

Warrant Pro 1: Market Price Synthesis with a Software Agent and a Neurosimulator¹

Dipl.-Ök. Patrick Bartels² and Prof. Dr. Michael H. Breitner³
bartels@iwi.uni-hannover.de and breitner@iwi.uni-hannover.de

Discussion Paper No. 300
July 2004
ISSN: 0949-9962

JEL-Classification: G13, M19, C80, F31
Keywords: Market prices, software agent,
artificial neural networks, neurosimulator, derivatives.

Universität Hannover 
Wirtschaftswissenschaftliche Fakultät

¹ This paper is submitted to the subconference eFinance of the conference Wirtschaftsinformatik 2005 (www.wi2005.de) in Bamberg, Germany.

² Institut für Wirtschaftsinformatik, Universität Hannover, Königsworther Platz 1,
D-30167 Hannover, Germany. Tel.: +49-(0)511-762 4979, Fax: +49-(0)511-762 4013.
Internet: <http://www.iwi.uni-hannover.de>.

³ Institut für Wirtschaftsinformatik, Universität Hannover, Königsworther Platz 1,
D-30167 Hannover, Germany. Tel.: +49-(0)511-762 4901, Fax: +49-(0)511-762 4013.
Internet: <http://www.iwi.uni-hannover.de>.

Table of Contents

1	Introduction	1
2	The WARRANT PRO 1 System	3
3	Option Market Price Functions with the Neurosimulator FAUN	4
4	Web Mining with the Software Agent PISA	7
	4.1 Requirements	7
	4.2 Design and Implementation	9
5	Test Example: Calculating Option Market Prices for DAX Put Options	11
	5.1 Introduction	11
	5.2 Extraction Process with PISA	12
	5.3 Training of Neural Networks with FAUN	13
	5.4 Results	15
6	Conclusions and Outlook	18
	References	19

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

Patrick Bartels, Michael H. Breitner

Universität Hannover

***Abstract:** Today's derivative pricing base on stochastic models developed in the 70's. These models base on some unrealistic assumptions. The system 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. PISA automatically extracts data from the internet or other (semi-)structured text sources. High quality neural network training and validation data with predefinable denseness are provided cost free. FAUN learns true market price functions and computes real market prices. An example for German DAX options is outlined.*

***Keywords:** Market prices, software agent, artificial neural networks, neurosimulator, derivatives.*

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 [Brei99] 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 system. WARRANT PRO 1 synthesizes real mar-

ket 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 The WARRANT PRO 1 System

Like presented above current analytical models of pricing warrants and options are inaccurate. Therefore, the program 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 com-

mon 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 [Buh99]. 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 refer-

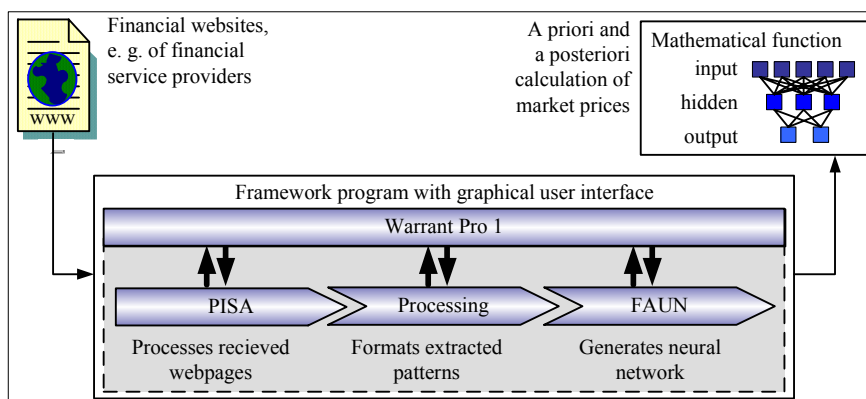


Figure 1: Overview of the WARRANT PRO 1 program.

ence 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 [GöRo⁰³] and [Stub01].

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 [Brei99]. 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 BLAS⁴. The SQP methods NPSOL⁵ and NLSSOL⁶ 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 [BrBa99] and [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 BLAS: Basic Linear Algebra Subprograms, see <http://www.netlib.org/blas/>.

⁵ NPSOL: Nonlinear programming problem solver.

⁶ NLSSOL: Nonlinear least squares problem solver.

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. 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. Accuratness of the extracted data is very important. The agent has to provide rules with which extracted data can be checked on plausibility.

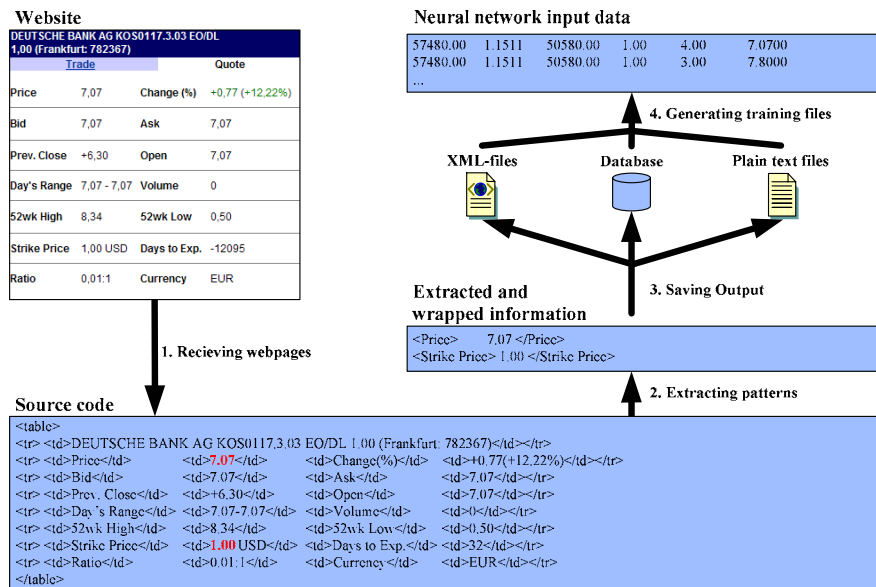


Figure 2: Complete extraction process.

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 alphanumeric 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 processed 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.

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.

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

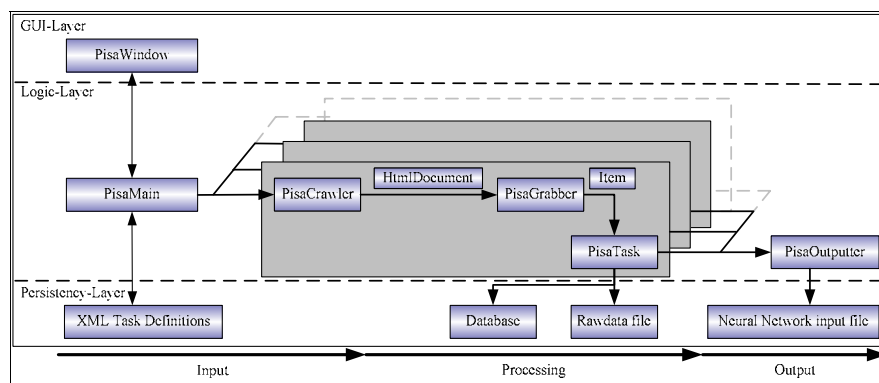


Figure 3: Major modules of PISA.

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.

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.

Seconds to expiration	Strike price	Quote time in seconds	Price of the underlying
8300580	3900.000	3536220	3999.01
0.05601636405010245	3900.000	3536100	3999.09
8300700	3900.000	3536220	3999.01
0.4883	3900.000	3536220	3999.01
8300580	3500.000	3536220	3999.01
0.13578610706149777	3500.000	3536220	3999.01
8473320	4000.000	3536280	3999.71
0.049181065627257985	4000.000	3536280	3999.71
8473440	3600.000	3536160	3998.50
0.11891959484806802	3600.000	3536160	3998.50
8473380	3800.000	3536220	3999.01

Figure 4: Exemplary neural network input file for FAUN.

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 are used to calculate options 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 asked by companies that want to hedge an individual problem for which no standard option matches. Issuers price (OTC-)options within short

time. 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. 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 analyses. It is not used for training.

Generated values must not have any data gaps. Data gaps are caused by poor quality of the offered data and by poor webserver availability. Here, all values for the options were extracted from the Onvista website, see <http://www.onvista.de>. Onvista is market leader in the segment of financial realtime quotes. Previous tests showed that Onvista offers high and reliable data quality, see [BaBr03]. Instead of Onvista all other HTML formatted websites can be used. Webserver availability is also very high. Long term tests showed that within weeks Onvista does not have any webserver breakdowns. Onvista offers all necessary values for the options within the webpage for an option except a usable price of the underlying asset. Here, the underlying is the German stock index DAX. DAX values are extracted from other websites 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 extracted from three different websites: Onvista, Comdirect (see <http://www.comdirect.com>) and CortalConsors (see <http://www.cortalconsors.com>).

On June, 7 2004 Onvista offered values for 44 different DAX put options with a strike price between 3400 and 4000 from 10 different issuers. Values for all 44 options are extracted for one week from Monday June, 7 2004 to Friday June, 11

2004. 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.

Pisa has been tested in detail in different environments and for different tasks in [BaBr04] and [BaBr04b]. Here, a common 2.4 GHz Notebook with 1 Gigabyte Memory is used. Microsoft Windows is used as operating system. The extraction interval is set to 60 seconds for each option and 30 seconds for DAX values. 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.

The overall extracted amount of data is 90 Megabytes. The net file size of used values is about 5 Megabyte for the whole period. The difference in size is caused by redundant values as invalid values are discarded for FAUN input files. 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 size of the FAUN input files for each day is about 1 Megabyte and contains about 4500 different patterns. The number of patterns per day is consistent for each day of the week. Therefore, for intraday training all days can be used. For neural network training values of June, 11 2004 are used. Table 1 illustrates the number of extracted values for this day.

5.3 Training of Neural Networks with FAUN

For numerical performance FAUN requires standardized input files. Therefore, FAUN uses a preprocessing 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. Minima and Maxima of the extracted values are shown in Table 2. 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. The available data of 4480 patterns is departed into training and validation data by a ratio of about 5:1. The validation data is taken out of the extracted data in five blocks of 200 patterns each.

The approximation quality of the neural network can be estimated with the training and validation error functions $\varepsilon_t(W)$ and $\varepsilon_v(W)$. W is the matrix of the weights that constitute a neural network. As usual the perceptron is trained iteratively, i. e. ε_t is decreased by adaption of W , as long as $\varepsilon_v < \varepsilon_t$ or $\varepsilon_v \approx \varepsilon_t$ 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.

Issuer	Security identification number	Number of values	Issuer	Security identification number	Number of values
BNP	344638	102	DZ-Bank	833725	70
	A0CQKK	97		833726	96
	BNP028	98		833727	97
	BNP029	88		833728	96
Citigroup	A0CLGK	120	Goldman Sachs	389101	87
	A0CLGL	101		389102	87
	A0CLGM	105		389130	93
	A0CLGN	92		389131	83
Commerzbank	A0CSZB	95	HSBC Trinkaus & Burkhardt	TB8PUK	111
	A0CSZC	97		TB8PUL	103
	A0CSZD	102		TB8PUM	99
Deutsche Bank	349887	112		TB8PUN	96
	349889	116		TB8PUP	98
	349891	90		TB8PUQ	92
	349893	98	TB8PUR	94	
Dresdner Bank	959795	179	Lang & Schwarz	477768	177
	DR0AFG	97		477828	97
	DR0AFH	98		477885	95
	DR0AFJ	117	UniCredito Italiano	A0CL7F	107
	DR0AFK	116		A0CL7H	97
	DR8PUX	99		A0CL7K	90
DR8PUY	97	A0CL7M	99		
Total number of values				4480	

Table 1: Summarization of the number of extracted options values for June, 11th 2004.

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 ε_v and ε_t . 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 5 shows the online error for the training process of the given task. Each chart represents the error of a single neural network. FAUN optimizes each network by minimization of $\varepsilon_t(W)$ until the overlearning limit is reached. The nearer a networks minimum $\varepsilon_t(W)$ approaches zero the better a network is. FAUN stops automatically when a specified number of adequate networks have been found, here 30 adequate networks are found. A higher number of adequate networks does not lead to any enhancement of training or validation error. Finally the best network is used with $\varepsilon_t(W)=1.413578$.

	Input variables				Output variable
	Strike price	Time to expiration in seconds	DAX value	Quote time in seconds	Real premium
Minimum	3400	8273880	3994.74	3531660	0.0392365961
Maximum	4000	8737020	4018.48	3562920	0.1653128955

Table 2: Minima and maxima of all trained values.

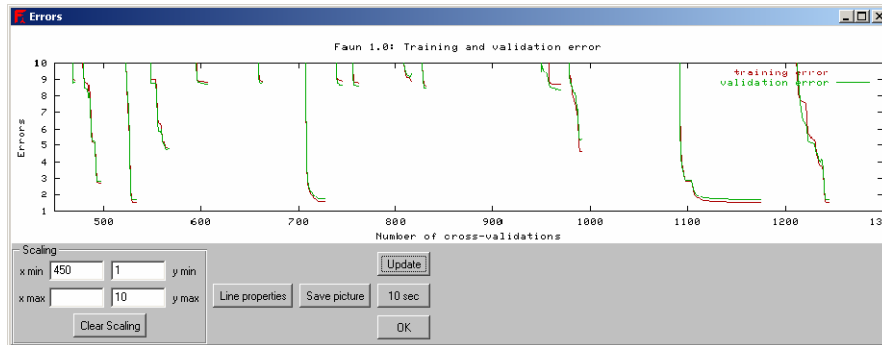


Figure 5: FAUN online error window with four very good networks.

The training process was conducted on the same computer the extraction process was conducted on, see section 5.2. The whole training process with FAUN lasts less than 5 minutes for the given task. Training duration is highly dependent on the used hardware power.

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.

The results of the training process are illustrated with the program Gnuplot. Gnuplot is a portable command-line driven interactive datafile (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 big 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. This makes the resulting output file applicable for neural network training with FAUN. Here, a more powerful computer would not have led to any advantage, as average process-

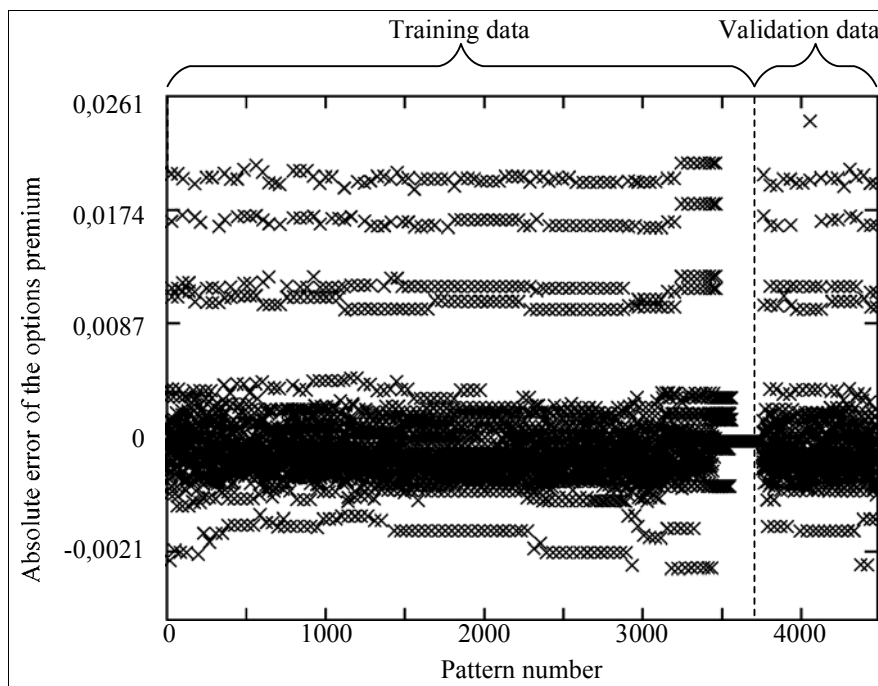


Figure 6: Absolute error of training data and validation data.

ing 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 more than the actual extraction interval of 120 seconds.

The quality of a trained network is measured by the absolute error. The absolute error is the difference of the extracted option's bid price and the calculated market price. If the extracted price is higher than calculated market price this results in a positive error. Under the assumption that the generated neural network is correct this means that options with a positive error are overpriced. Figure 6 shows the absolute errors for each training and validation pattern of the best neural network. The error for most values is small. Comparison of the error for training and validation data indicates that the validation data are adequately calculated. The data gap between training and validation data is caused by the fact that some options were not traded at the end of the extraction day. Their value is missing. In Figure 7 a number of options have a relatively high absolute error. The error is consistent over the training period. This error is not caused by an imprecise function generated by FAUN. Comparison of the issuer of the outliers shows that options with similar high error are issued by the same issuer. Table 3 shows the average error for each issuer and each strike price. The average error is the average of the absolute values of the calculated error for each issuer. It shows how accurate an option

Average error for issuer		
Issuer	Average error	Number of values
Commerzbank	0.01021302	235
DZ-Bank	0.01026423	292
Citigroup	0.01027908	335
Goldman Sachs	0.01091870	292
BNP	0.01215663	318
Dresdner Bank	0.01224107	711
Deutsche Bank	0.01298448	342
Lang & Schwarz	0.02067538	344
HSBC Trinkaus & Burkhardt	0.02232101	569
UniCredito Italiano	0.04765315	315
Total: 3753		

Table 3: Average training errors for each issuer.

of a specific issuer is generated. Options of HSBC Trinkaus & Burkhardt, Lang & Schwarz and UniCredito Italiano differ the most from the calculated market price. The difference is caused by a low transaction volume of these issuers for the related options. Therefore, these options depart the most from the market price. Options of these three issuers are harder to price than options of other issuers.

Table 4 shows the average error for each strike price. The absolute error for options with the strike price 3500 is significantly higher than the rest. Only two issuers offer such an option, Dresdner Bank and HSBC Trinkaus & Burkhardt. The trading volume is very small and only few prices are offered during the training day. This results in an imprecise prediction of options prices. The more values of an option are used for training the more accurate this option is trained.

Average error for strike price		
Strike price	Average error	Number of values
3700	0.01387607	260
3400	0.01450837	892
3600	0.01579086	640
4000	0.01717910	772
3900	0.01799623	227
3800	0.01934073	798
3500	0.02698710	164
Total: 3753		

Table 4: Average training errors for each strike price.

Using the Maple program market prices for nonexistent options can be calculated. The four input variables time to expiration, quote time, DAX value and strike price are predefinable. Maple calculates the market price. Table 5 shows a comparison of the calculated and the real options premium of selected existent options. The absolute error is small. Non existing options are calculated analogously with fictitious input values. Three nonexistent options with fictitious strike prices are shown in Table 5.

Issuer	Security identification number	Strike price	Time to expiration in seconds	DAX value	Quote time in seconds	Real premium	Market premium	Absolute error
Existent options								
DZ-Bank	833726	3600	8462820	4004.54	3546780	0.11500	0.11541	-0.00041
Citigroup	A0CLGL	3600	8305080	4013.09	3531720	0.11664	0.11626	0.00038
Commerzbank	A0CSZB	3600	8293200	4001.33	3543600	0.11404	0.11431	-0.00027
Nonexistent options								
n.a.	n.a.	3575	8293200	4001.33	3543600	n.a.	0.11950	n.a.
n.a.	n.a.	3625	8293200	4001.33	3543600	n.a.	0.10916	n.a.
n.a.	n.a.	3650	8293200	4001.33	3543600	n.a.	0.10418	n.a.

Table 5: Real prices and market prices for existent and nonexistent options.

Both PISA and FAUN are efficient enough to be run on a common desktop computer, here a notebook with 2.4 MHz and Microsoft Windows is used. Linux is supported by all used components. This enables WARRANT PRO 1 to run on nearly every common computer like it is available in companies. No special hardware is required.

6 Conclusions and Outlook

Software agents and artificial neural networks offer a wide range of financial applications. The WARRANT PRO 1 system 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 option market 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 put 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 and a posteriori. 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 44 German DAX put warrants shows that the generated neural network calculates prices for relevant options satisfactorily. Issuers with low option volumes often calculate prices imprecisely. Currently the options issuers are not

trained. In future tests the issuers will be trained, too. Thereby, the neural network learns true market price functions dependent on the issuer. This increases accuracy and comparability even if the trading volume is small. The test also shows that market prices for nonexistent, not standardized options can be successfully calculated. The WARRANT PRO 1 system 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. 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 co-operation). 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 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.
- [BrBa99] Breitner, Michael H. and S. Bartelsen: Training of Large Three-layer Perceptrons with the Neurocomputer SYNAPSE. In P. Kall, ed.: Operations Research '98. Springer: Berlin, 1999, p. 562 – 570.
- [Brei03] Breitner Michael H.: Nichtlineare, multivariate Approximation mit Perzeptronen und anderen Funktionen auf verschiedenen Hochleistungsrechnern (extended venia legendi thesis). Akademische Verlagsgesellschaft: Berlin, 2003.
- [Brei99] Breitner M.: Heuristic Option Pricing with Neural Networks and the Neurocomputer Synapse 3, Optimization 81, 1999, 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.