DESIGNING A SYSTEM FOR STOCK MARKET TRADING ASSISTANCE USING FUZZY LOGIC

Pruteanu Eusebiu[1*], POPA SORIN EUGEN[1]

[1] *"Vasile Alecsandri" University of Bacau, Calea Marasesti 156, Bacau, 600115, Romania*

Abstract: In this paper we present the design of a system for stock market trading assistance. The system will provide buy/sell signals. The aim is to maximize the accuracy of such signals, even with the cost of missing some of them. The architecture of the network will be determined using a genetic algorithm and for training differential evolution will be used. The support vector machine will perform a three-class classification (buy/sell/no action). For this, two support vector machines will be used, one classifying buy against don't buy, and the other one classifying sell against don't sell.

Keywords: stock, trade data, trading strategy, vector machines.

1. INTRODUCTION

The problem consists of designing a system for stock market trading assistance. The system will provide buy/sell signals. The aim is to maximize the accuracy of such signals, even with the cost of missing some of them.

The inputs of the system can consist of certain fundamental data (consider linguistic variables describing fundamentals), past trade data (prices, volumes, technical indicators), or trade data for other securities (exchange rates, commodity prices). Feature selection should be performed to limit the number of input data.

The system will be evaluated with regard to the accuracy of the signals, and, more importantly, a trading strategy will be devised and the return on investment will be taken into account. The system will be considered successful if it can beat several markets in the long run (for several years, both in bull and bear markets).

2. PROBLEM ANALYSIS

For solving the problem, both a support vector machine and a (probably recurrent) neural network will be considered, and their results will be aggregated.

The support vector machine will perform a three-class classification (buy/sell/no action). For this, two support vector machines will be used, one classifying buy against don't buy, and the other one classifying sell against don't sell. If a certain input is classified both as sell and as buy, it will be considered no action, since this contradiction can, most of the times, lead to invalid signals (and our aim is to improve the accuracy of the signals, so no signal at all is better than a wrong signal).

The training method used for the SVM remains to be decided.

The neural network will perform a regression, returning two inputs in the range [-1, 1], signifying the strength of a buy and sell signal respectively. A threshold will be used to separate a signal from a non-signal. This threshold will be adaptable during the operation of the network (it will change according to the rate of success).

[*] Corresponding author, email pruteanue@yahoo.com

The architecture of the network will be determined using a genetic algorithm and for training differential evolution will be used.

A signal will be outputted by the system only when both components (ANN and SVM) agree on the signal.

3. DESIGN AND IMPLEMENTATION OF THE NEURAL NETWORK

The neural network will be developed incrementally and tested at each step before proceeding to the next one. The following steps will be performed:

1. Design simple feed-forward neural network.
2. Test neural network for a simple problem (letter recognition).
3. Preprocess stock market data.
4. Test neural network for stock market data.
5. Change simple MLP to a recurrent neural network.
6. Test recurrent neural network.
7. Implement genetic algorithm for determining optimal network architecture.
8. Test genetic algorithm.
9. Implement differential evolution for training.
10. Perform testing of new neural network.
11. Perform feature selection.
12. Final testing phase.

Step 1: Design simple feed-forward neural network

The basic structure of the neural network is described in the following diagrams:

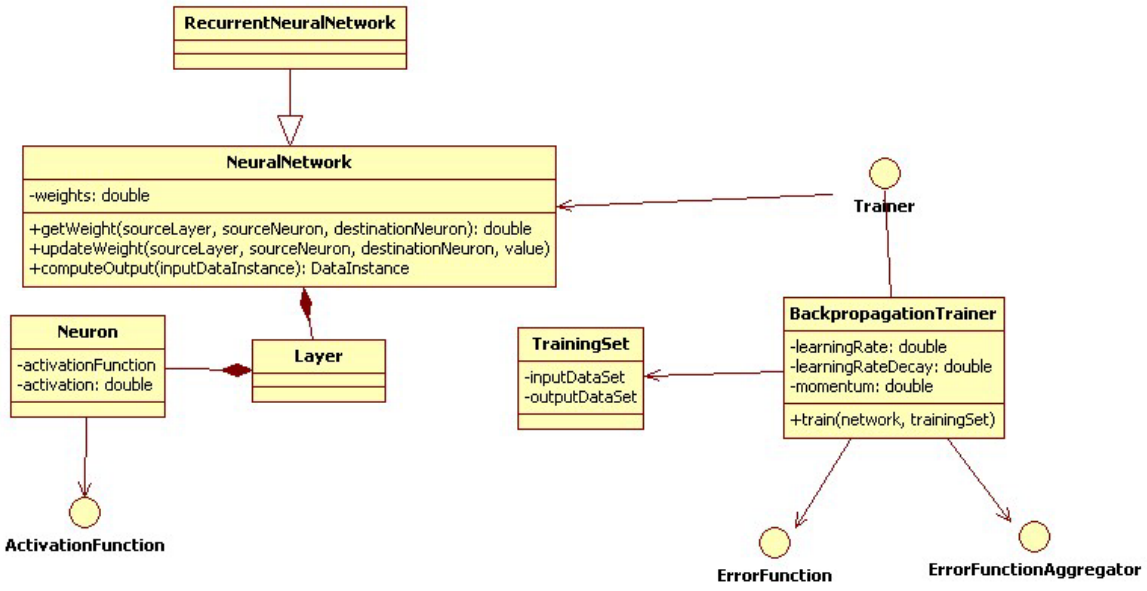

Fig. 1. General structure.

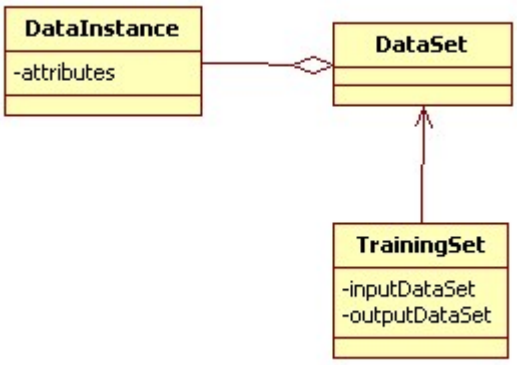

Fig. 2. Data.

While implementing the network the main aim has been to make it highly flexible and extensible, so that it can be tested for several different problems and improvements could be easily developed and plugged-in.

Step 2: Test neural network for a simple problem (letter-recognition)

For this purpose the data set located at http://archive.ics.uci.edu/ml/datasets/Letter+Recognition has been used.

- Data set size: 20000
- Training set size: 12000
- Validation set size: 3000
- Testing set size: 5000
- Input data: As input 16 parameters describing the image were used. Their values were integers between 0 and 15 and were scaled to the interval [0,1].
- Output data: The output consisted of a vector of 26 values, all of which were 0 except for the one corresponding to the letter described by the input, which was 1

The RegularErrorFunction (expectedOutput – actualOutput) was used to compute the error and as an aggregator RootMeanSquareError was chosen.

Sigmoid activation function was used both for the hidden and output layers.

Weights were initialized using a standard distribution with mean = 0 and stdDev = 0.05.

The optimal architecture and parameters for the network were determined through trial and error. Three situations are considered, in a sense, optimal. There description and results are presented below:

- 1 hidden layer (30 neurons), 0.2 non-decaying learning rate, 0.3 momentum, with bias neurons, training for 100 epochs => 81.5% hit rate;
- 1 hidden layer (30 neurons), 0.2 non-decaying learning rate, 0.3 momentum, with or without bias neurons, training for 500 epochs => 82.5% hit rate;
- 2 hidden layer (20+26 neurons), 0.2 learning rate with 0.005 decay, with bias neurons, 0.3 momentum, training for 500 epochs => 85.6% hit rate.

Step 3: Preprocess stock market data

For the time being, only trade data will be used as inputs for the neural network, meaning data that represents or can be derived from past price information.

Only one data set was chosen for testing, the daily FOREX exchange rate for EUR/USD, starting from.. until…

The following inputs were chosen:

- High/low/open/close prices for the last 3 time units;
- High in past 15 time units;

- Low in past 15 time units;
- RSI;
- ROC;
- MACD;
- Stochastic;
- Parabolic SAR;
- CCI;
- SMA – long term;
- EMA – short term;
- ADX.

All indicators were computed based on past prices. The parameters for the indicators were determined based on the observed relevance of different indicators with respect to the evolution of the exchange rate.

The output is two real numbers in the interval [-1,1], representing the strength of a buy and respectively sell signal. This is computed by taking into account the following elements:

- (next 15 time units high) / (current price) divided by a preset REFERENCE_PERCENTAGE and bounded in [-1,1] with a weight of 0.3 and 0.15 respectively;
- (next 15 time units low) / (current price) divided by the REFERENCE_PERCENTAGE and bounded in [-1,1] with a weight of 0.15 and 0.3 respectively;
- (next 15 time units average) / (current price) divided by the REFERENCE_PERCENTAGE and bounded in [-1,1] with a weight of 0.3;
- (closing price in the 15th time unit) / (current price) divided by the REFERENCE_PERCENTAGE and bounded in [-1,1] with a weight of 0.15;
- (sum of negative/positive percentual variations in the next 15 time units) / (current price) divided by a preset second REFERENCE_PERCENTAGE and bounded in [-1,1] with a weight of 0.10;

For this data set the first REFERENCE_PERCENTAGE was set to 15% and the second one was set to 15%.

Also, there is an exception: if both the first/second and third elements are >0.5/<-0.5 and the fourth is positive/negative, then a buy/sell signal with the strength 0.5 will be given.

4. CONCLUSIONS

All indicators were computed based on past prices. The parameters for the indicators were determined based on the observed relevance of different indicators with respect to the evolution of the exchange rate.

While implementing the network the main aim has been to make it highly flexible and extensible, so that it can be tested for several different problems and improvements could be easily developed and plugged-in.

REFERENCES

[1] http://www.fon.hum.uva.nl/praat/manual/Feedforward_neural_networks_1__ What_is_a_feedforward_ne.html
[2] https://www.learnopencv.com/understanding-feedforward-neural-networks/
[3] Daniel Svozil , Vladimir Kvasnicka, JiE Pospichal, Introduction to multi-layer feed-forward neural networks, Chemometrics and Intelligent Laboratory Systems 39 (1997) 43-62;
[4] George S.AtsalakisaKimon P.Valavanisb, Surveying stock market forecasting techniques – Part II: Soft computing methods, Expert Systems with Applications, Volume 36, Issue 3, Part 2, April 2009, Pages 5932-5941
[5] Jarmo Ilonen, Joni-Kristian Kamarainen, Jouni Lampinen, Differential Evolution Training Algorithm for Feed-Forward Neural Networks, Neural Processing Letters, February 2003, Volume 17, Issue 1, pp 93–105.