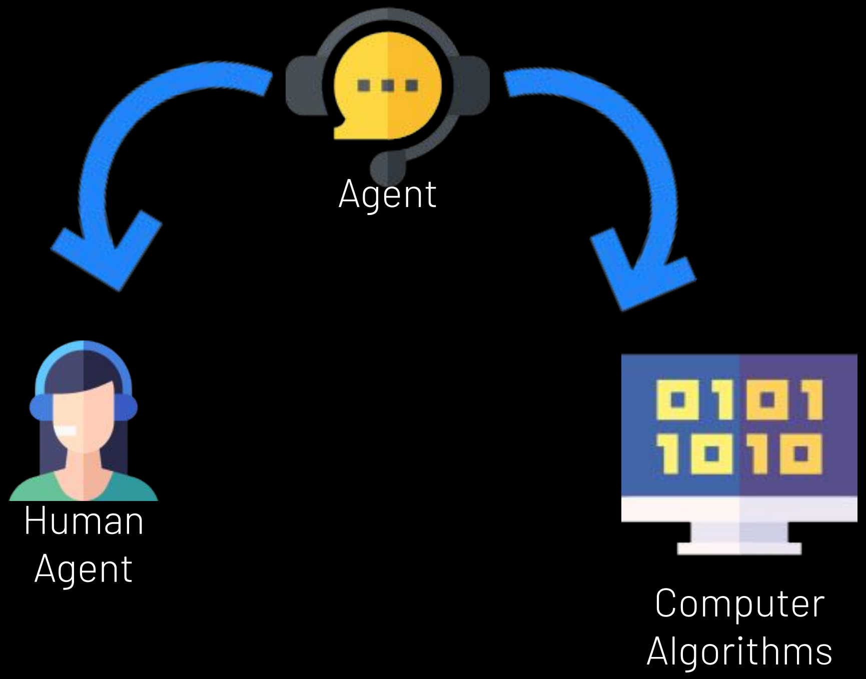
StrategicTradeAI: Leveraging Reinforcement Learning for Dynamic Market Decisions

CHENNAI MATHEMATICAL INSTITUTE

Financial Trading Task

"An agent interacts with the market trying to achieve some intrinsic goal."



”Can an artificial agent learn to trade successfully?”

Reinforcement Learning

Learning paradigm where an agent learns to take actions in an environment to maximize cumulative reward

Agent : The decision maker.

Environment : The world the agent interacts with.

State : The current situation.

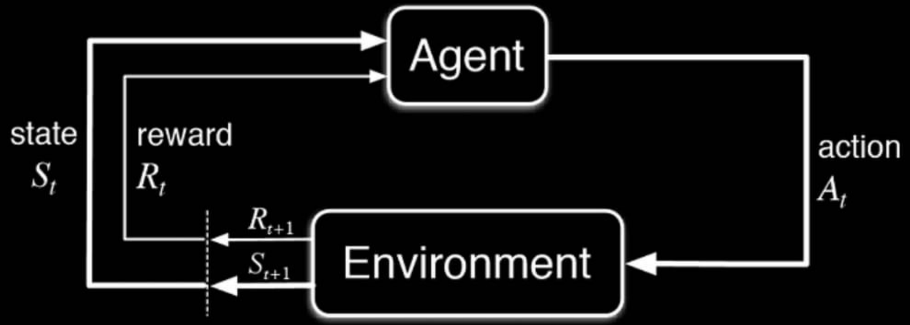
Action : What the agent can do.

Reward : Feedback signal.

Bellman Equations

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} P(s'|s,a) \left[R(s,a) + \gamma V^{\pi}(s')\right]$$

$$Q^{\pi}(s,a) = \sum_{s' \in S} P(s'|s,a) \left[R(s,a) + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s',a')\right]$$



Agent : The decision maker.

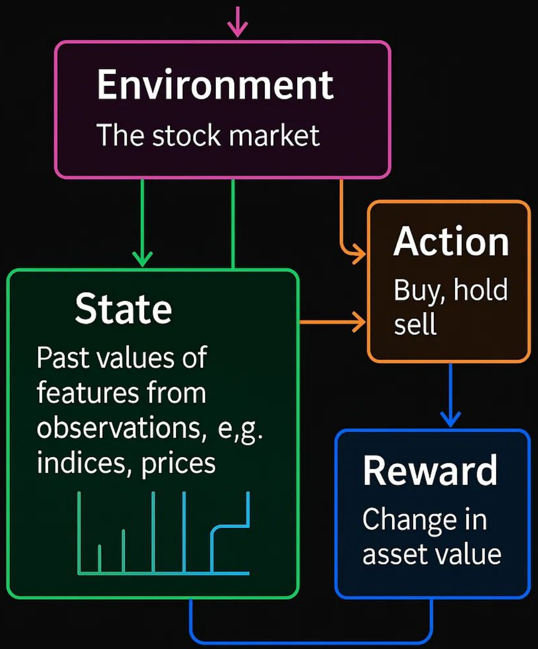
Environment : The world the agent interacts with.

State : The current situation.

Action : What the agent can do.

Reward : Feedback signal.

RL FRAMEWORK FOR STOCK MARKET



Environment : Stock Market

State : Past values of features(Value, Return , MA5, MA10, Volume) from last 10 days

Action : Buy, Hold or Sell one stock

Reward : Change in asset compared to last day

DATA

Data Source

- Yahoo Finance API (yfinance)
- Ticker: Microsoft (MSFT)
- Date Range: **January 1, 2000** to **May 5, 2025**

Data Processing

- **Normalization**: StandardScaler applied to features for scaling to [0,1]
- **Sequence Creation**: 10-day window sequences for LSTM model

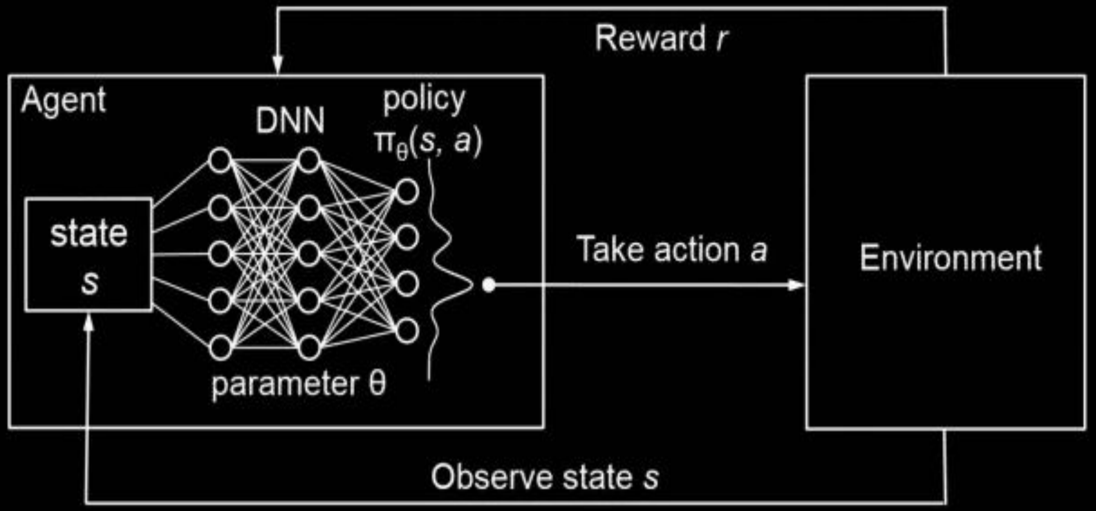
Train-Test Split

- Training Period: **January 1, 2000** to **February 28, 2025**
- Testing Period: **March 1, 2025** to **May 5, 2025**

Key Metrics

- **Training Sequences**: 6363
- **Testing Sequences**: 44
- **Features per Sequence**: Open, High, Low, Close, Volume, MA5, MA10, Return

DEEP Q LEARNING



- Learn a Q-value function :

Q(s, a) = expected cumulative reward of taking action a in state s

- Goal :

Learn the optimal policy π* by approximating Q*

Two Networks in DQN: Policy Net and Target Net

1. Policy Network (`policy_net`)

- Also called the **online network**.
- This is the main neural network being trained.
- It estimates Q-values Q(s,a;θ)Q(s, a; \theta)Q(s,a;θ) given the current state.
- Used to select the action a=argmaxaQ(s,a)a = \arg\max_a Q(s, a)a=argmaxaQ(s,a) during interaction with the environment.
- Parameters θ\thetaθ are updated via gradient descent to minimize the difference between predicted Q-values and target Q-values.

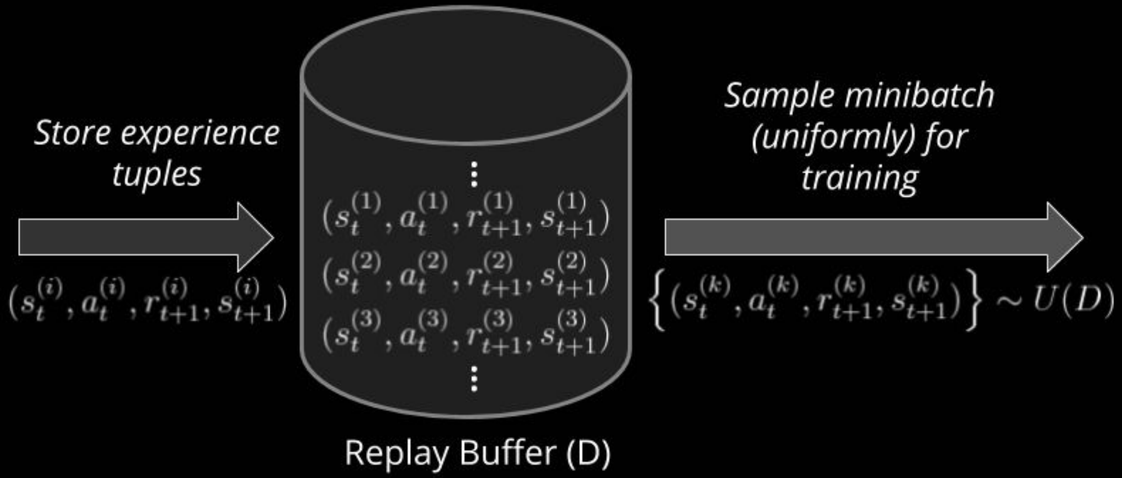
2. Target Network (`target_net`)

- A **stabilizing trick** in DQN.
- It has the **same architecture** as the policy network, but its weights θ−\theta^-θ− are **not updated every step**.
- Instead, it's periodically updated to slowly track the policy network

UPDATION RULE:

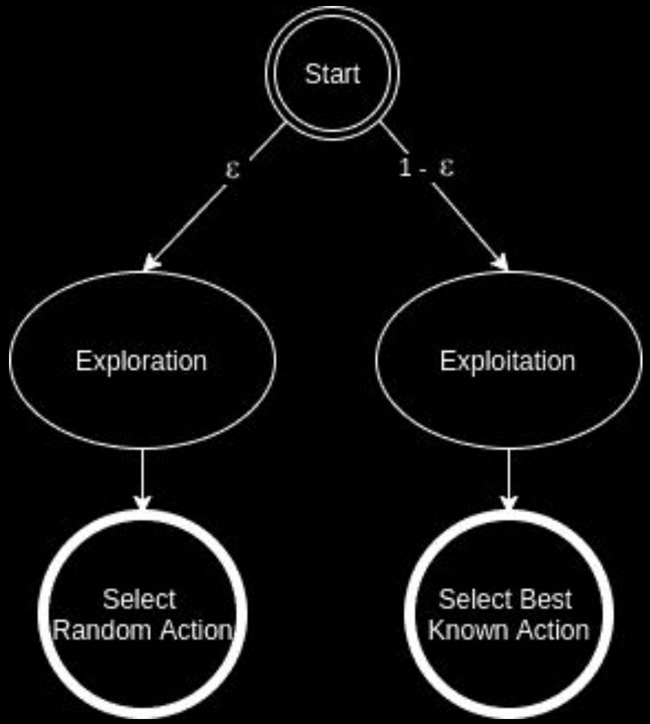
$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s', a'; \theta^{-}) - Q(s, a; \theta) \right)$$

Replay Buffer



- Stores past experiences (s, a, r, s').
- Random sampling helps break correlation and improve stability

Epsilon-Greedy Algorithm



Epsilon decays over time!!!

Actor-critic model

Actor-Critic has 2 components:

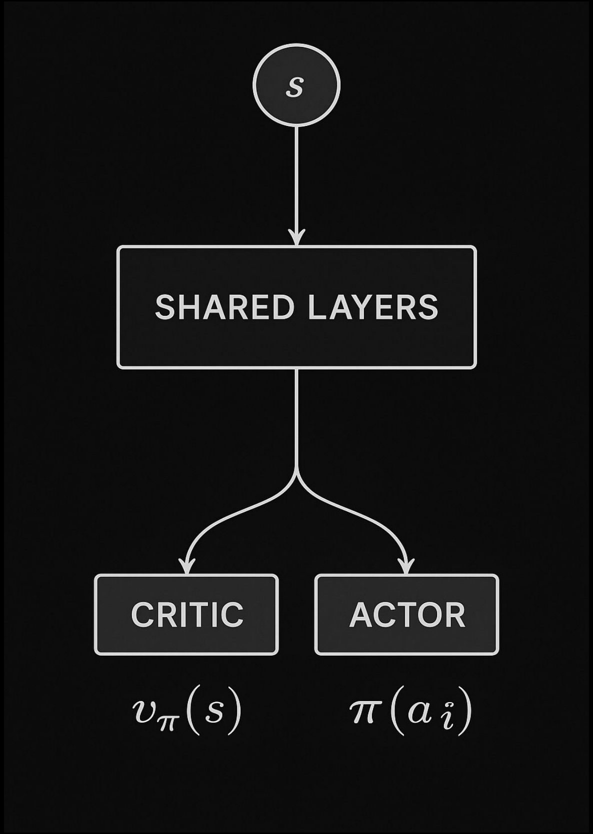
- **One decides what action to take (Actor)**
- **The other judges how good the action is (Critic)**

Actor

- **Takes the current state as input and outputs the probabilities of each possible action (π(a|s))**
- **Samples an action based on this distribution**

Critic

- **Takes the same state as input and output the value of that state (V(s))**



Our target is to maximise π(a|s) for the actions which increases the cumulative reward

Advantage estimate:

$$A(s, a) = r + \gamma * V(s') - V(s)$$

Critic loss:

$$\text{Critic Loss} = (r + \gamma * V(s') - V(s))^2$$

Actor loss:

$$\text{Actor Loss} = -\log(\pi(a|s)) * A(s, a)$$

Proximal Policy Optimization (PPO)

Why PPO?

- **Standard Actor-Critic methods can make unstable or overly aggressive policy updates.**
- **PPO solves this by restricting how much the policy changes during training.**

Key idea

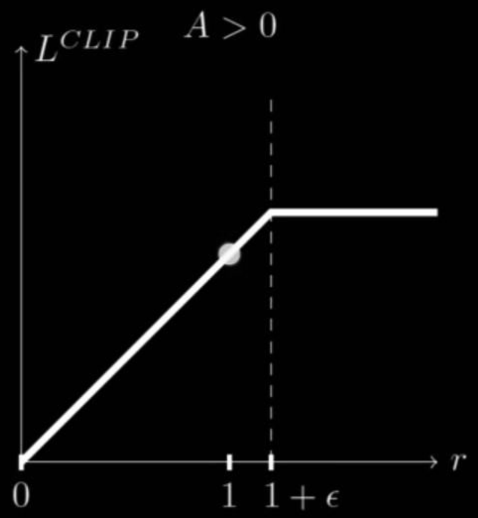
`r(θ) = π_θ(a|s) / π_old(a|s)`

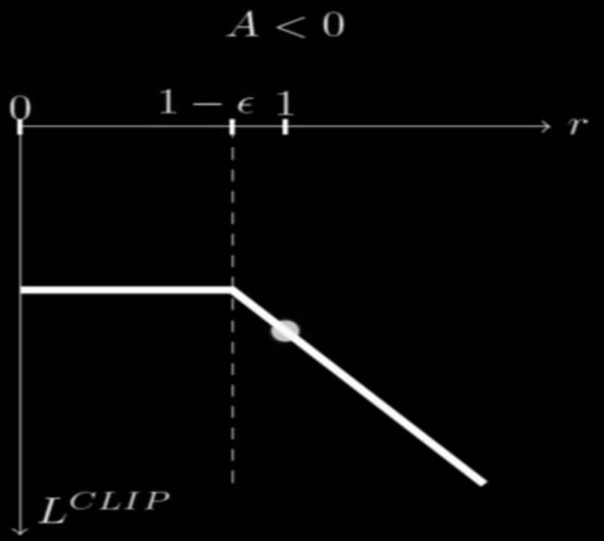
Maximise `A(s,a)*r(θ)` such that `r(θ)` **is not too small not too large**

Clipped Objective of actor:

```
L_clip = min(r(θ) * A(s,a), clip(r(θ), 1-ε, 1+ε) * A(s,a))
```

$$\mathrm{clip}(r(\theta), 1-\epsilon, 1+\epsilon) = \begin{cases} 1-\epsilon & \text{if } r(\theta) < 1-\epsilon \\ r(\theta) & \text{if } 1-\epsilon \leq r(\theta) \leq 1+\epsilon \\ 1+\epsilon & \text{if } r(\theta) > 1+\epsilon \end{cases}$$





RESULTS

Model	Profit
DQN	4.24
A2C	2.12
PPO	2.31

Challenges

- Data Quality & Availability
- Partial Observability of Markets
- Sensitivity to initialization and system-level randomness

Future Prospects

- Action Volume Optimization
- Portfolio-Level Extension
- Expanded Feature Space

THANK YOU