International Journal of Pure and Applied Mathematics
Volume 115 No. 4 2017, 883-893
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
doi: 10.12732/ijpam.v115i4.22

ijpam.eu

SEPARATE TRAINING OF HYBRID NEURAL NETWORK

Grigoriy Belyavskiy[1], Evgeny Puchkov[2 §]
[1]South Federal University
8a, Milchakova Street, Rostov-on-Don, 344058, RUSSIAN FEDERATION
[2]Don State Technical University
162, Socialisticheskaja Street
Rostov-on-Don, 344022, RUSSIAN FEDERATION

Abstract: Our research is related to the hybrid neural network training. It consists of local elements, each of which solving its own problem, and recurrent control element producing the general solution. Local elements have identical perceptron inputs and outputs. It is assumed that the training sample is clustered and each element is trained beforehand and independently. The training of local elements and the control element is considered in detail. The methods of regret assessment for the control element are shown. A computational experiment of the hybrid neural network training is carried out through a recognition problem. As a result of the training, the control element selected the local element with the least loss.

AMS Subject Classification: 68T05, 68T42, 62M20, 62M45
Key Words: online learning, hybrid neural network, coordination of expert assessments, regret, perceptron

1. Introduction

The training methods applied in the work are widely used for coordination of expert assessments [1,2] in sequential forecasting problems. The attention of researchers is primarily focused on the development of algorithms that give an es-

Received: June 20, 2017
Revised: July 5, 2017
Published: August 10, 2017

© 2017 Academic Publications, Ltd.
url: www.acadpubl.eu

[§]Correspondence author

timate close to the best expert. Some successful results of such approaches were presented in the works of Cesa-Bianchi et al.[3], Hazan and Kale [4], Chiang et al. [5], where sequence prediction based on expert predictions was demonstrated and a convex online optimization problems were solved. Alexander Rakhlin Karthik Sridharan are actively working in the field of online learning [6, 7]. Their works not only present theoretical proof of boundary estimation and choice of the control element model, but also describe practical applications of adaptive online learning methods [8].

Our research is devoted to the development of the method for separate training of a hybrid neural network. Each neuron or a separate group of neurons (a local element) of such a network solves its own task and the control element produces the general solution. To achieve successful results, it is assumed that the training sample is clustered. The mutual influence of elements on each other in this training is absent. This usually has a good effect on the training outcomes. Each local element is trained beforehand and independently on its data set. In this work, the training of local and control elements was considered in detail. The results of computational experiments through the example of the recognition problem were presented.

2. Methodology of Separate Training

The control element is trained online. The task of control element consists of the coordination of expert assessments. In our case, the experts are local elements. Training is seen as a game with the environment (Fig. 1).

2.1. Training of a Local Element

We will train local elements on a clustered sample, that is

$$V = \bigcup_{i=1}^{M} V_i,\ V_i = V_i^1 \bigcup V_i^2, V_i \bigcap V_j = \emptyset.$$

In our study, each cluster contains noisy images of the first or second class, distinguished by a small shift. Each local element is trained independently of others on its cluster.

We assume that the convex hulls V_i^1 and V_i^2 are linearly separable. In this case, the operation of the neuron can be represented as a predicate: $\lceil (W_i, X) \geq \theta_i \rceil$, X - the input binary vector, W_i- the vector of weights, θ_i - the threshold. The

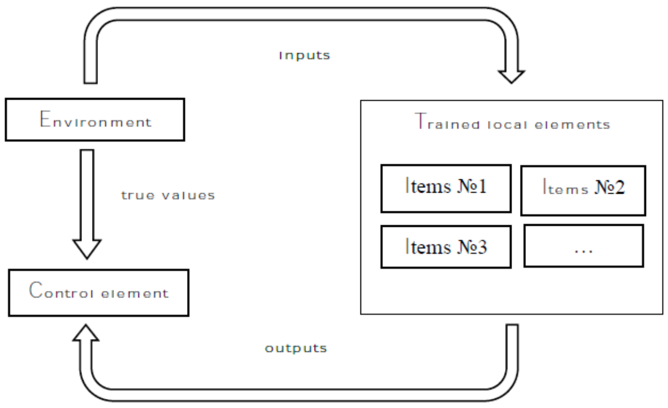

Figure 1: The scheme of the learning control item

problem of the local element training is to calculate the vector of weights and the threshold. The most common in the case of linear separation is Kozinets algorithm [9]. The learning algorithm calculates the vector with minimal norm in the convex hull of the set $B_i = \left\{ b : b = v^1_{i,j} - v^2_{i,k} \right\}$, $v^1_{i,j}$ and $v^2_{i,j}$- the elements of the sets V^1_i and V^2_i accordingly. The vector of weights is equal to the minimum vector W^*_i, the threshold is

$$\theta^*_i = \frac{\min\limits_j \left(W^*_i, v^1_{i,j} \right) + \max\limits_k \left(W^*_i, v^2_{i,k} \right)}{2}.$$

Only two elements of the weight vector change on each iteration of the Kozinets algorithm, so you can expect that the method of projecting the conjugate gradient with a constant step will converge faster. The problem of computing the minimal norm in a vector in a convex hull of a set $B = \{b_1, b_2, ..., b_r\}$ is equivalent to the problem:

$$W^* = \arg\min \left(G^T G x, x \right), \text{ under constraints: } x_i \geq 0, \sum_i x_i = 1. \quad (1)$$

In (1) the columns of the matrix Gcoincide with the elements of the set$B$. The following calculations at each iteration of the learning algorithm are made:

$$u_{t+1} = x_t - \eta g\left(u_t\right), \quad x_{t+1} = \max\left(u_{t+1} - \lambda, 0\right). \quad (2)$$

(2) g - the conjugate gradient; the maximum is calculated from the coordinates; λ - solution of the equation:$\sum\limits_i \max\left(\left(u_{t+1}\right)_i - \lambda, 0\right) = 1$.

2.2. Training of a Control Element

We call Y - the space of the environment choices, Z - the space of outputs of local and control elements. The loss function of the hybrid neural network when interacting with the environment on the Cartesian product is defined as $l : Z \times Y \to R^{+}$. It has standard properties of the loss function, for example, it is convex with respect to the first argument and bounded.

For any t and for each i cumulated losses can be calculated:

$$L_{i,t} = \sum_{t=1}^{n} l\left(z_{i,t}, y_t\right),$$

$y_1, ..., y_n$- choices of environment, $z_{i,1}, ..., z_{i,n}$ - Outputs of the i-the local element. The accumulated losses of a hybrid network over the same period are: $R_n = \sum\limits_{t=1}^{n} l\left(z_t, y_t\right)$, $z_1, ..., z_n$ - outputs of the control element. Regret is the difference $D_n = R_n - \min\limits_i L_i$. The interaction of hybrid neural networks with the environment is as follows: at a (concentrated moment) local time t outputs of local elements $z_{i,t+1}$ and the output of the control element z_{t+1} are calculated. After that, the environment makes a choice y_{t+1}. The control element in the calculation z_{t+1} uses the outputs of local elements and the information about the quality of the functioning of local elements. The algorithm of control element functioning is considered successful if $\lim\limits_{n\to\infty} \frac{D_n}{n} = 0$. Successful algorithms include weighed estimation and random selection.

The weighed estimation is calculated as the average of the local elements outputs:

$$z_{t+1} = \sum_i p_{i,t} z_{i,t+1}. \tag{3}$$

It is assumed here that the set Z is a convex set. Exponential weighing is considered good:

$$p_{i,t} = \frac{\exp\left(-hL_{i,t}\right)}{\sum\limits_j \exp\left(-hL_{j,t}\right)}. \tag{4}$$

The output of the control element with random selection is

$$z_{t+1} = z_{\xi_t, t+1}. \tag{5}$$

In (5) ξ_t- independent random variables, $\xi_t \in \{1, ..., M\}$ and $P\left(\xi_t = i\right) = p_{i,t}$.

The calculation of the probabilities that are used in both the first and second methods can be organized in a form more suitable for the control element. The corresponding recurrence equations have the following form:

$$u_{i,t} = u_{i,t-1} \exp\left(-hl\left(z_{i,t}, y_t\right)\right), \quad p_{i,t} = \frac{u_{i,t}}{\sum\limits_j u_{j,t}}, \quad u_{i,0} = 1. \tag{6}$$

2.3. Regret Assessment

A very important point is the choice of values for the parameter h. [2] proposes different values for the parameter. If $h = \sqrt{8 \ln M/T}$, it is assumed that $n \le T$, $l\left(z, y\right) \in [0, 1]$; M- the number of local elements, then regret assessment is:

$$D_n \le \sqrt{\frac{T}{2} \ln M}. \tag{7}$$

The advantage of the assessment (7) as well as of all subsequent ones is their independence from the choice of environment. The disadvantage of this choice of parameter is the dependence on T. It is necessary to choose a large value for T, because in real problems the number of cycles $n$is unknown beforehand. This leads to an unjustifiably overestimated regret assessment. It is suggested to restart the algorithm in points $T_j = 2^j$ to get rid of this drawback. On the interval $\left\{2^j, ..., 2^{j+1} - 1\right\}$, parameter $h_j = \sqrt{\frac{8 \ln M}{2^j}}$. This choice of parameter leads to the following regret assessment:

$$D_n \le \frac{1}{\sqrt{2} - 1} \sqrt{\frac{n}{2} \ln M}. \tag{8}$$

This assessment doesn't have disadvantage of (7) because it does not depend on T. However, the restart of the algorithm is not always justified, because it leads to the loss of the information about the behavior of local elements. Very often the restart occurs at the beginning of training, and then less frequently.

Provided that $\exp\left(-hl\left(z, y\right)\right)$is the oncave function on the first argument, the regret assessment is valid: $D_n \le \frac{\ln M}{h}$, which does not depend on n. Obviously, it is necessary to choose h as large as possible, while guaranteeing the exponential convexity of the loss function. For example, with

$$Z = Y = [0, 1]^d, \, l\left(z, y\right) = \sum_{i=1}^{d} \left(z_i - y_i\right)^2,$$

$h = \frac{1}{2(d+1)}$ for $d \geq 2$, and $h = \frac{1}{2}$ with $d = 1$. The regret assessment is:

$$D_n \leq \left\{ \begin{array}{l} 2\,(d+1)\ln M,\ d \geq 2 \\ 2\ln M, d = 1 \end{array} \right. . \tag{9}$$

Comparing assessments (7) and (8), we see that the assessment (9) is advantageous for

$$T \geq \left\{ \begin{array}{l} 4\,(d+1)\ln M,\ d \geq 2 \\ 4\ln M,\ d = 1 \end{array} \right. .$$

The regret of random selection is calculated as follows $D_n = ER_n - \min\limits_i L_i$. The expectation is used because the loss of the control element is a random variable. The regret assessments shown before for weighed estimation are also valid for random selection.

3. Experimental Analysis

In this section, the problem of recognizing two classes of black and white images is considered. Images are set on a raster size $S \times S$. Images contain noise and differ in shift. In addition, images can be of different sizes and rotate about their centers of mass. The control element must recognize the images without preliminary noise suppression, calculation of the shift, size and rotation. Local elements must recognize images with the same size and orientation. The experiment was implemented using PIL libraries for working with raster graphics and NumPy for working with multidimensional arrays in Python. In our experiment, the training of the control element was carried out in accordance with the algorithm shown below.

Algorithm. Training of a control element

Inputs: $X_1, ..., X_n$ - *vectors with training examples,* $y_1, ..., y_n$- *choices of environment ,* $Z_1, ..., Z_n$ - *vectors of local elements outputs*

Outputs: $z_1, ..., z_n$ - *the control element outputs.*
Steps:

1. Set U_0 =1, h= 0.5 and $l(Z, y) = (Z - y)^2$, where U_t – vector of local elements weights, $l-$ local element loss function.

2. While (isData) // to continue while the training examples are being submitted

$P_t = \frac{U_t}{\sum U_t}$// calculate the probability of selecting local elements, where in the denominator - the element-by-element vector sum of the local elements weights.

$z_t = z_{\xi_t,t}$ // calculate the output of the control element by the method of random selection (see formula 5).

$Z_t = X_t W^*$// calculate the outputs of local elements, where W^* - the matrix of trained local elements weights.

$L_t = l\left(Z_t, y_t\right)$ // submit the choice of environment to the control element and calculate local elements loss.

$U_{t+1} = U_t \exp\left(-hL_t\right)$ // calculate the weight vector of the local elements for the next t.

4. Experiment Setup

Step 1. Generating a training sample with linearly separated classes. A 30 * 30 raster and letters A and B were used when generating a training sample. The sizes of the images in pixels were calculated taking into account the specified font size. It is also worth noting that when letter rotating, the size of the letter area increases in height, so the maximum size along the diagonal of the letter area was selected. At the first stage, a sample was generated with letters shifted to the left, up, right, down by 2 pixels on the raster. At the second stage, the resulting sample was transformed by turning the letters 3 times (the degree of rotation is calculated automatically) and by a decrease in the size of the letters 3 times. 20 pictures with noise were added to all received samples. Thus a total of 12 samples, each of size 105 of letters A and B were received. (fig.2)

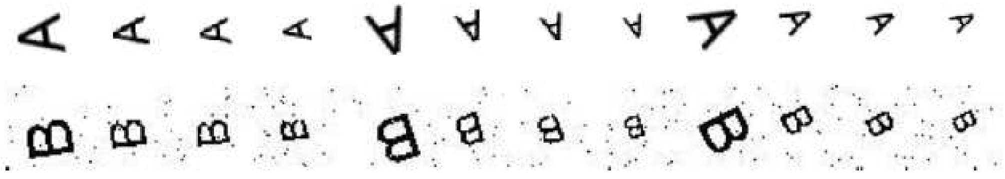

Figure 2: The examples of generated images for each sample (from 1 to 12)

Step 2. Training of local elements. 12 local elements were trained with the use of the generalized method of Kozinets.

Step 3. Environment setup. 12, 9, 6, and 3 samples were recorded. Letters from samples were submitted to the control element randomly with an

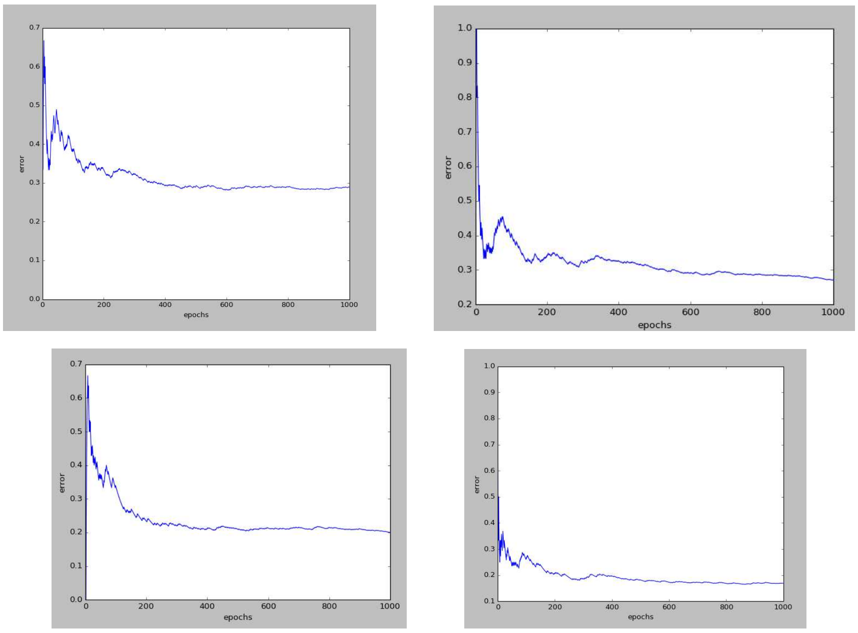

Figure 3: Graphs of errors of the hybrid neural network training. The abscissa axis is the number of examples, the ordinate axis is the error rate.

established probability for each sample.

Step 4. Training of a control element. The parameter h is set to a constant of 0.5. The interruption in the submission of examples was stopped when the error rate of the hybrid neural network stabilized. The error rate at each iteration was calculated as the ratio of the number of incorrectly recognized sample elements to the current number of submitted examples.

5. Experiment Results

Fig. 3 shows the results of four experiments. On the first two the error rate is close to 0.3, on the third and fourth ones it is close to 0.2 with stabilization. The first and second graphs correspond to the submission of images from 12 and 9 sample, the third and fourth graphics - the submission of 6 and 3 sample respectively. The results of the experiments are shown in Tables 1-4.

The tables show that each of the cases has a leader.

index	**image probability**	**expert probability**	**error**
1	0.00471885619097	6.37373143258e-59	0.41
2	0.116850264592	7.98222650701e-30	0.343
3	0.0151374689018	0.999664649869	0.276
4	0.141662795409	1.42468615378e-21	0.324
5	0.146148205655	2.3447647576e-59	0.411
6	0.118887351042	4.29458645791e-61	0.415
7	0.0316883876363	3.44132267332e-14	0.307
8	0.0956557552995	2.08358387806e-52	0.395
9	0.155103670378	0.000335350130466	0.284
10	0.0189233772149	2.00079893477e-75	0.448
11	0.0465458219924	9.5994597945e-24	0.329
12	0.108678045688	9.5994597945e-24	0.329

Table 1: Training results of the control element when submitting images from 12 samples.

index	**image probability**	**expert probability**	**error**
1	0.00471885619097	7.36299712224e-76	0.413
2	0.116850264592	5.03457535876e-45	0.342
4	0.156800264311	2.03109266273e-42	0.336
5	0.146148205655	1.3485799643e-77	0.417
6	0.118887351042	2.05388455404e-85	0.435
8	0.127344142936	0.999999999998	0.24
9	0.155103670378	9.35762296882e-14	0.27
10	0.0189233772149	1.07682818826e-106	0.484
12	0.15522386768	5.52108227702e-42	0.335
3		1.87952881654e-12	0.267
7		5.38018616001e-32	0.312
11		5.52108227702e-42	0.335

Table 2: Training results of the control element when submitting images from 9 samples

6. Conclusion

The computational experiment showed that the proposed method of separate training of a hybrid neural network works quite successfully in a complex sit-

index	**image probability**	**expert probability**	**error**
1	0.00471885619097	3.96142952134e-107	0.432
4	0.273650528903	8.7565107627e-27	0.247
5	0.146148205655	3.96142952134e-107	0.432
8	0.118887351042	2.1871378322e-148	0.527
9	0.301371190529	3.01440878507e-40	0.278
12	0.15522386768	1.64581143108e-38	0.274
2		1.06487866024e-63	0.332
3		0.999999999999	0.187
6		2.06960748968e-204	0.656
7		2.1871378322e-148	0.527
10		3.52017005458e-100	0.416
11		1.64581143108e-38	0.274

Table 3: Training results of the control element when submitting images from 6 samples

index	**image probability**	**expert probability**	**error**
1	0.278369385094	6.89001508487e-89	0.345
6	0.265035556697	1.32891155474e-110	0.395
12	0.456595058209	4.71116579184e-58	0.274
2		6.89001508487e-89	0.345
3		1.22540880294e-156	0.501
4		6.89001508487e-89	0.345
5		3.03748473759e-159	0.507
7		2.06115361819e-09	0.162
8		0.999999997939	0.142
9		1.22540880294e-156	0.501
10		3.03748473759e-159	0.507
11		4.71116579184e-58	0.274

Table 4: Training results of the control element when submitting images from 3 samples

uation involving the recognition of very different images without preliminary processing (noise suppression, image normalization). This is proved by the graphs of the hybrid neural network errors (fig.3). As a result of the training, the control element chose the local element that was least likely to have errors in the examples given by the environment. The proposed method can also be recommended for speech recognition of multiple speakers or for time series forecasting, when the time series model is not known in advance.

Acknowledgments

The reported study was partially supported by RFBR, research project No. 17-01-00888a.

References

[1] R.A. Jacobs, M.I. Jordan, S. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, *Neural Computation*, **3** (1991), 1-12, **doi:** 10.1162/neco.1991.3.1.79.

[2] N. Cesa-Bianchi, G. Lugosi, *Prediction, Learning, and Games*, Cambridge University Press, Cambridge, 2006, **doi:** 10.1017/CBO9780511546921.

[3] N. Cesa-Bianchi, Y. Mansour, G. Stoltz, Improved second-order bounds for prediction with expert advice, *Machine Learning*, **66**, No. 2 (2007), 321-352, **doi:** 10.1007/s10994-006-5001-7.

[4] E. Hazan, S. Kale, Extracting certainty from uncertainty: Regret bounded by variation in costs, *Machine Learning*, **80**, No. 2 (2010), 165-188, **doi:** 10.1007/s10994-010-5175-x.

[5] C.-K. Chiang, T. Yang, C.-J. Lee, M. Mahdavi, C.-J. Lu, R. Jin, S. Zhu, Online optimization with gradual variations, *Journal of Machine Learning Research*, **23** (2012).

[6] A. Rakhlin, K. Sridharan, A. Tewari, Online learning: random averages, combinatorial parameters, and learnability, *Adv. Neural Inf. Process. Syst.*, **23** (2010), 1984-1992.

[7] Dylan J. Foster, Alexander Rakhlin, Karthik Sridharan, ZigZag: A new approach to adaptive online learning, CoRR abs/1704.04010 (2017).

[8] Alexander Rakhlin, Karthik Sridharan, *A Tutorial on Online Supervised Learning with Applications to Node Classification in Social Networks*, CoRR abs/1608.09014 (2016).

[9] B.N. Kozinets, *Recursive Algorithm for Separation of Two Sets*, Soviet Radio (1973).