Deep Reinforcement Learning for financial portfolio management

RL Project

Adil Rhoulam

MVA student
Promo 2017

Contents

Overview

- **Financial Portfolio management** is the art of wise investment of assets, buying or selling them, in order to maximize profit while restrain the risk.
- It's all about you can making profitable actions that **balance risk against performance**.
- **Financial Portfolios are hard to manage**: capital market returns which are hard to predict in the short term future, unexpected decisions and moves by shareholders and many other impact factors...
- **Traditional techniques** : Follow-The-Winner (Universal Portfolios, Exponential Gradient...), Follow-The-Loser (Online Moving Average Reversion, Passive Aggressive Mean Reversion...)

Overview

- Deep RL is making remarkable achievements such as in video games, board games.
- These RL problems have **discrete action** spaces unlike **financial portfolio management where the actions are continuous**.
- Discretization of market actions have a lot of drawbacks: don't approximate the true risk, not scalable with the number of assets...
- **Actor-critic Deterministic Policy Gradient Algorithms** are general-purpose continuous deep RL framework: two deep neural network are trained to approximate the policy and the reward function, however it's **difficult and sometimes unstable**

The proposed framework: EIIE

- - The proposed framework is **designed specially** to financial portfolio
- - The core of the framework is **the Ensemble of Identical Independent Evaluators (EIIE) topology**
- - An IIE is a **neural network** whose job is to **inspect the history of an asset and evaluate its potential growth for the immediate future.**
- - Financial portfolio is a **set of** m **assets**, each asset has **its own weight**, the change in weights what we should predict, then **the output our network will be a softmax layer whose outcome will be the new portfolio weights**.

Assumptions and notations

- \- **Trading Period T** : the investment decisions and actions are made periodically not continually.
  \- **First asset will be cash**, so $as_1 =$ Amount of cash in our Portfolio
  \- Each period contains four important instants that correspond to the **opening**, **lowest**, **highest** and **closing** prices of the portfolio
- \- $\mathbf{v_t}$: Closing unit price vector of all asset, so ${v_t}^{(i)}$ corresponds to the price in cash of one unit of the asset $as_{(t+1,i)}$.
  \- $\mathbf{v_{t,h}}$, $\mathbf{v_{t,l}}$: The highest and lowest price vector of the period t respectively
  \- $\mathbf{y_t} = \mathbf{v_t} \ / \ \mathbf{v_{t-1}}$: The price relative vector of the t^{th} trading period
  \- p_t : The Portfolio value at the end of the Period t.
  \- $\mathbf{w_{t-1}}$: The portfolio weight vector at the beginning of the period t
  \- $r_t = \log \frac{p_t}{p_{t-1}}$: The logarithmic rate of return of the Period t

Mathematical Formalism: Not taking into account different costs

- \- $\mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$ is the quantity that control the change of the portfolio value and we have : - $p_t = p_{t-1}\ \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$
- \- $r_t = \log \frac{p_t}{p_{t-1}} = \log \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$
- \- The final value of the Portfolio p_f is : $p_f = p_{t_f} = p_0 \exp(\sum_{t=1}^{t_f} r_t) = p_0 \prod_{t=1}^{t_f} \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$
- \- The goal of the portfolio manager is to maximize the value p_f.

Mathematical Formalism: taking into account different costs

- - Buying or selling actions is not free, we have to pay the **transactions costs** and also the **commission fees**.
- - we will consider that **commissions fees are constant** and **transactions costs are variables** within each period
- - $p_t = \nu_t p'_t$ where p'_t is **the value of the Portfolio at the end of the period** t and p_t is **the value of it at the beginning of period** $t+1$ and ν_t is **transaction reminder coefficient** $\in (0,1]$
- - $r_t = \log \frac{p_t}{p_{t-1}} = \log \frac{\nu_t p'_t}{p_{t-1}} = \log \nu_t \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$
  - $p_f = p_0 \exp(\sum_{t=1}^{t_f} r_t) = p_0 \prod_{t=1}^{t_f} \nu_t \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$

What's is the explicit value of ν_t

- - $\mathbf{w_t'} = \frac{\mathbf{y_t \circ w_{t-1}}}{\mathbf{y_t^T \cdot w_{t-1}}}$ where $\circ$ is the element-wise product is **the weight vector at the end of period t**.
- - $(1 - c_s)p_t' \sum_{i=2}^{m} \text{ReLu}(w\ {}_t'^{(i)} - \nu_t w_t^{(i)})$ is The total amount of cash obtained by all selling.
- - We add this money to the cash that we have $p_t' w\ {}_t'^{(1)}$ to buy new assets:
  $(1\text{-}c_s)p_t' \sum_{i=2}^{m} \text{ReLu}(w\ {}_t'^{(i)} - \nu_t w_t^{(i)}) + p_t' w\ {}_t'^{(1)}$ - $p_t w_t^{(1)} = (1 - c_p)p_t'$ $\sum_{i=2}^{m} \text{ReLu}(\nu_t w_t^{(i)} - w\ {}_t'^{(i)})$

What's is the explicit value of ν_t

after simplifying the last formula we get:

$$\nu_t = \frac{1}{1-c_p w_t^{(1)}} \left(1 - c_p w\ {}'^{(1)}_t - (c_s + c_p - c_s c_p) \sum_{i=2}^m \mathrm{ReLu}(w\ {}'^{(i)}_t - \nu_t w_t^{(i)})\right)$$

- ν_t has **no explicit expression**, define $f_t(\nu)$ such that:

$$f_t(\nu) = \frac{1}{1-c_p w_t^{(1)}} \left(1 - c_p w\ {}'^{(1)}_t - (c_s + c_p - c_s c_p) \sum_{i=2}^m \mathrm{ReLu}(w\ {}'^{(i)}_t - \nu w_t^{(i)})\right)$$

- it's shown that this function **admits a fixed point**.

- In practice, we approximate ν_t by a sequence $\nu_{t,k}$ such that:

$$\nu_{t,k+1} = \mathrm{f}_t(\nu_{t,k}) \text{ and } |\nu_{t,k+1} - \nu_{t,k}| < \epsilon.$$

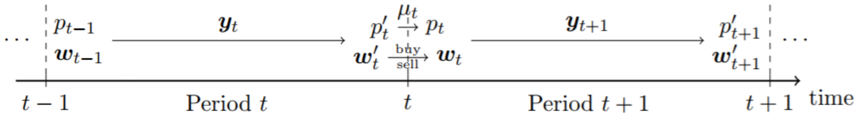

Figure: Portfolio pipeline when costs are taken into account

The Reinforcement Learning framework for the financial Portfolio Management Problem

- -**The agent** : is the Portfolio Manager who takes actions at the end of each Period, given w'_t the agent comes up, w.r.t some policy, with the portfolio vector w_t.
- -**The environment** : defines the space of actions, states and returns back a reward. It's the financial market.
  Problems: The financial market is very large and complex (contains all the available assets and the actions of all the agents...).
  A possible solution: Define it as the market's history of the prices of all orders up to the position where the state is at. It's intractable for the software agent $\Rightarrow$ sub-sampling to avoid processing all the history of prices.

Sub-sampling methods used

- **Periodic features extraction** : It's the discretizes of time into periods that we have discussed previously, and then the extraction of the highest, lowest and closing prices from each period.
- **Asset pre-selection** : Since the number of assets in a market history is very large, we will select only the most-volumed non-cash assets (which have a rich history) to represent our Portfolio.
- **History cut-off** : To represent the current state of the environment, we take the price-features history of only recent number of periods.

Price Tensor

- \- It will be the input feed to our network to predict the ending weight of a period t $\mathbf{w'_t}$. its shape is *(f,n,m-1)*
  \- f : is the number of relevant prices (High, low, closing).
  \- *m-1* : The number of pre-selected non-cash assets.
  \- n : the history length of periods taking into account to present the current state; with T = 30 min $\Rightarrow$ n = 50*30 $\approx$ 1 day and an hour.
- \- Since only the changes in prices within periods that contains relevant informations we define it as: $\mathbf{X_t} = \begin{pmatrix} \mathbf{Y_t} , \mathbf{Y_{t,h}} , \mathbf{Y_{t,l}} \end{pmatrix}^{\mathbf{T}}$ where

$$\mathbf{Y_t} = \left[\mathbf{v_{t-n+1}}/\mathbf{v_t} \middle| \mathbf{v_{t-n+2}}/\mathbf{v_t} \middle| \ldots \middle| \mathbf{v_{t-1}}/\mathbf{v_t} \middle| \mathbf{1} \right]$$

- \- At the end of the period t, the neural network comes with a vector $\mathbf{w'_t}$ using the informations stored in $\mathbf{X_t}$ and $\mathbf{w_{t-1}}$, after the trades-operations the agent comes with a vector $\mathbf{w_t}$ according to some policy π such that $\mathbf{w_t} = \pi(\mathbf{X_t}, \mathbf{w_{t-1}})$ by equivalence.

Actions, states and rewards

- -**Actions** : The action of the agent at Period t can be represented by the portfolio vector $\mathbf{w_t}$ such that : $\mathbf{a_t} = \mathbf{w_t} = \pi(\mathbf{w'_t})$
- -**States** : The state at time t can be represented by the pair $(\mathbf{X_t}, \mathbf{w_{t-1}})$, then : $\mathbf{s_t} = (\mathbf{X_t}, \mathbf{w_{t-1}})$
- -**Rewards** : The reward could be represented by the *logarithmic rate of return* defined previously by: $r_t = \log \frac{p_t}{p_{t-1}} = \log \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$
- -**Reward function** : $R(\mathbf{s_1}, \mathbf{a_1}, ..., \mathbf{s_{t_f-1}}, \mathbf{a_{t_f-1}}, \mathbf{s_{t_f}}) = \frac{1}{t_f} \log \frac{p_f}{p_0} = \frac{1}{t_f} \sum_{t=1}^{t_f} \log \nu_t \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}) = \frac{1}{t_f} \sum_{t=1}^{t_f} r_t$: R is then the average sum of the rewards of each episode (Period), Remark that we have taken a discount factor equal to 1 since all the reward have the same importance

Policy

- - **Policy** : **fully-exploitation** policy since we want to maximize the reward. **Deterministic policy** to avoid some stochastic decisions.
  - To find the optimal policy we use **the gradient ascent algorithm**.

- - **The performance metric** of the policy $\pi_{\mathbf{\Theta}}$ where $\mathbf{\Theta}$ is a set of parameters controlling the policy, on the time interval $[0, t_f]$ by:
  $J_{[0,t_f]}(\pi_{\mathbf{\Theta}}) = \mathrm{R}(\mathbf{s_1},\ \pi_{\mathbf{\Theta}}(\mathbf{s_1}),\ldots,\mathbf{s_{t_f-1}},\pi_{\mathbf{\Theta}}(\mathbf{s_{t_f-1}}),\mathbf{s_{t_f}})$ Where $\mathbf{a_t} = \pi_{\mathbf{\Theta}}(\mathbf{s_t})$.

- - $\mathbf{\Theta_{t+1}} = \mathbf{\Theta_t} + \Delta_t(\nabla_{\Theta} J_{[0,t_f]}(\pi_{\mathbf{\Theta}})|_{\mathbf{\Theta}=\mathbf{\Theta_t}})$

- - In practice, to avoid gradient vanishing and machine calculation errors, we use a mini-batch Gradient ascent where we update on a limited time-range $[t_{b_1}, t_{b_2}]$ instead of on the whole time-range $[0, t_f]$.

Learning the policy

- - The optimal policy will be learned using deep neural networks models (CNN, RNN or LSTM)

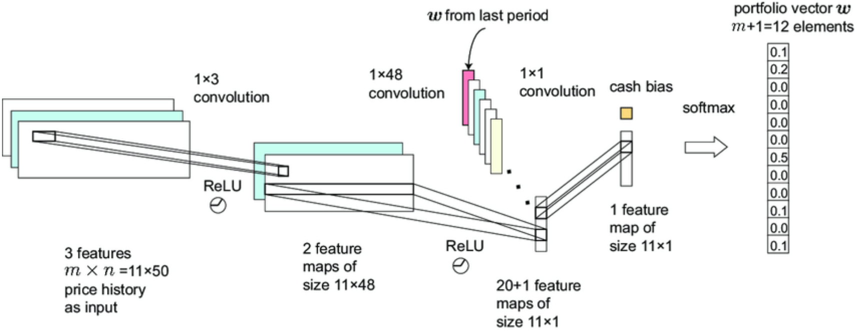

Figure: CNN implementation to learn the policy

Online Stochastic learning

- - we are facing time series, the **chronological order of the data is important**
- - Data generated in a time range is huge, dividing it into mini-batches and training over all mini-batches is intractable
- - Use of Stochastic learning by choosing N_b random batches
- - The correlation between two market price event decreases exponentially with the temporal distance between them. Prefer recent batches $\Rightarrow$ use a geometric distribution for batch selection:
  For a batch starting at period t_b, it's necessary that $t_b \leq t - n_b$, the probability of its selection is : $\mathrm{P}(t_b) = \beta(1-\beta)^{t-t_b-n_b}$ where $\beta \in (0,1)$

Experiments

- Data used : cryptocurrencies (virtual money), cash will be Bitcoin and non-cash assets will be other virtual moneys : (Ethereum for instance), $T = 30min$, $c_p = c_s = 0.25\%$. Other hyperparameters are cross validated such as β, regularization parameters...
- Compared methods; - **Follow-The-Loser methods** :
  Online Moving Average Reversion (OLMAR) :
  initialization : $w_0 = \frac{1}{m}\mathbf{1}$
  Iteration : $\mathbf{y_{t+1}} = \frac{1}{w}\left(1 + \frac{1}{\mathbf{y_t}} + \frac{1}{\mathbf{y_t y_{t-1}}} + ... + \frac{1}{\prod_{k=1}^{w-2}\mathbf{y_{t-k}}}\right)$
  $$\mathbf{w_{t+1}} = argmin_{\mathbf{w}\in\Delta_m} ||\mathbf{w} - \mathbf{a_t}||$$
  where $\mathbf{a_t} = \mathbf{w_t} + \lambda_{t+1}(\mathbf{y_{t+1}}$ - $s_{t+1}\mathbf{1})$; $\lambda_{t+1} = \max\left(0, \frac{\epsilon - \mathbf{w_t^T}\cdot\mathbf{y_{t+1}}}{||\mathbf{y_{t+1}} - s_{t+1}\mathbf{1}||}\right)$
  and $s_{t+1} = \frac{\mathbf{1^T}\cdot\mathbf{y_{t+1}}}{m}$

Experiments

- **Passive Aggressive Mean Reversion** (PAMR) :
  initialization : $w_0 = \frac{1}{m}\mathbf{1}$
  Iteration : $\mathbf{w_{t+1}} = argmin_{\mathbf{w}\in\Delta_m} ||\mathbf{w} - \mathbf{a_t}||$
  where $\mathbf{a_t} = \mathbf{w_t}$ - $\tau_t(\mathbf{y_t}$ - $s_t\mathbf{1})$; $\tau_t = \frac{l_\epsilon^t}{||\mathbf{y_t} - s_t\mathbf{1}||^2}$ (PAMR) or $\min\left(C, \frac{l_\epsilon^t}{||\mathbf{y_t} - s_t\mathbf{1}||^2}\right)$ (PAMR-1) or $\frac{l_\epsilon^t}{||\mathbf{y_t} - s_t\mathbf{1}||^2 + \frac{1}{2C}}$ (PAMR-2) and $l_\epsilon^t = max\left(0, \mathbf{w_t^T} \cdot \mathbf{y_t} - \mathbf{1}\right)$
- **Follow-The-winner methods** :
  Exponential Gradient (EG) : the update formula for the weight is given by :

$$w_{t+1}^i = w_t^i \exp\left(\frac{\eta y_t^i}{\mathbf{y_t^T} \cdot \mathbf{w_t}}\right) \Big/ \sum_{j=1}^m w_t^j \exp\left(\frac{\eta y_t^j}{\mathbf{y_t^T} \cdot \mathbf{w_t}}\right)$$

Experiments

- **Universal Portfolios** (UP) : the update formula for the weight vector is given by :
$$w_{t+1} = \int wS_t(w)dw \Big/ \int S_t(w)dw$$
where $S_t(w) = \prod_{k=0}^{t} \mathbf{y_k^T} \cdot \mathbf{w_k}$
- - The starting date of the uploaded data is 2015/07/01 and the ending date is 2017/07/01, the proportion of test data is 8% which is about the last two months and gives about 2775 periods of 30 minutes.

Experiments

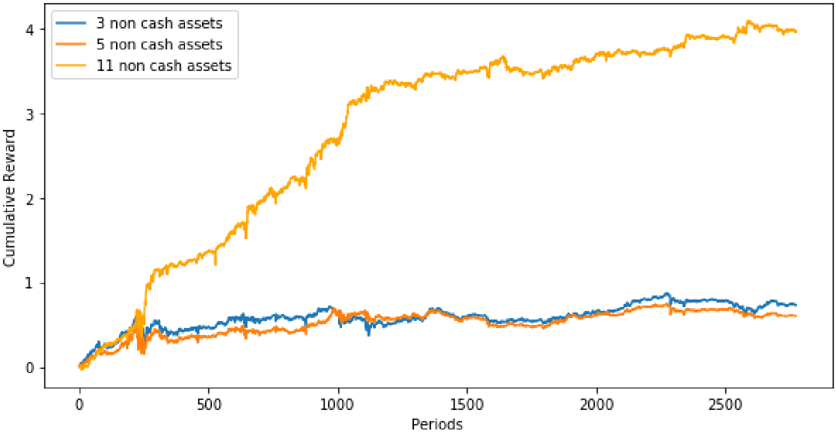

Figure: Cumulative reward corresponding to 3 Portfolios with 3, 5 and 11 non-cash assets

Experiments

Number of assets	4	6	12
training time	19min	38 min	1h32min

Table: the training time scales roughly linearly with number of assets

Experiments

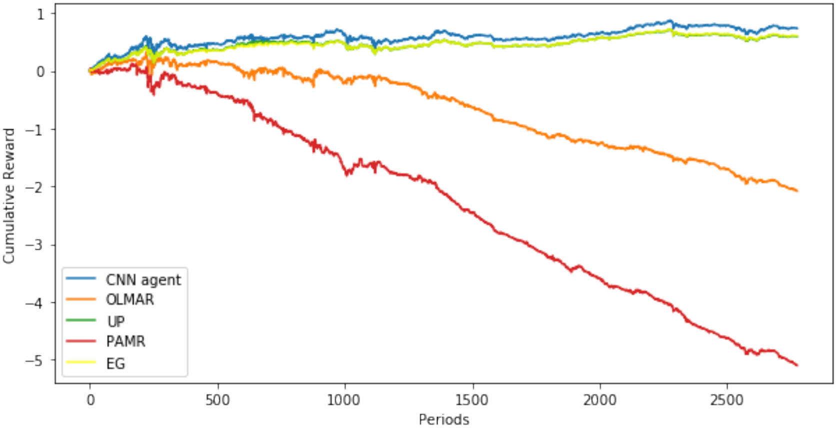

Figure: Comparison of the cumulative reward given by the CNN agent, OLMAR, PAMR, UP and EG algorithms using 3 non-cash assets.

Experiments

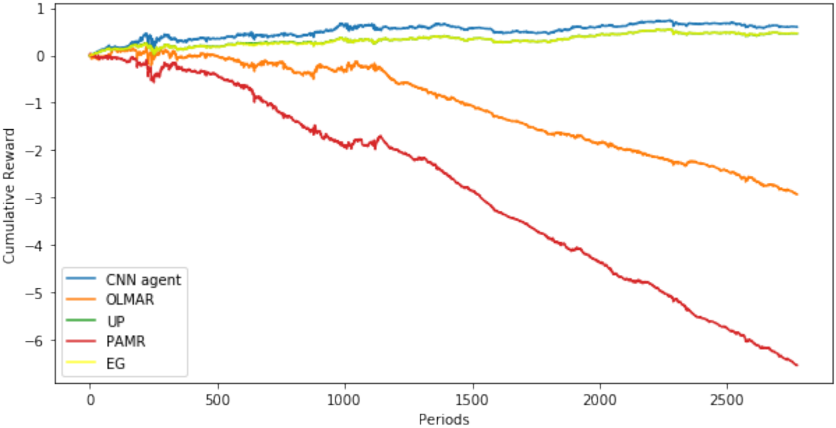

Figure: Comparison of the cumulative reward given by the CNN agent, OLMAR, PAMR, UP and EG algorithms using 5 non-cash assets.

Experiments

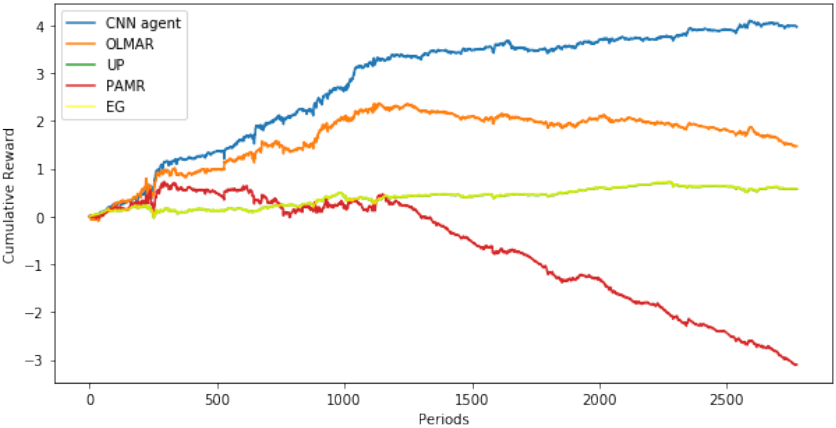

Figure: Comparison of the cumulative reward given by the CNN agent, OLMAR, PAMR, UP and EG algorithms using 11 non-cash assets.

Thank you