Report of Reinforcement Learning Project Subject: Financial Portfolio Management using deep RL techniques

Author : Adil Rhoulam
MVA student
adil.rhoulam@telecom-paristech.fr

Abstract

In this small research project, I studied a deep Reinforcement Learning framework applied to finance to face the problem of Portfolio Management. In this report, I first give an overview of the financial Portfolio Management problem and why it needs a Machine Learning framework. Then I explain the mathematical formalism along with the required hypotheses. Afterward, I present the Reinforcement Learning formulation of the problem. Finally, I will present the experiments and discuss the results then compare the performances of this framework with other existing algorithms.

1 Presentation

Financial portfolio management is the art of wise investment of assets, buying or selling, in order to maximize profit while restrain the risk. In fact, Portfolio management is all about determining the opportunities and threats which, based on them, you can make profitable decisions that balance risk against performance. There exists some traditional techniques dedicated to manage the portfolio. For instance, "Follow-The-Winner" and "Follow-The-Loser" are methods based on models that are constructed on prior (doesn't need to interact with the flow of the data to learn the model as like Machine or Reinforcement Learning techniques do), the performance of those two techniques depends highly on the model used. They may be assisted with machine learning optimization techniques for parameter tuning, note that those methods are not fully machine learning.

There exists a lot of factors that make the financial portfolio hard to manage, such as the capital market returns which are hard to predict in the short term future, unexpected decisions and moves by shareholders and many other impact factors. This mean that even with using powerful machine learning techniques in order to predict future trends, prices of assets or market actions can mislead to very poor decisions. However, their use still very important since it gives us an overview about the market.

Recently, a lot of successful fully machine learning models applied to algorithmic trading problems were developed and that treat those problems as a Reinforcement Learning ones. In this paper, we will explore a much deeper framework which is the fruit of the combination between the Deep Learning and Reinforcement Learning techniques.

2 Model Overview

In this project, the RL model used is specially dedicated for the task of financial portfolio management. It's based essentially on the EIIE topology (Ensemble of Identical Independent Evaluators) which is

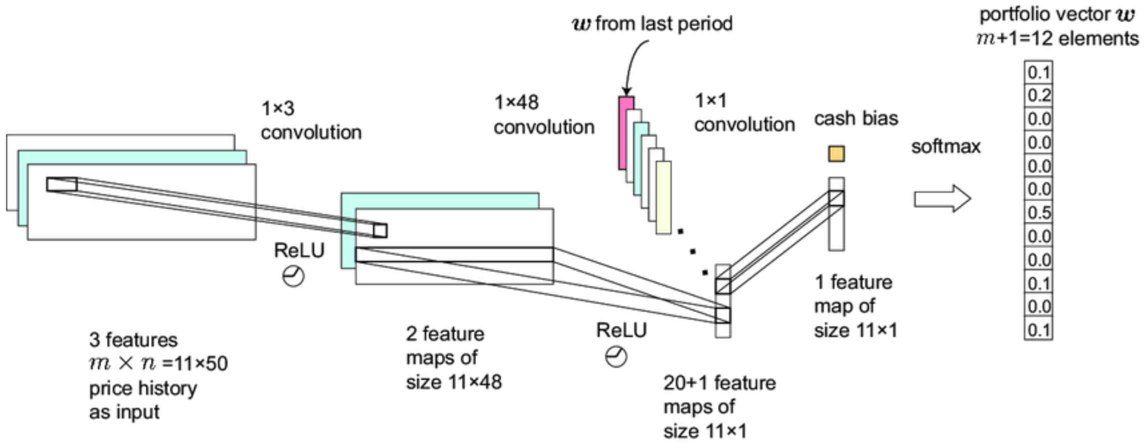

FIGURE 1 – CNN implementation of the EIIE where we see that the last layer is a softmax layer that gives the weights of the next period and then stored in the PVM (Portfolio Vector Memory), further explanation about the other layers will be given shortly

based on a neural network (CNN or RNN supported by an LSTM) that is responsible of inspecting the history of an asset and evaluate its changes in the short term future. In fact, a portfolio is a combination of assets, the importance of an asset in the portfolio is determined by its weight, those weights are changing continuously in time duo to different impact factors mentioned previously. Automatically, the output of the neural network should be a softmax layer that gives the new values of the weights for the next coming trading period (The period will be defined afterward). Portfolio's weights are also inputs of the neural network, we feed the network with those corresponding to the current trading period and we collect the ones for the next trading period. The history of weights will be stored in a vector (Portfolio Vector Memory) for next use which will be detailed next. The Figure[1] gives an overview of the EIIE implemented as a Convolutional Neural Network, further explanations of this architecture will be given along the report.

3 Mathematical Formalism of the financial Portfolio Management Problem

3.1 The periodical paradigm

As mentioned before, the change of the assets' weights in reality is continuous. However, for a computer, we need time to let it learn the environment (different factors causing the change) to give more precise values of the weights. For this reason, we will choose periodical change of weights, the period will be denoted by T. It is important to mention that each period contains four important instants that correspond to the opening, lowest, highest and closing prices of the portfolio.

3.2 Notations

Let P^t denotes the portfolio at the opening of the Period t, it's a set of m assets : $P^t = (as_{(t,1)}, as_{(t,2)}, ..., as_{(t,m)})^T$. Here are some useful notations that we're going to use from now on :

— $\mathbf{v_t}$: 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. It's simply the quotient of the closing price vector and the opening price vector of the t^{th} period. Where / is the point-wise division.

— p_t : The Portfolio value at the end of the Period t. If we ignore different costs (commissions fees, transactions cost...) of operations (Buying, selling assets) that the agent did between periods, it's also the opening value of the portfolio at Period $t+1$, in such case, p_t is given by :

$$p_t = \mathbf{v_{t+1}^T} \cdot \mathbf{P^{t+1}} \text{ for } t \geq 1$$

— $\mathbf{w_{t-1}}$: The portfolio weight vector at the beginning of the period t (called *portfolio vector*), it has the constraint that $\sum_{i=1}^{m} w_{t-1}^{(i)} = 1$. The weight of an asset is simply its total value divided by the portfolio value.

— $\rho_t = \frac{p_t}{p_{t-1}} - 1$: The rate of return of the Period t.

— $r_t = \log \frac{p_t}{p_{t-1}}$: The logarithmic rate of return of the Period t.

3.3 Formulation of the problem without taking into account the different costs between periods

Since the measurement of the value of the assets is cash, we will put as our first asset the amount of cash that we have a_1 = Amount of cash in our Portfolio, then ${v_t}^{(1)}$, ${v_{t,h}}^{(1)}$ and ${v_{t,l}}^{(1)}$ are equal to 1. Explicitly, $\mathbf{y_t} = (1, \frac{{v_t}^{(2)}}{{v_t}^{(2)}}, \frac{{v_t}^{(3)}}{{v_t}^{(3)}}, ..., \frac{{v_t}^{(m)}}{{v_t}^{(m)}})^T$, it's straightforward to see that the dot product $\mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$ is the quantity that control the change of the portfolio value in a Period since it's the sum of the product between the rate of change in price of each asset and its weight, then the value of the Portfolio at the end of the Period t, if we ignore the all costs i.e Transaction costs for instance, is given by the formula : $p_t = p_{t-1}\, \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$. Then the logarithmic rate of return and the rate of return are :

$$\rho_t = \frac{p_t}{p_{t-1}} - 1 = \mathbf{y_t^T} \cdot \mathbf{w_{t-1}} - 1$$

$$r_t = \log \frac{p_t}{p_{t-1}} = \log \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$$

The final value of the Portfolio p_f is then given by :

$$\prod_{t=1}^{t_f} \frac{p_t}{p_{t-1}} = \prod_{t=1}^{t_f} \exp(r_t) \Rightarrow p_f = p_{t_f} = p_0 \exp(\sum_{t=1}^{t_f} r_t) = p_0 \prod_{t=1}^{t_f} \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}.$$

p_0 and $\mathbf{w_0}$ are the initial values of the portfolio and of the weight vector respectively. $\mathbf{w_0}$ is usually initialized by $(1, 0, 0, .., 0)^T$ which often means that at the beginning our Portfolio contains only cash (we didn't begin the investments and trading operations). The goal of the portfolio manager is to maximize the value p_f.

3.4 Taking into account the different costs between periods

As mentioned before, buying or selling actions is not free, we have to pay the costs of transactions and also the commission fees, for instance when trading with Bitcoin, the plateforme that manages the transactions get for each transaction a pourcentage of cash flowing. In this part, we will consider that commissions fees is constant and transactions cost are variables within each period. Paying transaction costs comes back to multiply the portfolio value by a *transaction reminder coefficient* ν_t that belongs to $(0, 1]$, this means that those transactions shrink the value of the Portfolio. Eventually $p_t = \nu_t p'_t$ 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 $t+1$.
The new expressions of the rate of return and the logarithmic rate of return are now :

$$\rho_t = \frac{p_t}{p_{t-1}} - 1 = \frac{\nu_t p'_t}{p_{t-1}} - 1 = \nu_t \mathbf{y_t^T} \cdot \mathbf{w_{t-1}} - 1$$

$$r_t = \log \frac{p_t}{p_{t-1}} = \log \frac{\nu_t p'_t}{p_{t-1}} = \log \nu_t \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$$

As before by taking the product of the fraction $\frac{p_t}{p_{t-1}}$ between the initial Period and the final one t_f, we get :

$$p_f = p_0 \exp(\sum_{t=1}^{t_f} r_t) = p_0 \prod_{t=1}^{t_f} \nu_t \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}.$$

What's the explicit expression of the *transaction reminder coefficient* ν_t ?

If $\mathbf{w'_t}$ is the assets' weight vector at the end of the Period t, then it's given by :

$$\mathbf{w'_t} = \frac{\mathbf{y_t} \circ \mathbf{w_{t-1}}}{\mathbf{y_t^T} \cdot \mathbf{w_{t-1}}} \qquad (*)$$

where $\circ$ is the Hadamard product (element-wise product). So by selling and buying relevant assets, the Portfolio manager reallocates the portfolio weight vector from w'_t to w_t. By doing those trades, all or an amount of the asset need to be sold, this implies that the total value of the asset after the operations has decreased. Mathematically, this is equivalent to write :

$$w\,'^{(i)}_t p'_t \geq w^{(i)}_t p_t \Rightarrow w\,'^{(i)}_t \geq \nu_t w^{(i)}_t$$

After finishing the operation of selling assets, the amount of cash that we obtained is :

$$(1-c_s)p'_t \sum_{i=2}^m \text{ReLu}(w\,'^{(i)}_t - \nu_t w^{(i)}_t)$$

Where $0 \leq c_s < 1$ is the commission cost for selling and ReLu(x) = x if x ≥ 0 ; 0 elsewhere. We add this money to the cash that we have $p'_t w\,'^{(1)}_t$ to buy new assets, we get the next formula :

$$(1-c_s)p'_t \sum_{i=2}^m \text{ReLu}(w\,'^{(i)}_t - \nu_t w^{(i)}_t) + p'_t w\,'^{(1)}_t - p_t w^{(1)}_t = (1-c_p)p'_t \sum_{i=2}^m \text{ReLu}(\nu_t w^{(i)}_t - w\,'^{(i)}_t)$$

Where the left term is the amount of cash that we have spent to buy the new assets and the right one is the amount of assets that we have bought multiplied by the $(1-c_p)$ where $0 \leq c_p < 1$ is the commission rate for purchasing, after dividing the two terms by p'_t and using the fact that $p_t = \nu_t p'_t$ and $\sum_{i=2}^m w^{(i)}_t + w^{(1)}_t = 1$ and $\sum_{i=2}^m w\,'^{(i)}_t + w\,'^{(1)}_t = 1$, we get the following expression :

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

We see that inside the ReLu function we still have ν_t which means that the *transaction reminder coefficient* don't have an explicit expression. Then, we should consider the function :

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

and seek for a fixed point as we have done with the Bellman Operator in the RL courses, indeed it admits a unique fixed point ν which is the limit of the sequence $\nu_{(t,n+1)} = f_t(\nu_{(t,n)})$ with $\nu_{(t,0)}$ in (0,1], the demonstration could be found in [1] reference. When $c_p = c_s = c$ there is a practice (Moody et al., 1998) to approximate ν_t with c$\sum_{i=2}^m |w\,'^{(i)}_t - w^{(i)}_t|$. In practice, this value is used to initialize the sequence : $\nu_{(t,0)}$ = c$\sum_{i=2}^m |w\,'^{(i)}_t - w^{(i)}_t|$. In general ν_t is a function of $\mathbf{w_t}$ and $\mathbf{w'_t}$ since $\mathbf{w'_t}$ depends on $\mathbf{w_{t-1}}$ and $\mathbf{y_t}$ then by transitivity $\nu_t = \nu_t(\mathbf{w_{t-1}},\mathbf{y_t},\mathbf{w_t})$.

4 The Reinforcement Learning framework for the Portfolio Management ProbLem

4.1 Definition of the environment and the agent

4.1.1 The agent

In the Portfolio management problem, the agent is the Portfolio Manager who takes decisions (trades-actions) at the end of each Period, his job is to maximize p_f for a given time frame, given w'_t and the other relevant informations that will be made clear next (the previous weight vector and the change in prices along the period), the agent comes up, w.r.t some policy, with the portfolio vector w_t which will be the one to feed the network by at the beginning of the Period t.

4.1.2 The environment

The environment defines the space of actions, states and returns back a reward which could be positive or negative. In the Portfolio management problem, the environment is the financial market.

The financial market is a large market in which different agents (Portfolio managers) trade with their assets at low transaction costs and at prices that reflect the supply and demand model. Naturally, this market is huge since it comprises all the available assets and the interactions of all the agents towards them. Duo to this large and complex environment, it's difficult even impossible for the agent to get complete information about a state. Moreover, not all the informations are accessible to the agent. However, the history of the assets' prices are open source for every agent and it's believed that the prices contains all relevant informations. The environment will then be defined by the prices of all orders throughout the market's history up to the position where the state is at. Nevertheless, it's intractable, for the computer (software agent), to process those informations duo to its largeness. One solution that could be proposed to avoid processing all the history of prices is to use sub-sampling methods of the most informative moments, this would simplify the representation of the state space in the environment. The sub-sampling methods that are used to sample from the financial market in this project are :

Periodic feature 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, their numbers would be m-1 following the notations that we have used.

History cut-off : To represent the current state of the environment, we take the price-features history of only recent number of periods (not all the history which would be intractable by the software agent).

After those sampling methods, the representation of the input of the neural network will be a tensor $\mathbf{X_t}$ called the *Price Tensor.*

Price Tensor It will be the input feed to our network to predict the ending weight of a period t $\mathbf{w'_t}$, it should then contain all the relevant informations about this period. Our tensor will have the shape *(f,n,m-1)* where *f* : is the number of relevant prices (High, low, closing), *m-1* : The number of pre-selected non-cash assets and *n* : the history length of periods taking into account to present the current state. Since only the changes in prices within periods that contains relevant informations, we will consider only the relative price vectors by normalizing over the last period closing price vector $\mathbf{v_t}$. Explicitly, we write :

$$\mathbf{X_t} = \left(\mathbf{Y_t}\ ,\mathbf{Y_{t,h}}\ ,\mathbf{Y_{t,l}} \right)^{\mathbf{T}} \text{ Where :}$$

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

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

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

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 The action of the agent at Period t can be represented by the portfolio vector $\mathbf{w_t}$:

$$\mathbf{a_t} = \mathbf{w_t} = \pi(w'_t)$$

Since it's the redistribution of weights that the agent do when taking decisions by selling-buying assets that modify the vector of weights from $\mathbf{w'_t}$ (the output of the neural network) to $\mathbf{w_t}$.

States The state at time t can be represented by the pair $(\mathbf{X_t}, \mathbf{w_{t-1}})$, since at the beginning of the state at period t we feed the network with the tensor $\mathbf{X_t}$ and the action that would be taken depends on $\mathbf{w'_t}$ that itself depends on $\mathbf{w_{t-1}}$ by the relation $(*)$, therefore, it's reasonable to consider $\mathbf{w_{t-1}}$ as a part of the state at period t, so :

$$\mathbf{s_t} = (\mathbf{X_t}, \mathbf{w_{t-1}})$$

Rewards The reward could be represented by the *logarithmic rate of return* defined previously by :

$$r_t == \log \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}$$

so when taking an action at the beginning of period t $(\mathbf{w_{t-1}})$, we get a reward at the end of this Period given by r_t, the reward could be positive or negative since it's also equal to $\log \frac{p_t}{p_{t-1}}$, then if our action has increased the Portfolio value at the end of period t the reward will be positive since $p_t \geq p_{t-1}$ and will be negative otherwise.

The Reward Function Two parameters that the agent doesn't control when trying to maximize the final portfolio value p_f. They are the final period t_f, which is also the length of the portfolio management process, and the initial value of the portfolio p_0. So it's equivalent and more efficient to maximize the cumulative reward function R defined by :

$$\text{R}(\mathbf{s_1}, \mathbf{a_1}, ..., \mathbf{s_{t_f-1}}, \mathbf{a_{t_f-1}}, \mathbf{s_{t_f}}) = \frac{1}{t_f}\log \frac{p_f}{p_0} = \frac{1}{t_f}\sum_{t=1}^{t_f}\log \nu_t \mathbf{y_t^T} \cdot \mathbf{w_{t-1}}) = \frac{1}{t_f}\sum_{t=1}^{t_f} r_t$$

R is then the average sum of the rewards of each episode (Period), where we have taken the average (divided by t_f) to guarantee the fairness of R between runs of different lengths. Remark that we have taken a discount factor equal to 1.

Policy The current RL model need a fully-exploitation policy since we want to maximize the reward by taking the actions that gives it. The policy needs then to be deterministic to avoid some stochastic decisions. In this framework, to find the optimal policy we use the gradient ascent algorithm (such as the one used in the final homework). Let define 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 :

$$J_{[0,t_f]}(\pi_{\mathbf{\Theta}}) = \text{R}(\mathbf{s_1}, \pi_{\mathbf{\Theta}}(\mathbf{s_1}), ..., \mathbf{s_{t_f-1}}, \pi_{\mathbf{\Theta}}(\mathbf{s_{t_f-1}}), \mathbf{s_{t_f}}) \text{ Where } \mathbf{a_t} = \pi_{\mathbf{\Theta}}(\mathbf{s_t})$$

The parameters are updated by the Gradient Ascent Formula :

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

Sometimes to avoid gradient vanishing and machine calculation errors, it's better to 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 $[0, t_f]$. Then the updating formula becomes :

$$\mathbf{\Theta_{t+1}} = \mathbf{\Theta_t} + \Delta_t(\nabla_{\Theta} J_{[t_{b_1}, t_{b_2}]}\big(\pi_{\mathbf{\Theta}})|_{\mathbf{\Theta}=\mathbf{\Theta_t}}\big)$$

where we re-affine J and R by :

$$J_{[t_{b_1}, t_{b_2}]}(\pi_{\mathbf{\Theta}}) = \text{R}(\mathbf{s_{t_{b_1}}}, \pi_{\mathbf{\Theta}}(\mathbf{s_{t_{b_1}}},), ..., \mathbf{s_{t_{b_2}-1}}, \pi_{\mathbf{\Theta}}(\mathbf{s_{t_{b_2}-1}}), \mathbf{s_{t_{b_2}}})$$

$$\text{R}(\mathbf{s_{t_{b_1}}}, \pi_{\mathbf{\Theta}}(\mathbf{s_{t_{b_1}}}), ..., \mathbf{s_{t_{b_2}-1}}, \pi_{\mathbf{\Theta}}(\mathbf{s_{t_{b_2}-1}}), \mathbf{s_{t_{b_2}}}) = \frac{1}{t_{b_2}-t_{b_2}}\sum_{t=t_{b_1}}^{t_{b_2}} r_t$$

4.2 Learning the Policy

The optimal policy will be learned using deep neural networks models. The deep neural network could be a CNN, RNN or a LSTM. The Figure[1] showed in the overview is a CNN implementation for solving the Portfolio management problem. As we have discussed in the previous sections, the input to the neural networks will be the price tensor $\mathbf{X_t}$ and the output will be the Portfolio weight vector $\mathbf{w_t}$. The last hidden layer in the Figure[1] is a softmax layer which output the scores (new weights) for all non-cash assets, a cash bias is added to construct the new portfolio weight vector $\mathbf{w_t}$. The transactions costs are taking into account by adding just before the softmax layer the portfolio weight vector from last period $\mathbf{w_{t-1}}$ as a feature map, so the agent could minimize the transactions costs by restraining from large changes between $\mathbf{w_{t-1}}$ and $\mathbf{w_t}$. The question is how we could remember the previous weight vector $\mathbf{w_{t-1}}$ till we add it in our network.

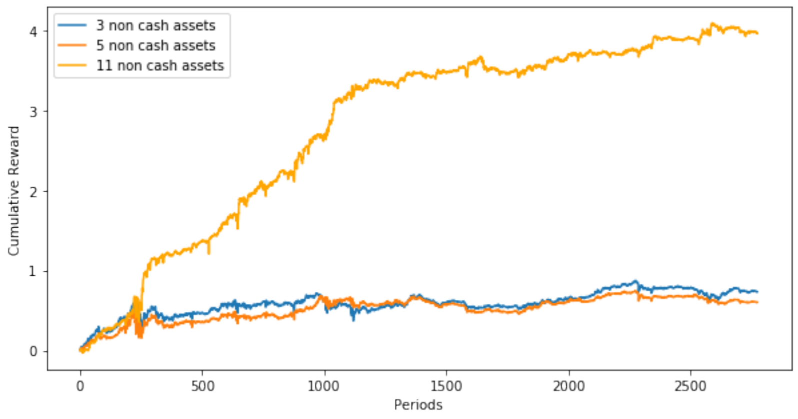

FIGURE 2 – Cumulative reward corresponding to 3 Portfolios with 3, 5 and 11 non-cash assets, i.e m = (4, 6 and 12). The starting date of the data uploaded is $2015/07/01$ and the ending date is $2017/07/01$, the proportion of test data is 8% which is about the last two months and gives about 2775 periods of 30 minutes.

PVM As mentioned in the introduction, to do this we need a Portfolio Vector Memory to store the network successive outputs. The PVM is a stack of the portfolio vectors in chronological order. It's initialized with uniform weights and along each training step, the network loads the portfolio weight vector of the previous period from the PVM at position $t-1$, and at the end of the training step, the output is written in the PVM at the t^{th} position.

The need of a Stochastic Online Learning algorithm In the Portfolio Management problem, we are facing time series data set (data flows along time in a chronological order). In fact, the chronological order of the data is important when learning the policy unlike a problem of supervised learning where the order of the data doesn't matter. Mini-batch learning is then reasonable for learning the policy on successive chronological batches even if there is an overlap between them as long as they respect the chronological order of their apparence. However, since the financial markets are very large, the amount of data generated in a range of time is big, then dividing this data into mini-batches and training the policy over all the batches becomes intractable. For this we will use a Stochastic online learning algorithm which consist that after the end of the t^{th} period, the price movement of this period $\mathbf{Y_t}$ are added to the training data. The policy network will then be trained on N_b (Number of batches) random batches of length n_b. Since it's believed that the correlation between two market price event decreases exponentially with the temporal distance between them, we prefer recent batches than old ones, for this we will use a geometrical distribution. For a batch starting at period t_b, it's necessary that $t_b \leq t - n_b$, the probability of its selection is : P(t_b) = $\beta(1-\beta)^{t-t_b-n_b}$ where $\beta \in$ (0,1) determining how important are recent batches.

5 Experiments and conclusion

The data that would be used for this framework is proposed by the paper author. It's constituted from cryptocurrencies (virtual money, i.e Bitcoin) which is uploaded from $Poloniex.com$ and adapted to the given code. The policy will be learned by the CNN deep neural networks with different architectures. The performance metric is the cumulative reward which is equivalent to the cumulative-product variation of the portfolio value between periods. The trading period will be taking equal to $T = 30min$. The maximum commission rate in the $Poloniex$ platform is 0.25% for either both purchasing and selling, so we will take $c_p = c_s = 0.25\%$. The cash asset will be $Bitcoin$, and non-cash assets will be chosen as the other cryptocurrencies such as $Ethereum$ (They are chosen respectively by their history volume as mentioned before). The data will be divided in three sets : training, validation and test.

In the Figure[2], we have trained the policy using a CNN architecture as described in Figure[1] (. This figure shows three different Portfolios with 3, 5 and 11 non-cash assets, we can see that the cumulative reward or equivalently the portfolio value increases in average and that the volatility or

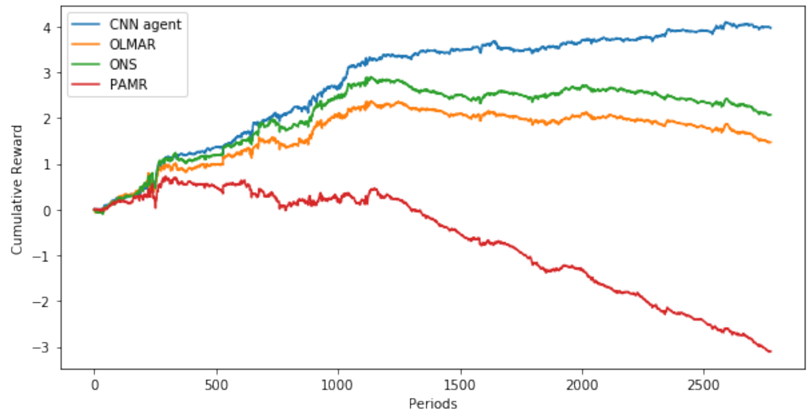

FIGURE 3 – Comparison of the cumulative reward given by the CNN agent and OLMAR, PAMR, ONS algorithms using 11 non-cash assets.

the fluctuation is not very large. The time spent by the network to train increases dramatically with the number of assets : around 1h32 for 11 assets, 38 minutes for 5 assets and 19min for 3 assets using CPU only. We can remark that the portfolio values for the case of 3, 5 non-cash assets are the same approximately, this maybe because the value of the two added assets is negligible in front of the three first non-cash assets.

Comparison of the EIIE framework with other frameworks In this part, we're going to compare the EIIE framework with other existing methods such as OLMAR (On-Line Portfolio Selection with Moving Average Reversion) which is based on representing multi-period mean reversion as ?Moving Average Reversion ? (MAR), which explicitly predicts next price relatives using moving averages, and then learn portfolios by online learning techniques. Explicitly the OLMAR algorithm consists : we initialize the portfolio value and the weights as before by 1 and $\frac{1}{m}\mathbf{1}$ respectively, then the next relative price $\mathbf{y_{t+1}}$ is predicted such that : $\mathbf{y_{t+1}} = \frac{1}{w}\left(1 + \frac{1}{\mathbf{y_t}} + \frac{1}{\mathbf{y_t y_{t-1}}} + ... + \frac{1}{\prod_{k=1}^{w-2}\mathbf{y_{t-k}}}\right)$ where $w \geq 3$ is a window size, and then the next portfolio weight is calculated by $\mathbf{w_{t+1}} = argmin_{\mathbf{w}\in\Delta_m}$ $||\mathbf{w} - \mathbf{a_t}||$ where $\mathbf{a_t} = \mathbf{w_t} + \lambda_{t+1}(\mathbf{y_{t+1}}$ - $s_{t+1}\mathbf{1})$ and $\lambda_{t+1} = \max\left(0, \frac{\epsilon - \mathbf{w_t^T} \cdot \mathbf{y_{t+1}}}{||\mathbf{y_{t+1}} - s_{t+1}\mathbf{1}||}\right)$ and $s_{t+1} = \frac{\mathbf{1^T} \cdot \mathbf{y_{t+1}}}{m}$, ϵ is the Reversion threshold and Δ_m is the set of positive vectors whose elements sum to 1. Another method is Passive Aggressive Mean Reversion (PAMR), it's an algorithm that predict also the portfolio weight vector by also using an optimization formula that is similar to OLMAR one. It's given mathematically by : $\mathbf{w_{t+1}} = argmin_{\mathbf{w}\in\Delta_m}$ $||\mathbf{w} - \mathbf{a_t}||$ where $\mathbf{a_t} = \mathbf{w_t}$ - $\tau_t(\mathbf{y_t}$ - $s_t\mathbf{1})$ where $\tau_t = \frac{l_\epsilon^t}{||\mathbf{y_t} - s_t\mathbf{1}||^2}$ (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) where $l_\epsilon^t = max\left(0, \mathbf{w_t^T} \cdot \mathbf{y_t} - \mathbf{1}\right)$. Online Newton Step (ONS) is another algorithm used for portfolio management which uses also an optimisation formula to predict the next portfolio weight vector that is different from the two previous algorithms, it searches for an optimal projection function that project the previous weights vector scaled by a matrix into the space of the new weights.

We can see from Figures[3,4] that the deep neural network (Convolutional network) which learn the policy gives better performance than the other methods, for two different sets of portfolios. Other test will be presented in the presentation duo to the large time demanded by training a new CNN with different parameters. However, the deep Reinforcement learning framework (EIIE) outperforms other frameworks largely. It's also adaptable to different network architectures for learning the policy (RNN, LSTM and many architectures of CNN with different layers (Convolutional layers, Fully connected ones, Drop-out layers and many others). This framework is also scalable with the amount of data and assets (portfolio size).
Finally, it's worth to note that deep Reinforcement Learning is lately making the buzz thanks to to its remarkable achievements in playing video games and board games (Atari game). Those problems

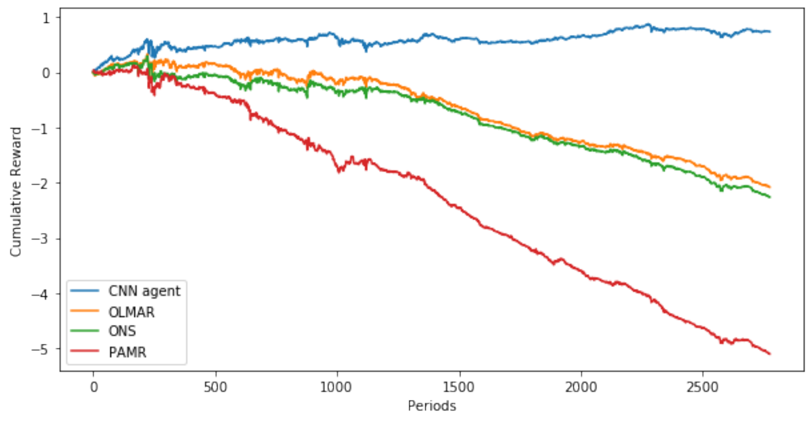

FIGURE 4 – Comparison of the cumulative reward given by the CNN agent and OLMAR, PAMR, ONS algorithms using 3 non-cash assets.

have discrete action spaces, and can not be directly applied to portfolio management problems, where actions are continuous.

References

[1] Zhengyao Jiang &Dixing Xu. &Jinjun Liang A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem and github for data and code. Xi ?an Jiaotong-Liverpool University.

[2] Bin Li &Steven C. H. Hoi. On-Line Portfolio Selection with Moving Average Reversion Nanyang Technological University, Singapore.

[3] Bin Li &Peilin Zhao. &Chu Hong HOI PAMR : Passive-Aggressive Mean Reversion Strategy for Portfolio Selection Singapore Management University.

[4] Amit Agarwal &Elad Hazan. &Satyen Kale & Robert E. Schapire Algorithms for Portfolio Management based on the Newton Method. SPrinceton University, Department of Computer Science.