Numerical conditioning of neural network components in time series forecasting

Stanislav Sholtanyuk
the Department of Computer Applications and Systems
Faculty of Applied Mathematics and Computer Science
Belarusian State University
Minsk, Belarus
SSholtanyuk@bsu.by

***Abstract*—In this paper, an issue of stability of fully connected neural network (perceptron), forecasting time series with sliding window method, is considered. A task of finding conditions, providing best results on accuracy and stability, was set. To accomplish it, training of the neural network with various sets of hyperparameters, such as amount of neurons on input and hidden layers, as well as activation function, has been performed. Training was performed on modeled data, which have been perturbed with unbiased normally distributed random variables with various variance. Such training results as number of conditioning of weight matrix between input and hidden layers, number of conditioning of activation functions, as well as statistics on retrospective forecasting absolut error, have been shown. An approach for estimation of condition number for some activation functions has been proposed. Impact of activation function on accuracy and stability of forecasting perceptron has been shown.**

***Keywords*—time series forecasting, neural network, perceptron, numerical conditioning, activation function**

I. Introduction

Artificial neural networks are widely used in various tasks, for example, pattern classification, speech recognition, decision making. Neural networks have also proved themselves as powerful tool for time series forecasting [1], [2]. There are plenty implementations and libraries on various programming languages, which can build neural networks with given architecture.

One of the modern researchers' focus is on stability and robustness of neural networks. There are different approaches for understanding and considering of neural networks stability. For example, dynamical stability (low sensitivity of the result to perturbations of input data) and structural stability (insignificant error of output in case of changes in numerical characteristics of the neural network itself) are distinguished [3]. The most vivid examples of neural networks instability appear in pattern recognition with deep convolutional networks. When training on certain dataset, these networks become unable to classificate objects on noisy images, which are indistinguishable from a human's point of view, but are completely different from a point of view of neural nets [4], [5]. Techniques, using such vulnerabilities of intelligence systems, called adversarial attacks. Connection between existence of adversarial examples and conditioning of weight matrices in neural networks has been revealed and systemically considered in [6]. In [7], it's also argued that stability of neural network training is an analogue of identification whether specified matrix is well- or ill-conditioned, besides, a measure of stability of neural networks training has been proposed for various applications and tasks in biometrics.

The aim of this paper is revealing the factors, which have impact on dynamical stability of neural networks in forecasting time series. For this, an issue of conditioning of neural nets and their components, calculation of condition numbers and affecting these indicators on neural network performance have been considered. Comparative analysis of neural networks with different hyperparameters sets has been performed as well.

II. Architecture of the forecasting perceptron and its training

The object of study of this article is three-layered perceptron with the following architecture:

- input layer contains arbitrary amount of neurons p. p sequential values of standardized time series are given to these p neurons;
- a hidden layer with arbitrary amount of neurons n;
- output layer contains the only neuron, where a forecasted value for the next, $(p+1)$th value, is received;
- input and hidden layers also contain bias neurons;
- the neural network is fully connected, i.e. all neurons on every layer connected with all neurons on the next layer.

Forecasting perceptron works with standardized time series, which is calculated by the following formula:

$$\tilde{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}},\ i = \overline{1, N}, \tag{1}$$

where x_i is i-th value of the initial time series, $N \in \mathbb{N}$ – length of the time series, $x_{\min}$ and $x_{\max}$ – minimal and maximal values of the time series respectively. Values of the standardized time series $\{\tilde{x}_i\}_{i=1}^{N}$ lie between 0 and

1. To obtain final result of forecasting, the output value should be processed with inverse transformation:

$$y = \tilde{y}(x_{\max} - x_{\min}) + x_{\min}. \tag{2}$$

On every neuron of the hidden layer, an activation function is acting. In this article the following activation functions are considered:

- rectifier linear unit (ReLU):

$$\mathrm{ReLU}(x) = \max(0, x); \tag{3}$$

- leaky rectifier linear unit (LeakyReLU, or LReLU):

$$\mathrm{LeakyReLU}(x) = \begin{cases} \alpha x, x < 0, \\ x, x \geq 0, \end{cases} \tag{4}$$

  where $\alpha << 1$ - parameter, preventing from vanishing gradient problem [8];
- softplus, which can also decrease the affect of vanishing gradient problem [9]:

$$\mathrm{soltplus}(x) = \ln(1 + e^x). \tag{5}$$

The perceptron is trained on the dataset, being obtained with sliding window method, i.e. vectors $\vec{x}^j = (\tilde{x}_j, \tilde{x}_{j+1}, ..., \tilde{x}_{j+p-1})$ with their expected outputs $\tilde{x}_{j+p},\ j = \overline{1, N-p}$ represent the training set. After the training, predicting of future values of considered time series is possible, for example, one can give vector $(\tilde{x}_{N-p+1}, \tilde{x}_{N-p+2}, ..., \tilde{x}_N)$ to the input of the neural net in order to obtain an estimation for $\tilde{x}_{N+1}$.

III. Conditioning of the neural network

The forecasting perceptron can be considered as a vector operator $\mathfrak{N} : \mathbb{R}_{p,1} \to \mathbb{R}$. One of the measures of vector and matrix operators stability is number of conditioning. For a non-degenerate operator A can be calculated as follows:

$$\mathrm{cond}\, A = \|A\| \left\|A^{-1}\right\|, \tag{6}$$

where $\|\cdot\|$ is operator norm, induced by Euclidean vector norm $\|\vec{x}\|_2 = \sqrt{\sum\limits_{i=1}^{p} x_i^2}$, i.e.

$$\|A\| = \sup_{\|\vec{x}\| \neq 0} \frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2}. \tag{7}$$

Operator $\mathfrak{N}$ can be decomposed into the following operators $\mathfrak{N}_i,\ i = \overline{1, 7}$:

- standardizing of the initial vector $\vec{a} \in \mathbb{R}_{p,1}$ with formula (1):

$$\mathfrak{N}_1 : \mathbb{R}_{p,1} \to \mathbb{R}_{p,1},$$
$$\mathfrak{N}_1(\vec{a}) = \frac{1}{x_{\max} - x_{\min}}\vec{a} - \frac{x_{\min}}{x_{\max} - x_{\min}}\vec{e}, \tag{8}$$

  where $\vec{e} = \underbrace{(1, 1, ..., 1)}_{p}^T$;
- left multiplicating received vector by weight matrix $A_1 \in \mathbb{R}_{n,p}$, whose element on i-th row, j-th column is equal to weight of the synapse, connecting i-th neuron on the input layer with j-th neuron on the hidden layer:

$$\mathfrak{N}_2 : \mathbb{R}_{p,1} \to \mathbb{R}_{n,1},\ \mathfrak{N}_2(\vec{a}) = A_1\vec{a};$$

- adding received vector to bias vector $\vec{b}^1 \in \mathbb{R}_{n,1}$, whose i-th component is equal to weight of the synapse, connecting i-th bias neuron on the input layer with i-th regular neuron on the hidden layer:

$$\mathfrak{N}_3 : \mathbb{R}_{n,1} \to \mathbb{R}_{n,1},\ \mathfrak{N}_3(\vec{a}) = \vec{a} + \vec{b}^1;$$

- activation function, acting on every neuron of the hidden layer:

$$\mathfrak{N}_4 : \mathbb{R}_{n,1} \to \mathbb{R}_{n,1},\ \mathfrak{N}_4(\vec{a}) = (\mathrm{act}(a_1), ..., \mathrm{act}(a_p))^T,$$

  where act – one of the activation functions (3)-(5);
- left multiplicating received vector by weight matrix $A_2 \in \mathbb{R}_{1,n}$, whose element on i-th row is equal to weight of the synapse, connecting i-th neuron on the hidden layer with the neuron on the output layer:

$$\mathfrak{N}_5 : \mathbb{R}_{n,1} \to \mathbb{R},\ \mathfrak{N}_5(\vec{a}) = A_2\vec{a};$$

- adding to weight b of the synapse between the bias neuron on hidden layer and the neuron on the output layer:

$$\mathfrak{N}_6 : \mathbb{R} \to \mathbb{R},\ \mathfrak{N}_6(a) = a + b;$$

- inverse of the standardizing transformation with formula (2):

$$\mathfrak{N}_7 : \mathbb{R} \to \mathbb{R},\ \mathfrak{N}_7(a) = a(x_{\max} - x_{\min}) + x_{\min}.$$

Number of conditioning of the operator $\mathfrak{N}$ can be estimated from above with multiplication of condition numbers of $\mathfrak{N}_i,\ i = \overline{1, 7}$ according to submultiplicativity of the norm (7):

$$\mathrm{cond}\, \mathfrak{N} \leq \prod_{i=1}^{7} \mathrm{cond}\, \mathfrak{N}_i.$$

Thus, conditioning of the whole neural network depends on conditioning of its components, such as weight matrices, activation functions, as well as on preprocessing data techniques, being used before training. By estimating of condition numbers for these operators, we can judge conditioning of the neural network in general.

Let us estimate condition numbers of the operators $\mathfrak{N}_i,\ i = \overline{1, 7}$. Initially, it can be proved, that $\mathrm{cond}\, \mathfrak{N}_1 = 1$. For proving, the following formula will be used [10]:

$$\mathrm{cond}\, f = \sup_{\|\vec{a}\| \neq 0} \frac{\|J\|}{\|f(\vec{a})\|/\|\vec{a}\|}, \tag{9}$$

where J is Jacobian matrix for function f, and matrix norm $\|J\|$ is induced by the vector norm from the same

Table I
STABILITY INDICATORS OF THE FORECASTING PERCEPTRON IN DEPENDANCE ON ACTIVATION FUNCTION AND STANDARD DEVIATION OF PERTURBATIONS IN INPUT DATA

Standard deviation of perturbations	**Mean estimation of condition number of A_1**			**Mean estimation of condition number of activation function**			**Mean of norm of absolute forecasting error w.r.t. the input series**			**Standard deviation of norm of absolute forecasting error w.r.t. the input series**		
	ReLU	**LReLU**	**sp**	**ReLU**	**LReLU**	**sp**	**ReLU**	**LReLU**	**sp**	**ReLU**	**LReLU**	**sp**
0	1.58	1.70	1.75	2.08	1.56	0.37	588	602	749	237	191	322
1	1.63	1.67	1.71	2.13	1.52	0.37	582	607	786	215	252	474
5	1.65	1.65	1.70	2.00	1.54	0.37	647	643	787	235	239	351
10	1.66	1.67	1.71	1.94	1.50	0.37	692	743	927	202	250	482
50	1.66	1.70	1.70	1.95	1.56	0.37	1975	2018	2398	231	347	733
100	1.65	1.74	1.71	1.94	1.55	0.37	3583	3623	4116	266	302	916
500	1.63	1.79	1.74	1.86	1.51	0.35	16915	17052	18236	623	751	2416

formula. For function from formula (8), matrix J is equal to

$$\begin{pmatrix} \frac{1}{x_{\max}-x_{\min}} & 0 & \cdots & 0 \\ 0 & \frac{1}{x_{\max}-x_{\min}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{x_{\max}-x_{\min}} \end{pmatrix},$$

and its norm is $\frac{1}{x_{\max}-x_{\min}}$. After substitution into (9) we got

$$\operatorname{cond}\mathfrak{N}_1 = \sup_{\|\vec{a}\|\neq 0} \frac{\|J\|}{\|\mathfrak{N}_1(\vec{a})\|/\|\vec{a}\|} = \frac{1}{x_{\max}-x_{\min}} \times$$
$$\times \sup_{\|\vec{a}\|\neq 0} \frac{\|\vec{a}\|}{\|\mathfrak{N}_1(\vec{a})\|} = \frac{1}{x_{\max}-x_{\min}} \left\|\mathfrak{N}_1^{-1}\right\|. \quad (10)$$

Similary, for operator $\mathfrak{N}_1^{-1}(\vec{a}) = (x_{\max}-x_{\min})\vec{a} + x_{\min}$ we got

$$\operatorname{cond}\mathfrak{N}_1^{-1} = (x_{\max}-x_{\min}) \|\mathfrak{N}_1\|. \quad (11)$$

Multiplication of (10) and (11) with taking into account (6) leads to the identity $\|\mathfrak{N}_1\| \left\|\mathfrak{N}_1^{-1}\right\| = 1$, whence follows that condition number of operator $\mathfrak{N}_1$ is equal to 1. $\operatorname{cond}\mathfrak{N}_i = 1$ for $i = 3, 6, 7$ is proved in the same way.

For operators $\mathfrak{N}_2$, $\mathfrak{N}_5$ condition numbers are equal to condition numbers of matrices A_1 and A_2 respectively, besides, $condA_2 = 1$, as far as the matrix $A_2 \in \mathbb{R}_{1,n}$ has only one singular number, i.e. such ς that $\exists \vec{u}, \vec{v}$: $A_2\vec{v} = \varsigma\vec{u} \wedge A_2^T\vec{u} = \varsigma\vec{v}$. Estimation of condition number for matrix A_1 was performed after forecasting perceptron training with various hyperparameters and training data.

As regards operator $\mathfrak{N}_4$, its Jacobian matrix is diagonal, hence, the matrix norm, induced by Euclidean vector norm, is equal to maximum among absolute values of diagonal elements. For functions ReLU and LeakyReLU $\|J\| = 1$, because $\mathrm{ReLU}'(x) = \mathrm{LeakyReLU}'(x) = 1$ for positive x, and $\mathrm{ReLU}'(x) = 0$, $\mathrm{LeakyReLU}'(x) = \alpha << 1$ for negative x. In case of using softplus function the following estimation is taking place:

$$\|J\| \leq \sup_x \mathrm{softplus}'(x) =$$
$$= \sup_x (\ln(1+e^x))' = \sup_x \frac{e^x}{1+e^x} = 1.$$

By using (9), we got

$$\operatorname{cond}\mathfrak{N}_4 = \sup_{\|\vec{a}\|\neq 0} \frac{\|J\|}{\|\mathfrak{N}_4(\vec{a})\|/\|\vec{a}\|} \leq$$
$$\leq \sup_{\|\vec{a}\|\neq 0} \frac{\|\vec{a}\|}{\|\mathfrak{N}_4(\vec{a})\|}. \quad (12)$$

IV. COMPUTATIONAL EXPERIMENT AND ITS RESULTS

Testing of the forecasting perceptron was performed on the examples of time series $\vec{x} = (x_1, x_2, ..., x_N)$, modeled with the following formula:

$$x_t = t^\alpha + \beta \sin \gamma t, \ t = \overline{1, N}.$$

Time series $\vec{x}^j = \vec{x} + \vec{\xi}^j$, $j = \overline{1, Q}$, where $\vec{\xi}^j = (\xi_1^j, \xi_2^j, ..., \xi_N^j)$, $\xi_t^j \in N(0, \sigma_j^2)$ are random variables with unbiased normal distribution with variance σ_j^2, were also considered. Training of the forecasting perceptron was carried out with various values of p, n, as well as various activation functions (3)-(5) for 7 epochs. An optimization algorithm, called AdaGrad (adaptive gradient algorithm) [11], which has shown the best results on stability of the neural network [12], was used. Condition numbers of activation functions were estimated according to formula (12), where components of the vector $\vec{a}$ are equal to values on inputs of neurons on the hidden layer, and vector $\mathfrak{N}_4(\vec{a})$ consists of values outputs of the same neurons.

During the experiment neural networks with different architectures have shown approximately equal results on stability and conditioning. Forecasting perceptron performance for $N = 1000$, $\alpha = 1.2$, $\beta = 100$, $\gamma = 0.05$, $Q = 60$ in dependance on standard deviations σ_j and an activation function is shown on the I. As

this table shows, when using activation function ReLU, matrix A_1 has lower condition number, then with other considred activation functions. What about LeakyReLU and softplus, the first function gives lower condition number of A_1, when perturbations in input data have standard deviation σ, not exceeding 10, and the second one – when $\sigma \geq 100$. In case $\sigma = 50$ both functions give approximately equal condition numbers of A_1. On the contrary, for the operator $\mathfrak{N}_4$, ReLU gives the highest condition number, and softplus gives the lowest one, moreover, when using ReLU, increasing of standard deviation of perturbations in input data leads to condition number decreasing, which can be explained with the aforementioned issue of vanishing gradient. The best results on accuracy and stability have been obtained with using ReLU function.

V. Conclusions and further research

Results of the forecasting perceptron training evidence that ReLU function is the most suitable as activation function in the neural network. Compared with LeakyReLU and softplus functions, ReLU has given better conditioning of weight matrix, as well as better accuracy and stability of retrospective time series forecasting. These results, however, don't correlate neither matrix A_1 conditioning nor activation functions conditioning. Thus, accuracy and stability of the forecasting perceptron don't directly depend on conditioning of its components.

In further researches, stability of the forecasting perceptron in sense of numerical conditioning will be considered. For exploring this problem, theoretical estimation of condition number of activation functions is necessary, as well as considering of factors, affecting on weight matrices and their conditioning, for example, weight initialization methods.

References

[1] L. S. Maciel, R. Ballini. Design a Neural Network for Time Series Financial Forecasting: Accuracy and Robustness Analysis. *Anales do 9º Encontro Brasileiro de Finanças*, 2008. Availaible at: https://www.cse.unr.edu/~harryt/CS773C/Project/895-1697-1-PB.pdf (accessed 2019, December)

[2] L. Falat, Z. Stanikova, M. Durisova, B. Holkova, T. Potkanova. Application of Neural Network Models in Modelling Economic Time Series with Non-Constant Volatility. *Business Economics and Management Conference*, 2015, pp. 600-607

[3] B. Sengupta, K. J. Friston. How Robust are Deep Neural Networks? *arXiv preprint arXiv:1804.11313*, 2018.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, R. Fergus. Intriguing properties of neural networks. *International Conference on Learning Representations, arXiv:1312.6199*, 2014

[5] I. J. Goodfellow, J. Shlens, C. Szegedy. Explaining and Harnessing Adversarial Examples. *International Conference on Learning Representations, arXiv:1412.6572*, 2015.

[6] M. Singh, A. Sinha, B. Krishnamurthy. Neural Networks in Adversarial Setting and Ill-Conditioned Weight Space. *arXiv preprint arXiv:1801.00905*, 2018.

[7] S. V. Kachalin. Otsenka Ustoichivosti Algoritmov Obucheniya Bol"shikh Iskusstvennykh Neironnykh Setei Biometricheskikh Prilozhenii [Assessment of Stability Learning Algorithms Large Artificial Neural Networks of Biometric Application]. *Vestnik SibGAU* [*SibSAU Bulletin*], 2014, no. 3, pp. 68-72.

[8] A. L. Maas, A. Y. Hannun, A. Y. Ng. Rectifer Nonlinearities Improve Neural Network Acoustic Models. *International Conference on Machine Learning*, 2013.

[9] H. Zheng, Z. Yang, W.-J. Liu, J. Liang, Y. Li. Improving Deep Neural Networks Using Softplus Units. International Joint Conference on Neural Networks, 2015. Available at https://www.researchgate.net/publication/30886346 (accessed 2019, December).

[10] L. N. Trefethen, D. Bau. *Numerical Linear Algebra*, Philadelphia, Society for Industrial and Applied Mathematics, 1997, 390 p.

[11] J. Duchi, E. Hazan, Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 2011, vol. 12, pp. 2121-2159.

[12] S. Sholtanyuk. Sravnitel'nyi analiz neirosetevoi i regressionnykh modelei prognozirovaniya vremennykh ryadov [Comparative Analysis of Neural Networking and Regression Models for Time Series Forecasting]. *Tsifrovaya transformatsiya* [*Digital Transformation*], 2019, no. 2, pp. 60-68.

Обусловленность компонентов нейронной сети в задачах прогнозирования временных рядов

Шолтанюк С.В.

В данной работе рассмотрен вопрос устойчивости полносвязной нейронной сети (персептрона), прогнозирующей временные ряды методом скользящего окна. Поставлена задача нахождения условий, обеспечивающих лучшие результаты по точности и устойчивости, для чего было проведено обучение нейронной сети при различных наборах гиперпараметров, таких как количество нейронов на входном и скрытом слоях, а также функция активации. Обучение проводилось для смоделированных данных, которые подвергались зашумлению несмещёнными нормально распределёнными случайными величинами с различной дисперсией. Представлены такие результаты обучения нейронной сети, как число обусловленности матрицы весов между входным и скрытым слоями, число обусловленности функций активации, а также статистические характеристики абсолютной погрешности ретроспективного прогноза. Предложен подход к оцениванию числа обусловленности некоторых функций активации, использующихся при решении задач прогнозирования. Показано влияние выбора функции активации на точность и устойчивость прогностического персептрона.

Received 29.12.19