Environment: Easily take batches of data to feed into neural network. Calculate rewards based on returns between time steps.
2. Deep Deterministic Policy Gradient [5]:

Algorithm 1 DDPG algorithm

---

```
Randomly initialize critic network  $Q(s, a|\theta^Q)$  and actor  $\mu(s|\theta^\mu)$  with weights  $\theta^Q$  and  $\theta^\mu$ .
Initialize target network  $Q'$  and  $\mu'$  with weights  $\theta^{Q'} \leftarrow \theta^Q$ ,  $\theta^{\mu'} \leftarrow \theta^\mu$ 
Initialize replay buffer  $R$ 
for episode = 1, M do
    Initialize a random process  $\mathcal{N}$  for action exploration
    Receive initial observation state  $s_1$ 
    for t = 1, T do
        Select action  $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$  according to the current policy and exploration noise
        Execute action  $a_t$  and observe reward  $r_t$  and observe new state  $s_{t+1}$ 
        Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $R$ 
        Sample a random minibatch of N transitions  $(s_i, a_i, r_i, s_{i+1})$  from  $R$ 
        Set  $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$ 
        Update critic by minimizing the loss:  $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$ 
        Update the actor policy using the sampled policy gradient:
            
$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

        Update the target networks:
            
$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

            
$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

    end for
end for
```

---

3. Backtest: Store Results of Portfolio Returns for each strategy
4. Benchmark Strategies: Tests agents results against well documented trading strategies e.g. "Follow the winner", Mean Reversion, Online Newton Step Algorithm, and others.
5. Evaluation: Present cumulative returns of each agent.

# Results

---

# Summary

---

## Current Portfolio Management Trends:

- Data-driven investing is taken over, as it removes emotional trading.
- More focus on ESG investing than ever before, and only set to increase.

## Why Deep Reinforcement Learning?

- Non-stationary nature of financial time series has made current supervised learning prediction models unsuccessful.
- Deep Reinforcement Learning enables the agent to learn directly from playing in the simulation environment.

# Summary of Research Questions

---

## New Research Questions:

- Can the RL Agents reward function be tailored to include ESG performance?
- Can we create a portfolio that outperforms the market index in ESG impact and financial returns?
- How does our strategy compare against other off the shelf algorithmic trading strategies?
- What is the environmental impact of current algorithmic trading strategies?

## References i

---

-  H. Markowitz, "Portfolio Selection," 1952. [Online]. Available: [https://www.math.ust.hk/~maykwok/courses/ma362/07F/markowitz\\_JF.pdf](https://www.math.ust.hk/~maykwok/courses/ma362/07F/markowitz_JF.pdf)
-  N. N. Y. Vo, X. He, S. Liu, and G. Xu, "Deep learning for decision making and the optimization of socially responsible investments and portfolio," *Decision Support Systems*, vol. 124, p. 113097, Sep. 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167923619301265>
-  D. Silver, "UCL Course on RL," library Catalog: [www.davidsilver.uk](http://www.davidsilver.uk). [Online]. Available: <https://www.davidsilver.uk/teaching/>

## References ii

-  Z. Jiang, D. Xu, and J. Liang, "A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem," *arXiv:1706.10059 [cs, q-fin]*, Jul. 2017, arXiv: 1706.10059. [Online]. Available: <http://arxiv.org/abs/1706.10059>
-  T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971 [cs, stat]*, Jul. 2019, arXiv: 1509.02971. [Online]. Available: <http://arxiv.org/abs/1509.02971>

## Q&A

---