Warrant Pro 1: Market Price Synthesis with a Software Agent and a Neurosimulator

Patrick Bartels, Prof. Dr. Michael H. Breitner
Universität Hannover, Institut für Wirtschaftsinformatik
Königsworther Platz 1, 30167 Hannover
Email: bartels@iwi.uni-hannover.de and breitner@iwi.uni-hannover.de

Warrant Pro 1: Market Price Synthesis with a Software Agent and a Neurosimulator

Abstract: *Inherently today's derivative pricing is based on stochastic models developed since the 1970's. Especially the Cox/Ross/Rubenstein model is widely used. These models base on some unrealistic assumptions. Especially the necessary estimation of future volatility leads to imprecise prices. Derivatives are priced below or above the real market price. In both cases either the issuer or the customer disadvantaged. The imprecision is avoided by using software agents and high precision neural networks. The software WARRANT PRO 1 presented here combines the software agent PISA (Partially Intelligent Software Agent) and the neurosimulator FAUN (Fast Approximation with Universal Neural Networks) to synthesize market price functions instead of theoretical price functions. The architecture of WARRANT PRO I is described in detail. PISA automatically extracts data from the internet or other (semi-)structured text sources, e. g. videotext. Afterwards PISA automatically analyzes the resulting data, eliminates redundant and invalid values and creates neural network input files. High quality neural network training and validation patterns with predefinable denseness are generated. Using free sources like the internet the input patterns can be generated cost free for any available data. The neurosimulator FAUN learns true market price functions from the input patterns and generates a neural network. The neural network computes real market prices. It enables customers to single out overpriced and underpriced options a priori (extrapolation) and a posteriori (interpolation). On the other hand issuers are enabled to price over-the-counter-options (OTC-options) just-in-time. Future versions of WARRANT PRO I will contain the possibility to export the market price function in a platform independent executable file. This enables the user to calculate accurate market prices on nearly every computer.*
This paper outlines an example with 53 German DAX call warrants. With statistical analysis the quality of the issuers' pricing mechanism is analyzed. Issuers and options with prices below and over the market price are singled out.

Keywords: *Derivatives, market prices, software agent, artificial neural networks, neurosimulator.*

1 Introduction

Risk management is essential in modern market economy. Financial markets enable companies and households to select an appropriate level of risk in their transactions by redistributing risks towards other agents who are willing and able to accept them. Markets for options, futures and other so-called derivative instruments – derivatives for short – have a particular status. Futures allow agents to hedge against upcoming risks. These contracts promise future delivery of a certain item at a certain strike price. Options allow agents to hedge against one-sided risks. Options give the right, but not the obligation, to buy (call option) or sell (put option) something at a predefined strike price at expiration (European style option) or at any time up to expiration (American style option). For details regarding derivatives, especially options, see [Hull02].

Money can be invested in stocks to benefit from capital gains. To spread the risk often index like portfolios are realized. To minimize the risk of loss standardized and over-the-counter (OTC) index options can be used. As many different options are available on the market there is the necessity for the investing companies to single out overpriced and underpriced options. On the other hand issuers have to price (OTC-)options in minutes today. In both cases real time market prices are required. Therefore, an important neural network application is the synthesis of option and other derivative market prices. Today's options pricing often bases on the so called Black/Scholes-model. Another important approach is the Cox/Ross/Rubinstein-model with various successors, which is also widely spread. For an overview on derivative pricing, see [Hull02], [Pris00] and [JoCv⁺01]. Here, we present an often more accurate approach to option pricing which is fully market oriented (instead of theoretic).

Common to both theoretic models is that the correct option price is investigated by an option-underlying portfolio with risk-free profit. The theoretically fair option price p_{BS} (Black/Scholes) depends on the underlying price *s*, the strike price *b*, the time to expiration *r*, the risk-free interest rate *ir* to expiration and the future volatility σ_s of the underlying price. The latter is estimated by the annualized standard deviation of percentage change in daily price. The Cox/Ross/Rubinstein-model depends on *s*, *b*, *r* and *ir*, too. The future volatility is represented by the likelihood of rising and falling of the underlying price *s* in time steps with appropriate length.

Both mentioned analytic pricing models base on two problematic assumptions: First the markets are assumed to be efficient so that a prediction of the direction of the market or an individual underlying is not possible. Second the future volatility σ_s of the underlying price is assumed to be accurately estimatable and is a priori known to seller and buyer of an option. As a result of estimating the volatility in both models, σ_s "varies" and often neither the Black/Scholes-model nor the Cox/Ross/Rubinstein-model capture option market prices accurately. In particular the very important option price sensitivities ("option Greeks") Γ, Θ, Δ, and Ω usually are inaccurate. These problems do not appear when instead of an analytical model market price functions are used. The generation of market prices depends on historical and actual prices of a set of appropriate options. For details of the option pricing process with neural networks see [Brei00] and [Brei03]. In contrast to the theoretical pricing models high accurate neural networks can learn true market price functions of options, warrants and other derivatives. Like the theoretical option price $p_{BS}(s, b, r, ir, \sigma_s)$ or $p_{CRR}(s, b, r, ir, \sigma_s)$ the market price $p_M(s, b, r, t)$ depends on the permanently available underlying price *s*, strike price *b* and time to expiration *r*. But instead of *ir* and the artificially estimated σ_s the time *t*, e. g. day and hour, is used as direct input for the pricing model.

During the last years business competition forced financial service providers to increase efficiency. Especially Banks invest in new information technologies to gain advantages in competition, see [LeWi02]. Usually this leads to automation and computerization as the margins are constantly getting smaller and computer work is much cheaper than substitutable human work. Therefore, research concentrates on the development of innovative computer programs and information systems that enable a necessary increase of efficiency. For an overview of information systems in finance see [Buhl99] and [BuKr⁺01]. This paper shows the first prototype of the WARRANT PRO 1 software. WARRANT PRO 1 synthesizes real market prices for options and warrants using a software agent and a neurosimulator. Changing the training variables also other market prices or values can be synthesized. This enables interpolation, extrapolation and forecast of many input-output-relations. WARRANT PRO 1 incorporates the web mining software agent PISA and the neurosimulator FAUN. Usually for neural network training approaches commercial databases are used. These are usually expensive. PISA automatically extracts option prices from the internet for free and generates input files for the neurosimulator FAUN. The input files are of high quality, i. e. outlier free, and have predefinable denseness. FAUN uses the resulting files as input data and learns true option market price functions. WARRANT PRO 1 combines PISA and FAUN to offer a framework to calculate highly accurate option market prices for free. Currently an advanced prototype is used to analyze feasibility. The prototype and a test for German DAX put options and warrants are presented and evaluated, here. The final version of WARRANT PRO 1 will be platform independent and efficient on common computers. A graphical user interface (GUI) offers a common interface to all implemented programs.

2 WARRANT PRO 1

Like presented above current analytical models of pricing warrants and options are inaccurate. Therefore, WARRANT PRO 1 is developed. Primary target is to develop a program that extracts financial data from the internet and uses them to train neural networks that calculate real market prices for warrants and options. Other applications are intended and developed. In difference to other approaches of pricing warrants or options with neural networks WARRANT PRO 1 uses data that are gathered from the internet instead of commercial databases. As the data are extracted free of charge WARRANT PRO 1 offers an inexpensive way to calculate real market prices for warrants and options.

WARRANT PRO 1 is a framework program that is still under development. It bases on two research programs of the Institut für Wirtschaftsinformatik of the University of Hanover. FAUN is a neurosimulator developed by the second author since 1996. FAUN supports supervised learning with different topologies using neural networks. FAUN features a fast and efficient training method. Other multivariate approximation approaches are inferior, here. Here, FAUN learns real market price functions for warrants and options. For more details see http://www.iwi.uni-hannover.de/faun.html. The FAUN training process needs high quality input data. PISA is a software agent that extracts the needed values from the internet automatically. Beside internet pages any other semistructured or structured text sources are supported, e. g. videotext or XML-feeds. It is developed by the first author since 2002. For latest information, see http://www.iwi.uni-hannover.de/pisa.html. Currently both programs are used separately as no common user interface is available. The current version is a prototype to analyze feasibility. When test approve that the idea is realizable and the results a superior a complete software development process will be initiated. In future versions WARRANT PRO 1 will offer a graphical user interface to access and control every underlying module. The currently used programs Maple and GnuPlot will be substituted by internal and more efficient programs. The final version of WARRANT PRO 1 will be available in summer 2005. The architecture of WARRANT PRO 1 is illustrated in Figure 1.

In the current development status the modules PISA and FAUN both work separately. Both programs work satisfactorily as shown in Section 5. Webpages are processed and gathered data are used for neural network training successfully. Generated networks calculate real market prices for options and warrants with high quality. WARRANT PRO 1 only uses components that are free of charge. Used programs, programming languages, runtime environments and extracted data are also free of charge. Using neural networks usually requires as much system resources as possible. To increase platform independency WARRANT PRO 1 supports all major operating systems: Microsoft Windows, Unix, Linux and MacOS.

3 Option Market Price Functions with the Neurosimulator FAUN

Neural networks are a very popular technique in financial market research for many years now, see [KiWe94] and [Buhl99]. Neural networks are information-processing systems inspired by the way the densely interconnected, parallel structure of the mammalian brain processes information. Neural networks are mathematical models that emulate some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning. That is why neural networks sometimes are called learning networks. Key element of the artificial neural network system is the novel structure of the information processing system. It is composed of an, eventually large, number of highly interconnected processing elements. These elements are analogous to neurons and tied together with weighted connections analogous to synapses. For further information about neural network's functionality see [Brei03].

Using neural networks mathematically means optimization, here. Input data are processed and mathematical functions like $f_{app}(x;p)$ are returned as a result. This function depends on a vector x that contains the input data and a vector p that contains the adjustable network parameters. The returned function approximates a correlation of the input variables. With this function either future values can be extrapolated or missing values can be interpolated or analyzed. To estimate the function's quality approximated and desired output data are compared. As reference either a dataset with validated data or a known reference function $f_{ref}(x)$ is used. The calculated error, e. g.

$$\varepsilon(f_{app}(x;p) \, ; f_{ref}(x)) = (f_{ref}(x) - f_{app}(x;p))^2,$$

should be as small as possible after the neural network training for all patterns. In order to minimize ε, the network tries other input/output-relations and compares the results again. If ε is less than before the new function is accepted as the currently best one. Otherwise the function is discarded and another one is tried. The described process is a kind of a learning process. For details regarding neurosimulators and neural networks, see [GöRo$^+$03] and [Stub01].

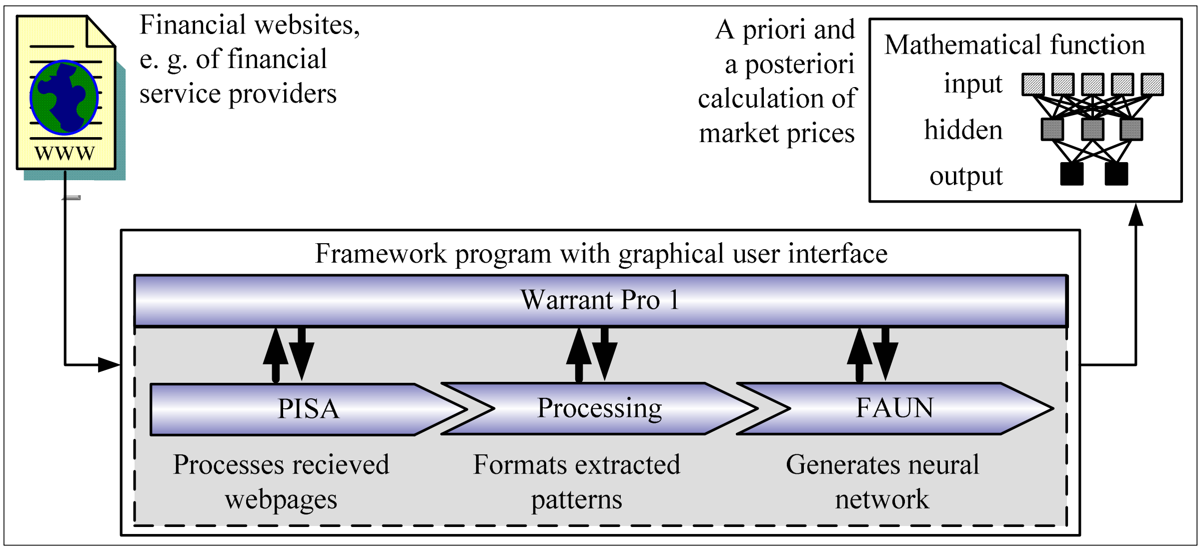

WWW
Financial websites, e. g. of financial service providers
A priori and a posteriori calculation of market prices
Mathematical function
input
hidden
output
Framework program with graphical user interface
Warrant Pro 1
PISA
Processing
FAUN
Processes recieved webpages
Formats extracted patterns
Generates neural network

Figure 1: Overview of the WARRANT PRO 1 program.

The described learning process is utilized for different kinds of applications. Among others recent financial market research concentrates on two application areas. Forecasting time series using neural networks has been discussed for many years now, e. g. price forecasts. The usual aim is to synthesize a mathematical function that predicts future values of the underlying time series. For the necessary training neural networks usually need large datasets, e. g. options prices, currency exchange rates or interest rates. The network generated mathematical functions usually are validated using their market prices as reference. The needed data usually are taken from commercial databases. Second application area is pricing derivatives. Today's analytical methods use several, partially estimated variables. The derivative prices of different issuers usually are not equal. In cases where highly accurate market prices are needed, neural networks can be used to synthesize market prices instead of analytical prices. New approaches are presented by [Brei03] and [Brei00]. The outcome of the network training process is a function that interpolates input prices. Changing the input variables market prices are generated for each variable combination without any estimation. As input data time price series of historical prices are needed. The series have to be as dense as possible and must not have large data gaps.

There are very many commercial and public domain (free- and shareware) neurosimulators, but the software FAUN 1.0 is standing out by special features. Based on sequential quadratic programming (SQP) FAUN efficiently trains three and four layer perceptrons with and without shortcuts and radial basis networks, too. A perceptron's error function and its gradient are computed with matrix algorithms implemented with the BLAS[1]. The SQP methods NPSOL[2] and NLSSOL[3] used are based on the BLAS, too. Optimized and fine-grained parallelized implementations of the BLAS exist for various hardware platforms. The coarse-grained FAUN 1.0 parallelization uses the PVM subroutines and runs on heterogeneous and decentralized networks interconnecting many general-purpose workstations and PCs and also high-performance computers. Most important features are

- the portability of FAUN 1.0 on LINUX, UNIX and WINDOWS;
- the easy to handle graphical user interface with detailed documentation;
- the comprehensive online graphics to control the program run and
- the offline graphics for a posteriori analyses and evaluation of the program run.

FAUN 1.0 synthesizes functions from high-dimensional input-/output-relations. The synthesis of functions is of importance e. g. for mathematical modelling and the calculation of optimal feedback controls for optimal control problems. Time series-analyses and -forecasts, are accomplish for many economical and technical applications, e. g. for interest- or exchange rate forecasts. Exemplarily option and warrant pricing with market prices is considered here, see Chapter 5. For more detailed information on FAUN see [Brei03].

For optimizing output data market price function generating applications need validated data as a reference. For functional correctness the input data have to be correct. Those data usually are bought in databases. Financial service providers buy the data directly from stock exchanges and resell them. Such databases usually are very expensive. An alternative approach is to get that information from the Internet. Possible sources of data are for example websites of banks, warrant issuers, or financial services companies. Also time delayed data can be used as neural networks are trained with historical series. Here, the needed pieces of information are automatically collected from webpages. To achieve the mentioned goals, the *partially intelligent software agent* PISA is developed. The agent enables extracting several stock, option or derivative prices efficiently. The agent loads specified webpages, extracts the demanded data, stores the data on computers and automatically generates the input file for neural network training.

4 Web Mining with the Software Agent PISA

4.1 Requirements

The web mining process is separated in four steps. These steps are illustrated in Figure 2. The illustrated subtasks require specific abilities of a web-mining agent in each step of the process.

[1] BLAS: Basic Linear Algebra Subprograms, see http://www.netlib.org/blas/.
[2] NPSOL: Nonlinear programming problem solver.
[3] NLSSOL: Nonlinear least squares problem solver.

These requirements lead to a specific design of the software agent PISA. For further details on software agents and their applications see [WoJe95] and [BrZa+98]. A detailed analysis of the requirements for the given tasks is published in [BaBr04] and [BaBr04b].

Receiving webpages: The software agent has to be able to crawl websites and to decide whether a webpage contains required values. For neural network training a continuous data flow is mandatory. The agent has to be permanently available during trading hours. In times with only few changes of the presented data processing frequency can be decreased.

Extraction: After the webpage is loaded it is processed and the needed data is extracted. A parsing mechanism is important for financial website extraction as it deals with quotes. These Quotes are usually numbers and an extraction using simple regular expressions is error-prone. The expressions might not match exactly the needed information. More complex patterns might match the exact information but might not be flexible enough if the websites structure changes. Therefore, the position of the quote containing tag is considered for the extraction. Dealing with HTML documents the structure is not always completely defined and can be irregular. HTML documents can contain errors; e. g. missing tags. The agent has to recognize and handle such problems to assure error tolerant HTML parsing. Once a webpage's source code is received and parsed regular patterns are advisable to identify and extract complex patterns from HTML source code. The Internet offers financial data from all over the world. Dependent on the target group of a website different international number, time and date formats are used. These have to be recognized and reformatted in user defined formats to increase flexibility for further processing programs. The formats need to be adjustable.

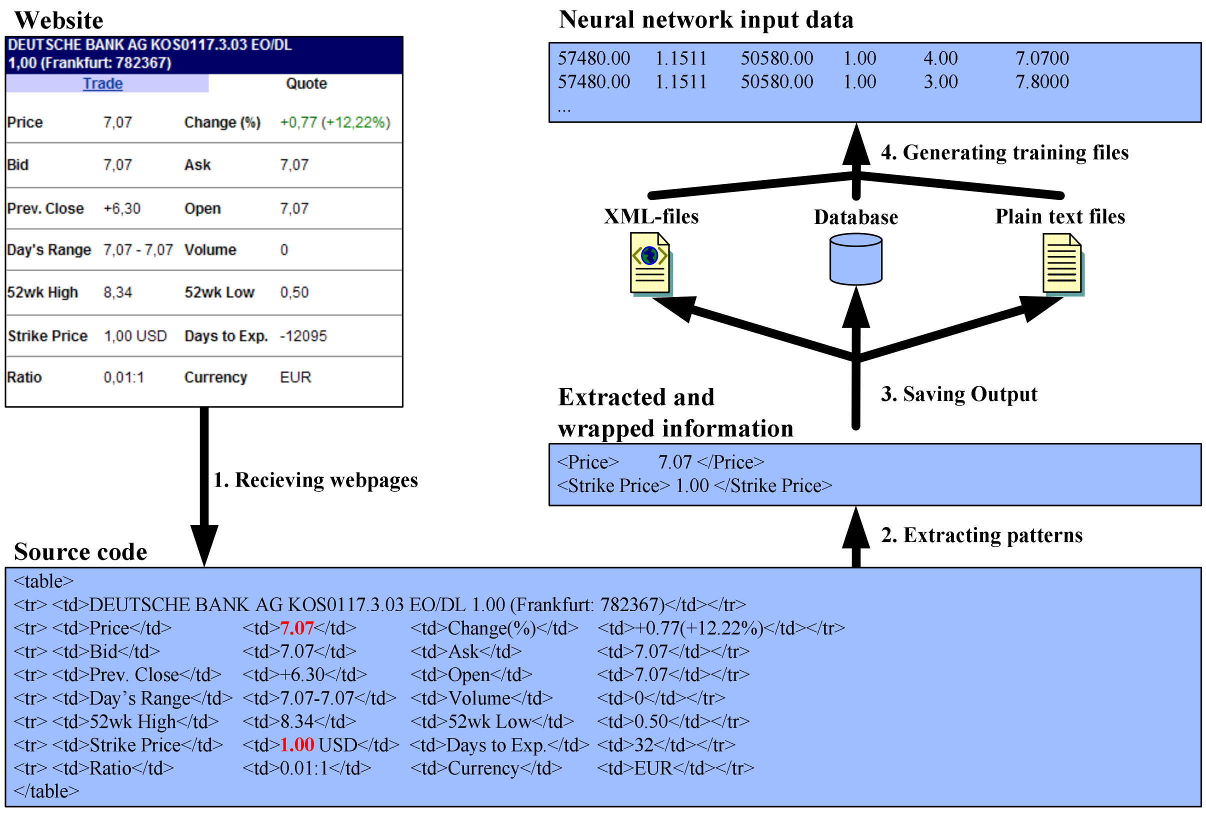
Data storage: File handling methods are mandatory to save extracted patterns. The user should be able to choose an output file format. Both plain text files and XML-files (XML = Extensible Markup Language) should be supported. Large amounts of data can be handled easier stored in a database instead of text files. The most common protocol for accessing databases is the ODBC-Protocol (Open database connectivity) which should be supported. The agent should be as platform independent as possible to assure flexible application. The hazard of breakdowns has to be minimized to assure a continuous data flow. Accurateness of the extracted data is very important. The agent has to provide rules with which extracted data can be checked on plausibility.

Post-Processing: Once rawdata are extracted they need to be processed. Different websites are merged and edited to create an input file for the neurosimulator FAUN. E. g. to synthesize real market price functions for the German stock index DAX the input function has to contain the current value of the option and the related price of the underlying index. Usually these information items are extracted from different websites. Exchange rate and options price are matched by their quote time. Other applications with neural networks need special technical indicators as input. The post-processing phase generates these values from given rawdata. The generated input file have particular requirements regarding formatting. FAUN requires one text file containing training data and another one containing validation data. Validation data are used to verify the trained neural network's results. Both files have to contain the input variables in one row and the related output variables in the next row. Only numbers are allowed as alphanumerical characters can not be processed. PISA has to create the neural network input files without any intermediate processing steps.

The listed requirements are achieved by some commercial and shareware software agents. For further information of their applicability see [BaBr04]. Usually web mining programs save extracted values to a text file with little possibilities to adjust data formatting and filter options. As programmers usually do not allow source code modifications formatting and filtering require an intermediate processing step. Furthermore, existing agents usually do not provide an application programming interface. This is necessary to control the agent by the framework program. The software agent has to be as platform independent as possible to assure platform independency of the whole Warrant Pro 1 framework. These are the major arguments to develop an own agent instead of using an existing one.

4.2 Design and Implementation

The mentioned requirements lead to special demands for the PISA's design. Major considerations are described in detail in [BaBr04] and [BaBr04b]. PISA is completely realized in Java. The major modules are shown in Figure 3. The component *PisaMain* initiates and starts all user defined extraction tasks. Specifications are taken from a configuration file. The initiated tasks run as independent threads and are executed simultaneously. Each task is represented by a single *PisaCrawler* object. These objects request the defined webpages and approve that during the crawling process no webpage is requested multiple times. Each received webpage is process by the *HtmlDocument* component that parses the passed website's source code. The source code is further processed by the *PisaGrabber* component which identifies and returns the user wanted patterns. The extracted data are saved either in a plain text file, XML-file or a database. Due to the dynamic nature of the web, most information extraction systems focus on specific extraction tasks. Here, we concentrate on agent based generation of dense datasets. The specific problems are focused here.



DEUTSCHE BANK AG KOS0117.3.03 EO/DL
1,00 (Frankfurt: 782367)

Trade		Quote	
Price	7,07	Change (%)	+0,77 (+12,22%)
Bid	7,07	Ask	7,07
Prev. Close	+6,30	Open	7,07
Day's Range	7,07 - 7,07	Volume	0
52wk High	8,34	52wk Low	0,50
Strike Price	1,00 USD	Days to Exp.	-12095
Ratio	0,01:1	Currency	EUR

```
57480.00   1.1511   50580.00   1.00   4.00   7.0700
57480.00   1.1511   50580.00   1.00   3.00   7.8000
...
```

```
<Price>       7.07 </Price>
<Strike Price> 1.00 </Strike Price>
```

```
<table>
<tr> <td>DEUTSCHE BANK AG KOS0117.3.03 EO/DL 1.00 (Frankfurt: 782367)</td></tr>
<tr> <td>Price</td>        <td>7.07</td>        <td>Change(%)</td>    <td>+0.77(+12.22%)</td></tr>
<tr> <td>Bid</td>          <td>7.07</td>        <td>Ask</td>          <td>7.07</td></tr>
<tr> <td>Prev. Close</td>  <td>+6.30</td>       <td>Open</td>         <td>7.07</td></tr>
<tr> <td>Day’s Range</td>  <td>7.07-7.07</td>   <td>Volume</td>       <td>0</td></tr>
<tr> <td>52wk High</td>    <td>8.34</td>        <td>52wk Low</td>     <td>0.50</td></tr>
<tr> <td>Strike Price</td> <td>1.00 USD</td>    <td>Days to Exp.</td> <td>32</td></tr>
<tr> <td>Ratio</td>        <td>0.01:1</td>      <td>Currency</td>     <td>EUR</td></tr>
</table>
```

Figure 2: Complete extraction process.

PisaMain: PISA starts with executing the main module *PisaMain* that initiates the extraction tasks. They are predefinable in an XML-file. The PisaMain-object starts one PisaCrawler-object per URL. These PisaCrawler-objects are started with a one second time delay, each. For dense datasets the request interval is very small. If too many pages are requested from the same webserver too frequently, the webserver might crash or it does not answer some requests. Latter results from a standard mechanism to avoid webserver overload by ignoring requests when a specified number of requests in a period is exceeded. This results in information gaps. The delay time is adjustable.

PisaCrawler: Each PisaCrawler object is executed in an adjustable interval. The interval is defined in seconds. Shorter intervals are possible but not reasonable since financial quotes are updated at most every second. The PisaCrawler component crawls websites either at every execution or just once at the first run. In this case PISA recognizes interesting webpages by user defined requirements and memorizes the URL. Future accesses use this address. Each requested webpage is represented by an HtmlDocument object. The wanted data are extracted from each object by a PisaGrabber object.

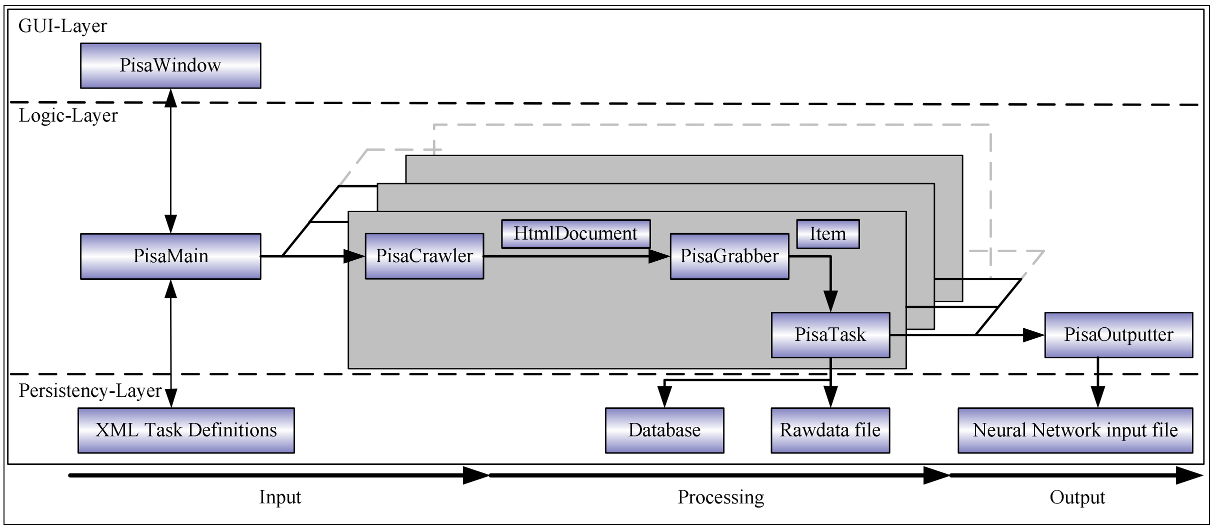
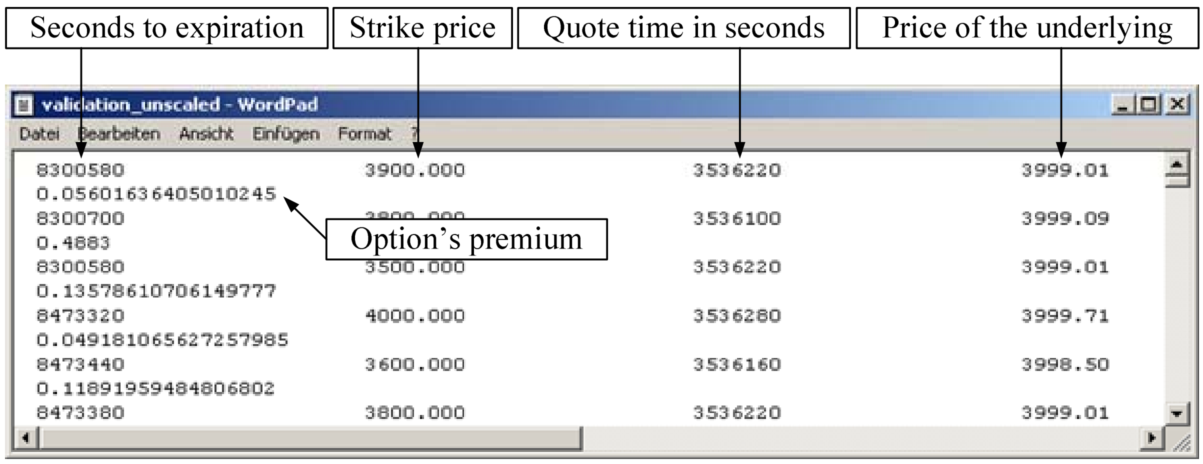


Figure 3: Major modules of PISA.

HtmlDocument: The evoking class passes an URL. The HtmlDocument component requests and analyzes the according webpages. PISA handles common syntactical errors reliably. The source code is converted into XHTML compliant text. Afterwards, all needed tags are identified using regular expressions for start- and end-position of a tag. Only needed tags are processed to decrease processing time. Once a tag is identified its attributes like size, colour and content are analyzed and stored in a tag-object. This tag-object represents the HTML tag. For each kind of tag an array is created that contains the objects of a kind in order of appearance. This enables a successive comparison of the array fields and the search for a special pattern.

PisaGrabber: The PisaGrabber module extracts the demanded patterns from a passed HtmlDocument-object. The location of a requested pattern is defined relative to an anchor pattern. Both patterns are specified by regular expressions. Using this approach the user has to know four things: 1. Pattern of the requested information; 2. Anchor pattern; 3. Number of tags between anchor-pattern and wanted information; 4. Kind of tag the patterns are formatted with. Optionally names for each extracted bit of information can be defined to store the values in XML-files and for calculation. To assure high data quality, accurateness of the extracted data are very important. Information items can be defined as mandatory. If an object is declared as mandatory and not available on a webpage the whole dataset is abolished. Extracting data from several webpages leads to the problem that the display format of numbers and dates do most likely differ from each other. PISA formats text, numbers and dates in adjustable formats.



```
8300580                      3900.000                    3536220                    3999.01
0.05601636405010245
8300700                      3800.000                    3536100                    3999.09
0.4883
8300580                      3500.000                    3536220                    3999.01
0.13578610706149777
8473320                      4000.000                    3536280                    3999.71
0.049181065627257985
8473440                      3600.000                    3536160                    3998.50
0.11891959484806802
8473380                      3800.000                    3536220                    3999.01
```

Figure 4: Exemplary neural network input file for FAUN.

As a consequence of the rapid growth of the internet it becomes more and more difficult to find, extract and organize information in the internet. The vision of the semantic web is developed to solve that problem. The semantic web is an enhancement of the existing internet that contains well defined data in a computer readable and processable meaning. Major idea is to annotate the information of the World Wide Web in a computer processable semantic. Thereby software agents are enabled to access the information in an easier way to convey them to the user. For more information see and http://www.semanticweb.org. PISA easily can be adjusted to extract values from webpages of the semantic web.

PisaOutputter: The PisaOutputter module collects the extracted items and matches them to related values if possible. Redundant values are discarded. The resulting patterns are automatically split into training and validation data. The ratio of training data to validation data and their allocation can be adjusted. The generated input file accomplishes all requirements given by FAUN and can be directly used for training. An example of a FAUN training file is shown in Figure 4.

5 Test Example: Calculating Option Market Prices for DAX Put Options

5.1 Introduction

Money can be invested in stocks to benefit from capital gains. Usually the investment period is predefined. To increase dependency usually portfolios are blended with stocks from different branches. To hedge the risk of stock investments options are used. Options usually have single stocks or indexes as underlying asset. Therefore, many companies use portfolios similar to major stock indexes. DAX (Deutscher Aktien Index) is the major German stock index.

For each hedging scenario usually options from many issuers are offered on the market. As described in Section 3 usually analytical functions or numeric simulations are used to calculate option prices. The used approximations are individual for each issuer. Therefore, prices of offered options differ as well. Companies need to single out overpriced and underpriced options. A reliable market price model is needed.

On the other hand, issuers need to calculate a price for (OTC-)options on-the-fly. (OTC-)options are e. g. asked by companies that want to hedge an individual problem for which no standard option matches. Issuers price (OTC-)options within short time, usually minutes. If the price is too small issuers might not find a hedge for the offsetting item. If the calculated price is too high the customer does not buy the option. Therefore, a reliable market model is needed by issuers as well.

Here, WARRANT PRO 1 is used to synthesize real options market price functions for German DAX options. To hedge stock investments put options are needed. A put option guarantees the sell of an underlying asset at a specified strike price at expiration. The following sections describe the extraction process of the needed data from the internet with PISA, the training process of FAUN using the extracted data and a validation of the test.

5.2 Extraction Process with PISA

Referring to the problem of option pricing we used the neurosimulator FAUN to synthesize real market price functions for German options. The needed variables are derived from the input function see Section 3. The influencing variables for each price/time combination are mainly: The derivative price, including bid- and ask-price, related date and time of the price and the time to expiration. The price related date and time of a quote are extracted because mostly published prices are no realtime prices. So the extraction date can not be used. The time to expiration is calculated by PISA as well as the options' premium. The options' premium is usually not published on the webpages. In case it is given it is usually time delayed, so that the premium does not match the published price on the same page. The following static values are extracted for each option: Security identification number, expiration date and strike price. Additionally the option's omega is extracted for further analyses. It is not used for training.

Generated values must not have any data gaps. Data gaps can be caused by poor quality of the offered data, poor webserver availability or networks failures. The source of most quotes is the EUWAX, Europe's leading derivative exchange, see http://www.euwax.de. Alternatively financial service providers like Onvista (http://www.onvista.de) also are a high quality source. Onvista is market leader in the segment of financial realtime quotes. Previous tests showed that both EUWAX and Onvista offer high and reliable data quality and webserver availability is very high, see [BaBr03]. Long term tests showed that within weeks both websites did not have any webserver breakdowns. As EUWAX offers all necessary values for the options within the webpage for an option except a usable price of the underlying asset the pattern are only extracted from one website. Here the EUWAX website is used as financial service providers receive their quotes directly from exchanges. The underlying German stock index DAX is calculated by the Deutsche Börse AG (http://www.deutsche-boerse.de) and published on many financial websites. Most of them are adequate to be used as resource here. Here, DAX values are extracted from the Deutsche Börse AG and matched to the option values automatically. The usual update interval for stock index webpages is about 30 seconds, see [BaBr04]. To assure availability of a DAX value for each extracted option values are additionally extracted from three different websites: Onvista, Comdirect (see http://www.comdirect.com) and CortalConsors (see http:// www.cortalconsors.com).

On February 17, 2005 EUWAX offered values for 53 different DAX call options with a strike price between 4000 and 4800 from the 12 major different issuers. Values for all 53 options are extracted for one week from Monday February 14, 2005 to Friday February 18, 2005. PISA automatically generates training and validation files for FAUN for each day. Without any intermediate processing steps each of the five day's data can be used for training. Data are stored in a rawdata file additionally for analyses. Note that using the pattern for Monday or

Friday might result in incorrect networks as the market behaviour is special shortly before and after weekends.

Pisa has been tested in detail in different environments and for different tasks in [BaBr04] and [BaBr04b]. Here, a common 2.4 GHz computer with 1 Gigabyte memory is used. Microsoft Windows XP is used as operating system. To reduce redundant extraction the extraction interval is set to 5 minutes for each option and 30 seconds for DAX values. Tests indicated that a higher extraction interval increases the extracted data volume with no proportional increase of the usable pattern. Most values are redundant and should not be used for network training. Note that this kind of redundancy is unintentional. Redundancy is eliminated to avoid an over proportional weight of a single option with too many pattern. On the other hand redundancy is deliberately generated during generation of the training patterns to raise the number of patterns for under represented options or issuers to equal weighting. The extracted number of values is adequate for the given task. Average processing time for each webpage is less than one second. Here, a more powerful computer like described in [BaBr04b] would not lead to any advantage.

Issuer (alias number)	**Security identification number (alias number)**	**Number of values**
Merrill Lynch (9)	A0CT2L (40)	22
	A0CT2M (41)	2
Unicredito Italiano (11	A0CVC6 (49)	52
	A0CVC7 (47)	91
	A0CVC8 (48)	91
BNP Paribas (0)	BNP13U (0)	48
	BNP13V (2)	47
	BNP13W (5)	48
	BNP5R0 (4)	48
	BNP5R1 (50)	47
	BNP5RY (1)	48
	BNP5RZ (3)	48
Vontobel (1)	BVT14H (6)	51
	BVT14K (7)	34
	BVT14M (8)	8
Commerzbank (3)	CB0AE3 (14)	18
	CB0AE4 (11)	97
	CB6D0A (15)	95
	CB6D0B (13)	69
	CB6D0C (12)	43
Citigroup (2)	CG6D0Q (9)	97
	CG6D0S (10)	48
	CG6D0U (51)	42
Deutsche Bank (4)	DB0C7B (17)	96
	DB0C7D (19)	45
	DB0C7F (20)	89
	DB1288 (16)	96
	DB1290 (18)	96
Dresdner Bank (5)	DR6EBS (25)	94
	DR6EBT (23)	95
	DR6EBU (24)	93
	DR6EBV (22)	91
	DR6EBW (21)	81
DZ-Bank (6)	DZ1A42 (26)	96
	DZ1A43 (27)	96
	DZ1A44 (28)	94
Goldman Sachs (7)	GS0BNC (30)	96
	GS0BND (34)	94
	GS0BNE (31)	96
	GS6DYP (32)	96
	GS6DYQ (33)	48
	GS6DYR (29)	56
	GS6DYS (52)	3
Sal. Oppenheim (10)	SAL5YE (45)	54
	SAL5YF (42)	54
	SAL5YG (43)	54
	SAL5YH (44)	94
	SAL5YJ (46)	57
HSBC Trinkaus & Burkhardt (8)	TB8PUF (36)	95
	TB8PUG (35)	95
	TB8PUH (38)	95
	TB8PUJ (37)	95
	TB8QDE (39)	85
Total number of values		**3623**
Total number of options		**53**
Total number of issuers		**12**

Table 1: Summarization of the number of extracted options values for February 17, 2005.

The overall extracted amount of data is 3 Megabytes per day. After eliminating redundant and incorrect patterns the net file size of used values is about 1 Megabyte per day. Major cause for redundant values is an extraction interval that is higher than the update interval of the extracted data. Future versions of PISA will adjust the extraction interval automatically to increase efficiency and decrease webserver load. The net file contains about 3600 different patterns. The number of patterns per day is consistent for each day of the week. Technically, all days can be used for intraday training. For neural network training values of February 17, 2005 are used. Table 1 illustrates the number of extracted values for this day. Two options, A0CT2M of Merrill Lynch and GS6DYS of Goldman Sachs, have significantly less extracted patterns that all other options. Checking the extraction log files showed that all webpages for these two options have been correctly processed for each extraction. The values have been extracted redundantly and only two and three were different. As the extracted webpages contained a nearly real-time quote time it is reasonable to say that the Euwax website published the values the way they were extracted. There are two major possible reasons. First, the website's data base contained more quotes but they were not published for any reason, e. g. technical problems. Second possibility is that the website's database did not contain more quotes because for these options prices no more prices existed. Nevertheless we used these values to avoid a loss of information although the influence of these two options on the market price function is small.

The neurosimulator FAUN only works with numbers as input values. Therefore the alphanumeric values for the security identification number and for the issuer are automatically replaces with consecutive numbers in order of appearance. This way identification number and issuer specific price differences can be trained. The alias number is also shown in Table 1.

The number of extracted values is not equal for each option and especially not for each issuer. Using training pattern with non equal numbers of pattern for an option or an issuer would result in an inadequate price function. The generated market price function would be over proportionally influenced by the options and/or issuers with an above-average number of values. Therefore the number of values is adjusted by copying the patterns of the under represented options until the number of pattern equals the maximum number of values. To avoid an over average influence of other factors like time of day the copied values are picked randomly. Instead values of the options with too many patterns could be deleted. This would result in a loss of information and is refused here. For the first training the number of pattern per security identification number is adjusted. In a second training run the number of pattern per issuer is adjusted. Table 2 shows the resulting numbers of values over-all and for training and validation using a ratio 20 % as validation data.

	Total numbers of values	**Numbers of values used as training patterns**	**Numbers of values used as validation patterns**
Original values extracted	**3623**	--	--
Normalized for options	4947	3954	993
Normalized for issuers	5378	4298	1080

Table 2: Overview of the number of patterns used as training and validation data.

5.3 Training of Neural Networks with FAUN

For numerical performance FAUN requires standardized input files. Therefore, FAUN uses a pre-processing program to scale training and validation files automatically. Input variables are scaled within an interval from -1 to 1. The output variable option's premium is scaled from -0.95 to 0.95. For all variables linear scaling is used. For further details of the scaling process, see [Brei03]. For the given task the neural network type three-layered perceptron with two neurons in the first hidden layer is used. Shortcut connections are activated. In addition to the described settings different other scenarios are tested but results are significantly inferior to the ones generated with the described settings. Available patterns are departed into training and validation data by a ratio of about 4:1 which means that 20 % of the data are used for validation. The anticipate differences in the data depending on the time of day validation data are taken out of the extracted data in five equally spread blocks. Each block contains 4 % of the validation data. Figure 5 illustrates the partitioning.

The approximation quality of the neural network can be estimated with the training and validation error functions εt (W) and εv (W). W is the matrix of the weights that constitute a neural network. As usual the perceptron is trained iteratively, i. e. εt is decreased by adaption of W, as long as εv < εt or εv ≈ εt holds (prevention of overtraining). Thousands of multi-layer perceptrons with various topologies and with different weight initializations are trained with a fast sequential quadratic programming (SQP) method.

FAUN trains neural networks until a specified number of adequate networks is generated. The quality of a generated neural network is defined, among other criteria, by the cross-validation quality. The cross-validation quality is the quotient of εv and εt. Here, the worst accepted cross-validation quality is 1.00. FAUN shows the cross-validation error of the trained networks during the program run in a separate window. Figure 6 shows the online error for the first training process. Each chart represents the error of a single neural network. FAUN optimizes each network by minimization of εt(W) until the overlearning limit is reached. The nearer a networks minimum εt(W) approaches zero the better a network is. FAUN stops automatically when a specified number of adequate networks have been found, here the limit is set to 100 adequate networks. Test indicated that a higher number of adequate networks does not lead to any enhancement of training or validation error. All settings for the two trainings processes are illustrated in Table 3.

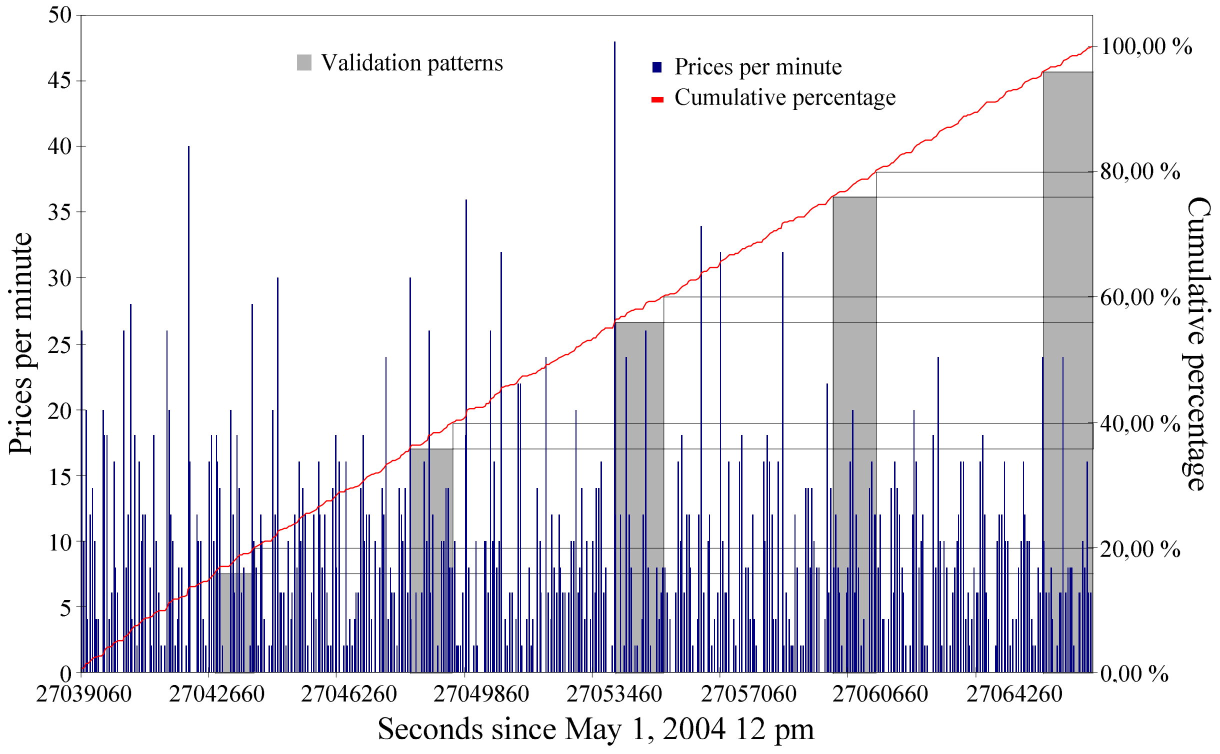


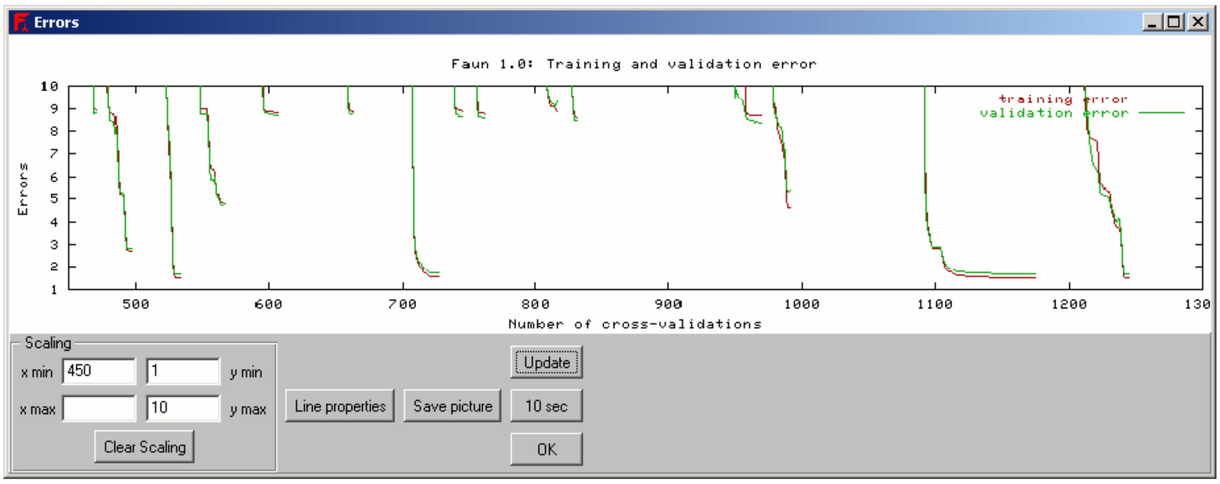
Figure 5: Partitioning of the extracted data in training and validation data.

The training process was conducted on the same computer the extraction process was conducted on, see Section 5.2. In a first training the security identification number's alias number is used as fifth input variable. In an additional training process the issuer's alias number is used instead. Each training process with FAUN lasts less than 10 minutes. Note that the training duration is highly dependent on the used hardware power.

	First training	Second training
Layers	3	3
Number of networks to be trained successfully	100	100
Number of parallely trained networks	1	1
Maximum number of trainingstops	2,000	3,000
Worst accepted cross-validation quality	1.00	1.00
Number of minimizing iterations without cross-validation	20	30
Number of cross-validation error comparisons after which weights will be saved	10,000	10,000
Shortcuts	used	used
Number of neurons in the input layer	5	5
Number of neurons in the first hidden layer	2	2
Number of neurons in the output layer	1	1
Input variables: extraction time, strike price, time to expiration, underlying price and …	numeric option identifier	numeric issuer identifier

Table 3: Overview of the number of patterns used as training and validation data.

The best neural network is exported in the Maple computer algebra system. Maple is an interactive symbolic mathematical system and a very sophisticated electronic calculator. Maple has the ability to algebraically manipulate unbounded integers, exact rational numbers, real numbers with arbitrary precision, symbolic formulae, polynomials, sets, lists, and equations. It can solve systems of equations and differentiate and integrate expressions. For further information, see http://www.maplesoft.com/products/maple/. Maple is used to evaluate the trained networks and to calculate market prices for any input value combinations. The results are presented in the next subchapter. Future versions of WARRANT PRO I will automatically generate Java sourcecode for the resulting neural networks. The java functions will be used to analyse the generated neural networks. They can be applied to any variation of the input variables to analyze the resulting changes in the market price. Using the resulting java function market prices can be calculated for non existing options. The source code is going to be a platform independent executable file. This means that the market price function can be used autonomously without the WARRANT PRO I framework.



x min 450
y min 1
x max
y max 10

Figure 6: FAUN online error window with four very good networks.

The results of the training process are illustrated with the program Gnuplot. Gnuplot is a portable command-line driven interactive data file (text or binary) and function plotting utility all major operating platforms. The software is copyrighted, but shareware. It was originally intended as graphical program which would allow scientists and students to visualize mathematical functions and data, for more information see http://www.gnuplot.info. Major reasons to use Gnuplot instead of other spreadsheet programs are the free distribution and the availability for all major operating systems. Furthermore, it can be integrated in the WARRANT PRO 1 program which is not possible with spreadsheet programs like Microsoft Excel.

5.4 Results

The extraction process with the software agent PISA works satisfactorily. The generated datasets are of high quality and no data gaps occur. The resulting output file is applicable for neural network training with FAUN. Here, a more powerful computer would not have led to any advantage, as average processing time is less than one second for each webpage. The given environment even offers the possibility to shorten extraction intervals and to process more webpages. Here, a shorter interval is not reasonable as the update interval of the underlying option is longer than the actual extraction interval.

Quality of a neural network is measured by the cross-validation error $\varepsilon_t(W)$, see Section 5.3. The nearer a networks minimum $\varepsilon_t(W)$ approaches zero the better a network is. Here, the best network produced by the first training session has an error of $\varepsilon_t(W)$= 0.0680357. The minimal error $\varepsilon_t(W)$ of the second training process is $\varepsilon_t(W)$=0.0464518. Both errors are considered as very good. As a result prices for non existing options can be calculated precisely.

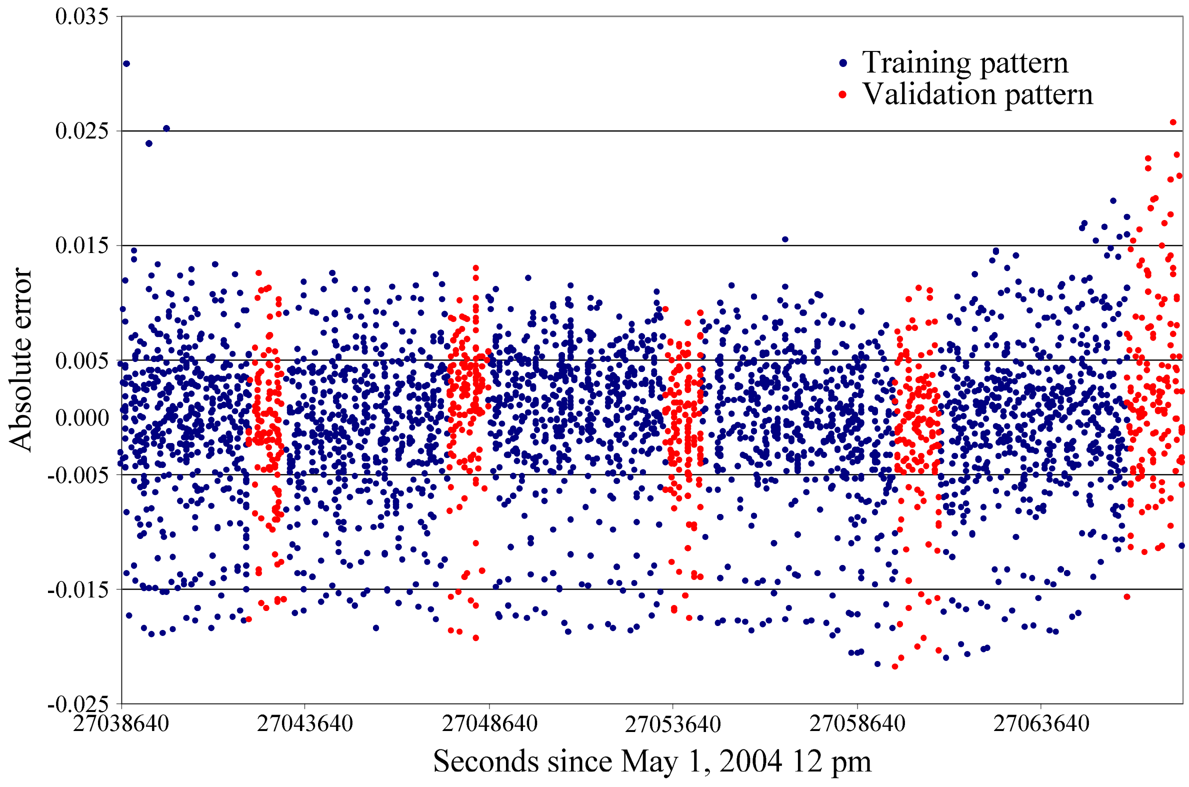


Figure 7: Absolute error of training and validation data for the first training (interval from -0.95 to 0.95).

Quality of a trained network is also measured by the absolute error. The absolute error is the difference of the extracted option's bid price and the calculated market price for the given input values of an option. If the extracted price is higher than the calculated market price this results in a positive error. The absolute error is calculated for both training and validation data. Under the assumption that the generated neural network is correct this means that options with a positive error are overpriced and options with a negative error are priced below the market price. Figure 7 shows the absolute errors for each training and validation pattern of the best neural network of the first training process. In the first training the input values are: extraction time, strike price, time to expiration, price of the underlying and the numeric option identifier. Figure 8 illustrates the absolute for the second training process where the numeric issuer identifier is used instead of the option identifier. The error for most values of both trainings is very small. Comparison of the error for training and validation data indicates that the validation data are adequately calculated. Note that identical training or validation patterns produce an identical error, so redundant values do not appear multiple times in the illustrations.

If noticeable and regular patterns in the chart appear the reasons for these patterns have to be analyzed. Usually those patterns have common attributes. They are usually based on specific combination of the input variables, e. g. a specific issuer with over-prised options or a specific option that is badly prised. Analyzing the training results neural networks can be used to single out over-prised and poorly-prised options. In Figure 7 most patterns' errors are very small. Only a few of the first patterns have an over-proportional high absolute error. All three patterns are from the option with the numeric option identifier 52. The according security identification number is GS6DYS and the issuer is Goldman Sachs. As only these three prices are higher than the market price the option is not generally over-priced. Further analyzes produced that the error is quite similar for all groups of input variables.

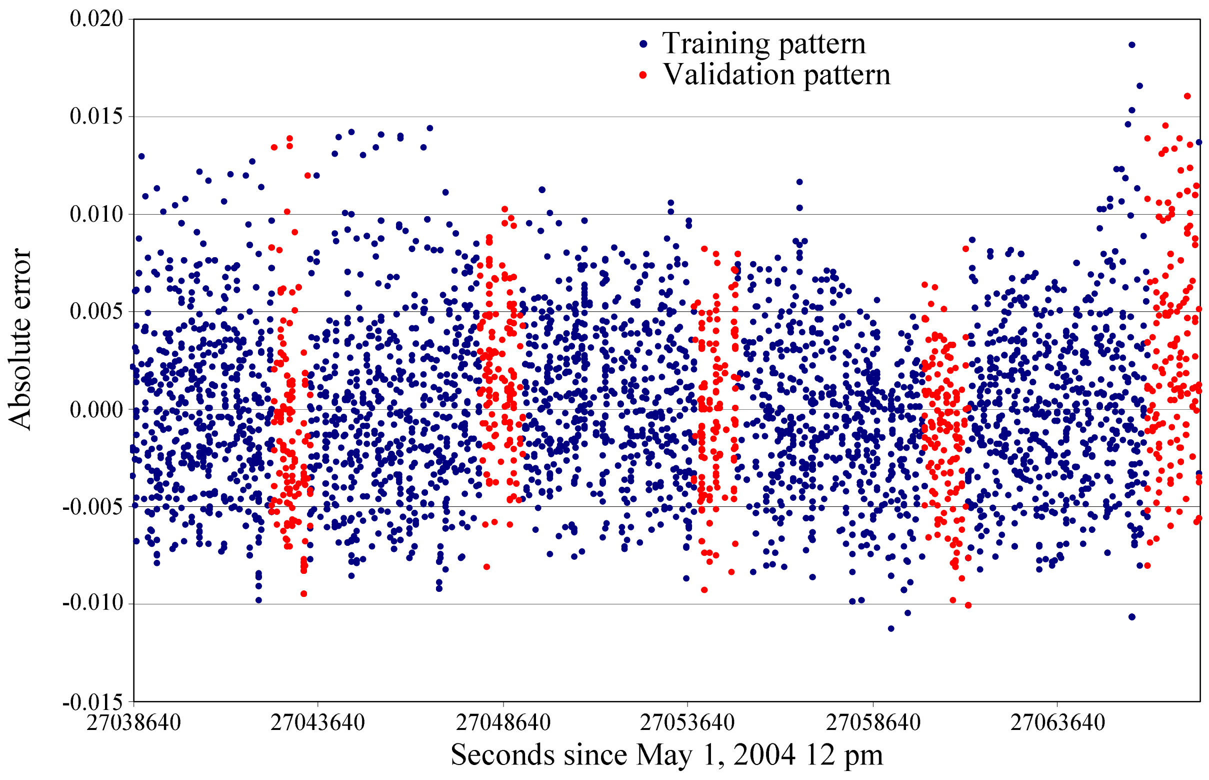


Figure 8: Absolute error of training data and validation data for the second training (interval from -0.95 to 0.95).

Figure 8 illustrates the absolute error for the second training. The numeric option identifier is replaced by a numeric issuer identifier. The resulting error is dependant on quote time, strike price, time to expiration, price of the underlying asset (DAX) and the issuer. The single option has no influence on the resulting neural network. As mentioned before noticeable patterns in

the chart might indicate poorly priced options depending on one of the input variables. The illustration itself does not show such patterns. All prices seem to be well distributed around the market price. The market price is indicated by the neutral axis.
Analyzing the average error for each issuer shows that there are differences anyway. Figure 9 shows the average error combined with the minimum and maximum error for each issuer. The average error shows if an option of an issuer is averagely below or over the market price. In the described test scenario the average option of Dresdner Bank and Deutsche Bank has a lower price than the market price. On the other hand the average option of Commerzbank and DZ-Bank has a price that is higher than the market price. Prices for options of Citigroup and BNP Paribas averagely equal the market price. The average error is only one quality criterion for an issuer's pricing mechanisms. E. g. if half of the options are below the market price and the other half is higher than the market price this results in an average error that is quite equal to the market price although the options are not well priced. Additionally the absolute average of the errors of an issuer indicates how close the prices are to the market price. Together with the absolute average error the range of the errors has to be small. As the range of Citigroup is one of the smallest their pricing mechanism calculates good prices. Note that the number of values per issuer has been adjusted before the training. The difference in the number of extracted patterns per issuer has no influence here. It concludes from this analysis that the option prices of the markets biggest issuer Citigroup are close to the market price and can be precisely calculated by the trained neural network. Dresdner Bank and Deutsche Bank are averagely cheaper than the market price and Commerzbank and DZ-Bank are dearer than the market price. Some issuers' options' prices are quite close to the market price in-average but their absolute average error is higher than the average. This indicates that only a few option prices of this issuer are quite close to the market price. They are distributed below and over the market price so the average hits the market price. E. g. BNP Paribas' average error indicates a good pricing mechanism but it is not, because the absolute average error is higher than the average. In fact BNP Paribas has the highest absolute average error, which indicates an imprecise pricing mechanism.

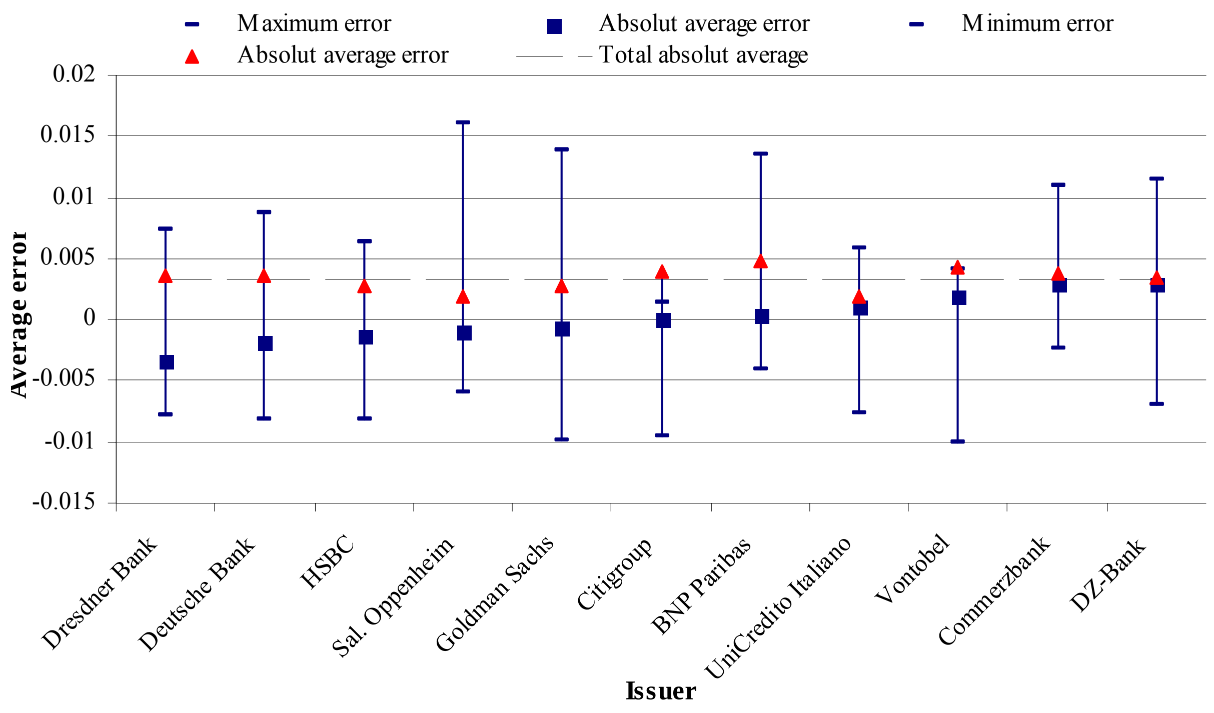


Figure 9: Average errors of the second training for each issuer.

With the same analytic method as for the issuers the pricing for each strike price can be analyzed. Figure 10 illustrated the average error with the according error range and the absolute average error for each strike price. Additionally the number of extracted values for each strike price illustrated. Note that the numbers of values for each strike price has not been adjusted before the training. Therefore, it is not possible to conclude from this illustration to the pricing of such options. Nevertheless it is noticeable that the options with many training patterns have very good prices. Adjusting the numbers of values for the under-represented strike prices would lead to an enhancement in calculating their prices. This would result in an over-all quality enhancement of the option pricing mechanism of WARRANT PRO I.

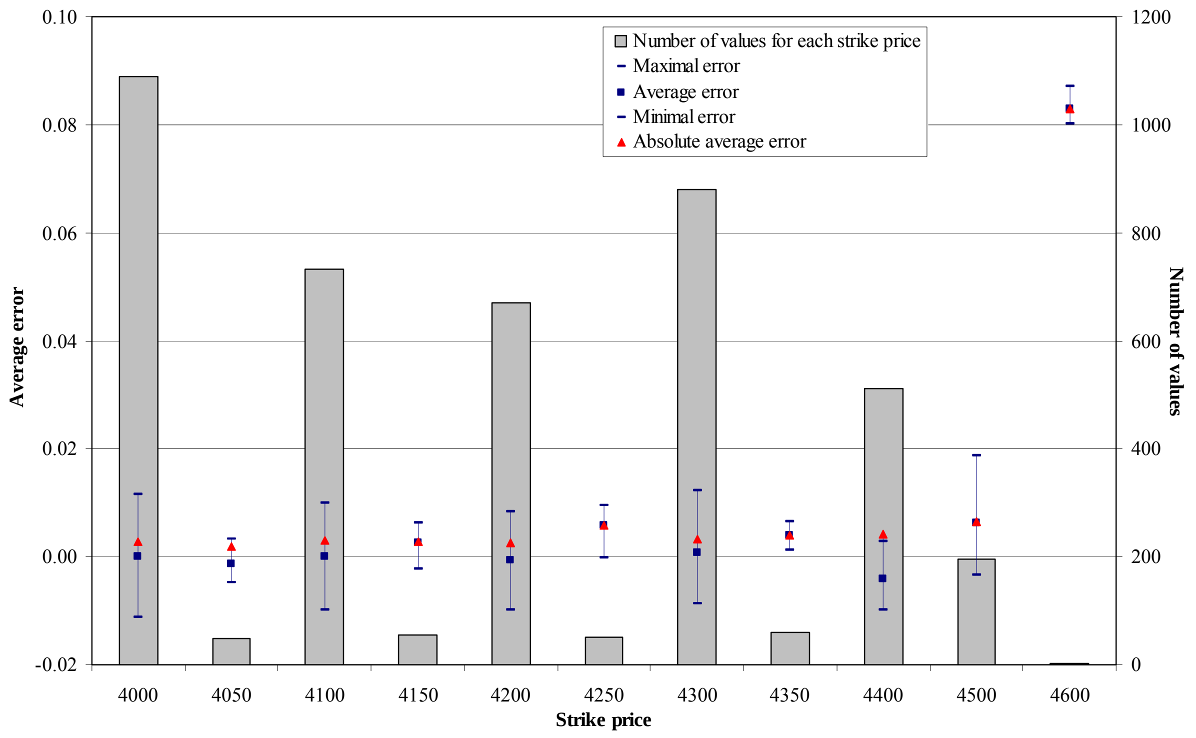


Figure 10: Average errors and number of training patterns of the second training for each strike price.

6 Conclusions and Outlook

Software agents used with artificial neural networks offer a wide range of financial applications. WARRANT PRO 1 incorporates the software agent PISA and the neurosimulator FAUN. It offers the possibility to use financial market data extracted from the internet to synthesize real market (option) price functions. Currently a prototype is developed to analyze feasibility. For analyses the programs Maple and GnuPlot are used, here. The final version will contain own analyses tools and a common graphical user interface (GUI). Here, a test for German DAX call options is presented that confirms feasibility.

The software agent PISA extracts financial market values efficiently. The extracted values are of high quality, i. e. usually have no data gaps and no outliers. They are stored in text files which are successfully used for neural network training with the neurosimulator FAUN. German warrant prices are extracted, here. They are used to build a reliable heuristic pricing model for German options based on highly accurate neural networks. In contrast to theoretical pricing models this heuristic pricing model learns true market price functions of options. It enables customers to single out overpriced and underpriced options a priori (extrapolation) and a posteriori (interpolation). On the other hand issuers are enabled to price (OTC-)options just-in-time. Currently theoretical methods are used which base on unrealistic assumptions.

A test with 53 German DAX call warrants shows that the generated neural network calculates prices for relevant options satisfactorily. Issuers calculate warrant prices differently. Using the issuer as an input variable for the neural network training the neural network learns true market price functions dependent on the issuer. This shows issuer dependant price differences and enables the user to find out issuers with high prices. The test also shows that market prices for nonexistent, not standardized options can be successfully calculated. WARRANT PRO 1 is cost efficient as the used data from the internet are for free. Usually commercial databases are used which are expensive. Here, no licenses have to be acquired.

Today's and future development concentrates on two major aspects. First PISA and FAUN will be better integrated by the framework program. Currently the two programs are used separately. A common GUI will enable the user to control PISA and FAUN within one program. An additional module will replace Maple for analyses and will increase usability. Major component of the analysis module is the market price function which is realized in Java. The Java function implements the weights of the best neural network and provides a real market price function. After the source code is compiled the resulting executable file can be used to either analyze neural networks or calculate market prices for non existing options. Secondly performance of PISA will be enhanced. As long as the webpage structure is correct invalid data are caused by wrong data in the underlying database of the website. Currently, such data can not be differentiated from valid data as long as they are syntactically correct. Beside the fact that webservers can fail the network the agent is executed in can fail as well. Webserver failure can be bypassed by using redundant sources. Local network failure can only be managed by multiple cooperating agents in independent networks, working simultaneously and redundantly (multi-agent system with cooperation). Furthermore, the extraction interval will be automatically adjusted to the data updates to decrease download traffic and webserver load.

Note that extraction and usage of internet data requires special regards towards legal aspects. Usually the general terms and conditions of the financial service providers impose legal restrictions to the usage of the published data. Usually usage is restricted to private purposes. Especially reselling extracted data to third parties is not allowed. Most financial service providers reserve themselves the right of compensation in case of reselling of if adverse effects to the provider's technical infrastructure are determined. WARRANT PRO 1 uses extracted data to provide users the service of generating derivative market price functions. The extraction methods do not harm the system in any way. Neither extracted data nor generated functions are used for profit purposes today. WARRANT PRO 1 does nothing the user can not do manually. As usage of the published data is permitted the usage of WARRANT PRO 1 is legal in terms of the general terms and conditions of most financial service providers, too.

References

[BaBr04] Bartels P., Breitner M.: Finance Applications with the Web Mining Software Agent PISA. In: S. Geber, S. Weinmann und D. F. Wiesner, ed.: Impulse aus der Wirtschaftsinformatik, 5. Liechtensteinisches Wirtschaftsinformatik-Symposium an der Fachhochschule Liechtenstein. Physica-Verlag: Heidelberg, 2004, p. 135 – 150.

[BaBr04b] Bartels P., Breitner M.: Financial Market Web Mining with the Software Agent PISA. In D. Ahr, R. Fahrion, M. Oswald und G. Reinelt, ed.: Operations Research '03. Springer: Berlin, 2004, p. 467 – 474.

[Brei03] Breitner Michael H.: Nichtlineare, multivariate Approximation mit Perzeptrons und anderen Funktionen auf verschiedenen Hochleistungsrechnern (extended venia legendi thesis). Akademische Verlagsgesellschaft: Berlin, 2003.

[Brei00] Breitner M.: Heuristic Option Pricing with Neural Networks and the Neurocomputer Synapse 3, Optimization 47, 2000, p. 319 – 333.

[BrZa$^+$98] Brenner W., Zarnekow R. and Wittig H.: Intelligent Software Agents: Foundations and Applications. Springer: Berlin, 1998.

[Buhl99] Buhl H. U., ed.: Informationssysteme in der Finanzwirtschaft. In: Wirtschaftsinformatik 41, 1999, p. 103 – 144.

[BuKr$^+$01] Buhl H. U., Kreyer N., Steck W., ed.: e-Finance: innovative Problemlösungen für Informationssysteme in der Finanzwirtschaft. Springer: Berlin, 2001.

[GöRo$^+$03] Görz G., Rollinger C.-R., Schneeberger J.: Handbuch der Künstlichen Intelligenz. Oldenbourg: München, 2003.

[Hull02] Hull, John C.: Options, Futures and other Derivatives. Prentice Hall: Upper Saddle River, NJ., 2002.

[JoCv$^+$01] Jouini E., Cvitanic J., Musiela M., ed.: Handbooks in Mathematical Finance: Option Pricing, Interest Rates and Risk Management. Cambridge University Press: Cambridge, 2001.

[KiWe94] Kirn S., Weinhardt C. ed.: Künstliche Intelligenz in der Finanzberatung: Grundlagen, Konzepte, Anwendungen. Gabler: Wiesbaden, 1994.

[LeWi02] Leist S, Winter R., ed.: Retail Banking im Informationszeitalter: integrierte Gestaltung der Geschäfts-, Prozess- und Applikationsebene. Springer: Berlin, 2002.

[Pris00] Prisman E.: Pricing Derivative Securities: An Interactive, Dynamic Environment with Maple V and MATLAB. Academic Press: San Diego, 2000.

[Stub01] Stubblefield L.: Künstliche Intelligenz. Strategien zur Lösung komplexer Probleme. Pearson Studium: München, 2001.

[WoJe95] Woodridge M. and Jennings N.: Intelligent Agents: Theory and Practice, Knowledge Engineering Review 10, 1995, p. 115 – 152.