UDC 004.415.2
DOI https://doi.org/10.32838/2663-5941/2020.5/11

Yerastova V.V.
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

Oleshchenko L.M.
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

Yurchyshyn V.Ya.
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

FORECASTING SOFTWARE MARKET PRICE USING BACK PROPAGATION NEURAL NETWORK

This research uses backpropagation artificial neural network to examine whether it is capable of adequately capturing software cost complexities in weight space, to enable it to make accurate estimates.

The input for this task is a set of open information about the software of a certain type. Because of the openness criterion, it was decided to choose mobile apps for Android, full information of this type of software is available on an open source Google Play Market. The purpose of the developed software is to reach the Google server in real time and to receive up-to-date information on the current situation in the mobile application market.

Within the collected data set, backpropagation artificial neural network appears to indicate the potential to be developed into good software price estimation models. The model is not difficult to develop and has the flexibility of being able to incorporate additional attributes as input if special circumstances warrant their inclusion. Neural network has the ability to capture knowledge of the complex interrelationships in weight matrix to make predictions.

For this research the data were divided into three sets. The training set, the test set, and the validation set. The data for each category were randomly chosen, except that the data in the test and validation sets was not allowed to be larger or smaller than the largest and smallest features respectively in the training set. This was done so that predictions were not made outside the data range on which the network had been trained. The inputs were rating, number of ratings, number of downloads, number of reviews, in-app purchases, number of supported languages. The accuracy of the price estimate was taken as the Root Mean Square Error (RMSE).

Key words: *software, market price, software price estimation models, dataset, backpropagation artificial neural network.*

Introduction. Problem statement. In today’s economic environment the development and implementation of new technologies is especially important for the successful competition of companies. An important element in making investment decisions for technological projects is to evaluate their effectiveness. As the market of the purchase/sale of new technologies exists and functions, there is a need to determine the value of the development.

Estimating the cost of technology is needed to analyze the profitability of current and future technological projects and the feasibility of investments.

The ability to estimate the potential benefits and losses from the project in the early stages, to analyze possible scenarios of the development of events becomes very important. According to statistics, about a quarter of all projects is completed on time, a quarter is canceled and about half of all projects are over budgeted or overdue. Most of the reasons are due to incorrect project cost estimates.

Related research. The beginning of intensive studies in the field of cost estimation modeling dates back to 1965–1966, when E.A. Nelson, a member of the System Development Corporation (SDC), who carried out a study on calculating the cost of software for the US Air Force, published a work entitled “Management Handbook for the Estimation of Computer Programming Costs” [1].

Since that time, a lot of models for estimating the cost of software have appeared. Depending on the method of obtaining the initial data in determining the cost and labor costs of developing a software product different methods and models of software cost estimation are offered.

Among the non-algorithmic methods the most commonly used are:

− Price-to-win. The method is based on the principle that the client is always right. The essence of the method is that regardless of the estimated actual costs of project development, the cost estimate of the software is adjusted according to the wishes of the customer [2].

− Expert evaluation. The method is based on the principle of peer review and is used in projects that use new technologies, new processes or solve innovative tasks. The evaluation process involves design engineers who self-evaluate their part of the project. Thereafter, a meeting is convened, at which the results of the individual assessments are integrated into a single, coherent system [3].

− Evaluation by analogy. As a kind of peer review, it often stands out as a separate method. The method is based on the principle of analogy. Evaluation by analogy, like algorithmic models, uses empirical data on the characteristics of completed projects. The key difference is that algorithmic models use this data in a way, for example, to calibrate model parameters, and the method of estimation by analogy with empirical data allows us to select similar projects [4].

The software cost estimation model is one or more functions that describe the relationship between the characteristics of the project and the costs of its implementation.

Models are divided according to the type of functions into linear, multiplicative, power ones and the use of historical data into empirical and analytical ones. The most commonly implemented and well-documented models are SLIM [5] and COCOMO [6].

Nevertheless, it is equally important to be able to estimate the competitive price of a product within the existing software market, that is not allowed in existing evaluation models. For this purpose, the use of assessment methods with the ability to study is the most appropriate, these are updated methods of evaluation by analogy with the subsequent training using artificial intelligence techniques (such as neural networks).

Nowadays, machine learning uses analytical models and algorithms that are continually refined through data-driven learning so that computers can capture hidden meaning without being programmed to where to look for it. This means that scientists and data analysts can teach computers to solve problems without setting rules for each new set of data. Using algorithms that are learned by studying hundreds and thousands of samples of data, computers can solve a similar problem in a new situation, making a prediction based on experience. And they do it with an accuracy that is already comparable to the human intellect.

The value of neural network modelling techniques in performing complicated pattern recognition and non-linear estimation tasks has been demonstrated across an impressive spectrum of applications.

This study uses backpropagation artificial neural networks to examine whether they are capable of adequately capturing software cost complexities in their weight space, to enable them to make accurate estimates.

Problem solution and results. The prediction task is first and foremost the task of data analysis. For any data analysis problem, there are major steps to solving it. General scheme of the forecasting task is presented in the Fig. 1.

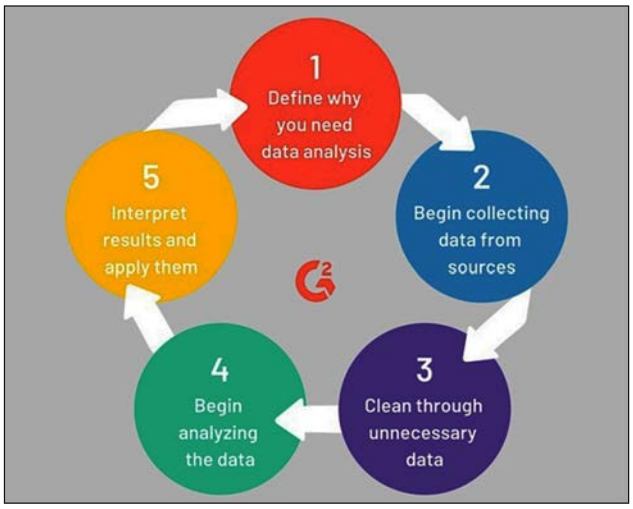

1
Define why you need data analysis
2
Begin collecting data from sources
3
Clean through unnecessary data
4
Begin analyzing the data
5
Interpret results and apply them

Fig. 1. General scheme of the forecasting task

In this case, the goal is to predict the market value of the software. The next step is data preparation, which involves identifying data sources for analysis, combining and cleaning them. The required data can be in different databases and on different servers. Moreover, they can be presented in the form of text files, tables, in different formats.

The collected data usually requires additional processing, which called cleaning. The output of this step should be structured data in the form of a rectangular table, where each row represents a separate case, object or condition of the investigated object, and each column – the parameters, properties or characteristics of all investigated objects.

The next step is to build a model. The model includes a reference to the data mining algorithm and its parameters, as well as the analyzed data. The model may be trained in applying the selected algorithm to the training dataset. After that, it stores the identified patterns.

The last step is to test the model. The purpose here is to evaluate the quality of the model created. When solving forecasting problems, the quality of the forecast given by the model can be verified on a test set that knows the value of the forecast parameter.

Data collection. The input for this task is a set of open information about the software of a certain type. Because of the openness criterion, it was decided to choose mobile apps for Android, full information of this type of software is available on an open source Google Play Market. The purpose of the developed software is to reach the Google server in real time and to receive up-to-date information on the current situation in the mobile application market. At this time, Google does not provide the official API to access mobile app data hosted on the Play Market. Several API's for access to data were considered:

1. Google Play Unofficial Python API (https://github.com/egirault/googleplay-api).

2. Python Android Market Library (https://github.com/liato/android-market-api-py).

3. 42matter (https://42matters.com).

The third option was the most suitable for this prediction task, because HTTP-request GET that POST of this RESTful API allows to get all the meta-data of any application in a convenient for the next processing format JSON (Fig. 2).

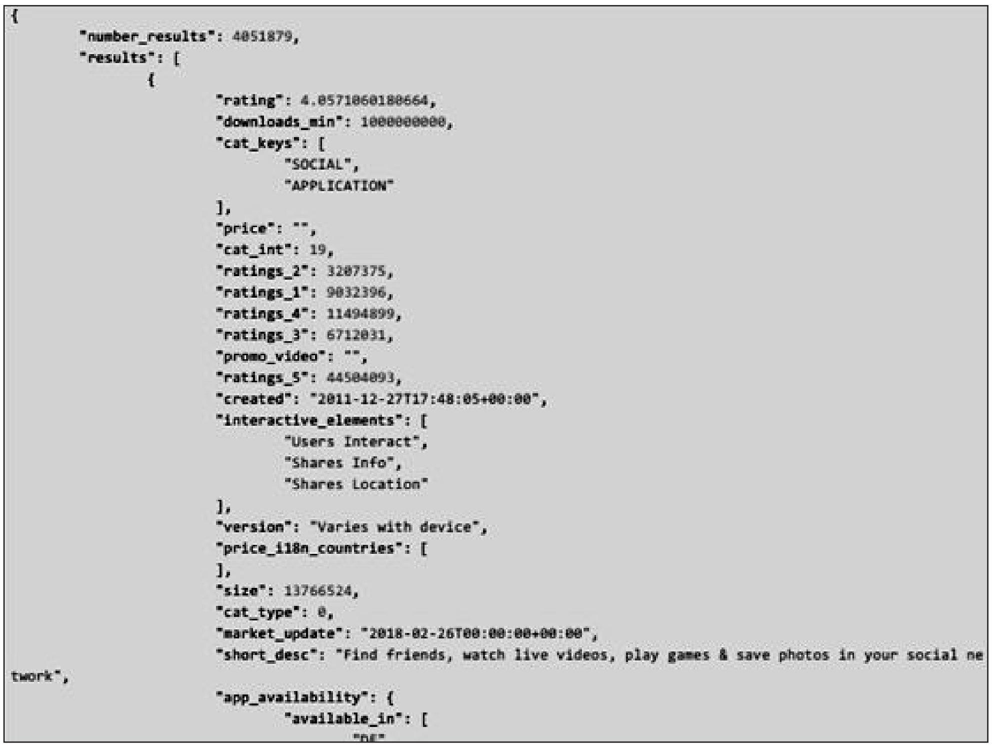
It was decided to select software features that could affect the price of the analogue product as a key feature of the received information.

The first feature is an app rating. Rating is the definition of an estimation parameter or group of parameters according to a certain estimation algorithm, according to a given ranking scale. In fact, rating is a measure of the popularity of anything. The rating is determined by the survey method of a large target group or a limited group of experts. The Google Play Store service is ranked by users. It is presented as 5 stars, with 5 stars being the best rating and 1 star being the lowest.

The next feature is the number of ratings. This parameter represents the total number of users who rated the app from 1 to 5 stars. Also, the attributes are the number of individual app ratings, that is, the number of users who rated the app 1 star, the number of users rated 2 stars, and so on. We get five more features.

Another feature is the number of downloads. Downloading an application is transferring the application data from the Google Play Market to the device and installing it. That is, this parameter shows the number of users who installed the application on their device. Another important feature is the number of feedbacks. Feedback is an opinion of users is based on an affirmative attitude toward the app on Google Play Market. Another feature is the presence of in-app purchases provided by the in-app purchase of virtual goods, services, or digital content.

Also, quantity of supported languages in the application was selected as feature. This feature shows how many languages are available in the Google Play Market app. The dependent characteristic, which is called the target variable, is the price of the applica-



```
{
        "number_results": 4051879,
        "results": [
                {
                        "rating": 4.0571060180664,
                        "downloads_min": 1000000000,
                        "cat_keys": [
                                "SOCIAL",
                                "APPLICATION"
                        ],
                        "price": "",
                        "cat_int": 19,
                        "ratings_2": 3207375,
                        "ratings_1": 9032396,
                        "ratings_4": 11494899,
                        "ratings_3": 6712031,
                        "promo_video": "",
                        "ratings_5": 44504093,
                        "created": "2011-12-27T17:48:05+00:00",
                        "interactive_elements": [
                                "Users Interact",
                                "Shares Info",
                                "Shares Location"
                        ],
                        "version": "Varies with device",
                        "price_i18n_countries": [
                        ],
                        "size": 13766524,
                        "cat_type": 0,
                        "market_update": "2018-02-26T00:00:00+00:00",
                        "short_desc": "Find friends, watch live videos, play games & save photos in your social ne
twork",
                        "app_availability": {
                                "available_in": [
                                        "DE"
```

Fig. 2. Result of the request to API

tion. After the data is collected, it must be prepared. This stage is called preprocessing. The main task of preprocessing is to display data in a format suitable for learning the model. For this purpose, unnecessary information was deleted and all data was normalized.

Neural network model. For this research, backpropagation artificial neural network models were used. Backpropagation networks are the most generalised neural networks currently in use and this approach was chosen in preference to Hopfield and Kohonen networks. As software development estimation is not a time series problem, approaches such as finite impulse response (FIR) and recurrent networks were not considered.

The backpropagation network requires data from which to learn. To learn the network calculates the error, which is the difference between the desired response and the actual response, and a portion of it is propagated backward through the network. At each neuron in the network the error is used to adjust weights and threshold values of the neuron, so that at the next epoch the error in the network response will be less for the same inputs. This corrective procedure is called backpropagation and is applied continuously for each set of inputs or training data. The training data should consist of as much relevant data as possible. In practice one does not usually have the luxury of a perfect training data set. The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem (Fig. 3).

For this research the data were divided into three sets. The training set, the test set, and the validation set. The data for each category were randomly chosen, except that the data in the test and validation sets was not allowed to be larger or smaller than the largest and smallest features respectively in the training set. This was done so that predictions were not made outside the data range on which the network had been trained. The inputs were rating, number of ratings, number of downloads, number of reviews, in-app purchases, number of supported languages. The target against which the network was trained was the price of the application in the training set. The accuracy of the price estimate was taken as the Root Mean Square Error (RMSE).

To try and improve the network performance, the learning rate and momentum were varied, as was the network architecture. Models with one through to six hidden layers were developed. Consistently the models with just a single hidden layer performed better, while the models with multiple hidden layers in many instances did not converge. Various activation functions were tried, and the popular sigmoid function consistently gave the best results.

There is no clearly defined theory which allows for the calculation of the ideal parameter settings and as a rule, even slight parameter changes can cause major variations in the behavior of almost all networks. It is through a process of trial and error and experience that settings are selected which will result in a reduced average prediction error. The settings of the learning rate and momentum control the way in which the error is used to correct the weights in the neural network foreach training case. When the learning rate is set to high values (close to 1) there is the possibility of unstable behavior, as evidenced by widely varying average error values. When the learning rate is set lower, the

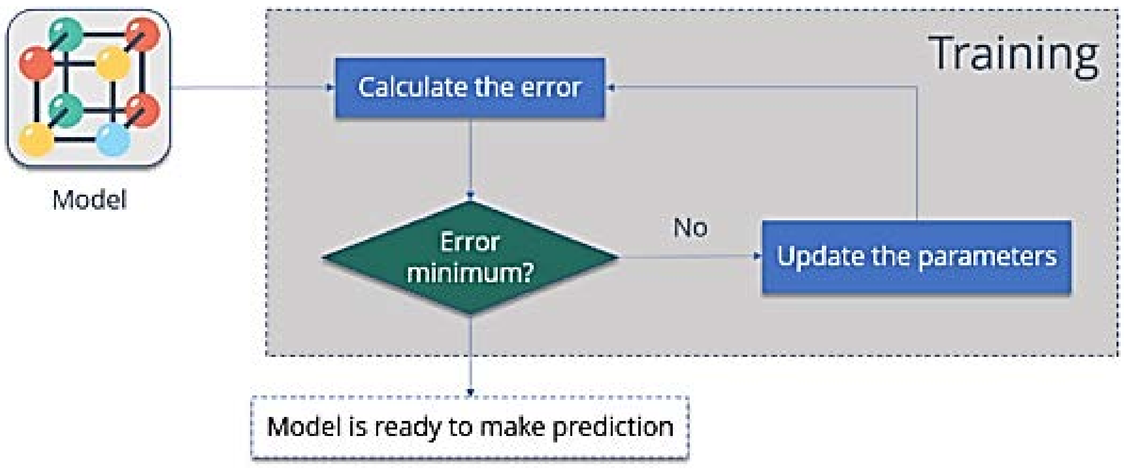


Fig. 3. Training a model

possibility of unstable behavior is reduced, but training times are increased and there is a greater probability of getting stuck in local error minima. The higher the momentum, the larger the percentage of previous errors that is applied to the weight adjustment in each training case [8]. For this set of data a learning rate of 0.2 and a momentum of 0.5 gave good results. The neural network architecture of a single hidden layer using a sigmoid activation function tended to result in the lowest average prediction error. The best results were obtained with a 17-5-1 architecture.

Analysis of data. Network models were developed with various combinations of inputs selected from the attributes mentioned above. The results were ambiguous, because prediction errors were erratic. An examination of the results showed that the network overestimate the price of the application with lower rating and worse indicators, as well as underestimate price of the application with higher rating. For the remaining applications, the estimated price was more accurate.

Conclusions and future work. Within the collected data set, backpropagation artificial neural networks appear to indicate the potential to be developed into good software price estimation models.

The model is not difficult to develop and has the flexibility of being able to incorporate additional attributes as input if special circumstances warrant their inclusion. Neural networks have the ability to capture knowledge of the complex interrelationships in their weight matrix to enable them to make predictions. Further research will be conducted to use larger set of training examples, that covers all possible values of software characteristics. It will allow to make training networks more stable. Also semantic analysis of application reviews is of interest. This type of analysis can provide an opportunity to identify positive and negative references in order to better prioritize applications.

References:

1. Nelson E.A. Management Handbook for the Estimation of Computer Programming Costs. AD-A648750, Systems Development Corp., 1966. 141 p.
2. Boehm B.W. Software engineering economics. Prentice-Hall, 1981. 320 p.
3. Coates J. Technological Forecasting and Social Change. Elsevier Science Inc., 1999. 235 p.
4. Shepperd M.C. Schofield Estimating software project effort using analogy. IEEE Trans Software Eng., 1997. P. 736–743.
5. Heires J., Doing T. More with Less: SLIM-Estimate 5.0 Product Review. QSM Software, 2002. 46 p.
6. Boehm B.W. The COCOMO 2.0 Software Cost Estimation Model. American Programmer, 2000. 586 p.
7. Menzies T., Shepperd M. Special Issue on Repeatable Results in Software Engineering Prediction. *Empirical Software Eng*. 2012. Vol. 17. No. 1. P. 1–17.
8. Tadayon N. Neural network approach for software cost estimation. *Information Technology: Coding and Computing* : International Conference on. 2005. ITCC 2005. Vol. 2. IEEE, 2005. P. 815–818.

Єрастова В.В., Олещенко Л.М., Юрчишин В.Я. ПРОГНОЗУВАННЯ РИНКОВОЇ ЦІНИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ НЕЙРОННОЇ МЕРЕЖІ ЗВОРОТНОГО ПОШИРЕННЯ

У дослідженні використовується нейронна мережа зворотного поширення для того, щоб вивчити, чи здатна вона адекватно фіксувати складність витрат на програмне забезпечення у ваговому просторі для отримання точних оцінок.

Вхідні дані для завдання – це набір відкритої інформації про програмне забезпечення певного типу. Для забезпечення критерію відкритості було обрано мобільні додатки для Android, повна інформація про цей тип програмного забезпечення доступна на відкритому коді Google Play Market. Метою розробленого програмного забезпечення є доступ до сервера Google в режимі реального часу й отримання актуальної інформації про поточну ситуацію на ринку мобільних додатків.

У межах зібраного набору даних нейронна мережа зворотного розповсюдження вказує на потенціал, який можна перетворити на хороші моделі оцінки цін на програмне забезпечення. Модель гнучка, оскільки може містити додаткові атрибути, такі як вхідні дані, якщо особливі обставини вимагають їхнього включення. Нейронна мережа зворотного поширення має здатність фіксувати знання про складні взаємозв'язки у своїй ваговій матриці, що дозволяє їй робити прогнози.

Для дослідження дані були розділені на три групи: навчальний набір, набір тестів і набір перевірок. Дані для кожної категорії були обрані випадковим чином, за винятком того, що дані в наборах для тестування та перевірки не мали права бути більшими чи меншими, ніж найбільші й найменші ознаки відповідно в навчальному наборі. Це було зроблено для того, щоб прогнози не робили за межами діапазону даних, на якому тренувалася мережа. Вхідними даними були рейтинг, кількість оцінок, кількість завантажень, кількість оглядів, покупки в додатку, кількість підтримуваних мов. Точність оцінки ціни програмного забезпечення була прийнята як середньоквадратична помилка (RMSE).

Ключові слова: *програмне забезпечення, ринкова ціна, моделі оцінки ціни програмного забезпечення, набір даних, нейронна мережа зворотного поширення.*