Comparison of Methods for Improving the Quality of Prediction Using Artificial Neural Networks

Kirill Uryvaev[0000−0001−5275−5024], spore9@yandex.ru
Alena Rusak[0000−0002−2803−4777], alena@cde.ifmo.ru

ITMO University, 49 Kronverksky Pr., St. Petersburg, 197101, Russia

Abstract. In this paper, the methods for improving the quality of forecasting using artificial neural networks are researched. Nowadays due automatisation of many systems, that using prediction algorithms, such as artificial neural networks, forecasting error cost are increasing, especially in expensive or dangerous areas. In problems with a small set of initial data or a large algorithmic complexity, the solution of the forecasting problem can give unsatisfactory results or lead to large amounts of computation. The speed and accuracy of prediction are of critical importance since in many real practical problems the cost of forecasting errors is extremely high. There are of methods aimed at improving the quality of training of artificial neural networks. The methods selected for this paper are not exclude each other, so they can composited without any conflict, but some methods in certain situation can worsen the result, some methods can increase accuracy, but decrease speed and vice versa. And because of that condition, methods should be compared separately and only after that composite, and result should improve, because their disadvantages compensated by advantages of others methods. The experiment showed that although individually these methods do not have a significant impact on accuracy, a combination of these approaches improves the quality of forecasting.

Keywords: Artificial neural networks · Machine learning · Forecasting · Nonlinear regression · Extrapolation.

1 Prediction task

Prediction is a non-linear and often quite difficult task, which is obviously relevant nowadays. It nds its application both in purely scientific fields (physics, chemistry, biology, etc.) and in practice (marketing, statistics, etc.). The essence of the forecasting problem is to predict the future reaction of the system according to its previous behavior. Currently, many approaches have been developed to solve this problem, among which artificial neural networks (ANN). The main

advantage of neural network models is their nonlinearity, i.e. the ability to establish non-linear relationships between future and current process values, as well as their adaptability and scalability. The disadvantages of neural network models are the opacity of the modeling process, the complexity of the choice of architecture and the complexity of artificial neural network learning.

The accuracy of forecasting is largely influenced by a careful analysis of the input data, selection of features, the ability to take into account external factors, i.e. other processes occurring in parallel with the researching process and affecting it. In problems with a small set of initial data or a large algorithmic complexity, the solution of the prediction problem can give unsatisfactory results or lead to large amounts of computation. At the same time, taking into account the ongoing automation, in systems whose functionality covers important and dangerous areas, the price of forecasting error is of critical importance. One approach to improving forecast accuracy is the use of combined models, for example, the use of ANN in combination with autoregressive models [1]. Another way to increase accuracy is to use a consensus forecast, i.e. forecast, which is a linear combination of several independent forecasts [2]. However, these approaches are rather laborious, because require the development of several forecasting models. In this work, we study the influence on the forecast accuracy of some standard methods used in the training of neural networks.

For ANN, forecasting is the task of nonlinear regression. Having information about the values of the variable x at the moments preceding the prediction of *x(k-1), x(k-2), ..., x(k-N)*, the network makes a decision on what will be the most probable value of the sequence at the current moment k. To adapt the network weighting coefficients, the actual forecasting error $\epsilon = x(k)\text{-}\bar{x}(k)$ and the values of this error at previous time instants are used [3].

Despite the advantages, neural networks also have minuses. To successfully solve the problem, the neural network needs to be trained at a accepted level, which requires sufficient availability of data, time and computing resources. But even under such conditions, training may be wrong, for example, a network may overfit. This means that the network adapts too much to the answers in the training set, and all other values are very likely to be incorrectly predicted.

But existing a methods that can help smooth the artificial networks problems. Although there are many methods to improve the results of ANNs, not all of them are universal, and what works well, for example, to classify images, may not work at all for predicting time series. Therefore, it is necessary first to consider the effectiveness of these methods in general, and then for the forecasting problem.

2 Methods for improving prediction quality

The learning outcomes are greatly influenced by the selection of the initial values of the network weights. The wrong choice of a range of random values of weights can lead to an excessive slowdown in the learning process. **Xavier's initialization** (also known as glorot) is to generate random initial bond weights based on the number of input and output links of a given neuron. This method allows

you to accelerate the training of ANN [4] and improves the quality of prediction because, with several training iterations, the ANN does not fall into the same local minimum. The initial bond weight is calculated by (1).

$$W_i \; U(-\frac{\sqrt{6}}{\sqrt{Count_{in} + Count_{out}}}, \frac{\sqrt{6}}{\sqrt{Count_{in} + Count_{out}}}) \tag{1}$$

where W_i is the neuron connection weight, U is the uniform distribution, $Count_{in}$ is the number of neuron inputs, $Count_{out}$ is the number of neuron outputs.

Data shuffle is a random arrangement of data sets in the training and test samples. This method avoids the dependence of the result on the sequence of data. However, this technique is not always suitable for forecasting problems [5], since in such problems the data often are sequence-dependent.

To prevent overfitting when building a neural network, various **regularization** methods are used. One approach is to introduce an additional term (regularizer) in the objective function, which does not allow the scales to acquire very large values. The regulator may have the following form (2).

$$regularization = \frac{\lambda}{2m} \sum_{j=1}^{n} W_j^2 \tag{2}$$

where m is the sample size, λ is the regularization coefficient, n is the number of neuron connections, W_j is the weight of the connection.

In the theory of neural networks, such a regularization is called weight decay, because it leads to a decrease in their absolute values. There is another regularization method called "dropout". Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets. [6].

Using an **adaptive learning coefficient** allows you to vary the learning step depending on the situation. To change the coefficient, you must have a criterion, the simplest and most obvious is the use of the cost function. This implies that when the network starts to diverge, you can try to reduce the step to achieve a more accurate minimum value [7]. Or vice versa, you can increase the step if, for example, we are sure that the network has hit the local minimum, and it should look for a solution further.

Currently, adaptive optimization methods are widely used to configure neural networks. The following training algorithms were compared: Adadelta, Adagrad, Adam, Adagrad, Adagrad with low LR, Adamax, RMSprop, SGD (see Fig. 1).

This result was obtained by using ResNet50 with pre-trained weights of imagenet. The input shape is (224, 224, 3). As can be seen, Adamax showed the best result, but this study was conducted on a large scale of data, and Nadam was not researched because of memory problems with it.

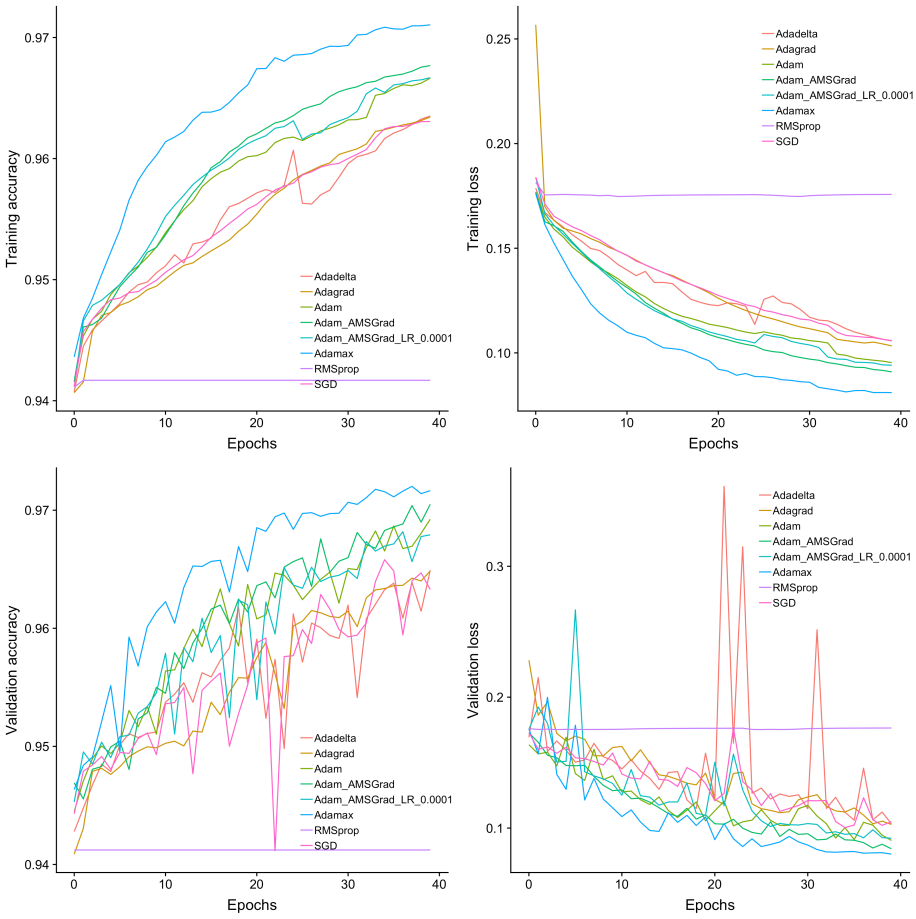


Fig. 1: Comparison of adaptive learning coefficient optimization methods.

The pruning method consists in equating to zero the most uninformative relations with relative to other relations. This method is more controversial because it often worsens the result because can nullify important connections, but at the same time in large networks, it can significantly simplify calculations [8]. Also, to implement pruning, it is necessary to use sparse matrices, i.e. matrices with lots of zeros.

Sparse matrix is a matrix in which most of the elements are zero. Sparse matrices, by itself, without pruning, can improve the results of the ANN [9]. On small networks, they accelerate learning, and on large networks they allow you to make calculations faster [10] (see Fig. 2). Large sparse matrices are common

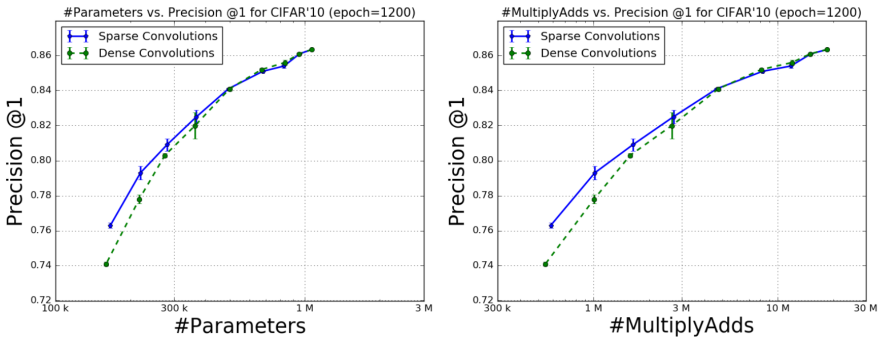


Fig. 2: Comparison of sparse and dense matrices [10].

in datasets that contains counts, data encodings that map categories to counts, and even in whole subfields of natural language processing.

It is computationally expensive to represent and work with sparse matrices as though they are dense, and much improvement in performance can be achieved by using representations and operations that specifically handle the matrix sparsity.

3 Testing the methods

To check the described methods, tests were performed on the Building problem from the Proben1 set [11]. The set contains data for predicting the hourly consumption of electricity, hot and cold water based on the date, time of day and weather data. Complete hourly data for four consecutive months is used for training, and output data for the following two months should be predicted. In all there are 14 inputs and 3 outputs. To implement it, the Keras framework on TensorFlow was selected and ANN architecture is: 2 dense layers, first with 64 neurons, second with 32 neurons. Mean absolute percentage error (MAPE) (3) will be used as accuracy

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{3}$$

where A_t is actual value, F_t is forecast value, n is values set size.

To evaluate the quality of the forecast, the determination coefficient R2 (4) are used, which is considered as a universal measure of the dependence of one

random variable on many others. It takes values from 0 to 1, the closer the coefficient is to 1, the stronger the dependence. When evaluating regression models, this is interpreted as matching the model with data.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{4}$$

where SS_reg (5) is the sum of explained squares , SS_tot (6) is the total sum of squares.

$$SS_{res} = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{5}$$

where y_i is actual value, $\bar{y}$ is the mean value.

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \hat{y})^2 \tag{6}$$

where y_i is actual value, $\hat{y}$ is the predicted value. After training and solving the problem on the test sample, the next results were obtained (Table 1).

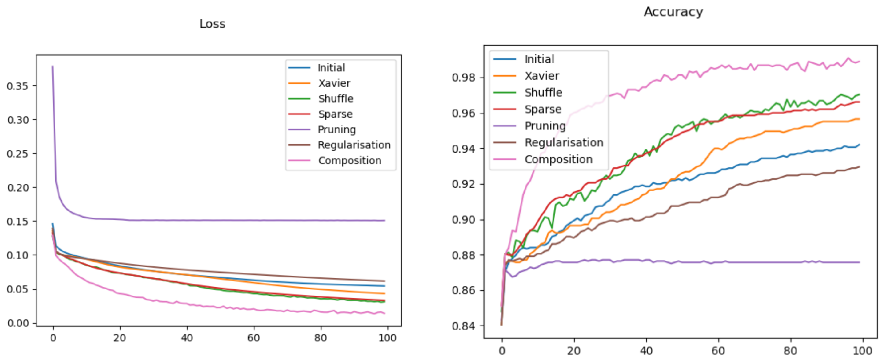
Table 1: Results of Building problem solving.

ANN type	Loss function	Accuracy (MAPE)	Determination coef.
Initial ANN	0.0542	0.9421	0.7006
Xavier initialization	0.044	0.9566	0.7501
Regularization	0.0632	0.9297	0.6953
Data shuffle	0.0319	0.9704	0.7871
Pruning	0.1516	0.8761	0.5685
Sparse Matrices	0.0337	0.9661	0.8348
Method Composition	0.0138	0.9887	0.8843

In benchmark they obtained result of 0.92 on validation set, which is lower than all results except of pruning. This can be explained by the fact that the task was solved by old artificial neural network, so even the inital Keras ANN did it better. Also it took for their neural network about 400 epoch to learn, while our ANN has learned only with 100 epochs.

For comparison, the forecasting problem with the help of ANN was solved without using methods for improving the quality of forecasting. (Fig. 3a) shows a graph of the learning speed of all networks, and (Fig. 3b) the accuracy of this networks. The problem was solved with fairly good accuracy, but the determination coefficient was only 0.7.

In the researching problem were not a big set of zeros, i.e. the matrices were not sparse, therefore, they did not have a big impact on the result, but at the same time, the calculations themselves were reduced, albeit slightly (due to the small size of the problem), 159.546 seconds versus 157.486 seconds. In this task,



(a) (b)

Fig. 3: Methods comparison (a) learn curve (b) accuracy

the pruning worsened the result, because there were not a large number of zeros, and the task was generally not large, but now the model can be compressed without losses due to zeros, the array of which is easy to compress.

Applying the composition of all methods except for pruning, the best result was obtained: the forecasting accuracy increased by about 5.6%, while the determination coefficient was 0.8843.

Thus, the study showed that despite the fact that separately the standard methods used for tuning neural networks did not show a significant increase in the accuracy of forecasting, the use of a combination of these methods allowed us to achieve a fairly high forecast accuracy (97.26%). However, when solving problems using ANN, it is necessary to take into account its specificity and, on the basis of this, select more controversial methods such as pruning.

References

1. Bunnoon, P., Chalermyanont, K., Limsakul, C.: A Computing Model of Artificial Intelligent Approaches to Mid-term Load Forecasting: a state-of-the-art-survey for the researcher. IACSIT International Journal of Engineering and Technology, (2010)
2. Olivia, R., Watson, N.: Managing Functional Biases in Organizational Forecasts: A Case Study of Consensus Forecasting in Supply Chain Planning. Production and Operations Management Society, 138-151 (2009)
3. Osovsky S.: Neural networks for information processing. 2nd edn. Finance and Statistics, Moscow (2017)
4. LeCun Y., Bottou L., Orr G., Muller K.: E Efficient BackProp. Neural Networks: Tricks of the trade, Springer , 9-48 (1998)
5. How to Backtest Machine Learning Models for Time Series Forecasting, https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting. Last accessed 14 Oct 2019

6. Hinton G., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R.: Improving neural networks by preventing co-adaptation of feature detectors. (2012)
7. Moreira M., Fiesler E.: Neural Networks with Adaptive Learning Rate and Momentum Terms. IDIAP Technical report , (1995)
8. Pruning deep neural networks to make them fast and small, https://jacobgil.github.io/deeplearning/pruning-deep-learning. Last accessed 19 Oct 2019
9. Sparse Matrices For Efficient Machine Learning, https://dziganto.github.io/Sparse-Matrices-For-Efficient-Machine-Learning/. Last accessed 23 Oct 2019
10. Changpinyo S., Sandler M., Zhmoginov A.: The Power of Sparsity in Convolutional Neural Networks. ICLR 2017 , (2017)
11. Prechelt L.: Proben1: A Set of Neural Network Benchmark Problems and Benchmarking Rules. In: Technical Report 21, Karlsruhe, Germany (1994)