CS229 Final Report: Predicting Cycles in the Foregin Exchange Market

Manuel R. López-Morales* Nicholas Shelly† Inna Brodkin‡

Autumn Quarter 2012

Abstract

Professional Foreign Exchange (FOREX) traders successfully predict cycles in the market. We hypothesize that a machine learning algorithm would be able to identify those patterns as well. In the current project, we develop a trading algorithm which takes as input a segment of the tick-by-tick history of the bid-ask values of the Euro-U.S. Dollar (EUR/USD) and makes the decision of buying or selling the pair with the purpose of making a profit.

The project has three main phases: first, to acquire and manipulate raw data to train the algorithm; second, to implement a Neural Network (NN) to predict whether or not the most recent price is a local optimum in a cycle; third, to implement a Markov Deicison Process (MDP) algorithm to make the decision of buying or selling the pair based on the NN prediction.

We predict the first and second derivatives of the price trends using a NN trained online. The training is done with the latest first and second derivatives of the trends that could be computed. In our case, the latest training occurs with the trend derivatives found 20 ticks before the tick whose trend derivatives we are predicting.

The NN successfully predicted 73.21% of the price cycles of the EUR/USD pair between November 5th and 9th, 2012. The MDP was not able to take advantage of the predictions to make a profit. Nevertheless, a hard-coded logic was able to generate a profit of 0.3

1 Training

We chose to do online training of the NN in order to use the latest patterns found to predict the current trends. The training proceeds as follows: process the latest data to find the trend derivatives from 20 ticks ago, show these "real" values to the NN, and use backpropagation to update the weights in all nodes.

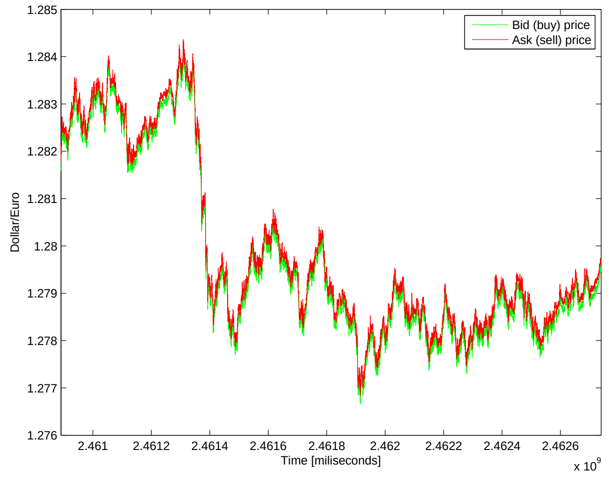

Figure 1: EUR/USD tick-by-tick bid-ask prices on November 5th, 2012

1.1 Data

The data used during development, shown in Figure 1, consists of tick-by-tick bid-ask prices of the EUR/USD pair relative to time. The data shown corresponds to one day of trading and contains 40,812 ticks. One tick represents a change in bid-ask. The frequency of the change in the price depends on the liquidity of the market, or the volume of the pair being bought and sold by all participants. It is important to note that access to high-quality data is expensive. Most of the freely available data is trimmed. The data used in this project was found freely in histdata.com and, even though it does not contain large gaps, the accuracy of the values is not guaranteed.

1.2 Cycle Detection

The bid and ask values tend to move together with a constant difference, called spread. To detect cycles, we analyze the average of the values. As Figure 1 shows, the data contains vast fluctuations. The identification of data points that belong to cycles is not so straightforward. Our approach to identifying points that mark a minimum or maximum in the general trend was to smooth the data using Locally Weighted Linear Regres-

*Ph.D. Candidate, Department of Aeronautics and Astronautics, Stanford University; mlopez14@stanford.edu

†M.S. Student, Computer Science Department, Stanford University; nshelly@stanford.edu

‡Software Engineer, Ketchum Trading; inna.brodkin@gmail.com

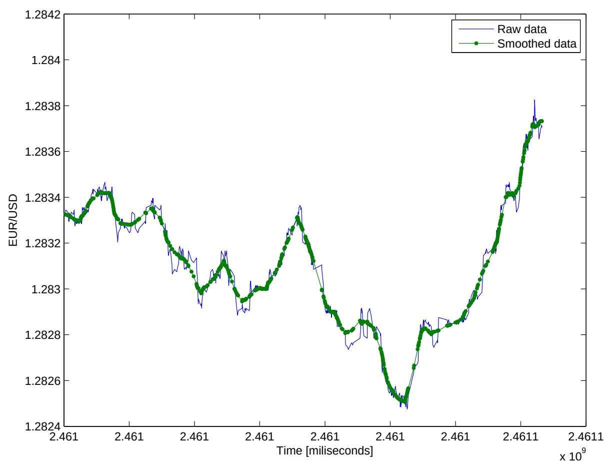

Figure 2: Comparison of raw data and smoothed data

sion (LWLR) on selected data points around the tick being analyzed. Figure 2 shows a comparison between the raw data and the smoothed data.

Interestingly, by modifying the weighting in the LWLR, it would be possible to capture cycles of different sizes and, hence, train the algorithm to identify large or small-scale cycles.

The final stage in the cycle detection for the training data is identifying the points near the local minima and maxima – using the first derivative of the smoothed data– and their corresponding tendency –using the second derivative of the smoothed data. In this process there is another arbitrary threshold introduced: the range of the derivatives used to label data points as local optima. We can see that the labeling ignores small fluctuations in the major price changes, so only a potentially profitable cycle is labeled as a cycle. The labeling of local maxima or minima is not used in the training, only the first and second derivative values are used for the training. The detection of local optima is only a measure of the effectiveness and validity of using derivatives to detect and predict cycles.

Figure 5 shows the result of the classification of points as belonging to local maxima, minima or neither by using the entire data set.

2 Predicting

We used a feedforward Neural Network to predict the first and second derivatives of the price trend. For the results reported here, we used a network with 5 layers and 5 nodes. The learning rate was 0.07.

An interesting result we obtained from the NN was that the derivatives of the price trend at tick i –which were predicted right after training with the price derivatives at tick $i-20$– are very similar to the known derivatives at tick $i-20$. In other words, the predicted values seem to simply lag the training labels with a constant shift. However, upon closer examination, it becomes apparent that the shift of the values is not constant. In some occasions, the predicted values match the behavior of the training values; at other times, the predicted values lag significantly. A good way to see this this variation in the lag is by comparing the peaks of the training and predicted values in Figures 3 and 4. The spacing of the ticks in time is almost constant in Figures 3 and 4, so the change in lag cannot be attributed to the mere irregularity of the ticks in time.

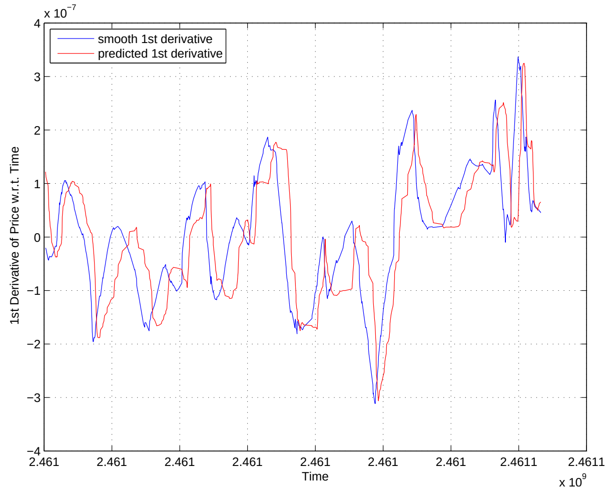

Figure 3: Comparison of 1st derivative values from smoothed data and the NN predictions

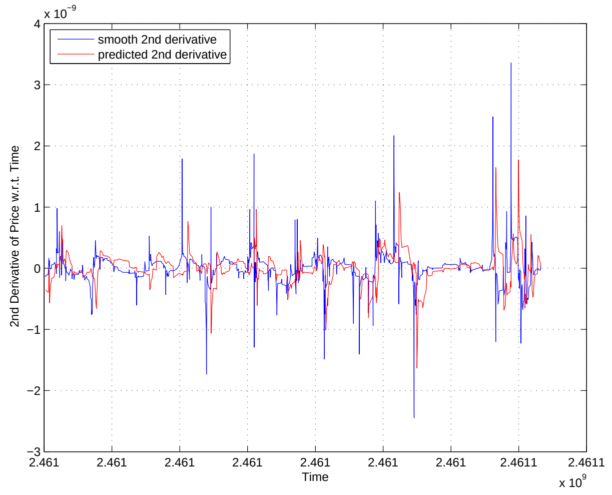

Figure 4: Comparison of 2nd derivative values from smoothed data and the NN predictions

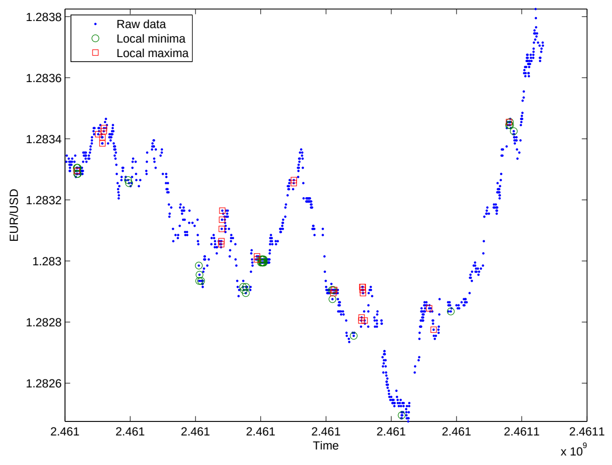

Figure 5: Local optima recognition using the derivatives of the smoothed data

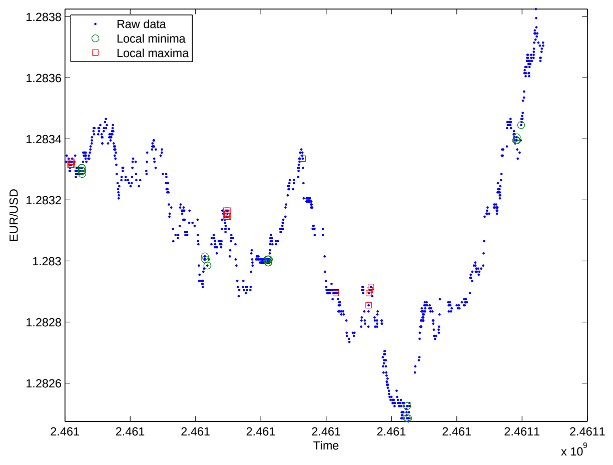

Figure 6: Local optima recognition using the derivatives predicted by the NN

2.1 Cycle Prediction

Figure 6 demonstrates the predictive capabilities of the NN. By using only previous tick values and constant online training, the NN is able to identify upward and downward cycles. It is important to note that the trading portion of the algorithm does not act on these labels. The labels shown here simply pinpoint when the first derivative is close to zero and the second derivative is significantly positive or negative. The fact that the NN can predict accurately some local optima adds validity to the derivative predictions and, hence, merits the inclusion of these predictions in the state-space of the Markov Decision Process (MDP) used for making the decision of selling, buying or holding.

The NN predicted 73.21% of the price cycles between November 5th and 9th that were detected using the LWLR. We counted a cycle as detected when the tick that the NN labeled as being a local maximum or minimum was within 4 ticks of a tick labeled in the same way using the LWLR techniques.

3 Trading

To act upon the predicted trends, we implemented an MDP algorithm. We were expecting the MDP to account for the lack of accuracy in some of the predictions. Given that the focus of our project was on the prediction of the cycles, the MDP described here was our first iteration in its design and a naive implementation. A more current version of the model is able to make a profit, albeit unreliably, by changing the meaning of the transition matrix.

3.1 Markov Decision Process

The current position[1]and the prediction of the first and second derivatives were the only components in the state description. In order to bias the model towards correlating the making of a profit when buying during a local minimum and selling during a local maximum, the states were discretized coarsely.

3.1.1 State Space

The position was discretized into three values: negative, close to zero, and positive. The predicted derivatives were normalized with respect to their own standard deviation (their average can represent trends we don't want to remove) and then categorized. The state of the normalized first derivative was 1 if it was within 0.05 standard deviations away from zero and 2 otherwise. The state of the normalized second derivative was labeled 1 if it was negative and farther than 0.7 standard deviations away from zero (meaning the cycle is strongly negative); 2 if it was within 0.7 standard deviations away to zero; 3 if it was positive and farther than 0.7 standard deviations away from zero (meaning the cycle is strongly positive).

The possibilities of action were only three: buy, sell or do-nothing. As a first iteration, we constrained the model to As it can be seen, the state space consisted of only 9 states and 3 actions. Hence, the value iteration algorithm converged rapidly.

3.1.2 Rewards

The rewarding scheme aimed to encourage the policy that makes a profit. Rewards were given at the state in which the policy made a profit and at the state in which the policy entered the market before making the profit. For example, if the policy decided to buy while at state

[1]Here we use the term position, for brevity, as equivalent to unrealized profits and losses. In other words, position in this report refers to the profit or loss that would be made if the current assets being held were sold immediately

3, held on to the purchase, and then sold while at state 9, both states 3 and 9 received a reward. The converse also applied, although the negative reward was smaller than the positive one. The positive reward was 0.5, the negative one was -0.2 .

Additionally, to encourage the algorithm to hold on to a purchase while its potential profit increases, the state in which the policy decided to do-nothing received a reward when its position increased in value. If its position decreased in value, it received a negative reward. In this case, the reward values were 0.001 and -0.001 .

3.1.3 Performance

The MDP always chose the do-nothing policy despite the fact that in its training all of the states were reached and all rewards were given.

3.2 Alternatives

We tried two alternative ways of making a profit given the high-quality of predictions the NN outputs. We hard-coded a decision and are implementing an alternative take on the Markov Decision Process.

3.2.1 Hard-Coded Decision

As Figure 6 demonstrates, the prediction from the NN can be used to label local optima. Then, we hard-coded the decision to enter the market –buy or sell– at those labeled points, do-nothing until the predicted first derivative reached a value close to zero, and liquidate the current position (e.g. sell if we bought before) if the predicted second derivative indicated a change in the trend. With this hard-coding, we achieved a profit of 0.3% for the week of data analyzed. Even though this is far from being machine learning, the encouraging results add practical validity to the predictions. With this method, the most profitable position was 0.35%, and the largest loss was -0.2%. The percentages are with respect to the initial investment.

3.2.2 Alternative MDP

A major flaw we now recognize with our initial iteration in the MDP design is that we are assuming that the states are in some way controllable by the system or interrelated. By the nature of the FOREX market, a combination of the first derivative, second derivative, and current price is almost completely independent of another such combination. As a consequence, the new MDP we are working on contains a transition matrix linking the state space described in Section 3.1.1 and a profitability state, not the state space itself. Hence, the transition matrix contains the probability of going from a price and first and second derivative state to a profitability state. The preliminary results of this algorithm are more encouraging than those from the Hard-Coded Decision.

4 Conclusions

We developed a technique to pre-process and analyze FOREX price data that yields practical predictions regarding the trend of the price. The data used comprised of prices of the EUR/USD pair for 120 hours starting on November 5$^{\text{th}}$, 2012.

To smoothen the data, we used Locally Weighted Linear Regression (LWLR) with a parameter that allowed us to control the time-scale of the trends to be observed. To predict the first and second derivatives of the price trend, given the latest price available, we used a Neural Network (NN). It was able to predict 73.21% of the cycles identified with the LWLR.

To assess the applicability and practical validity of the predictions, we used them, together with the latest price available, as the state space in a Markov Decision Process (MDP). Our first implementation of the MDP was not successful in making a profit; the structure of the transition matrix encouraged the policy of do-nothing to be the preferred action. Nevertheless, two alternatives to circumvent this problem arose.

Hard-coding a decision based on the NN predictions of the price derivatives yielded a modest profit of 0.3% during the week of data. This result confirms further the usability of the predictions. In addition, an alternative MDP is in the works. The transition matrix in this MDP relates price and derivative states with profitability states.

To conclude, we have shown that predicting the first and second derivatives of the FOREX market is viable and usable to make a profit. The automation of the decision to buy or sell is a different matter.

Future work involves optimizing the parameters of the new MDP and implementing the sequence of training, predicting and trading described here in a real scenario.

5 Acknowledgements

We thank Prof. Ng and the TA team for their excellence in teaching. The notes were an essential resource in the making of this project. Additionally, the book "Pattern Recognition and Machine Learning" by Christopher M. Bishop, 2006, was the main resource for understanding the implementation of Neural Networks.