Stock Trading with Reinforcement Learning

Zeyu Chen, Manish Kumar

Executive Summary

- *Utilization of reinforcment learning for stock trading*
- *Examination using AAPL over ~12 years of training data and ~1 year of testing data*
- *Agenda:*
  - *Discussion of trading environments: states, transitions, rewards and steps*
  - *Application across various approaches varying in algorithmic complexity*
  - *Performance comparison, additional application and conclusion*

Experiment	**Libraries Utilized**	**Algorithms**
#1	Custom built (leveraged anytrading) + ta	Q-Learning
#2	gym-anytrading + StableBaselines3	A2C , DQN
#3	FinRL	A2C, DDPG, PPO

Section #1: Custom Trading Environment + Q-Learning

Data Processing and Technical Indicators

Data preparation

- Formatting: Convert dates to pandas datetime format
- Technical indicators as state variables

Technical Indicators Overview

- Stochastic Oscillator
- Relative Strength Index (RSI)
- MACD (Moving Average Convergence Divergence)
- Bollinger Bands
- Commodity Channel Index (CCI)
- Directional Movement Index (DX)
- Simple Moving Averages (SMA)

Ten buckets to convert from continuous to discrete space variables for experiments #1 and #2

Continuous state-space for FinRL

States Overview

- Sequential time series data with algorithm driven based on index location
- States based on combinations of variables / indicators
- Performance based on day shifted percentage changes

	Date	Open	High	Low	Close	Adj Close	Volume	smi	rsi	rsi_bucket	smi_bucket	state	next_state	buy_reward	sell_reward
0	2010-12-03	11.322	11.380	11.298	11.337	9.610	342092800	-136.780	55.050	4	1	(4, 1)	(5, 1)	0.009	0
1	2010-12-06	11.380	11.512	11.372	11.434	9.692	448481600	-124.833	58.552	5	1	(5, 1)	(4, 1)	-0.006	0
2	2010-12-07	11.564	11.571	11.361	11.365	9.634	391454000	-116.388	55.235	4	1	(4, 1)	(5, 1)	0.009	0

Daily data

Adjusted stock price

Technical indicators

Bucketed technical indicators as state variables

Sequential time series where next date is the next state

Long-only / "in" or "out" of the market…buy reward is one day shifted percentage change in stock price. Sell reward is zero

Simplified Trading Environment Overview (1 of 2)

General Setup

```python
class CustomTradingEnv(gym.Env):
    def __init__(self, df):
        super(CustomTradingEnv, self).__init__()

        # Define the action space (0 for buy, 1 for sell)
        self.action_space = spaces.Discrete(2)

        # Define the observation space (state)
        # Define the observation space (state)
        self.observation_space = spaces.Box(
            low=np.array([0, 0], dtype=np.float32),
            high=np.array([10, 10], dtype=np.float32),
            shape=(2,)
        )
```

```python
        # Set the DataFrame for price data
        self.df = df

        # Initialize the environment
        self.index_loc = 0
        self.state = self.df['state'].iloc[self.index_loc]
        self.done = False
        self.history = []

    def reset(self):
        # Reset the environment to its initial state
        self.index_loc = 0
        self.state = self.df['state'].iloc[self.index_loc]
        self.done = False
        self.history = []
```

0: Buy and 1: Sell action space

Ten buckets for states

Start at first index location

State driven by index location

Simplified Trading Environment Overview (2 of 2)

Steps

```
def step(self, action):

    # Ensure the action is valid
    assert self.action_space.contains(action)

    # Calculate the reward based on the percentage change and balance
    self.reward = 0.0  # Initialize reward

    if action == 0:  # Buy
        self.reward = self.df.loc[self.index_loc, 'buy_reward']

    self.done = (self.index_loc == len(self.df) - 1)  # Episode is done
    self.info = {}  # Additional information

    # The state is taken from the observation space
    self.state = self.df['state'].iloc[self.index_loc]

    # The current price is taken from the observation space
    self.current_price = self.df['Adj Close'].iloc[self.index_loc]
```

Long-only model with reward driven by buy / sell ‘signal’

Stop at last index location

```
    self.episode_step = {
        'date': self.df['Date'].iloc[self.index_loc], # Include the dat
        'index_loc': self.index_loc,  # Renamed 'index' to 'step'
        'state': self.state,  # Use self.state instead of state
        'action': action,
        'reward': self.reward,
        'price': self.current_price  # Use self.current_price instead o
    }
    self.history.append(self.episode_step)

    # Retrieve the next state
    self.next_state = self.df['next_state'].iloc[self.index_loc]

    # Assign next_step
    self.index_loc = self.index_loc + 1
    self.state = self.df.loc[self.index_loc, 'state']

    # Return the updated information
    return self.index_loc, self.state, action, self.reward, self.done
```

Move down the index and select the next state

Return data for reinforcement learning

Q-Learning: Performance calculation and results

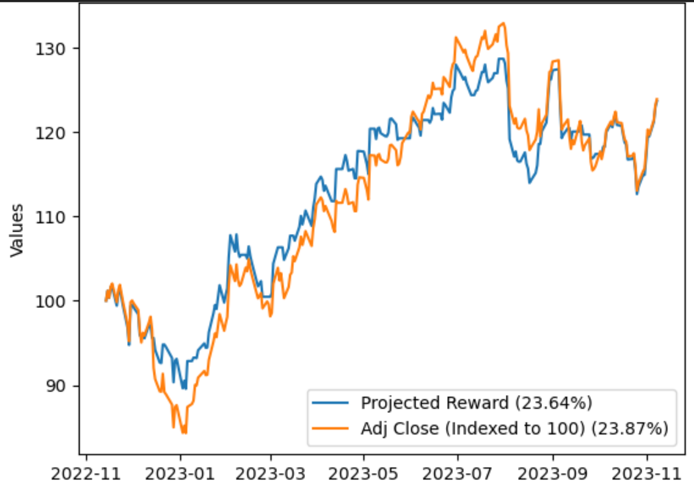
- Application of the optimal policy from Q-learning – no holdout dataset
- **Episodes & steps_per:** 10 & 2,000 / **epsilon (non-greedy)**: 0.10 / **alpha (learning rate)**: 0.35 / **gamma:** 0.99

Value calculation formula accounts for in / out status

	Date	Adj Close	buy_reward	sell_reward	projected_reward	optimal_policy	in_out
0	2022-11-14	147.456	0.012	0	100.000	0	in
1	2022-11-15	149.206	-0.008	0	101.177	0	in
2	2022-11-16	147.963	0.013	0	100.324	0	in
3	2022-11-17	149.882	0.004	0	101.615	1	in
4	2022-11-18	150.449	-0.022	0	101.615	0	out
...	...	...	...	...	...	...	...
243	2023-11-02	177.336	-0.005	0	122.427	1	in
244	2023-11-03	176.418	0.015	0	122.427	0	out
245	2023-11-06	178.994	0.014	0	124.203	0	in
246	2023-11-07	181.581	0.006	0	125.985	0	in
247	2023-11-08	182.649	-0.003	0	126.714	0	in

"in_out" status based on action taken at the end of the prior state

Performance



Note: fee estimate of 0.0001 applied to rewards at sale

Section #2: gym-anytrading + StableBaselines3

Library Overview

StableBaselines3

- A set of RL algorithms implemented in PyTorch
- Offers a unified structure for a wide range of RL algorithms
- Predefined policies for each model

gym-anytrading

- A collection of OpenAI Gym environments for RL-based trading algorithms
- Offers two Gym environments: ForexEnv and StocksEnv

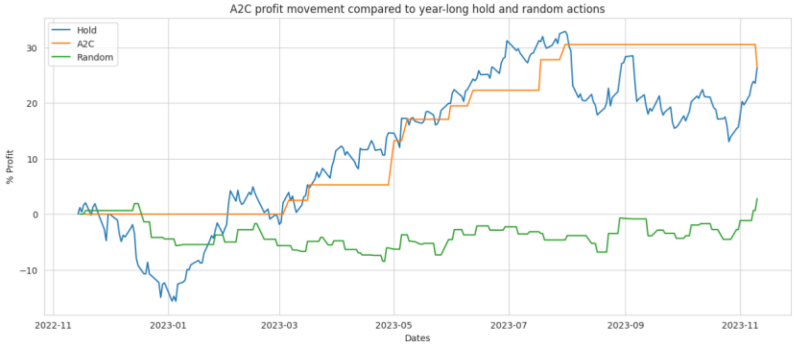
```python
import gymnasium as gym
import gym_anytrading
from stable_baselines3 import PPO

env = gym.make('stocks-v0')

model = PPO("MlpPolicy", env)
model.learn(total_timesteps=10000)
```

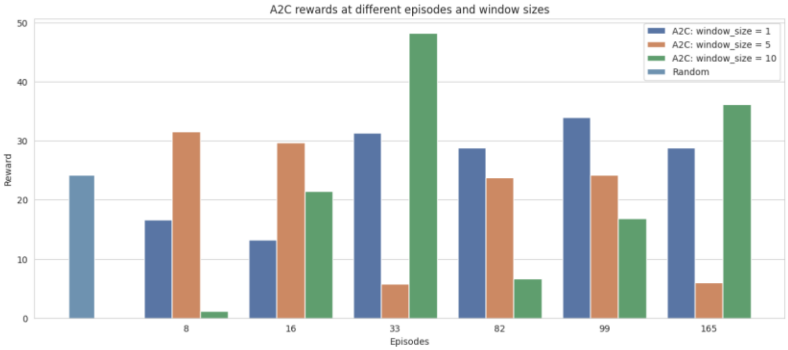
Environment Setup

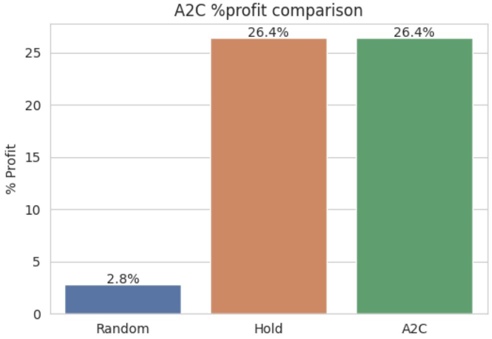
- Actions : Buy (1) and Sell (0)
- Indicators : RSI, SMI (Discreet space of 10)
- Trading Fee (Buy/Sell) : 0.1%
- Prices : Adjusted Close
- Reward : Price difference between trades
- Training Data : 12 years (Nov 2010 – Nov 2022)
- Testing Data : 1 year (Nov 2022 – Nov 2023)

A2C: Advantage Actor-Critic

- Actor: Policy-based; decides what action to take
- Critic: Value-based; critiques the action that the Actor selected, provides feedback on how to adjust
- Advantage: How much better an action is compared to the average action at a given state
- Hyperparameters - policy: MlpPolicy, α: 5e-4, γ: 0.99

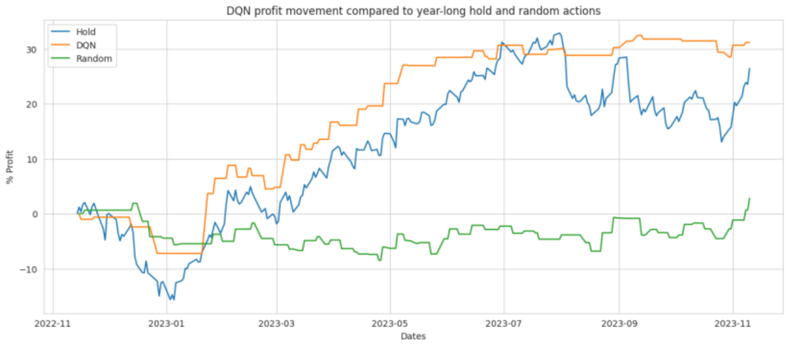


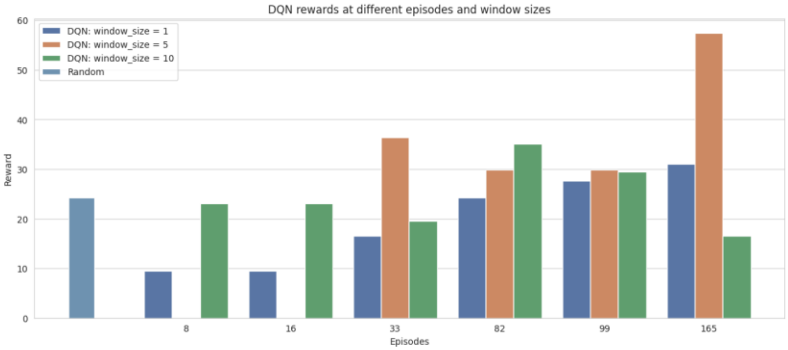


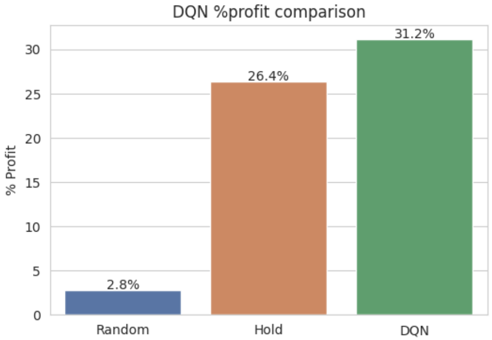


DQN: Deep Q-Learning

- Combines traditional Q-learning with deep neural networks
- Can learn optimal policies for high-dimensional and complex environments
- Hyperparams - policy: MlpPolicy, α: 1e-4, batch_size: 32, γ: 0.99







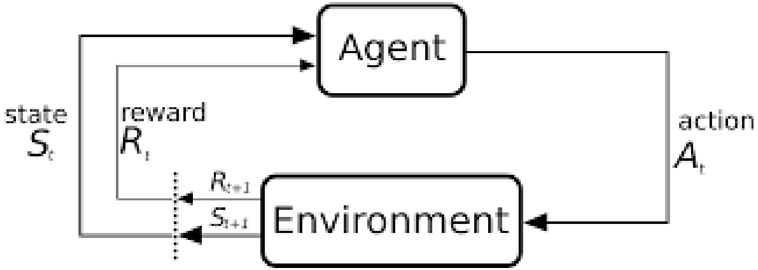
Section #3: Trading in FinRL

FinRL: Overview

Why FinRL

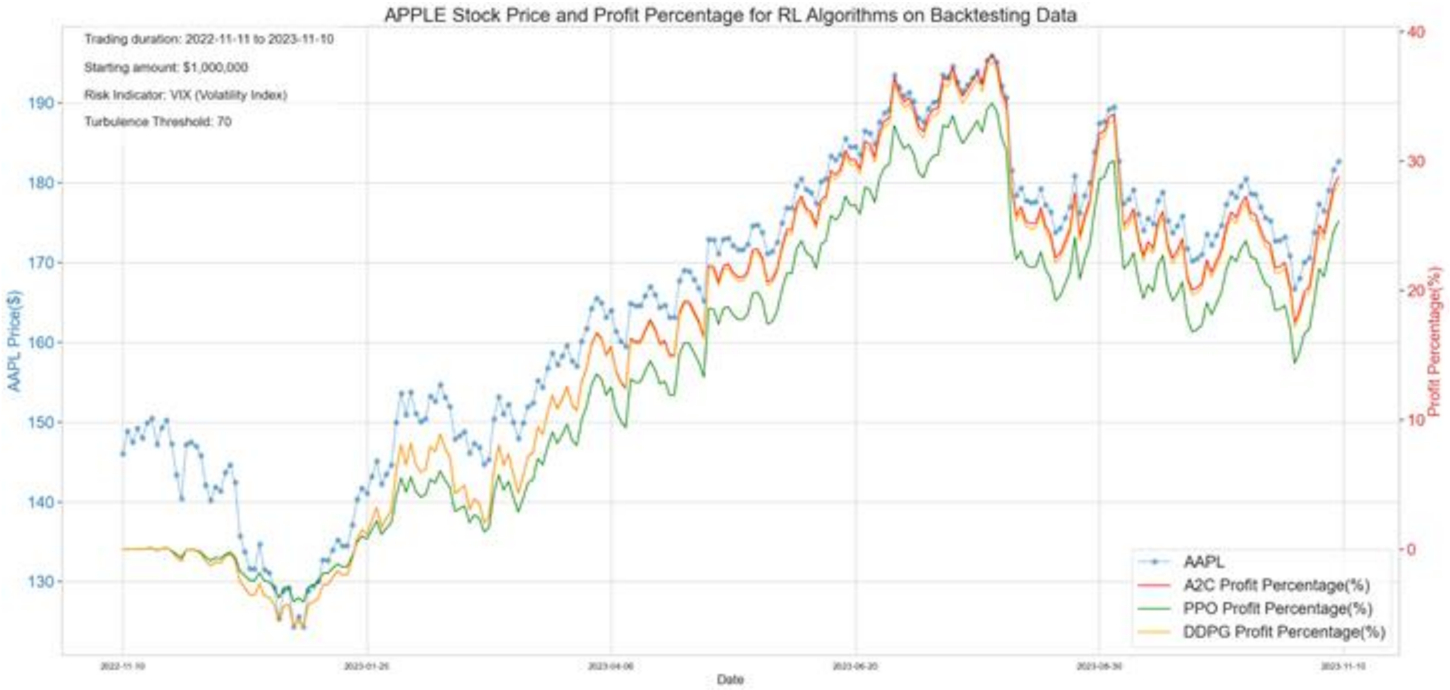
- Open source library that specify for finance based of OpenAI Gym
- Easy integration with ElegentRL and Stablebaseline
- Ensemble trading agents training
- Explorability with portfolio allocation
- Three layer architecture that link to real financial/crypto data

Training Environment



- **Training Duration**: 200,000 Timesteps
- **Model Selection**: A2C, PPO, DDPG
- **Reward**: Change in Account Value
- **Max Shares per Transaction**: 100 shares
- **Training Budget**: $ 1 million
- **Transaction Fee**: $0.001 per stock
- **Action Space**: Continuous between -1 and 1, represent the proportion of max shares
- **State Space**: Continuous Space of Market information with added technical indicators

Backtest



APPLE Stock Price and Profit Percentage for RL Algorithms on Backtesting Data
Trading duration: 2022-11-11 to 2023-11-10
Starting amount: $1,000,000
Risk Indicator: VIX (Volatility Index)
Turbulence Threshold: 70
AAPL Price($)
Profit Percentage(%)
Date
AAPL
A2C Profit Percentage(%)
PPO Profit Percentage(%)
DDPG Profit Percentage(%)

Results and Key Takeaways

- Reinforcement Learning Algorithms exhibit a keen ability to discern and capture trends of test data.
- These algorithms excel in making accurate trading decisions.
- Reinforcement Learning in single-stock trading encounters challenges, particularly in its sensitivity to the stock's volatility.

	Annual Return
Hold Apple Stock	~25.0%
PPO Trading	25.3%
A2C Trading	29.8%
DDPG Trading	29.3%

Section #4: Conclusions and Future Opportunities

Comparison between Trading strategies (AAPL)

Trading Strategies	Annual Return (%)
Q-Learning in Custom Environment	23.9%
DQN in Gym-Anytrading	31.2%
A2C in Gym-Anytrading	26.4%
PPO in FinRL	25.3%
A2C in FinRL	28.8%
DDPG in FinRL	28.3%

Conclusions

- Reinforcement Learning models have the potential to capture non-linear patterns and adapt to changing market conditions.
- Preprocessing financial data and selecting relevant features are critical for performance.
- Trading with Reinforcement Learning poses difficulties when dealing with volatile stocks.
- Trading with ensemble agents and investment portfolio can mitigate risk.
- Further hyperparameter tuning can improve the agents' learning capability and make better decisions in trading.

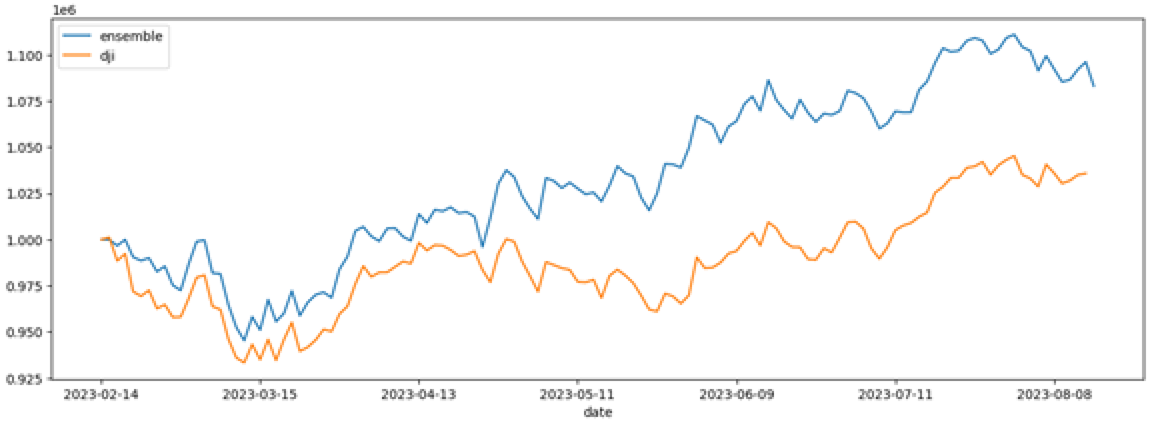
Future Opportunity: Portfolio Trading

- DJI 30 Companies
- Various Industry
- Trained with **A2C, DDPG, PPO**
- Compare with traditional trading method **Mean Variance Optimization**

Trading Strategies	Annual Return
A2C	7.7%
DDPG	7.8%
PPO	16.9%
MVO	12.5%

Future Opportunity: Ensemble Agents

- Training and validating 3 agents (**A2C, PPO, DDPG**)
- Rolling-window Ensemble Method
- Turbulence Threshold Tuning



	Annual Return	Sharpe Ratio
Ensemble	17.3%	1.309
Dow Jones	7.4%	0.660

Future Opportunity: Hyperparameter Tuning

Q-Learning (DQN)

- Learning Rate: Adjusts the step size
- Discount Factor: Balances the importance of immediate and future rewards.
- Exploration Rate
- Batch Size and Memory Size
- Online-offline models update frequency

Deep Deterministic Policy Gradient (DDPG)

- Actor/Critic Network Learning Rate
- Discount Factor
- Batch Size and Memory Size

Actor-Critic

- Actor Learning Rate
- Critic Learning Rate

Proximal Policy Optimization (PPO)

- Learning Rate
- Clip Parameter: Constrains the ratio of new and old policy probabilities.
- Value Function Coefficient