Peculiarities of regression model design based on neural networks

N Yu Grigoryeva¹, A S Perkov², T R Zhangirov² and A A Liss²

¹Saint-Petersburg Federal Research Center of Russian Academy of Sciences, Scientific Research Center for Ecological Safety, ul. Korpusnaya, 18 197110 Saint-Petersburg, Russia
²Saint-Petersburg Electrotechnical University “LETI”, ul. Prof. Popov, 5 197376 Saint-Petersburg, Russia

E-mail: renes3@mail.ru

Abstract. In this article a neural-network regression model for prediction of the bacterioplankton abundance according to physicochemical parameters of the environmental conditions is considered and some of the peculiarities of its development are described. A particular case of small and very heterogeneous data sample, typical for biological applications, is analysed. To solve this problem, a number of multi-layer feed-forward neural networks with different architectures are studied. The regression results are estimated on the base of determination coefficient and standard deviation of predicted values in the test sample. The effect of the dropout, applied to one of the hidden layers, on learning process and obtained results is analysed.

1. Introduction
The regression problem is one of the most common issue in the field of machine learning. The solution of this problem is reduced to the determination of a certain function associating a set of independent variables (features) with a set of dependent variables (output parameters). This function is also called a regression model. All regression models can be divided into two groups according to the type of the dependence between features and output parameters: linear and non-linear models. Actually, linear relationships in the real world are much less common than non-linear ones. More frequently the analyzed data have a non-linear behaviour. Therefore, the non-linear regression models are more relevant for many subject areas.

One of the main problems in a non-linear regression model elaboration is to determine the type of non-linearity. In the case when the type of non-linearity is beforehand known, e.g. natural exponential or simple exponential dependence, the least squares method is used, as in the case of linear regression. When the type of non-linearity is uncertain, the solution of the regression problem includes not only the derivation of a specific functional dependence, but also the establishing of a non-linearity type. In recent decades, artificial neural networks (ANNs) are often used to elaborate such regression models, that allow to change the type and degree of non-linearity by adjusting the network architecture. In addition, a specific ANN training algorithm enables to find a unique relationship between dependent and independent variables for a particular data set.

A lot of research works are devoted to the estimation of ANN effectiveness in solving regression problems [1–3]. The authors of [4] present a solution of regression problem for determination of phytoplankton in an optically complex coastal region. During this study, ANNs were compared with
other regression models and it was established that according to determination coefficient the ANNs give better solution with respect to other methods. Similar results were demonstrated in [5], concerning the prediction of the soil enzymes activity based on the geographical features of considered area. Moreover, in this paper [5] is shown, that multiple linear regression can simply ignore a set of some features, thereby it produce a serious prediction error. Several other examples of ANN utilization for solving regression problems in biology were presented in [6, 7].

In the presented work, ANN regression model is used to reveal the relation between physicochemical environmental parameters and bacterioplankton abundance on littoral of the Kola bay (The Barents sea). The difficulties in obtaining right ANN model owing to a small sample size are emphasized. The dependence of the results on ANN architecture and training parameters is also investigated.

2. Materials and methods

2.1 Data Description

Input data for the regression model is the experimental observations obtained at three biological scientific stations on southern and middle knee of the Kola bay littoral (The Barents sea). Data was collected every month from September, 2012 to September, 2013 – the total sample consists of 36 observations [8]. The input data include physicochemical parameters of the water (temperature, salinity, pH, dissolved oxygen, oxygen saturation of the water, biochemical oxygen consumption, ammonium nitrogen, nitrite nitrogen, nitrate nitrogen, phosphates), as well as the target bacterioplankton abundance to be predicted. In table 1 the statistical data parameters are presented.

| Parameters                                      | Min value | Max value | Mean value | Standard deviation |
|------------------------------------------------|-----------|-----------|------------|--------------------|
| Temperature, °C                                 | -2        | 15        | 5.48       | 4.85               |
| Salinity, %                                     | 2         | 34        | 18.72      | 9.04               |
| pH                                              | 7.03      | 8.35      | 7.67       | 0.32               |
| Dissolved oxygen, mg/l                          | 7.02      | 16.08     | 9.61       | 2.59               |
| Oxygen saturation of the water, %               | 51.3      | 118.9     | 75.39      | 18.7               |
| Biochemical oxygen consumption, mgO₂/l          | 1.1       | 6.9       | 2.61       | 1.36               |
| Ammonia nitrogen, mcgNH₄⁺/l                     | 18.2      | 543.2     | 158.53     | 139.21             |
| Nitrite nitrogen, mcgNO₂⁻/l                     | 1.15      | 5.23      | 2.03       | 0.98               |
| Nitrate nitrogen, mcgNO₃⁻/l                     | 21.1      | 90        | 42.81      | 17.75              |
| Phosphates, mcgPO₄³⁻/l                          | 6.9       | 69.8      | 40.38      | 22.11              |
| The total number of bacterioplankton, cells/ml  | 0.3       | 3.02      | 1.42       | 0.66               |

It is obvious, that to build a correct model input data should be normalized, since some of them differ by an order of magnitude. In this work, linear normalization was carried out to bring the input and output values to the range [0, 1]. It was revealed that a number of features (salinity, pH, biochemical oxygen consumption, ammonium nitrogen, nitrate nitrogen, phosphates) at different stations have significant differences in the mean value and standard deviation, but their behaviour during a year is similar. As a result, it was concluded that since the temporal behaviour carries more information than the absolute value for this group of features, the independent normalization for each station is possible. After such normalization procedure, the features became comparable in terms of mean values and standard deviation and their normalized values were fitting in the range [0, 1].
2.2. Artificial Neural Network

Artificial neural network (ANN) is a mathematical model consisting of a combination of simple computational elements – neurons, which allows non-linear data conversion [3]. A feed-forward neural network with a number of input neurons, corresponding to input features, and with one output neuron was used to build a multiple regression model with one dependent parameter (figure 1).

![Neural network architecture](image_url)

**Figure 1.** Neural network architecture for multiple regression model: (a) an example of a four-layer ANN with two hidden layers, (b) the same network with a dropout layer after the first hidden layer.

The quality of elaborated ANN model is affected by two factors: the model’s hyperparameters and the parameters of learning algorithm. Hyperparameters of the model define the architecture of ANN and determine the way of input data transformation to output data. In the case of the regression problem, hyperparameters determine the type of non-linearity.

During this investigation several ANN architectures were considered. Networks with one, two, and three hidden layers were studied. The number of neurons on hidden layers was varied, since the number of neurons determines the number of hidden factors, which can be distinguished in the input data. Among four-layer networks, the models with 5 till 12 neurons on each hidden layer (excluding a bias neuron) were studied. The right selection of activation function is also very important during ANN model design, since this function determines the character of data conversion. In this work, ANN models with activation functions typical for regression problem were investigated. Usually at the hidden layers a hyperbolic tangent function (tanh) with the range from $-1$ to $1$ is used as an activation function, and at the output layer a linear function is recommended. But since the output values in the considered problem lay in the interval $[0, 1]$, the activation function ReLU, which turns all negative values to 0, was involved initially on the output layer. However, later it was revealed that such zeroing of negative values affects the learning process incorrectly, namely, in some cases, ANN stops learning and for all observations of the dataset gives a zero result. To fix this problem, the activation function at the output layer was replaced with ELU, which has a non-zero gradient for negative values.

The weights initialization, i.e. the distribution law, by which the initial weights are generated, is the most important issue in an ANN design. During this study the uniform and normal distribution laws with various parameters were considered for weights initialization. At the same time we took into account, that our main goal is not just to build the best model, but also to find an optimal conditions for initial weights initialization, under which the learning process for ANNs with different architectures will lead to consistently good result. Finally, it was found, that different models with the same architecture are trained most stably, when the normal distribution law with a standard deviation of 0.65 and mean value 0 for initializing weights on the basic neurons was used, and the uniform distribution in the range
[−1.5, 1.5] for a bias neurons at the hidden layers and the fixed value of 0.35 for a bias neuron at the output layer were applied (figure 1).

2.3. Learning Algorithm Selection

The ANN training algorithm determines a weights variation method for loss minimization at each learning cycle (epoch). The law of weights variation is based on a selected loss function. In this paper, mean square error (MSE) was used as a loss function:

\[ E = \frac{1}{n} \sum_{i=1}^{n} (y_i - h(x_i))^2, \]  

where \( n \) – number of observations in the sample, \( y_i \) – expected value of the \( i \)-th observation, \( h(x_i) \) – predicted value for \( i \)-th observation. It should be noted, that this loss function is conventional for all regression models, not only for ANN based.

The training method selection was based on the analysis of several modifications of the first order gradient descent method: the standard gradient descent method (SGD), the root mean square propagation method (RMSProp), and the Nesterov adaptive moment estimation method (Nadam). As a result, the Nadam training method was chosen as a primary method for this case, since, according to the authors opinion [9], it is simple in implementation, efficient, well suits for problems with noisy or fuzzy gradients, and training parameters do not require multiple customization. Besides, this method displayed the best result during experimental study. Nadam training method approximates second-order learning algorithms and takes in account the error value on the previous training steps, in contrast to the standard gradient descent method. It represents a modification of Adam algorithm with additional calculation of Nesterov moment, allowing to slow down training process if necessary.

In Nadam method weight change, based on calculation of mean square error, is expressed by following equation:

\[ (\Delta W)_i = \frac{\eta}{\varepsilon + S_i} \left[ \mu_{i+1} \cdot \frac{m_i}{1 - \prod_{i=1}^{\infty} \mu_i} + (1 - \mu_i) \cdot \frac{(\nabla E)_i}{1 - \prod_{i=1}^{\infty} \mu_i} \right], \]  

where \( S_i = (\alpha \cdot S_{i-1} + (1 - \alpha) \cdot (\nabla E)_i^2) / (1 - \alpha^i) \), \( m_i = \beta m_{i-1} + (1 - \beta) (\nabla E)_i \), \( \mu_i = \beta \cdot (1 - 0.5 \cdot 0.96^{0.004 \cdot i}) \), \( m_0 = 0, S_0 = 0, \alpha = 0.999, \beta = 0.9, \varepsilon = 10^{-7} \).

It should be noted especially, that the learning rate \( \eta \) in equation (2), which determines the step size of weights variation, should be selected carefully, since it determines the possibility of achieving a global minimum during training process. The optimal value of the parameter \( \eta \) for different training algorithms varies drastically. For the problem under consideration, the best values of \( \eta \) were determined experimentally for different training methods: SGD (\( \eta = 0.06 \)) and Nadam (\( \eta = 0.0003 \)). In this work, supervised learning in batch mode was used. The considered neural network and selected traning algorithm were implemented on Python using Keras.

2.4. Estimation of Regression Results

In this work, the determination coefficient [10] was used as the main parameter for assessing the results of regression analysis, instead of MSE. The MSE error function, involved in training process, is applied occasionally to evaluate the model quality. However, the main problem of this function is the difficulty of the interpretation of its numerical value in the framework of the regression model. In this case the determination coefficient seems to be more suitable. It is determined after each training epoch as:
where \( \sigma_y^2 \) – variance of expected values, \( E \) – mean square error, defined by equation (1).

The determination coefficient lies in the range \((-\infty;1]\). The higher value of determination coefficient, the more accurate values are predicted by the regression model. It should be noted, that in order to control the training process, it is necessary to calculate both functions (MSE and \( R^2 \)) for training and test samples at each epoch.

To assess the quality of the regression model the so-called expected/predicted values plots are used in addition to calculating the determination coefficient. On the plot, each observation is displayed as a point in the plane of values, that were calculated by the regression model (predicted values), versus expected values, that should be predicted (measured values). This diagram visualizes model quality and allows to identify areas, in which considered model has a low prediction accuracy. If the computer model predicts all observations without significant error, then all points lay near the straight line \( y=x \), while the determination coefficient tends to 1.

According to the expected/predicted value diagram, it is convenient to identify observations for which a prediction turned out to be worse of all in a set of observations. To facilitate this process, two lines \( y= x \pm SD \), here SD is the standard deviation) are drawn on the plot. The narrower SD range and the smaller a number of points outside the range, the better is the prediction results.

3. Results and discussion
The selection of the criteria for analyzing quality of ANN should be conducted in accordance with the considered problem in each particular case. The optimal model, while solving a given problem, should predict the bacterioplankton abundance on the base of input physicochemical parameters equally well for both training and test samples. Moreover, the determination coefficient for the test sample should also tend to 1. Only in this case, the result can be considered acceptable. In addition, while choosing the best ANN model, should be monitoring the appearance of the overfitting at the optimal epoch (i.e. the epoch where the maximum determination coefficient for the test sample is achieved). This is necessary to exclude cases of low generalization quality with an overall high accuracy of the regressor. In this work, we use a wide range of methods for assessing the quality of the designed ANN model and a number of specific methods for increasing its efficiency.

3.1. Analysis of ANN Architecture
While analyzing various ANN architectures and training methods the following criteria were selected. Firstly, a correctly trained model should have no overfitting and early hit into the local minimum. And at the same time the selected models should have a determination coefficient value of at least 0.3 on the test sample and not less than 0.5 on the training sample. The number of neurons on hidden layers for 36 studied architectures was varied from 5 to 12, and the number of neurons on the second hidden layer was less or equal to the number of neurons on the first hidden layer. For each network architectures 10 different models with randomly generated input weights were trained. In order to compare different learning algorithms, training for each algorithm was carried out from the same initial state, i.e., with the same weights initialization of the model. The interruption criteria for training process, was the obtaining maximum determination coefficient for the test sample. Table 2 represents the results of training a set of networks via two different methods: Nesterov adaptive moment estimation the method (Nadam) and standard gradient descent method (SGD). The percentage of correct models from 10 trained (CM%) and average determination coefficients (DC) for learning (LS) and test (TS) samples are given. The determination coefficients presented in table 2 are averaged over a set of determination coefficients for correctly trained models.
Table 2. Results for various architectures and two training algorithms.

| ANN  | Training algorithms |   |   |
|------|---------------------|---|---|
|      | CM% | LS (max) | TS (max) | % CM | LS | TS |
| HL1  | HL2 |   |   |   |   |   |
| 9    | 5(57) | 50% | 0.839 (0.947) | 0.414 (0.514) | 10% | 0.812 | 0.301 |
| 5    | 5(4) | 50% | 0.834 (0.678) | 0.411 (0.493) | 0% | n/c* | n/c* |
| 12   | 8(55) | 40% | 0.913 (0.994) | 0.452 (0.649) | 20% | 0.746 (0.726) | 0.55 (0.771) |
| 11   | 5(59) | 30% | 0.79 (0.999) | 0.49 (0.626) | 10% | 0.71 | 0.427 |
| 7    | 6(81) | 30% | 0.967 (0.951) | 0.455 (0.544) | 10% | 0.608 | 0.401 |
| 6    | 6(22) | 30% | 0.809 (0.851) | 0.447 (0.517) | 20% | 0.717 (0.719) | 0.39 (0.448) |
| 6    | 5(88) | 30% | 0.998 (0.999) | 0.436 (0.559) | 10% | 0.533 | 0.385 |
| 10   | 7(10) | 30% | 0.928 (0.888) | 0.419 (0.443) | 30% | 0.857 (0.951) | 0.472 (0.630) |
| 10   | 5(07) | 30% | 0.705 (0.735) | 0.403 (0.473) | 20% | 0.797 (0.842) | 0.401 (0.411) |
| 11   | 6(68) | 30% | 0.837 (0.777) | 0.4 (0.534) | 20% | 0.691 (0.818) | 0.405 (0.422) |
| 8    | 5(35) | 30% | 0.848 (0.752) | 0.352 (0.378) | 10% | 0.596 | 0.301 |

* n/c – no correct models

According to the results of training several ANN architectures (Table 1), we can conclude, that the architecture with the best and most stable result is a model with 9 hidden layers and 5 neurons, trained by NAdam. The less stable results for SGD are due to the fact, that training by this method does not have a mechanism to escape from a local minimum, while the NAdam allows achieving a global minimum due to the adaptive changes in learning rate and moment, therefore, the probability of getting the correct result with NAdam training method is higher.

Figure 2. Determination coefficients (DC) for training and test set for five models with architecture 9-5. The first two columns in each series correspond to the case without dropout and the last two columns in each correspond to the case with dropout. Numbers under DC series indicate the number of correctly trained model.

For the selected architecture 9-5, named by the numbers of neurons on two hidden layers, the average value of the determination coefficients for the training and test samples are 0.839 and 0.414, respectively. However, the maximum values of these parameters for individual models may be much higher. Figure 2 represents a bar chart of the values of determination coefficient (DC) for five correctly trained models correspond to 9-5 architecture. The first two columns in each series correspond to the DC values for the training and test samples, respectively. Obviously, for models with numbers 1-3, the
values of the determination coefficients in the corresponding sets have close values, which indicates a rather high learning stability.

Let us describe in details one of the successful models (but not the best) – number 2. Figure 3 (a) shows MSE error function during the learning process. According to the MSE function, the beginning of overfitting is clearly traced at the 2000-epoch area. At this area, the determination coefficient for the test sample reaches its maximum value (0.374), while the determination coefficient for training dataset is 0.905. Taking into account, that for the regression problem determination coefficient is easier to analyze, than the mean square error, and that the dependence of the determination coefficient represents better the real quality of trained model, we plot this dependence in figure 3 (b).

![Figure 3](image)

**Figure 3.** Training results for model 2. (a) – mean square error (MSE) function, (b) variation of the determination coefficient during training process, (c) – diagram of expected/predicted values at the optimal 1894 epoch (SD range ±0.0846).

A diagram of expected/predicted values calculated at the optimal 1894 epoch (figure 3 (c)) shows, that despite the rather low values of the determination coefficients, almost all experimental points fit into the SD range (± 0.0846) obtained for test sample. The predicted values for the training dataset are close enough to the measured values, but are not equal to them exactly. This allows the developed model to possess a good generalization quality for a new data, on which training is not previously performed. The predicted values for the test sample are slightly farther from the true values, but by 66% fit in the SD range. Four observations get out of this range, which explains a low determination coefficient for the testing dataset.

### 3.2. Dropout Application

One of the methods to improve the generalization quality of ANNs is to use a dropout layer during training [11]. This layer does not perform any non-linear transformations, but placing between layers it randomly zeroing input signals at each neuron during training process. Thereby, this layer prevents neurons in a subsequent layer from learning only by one hidden feature.

In this research, a dropout layer was added after the first hidden layer. The zeroing probability for each neuron was set at 10%. Determination coefficients for 5 previously selected models are presented in figure 2, as the last two columns in each series. ANN training was carried out under exactly the same initial conditions as in the case without dropout (i.e. network architecture, initial weights, learning rate). For some models, the result has not changed much, but for models with numbers 2, 4, 5, the determination coefficient for the test sample has increased significantly, which indicates an improvement of ANN generalization quality.

For the considered model 2, training results in the presence of a dropout layer are presented in figure 4. For this model a significant increase in the determination coefficient on the test sample was observed. However, the number of requisite epochs for achieving maximum result increased up to 6231, which is
associated with a partial zeroing of signals on the first hidden layer during training process. Comparison of the prediction results at the optimal epoch for the model 2 (figure 3 (c) and figure 4 (c)), gives an improvement of the results in presence of the dropout layer, i.e. the predicted values lie in a smaller SD range = 0.0695, and for the test sample now they fit in 83% SD.

Figure 4. Training results for the model 2 with a dropout layer. (a) mean square error (MSE) function, (b) variation of the determination coefficient during training process, (c) diagram of expected/predicted values at the optimal 6231 epoch (SD range ± 0.0695).

The procedure of adding dropout layer is rather specific and ambiguous method for improving the generalization quality of ANNs. Sometimes it can lead to unpredictable results, since the zeroing of individual neurons occurs randomly. In this research, conducted on 360 ANN models, both improvement and worsening of the initial results were observed. For five models, shown in figure 2, all three possible cases are presented: the improvement of the quality of the model is observed for models 2 and 5; the results' worsening displays model 4; and the indifference to the introduction of a dropout layer show models 1 and 3. The situation when dropout does not change the quality of the model usually occurs if the weights initialization did not work correctly, that is, the weights were generated in such a way that the model began to give a priority to a single feature or group of features. In this case, the model begins to ignore some features during training process, which leads to a decrease of generalization quality, and dropout of course cannot solve this problem. If the weights initialization was carried out correctly, but the model got into a local minimum at the early stage of training, then dropout may worsen the prediction quality of the model. In this case, there a decrease of the determination coefficient is observed for the training sample and, at the same time, an increase for the test sample can occur. It means that dropout increased the generalization quality of the model by lowering its overall quality. A significant improvement in the model's quality occurs when the model, along with the correct weights initialization, is able to get out of the local minimum and successfully reach the global minimum. Such a case is shown in figure 3 and 4 (model 2).

While working with ANNs usually it is necessary to train a set of models, and the application of the dropout layer helps to increase a number of correctly trained models. This technique is quite useful in the case, when the initial data have a complex dependence, not amenable to preliminary analysis, as well as when a set of observations is small. The effectiveness of dropout application for small samples is due to the fact, that during the training process an indirect augmentation of the input data occur.

4. Conclusion
In this investigation the characteristic neural-network regression model for prediction of the bacterioplankton abundance according to physicochemical parameters of the environment is presented. The best ANN architecture 10-9-5-1 with the determination coefficients of 0.97 and 0.69 for training and test samples, respectively, was selected from a number of variants. The achieved determination
coefficients, as well as expected/predicted values function obtained at the optimal epoch, showed that the trained model has a high prediction level and a good generalization quality. The analysis of the dropout effect on the training process quality was made. Several hints of dropout application were given, and some reasons for possible worsening or improvement of the results in the presence of dropout layer are pointed out.

References
[1] Chukwu A U and Adepoju K A 2012 On the power efficiency of artificial neural network (ANN) and the classical regression model Progress in Applied Mathematics 3 (2) 28–34
[2] Livni R, Shalev-Shwartz S and Shamir O 2014 On the computational efficiency of training neural networks Advances in Neural Information Processing Systems 1 855–863
[3] López-Martin C 2015 Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects Applied Soft Computing 27 434–449
[4] D’Alimonte D and Zibordi G 2003 Phytoplankton determination in an optically complex coastal region using a multilayer perceptron neural network IEEE Transactions on Geoscience and Remote Sensing 41 (12) 2861–2868
[5] Tajik S, Ayoubi S and Nourbakhsh F 2012 Prediction of soil enzymes activity by digital terrain analysis: comparing artificial neural network and multiple linear regression models Environmental Engineering Science 29 (8) 798–806
[6] Mattei F, Franceschini S and Scardi M 2018 A depth-resolved artificial neural network model of marine phytoplankton primary production Ecological Modelling 382 51–62
[7] Millie D F at al. 2006 Neural net modeling of estuarine indicators: Hindcasting phytoplankton biomass and net ecosystem production in the Neuse (North Carolina) and Trout (Florida) Rivers, USA Ecological Indicators 6 (3) 589–608
[8] Lutsenko E S and Moskvin M I 2014 Annual dynamics of bacterioplankton and bacterioperiphyton populations of rock substrates on littoral of the Kola bay (The Barents sea) Fundamental Research 8 79–83
[9] Kingma D P and Ba J 2015 Adam: A method for stochastic optimization arXiv preprint arXiv:1412.6980. 2014 Available at: https://arxiv.org/abs/1412.6980 (accessed: 21.02.19)
[10] Nakagawa S, Johnson P C D and Schielzeth H 2017 The coefficient of determination R 2 and intra-class correlation coefficient from generalized linear mixed-effects models revisited and expanded Journal of the Royal Society Interface 14 (134) 20170213
[11] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 Dropout: a simple way to prevent neural networks from overfitting Journal of Machine Learning Research 15 (1) 1929–1958