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 presumed 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 straightforward 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

\[
\sum_{k=1}^{C} (b_{i0k} - \tilde{b}_{i0k}) \neq 0 \quad (1)
\]

or

\[
\sum_{i=1}^{L} (b_{i0k} - \tilde{b}_{i0k}) \neq 0 \quad (2)
\]

for at least one \( i_0 \in \{1, L\} \) and one \( k_0 \in \{1, C\} \). 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, while being tested, the trained classifier (5) or perceptron (5) over which the surface \( p(H,r) \) should be minimized in (7).

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 \( P_b((I,H,F)) \) will be trained and perceptrons (5) will be simulated for covering the product

\[
\{(H_{\text{min}};H_{\text{max}}) \cap \mathbb{N}) \times [r_{\text{min}};r_{\text{max}}]\}
\]

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 \( H_{\text{min}} \) and \( H_{\text{max}} \) for the range of hidden layer neurons number \( H \).
5. To define boundaries \( r_{\text{min}} \) and \( r_{\text{max}} \) for the ratio (3) range.
6. To run perceptrons (5) through the rectangle (8) of \( (H, r) \) 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

\[
P\left((I,H,F), (\gamma_{\text{max}},r), (R,S,Q_{\text{pass}})\right)
\]

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 \( 60 \times 80 \) image of an enlarged English alphabet capital letter [6]. Its medium format is good for obtaining results promptly. The general totality

\[
G = \left\{ \{B_c\}_{c=1}^{26}, \{\bar{B}_m\}_{m=1}^{24800-26} \right\}
\]

is of 26 pure representatives \( \{B_c = [b_{uv}]_{60 \times 80}\}_{c=1}^{26} \) and the rest \( 2^{24800} - 26 \) monochrome \( 60 \times 80 \) images as \( 60 \times 80 \) matrices \( \{\bar{B}_m\}_{m=1}^{24800-26} \). Each of \( 2^{24800} \) 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 \( P_b((I,H,F)) \) is fed with samples from the extended general totality

\[
E = G \cup Z
\]
by the set $Z$ of $60 \times 80$ matrices
\[ \tilde{Z} = G + \partial \cdot \Xi \] 
(12)
at a standard deviation $\theta$ in forming pixel-distorted monochrome
$60 \times 80$ images and $60 \times 80$ matrix $\Xi$ of values of normal variate
with zero expectation and unit variance, where $G \in G$. 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 builds-up are within the program envi-
ronment MATLAB, in numerical and symbolical views. MA-
TLAB Neural Network Toolbox is one of the best for simula-
tion environment MATLAB, in numerical and symbolical views. MA-
TLAB function for training the perceptron $P_0((4800, H, 26))$ there is MATLAB function “traingda” [16, 17]. It is one of the fastest methods of backpropa-
gation 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 $B_c = [b_{uv}^{(c)}]_{60 \times 80}$ 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
\[ x(\gamma_s) = \varphi(0.1C \gamma_s \cdot \zeta_s) \cdot \frac{1 - \text{sign}(\varphi(0.1C \gamma_s \cdot \zeta_s) - C)}{2} + C \cdot \frac{1 + \text{sign}(\varphi(0.1C \gamma_s \cdot \zeta_s) - C)}{2} \] 
(13)
pixels and vertical shift is
\[ y(\gamma_s) = \varphi(0.1L \gamma_s \cdot \zeta_s) \cdot \frac{1 - \text{sign}(\varphi(0.1L \gamma_s \cdot \zeta_s) - L)}{2} + L \cdot \frac{1 + \text{sign}(\varphi(0.1L \gamma_s \cdot \zeta_s) - L)}{2} \] 
(14)
pixels, where standard deviation
\[ \gamma_s = \frac{\gamma_{\text{max}}}{S} \cdot s \quad \text{for} \quad s = \overline{1,S} \] 
(15)
and function $\varphi(\alpha)$ rounds $\alpha$ to the nearest integer less than or
equal to $\alpha$, where $\zeta_s$ and $\zeta_s$ 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 $B_c = [b_{uv}^{(c)}]_{L \times C}$
changes into matrix $A_c(s) = [a_{uv}^{(c)}(s)]_{L \times C}$. For $x(\gamma_s) > 0$
the elements of this intermediate matrix are
\[ a_{uv}^{(c)}(s) = 1 \quad \text{for} \quad v = \overline{1,C} \quad \text{and} \quad a_{uv}^{(c)}(s) = b_{uv}^{(c)} \] 
(16)
at $t = v - x(\gamma_s)$ for $v = x(\gamma_s) + 1, C \forall u = \overline{1,L}$.

For $x(\gamma_s) < 0$ those elements are
\[ a_{uv}^{(c)}(s) = b_{uv}^{(c)} \quad \text{at} \quad t = v - x(\gamma_s) \quad \text{for} \quad v = \overline{1,C} + x(\gamma_s) \]
(17)
and $a_{uv}^{(c)}(s) = 1$ for $v = C + x(\gamma_s) + 1, C \forall u = \overline{1,L}$.

For $x(\gamma_s) = 0$ the $c$-th image is not shifted horizontally:
\[ a_{uv}^{(c)}(s) = b_{uv}^{(c)} \quad \forall u = \overline{1,L} \quad \text{and} \quad \forall v = \overline{1,C}. \] 
(18)
The vertical shift goes second, right after the matrix $A_c(s) = [a_{uv}^{(c)}(s)]_{L \times C}$ has been formed. Here matrix $A_c(s) = [a_{uv}^{(c)}(s)]_{L \times C}$
changes into matrix $Z_c(s) = [z_{uv}^{(c)}(s)]_{L \times C}$. For $y(\gamma_s) > 0$
the elements of this shift final matrix are
\[ z_{uv}^{(c)}(s) = a_{uv}^{(c)}(s) \quad \text{at} \quad r = u + y(\gamma_s) \quad \forall u = \overline{1,L - y(\gamma_s)} \]
and $z_{uv}^{(c)}(s) = 1$ for $u = L - y(\gamma_s) + 1, L \forall v = \overline{1,C}$.

For $y(\gamma_s) < 0$ those elements are
\[ z_{uv}^{(c)}(s) = 1 \quad \text{for} \quad u = \overline{1,L - y(\gamma_s)} \quad \text{and} \quad z_{uv}^{(c)}(s) = a_{uv}^{(c)}(s) \] 
(20)
at $r = u + y(\gamma_s)$ for $u = y(\gamma_s) + 1, L \forall v = \overline{1,C}$.

For $y(\gamma_s) = 0$ the $c$-th horizontally shifted image is not shifted
vertically:
\[ z_{uv}^{(c)}(s) = a_{uv}^{(c)}(s) \quad \forall u = \overline{1,L} \quad \text{and} \quad \forall v = \overline{1,C}. \] 
(21)

After having shifted with (13) – (21) by $L = 60$ and $C = 80$
all the pure representatives $\{B_c = [b_{uv}^{(c)}]_{60 \times 80}\}_{c=1}^{26}$
separately, the $c$-th shifted image as matrix $Z_c(s) = [z_{uv}^{(c)}(s)]_{60 \times 80}$ is reshap-
ed into $4800 \times 1$ matrix (column), and all these 26 columns are
concatenated horizontally into $4800 \times 26$ matrix $Z_\tau$ for $s = \overline{1,S}$.
Paralleling, pure representatives $\{B_c = [b_{uv}^{(c)}]_{60 \times 80}\}_{c=1}^{26}$, reshap-
ed into $4800 \times 1$ column each, are concatenated horizontally
into $4800 \times 26$ matrix $B$. Then the training set (4) has $26 \cdot S$ reshap-
ed elements
\[ \overline{Z}_\tau = Z_\tau + \partial_s \cdot \Omega_s \quad \text{for} \quad s = \overline{1,S} \] 
(22)
from the general totality (11), where standard deviation
\[ \partial_s = \frac{\partial_{\text{max}}}{S} \cdot s \quad \text{for} \quad s = \overline{1,S} \] 
(23)
is multiplied by $4800 \times 26$ matrix $\Omega_s$ 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 $P_0((4800, H, 26))$ for $Q_{\text{pass}}$ 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, train-
ing on noised representatives, and training on pure representa-
tives again to make sure the trained perceptron didn’t lose
the ability to recognize pure representatives. In the first stage, the perceptron \( P_0((4800, H, 26)) \) is trained on the single replica \( \mathbf{B} = [w_{wc}]_{4800 \times 26} \). In the second stage, it is trained on the training set (4). Finally, in the third stage, the perceptron (5)

\[
P((4800, H, 26), (\gamma_{\text{max}}, r), (R, S, Q_{\text{pass}}))
\]

is re-trained on the single replica \( \mathbf{B} = [w_{wc}]_{4800 \times 26} \). The integer range \( [H_{\text{min}}; H_{\text{max}}] \cap \mathbb{N} \) of hidden layer neurons number \( H \) should be defined with boundaries \( H_{\text{min}} \) and \( H_{\text{max}} \) on the training quality criterion. By \( H < H_{\text{min}} \) the perceptron \( P_0((4800, H, 26)) \) or (24) is trained slower than ordinarily or cannot be trained at all, starting from the first training stage. By \( H > H_{\text{max}} \) there is a risk of getting the overtrained perceptron (24) in the second training stage, or the perceptron \( P_0((4800, H, 26)) \) 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 \( H_{\text{min}} = 200 \) and \( H_{\text{max}} = 350 \) independently of the pixel-to-shift standard deviations ratio (3) with an appropriate maximal standard deviation \( \gamma_{\text{max}} \).

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

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

\[
P((4800, H, 26), (1, r), (2, 8, 240))
\]

poor performance becomes unacceptable. So, boundaries

\[
\begin{align*}
\gamma_{\text{min}} = \frac{\partial_{\text{min}}^{(\gamma_{\text{min}})}}{\gamma_{\text{max}}} = 0.01 \\
\gamma_{\text{max}} = \frac{\partial_{\text{max}}^{(\gamma_{\text{max}})}}{\gamma_{\text{max}}} = 1
\end{align*}
\]

enclose the segment \([0.01; 1]\) of pixel-to-shift standard deviations ratio.

**Running through the rectangle (8)**

For solving the problem (7)

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

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 \( \gamma \in [0; \gamma_{\text{max}}] = [0; 1] \). 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

\[
p(H, r) = \frac{1}{\partial_{\text{min}}^{(\gamma_{\text{min}})}} \int_{0}^{1} p(H, r, \gamma) d\gamma
\]

can be numerically evaluated on the subset \( \{\gamma \}_{\gamma=0}^{\gamma=1} \subset [0; 1] \) as

\[
p(H, r) \approx \frac{1}{\partial_{\text{min}}^{(\gamma_{\text{min}})}} \sum_{j=0}^{10} p(H, r, 0.1j)
\]

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

\[
p(H, r) = p(H, r, \gamma_{\text{max}}) = p(H, r, 1).
\]

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

\[
\begin{align*}
&\{0.01 + 0.01i\}_{i=0}^{9}, \{0.1 + 0.1i\}_{i=1}^{9} \\
&\subset [r_{\text{min}}; r_{\text{max}}] = [0.01; 1]
\end{align*}
\]

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

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

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):

\[
\begin{align*}
&\{200 + 10i\}_{i=0}^{15} \times \{0.01 + 0.01i\}_{i=0}^{9}, \{0.1 + 0.1i\}_{i=1}^{9} \\
&\subset \{[200; 350] \cap \mathbb{N}\} \times [0.01; 1].
\end{align*}
\]

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

\[
r \in \{0.01 + 0.01i\}_{i=0}^{9}, \{0.1 + 0.1i\}_{i=1}^{9}
\]

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.
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

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

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

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

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

$$r \in \{(0.01 + 0.01i)_{i=0}^{9} \times 0.2\}.$$
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 remain to re-evaluate. Hence, the fourth bunch of evaluations is on the lattice

\[
\{0.01, 0.02, 0.03\}
\]

with the number of evaluations doubled more. Fig. 4, showing the average of 80 evaluations of surface (29) on 15-point lattice (36), 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

\[
p(H, r) < p_0 \quad (38)
\]

and

\[
p(H, r) > p_1 \quad (39)
\]

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

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
\[ r* \in \{0.01, 0.02\}. \] (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,
\[ H* \in [(390; 400) \cap \mathbb{N}]. \] (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
\[ [H \star r*] = [400, 0.02] \] (42)
by a supplementary criterion. This criterion is the training process duration, which is statistically the shortest for point (42).

Perceptron
\[ P((4800, 400, 26), (1, 0.02), (2, 8, 240)) \] (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
\[ P((4800, 410, 26), (1, 0.03), (2, 8, 240)) \] (44)
and
\[ P((4800, 390, 26), (1, 0.01), (2, 8, 240)) \] (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
\[ P((4800, 390, 26), (1, 0.02), (2, 8, 240)) \] (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).

Verifying the problem (7) solution

According to the memberships (40) and (41), the problem (7) solution is stated as
\[ [H \star r*] \in [(390; 400) \cap \mathbb{N}] \times \{(0.01; 0.02)\}. \] (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
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 $N \in \mathbb{N}\setminus\{1,2\}$ 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 $\mathbf{B} = [b_j]_x$ of the format $\mathcal{F} = \times_{d=1}^N L_d$ and subscript $J$ is reshaped into $\left(\prod_{d=1}^N L_d\right) \times 1$ column. Consequently, any color image 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)

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

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


Received: 2017
Accepted: 2017