Forecasting financial markets using advanced machine learning algorithms

Aleksandr Medvedev¹, and Artem Medvedev¹

¹Russian Biotechnological University (ROSBIOTECH), Departments of Computer Science and computer technology of food production, Moscow, Russian Federation

Abstract. This article explores the application of advanced data analysis techniques in the financial sector using neural networks for price forecasting in financial markets. Neural networks, with their ability for self-learning and capturing complex dependencies, offer great potential for accurate financial trend predictions. The article describes the development and utilization of a mathematical model based on convolutional neural networks for forecasting the state of financial markets. The model is trained on historical data, uncovering hidden relationships among various factors and predicting future prices based on acquired knowledge. However, additional research and algorithm optimization are needed to further enhance the accuracy and reliability of the forecasts. The application of neural networks in financial market forecasting represents a crucial area of research that can significantly impact decision-making and the performance of financial operations. Improving the accuracy and reliability of such models can contribute to more effective risk management and better outcomes in the financial sector.

Keywords: price forecasting, financial markets, data analysis, neural networks, self-learning.

1 Introduction

Nowadays, the issue of forecasting the stock market is still relevant because its success brings significant profits to the participants. Ultimately, it is important to predict the moments when the price of a stock is at a local minimum, which opens up an opportunity to buy a stock, and when the price will be at its maximum, to sell it and take profit.

Along with traditional forecasting methods (fundamental analysis and technical analysis), the methods of data mining stand out, the most famous of which is the method using neural networks. Neural networks are quite effective in finding connections between data and can predict (or classify) new data on their basis. Among neural networks, various types of orders are distinguished, the emergence and development are largely caused by the increased needs in solving computer vision problems [1].
The article discusses a mathematical model based on a convolutional neural network, which is used not for the classical problem of image classification and detection, but for recognizing the position of the money market (uptrend, downtrend or its absence - sideways) and predicting the upcoming moments of directional change. The practical importance of the study lies in identifying the optimal moments for opening a position – buying assets on an uptrend and their implementation on a downtrend, which should be done as close as possible to the moment of the next change in the market position to maximize the investor’s profit.

1.1 Statement of the problem

The basis of this work is the creation of a tool based on a convolutional neural network for asset management in the stock market.

The idea of the tool is following: instead of predicting the value of the shares for the next trading day, it is proposed to create an intelligent model to predict whether the market position the next day will remain the same as it is now. The assumption is made that there are only two possible market positions: trend and non-trend (flat). If the investor understands that this is a trend, then he needs to keep the position, and when the position of the trend changes to a flat position (non-trend), the position (long or short) needs to be closed.

The solution to this problem is to develop a software product in Python that implements a machine learning algorithm for predicting future moments of changes in market conditions using a convolutional neural network [2].

1.2 Description of the main approaches

The basis for building a mathematical model is the historical data of the financial market, divided by experts into markup windows, each of which corresponds to a certain situation on the market. The markup was performed in a specially designed graphical interface (Fig. 1). The initial markup contains three types of windows - trend, sideways or unknown position. Each corresponds to some unchanging state of the market [3].
Fig. 1. GUI for layout windows.

1.3 Overview of raw data

The characteristics of the data array used to build (train and validate) the mathematical model are shown in Table 1.

Table 1. Overview of raw data

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of files</td>
<td>1389</td>
</tr>
<tr>
<td>Number of experts</td>
<td>2</td>
</tr>
<tr>
<td>Total number of records</td>
<td>3,304,488</td>
</tr>
<tr>
<td>Number of financial instruments</td>
<td>700 (stocks included in the S&amp;P index)</td>
</tr>
</tbody>
</table>

Each file contains on average about 2600 daily quotes in a certain time interval. The description of the main fields is given in Table 2.

Table 2. Description of the main fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Quotation Date</td>
</tr>
<tr>
<td>Open</td>
<td>Daily Opening Price</td>
</tr>
<tr>
<td>High</td>
<td>Maximum price per day</td>
</tr>
<tr>
<td>Low</td>
<td>Minimum price per day</td>
</tr>
<tr>
<td>Close</td>
<td>Daily close price</td>
</tr>
<tr>
<td>Volume</td>
<td>Trading volume</td>
</tr>
<tr>
<td>Stock name</td>
<td>Financial instrument name</td>
</tr>
<tr>
<td>ID select</td>
<td>The serial number of the “markup window” - the period during which the market condition remains unchanged. In each file, numbering re-starts. Null values are allowed - this means that the type of market condition in this area is not defined.</td>
</tr>
<tr>
<td>Type</td>
<td>Market condition type. Can be “Trend”, “Flat”, or “N / A” (undefined). Based on the economic meaning of the problem, the “N / A” value is equated to “Flat” and is not considered as a separate category.</td>
</tr>
<tr>
<td>Username</td>
<td>The identifier of the Expert Advisor that produced the markup</td>
</tr>
</tbody>
</table>

2 Project implementation

2.1 Basic idea

The development consists of several subtasks, performed in Python [4].

The Python software product consists of five parts: the program code for training and validating the model predicting the presence or absence of points of change in the market
state at a certain time interval (Model ChangePoints_classifier, abbreviated to ChP-c); program code for training and validating a model that predicts the position of the point of the last change in the market state at a certain time interval, provided that at least one change of state has occurred (Model ChangePoints_regression, abbreviated ChP-r); program code for training and validation of a classification model that determines the type of trend observed in a given markup window (TrendOrFlat Model, abbreviated as TF); program code implementing the trading simulation algorithm based on the interaction of the trained ChP-c, ChP-r and TF models; software module (trend_functions) containing auxiliary functions for processing data and supporting the implementation of the trading simulation algorithm.

It is necessary to teach the network to distinguish trends from flat (classification).

It is necessary to teach the network to find and recognize objects in a large picture (detection) [5].

Matrices are fed to the input of the convolutional network, each of which corresponds to one markup window (one object). First, we will teach the network to classify 3 types of market conditions (uptrend, downtrend and no trend), and then we will teach to recognize their boundaries. The code (snippet) below (Fig. 2) is intended to solve the first problem [6, 7-9].

![Fig. 2. Snippet of application code](image)

Structurally, the code for these modules consists of the following sections:

- data preparation (formation of features and labels),
- loading data in the required format,
- creating a model (setting the network structure)
- training the model (setting the loss function, learning algorithm and validation),
- predictions (model validation on a test sample, calculation of metrics and visualization of the result) [10].

The program code implementing the trading simulation algorithm is based on the interaction of the trained ChP-c, ChP-r and TF models.

The logic of the algorithm is following: the first is the ChP_s model, which is responsible for determining the presence of trend change points in a window with a width of n_days (for example, n_days = 25 corresponds to about 5 weeks, and n_days = 75-15). To reduce the calculation time, the windows are taken with a skip step (for example, skip=5
days), which gives a non-critical error. If the ChP_c model detects the presence of a trend change in the window (ChP_c_pred = 1), the ChP_r model is connected to determine exactly where the last trend started (win_srt). If the ChP_c_pred value = 0 (there was no trend change), then the trend start point (win_srt) is the value determined from the previous window (or the beginning of the data file). The end of the trend (win_end) is by default the date on which the ChP_c model is started. Stock quotes taken for the dates between win_srt and win_end are transferred to the TF model, which determines the trend type (uptrend, downtrend or flat) and, accordingly, the Scored New Type Bool value for the current date. It is valid for the next skip days until the next start of the calculation. For the trading simulation algorithm, the final estimate is given by the TF model, therefore, after full processing, the actual points of the ScoredNewTrigger trend change are determined. Intermediate values (win_start points) are stored in the ScoredNewTrigger_tmp variable.

The code works differently for cases when marked or unmarked data is supplied to the input, as well as depending on several other additional parameters that are set before starting the calculation.

The algorithm for the interaction of submodels in real-time simulation is shown in the diagram below (Fig. 3).
2.2 Preparing images

One of the possible options for constructing a mathematical model, subject to its universality (that is, there is no binding to a specific financial instrument, and, in particular, to the range of changes in its prices or trading volumes at any time interval) is to interpret the quotes of a financial instrument not as a set of quantitative indicators, but as images, for example, a chart with Japanese candlesticks, which corresponds to one or another label. This interpretation of the input data contributed to the authors' choice of a convolutional neural network, since the development of a convolutional network provides image classification. Convolutional networks are used not only in the problem of classification, but also in image detection. Image detection assumes recognition of both the type of object (classification problem) and determination of the boundaries of its location on an image.
containing several types of objects of different sizes (regression problem). In our case, the problem of detection can be interpreted as determining the type of trend and its boundaries (beginning and end) on a time interval that acts as a snapshot. Thus, the input data in all three submodels (ChP-c, ChP-r, TF) are matrices of digitized images corresponding to the quotes charts on certain time intervals - data slices.

One of the most common types of chart that allows taking into account all price indicators (open price, close price, high and low) is the candlestick chart (Fig. 4). In this case, the image turns out to be colored, where the green color of the candlestick body, as a rule, corresponds to an increase in the price (the closing price is higher than the opening price), and red - to a decrease. The choice of colors, however, is not critical, as long as they are contrasting [11].

![Candlestick chart diagram](image)

**Fig. 4. Candlestick chart.**

The candlestick chart is a set of pixels arranged in a given order [12]. Each pixel is characterized by intensity, measured in the range from 0 to 255. For black and white images, the value 0 corresponds to the white colour of the pixel, and 255 to black, while intermediate values correspond to different shades of grey. The RGB scheme is often used to digitize colour images. In this case, each pixel is described by a three-dimensional vector containing the colour intensity in three channels - red, green and blue. Thus, the red pixel is described by the vector (255, 0, 0), the green by the vector (0, 255, 0), and the blue by the vector (0, 0, 255). All other colours are obtained by combinations of colour intensities in different channels [13].

The intensity value in each pixel channel corresponds to one variable, therefore, the total number of input variables depends on the image resolution (dpi is the number of pixels per inch) and the number of channels that describe one pixel. So, at dpi = 10 dots per inch, a 3 × 4 colour image will contain 48 × 64 pixels with a dimension of 3, that is, it will be described by a matrix of 48 (height) × 64 (width) × 3 (number of channels) = 9216 elements. Using dpi = 20 resolution increases the number of features by 4 times: the image matrix will already contain (2 × 48) × (2 × 64) × 3 = 4 × 9216 = 36864 elements. Refusal from the colour scheme in favour of monochrome (one black and white channel), on the contrary, leads to a threefold decrease in the dimension of the feature space to 48 × 64 = 3072 elements.
A decrease in the resolution and the number of channels (and, as a consequence, a decrease in the dimension of the feature space) increases the speed of calculations but can lead to the loss of some information [14].

### 2.3 General scheme for building a model based on a convolutional neural network

The general scheme for building a model based on a convolutional neural network includes several stages. At each stage, it is necessary to fix some parameters (more precisely, hyperparameters, models), on which the learning outcome will largely depend. Solution algorithm is implemented in Python using the CNTK library from Microsoft [15]. A similar solution can also be implemented in the libraries TensorFlow [16], Caffe [17] and others, but will require adaptation taking into account the specifics of syntax, requirements for data formats and compatibility with software platforms.

#### 2.3.1 Formation of signs and labels

At this stage, it is necessary to fix the following data preprocessing parameters:

- image characteristics: resolution, number of channels. In the present study, colour images with a resolution of dpi = 10 were used to train the ChP-c and ChP-r models. For the TF model, for which the sample size was smaller, the dpi = 20 and dpi = 60 options were also tested, which did not reveal any significant quality benefits but led to an increase in data volume and processing time.

- the procedure for dealing with duplicates and contradictions.

- the width of the data slice (n_days) and the step (skip) with which they are taken for models that determine the point of trend change (ChP-c and ChP-c). In the constructed models, the options n_days = 25 and n_days = 75 were used, which is slightly longer than the monthly and quarterly intervals. The skipped step is in many respects a technical hyperparameter that regulates the size of the resulting sample (reaches several gigabytes) and, accordingly, the time of its generation (can take tens of hours).

The generated sets of features and labels are saved in a special CTF text format compatible with the features of machine learning procedures for models using the CNTK library.

An example of data recording in CTF format is shown in the figure 5 below (each line corresponds to one observation):

```
labels 0 1 0|features 255 0 0 255 ....255
labels 0 0 1|features 0 0 0 255 ... 0
....
```

**Fig. 5.** Example of data recording in CTF format.

The size of such a file, containing about 100 thousand lines with 9216 thousand variables (one colour image with dpi = 10), is about 3-4 GB, however, the physical volume can be halved if you choose a colour scheme containing only values 0 and 255. In this case, they can be normalized by dividing by 255 and stored as 0 and 1.
2.3.2 Determining the structure of a convolutional neural network

At this stage, the structure of a convolutional neural network is determined, the model of which can have three types of layers (convolutional, unions, fully connected) characterized by different sets of hyperparameters. A multidimensional array containing input variables (features) is fed to the input of the neural network; at the output, the network returns labels.

2.3.3 Training a convolutional neural network

The process of training a convolutional neural network in the CNTK library requires the definition of the following hyperparameters: the type of the loss function, the optimization algorithm, the size of the minibatch, the number of runs / iterations, and the learning rate \(\alpha\). The choice of model hyperparameters can have a decisive influence on the simulation result.

2.3.4 Convolutional neural network validation

The procedure for validating a convolutional neural network is determined by the type of problem. For the classification problem, the main quality metrics are Accuracy and the conjugacy matrix (ChP-c, TF models), as well as indicators AUC, Precision, Recall, F-Score in the case of binary classification (ChP-c model). For the regression problem (ChP-r model), the determination coefficient R2 or the mean absolute error MAE is usually used.

3 Analysis of trading simulation results and conclusions

The table below (Table 3) shows the results of trading simulation for \(n_{\text{days}} = 25\) (ChP-c_4, ChP-r_15, TF_19 submodels) and \(n_{\text{days}} = 75\) (ChP-c_6, ChP-r_13, TF_19 submodels). In both cases, the experiment was carried out for the entire test dataset (that is, data for dates after 10/17/2014 and before 5/13/2017) for all 700 stocks. The skip step was set to 5 to reduce the amount of iteration. However, even in this case, the running time of the algorithm, due to a large number of shares and the time period) for each of the simulations was about 2 days. The total number of data points for which predictions were generated was 477094 for \(n_{\text{days}} = 25\) and 442115 for \(n_{\text{days}} = 75\).

<table>
<thead>
<tr>
<th>Models combination</th>
<th>(n_{\text{days}})</th>
<th>Profit (%)</th>
<th>Days_in</th>
<th>Times_in</th>
<th>Year (%)</th>
<th>Profit (%)</th>
<th>Year Profit_avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChP-c_4, ChP-r_15, TF_19</td>
<td>25</td>
<td>811.89</td>
<td>101359</td>
<td>5088</td>
<td>2.00</td>
<td>0.43</td>
<td></td>
</tr>
<tr>
<td>“Average” Expert</td>
<td>25</td>
<td>14680.46</td>
<td>145293</td>
<td>832</td>
<td>25.26</td>
<td>7.69</td>
<td></td>
</tr>
<tr>
<td>ChP-c_6, ChP-r_13, TF_19</td>
<td>75</td>
<td>1858.58</td>
<td>130529</td>
<td>4747</td>
<td>3.56</td>
<td>1.05</td>
<td></td>
</tr>
<tr>
<td>“Average” Expert</td>
<td>75</td>
<td>11158.39</td>
<td>129325</td>
<td>795</td>
<td>21.57</td>
<td>6.3</td>
<td></td>
</tr>
</tbody>
</table>

The results of trading simulation and their comparison with the results of the “average” Expert Advisor indicate the need to refine the models. Despite the positions are opened
about 6 times more often (increases transaction costs), the final profit on them is several times less.

To better understand the causes of errors, let us analyze the contingency matrices obtained as a result of trading simulation (Figures 6, 7).

**Fig. 6.** Contingency matrix and its normalized version for \( n\_days = 25 \).

The analysis shows that for the case \( n\_days = 25 \) (Fig. 6) in the final counting 63% of the points of upward trends and 86% of the points of downtrends are mistakenly recognized as sideways, which does not allow making money on these tendencies. However, the model at least rarely confuses uptrends with downtrends, which would immediately lead to a loss.

**Fig. 7.** Contingency matrix and its normalized version for \( n\_days = 75 \).

In the case, \( n\_days = 75 \) (Fig. 7) the quality of the final marking is a bit higher. 85% of downtrend points are still recognized as sideways, but the share of unrecognized uptrends is 49% of points, which allows you to earn a little more on this trend in comparison with the case \( n\_days = 25 \). Taking into account the fact that most of the trends marked by experts are long-term and medium-term, we can conclude that using a wider data slice (\( n\_days = 75 \)) is more preferable to correctly identify the points of market change.

The main reason for the unsatisfactory accuracy of predictions is the shortcomings of the ChP-c and ChP-r submodels described in the sections above. They, in turn, can be caused by inconsistencies in the original data markup. One should also take into account the
imbalance concerning the class of downtrends, which led to a relatively lower accuracy of their recognition by the TF model. In general, however, the developed mathematical model for predicting future moments of changes in market conditions using a convolutional neural network is valid, although it can be improved. Improving the prediction accuracy of the ChP-c, ChP-r and TF models is the main direction of further research.

The most successful trading simulation results are shown below as an illustration (Fig.8, 9).

![Image](image1.png)

**Fig. 8.** Example of trading simulation for n_days = 25.

![Image](image2.png)

**Fig. 9.** Example of trading simulation for n_days = 75.

## 4 Conclusion

The article provides a working example of predicting the dynamics of the stock market based on a convolutional neural network model using the preliminary processing of input data.

The practical purposes of applying the results impose some limitations. So, the built model must be universal: it must not be tied to a specific financial instrument, market or time period. After training, it should be applicable for any financial instrument for which standard data on quotes (daily open and close values, maximum, minimum) and trading sizes are available. Another feature is that when training a model based on historical data, you need to keep in mind the absence at the time of calculations of information about the upcoming asset value and trading volume.
The results obtained by the authors indicate a high potential for using this technology. However, the process of training neural networks is quite expensive both in terms of computational resources and time. Note that this study is only one of the steps toward building an effective stock market forecasting tool.

References


The results obtained by the authors indicate a high potential for using this technology. However, the process of training neural networks is quite expensive both in terms of computational resources and time. Note that this study is only one of the steps toward building an effective stock market forecasting tool.

References


