Trading via Recurrent Reinforcement Learning

Abhishek Kumar
Lovejeet Singh
Nimit Bindal
Raghav Goyal

November 16, 2013

Contents

1 Problem Definition

We propose to investigate trading (ForEx/stocks/derivatives) with neural networks [1] trained via Recurrent Reinforcement Learning (RRL) [2].

2 Introduction

A lot of work is being done on stock market prediction using various supervised learning techniques. The trade is executed by relying on the forecasted time series, whereas the real objective function in these methods is to optimise some relevant measure of trading system performance, such as profit, economic utility or risk-adjusted return. Moreover, all of these techniques does not include transaction cost as well.

Moody [2] proposed a better method for trading using reinforced learning algorithms to directly optimize such trading system performance functions. When transaction costs are included, the trading system must be recurrent, thus requiring a recurrent reinforcement learning (RRL) approach. Carl Gold [3] implemented the same technique for ForEx trading.

Over the past years, there has been a lot of buzz in the industry around the idea of recognize the invisible patterns of time series data, along with the data mining technology and time series analysis theory. One relatively new approach to financial trading is to use machine learning algorithms to predict the rise and fall of asset prices before they occur. An optimal trader would buy an asset before the price rises, and sell the asset before its value declines.

Already, some researchers have reported experimental results of SVM application in finance engineering field. We try to achieve this with the use of neural networks trained via Recurrent Reinforcement Learning as proposed by Moody and Saffell.

3 Methodology

3.1 Neural network trading functions

Here we provide the basic overview of the neural network which is used to make trading decisions on single price series. The input of the neural network is only the recent price history and the previous position taken. Because the previous output is fed back to the system as part of input, the neural networks are called recurrent. The output of the network $F_t \ \epsilon$ {-1, 1} is the position (long/short) to take at the time t. For a single layer neural network (also known as a perceptron) the trading function is

$$F_t = sign(\sum_{i=0}^{M} w_i r_{t-i} + w_{M+1} F_{t-1} + v) \tag{1}$$

where, $\vec{w}$ and v are the weights and threshold of the neural network, and r_t is the price return at time t.

For trading where a fixed amount is invested in every trade , the price returns are given by

$$r_t = p_t - p_{t-1} \tag{2}$$

In practice the *sign* function is replaced with a *tanh* function so that derivatives can be taken with respect to the decision function for training. The *tanh* function is then thresholded to produce the output.

3.2 Returns and utility function

Profit after a time period T is given by

$$P_T = \sum_{t=1}^{T} R_t \tag{3}$$

where,

$$R_t = \mu(F_{t-1} r_t - \delta |F_t - F_{t-1}|) \tag{4}$$

where, μ is the number of shares traded and δ is the transaction cost per share traded.

We use the **Sharpe Ratio** to evaluate the performance of the system for training. The Sharpe Ratio is given by

$$S_T = \frac{Average(R_t)}{StandardDeviation(R_t)}; t = 1, 2, ..., T \tag{5}$$

where R_t is the return on investment at trading period t. Intuitively, Sharpes ratio rewards investment strategies that rely on less volatile trends to make a profit.

Calculating the exact Sharpe Ratio at every point in time results in an $O(T^2)$ algorithm, so in order to evaluate the traders performance we use the **Differential Sharpe Ratio**. The Differential Sharpe Ratio is derived by considering a moving average version of the simple Sharpe Ratio given by

$$\hat{S}(t) = \frac{A_t}{B_t} \tag{6}$$

where,

$$A_t = A_{t-1} + \eta(R_t - A_{t-1}) = A_{t-1} + \eta \Delta A_t \tag{7}$$

$$B_t = B_{t-1} + \eta(R_t^2 - B_{t-1}) = B_{t-1} + \eta \Delta B_t \tag{8}$$

where, A_t and B_t are exponential moving estimates of the first and second moments of R_t respectively. The Differential Sharpe Ratio is derived by expanding the moving average to first

order in the adaptation parameter η, and using the first derivative term as the instantaneous performance measure, given by

$$D_t = \frac{d\hat{S}_t}{d\eta}|_{\eta=0} = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{\frac{3}{2}}} = U_t \tag{9}$$

This differential sharpe ratio provides a convenient assessment of the traders performance and hence it is used as the utility function U_t for use in Recurrent Reinforcement Learning.

3.3 Gradient ascent

The goal of Recurrent Reinforcement Learning is to update the weights in a recurrent neural network trader via gradient ascent in the performance function,

$$\vec{w}_t = \zeta \vec{w}_{t-1} + \rho \frac{dU_t}{dw_t} = \zeta \vec{w}_{t-1} + \Delta w_t \tag{10}$$

where, $\vec{w}_t$ any weight of the network at time t, U_t is the utility function, ζ is the decay parameter and ρ is the learning rate.

Δw_t is given by,

$$\Delta w_t = \rho \frac{dU_t}{dw_t} \approx \rho \frac{dU_t}{dR_t}\{\frac{dR_t}{dF_t}\frac{dF_t}{dw_t} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{dw_{t-1}}\} \tag{11}$$

For the Differential Sharpe Ratio the derivative of the performance function with respect to the returns is,

$$\frac{dU_t}{dR_t} = \frac{dD_t}{dR_t} = \frac{B_{t-1} - A_{t-1}R_t}{(B_{t-1} - A_{t-1}^2)^{\frac{3}{2}}} \tag{12}$$

For trading returns, the derivatives of the return function are,

$$\frac{dR_t}{dF_t} = -\mu\delta sign(F_t - F_{t-1}) \tag{13}$$

$$\frac{dR_t}{dF_{t-1}} = r_t + \mu\delta sign(F_t - F_{t-1}) \tag{14}$$

The neural network trading function is recurrent so the derivatives $\frac{dF_t}{dw}$ for on-line training are calculated in a manner similar to back-propogation through time,

$$\frac{dF_t}{dw_t} \approx \frac{\partial F_t}{\partial w_t} + \frac{\partial F_t}{\partial F_{t-1}}\frac{dF_{t-1}}{dw_{t-1}} \tag{15}$$

3.4 Algorithm for training and testing

The algorithm used for training and trading is as follows:

Train the neural network in an initial training period of length L_{train}. The trades and performance during the training period are used to update the network weights but are then discarded so that they do not contribute to the final performance. The training period may be repeated for

any number of *epochs*, n_e. The training period is then followed by a trading period of length L_{test}.

The trades made during the trading period are the actual trades for the period, and also update of the network weights continues during the trading period. After the trading period, the start of the training period is advanced by L_{trade} and the process is repeated for the entire sequence of data.

4 Experiments with currency trading (FOREX)

The time series is of US Dollar vs. Euro exchange rate between 01/01/2008 until 31/12/2012 on 15 minutes data points taken from 5 year data. We sampled out data between 01/01/2008 until 10/02/2008 with nearly 2470 points.

This sample data is divided into training and test data.
Training Set consist of data between 01/01/2008 until 01/02/2008, 2000 data points.
Test Set consist of data between 03/02/2008 until 10/2/2008, 470 data points.

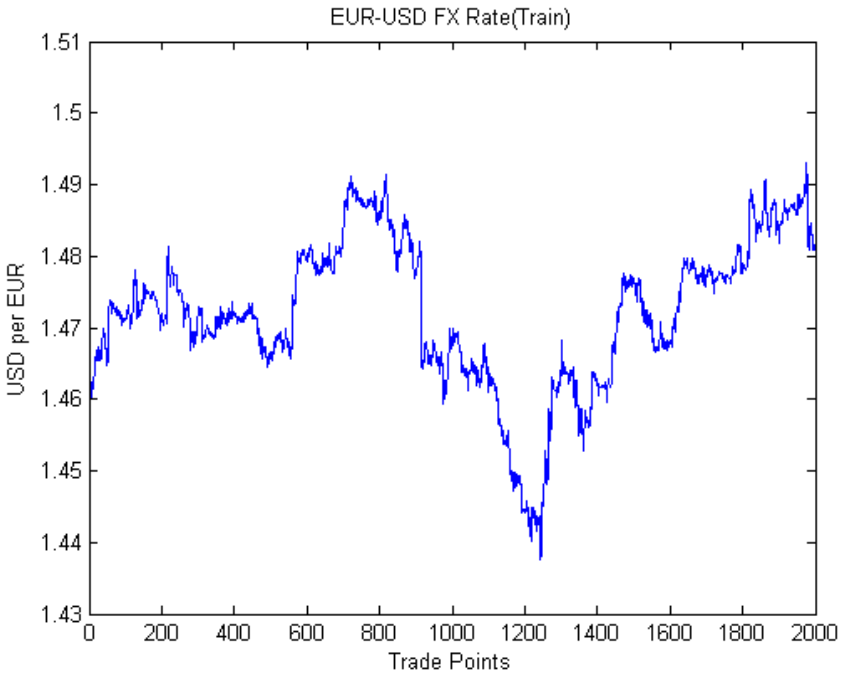

Figure 1: Train Data

All price returns were normalized before being input to the neural networks. In order to achieve this, the mean and variance of the price returns were calculated over the first training period (before beginning training) and then all price returns were normalized to zero mean and unit variance with respect to these values before being input to the neural network.

4.1 Validation

Parameters of the algorithm include the learning Rate ρ, the coefficient of weight decay ζ, adaptation rate η, the size of the training and test windows, M and the number of epochs of training n_e.

The large number of parameters made it quite impossible to systematically test all but a small number of parameter combinations. The parameter were tuned by systematically varying each parameter while holding the other parameters fixed. After all parameters had been tuned, the previous values would be re-checked to see if their optimum value changed due to the change in the other parameters. (This technique is commonly known as the greedy graduate student algorithm.)

Substantial amount of improvements are observed during validations of η and ρ which are as shown below:

$$\eta(adaptationrate) = 0.01$$
$$\zeta(DecayParameter) = 1$$
$$\rho(LearningRate) = 0.055$$

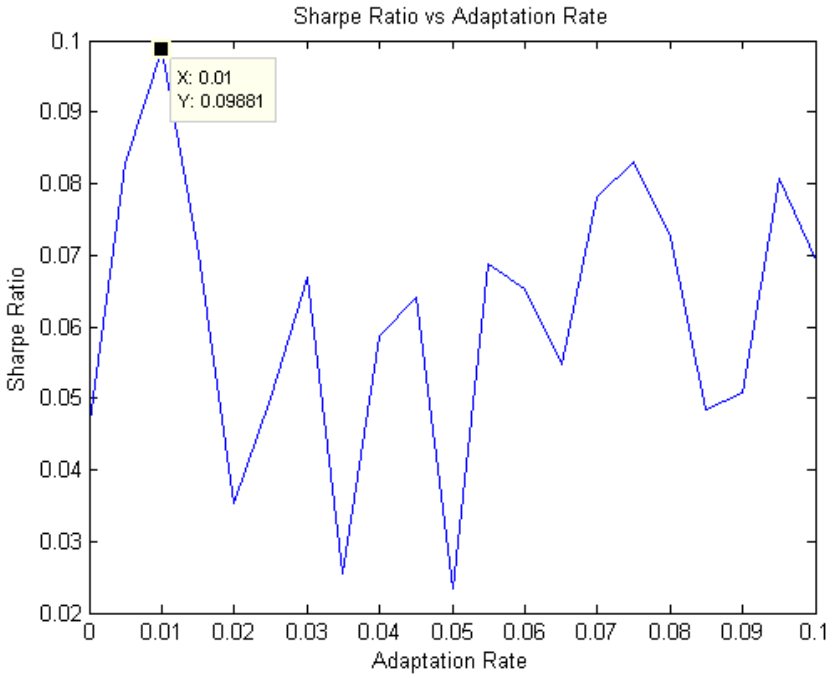

Figure 2: Validation of η

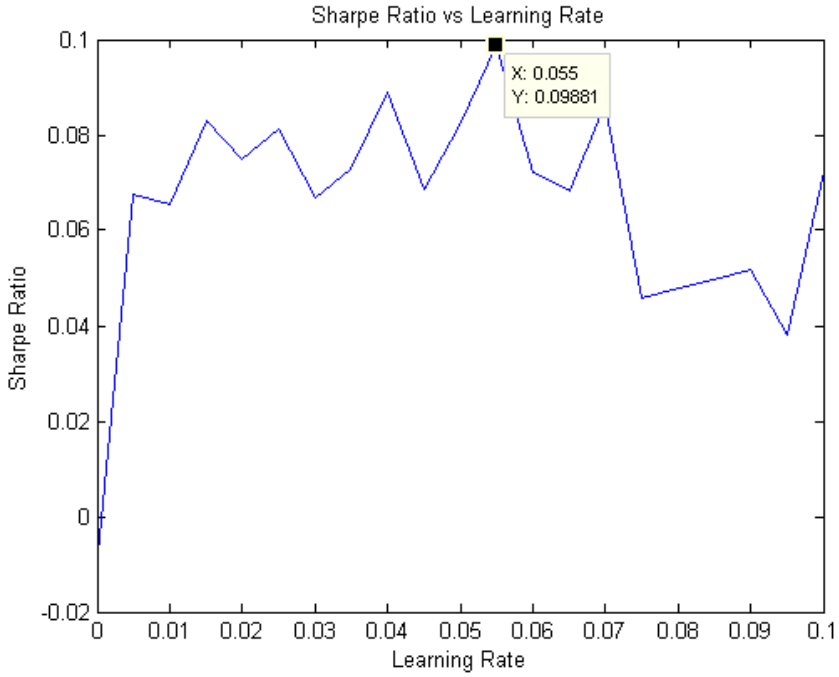

Figure 3: Validation of ρ

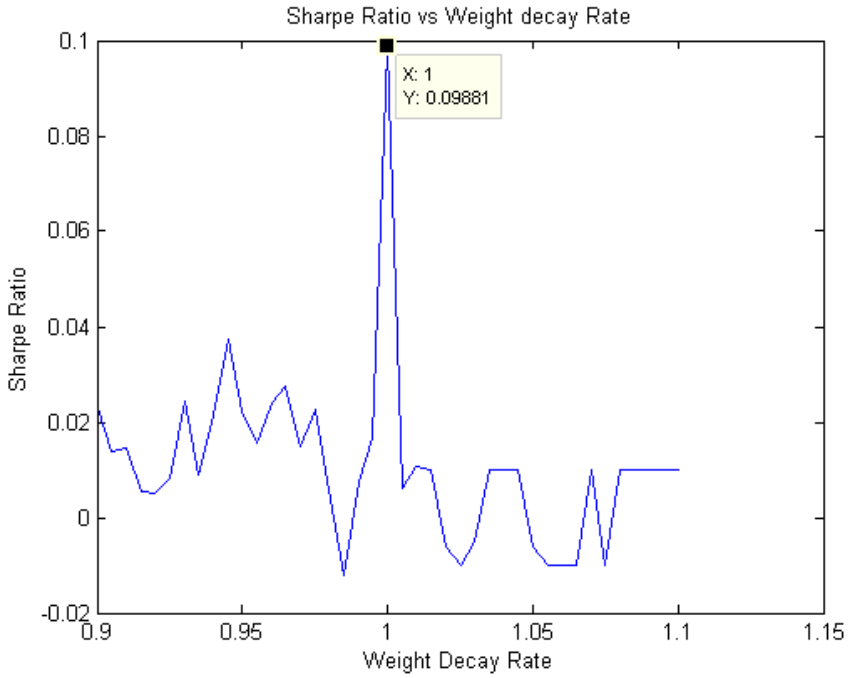

Figure 4: Validation of ζ

4.2 Performance on training data

We observe that the Sharpes Ratio converges as the number of epochs are increased, ensuring better values of weights. The following results are obtained with no transaction costs, $\delta = 0$.

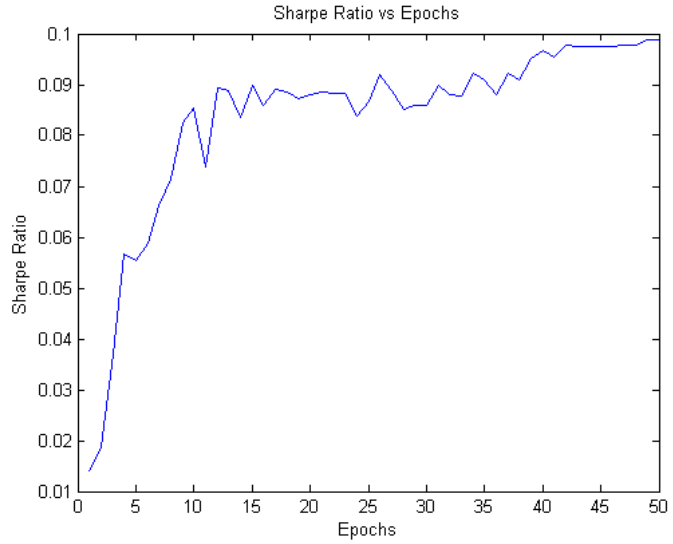

Figure 5: Convergence of Sharpe Ratio

Now using these weights we find cumulative returns on training data. Also, we compare our strategy with benchmark strategies such as:
1. Random/Monkey Strategy: Takes position of trades randomly between +1 and -1.
2. Buy and Hold Strategy: Buys the stock and hold it till the end.

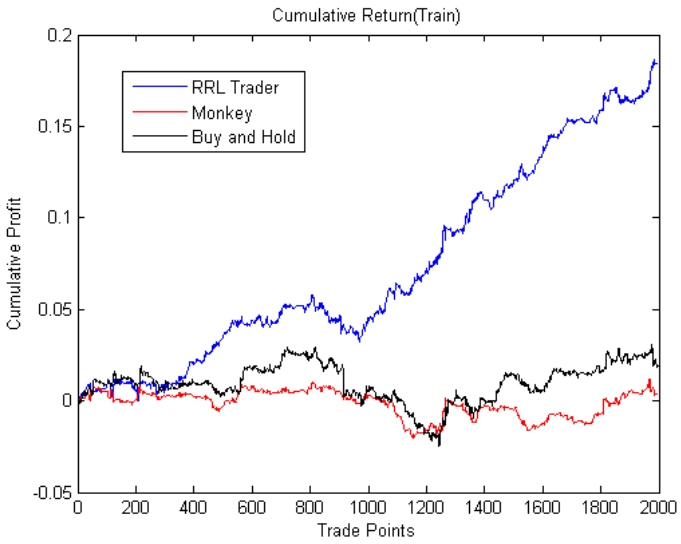

Figure 6: Cumulative Returns on Training Data

This shows that our strategy, RRL performs better than Monkey and Buy/Hold Strategy.

4.3 Performance on test data

Test Set consists of data between 03/02/2008 until 10/2/2008, about 470 data points.

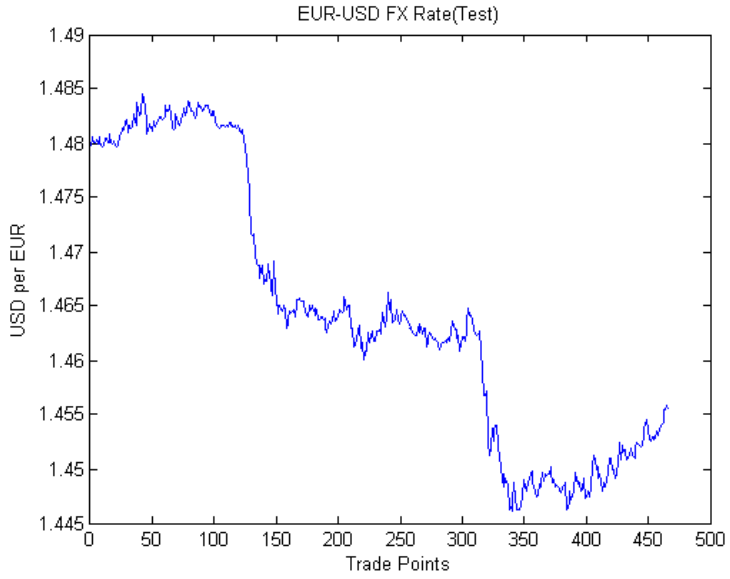

Figure 7: Test Data

We tested our strategy on this Test set with Benchmark strategies,

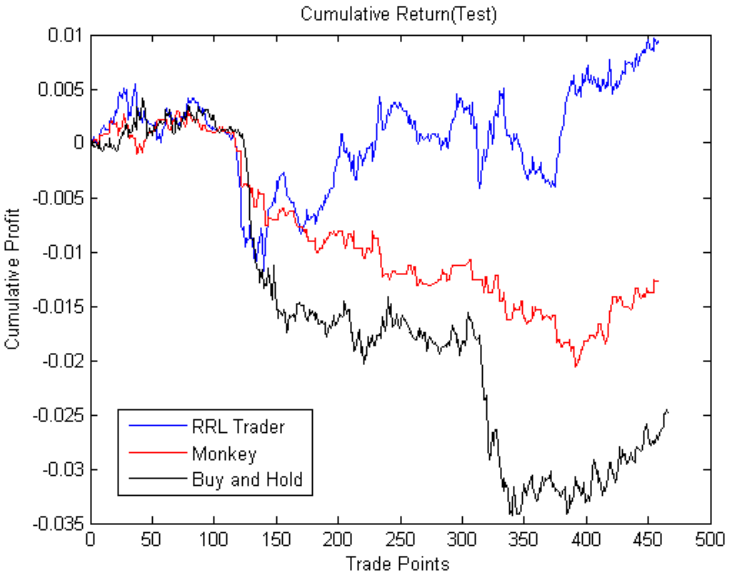

Figure 8: Cumulative Returns on Test Data

We observe that the returns are maximized with the RRL approach.

4.4 Variation with transaction costs

Boxplots of Profit/Loss obtained during testing with transaction rates of 0, 0.0001 and 0.001 (actual transaction cost) are shown below,

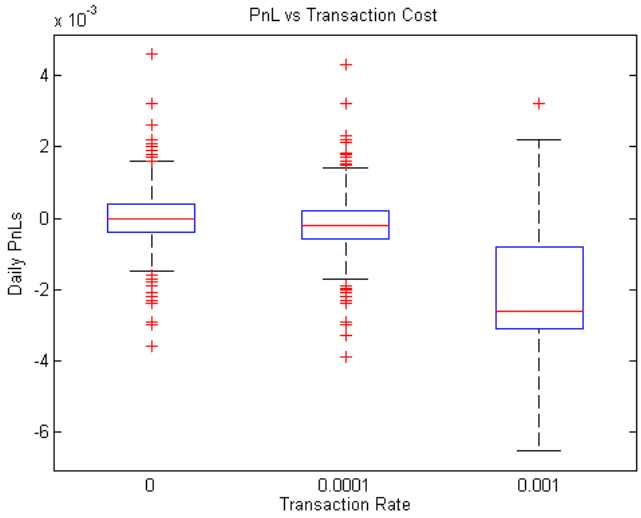

Figure 9: Boxplots

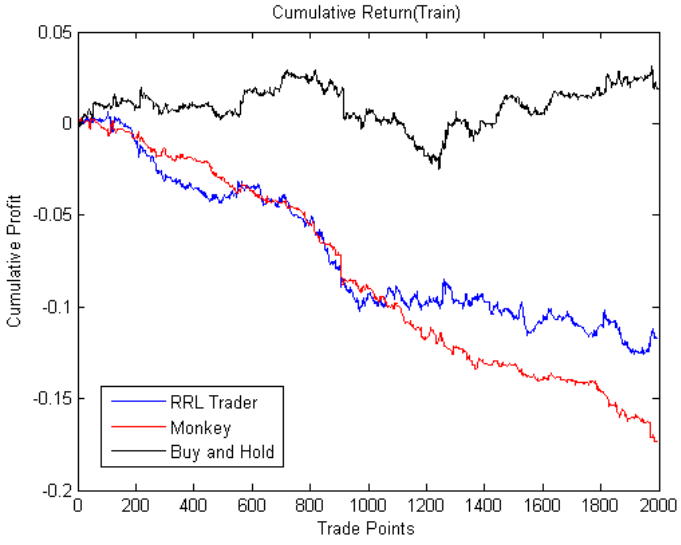

Figure 10: Cumulative returns with transaction costs.

RRL performs badly as transaction rate increases. Additional layers to control the frequency

of trade is required to minimize losses due to transactions. We observe that RRL performs better than random strategy. Buy and Hold strategy depends on data therefore, it is irrevalent.

4.5 Training iteratively on rolling data samples

Our algorithm takes Rolling data samples of size 60 days and further divides into training and test set of size 50 and 10 days respectively. We trained our model using training set and trade using this model on test set to generate cumulative returns for training set. We iteratively roll our training set starting from 51th day till end with interval of 10 days.

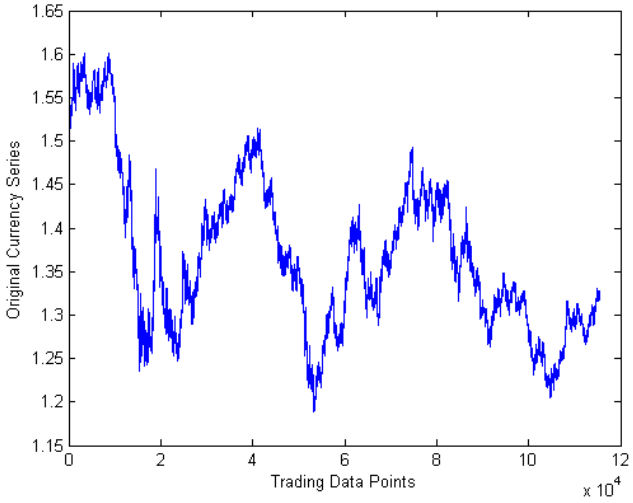

Figure 11: Original Series

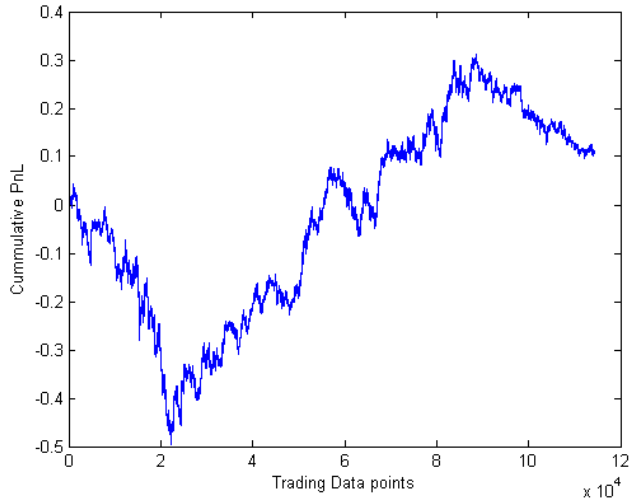

Figure 12: Rolling Cumulative Return

We observe profits at the end of trade, but there are losses in several trade cycles due to unpredictable change in behaviour of data.

5 Conclusion

One cannot deny the fact that there will be some unprofitable transactions, since our system at this point is trading at every step hence the profits incurred are nullified by the losses because of the transaction cost component. When trading with real data, the transaction cost is major factor.

RRL seems to struggle during **volatile** periods. During volatile periods, variance in the data is high. Therefore, changes in market conditions lead to waste of all the systems learning during the training phase.

6 Future work

Current system suffers from transaction costs, therefore, we need to add more risk management layers (e.g. Stop-Loss, retraining trigger, shut down the system under anomalous behavior). This will manage risk in an intelligent manner so that it protects past gains and avoid losses by restraining or even shutting down training activity in times of high uncertainty.

Dynamic optimization of external parameters (such as learning-rate) in order to identify optimal parameter values for a particular trade.

Working with variable number of stock size at each trade.

References

[1] C Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995

[2] J Moody, M Saffell, Learning to Trade via Direct Reinforcement, IEEE Transactions on Neural Networks, Vol 12, No 4, July 2001

[3] Gold, C. (2003). FX trading via recurrent reinforcement learning. Proceedings of the IEEE International Conference on Computational Intelligence in Financial Engineering 2003, op.cit., 363371.

[4] G Molina, Stock Trading with Recurrent Reinforcement Learning(RRL).