DIPLOMA PROJECT

Forecasting Foreign Exchange Rates with Neural Networks

Project Report

A project presented to the Artificial Intelligence Laboratory
School of Computer and Communication Sciences
Swiss Federal Institute of Technology
(EPFL)

Supervisor: Marius-Călin Silaghi, EPFL
Professor: Prof. Boi Faltings, EPFL
Reviewer: Prof. Kilian Stoffel, University of Neuchâtel

By

Eric Simon
Computer Science Institute
University of Neuchâtel
E-mail: eric.simon@unine.ch

Lausanne, 15 February 2002

Abstract

This document reports empirical results that tend to confirm the applicability of a simple neural network model to the prediction of foreign exchange rates. First, we introduce the foreign exchange market, then the neural network model. Simple weekly average indicators are fed to a three-layer perceptron to capture the movement of exchange rates. The exchange rates between five currencies quoted against the U.S. Dollar are forecast by the model for different time periods and segmentations of the data, using different input models and indicators. The results are evaluated in terms of Normalized Mean Squared Error and profit, using simple trading strategies. The results show that this model is promising for forecasting and requires further investigations to try to improve the meager results regarding the profit. The inclusion of political factors is especially interesting. After presenting and discussing the experimental results, a discussion on further research concludes the report.

"Bistromathics itself is simply a revolutionary new way of understanding the behaviour of numbers. Just as Einstein observed that space was not an absolute but depended on the observer's movement in time, and that time was not an absolute, but depended on the observer's movement in space, so it is now realized that numbers are not absolute, but depend on the observer's movement in restaurants."

Douglas Adams, "The Hitchhiker's Guide to the Galaxy"

Contents

1 Introduction **1**
- 1.1 Currency Exchange . . . 1
- 1.2 Currency Exchange Rates . . . 2
- 1.3 Foreign Exchange Market . . . 3
- 1.4 Neural Networks as a Forecasting Tool for Forex . . . 3

2 Neural Network Model **5**
- 2.1 Formal Neuron . . . 5
- 2.2 Perceptron . . . 6
- 2.3 Multilayer Perceptron . . . 7
- 2.4 Backpropagation Algorithm . . . 8
- 2.5 Other Neural Networks . . . 10

3 Data **11**
- 3.1 Historical Exchange Rates . . . 11
  - 3.1.1 The Euro Problem . . . 12
  - 3.1.2 Data Retrieval . . . 13
  - 3.1.3 Data Preparation . . . 14
- 3.2 Other Factors . . . 15
- 3.3 Time Periods . . . 15
- 3.4 Data Segmentation . . . 16
- 3.5 Input Models . . . 17
  - 3.5.1 Daily Averages . . . 17
  - 3.5.2 Weekly Averages . . . 18
  - 3.5.3 Progressive Averages . . . 18
  - 3.5.4 Long Progressive Averages . . . 19
  - 3.5.5 Other Input Models . . . 19

4 Evaluation of Results **21**
- 4.1 Measurement of Neural Networks . . . 21
  - 4.1.1 NMSE . . . 21

4.1.2 Profit 22
4.2 Trading Strategies 23
4.3 Potential Problem with Results 24
4.4 Comparison with Related Work 25

5 Results and Discussion **27**
5.1 Influence of Time Period 28
5.2 Influence of Segmentation 35
5.3 Influence of Input Number 39
5.4 Influence of Input Model 43
5.5 Influence of Other Factors 50
5.6 Comparison with Related Work 57

6 Conclusions and Further Work **59**

A Implementation **61**
A.1 General Conception 61
A.2 UML Diagrams 62

B User's Manual **69**
B.1 What is FXAgent? 69
B.2 Interface 70
B.2.1 Information Panel 70
B.2.2 Graph Panel 71
B.2.3 Data Panel 71
B.2.4 Control Panel 72
B.2.5 Result Window 73
B.2.6 Configuration Panel 73
B.2.7 Trade Panel 74

Bibliography **75**

Index **77**

List of Figures

2.1 A `6-3-1` multilayer perceptron 7
2.2 The sigmoid $\frac{1}{1+e^{-\Sigma}}$ 8

3.1 Exchange rates for USD/HKD 14

4.1 NMSE and profit 22
4.2 Exchange rates for USD/EUR over the last ten years 25

5.1 Influence of time period on USD/CHF 29
5.2 Influence of time period on USD/EUR 30
5.3 Influence of time period on USD/GBP 31
5.4 Influence of time period on USD/JPY 32
5.5 Influence of time period on USD/ZAR 33
5.6 Influence of segmentation on USD/CHF 36
5.7 Influence of segmentation on USD/EUR 36
5.8 Influence of segmentation on USD/GBP 36
5.9 Influence of segmentation on USD/JPY 37
5.10 Influence of segmentation on USD/ZAR 37
5.11 Influence of input number on USD/CHF 40
5.12 Influence of input number on USD/EUR 40
5.13 Influence of input number on USD/GBP 40
5.14 Influence of input number on USD/JPY 41
5.15 Influence of input number on USD/ZAR 41
5.16 Influence of input model on USD/CHF 44
5.17 Influence of input model on USD/EUR 45
5.18 Influence of input model on USD/GBP 46
5.19 Influence of input model on USD/JPY 47
5.20 Influence of input model on USD/ZAR 48
5.21 Influence of boolean factors on USD/CHF 51
5.22 Influence of boolean factors on USD/EUR 52
5.23 Influence of boolean factors on USD/GBP 53

5.24 Influence of boolean factors on USD/JPY 54
5.25 Influence of boolean factors on USD/ZAR 55

A.1 UML diagram for the agent applet 62
A.2 UML diagram for the MVC 63
A.3 UML diagram for the message 64
A.4 UML diagram for the MVC view 65
A.5 UML diagram for the engine 66
A.6 UML diagram for the data 67
A.7 UML diagram for the strategy 67

Chapter 1

Introduction

This chapter presents some background about the foreign exchange market and introduces the neural network as a tool for forecasting economic time series.

First, the concepts of currency exchange and currency exchange rate are explained, followed by some elements about the foreign exchange market. Then, the problem of time series prediction is developped in the context of neural networks, and some existant research in this field is mentioned.

Some books that were consulted to write this introduction are mentioned in the bibliography (*see [4], [5], [9] and [10]*). There is also a good introduction to Forex on OANDA.com's web site [1].

1.1 Currency Exchange

Currency exchange is the trading of one currency against another, and usually takes place as follows.

Consumers typically come into contact with currency exchange when they have to convert their "home currency" into the currency of a foreign country they intend to travel to. They may also purchase goods in a foreign country via the Internet, in which case the amount paid in the foreign currency will have been converted to their home currency on their credit card statement. Those transactions are relatively small.

Businesses typically have to deal with currency exchange when they conduct trade outside their home country. For example, if they export goods to a foreign country

and receive payment in the currency of that country, the payment must often be converted back to the home currency. The amounts converted each year can be huge.

Investors and speculators have to care about currency exchange when they trade in any foreign investment, like bonds, bank deposits or others.

Investors and speculators also trade currencies directly in order to benefit from movements in the currency exchange market. For example, if an American investor believes that the Swiss economy is strenghtening, and as a result expects the Swiss Franc to appreciate in value (go up) relative to the U.S. Dollar, the investor may want to buy Swiss Francs and take what is referred to as a long position. Similarly, if the Swiss Franc is believed to go down, the investor may want to sell Swiss Francs against U.S. Dollars, thus taking a short position. Those transactions are often conducted to take advantage of movements in the market over very short time periods, hours or even minutes.

Commercial and investment banks also participate in the currency market, as well as governments and central banks when they try to intervene for adjusting economic imbalances.

1.2 Currency Exchange Rates

Currency exchange rates are always quoted for a currency pair using ISO code abbreviations[1]. For example, EUR/CHF refers to the two currencies Euro and Swiss Franc. The first is the base currency and the second the quote currency. Therefore, the EUR/CHF exchange rate specifies how many Swiss Francs you have to pay to buy one Euro, or conversely how many Swiss Francs you obtain when you sell one Euro.

A currency exchange rate is typically given as a pair consisting of a bid price and an ask price, the difference between those two beeing the spread.

[1] ISO codes are available on OANDA.com's web site [1].

1.3 Foreign Exchange Market

The currency exchange market, also referred to as Forex market[2], was established in 1971, when floating exchange rates began to materialize.

In terms of trading volume, Forex is the world's largest market, with daily trading volumes in excess of $1.5 trillion U.S. Dollars (source: OANDA.com [1]). It is by far the most liquid market[3].

Today, 85% of all Forex transactions involve a few major currencies: the U.S. Dollar (USD), Japanese Yen (JPY), Euro (EUR), Swiss Franc (CHF), British Pound (GBP), Canadian Dollar (CAD) and Australian Dollar (AUD) (source: OANDA.com [1]). Most of the currrencies are rated only against the U.S. Dollar. Trading between two non-dollar currencies usually occurs by first trading one against USD and then trading the USD against the second non-dollar currency (GBP/EUR or EUR/CHF are traded directly). The exchange rate in this situation is then referred to as cross rate.

Forex is a true 24-hour market, 5 days a week. Actually, trading is possible 7 days a week on the Internet.

1.4 Neural Networks as a Forecasting Tool for Forex

The forecasting of foreign exchange rates poses many theoretical and experimental challenges. Foreign exchange rates are influenced by many correlated economic, political and psychological factors. Like many economic time series, Forex has its own trend, cycle and irregularity.

As mentioned by Yao *et al.* [18], classical time series analysis doesn't perform satisfactorily on economic time series. Hence the idea of applying non-linear models, like neural networks, to this problem.

Many consider White to be the first to have applied a two-layer neural network on financial time series in 1988. He did it on series of IBM stocks to test the Efficient Market Hypothesis[4]. More recent work reported mixed evidence as to forecasting results of neural networks (*see Herbrich et al. [8] pp 13-14*). It seems

[2]Forex and FX are acronyms for Foreign Exchange used to refer to this particular market. Those acronyms will be used from now on in this document.

[3]A liquid market is a market allowing the buying or selling of large quantities of an asset at any time and at low transaction costs.

[4]The Efficient Market Hypothesis, known as EMH, states that the market fully reflects all of

to work as well as classical statistical methods, sometimes better, but under certain conditions.

Yao *et al.* [16] demonstrated that a simple backpropagation perceptron and a simple set of technical indicators serve well as a predictive model for a six month forecasting period for CHF/USD, and that significant paper profit could be made with simple trading strategies. However, with efficient markets, it is not easy to make profits using time series neural networks, and some currency pairs behave differently than others [18]. Also, capturing the behaviour of Forex depends not only on which indicators are chosen, but also on which period is considered the most representative of the whole system [17].

the available information and prices are adjusted fully and immediately once new information becomes available.

Chapter 2

Neural Network Model

This chapter presents briefly the neural network model used in this study: the multilayer perceptron. First, we explain the concept of formal neuron, the base constituent of neural networks. Then some historical background is followed by an introduction to the perceptron and its extension, the multilayer perceptron. Finally we give some elements about the underlying learning algorithm: backpropagation of errors.

2.1 Formal Neuron

Neural networks are based on a simple model of neuron, called the formal neuron. It is directly inspired from the real neuron present in our nerve system.

A real neuron usually consists of three parts: a dendrite, receiving inputs from other neurons or from an external stimulus; a soma, that performs a non-linear processing of the signals from the dendrites; an axon, transmitting the output signal to other neurons or organs. The connection between two neurons is made through a synapse, transforming the electrical stimulus travelling along the axon into a chemical one that serves as activator for the other neuron's connection.

Similarly, the formal neuron works in three steps: an integration step, during which all inputs x_i are weighted with parameters $w_{i,j}$; a non-linear step that transforms the weighted sum of inputs into a non-linear function, depending on the type of output the network should provide; a propagation step, where the output is propagated to subsequent neurons or objects.

Therefore, the output of a single formal neuron is defined as

$$y = f\left(\sum_{j=0}^{n-1} w_{i,j}x_j - \theta_i\right)$$

where y is the output, $f(x)$ is a non-linear function, n is the number of input lines of neuron i and θ_i is a threshold. The threshold can be replaced by another input (*see Gerstner [6]*) so we can rewrite the previous formula as

$$y = f\left(\sum_{j=0}^{n} w_{i,j}x_j\right)$$

$f(x)$ can be chosen to be a strict threshold, producing a binary output, or a non-linear activation function, like $tan(x)$ or the sigmoid that was chosen for this study (*see Section 2.3*).

Each formal neuron constitutes what is often referred to as a processing element (PE).

2.2 Perceptron

As mentioned in the introduction (*see Section 1.4*), the first idea that comes to mind when trying to predict time series is the multilayer perceptron using back-propagation as a learning algorithm. That is what Yao *et al.* [18] did. This is also the model we chose for this study.

The perceptron, first invented by Rosenblatt in the late fifties, consists in a single layer of processing elements (PEs). It was soon to be extended to multiple layers. For more information about different neural network models and their associated learning algorithms, see Gerstner [6].

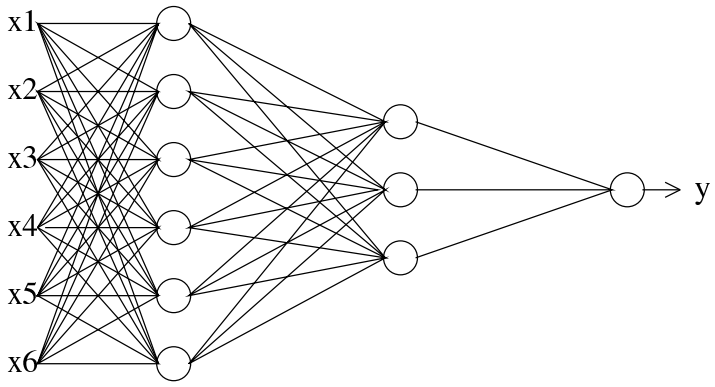


Figure 2.1: A `6-3-1` multilayer perceptron

2.3 Multilayer Perceptron

The perceptron chosen for this study contains three layers[1]. The input layer contains as many PEs as inputs, depending on the data input model (*see Section 3.5*). The hidden layer contains an arbitrary number of PEs, often a fraction of the number of PEs in the input layer (half as much, for example). The output layer contains as many PEs as there are outputs. For this specific problem, the number of outputs is the number of currencies to predict.

For example, a network for predicting the currency pair USD/CHF taking the six previous weeks' exchange rate averages in input would have the structure `6-n-1`, that is six PEs in the input layer, an arbitrary number `n` in the hidden layer, and one PE in the output layer (*see Figure 2.1 for a* `6-3-1` *perceptron*). This notation, inspired by Yao *et al.* [18], will be used in the rest of this document.

Each PE uses the following sigmoid (*see Figure 2.2*) as activation function,

$$\frac{1}{1+e^{-\gamma\Sigma}}$$

where γ represents a gain (the "slope" of the sigmoid) and Σ is the result of the

[1] The best number of hidden layers depends on the task the network has to perform, and there is no rule for determining it in general. Some tests were made at the beginning of this project with more than three layers, but at first sight it didn't seem to improve the performance, so the idea was abandonned.

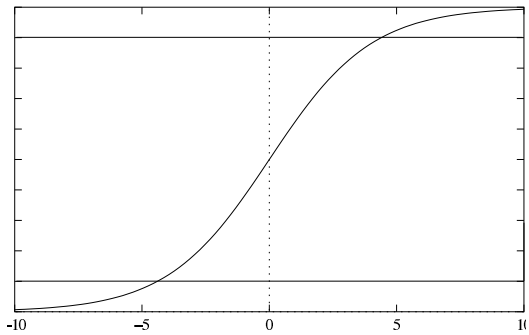


Figure 2.2: The sigmoid $\frac{1}{1+e^{-\Sigma}}$

integration phase, that is

$$\Sigma = \sum_{j=0}^{n} w_{i,j} x_j$$

where n represents the number of inputs, $w_{i,j}$ represents the weight associated with PE i and input j, and x_j represents the jth input.

The given sigmoid gives an value between 0 and 1, which forces a normalization of the data (*see Section 3.1.3*).

2.4 Backpropagation Algorithm

The backpropagation learning algorithm, derived from the Widrow-Hoff algorithm, appeared in the seventies. It consists in propagating the errors on the output layer back to the first layer to correct the weights of all layers in-between. The algorithm is guaranteed to converge towards a local minimum [6]. There are methods that find better minima, like NOVEL by Wah *et al.* [15].

Here is the backpropagation algorithm in pseudo-code:

Initialize weights $w_{i,k}^q$ with random values $\in [-0.5; 0.5]$
for $(\sigma = 0, ..., eN)$ **do**
  Compute output by propagation:
  $\hat{y}^0 = x$
  for $(q = 1, ..., o)$ **do**
    for $(i = 1, ..., n_q)$ **do**
      $\hat{y}_i^q = f\left(\sum_k w_{i,k}^q \hat{y}_k^{q-1}\right)$
    end for
  end for
  for $(i = 1, ..., n_o)$ **do**
    Compute output error: $\varepsilon_i = \frac{1}{2}(y_i - \hat{y}_i^s)^2$
    Compute gradient δ_i^s for the output layer:
    $\delta_i^s = \varepsilon_i f'(\hat{y}_i^o)$
  end for
  Backpropagatation:
  for $(q = o - 1, ..., 1)$ **do**
    for $(i = 1, ..., n_q)$ **do**
      $\delta_i^q = f'\left(\hat{y}_i^q\right) \sum_k \delta_k^{q+1} w_{i,k}^{q+1}$
    end for
  end for
  Update weights:
  for $(q = 1, ..., o)$ **do**
    for $(i = 1, ..., n_q)$ **do**
      for $(k = 1, ..., n_{q-1})$ **do**
        $w_{i,k}^q = w_{i,k}^q + \eta \delta_i^q \hat{y}_k^q + \alpha \Delta w_{i,k}^q$
        $\Delta w_{i,k}^q = \eta \delta_i^q \hat{y}_k^q$
      end for
    end for
  end for
end for

$\hat{y}^q$ represents the output of layer q, and by extension $\hat{y}_i^q$ the ith output of layer q. x represents the input of the first layer. N is the number of samples, n_q is the number of PEs in layer q, o is the number of layers. $w_{i,k}^q$ represents the weight associated with PE i and input k, $f(x)$ is the activation function, $f'(x)$ its derivative. Important parameters of this algorithm are the number of **epochs** e, the **momentum** α and the **learning rate** η.

2.5 Other Neural Networks

It may be interesting to try other neural network models to make predictions. The perceptron is certainly a good candidate because of its simplicity, but recursive neural networks also give interesting results [7]. Diffusion networks [11] could also be investigated, although the level of mathematics involved (stochastic methods) is more complicated.

Chapter 3

Data

This chapter presents the data that is specific to the current problem: foreign exchange rates.

The first part mentions some issues related to foreign exchange rates, and the solutions envisaged in this study. Those issues comprise the problem of the Euro, the availability of historical data, its retrieval and the preparation of this data necessary for it to be used in our model.

The second part begins with a brief word about other factors that were included in this study, and explains how the data was actually used in the model, that is how much of it was taken, how it was segmented and how it was fed to the perceptron. For each problem, different solutions that were considered are described, and some other possibilities are proposed.

3.1 Historical Exchange Rates

As stated in the introduction (*see Section 1.2*), a currency exchange rate usually consists in two numbers, for the bid price and the ask price respectively. For simplicity, we consider a rate as a single number in the model, and neglect the spread.

Trading takes place on a 24-hours market, and rates change very quickly. It's therefore necessary to fix an arbitrary closing hour and take into account only an average of the exchange rates for the period between two closing hours. Here, the data is constituted of daily averages (*see Section 3.1.2*).

In practice, there is always some transaction cost involved. We didn't take this into account. The cost of transaction is typically in the order of 1% of the exchange rate when using financial services on the Internet. Hence, trading too often will lead to unnecessary costs, and that's why our strategies (*see Section 4.2*) trade on a weekly basis.

Historical exchange rates were retrieved from the Internet (*see Section 3.1.2*) for currencies quoted against USD. There are two reasons that dictated the choice of restricting the pairs to those with USD as the base currency. First, it was a bit more difficult to compute pseudo-exchange rates for the Euro using other base currencies than the U.S. Dollar (*see Section 3.1.1*). Secondly, as stated in Section 1.3, most currencies are rated only against USD, cross-rating taking place for trading two non-dollar currencies. This way, we avoid unnecessary redundancy in the data.

3.1.1 The Euro Problem

The Euro exists only since 01.01.1999[1]. Therefore, in order to still have a valid model after this date when training it with non-euro data, it was necessary to find a way of computing pseudo-exchange rates for the chosen period.

The first method is to use the exchange rates for the ECU. This is convenient because the Euro was linked 1:1 to the ECU when the European Council voted the irrevocable conversion rates between the Euro and the participating currencies[2]. However, three countries are part of the ECU basket (Britain, Denmark and Greece) that are not part of the Euro group, and two other countries (Finland and Austria) are part of the Euro group but not part of the ECU basket.

A superior method [3] is to calculate a weighted average of the Euro constituent currencies for the pre-1999 period using the following equation:

$$x_{e,j} = \exp(z_{e,j} + \sum_{i=1}^{11} w_i \log x_{i,j})$$

where $x_{e,j}$ is the exchange rate of the Euro e with respect to currency j, $x_{i,j}$ is the exchange rate of one of the eleven constituent currencies i of the Euro with

[1] This is not strictly true, the Euro existed before this date, and exchange rates for the various currencies participating in the Euro were already fixed. However, the currency was introduced officially at this date.

[2] The participating currencies are FRF, ITL, ESP, NLG, BEF, IEP, FIM, ATS, PTE and DEM.

respect to currency j, and w_i is a weight. Finally, $z_{e,j}$ is a scaling factor that normalizes the exchange rate so it matches the actual Euro exchange rate on the date of its introduction. For example, when combining exchanges rates DEM/USD, FRF/USD, ITL/USD, and so on, a correction factor of $z_{EUR,USD} = -2.3586193$ must be used.

The weights are based on the size of international trade with a group of countries outside the Euro zone. For more details about this method, the weights, and the Euro in general, see Antweiler [3].

For this study, the second method was used to compute pseudo-exchange rates for the Euro against the U.S. Dollar (USD/EUR).

3.1.2 Data Retrieval

Historical exchange rates are available from some web sites in the form of daily averages. We chose to retrieve the data from OANDA.com [1]. It's easy to find historical data for the major currencies (*see Section 1.3*) quoted against USD. The only exception is for the pair USD/AUD, where data is available only from 1993, so we decided not to include this pair.

The currency pairs for which the data has been retrieved are USD/CAD, USD/CHF, USD/DKK, USD/EUR, USD/GBP, USD/HKD, USD/JPY, USD/NOK, USD/NZD and USD/ZAR.

There are other exchange rate systems than the freely floating one adopted by industrialized countries in the seventies. Currencies that aren't floating, like the Yuan[3], were avoided, except HKD. We noticed HKD was managed floating only when we obtained a horizontal line (or almost) for USD/HKD when doing the tests. It was left in the model as an illustration of how such a currency behaves (*see Figure 3.1*). The predicted line looks much lower than the real one (*see Section 4.3*), but beware of the scale. The same applies for all plots in the rest of the document.

Perl scripts were written to automate the process of downloading and formatting the data. From a practical point of view, the data consists in an ASCII file containing a column for each currency pair or other indicator.

[3]The Yuan, the currency of China, is what is referred to as managed floating, like the Hong-Kong Dollar (HKD).

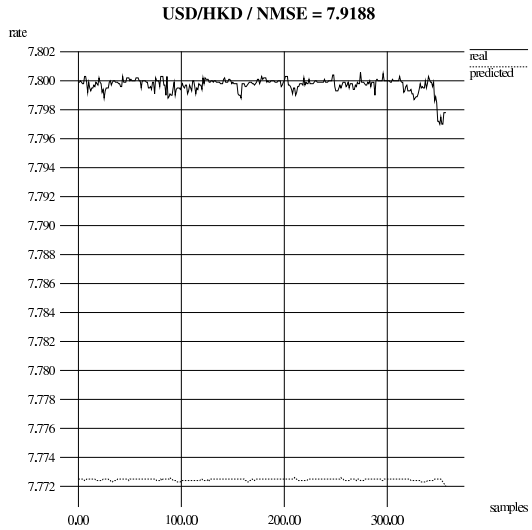


Figure 3.1: Exchange rates for USD/HKD

3.1.3 Data Preparation

Since the sigmoid chosen as activation function gives an output in the interval $[0;1]$, it was necessary to normalize the data into this interval (*see Section 2.3*). The data is then denormalized when displaying the results.

The formula used to normalize the data is the following,

$$y = \frac{x - min}{max - min}(\omega - \alpha) + \alpha$$

where x represents the number to normalize, y the result, min and max the minimum and maximum of the data over the whole set respectively, and α and ω the lowest and uppest bound of the interval respectively.

The denormalization formula is then the inverse of the above,

$$y = \frac{x - \alpha}{\omega - \alpha}(max - min) + min$$

In practice, the data is normalized into the interval $[0.1;0.9]$ to avoid working near the asymptot of the sigmoid (*see Figure 2.2, where the two horizontal lines show the boundaries of the interval*).

The boolean factors (*see Section 3.2*) were also normalized using the same formula, which means that with $\alpha = 0.1$ and $\omega = 0.9$, for the neural network `true` is represented by 0.9 and `false` by 0.1.

3.2 Other Factors

As mentioned in Section 1.4, Forex is influenced not only by economic, but also by political and psychological factors. The idea at the beginning of the project was to try to include the political factor by adding simple boolean indicators, giving information such as whether the government of the United States is Republican or Democratic, or whether an important war is going on.

It might be interesting to try to include also the psychological factor in a similar way, for example by adding a boolean indicator capturing the state of mind of the trader. Another idea would be to use a number to capture those factors, giving information like how sure is the trader that the market is in a good or bad period.

In this study, two boolean indicators were included:

- is the government of the USA Republican (`true`) or Democratic (`false`);
- is there a war[4] going on (`true`) or not (`false`).

3.3 Time Periods

The maximum time period chosen for training and testing the network ranges from 01.01.1992 to 31.12.2001, ten years in total. The problem was to find a balance between having enough currencies to trade with, and having enough data to train the network. It happens that historical exchange rates are usually not available before 1992 for many currencies. Also, it's likely that the market situation before 1989 was very different from what it has been afterwards[5], and we chose from the beginning not to take the risk of teaching the network a behaviour that is too likely to be different from what it is now. It would have been interesting to try, considering the results afterwards, though (*see Chapter 5*).

[4] Wars that were considered since 1992 are Yugoslavia and the so-called "War against terror" in retaliation after the attack of the World Trade Center in September 2001.

[5] Remember the events related to the collapse of the USSR, in particular the destruction of the Berlin Wall.

Another constraint for the time period was the Euro (*see Section 3.1.1*).

As mentioned in the introduction (*see Section 1.4*), the behaviour of Forex isn't constant over long periods, so that training a neural network over a particular time period is likely to produce bad results when the network is tested on a subsequent period. In other terms, the past isn't always the key to the present.

Yao *et al.* [18] proposed to use statistical methods to investigate this problem and try to find the best time period for training the neural network. This is well beyond the purpose of this study.

However, some empirical results are always interesting, so different time periods were used to train and test the perceptron:

from	to	duration
01.01.1992	31.12.2001	10 years
01.01.1992	31.12.1996	5 years
01.01.1997	31.12.2001	5 years
01.01.1999	31.12.2001	3 years *with real USD/EUR rates*

3.4 Data Segmentation

Yao *et al.* [18] made all their tests with the same data segmentation, based on a rule of thumb derived from their experience, namely 70% for training, 20% for cross-validation and the remaining 10% for testing. This is generally considered a good segmentation because it allows enough data for the training phase while keeping enough for the cross-validation so that the network usually generalizes best on out-of-sample data. This is also the default segmentation that was applied in this study for producing results.

However, playing with those numbers showed some influence of the segmentation on the results. That's why different segmentations were tried to study their impact on the network. The different segmentations are the following:

training	cross-validation	testing
70%	20%	10%
80%	15%	5%
90%	5%	5%

In the rest of this document, a segmentation of 70% for training, 20% for cross-validation and 10% for testing will be denoted `70-20-10`.

Wah *et al.* [14] have formulated the learning as a constrained optimization problem. Their method allow them to use all the available data for training, without sacrifying samples for the cross-validation. This would be a good improvement, considering the obtained results (*see Section 5.2*).

3.5 Input Models

As mentioned by Yao *et al.* [18], the trend of the market can be categorized using Dow's[6] categories, namely the major trend, the intermediate trend and the minor trend. The major trend lasts more than one year, the intermediate trend usually lasts from three weeks to three months, the minor trend is meaningless. That was what pushed the authors in the direction of moving averages. According to their study, this works well for certain currencies and certain training periods.

Moving averages is a well-known technique used to approach a curve while smoothing down its instability. A daily average is in itself a moving average of the exchange rate over a whole day. We used this approach too.

In the rest of this document, and in the implementation, an input model refers to the factors used as input for the network, as well as the target used for comparing the prediction against the reality. For example, using purely time-delayed daily averages and predicting the value one day ahead constitutes an input model.

Four input models were tested:

- time-delayed daily averages
- time-delayed weekly averages
- progressive averages (6 months back)
- long progressive averages (2 years back)

3.5.1 Daily Averages

This model consists in using the very daily averages downloaded from the Internet and feed them to the first layer of the perceptron. The target is the daily average one day after the last input.

[6] Charles Dow, father of Dow-Jones.

input #	name	value
1	D1	i
2	D2	$i-1$
3	D3	$i-2$
4	W1	$\mu(]i-7;i])$
5	W2	$\mu(]i-14;i])$
6	M1	$\mu(]i-30;i])$
7	M3	$\mu(]i-90;i])$
8	M6	$\mu(]i-180;i])$

Table 3.1: Progressive averages model, $\mu(I)$ is the average on interval I and i is the day of reference

This model should capture the minor trend only (*see Section 3.5*).

3.5.2 Weekly Averages

This model consists in computing weekly averages and feed them to the first layer of the perceptron. The target is the daily average one week ahead, regardless of which day that is.

This model should be able to capture the minor trend, but smoothing down it's instability (*see Section 3.5*).

3.5.3 Progressive Averages

Instead of using purely time-delayed time series, which are known to produce time-delayed responses, another input model was tested that consists in eight inputs of different size, named D1, D2, D3, W1, W2, M1, M3 and M6. They refer to daily averages for the current day, one day before, two days before, weekly averages for last week, last two weeks, monthly averages for last month, last three months and last six months respectively (*see Table 3.1*).

The target is the daily average one week ahead.

This model should be able to capture both the minor trend and the intermediate trend (*see Section 3.5*).

input #	name	value
1	D1	i
2	D2	$i-1$
3	D3	$i-2$
4	W1	$\mu(]i-7;i])$
5	W2	$\mu(]i-14;i])$
6	M1	$\mu(]i-30;i])$
7	M3	$\mu(]i-90;i])$
8	M6	$\mu(]i-180;i])$
9	Y1	$\mu(]i-365;i])$
10	Y2	$\mu(]i-730;i])$

Table 3.2: Long progressive averages model, $\mu(I)$ is the average on interval I and i is the day of reference

3.5.4 Long Progressive Averages

This is the same idea as the progressive averages (*see Section 3.5.3*), but going back two years (*see Table 3.2*) to try to capture also the major trend (*see Section 3.5*).

3.5.5 Other Input Models

It might be interesting to investigate further input models to capture different factors. There are all sorts of other models that are imaginable. The Java applet written to investigate the problem can easily be extended by adding new input models to it (*see Appendix A*). Among the things that could be tried are the following:

- using extrema along with averages to capture the instability;
- try to include a longer time period in the progressive average model, for example always going back as long in the past as possible, depending on the reference.

Chapter 4

Evaluation of Results

This chapter presents the methods that were applied to measure the performances of the forecasting model. First, we introduce the Normalized Mean Squared Error, that gives a measure of the fitting of the predicted curve. Then the notion of profit is developped. Finally, the concept of trading strategy necessary to define the profit is explained together with the two strategies that were applied in this study, and a comparison with related work concludes the chapter.

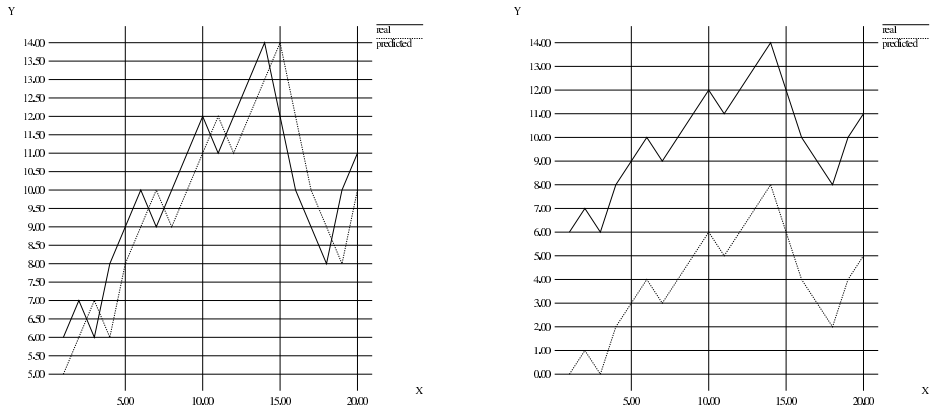
4.1 Measurement of Neural Networks

4.1.1 NMSE

A usual measure to evaluate and compare the predictive power of a neural network is the Normalized Mean Squared Error (NMSE). It is basically the Mean Squared Error (MSE) divided by the variance over the data set. Let E be the MSE, defined as

$$E = \frac{1}{|S|} \sum_{i \in S} (x_i - \hat{x}_i)^2$$

where S represents the data set, x_i the real value, $\hat{x}_i$ the predicted value and $|S|$ denotes the cardinal of S, that is the number of elements in the set, and let σ^2 be



(a) small NMSE but small profit (b) big NMSE but big profit

Figure 4.1: NMSE and profit

the variance over S,

$$\sigma^2 = \frac{1}{|S|} \sum_{i \in S} (x_i - \mu_S)^2$$

where μ_S is the mean of S. Then, the Normalized Mean Squared Error $\hat{E}$ is defined as

$$\hat{E} = \frac{E}{\sigma^2} = \frac{\sum_{i \in S} (x_i - \hat{x}_i)^2}{\sum_{i \in S} (x_i - \mu_S)^2}$$

4.1.2 Profit

The NMSE is a good indicator of whether the predicted curve fits well with the real one. However, the real aim of forecasting economic time series is of course the profits one can make using the model. The problem is that a generally good model in terms of NMSE can lead to bad results in terms of profit. For example, if the predicted curve is a bit late compared to the real one (*see Figure 4.1 (a)*), the NMSE will be small compared to a situation where the prediction is well below the real line but the trend follows the real trend (*see Figure 4.1 (b)*). However, the first situation is likely to produce worse results in terms of profit than the second one.

Therefore, it is necessary to find a measure of the profit the system is able to achieve. Yao *et al.* [18] defined one as follows,

$$\text{Profit} = \left(\frac{\text{money obtained}}{\text{seed money}}\right)^{\frac{52}{w}} - 1$$

where **money obtained** is the amount of money obtained on the last day of testing, **seed money** is the amount of money used for trading on the first day of testing, and w is the number of weeks of the testing period. Note that this value was defined for a strategy where trading takes place always the same day, at the end of the week, hence the exponent $\frac{52}{w}$ (there are 52 weeks in a year).

This is also the measure that was used in this study, with a variant for daily trading,

$$\text{Profit} = \left(\frac{\text{money obtained}}{\text{seed money}}\right)^{\frac{365}{d}} - 1$$

where d is the number of days of the testing period[1].

4.2 Trading Strategies

Of course, a profit cannot be defined without a related trading strategy. There are quite a lot of trading strategies used by people gambling on financial markets. The probably best-known is speculation[2].

It would be very interesting to implement trading strategies inspired from reality, but this would go much beyond the subject of the current project. Therefore, the two strategies chosen for evaluation are very simple. Both use the difference between the predictions to take a decision of the form

$$\text{if} \quad (\hat{x}_{t+1} - \hat{x}_t) > 0 \quad \text{then } \textbf{buy} \quad \text{else } \textbf{sell}$$

where $\hat{x}_t$ represents the predicted value at time t.

[1] There are 365 days in a year, although in the economic world people tend to use the more convenient year of 360 days when it comes to calculations.

[2] Speculation consists in delibarately taking risks by taking long or short positions on the market. Another strategy, hedging, consists in using the forward market to cover the risk. For more information about those concepts, see the "Foreign Exchange Handbook" by S. Bell and B. Kettel [5].

Two strategies were evaluated. The first one, **strategy 1**, consists in buying/selling the whole amount at hand whenever an increase/decrease in the exchange rate is predicted; the second, **strategy 2**, consists in buying/selling only a fraction of the amount at hand, here 10%, thus reducing the risk of loosing too much at once (but also reducing the potential gain when big differences occur). Both strategies use U.S. Dollar as the currency for the initial amount. The amount at the end of the testing period is converted to U.S. Dollar too.

Taking decisions based on differences between predicted and real values [18] was envisaged, but it happened that the resulting predictions were often well below the real values (*see Section 4.3*), so this idea was abandonned. Predicting the differences between two exchange rates themselves instead of the actual rates would be another idea, but it would require some tuning of the perceptron so that the output includes negative values, by changing the activation function (*see Section 2.3*).

Another good idea would be to add a measure of confidence in the prediction, so as to decide how much to buy/sell at each step. There are of course all sorts of possible strategies that can be implemented using the developped platform (*see Appendix A*).

4.3 Potential Problem with Results

During all the tests, many predictions were much lower than the real value (*see Figure 5.1 (a)*). Although we tried to find a satisfying explication, we couldn't really spot what went wrong. Here are possibilities that were investigated:

- There might very well be a bug in the implemented perceptron that causes this behaviour, but in this case why do some diagram fit almost perfectly (*see Figure 5.1 (b)*)?

- Some currency pairs, especially USD/CHF and USD/EUR were very low during a long period, and suddenly went up to show a lot of instability over shorter periods (*see Figure 4.2*). Training the perceptron with very low values to test it on a market with a high instable peak might produce the results shown in Figure 5.2, although it's not clear why. The countermeasure in this case is to shorten the time period. That's what was done in Section 5.1.

- There might not be enough layers or PEs in the hidden layer, problem that arises only for some currencies while other are more easily captured. How-

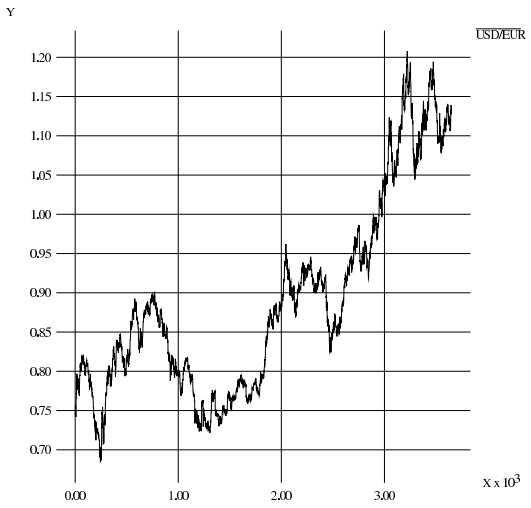


Figure 4.2: Exchange rates for USD/EUR over the last ten years

ever, Yao *et al.* [18] used a three-layer perceptron too, and they didn't mention this problem.

The results proved interesting despite this problem, so we drawed conclusions anyway, while keeping the issue in mind. Chances are big that if there is a bug in the perceptron, results will be better than those showed in this study, not worse.

4.4 Comparison with Related Work

The purpose of this section is to compare our methodology with the one chosen by Yao *et al.* [18]. The results obtained will then be compared further in this document (*see Section 5.6*).

We used a similar network model as Yao *et al.* [18]. We also applied the NMSE measure to our results. However, their study went much deeper in certain areas than ours. For example, they compared their results to those obtained using statistical methods like ARIMA, and they used other measures of performance in addition to NMSE. This is well beyond our goal.

Yao *et al.* produced results for the five currency pairs AUD/USD, CHF/USD, DEM/USD, GBP/USD and JPY/USD. The time periods chosen by the authors are in the same order than ours, but they range from 1984 to 1995.

The input models they used are similar to the **weekly averages** and **progressive averages** used in this study (*see Section 3.5*).

The first strategy (**strategy 1**, *see Section 4.2 and Chapter 5*) is similar to the first strategy they proposed. They also proposed a second one using the difference between the predicted and the real value, but as mentioned in Section 4.2, we abandonned this idea.

Chapter 5

Results and Discussion

This chapter presents the results of this project. Each section below studies the influence of a parameter of the data on the forecasting results and on the profit.

All the experiments were conducted with the following parameters (*see Section 2.4*):

maximum iterations	1000
maximum non-updating iterations	100
momentum (α)	0.9
learning rate (η)	0.05
epochs (e)	1
hidden layer type	half

The five experimental currency pairs are USD/CHF, USD/EUR, USD/GBP, USD/JPY and USD/ZAR. The two pairs USD/NOK and USD/DKK showed results very similar to those of USD/EUR and were therefore not included. USD/CAD and USD/NZD were excluded for similar reasons, they wouldn't have brought much more to the argumentation. The exchange rate of HKD beeing fixed against the exchange rate of USD, the pair USD/HKD was ignored as well, it's plot beeing almost a horizontal line (*see Section 3.1.2*).

The profits were calculated for the two strategies mentioned in Section 4.2, called **strategy 1** and **strategy 2**. Given profits are those calculated during the generation of the graph. Therefore, they may not reflect the profit one can expect on average, nor the best profit one can achieve.

5.1 Influence of Time Period

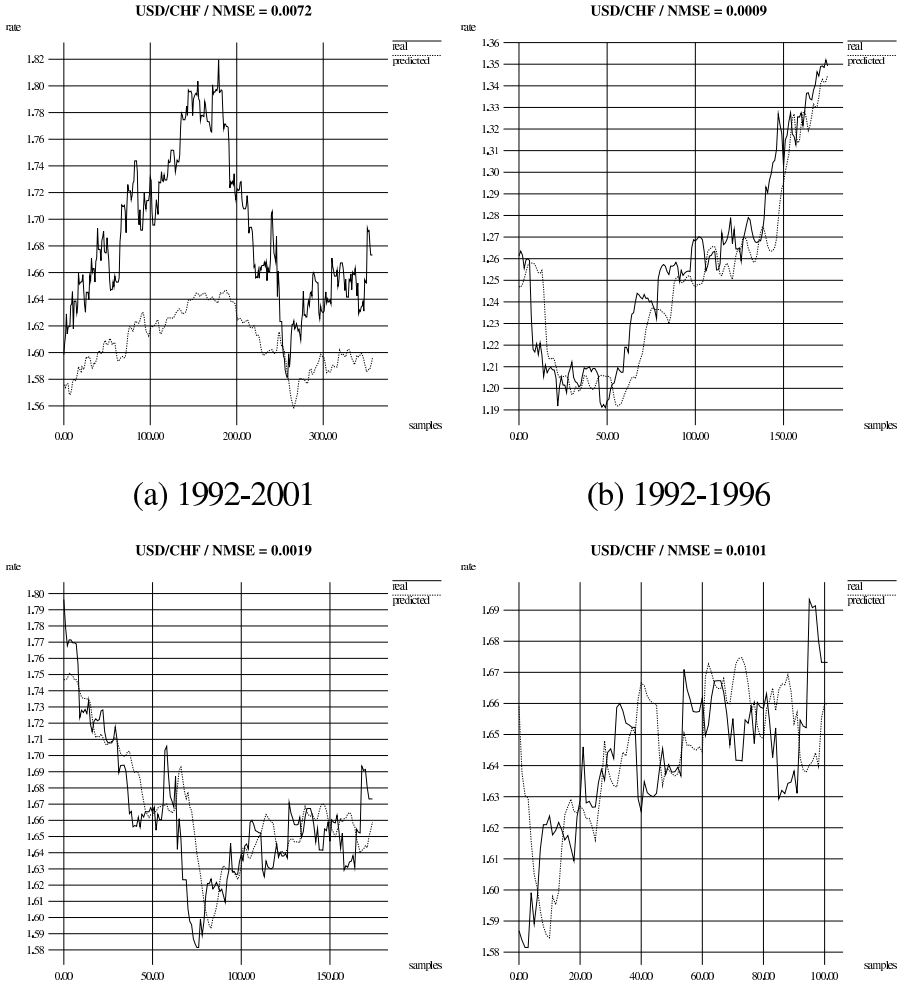
The measurements of the forecasting results for the four time periods mentioned in Section 3.3 are shown in Figures 5.1 to 5.5 for the five experimental currency pairs. The influence of the time period on NMSE and profits is shown in Table 5.1. The segmentation chosen is `70-20-10` (*see Section 3.4*) and the input model is **progressive averages** (*see Section 3.5.3*).

From the diagrams, we can see that the best time period in terms of NMSE is 1992 to 1996, that is with the chosen segmentation a training of 3.5 years, a cross-validation of 1 year and a testing period of 6 months, ranging from 01.07.1996 to 31.12.1996. The only exception here is USD/ZAR, for which it is actually the worse period. However, we cannot conclude that a time period of five years is better than one of ten years, even if it first seems to be the case for USD/CHF and USD/EUR. Indeed, the results for 1997 to 2001 are worse than those for 1992 to 2001 for the three other currencies.

If we look at the profits in Table 5.1, an important clue appears: the profit seems to be generally better for shorter time periods. If we exclude USD/GBP, that gives small profits anyway, the best period seems to be 1999 to 2001. Even for USD/GBP, the loss is not too bad compared with the profits of the four other pairs. But such a small period is probably not a good idea, which is confirmed by the results for USD/ZAR.

Finally, if we look at the profits one can make on USD/ZAR, they really look impressive. When we look at the historical exchange rates for this pair on Figure 5.5, we see the pair has been going up a lot since 1992, which is of course an ideal situation for our trading strategies.

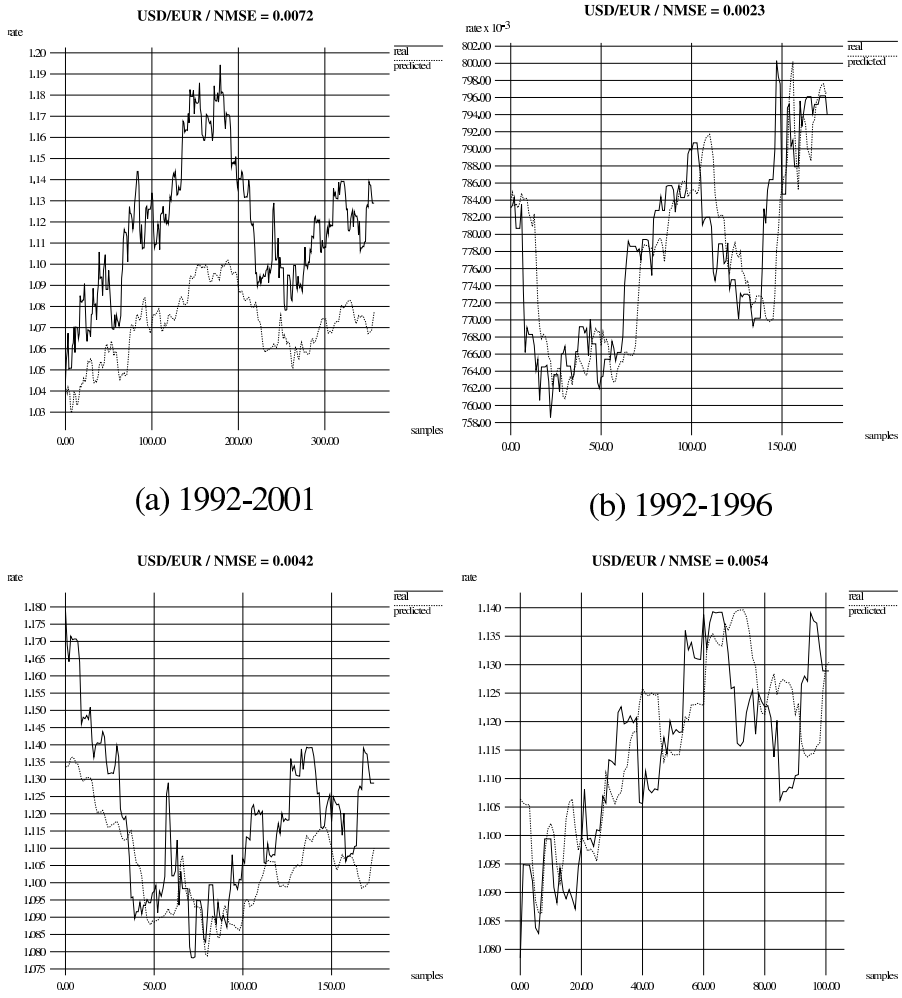
The plots show a predicted curve that is almost always late compared to the real one. This is a major problem for all prediction models, and it probably explains the low profit and the losses achieved with the simple strategies involved.



(a) 1992-2001 (b) 1992-1996

(c) 1997-2001 (d) 1999-2001

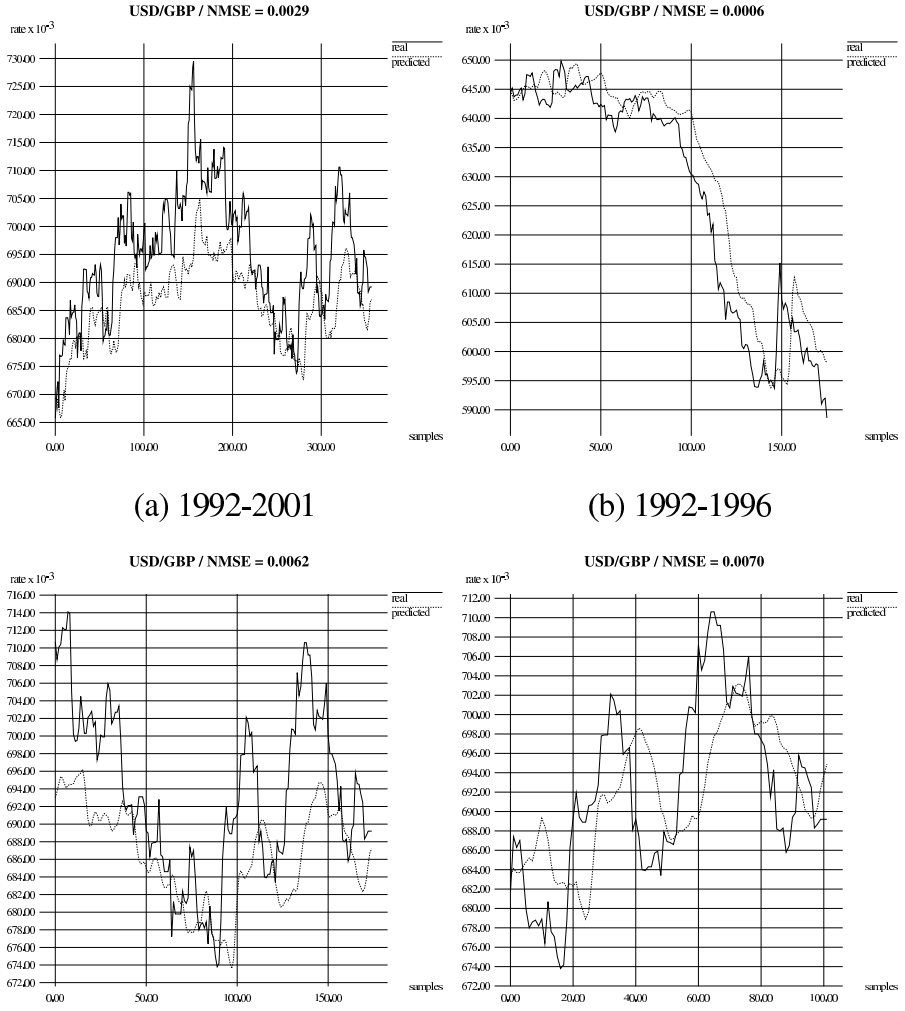
Figure 5.1: Influence of time period on USD/CHF



(a) 1992-2001 (b) 1992-1996

(c) 1997-2001 (d) 1999-2001

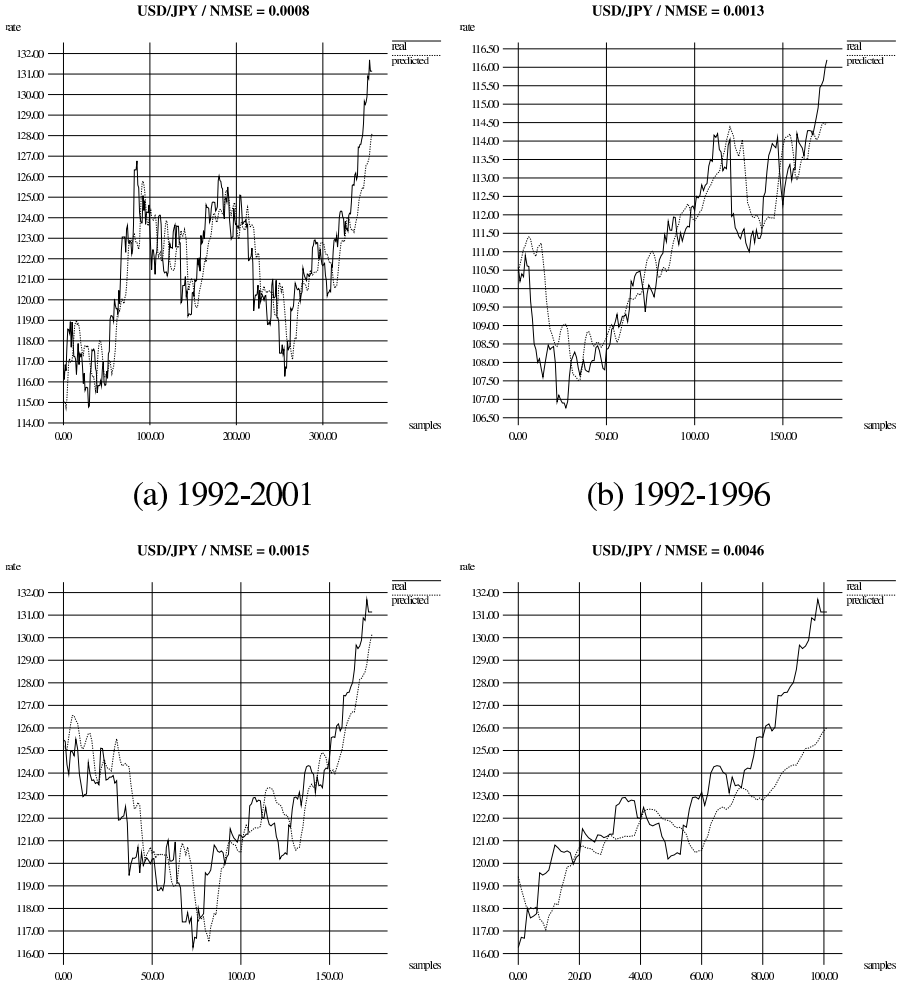
Figure 5.2: Influence of time period on USD/EUR



(a) 1992-2001 (b) 1992-1996

(c) 1997-2001 (d) 1999-2001

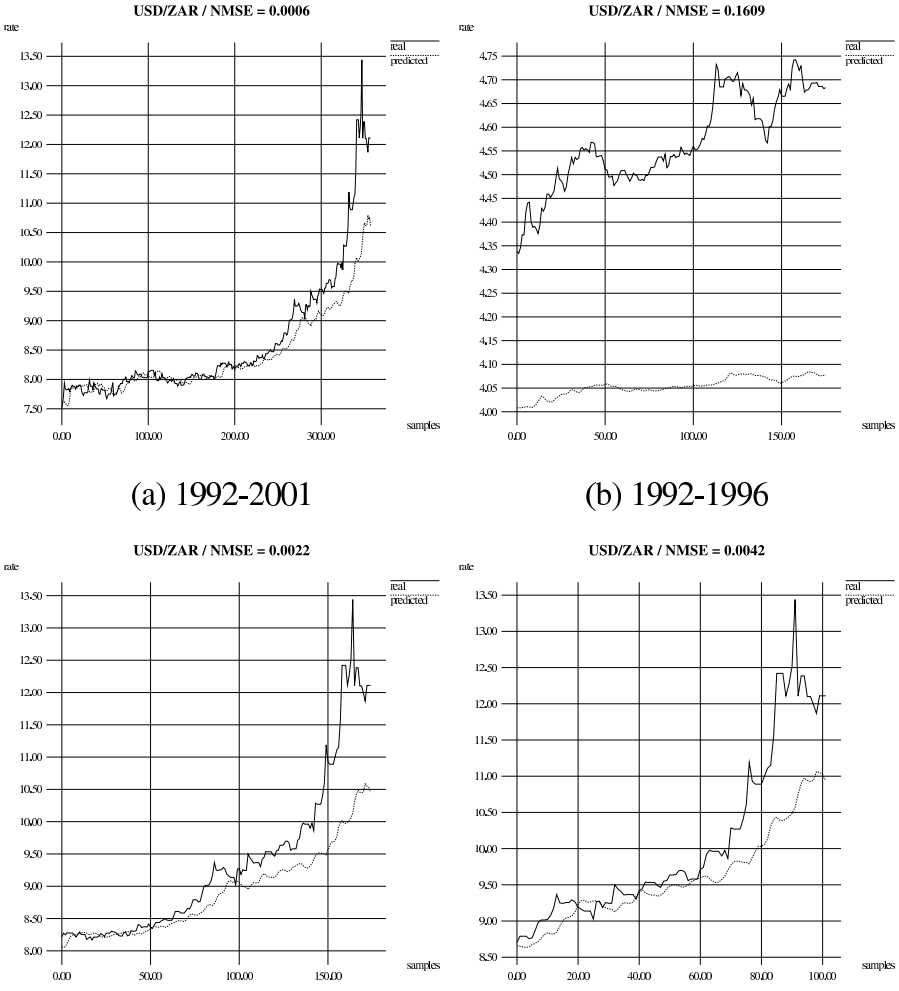
Figure 5.3: Influence of time period on USD/GBP



(a) 1992-2001 (b) 1992-1996

(c) 1997-2001 (d) 1999-2001

Figure 5.4: Influence of time period on USD/JPY



(a) 1992-2001 (b) 1992-1996

(c) 1997-2001 (d) 1999-2001

Figure 5.5: Influence of time period on USD/ZAR

currency pair	period	NMSE	strategy 1 [%]	strategy 2 [%]
USD/CHF	1992-2001	0.0072	1.64	-1.04
USD/CHF	1992-1996	0.0009	14.42	9.66
USD/CHF	1997-2001	0.0019	0.99	0.38
USD/CHF	1999-2001	0.0101	4.09	0.93
USD/EUR	1992-2001	0.0072	3.56	0.30
USD/EUR	1992-1996	0.0023	2.80	2.09
USD/EUR	1997-2001	0.0042	-6.67	0.77
USD/EUR	1999-2001	0.0054	6.86	2.07
USD/GBP	1992-2001	0.0029	-0.30	-2.13
USD/GBP	1992-1996	0.0006	-17.08	-2.68
USD/GBP	1997-2001	0.0062	-4.81	-2.41
USD/GBP	1999-2001	0.0070	-2.82	-1.67
USD/JPY	1992-2001	0.0008	6.99	4.80
USD/JPY	1992-1996	0.0013	12.01	7.52
USD/JPY	1997-2001	0.0015	7.86	6.70
USD/JPY	1999-2001	0.0046	32.28	18.66
USD/ZAR	1992-2001	0.0006	44.81	45.73
USD/ZAR	1992-1996	0.0599	15.19	5.67
USD/ZAR	1997-2001	0.0022	80.55	89.04
USD/ZAR	1999-2001	0.0042	151.49	90.07

Table 5.1: Influence of time period on NMSE and profits

5.2 Influence of Segmentation

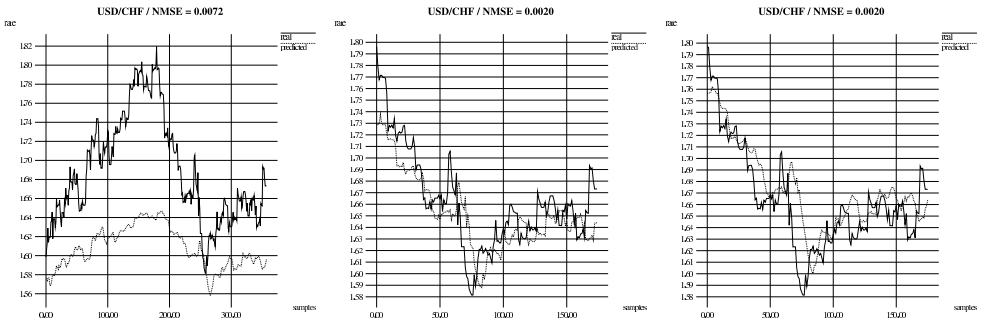
In Section 5.1, we couldn't decide which time period was best for capturing the behaviour of the market, given a certain segmentation, even if five years seemed to be a generally good idea. But what if we change the segmentation to try to give the network more data for training?

The measurements of the forecasting results for the three segmentations mentioned in Section 3.4 are shown in Figures 5.6 to 5.10 for the five experimental currency pairs. The influence of the time period on NMSE and profits is shown in Table 5.2. The time period is 01.01.1992 to 31.12.2001 (*see Section 3.3*) and the input model is **progressive averages** (*see Section 3.5.3*).

From the diagrams, we can see a propensity of the NMSE to decrease if we restrict the testing period from 10% (1 year) to 5% (6 months). This is again not true for USD/ZAR, as showed in Figure 5.10, and this is not true either for USD/JPY (*see Figure 5.9*).

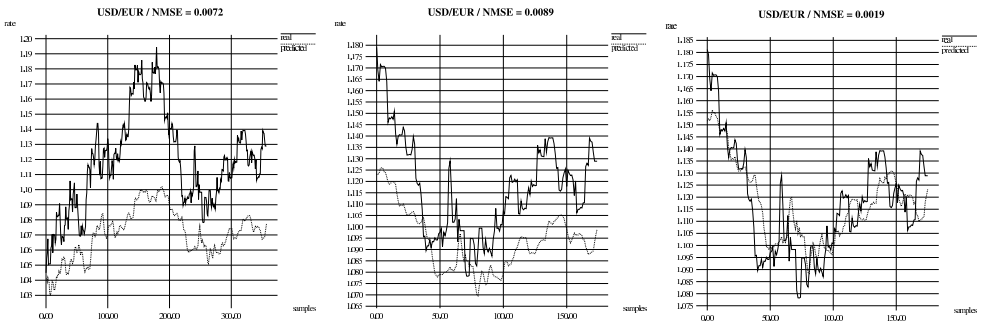
But the profits for the two last currencies increase for a testing period of 5% (6 months). However, the profits for the three currency pairs USD/CHF, USD/EUR and USD/GBP decrease while their NMSE decreases. This illustrates dramatically the issue mentioned in Section 4.1.2.

It is very difficult to conclude anything here, except that it seems to be better not to try to predict periods longer than 6 months. The network could be trained again after this period. Another thing this section illustrates is a certain correlation of the time period and the segmentation on the results for a particular currency. If we look at European currencies, CHF and EUR, we see that a shorter time period and a shorter testing phase are better in terms of forecasting results. It is the opposite for JPY, where a longer time period and a longer testing are better. Like the whole market, currencies have their own behaviour.



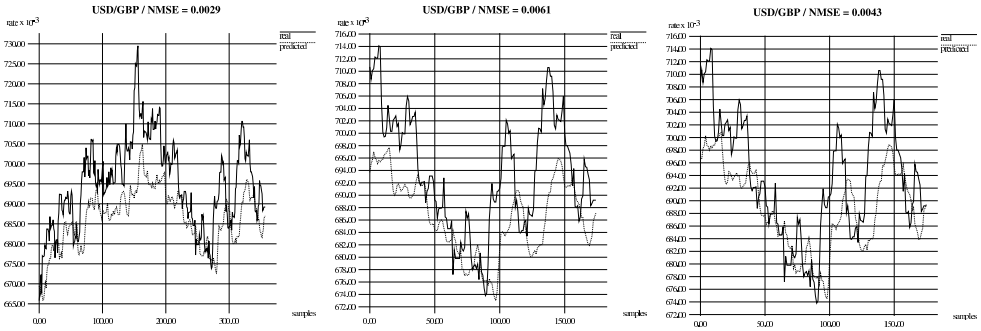
(a) 70-20-10 (b) 80-15-05 (c) 90-05-05

Figure 5.6: Influence of segmentation on USD/CHF



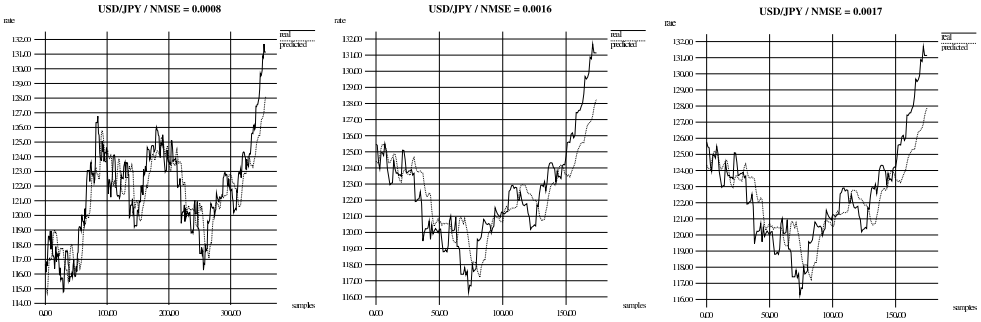
(a) 70-20-10 (b) 80-15-05 90-05-05

Figure 5.7: Influence of segmentation on USD/EUR



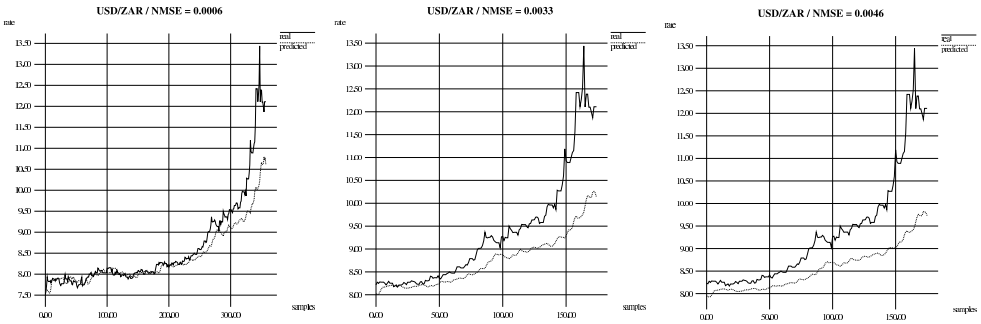
(a) 70-20-10 (b) 80-15-05 90-05-05

Figure 5.8: Influence of segmentation on USD/GBP



(a) 70-20-10 (b) 80-15-05 90-05-05

Figure 5.9: Influence of segmentation on USD/JPY



(a) 70-20-10 (b) 80-15-05 90-05-05

Figure 5.10: Influence of segmentation on USD/ZAR

currency pair	segmentation	NMSE	strategy 1 [%]	strategy 2 [%]
USD/CHF	70-20-10	0.0072	1.64	-1.04
USD/CHF	80-15-05	0.0020	3.09	-1.38
USD/CHF	90-05-05	0.0020	-13.30	-1.23
USD/EUR	70-20-10	0.0072	3.56	0.30
USD/EUR	80-15-05	0.0089	-6.66	0.77
USD/EUR	90-05-05	0.0019	-8.66	-0.47
USD/GBP	70-20-10	0.0029	-0.30	-2.13
USD/GBP	80-15-05	0.0061	-4.81	-3.02
USD/GBP	90-05-05	0.0043	-5.43	-2.17
USD/JPY	70-20-10	0.0008	6.99	4.80
USD/JPY	80-15-05	0.0016	7.86	6.70
USD/JPY	90-05-05	0.0017	8.82	7.50
USD/ZAR	70-20-10	0.0006	44.81	45.73
USD/ZAR	80-15-05	0.0033	87.45	91.19
USD/ZAR	90-05-05	0.0046	103.16	93.73

Table 5.2: Influence of segmentation on NMSE and profits

5.3 Influence of Input Number

Section 5.2 showed that currencies have their own behaviour. The results so far also illustrates a linkage of certain currencies with other[1]. For example, if we look at how USD/CHF and USD/EUR behave, we see a similitude. In this section, we try to increase the number of currencies used as inputs to teach the model this dependency and smooth down the individual behaviours of currencies.

The measurements of the forecasting results for three different numbers of inputs are shown in Figures 5.11 to 5.15 for the five experimental currency pairs. The three situations are the following:

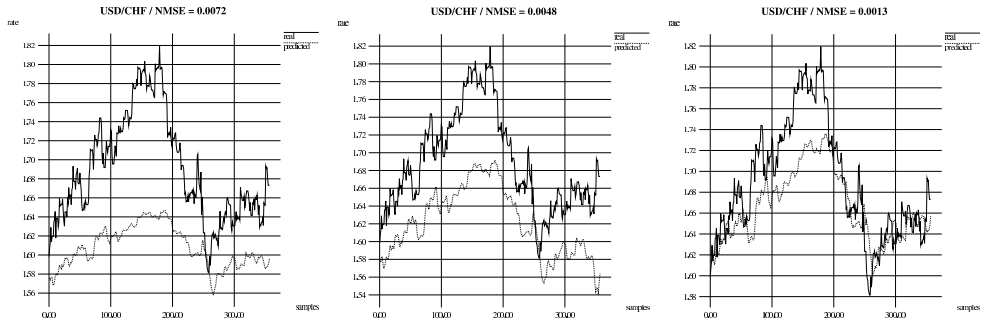
- **1 currency** refers to a situation where the input pair is the same as the output pair, for example, for USD/CHF, the input is only USD/CHF;
- **5 currencies** refers to a situation where the input pairs are the five experimental pairs USD/CHF, USD/EUR, USD/GBP, USD/JPY and USD/ZAR;
- **10 currencies** refers to a situation where all the available pairs are used as input (*see Section 3.1.2*).

The influence of the number of inputs on NMSE and profits is shown in Table 5.3. The time period is 01.01.1992 to 31.12.2001 (*see Section 3.3*), the segmentation is `70-20-10` (*see Section 3.4*) and the input model is **progressive averages** (*see Section 3.5.3*).

From the diagrams, we see that increasing the number of currency pairs in input provides better results in terms of forecasting performances for USD/CHF and USD/EUR. Again, USD/JPY and USD/ZAR have the opposite behaviour, and the prediction performances for USD/GBP are also decreasing when increasing the number of inputs.

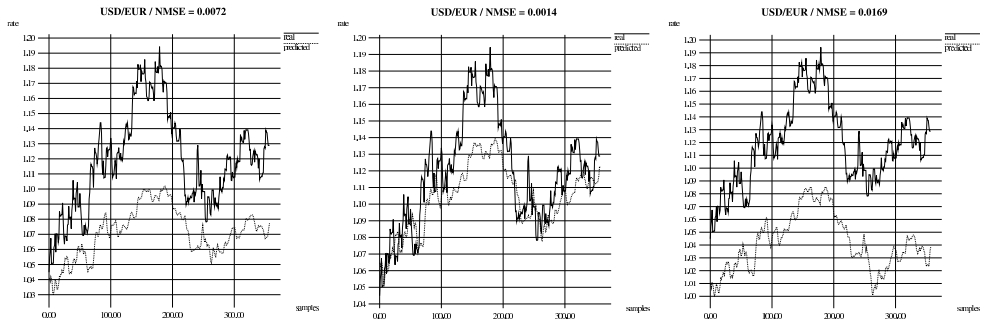
However, the profit here seems to decrease when the number of inputs increases, this for all five experimental currency pairs. We can safely conclude that increasing the number of currencies without taking care of their linkage is a bad idea. It would be very interesting to try to determine which currencies are linked with others from this model, see if it fits with the actual linkages known in Economy, and maybe see if increasing the number of correlated currencies in input could lead to better results in terms of profit.

[1]This linkage is well-known in Economy, and was particularly spectacular when there still existed many different currencies in Europe, before the Euro era.



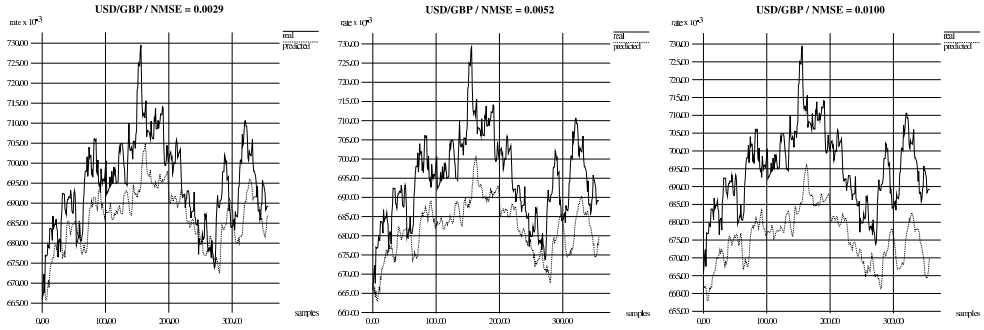
(a) 1 currency (b) 5 currencies (c) 10 currencies

Figure 5.11: Influence of input number on USD/CHF



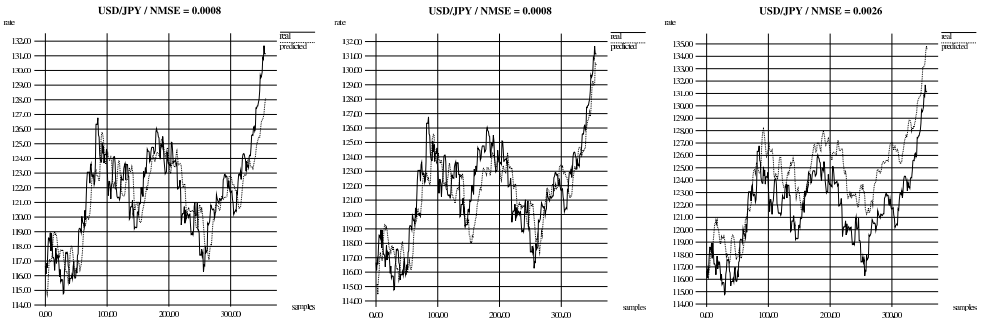
(a) 1 currency (b) 5 currencies (c) 10 currencies

Figure 5.12: Influence of input number on USD/EUR



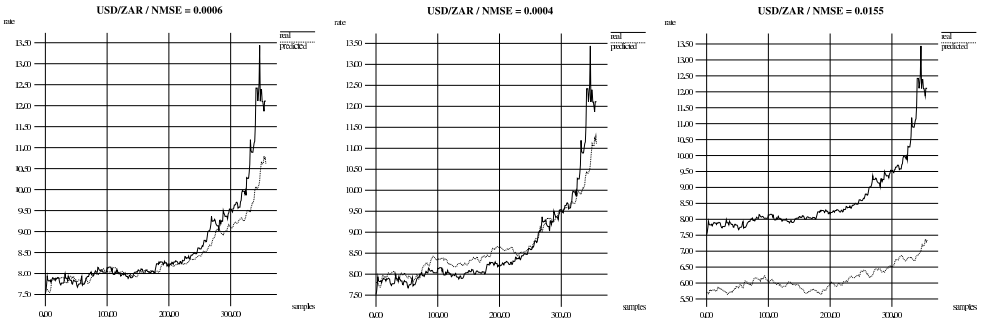
(a) 1 currency (b) 5 currencies (c) 10 currencies

Figure 5.13: Influence of input number on USD/GBP



(a) 1 currency (b) 5 currencies (c) 10 currencies

Figure 5.14: Influence of input number on USD/JPY



(a) 1 currency (b) 5 currencies (c) 10 currencies

Figure 5.15: Influence of input number on USD/ZAR

currency pair	inputs	NMSE	strategy 1 [%]	strategy 2 [%]
USD/CHF	1	0.0072	1.64	-1.04
USD/CHF	5	0.0048	1.85	1.55
USD/CHF	10	0.0013	1.59	-2.85
USD/EUR	1	0.0072	3.56	0.30
USD/EUR	5	0.0014	3.56	0.30
USD/EUR	10	0.0169	1.58	-0.40
USD/GBP	1	0.0029	-0.30	-2.13
USD/GBP	5	0.0052	-0.30	-2.05
USD/GBP	10	0.0100	-0.55	-2.12
USD/JPY	1	0.0008	6.99	4.80
USD/JPY	5	0.0008	5.45	3.45
USD/JPY	10	0.0026	7.02	5.77
USD/ZAR	1	0.0006	44.81	45.73
USD/ZAR	5	0.0004	46.09	45.01
USD/ZAR	10	0.0155	34.34	33.06

Table 5.3: Influence of input number on NMSE and profits

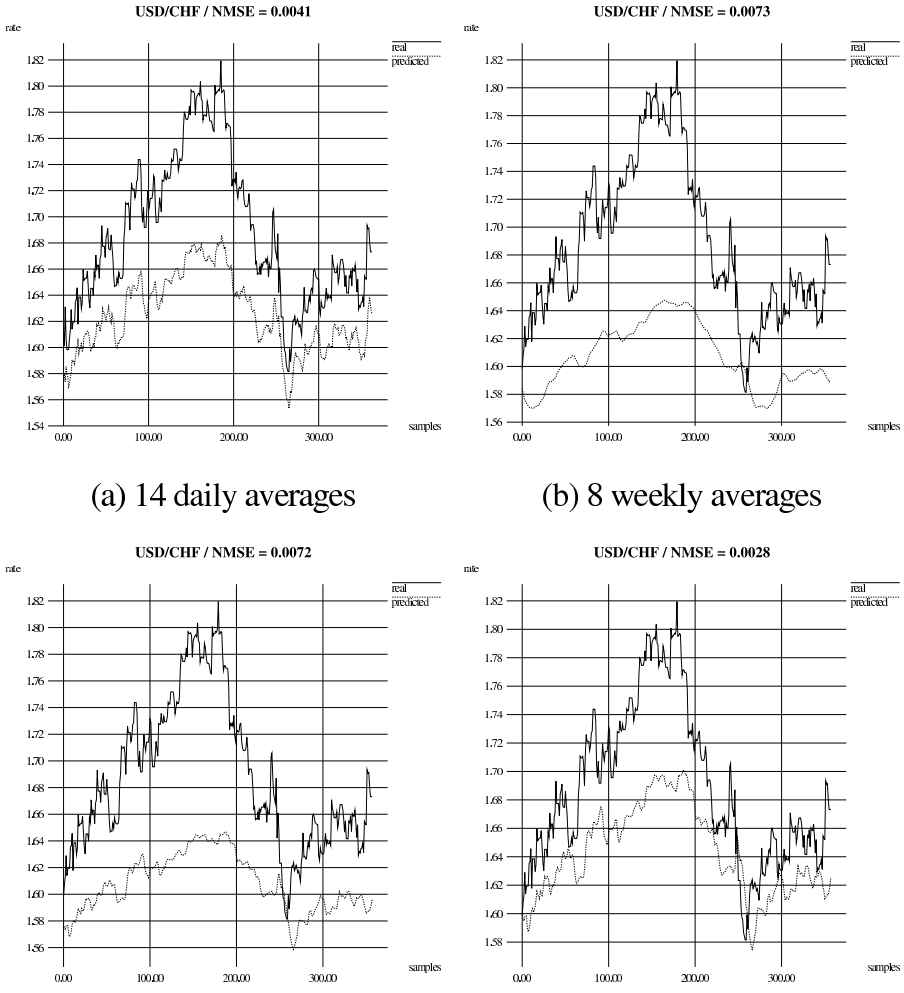
5.4 Influence of Input Model

The measurements of the forecasting results for the four input models mentioned in Section 3.5 are shown in Figures 5.16 to 5.20 for the five experimental currency pairs. The influence of the input model on NMSE and profits is shown in Table 5.4. The time period is 01.01.1992 to 31.12.2001 (*see Section 3.3*) and the segmentation is `70-20-10` (*see Section 3.4*).

From Table 5.4, we can see that progressive averages produce better results in terms of NMSE than time-delayed averages. This is what was expected when we designed the four input models to capture the various trends of the market (*see Section 3.5*). It seems that the **long progressive averages** model produces slightly better results than the shorter model **progressive averages**, except for USD/EUR. The plots tend to confirm this.

But again, the profit doesn't follow at all the forecasting results. The best profits in Table 5.4 are achieved with the purely time-delayed **14 daily averages** input model, the worse in term of NMSE. This could very well be that the strategies aren't adapted to the input model. Indeed, both strategies work on a weekly basis, whereas the intermediate and major trends range in the order of several months. Trying long-term strategies may improve this situation.

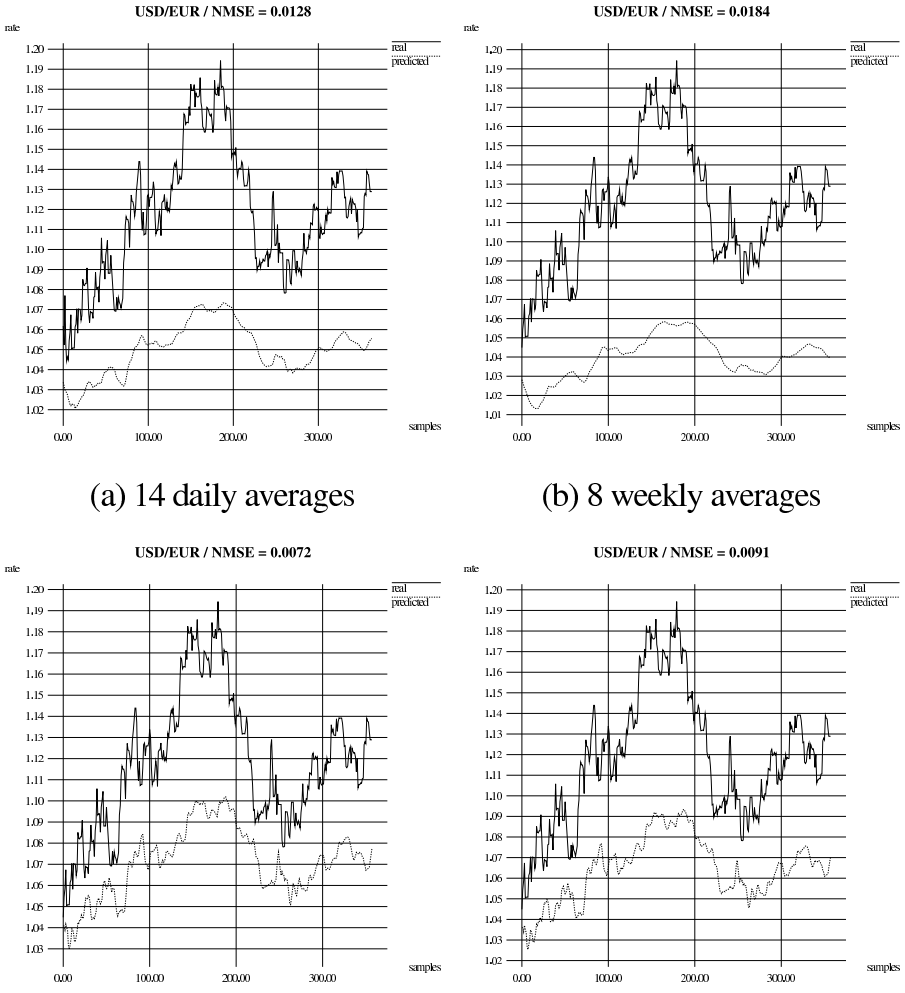
Also, saying that the **14 daily averages** model is the worse isn't actually exact. If we look at the plots, this model outperforms at least the **8 weekly averages** and the **progressive averages** models from a visual point of view, if not from a NSME point of view. Exceptions are USD/ZAR and maybe USD/EUR where it is difficult to deduce something from the plots. Maybe capturing the minor trend only is enough to achieve profit with such a system. However, the **14 daily averages** model implies trading every day in our implementation, and there could be a lot of unnecessary trading costs involved that weren't taken into account (*see Section 3.1*).



(a) 14 daily averages (b) 8 weekly averages

(c) progressive averages (d) long progressive averages

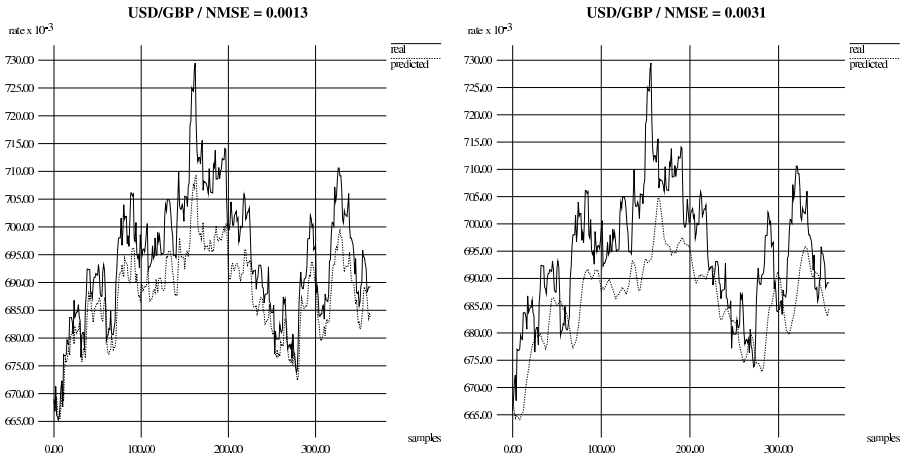
Figure 5.16: Influence of input model on USD/CHF



(a) 14 daily averages (b) 8 weekly averages

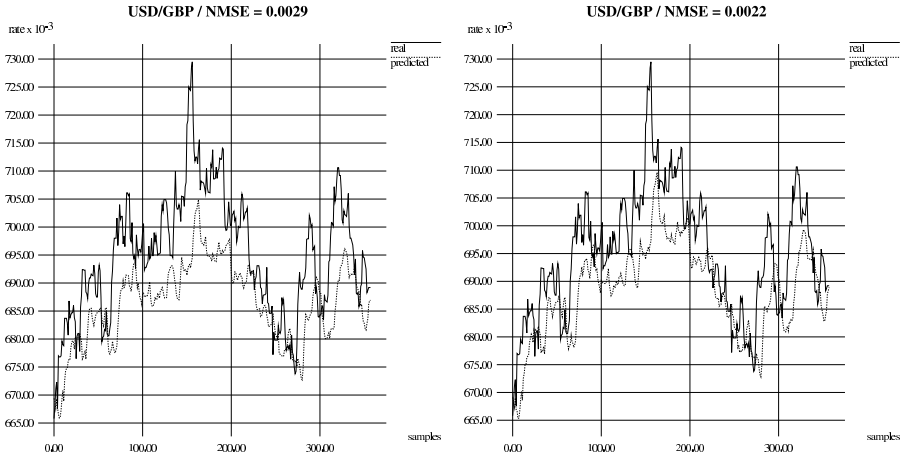
(c) progressive averages (d) long progressive averages

Figure 5.17: Influence of input model on USD/EUR



(a) 14 daily averages

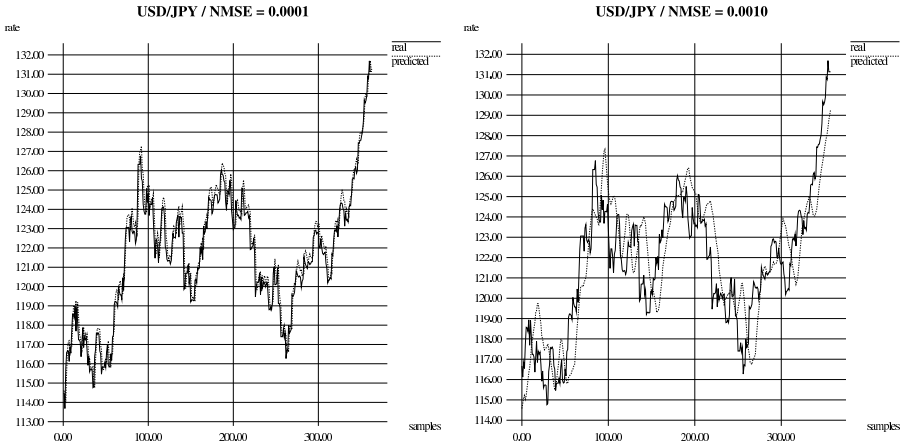
(b) 8 weekly averages



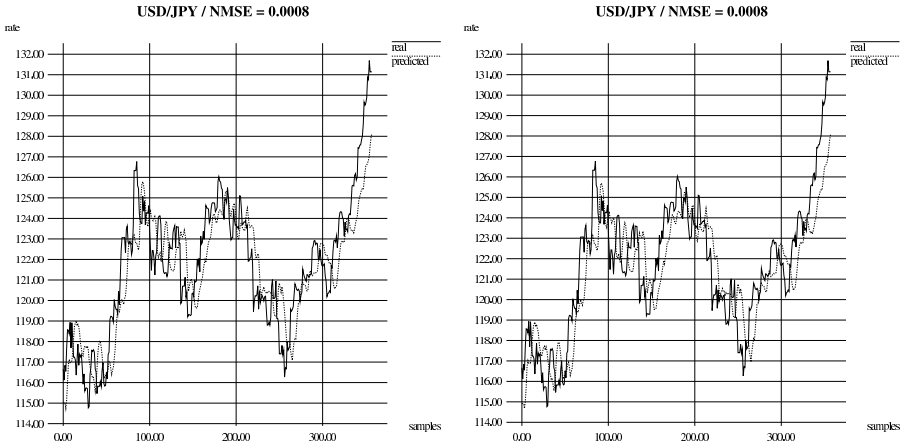
(c) progressive averages

(d) long progressive averages

Figure 5.18: Influence of input model on USD/GBP

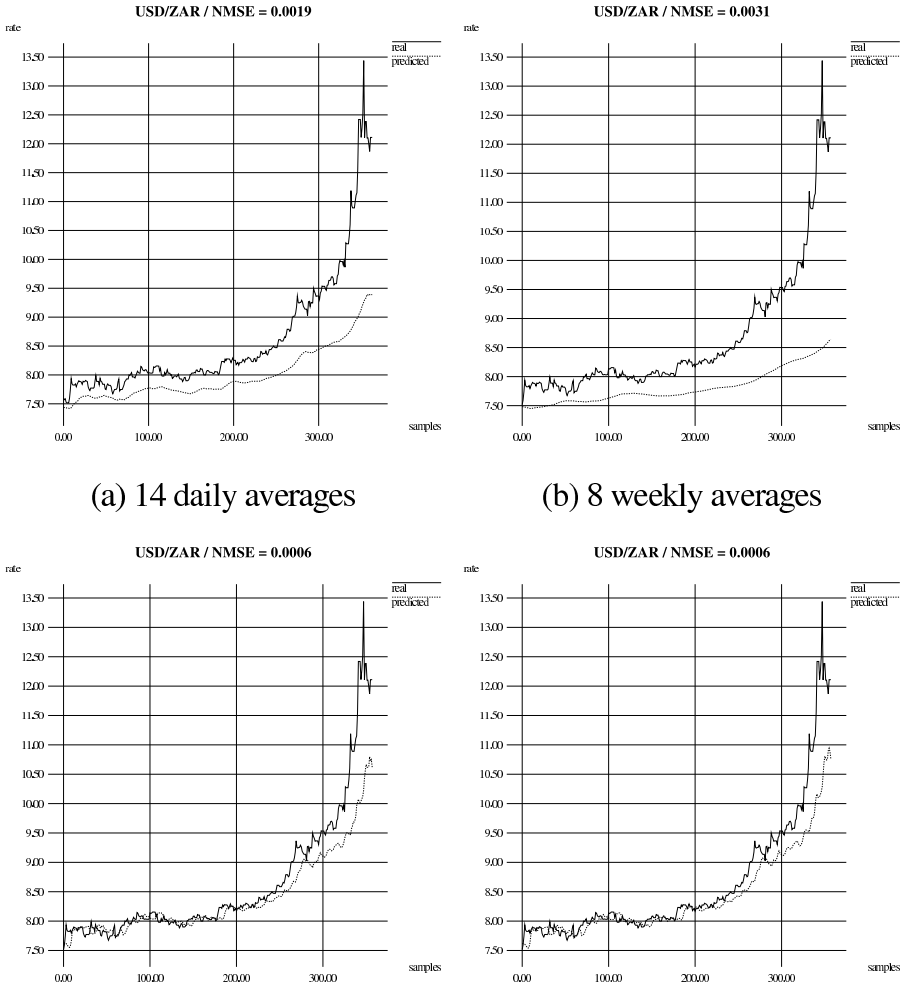


(a) 14 daily averages (b) 8 weekly averages



(c) progressive averages (d) long progressive averages

Figure 5.19: Influence of input model on USD/JPY



(a) 14 daily averages (b) 8 weekly averages

(c) progressive averages (d) long progressive averages

Figure 5.20: Influence of input model on USD/ZAR

currency pair	model	NMSE	strategy 1 [%]	strategy 2 [%]
USD/CHF	14 daily avg	0.0183	26.20	6.18
USD/CHF	8 weekly avg	0.0202	1.11	-0.23
USD/CHF	progressive avg	0.0072	1.64	-1.04
USD/CHF	long progressive avg	0.0028	1.64	-1.04
USD/EUR	14 daily avg	0.0128	26.60	4.28
USD/EUR	8 weekly avg	0.0184	5.34	1.18
USD/EUR	progressive avg	0.0072	3.56	0.30
USD/EUR	long progressive avg	0.0091	3.56	0.30
USD/GBP	14 daily avg	0.0013	23.09	2.13
USD/GBP	8 weekly avg	0.0031	2.43	-1.11
USD/GBP	progressive avg	0.0029	2.36	-2.13
USD/GBP	long progressive avg	0.0022	-0.30	-2.05
USD/JPY	14 daily avg	0.0001	36.84	7.82
USD/JPY	8 weekly avg	0.0010	9.82	2.07
USD/JPY	progressive avg	0.0008	6.99	4.80
USD/JPY	long progressive avg	0.0008	6.99	4.80
USD/ZAR	14 daily avg	0.0019	59.02	50.31
USD/ZAR	8 weekly avg	0.0031	54.46	51.09
USD/ZAR	progressive avg	0.0006	44.81	45.73
USD/ZAR	long progressive avg	0.0006	44.81	45.73

Table 5.4: Influence of input model on NMSE and profits

5.5 Influence of Other Factors

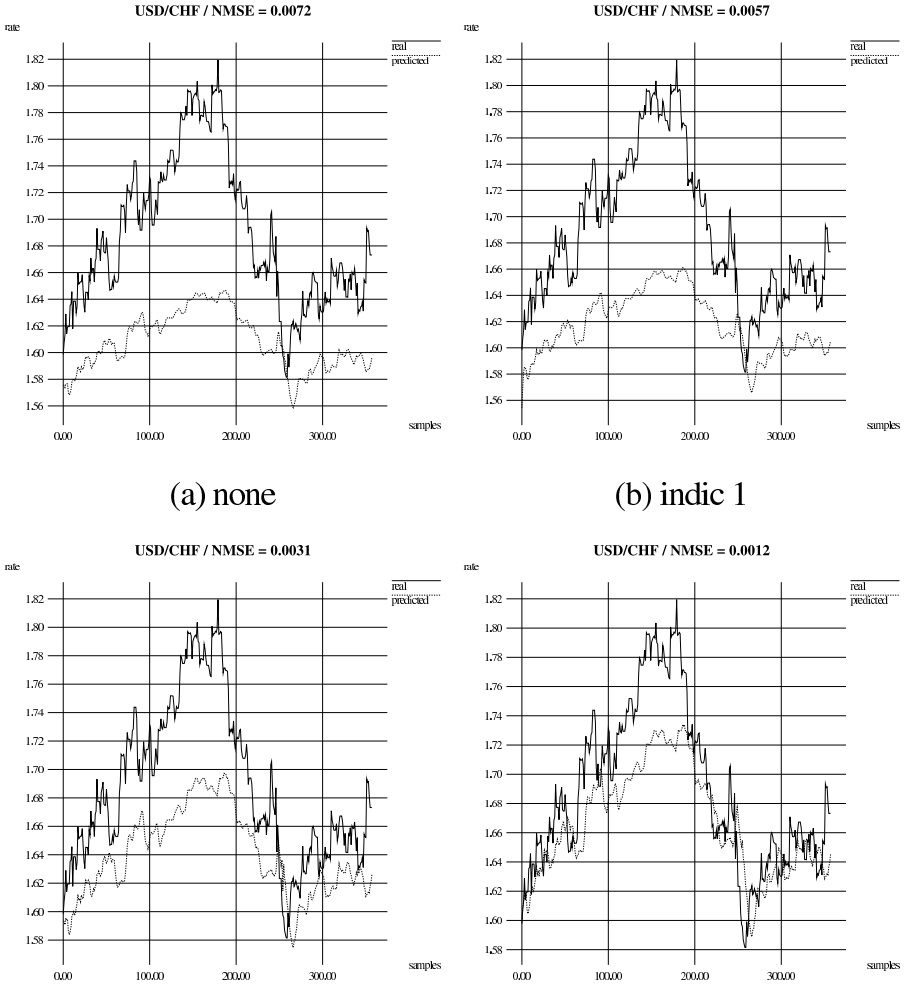
The measurements of the forecasting results for four different situations involving boolean indicators (*see Section 3.2*) are shown in Figures 5.21 to 5.25 for the five experimental currency pairs. The four situations are the following:

- **none** refers to the situation with no boolean indicator;
- **indic 1** refers to the situation where the first boolean indicator, the one saying whether the government of the U.S.A. is Republican or Democratic, is added in input;
- **indic 2** refers to the situation where the second boolean indicator, the one saying whether there is a war going on, is added in input;
- **both** refers to the situation where both indicators are added in input.

The influence of those four situations on NMSE and profits is shown in Table 5.5. The time period is 01.01.1992 to 31.12.2001 (*see section 3.3*), the segmentation is `70-20-10` (*see Section 3.4*) and the input model is **progressive averages** (*see Section 3.5.3*).

If we take a brief look at the plots, adding simple indicators seems a good idea for improving the predicting power of the model. The increase in performance is spectacular on USD/CHF (*see Figure 5.21*). USD/ZAR behaves differently again, and also USD/JPY, but when we look at this last pair's plot (*Figure 5.24*) we see that the results are already so good without any boolean indicator that it would be difficult to do better.

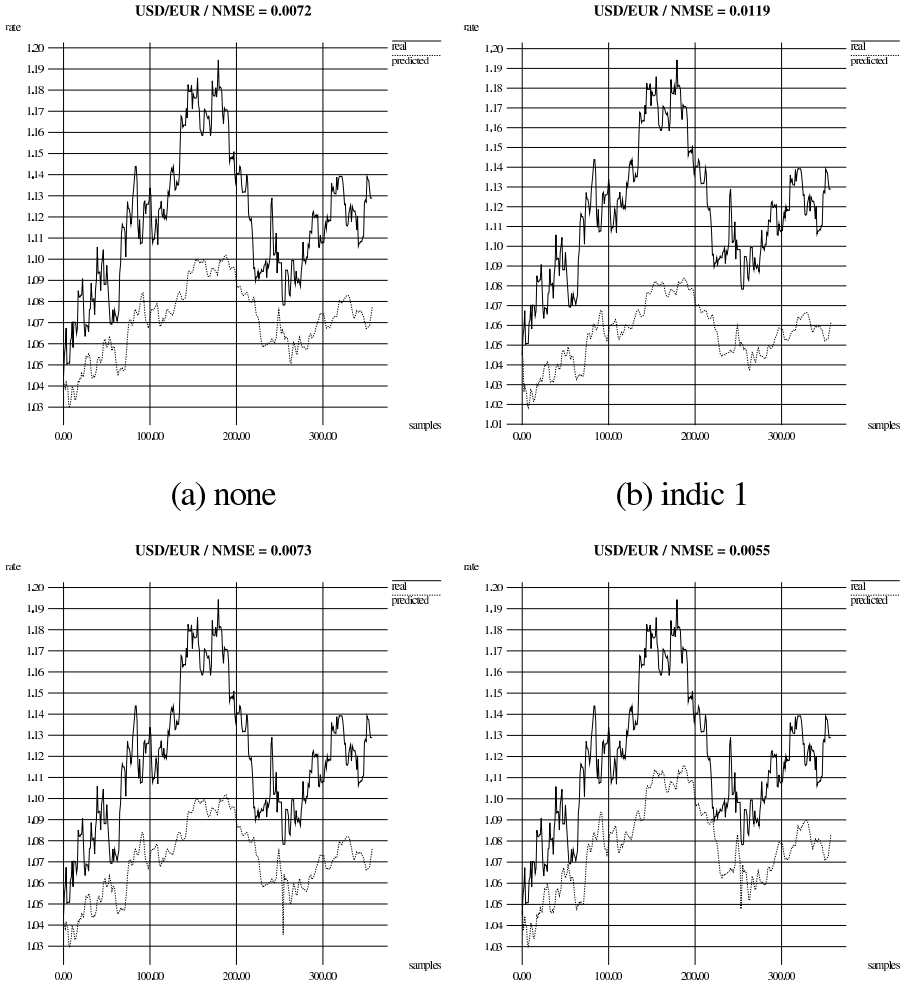
Unfortunately, in terms of profit, adding those boolean indicators doesn't seem to improve the situation. At least it doesn't seem to cause a loss of performance either.



(a) none (b) indic 1

(c) indic 2 (d) both

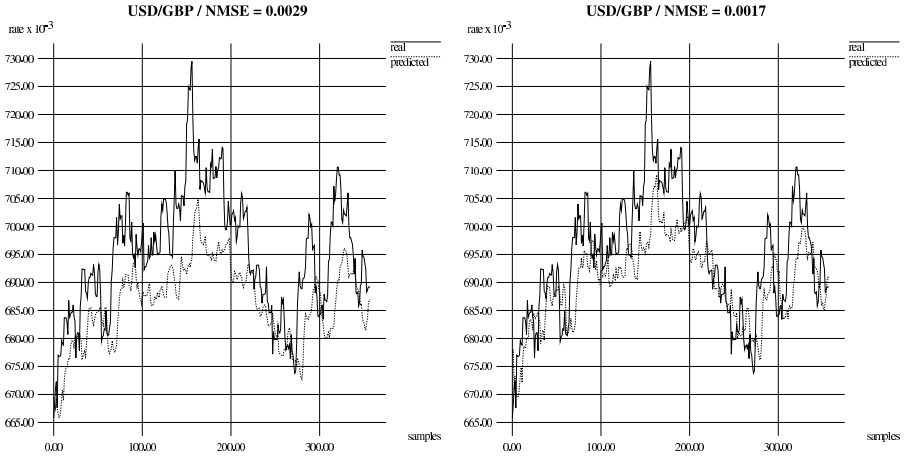
Figure 5.21: Influence of boolean factors on USD/CHF



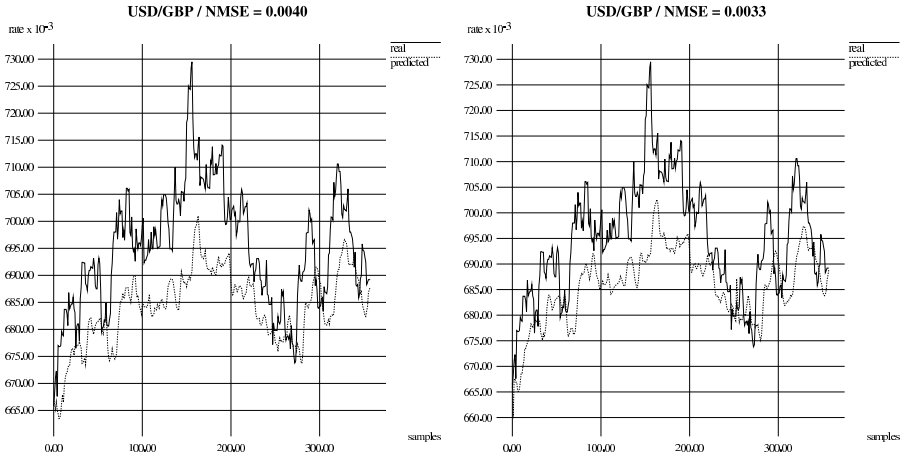
(a) none (b) indic 1

(c) indic 2 (d) both

Figure 5.22: Influence of boolean factors on USD/EUR

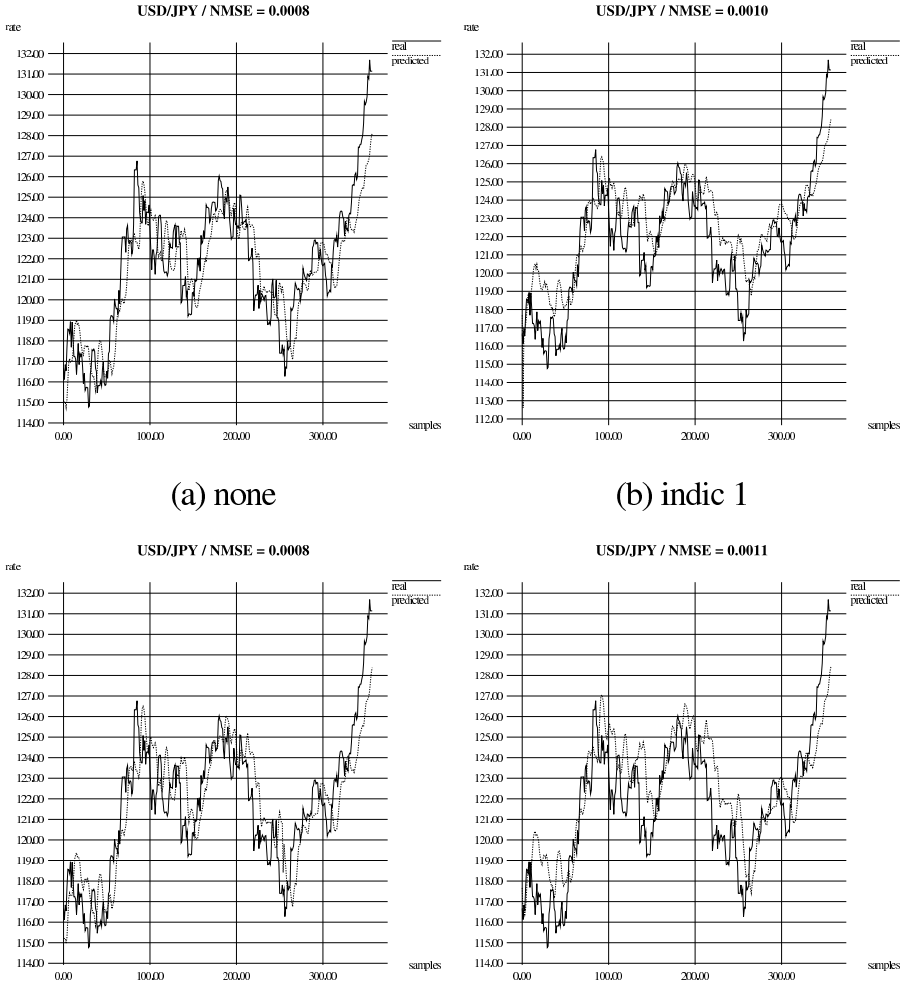


(a) none (b) indic 1



(c) indic 2 (d) both

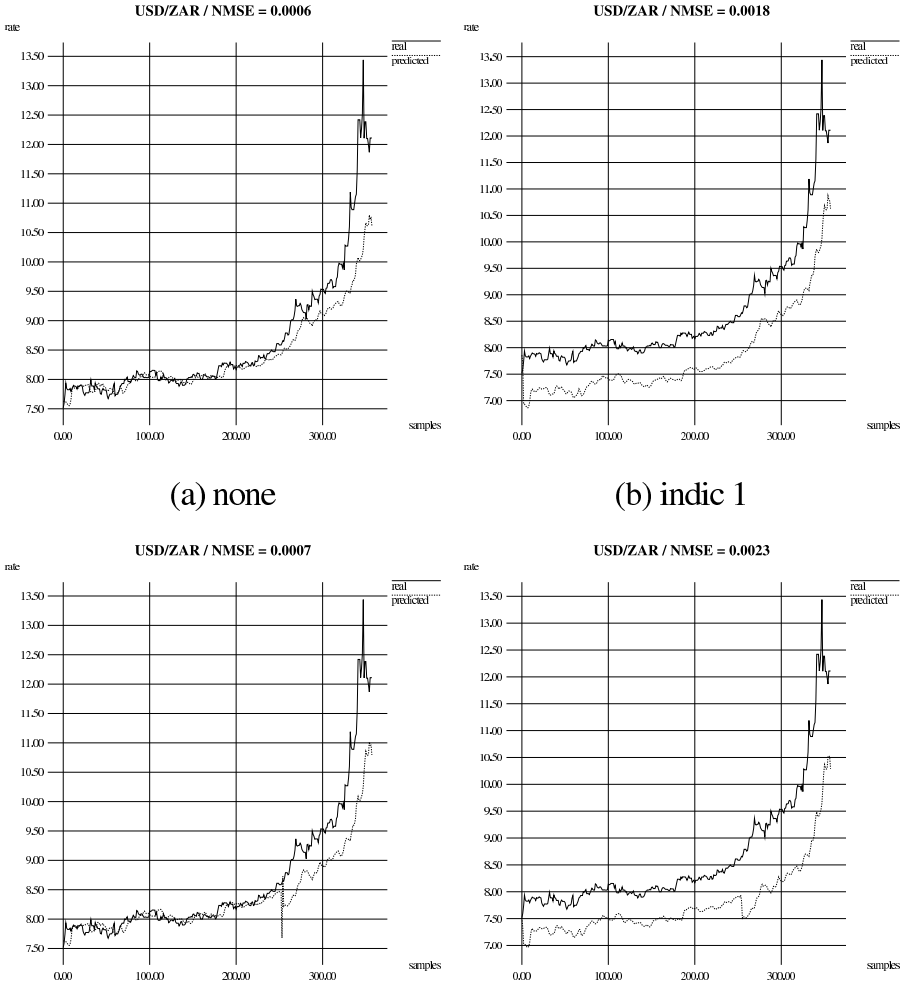
Figure 5.23: Influence of boolean factors on USD/GBP



(a) none (b) indic 1

(c) indic 2 (d) both

Figure 5.24: Influence of boolean factors on USD/JPY



(a) none (b) indic 1

(c) indic 2 (d) both

Figure 5.25: Influence of boolean factors on USD/ZAR

currency pair	indicator	NMSE	strategy 1 [%]	strategy 2 [%]
USD/CHF	none	0.0072	1.64	-1.04
USD/CHF	indic 1	0.0057	1.64	-1.04
USD/CHF	indic 2	0.0031	1.64	-1.04
USD/CHF	both	0.0012	1.69	-0.57
USD/EUR	none	0.0072	3.56	0.30
USD/EUR	indic 1	0.0119	3.56	0.30
USD/EUR	indic 2	0.0073	3.56	0.30
USD/EUR	both	0.0055	3.56	0.30
USD/GBP	none	0.0029	-0.30	-2.13
USD/GBP	indic 1	0.0017	-0.30	-2.22
USD/GBP	indic 2	0.0040	-0.30	-1.86
USD/GBP	both	0.0033	-0.30	-1.77
USD/JPY	none	0.0008	6.99	4.80
USD/JPY	indic 1	0.0010	6.99	4.80
USD/JPY	indic 2	0.0008	6.99	4.80
USD/JPY	both	0.0011	6.99	4.80
USD/ZAR	none	0.0006	44.81	45.73
USD/ZAR	indic 1	0.0018	44.81	45.73
USD/ZAR	indic 2	0.0007	41.54	44.72
USD/ZAR	both	0.0023	41.54	44.72

Table 5.5: Influence of boolean indicators on NMSE and profits

5.6 Comparison with Related Work

In Section 4.4, we compared our methodology with the one chosen by Yao *et al.* [18]. Now we try to compare the results.

Yao *et al.* achieved good forecasting results for their currency pairs, the best results beeing for a testing period of six months. They mentioned problems about JPY/USD, whereas we got good results for USD/JPY. Maybe the behaviour of Yen has changed since.

Their more in-depth study about USD/CHF [16] showed similar NMSE's for similar situations (structure `6-3-1`, same time period length, same segmentation), but they achieved far better results in term of profits (up to more than 60% for some situations, when the best result for USD/CHF we got was 26%, *see Figure 5.4*). Their profits for JPY/USD are of the same order than ours for USD/JPY.

The conclusion of the authors of the case study is basically that a simple neural network model is applicable to the prediction of foreign exchange rate, which seems to be the case from the results obtained here. But when it comes to making profit, they nuance their conclusions, and so do we, although the direction taken seems promising.

Chapter 6

Conclusions and Further Work

The present study has led to the following observations:

- The forecasting results are promising, despite a doubt remaining concerning the implementation. The system seems to achieve better results with shorter time periods, for which the profit is also generally better. Shorter testing periods tend to produce worse results in terms of NMSE, but better profits.
- Increasing the number of currency pairs in input led to very different results, if not always better ones. A further study will have to be conducted, taking care of the linkage between currencies. The profit seems not to suffer much from increasing the currency pairs used in input.
- As expected, the input model that performs best is the **long progressive averages** model, that should capture all trends. Surprisingly, the **14 daily averages** model performs also very well, and gives much better results in terms of profit. This is probably caused by the fact that both strategies work on a short-term basis. Long-term strategies will have to be investigated.
- Boolean political factors tend to improve the forecasting results without causing a drop-off of the profit. This particular point should be investigated further by entering better factors.
- Despite the good forecasting results, profit seems to be more difficult to achieve with simple strategies. More elaborate strategies have to be investigated.
- The evolution of the profit during the training shows a lot of instability for some situations. In particular, it often decreases when training the network

further. A system that maximizes the profit instead of minimizing the MSE could be imagined, or at least a system that includes the profit somehow in the neural network.

- It will be interesting to try different learning approaches, as well as different neural architectures. Recursive neural networks and diffusion networks are good candidates. NOVEL [15] and Constrained Learning [14] are optimizations that could be worth investigating as well.

To compare the present study with the work of Yao *et al.* [18], we draw more or less the same conclusions regarding the forecasting results of a comparable simple model. However, we are less enthusiastic concerning the profit. In short, we wouldn't use this model as it is with real money.

Appendix A

Implementation

This appendix describes briefly the implementation of the FXAgent platform that was developped to make the tests for this study. The first section gives information about the general conception of the program, and the second section contains UML class diagrams for the most important classes.

A.1 General Conception

FXAgent is a Java Applet, developped using Sun's Java2 Development Kit version 1.3.1 [13]. It consists in a full MVC[1] GUI and we tried to make it as easy as possible to extend by adding more data, input models and strategies.

The core of the program is the *engine* package (*see the API documentation and Figure A.5*). This package regroups all the classes necessary to run the implemented perceptron. The perceptron itself implements the algorithm described in Section 2.4. The GUI is implemented as a MVC separate from the engine (*see Figure A.2*). Finally, all utilitary classes necessary for the communication between the GUI and the engine are included in the *communication* package (*see the API documentation and Figure A.3*).

[1]MVC stands for Model View Controller and is a well-known paradigm for programming graphical user's interfaces (GUI).

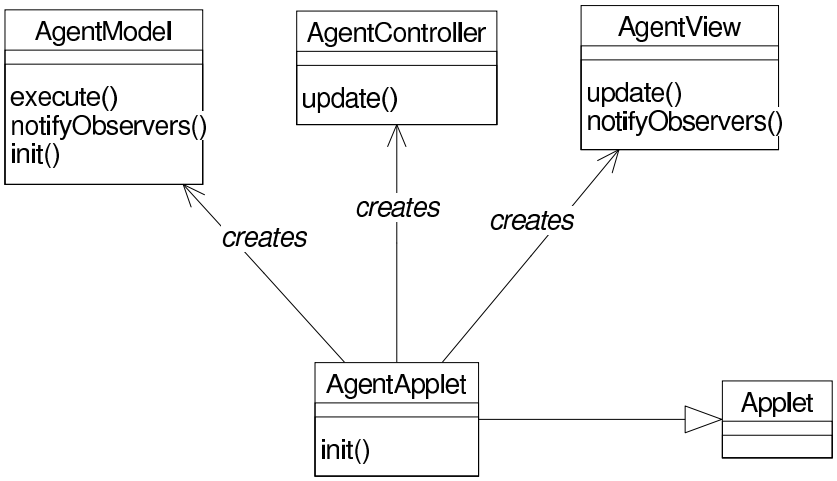


Figure A.1: UML diagram for the agent applet

A.2 UML Diagrams

Figures A.1 to A.7 are UML class diagrams showing the most important classes of the program, together with their most important interface methods and relations. The notation used is the one of Rational Rose [2], and may not follow strict UML specifications as defined by OMG [12]. The purpose of the diagrams is to illustrate the general conception of the program, not to describe it in details. Therefore, not all attributes, methods and relations are drawn.

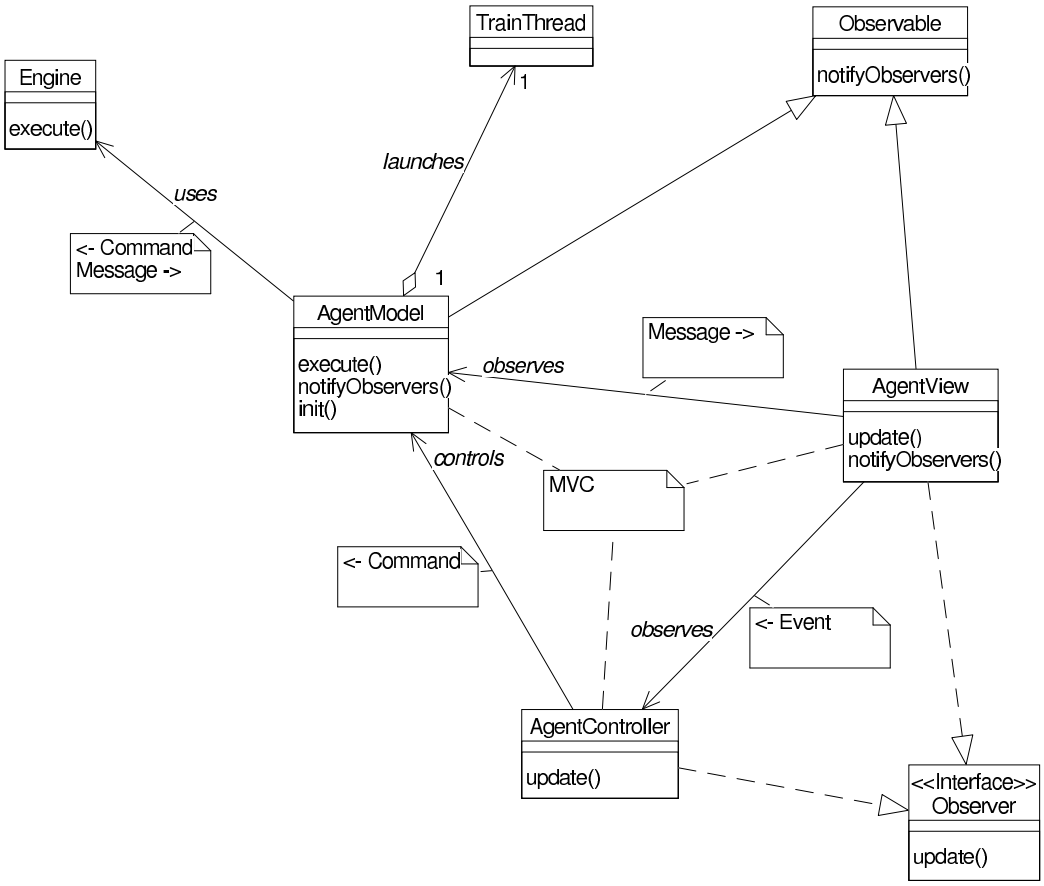


Figure A.2: UML diagram for the MVC

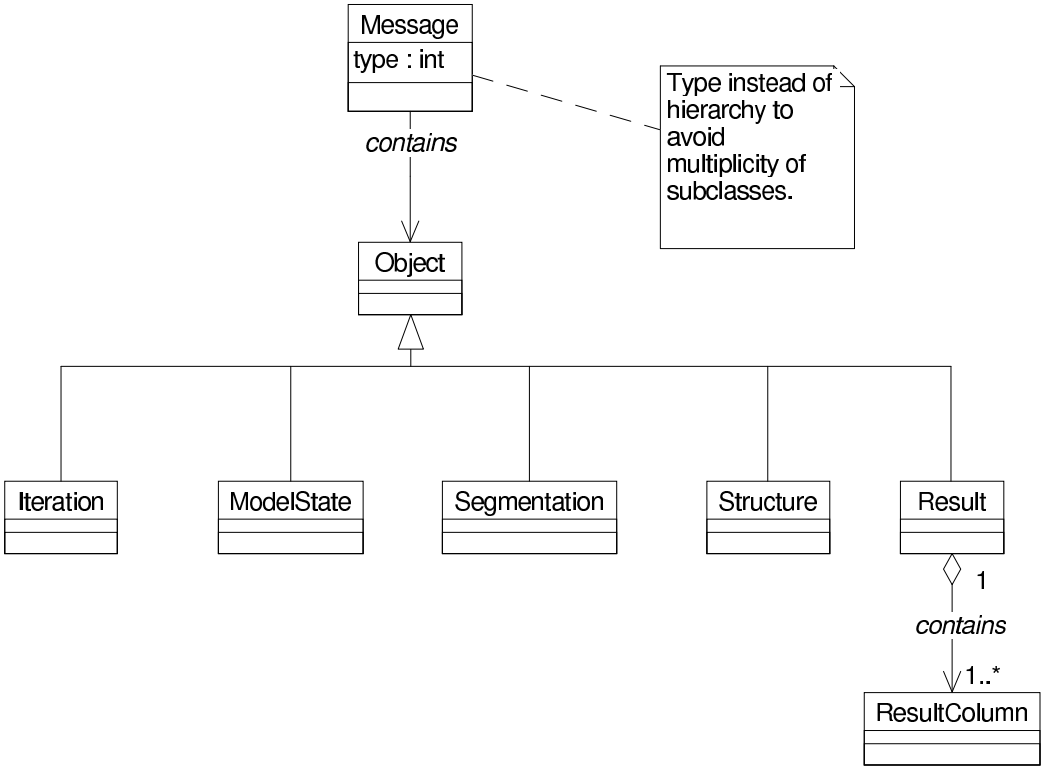


Figure A.3: UML diagram for the message

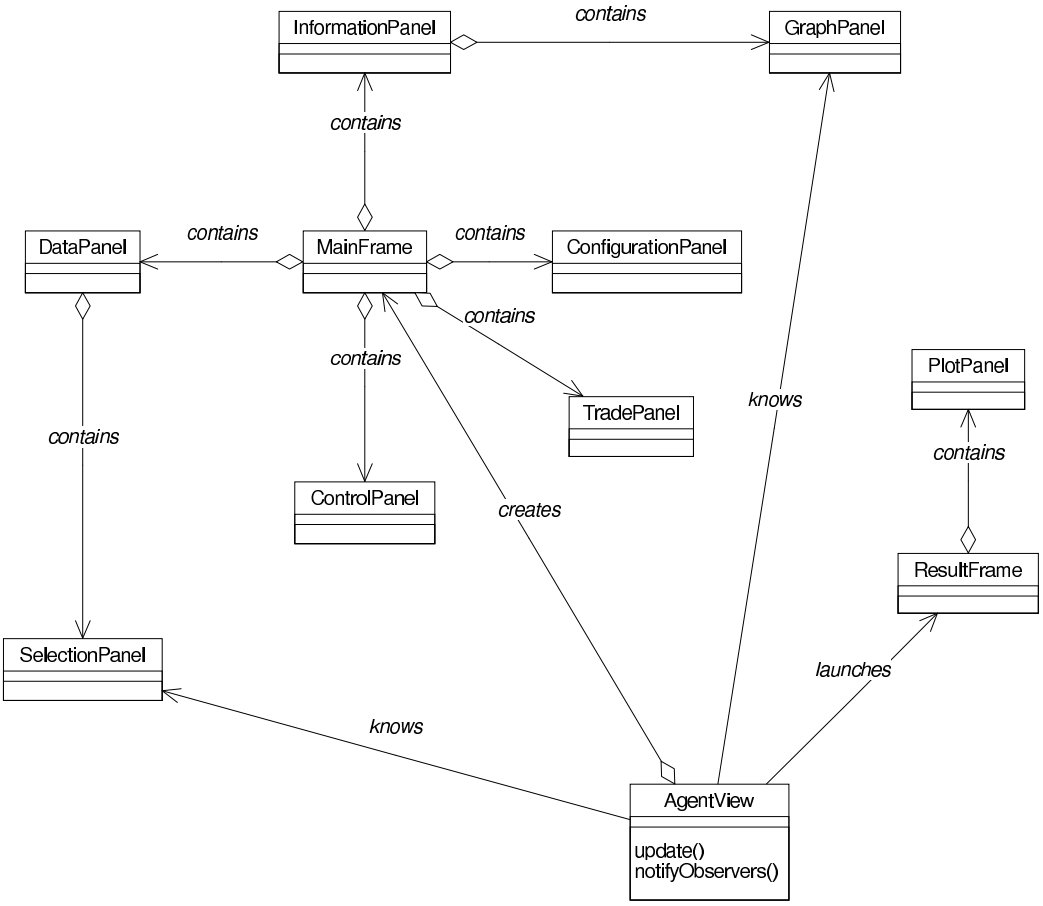


Figure A.4: UML diagram for the MVC view

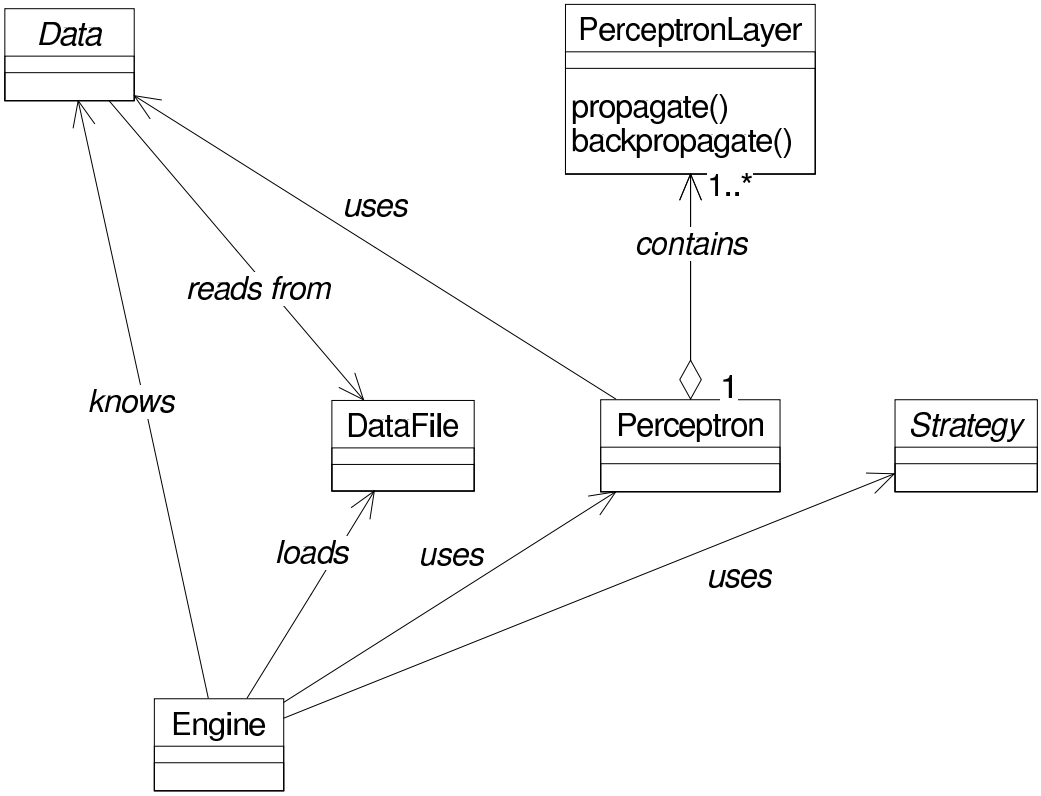


Figure A.5: UML diagram for the engine

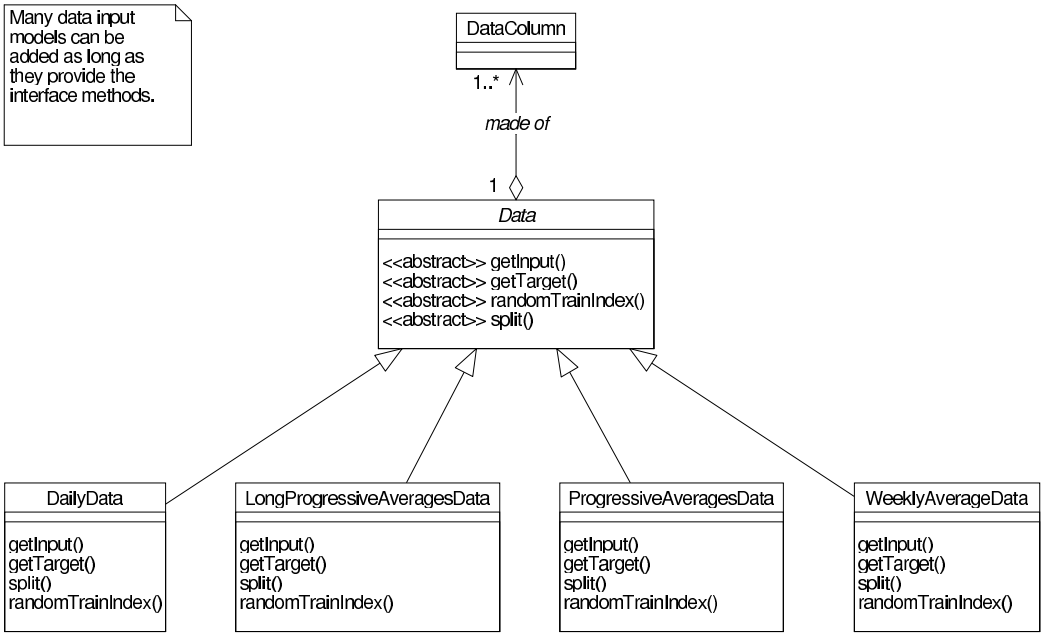


Figure A.6: UML diagram for the data

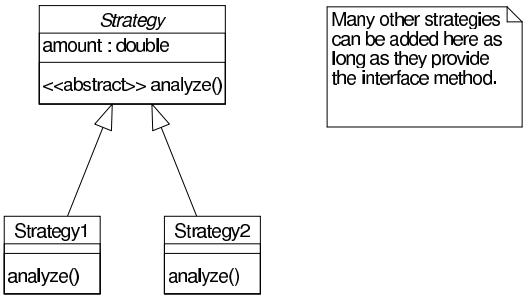


Figure A.7: UML diagram for the strategy

Appendix B

User's Manual

B.1 What is FXAgent?

FXAgent is the platform developped to play with currency exchange rates predictions made by the neural network. It is a Java Applet that runs on browsers that include a Java Virtual Machine 1.3 or later. Popular browsers do by mean of the Java Plugin, that can be downloaded from Sun's Java web site.

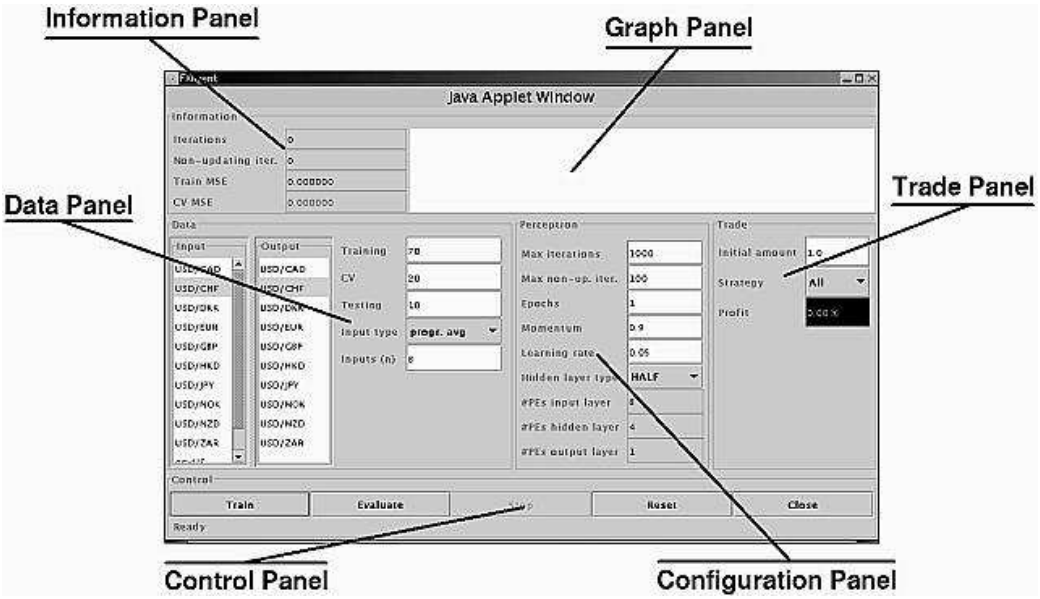
The interface was developped to provide the necessary features to change the various parameters of the model, and the necessary information to see the results in a human readable form. It should be pretty straightforward, provided you under-

stand a minimum about currency exchange and neural networks.

Enjoy!

B.2 Interface

Here is how what the interface looks like when the applet is started:



B.2.1 Information Panel

The **Information Panel** gives the following information (up to bottom):

- **Iterations** is the number of iterations since starting the training phase.
- **Non-updating iter.** is the number of non-updating iterations since the last time weights were saved.
- **Train MSE** is the Mean Squared Error over the training set.
- **CV MSE** is the Mean Squared Error over the cross-validation set.

The colors for **Train MSE** and **CV MSE** are the same as in the **Graph Panel** (*see Section B.2.2*).

B.2.2 Graph Panel

The **Graph Panel** plots in real time the **Train MSE** and **CV MSE** values (*see Section B.2.1*). The scale is adjusted dynamically every ten iterations so that the maximum value is plotted at middle-height in the area. The plot always spans over the whole width of the graph, compressing the curves to the left at each iteration. The colors are the same as in the **Information Panel** (*see Section B.2.1*).

The **Graph Panel** also plots the profit (*see Section B.2.7*). Since the profit is usually two orders of magnitude smaller than the Mean Squared Errors, the plot uses another scale to draw the curve. The scale is also adjusted dynamically. Negative profits aren't visible on the plot.

B.2.3 Data Panel

The **Data Panel** allows you to play with the parameters associated with the data. The panel is made of two subpanels: **Selection Lists** (*see Section B.2.3*) and **Configuration** (*see Section B.2.3*).

Selection Lists

The **Selection Lists** allow you to select which factors you want for input and output respectively. The currencies are upper-case, and the boolean indicators are lower-case and appear only in the **Input List**.

- **To select a factor**, click on its name in the desired list.
- **To select multiple entries**, hold the Ctrl key pressed while clicking on the entries you want to select.

Configuration

The **Configuration** part allows you to change various parameters of the data. The first three fields are for the segmentation, and works as follows:

- **To change the percentage used for training**, enter the new value in the **Training** field and press enter to validate. The **CV** field will be modified accordingly to ensure the total is 100%.

- **To change the percentage used for cross-validation**, enter the new value in the **CV** field and press enter to validate. The **Training** field will be modified accordingly to ensure the total is 100%.

- **To change the percentage used for testing**, enter the new value in the **Testing** field and press enter to validate. The **Training** field will be modified accordingly to ensure the total is 100%.

In order to define a new segmentation, you should always start by entering the **Testing** field, because it will never be changed when you modify the other two, whereas the **Training** and **CV** fields are linked together to ensure the total is 100%.

B.2.4 Control Panel

The **Control Panel** allows you to control the system. There are five functions:

- **Train** starts a training phase.

- **Test** opens a window showing a graph of the results (*see Section B.2.5*).

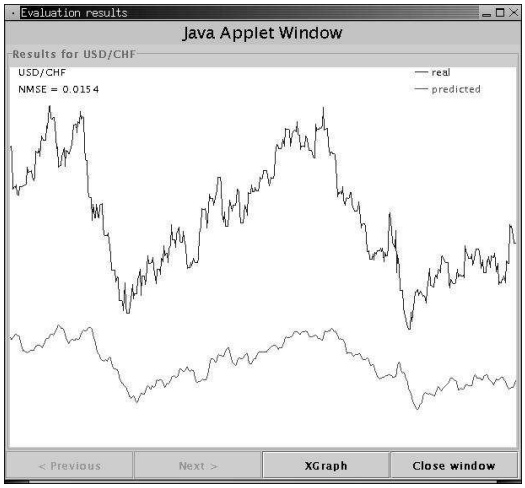
- **Stop** stops a training phase.

- **Reset** resets the system, forgets all the weights.

- **Close** closes the window.

When you click on a button, the whole interface puts itself in a different mode and may allow/disallow components according to what it is doing and what can be done. For example, if you start a training phase, you cannot interact with the interface anymore, except for stopping it or closing the window. Once you stop the training, you cannot change parameters related to the data anymore unless you reset the system, etc.

B.2.5 Result Window



The **Result Window** displays the plots for the predicted and real values over the testing set. Under the graph are four buttons:

- **Previous** allows you to cycle through the selected outputs. It isn't available if only one output has been selected.
- **Next** allows you to cycle through the selected outputs. It isn't available if only one output has been selected.
- **XGraph** allows you to switch to the **XGraph mode**, displaying the results in a way that is understood by *xgraph*, an open-source program for plotting functions in X-Window. The text is copied to the standard output, the Java console if you are in a browser, or the console that started the applet if you use *appletviewer*.
- **Close window** allows you to close the window.

B.2.6 Configuration Panel

The **Configuration Panel** allows you to set various parameters of the perceptron. There are five text fields and one combo box:

- **Max iterations** is the number of maximum iterations for training. The training stops when reaching this number.

- **Max non-up. iter.** is the number of maximum non-updating iterations. The training stops when reaching this number, that is when no weights have been saved for this number of iterations.
- **Epochs** is the number of epochs for training (e).
- **Momentum** is the momentum term ($0 < \alpha \leq 1$).
- **Learning rate** is the learning rate ($0 < \eta \leq 1$).
- **Hidden layer type** is the type of hidden layer. The type is a fraction of the number of processings elements (PEs) in the input layer.

Below are three informational text fields:

- **# PEs input layer** is the number of PEs in the input layer. It depends on the number of inputs selected and the input model.
- **# PEs hidden layer** is the number of PEs in the hidden layer. It depends on the number of PEs in the input layer and on **Hidden layer type**.
- **# PEs output layer** is the number of PEs in the output layer. It depends on the number of outputs selected.

All text fields require that you validate the entry by pressing *enter* when done. This is necessary to ensure that no silly value is entered, like a momentum > 1.0.

B.2.7 Trade Panel

The **Trade Panel** allows you to select the trading strategy, set the initial amount if applicable and see the profit. There are three items:

- **Initial amount** allows you to set the initial amount for trading. This doesn't affect all strategies, only those that depend on the initial amount. The currently implemented strategies don't.
- **Strategy** allows you to choose the trading strategy.
- **Profit** indicates the profit in real time. The profit is also plotted on the **Graph Panel** (*see Section B.2.2*).

Bibliography

[1] "OANDA.com, The Currency Site".
http://www.oanda.com.

[2] "Rational Software".
http://www.rational.com.

[3] W. Antweiler. "The EURO, Europe's New Currency".
http://pacific.commerce.ubc.ca/xr/euro/euro.html.

[4] R. Baillie and P. McMahon. *The Foreign Exchange Market, theory and econometric evidence*. Cambride University Press, 1990.
ISBN 0-521-39690-5.

[5] S. Bell and B. Kettel. *Foreign Exchange Handbook*. Graham & Trotman, London, 1983.
ISBN 0-86010-385-4.

[6] W. Gerstner. Supervised Learning for Neural Networks: A Tutorial with JAVA-exercises. Technical report, 1988.
http://diwww.epfl.ch/~gerstner/wg_pub.html.

[7] C. L. Giles, S. Lawrence, and A. C. Tsoi. Noisy Time Series Prediction Using a Recurrent Neural Network and Grammatical Inference. *Machine Learning*, 44(1/2):161–183, July/August 2001.
http://citeseer.nj.nec.com/giles01noisy.html.

[8] R. Herbrich, M. Keilbach, T. Graepel, P. Bollmann-Sdorra, and K. Obermayer. Neural Networks in Economics: Background, Applications, and New Developments. *Advances in Computational Economics, volume 11*, pages 169–196, 1999.
http://stat.cs.tu-berlin.de/publications.

[9] J. Hicks. *Monnaie et Marché*. Economica, Paris, 1991.
ISBN 2-7178-1919-3.

[10] J. C. Hull. *Options, futures and other derivatives*. Prentice Hall Intl. Editions, Upper Saddle River, 4th edition, 2000.
ISBN 0-13-015822-4.

[11] J. R. Movellan, P Mineiro, and R. J. Williams. A Monte-Carlo EM Approach for Training Partially Observable Diffusion Networks. November 2000.

[12] Object Management Group (OMG). "UML".
http://www.omg.org/uml.

[13] Sun. "The Source for Java Technology".
http://java.sun.com.

[14] B. W. Wah and M. Qian. Time-Series Predictions Using Constrained Formulations for Neural-Network Training and Cross Validation.
http://www.manip.crhc.uiuc.edu.

[15] B. W. Wah and Y. Shang. Global Optimization for Neural Network Training. *IEEE Computer, volume 29, no. 3*, pages 45–54, March 1996.
http://www.manip.crhc.uiuc.edu.

[16] J. Yao, Y. Li, and C. Tan. Forecasting the Exchange Rates of CHF vs USD Using Neural networks. *Journal of Computational Intelligence in Finance, Vol.5, No.2*, pages 7–13, 1997.
http://citeseer.nj.nec.com/yao97forecasting.html.

[17] J. Yao, H. Poh, and T. Jasic. Foreign Exchange Rates Forecasting with Neural Networks. *International Conference on Neural Information Processing, Hong Kong*, pages 754–759, September 1996.
http://citeseer.nj.nec.com/yao96foreign.html.

[18] J. Yao and C. Tan. A Case Study on Using Neural Networks to Perform Technical Forecasting of Forex. June 1999.
http://citeseer.nj.nec.com/235723.html.

Index

activation function, 7
applet, 61
ask price, 2
axon, 5

backpropagation, 8
base currency, 2
bid price, 2

class diagram, 61
cross rate, 3
currency exchange, 1

dendrite, 5

ECU, 12
Efficient Market Hypothesis, 3
EMH, 3
epochs, 9
Euro
    participating currencies, 12
    pseudo-exchange rates, 12
exchange rate, 2, 3

Forex, 3
formal neuron, 5
FX, 3

GUI, 61

Java, 61

learning rate, 9
liquid market, 3
long position, 2

momentum, 9
multilayer perceptron, 7
MVC, 61

neural network, 5
NMSE, 21
normalized mean squared error, 21

perceptron, 6
processing element, 6
profit, 22

quote currency, 2

short position, 2
sigmoid, 7
soma, 5
spread, 2
synapse, 5

trading strategy, 23

UML, 61