Metric analysis of the parameter space of deep neural networks

Ruslan R. Nasyrov, Vadim V. Strijov
nasyrov.rr@phystech.edu

The problem of dimensionality reduction in the parameter space of machine learning models is being investigated. Regression and classification tasks are solved using fully connected neural networks. A metric analysis of the parameter space of the model is conducted. It is assumed that individual model parameters, random variables, are collected into vectors, which are multidimensional random variables. The analysis of their mutual arrangement in space represents the subject of investigation in this work. This analysis reduces the number of model parameters, estimates their significance, and selects them. To determine the position of the parameter vector in space, its mean and covariance matrix are estimated using bootstrap methods, SGD estimation, and Bayesian neural networks. Experiments are conducted on tasks of synthetic time series recovery, quasi-periodic accelerometer readings, and IRIS and MNIST datasets.

Keywords: *Time series; dimension reduction; relevance of parameters; parameter space; model selection.*

1 Introduction

High-dimensional data is often excessive, which poses a challenge for their efficient processing and utilization. The problem of dimensionality reduction in the feature space of an object is addressed in this work. Its basic principle is to map the high-dimensional feature space into a low-dimensional one while preserving important information about the data [Jia et al., 2022].

Currently, there are many methods for reducing the dimensionality of data. In [Örnek and Vural, 2019], dimensionality reduction is achieved by constructing a differentiable embedding function into a low-dimensional representation, while linear methods are discussed in [Cunningham and Yu, 2014]. In [Isachenko and Strijov, 2022], the problem of dimensionality reduction is solved for predicting human limb movement from electrocorticogram using the QPFS (quadratic programming feature selection) method, which takes into account the multicollinearity of both input and target features.

In addition to the problem of reducing the dimensionality of input data, there is also the problem of selecting the optimal structure of a model. In the case of optimizing the structure of a neural network, much attention has been paid to studying the feature space of the model. The OBS (Optimal Brain Surgeon) method, which involves removing network weights while maintaining its approximation quality, with the choice of weights to be removed determined by computing the Hessian of the error function with respect to weights, is applied in [Hassibi et al., 1993] and [Dong et al., 2017].

The article [Грабовой et al., 2019] presents a first-order method for weight removal based on finding the variance of the gradient of the error function with respect to parameters and analyzing the covariance matrix of parameters, while in [Грабовой et al., 2020], irrelevant weights are not removed but their training is stopped.

The aforementioned tasks of reducing data dimensionality and selecting optimal neural network structure are based on exploring the input data space and feature space, respectively. A significant drawback of these works is that they analyze *individual* parameters (scalars) of the models and their interdependence. This overlooks the fact that input data is affected by *vectors* of parameters through scalar products, thus neglecting the simple *structure* of transformation.

This work addresses the problem of time series restoration, within which the issue of reducing the dimensionality of the model parameters space is investigated. Dimensionality reduction is based on analyzing the conjugate space to the input space, which connects the input space and the parameter space.

In this study, we will consider neural networks with a simple structure that are composed of linear and basic non-linear functions (activation functions). Their composite block is described by the form:

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}),\ \mathbf{y}, \mathbf{b} \in \mathbb{R}^m,\ \mathbf{x} \in \mathbb{R}^n,\ \mathbf{W} \in \mathbb{R}^{m \times n},\ \sigma : \mathbb{R} \to \mathbb{R}.$$

In the OBS and OBD methods, as well as in the article [Грабовой et al., 2019], the elements $\mathbf{W}_{ij}$ were investigated individually as scalars. The authors propose to study them as vector-rows:

$$\mathbf{w}_1, \ldots, \mathbf{w}_m : \mathbf{W} = \begin{pmatrix} \mathbf{w}_1^\mathsf{T} \\ \ldots \\ \mathbf{w}_m^\mathsf{T} \end{pmatrix}.$$

In neural networks, these rows are commonly referred to as *neurons*. In SSA (singular spectrum analysis), $\sigma = Id$, and the matrix $\mathbf{W} = \mathbf{W}_k$ is an approximation of the true matrix of phase trajectories $\mathbf{X}$ (Hankel matrix) as a sum of k elementary matrices.

Let $\mathbf{x} = [x_1, \ldots, x_N]^\mathsf{T}, x_i \in \mathbb{R}$ be a time series and $1 \leqslant n \leqslant N$ be the window width. The point $\mathbf{x}_t = [x_t, \ldots, x_{t+n-1}]^\mathsf{T}$ is a point of the phase trajectory of the time series in the trajectory space $\mathbb{H}_\mathbf{x} \subset \mathbb{R}^n$. It is assumed that each point of the phase trajectory is normally distributed around its mean. Therefore, the time series is also random, and the result of training a model on it, i.e., the parameters of the trained model, will also be random.

In this work, the position of the random parameter vectors $\mathbf{w}_i$ in the metric space is studied. Their means $\mathbf{e}_i = \mathsf{E}(\mathbf{w}_i)$ and covariance matrices $\mathsf{D}(\mathbf{w}_i) = \mathbf{A}_i^{-1}$ are estimated using bootstrap and variational inference methods [Hastie et al., 2009]. We assume that these vectors $\mathbf{w}_i$ are normally distributed, so the pair $(\mathbf{e}_i, \mathbf{A}_i^{-1})$ fully describes the probability distribution of the vector $\mathbf{w}_i$.

As a graphical analysis, the positions of these vectors are represented as Gaussian mixtures in the metric space. Figure 1 shows the probability density functions of three Gaussian vectors (vertical axis) depending on their positions in the plane. At each point, the density is the sum of the densities of the three distributions, normalized such that the area under the graph equals 1. The peaks of the distribution correspond to the means of the vectors, and their shape is determined by the covariance matrix $\mathbf{A}^{-1}$. Thus, the lower and wider the peak, the greater the variance and vice versa.

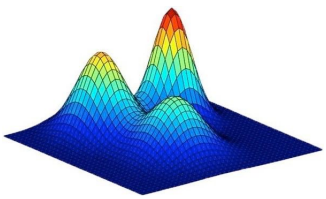
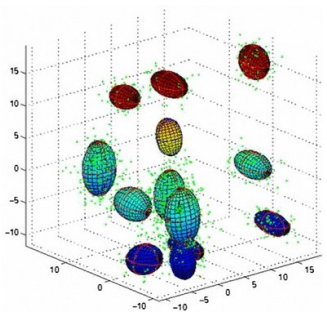


Рис. 1 Gaussian mixture of three 2-dimensional vectors.

Figure 2 depicts ellipses corresponding to 95% confidence areas for Gaussian vectors in a 3-dimensional space. The greater the width of the ellipse along a certain direction, the greater the variance of the vector in that direction. Dimensionality reduction is achieved through metric analysis of the parameter vector space by selecting relevant rows (with low variance), replacing multicorrelated rows with their linear composition using an extension of the QPFS algorithm, and studying the



structure of row communities. The basic models used are SSA (singular spectrum analysis) ([Golyandina et al., 2001]), 2-layer and 3-layer fully connected neural networks. The task of reconstructing a time series is solved on synthetic data with noisy sin and accelerometer readings from the MotionSense3 dataset [Malekzadeh et al., 2018].

2 Problem statement

Let $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_m\}, \mathbf{s}_i = [\mathbf{s}_i^1, \ldots, \mathbf{s}_i^T], \mathbf{s}_i^j \in \mathbb{R}$ be a set of m time series, where n is the length of the signals. Each time series is a sequence of measurements of a quantity over time.

Definition 1. *The temporal representation* $\mathbf{x}_t = [\mathbf{s}_1^t, \ldots, \mathbf{s}_m^t]^\mathsf{T} \in \mathbb{R}^m$ *consists of the measurements of the time series at time* t*.*

Definition 2. *The history of length* h *for time* t *of the set of time series* $\mathcal{S}$ *is the matrix* $\mathbf{X}_{t,h} = [\mathbf{x}_{t-h+1}, \ldots, \mathbf{x}_t]^\mathsf{T} \in \mathbb{R}^{h\times m}$*.*

Definition 3. *The prediction horizon of length* p *for time* t *of the set of time series* $\mathcal{S}$ *is the matrix* $\mathbf{Y}_{t,p} = [\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+p}]^\mathsf{T}$*.*

Definition 4. *The forecasting model* $\mathbf{f}^{\mathsf{AR}} : \mathbb{R}^{h\times m} \to \mathbb{R}^{p\times m}$ *is an autoregressive model that predicts the prediction horizon* $\mathbf{Y}_{t,p}$ *from the history* $\mathbf{X}_{t,h}$*.*

The task of autoregressive decoding is to construct a forecasting model $\mathbf{f}^{\mathsf{AR}}$ that gives the *prediction horizon* of a set of time series based on their *history*. In the following, we assume that we are reconstructing one time series, i.e., $m = 1$.

Let $\mathbb{S}$ be the set of all one-dimensional time series:

$$\mathbb{S} = \bigcup_{n=1}^{+\infty} \{[s_1, \ldots, s_n] \in \mathbb{R}^n\}.$$

We fix the length of the prediction horizon to be 1. Then the forecasting model is a function $f^{\mathsf{AR}} : \mathbb{S} \to \mathbb{R}$. The original time series $\mathbf{s} = [s_1, \ldots, s_T]$ is divided into two parts $\mathbf{s} = [\mathbf{s}^H|\mathbf{s}^T], \mathbf{s}^H = [s_1, \ldots, s_h], \mathbf{s}^T = [s_{h+1}, \ldots, s_T]$. The task is to predict $\mathbf{s}^T$ with maximum accuracy. The prediction is made as follows:

1. Using the model f, $\hat{s}_{h+1}$ is predicted.
2. The predicted element $\hat{s}_{h+1}$ together with the original time series $\mathbf{s}^H$ are fed into f to predict $\hat{s}_{h+2}$.
3. Steps $1-2$ are repeated until the entire $\hat{\mathbf{s}}^T = [\hat{s}_{h+1}, \ldots, \hat{s}_T]$ is predicted.

For simplicity of notation, we denote $f(\mathbf{s}^H) = \hat{\mathbf{s}}^T = [\hat{s}_{t+1}, \ldots, \hat{s}_T]$.

The model $f = f(\mathbf{w}, \mathbf{s}), \mathbf{w} \in \mathbb{W}, \mathbf{s} = [s_1, \ldots, s_t] \in \mathbb{R}^t$ is chosen from some parametric family. The model parameters are chosen such that they minimize the error function $S = S(\mathbf{w}|\mathbf{s}, f)$:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{W}} S(\mathbf{w}|\mathbf{s}, f).$$

In this work, the MSE error function is used, i.e.,

$$S(\mathbf{w}|\mathbf{s}, f) = \sum_{t=h+1}^{T} (\mathbf{s}_t - \hat{\mathbf{s}}_t)^2.$$

The input data for the model is a matrix of phase trajectories. It is constructed from the time series $\mathbf{s} = [s_1, \ldots, s_T]$ and the window width h as follows:

$$\mathbf{X} = \begin{pmatrix} s_1 & \ldots & s_k \\ s_2 & \ldots & s_{k+1} \\ \ldots & \ldots & \ldots \\ s_h & \ldots & s_T \end{pmatrix} = \big[X_1 : \cdots : X_k\big], k = T - h + 1, X_i = [s_i, s_{i+1}, \ldots, s_{i+h-1}] \in \mathbb{R}^h.$$

The trajectory matrix is a Hankel matrix, as each diagonal of the form $i + j = const$ contains identical elements. The vectors X_i are points on the phase trajectory of the signal.

As an alternative model for time series reconstruction, we will use a two-layer neural network with orthogonal transformations. Let us discuss each of them in more detail.

2.1 SSA

The reconstruction of a time series in the SSA model is achieved by decomposing the phase trajectory matrix into a sum of rank-one matrices.

Then, the eigenvalues and corresponding orthonormal systems of vectors of the matrix $\mathbf{S} = XX^T \in \mathbb{R}^{h\times h}$ are computed, where X is the trajectory matrix. Let $\lambda_1 \geqslant \ldots \geqslant \lambda_L \geqslant 0$ be the eigenvalues of $\mathbf{S}$ and let $u_1, \ldots u_h$ be the orthonormal system of eigenvectors corresponding to the eigenvalues. Then, the vectors $v_i = \frac{\mathbf{X}^T u_i}{\sqrt{\lambda_i}}$ for $i = 1, \ldots, h$ are computed. The time series is reconstructed as $\mathbf{X} = \mathbf{X}_1 + \cdots + \mathbf{X}_d$, where $\mathbf{X}_i = \sqrt{\lambda_i} u_i v_i^T$ is the SVD decomposition of $\mathbf{X}$. The matrices $\mathbf{X}_i$ are grouped, Hankelized, and then the reconstructed signal matrix is obtained as $\hat{\mathbf{X}} = \tilde{\mathbf{X}}_{I_1} + \cdots + \tilde{\mathbf{X}}_{I_m}$, where $I_1 \sqcup \cdots \sqcup I_m \subset \{1, \ldots, h\}$ and $\tilde{\mathbf{X}}_I = hank\big(\sum\limits_{i\in I} \mathbf{X}_i\big)$. Here, hank$(X)$ is the Hankelization of matrix X, which replaces all elements on each diagonal of the form $i + j = const$ with their arithmetic mean. The parameters of the SSA model are the sets $I_1, \ldots, I_m$.

2.2 Neural network

The two-layer neural network model with orthogonal transformations is defined as follows:

$$\mathbf{f}(\mathbf{x}) = \sigma(\mathbf{W}_1^{\mathsf{T}} \cdot \sigma(\mathbf{W}_2^{\mathsf{T}}\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2),$$

where

$$\mathbf{x} \in \mathbb{R}^h,\ \mathbf{W}_2 \in \mathbb{R}^{h\times d}, \mathbf{W}_1 \in \mathbb{R}^{d\times h}\ b_1 \in \mathbb{R}^h,\ b_2 \in \mathbb{R}^h; \mathbf{W}_1^{\mathsf{T}}\mathbf{W}_1 = \mathsf{I}, \mathbf{W}_2\mathbf{W}_2^{\mathsf{T}} = \mathsf{I}.$$

The last two conditions ensure that the transformations are orthogonal. Here, the neural network reconstructs the values of a signal of length h time points.

A hypothesis is made about the input time series that the points on the phase trajectory are distributed according to a normal law, that is, $\mathbf{x}_t = [s_t, \ldots, s_{t+h-1}] \sim \mathcal{N}(\hat{\mathbf{x}}_t, \mathbf{B}^{-1})$, where $\hat{\mathbf{x}}_t$ is the expectation of the point on the phase trajectory at time t, and $\mathbf{B}^{-1}$ is the covariance matrix.

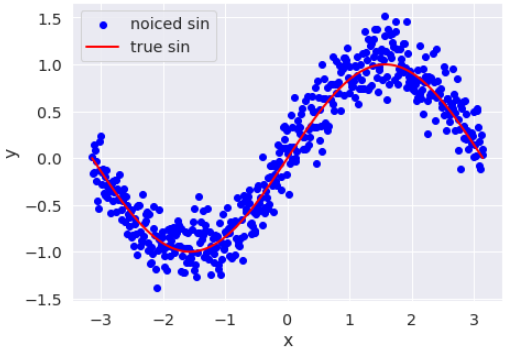
For metric analysis of the parameter space, a set of index sets $\mathcal{I}_1 \sqcup \cdots \sqcup \mathcal{I}_k \subset \{1, \ldots, len(\mathbf{w})\}$ is chosen, and the corresponding parameter subvectors $\mathbf{w}_{\mathcal{I}_1}, \ldots, \mathbf{w}_{\mathcal{I}_k}$ are considered. It is assumed that each $\mathbf{w}_{\mathcal{I}_j}$ corresponds to a "semantic unit"of the model. In neural network models, these are the rows of the linear transformation matrix $\mathbf{W}$, which are usually referred to as "neurons".

For each $\mathbf{w}_{\mathcal{I}}$, the expectation $\hat{\mathbf{w}}_{\mathcal{I}}$ and the covariance matrix $\mathbf{A}_{\mathcal{I}}^{-1}$ are estimated, which are then used for metric analysis of the parameter space.

3 Computational experiment

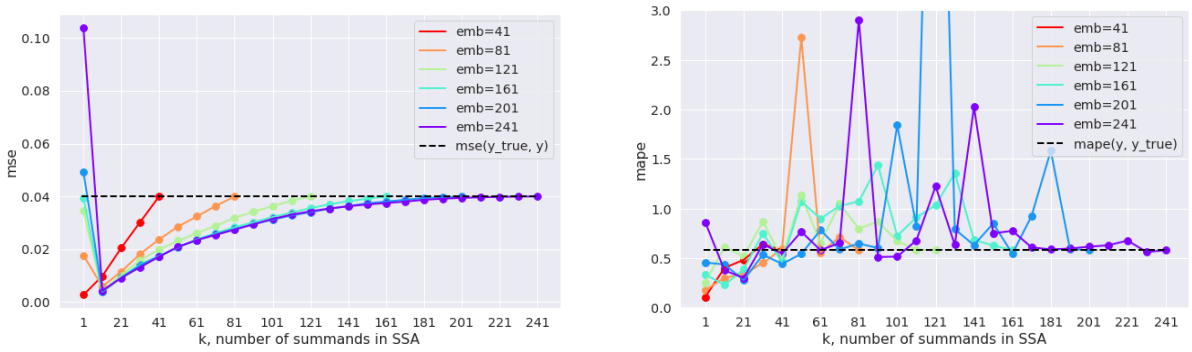
3.1 SSA

The scalar time series $\tilde{\mathbf{x}} = \mathbf{x} + \varepsilon$ consists of $n = 500$ values of a noisy sine wave measured at points in the interval $[-\pi, \pi]$ with uniform spacing. The noise used is from the distribution $\varepsilon \sim \mathcal{N}(0, 0.2)$. The data is shown in Figure 3.



Using the SSA method, the accuracy of reconstructing the time series $\mathbf{x}$ from the noisy series $\tilde{\mathbf{x}}$ is plotted as a function of window width h and number of principal components k used to reconstruct the matrix $\mathbf{X}$.

MSE and MAPE were used as quality criteria, measured between the reconstructed series $\hat{\mathbf{x}}$ and the true value of the series $\mathbf{x}$. The results are shown in Figure 4.



(*a*) MSE (*б*) MAPE

Рис. 4 Metrics for different values of h and k. The black line highlights the metric between the pure and noisy sine.

The different lengths of the plots are due to the fact that the window width h cannot be greater than the number of terms in the SVD reconstruction of the series. It can be seen that the true value of the series is accurately reconstructed with a relatively small number of terms (< 20), which corresponds to the minimum point of all the plots on the left-hand side. As more components are used, the error increases, and when the number of components is equal to the window length, MSE becomes the same as MSE between the true value of the series and its noisy version (the plots approach a horizontal dotted line).

The results confirm the practical consistency of SSA: the algorithm reconstructs the main trend with a small number of singular terms $\mathbf{X}_i$.

3.2 2-layer neural network with orthogonal transformations.

The scalar time series $\tilde{\mathbf{x}} = \mathbf{x} + \varepsilon$ consists of $n = 500$ values of a noisy sine wave measured at points in the interval $[-4\pi, 4\pi]$ with uniform spacing. The noise used is from the distribution $\varepsilon \sim \mathcal{N}(0, 0.2)$. The data is shown in Figure 5.

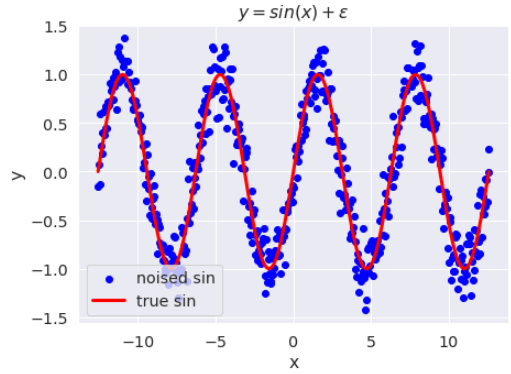


Рис. 5 Noised sin.

Using a neural network, the task of reconstructing the time series was solved. The hidden dimension of the neural network was $d = 3$, and the dimension of the trajectory space was $h = 20$.

On random subsets of the training set, $N = 100$ neural networks were trained for 25 epochs. The means and covariance matrices of the columns of the parameter $\mathbf{W}_1 \in \mathbf{R}^{3\times 20}$ were estimated. The resulting matrices were visualized in a 3D space, as shown in Figure 6.

All the parameters-neurons in Fig. 6 have the shape of elongated ellipses, located at a distance from each other. Some of the ellipses are elongated in the same direction (reducing the OY axis). Overall, these neurons appear to be quite independent and do not have a pronounced community structure.

In the next neural network experiment, an architecture of $10 - 3 - 20$ was taken, which means that $h_1 = 10$-dimensional vectors are input, the hidden space dimension is $d = 3$, and $h_1 = 20$ time steps need to be reconstructed.

On randomly selected subsets of the training set, $N = 100$ neural networks were trained for 70 epochs. The expectations and covariance matrices of the columns of the parameter $\mathbf{W}_1 \in \mathbf{R}^{3\times 10}$ were estimated. The resulting matrices were visualized in 3D space in Fig. 7.

In contrast to the previous experiment, in the current one, a separation into three communities is observed, corresponding to the orange, red, and purple ellipsoids, which have the highest variance among neurons (since they have a larger volume).

This partially confirms the hypothesis of the existence of community structure among neurons.

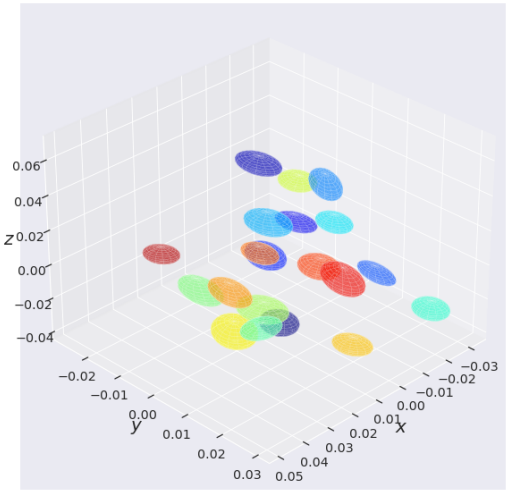


Рис. 6 Confidence areas of 3-dimensional vectors of a 2-layer neural network architecture 20-3-20.

4 Covariance estimation

The estimation of the covariance matrix of a neuron will be performed using three methods: 1) the bootstrap technique, 2) computation of the error Hessian with respect to the parameters, and 3) the Bayesian neural network approach.

4.1 Bootstrap

$N = 100$ models are trained using stochastic gradient descent (SGD), and unbiased estimates of the covariance and variance of neurons are computed using the following formulas:

$$\mathsf{E}[\mathbf{w}] = \frac{1}{n}\sum_{t=1}^{n}\mathbf{w}^t, \;\; cov(\mathbf{w}_i, \mathbf{w}_j) = \frac{\sum\limits_{t=1}^{n}(\mathbf{w}_i^t - \overline{\mathbf{w}_i})(\mathbf{w}_j^t - \overline{\mathbf{w}_j})}{n-1}$$

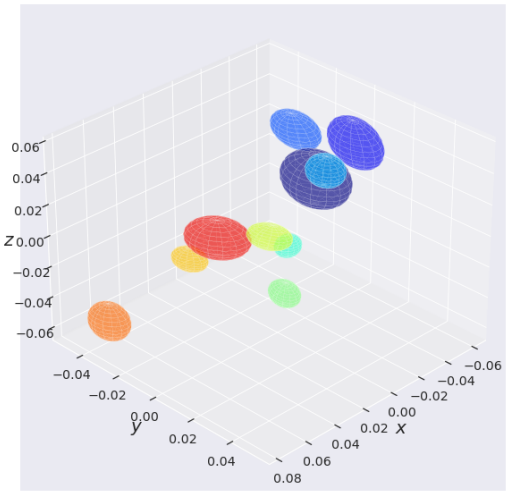


Рис. 7 Confidence areas of 3-dimensional vectors of a 2-layer neural network architecture 10-3-20.

4.2 Hessian

The calculation of covariance is carried out in the following ways, presented in the article[Chen et al., 2020].

Let $\mathbf{w}^* \in \mathbb{R}^d$ be the true vector of the model parameters, that is

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{W}} F(\mathbf{w})$$

$$F(\mathbf{w}) = \mathsf{E}_{\xi\sim\mathcal{D}} S(\mathbf{w}, \xi)$$

Where ξ is the element sampled from the training sample, S is the error function.

When optimizing the model parameters by the SGD method with the starting point $\mathbf{w}_0$, an iterative process occurs:

$$\mathbf{w}_i = \mathbf{w}_{i-1} - \nu_i \nabla S(\mathbf{w}_{i-1}, \xi_i)$$

Where ξ_i is sampled from the training sample $\mathcal{D}$, $\nabla S(\mathbf{w}_{i-1}, \xi_i)$ - the gradient of the error function relative to the weights of the model.

In the ASGS version (averaged SGD), which will be used later, the response is returned $\overline{\mathbf{w}_n} = \frac{1}{n}\sum\limits_{i=1}^{n}\mathbf{w}_i$.

Denote $A = \nabla^2 F(\mathbf{w}^*)$ is the Hessian of the error expectation, $S = \mathsf{E}\left[\nabla \mathsf{S}(\mathbf{w}^*, \xi) \cdot \nabla \mathsf{S}(\mathbf{w}^*, \xi)^{\mathsf{T}}\right]$ - covariance matrix $\nabla S(\mathbf{w}^*, \xi)$.

The article mentions that under the conditions of convexity F, the following asymptotic equality follows:

$$\sqrt{n}(\overline{\mathbf{w}}_n - \mathbf{w}^*) \to \mathcal{N}(0, A^{-1}SA^{-1})$$

Thus, a consistent estimate of the covariance matrix $\sqrt{n}\overline{\mathbf{w}}_n$ это $A^{-1}SA^{-1}$.

Алгоритм 1 Offline method for estimating the covariance matrix of parameters

1. Train the model by getting an approximation $\mathbf{w}^*$.
2. Approximate matrices A, S by sampling:

$$A_n = \frac{1}{n}\sum_{i=1}^{n} \nabla^2 S(\mathbf{w}^*, \xi_i),\ S_n = \frac{1}{n}\sum_{i=1}^{n} \nabla S(\mathbf{w}^*, \xi_i) \cdot S(\mathbf{w}^*, \xi_i)^\mathsf{T}$$

3. Estimate the covariance matrix $cov(\sqrt{n}\overline{\mathbf{w}}_n) \approx A_n^{-1} S_n A_n^{-1}$.

4.3 Bayes

A Bayesian neural network is being trained, where each neuron in the network has a mean and covariance matrix as its parameters. As a result of training, the necessary parameters are obtained automatically.

The training of the Bayesian model is performed using variational inference based on the joint likelihood of the data:

$$\mathcal{L}(\mathcal{D}) = \log p(\mathcal{D}) = \log \int\limits_{\mathbf{w}\in\mathbb{R}^k} p(\mathcal{D}|\mathbf{w}) \cdot p(\mathbf{w}) d\mathbf{w},$$

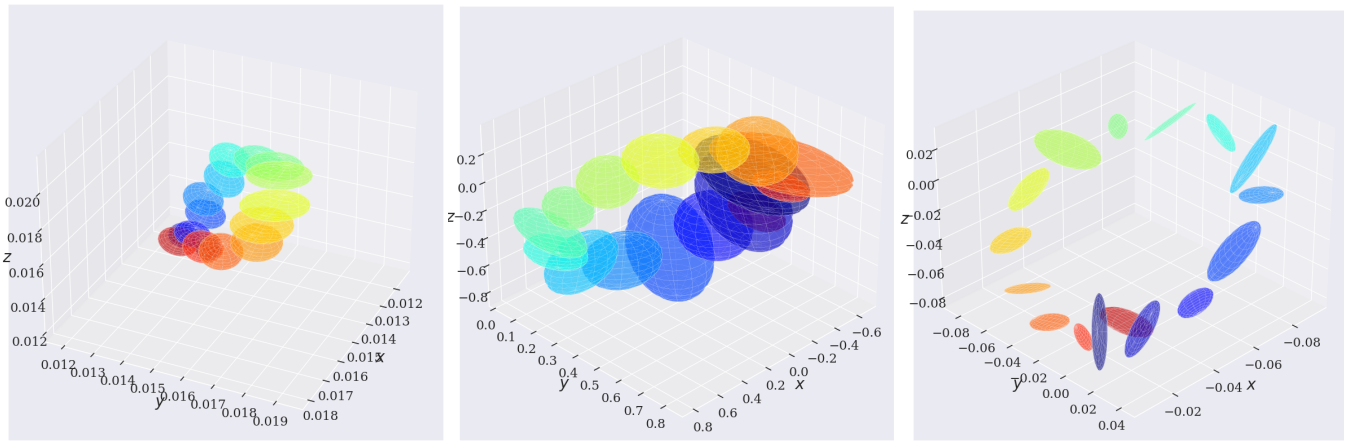
where $p(\mathbf{w})$ is the a priori distribution of the model parameters. Taking into account our assumption that it is normal, the student's training is reduced to solving the problem:

$$\widehat{\mathbf{w}} = \arg\min_{\mu,\mathbf{\Sigma},\mathbf{w}} \mathsf{D}_{\mathsf{KL}}(p(\mathbf{w}|\mathcal{D})||p(\mathbf{w})) - \log p(\mathbf{y}|\mathbf{X}, \mathbf{w}).$$

Where the second term is the logarithm of the sample likelihood, and the first is KL-the divergence between the a priori and a posteriori distribution of the model parameters.

5 Visualization of the network neuron space.

The problem of reconstructing the Accelerometer Motion Sense time series is solved using a two-layer neural network with an architecture of $100 - 3 - 100$. The network is trained for 50 epochs on the Accelerometer Motion Sense dataset, which consists of 10,000 measurements of device acceleration. After training, the covariance matrices of the model neurons are computed using the Bootstrap technique, according to Algorithm 1 (Hessian technique), by training a Bayesian neural network (Bayes technique). The results are presented in Figure 8. The structure of neurons is identical for all three methods, with all neurons arranged in a circular pattern. Given that the time series period is 100 measurements, it is evident that the size of the output layer of the neural network can be reduced to 16. The scale differs significantly, with the Bootstrap method yielding the smallest variances and neuron dispersion, while the Hessian method yields the largest. Additionally, it is observed that the Hessian and Bayes methods produce flat neurons (one of the eigenvalues of the covariance matrix is much smaller than the other two), unlike the Bootstrap method. This provides a potential opportunity to perform linear transformations of hidden layer neurons to reduce the size of the hidden space from 3 to 2 and is of interest for further research.



(*а*) Bootstrap (*б*) Hessian (*в*) Bayes

Рис. 8 The first 16 neurons of $100 - 3 - 100$ neural networks trained on the task of restoring the Motion Sense time series.

6 Dimensionality reduction using clustering.

Classification and regression tasks were solved using fully connected neural networks on three datasets: MNIST, Iris flower, and Motion Sense. The architectures used were $784 - 8 - 100 - 10$, $4 - 128 - 3$, and $100 - 10 - 100$, respectively.

Covariance and mean estimates of neurons were obtained using the bootstrap technique. Neuron clustering was performed using the k-means algorithm in two ways: in the first method, the distance between neurons was calculated as the Euclidean distance between their means, while in the second method, it was calculated as the Wasserstein distance between two normal distributions.

$$\rho_1(\mathbf{w_1}, \mathbf{w_2}) = ||\mathbf{m}_1 - \mathbf{m}_2||_2^2$$

$$\rho_2(\mathbf{w_1}, \mathbf{w_2}) = ||\mathbf{m}_1 - \mathbf{m}_2||_2^2 + Tr(\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2 - 2(\mathbf{\Sigma}_1^{0.5}\mathbf{\Sigma}_2\mathbf{\Sigma}_1^{0.5})^{0.5})$$

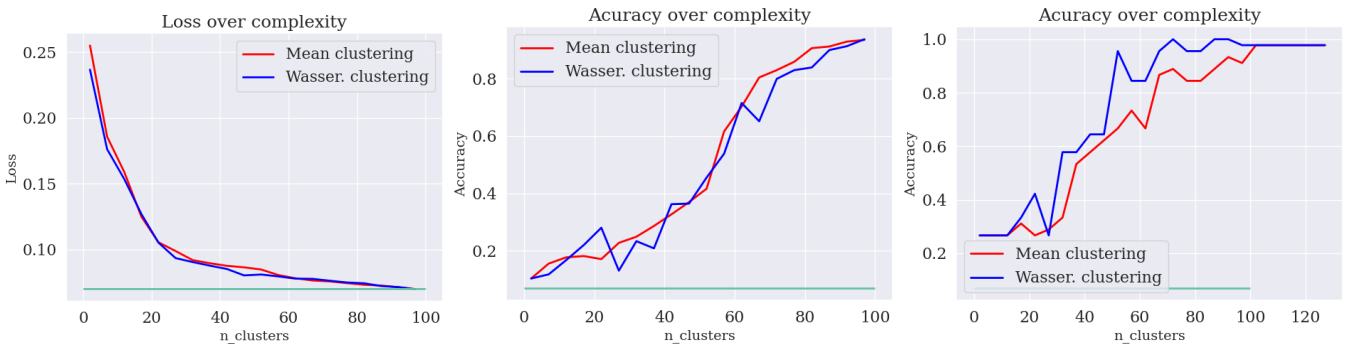
After clustering, each neuron was replaced with the centroid of its cluster and quality measurements were taken: mean squared error (MSE) for regression tasks and accuracy for classification.

The experimental results are presented in Figure 9.

The graphs depict the relationship between model accuracy and complexity. As the number of clusters increases, the model has more parameters and becomes more complex, resulting in higher prediction quality. Both clustering methods show similar results on the first two datasets, while the Wasserstein clustering method performs better on the IRIS dataset. This is likely due to the dimensionality of the neurons: smaller dimensionality allows for a more accurate estimation of the covariance matrix and better clustering results.

7 Conclusion

The study investigated the structure of neurons in neural networks when applied to time series reconstruction tasks. Using the MotionSence dataset, it was demonstrated that 3-dimensional neurons have a regular structure and are arranged in a circular pattern with a period of 16, which is 5 times smaller than the period of the time series. This presents an opportunity to reduce the size of the network's hidden space by more than 5 times.



(*а*) MotionSense. 10-dimensional neurons. (*б*) MNIST. 8-dimensional neurons. (*в*) IRIS. 4-dimensional neurons.

Рис. 9 Dependence of model accuracy on complexity on 3 datasets. The blue curve corresponds to clustering by the Wasserstein metric, and the red one— by the Euclidean metric.

Additionally, the possibility of reducing the dimensionality of parameter space in neural networks through neuron clustering was explored using three datasets of different natures: MotionSence - scalar time series for regression tasks; IRIS - tabular data for classification tasks; and MNIST - images for classification tasks. It was shown that up to 50% of neurons can be removed from the IRIS dataset while maintaining more than 90% classification accuracy. The effect was not as significant on the other datasets.

Future research will focus on compressing large industrial neural networks, with particular attention given to networks that reconstruct time series and potentially have redundant parameters. New methods for estimating covariance matrices of parameters, such as low-rank Bayesian networks, semi-Bayesian networks, and block-diagonal Bayesian networks, will also be developed.

8 *

Список литературы

Weikuan Jia, Meili Sun, Jian Lian, and Sujuan Hou. Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3):2663–2693, 2022.

Cem Örnek and Elif Vural. Nonlinear supervised dimensionality reduction via smooth regular embeddings. *Pattern Recognition*, 87:55–66, 2019.

John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509, 2014.

RV Isachenko and VV Strijov. Quadratic programming feature selection for multicorrelated signal decoding with partial least squares. *Expert Systems with Applications*, 207:117967, 2022.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.

Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in Neural Information Processing Systems*, 30, 2017.

302 Андрей Валериевич Грабовой, Олег Юрьевич Бахтеев, and Вадим Викторович Стрижов.
303 Определение релевантности параметров нейросети. *Информатика и её применения*, 13
304 (2):62–70, 2019.

305 Андрей Валериевич Грабовой, Олег Юрьевич Бахтеев, and Вадим Викторович Стрижов.
306 Введение отношения порядка на множестве параметров аппроксимирующих моделей.
307 *Информатика и её применения*, 14(2):58–65, 2020.

308 Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements*
309 *of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

310 Nina Golyandina, Vladimir Nekrutkin, and Anatoly A Zhigljavsky. *Analysis of time series*
311 *structure: SSA and related techniques*. CRC press, 2001.

312 Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. Protecting
313 sensory data against sensitive inferences. In *Proceedings of the 1st Workshop on Privacy by*
314 *Design in Distributed Systems*, pages 1–6, 2018.

315 Xi Chen, Jason D Lee, Xin T Tong, and Yichen Zhang. Statistical inference for model parameters
316 in stochastic gradient descent. 2020.