STAT 497 - Final Project

Concordia University

Bakr Abbas, Matthew Liu, Frederic Siino

Day Month Year

0 Introduction

One of the main challenges of managing a portfolio or a fund, is the decision process of how to allocate investment resources to each assets. The goal is to maximize returns but also to diversify, putting all the fund in the most profitable asset could be beneficial in the short run, but the possibility of that one asset crashing in value is larger than the possibility of multiple assets crashing in value. So diversifying is important to hedge against that outcome, and portfolio management in the modern day involves more automation than ever. Reinforcement & Deep learning methods are being used to optimize this decision making process.

Cryptocurrencies are decentralized digital assets that behave as an alternative to government issued money. They are being introduced into most investment portfolios, this paper address the challenge of managing five different cryptocurrencies with Bitcoin(BTC) being cash.Meaning that the four other currencies are quantified as the price relative to BTC. DIfferent methods will be used to manage the weights assigned to each asset in the portfolio. The first three methods will be based on the contextual bandits model.The final method will be based on a policy gradient model. A more realistic model would also take transaction costs into account, but this will be set aside for simplicity.

A previous research paper (Jiang, Xu, Liang, 2017) has been published on this exact topic based on a framework consisting of Ensemble of Identical Independent Evaluators (EIIE), a Portfolio-Vector Memory (PVM), an Online Stochastic Batch Learning (OSBL) scheme and a fully exploiting strategy. The actions were determined using a policy gradient method and the tested evaluators were a Convolutional Neural Network (CNN), a Recurrent Neural Network (RNN) and a Long Short-Term Memory Network (LSTM). Experiments were run with a trading period of 30 minutes and the RL framework was observed to behave far better than other optimization frameworks in the test period of 2017 but inferior to some frameworks in the test period of 2018.

In this project, we opt for a much simpler approach, relying instead on an action preference function and a policy gradient when framing the task as a contextual bandits problem. We do however borrow some aspects of Jiang, Xu and Liang such as definitions for states and returns.

1 Problem description

In this project, the environment is solely dictated by the data of the historic prices of all cryptocurrencies relative to BTC. In the experiments, a trading period of $T = 120$ minutes is used, and we only consider the closing prices of each period (which are equivalent to the opening prices of the next period). In addition, the experiments rely on two key assumptions (Jiang, Xu, Liang, 2017):

- Instant liquidity: The liquidity of the market allows each trade to be carried out immediately at the price listed when the order is placed.
- No market impact: The capital invested by the agent is so insignificant as to have no impact on the market.

This means that the sequence of states and transition probabilities are already predefined based on the data at hand. In particular, we define a price relative vector y_t of the tth trading period like Jiang, Xu and Liang, namely:

$$y_t := v_t \oslash v_{t-1} = \left(1, \frac{v_{2,t}}{v_{2,t-1}}, \ldots, \frac{v_{m-1,t}}{v_{m-1,t-1}}, \frac{v_{m,t}}{v_{m,t-1}}\right)^T$$

where $v_{t,i}$ denotes the closing price of the ith asset in the tth trading period. Note here that asset 1 and thus $v_{t,1}$ is always equal to 1. This is because the relative prices $v_{t,i}$ are relative to the first asset (in our case, BTC), also known as the cash. This price relative vector can then be used to calculate the reward, defined in general as

$$r_t = y_t \cdot w_{t-1}$$

where w_{t-1} is the portfolio weight vector at time $t-1$. This reward is effectively the change in portfolio value for period t. The action a_t at time t is the portfolio weight vector w_t, where the ith element is the proportion of asset i in the portfolio. The goal of the agent is then to maximize the growth rate (or cumulative return) r_t of the portfolio over the experimental time frame.

In addition, we also define a state vector $S_T := (s_{1,T}, ..., s_{m,T})$ where $s_{i,T}$ denotes the state of the previous price changes of asset i at time T. Namely,

$$s_{i,T} = \sum_{t=T-n}^{t=T} \gamma^t y_{i,t}$$

where $y_{i,t}$ is the ith element of the previously defined price relative vector y_t, γ is a discount factor and n is the size of the history we consider. As such the state is a vector of n-step backward relative price changes for each asset discounted by a rate gamma.

Thus, based on the environment and problem definition, we have

$$p(s_{t+1}|s_t, a) := Pr[S_{t+1} = s_{t+1}|S_t = s_t, A_t = a] = 1 \; \forall a \in A$$

for the deterministic sequence of states

$$S_1, S_1, ..., S_{T-n}$$

which follows since the sequence $y_1, y_2, ..., y_T$ is determined by the data.

For the sake of simplicity, an additional assumption that there are no transaction fees is added. Thus the focus turns to wether or not the agent is able to learn the optimal re-weighting of the portfolio given the state at each time, i.e. the agent must identify promising assets. The question of transaction fees and practicality will be judged in a seperate analysis.

Lastly, since we let the agent run through all the available data when conducting an experiment, the problem becomes an episodic task where $t = 0, ..., T$ and T is the number of trading periods in our data set. In practice however, the problem is a continuing task since the agent is tasked to a manage a portfolio forever.

2 Implementations and Analyses

Analysis goes here

2.1 Method 1

2.1.1 Implementation

Method 1 is an ad-hoc method based on the contextual bandits problem. The portfolio optimization problem is reframed as a k-armed bandits problem where we have one arm for each asset in the portfolio (including cash). The action preference method is used, where

$$H_t(a),\ a = 1, ..., k$$

are the action preferences for all k assets. However, a slight modification is done when defining actions and updating the preference functions. When computing probabilities of chosing a given action, we define $\pi_t(a)$ in the standard way, i.e.

$$\pi_t(a) := Pr[A_t = a | R_1, ..., R_{t-1}] = \frac{e^{H_t(a)S_{a,t}}}{\sum_{b=1}^{k} e^{H_t(b)S_{b,t}}},\ a = 1, ..., k$$

where the previously defined state vector S_t is included due to the contextual nature of the problem. The n-size history of discounted price changes represented by S_t will thus change the preferences of each asset based on whether or not their prices are observed to have declined or increased in the past n periods (uptrends increase preference, and downtrends decrease preference). Following the standard contextual bandits framework, this leads to the action at time t of

$$A_t = \arg\max_a \pi_t(a)$$

In the form of a portfolio weight vector, $A_t = w_t$ is a k-length vector where the A_tth co-ordinate is 1 and all other co-ordinates are 0. This means that rewards are

$$R_{t+1} := y_{t+1} \cdot A_t = y_{t+1,A_t}$$

i.e., the relative price change of asset A_t. The problem then reduces to a simple contextual bandits problem where one and only one arm is pulled by the agent at each time steps.

However, in the ad-hoc method the actual action taken by the agent at time t is the vector of preference values, i.e.

$$A'_t = (H_t(1), H_t(2), ..., H_t(k))^T$$

Since all $H_t(k)$'s sum to 1, this is a valid portfolio weight vector, and we can set $A'_t = w_t$. The rewards are then computed as

$$R'_{t+1} := y_{t+1} \cdot A'_t$$

and the preference functions are updated based on R'_t as follows:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R'_t - \bar{R}_t)(a - \pi_t(A_t))$$

$$H_{t+1}(a) = H_t(a) - \alpha(R'_t - \bar{R}_t)\pi_t(a),\ a \neq A_t$$

Intuitively, this method relies on the simplfying assumption that rewards produced by actions A'_t is a good enough approximation of rewards produced by actions A_t. In other words,

$$E[R'_t] \approx E[R_t]$$

In reality however, we can predict that this is far from the case (hence the ad-hoc nature of this method). It is safe to assume that computing the returns of a preference vector of assets is not the same as computing the total one-step return of the single most preferred asset. Nevertheless, the ad-hoc method provides a straightforward way to implement an algorithm learning to optimize a portfolio of assets since the softmax produces valid actions that represent the preference of the agent at time t based on S_t. However, the actual rewards used to update the preference function is only a weak approximation of the actual reward R_t.

The function used to simulate a single episode of method 1 is `simulate_contextual1` and the parameters are as follows:

- `n_curren`: The number of currencies in the problem including cash (5 for this project)
- `n_steps`: The number of timesteps T in the episode. To use all available data set `n_steps = nrow(data)`
- `alpha`: Learning rate α when updating action preference function (as defined above)
- `window_size`: Size of historic prices to consider when computing S_t (`window_size` corresponds to the value of n)
- `discount`: Discount factor of previous relative price changes when computing S_t (`discount` corresponds to the value of γ)

2.1.2 Analysis

Analysis of method 1

2.2 Method 2

2.2.1 Implementation

Implementation of method 2

2.2.2 Analysis

Analysis of method 2

2.3 Method 3

2.3.1 Implementation

Method 3 (dubbed the "all-or-nothing" method" is the original contextual bandits with action preference function method outlined in the section for method 1. We use the softmax and the previously defined state vector S_t to form our action probabilities

$$\pi_t(a) := Pr[A_t = a|R_1, ..., R_{t-1}] = \frac{e^{H_t(a)S_{a,t}}}{\sum_{b=1}^{k} e^{H_t(b)S_{b,t}}},\ a = 1, ..., k$$

The action taken by the agent is then

$$A_t = \arg\max_a \pi_t(a)$$

More formally, when computing the reward, the action A_t becomes the standard basis vector w_t where

$$w_t = e_{A_t}$$

and the vectors e are defined as

$$e_1 := (1,0,0,0,0)^T,\ e_2 := (0,1,0,0,0)^T,\ e_3 := (0,0,1,0,0)^T,\ e_4 := (0,0,0,1,0)^T, e_5 := (0,0,0,0,1)^T$$

when $k = 5$. This leads to the aforementioned definition of the reward,

$$R_{t+1} := y_{t+1} \cdot A_t = y_{t+1,A_t}$$

which is the relative price change of asset A_t and the preference can now be correctly updated as

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(a - \pi_t(A_t))$$

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a),\ a \neq A_t$$

The all-or-nothing method is essentially the simplest algorithm for tackling the portfolio optimization problem, since it is equivalent to the k-armed contextual bandits framework where we can assume rewards for each arm are non-stationary and we only pull one and one arm only at each time step. In practice, this means that the agent reallocates all the capital of the portfolio into one currency during each trading period based on its preference function, which can lead to extremely volatile rewards as well as large transaction costs (if such fees were considered). In addition, since the we assume non-stationarity, an epsilon greedy option is available (and recommended).

The function used to simulate a single episode of method 3 is `simulate_contextual3` and the parameters are as follows:

- `n_curren`: The number of currencies in the problem including cash (5 for this project)
- `n_steps`: The number of timesteps T in the episode. To use all available data set `n_steps = nrow(data)`
- `alpha`: Learning rate α when updating action preference function (as defined above)

- `window_size`: Size of historic prices to consider when computing S_t (`window_size` corresponds to the value of n)
- `discount`: Discount factor of previous relative price changes when computing S_t (`discount` corresponds to the value of γ)
- `epsilon`: ϵ value (for epsilon greedy)

2.3.2 Analysis

Analysis of method 3

3 Conclusion

Conclusion goes here

4 References

References go here