The University of Birmingham
School of Computer Science
MSc in Natural Computations

Summer-project

Financial Time Series Prediction

Marcin Radlak

Supervisor: Dr John Bullinaria

September 2007

Abstract

The future is often predictable from present and past events. Which aspects of those events are more important? Which are less? What is prediction in general? What factors influence its accuracy? Is it possible to build software which will be able to imitate experienced investors decision processes, or even outperform them? What should the system consists of? How good are the results that can be obtained? These are only a few of many questions which have motivated research in Time Series Prediction. This project is an attempt to begin to answer the above questions by investigating the application of Artificial Neural Networks trained ("evolved") using Evolutionary Programming techniques. Taking advantage of an ensemble approach, the idea of integrating outputs for different time frames is presented. Additionally, as Financial Time Series Prediction is often time critical, emphasis has been put into the architecture and implementation of the prediction system to achieve the highest speed possible. Experiments with various parameters settings are performed and analyzed.

Keywords: Time Series Prediction, Evolving Neural Networks, Evolutionary Programming, Artificial Neural Network, Financial Prediction.

Contents

List of Figures

Chapter 1

Introduction

29th of October 1929, 19th of October 1987, August 1998. What do these dates have in common? They are described as "black days", because they are dates when world stock markets crashed. Major indexes dropped more than 10% during few sessions ended up even more than 25% down from their highest. Billions of dollars melted within hours, putting many individuals into bankruptcy.

Was it possible to predict those turmoils? Is it possible to predict how price of a stock share will change during next few days, months or years. When is the best time to enter the market and when to quit in order to achieve high profit?

To be able to answer above question, one would have to engage not only into laws of economics, but mathematics, physics, computer science, psychology, etc. There are always overwhelming number of factors influencing behaviors of investors all over the world. And there are always many possible scenarios. One thing is beyond all doubts: there is no certainty which one is most probable to occur.

The main motivation of this project is to develop a system which will eventually predict future changes in stock or currency markets. Achieving high profitability with simultaneous reduction and minimization of loses is the main goal of the problem. How was it approached? This report will explain every aspect in detail.

1.1 Based on Nature

Decision process is very complex task and includes tremendous number of factors to be included. Drew (1941), concluded this situation as:

"In fact, simplicity or singleness of approach is a greatly underrated factor of market success. As soon as the attempt is made to watch a multiplicity of factors, even though each has some element to justify it, one is only too likely to become lost in a maze of contradictory implications (...). The various factors involved may be so conflicting that the conclusion finally drawn is no better than a snap judgment would have been"

Despite the fact, that this work has been done more than 65 years ago, it provides interesting conclusion: there are systems required to aid investor in a decision making

process. This should allow to uncover hidden relationships between mentioned factors, thus as a results - provide a solid founded information, which further can be interpreted according to experience and knowledge of the investor. Obviously, the goal is to provide those kind of results, that will not require much experience from a person using it.

To meet the requirement, methodology adopted in this project has been based on psychology. To be more specific - human judgement (Slovic (1972)). If one would observe decision process of experienced investor, following actions could be extracted:

- **Collect as much information as possible**

  Through research, statistical analysis or planning, information collected is a basis for a future decision. At this stage, investor goes through financial reports, technical or fundamental analysis indicators, domestic indicators and any other sources of information. News agencies are also a source of up to date political, economical and social information. These often have a substantial impact on world of investment.
- **Decide what is significant and exclude unimportant factors.** After preliminary data collection and familiarization with different values, one tries to build a model of dependencies in data, which factors are most influential to a decision, which can be excluded immediately. This step is to allow elimination of redundant data, thus decision could be made based on lower number but strong arguments.
- **Analyze preprocessed information creating different scenarios.** Having significant data, one tries to predict what are possible scenarios that may occur.At this time it is required that dimension of information from first step has been properly reduced in second step. To much data may be overwhelming, reduce clarity and significantly increase difficulty of the task.
- **Decide which predictions are more likely to happen.** According to investor's experience, current situation is analyzed. This is often done with respect to the past. When similar events have occurred it is more likely that sequence of events may repeat again.
- **Give most favorable opinion.**After accomplishing previous steps, investor has to decide which sequence of events is more likely to happen. This is the time when actual decision is made.
- **Combine decision with previous experience.** Each time decision has to be done, investor builds his experience. It is crucial part of learning and for future reference, one has to combine new facts with knowledge already possessed.

1.2 Digital Analogy

This paper adopts general concept of the process described in section 1.1, but transformed into analytical form. Whole system is based on Artificial Neural Network which is a common tool for predictions (McNeils (2005), Brabazon & O'Neil (2006) or Weigend & Gershenfeld (1993)). Because of high number of parameters, which are required to be set

(i.e. weights, number of hidden neurons, activation functions), Evolutionary Programming algorithm has been used to help with this process. Idea of using Evolutionary Computations with Neural Networks is sometimes called "evolving Neural Networks" (i.e. Yao (1999)), because it is similar to Darwinian evolution, where characteristics of organisms change with each generation.

Input for the system is prepared as a collection of Economic Indicators, Stock Market Indexes, Inflation Rate, Interest Rate, Bond Rates, Currency Exchange Rates and many more. Because of very little understanding of which factors influence final prediction, method of dimension reduction (Principal Component Analysis) has been applied in order to reduce computation power demand for ANN evolution.

When output is produced by Neural Network, it's value is often far from desired output. Therefore, population of ANN's has been applied to provide a voting system, where output is mutually combined from each member of population. This method, Ensemble of Neural Networks, has been proved to lower overall prediction error.

Whole of this process is presented on figure 1.1.

1.3 Structure of the Work

Chapter 2 will provide a description of each part of the system according to figure 1.1. Additionally, problem of speed will be described and demonstration of how algorithm has been optimized to work more than 40 times faster using combination of Matlab and ANSI C programming languages. Chapter 3 will describe various tests for input and output data, results will be analyzed and conclusion provided. Chapter 4 will focus on overall performance of the system. Finally, chapter 5 will summarize findings and conclusion giving some directions for future work.

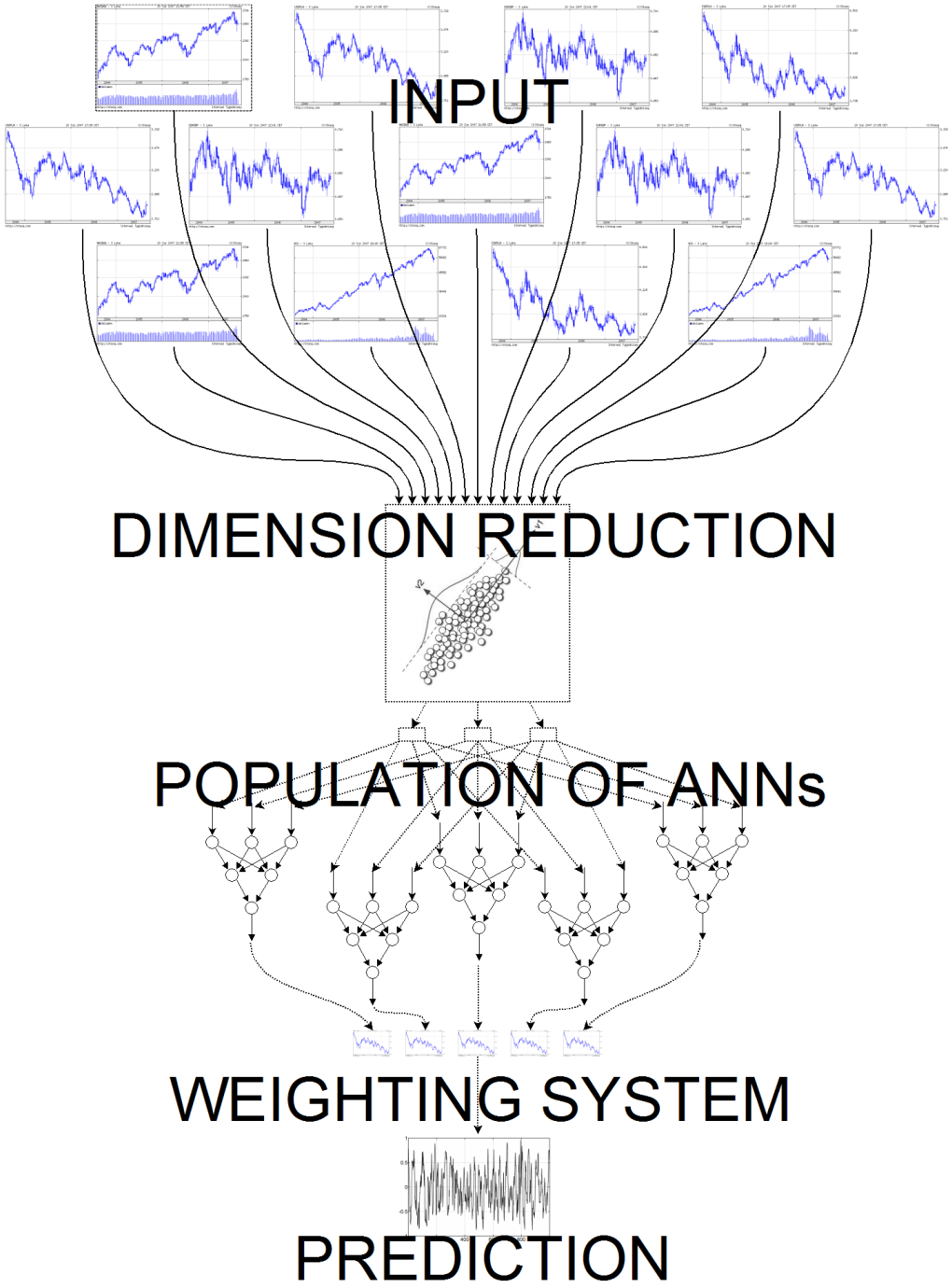


Figure 1.1: Modules of the project

Chapter 2

Core engine: ANN + EC

To meet requirements of financial market investor, trading system should provide approximate information about future. According to their accuracy, they will be used in decision making. The goal is to produce a system which will approximate future very well. How this can be achieved?

If one was to simplify process of prediction, this could be expressed as a series of events which lead into some action. This is generally adapted process of thinking which can be observed in people everyday behavior, i.e. weather prediction - if there are dark clouds in the sky it is more likely that it will start raining soon, or if there was lots of rain during past few weeks, it is very possible that some areas will be flooded.

No matter what area prediction is being performed in, the task is very difficult. Additionally, there are areas which are very restrictive regarding the output produced. This area is undoubtedly finance, where any wrong prediction may cost fortune.

These examples are trivial, but fit to general concept of future prediction:

Hypothesis 1 *Current and past conditions determine future action*

The idea behind hypothesis 1 is that future is a function (probably non-linear) of n arguments

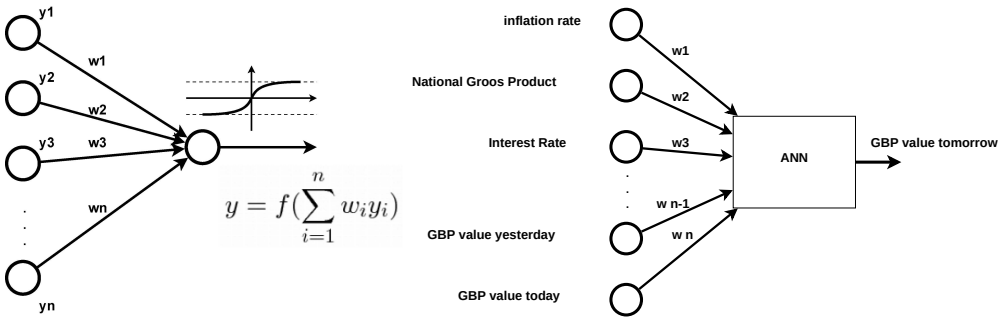
$$y = f(x_1, x_2, ..., x_n) \tag{2.1}$$

Therefore, the first part of the process is to determine what are those arguments $x_1, x_2, ..., x_n$ that influence value of f - y, which defines future event. Secondly, by knowing input and output (which are usually known based on historical data), one has to construct a model of function f which will be able to respond for known as well as for unknown input accurately. In other words this process is called approximation. And when it comes to approximation of function, Neural Networks as mathematical tool have many significant advantages amongst other methods.

2.1 Artificial Neural Networks

Artificial Neural Networks are great invention of past years from mathematical point of view. In short, this is a tool which mathematically imitate biological brain. As real brain is able to learn patterns, so are the ANNs. As real brain can produce deterministic action based on well known input, so are ANNs. There are many more similarities, but the scope of this project is limited and no detailed description of ANNs will be provided. More details on the subject can be found in i.e.: Bishop (1995), Hu & Hwang (2002), Krose & van der Smagt (1996) or Haykin (2005). There are many areas where they are successfully applied, just to mention text or speech recognition, objects identification etc. They can be trained to do specific task (produce desired output for given input) by adjusting weights during the learning process. This process require training data which consists of an input series and relating output series.

In case of time series prediction, ANNs could be excellent tool because of their properties. Basic structure and analogy to decision process is presented on figure 2.1



(a) General ANN (b) Schmatic prediction of GBP value

Figure 2.1: ANN properties according to prediction properties

Figure 2.1(a) presents General ANN architecture (simplest part of network - neuron). It has inputs and produce an output by assigning importance weights to each input value. Afterwards, sum of product between weights and inputs is calculated and modified by nonlinear activation function. This process is analogical to expert reasoning (presented on GBP value at figure 2.1(b)):

1. decide what factors are important for the GBP value (or any other value prediction is required for)
2. assign significance weights
3. collect those values and analyze based on past knowledge
4. produce decision.

Therefore ANNs have been chosen for their analogy to real world data processing.

By creating links between neurons, network is created. Many neurons can be connected to one neuron (its inputs), and each connection has assigned weight value, which indicates

how important the connection is (higher value - higher importance). This neuron produce output in the following sequence:

1. calculate sum of products of connected neuron inputs and weight value

$$\sum_{j=1}^{n} w_{ij} x_j \tag{2.2}$$

2. add bias

$$\sum_{j=1}^{n} w_{ij} x_j + 1 w_{1b} \tag{2.3}$$

3. pass it through nonlinear transfer function.

$$y_1 = f_i(\sum_{j=1}^{n} w_{ij} x_j + 1 w_{1b}) \tag{2.4}$$

This process and basic ANN architecture is visualized on figure 2.2:

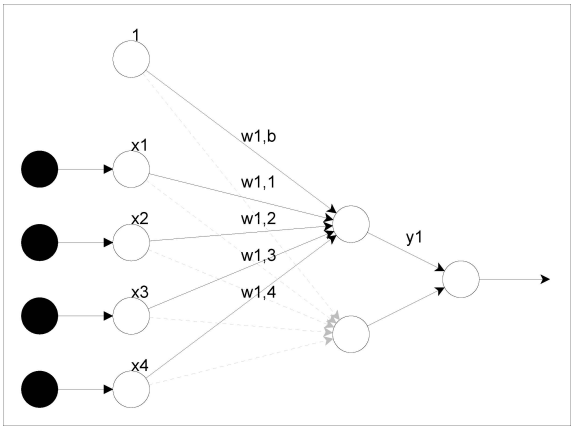


Figure 2.2: Basic Feed Forward ANN architecture

Units which are completely separated from each other form layers. There are unlimited number of layers possible. Complexity of network is measured as number of neurons and number of layers. Higher those numbers - more complex and more computational demanding the structure.

2.1.1 Applied methodology of calculation

All calculations performed for the purpose of this project have been done using Matlab IDE. As it is most efficient with matrix calculations, accurate representation has to be designed, to make it run fast enough. The reason why Matlab has been chosen is because it allows amazingly fast design, programming and testing of algorithms with great visualization features. By designing working algorithm in Matlab, it will be easier to translate it into lower level language like C or C++ which are much faster in execution of the pro-

grams which Matlab is not designed for. Therefore, ANN architecture has been converted into matrix notation and this is described as follows.

One way of visualizing Neural Network is a graph notation: neuron is a vertex, connection between neurons is an edge. Obviously, if network has only forward connections, graph is unidirectional: connection from unit u_1 to u_2 is the same as from unit u_2 to u_1. On the other hand, recurrent network is represented by directed graph, where connections introduced above describe two completely different values.

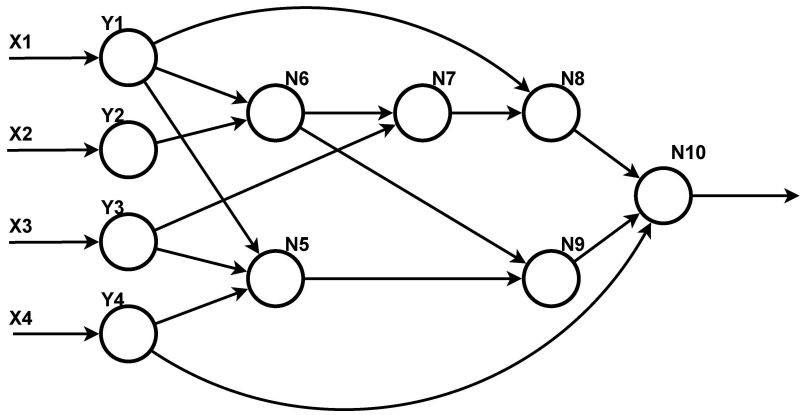


Figure 2.3: Structure according to example weights matrix

y		1	2	3	4	5	6	7	8	9	10
y_1	1		0	0	0	.1	.2	0	.1	0	0
y_2	2	0		0	0	0	−.2	0	0	0	0
y_3	3	0	0		0	.7	0	5	0	0	0
y_4	4	0	0	0		.3	0	0	0	0	.1
y_5	5	0	0	0	0		0	0	0	.1	0
y_6	6	0	0	0	0	0		.7	0	−.3	0
y_7	7	0	0	0	0	0	0		.3	0	0
y_8	8	0	0	0	0	0	0	0		0	−.4
y_9	9	0	0	0	0	0	0	0	0		.3
y_{10}	9	0	0	0	0	0	0	0	0	0	

Figure 2.4: Example ANN representation in matrix notation

This implies the fact, that whole structure can be represented as a connection matrix. Moreover, if instead of using 1 and 0 as values indicating presence or absence of connection respectively, one would use value of weight assigned to each connection. To explain this concept matrix 2.4 has been prepared. It is a feed forward structure which has been visualized on figure 2.3 as a diagram. By analyzing provided examples and drawing from Yao (1999), general concept has been developed and presented on figure 2.5

One could ask a question how to detect layers from this notation? The answer is based on the assumption that neurons at one layer cannot be connected. Thus layers should

		x_1	x_2	...	x_{in}	y_{n1}	y_{n2}	...	$y_{n(n-1)}$	y_{nn}
y_1	x_1		0	...	0	$w_{1,n1}$	$w_{1,n2}$	...	$w_{1,n(n-1)}$	$w_{1,n}$
y_2	x_2	0		...	0	$w_{2,n1}$	$w_{2,n2}$	...	$w_{2,n(n-1)}$	$w_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
y_{in}	x_{in}	0	0	0		$w_{in,1}$	$w_{in,n2}$	...	$w_{in,n(n-1)}$	$w_{in,n}$
y_{n1}	x_{n1}	0	0	0	0		$w_{n1,n2}$	...	$w_{n1,n(n-1)}$	$w_{n1,n}$
y_{n2}	x_{n2}	0	0	0	0	0		...	$w_{n2,n(n-1)}$	$w_{n2,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$y_{n(n-1)}$	$x_{n(n-1)}$	0	0	0	0	0	0	0		$w_{n(n-1),n}$
y_{nn}	x_{nn}	0	0	0	0	0	0	0	0	

Figure 2.5: General ANN representation in matrix notation

be detected through columns, where more than one weigh is equal to 0, i.e. taken from figure 2.3 where $N5$ and $N6$ are called $Layer1$ (based on already provided assumption): $w_{56} = w_{65} = 0$ and $w_{89} = w_{98} = 0$).

By rewriting columns in a way, that zeros are omitted, and each subsequent column is stacked on top of previous one, whole network can be represented as one column, thus whole Neural Network can be represented as a real value string, where each parameter corresponds to a weight between specified neurons. At this stage, table 2.5 is very useful. One simply has to rewrite two dimensional matrix into one dimension to achieve desired representation of ANN, which is fully suitable for many optimization techniques, i.e. Evolutionary Programming (EP) described in detail in section 2.2.

2.1.2 Issues

At first sight, ANN seems to be superior tool specially designated for kinds of problems, that have been faced during this project. But there are many issues which arise during design process, which are mostly based on the Neural Network itself, i.e.: weights initialization, weights values, stopping criteria, architecture, input, output, etc. These are non-trivial problems, because i.e. properly designed structure can significantly reduce computation complexity. This has been illustrated on exclusive-or (XOR) problem (i.e. Bemley (2001)), where after adding connection between input and output, size and training time of the network has been reduced, while keeping correct results. Major issues will be described in following sections. It is important to remember, that while constructing this model, many trade offs are required, like speed has to be chosen over accuracy. This is to illustrate that the task is very difficult and path towards satisfying results is quite long.

Input

One of the most important decision one has to make regarding Artificial Neural Networks:

- input size - high number of inputs may results in increased complexity of Neural Network, thus more time is required to be able to fully train the net,
- input structure - what exactly should be provided as input and what should not; some data do not provide any useful information thus input should be selected carefully,
- size of input - it is also very important, because if the training period, in terms of time series prediction, may result in network memorizing past to much, thus will be unable to perform well in rapidly changing environment, like stock market is; if training period is to short, ANN may be unable to learn patterns hidden in data set, thus only noise will be produced,
- data significance - this factor has already been mentioned briefly; what is meant by data significance is the fact, that investor may already have some understanding of the data and based on his experience, would be able to choose only those factors, which might influence predicted value the most.
- preprocessing methods - this part of a system is mainly to help in choosing right data as an input; once again - it is possible to collect all possible data, which might be correlated with predicted value, but speed of calculation would such slow, that it would take years to produce at least satisfactory results; very common technique used at this stage is Principal Component Analysis (Jackson (1991))

Output

It is not enough to know if the price will increase following day or at any desired point in the future. Much more desired value would be how much that increase will be. By choosing accurate representation of the output, significant improvement in ANN's performance can be observed.

- choosing raw value as an output; if value of FTSE100 was approx 6000, with daily changes of up to 2% (5780 - 6120), it seems to be extremely difficult task to predict this value.
- choosing percentage change as an output; one has to find a value between i.e. (-2;2); moreover, exact value is not required, as even prediction with precision set to $\pm$ 1% gives a lot of information. But what is more important, this is huge complexity reduction according to previous example
- to summarize - output structure - i.e. future price, ratio of change or direction of change are only few of values, which could be applied as an output to neural network
- output transfer function - depends on the value which will be used; if data mean would be approx. 0, then tangent sigmoid function is recommended; otherwise, if mean is equal to approx. 0.5, then logistic sigmoid function might be a better choice.

Architecture

- by choosing the right one, less complex structure may give the same results than more complex one. Computation complexity may be significantly reduced.

- network architecture - i.e. feed-forward or recurrent, or maybe other structure
- weights update - i.e. what algorithm should be used (back-propagation, simulated annealing)
- transfer function for each layer (unit) - i.e. sigmoid, logistic, linear, radial basis.

Fitness evaluation

When evaluating performance of the model, one has to construct accurate measure, i.e

- Root Mean Square Error (RMSE) - this is very common method of comparing output produced by network to desired output. This is a square root of squared sum of differences between desired output and actual output

$$RMSE = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(d_i - y_i)^2} \tag{2.5}$$

  One has to be careful relying on RMSE measure, as trained network with very low error, may still produce bad prediction, i.e. negatively correlated to desired prediction
- very intuitive overall measure of trading system would be its ability to earn money. This could be tested by simulation of trading with signals generated on historical data

Weights

Weights are fundamental part of Neural Networks. High values are assigned to significant data, low to insignificant.

- Every weight influence each other. Thus method for appropriate adjustment is highly desired. Therefore, in this project Evolutionary Programming as real value optimization method has been used
- Size of training data. Learning process require samples evaluate them, at assign weight, so they correct output is produced. Fitness function is used during this process.

Overall success project is based mostly on mentioned above factors. Each of them are characterized by tradeoff one has to make when choosing specific values. Usually, only by trials and errors it is possible to obtain satisfying results. Achieving best set of values is very laborious task, because every aspect should be considered and implemented with the best available solutions.

To illustrate issues described above, figure 2.6 has been prepared. As clearly presented, basic part of the system, single neural network consists of 3 main modules: input, inside and output.

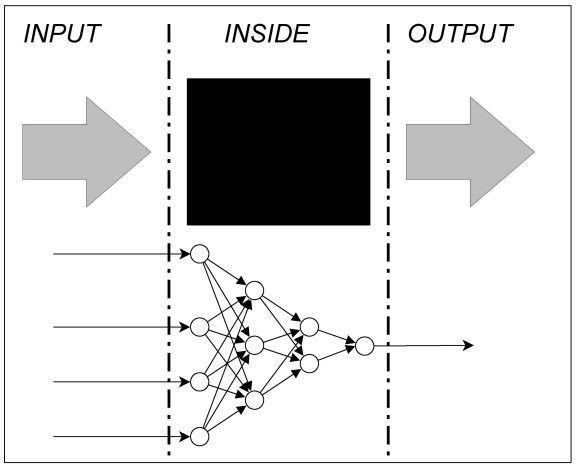


Figure 2.6: Prediction issues diagram

2.2 Evolutionary Computations

Evolutionary Computations (Holland (1975)) have become very popular in many areas of science and technology. These methods of population based, global optimization have been found to produce satisfying results for many optimization problems, i.e. Vehicle Routing, Scheduling, Packing, classification and many more. They have also been successfully applied to 'evolve Artificial Neural Networks' (i.e. Yao (1999)).

The idea behind EC is that individuals from population of solutions compete between each other. Those with best fitness, based on Darwinian Evolution Theory, have the highest chances of survival. But, not only competition is present within a population. As in real evolution, artificial one is based on mating habits. Parents are selected in a selection process. Furthermore, their genetic material is mixed by applying different genetic operators, i.e. mutation or recombination and offsprings are produced. Finally, according to selection for survival scheme, population is reorganized to include new offsprings, usually better than their parents. For more details, please refer to Back et al. (2000*a*), Back et al. (2000*b*), Mitchell (1996) or Michalewicz (1996).

The most common goal of calculation based on Artificial Neural Network is to 'teach' network to respond correctly for input stimulation. If one wish to design a model of this training, it has to be transformed into optimization problem, where:

1. Search space = weights of ANN
2. Dimension = number of weights to adjust + eventually other parameters
3. Constraints = limits for weights values
4. Fitness function = measure of an output accuracy (i.e. RMSE)

Branch of Evolutionary Computation provide tool which fits very well to above specification - Evolutionary Programming (EP). It is a real value optimization technique, which makes use of string of real values. Recombination is hardly ever used. Major genetic operator in this case is mutation (most common - Gaussian and Cauchy). Suggestions about

choosing mutation type have been described by Yao et al. (1999) and these guidelines were followed in this work. In short, gaussian mutation allows detailed walk through search space, thus is better for narrow extremum. Disadvantage is the exploration speed of fitness landscape. Cauchy mutation is much faster in exploration of fitness landscape because of its larger steps, thus it is good choice for wide extremum. Yao et al. (1999) also suggests, that good results may be achieved by using mix of cauchy and gaussian mutation, even though it takes two times more function evaluations.

At first, decision had to be made how to represent Neural network in a way, that is familiar with Evolutionary Programming. Ideally, this would be one dimensional string of parameters which one has to optimize. This results in another problem, how to calculate output of the network? ANN scheme presented on figure 2.3, described by matrix 2.4 allows it to be reasonably simple using following algorithm:

1. Set input $x_1...x_{in}$ as $y_1...y_{in}$
2. For $c = (in + 1) : n$

$$y_c = f(\sum_{i=1}^{c-1} w_{i,c} y_i)$$

This procedure is the most important for the success of a project. First, it is recursive what might create a bottle neck while running algorithm. Value for each neuron can be calculated if outputs of previous neurons connected to it have already been calculated. Output for the last neuron $c = n$, is equal to output of the whole network. Second, this allows unlimited evolution of the network. None of the layers in this network are artificially created or predefined before training. They are individually controlled by evolution process. Method for output calculation is not dependant on number of layers. What cause increase in complexity is only number of nodes.

2.2.1 Evolved structure

After running EP, ANN is represented as a string of real values. How to calculate its output has been presented in previous section. But the question is: how does an evolved structure look like? It depends on the string. Each zero value indicates that there is no connection. One has to investigate which neuron does it correspond to by transforming string into connectivity matrix and if all of the neurons are not connected, this may be considered that they belong to one layer. However, as results show, situation that there is 0 at any position in chromosome is very rare. Therefore, it is very common, that final structure of ANN is similar to one presented on figure 2.7. Source of differences between individuals were values of weights.

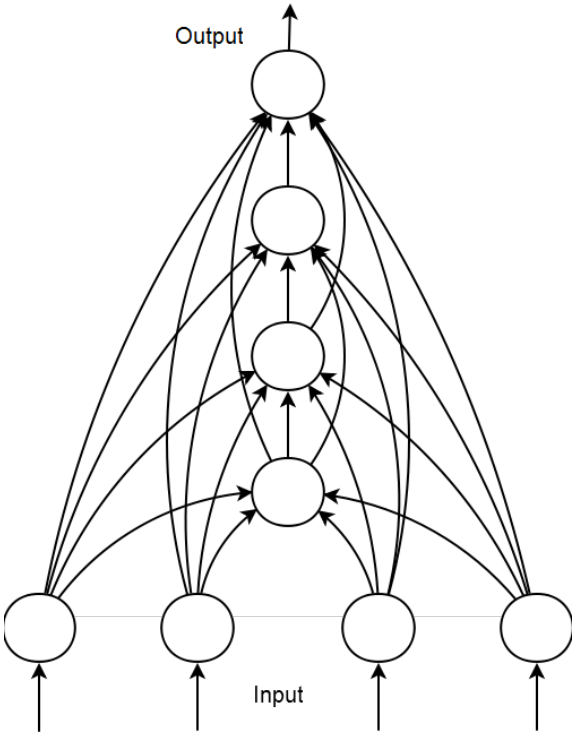


Figure 2.7: Evolved architecture of Neural Network

2.3 Integrating algorithms into one system

Performance of Neural Network is measured by RMSE (2.5) which is a square root of MSE. Significant is the fact, that it consists of two measures of Neural Network: bias and variance as presented on equation:

$$MSE = bias + variance \tag{2.6}$$

Bias describes the degree of fitting NN to the given training data. Variance on the other hand, describes the generalization ability of NN, thus whether or not the neural network fits the examples without regarding specificities of provided data. Those two measure have to be balanced to achieve best possible generalization error on data not previously known.

To rise to a challenge, population of Neural Networks is required - Ensemble of Neural Networks. This solution is based on the fact, that the error of ensemble output is always smaller or equal to a mean of individual networks output.

By applying EP algorithm, which is population based, this requirement has been met. Instead of choosing only best individual, all of them are selected. Output of each is included in final output according to weight assigned to each member of population. In this project, weights of each individual are equally important. Practically speaking, mean value of all ANN's outputs is calculated in order to produce final output.

2.4 Implementation - integrating modules

After brief introduction of theory standing behind the project, this section will bring implementation issues closer as they will be discussed in more detail.

Programming language chosen for the purpose of this project has been Matlab[1]. Following list provides motivation of this decision

- it provides quick and easy matrix manipulation techniques
- is highly optimized for mathematical computations
- increase speed of algorithms design
- provides very easy, but very advanced interface for different types of data manipulation
- provide very easy to use visualization techniques
- enable integration with procedures written in other high level languages

This programming language made process of design very efficient, thus more time has been spent on analysis of the system rather than on correcting faults, if other high level language had been chosen (i.e. C, C++, Java or C#). It requires specific approach to make full use of fast matrix calculation abilities provided by Matlab. This was not the problem, as description of all algorithms has already been provided in matrix notation.

Software developed for the purpose of a project works as follows (also visualized on figure 2.8:

1. Randomly generate population of ANNs (in a string form)
2. Fetch input and output data according to actual pointer
3. Run EP to train the network
4. Calculate RMSE for each individual
5. Combine population into Ensemble
6. Calculate RMSE of an Ensemble
7. Move pointer to the next day (re-use already trained network and fine tune it)
8. Return to 1

Point 7 from above list is an innovation of this project. It is based on the hypothesis, that

Hypothesis 2 *ANN does not have to be trained from the beginning, each time new data point becomes available*

Instead of using new population for each time step, every member of existing population is retrained as soon as new data become available to include it. This way, system base on the knowledge absorbed from the past, adding "new experience", each time new data point becomes available. What is the advantage of this approach? There are many:

- Huge time save, because there is no need to train network from the beginning

[1] www.mathworks.com

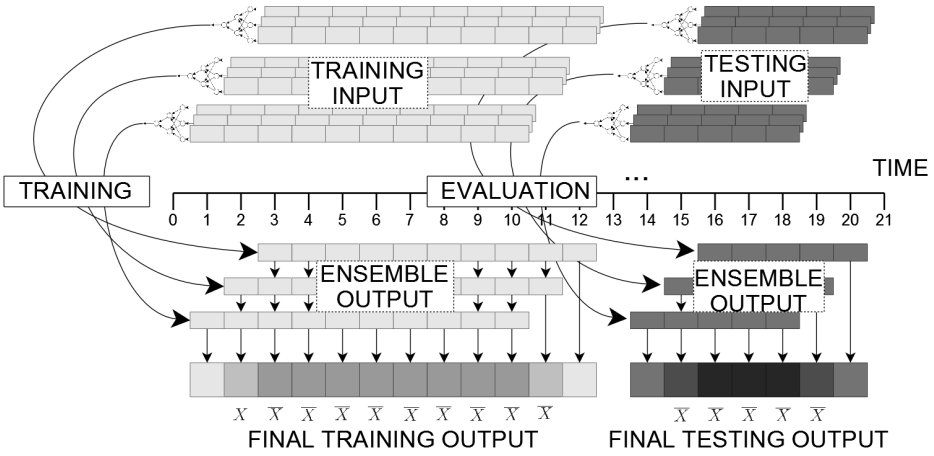


Figure 2.8: Structure of data fetching for subsequent time window

- long term memory - Neural Network can remember past data series, thus future patterns in data might be recognized easier
- this approach is similar to human approach: adding new knowledge to the past experience

Although it seems to be an ideal solution, one has to be aware, that by keeping one population of networks, super-individuals may be created. Therefore, it is a good practice to introduce few new individuals into the system each time training is performed, removing few old ones. This will keep diversity in population and will increase ability of avoiding local minima during learning process.

2.5 Encountered Difficulties

The most difficult obstacle to overcome was speed of calculation. This was partly caused by Matlab, which is the slowest programming language when compared to other high level languages, i.e. C. It is slower than commonly considered slow - Java. Problem was indeed serious. By training network on 10 subsequent time windows with population size $p = 25$, number of hidden neurons $h = 20$ for 1000 epochs, evaluation function calculating output of Neural Network has been called more than 1 million times. Time required to perform those calculations was approx 7 minutes on Intel Pentium Centrino 1.6 GHz. This was unacceptable, because time required to perform calculation for the whole 1000 time windows in training data took more than 12 hours. Source of the problem has been diagnosed using Matlab tool called Profiler which produce a report of time spent in each function of the program. It was found, that bootle neck which caused such a huge slow down of whole algorithm was calculation of fitness - netoutput function. Steps have been performed to optimize the function, but only small gain has been achieved. Therefore, decision has been done to rewrite netoutput in ANSI C. Matlab version of netoutput is presented on Listing 5.1. ANSI C version of the same function is presented on Listing

5.2. Matlab allows to use code written in different programming languages, thus it was possible to optimize this function by programming it in much faster language.

Result. There was amazing acceleration in calculation time. ANSI C version of netoutput was more than 30 times faster than its Matlab version. Both results are presented on figure 2.9.

Profile Summary

Generated 27-Aug-2007 15:54:48 using real time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
mainf	1	436.203 s	1.266 s	
gnn	10	422.375 s	10.797 s	
nnfit	10250	411.578 s	16.078 s	
netoutput	1055000	407.281 s	407.281 s	

(a) Matlab version

Profile Summary

Generated 27-Aug-2007 15:59:38 using real time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
mainf	1	44.890 s	1.185 s	
gnn	10	43.110 s	9.987 s	
nnfit	10250	33.123 s	21.019 s	
netoutput (MEX-function)	1055000	12.038 s	12.038 s	

(b) ANSI C version

Figure 2.9: Profiler report for Matlab functions performance

Together with speed-up of C version of netoutput function, whole algorithm is able to run faster. In this example, it is more than 10 times better. This allow much more experiments to be performed to better understand the structure of data and whole problem of prediction in general.

Chapter 3

Data manipulation

Performance of predictive engine depends largely on the data that are feed into the system. Therefore it is one of the most important part of the project to prepare data and adjust them in such a way, that significant patterns will be uncovered. Additionally, redefining goal of the system may significantly reduce complexity of prediction, i.e. if the goal set for an engine was to predict next day value of index WIG (current value is approx. 65 000) to within 10 points, the accuracy would be 0.01 % . If on the other side, the goal was redefined to be percentage change of the index value for the next day to within 0.5% - accuracy would be 5 out of 20. Therefore, it is crucial to adopt correct methodology which will guarantee a success.

3.1 Preprocessing Input

3.1.1 Collecting input data

There is a huge amount of different data available which could be used for the purpose of this project but not every data is appropriate and cannot be used because may simply be irrelevant.

Source of data used in the project is a website, financial database, called Stooq (2007). This is a source of i.e. historical share prices, economical indicators, exchange rates, options and many more. Based on past research in the field (i.e. Brabazon & O'Neil (2006)) and author's experience, following data has been selected for initial analysis:

- Forex currency pair GBP/PLN
- 5 day cumulative percentage change of GBP/PLNx
- Polish Stock Market (GPW) index value WIG
- Polish National Bank currency exchange rate for GBP/PLN
- GPW Macro Sector - Finance Companies
- GPW Macro Sector - Industry Companies
- GPW Macro Sector - Services Companies
- Index of GPW investors' mood - Wigometr

- Inflation Rate (month to month)
- Inflation Rate (year to year)
- Gross National Product (PKB)
- Unemployment Rate
- Interest Rate
- Balance of Current Turnover
- GBP London Interbank Offered Rate GBP (LIBOR) - 1 month
- PLN Warsaw Interbank Offered Rate PLN (WIBOR) - 1 month
- British Stock Market Index (FTSE 100)
- USA Stock Market Index (S&P 500)
- Oil Brent Price
- Cooper
- Gold

Above list could be extended by adding many other available indexes but this would significantly reduce speed of calculations, thus it may be considered as a future task, to work on larger data. The case is to decide what the output should be and investigate what factors influence it mostly. By performing this task, input data should be reduced down to approx. 5 different series which will be further preprocessed and feed into Neural Network. It is worth mentioning, that the goal is to form most influential set of data, simultaneously reducing number of variables because it will allow to produce less complex Neural Network. And less complexity results in faster system.

Finally, one has to remember, that input is very important. Poor input will result in poor output. In this case GIGO (Brabazon & O'Neil (2006)) applies (Garbage In - Garbage Out). Therefore, focus should be put towards careful preparation of this element of the system. One of the methods, used for input preparation, is dimension reduction technique called Principal Components Analysis (PCA) which will be described in following sections.

3.1.2 Dimension Reduction

The idea of this project is to collect as many data as possible, which may be significant to prediction. Ideally, one would then use collected set as an input to Neural Network. Output produced would be solely dependant on the ability to learn patterns in data. This approach would unfortunately require huge computing power and is simply waste of time, because some of the data are redundant. Therefore, to optimize this process, Principal Components Analysis (PCA) has been employed as one of excellent linear dimension reduction technique. This method is well known for ability to explain data in terms of principal components. It searches for relationship within data set and form principal components from those with largest variance. Finally, cumulative energy content for each eigenvector provides overview of importance each principal component has. Online dictionary (*Online Encyclopedia - Wikipedia* (2007)) defines PCA as:

"PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on (...). PCA can be used for dimensionality reduction in a data set by retaining those characteristics of the data set that contribute most to its variance, by keeping lower-order principal components and ignoring

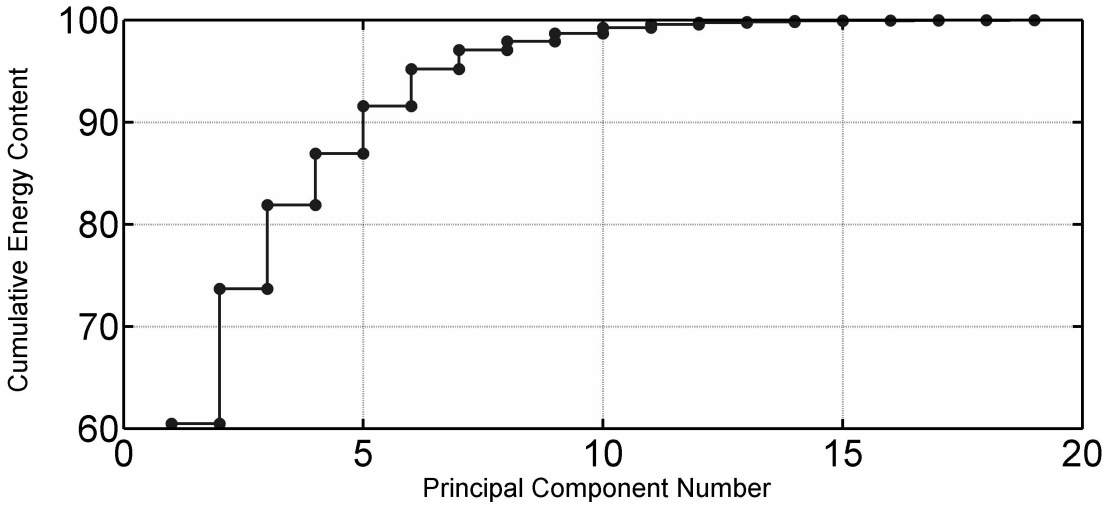


Figure 3.1: Energy Content of each Principal Component

higher-order ones. Such low-order components often contain the "most important" aspects of the data (...). PCA has the distinction of being the optimal linear transformation for keeping the subspace that has largest variance."

There are some disadvantages of this method, i.e. it uncovers only linear dependencies in data. Thus if there are any nonlinearities, which may have significant impact on the prediction, this unfortunately would be omitted by this method and sometimes might even be lost.

Whole of the data has been represented in terms of principal components. Each of them has specified significance, called Energy Content of each Principal Component. Figure 3.1 shows Energy Content for data used in this project.

As expected, more than 99% of energy is distributed through 10 components with more than 60% in first Principal Component. If one were to accept those 10 components, this would reduce size of data of approx 50%.

One has to remember, that this is not huge dimension reduction. Some experiments noted decrease of more than 90% of initial data or even more. But this depends on the structure of data and does not apply for data used in this project. One more conclusion drawn from these results is, that many of data prepared to be an input, are not linearly correlated, thus it is an ANN task to choose which of them are more significant than the other. Final note regarding this project: for further analysis, threshold has been set at $th = 90\%$ on cumulative energy content, to include components up to th value. This resulted in ANN input size equal to $ins = 4$.

3.2 Postprocessing Output

Methodology regarding input, adopted in this project, is very straightforward: collect many data series, reduce their dimension by choosing only those with significant features and use with Neural Network. But problem arises when one has to decide about output,

which should be produced by ANN. It significantly increase (or decrease) performance of the whole system. There are many issues to consider. First of them is scaling: range based or mean and standard deviation based.

3.2.1 Scaling output

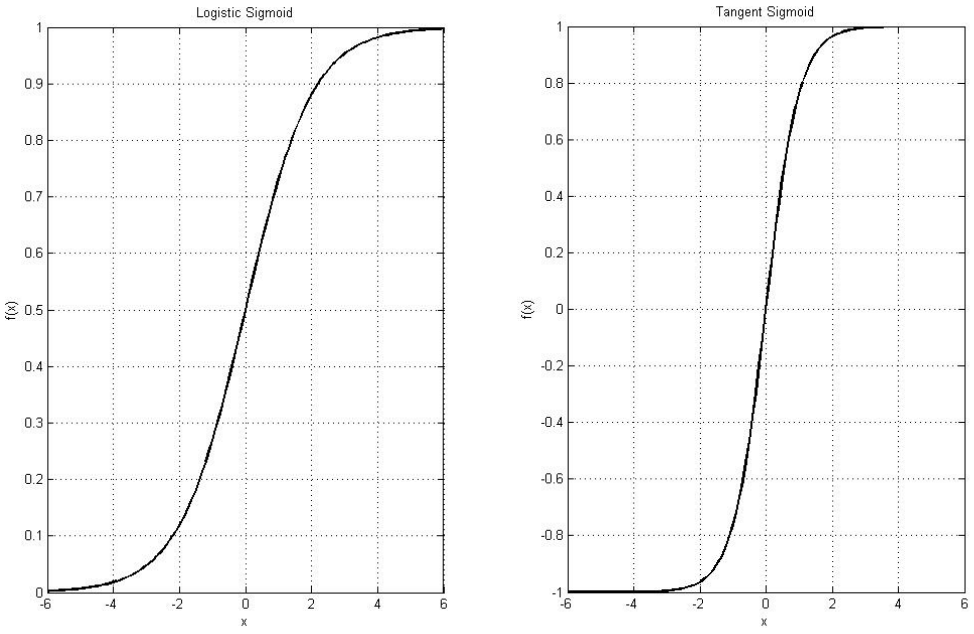
Range based scaling. Shifts data to fit between specified values. To obtain values in range $(0; 1)$ equation 3.1 should be used:

$$y_t = \frac{x_t - min_n}{x_{max} - x_{min}} \quad y \in (0; 1) \tag{3.1}$$

or if range $(-1; 1)$ is requirement, equation 3.2 should be used:

$$y_t = 2\frac{x_t - min_n}{max_n - min_n} - 1 \quad y \in (-1; 1) \tag{3.2}$$

Output range depends strictly on output neuron's activation function applied. If logistic sigmoid has been used (3.2(a), equation 3.1 would be desired, because it scales data into $(0; 1)$ range. On the other side, using tangent sigmoid (3.2(b)) as an activation function for output neuron, would require equation 3.2 to be applied to selected output data.



(a) Logistic Sigmoid

(b) Tangent Sigmoid

Mean and standard deviation based. Scales data by subtracting mean value of the set n and dividing it by its standard deviation. The aim is to produce sample for which

mean value is 0 and standard deviation is 1.

$$y_t = \frac{x_t - mean_n}{\sigma_n} \tag{3.3}$$

Selecting appropriate scaling method significantly influences performance of Neural Network, i.e. scaling desired output to be in range $(0; 1)$ while applying tangent sigmoid transfer function, half of the output range is unused. This situation introduces artificial error, which should be avoided.

For this work, tangent sigmoid transfer function has been chosen, thus all possible output has been scaled to be in range $(-1; 1)$.

Two main data manipulation techniques can be performed: preprocessing the input and postprocessing the output and these will be described as follows.

3.3 Output selection

As mentioned previously, deciding about an output may significantly simplify task of the system. There were many suggestions to what should be chosen. Table 3.1 collects those ideas into one place and provide properties, advantages and disadvantages of each possibility. Furthermore, those indicators are plotted on figure 3.3 to provide general overview of those indicators. Value plotted is Forex pair between GBP/PLN (Great Britain Pound / Polish Zloty) for a period between 31-01-2003 and 11-04-2007 which resulted in 1059 samples, after excluding days free of work.

After brief examination of figure 3.3, two indicators will not be very useful for further analysis: ROC and DROC because their behavior is very similar to original price which is probably not very good and easy target to approximate, but this will be examined further.

3.3.1 Analysis

Interesting pattern may be observed for DPC and CDPC on figure 3.3. First of them, DPC is similar to white noise, whereas for CDPC it is possible to extract some pattern, which is hidden behind a noise. Intuitively, this signal may be periodic, thus hypothesis has been developed:

Hypothesis 3 *DPC and CDPC indicators are periodic*

To investigate periodicity in given signal Discrete Fourier Transform (DFT) will be used ((Lyons (2001)). DFT - translates signal into its frequency domain. By plotting Frequency Spectrum it is possible to extract information about main frequencies in the signal. As an example, figure 3.3 has been presented. On figure 3.3(a) sample signal $y = sin(2\pi 50t) + sin(2\pi 120t)$ has been modified with noise thus it is difficult to extract find periodicity within it by looking at the plot in time domain. After applying Fourier Transform, signal has been transformed into frequency domain. Base frequencies used in

Table 3.1: Original price and possible output indicators

Symbol	Indicator	Formula	Advantages	Disadvantages
DPC	Day-to-Day Percentage Change	$DPC = \frac{x_{k+1}}{x_k} - 1$	Presents information about price change from one point to another thus provides overview about dynamics of the market during following days	prices are only with respect to previous point thus nonlinear data transformation is performed
CDPC	Cumulative DPC calculated over a period t	$CDPC = \sum_{k=1}^{t} DPC_k$	provides information about future changes and what will their result be. By summing Day-to-Day percentage changes of asset through a period of time, this indicator takes size of price movement into account	Percentage changes are only with respect to preceding values thus are not normalized as a ratio to i.e. first value.
ROC	Rate of Change of asset measured after a period k	$ROC = \frac{x_k}{x_1} - 1$	provides investor with an indication of profitability of investment after a period of time	does not include price change through desired period t thus reduce information in data
CROC	Cumulative ROC for each value k over a period t	$CROC = \sum_{k=1}^{t} ROC_k$	It smoothes original data, is linear transformation of it	Does not introduce any transformation of data thus is not much different than original data

sample signal has been extracted and has been displayed as peaks at $50[Hz]$ and $120[Hz]$ on figure 3.3(b). Their height is at least 3 times more than the level of noise.

DFT has been used to test hypothesis 3 which states that the CDPC indicator is a periodic signal. Time frame used in this experiment has been slightly shortened to contain 1024 (power of 2) number of samples as required by Fast Fourier Transform (FFT) algorithm used in this case (Cormen et al. (2001)). Therefore, last redundant samples were excluded. This operation resulted in no loss of information which time series contains.

As a result of experiments, 10 different Frequency Spectra have been plotted on figure 3.4. Peaks are much more significant for large window. For small window, there is no base frequency detected by DFT. There are 3 main significant peaks at: 13 days, 20 days and 40 days. While increasing window size, peak at 40 days becomes less noticeable.

Based on above analysis, DFT provides very useful spectra for larger windows, confirming hypothesis 3. If data are periodic indeed, it will be easier for Neural Network to

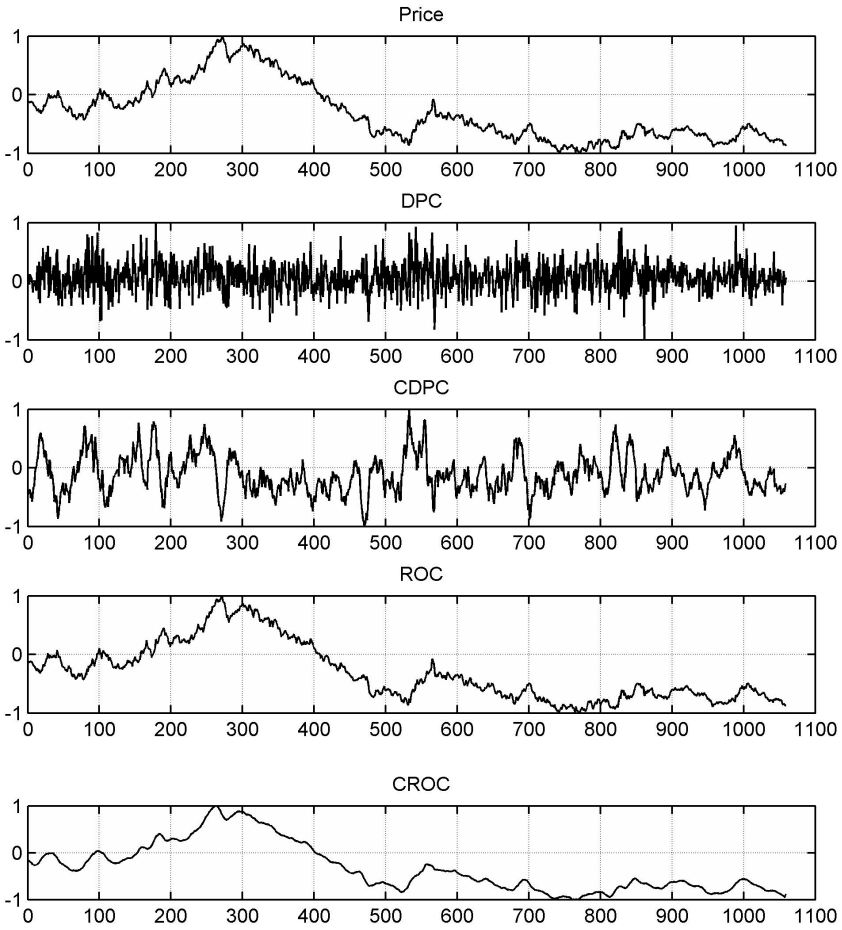
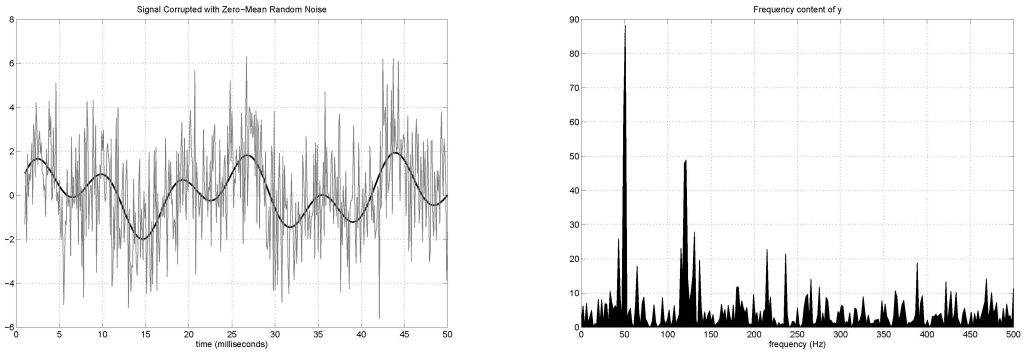


Figure 3.2: Original price and output indicators scaled to range $(-1; 1)$



(a) Signal corrupted with zero-mean random noise

(b) Frequency Spectrum

Figure 3.3: Sample Fourier Transform of signal: $y = sin(2\pi 50t) + sin(2\pi 120t)$

approximate this function.

To further analyze CDPC, figure 3.6 has been prepared. When window size is increased, signal becomes less noisy. It is very important to notice, that at the time when CDPC crosses value 0 from - to +, price is at its local bottom. When CDPC crosses value 0 from

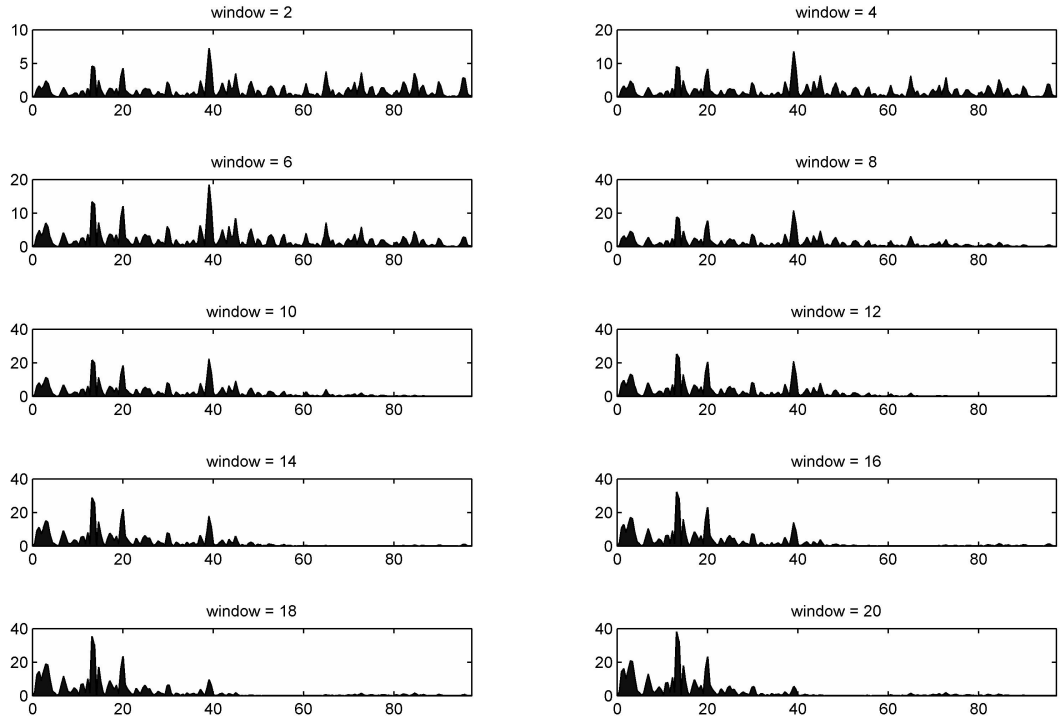


Figure 3.4: Frequency Spectra for CDPC with different window size

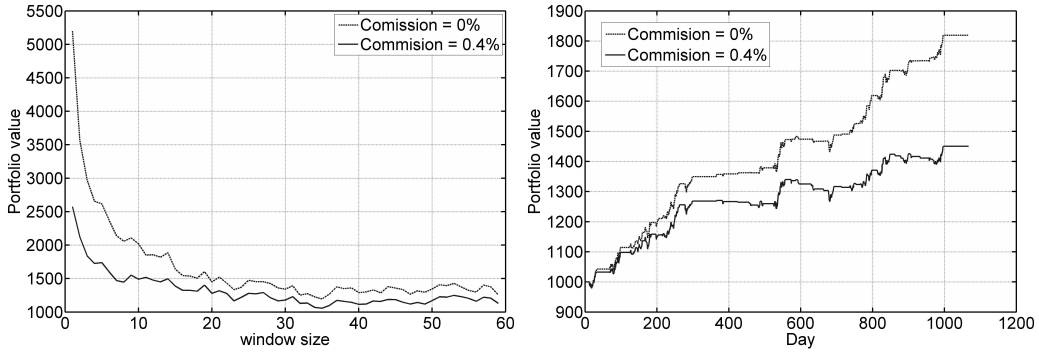
+ to -, price is at its local top. This property could be well used as a trading rule.

Trading Rule 1 *CDPC as an indicator*

- *Buy signal - when CDPC changes sign from - to +*
- *Sell signal - when CDPC changes sign from + to -*
- *To adjust its effectiveness, window can be modified.*

Trading Rules performance To confirm effectiveness of trading rules, virtual transactions have been simulated on portfolio of initial value Ł1000. Results for trading rule 1 have been presented on figure 3.3.1. There are two situations presented. One is when trading has been performed in ideal conditions. On the other side, more real conditions have been introduced - trading including example commission. In this case, rule for adding transaction cost was as follows: if the commission is smaller than some threshold t, than threshold is subtracted from transaction value. On the other hand when it is larger that t, commission value is subtracted from price. This way most of brokerage houses operate.

Comments on hypothesis 3. Although CDPC has been shown to be a good candidate for trading rule, there are some issues regarding this indicator. First of all, to prepare plot 3.6, future values of price have been chosen. In other words, for a price at time $t = 1$, values $t = 1...w$ have been used (where w is a windows size). As a result, in order to find



(a) Portfolio value for different window size

(b) Portfolio value during trading period $window = 14\,days$

Figure 3.5: Trading rule 1 performance

a value of CDPC at time t, ANN would have to predict w steps ahead from $t - w$ which is the last determined point of a data set. Additionally, to achieve best correlation with price, windows size should be selected based on the results from Fourier Transform. As an example, using Frequency Spectrum at figure 3.6, where peaks are detected at $t = 14$, $t = 20$ or $t = 40$.

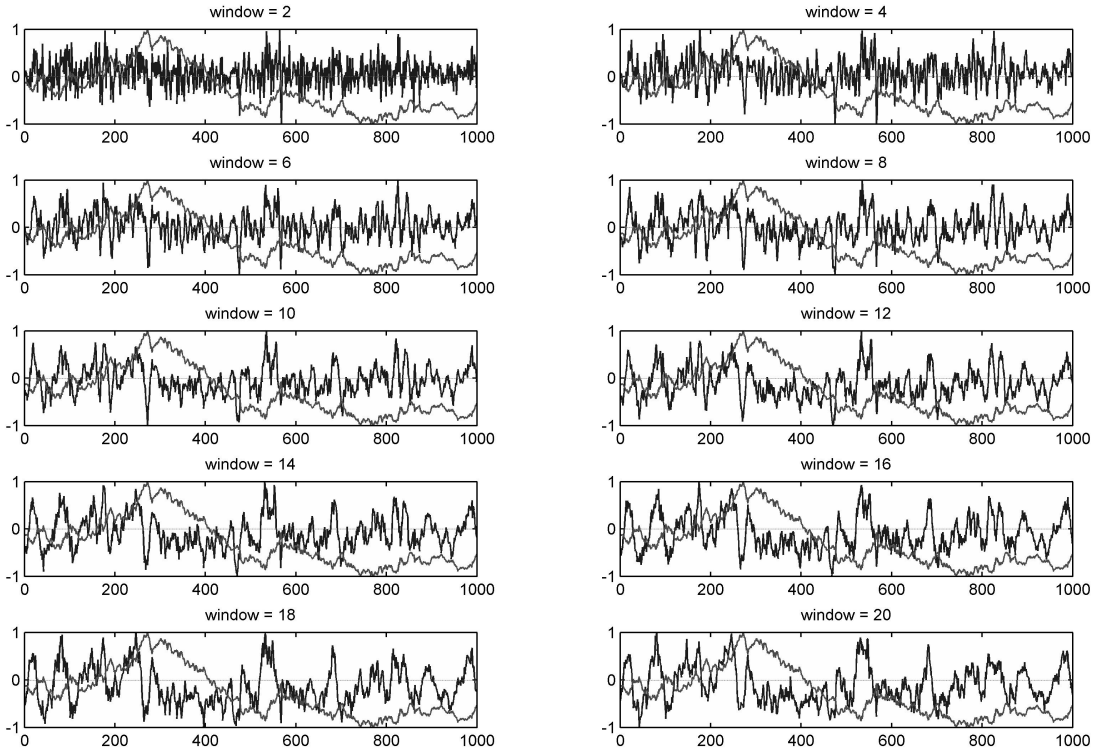
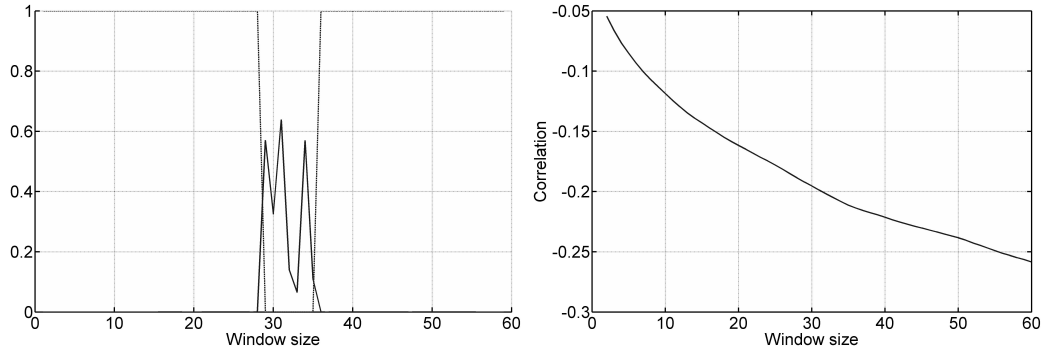


Figure 3.6: CDPC with different window size value, plotted over price, standardized between $(-1; 1)$

Additionally, T-Test has been performed for CDPC based on the window size. Results are presented graphically on figure 3.7(a).



(a) T-Test according to window size

(b) Correlation between price and CDPC according to window size

Figure 3.7: Basic statistics of CDPC according to price with different window size

For window size between $(29; 34)$ results from T-Test say that null hypothesis cannot be rejected. In this case, null hypothesis tested was that two matched (or paired) samples in the vectors x and y come from distributions with equal means. The difference x-y is assumed to come from a normal distribution with unknown variance.

Question can be asked about usability of T-Test for this data. Data, do not have to come from distribution with equal means to influence each other. Moreover, if in this case data are similar, they may be not very useful to use with neural network.

Therefore, correlation test has also been performed between price and CDPC with varying window size. Results are presented on figure 3.7(b). Provided figure also suggest that correlation is low in this case, but rises together with window size. This does not mean that data do not influence each other at all. Correlation detects only linear dependency, thus further steps are required to investigate influence of each set. In this paper, the author(s) develops a layered architecture for a MRS composed by heterogeneous mobile robots is proposed.

3.4 Conclusions

This chapter was to present data preparation process.

Data and results provided in this chapter were based on the past. To be able to successfully trade on the financial markets it is important to work on current data, thus system should be able to provide information about future. Therefore, next chapter will describe how data introduced in this chapter will be manipulated to produce future predictions.

Chapter 4

System evaluation

Previous chapters were presented to introduce theory standing behind this project as well as provide an overview of possible scenarios regarding system evaluation. Chapter 2 introduced detailed description of overall concepts regarding proposed solutions. Artificial Neural Networks and Evolutionary Programming algorithms have been presented, as they are core elements. Additionally, implementation issues have been described. In chapter 3 exhaustive introduction to input and output data has been described. This chapter is to provide results achieved after assembling each element into one wholeness.

4.1 Principles of system operation

Whole system operates in a recursive way. For each iteration, time period is selected. It is further divided into training set and evaluation set as presented on figure 2.8. The goal of each iteration is to "evolve" ANN to achieve minimum evaluation error (RMSE). This is partly done by aim to train ANN to obtain minimum training error. But as results show, minimum training error often does not guarantee minimum evaluation error. There are minimum three different approaches for training:

- population of ANNs is initialized randomly for each iteration
- population of ANNs is not initialized randomly for each iteration. Instead, already trained population for previous iteration is being used instead of randomly initialized. Training is performed on this set.
- first half of population (best individuals) are used from previous iteration and second half of population is initialized randomly. Further training is performed.

An idea of using already trained population of network in subsequent iteration is based on following hypothesis:

Hypothesis 4 *For each neighboring time window (iteration), architecture and parameters of ANN differ a little*

Conclusion from hypothesis 4 is very straightforward: for each subsequent iteration, it might be enough to use already trained population of ANNs and fine tune them instead of training from the beginning. This has been tested further in described experiments.

4.2 Day-to-Day Cumulative Percentage Change

Performance of the system based solely on the past data, using system described in section 3.3.1 presented very promising results (profit of approx 50% after 3 years). This has been tested using proposed system. Results have been visualized on figure 4.1

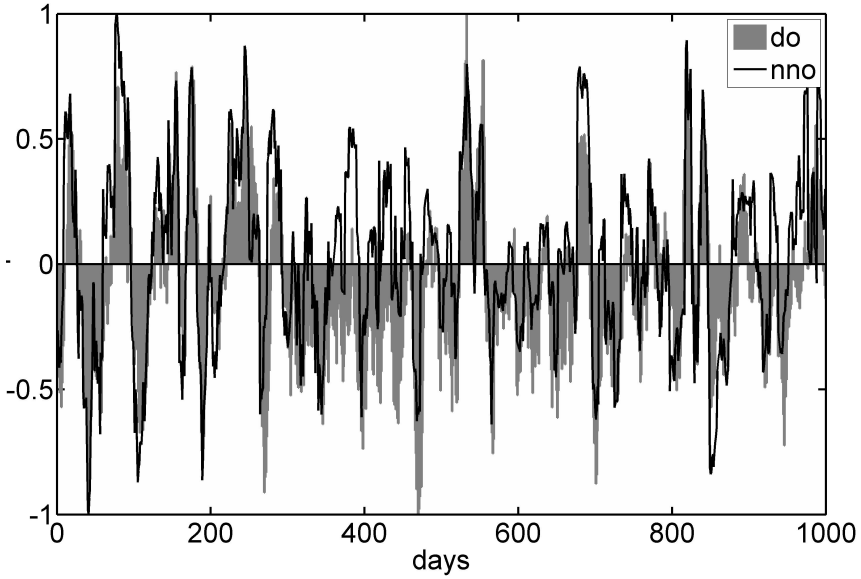


Figure 4.1: Prediction of CDPC using developed system

Based on trading rule 1, prediction obtained for CDPC (fig. 4.1) has been used, to compare its performance to ideal conditions on past prices. This is presented on figure 4.2

As presented, system would not cause investor to loose money, if commission would not be charged for each transaction. Including more realistic condition, where usual commission is equal to 0.4%, system caused loses of 20% of initial capital equal p=1000.

4.3 Raw Price Output

At first, prediction of raw price has been performed. Following parameters have been used:

- data type: Forex Exchange Rate GBP/PLN
- period: 21/01/03 - 12/01/07
- maximum epochs: 1000
- minimum gradient: 10^{-6}
- desired RMSE: 0.02
- number of hidden Neurons: 5

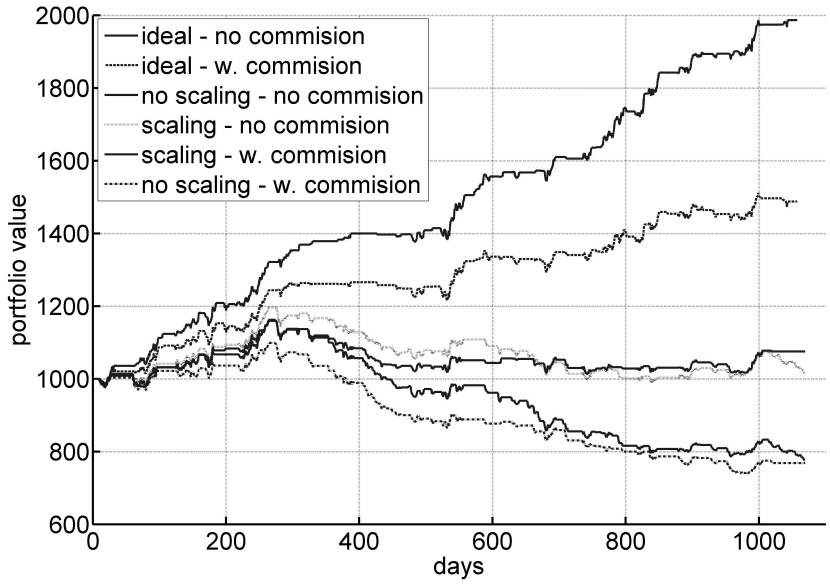


Figure 4.2: System performance based on prediction of the system

There is also a parameter of how many days ahead one would like to obtain a prediction for. This is strictly up to investors preference. The scope is unlimited, from 1 day prediction up to half of available data set size, which has to be divided into training and evaluation, thus if one would assign more data to training - it might happen that there is no data for evaluation left. However, the size of prediction is unlimited, one has to keep in mind, that:

Hypothesis 5 *Longer the prediction period, less accurate this prediction is.*

Unfortunately, due to to time limits of this project, I was unable to provide a proof of above hypothesis which has been developed through initial tests on data used in this project.

Additionally, when considering size of prediction period, one has to remember to include commissions for trading assets. These are the hidden costs, which are on top of actual gains or loses derived from correct investments decision. Therefore, if one decide to choose prediction period to be 1 day, gain of 0.5% is not very impressive, because to buy a share, very often one would be charged approx 0.4% - the same for selling it, which would result in loss equal to -0.3%. Assuming that trading system predicted price rise correctly, the loss is significant. That is the reason, why performance has been tested for 50 trading days ahead (excluding holidays and weekends, this is almost 3 months ahead) and 100 days ahead (approx half of the year).

In next subsections, experiments will be conducted in order to uncover how system performance is influenced by different factors.

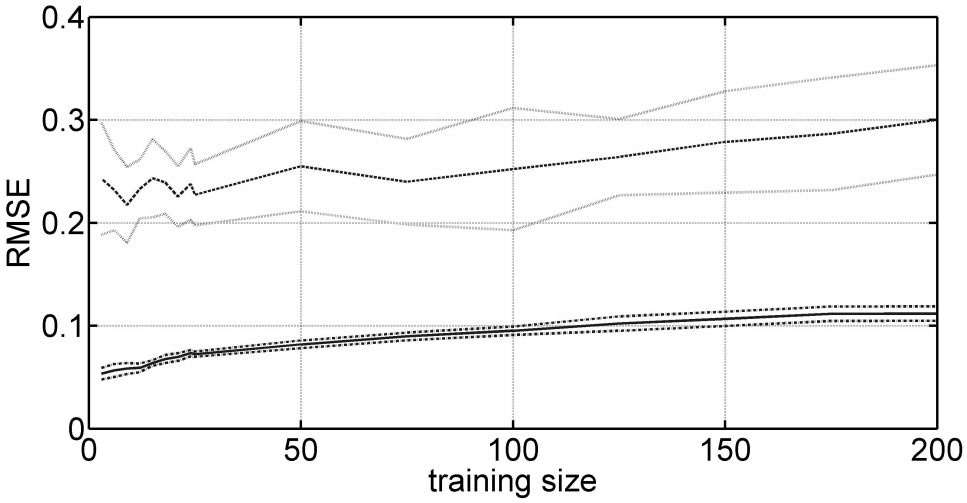


Figure 4.3: RMSE dependance on training size

4.3.1 Size of training

First of them, is the influence of training size for RMSE. It depends on how many training data has been used. As figure 4.3 presents, for low training size, there are fluctuations in RMSE, thus it is not easy to say which value to choose for further experiments.

It cannot be to small, because will be to much dependent on data which are actually being processed, if it is to large, memory of events from the past might be to long. And because changes on the market are very rapid, this might reduce performance. Confirmation of above statements can also be found on figure 4.3. Additionally, larger training size creates the requirement for longer computation time. In this work, great concern is about critical matters, therefore adopting to large training size might significantly slow down computations. Therefore, for the purpose of further experiments, training size has been chosen to be 80 days.

4.3.2 Population initialization for each time window

Three different strategies were adopted for the purpose of this project, regarding population initialization for each time window - random, half random or population parameters from previous iteration.

Each experiment has been performed 30 independent times to obtain statistically significant sample. Results have been presented in table 4.1.

Results for 100 days ahead have been visualized on figure 4.3.2. This figure confirms results presented in table 4.1. Figure 4.5(a) shows system performance for prediction 100 days ahead. As initially stated, 30 independent runs have been performed and standard deviation of mean has been plotted. It is clearly seen that standard deviation is high in this case. Introducing half of population to be randomly initialized did not lower standard deviation (4.5(b)) and according to table 4.1, it has increased. Significant reduction is visible on figure 4.5(c) and this setting resulted in lowest evaluation error. Another inter-

Table 4.1: System performance based on how population is initialized for each step, with varying steps ahead

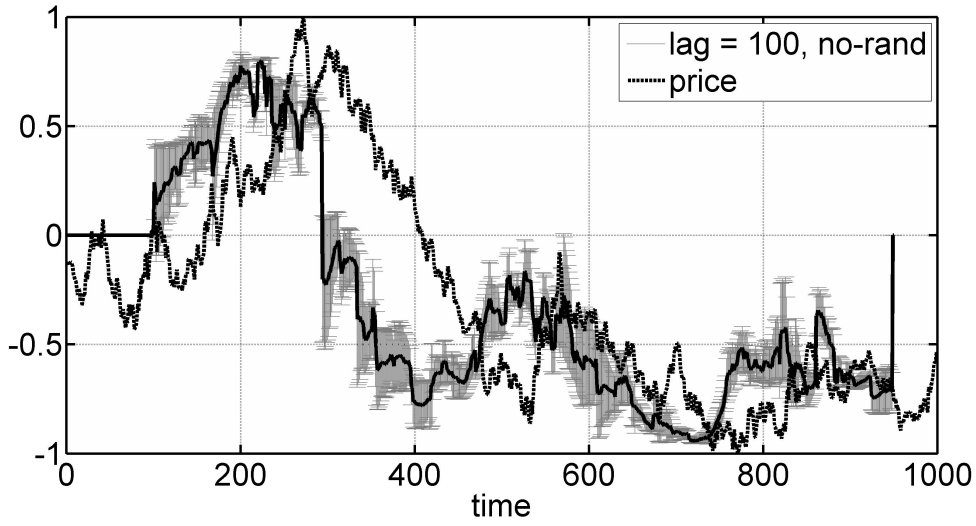
		No rand	Half rand	All rand
50 days ahead	rmse Mn(TR)	0.090	0.093	0.115
	rmse Std(TR)	0.002	0.002	0.002
	rmse Mn(EV)	0.233	0.243	0.211
	rmse Std(EV)	0.002	0.037	0.005
	FCalls Mn	76	98	498
	FCalls Std	117	175	278
	T-Test (% of 0)	13	30	70
	Corr Coeff	0.89	0.89	0.91
100 days ahead	rmse Mn(TR)	0.086	0.088	0.112
	rmse Std(TR)	0.004	0.004	0.002
	rmse Mn(EV)	0.224	0.236	0.200
	rmse Std(EV)	0.021	0.020	0.005
	FCalls Mn	52	53	447
	FCalls Std	26	29	200
	T-Test (% of 0)	40	33	76
	Corr Coeff	0.90	0.89	0.92

esting issue, presented in table is correlation coefficient and t-test variable. As correlation coefficient indicates, that for every instance of experiment, results are highly correlated with value to be predicted. T-test variable is a percentage of t-tests, which confirmed the null hypothesis. As results show, the larger number of randomly initialized population, the more null hypothesis could not be rejected. It may indicate, that those results with highes rate of null hypothesis rejection, are statistically more significant and relate in more detail to predicted value.

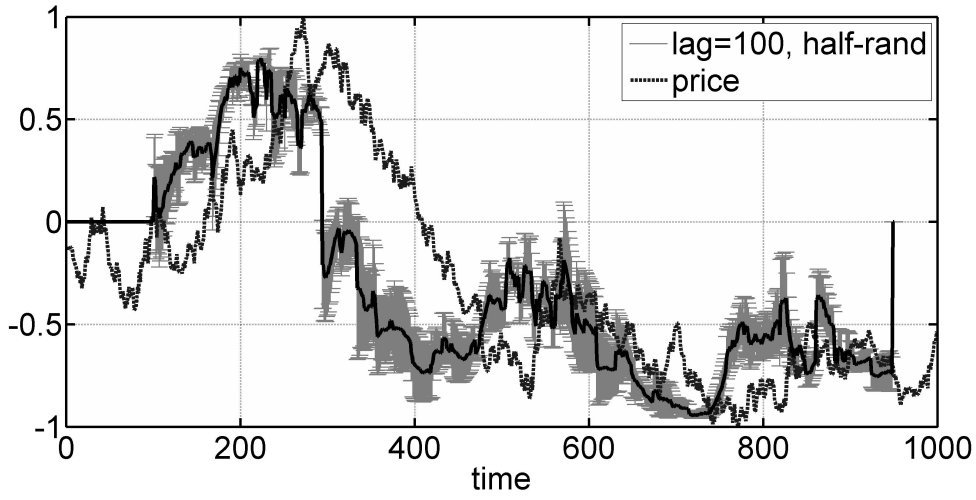
4.3.3 Time of training

It was a challenge to make experiments with the system because of its long running time. Especially, this was visible when larger training period has been considered. In this example, 100 data points were used for training. Results are presented on figure 4.3.2. It is clearly seen, that if members of population were not initialized randomly (or half of them were), average training time was approx 50 epochs (with low standard deviation), whereas for members initialized randomly for each sliding window, this time was approx 500 epochs (with high standard deviation). These results have been presented in table 4.1.

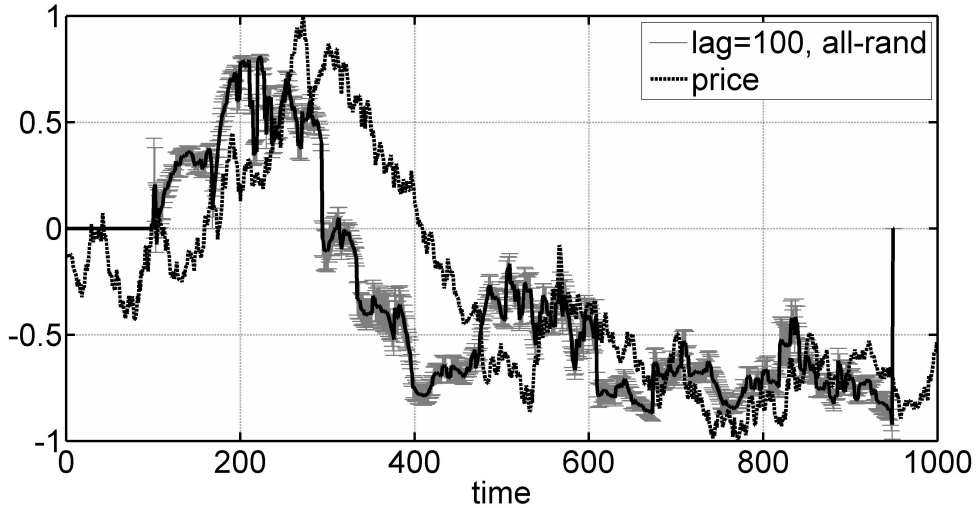
Biggest challenge was time required to train the system. If one was to use retraining rather than random initialization, average time required to train the system down to



(a) No randomly initialized members in population

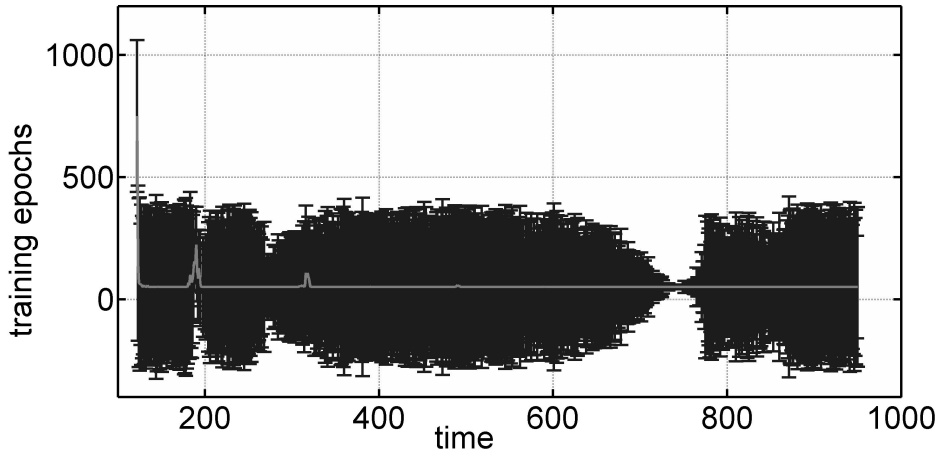


(b) Half of population members initialized randomly

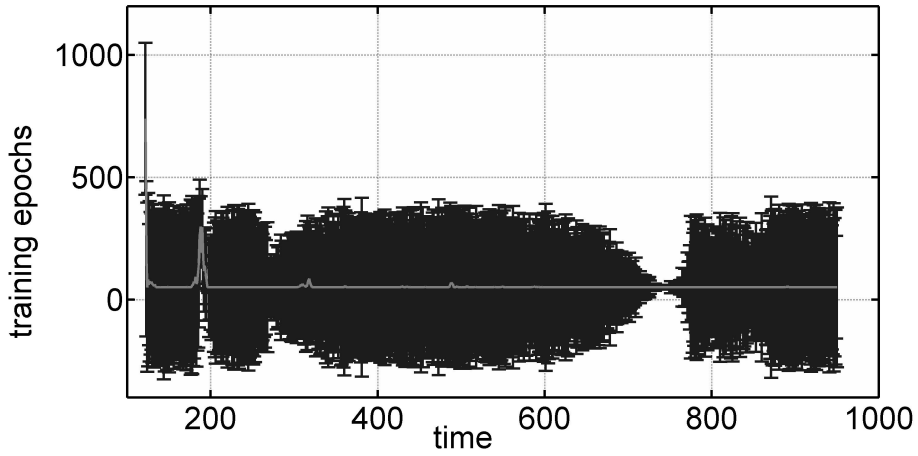


(c) Each population members initialized randomly

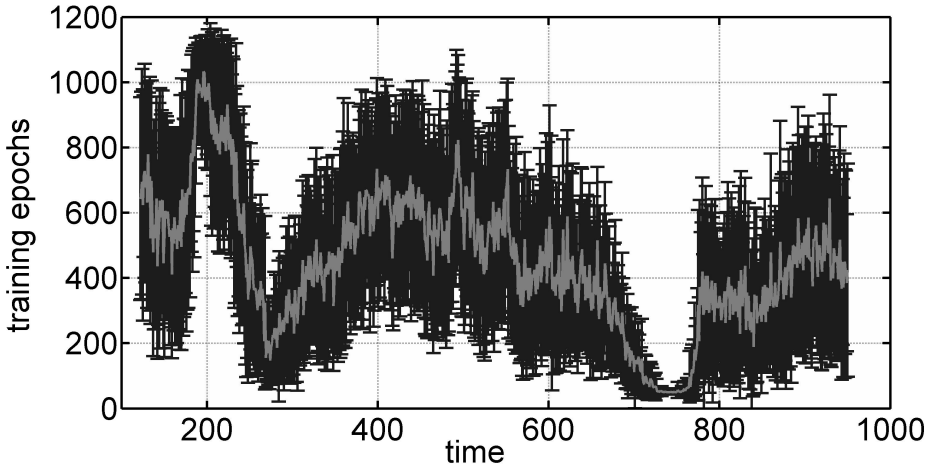
Figure 4.4: System Prediction according to population initialization in subsequent iteration (100 days ahead, evaluation)



(a) No randomly initialized members in population



(b) Half of population members initialized randomly



(c) Each population members initialized randomly

Figure 4.5: Training length according to population initialization

desired RMSE value was approx. 50 epochs. When using re-initialization, this time was approx. 10 times longer - approx 450 epochs. This, very much decrease overall performance of the whole system. And with time critical properties, it is significant difficulty which should be overcome.

Chapter 5

Conclusions and future works

29th of October 1929, 19th of October 1987, August 1998. As described at the beginning, during those days billions of dollars melted within hours on many stock markets, putting many individuals into bankruptcy.

Question asked at the beginning of this work was:

Was it possible to predict those turmoils?

The answer is:

No.

The answer for above question is a conclusion after conducting research in the field of Financial Time Series Prediction described in this work.

One would then ask a question:

Was it a waste of time, performing all these experiments?

The answer would again be:

No.

In this work, many examples have been provided, to support above answer. One main conclusion would be, that future is indeed based on present and past. Up to some extent. Because, as outlined in chapter 4, there is a noise present in the data and its level is very high thus prediction becomes even more difficult. Correctness of the model can be increased, by properly setting parameters (i.e. choosing right training period resulted in lower error). Thus, it is possible to achieve even better accuracy by performing test to understand which values are the most influential to overall output. However, one has to remember, not to go to far in trying to find ideal settings, because this might end up trying to fit model to specific properties of actual data. In this kind of work, generalization is much more desired than over fitting.

Methodology applied for the purpose of this project is unreliable. Although results achieved were satisfying, it was found that perfect predictions at financial markets are impossible using mathematical tools only. Why? The answer draws from foundation of stock market: trading is a psychological process. One would not have to look any further than events from past few days. To be more specific, "black Thursday" - 16th of August, 2007, when markets fell more than 5% during one session, i.e. FTSE surged 5.4 % from

its low and ended with its biggest daily rise since 2003 as announced by Guha & Callan (2007) in Weekend edition of Financial Times. While rise during next few days might be explained by FED cutting discount rate by 0.5%, it is difficult to find sensible reason for such a high fall. After conducting experiments described in this report, I am confident that:

Hypothesis 6 *Raw mathematical methods would not be able to predict market collapse*

There are many reason supporting above statement, i.e.:

- system has been trained using historical data, which did not contained such incidents, therefore had no example to relate to;
- fall was not caused by any rapid and significant event, i.e. September 11 2001, when markets fell a lot. This time, there was fear about mortgage market in USA but investors were informed about that situation during past weeks
- collapse of the market is often a result of psychological panic, which is very difficult to include as an input to neural network

Does it mean that designed system is completely useless? Obviously not. As presented on many examples, it is possible to approximate general trend, which data follow. Additionally, market crashes happen once a decade, therefore they can be simply ignored during training. Most important is the fact, that during periods when markets behave reasonably, proposed system predicts accurately.

This research provided tremendous amount of ideas and directions for future work which is a must to be performed in order to develop a trading system which might be used for profit making on stock market.

Sometimes, one would like to know, how to predict significant market falls. How this could be achieved? There are probably many routes to follow, but one seems to be very interesting. Research should be done towards analysis of people behavior, psychology of financial markets and global trading. It is often a game of psychology. There are big winners and big losers. The difference between them is experience in predicting. Based on this factor, trading system has been prepared. But there is another factor not included in the system - ability to understand behavior of crowd. As described by Elder (2001), people act completely different in a group and as individuals. To be more specific, what seems to be irrational for individual, one may forget while in a group, and act completely foolish. Many trading systems are based on the assumption that people act rationally. This shows deficiency of not only proposed method, but also most of others. Therefore, an opportunity arise for interdisciplinary work towards the task of constructing a model, which would simulate how individuals act. Afterwards, simulation of group reaction would possible give valuable information and is strongly desired. In my opinion, this is very much required to be able to make significant step forward in this field.

Finally, more research is required in order to improve accuracy of the system. There is still large scope of tests which were not performed, or were conducted briefly. There is a need to improve speed and reduce output error as just explained - it is still possible.

Acknowledgements

I would like to thank Dr John Bullinaria for supervising my project and not helping me solving problems but always pointing at the right direction instead, what allowed me to solve obstacles on my own, significantly improving my understanding of the subject.

I also wanted to thank Dr Jonathan Rowe for fantastic lecture in Nature Inspired Optimization which helped me to understand optimization in general, Ata Kaban for a lecture in Evolutionary Computation which introduced me to these interesting methods and partly inspired to do the work in this field. Finally, Peter Tino, for lecture in Neural Networks, which introduced me to the aspects of ANNs. Those lectures were also a source of many great ideas which I have implemented and described in this work.

Appendix: Statement of Information Search Strategy

Parameters of Literature

Forms of literature

Work has been conducted in recent area of research, despite of that there were some books available. Methods applied were described separately, in those books, but to find information on coupling ANN and EP articles have been searched for from within past 10 years. To understand topic, some web sites have also been used. Additionally, some older literature in the field of psychology of stock markets was investigated.

Geographical and linguistic coverage

Number of papers were examined from UK and USA research centers, however some European, Korean and Chinese were also considered, as some of them contained very useful content.

Most of the papers were written in English and very few were written in Polish.

Search tools

Citation Index

ISI Web of Knowledge, as described on the website, is an integrated Web-based platform which provides high-quality content and tools to access, analyze, and manage research information.

Google Scholar

Provides a search of scholarly literature across many disciplines and sources, including theses, books, abstracts and articles.

Scirus

Scirus is comprehensive science-specific search engine. Driven by the latest search engine technology, Scirus searches over 300 million science-specific Web pages.

Google

Google commercial search engine, was able to produce many basic tutorials which significantly helped in understanding the subject.

Search statements

Search statements were based on:

pars* AND unification AND grammar*

Initially following keywords were used: mass spectrometry, ovarian cancer, seldi cancer detection.

Selected source code

Listing 5.1: Matlab version of netoutput

```
function o = netoutput2( vw, in, n )
% n - number of hidden neurons
% vw - weights string
% in - input neurons values

sIn = max( size( in, 2 ) ); % number of input neurons
Y = in'; k = 1; j = 1;

for i = sIn : sIn + n - 1
    tmp = sum(vw( k : k + i -1 )'.*Y );
    k = k + i;

    tmp = exp( 2 * tmp );
    tmp = ( tmp - 1 )./ ( tmp + 1 );

    Y( i+1, 1 ) = tmp;
    j = j + 1;
end
o = Y( end, 1 );
```

Listing 5.2: ANSI C version of netoutput

```
#include "mex.h"
#include "math.h"

void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray
*prhs[] ) {

    double *vw, *in;
    double *y;
    double *out;
    double *n;
    double tmp;

    int sIn, sVw, i, m, z, sm;
    int k = 0;
    int j = 0;

    double yy[1000];

    vw = mxGetPr(prhs[0]);
    in = mxGetPr(prhs[1]);
    n =  mxGetPr(prhs[2]);

    sVw = mxGetN(prhs[0]);
    sIn = mxGetN(prhs[1]);

    sm = sIn + (int)n;

    y = yy;

    for (z = 0; z < sIn; z++)
    {
        y[z] = in[z];
    }

    plhs[0] = mxCreateDoubleMatrix(1,1,mxREAL);
    out = mxGetPr(plhs[0]);

    for (i = sIn; i < sIn + n[0]; i++ ){
    tmp = 0;
        for (m = 0; m < i; m++ ){
            tmp = tmp + vw[ m + k ] * y[ m ];
        }
        k = k+i;
```

```
            tmp = exp( 2 * tmp );
            tmp = ( tmp - 1 )/( tmp + 1 );

            y[i] = tmp;
            j = j + 1;
        }
        out[0] = y[i-1];
}
```

Bibliography

Back, T., Fogel, D. B. & Michalewicz, Z. (2000*a*), *Evolutionary Computation 1: Basic Algorithms and Operators*, Institute of Physics Publishing.

Back, T., Fogel, D. B. & Michalewicz, Z. (2000*b*), *Evolutionary Computation 2: Advanced Algorithms and Operators*, Institute of Physics Publishing.

Bemley, J. (2001), 'Neural networks and the xor problem', *Neural Networks. Proceedings. IJCNN '01. International Joint Conference* **1**, 540 – 543.

Bishop, C. M. (1995), *Neural Networks for Pattern Recognition*, Oxford University Press.

Boers, E., Kuiper, H., Happel, B. & Sprinkhuizen-Kuyper, I. (1993), 'Designing modular artificial neural networks', *Computing Science in the Netherlands (CSN)* pp. 87–96.

Brabazon, A. & O'Neil, M. (2006), *Biologically Inspired Algorithms for Financial Modelling*, Springer.

Brock, W., Lakonishok, J. & LeBaron, B. (1992), 'Simple technical trading rules and the stochastic properties of stock returns', *Journal of Finance* **5**(47).

Chan, L. K. C., Jegadeesh, N. & Lakonishok, J. (1996), 'Momentum strategies', *Journal of Finance* **5**(51).

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2001), *Introduction to Algorithms*, second edition edn, The MIT Press.

Dissanaike, G. (1997), 'Do stock market investors overreact', *Journal of Business Finance and Accounting* **24**(1).

Drew, G. A. (1941), *New Methods for Profit in the Stock Market*, Boston: Metcalf Press.

Elder, A. (2001), *Zawod Inwestor Gieldowy (trans. Proffesion: Stock Market Investor)*, second, polish edn, ABC.

Guha, K. & Callan, E. (2007), 'Fed brings relief to markets', *Financial Times* .

Haykin, S. (2005), *Neural Networks: A Comprehensive Foundation*, 2nd edition edn, Prentice Hall.

Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

Hu, Y. H. & Hwang, J.-N. (2002), *Handbook of Neural Network Signal Processing*, CRC Press.

Jackson, J. E. (1991), *A User's Guide To Principal Components*, Wiley.

Krose, B. & van der Smagt, P. (1996), *An Introduction to Neural Networks*, The University of Amsterdam.

Lyons, R. G. (2001), *Understanding Digital Signal Processing*, Prentice Hall PTR.

McNeils, P. D. (2005), *Neural Networks in Finance: Gaining Predictive Edge in the Market*, Elsevier.

Michalewicz, Z. (1996), *Genetic Algorithms + Data Structures = Evolution Program*, Springer.

Mitchell, M. (1996), *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*, MIT Press.

Murphy, J. J. (1999), *Technical Analysis of the Financial Markets*, New York Institute of Finance.

Online Encyclopedia - Wikipedia (2007).
URL: *www.wikipedia.org*

Slovic, P. (1972), 'Psychological study of human judgment: Implications for investment decision making', *The Journal of Finance* **27**(4).

Stooq (2007), 'Financial database', Website. `http://www.stooq.pl`.

Weigend, A. S. & Gershenfeld, N. A. (1993), *Time Series Prediction: Forecasting the Future and Understanding the Past: Proceedings of the NATO Advanced Research*, Proceedings Volume, Santa Fe Institute Studies in the Sciences of Complexity, Addison Wesley.

Yao, X. (1999), 'Evolving artificial neural networks', *Proceedings of the IEEE* **87**(9).

Yao, X., Liu, Y. & Lin, G. (1999), 'Evolutionary programming made faster', *IEEE Transactions on Evolutionary Computation* **3**(2), 82–102.