Investigations on TD methods for time series predictions
Claudio Sole - Greta Laage - RL Class Project W2017

Abstract

This project proposes an investigation on methods applying reinforcement learning to the problem of financial time series predictions. Those methods seem to be more suitable to this kind of problems since they permit modeling and learning various kinds of interactions in real situations, with long term goals. For instance, the $TD(0)$ method learns only from raw experience. The investigation focused on 2 types of temporal difference methods: the conventional one and the recently introduced emphatic temporal difference learning. First, we compare the TD algorithm in the non-linear function approximation case (artificial neural networks). Then we compare the supervised learning approach with TD and ETD algorithm in the case of linear function approximation. The experimental results, led in a quite restricted function approximation framework show no improvement from reinforcement learning methods compared to supervised learning approach, the latter giving nonetheless not satisfying enough results. This confirm the difficulty of the financial time series predictions problem.

1. INTRODUCTION

The recent systems using deep learning methods to predict stock prices has showed better performance than those using only economic technical indicators. However, these news methods are supervised learning, which are often not adequate for learning problems with long-term goals. In supervised learning for stock price predictions, the target values are usually the absolute prices after a fixed time horizon while in reinforcement learning, it would be the relative changes in price since the previous time step, as proposed in [1]. In this general paper introducing TD methods, the author proved the same convergence properties for the RL based updates and the conventional supervised learning approach. The theoretical results are proved for TD(O) but insights are also given when adding eligibility traces. The recently proposed emphatic temporal difference algorithm [2] introduced a generalization of $TD(\lambda)$ algorithm which allows an explicit definition of states in which one are interested. Due to the recency of this paper, ETD has not been tested on large scale examples and compared with other TD methods such as GTD,$GTD2$ and TDC [4]. While we do not pretend in giving a theoretical convergence proof for ETD in this project, we investigated the performances of TD methods and ETD for financial time series predictions.

At each time step is associated a price or a indicator that we want to predict (such as the price trend for example). It is mapped to a state such as defined in reinforcement learning framework, represented by numerical features. The stock price changes process is considered as a Markov process: at each time step t, we suppose the previous state representation contains the values of a fixed number of previous states and s_t depends only on s_{t-1}. Since the total number of states is not finite, we used a generalization method to approximate the prediction function, as defined below.

This paper is organized as following: first section presented the general context of our project. The next section presents the theoretical framework with the learning update rules. We then present the first batch of experiments with artificial neural network function approximation and TD learning. Finally, the last section presents the experimental results on the comparison of TD and ETD predictions with linear function approximation.

2. Reinforcement learning

2.1. Learning to predict approach

The approach developped in [1] focuses on multi-step prediction problems in the context of time series. The TD methods detailed can learn from experience without requiring the complete environment's dynamics. Experience are of the form observation-outcome sequences. Observations are real-valued component vectors and will be represented by the notation $x_1, x_2, .., x_m$ while the outcome is a scalar z. For each sequence, the objective is to predict z at each time step. The prediction is noted P_t and is a function of

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

x_t and a vector of parameters ω: $P_t = P(x_t, \omega)$.

A learning procedure is then defined as an update rule for updating the weights ω. For conventional supervised learning, this update rule is based on the error between the value predicted and the actual outcome. TD methods however do not learn by the difference to the final outcome but rather by the difference between successive predictions within a sequence.

When updating the weights after each complete sequence, the update rule can be expressed as below, with $\Delta\omega_t$ the increment after each observation :

$$\omega \leftarrow \omega \sum_{t=1}^{m} \Delta\omega_t$$

In the supervised learning case, the update are based on the real outcome. The increment is then computed as below: with α the rate learning (positive parameter)

$$\Delta\omega_t = \alpha(z - P_t)\nabla_\omega P_t. \tag{1}$$

The TD procedure, which produces the same result as (1) can be computed in an incremental way. It starts from the equality $z - P_t = \sum_{k=1}^{m}(P_{k+1} - P_k)$ where $P_{m+1} = z$. It is showed in (**?**) that this equality leads to

$$\Delta\omega_t = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t} \nabla_\omega P_k \tag{2}$$

This procedure can be incrementally computed and needs to remember the sum of all past values of the gradient insted of each values instead.

This TD update can be generalized with an exponential weighting with recency on past values. (2) is then modified as following:

$$\Delta\omega_t = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t} \lambda^{t-k}\nabla_\omega P_k \tag{3}$$

By defining $e_t = \sum_{k=1}^{t} \lambda^{t-k}\nabla_\omega P_k$, then $e_{t+1} = \nabla_\omega P_{t+1} + \lambda e_t$. $TD(\lambda)$ can also be computed incrementally :

$$\begin{aligned} \Delta\omega_t &= \alpha(P_{t+1} - P_t)e_t \\ e_t &= \nabla_\omega P_t + \lambda e_{t-1} \end{aligned} \tag{4}$$

The supervised learning with the updates as described in (1) and the TD updates as described in (2) converges to the same evaluation but TD learns faster.

It is showed in [1] that in the linear function approximation case, the updates converge in expected value asymptotically towards the ideal prediction $\mathbb{E}[z|i]$ the true outcome value given that the sequence starts in i.

$$\lim_{n\infty} \mathbb{E}[x_i^T w_n] = \mathbb{E}[z|i]$$

Later in this project, we investigate on the performances of the recently introduced algorithm $ETD(\lambda)$ for time series prediction. In order to generalize the updates rules such as described in [1], we present the algorithm in the conventional TD case.

2.2. Temporal Difference methods

In the classical reinforcement learning approach, an agent interacts with its environment. From a certain state, it takes an action to go the next state and gets a reward.

The $TD(0)$ algorithm, used for the policy evaluation problem (ie estimating the value function) is described in 5. V is the estimated state dependent value function, R is the reward and γ the discount factor.

$$V(S_t) = V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right) \tag{5}$$

In the linear function approximation case, we suppose $V(s) = \theta^T\phi(s)$ where θ is a vector of parameters and $\phi(s)$ a feature vector of the state s. The objective is to estimate the value of the parameters.

Then the $TD(0)$ udapte can be written, in an on policy prediction with approximation problem :

$$\theta_{t+1} = \theta_t + \alpha\left(R_{t+1} + \gamma\theta_t^T\phi(S_{t+1}) - \theta_t^T\phi(S_t)\right)\phi(S_t) \tag{6}$$

The $TD(\lambda)$ algorithm, which is based on the same principle as $TD(0)$ but with a recency weigthing is described in (7) with $e_0 = 0$ in the on-policy case.

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha\left(R_{t+1} + \gamma\theta_t^T\phi(S_{t+1}) - \theta_t^T\phi(S_t)\right)e_t \\ e_t &= \gamma\lambda e_{t-1} + \phi(S_t) \end{aligned} \tag{7}$$

The $ETD(\lambda)$ algorithm, introduced in the paper [2] and proved convergence in the paper [3] is described in 8, in the general off-policy case with linear function approximation. ρ is the importance sampling ratio. It is based on interests functions: $\mathcal{S} \rightarrow [0, \infty[$ that allows to specifically define the states we are interested in, an state-dependent discount function γ and a state-dependant bootstrapping factor λ.

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha\left(R_{t+1} + \gamma\theta_t^T\phi(S_{t+1}) - \theta_t^T\phi(S_t)\right)e_t \\ e_t &= \rho_t(\gamma(S_t)\lambda(S_t)e_{t-1} + M_t\phi(S_t)) \\ M_t &= \lambda(S_t)i(S_t) + (1 - \lambda(S_t))F_t \\ F_t &= \rho_{t-1}\gamma(S_t)F_{t-1} + i(S_t) \end{aligned} \tag{8}$$

M_t is the emphasis for each state. It basically indicates how much each state is important. A state can be important for two reasons: either it is because it is used for another update, either it is because the learner said so.

F_t is the followon trace for each state S_t. It is the expected number of times the agent would be in that state assuming it started from here with the behaviour policy distribution and continue its trajectory following the target policy.

2.3. Assumed ETD updates for time series predictions

In order to investigate on the performances of ETD, we generalized the update rules described in (4) with the original $ETD(\lambda)$ update rules (8). The rules we used for ETD learning are written in (9). We can indeed consider that we are in the on-policy case so the importance sampling ratios are all equal to 1.

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha \left(R_{t+1} + \gamma \theta_t^T \phi(S_{t+1}) - \theta_t^T \phi(S_t)\right) e_t \\ e_t &= \gamma(S_t)\lambda(S_t) e_{t-1} + M_t \phi(S_t) \\ M_t &= \lambda(S_t) i(S_t) + (1 - \lambda(S_t)) F_t \\ F_t &= \gamma(S_t) F_{t-1} + i(S_t) \end{aligned} \tag{9}$$

The first batch of our experiments was the comparison of TD learning and supervised learning on non-basic fonction approximation. This part is described in the following section.

Then, the objective was to compare TD learning, ETD learning and supervised learning. Since ETD is defined in the linear function approximation case, we kept this framework in the second batch of experiments.

3. Neural Network function approximation and TD learning

Financial time series provide an interesting task for analyzing the difference between the supervised learning and the TD-learning approach for sequence prediction. Given their fractal nature, it is indeed easy to cast a single-step next day prediction into a multi-step prediction, considering a day as, for example, a 24 steps sequence in the hourly timeframe (Figure 3).

The purpose of our first experiment is to predict next daily direction using hourly data. The problem can be interpreted as a 24-steps prediction where, given a sequence of states $x_1, ..., x_{24}$ (one for each hour of the day) the outcome to be predicted is 1 (positive day) or 0 (negative day).

We started our investigation implementing the Supervised Learning approach. As mentioned above, this imply that for each hour t of a day d, we will have a training example of the form (x_{td}, z_d), thus not taking into account the temporal information the lead to that outcome. For the description of a state x_{td} the following features have been used:

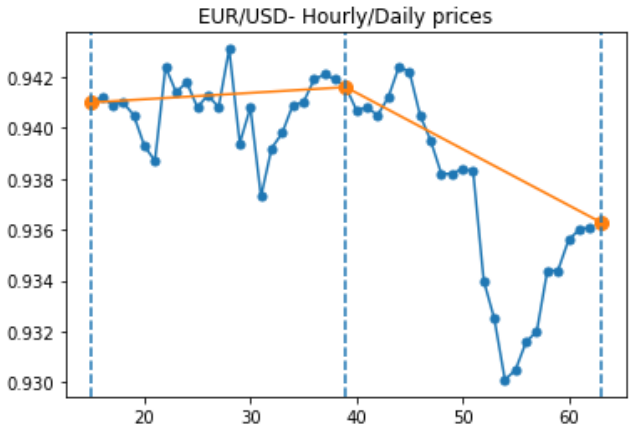

Figure 1. Daily (orange) and Hourly (blue) returns

- 14 past hourly High, Low, Open and Close prices
- past 24 Moving Average values for various time windows: 5,20,50 and 200
- hour of the day
- past 6 crossover points of the 5 and 20 periods moving average in terms of time and price. Based on very spread idea developed by Richard Donchian, if the 5 periods moving average crosses the 20 periods moving average in the upside direction, this can be interpretd as a strength sign, while the opposite is true for a crossover in the downside direction.

This features were then given as input to a Multilayer Perceptron to predict the daily outcome (1 or 0).

The same state description and network architecture were then used for the TD(0) approach. For each observation x_{td} a prediction P_{td} is computed for z_d, and the same is done at the following step $x_{t+1,d}$ obtaining $P_{t+1,d}$. We then use (4) to compute the network updates (with the gradient being computed by the backpropagation algorithm)

The experiment was conducted on EUR/USD currency pair data from 2009-02 to 2017-03. The results in Table (3) show the accuracy performance in predicting daily direction at each hour. In plotting figure (2) we decided to simulate a possible trading scenario where at each day we want to decide if buying or selling the EUR/USD pair. In particular, the *red* line indicates a day for which at 00:00 a negative return was predicted, while the *green* line represents a positive return prediction. Thus leading for example, to

	Supervised Learning	TD(0)
Accuracy	64.35%	57.49%

a correct prediction for the first day (between the first and second vertical lines) and fourth one (between the fourth and the fifth vertical lines), and to a wring prediction for the sixth and the seventh days (from sixth to eight vertical lines).

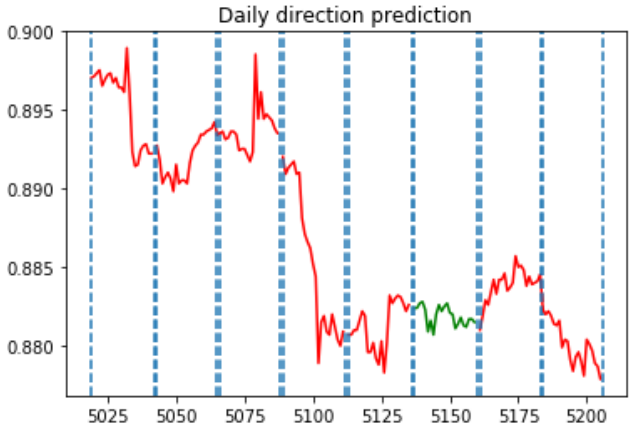

Figure 2. Direction prediction at 00:00 for each day. *Green* for positive return and *red* for negative return

4. ETD Learning experiments

In this experiments, we stayed in the framework of financial time series prediction but with a simpler model for the prediction.

The data used in those experiments were the normalized daily stock prices of the **Procter & Gamble Company** from January 1970 to April 2017.

For a fixed number p of days, the objective was to predict the open stock price of the $(p+1)^{th}$ day.

We made several experiments, varying the following parameters :

- Method to normalize the data
- λ, constant or not
- γ, constant or not
- p which is also the size of the sequence presented to the algorithm
- Features of each state
- interest function

Regarding the learning rate, we considered $\alpha_t = 1/t$ with t the time step since the convergence proof of (**?**) requires a decreasing learning rate over time.

We considered two models of features for a state. The first one contains the open stock prices of the p previous days normalized. The second one contains those last values, the moving average over the 5 previous days, the moving average over the 20 previous days, the trend of stock prices (-1 if it was decreasing for the h past days, +1 if it was increasing and 0 otherwise, with h a parameter we varied).

The graphs (3) represents the average error between predictions and real outcome within a sequence using the standardized data. In this experiment, $p = 7$, $h = 3$ λ is increasing as we go closer in time to the day we want to predict the open price. The interest is equal to 1 for all state and discount factor is constant at 0.9 .

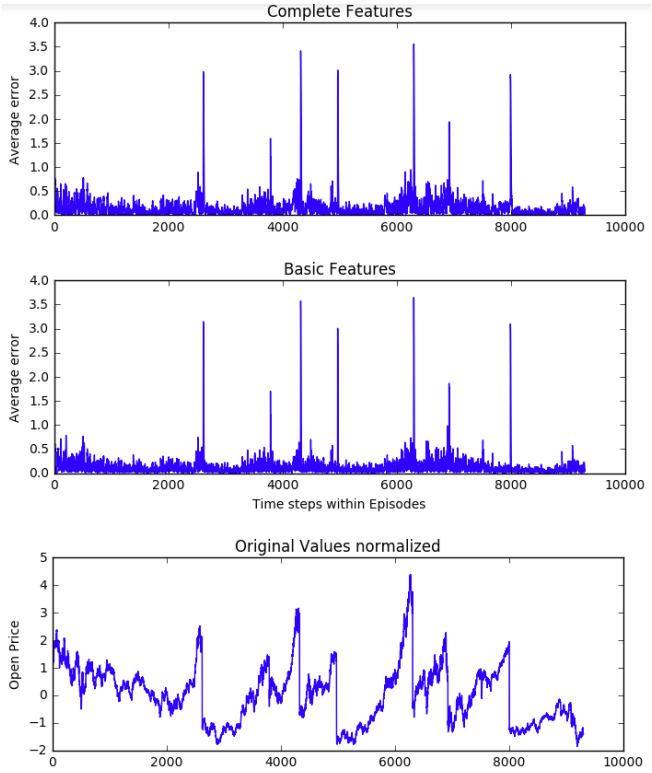

Figure 3. First run of experiments

The error values seems a bit smaller with the basic features, we will use only those ones for the following experiments. However, the graphs are of the same shape in the two cases. We observe some high peaks of the error, which corresponds to moments in time where the stock prices bumped. This behavior is hard to predict, as we can see the important value of the error.

The graphs (4) compares on a sample of sequences the error for supervised learning, $ETD(\lambda)$ and $TD(\lambda)$.

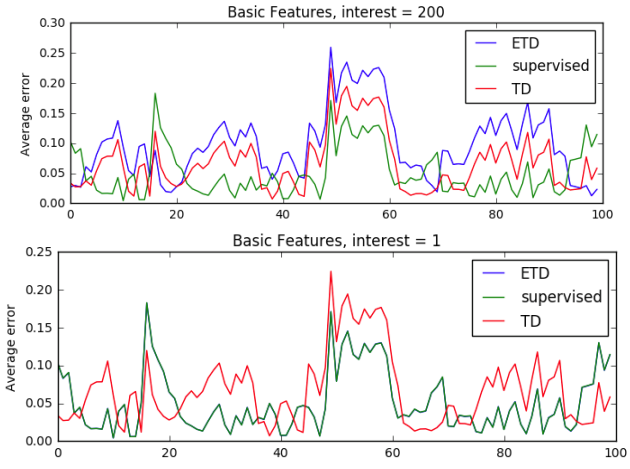

Figure 4. Second experiments. $p = 7, \gamma = 0.9$, λ increasing as we go closer in time to the day to be predicted.We observe that when the interest is constant to 1, the predictions are the same for ETD and supervised learning

We also ran some experiments regarding the interest functions. We tried a random interest function, which map each state to a random value from a uniform distribution between 0. and 100 and another one relative to the bumps. As soon as there is a bump in the previous data, we add a much bigger value of the interest to the states after the bump. However, the results we found were not significant.

In conclusion, in the case of linear function approximation for financial time series, the different algorithms (supervised learning, ETD and TD) are not able to make satisfying predicitions and we could not observe improvements from ETD compared to supervised learning. But the linear case is really restrictive in this case, financial time series data are known to be highly non linear and with a high variability. A more complicated model for prediction function as the one developped in the previous section might give better results, but it cannot be compared with ETD though.

5. Conclusion

In this report we investigated the temporal difference approach for multi-step time series prediction. In particular, the main idea was to verify if valuable information could be added to the prediction task taking into account the sequential nature of the time series, as opposed to the Supervised Learning approach, where observations are supposed to be i.i.d. Our result seem to confirm the well known difficulty of predicting financial time series, independetly of the choosen approach. Moreover, it should be considered that the process underlying financial markets is not Markovian, thus making the application of TD kind of algorithms even harder. We tried to mitigate this problem including features representing the past evolution of the time series in the state representation. However, Supervised Learning significantly outperforms the $TD(0)$ approach in out first experiment with nonlinear function approximation using Neural Netowrks, while comparable results are obtained by $ETD(\lambda)$ with $\lambda < 1$ and the supervised learning approach, in the second experiment, where a linear function approximation has been used.

6. Bibliography

[1] R.S. Sutton,1988 *Learning to predict by the method of temporaldifferences*, Machine Learning, $3 : 9 - 44$

[2] R.S. Sutton, A.R. Mahmood and M. White, 2015, *An emphatic approach to the problem of off-policy temporal-difference learning*

[3] H. Yu,2015, *On Convergence of Emphatic Temporal-Difference Learning*,CoRR