SCIE4500 Final Presentation
Policy Gradient Trading Algorithm by Maximizing Sharpe Ratio

Supervisor: Prof. Yao, Yuan
Student: Wang, Xinyi

MATH IRE, HKUST

September 29, 2019

Outline

Problem Description

Introduction and Background

Project Setup

Conclusion

Problem Description

- A trading algorithm to trade a single or multiple securities.
- Optimize some relevant measure of trading system performance. (e.g. profit, economic utility, risk-adjusted return, etc)
- Sharpe ratio:

$$S_T = \frac{Average(R_t)}{Standard\ Deviation(R_t)}$$

Where R_t is the return of the system at step t.
Commonly used to understand the return of an investment compared to its risk over a period of time

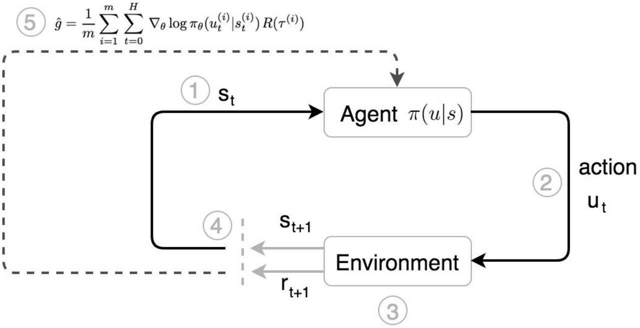
Trading problem as a reinforcement learning problem

Reinforcement learning as a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$:

$$p(s', r|s, a) = Pr\left[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\right]$$

π is the policy that the agent should follow to maximize the total reward (the probability distribution of actions given a state):

$$\pi(A_t = a|S_t = s) \;\; \forall A_t \in \mathcal{A}(s), S_t \in \mathcal{S}$$



Trading problem as a reinforcement learning problem

- Want to learn the best trading policy π given the current market data. i.e. gaining the best Sharpe ratio on the validation set.
- Environment: define the current state s by four scalar: ['Open', 'High', 'Low', 'Close', 'Volume']. Define p_t as the close price at time step t.
- Actions: *long* $(a_t = 1)$ or *short* $(a_t = -1)$ position
- Immediate return: $R_t = (\log p_{t+1} - \log p_t)a_t$
- Average return over a trajectory $\tau = (s_1, a_1, R_1; ...; s_T, a_T, R_T)$: $\bar{R_\tau} = \frac{1}{T}\sum_{i=1}^{T} R_i$
- Sharpe ratio over τ: $S_\tau = \frac{\bar{R_\tau}}{\frac{1}{T}\sqrt{\sum_{i=1}^{T}(R_i - \bar{R_\tau})^2}}$
- Optimization algorithm: policy gradient

Introduction: Policy Gradient

The learning objective of policy gradient:

$$\arg\max_{\pi} \mathbb{E}_{\pi}\left[r(\tau)\right]$$

- $\tau = (s_1, a_1, R_1; ...; s_T, a_T, R_T)$: a given trajectory.
- $r(\tau) = \sum_i r_i$: the total reward received on this trajectory. Instead of directly using the sum of immediate return, define $r(\tau)$ as the Sharpe ratio S_τ.

The update rule using gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} \mathbb{E}_{\pi}\left[r(\tau)\right]$$

Introduction: Policy Gradient

The policy for a trajectory: (s_{t+1}, r_{t+1} only depend on s_t, a_t)

$$\begin{aligned}\pi_\theta(\tau) &= \mathcal{P}(s_0) \prod_{t=0}^{T} P(s_{t+1}, r_{t+1}, a_t | s_t) \\ &= \mathcal{P}(s_0) \prod_{t=0}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}, r_{t+1}|s_t, a_t)\end{aligned}$$

Then by using the trick of $\nabla_\theta \pi(\tau) = \pi(\tau) \nabla_\theta \log \pi(\tau)$, we can get:

$$\begin{aligned}\nabla_\theta \mathbb{E}_\pi \left[r(\tau)\right] &= \nabla_\theta \int \pi_\theta(\tau) r(\tau) d\tau \\ &= \int r(\tau) \nabla_\theta \pi_\theta(\tau) d\tau \\ &= \int r(\tau) \pi(\tau) \nabla_\theta \log \pi_\theta(\tau) d\tau\end{aligned}$$

The Policy Gradient Theorem:

$$\nabla_\theta \mathbb{E}_\pi \left[r(\tau)\right] = \mathbb{E}_{\pi_\theta} \left[r(\tau) \nabla_\theta \log \pi_\theta(\tau)\right] \tag{1}$$

Introduction: Policy Gradient

Eliminate $\mathcal{P}(s_0)$ and $p(s_{t+1}, r_{t+1}|s_t, a_t)$:

$$\log \pi(\tau) = \log \mathcal{P}(s_0) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T} \log p(s_{t+1}, r_{t+1}|s_t, a_t)$$

$$\nabla_\theta \log \pi(\tau) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$\nabla_\theta \mathbb{E}_\pi \left[r(\tau)\right] = \mathbb{E}_\pi \left[r(\tau) \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \right]$$

Replace the expectation by sampling and averaging:

$$\nabla_\theta \mathbb{E}_\pi \left[r(\tau)\right] \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} r(\tau) \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})$$

Proposed algorithm

At the t-th training step, sample a trajectory τ from the training set:
Forward propagation: (recall that $r(\tau) = S_\tau$ is the Sharpe ratio)

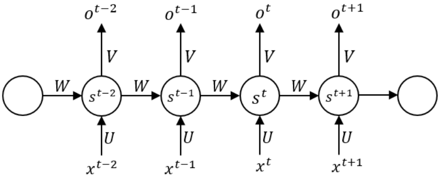
$$\mathcal{L}(\theta)_t = -S_\tau \log \pi_\theta(a_t | s_t)$$

Back propagation:

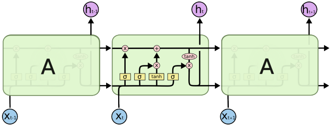
$$\frac{\partial \mathcal{L}(\theta)_t}{\partial \theta} = S_\tau \frac{\partial \log \pi_\theta(a_t | s_t)}{\partial \theta}$$
$$\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}(\theta)_t}{\partial \theta}$$

- The policy π_θ is approximated directly by a neural network (LSTM) with parameter θ.
- The loss function $\mathcal{L}(\theta)_t$ is especially designed to have the negative objective gradient.

Approximate policy π using neural network



(a) RNN: recurrent neural network



(b) LSTM: long short term memory unit

- Use LSTM units.
- Keep a sequence length of T (same as the trajectory length)

Approximate policy π using neural network

- suppose the output of the above network is $(f_\theta(s_t|long), f_\theta(s_t|short))$
- Use softmax function to approximate the conditional probability distribution of the action a_t:

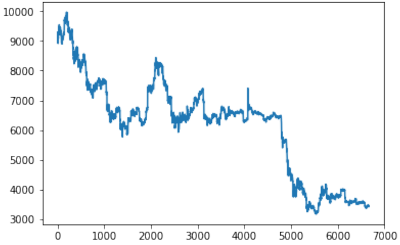
$$\pi_\theta(a_t|s_t) = \frac{\exp f_\theta(s_t|a_t)}{\exp f_\theta(s_t|long) + \exp f_\theta(s_t|short)}$$

Recall the back propagation:

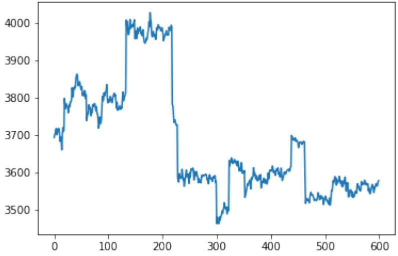
$$\frac{\partial \mathcal{L}(\theta)_t}{\partial \theta} = S_\tau \frac{\partial \log \pi_\theta(a_t|s_t)}{\partial \theta}$$
$$\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}(\theta)_t}{\partial \theta}$$

Data

- Bitcoin data from 2019-01-31 01:20:00 to 2018-04-28 03:05:00 obtained from the MAFS6010U course.
- The per-minute data is aggregated to per-hour data to reduce the variance.
- 5336 data is used for training; 600 data is used for validation and testing respectively
- Sharpe ratio of always taking *long* position on the test set: **-0.0098**



(c) Close price of whole dataset



(d) Close price of test data

Data pre-process for state definition

- $s_t = ('Open', 'High', 'Low', 'Close', 'Volume')$
- 'Open': price at the first minute of hour t.
- 'High': highest price in hour t. 'Low' is the lowest price in hour t.
- 'Close': price at the last minute of hour t. Use 'Close' as p_t to calculate the return R_t.
- Take their log form and then subtract the number in the previous hour.
- Scaled by the mean and standard deviation of the train data.

Experiment Setup: Hyper-parameters

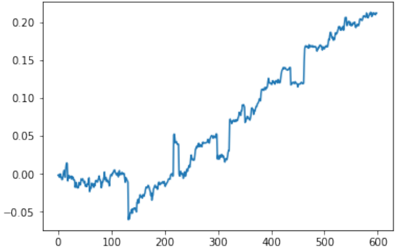
- Learning rate: $\alpha = 1e - 4$
- Optimizer: Adam
- Batch size: 4
- Trajectory (sequence) length $T = 20$
- Random sampling a trajectory from the train data at each episode.

Experiment results

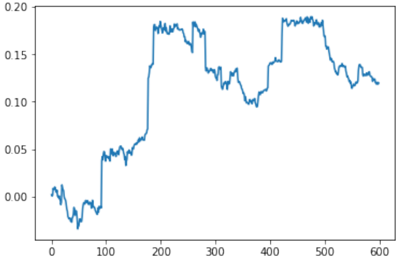
Cumulative return $\hat{R}$ on the test set:

$$\hat{R} = \sum_t (\log p_{t+1} - \log p_t) a_t$$

Trajectory (episode) length $T = 20$.



(a) Policy gradient (proposed)
Sharpe ratio: **0.0654**



(b) Q learning
Sharpe ratio: **0.0485**

Baseline: Q Learning

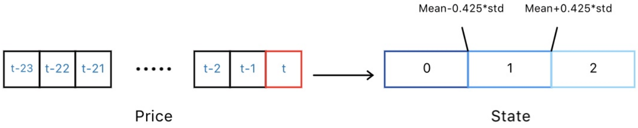
Define accumulated reward as $\sum_t \gamma^t S_\tau R_t$. (Sharpe ratio: take previous performance into consideration.)
Use a Q function $Q(S_t, A_t)$ to approximate the maximum accumulated reward, which can be updated by the Bellmen equation:

$$Q(S_t, A_t) = S_\tau R_t + \gamma \max_h Q(S_{t+1}, A_h)$$

```
Initialize Q(s,a) arbitrarily
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      Choose a from s using policy derived from Q
      Take action a, observe r, s'
      Update
        Q(s,a) ← Q(s,a) + α[r + γ max_a' Q(s',a') − Q(s,a)]
      s ← s';
   Until s is terminal
```

Baseline: Q Learning

- Only keep a finite number of states so that it is possible to remember the Q value at each state given the action.
- Discretize the states using a moving window of the previous states to fit the current price into one of the three slots.

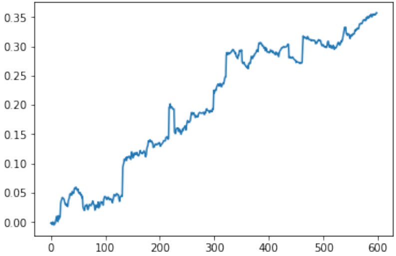


More results

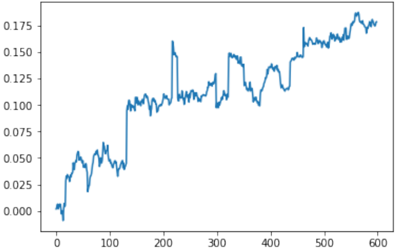
Cumulative return $\hat{R}$ on the test set:

$$\hat{R} = \sum_t (\log p_{t+1} - \log p_t) a_t$$

Trajectory (episode) length $T = 100$.



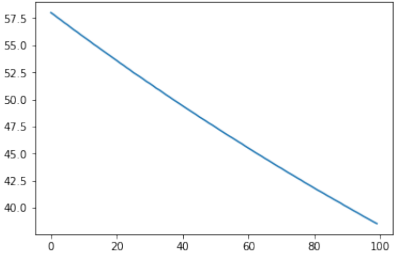
(c) Policy gradient (proposed)
Sharpe ratio: **0.1108**



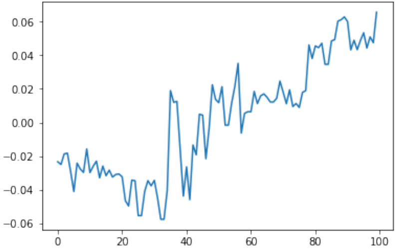
(d) Q learning
Sharpe ratio: **0.0551**

More results

Some training data of policy gradient of the first 100 epoch.
Trajectory (episode) length $T = 100$.



(e) train loss



(f) validation Sharpe ratio

Conclusion and Analysis

- By using the policy gradient algorithm, the agent can learn from the market environment to take some meaningful trading decisions.
- Both policy gradient and Q learning tends to perform better with longer trajectory length ($T = 100$).
- This might because that longer trajectory could explore the environment better under the current policy.

- Thank you!