TASK QUARTERLY 7 No 1 (2003), 93–114

MATHEMATICAL MODEL OF ARCHITECTURE AND LEARNING PROCESSES OF ARTIFICIAL NEURAL NETWORKS

ANDRZEJ BIELECKI

Institute of Computer Science, Jagiellonian University,
Nawojki 11, 30-072 Cracow, Poland
Bielecki@softlab.ii.uj.edu.pl

(Received 7 November 2001; revised manuscript received 9 September 2002)

Abstract: A mathematical model of architecture and learning processes of multilayer artificial neural netwoks is discussed in the paper. Dynamical systems theory is used to describe the learning precess of networks consisting of linear, weakly nonlinear and nonlinear neurons. Conjugacy between a gradient dynamical system with a constant time step and a cascade generated by its Euler method theorem is applied as well.

Keywords: artificial neural network, neuron, learning process, topological conjugacy, gradient dynamical system, Euler method

1. Introduction

In recent years, artificial neural networks (ANNs) have been studied very intensively. There are many papers describing applications of ANNs to solve problems in control theory (see [1] Section 6.3, [2] Section 7.5, [3] Section 4.4, [4] Chapter 5, [5] Section 8.3), robotics ([5] Section 8.4), speech recognition ([1] Section 6.3, [6] Section 10.1), pattern recognition ([1] Section 6.3, [3] Section 4.1, [5] Section 8.2), data compression ([1] Section 6.3, [3] Section 4.2), expert systems ([5] Section 8.3) and many others. Since problems with neural nets learning processes emerged, their mathematical models have also been created. Graphs and matrices are used for modeling the architecture of ANNs (see, for instance, [7, 5]), whereas dynamical systems theory is used to analyse the behaviour of recurrent networks ([1] Section 2.2), and it seems to be a suitable tool for investigating learning processes in layer neural networks ([8] and [6] Chapter 9). But, though there are numerous mathematical results concerning both acting and learning processes of ANNs, these results are rather isolated facts. It seems that a coherent mathematical theory concerning ANNs has not been developed so far. This paper aims at filling this gap. Thus, a mathematical model of both the architecture and the learning process of artificial neural networks is discussed below. We focus on multilayer ANNs, but the model can be extended to recurrent ANNs as well.

TQ107G-G/93 10 X 2003 BOP s.c., http://www.bop.com.pl

Several papers have recently been devoted to the study of the qualitative properties of discrete-time dynamical systems obtained via discretization methods. The basic question is whether the qualitative properties of continuous-time systems are preserved under discretization. Various concepts of differentiable dynamics were investigated. Results on stability and attraction properties [9], the saddle-point structure about equilibria [10, 11], invariant manifolds [12, 13], averagings [14] and algebraic-topological invariants [15, 16] can be mentioned as examples. Numerical applications are studied as well [17]. The investigations are concerned with both local (see, for instance, [18, 19]) and global conjugacy [20–23]. Some of these results are used in this paper.

This paper discusses the mathematical model of both the architecture and the learning process of artificial neural networks. Dynamical systems theory is used to describe and analyse the learning process of networks consisting of linear, weakly nonlinear and nonlinear neurons. The theorems about conjugacy between a gradient dynamical system with a constant time step and a cascade generated by its Euler method [20, 18] are applied as well.

This paper is divided into four logical parts. First (Section 2 and Subsection 3.1) it is shown that the known results (see, for instance [1, 6, 24]), are simple theorems based on this model. In the second part (Subsections 3.2), theorems are presented which are slight generalizations of the known ones. In the third part (Subsection 3.3) a definition of a weakly nonlinear neuron is introduced, and its properties are analysed. Finally (Subsection 3.4), the properties of a nonlinear neuron are analysed. It seems that results presented in Subsections 3.3 and 3.4 are new (they have been presented in [25, 26] in a shortened form), though the problem of a neural network stability has been considered by other authors [27–31]. Furthermore, some of the problems considered in this paper have been analyzed in [32–34].

2. Mathematical model of an artificial neural network and its learning process

2.1. Model of a neuron

An artificial neuron is a unit having several weighted inputs and one output. Thus, we can say that a neuron is a function of two vector variables.

Definition 2.1.1 *A neuron with k inputs transforming a set $X \subset \mathbb{R}^k$ of input signals (a k-neuron on X) is a function*

$$F : \mathbb{R}^k \times X \ni (\vec{w},\vec{x}) \longmapsto F(\vec{w},\vec{x}) = f(\langle \vec{w},\vec{x} \rangle) \in \mathbb{R},$$

where $\vec{w}$ is a weights vector, $\langle \cdot,\cdot \rangle$ is a real scalar product, and $f : \mathbb{R} \longrightarrow \mathbb{R}$ is called an activation function of the neuron. If f is a linear operator, then the neuron is called linear. A function

$$F^* := F(\vec{w},\cdot) : X \ni \vec{x} \longmapsto F^*(\vec{x}) \in \mathbb{R},$$

is said to be a trained k-neuron on X.

Remarks

1. In applications, a standard real scalar product is used:
$$\langle \vec{w}, \vec{x} \rangle = \sum_{k'=1}^{k} w_{k'} \cdot x_{k'} := \vec{w} \circ \vec{x}.$$
2. Without losing generality, an identity can be used as an activation function of a linear neuron (see Proposition 2.2.11).
3. Bounded functions are most often used as activation functions of nonlinear neurons.
4. Definition 2.1.1 implies that a neuron is a function of two vector variables, whereas, in a trained neuron, a weights vector is fit. Thus, a trained neuron is a mapping of only one vector variable.
5. The most classical approach to multilayer ANNs is analyzed in this paper, so we have restricted our considerations to the composition of a scalar product and a real function as an activation function of a neuron. However, it should be stressed that it is not the only possibility (see [35]).

An artificial neural network is a system of neurons which are connected in such a way that the output signal of a neuron is given to the input of another neuron. Multilayer networks are considered in this paper.

2.2. Mathematical model of a multilayer ANN

The proposed model is based on graph theory. Let us recall its basic definitions.

Definition 2.2.1 *An ordered pair $(A, \mathscr{E})$, where A is a finite set of nodes and $\mathscr{E} \subset A \times A$ is a set of oriented edges, is said to be an oriented graph (orgraph).*

Set that $(a_i, a_j) \in \mathscr{E}$ is the edge directed from the node a_i to the node a_j.

Definition 2.2.2 *The nodes set power of a graph G is said to be the degree of the graph G and will be denoted by δ_G.*

Definition 2.2.3 *Let a graph $G = (A, \mathscr{E})$ be given. The power of the set $\{a_j : (a_j, a_i) \in \mathscr{E}\}$ is said to be an input semidegree of the node a_i and is denoted by $\delta^+_{a_i}$, whereas the power of the set $\{a_j : (a_i, a_j) \in \mathscr{E}\}$ is called an output semidegree of the node a_i and is denoted by $\delta^-_{a_i}$.*

Orgraphs are often used in the definition of an ANN (see, for instance, [36, 37]). The orgraph nodes are identified with neurons of an ANN, whereas directed edges determine inputs where the output signal from a given neuron is sent.

Assume that the following objects are given:

$G := (A, \mathscr{E})$ – an orgraph of a degree δ_G such that $\{a \in A : \delta^+_a = 0\} \neq \emptyset$;
$\gamma : A \ni a \longmapsto \gamma(a) \in \{1, \ldots, \delta_G\}$ – a bijection mapping;
$\mathscr{F}$ – the set of all neurons;
$\alpha : A \ni a \longmapsto \alpha(a) \in F$; if $\delta^+_a = 0$ then $\alpha(a)$ is a k-neuron, otherwise $\alpha(a)$ is a δ^+_a-neuron;
W – a set indexing all inputs of neurons in the ANN;
$\beta : \mathscr{E} \longrightarrow W$ – a bijection mapping.

Definition 2.2.4 *A quintuple*

$$\mathscr{A}_k := (G, \gamma, \alpha, W, \beta)$$

is called a k-ANN architecture. If $\mathscr{F}$ is the set of all trained neurons, then the five-tuple is said to be a trained k-ANN architecture.

Remarks

1. By the condition $\{a \in A : \delta_a^+ = 0\} \neq \emptyset$ there exist input neurons in an ANN. But the definition does not imply the existence of output neurons, as sometimes, recurrent networks are considedred in which such neurons do not exist. Then, the dynamical excititions of neurons in general and asymptotic equilibrium in particular are the response to the input signal.
2. The mapping γ assigns natural numbers to graph nodes.
3. The mapping α assigns neurons to graph nodes in such a way, that a k-neuron (if a network is a k-ANN) is assigned to a node in which input semidegree is equal to zero, whereas a δ_a^+ neuron is assigned to a node with an input semidegree δ_a^+.
4. According to the indexing convention, W can be a subset of natural numbers – in this case it will be denoted by W_1, a set of ordered pairs of natural numbers (in which case the first one is the neuron number and the second one – a number of the neuron input), denoted by W_2, or, mainly in multilayer networks, a set of three-tuples of natural numbers (the first one is the number of a layer, the second one – the number of a neuron in the layer, and the third one – the number of an input of the neuron), denoted by W_3.
5. The mapping β determines to which input of a neuron the signal from other neuron or the ANN input signal is given. Since every input of a neuron is weighted and β is a bijection, every weight in noninput neurons of the ANN can be identified with an edge of the graph describing the ANN architecture.
6. Usually, all neurons in an ANN have the same activation function. Sometimes, however, neurons in differet layers of multilayer ANNs (see Definitions 2.2.5 and 2.2.6) have different activation functions.
7. Matrices can also be used to define neural networks (see, for instance, [5]). Such approach is equivalent.

In this paper, we consider multilayer ANNs. We shall now define this kind of ANNs.

Definition 2.2.5 *Let*

$$\mathscr{A}_k := (G = (A, E), \gamma, \alpha, W, \beta)$$

be an architecture of an k-ANN, trained or not. If the set A of nodes of the corresponding orgraph G can be decomposed into a finite family of disjoint sets $A_1, \ldots, A_R$ in such a way, that each graph edge (a_i, a_j) satisfies the condition $a_i \in A_r$, and $a_j \in A_{r+1}$, where $r \in \{1, \ldots, R-1\}$, then such an architecture is said to be an architecture of R-layer k-ANN. If, furthermore, for every pair a_i, a_j of nodes such that $a_i \in A_r$ and $a_j \in A_{r+1}$ the ordered pair (a_i, a_j) is an edge of the graph, then the multilayer ANN is said to be complete.

Definition 2.2.6 *Let*

$$\mathscr{A}_k := (G = (A,E),\gamma,\alpha,W,\beta)$$

be an architecture of an R-layer k-ANN, trained or not. Let also be given two sets $X \subset \mathbb{R}^k$ *and* $Y \subset \mathbb{R}^m$, *where* m *is the number of the output layer neurons. The three-tuple*

$$\mathscr{S}_k(X,Y) := (X,\mathscr{A}_k,Y)$$

is said to be an R-layer k-ANN on X.

Remarks

1. The set X is a set of input signals, whereas the set Y is a set of output signals.
2. A recurrent ANN can be defined in a similar way. The set Y would be a set of dynamic excitations of neurons or of equilibrium states attained in response to input signals.

Define

$$\mathscr{S} := \{(\vec{x},\vec{y}(\vec{x})) : \vec{x} \in X, \vec{y}(\vec{x}) \in Y\},$$

where $\vec{y}(\vec{x})$ is an output signal of the network $\mathscr{S}_k(X,Y)$ if the vector $\vec{x}$ is given to the ANN's input. In this way we have constructed a function

$$\mathscr{S} : X \ni \vec{x} \longmapsto \mathscr{S}(\vec{x}) = \vec{y}(\vec{x}) \in Y,$$

corresponding to the multilayer network $\mathscr{S}_k(X,Y)$. This construction implies the following corollaries.

Corollary 2.2.7 *An ANN consisting of a single linear M-neuron on* $X \subset \mathbb{R}^M$ *generates a linear function* $\mathscr{S} : X \longrightarrow \mathbb{R}$.

Proof

The corollary is simply implied by the construction of the function $\mathscr{S}$ and properties of the scalar product. □

Corollary 2.2.8 *Let a one layer k-ANN consisting of T neurons*

$$F_1,\ldots,F_T : F_t : \mathbb{R}^k \supset X \longrightarrow Y \subset \mathbb{R}, t \in \{1,\ldots,T\}$$

be given. Then the mapping $\mathscr{S}$ *corresponding to the ANN is of the form*

$$\mathscr{S} : X \ni \vec{x} \longmapsto \mathscr{S}(\vec{x}) = (F_1(\vec{x}),\ldots,F_T(\vec{x})) \subset Y^T.$$

Assume that two trained k-ANNs are given: $(X,\mathscr{A}_k,Y)$ and $(X,\mathscr{A}_k^*,Y)$, with corresponding functions $\mathscr{S} : X \longrightarrow Y$ and $\mathscr{S}^* : X \longrightarrow Y$, respectively.

Definition 2.2.9 *If functions* $\mathscr{S}$ *and* $\mathscr{S}^*$ *are identically equal then the networks* $(X,\mathscr{A}_k,Y)$ *and* $(X,\mathscr{A}_k^*,Y)$ *are said to be equivalent.*

The next corollary follows directly from Definition 2.2.9.

Corollary 2.2.10 *Let two trained R-layer k-ANNs be given:* $(X,\mathscr{A}_k,Y)$ *and* $(X,\mathscr{A}_k^*,Y)$, *and let only the second one be complete. Let*

$$\mathscr{A}_k := (G = (A,\mathscr{E}),\gamma,\alpha,W,\beta)$$

be the architecture of the first one, and let the architecture of the second one,

$$\mathscr{A}_k^* := (G^* = (A^*,\mathscr{E}^*),\gamma^*,\alpha^*,W^*,\beta^*),$$

satisfy the condition $A^* = A$, $\mathscr{E} \subset \mathscr{E}^*$. *If weights of the network* $(X,\mathscr{A}_k^*,Y)$, *corresponding to edges which do not exist in* $(X,\mathscr{A}_k,Y)$ *are all equal to zero, and the*

remaining weights of $(X,\mathscr{A}_k^*,Y)$ *are equal to the corresponding weights of* $(X,\mathscr{A}_k,Y)$, *then the networks are equivalent.*

Remark

Since, by Corollary 2.2.10, every incomplete ANN is equivalent to a complete ANN, it is sufficient, without losing the generality, to consider only complete ANNs.

Proposition 2.2.11 *Every trained ANN consisting of one linear* M*-neuron is equivalent to a trained ANN consisting of one linear* M*-neuron having the identity as its activation function.*

PROOF

The considered M-neuron F is linear, so its activation function $f:\mathbb{R}\longrightarrow\mathbb{R}$ is of the form $f(\vec{x})=a\cdot\vec{x}, a\in\mathbb{R}$. An M-neuron defined as $F_1^*(\vec{x})=\langle\vec{x},a\cdot\vec{w}\rangle$ is equivalent to the considered neuron, because

$$F^*(\vec{x})=f(\langle\vec{x},\vec{w}\rangle)=a\cdot\langle\vec{x},\vec{w}\rangle=\langle\vec{x},a\cdot\vec{w}\rangle=F_1^*(\vec{x}).$$

□

By the following theorem, it is sufficient to consider only one-layer linear ANNs.

Theorem 2.2.12 *Every linear multilayer* M*-ANN is equivalent to a one-layer* M*-ANN.*

PROOF

Since every multilayer network consists of a finite number of layers it is sufficient to prove the theorem for a two-layer network. By Proposition 2.2.11, we can consider neurons in which the activation function is an identity function.

Consider a neuron of the output layer. Its output signal is given by

$$y_{2,t}=\langle\vec{y}_{we},\vec{w}_{2,t}\rangle,$$

where $\vec{y}_{we}$ is its input signal with components being output signals of the input layer neurons. The vector $\vec{w}_{2,t}$ is the weights vector of the t^{th} neuron of the output layer. The vector $\vec{y}_{we}$ in an orthonormal base is of the form:

$$\vec{y}_{we}=\sum_{m=1}^{M_2}\vec{y}_{we_m}\hat{e}_m=\sum_{m=1}^{M_2}\langle\vec{x},\vec{w}_{1,m}\rangle\hat{e}_m,$$

where $\vec{x}$ is the network input signal. By scalar product properties we have:

$$y_{2,t}=\sum_{m=1}^{M_2}\langle\langle\vec{x},\vec{w}_{1,m}\rangle\hat{e}_m,\vec{w}_{2,t}\rangle=\sum_{m=1}^{M_2}\langle\vec{x},\vec{w}_{1,m}\rangle\langle\hat{e}_m,\vec{w}_{2,t}\rangle.$$

By the orthonormality of the base:

$$y_{2,t}=\sum_{m=1}^{M_2}\langle\vec{x},\vec{w}_{1,m}\rangle\cdot w_{2,t,m}=\sum_{m=1}^{M_2}\langle\vec{x},w_{2,t,m}\cdot\vec{w}_{1,m}\rangle.$$

The last equation means that, for the same input signal, the output signals of the considered two-layer network and a one-layer network with proper weights are identical. This means that these networks are equivalent. □

Propositions 2.2.8 and 2.2.7 and Lemma 2.2.12 imply the following corollary.

Corollary 2.2.13 *The function* $\mathscr{S}$ *generated by a trained* M*-ANN over a set* X *is a linear operator on* X.

Assume that a R_1-layer k-ANN $(X,\mathscr{A}_k,Y)$, $Y\subset\mathbb{R}^m$, with the corresponding function $\mathscr{S}:X\longrightarrow Y$, and an m-ANN R_2-layer ANN, with the function $\mathscr{S}^*:Y\longrightarrow Z$, are given. Let, furthermore, output signals of $(X,\mathscr{S}_k,Y)$ be put to the input of $(Y,\mathscr{S}^*,Z)$. In this way, a R_1+R_2-layer network is obtained. This can be defined formally as follows.

Definition 2.2.14 *Let architectures A_k and A^*_m of the networks $(X,\mathscr{A}_k,Y)$, $Y\subset\mathbb{R}^m$ and $(Y,\mathscr{A}^*,Z)$ specified above be of the following form:*

$$\mathscr{A}_k:=(G=(A,\mathscr{E}),\gamma,\alpha,W,\beta),$$

$$\mathscr{A}^*_m:=(G^*=(A^*,\mathscr{E}^*),\gamma^*,\alpha^*,W^*,\beta^*).$$

Construct the new k-ANN $(X,\tilde{A}_k,Z)$ in such a way that its architecture

$$\tilde{\mathscr{A}}_k:=(\tilde{G}=(\tilde{A},\tilde{\mathscr{E}}),\tilde{\gamma},\tilde{\alpha},\tilde{W},\tilde{\beta})$$

is defined as follows:

$\tilde{A}:=A\cup A^*$,
$\tilde{\mathscr{E}}:=\mathscr{E}\cup\mathscr{E}^*\cup\{(a_i,a_j):a_i\in A,\delta^-_{a_i}=0,a_j\in A^*,\delta^+_{a_j}=0\}$,
$\tilde{\gamma}:\tilde{A}\ni a\longmapsto\tilde{\gamma}(a)\in\{1,\ldots,\delta_{\tilde{G}}\}$ – *a bijection mapping,*
$\tilde{\alpha}:\tilde{A}\ni a\longmapsto\tilde{\alpha}(a)\in F$ *is a mapping which satisfies the conditions* $\tilde{\alpha}|A\equiv\alpha$ *and* $\tilde{\alpha}|A^*\equiv\alpha^*$,
$\tilde{\beta}:\tilde{\mathscr{E}}\longrightarrow\widetilde{W}$ *is a bijection mapping, whereas* $\widetilde{W}$ *is a set of indices (see Definition 2.2.4).*

Such operation is said to be a superposition of networks A_k and A^*_m.

The definitions of multilayer networks and network superposition imply the following two corollaries.

Corollary 2.2.15 *A multilayer ANN is a superposition of one-layer networks of which it consists.*

Corollary 2.2.16 *Let a multilayer ANN $(X,\tilde{A}_k,Z)$ with the corresponding function $\tilde{\mathscr{S}}:X\longrightarrow Z$ be a superposition of $(X,\mathscr{A}_k,Y)$ and $(Y,\mathscr{A}^*_m,Z)$, with corresponding functions $\mathscr{S}:X\longrightarrow Y$ and $\mathscr{S}^*:Y\longrightarrow Z$, respectively. Then:*

$$\tilde{\mathscr{S}}=\mathscr{S}^*\circ\mathscr{S}.$$

Mulitilayer ANNs should act correctly. This means that in response to a given input signal the output signal should be equal to the desired value. If some of these desired output values are known, the training sequence, used in the sequel in the training process, can be defined.

Definition 2.2.17 *A finite sequence of pairs*

$$\left((\vec{x}^{\,(1)},\vec{z}^{\,(1)}),\ldots,(\vec{x}^{\,(N)},\vec{z}^{\,(N)})\right),$$

where $\vec{x}^{\,(i)}$ are input signals of a k-ANN on X and $\vec{z}^{\,(i)}$ is a required output signal, is called a training sequence of the multilayer k-ANN on X.

An adjustment of weights in such a way that differences between real and desired outputs in response to a given input signal are as small as possible is called the training process of an ANN. This problem can be reduced to finding

a minimum of a certain function. Assume that a multilayer ANN and a training sequence $\left((\vec{x}^{(1)},\vec{z}^{(1)}),\ldots,(\vec{x}^{(N)},\vec{z}^{(N)}\right)$ are given.

Definition 2.2.18 *A mapping*

$$E:\mathbb{R}^{s_1}\ni\mathbf{w}\longmapsto E(\mathbf{w})\in[0,\infty)$$

is said to be the error function if $E(\mathbf{w})$ *is of the form:*

$$U\left(\vec{y}^{(1)}(\mathbf{w}),\ldots,\vec{y}^{(N)}(\mathbf{w})\right),$$

where

$$U:\left(\mathbb{R}^{M_R}\right)^N\longrightarrow[0,\infty)\ \textit{and}\ \vec{y}^{(n)}:\mathbb{R}^{s_1}\longrightarrow\mathbb{R}^{M_R}.$$

The number of inputs in an output layer neuron is equal to s_1 *whereas* M_R *is the number of neurons in the output layer. Furthermore, it is required that satisfying the following equalities*

$$\vec{y}^{(1)}(\mathbf{w})=\vec{z}^{(1)},\ldots,\vec{y}^{(N)}(\mathbf{w})=\vec{z}^{(N)}$$

for every training sequence

$$\left((\vec{x}^{(1)},\vec{z}^{(1)}),\ldots,(\vec{x}^{(N)},\vec{z}^{(N)})\right),$$

is a necessary and sufficient condition of zeroing the function E *at the point* $\mathbf{w}$*.*

Remarks

1. The vector of all weights in an ANN is denoted by $\mathbf{w}$. Components of this vector are indexed by the set W_1.
2. Vectors $\vec{y}^{(n)}(\mathbf{w})\in Y$ are network output signals which are responses to the input signals $\vec{x}^{(n)}$ being elements of a training set.
3. In most cases, it is additionaly assumed that an error function is at least of class C^1.
4. In most cases, it is impossible to find the global minimum of the deviation function. Therefore, numerical methods which allow to find its local minimum are used.

There are several methods of artificial neural networks training. Most of them are iterative processes. Below, we shall consider the gradient descent method, which is one of the most common ways of training multilayer ANNs.

Let a training sequence $\left(\vec{x}^{(n)},\vec{z}^{(n)}\right)_{n=1,2,\ldots,N}$ and a multilayer ANN $\mathscr{S}_k(X,Y)$ with differentiable activation functions of all neurons be given. If the descent gradient method is used in ANN training, then the iteration rule of weights modification is of the form (*cf.* [2] Section.2.4.1, [3] Section.2.5.2):

$$w_\iota^{(p+1)}=w_\iota^{(p)}-\eta\cdot\frac{\partial E(\mathbf{w}^{(p)})}{\partial w_\iota},\tag{1}$$

where the superscript index (p) is the number of an iteration and $\mathbf{w}=\{w_\iota\}_{\iota\in W_1}$ is a vector of all weights in the ANN. Formula (1) is the Euler method for the gradient differential equation:

$$\frac{dw_\iota}{dt}=-\operatorname{grad}E(\mathbf{w}).\tag{2}$$

Thus, an error function plays the role of the potential E in the gradient of Equation (2). Since, for a sufficiently small η, Equation (1) generates a discrete dynamical

system (a cascade), whereas Equation (2) generates a continuous dynamical system (a flow), dynamical systems theory can be used to analyse the training process.

Remark

The square deviation function is used most often. It is of the form:

$$E(\mathbf{w}) = \sum_{n=1}^{N} \sum_{t=1}^{T_R} \left[y_{R,t}^{(n)} - z_t^{(n)} \right]^2, \tag{3}$$

where $y_{R,t}^{(n)} = f_{R,t}\left(\vec{w}_{R,t} \circ \vec{x}^{(n)}\right)$, $f_{R,t}$ is an activation function of the t^{th} neuron in the R^{th} layer, and $\vec{w}_{R,t} = [w_{R,t,1}, \ldots, w_{R,t,m}]$. Thus, the indexing convention is as follows:

$n = 1, \ldots, N$ – the number of a vector in the training set – the superscript in brackets,

$r = 1, \ldots, R$ – the number of an ANN layer,

$t = 1, \ldots, T_r$ – the number of a neuron in the r^{th} layer,

$m = 1, \ldots, M_r$ – the neuron input number in the r^{th} layer.

Derivatives of hidden layer neurons in Equations (1) and (2) are calculated using the back-propagation method.

3. Learning process analysis

3.1. Linear ANNs

In most cases, the least-squares method is used for analysis of the learning process of a linear ANN [38].

The properties of Gram matrices will be used to analyse linear training processes of ANNs analysis. Let us recall their definition.

Definition 3.1.1 *A real matrix G of a dimension $n \times n$ is said to be a Gram matrix of a vector family $\vec{v}_1, \ldots, \vec{v}_n$ if $g_{ij} = \langle \vec{v}_i, \vec{v}_j \rangle$, where $\langle \vec{v}_i, \vec{v}_j \rangle$ denotes a real scalar product of vectors $\vec{v}_i$ and $\vec{v}_j$. The Gram matrix will be denoted by $G(\vec{v}_1, \ldots, \vec{v}_n)$.*

Consider an ANN consisting of a single linear M-neuron. By Proposition 2.2.11, it is sufficient to consider a neuron whose activation function is identity. In such a case, the square deviation function is given by:

$$E(w_1, \ldots, w_M) = \sum_{n=1}^{N} \left[y(w_1, \ldots, w_M)^{(n)} - z^{(n)} \right]^2,$$

where

$$y(w_1, \ldots, w_M)^{(n)} = \sum_{m=1}^{M} x_m^{(n)} w_m.$$

Calculate a value of the right side of the equation describing the learning process:

$$\frac{\partial E(w_1, \ldots, w_M)}{\partial w_{m'}} = \frac{\partial}{\partial w_{m'}} \sum_{n=1}^{N} \left[\sum_{m=1}^{M} x_m^{(n)} w_m - z^{(n)} \right]^2.$$

Setting

$$H^{(n)} := \sum_{m=1}^{M} x_m^{(n)} w_m - z^{(n)},$$

we obtain

$$\frac{\partial E(w_1,\ldots,w_M)}{\partial w_{m'}} = \sum_{n=1}^{N} 2\cdot H^{(n)}\cdot\frac{\partial(y^{(n)}-z^{(n)})}{\partial w_{m'}} = 2\cdot\sum_{n=1}^{N} H^{(n)}\cdot\frac{\partial y^{(n)}}{\partial w_{m'}} =$$

$$= 2\cdot\sum_{n=1}^{N} H^{(n)}\cdot\frac{\partial(\sum_{m=1}^{M} x_m^{(n)} w_m)}{\partial w_{m'}} = 2\cdot\sum_{n=1}^{N} H^{(n)}\cdot x_{m'}^{(n)} =$$

$$= 2\cdot\sum_{n=1}^{N} x_{m'}^{(n)}\cdot\left[\left(\sum_{m=1}^{M} x_m^{(n)} w_m\right) - z^{(n)}\right].$$

We have seen that the equation describing the training process of a linear neuron is a homogeneous linear differential equation which can be written as follows:

$$\frac{d\vec{w}}{dt} = -2\cdot(A\cdot\vec{w} - B),$$

where A is a Gram matrix

$$A = G(\overline{x}_1,\ldots,\overline{x}_M).$$

The signals on the i^{th} neuron input are components of the N-dimensional vector $\overline{x}_m$, $m = 1,\ldots,M$.

Thus if a neuron has M inputs and a training sequence consists of N vectors $\vec{x}^{\,(n)}$, then the matrix $A = G(\overline{x}_1,\ldots,\overline{x}_M)$ and

$$a_{ij} = \overline{x}_i \circ \overline{x}_j.$$

The components of the M-dimensional vector B are of the form:

$$b_m = \overline{x}_m \circ \vec{z}, \text{ where } \vec{z} = (z^{(1)},\ldots,z^{(N)}).$$

Remark

A linear one-layer ANN learning process is described by the system of T differential equations which are independent of each other. Each of them models the learning process of a single neuron. Indeed:

$$\frac{\partial E}{\partial w_{t',m'}} = \frac{\partial}{\partial w_{t',m'}}\sum_{n=1}^{N}\sum_{t=1}^{T}\left[\left(\sum_{m=1}^{M} x_m^{(n)} w_{t,m}\right) - z_t^{(n)}\right]^2 =$$

$$= \frac{\partial}{\partial w_{t',m'}}\left\{\sum_{n=1}^{N}\left\{\sum_{t=1,t\neq t'}^{T}\left[\left(\sum_{m=1}^{M} x_m^{(n)} w_{t,m}\right) - z_t^{(n)}\right]^2\right\} + \right.$$

$$\left. + \frac{\partial}{\partial w_{t',m'}}\sum_{n=1}^{N}\left[\left(\sum_{m=1}^{M} x_m^{(n)} w_{t',m}\right) - z_t^{(n)}\right]^2\right\}.$$

Since the first component does not depend on $w_{t',m'}$, it is equal to zero. Thus:

$$\frac{\partial E}{\partial w_{t',m'}} = \frac{\partial}{\partial w_{t',m'}}\sum_{n=1}^{N}\left[\left(\sum_{m=1}^{M} x_m^{(n)} w_{t',m}\right) - z_t^{(n)}\right]^2 =$$

$$= 2\cdot\sum_{n=1}^{N} x_{m'}^{(n)}\left[\left(\sum_{m=1}^{M} x_m^{(n)} w_{t',m}\right) - z_{t'}^{(n)}\right].$$

We have developed a system of equations indexed by $t' \in \{1,\ldots,T\}$. It can be recapitulated in the following way.

Proposition 3.1.2 *A learning process of a linear M-ANN consisting of T neurons is modelled by the system of equations:*

$$\frac{d\vec{w}_t}{dt} = -2\cdot(A\cdot\vec{w}_t - B_t), \quad t\in\{1,\ldots,T\}, \tag{4}$$

where $A = G(\overline{x}_1,\ldots,\overline{x}_M)$ *is the matrix from the equation describing the learning process of a single neuron and, for a given* $t = t'$, B_t *is a vector with components being given by formulae*

$$b_{m,t'} = \overline{x}_m \circ \vec{z_{t'}}, \quad \vec{z_{t'}} = (z_{t'}^{(1)},\ldots,z_{t'}^{(N)}).$$

Remarks

1. By Lemma 2.2.12 each multilayer linear ANN is equivalent to a one-layer linear ANN (see, for instance, [6]). Therefore Equation (4) describes the learning process of a linear ANN in the most general terms.
2. The found system of equations can be written in the following way:
$$\frac{d\mathbf{W}}{dt} = -2\cdot(A\cdot\mathbf{W} - \mathbf{B}),$$
where $\mathbf{W}$ is the matrix with the t^{th} column being a vector $\vec{w}_t$, whereas $\mathbf{B}$ is the matrix consisting of elements $b_{m,t} = \overline{x}_m \circ \vec{z}_t$.

Example. Consider the learning process of a single linear neuron having two inputs. Let a learning sequence consists of three components.

$$\frac{dw_1}{dt} = -2\sum_{n=1}^{3} x_1^{(n)}\cdot[(x_1^{(n)}w_1 + x_2^{(n)}w_2) - z^{(n)}] =$$
$$-2\cdot(x_1^{(1)}x_1^{(1)}w_1 + x_1^{(1)}x_2^{(1)}w_2 - x_1^{(1)}z^{(1)} + x_1^{(2)}x_1^{(2)}w_1 + x_1^{(2)}x_2^{(2)}w_2 - x_1^{(2)}z^{(2)} +$$
$$+x_1^{(3)}x_1^{(3)}w_1 + x_1^{(3)}x_2^{(3)}w_2 - x_1^{(3)}z^{(3)}) =$$
$$= -2\cdot[(x_1^{(1)}x_1^{(1)} + x_1^{(2)}x_1^{(2)} + x_1^{(3)}x_1^{(3)})w_1 + +(x_1^{(1)}x_2^{(1)} + x_1^{(2)}x_2^{(2)} + x_1^{(3)}x_2^{(3)})w_2 -$$
$$-x_1^{(1)}z^{(1)} - x_1^{(2)}z^{(2)} - x_1^{(3)}z^{(3)}] =$$
$$= -2\cdot[(\overline{x}_1\circ\overline{x}_1)\cdot w_1 + (\overline{x}_1\circ\overline{x}_2)\cdot w_2 - (\overline{x}_1\circ\vec{z})]$$

The derivative $\dfrac{dw_2}{dt}$ can be calculated in the same way. Thus, we have procured that:

$$\frac{dw_1}{dt} = -2\cdot[(\overline{x}_1\circ\overline{x}_1)\cdot w_1 + (\overline{x}_1\circ\overline{x}_2)\cdot w_2 - (\overline{x}_1\circ\vec{z})],$$
$$\frac{dw_2}{dt} = -2\cdot[(\overline{x}_2\circ\overline{x}_1)\cdot w_1 + (\overline{x}_2\circ\overline{x}_2)\cdot w_2 - (\overline{x}_2\circ\vec{z})],$$

which can be written as:

$$\frac{d\vec{w}}{dt} = -2\cdot(A\cdot\vec{w} - B),$$

where

$$A = \begin{pmatrix} \overline{x}_1\circ\overline{x}_1 & \overline{x}_1\circ\overline{x}_2 \\ \overline{x}_2\circ\overline{x}_1 & \overline{x}_2\circ\overline{x}_2 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} \overline{x}_1\circ\vec{z} \\ \overline{x}_2\circ\vec{z} \end{pmatrix}.$$

3.2. Stability of the learning process of a linear ANN

The definitions of stability and asymptotical stability of a dynamical system can be found in [39], Chapters 2.1 and 2.6, and in [40], Chapter 1.1. Furthermore, the following theorem is satisfied for linear flows (see [39], page 95).

Theorem 3.2.1 *A flow generated by a linear differential equation is asymptotically stable if and only if every orbit converges to zero.*

Thus, a linear flow is asymptotically stable if and only if it is globally asymptotically stable.

In order to show asymptotical stability of a neural network learning process we will use the Hurwitz criterion ([39], page 116).

Theorem 3.2.2 *Let*

$$\frac{d\vec{x}}{dt} = A\vec{x}$$

be a linear equation in $\mathbb{R}^n$*, where* A *is a given real matrix. The flow generated by the equation is asymptotically stable if and only if the following conditions are satisfied:*

$$\begin{aligned}
&\Delta_1 = -A_1 > 0,\\
&\Delta_2 = -A_1A_2 + A_3 > 0,\\
&\Delta_3 = (-1)^3 A_3\Delta_2 > 0,\\
&\vdots\\
&\Delta_n = (-1)^n A_n\Delta_{n-1} > 0,
\end{aligned}$$

where A_k *is the sum of all principal minors of the matrix* A*.*

The matrix A is a Gram matrix in the equation describing a learning process of a linear neuron. Let us recall the basic properties of Gram matrices.

Theorem 3.2.3 *If vectors* $v_1,\ldots,v_k$ *are linearly independent, then* $\det G(v_1,\ldots,v_k)>0$*; otherwise* $\det G(v_1,\ldots,v_k)=0$*.*

Theorem 3.2.4 *The inequality*

$$\det G(v_1,\ldots,v_m,v_{m+1},\ldots,v_k) \le \det G(v_1,\ldots,v_m)\cdot\det G(v_{m+1},\ldots,v_k).$$

is satisfied for every finite family $\{v_k\}_{k=1}^K$ *of vectors.*

Proofs of these theorems can be found in [41], pages 335, 337 and 338.

Considering the learning process of a linear artificial neural network we can restrict our considerations, without losing generality, to the case of only one neuron (see Subsection 3.1). Let us consider a dynamical system modelling such a case.

Theorem 3.2.5 *The flow generated by equation*

$$\frac{d\vec{w}}{dt} = -2\cdot(G(\overline{x}_1,\ldots,\overline{x}_M)\cdot\vec{w} - B) \tag{6}$$

is asymtotically stable if and only if vectors $\{\overline{x}_1,\ldots,\overline{x}_M\}$ *are linearly independent.*

Proof

It is sufficient to prove asymptotical stability of the equation

$$\frac{d\vec{w}}{dt} = -2\cdot G(\overline{x}_1,\ldots,\overline{x}_M)\cdot\vec{w}.$$

(see [39], page 91, Corollary 2).

It is easy to show that the assumptions of the Hurvitz criterion are satisfied. Indeed:

$$\Delta_1 = -A_1 = -(-2)\cdot \mathrm{Tr}\, G(\overline{x}_1,\ldots,\overline{x}_M) = 2\cdot \sum_{m=1}^{M} \overline{x}_m \circ \overline{x}_m > 0,$$

$$\Delta_2 = -A_1\cdot A_2 + A_3 > 0.$$

This can be written as:

$$A_3 > A_1\cdot A_2, \text{ thus } (-2)^3\cdot G_3 > (-2)^3\cdot G_1\cdot G_2,$$

where G_k is the sum of all principal minors of the rank k of the matrix G.

We have obtained that:

$$G_3 < G_1\cdot G_2. \tag{7}$$

On the left side of the inequality, we have the sum of all principal minors of rank 3. It is easy to show that for every left side component there exists a component on the right side such that Theorem 3.2.4 can be applied. Furthermore, on the right side there exist additional positive components (their positivness is implied by the linear independence of the vectors of the matrix G). Thus inequality (7) is satisfied.

The remaining assumptions of the Hurvitz criterion:

$$\Delta_n = (-1)^n\cdot A_n\cdot \Delta_{n-1} > 0,$$

where $A_n = (-2)^n\cdot G_n$, are satisfied if and only if the vectors $\overline{x}_1,\ldots,\overline{x}_M$ are linearly independent (see Theorem 3.2.3). □

We have proved that the dynamics generated by the equation

$$\frac{d\vec{w}}{dt} = -2\cdot A\cdot \vec{w} - B,$$

which models the artificial neural network learning process, is asymptotically stable, thus, it is globally asymptotically stable (see Theorem 3.2.1) if only vectors $\overline{x}_1,\ldots,\overline{x}_M$ are linearly independent. By Theorem 3.2.1, if the matrix G is nonsingular, then the flow generated by the differential Equation (6) has only one hyperbolic stable point, which is globally attracting.

Consider topological conjugacy between the cascade obtained via discretization of the flow $(\phi_h, \mathbb{R}^n)$ which describes the learning process and the cascade $(\psi_h, \mathbb{R}^n)$, generated by the Euler method for a sufficiently small h. By the Grobman-Hartman Theorem (see [42], page 60, Theorem 4.1), in a neighbourhood U of the hyperbolic fixed point of the flow ϕ, the cascades ϕ_h and ψ_h are locally topologically conjugate. Assuming that a homeomorphism H is conjugating, it can be shown that the mapping $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ defined by:

$$F(x) = \psi_h^n(H(\phi_h^n(x))),$$

is a globally conjugating homeomorphism (n is the smallest natural number such that $\phi(x, nh) \in U$). The existence of global conjugacy implies that the dynamics of the model of the learning process and the Euler method implemented on a computer are the same.

It has been proven that the learning process of a linear ANN is globally asymptotically stable if vectors $\overline{x}_1,\ldots,\overline{x}_M$ are linearly independent. It is easy to show that the linear independence is a generic property. Let us recall the definition (see [39], page 107).

Definition 3.2.6 *Let X be a topological space. A property is said to be generic if the set having this property is open and dense in X.*

Proposition 3.2.7 *Let X be a family of all sets $\{Y_M\}_J$ such that each of them consists of M vectors in $\mathbb{R}^N$ and let $M \leq N$. The linear independence of elements of a set $\{Y_M\}_J$ is a generic property in X.*

PROOF

Assume that a family consisting of M vectors from N-dimensional space is given. Let an $N \times M$ matrix be generated by the family in such a way that its columns are vectors from the family. The vectors are linearly independent if and only if each square submatrix is nonsingular. Thus, the proposition is implied by a well known fact that the set of all nonsingular $L \times L$ real matrices is open and dense in the set of all $L \times L$ real matrices. □

Thus, the linear independence of training sequence vectors $\overline{x}_1, \ldots, \overline{x}_M$ is a generic property. This means that the set of learning sequences consisting of linearly independent vectors is open and dense in the set of all learning sequences. Therefore the possibility of accidental generation of a learning sequence consisting of linearly dependent vectors can be neglected.

3.3. Weakly nonlinear neural networks

Dynamics of linear dynamical systems is a well investigated problem. We can take as an example the theorem about global topological conjugacy between cascades generated by a linear flow. In such problems, theorems describing properties of linear systems are also frequently true for weakly distorted linear systems. Thus, for instance, according to the Hartman theorem (see [43], page 114, Theorem 5.15), cascades generated by weakly distorted operators in a Banach space are globally conjugate, whereas the conjugacy of weakly distorted linear flows is the subject of the Grobman Theorem (see [43], page 117, Theorem 5.25). In the Fečkan Theorem [18], the relation between a discretization of a weakly perturbed linear flow and its Euler method is considered. Let us recall this result.

Theorem 3.3.1

Let $(\phi, \mathbb{R}^m)$ be the flow generated by the equation:

$$\dot{x} = Ax + g(x), \tag{8}$$

where $A \in \mathscr{L}(\mathbb{R}^m)$ has no eigenvalues on the imaginary axis, $g \in C^1(\mathbb{R}^m, \mathbb{R}^m)$, $g(0) = 0$, and let for each $x \in \mathbb{R}^n$ and a sufficiently small b the following inequalities hold:

$$\sup |g(x)| < \infty \text{ and } |Dg(x)| \leq b.$$

Then, for a compact set $K \subset \mathbb{R}^m$ there exists a number $h_o > 0$ and a C^o-mapping

$$H_h : \mathbb{R}^n \longrightarrow \mathbb{R}^n, \qquad h \in (0, h_o),$$

such that on the set K the following equality holds:

$$\phi(\cdot, h) \circ H_h(\cdot) = H_h(\cdot) \circ \psi_h(\cdot),$$

where ψ_h is the map given by the Euler method:

$$\psi_h(x) = x + h \cdot Ax + h \cdot g(x).$$

The set K can be considered to be a ball $B_q := \{x, |x| \leq q\}$ (see [18], page 124).

We use the Grobman-Hartman and the Fečkan Theorems to analyse the iterative process which is used to train the so-called weakly nonlinear neurons. Let us introduce the following definition.

Definition 3.3.2 *A neuron is called weakly nonlinear if its activation function is of the form:*

$$f : \mathbb{R} \ni \beta \longmapsto f(\beta) = \beta + u(\beta),$$

where u is limited, and $u \in C^2(\mathbb{R})$, and, furthermore, for each $\beta \in \mathbb{R}$ we have $|u'(\beta)| < c_1$, $|u''(\beta)| < c_2$, $|\beta \cdot u'(\beta)| < c_3$ and $|\beta \cdot u''(\beta)| < c_4$, where constants c_1, c_2, c_3 and c_4 are sufficiently small.

It seems that, weakly nonlinear neurons have been considered so far only by Bielecki [26, 36].

Theorem 3.3.4, the main theorem of this subsection, implies that the learning process of a weakly nonlinear neuron has the same dynamics as that of a linear dynamical system. The following theorem demonstrates this.

Theorem 3.3.3 *The process generated by the descent gradient learning method of a weakly nonlinear neuron is modelled by a differential equation of a type (8) which satisfies assumptions of Theorem 3.3.1.*

Proof

Let a neuron having M-componental input and an N-componental training sequence $((\vec{x}^{(1)}, z^{(1)}), \ldots, (\vec{x}^{(N)}, z^{(N)}))$ be given. Then a total excitation of a neuron in the n^{th} step of a learning process is given by the formula:

$$\beta^{(n)} = \sum_{m=1}^{M} x_m^{(n)} w_m^{(n)}.$$

Assuming that the considered neuron is weakly nonlinear, its activation function is of the form:

$$f(\beta^{(n)}) = \beta^{(n)} + u(\beta^{(n)}),$$

whereas the square deviation function is given as follows:

$$E(w_1, \ldots, w_M) = \frac{1}{2} \sum_{n=1}^{N} \left[\beta^{(n)} + u(\beta^{(n)}) - z^{(n)} \right]^2.$$

Let us compute the k^{th} component of the deviation function gradient:

$$\frac{\partial E}{\partial w_k} = \sum_{n=1}^{N} \left\{ \left[\beta^{(n)} + u(\beta^{(n)}) - z^{(n)} \right] \cdot \left[x_k^{(n)} + u'(\beta^{(n)}) \cdot x_k^{(n)} \right] \right\} =$$

$$= \sum_{n=1}^{N} x_k^{(n)} \beta^{(n)} - \sum_{n=1}^{N} x_k^{(n)} \cdot z^{(n)} + \sum_{n=1}^{N} x_k^{(n)} \cdot \left[u(\beta^{(n)}) + u'(\beta^{(n)}) \cdot f(\beta^{(n)}) - z^{(n)} \cdot u'(\beta^{(n)}) \right].$$

Thus, the gradient differential equation modelling the learning process

$$\dot{\vec{w}} = -\operatorname{grad} E(\vec{w})$$

can be written in the following form:

$$\dot{\vec{w}} = -(A\vec{w} + B + \tilde{g}(\vec{w})).$$

Let us assume that the neuron input is M-componental and the learning sequence consists of N elements. The matrix A is given as follows:

$$A = \begin{pmatrix} \overline{x}_1 \circ \overline{x}_1 & \dots & \overline{x}_1 \circ \overline{x}_M \\ \vdots & & \vdots \\ \overline{x}_M \circ \overline{x}_1 & \dots & \overline{x}_M \circ \overline{x}_M \end{pmatrix}, \tag{9}$$

where $\overline{x}_m$ is an N-componental vector. Signals given on the m^{th} input of the neuron are its components and $m = 1,\dots,M$. Thus, the matrix A is a Gram matrix generated by the vectors of the training set:

$$A := G(\overline{x}_1,\dots,\overline{x}_M).$$

The matrix B is of the form:

$$B = \begin{pmatrix} \sum_{n=1}^{N} x_1^{(n)} \cdot z^{(n)} \\ \vdots \\ \sum_{n=1}^{N} x_M^{(n)} \cdot z^{(n)} \end{pmatrix}. \tag{10}$$

If the matrix $G(\overline{x}_1,\dots,\overline{x}_M)$ is nonsingular, then the flows generated by equations $\dot{\vec{w}} = -A\vec{w} + B$ and $\dot{\vec{w}} = -A\vec{w}$ are conjugate. As it was mentioned at the previous subsection, linear independence of vectors $\overline{x}_1,\dots,\overline{x}_M$ is a generic property. Since linear independence implies nonsingularity, the topological conjugacy of the flows is generic as well. Conjugacy is transitive, thus it is sufficient to consider the equation:

$$\dot{\vec{w}} = -(A\vec{w} + \tilde{g}(\vec{w})).$$

The k^{th} component of vector $\tilde{g}(\vec{w})$ has the form:

$$\tilde{g}(\vec{w})_k = \sum_{n=1}^{N} x_k^{(n)} \cdot \left[u(\beta^{(n)}) + u'(\beta^{(n)}) \cdot \beta + u'(\beta^{(n)}) \cdot u(\beta^{(n)}) - z^{(n)} \cdot u'(\beta^{(n)}) \right].$$

The derivative matrix of $\tilde{g}(\vec{w})$ can be written as follows:

$$\frac{\partial \tilde{g}(\vec{w})_k}{\partial w_s} = \sum_{n=1}^{N} x_k^{(n)} \cdot x_s^{(n)} \cdot [u'(\beta^{(n)}) + u''(\beta^{(n)}) \cdot \beta +$$

$$+ u''(\beta^{(n)}) \cdot u(\beta^{(n)}) + u'(\beta^{(n)}) \cdot (1 + u'(\beta^{(n)})) - z^{(n)} \cdot u''(\beta^{(n)})].$$

Thus, if the mapping u satisfies the assumptions specified in Definition 3.3.2, then the mapping $\tilde{g}$ satisfies the assumptions of Theorem 3.3.1. This completes the proof. □

The map

$$u(\beta) = \frac{c \cdot \beta}{\beta^2 + 1},$$

where the constant c is sufficiently small, can be put as an example of a function satisfying assumptions concerning the perturbing map g in the definition of the weakly nonlinear neuron.

The properties of the function u specified in Definition 3.3.2 imply that assumptions of the Grobman-Hartman Theorem are satisfied as well. This means that the following theorem holds.

Theorem 3.3.4 *The flow generated by the equation:*

$$\dot{\vec{w}} = -(A\vec{w} + \tilde{g}(\vec{w})) \tag{11}$$

is globally topologically conjugate to the flow ganerated by its linear part:

$$\dot{\vec{w}} = -A\vec{w}.$$

Furthermore, the cascade generated by the time-h-map discretization of the flow generated by the Equation (11) is, on a large ball, conjugate to the cascade generated by the Euler method of this equation.

Let us remark that the range of number values which can be represented in a computer is limited. Furthermore, in a biological neural cell, neurotransmitters are liberated in tiny amounts from vesicles – about 10^{-17} mol acetylcholin per impulse (see [44], page 5, [45], page 39). The input impulse cannot be too large or a cell will be destroyed. Thus, both in biological and artificial neural networks, absolute values of vectors $\vec{w}$ and $\vec{x}$ are bounded and, therefore, modelling numerically both biological and artificial neurons we can considered only bounded vectors $\vec{w}$ and $\vec{x}$. Therefore, considering topological conjugacy on a sufficiently large ball is adequate for training process analysis.

The learning process is implemented on a computer according to the Euler method. Theorem 3.3.4 ensures that its dynamics is the same as the dynamics of a linear cascade if only the matrix A has no eigenvalues on the imaginary axis. It implies, inter alia, asymptotical stability of a learning process. Furthermore, the cascade generated by the discretization of the Equation (11) is, on a sufficiently large ball, topologically conjugate to the cascade generated by the Euler method (for a sufficiently small time step). The conjugacy on a large ball is sufficient for analysis of neural networks, as norms of weights vectors and vectors of training sequence are limited. However, it is necessary to estimate the value of h_o (see Theorem 3.3.1) in order to apply the results of our analysis to network implementations. This estimation has the following form (see [46]).

Theorem 3.3.5 *Let $0 < h < \|A\|^{-1}$ be fixed. Under the notations of the Fečkan Theorem, if the following inequalities are true:*

$$\begin{aligned} &h \cdot b < (1-M) \cdot \|A_h\|^{-1}, \\ &h \cdot b \cdot (\|A\| + b) < (1-M) \cdot (\|A_h^{-1}\| \cdot \|e^{A \cdot h}\|), \end{aligned} \tag{12}$$

where $A_h := id + h \cdot A$ and $M := \max\{\|(A_u)^{-1}\|, \|A_s\|\}$, then the conclusion of the Fečkan Theorem holds.

As the matrix A has no imaginary eigenvalues, the linear operator A is hyperbolic, which means that there exists an invariant splitting $\mathbb{R}^n = E^s \oplus E^u$ in which $\|A_s\| \leq a < 1$, $\|(A_u)^{-1}\| \leq a < 1$, where $\|A_s\| = A|E^s : E_s \longrightarrow E_s$ and $\|A_u\| = A|E^h : E_u \longrightarrow E_u$.

3.4. Nonlinear neural networks

In this subsection, we apply the fact that, on a two-dimensional sphere S^2, a gradient dynamical system is, under some natural assumptions, correctly reproduced by its Euler method for a sufficiently small time step. This means that the time-h-map

of the induced dynamical system is globally topologically conjugate to the discrete dynamical system obtained using the Euler method. It can be expressed as follows.

Theorem 3.4.1 *Let S^2 be the two-dimensional sphere in $\mathbb{R}^3$ and let*

$$\phi : S^2 \times \mathbb{R} \longrightarrow S^2$$

be the dynamical system generated by a differential equation

$$\dot{x} = -\operatorname{grad} E(x), \tag{13}$$

where $E \in C^2(S^2, \mathbb{R})$ has a finite number of singularities, all of them hyperbolic. Let, furthermore, the dynamical system ϕ have no saddle-saddle connections. Moreover, let us assume that $\phi_h : S^2 \longrightarrow S^2$ is a discretization of the system ϕ, i.e. $\phi_h(x) := \phi(x,h)$, whereas $\psi_h : S^2 \longrightarrow S^2$ is a mapping generated by the Euler method for Equation (13). Then, for a sufficiently small $h > 0$, there exists a homeomorphism $\alpha = \alpha_h : S^2 \longrightarrow S^2$ globally conjugating cascades generated by ϕ_h i ψ_h, i.e. the following formula holds:

$$\phi_h \circ \alpha = \alpha \circ \psi_h. \tag{14}$$

The proof of the theorem is presented in [36] and [20]. In this paper we focus on the application of the presented result to the analysis of training process dynamics of a nonlinear neuron.

Remarks

1. As it is known, Axiom A and the strong transversality condition are equivalent to the structural stability of a dynamical system (see [42], page 171 and [47]). At the same time, for gradient dynamical systems, Axiom A implies that the system has only a finite number of singularities, all of them hyperbolic, whereas the strong transversality condition implies that the gradient system has no saddle-saddle connections. Thus, structural stability of the considered dynamical system (S^2, ϕ) implies assumptions of Theorem 3.4.1. Moreover, the set of structurally stable systems is open and dense in the space of gradient dynamical systems (see [42], page 116).
2. A dynamical system generated by Equation (13), having only a finite number of singularities, all of them hyperbolic, without saddle-saddle connections, is called a gradient Morse-Smale system.

Let us consider a one-layer artificial neural network consisting of nonlinear neurons. Let us also assume that an input of each neuron has two components. Furthermore, let us declare an activation function of each neuron to be a limited mapping of a class $C^2(\mathbb{R}, \mathbb{R})$, with the first and second derivatives bounded as well. Most types of activative functions used in practice (for instance bipolar and unipolar functions – see [3], page 38, most radial functions – see [3], page 168) satisfy the specified assumptions. Since neurons are trained independently in a one-layer net, we can consider the learning process only for a single neuron. Under such assumptions, Theorem 3.4.1 implies asymptotical stability of the learning process using the descent gradient method.

Theorem 3.4.1 has been proved for a sphere S^2, which can be identified, via homeographic projection, with the compactified plane $\overline{\mathbb{R}^2}$. This theorem can be used

to analyse the training process of a nonlinear neuron having two-componental input provided the potential E in gradient Equation (13) can become completed in the north pole of the sphere in such a way, that it remains a function of a class $C^2(S^2,\mathbb{R})$ and no nonhyperbolic fixed point shall appear. The error function plays the role of potential. The most commonly used square criterial function is, for a single neuron, given by the formula:

$$E(\vec{w}) = \sum_{n=1}^{N} \left[f(\beta^{(n)}) - z^{(n)} \right]^2, \qquad \beta^{(n)} := x_1^{(n)} \cdot w_1 + x_2^{(n)} \cdot w_2, \tag{15}$$

where $\vec{x}^{(n)} = [x_1^{(n)}, x_2^{(n)}]$ is an input vector, $z^{(n)}$ is a desired response of the neuron if a vector $\vec{x}^{(n)}$ is given to the input, and N is the number of input vectors used in the learning process.

The deviation function given by Equation (15) has various limits if an inverse image $\pi^{-1}(\vec{w})$ of a vector $\vec{w}$ (where π is the stereographic projection) converges to the north pole, *i.e.* $|\vec{w}|$ converges to infinity. It follows from the fact that we can converge to infinity in such a way, that the scalar product $\vec{x} \circ \vec{w}$ remains an arbitrary constant. However, it is possible to modify the criterial function so that, on a cartain ball $B((0,0),r)$, the potential will not be modified and it will be possible to complete it in the proper way. Furthemore, radius r can be as large as we need.

As it was mentioned in the preceding subsection, both in biological and in artificial neural networks, absolute values of vectors $\vec{w}$ and $\vec{x}$ are limited and, therefore, while mathematically modelling both biological and artificial neurons we can consider only bounded vectors $\vec{w}$ and $\vec{v}$.

Let us choose a sufficiently large radius a of the circle on which a training process is modelled. We can modify potential E using the function defined as follows:

$$g(\vec{w}) := \begin{cases} e^{(r-a)^n} & \text{for } r \geq a, \\ 1 & \text{for } r \in [0,a), \end{cases} \tag{16}$$

where $r := |\vec{w}|^2 = w_1^2 + w_2^2$ and a natural number n is selected depending on potential E and radius a in the way specified below. The function g is of a class $C^2(\mathbb{R}^2,\mathbb{R})$. Define the following mapping:

$$E^*(\vec{w}) := g(\vec{w}) \cdot E(\vec{w}). \tag{17}$$

Solutions of the equation

$$\dot{\vec{w}} = -\operatorname{grad} E^*(\vec{w}) \tag{18}$$

cut the circle $K((0,0),2a)$ transversally entering its interior, that is to say the scalar product $-\operatorname{grad} E^*(\vec{w}) \circ \vec{w}$ has negative values for $|\vec{w}|^2 = 2a$. Indeed, we have:

$$-\operatorname{grad} E^*(\vec{w}) \circ \vec{w} = \sum_{i=1}^{2} \left[E(\vec{w}) \cdot \frac{\partial g(\vec{w})}{\partial w_i} + g(\vec{w}) \cdot \sum_{n=1}^{N} \left[x_i^{(n)} \cdot \frac{df}{d\beta^{(n)}} \cdot \left(f(\beta^{(n)}) - z^{(n)} \right) \right] \right] \cdot w_i.$$

Because

$$\frac{\partial g(\vec{w})}{\partial w_i} := \begin{cases} 2 \cdot w_i \cdot n \cdot (r-a)^{n-1} \cdot e^{(r-a)^n} & \text{for } r > a, \\ 0 & \text{for } r \in [0,a], \end{cases} \tag{19}$$

thus for $r = 2a$, we obtain

$$-\operatorname{grad} E^*(\vec{w}) \circ \vec{w} = -e^{a^n} \sum_{i=1}^{2} \cdot \left[n \cdot a^{n-1} \cdot w_i^2 \cdot E(\vec{w}) + \sum_{n=1}^{N} \left[x_i^{(n)} \cdot \frac{df}{d\beta^{(n)}} \cdot \left(f(\beta^{(n)}) - z^{(n)} \right) \right] \cdot w_i \right].$$

Since the problem is considered on a closed ball, all variables and functions are bounded. The potential E is nonnegative, thus, for a large a, the first term in the sum asymptotically equals a^{n-1}, whereas the second one can be negative but is limited. Thus, as a is large, the number n can be chosen to be so large that the value of the first term is greater than the absolute value of the second component of the sum.

In order to properly complete the potential in the north pole, let us assume that, in a certain neighbourhood of the north pole, the potential is of the form $V(\vec{w}) = -s^2$, where s is the distance from the north pole. On the plain $\mathbb{R}^2$ this neighbourhood is equivalent, via homeographic projection, to values of $|\vec{w}|^2$ greater than, for instance, $3a$. Thus, it has its maximum in the north pole. Then, the dynamical system generated by the differential equation

$$\dot{\vec{w}} = -\operatorname{grad} V(\vec{w}),$$

considered on the abovementioned neighbourhood, has a repelling, hyperbolic fixed point in the north pole. The completed potential $\tilde{E} \in C(S^2, \mathbb{R})$ can be of the form:

$$\tilde{E}(\vec{w}) = \begin{cases} E^*(\vec{w}) & \text{for } |\vec{w}|^2 \leq 2a, \\ V(\vec{w}) & \text{for } |\vec{w}|^2 \geq 3a. \end{cases}$$

Since trajectories enter the interior of the circle $K((0,0), r = 2a)$, the closed ball $B((0,0), r \leq 2a)$ is an invariant set of the system. Therefore, though the potential is modifed, the dynamics in the domain of interest for us remains unchanged. Thus, if only a dynamical system generated by Equation (13) has finite number of singularities in the ball $B((0,0), r \leq a)$, all of them hyperbolic, then the dynamical system generated by Equation (18) satisfies the assumptions of Theorem 3.4.1.

Possible applications of Theorem 3.4.1 are limited to systems with finite numbers of singularities, all hyperbolic. It is important in these applications that the set of structurally stable systems is open and dense in the set of gradient dynamical systems (see [42], page 116) and that structural stability implies the assumptions of Theorem 3.4.1. This ensures that the properties specified in assumptions of Theorem 3.4.1 are generic (see remarks in Section 1).

The dynamics of gradient systems is very regular. Particularly, this dynamics can not be chaotic and there are no periodic orbits. These properties are preserved under discretization and, thanks to the global topological conjugacy, when the Euler method is applied. This implies asymptotical stability of the learning proccess of the layer artificial neural networks which are modelled by the cascade generated by the Euler method.

4. Concluding remarks

The presented mathematical analysis allows us to create a model describing trained multilayer ANNs and the dynamics of their learning process. The introduced mathematical model has the following structure: first, an untrained and a trained neuron are defined as real functions of two or one vector variable, respectively; then, using a description based on graph theory, a multilayer ANN is said to be a map $\mathscr{S} : \mathbb{R}^k \longrightarrow \mathbb{R}^m$. This function is used in the definition of a cost function assuming that a training sequence is given. Subsequently, since a training process simply aims at finding a cost function minimum, we can analyse the induced numerical process using dynamical systems theory.

It has been shown that by considering a linear network we can, without losing generality, study dynamics of only a single neuron with an activation function equal to identity. Asymptotical stability of a training process of such a network is a generic property.

Theorems describing properties of weakly perturbated linear systems, especially the Grobman-Hartman Theorem and the Fečkan Theorem, imply the properties of the so-called weakly nonlinear neuron, as introduced by Bielecki. Topological conjugacy of its learning process dynamics to the dynamics of a linear system has been proven. Estimations (12) of the constants in the Fečkan Theorem will allow us to implement this sort of a neuron.

The theorem about topological conjugacy of gradient cascades on a two-dimensional sphere allows us to study the dynamics of a learning process of a nonlinear neuron having a two-component input. Generalization of this theorem to higher dimensions will enable us to consider more complicated cases of nonlinear ANNs.

References

[1] Hertz J, Krogh A and Palmer R G 1991 *Introduction to the Theory of Neural Computation*, Addison-Welsey Publishing Company, Massachusetts

[2] Korbicz J, Obuchowicz A and Uciński D 1994 *Artificial Neural Networks – Backgrounds and Applications*, Akademicka Oficyna Wydawnicza PLJ, Warsaw (in Polish)

[3] Osowski S 1996 *Neural Networks – the Algorithmic Approach*, WNT, Warsaw (in Polish)

[4] Rutkowska D, Piliński M and Rutkowski L 1997 *Neural Networks, Genetic Algorithms and Fuzzy Systems*, PWN, Warsaw-Lodz (in Polish)

[5] Żurada J, Barski M and Jędruch W 1996 *Artificial Neural Networks*, PWN, Warsaw (in Polish)

[6] Tadeusiewicz R 1993 *Neural Networks*, Akademicka Oficyna Wydawnicza, Warsaw (in Polish)

[7] Podolak I T 1998 *IEEE Trans. on Systems, Men and Cybern., Part B: Cybernetics* **28** (6) 876

[8] Owens A J and Filkin D L 1989 *Int. Joint Conf. on Neural Networks* **II**, Washington, New York: IEEE, pp. 381–386

[9] Kloeden P E and Lorenz J 1986 *SIAM J. Numer. Anal.* **23** 986

[10] Alouges F and Debussche A 1991 *Numer. Funct. Anal. Optimiz.* **12** 253

[11] Beyn W J 1987 *SIAM J. Numer. Anal.* **24** 103

[12] Beyn W J and Lorenz J 1987 *Numer. Funct. Anal. Optimiz.* **9** 381

[13] Fečkan M 1991 *Proc. Amer. Math. Soc.* **111** 585

[14] Fečkan M 1991 *Proc. Amer. Math. Soc.* **113** 1105

[15] Mrozek M and Rybakowski K P 1992 *J. Dyn. Diff. Eq.* **4** 57

[16] Srzednicki R 1994 *J. Diff. Eq.* **111** 283

[17] Garay B 1996 *J. Diff. Eq. App.* **2** 67

[18] Fečkan M 1992 *Math. Slovaca* **42** (1) 123

[19] Garay B 1993 *Acta Math. Univ. Comenianae* **62** 245

[20] Bielecki A 2000 *Ann. Pol. Mat.* **73** 37

[21] Garay B 1994 *Acta Math. Univ. Comenianae* **63** 25

[22] Garay B 1996 *J. Numer. Math.* **4** 449

[23] Garay B 1998 *IMA J. Numer. Anal.* **18** 77

[24] Dudek-Dyduch E 1986 *Cybernetics of Neuro-like Systems*, Zakład Nar. im. Ossolińskich, PAN Publ., Wroclaw (in Polish)

[25] Bielecki A 2001 *Opuscula Mathematica* **20** 249

[26] Bielecki A 2001 *Nonlinear Analysis: Real World Applications* **2** 249

[27] Campbell S A 1998 *Differential Equations with Applications to Biology* (Ruan S, Wolkowicz G S K and Wu J, Eds.) **21** 65

[28] Deville Y 1996 *Signal Processing* **51** (3) 229

[29] Gelenbe E 1990 *Neural Computation* **2** (2) 239
[30] Hoppensteadt F C and Izhikevich E M 1997 *Weakly Connected Neural Networks*, Springer
[31] Tanaka K 1996 *IEEE Trans. on Neural Networks* **7** (3) 629
[32] Solla S 1995 *Phys. Rev. Lett.* **74** 4337
[33] Solla S and Levin E 1986 *Complex Systems* **2** 625
[34] Saad D and Solla S 1996 *Advances in Neural Information Processing Systems* **8** 302
[35] Duch W and Jankowski N 1999 *Neural Computing Surveys* **2** 163
[36] Bielecki A 1998 *Gradient Dynamical Systems and a Training Process of Multilayer Neural Networks*, PhD Thesis, Faculty of Mathematics and Physics, Jagiellonian University, Cracow (in Polish)
[37] Podolak I T 1998 *Comput. Phys. Commun.* **117** (1–2) 62
[38] Baldi P and Hornik K 1995 *IEEE Trans. on Neural Networks* **6** (4) 837
[39] Demidowicz B 1972 *Mathematical Stability Theory*, WNT, Warsaw (in Polish)
[40] Wiggins S 1990 *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, Springer-Verlag, New York
[41] Voyevodin V V 1980 *Linear Algebra*, Mir Publishers, Moscow
[42] Palis J and de Melo W 1982 *Geometric Theory of Dynamical Systems*, Springer Verlag, New York
[43] Irwin M C 1987 *Smooth Dynamical Systems*, Academic Press Ltd., London
[44] Müller B and Reinhardt J 1990 *Neural Networks*, Springer Verlag, New York
[45] Tadeusiewicz R 1994 *Problems of Biocybernetics*, PWN, Warsaw (in Polish)
[46] Jabłoński D and Bielecki A 2000 *Proc. 6^{th} National Conf. on Application of Mathematics in Biology and Medicine*, Zawoja, Poland, pp. 57–60
[47] Mañé R 1988 *Publ. Mathematiques IHES* **66** 161