Using Neural Networks for Dynamic Reduction of the Features Space Dimensionality

Yury Tsoy[1]
[1]Computing Engineering Department, Tomsk Polytechnic University, Tomsk Russia
yurytsoy@gmail.com

Abstract— **The paper presents a modification of the Principal Components Analysis (PCA) for dimensionality reduction. The peculiarity of the method is that linear subspaces are removed dynamically via special criterion, which utilizes variances of data projections onto eigenvectors' estimates, whose coordinates are defined approximately. The latter gives rise to the method's name: pseudo-PCA. Two algorithms based upon this method are proposed for dimensionality reduction. The first algorithm applies idea of the Generalized Hebbian Algorithm, while the second algorithm uses evolutionary approach for neural networks training. The algorithms are tested and studied on classification problems and obtained results show applicability and efficiency of the algorithms for dimensionality reduction even though linear subspaces for removal are defined approximately.**

Keywords-principal components analysis; dynamical features space reduction; generalized hebbian algorithm; neuroevolution.

I. INTRODUCTION

Principal Components Analysis (PCA) is one of the most widely-used methods for dimensionality reduction for solving various real-world problems [1]. The method utilizes simple yet powerful idea that there are linear subspaces in the features space that do not contribute much in the data description and thus can be neglected without much harm to the information. PCA can be considered as a lossy compression algorithm since linear subspaces to be removed often comprise some information about the data.

Technically speaking PCA requires finding eigenvectors and eigenvalues for the data covariance matrix **C**. This can be done with use of special numeric procedures and the fact that **C** is symmetric allows more efficient numeric computations [2]. The common approach for computing eigenvectors for **C** is calculation of all eigenvectors and correspondent eigenvalues and then removal of eigenspaces with small eigenvalues. However this approach scales badly for large features spaces especially when the number of 'informative' features is relatively small, because a lot of unnecessary computations are made to find low-contributing eigenvectors.

From the artificial neural networks (ANN) point of view it was found that for linear ANNs without hidden layers Hebbian learning [3] leads to convergence of neural weights to eigenvectors. Generalized Hebbian Algorithm (GHA) introduced by E. Oja [4] provides more stable procedure for computing data covariance matrix eigenvectors and is currently widely used for neural PCA. Application of GHA either uses the approach described in the paragraph above or assumes 'guessing' of the correct dimensionality of reduced features space, which is generally impossible to do.

The paper presents method for training of ANN to perform PCA with dynamical removal of transformed features. The method doesn't require computation of exact coordinates of low-contributing eigenvectors, which gives advantages from the computational point of view, neither it demands guessing the dimensionality of the features space. It will be shown that the method even doesn't require knowing exact eigenvectors, hence the method's name: pseudo-PCA. The method is easy to use and has only one parameter to tune.

The paper is organized as follows. Section II gives some theoretical insights of the method and presents criterion for removal of features. Section III describes two algorithms, based upon the method. Experimental settings and classification problems are described in Section IV and Section V presents results of the experiments and comparison with existing classification algorithms. Section VI describes analysis of the execution time of the modification of GHA algorithm, presented in the paper. Conclusion contains paper summary and suggests possible improvements and research directions.

II. IDEA OF THE METHOD

The main question, which appears when features are removed dynamically during the training process, is how we can be sure that the removed features are low informative? If some criterion for removal is used then it is possible that this criterion was calculated approximately, which doesn't seem to improve our confidence. In the context of the PCA the following proposition can be proved [5]:

Proposition 1: Let $\mathbf{X}_i = \{\mathbf{X}_i, i = 1,...,N\}$, $\mathbf{X}_i \in R^n$ be a set of data points and $\mathbf{Q}_i = \{\mathbf{q}_i, i = 1,...,n\}$ is an orthogonal basis in R^n. We'll denote projection of data points from **X** onto the coordinate vector $\mathbf{q}_i$ as $proj_{\mathbf{q}_i}(\mathbf{X})$ and variance of projections as $Var(proj_{\mathbf{q}_i}(\mathbf{X}))$. Then summation over all dimensions

$$\sum_{i=1,...,n} Var(proj_{\mathbf{q}_i}(\mathbf{X})), \tag{1}$$

is constant and doesn't depend on the selection of **Q**.

This research was supported by the Russian Foundation for Basic Researches (projects no. 11-07-00027-a and 12-08-00296-a)

The proposition can be treated in the way that sum variance of data points projections onto orthogonal coordinate vectors can be considered as a sort of finite resource, which is distributed among coordinate vectors. Hence if variance of projections onto certain vector is increased, there must be a vector (or vectors), for which variance of projections reduces. This result can serve as a basis for suggestion that low-contributing eigenvectors will not be ”enhanced” (in the sense of projection variance) on the consequent iterations and thus can be neglected.

Thus the following criterion for removal of low-informative eigenvectors (subspaces) can be proposed (eigenvectors are supposed to be sorted by decrease of projection variance):

$$\frac{Var(proj_{\hat{\mathbf{q}}_1}(\mathbf{X}))}{Var(proj_{\hat{\mathbf{q}}_i}(\mathbf{X}))} < \tau, \qquad (2)$$

where $\hat{\mathbf{q}}_i$ – estimate of the i-th eigenvector, τ – is a threshold. The less the value of τ , the more eigenvectors will be neglected and thus more effective dimensionality reduction is possible. Typical values for τ are 5, 10, 15, …. However it’s not clear for now how to adjust this parameter properly. There are some experimental evidences, described below, which show that acceptable values for τ for the presented algorithms could be 15 or 20, to trade-off between dimensionality reduction and classification accuracy, but this setting should definitely depend on the training data ”geometry”.

The question of selection of proper value for τ is relative to selection of the target number of features after the dimensionality reduction and doesn’t have a universal answer for now, since there’ s often no exact data on informativeness of various features. Nevertheless it seems that dynamical tuning of the number of features using τ is more reasonable than trying to guess a suitable features space dimensionality in advance.

III. Algorithms

This section contains description of two algorithms, utilizing the idea of removal of linear subspaces using information on data projection variances. The first algorithm is modification of the well-known Generalized Hebbian Algorithm (GHA) [6], while the second uses evolutionary approach to training linear neural networks for dimensionality reduction.

A. Dynamical Generalized Hebbian Algorithm

The algorithm for dynamical removal of output nodes of the linear ANN uses traditional GHA procedure with additional dimensionality reduction step. The algorithm (Dynamical GHA, DGHA) looks like follows:

1. **Initialize** ANN using random orthogonal weights vectors and maximal possible number of outputs.

2a. Compute ANN responses to each training sample and calculate vector **var** of variances of output signals.

2b. Starting from the last element in **var** check whether criterion (2) is satisfied. If it does for some k-th element, then remove correspondent node from the network. If not, then proceed to **Step 3**.

3. **Update** ANN weights using traditional formula for GHA (for the training sample n and weight w_{ij}):

$$\Delta w_{ij}^{(n)} = \eta \left[y_j^{(n)} x_i^{(n)} - y_j^{(n)} \sum_{k=1}^{j} w_{ik}^{(n)} y_k^{(n)} \right],$$

where $x_i^{(n)}$ and $y_j^{(n)}$ are i-th element on input and j-th ANN output signal respectively for the n-th training sample; η – training rate.

4. **Continue** until stopping criterion is met.

The algorithm resembles GHA except that Steps 2a and 2b now regulate the number of outputs.

B. Neuroevolutionary Pseudo-PCA

The algorithm for neuroevolutionary (NE) solution of the pPCA is as follows:

1. **Initialize** random population, each individual is a candidate solution for pPCA.

2. **Evaluate** population (see the next Subsection).

3a. Compute vector of mean variances $\overline{\mathbf{var}}$ of data points projections onto eigenvectors’ estimates by all individuals in the population.

3b. Starting from the last element in $\overline{\mathbf{var}}$ check whether criterion (2) is satisfied. If it does for some k-th element, then remove correspondent genes from all individuals in population. If not, then proceed to **Step 4**.

4. **Selection**

5. **Crossing** and **Mutation**.

6. If algorithm’s run is completed then proceed to **Step 7**, otherwise proceed to **Step 2**.

7. Return the best found individual.

Due to the **Step 3** there is a possibility to remove output nodes from the ANN under training. Thus this step implies reduction of the features space dimensionality. To implement evaluation of fitness and crossing of individuals special operators procedures are used, which are described below.

1) Fitness Evaluation

The following formula is used for calculation of fitness:

$$f = \alpha \sum_i Var(proj_{\hat{\mathbf{q}}_i}(\mathbf{X})) \to \max \qquad (3)$$

$$\alpha = (\hat{\mathbf{q}}_0^T \mathbf{r})^2, \mathbf{r} = \frac{\mathbf{C}\hat{\mathbf{q}}_0}{\|\mathbf{C}\hat{\mathbf{q}}_0\|},$$

Coefficient α is necessary to evaluate how close the most “principal” eigenvector estimate to the true eigenvector. Let $\hat{\mathbf{q}}_1$ be a normalized eigenvector then

$\alpha = \left(\frac{\hat{\mathbf{q}}_1^T \mathbf{C} \hat{\mathbf{q}}_1}{\left\| \mathbf{C} \hat{\mathbf{q}}_1 \right\|} \right)^2 = \left(\frac{\lambda \hat{\mathbf{q}}_1^T \hat{\mathbf{q}}_1}{\left\| \lambda \hat{\mathbf{q}}_1 \right\|} \right)^2 = 1$, and if $\hat{\mathbf{q}}_1$ is not an eigenvector then $\alpha < 1$.

Note that expression (3) contains sum of projection variances, which was proved to be constant. However this sum is constant only when estimates of all n principal components are considered. When some component is removed the sum over coordinate vectors for remaining subspaces will depend on those vectors' coordinates and since individuals in population will represent different principal components' estimates this sum will be different for reduced feature spaces.

Before the evaluation weights of ANN are transformed to the orthonormal vectors using Gram-Schmidt process. For more details on the evaluation process see [5].

2) Crossing

The crossing of two individuals i and j results in one offspring using expression:

$$\mathbf{c}^{(k)} = \mathbf{w}_i^{(k)} + \frac{|v_i^{(k)} - v_j^{(k)}|}{\left\| \mathbf{w}_i^{(k)} - \mathbf{w}_j^{(k)} \right\|} (\mathbf{w}_i^{(k)} - \mathbf{w}_j^{(k)}), \qquad (4)$$

where $\mathbf{w}_i^{(k)}$, $\mathbf{w}_j^{(k)}$, and $\mathbf{c}^{(k)}$ – weights of the k-th node of the i-th and j-th parent and offspring ANN respectively; $\|\mathbf{x}\|$ – Euclidian norm for vector **x**.

The fraction in (4) is necessary to approximate the absolute value of gradient of k-th vector so that overall expression (4) can be treated as linear approximation for the update of k-th weight vector moving from the $\mathbf{w}^{(k)}$.

IV. Experiments Description

Testing of the proposed pPCA method will be conducted using several classification problems from the Proben1 set [7], namely: cancer1, card1, diabetes1, glass1, heart1, horse1, soybean1, thyroid1. Some numerical characteristics of these problems are given in the Table I.

TABLE I. GENERAL PARAMETERS OF THE PROBLEMS FROM THE PROBEN1 TEST SET

Problem	# of features	# of classes	Training / Validation / Test sets sizes
cancer1 (9)	9	2	350 / 175 / 174
card1 (51)	51	2	345 / 173 / 172
diabetes1 (8)	8	2	384 / 192 / 192
glass1 (9)	9	6	107 / 54 / 53
heart1 (35)	35	2	460 / 230 / 230
horse1 (58)	58	3	182 / 91 / 91
thyroid1 (21)	9	3	3600 / 1800 / 1800

Algorithms, described in the Section III are to be used for reduction of the features space dimensionality and after that transformed objects descriptions will be used to train feed-forward ANN using traditional gradient training. Scheme for combination of the resulting ANNs is shown in Fig. 1. After ANN-1 and ANN-2 are trained they can be united into a single ANN in an obvious way.

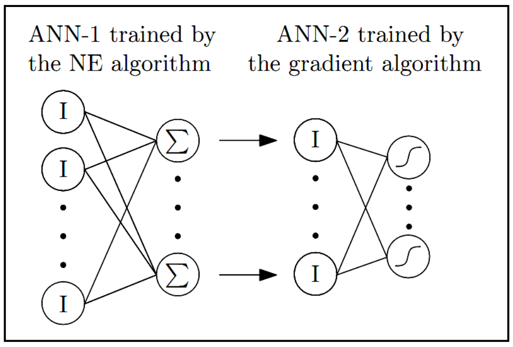
ANN-1 trained by the NE algorithm
ANN-2 trained by the gradient algorithm
I
Σ

Figure 1. Combination of ANN-1, trained using neuroevolutionary approach for dimensionality reduction, and ANN-2 trained for classification. Both networks doesn't contain hidden layers and output signals from ANN-1 serve as inputs for ANN-2. ANN-1 has linear activations, while output nodes for ANN-2 use "standard" sigmoid activation functions: $f(\mathbf{x}) = (1 + \exp(-\mathbf{w}^T\mathbf{x}))^{-1}$. "I" means identity activations, $f(x) = x$; x R. The same scheme applies for the DGHA algorithm.

The following values for criterion value τ for removal of non-informative linear subspaces according to (2) to be considered: 5, 10, 15, 20.

During each run a pPCA solution to be found and for this solution 10 ANNs will be trained, using the RPROP algorithm [8] with early stopping with 10% threshold. From these 10 ANNs only one is selected using classification error on a validation data set. This winning ANN is used for classification of samples from a test data set. For each problem 10 runs are performed in this way and the obtained mean classification accuracy is used for comparison and analysis.

Parameters settings for DGHA and NE pPCA are presented in the Table II. These parameters were picked experimentally and parameters for the RPROP algorithm were set according to the recommendations from [8].

TABLE II. PARAMETERS SETTINGS FOR DGHA AND NE PPCA

Parameter	Value
Encoding	Real
Population Size	50
Generations/epochs number	50
Selection	Tournament (tour size = 5)
Crossover probability	0.7
Mutation probability	0.05
Training rate	1/T, where T – current training epoch

Software implementation of pPCA and its application for solving classification problems uses the MentalAlchemy [9] and the Encog [10] open-source libraries and is available as part of the MentalAlchemy project.

V. Results of Experiments

Test set classification errors are given in the Tables III and IV. Also the number of ANN outputs (remaining principal components estimates) in result of pPCA for DGHA and NE algorithms is given in brackets to evaluate dimensionality reduction.

TABLE III. INFORMATION ON THE PROBLEMS FROM THE PROBEN1 TEST SET

Problem	$\tau = 5$		$\tau = 10$	
	DGHA	NE pPCA	DGHA	NE pPCA
cancer1 (9)	2.3 (1)	2.30 (1)	1.7 (2.5)	2.82 (1.2)
card1 (51)	14.48 (8.3)	16.28 (28.5)	13.66 (11.7)	15.41 (50.7)
diabetes1 (8)	24.74 (7.6)	24.95 (7.6)	24.38 (8)	25.00 (8)
glass1 (9)	71.7 (1)	36.23 (5.5)	40.38 (4.3)	33.02 (6.7)
heart1 (35)	21.3 (9.9)	21.13 (22.3)	21.74 (15.7)	19.91 (31.5)
horse1 (58)	34.07 (1)	28.79 (35.3)	32.86 (5.3)	29.23 (57.7)
thyroid1 (21)	7.24 (7)	7.19 (8.9)	7.21 (8)	6.03 (16.3)

TABLE IV. INFORMATION ON THE PROBLEMS FROM THE PROBEN1 TEST SET

Problem	$\tau = 15$		$\tau = 20$	
	DGHA	NE pPCA	DGHA	NE pPCA
cancer1 (9)	1.44 (4)	1.78 (4,6)	1.67 (5.6)	1.84 (6.3)
card1 (51)	16.8 (16.4)	15.64 (51)	16.4 (19.8)	15.76 (51)
diabetes1 (8)	24.43 (8)	25.00 (8)	24.64 (8)	25.00 (8)
glass1 (9)	37.55 (6.8)	32.07 (7.9)	34.91 (7.4)	32.26 (8.4)
heart1 (35)	22.52 (17.7)	20.00 (34.2)	21.13 (19.1)	20.04 (35)
horse1 (58)	29.89 (27.3)	30.66 (58)	26.81 (32.4)	29.56 (58)
thyroid1 (21)	6.78 (14)	5.87 (18)	6.73 (15)	5.92 (19.8)

It can be seen that DGHA produces more active removal of low-contributing subspaces, than NE pPCA. There were cases when DGHA was dramatically more efficient at dimensionality reduction, see results for *cancer1* and *horse1* problems. From the other hand this sometimes lead to bad classification using the transformed features space (see the *glass1* problem), because many important features were ripped off by the DGHA algorithm. This indicates that use of DGHA algorithm requires more accurate setting of the τ parameter value.

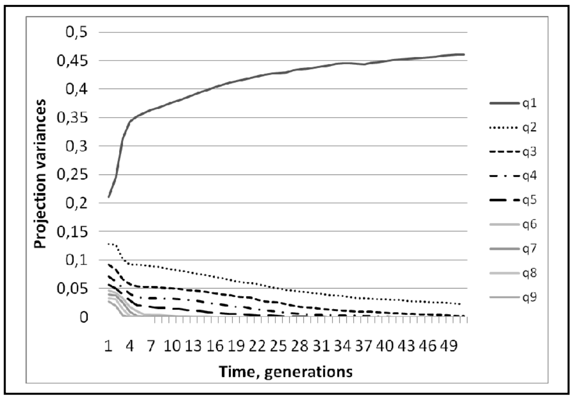

Figure 2. Change projection variances onto eigenvectors estimates for the NE pPCA algorithm for *cancer1* problem for $\tau = 10$. The results are averaged over 100 runs.

In order to figure out why DGHA is able to remove low-informative linear subspaces plots for change of projection variances were made (see examples for cancer1 problem in Fig. 2 and 3). While NE pPCA had only came close to the maximum projection onto the 1st eigenvector estimate, the DGHA had been there from the very first steps. Thus DGHA is able to obtain quite 'good' estimates for the eigenvectors much faster than the NE pPCA algorithm, but additional regularization is required to prevent 'ill-judged' features space dimensionality reduction.

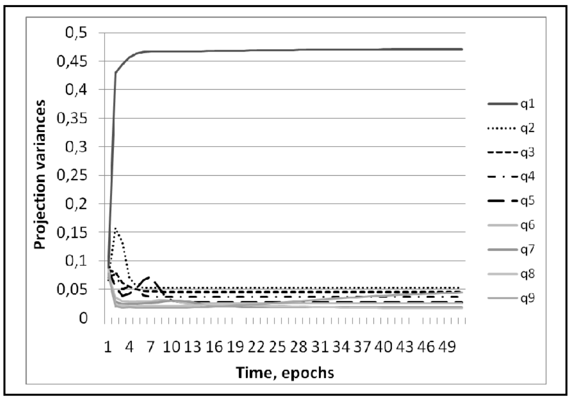

Figure 3. Change projection variances onto eigenvectors estimates for the DGHA algorithm for *cancer1* problem for $\tau = 10$. The results are averaged over 100 runs.

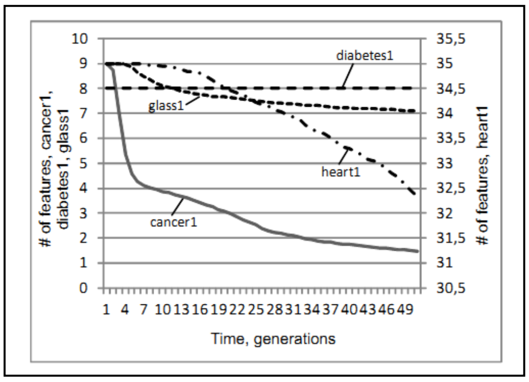

Figure 4. Change of a mean number of ANN outputs (features space dimensionality) trained using NE pPCA over time for the problems *cancer1*, *diabetes1*, *glass1*, *heart1*, for $\tau = 10$. The results are averaged over 100 runs.

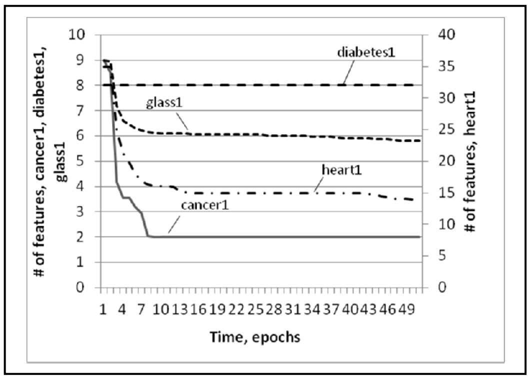

Figure 5. Change of a mean number of ANN outputs (features space dimensionality) trained using DGHA over time for the problems *cancer1*, *diabetes1*, *glass1*, *heart1*, for $\tau = 10$. The results are averaged over 100 runs.

Classification accuracy was compared to that of obtained by other methods, namely: traditional ANN training with manual tuning of a structure [7]; neuroevolutionary training of

ANN with fixed structure [11]; ANN training with pruning and further retraining [12]. The results are presented in the Table V.

TABLE V. CLASSIFICATION ERRORS (%) OBTAINED USING SOME OTHER APPROACHES. FOR [7], [11], [12] THE BEST OBTAINED RESULTS ARE GIVEN

Problem	Best classification result, %			NE pPCA, $\tau = 15$	DGHA, $\tau = 20$
	[7]	*[11]*	*[12]*		
cancer1 (9)	1.38	1.24	1.1	1.78	1.67
card1 (51)	14.05	14.27	13.7	15.64	16.4
diabetes1 (8)	24.10	23.70	20.8	25.00	24.64
glass1 (9)	32.7	47.62	30.2	32.07	34.91
heart1 (35)	19.72	21.87	18.5	20.00	21.13
horse1 (58)	29.19	26.44	26.9	30.66	26.81
soybean1 (82)	9.06	8.47	N/A	11.94	6.73
thyroid1 (21)	2.32	6.12	5.7	5.87	1.67

In order to estimate efficiency of the dimensionality reduction using the pPCA ($\tau = 15$) a comparison with results from the literature was made (Table VI). The methods are: Layered Genetic Programming (FLGP) [13]; Recursive Feature Elimination combined with multi-layered neural network (RFENN) and support vector machines (RFESVM) [14].

The results from the Table VI show that NE pPCA shows slightly worse performance considering the resulting dimensionality of the features space, but it is compensated by lower classification error rates. DGHA looks more competitive from the number of features point of view and also provides good features transform for classification.

TABLE VI. COMPARISON OF THE TEST SET CLASSIFICATION ERRORS (%) OBTAINED USING DIFFERENT FEATURES SELECTION METHODS FOR CANCER1, DIABETES1 AND HEART1 PROBLEMS. AVERAGE DIMENSIONALITY OF THE RESULTING

Method	cancer1	diabetes1	heart1
FLGP [13]	2.24 (5.2)	27.24 (6.1)	22.40 (11.0)
RFENN [14]	4.00 (7)	24.90 (2)	21.00 (27)
RFESVM [14]	3.70 (7)	24.50 (3)	20.00 (18)
DGHA, $\tau = 20$	1.67 (5.6)	24.64 (8)	21.13 (19.1)
NE pPCA, $\tau = 15$	1.78 (4.6)	25.00 (8)	20.00 (34.2)

It is necessary to stress that both NE pPCA and DGHA reduce dimensionality of the features space when the information on eigenvectors coordinates is inexact.

However not looking at the latter argument the method seems to have a ”self-repairing” feature: even if inexact linear subspaces were removed to cause a representation error, the remaining components’ coordinates will be defined more precisely on consequent steps of the algorithm, which means that the remaining linear subspace is rotated and this rotation should diminish the error.

VI. DGHA Execution Time Analysis

Since DGHA presents modification of GHA, which considers online removal of output nodes, it involves reduction of computational complexity and thus should provide advantage in terms of execution time. Dependence of the time for computation for different τ for the *heart1* problem is shown in Fig. 6. Also GHA execution time is given for reference.

It is clear that DGHA performs much faster due to significant reduction of the number of ANN weights. Note that the computations time for DGHA tends to grow more slowly as τ value increases and dimensionality of the resulting features space growths not very fast.

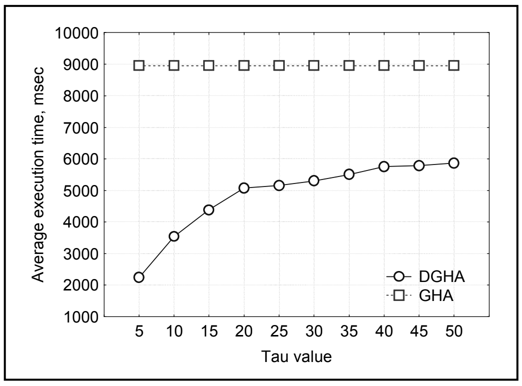

Figure 6. Change of a average computation time for 50 epochs for the GHA and DGHA algorithms on the *heart1* problem, for different τ values. The results are averaged over 100 runs.

Note that if no dimensionality reduction is performed, which means that all features are important, then DGHA should take slightly more time to operate due to necessity to compute output variances. For example in the *diabetes1* problem for $\tau \geq 10$ DGHA didn’t remove output nodes. The execution time for DGHA and GHA for the *diabetes1* are shown in the Fig. 7.

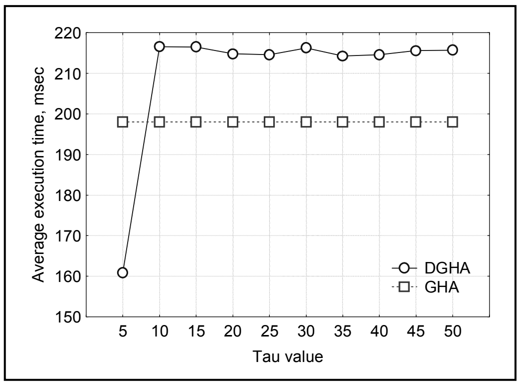

Figure 7. Change of a average computation time for 50 epochs for the GHA dna DGHA algorithms on the *diabetes1* problem, for different τ values. The results are averaged over 100 runs.

It can be seen that DGHA is indeed slightly slower than GHA, but slow down is within 10% for this particular problem. For $\tau = 5$ DGHA eventually reduced features space by 2, which lead to the execution time advantage.

VII. Conclusion

Two algorithms for online dimensionality reduction were presented based upon original method for dimensionality reduction. The first algorithm DGHA is a modification of the well-known GHA algorithm and the second, NE pPCA, is an neuroevolutionary algorithm for training linear ANNs.

Experimental research showed that DGHA is able to reduce dimensionality much faster than NE pPCA, however with incorrectly selected threshold parameter τ this removal may become too harsh. It was found that the reason for this is that DGHA improves projection variances very efficiently and hence presents faster procedure to estimate eigenvectors.

Comparison with existing methods for classification and dimensionality reduction showed that both DGHA and NE pPCA look competitive from the classification accuracy and the dimensionality reduction point of view.

Comparison of execution times for DGHA and GHA showed that for the cases when dimensionality reduction is possible DGHA performs faster because it doesn't involve unnecessary computations for the eigenvectors to be removed. The obtained speed-up scales with the number of removed features. If however no dimensionality reduction takes place then DGHA is slightly slower.

In order to avoid undesirable effect of removal of relevant features spaces by DGHA two solutions can be proposed:

1. Reduction of the DGHA training time. This can potentially prevent DGHA from reaching exact values of eigenvectors due to time limits. However this approach doesn't seem reliable and its usefulness is under question.
2. Constraining portion of information which can be removed, by say 10% threshold. This information can be roughly measured by sum of data projection variances onto the remaining linear subspaces and if this sum is smaller than 0.9 of the maximal, then no more outputs should be removed from ANN.

ACKNOWLEDGMENT

The author thanks Dr. Yu. Burkatovskaya for valuable comments on the pPCA method.

REFERENCES

[1] I.T. Jolliffe, Principal Component Analysis. Springer Series in Statistics. NY: Springer, 2002.

[2] G.H. Golub, C.F. Van Loan, Matrix Computations. Second Edition. Baltimor, London: The John Hopkins University Press, 1989.

[3] D. Hebb The Organization of Behaviour.John Wiley & Sons, 1949.

[4] E. Oja, "Simplified neuron model as a principal component analyzer" Journal of Mathematical Biology, vol. 15, no. 3. pp. 267-273, 1982.

[5] Tsoy Y. "Evolving Linear Neural Networks for Features Space Dimensionality Reduction", Proceedings of the 2012 IEEE International Joint Conference on Neural Networks (IJCNN 2012), 2012, in press.

[6] S. Haykin, Neural Networks: A Comprehensive Foundation (2 ed.). Prentice Hall, 1998.

[7] Prechelt L. "PROBEN1 – a set of neural network benchmark problems and benchmarking rules" Technical Report 21/94, Universitat Karlsruhe, Karlsruhe, Germany, 1994.

[8] M. Riedmiller "Rprop - Description and Implementation Details", technical report, Universtiy of Karlsruhe, 1994.

[9] http://code.google.com/p/mentalalchemy/

[10] http://www.heatonresearch.com/encog

[11] F.Y. Barrera "Busqueda de la estructura optima de redes neurales con Algoritmos Geneticos y Simulated Annealing. Verificacion con el benchmark PROBEN1 Inteligencia Artificial", Revista Iberoamericana de Inteligencia Artificial, vol. 11, no. 34, pp. 41-61, 2007.

[12] N. Weman "Empirical Investigation of the Effect of Pruning Artifical Neural Networks With Respect to Increased Generalization Ability", Linkopings universitet, Sweden, 2010.

[13] I. De Falco, A. Della Cioppa, E. Tarantino "Discovering interesting classification rules with genetic programming Applied Soft Computing", vol. 1, no. 4, pp. 257-269, 2002.

[14] T. Windeatt, R. Duangsoithong, R. Smith "Embedded Feature Ranking for Ensemble MLP Classifiers" IEEE Transactions on Neural Networks, vol. 22, no. 6, pp. 988-994, 2011.