CS230

Deep Reinforcement Learning For Forex Trading

Deon Richmond
Department of Computer Science
Stanford University
deonrich@stanford.edu

Abstract

The Foreign Currency Exchange market (Forex) is a decentralized trading market that receives millions of trades a day. It benefits from a large store of historical trend data for dozens of currencies, that many experienced traders use to predict price action for future trade prospects. The question is with this immense amount of data is it possible to train a Machine Learning model to trade profitably only using this historical data. While Machine Learning models typically learn through ground truth labels, more complex problems lack such examples to train on. Thus with the current achievements of Reinforcement Learning, specifically Deep Q Learning, in being able to train a model from experience rather than ground truths examples, this paper seeks to apply these techniques to produce a model that can profitably trade on a trading platform.

1 Introduction

The Foreign Currency Exchange market (Forex) is a decentralized trading market that receives millions of trades a day. Because of this there is a large dataset of spanning years of the historical price action of currency exchanges. It is well known that Machine Learning models benefit from large amounts of training examples to effectively learn a problem. However in these learning paradigms a model is given ground truth labels of what is the correct answer. In many more complex problems such luxuries are not available, and the model must learn on its own in an environment.

If the model has a finite set of actions it can take, then reinforcement learning has shown great progress it producing models that effectively navigate a given environment. Examples of where models have succeeded in this learning paradigm is playing games, which have very well defined environments. Trading also benefits from this in having a very small action space i.e. placing a trade. So is it possible for a model to learn to trade on the Forex Market?

This experiments seeks to train a model using recent historical trend data to trade. The input for this model is current "state" of the market represented as the most recent 100 candle sticks of data. These candle sticks contain the high, low, open, and close price in the period. In addition to this the model also receives the 200-period Exponential Moving Average, and its trade history in the 100 candle stick period (represented as a two-hot encoding vector), and a vector representing what two currencies are currently being traded. The model then predicts, given this input, they expected Q-value of the three actions it can take (buy, sell, or hold at the current close price).

2 Related work

Taking a similar approach to this problem David W. Lu used Reinforcement learning to train a model to trade on the stock market in his paper [1]. While this technique proved successful he used two loss functions the *Sharpe Ratio* and the *Downside Deviation Ratio*. These ratios are specific to analyzing the risk of trades, which is ideal for training a model to perform like other human experts but stifles its abilities to learn its own way of trading through experience.

I ended up taking an approach similar to Gyeeun Jeong and Ha Young Kim [2] and Xiang Gao[3] In using Deep Q Learning to predict the Q-value of a given state so the model could learn to generalize to the vast infinite state space of price actions that it would have no training experience on. In their paper they use a similar approach of a reward function of profits of the closing trade $r = (p_t - p_n)/p_n$ where p_t is the closing price and p_n is the price the trade was opened at. However in their approach they also allowed for the model to select the amount of shares to trade with. This added another layer of complexity in needing to train another Neural Network layer to predict the number of shares to trade after predicting the most profitable action. I have opted to omit this with the assumption my model trades equally on every trade.

I used a similar training strategy of Matthew Hausknecht and Peter Stone's paper [4] of full end-to-end weight update and using Back Propagation through time. Other techniques of pre-training the weights of the recurrent layer suggested in [3] wouldn't allow the model to learn salient feature extraction "filters" for the time series data simultaneously. This technique would be better suited in situation where the model is to be expanded to also select number of shares to trade with in each trade.

3 Dataset and Features

The whole dataset consisted of 32 currency exchanges on 12 unique currencies: AUD, CAD, USD, CHF, JPY, NZD, EUR, GBP, HKD, CNH, MXN, XAU. Historical Data was pulled from February 3rd 2020 to August 29th, 2020 on 4-hour candlestick intervals using Meta Trader 4 trading application. This led to a time series dataset with sequence length 1,056 and feature length 9.The input in the model was the price difference from one candle stick feature to the next and The features included: the low, high, open, close, and 200-EMA price. Giving the model the price difference instead of only the current prices allowed the model to distinguish current states more easily than without. The model also received a two hot encoded vector of whether the model bought or sold at the current sequence close price, the corresponding close price and the 200 EMA (both non-difference).

The model was trained on 28 of theses currency exchanges and 4 of them were separated for the test set. The test set was also larger sequences of length 6,200. Before being inputted into the model for training and testing the data is scaled by the standard deviation of the training data.

4 Methods

Deep Q Learning, Reinforcement learning algorithm was used for this problem. Deep Q learning involve an agent taking actions in an environment, based on a learned policy. At every time time step t the agent receives a state s_t from the environment. Then based on this state the agent takes an action a_t, and receives a reward based on some reward function $R(s_t, a_t)$. The Q-value of a state an action is then defined to be the reward associated with taking that action in the given state in addition to the Q-value of taking the next best action in the resulting state (discounted by a factor γ),

$$Q(s, a) = R(s, a) + \gamma * max_{a'}(Q(s', a'))$$

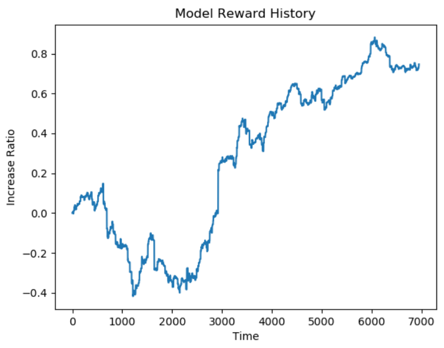
The agent then through experience in interacting with the environment learns the Q-value function in order to predict the value of taking an action in a given state. Why Deep Learning is useful for this is that in a state space that is continuous of even just infinite the generalization qualities of Deep Learning model can still predict relatively well the Q values of state it hasn't seen before. Since a higher Q-value means a higher anticipated reward the model will greedily select the action with higher value. In order to encourage the model to explore other actions an ϵ hyper parameter is chosen and random number is selected before taking an action if the random number is less than ϵ then a random action is taken otherwise the agent proceeds with its greedy approach.

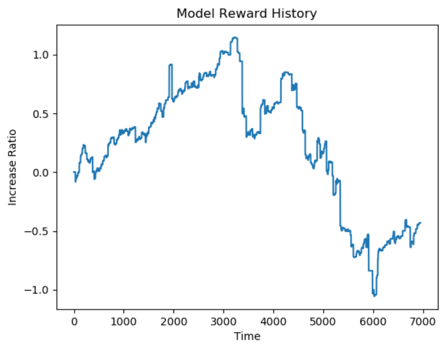
To connect this back to the current problem, Deep Q learning is used to predict the profitability of opening or closing trades. The time series input is fed forward through a Gated Recurrent Unit (GRU) layer with 128 units, which is then fed forward through a fully connected layer with 3 outputs, one for each action, with a linear activation function. The Gated recurrent Unit is a variant of the recurrent layer that allows for propagation of the previous hidden states through time, by use of update gate. This allows for dependencies to be made across long sequences which is important in this case.

5 Experiments/Results/Discussion

As there are 28 different currencies, the agent is trained in 28 separate environment simultaneously during training. Each environment simply keeps track of the current time step and returns the current 100 candlestick input to the agent. The environments also keep track of every action the agent has taken. If the agent makes a series of "Buy" actions at distinct time steps, it is as if in each candle stick the agent placed a buy order at that close price. But once the model makes a "Sell" order or whatever action closes the particular trade, then all "Buy" orders are closed at the current close price, and the reward is equal to the total percent increase or decrease in profit.

The hyper-parameters used for training were a mini batch size of 256, Gamma of .9, number of episodes 500, target update of 1000 trials, replay size of 4000, epsilon for random action selection of .05, and learning rate of .001. These hyper parameters selections were chosen because they are fairly common in the literature researched, and also after few trials of differing learning rate, Gamma and mini-batch sizes these parameters allowed for stable training. The loss function for back propagation was mean squared error loss, and the overall metric for performance evaluation during test time was maximum reward. Below are two figures of the summation of reward over time for the agent on the Test set. The model received about an 80 percent increase on one currency exchange, and 50 percent decrease on the other. Overall on the 4 test currency exchange the Agent had a total percent increase of 53 percent.





Over this 3-year test period the model ended up with about a 50 percent increase in profits. While this is fairly great news as it can be seen from the graphs the model isn't ideal. It still has significant losing trades and has inconsistency and oscillations seen during training between performing very well, and sub-par.

6 Conclusion/Future Work

Overall from these results it shows that it is possible to train an agent to trade on the market without the need to give explicit human tailored risk minimization heuristics. Instead a model can learn how to navigate this environment to maximize rewards and produce profitable trades. However this model is far from ideal. This model does not take into account any external factors other than prices and price patterns. Therefore it will undoubtedly fail to predict any down or upturns due to news, or human emotions.

However in lieu of this I believe improvements can still be made to this model to allow it to be a viable and more robust trader. First of all I believe that to extend this beyond the Forex market the model should also be able to trade stocks and therefore request how many shares to place an order with. This will allow the bottleneck of the model needing to spend multiple time steps buying up a stock if it deems it profitable and instead do it in one.

Also comparing the results of my agent to those presented in a Lu's paper [1], my results were not as strong. This is due to the fact human heuristics of risk are already fairly good and thus training a model to better learn this will produce better results. However I believe a model might still be able to learn on its own without these heuristics, perhaps with more training data and more feature than the few presented here.

7 Contributions

As this is a one person team, no explicit description for each team member is needed.

References

[1] Lu, D.W., 2017. Agent inspired trading using recurrent reinforcement learning and lstm neural networks. arXiv preprint arXiv:1707.07338.

[2] Jeong, G. and Kim, H.Y., 2019. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. Expert Systems with Applications, 117, pp.125-138.

[3] Gao, X., 2018. Deep reinforcement learning for time series: playing idealized trading games. arXiv preprint arXiv:1803.03916.

[4] Hausknecht, M. and Stone, P., 2015. Deep recurrent q-learning for partially observable mdps. arXiv preprint arXiv:1507.06527.