Pol. J. Appl. Sci., **2017**, *3*, 146-154

OPTIMIZING 2-LAYER PERCEPTRON NEURONS NUMBER AND PIXEL-TO-SHIFT STANDARD DEVIATIONS RATIO FOR TRAINING ON PIXEL-DISTORTED SHIFTED 60×80 IMAGES IN CLASSIFYING SHIFTING-DISTORTED OBJECTS

Vadim Romanuke

Faculty of Navigation and Naval Weapons
Polish Naval Academy, Gdynia, Poland

E-mail: romanukevadimv@gmail.com

Abstract: The problem of classifying shifting-distorted objects is considered. The object model is the flat monochrome 60×80 image of an enlarged capital letter from the English alphabet. A model of shifting-distorted monochrome images with pixel distortion is developed. The classifier is 2-layer perceptron, whose faster operation speed and poor performance on shifted objects stands against deep neural networks classifying shifted objects well enough with significant delays. The perceptron performance is the maximal classification error percentage which depends on two parameters. These ones are the perceptron single hidden layer size and pixel-to-shift standard deviations ratio. The ratio is assumed to advance the perceptron in training on pixel-distorted shifted images in order to classify shifting-distorted objects better. Thus, the problem of minimizing a function of two variables is stated, wherein the function is the maximal classification error percentage. Statistically, the optimal hidden layer neurons number is an integer from 390 to 400, and the optimal ratio should be set to a value from the segment [0.01; 0.02]. For the accepted object model, these parameters made it possible to obtain a 2-layer perceptron classifier whose performance is comparable to that one of deep learners. This is about 11.2% error rate and lower. Compared to the results obtained previously, the gain of this two-parameter optimization is at least about 6%.

Key words: shifting-distorted objects, 2-layer perceptron, perceptron single hidden layer size, pixel-to-shift standard deviations ratio, maximal classification error percentage, optimization problem.

Shifting-distorted objects and their classification

In computer-controlling objects nothing can be centered to compare the input object with the known one. Object decentralization is usually called shift. N-dimensional object shift by $N \in \mathbb{N}$ is described with N shift indicators, each indicator in its dimension. One-dimensional objects are shifted along an axis line (linear or curvilinear), and they are easily captured and classified. Three-dimensional objects, the highest-dimensional objects that have been visualized yet, are more difficult to be classified as their shift is of three indicators, and it is straightened sometimes to calculate them from computer vision data [1–3]. Two-dimensional objects, being often visualized in monochrome images, are relatively easy to expand their shift into horizontal and vertical shift indicators by projections. These flat objects mostly are ordinary for their classification [2, 4–6] whereas N-dimensional objects by $N \in \mathbb{N}\backslash\{1,2\}$ are projected in two-dimensional space.

For machine classifiers, the shifting of objects is equivalent to their distortion. If a flat object is presented by $L \times C$ matrix $\mathbf{B} = [b_{ik}]_{L\times C}$ then matrix $\tilde{\mathbf{B}} = [\tilde{b}_{ik}]_{L\times C}$ of the shifting-distorted object has a property that there is either

$$\sum_{k=1}^{C} \left(b_{i_0 k} - \tilde{b}_{i_0 k}\right) \neq 0 \qquad (1)$$

or

$$\sum_{i=1}^{L} \left(b_{i k_0} - \tilde{b}_{i k_0}\right) \neq 0 \qquad (2)$$

for at least one $i_0 \in \{\overline{1, L}\}$ and one $k_0 \in \{\overline{1, C}\}$. Both inequalities (1) and (2) are violated if the object by matrix $\mathbf{B} = [b_{ik}]_{L\times C}$ is background, and all elements of this matrix are identical. For example, this is white color in monochrome images when the object elements are featured with black-and-white on the white background; if the object is shifted, its horizontal and vertical stripes, which were removed, become the background color; for the object, being the background, shifting doesn't change it. Thus, the classification of shifting-distorted objects cannot be fulfilled through object-by-object comparison, where monochrome images are compared on pixel-by-pixel; even if the image is shifted off the single pixel line (or column), most of other pixels obtained new values, and matrices of the non-shifted image and shifting-distorted image are quite different although visually those two images are very similar.

Classifiers for shifted objects

Linear classifiers are not apt to recognize shifting distortions [1, 6–9]. Neural networks with nonlinear activation functions fit the classification of shifting-distorted objects after they have been trained on samples from general totality [6,8–10]. Neocognitrons and other hierarchical multilayered neural networks

(deep neural networks) classify shifted objects well enough, but with a bad delay, being scored in seconds and even minutes on object medium and larger formats [11–13]. Perceptrons classify shifting-distorted objects much worse. However, if 2-layer perceptron could be trained on shifted objects, it would excel other classifiers in operational speed. The matter just is in that 2-layer perceptron cannot be trained even on shifting-distorted flat objects [7, 8, 10, 14] not speaking about N-dimensional objects by $N \in \mathbb{N}\backslash\{1,2\}$. Nevertheless, perceptrons are trained very effectively on feature-distorted objects, especially when feature distortion is distributed normally [6,8,15]. Then, importing some part of feature distortion into shifting-distorted objects, there might be fixed a ratio between both types of distortions, at which 2-layer perceptron would be trained successfully. Moreover, the training process might be shortened by adjusting the single hidden layer size in this 2-layer perceptron.

The paper's goal and tasks to be accomplished

The aim is to formalize those two parameters of the classifier identification on the basis of 2-layer perceptron. Firstly, sizes of 2-layer perceptron are in the tuple $\langle I,H,F\rangle$ by the input layer size $I=LC$, the hidden layer size $H\in[H_{min};H_{max}]\cap\mathbb{N}$ and the output layer F being equal to the number of classes. Untrained 2-layer perceptron $P_0(\langle I,H,F\rangle)$ is merely for classifying objects, having I features. Secondly, the said ratio is [6]

$$r=\frac{\partial_{max}}{\gamma_{max}} \tag{3}$$

by maximal standard deviation ∂_{max} in forming feature-distorted objects and maximal standard deviation γ_{max} in forming shifting-distorted objects. In the training process the input of $P_0(\langle I,H,F\rangle)$ is fed with the training set

$$\left\{\{\overline{\mathbf{B}}\}_{d=1}^{R},\{\tilde{\overline{\mathbf{Z}}}_s\}_{s=1}^{S}\right\} \tag{4}$$

included $R\in\mathbb{N}$ replicas $\overline{\mathbf{B}}=[b_{wc}]_{LC\times F}$ of F non-distorted (pure) representatives $\left\{\mathbf{B}_c=[b_{uv}^{\langle c\rangle}]_{L\times C}\right\}_{c=1}^{F}$ and $S\in\mathbb{N}$ shares of $LC\times F$ matrices $\{\tilde{\overline{\mathbf{Z}}}_s\}_{s=1}^{S}$ of distorted objects by $R+S$ targets as identity $F\times F$ matrices. Thus, the trained perceptron $P_0(\langle I,H,F\rangle)$ is registered as

$$P\Big(\langle I,H,F\rangle,\langle\gamma_{max},r\rangle,\langle R,S,Q_{pass}\rangle\Big) \tag{5}$$

by the integer Q_{pass}, showing how many times the training set (4) has been passed through the perceptron. In registration (5) γ_{max} is fixed before getting started in training, and the ratio (3) is determined with ∂_{max} that varies from the lowest value $\partial_{max}^{\langle min\rangle}$ up to the highest $\partial_{max}^{\langle max\rangle}$. Varying ∂_{max} generates a segment

$$[r_{min};r_{max}]=\left[\frac{\partial_{max}^{\langle min\rangle}}{\gamma_{max}};\frac{\partial_{max}^{\langle max\rangle}}{\gamma_{max}}\right] \tag{6}$$

of tolerable ratios (3). Having put heuristically three integers $\langle R,S,Q_{pass}\rangle$ there is the goal to optimize 2-layer perceptron (5) by its two parameters H and r. The criterion of optimization is minimizing classification error percentage $p(H,r)$ in classifying shifting-distorted objects of a definite type:

$$[H*\quad r*]\in\arg\Big(\min_{[H\quad r]\in\{[H_{min};H_{max}]\cap\mathbb{N}\}\times[r_{min};r_{max}]}\{p(H,r)\}\Big). \tag{7}$$

To solve the problem (7) explicitly, the following tasks must be accomplished to achieve the paper's goal:

1. To define the object model, general totality and non-distorted representatives of the fixed number of classes F.
2. To select a program environment, where 2-layer perceptron $P_0(\langle I,H,F\rangle)$ will be trained and perceptrons (5) will be simulated for covering the product
$$\{[H_{min};H_{max}]\cap\mathbb{N}\}\times[r_{min};r_{max}] \tag{8}$$
over which the surface $p(H,r)$ should be minimized in (7).
3. To state a model of shifting-distorted objects of the chosen type blended with feature-distorted objects of that type.
4. To define boundaries H_{min} and H_{max} for the range of hidden layer neurons number H.
5. To define boundaries r_{min} and r_{max} for the ratio (3) range.
6. To run perceptrons (5) through the rectangle (8) of H and ratio (3) values for evaluating the surface $p(H,r)$ as an averaged classification error percentage.
7. To minimize the surface $p(H,r)$ over the rectangle (8), reaching the optimal values of H and ratio (3).
8. To verify the problem's (7) solution.
9. To reason whether the perceptron
$$P\Big(\langle I,H*,F\rangle,\langle\gamma_{max},r*\rangle,\langle R,S,Q_{pass}\rangle\Big) \tag{9}$$
performance might be optimized further.

Object model, general totality and pure representatives of the classes

A one object model must be accepted. There is no need to compose a big benchmark gallery of similar object models because a perceptron is indifferent to feature representation (which, in this way, can be varied). The object may be flat, and its model may be a monochrome 60×80 image of an enlarged English alphabet capital letter [6]. Its medium format is good for obtaining results promptly. The general totality

$$G=\left\{\{\mathbf{B}_c\}_{c=1}^{26},\{\tilde{\mathbf{B}}_m\}_{m=1}^{2^{4800}-26}\right\} \tag{10}$$

is of 26 pure representatives $\{\mathbf{B}_c=[b_{uv}^{\langle c\rangle}]_{60\times80}\}_{c=1}^{26}$ and the rest $2^{4800}-26$ monochrome 60×80 images as 60×80 matrices $\{\tilde{\mathbf{B}}_m\}_{m=1}^{2^{4800}-26}$. Each of 2^{4800} elements in (10) is the matrix of ones and zeros. While being tested, the trained classifier (5) or (9) input is fed with samples from the general totality (10). In training, the perceptron $P_0(\langle I,H,F\rangle)$ is fed with samples from the extended general totality

$$E=G\cup Z \tag{11}$$

by the set Z of 60×80 matrices

$$\tilde{\mathbf{Z}} = \mathbf{G} + \partial \cdot \Xi \tag{12}$$

at a standard deviation ∂ in forming pixel-distorted monochrome 60×80 images and 60×80 matrix Ξ of values of normal variate with zero expectation and unit variance, where $\mathbf{G} \in G$. Obviously, the extended general totality (11) is infinite.

MATLAB function for training the perceptron

The oncoming investigations are connected with vector and matrix algebra, whose buildups are within the program environment MATLAB, in numerical and symbolical views. MATLAB Neural Network Toolbox is one of the best for simulating neural networks or constructing classifiers. To train 2-layer perceptron $P_0(\langle 4800, H, 26\rangle)$ there is MATLAB function "traingda" [16, 17]. It is one of the fastest methods of backpropagation algorithm [15, 18–20] for training multilayer perceptrons. While "traingda" works, the perceptron weight and bias values are updated according to the gradient descent with adaptive learning rate [20–22]. All the trainings are going to be driven under the training MATLAB function "traingda".

Model of shifting-distorted monochrome images with pixel distortion

The c-th class pure representative $\mathbf{B}_c = [b_{uv}^{\langle c\rangle}]_{60\times 80}$ from (10) is shifted for the s-th share in the training set (4) as follows. In general, for $L \times C$ monochrome image horizontal shift is

$$x(\gamma_s) = \varphi(0.1C\gamma_s \cdot \varsigma_s) \cdot \frac{1 - \operatorname{sign}(|\varphi(0.1C\gamma_s \cdot \varsigma_s)| - C)}{2} + \\ + C \cdot \frac{1 + \operatorname{sign}(|\varphi(0.1C\gamma_s \cdot \varsigma_s)| - C)}{2} \tag{13}$$

pixels and vertical shift is

$$y(\gamma_s) = \varphi(0.1L\gamma_s \cdot \zeta_s) \cdot \frac{1 - \operatorname{sign}(|\varphi(0.1L\gamma_s \cdot \zeta_s)| - L)}{2} + \\ + L \cdot \frac{1 + \operatorname{sign}(|\varphi(0.1L\gamma_s \cdot \zeta_s)| - L)}{2} \tag{14}$$

pixels, where standard deviation

$$\gamma_s = \frac{\gamma_{max}}{S} \cdot s \qquad \text{for} \qquad s = \overline{1, S} \tag{15}$$

and function $\varphi(\alpha)$ rounds α to the nearest integer less than or equal to α, where ς_s and ζ_s are values of normal variate with zero expectation and unit variance, raffled for the s-th share independently.

The horizontal shift goes first, where matrix $\mathbf{B}_c = [b_{uv}^{\langle c\rangle}]_{L\times C}$ changes into matrix $\mathbf{A}_c(s) = [a_{uv}^{\langle c\rangle}(s)]_{L\times C}$. For $x(\gamma_s) > 0$ the elements of this intermediate matrix are

$$a_{uv}^{\langle c\rangle}(s) = 1 \quad \text{for} \quad v = \overline{1, x(\gamma_s)} \quad \text{and} \quad a_{uv}^{\langle c\rangle}(s) = b_{ut}^{\langle c\rangle} \\ \text{at} \quad t = v - x(\gamma_s) \quad \text{for} \quad v = \overline{x(\gamma_s) + 1, C} \;\forall\; u = \overline{1, L}. \tag{16}$$

For $x(\gamma_s) < 0$ those elements are

$$a_{uv}^{\langle c\rangle}(s) = b_{ut}^{\langle c\rangle} \quad \text{at} \quad t = v - x(\gamma_s) \quad \text{for} \quad v = \overline{1, C + x(\gamma_s)} \\ \text{and} \quad a_{uv}^{\langle c\rangle}(s) = 1 \quad \text{for} \quad v = \overline{C + x(\gamma_s) + 1, C} \;\forall\; u = \overline{1, L}. \tag{17}$$

For $x(\gamma_s) = 0$ the c-th image is not shifted horizontally:

$$a_{uv}^{\langle c\rangle}(s) = b_{uv}^{\langle c\rangle} \;\forall\; u = \overline{1, L} \quad \text{and} \quad \forall\; v = \overline{1, C}. \tag{18}$$

The vertical shift goes second, right after the matrix $\mathbf{A}_c(s) = [a_{uv}^{\langle c\rangle}(s)]_{L\times C}$ has been formed. Here matrix $\mathbf{A}_c(s) = [a_{uv}^{\langle c\rangle}(s)]_{L\times C}$ changes into matrix $\mathbf{Z}_c(s) = [z_{uv}^{\langle c\rangle}(s)]_{L\times C}$. For $y(\gamma_s) > 0$ the elements of this shift final matrix are

$$z_{uv}^{\langle c\rangle}(s) = a_{rv}^{\langle c\rangle}(s) \quad \text{at} \quad r = u + y(\gamma_s) \quad \text{for} \quad u = \overline{1, L - y(\gamma_s)} \\ \text{and} \quad z_{uv}^{\langle c\rangle}(s) = 1 \quad \text{for} \quad u = \overline{L - y(\gamma_s) + 1, L} \;\forall\; v = \overline{1, C}. \tag{19}$$

For $y(\gamma_s) < 0$ those elements are

$$z_{uv}^{\langle c\rangle}(s) = 1 \quad \text{for} \quad u = \overline{1, -y(\gamma_s)} \quad \text{and} \quad z_{uv}^{\langle c\rangle}(s) = a_{rv}^{\langle c\rangle}(s) \\ \text{at} \quad r = u + y(\gamma_s) \quad \text{for} \quad u = \overline{-y(\gamma_s) + 1, L} \;\forall\; v = \overline{1, C}. \tag{20}$$

For $y(\gamma_s) = 0$ the c-th horizontally shifted image is not shifted vertically:

$$z_{uv}^{\langle c\rangle}(s) = a_{uv}^{\langle c\rangle}(s) \;\forall\; u = \overline{1, L} \quad \text{and} \quad \forall\; v = \overline{1, C}. \tag{21}$$

After having shifted with (13) – (21) by $L = 60$ and $C = 80$ all the pure representatives $\left\{\mathbf{B}_c = [b_{uv}^{\langle c\rangle}]_{60\times 80}\right\}_{c=1}^{26}$ separately, the c-th shifted image as matrix $\mathbf{Z}_c(s) = [z_{uv}^{\langle c\rangle}(s)]_{60\times 80}$ is reshaped into 4800×1 matrix (column), and all these 26 columns are concatenated horizontally into 4800×26 matrix $\overline{\mathbf{Z}}_s$ for $s = \overline{1, S}$. Paralleling, pure representatives $\left\{\mathbf{B}_c = [b_{uv}^{\langle c\rangle}]_{60\times 80}\right\}_{c=1}^{26}$, reshaped into 4800×1 column each, are concatenated horizontally into 4800×26 matrix $\overline{\mathbf{B}}$. Then the training set (4) has $26 \cdot S$ reshaped elements

$$\tilde{\overline{\mathbf{Z}}}_s = \overline{\mathbf{Z}}_s + \partial_s \cdot \mathbf{\Omega}_s \quad \text{for} \quad s = \overline{1, S} \tag{22}$$

from the general totality (11), where standard deviation

$$\partial_s = \frac{\partial_{max}}{S} \cdot s \quad \text{for} \quad s = \overline{1, S} \tag{23}$$

is multiplied by 4800×26 matrix $\mathbf{\Omega}_s$ of values of normal variate with zero expectation and unit variance. Afterwards the training set (4), included R pure and S shifting-distorted monochrome images with pixel distortion (22) for each class, feeds (passing through) the perceptron $P_0(\langle 4800, H, 26\rangle)$ for Q_{pass} times.

Boundaries for the range of hidden layer neurons number

Commonly, the perceptron for a classification problem is trained for three stages: training on pure representatives, training on noised representatives, and training on pure representatives again to make sure the trained perceptron didn't lose

the ability to recognize pure representatives. In the first stage, the perceptron $P_0(\langle 4800, H, 26\rangle)$ is trained on the single replica $\overline{\mathbf{B}} = [b_{wc}]_{4800\times 26}$. In the second stage, it is trained on the training set (4). Finally, in the third stage, the perceptron (5)

$$P(\langle 4800, H, 26\rangle, \langle \gamma_{max}, r\rangle, \langle R, S, Q_{pass}\rangle) \tag{24}$$

is re-trained on the single replica $\overline{\mathbf{B}} = [b_{wc}]_{4800\times 26}$. The integer range $[H_{min}; H_{max}] \cap \mathbb{N}$ of hidden layer neurons number H should be defined with boundaries H_{min} and H_{max} on the training quality criterion. By $H < H_{min}$ the perceptron $P_0(\langle 4800, H, 26\rangle)$ or (24) is trained slower than ordinarily or cannot be trained at all, starting from the first training stage. By $H > H_{max}$ there is a risk of getting the overtrained perceptron (24) in the second training stage, or the perceptron $P_0(\langle 4800, H, 26\rangle)$ training process may hang in the first training stage. Empirically, for the problem of classifying shifting-distorted monochrome images by their shift intensity in (13) – (21), there are most likely boundaries $H_{min} = 200$ and $H_{max} = 350$ independently of the pixel-to-shift standard deviations ratio (3) with an appropriate maximal standard deviation γ_{max}.

Boundaries for the range of pixel-to-shift standard deviations ratio

Here, regarding (13) – (21), an appropriate maximal standard deviation for shift distortion is $\gamma_{max} = 1$. Let the tuple $\langle R, S, Q_{pass}\rangle$ empirically be $\langle 2, 8, 240\rangle$. The lowest value $\partial_{max}^{\langle min\rangle} = 0.01$ because in the case $\partial_{max} < 0.01$ the second training stage of the perceptron $P_0(\langle 4800, H, 26\rangle)$ flows very slow. The case $\partial_{max} > 1$ makes the second training stage be completed quicker, but the perceptron (24)

$$P(\langle 4800, H, 26\rangle, \langle 1, r\rangle, \langle 2, 8, 240\rangle) \tag{25}$$

poor performance becomes unacceptable. So, boundaries $r_{min} = \dfrac{\partial_{max}^{\langle min\rangle}}{\gamma_{max}} = 0.01$ and $r_{max} = \dfrac{\partial_{max}^{\langle max\rangle}}{\gamma_{max}} = 1$ enclose the segment $[0.01; 1]$ of pixel-to-shift standard deviations ratio.

Running through the rectangle (8)

For solving the problem (7)

$$[H* \quad r*] \in \arg\left(\min_{[H \quad r]\in\{[200;350]\cap\mathbb{N}\}\times[0.01;1]}\{p(H,r)\}\right) \tag{26}$$

the perceptron (25) shall be run through batch testing on the rectangle (8), which makes it possible to evaluate the surface $p(H, r)$ as either an averaged classification error percentage on (8) or a maximal classification error percentage on (8). The maximal classification error percentage is presumed to ensue from maximal shift distortion. While being tested, the input of the perceptron (25) is fed with shifting-distorted monochrome 60×80 images, formed by shift standard deviation $\gamma \in [0; \gamma_{max}] = [0; 1]$. They are fed 400 batches from the general totality (10), where each batch has 26 elements, by one representative of every class. The classification error percentage by shift standard deviation $\gamma \in [0; 1]$ is $p(H, r, \gamma)$. This is calculated as

$$p(H, r, \gamma) = \frac{q(H, r, \gamma)}{400 \cdot 26} \cdot 100 = \frac{q(H, r, \gamma)}{104}$$

by the number $q(H, r, \gamma)$ of classification errors, scored at parameters $\{H, r, \gamma\}$. The averaged classification error percentage

$$p(H, r) = \int_0^1 p(H, r, \gamma)d\gamma \tag{27}$$

can be numerically evaluated on the subset $\{0.1j\}_{j=0}^{10} \subset [0; 1]$ as

$$p(H, r) \approx \frac{1}{11}\sum_{j=0}^{10} p(H, r, 0.1j) \tag{28}$$

for $H \in [200; 350] \cap \mathbb{N}$ and $r \in [0.01; 1]$. The maximal classification error percentage is just

$$p(H, r) = p(H, r, \gamma_{max}) = p(H, r, 1). \tag{29}$$

It is clear that the segment $[0.01; 1]$ must be sampled. Let the sampling steps be 0.01 and 0.1, substituting that segment with the 19-elemented subset

$$\left\{\{0.01 + 0.01i\}_{i=1}^{9}, \{0.1 + 0.1i\}_{i=1}^{9}\right\} \subset [r_{min}; r_{max}] = [0.01; 1]. \tag{30}$$

Besides, the hidden layer size can be run with the step equal to 10 neurons. Hence, instead of the rectangular (8)

$$\{[200; 350] \cap \mathbb{N}\} \times [0.01; 1] \tag{31}$$

being actually the striped rectangular owing to one integer side (hidden layer neurons number), there is a lattice, constituted with set $\{200 + 10i\}_{i=0}^{15}$ and subset in (30):

$$\left\{\{200 + 10i\}_{i=0}^{15}\right\} \times \left\{\{0.01 + 0.01i\}_{i=0}^{9}, \{0.1 + 0.1i\}_{i=0}^{9}\right\} \subset \subset \{[200; 350] \cap \mathbb{N}\} \times [0.01; 1]. \tag{32}$$

Fig. 1 shows the average of four evaluations of the surface $p(H, r)$ on lattice (32). Each evaluation has been made up of 304 points of lattice (32), where every point is the averaged classification error percentage of perceptron (25), for

$$H \in \{200 + 10i\}_{i=0}^{15}$$

and

$$r \in \left\{\{0.01 + 0.01i\}_{i=0}^{9}, \{0.1 + 0.1i\}_{i=0}^{9}\right\}.$$

Note, that for plotting those four meshes 1216 perceptrons have been tested.

Naturally, the global minimum of the surface $p(H, r)$ can be found only numerically. Fig. 1 can't help with it, unless to watch a domain within the rectangle (31), and this domain shall have the minimum point. Therefore, the domain shall be re-sampled to find the minimum point by the higher accuracy.

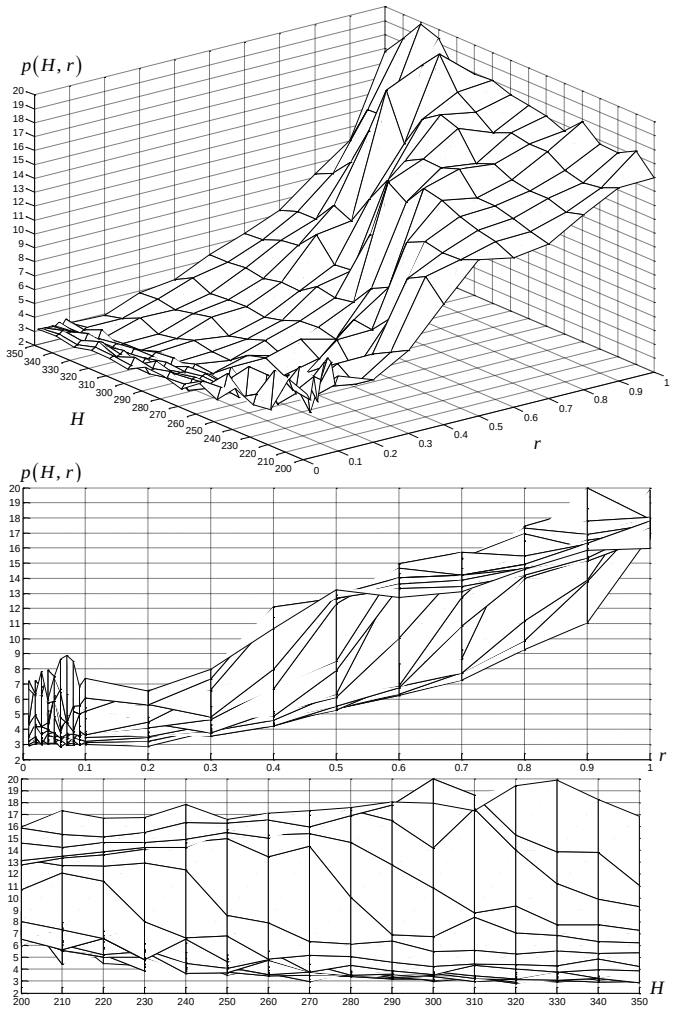

Fig. 1: The average of four evaluations of the surface $p(H,r)$ on 304-point lattice (32) and its single variable profiles.

Maximal classification error percentage minimization by the optimal point (26)

At first glance, Figure 1 hints at that the surface $p(H,r)$ minimum is reached at about a point, enclosed within the domain

$$\{[270;350]\cap\mathbb{N}\}\times[0.01;0.2]\subset\{[200;350]\cap\mathbb{N}\}\times[0.01;1]. \quad (33)$$

Without re-sampling the subsegments, domain (33) is substituted with the finer lattice

$$\left\{\{270+10i\}_{i=0}^{8}\right\}\times\left\{\{0.01+0.01i\}_{i=0}^{9},0.2\right\}\subset \subset\{[270;350]\cap\mathbb{N}\}\times[0.01;0.2]. \quad (34)$$

As the averaged classification error percentage has decreased significantly, the maximal classification error percentage is better to use. Fig. 2 shows the average of 20 re-evaluations of the surface $p(H,r)$ by (29) on 99-point lattice (34), where every point is the maximal classification error percentage of perceptron (25), for

$$H\in\{270+10i\}_{i=0}^{8}$$

and

$$r\in\left\{\{0.01+0.01i\}_{i=0}^{9},0.2\right\}.$$

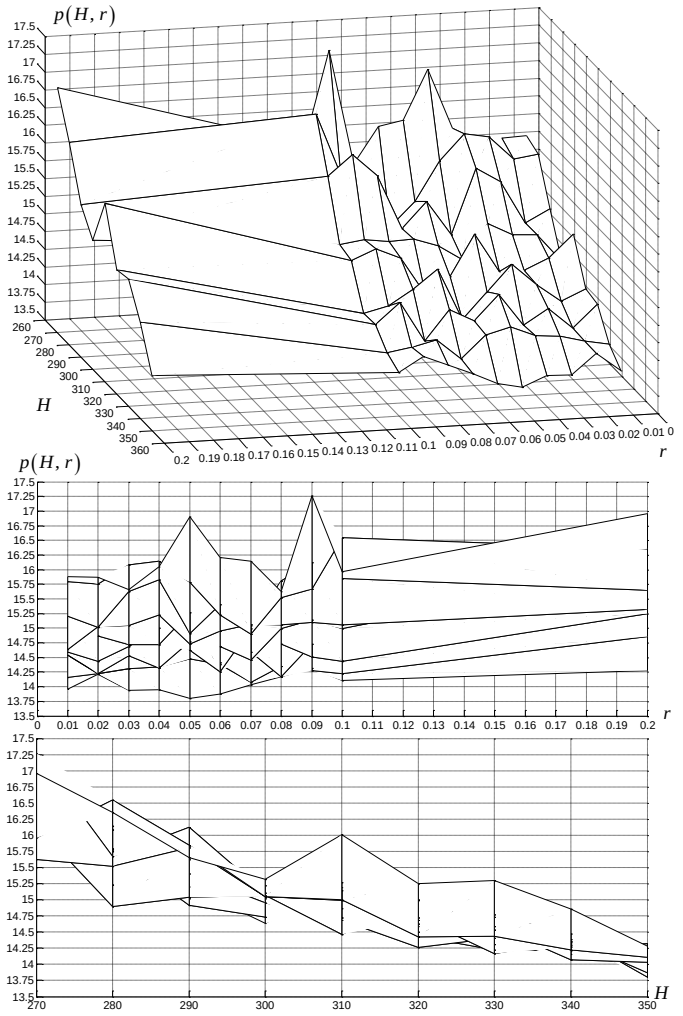

Fig. 2: The average of 20 evaluations of surface (29) on 99-point lattice (34) and its single variable profiles.

For plotting those 20 meshes 1980 perceptrons have been tested, including those 396 ones from Fig. 1.

It is seen clearly from Fig. 2 that

$$r*\in\{0.01i\}_{i=1}^{7} \quad (35)$$

but $H*>340$. This is why the third bunch of evaluations is going to be made on the lattice

$$\left\{\{350+10i\}_{i=0}^{4}\right\}\times\left\{\{0.01i\}_{i=1}^{7}\right\}\subset\{[350;390]\cap\mathbb{N}\}\times[0.01;0.7] \quad (36)$$

and number of evaluations is doubled. Fig. 3 shows the average of 40 evaluations of surface (29) on 35-point lattice (36). For plotting those 40 meshes there have been tested 1400 perceptrons. Here, only 140 perceptrons from Fig. 2 have been included which concerned the point $H=350$.

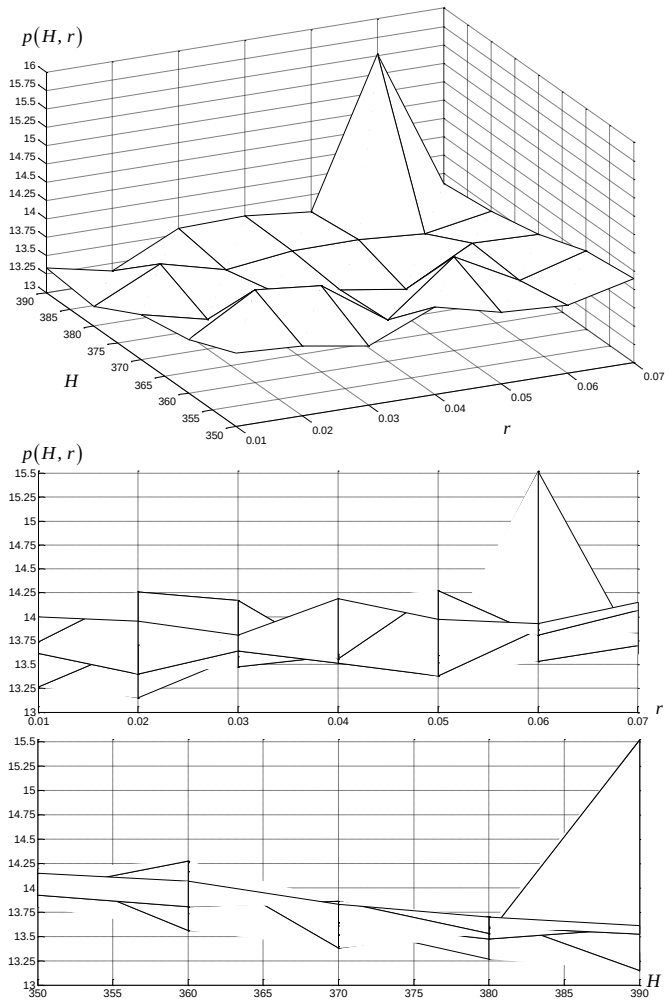

Fig. 3: The average of 40 evaluations of surface (29) on 35-point lattice (36) and its single variable profiles.

The H axis profile of the mesh in Fig. 3 shows that the case of $H* > 390$ cannot be excluded. This implies the H axis should be extended to the right. A decision on the ratio (3) is closer – only three points

$$\{0.01, 0.02, 0.03\}$$

remain to re-evaluate. Hence, the fourth bunch of evaluations is on the lattice

$$\left\{\{370+10i\}_{i=0}^{4}\right\}\times\left\{\{0.01i\}_{i=1}^{3}\right\}\subset\{[370;410]\cap\mathbb{N}\}\times[0.01;0.03] \tag{37}$$

with the number of evaluations doubled more. Fig. 4, showing the average of 80 evaluations of surface (29) on 15-point lattice (37), makes an evaluation of global minimum confusing. Although for plotting those 80 meshes 1200 perceptrons have been tested, including 360 perceptrons from Fig. 3, the high variance of classification error percentage has not decreased. Nonetheless, the final decision on what the solution (26) is can be made using statistics of those 80 meshes and other ones in Figures 1, 2, and 3. For instance, a number of cases where

$$p(H,r) < p_0 \tag{38}$$

and

$$p(H,r) > p_1 \tag{39}$$

can be counted, where p_0 and p_1 are desired (tolerable) CEP and undesired (intolerable) CEP, respectively.

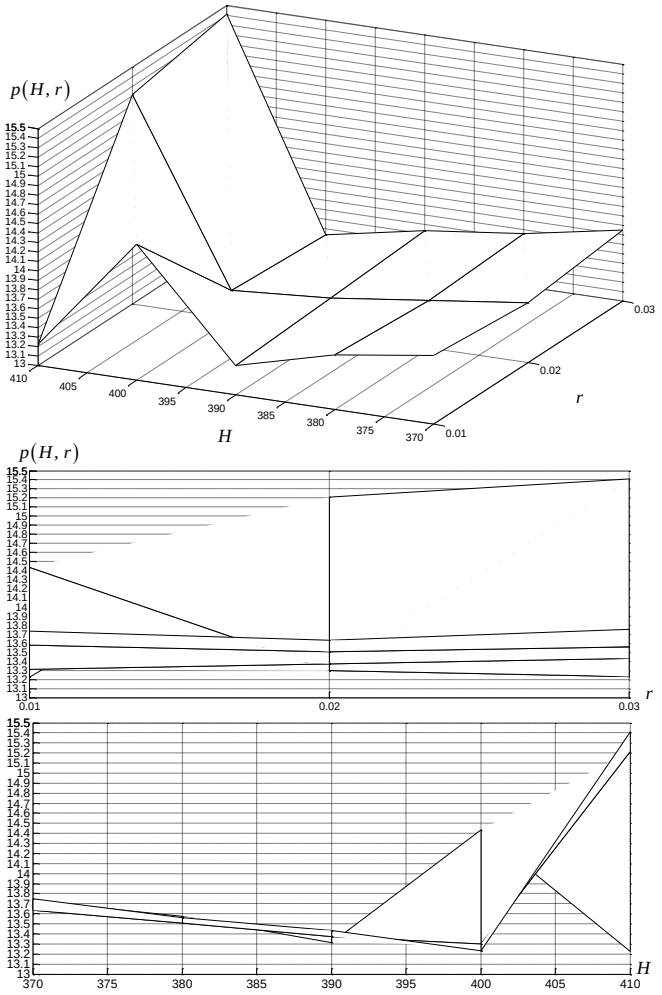

Fig. 4: The average of 80 evaluations of surface (29) on 15-point lattice (37) and its single variable profiles.

Denote by $c_H(p_0)$ the most frequent hidden layer neurons number for (38), and denote by $c_r(p_0)$ the most frequent ratio for (38). If $p_0 \in [3; 3.5]$ for the averaged classification error percentage, solely $c_H(p_0) = 350$ and $c_r(p_0) = 0.01$ (though, except a case with $c_r(3.23) = 0.02$), although $c_H(2.52) = 310$ and $c_r(2.52) = 0.05$ (related to Fig. 1). Denote by $u_H(p_1)$ the less frequent hidden layer neurons number for (39), and denote by $u_r(p_1)$ the less frequent ratio for (39), the results are much the same.

If $p \in [11.5; 13.5]$ for the maximal classification error percentage, solely $c_H(p_0) = 350$ and $c_r(p_0) \neq 0.01$ (related to Fig. 2), where $c_r(p_0) = 0.02$ in over 77% of all cases. Also $u_H(p_1) = 350$ in over 82% of all cases, but $u_r(p_1) = 0.07$ in nearly every third

case. Related to Fig. 3, solely $c_H(p_0) = 390$ and $c_r(p_0) = 0.01$ in nearly two from three cases, but $u_H(p_1) = 370$ at 71% rate and $u_r(p_1) = 0.02$ at 69% rate. Finally, relating to Fig. 4, $c_r(p_0) = 0.02$ at 70% rate and $u_r(p_1) = 0.02$ at 69% rate, but $c_H(p_0) = 410$ stands at 69% rate against $u_H(p_1) = 400$ at 47% rate.

Now it is clearer that

$$r* \in \{0.01, 0.02\}. \tag{40}$$

It might have seemed that the most appropriate hidden layer neurons number is 410, but there are four cases among those 80 ones with $H = 410$ when the training process just fully failed, where the maximal classification error percentage is equal to $\frac{2500}{26}$. The same fail concerned a single perceptron with $H = 400$. Therefore,

$$H* \in \{[390; 400] \cap \mathbb{N}\}. \tag{41}$$

Each of memberships (40) and (41) is a statistical decision. However, an evaluation of the global minimum of the surface $p(H, r)$ on lattice (37) could be the point

$$[H* \quad r*] = [400 \quad 0.02] \tag{42}$$

by a supplementary criterion. This criterion is the training process duration, which is statistically the shortest for point (42). Perceptron

$$P(\langle 4800, 400, 26\rangle, \langle 1, 0.02\rangle, \langle 2, 8, 240\rangle) \tag{43}$$

at point (42) performs at

$$p(400, 0.02) \approx 13.2994$$

on average. The best perceptron (43) has maximal classification error percentage

$$p(400, 0.02) \approx 11.0192.$$

Perceptrons

$$P(\langle 4800, 410, 26\rangle, \langle 1, 0.03\rangle, \langle 2, 8, 240\rangle) \tag{44}$$

and

$$P(\langle 4800, 390, 26\rangle, \langle 1, 0.01\rangle, \langle 2, 8, 240\rangle) \tag{45}$$

have the best performance

$$p(410, 0.03) = p(390, 0.01) = 10.625,$$

where just one of them solves the problem (7) if the memberships (40) and (41) are regarded as the solution. Perceptron

$$P(\langle 4800, 390, 26\rangle, \langle 1, 0.02\rangle, \langle 2, 8, 240\rangle) \tag{46}$$

performs at 11% error rate. Along with three perceptrons (43), (45), (46), being the problem (7) solutions, and perceptron (44) with 10.625% error rate, there are another three having very low error rates:

$$p(390, 0.03) \approx 11.0481,$$

$$p(410, 0.02) \approx 11.0385,$$

$$p(410, 0.03) \approx 10.9038,$$

where the last statement relates to another perceptron (44).

Verification of the problem (7) solution

According to the memberships (40) and (41), the problem (7) solution is stated as

$$[H* \quad r*] \in \{[390; 400] \cap \mathbb{N}\} \times \{[0.01; 0.02]\}. \tag{47}$$

For verification of the problem (7) solution (47), the best perceptrons (43), (45), (46) shall be re-tested at shift standard deviation $\gamma \in [0; 1]$ in shifting-distorted monochrome 60×80 images. The number of re-testing batches is 25 times increased (10,000 batches). Taken three testings by 10,000 batches each, the averaged polylines of functions $p(400, 0.02, \gamma), p(390, 0.01, \gamma), p(390, 0.02, \gamma)$ by $\gamma \in \{0.1j\}_{j=0}^{10}$ are shown in Fig. 5 against the background of those four ones reflecting performance of non-optimal perceptrons although close to optimum. Disclosures of those 10,000-batched tests are in Fig. 6.

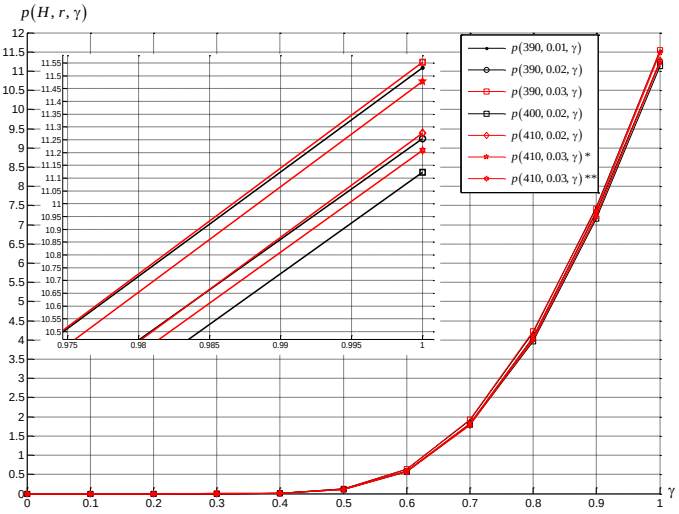

Fig. 5: Polylines of functions $p(400, 0.02, \gamma), p(390, 0.01, \gamma), p(390, 0.02, \gamma)$ for verifying the problem (7) solution (47); polylines of functions $p(390, 0.03, \gamma)$ and $p(410, 0.02, \gamma)$, and two polylines of function $p(410, 0.03, \gamma)$ representing two different perceptrons (44) are in the background for comparison, where (*) corresponds to the perceptron tested previously with its result $p(410, 0.03) \approx 10.9038$, and (**) corresponds to the perceptron tested previously with its result $p(410, 0.03) \approx 10.625$; each polyline is the average of three 10,000-batched tests.

Perceptron (43) reveals itself in both Fig. 5 and Fig. 6 that it has the best performance, which is

$$p(400, 0.02) \approx 11.1228$$

now, averaged over three 10,000-batched tests. Every single 10,000-batched testing gives the same result, i.e. the optimality of parameters (42) is confirmed. Note, that the performances of the rest classifiers are a little worse than expected after 400-batched testing (this happened because averaging over the 400-batched testings is very rough). That magnificent 10.625% error rate is not reached. However, the gain of this two-parameter optimization is at least about 6% comparing to results obtained in [3, 6, 8, 13].

It is worth pointing out that a few tests showed that an 11% error rate can be transcended at $H > 400$. Moreover, two

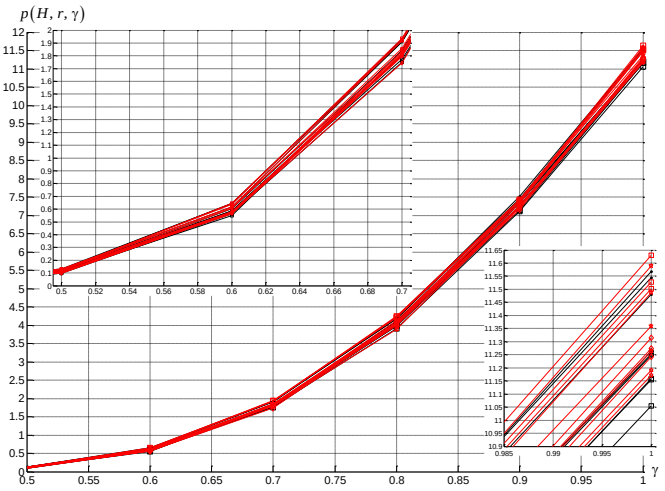

Fig. 6: Disclosures of those 10,000-batched tests whose averaged polylines are in Fig. 5 (designations of line markers are the same).

of those three perceptrons by $H = 410$ in Fig. 5 have a performance that does not exceed 11.3% error rate. Nevertheless, if the hidden layer neurons number is increased to starting off at point $H > 400$, the likelihood of the training process full fail increases. Besides, operation speed of 2-layer perceptron becomes lower as its single hidden layer size is increased. Here "operation speed" is treated in the sense of classifying multiple huge streams of objects (not a single object), where every superfluous byte weighs as a gigabyte. This is why $H = 400$. So, taking into consideration the results in Fig. 5 and the disclosures in Fig. 6 along with the restricted single hidden layer size, the problem (7) solution (47) has been verified and validated.

Reasoning into further 2-layer perceptron performance optimization

When another problem of classifying shifting-distorted objects is put forward, say, in another format of images or with non-imaged objects, then the hidden layer neurons number and the ratio (3) are optimized for 2-layer perceptron in a similar way to those first-numbered eight items, formulated as tasks for this paper's investigation. Furthermore, a 2-layer perceptron can be optimized by those items and for N-dimensional objects by $N \in \mathbb{N}\backslash\{1,2\}$ without projecting them flat. This is because the classifiers on perceptrons use the line-up (column) of object features, and whatever the object with a finite feature number is, its N-dimensional matrix $\mathbf{B} = [b_j]_{\mathscr{F}}$ of the format $\mathscr{F} = \times_{d=1}^{N} L_d$ and subscript J is reshaped into $\left(\prod_{d=1}^{N} L_d\right) \times 1$ column. Consequently, any color images shift problem may also be solved on 2-layer perceptron classifiers. For objects, whose number of features is close to (or comparable to) 4800 and number of classes is about 26, the solution (47) remains relevant. In addition, the corresponding best perceptron performance will be nearly the same as the perceptron (43) performance. If an even number of features is about a few thousands, the optimal single hidden layer size should be set to an integer from 390 to 400, and the optimal ratio (3) should be set to a value from the segment $[0.01; 0.02]$. The ratio is far less sensitive to changes in number of features and number of classes.

The perceptron (41) performance could have been optimized deeper if the integers in the tuple $\langle R, S, Q_{pass}\rangle$ hadn't been put empirically. To optimize them along with H and the ratio (3), however, wouldn't have been rational as that would have given a minimization problem of the hypersurface of five variables, what is always too hard, especially when this hypersurface must be evaluated (on five-dimensional line-pointed parallelepiped!) before [23, 24]. Instead of that the perceptron (9)

$$P(\langle 4800, 400, 26\rangle, \langle 1, 0.02\rangle, \langle R, S, Q_{pass}\rangle)$$

performance might be optimized further, where the hypersurface of three variables in the tuple $\langle R, S, Q_{pass}\rangle$) should be minimized on integer parallelepiped. After such optimization, all points in (47) may come off the optimum, but the perceptron performance is not presumed to change much to make the investigator re-optimize the perceptron neurons number and the ratio (3).

Literature

[1] Oreški G., Oreški S. Two stage comparison of classifier performances for highly imbalanced datasets. *Journal of Information and Organizational Sciences*, 39(2):209–222, 2015.

[2] Klette R. *Concise Computer Vision: An Introduction into Theory and Algorithms.* Springer, London, 2014.

[3] Romanuke V.V. Boosting ensembles of heavy two-layer perceptrons for increasing classification accuracy in recognizing shifted-turned-scaled flat images with binary features. *Journal of Information and Organizational Sciences*, 39(1):75–84, 2015.

[4] Forsyth D.A., Ponce J. *Computer Vision. A Modern Approach, 2nd edition.* Pearson, Upper Saddle River, 2012.

[5] Zhou H., Li X., Schaefer G., Celebi M.E., Miller P. Mean shift based gradient vector flow for image segmentation. *Computer Vision and Image Understanding*, 117(9):1004–1016, 2013.

[6] Romanuke V.V. An attempt for 2-layer perceptron high performance in classifying shifted monochrome 60-by-80-images via training with pixel-distorted shifted images on the pattern of 26 alphabet letters. *Radioelectronics, informatics, control*, (2):112–118, 2013.

[7] Fukushima K., Miyake S. A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455–469, 1982.

[8] Romanuke V.V. Two-layer perceptron for classifying flat scaled-turned-shifted objects by additional feature distortions in training. *Journal of Uncertain Systems*, 9(4):286–305, 2015.

[9] Costarelli D., Spigler R. Convergence of a family of neural network operators of the Kantorovich type. *Journal of Approximation Theory*, (185):80–90, 2014.

[10] Chen Z., Cao F., Hu J. Approximation by network operators with logistic activation functions. *Applied Mathematics and Computation*, (256):565–571, 2015.

[11] Schmidhuber J. Deep learning in neural networks: An overview. *Neural Networks*, (61):85–117, 2015.
[12] Weng J., Ahuja N., Huang T. S. Learning recognition and segmentation using the cresceptron. *International Journal of Computer Vision*, 25(2):109–143, 1997.
[13] Fukushima K. Artificial vision by multi-layered neural networks: Neocognitron and its advances. *Neural Networks*, (37):103–119, 2013.
[14] Arulampalam G., Bouzerdoum A. A generalized feedforward neural network architecture for classification and regression. *Neural Networks*, 16(5-6):561–568, 2003.
[15] Hagiwara K., Hayasaka T., Toda N., Usui S., Kuno K. Upper bound of the expected training error of neural network regression for a Gaussian noise sequence. *Neural Networks*, 14(10):1419–1429, 2001.
[16] Romanuke V.V. Setting the hidden layer neuron number in feedforward neural network for an image recognition problem under Gaussian noise of distortion. *Computer and Information Science*, 6(2):38–54, 2013.
[17] Romanuke V.V. Accuracy improvement in wear state discontinuous tracking model regarding statistical data inaccuracies and shifts with boosting mini-ensemble of two-layer perceptrons. *Problems of tribology*, (4):55–58, 2014.
[18] Moller M.F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.
[19] Kathirvalavakumar T., Jeyaseeli Subavathi S. Neighborhood based modified backpropagation algorithm using adaptive learning parameters for training feedforward neural networks. *Neurocomputing*, 72(16-18):3915–3921, 2009.
[20] Hagan M.T., Menhaj M.B. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, 1994.
[21] Nied A., Seleme S.I.Jr., Parma G.G., Menezes B.R. On-line neural training algorithm with sliding mode control and adaptive learning rate. *Neurocomputing*, 70(16-18):2687–2691, 2007.
[22] Yoo S.J., Park J.B., Choi Y.H. Indirect adaptive control of nonlinear dynamic systems using self recurrent wavelet neural networks via adaptive learning rates. *Information Sciences*, 177(15):3074–3098, 2007.
[23] Malalur S.S., Manry M.T., Jesudhas P. Multiple optimal learning factors for the multi-layer perceptron. *Neurocomputing*, (149 (p. C)):1490–1501, 2015.
[24] Trobec R., Vajteršic M., Zinterhof P. *Parallel Computing. Numerics, Applications, and Trends.* Springer, London, 2009.

Received: 2017
Accepted: 2017