UDC 004.855.5

O. Bondar, O. Lisovychenko

REVIEW AND ANALYSIS ONE OF THE APPROACHES TO TRAINING THE NEURAL NETWORK MODEL

Abstract: The article provides an overview and analysis one of the approaches to training a neural network model. The list of requirements for training the model was created. As the architecture of a neural network, feedforward networks (perceptron and multilayer perceptron) were considered and a multilayer perceptron was selected. The input data for training and testing the model were information from the electricity market for one year. Each stage of building and training a model that will serve to predict data was described. The trained model was tested using the mean absolute percentage error (MAPE). Graphs and tables were built that show the dependence of the accuracy of the model on the parameters and input data during training. The results obtained can be used for practical application, in particular for predicting economic indicators.

Keywords: artificial neural network, machine learning, deep learning, prediction, feedforward networks, perceptron, multilayer perceptron.

Introduction

It is difficult to imagine the modern IT industry without using artificial intelligence. The basis of this area is machine learning, which, in turn, contains a large number of algorithms.

Today, the most well-known class of machine learning algorithms is deep learning, which has become quite popular in recent years. This is due to the emergence of powerful computers, as well as a large number of marked data sets.

Deep learning uses a multi-layered filter system to hierarchically extracting the desired features, that is, each layer at the input receives the output from the previous layer. Usually, this type of training uses different types of neural networks.

An artificial neural network (ARN) is a system that can not only use the algorithms that have been chosen for it but also self-learn with the help of data that it has previously calculated. This training is created due to the fact that the neural network contains many layers, which one after another transmit to the next layer the found features of the object, thus creating more complex features and give the desired result.

Deep learning is used to solve the following tasks:

- medical diagnostics;
- data forecasting;
- automatic image recognition;
- recommendation systems;
- natural language processing.

 ISSN 1560-8956

Nowadays, the greatest attention is paid to data forecasting. This process is used in different situations. For example, weather forecasts, economic forecasting, political forecasting, pedagogical forecasting, military, and others.

Of these uses, the most useful is economic forecasting, as this factor is the most important in the formation of a successful state.

Accordingly, it was decided to consider and analyze one of the approaches to training the neural network model that will serve to predict data.

Problem overview

The accuracy of the system that will use the neural network model depends on the quality of this model, so model training is the most important step.

Accordingly, the main task of this article is to review and analyze one of the approaches to training the neural network model.

Based on this, the following requirements were formed, which are required for analysis:

- consider and choose one of the direct distribution networks;
- select the value of the optimization factor and the required number of input layers of the network;
- create and teach several models using a different number of cycles, hidden layers, and other parameters;
- for each model calculates the average absolute error in percentages (MAPE);
- compare the obtained results in the form of graphs and tables.

Review and analysis of feedforward networks

The first task of the study is to review feedforward networks and select the most suitable for forecasting data.

Classical and multilayer perceptrons are types of neural networks that are quite rectilinear, as they transmit information from the input layer to the output, passing through each layer. Figures 1.1-1.2 show examples of such networks.

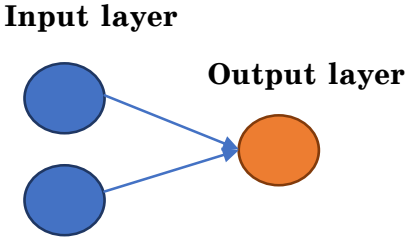

Figure 1.1. Perceptron

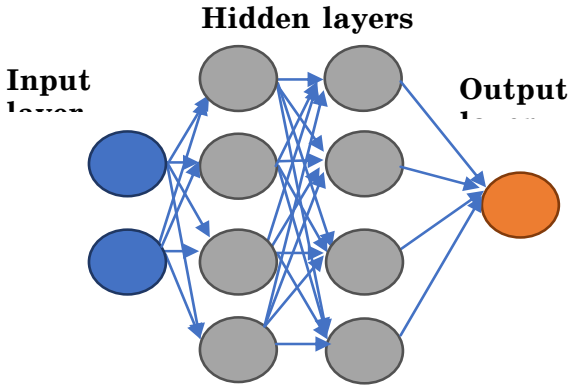

Figure 1.2. Multilayer perceptron

The classical perceptron is usually used to solve classification problems based on binary input data. Accordingly, this neural network architecture is not quite suitable, because, for data prediction, the input is not only binary signals.

The multilayer perceptron is one of the types of feedforward neural network, which is very similar to the classical perceptron, except that this neural network contains additional hidden layers in addition to the input and output. In this type of network, all neurons except inputs use a nonlinear activation function. When training, the multilayer perceptron uses "teacher training" and an error backpropagation algorithm. Practice proves that this grid works very well in solving problems of data forecasting and the construction of regression models.

Creating and training the model

Electricity market data were selected for research. Based on these data, need to create a model that can be used to forecast market values “for the day ahead”. The input data for training are the value of the intraday electricity (VDR) market and the value of the market “for the day ahead” (RDN) for twelve months, which are divided by time and day.

The best tools for creating and training a model are the Python programming language, the scikit-learn machine learning library, and pandas library for working with the dataset. Auxiliary are: Visual Code programming environment and interactive shell for IPython programming.

Using the pandas library, twelve datasets were obtained, which were combined by VDR and RDN, x and y, respectively. Data were checked for absence and correctness of values.

The next step was to divide the data into input and output. To do this, we used the train_test_split function from the sklearn.model_selection library. Figure 2 shows an example of using this feature.

```
X_train, X_test, y_train, y_test = train_test_split(
    X_VDR, Y_RDN, random_state=42, test_size=0.2
)
```

Figure 2. Splitting data

 ISSN 1560-8956

As can be seen from the figure, 20% of the total data size was selected for the x and y test samples, and the number 42 was selected (random_state) generation data.

A multilayer perceptron (MLPRegressor) from the sklearn.neural_network library was used as a neural network. In that the model will train with each cycle, the warm_start parameter was selected. The optimization algorithm – L-BFGS, the number of epochs (max_iter) was 2000 cycles (Fig. 3.).

```
mlp_regr = MLPRegressor(warm_start=True, solver='lbfgs',
max_iter=2000, alpha=0.001, hidden_layer_sizes=3)
```

Figure 3. Creating a model using multilayer perceptron

The next task was a selection of network parameters: the number of hidden layers, the regularization parameter. To automatically calculate these parameters, is used the Random Search method. However, the automatic calculation also has a disadvantage, which is that the learning speed of the model is reduced.

The following values of parameters were chosen for the initial analysis: the number of hidden layers - 3, regularization parameter - 0.001 (Fig. 3.).

As input for training and testing, it was decided to create two arrays (for values of x and y), which are filled by passing through datasets and taking for x at each iteration two days with VDR, and for y - the next day with RDN (Fig. 4.1-4.2.).

Hour 16	Hour 17	Hour 18	Hour 19	Hour 20	Hour 21	Hour 22	Hour 23	Hour 24
1,084.11	1,178.50	1,270.00	1,270.00	1,250.00	1,134.32	1,063.50	995.00	750.00
2,039.55	2,039.55	2,040.56	2,040.55	2,040.55	1,957.00	2,039.55	2,039.55	959.12
1,500.00	1,550.00	1,550.00	1,536.17	1,520.00	1,450.00	1,607.20	1,520.00	949.00
1,141.88	1,250.00	1,295.00	1,257.18	1,198.00	1,178.50	1,198.00	1,087.00	949.00
1,825.00	1,900.00	1,957.00	1,957.00	1,900.00	1,957.00	2,039.55	2,039.55	959.12
1,260.10	1,295.00	1,376.82	1,339.00	1,331.00	1,331.00	1,376.82	1,208.35	956.00
1,178.00	1,295.00	1,299.00	1,295.00	1,295.00	1,295.00	1,377.00	1,257.18	949.00
1,550.00	1,450.00	1,550.00	1,550.00	1,650.00	1,650.00	1,844.84	1,698.57	956.00
1,890.00	1,845.00	1,970.00	1,770.00	1,770.00	1,770.00	1,620.00	1,579.91	948.00
2,000.00	2,012.40	2,039.55	1,994.10	1,930.00	1,910.00	1,999.60	1,850.00	948.00

Figure 4.1. Input data for training

Hour 18	Hour 19	Hour 20	Hour 21	Hour 22	Hour 23	Hour 24
1,259.29	1,278.44	1,294.36	1,246.56	1,102.06	1,057.38	834.00
1,888.73	1,910.08	1,913.89	1,899.45	1,920.61	1,928.16	932.41
1,507.09	1,512.31	1,489.20	1,462.04	1,515.27	1,529.44	915.06
1,298.76	1,290.88	1,231.95	1,224.45	1,241.56	1,147.59	947.15
1,867.82	1,878.48	1,855.51	1,897.61	1,915.65	1,922.50	901.23
1,426.92	1,443.31	1,443.09	1,435.27	1,459.72	1,396.39	955.24
1,377.84	1,376.60	1,389.96	1,388.81	1,417.97	1,343.16	944.54
1,619.52	1,629.12	1,691.13	1,686.03	1,721.64	1,721.09	929.29

Figure 4.2. Output data for training

An example of filling arrays in the code is shown in Figure 5.

```
for data in range(len(data_x)):
    column = 0
    if data + 1 != len(data_x):
        if data + 3 <= len(data_x):
            array_x.append(np.concatenate(data_x.iloc[[data,data + 1]].T.values))
            array_y.append(data_y.iloc[data + 2].T.values)
        else:
            array_x.append(np.concatenate(data_x.iloc[[data+1,column]].T.values))
            array_y.append(data_y.iloc[column+1].T.values)
    else:
        array_x.append(np.concatenate(data_x.iloc[[column,column+1]].T.values))
        array_y.append(data_y.iloc[column + 2].T.values)
return array_x, array_y
```

Figure 5. Filling arrays

The figures show that data filling begins with the first element and continues until the end of the dataset is reached.

Below are all the operating conditions of this filling algorithm.

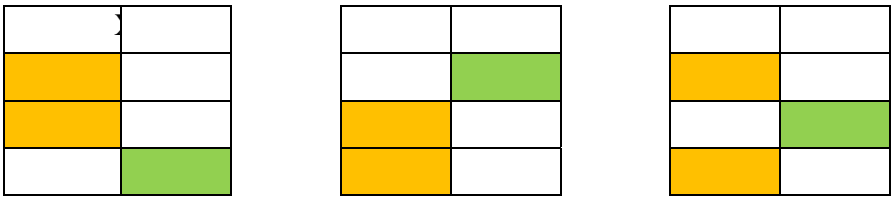

Figure 6. Conditions for filling arrays with data

The fit method is shown below which is used to train the model:

```
mlp_regr.fit(x, y)
```

Figure 7. Training the model

After training, the model was tested, finding the average absolute error in percent (Fig. 8.).

```
res = mlp[0].predict(data[0])
print('MAPE', np.mean(np.abs((data[1] - res)/data[1]))*100)
```

Figure 8. Testing the model

To calculate the MAPE, you need to get the predicted data using a trained model and take the original source data. The test result is shown below:

```
MAPE 18.990311881807948
```

Figure 9. The result of testing

Based on the steps described above, additional research was conducted and MARE was calculated to verify the accuracy of learning the neural network model depending on the parameters specified at the stage of creating the model. Based on this, a table with the results of the study was created (Table 1.).

 ISSN 1560-8956

Table 1.

Dependence of learning quality on input parameters

Number of hidden layers	Number of training epochs	Regularization parameter	Number of values for training (units /%)	MAPE
3	2	10	292/80%	82.0
3	20	10	292/80%	20.1
3	200	10	292/80%	19.7
3	2000	10	292/80%	19.0
50	2000	10	292/80%	26.5
50	200	10	292/80%	0.5
50	20	10	292/80%	1.4
50	2	10	292/80%	0.1
3	2000	1	292/80%	9.7
3	2000	0.1	292/80%	8.99
3	200	0.1	292/80%	9.5
3	20	0.1	292/80%	0.5
3	2000	0.01	292/80%	9.5
3	2000	0.001	292/80%	8.7
3	2000	0.001	183/50%	1.5
3	2000	0.1	183/50%	1.2

According to these data, a graph was constructed (Fig. 10.), which visually shows the dependence of the quality of the model on the input parameters. The smaller the average absolute error in percent, the higher the quality of the model, respectively.

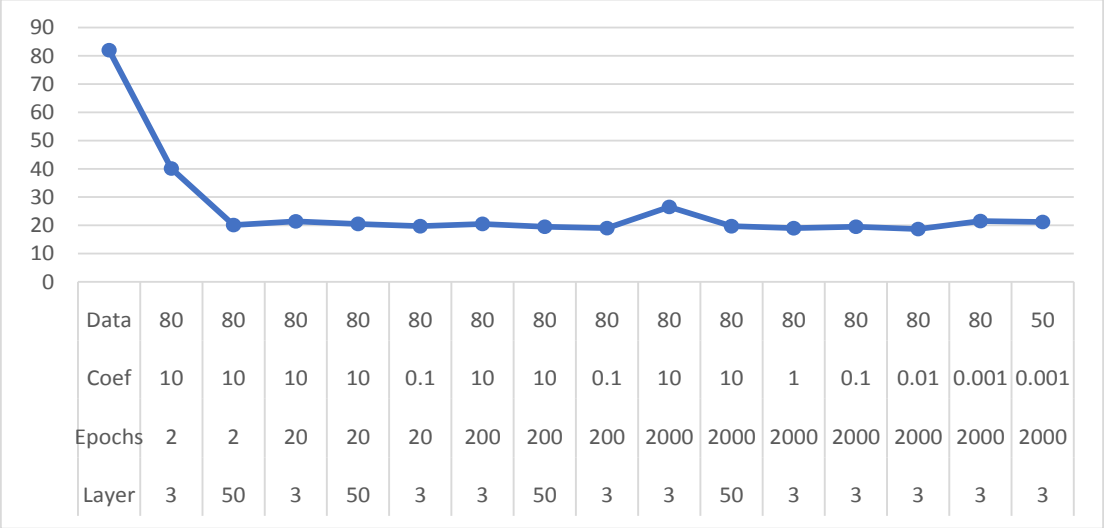

Figure 10. Dependence of MARE on various input parameters

From the graph and table above, it follows that the quality of the trained model depends primarily on the number of learning cycles and the number of hidden layers. Based on the results, we can say that the impact on the quality of the model also has the amount of selected data for its training.

Conclusions

In the course of the article, one of the approaches to training the neural network model that will be used for data prediction was considered and analyzed.

In accordance with the requirements, the following tasks were performed:

- feedforward networks were considered and analyzed, and a multilayer perceptron is selected for research;
- prepared input data for model training and testing;
- created a model using MLPRegressor and the L-BFGS optimization algorithm;
- network parameters were selected: number of hidden layers, regularization parameter, number of training cycles;
- model training was conducted;
- network testing was performed using the calculation of the mean absolute error in percentage (MARE);
- a study was conducted that included an analysis of the dependence of the accuracy of the model on the choice of various parameters and data at the training stage;
- the corresponding table and graph with results were constructed.

REFERENCES

1. Neurotechnologies and neurocomputer systems: textbook. Yampolskyi L. S., Lisovichenko O.I., Oliynyk V.V. Kyiv: Dorado-Druk, 2016. 576 p. (Ukr.)

2. Feedforward Neural Networks and Multilayer Perceptrons. URL: https://boostedml.com/2020/04/feedforward-neural-networks-and-multilayer-perceptrons.html (date of use: 26.11.2020)

3. Perceptrons and Multi-Layer Perceptrons: The Artificial Neuron at the Core of Deep Learning. URL: https://missinglink.ai/guides/neural-network-concepts/perceptrons-and-multi-layer-perceptrons-the-artificial-neuron-at-the-core-of-deep-learning/ (date of use: 26.11.2020)

4. Multilayer perceptron. URL: https://wiki.loginom.ru/articles/multilayered-perceptron.html (date of use: 27.11.2020) (Rus.)

5. MLPRegressor URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html (date of use: 27.11.2020)

6. Neural Architecture search: A Survey URL: https://www.jmlr.org/papers/volume20/18-598/18-598.pdf (date of use: 28.11.2020)

7. Random Search for Hyper-Parameter Optimization. URL: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf (date of use: 28.11.2020)

8. Limited-Memory Broyden-Fletcher-Goldfarb-Shanno. URL: https://software.intel.com/sites/products/documentation/doclib/daal/daal-user-and-reference-guides/daal_prog_guide/GUID-254D0216-3379-4733-95CA-138009F24A04.htm (date of use: 02.12.2020)

9. Basic estimates of the accuracy of time series forecasting. URL: https://www.mbureau.ru/blog/osnovnye-ocenki-tochnosti-prognozirovaniya-vremennyh-ryadov (date of use: 05.12.2020) (Rus.)