Assignment I Bandit Algorithm and Stock Investment

Group: RLight

Problem Setting

- Investment Goal: to maximize the next day return.
- Current Information: daily return distribution for each stock.
  - It is derived from historical data.
    - Data file: **./RL_A1_Data/hs300_daily.csv**
  - After statistical test, we assume most of stocks return follows t distribution.
    - Program: **Distribution_Estimation.py**
    - Detailed parameters are listed in:
      - **./RL_A1_Data/ hs300_stock_t_params.csv**
  - So our reward is a stochastic one.
- Actions to choose: we have 203 stocks to present 203-arm bandit problem and for each epoch, we try to select the arm with the highest value function.
- Algorithms: we mainly use epsilon-greedy to train our model and we add a small UCB parameter to optimize our exploration process.

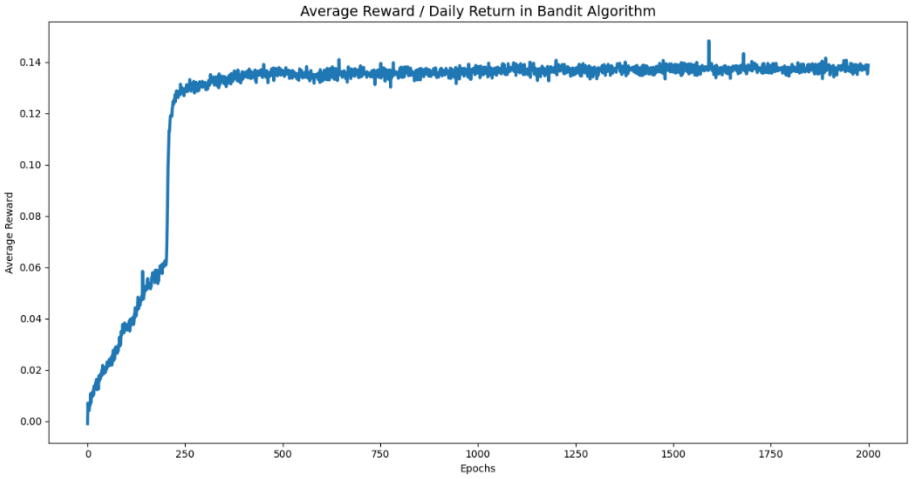
Algorithm Training

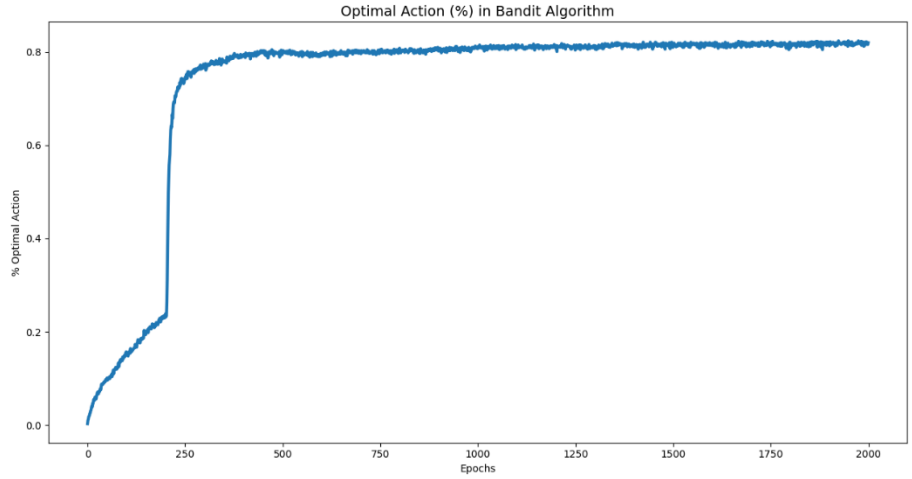
- Training part iteratively deals with act() and step() functions.
  - act(): to select one arm based on q_estimation
  - step(): to make q_estimation get close to q_true

```
def act(self):
    if np.random.rand() < self.epsilon:
        return np.random.choice(self.indices)

    if self.UCB_param is not None:
        UCB_estimation = self.q_estimation + \
                         self.UCB_param * np.sqrt(np.log(self.time + 1) / (self.action_count + 1e-5))
        q_best = np.max(UCB_estimation)
        return np.random.choice(np.where(UCB_estimation == q_best)[0])

    q_best = np.max(self.q_estimation)
    return np.random.choice(np.where(self.q_estimation == q_best)[0])

def step(self, action):
    reward = self.reward_deviation(action) + self.q_true[action]

    self.time += 1
    self.action_count[action] += 1

    # update estimation using sample averages
    self.q_estimation[action] += (reward - self.q_estimation[action]) / self.action_count[action]

    return reward
```

Algorithm Result

- Our model converges from perspectives of average reward and optimal action percentage.





- Also, we store the policy dataframe which could help us decide the best stock to invest at current time based on bandit algorithm, i.e., stock #76.
  - Remark. Stock_number 0 is NOT considered because we initialize our policy vector to be np.zeros which is explained in our code.

	A	B	C
1	stock_number	sign	
2	0	161	
3	76	18	
4	86	15	
5	35	13	
6	46	12	
7	56	11	
8	71	11	
9	81	10	
10	111	10	
11	123	9	