EPiC Series in Computing

Volume 50, 2017, Pages 221–232

GCAI 2017. 3rd Global Conference on Artificial Intelligence

Implementation of Incremental Learning in Artificial Neural Networks

Mariela Andrade, Eduardo Gasca, and Eréndira Rendón

Toluca Institute of Technology, Mexico.
andradeh.mariela@gmail.com, gasca_eduardo@hotmail.com

Abstract

Nowadays, the use of artificial neural networks (ANN), in particular the Multilayer Perceptron (MLP), is very popular for executing different tasks such as pattern recognition, data mining, and process automation. However, there are still weaknesses in these models when compared with human capabilities. A characteristic of human memory is the ability for learning new concepts without forgetting what we learned in the past, which has been a disadvantage in the field of artificial neural networks. How can we add new knowledge to the network without forgetting what has already been learned, without repeating the exhaustive ANN process? In an exhaustively training is used a complete training set, with all objects of all classes.

In this work, we present a novel incremental learning algorithm for the MLP. New knowledge is incorporated into the target network without executing an exhaustive re-training. Objects of a new class integrate this knowledge, which was not included in the training of a source network. The algorithm consists in taking the final weights from the source network, doing a correction of these with the Support Vector Machine tools, and transferring the obtained weights to a target network. This last net is trained with a training set that it is previously preprocessed. The efficiency resulted of the target network is comparable with a net that is exhaustively trained.

1 Introduction

Since the beginning of scientific progress, humans have to create machines capable of performing processes with certain intelligence. Initially, his goal was to create automatons, whose function was to make tasks typical of the humans. The human brain is a highly complex, non-linear, and parallel in function. This means that the brain can perform many operations simultaneously. This is unlike ordinary computers that are of sequential type, that is, the computers can only execute an operation at a time. Artificial Neural Networks (ANN) have shown their effectiveness and maturity in numerous applications, in particular the Multilayer Perceptron. One of the biggest frustrations which faces investigators working with classifiers is that most of them cannot be trained with new data, without forgetting previously acquired knowledge. That is, the classifiers are not equipped with Incremental Learning. To carry out these tasks of learning, the conventional classifiers would have to be fully re-trained when new samples are presented, together with the previous data. This type of solution is very impractical because it is necessary

C. Benzmüller, C. Lisetti and M. Theobald (eds.), GCAI 2017 (EPiC Series in Computing, vol. 50), pp. 221–232

to store all available training sets, computational processing time is high and in the most cases, the topology of the classifiers being used must be completely redefined [3].

Incremental Learning is the gradual acquisition of knowledge, without to have to discard the already obtained or to repeat the learning process completely [20]. In the case of ANN, such as MLP, when input new data, the network forgets the previous knowledge to give way to the learning of the new data. This is a manifestation of the so-called stability-plasticity dilemma [7]. To solve this problem, an algorithm of Incremental Learning must incorporate the new knowledge into that previously acquired. On the other hand, in view of the learning approaches of human beings, it seems natural to build successive learning upon prior results. This is essentially a feature of incremental learning [21].

Therefore, an algorithm that can learn from new data without requiring access to previously used data would ideally retain the formerly acquired knowledge, accommodate new classes, and estimate the confidence in its own classification. In this way, it can be a good method to classify, when it is intended to work on databases that become increasingly large, particularly if new samples arrive at any time [17]. The Incremental Learning may also be addressed in the context of training data manipulation [21]. For instance, in [6], an incremental learning strategy is implemented through the selection of most informative training samples.

The aim of this work is the implementation of the Incremental Learning in the ANN, especially in the MLP. In these approaches, knowledge resides in the connections between neurons (weights), so it is necessary a procedure that allows the modification the values of the weights to integrate elements of a new class, without losing the knowledge already acquired, all of this without exhaustive training. For such task, we consider that Incremental Learning consists in the addition of a new class into a target network, which was not considered in the training of the source network. We present an algorithm for Artificial Neural Networks of type Multilayer Perceptron, using Support Vector Machine for making the calculation of the new weights. We also used the preprocessing technique called Wilson's Editing.

Researchers have tried different ways to solve the Incremental Learning paradigm. For example, Robi Polikar and collaborators introduce Learn++, an algorithm for incremental training of an ANN pattern classifier. Learn++ use an ensemble of classifiers for generating multiple hypotheses. They consider that an algorithm has Incremental Learning if it is able to learn additional information from data. It should not require access to the original data, it should preserve previously acquired knowledge, and it should be able to accommodate new classes that may be introduced with new data [16].

Later, [14, 15] proposed a method based on introducing a forgetting function in an incremental on-line learning algorithm for two-layer feedforward neural networks. In this method, they call Incremental Learning when the ANN use a training sample that containing patterns that change over time (non-stationary environment).

Recently, M. Zribi and Y. Boujelbene [24] used neural networks with an Incremental Learning algorithm as a tool to classify a mass in the breast. The incremental learning in this algorithm consists of adding a neuron in the hidden layer until the network improvements are not significant. The expectations of achieving Incremental Learning have been many, and this has been applied in other classifiers, examples of which are the Support Vector Machines(SVM), which have demonstrated accuracy and speed in learning [12], for example, an incremental algorithm to train SVM by using the selected samples violating KKT conditions. In this example, Incremental Learning consists of adding a new sample, and verifying which support vectors violate KKT conditions, training the new samples, re-obtaining the support vectors, and the optimal hyperplane [23].

Lately, new proposals have emerged that could be confused with ours as mentioned below.

One of these is the transfer of knowledge proposed by Geoffrey Hinton and collaborators [9], in which it is possible to compress the knowledge in an ensemble into a single model which is much easier to deploy. This uses a different training that they call "distillation" to transfer knowledge from large ensemble of models to a single small model that is more appropriate for your deployment. This methodology is different to ours because unlike them, we do not use a set of neural networks; we only need a source network trained with the totality of objects that transfers knowledge of the input layer to the hidden layer, to the target network that has an additional node in the output layer. Previously Pratt [18] had already done a transfer knowledge but did not add a node in the output layer and passed only the best hyperplane.

The shared representations usually use deep learning and are based on the integration of different types of data in the same space [10]. Deep learning has begun exploring models that embed images and words in a single representation. In [11] the core idea is to extract deep knowledge of the Convolutional Neural Network model from a large data set and then transfer the knowledge to an ocean front recognition task on limited remote sensing (RS) images. On the other hand, in [8] the authors apply knowledge transfer to deep convolutional neural nets. This transfer is completed in such a way that one can envision creating the net that could learn new concepts throughout its lifetime. This is similar to what we are looking for with the application of Incremental Learning in Multilayer Perceptron networks. In contrast, the algorithm proposed here is only applicable to Multilayer Perceptron networks. We transfer knowledge of the classes learned previously so that when the network receives an unknown class, it does not forget the classes learned previously. Another important difference is that in our algorithm, the initial weights of the network are the weights transferred from the source network but only the weights from input layer to hidden layer.

The rest of this paper is organized as follows. Section 2 describes the Multilayer Perceptron and Linear Support Vector Machines; in Section 3 we describe the Incremental Learning Algorithm; Section 4 includes the experimental procedure and results; the conclusions are in Section 5.

2 Preliminaries

2.1 Multilayer Perceptron

Multilayer Perceptron is a feedforward artificial neural network. This network is composed of a layer of input units, another layer of output units and a certain number of intermediate layers of process units, also called hidden layers because the outputs of said neurons are not seen and have no connections to the outside. Figure 1 shows the Multilayer Perceptron (MLP) structure with a hidden layer. MLP is widely used for pattern classification, recognition, prediction, and approximation. Multilayer Perceptron can solve tasks that are not linearly separable.

There are different algorithms to perform the adjustment of the weights in a MLP; in this paper, we will make use of the Backpropagation (BP) algorithm. During the MLP learning process, it makes use of the training sample and the BP algorithm. It consists in the modification of the weights values, to the presentation of the patterns contained in the sample of training. Such a change is performed considering the minimization of the mean square error, which quantifies the difference between the correct class and the assigned class by the network to the pattern of entrance, equation (1).

$$E = \frac{1}{2N_s} \sum_{i=1}^{N_s} (\varepsilon_i^p)^2 \tag{1}$$

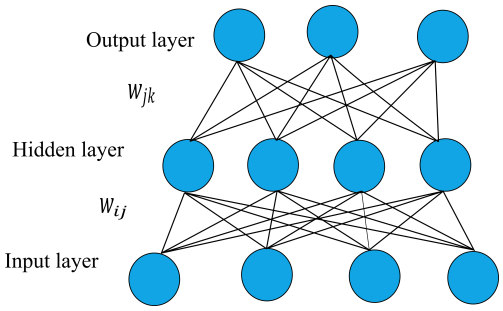

Figure 1: Multilayer Perceptron of three layers. The connections between nodes ($w..$) are denominated weights, and they are modified during the training of the network.

where: $\varepsilon_i^p = (d_i^p - s_i^p)$ is the error between the desired value (d_i^p) and the output produced by the network (s_i^p) of the $p-th$ pattern in the $i-th$ node of the output layer, and N_S is the number of nodes of the output layer.

For calculating the mean quadratic error, the BP algorithm employs the gradient descent to optimize the values of the weights that minimize the error, using the expression given in equation (2):

$$\Delta w(t) = -\eta \nabla_w E_p + \alpha \Delta w(t-1) \tag{2}$$

where: η is the learning rate, $\nabla_w E_p$ is the gradient of the function of error with respect to the weight, α is the momentum, and t is the number of iterations.

2.2 Linear Support Vector Machines

A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression purposes. SVM uses linear models to implement nonlinear class boundaries. It transforms the input space using a nonlinear mapping into a new space. In this work, we will assume that we have linearly separable classes, so we will explain this briefly.

According to Christopher J.C. Burges [4]. First, we label the data for training $\{\mathbf{x}_i, y_i\}$, with $i = 1, ..., l, y_i \in \{-1, 1\}$, and $x_i \in \mathbf{R}^d$. Then, we propose some hyperplane that separates the positive from the negative examples. The points $\mathbf{x}$ which lie on the hyperplane that satisfies $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ is a vector normal to the hyperplane, $\frac{|b|}{\|\mathbf{w}\|}$ is the perpendicular distance from the hyperplane to the origin, and $\|\mathbf{w}\|$ is the Euclidean norm of $\mathbf{w}$. Let $d_+(d_-)$ be the shortest distance from the separating to the hyperplane to the closest positive (negative) example. Define the "margin" of a separating hyperplane to be $d_+ + d_-$. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin . This can be formulated as follows: suppose that all the training data satisfy the following constraints:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad for \quad y_i = +1 \tag{3}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad for \quad y_i = -1 \tag{4}$$

These can be combined into one set of inequalities:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq -\forall_i \tag{5}$$

Now consider the points for which the equality in Eq. (6) holds. These points lie on the hyperplane $H_1 : \mathbf{x}_i \cdot \mathbf{w} + b = 1$ with normal $\mathbf{w}$, and perpendicular distance from the origin $\frac{|1-b|}{\|\mathbf{w}\|}$. Similarly, the points for which the equality in Eq. (7) holds lie on the hyperplane $H_2 : \mathbf{x}_i \cdot \mathbf{w} + b = 1$ with normal again $\mathbf{w}$, and the perpendicular distance from the origin $\frac{|-1-b|}{\|\mathbf{w}\|}$. Hence $d_+ = d_- = \frac{1}{\|\mathbf{w}\|}$ and the margin is simply $\frac{2}{\|\mathbf{w}\|}$. Note that H_1 and H_2 are parallel and that no training points fall between them. Thus, we can find the pair of hyperplane, which gives the maximum margin by $\|\mathbf{w}\|^2$, subject to constraints (5). Thus, we expect the solution for a typical two dimensional case to have the form shown in Figure 2. Those training points for which the equality in Eq. (5) holds, and whose removal would change the solution found, are called support vectors.

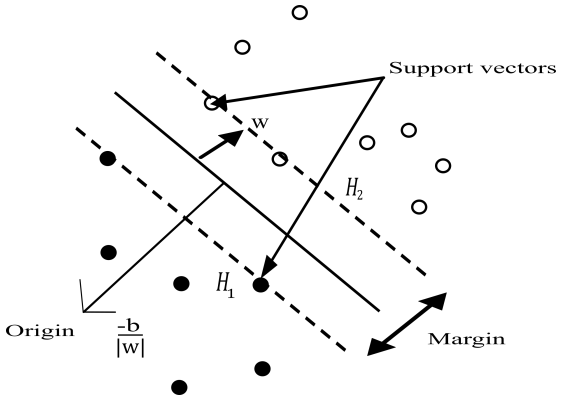

Figure 2: Linear separating hyperplanes for the separable case.

We will now switch to a Lagrangian formulation of the problem. The constraints (5) will be replaced by constraints on the Lagrange multipliers themselves, which will be much easier to handle. Thus, we introduce positive Lagrange multipliers $\alpha_i, i = 1, ..., l$, one for each of the inequality constraints (5). This gives Lagrangian:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{6}$$

requiring that the gradient of L_P with respect to $\mathbf{w}$ and b vanish give the conditions:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \tag{7}$$

$$\sum_i \alpha_i y_i \mathbf{x}_i = 0 \tag{8}$$

Since these are equality constraints in the dual formulation, we can substitute them into Eq. (6) to give

$$L_D = \sum_i \alpha_i - \frac{1}{2}\alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{9}$$

Support vector training therefore amounts to maximizing L_D with respect to the α_i, subject to constrains (8) and positive of the α_i, with solution given by (7). In the solution, those points

for which $\alpha_i > 0$ are called "support vectors", and lie on one of the hyperplanes H_1, H_2 and all other training points have $\alpha_i = 0$, and lie either on H_1 or H_2 [4]. Now we need to find the parameters and based on the linear equations as many support vectors as it has.

3 Incremental Learning Algorithm

Our proposed Incremental Learning Algorithm (ILA) is described below.

First, the training of the source network was carried out with the partially exposed sample. That is to say, all objects of a class were removed as shown in Table 2. Second, we identify the weights from the input layer to the hidden layer, which integrate the hyperplanes to transfer from the source network to target network. Third, we use SVM to perform the adjustment of the weights of the input layer to the hidden layer. Fourth, the target network is designed by adding a node in the output layer, (Figure 3), which allows adding a new class, except that the new node the rest of the topology remains the same as the source network.

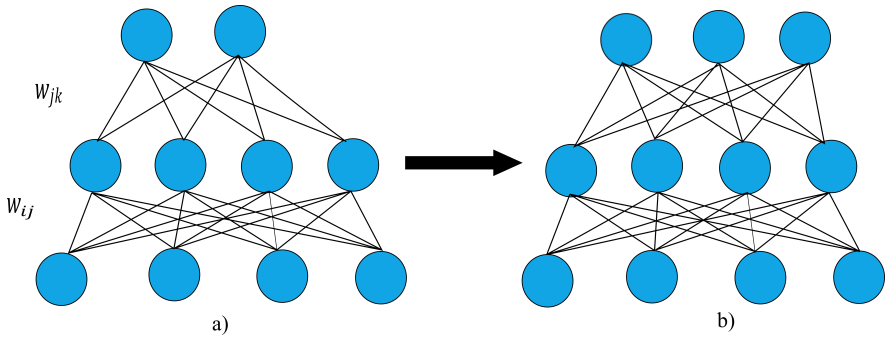

Figure 3: Transfer of weights, a) source network, and b) target network.

Fifth, the weights, set in the third step, are transferred to the target network. The rest of the weights are initialized randomly. Sixth, a non-exhaustive retraining is performed, using a training sample (TS) formed with the objects of the new class, and the result of the preprocessing of the source training sample. Furthermore, a smaller amount of epochs (50%) is used in the training.

The preprocessing technique employed in this work is the so-called Wilson's Editing [22], with the objective of to eliminate the elements found in the overlap between classes. To do this, it uses the k-nearest neighbors algorithm (k-NN). If the majority of the k neighbors belong to the class of the element in evaluation, it remains in the training sample, otherwise it is eliminated. For this study, we used k =3. We have used this algorithm to approximate our problem to linearly separable classes. All the designed ANN were Multilayer Perceptrons, with three layers described in Section 2.1: input, one hidden, and output layer. The Backpropagation algorithm for multiclass (Section 2.1) was employed for the training of all ANN. The following section contains the description of the weights calculation

3.1 Algorithm for Weights Calculation

The algorithm to calculate weights that are transferred from the source network to the target network is the following:

1. Read the training sample and identify the weights that integrate each one of hyperplanes of the source network. We have as many hyperplanes as nodes in the hidden layer.

2. Identify the two classes closest to each hyperplane. We propose the following equation to do so:

$$Dch = |dc - (dp2 - dp1)| \tag{10}$$

where Dch is the distance from the class to the hyperplane, the distance from the centroid of the class to the hyperplane is dc, $dp1$ is the distance from the closest pattern of the class to the hyperplane, and $dp2$ is the distance from the second closest pattern of the class to the hyperplane.

The equation that we have proposed considers that some classes may be very far from the hyperplanes in the majority of patterns. That is to say that their centroid is far, but they can have patterns closer to the hyperplane in comparison with other classes, as we can see in the star shapes in Figure 4. It is also possible to have classes with a near to hyperplane centroid, but with some elements so far away.

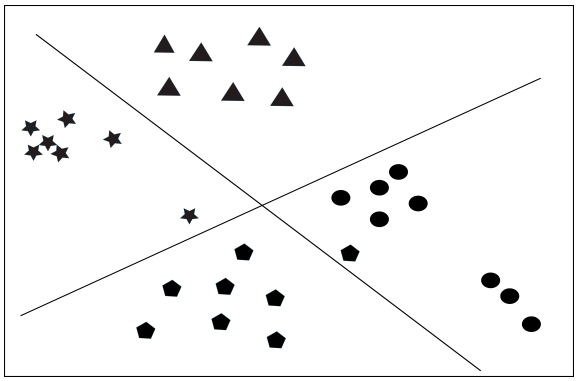

Figure 4: Hyperplanes in an artificial neural network.

To calculate the distance of the patterns to the hyperplanes, and the distances of the centroids of the classes to the hyperplane the following formula is used:

$$D = \frac{|b - (w_1x_1 + w_2x_2, ..., w_nx_n)|}{\sqrt{w_1^2 + w_2^2, ..., w_n^2}} \tag{11}$$

where D is the distance to hyperplane, b is the independent term or bias, the weights that make up the hyperplane are w_n, and x_n are the features of the pattern of the TS.

3. We adjust of the weights using the Support Vector Machines for two linearly separable classes describe in Section 2.2. We have considered finding the two closest classes to each hyperplane because in this work SVM is used as though these classes were linearly separable. To apply SVM, we must find as many support vectors as attributes have the patterns plus 1, the addition of this latter represents the independent term or bias. The latter generates a set of equations from which we obtain the Lagrange multipliers as shown in Section 2.2.

For linearly separable classes SVM has solution. It is for this reason that with the adjustment of weights with SVM, and the edited sample we try to transform the multiclass problem into linearly separable classes in retraining, except for the class that is inserted Therefore, the procedure seeks in a certain way to ensure convergence simulating a problem of linearly separable classes.

4 Experiments and Results

The experiments were made by applying the Incremental Learning Algorithm presented in sections 3 and 3.1. For executing the experiments, we employ the real datasets of UCI [1] shown in Table 1. The whole sample is divided into 10 trials, and each of these is divided into a training set and test set. The generalization power of the ANN was calculated using 10-fold cross-validation. This procedure is described in Section 4.1.

Datasets	Classes	Features	Patterns
Iris	3	4	147
Image	7	18	397
Glass	6	9	213
Vehicle	4	18	846
Vowel	11	10	528
Waveform20	3	20	4999
Waveform40	3	40	4999
Wine	3	13	178

Table 1: Description of the datasets used in the experiments.

Table 2 shows the partially exposed samples, one class was removed from each dataset as mentioned in Section 3 of the description of the algorithm.

Datasets	Classes	Features	Patterns
Iris	2	4	98
Image	6	18	303
Glass	5	9	184
Vehicle	3	18	647
Vowel	10	10	480
Waveform20	2	20	3303
Waveform40	2	40	3344
Wine	2	13	135

Table 2: Description of the partially exposed samples.

Table 3 shows the stopping criteria (error minimal and the epochs quantity), and the number of nodes in the hidden layer, for each training sample. For the training of source network and target network, we used the Backpropagation algorithm (Section 2.1) with learning rate of 0.7, and a momentum equal 0.9.

We compare our results with standard training. Similarly, as other authors did to validate their technique [13, 19, 2] comparing with a standard training. In Table 4, we show the results of the experiments with the different ANN for each dataset. In column 2, we present the generalization power of networks when the training was realized with all classes, that is to say,

Datasets	Nodes	Error	Epochs
Iris	4	0.0046	598
Image	7	0.00123	278
Glass	6	0.027	150
Vehicle	4	0.034	287
Vowel	11	0.019	145
Waveform20	13	0.053	277
Waveform40	6	0.0449	241
Wine	4	0.0005	190

Table 3: Stop criteria for the training.

an exhaustive training. We have called it standard training. Column 3 shows the generalization power of the source network with the sample partially exposed, without a class. Finally, in column 4, we present the generalization power, that was obtained with the incremental learning algorithm proposed in this work, in which a non-exhaustive re-training of the target network has been done.

Datasets	Standard training	Training source network	Non-exhaustive retraining target network ILA
Iris	96%	100%	96.70%
Image	89.90%	95.40%	91.20%
Glass	43.90%	44%	47.50%
Vehicle	70.70%	65.80%	66.10%
Vowel	45.70%	51.10%	53.30%
Waveform20	86.30%	88.80%	86.70%
Waveform40	81.70%	82.70%	82.30%
Wine	98.60%	97%	96.50%

Table 4: Generalization power of ANN.

4.1 Approximate Statistical Test for Comparing the Algorithms

We performed an approximate statistical test to compare the performance of our algorithm of Incremental Learning and the standard training explained in previous section. To validate our approach,we used the K-fold cross-validated paired t test, which consists of randomly dividing the sample of data into K disjoint sets of equal size, $T_1, .., T_k$. We then conduct K trials. In each trial i, the test is T_i, and the training set is the union of the other $T_j, j \neq i$ [5]. We used 10-fold cross-validation, and at the end of each run, we have obtained the average over the 10 folds [2]. For each of the defined fold, we obtain a difference of the classification errors of each one of the algorithms to evaluate. It is $p^{(i)} = p_A^{(i)} - p_B^{(i)}$ were drawn independently from normal distribution, then we can apply Student's t test, by computing the statistic

$$t = \frac{\bar{p} \cdot \sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n} \left(p^{(i)} - \bar{p}\right)^2}{n-1}}} \tag{12}$$

where $\bar{p} = \frac{1}{n} \sum_{i=1}^{n} p^{(i)}$ .Under the null hypothesis, this statistic has a t distribution with

$n-1$ degrees of freedom [5]. For 10 trials, the null hypothesis can be rejected if $|t| > t_{9,0.975} = 2.2622157$. The results are show in Table 5.

K-fold cross-validated paired t test	
Datasets	t score
	Standard training vs ILA
Iris	1 <2.2622157
Image	0.3005 <2.2622157
Glass	0.9153<2.2622157
Vehicle	2.8494 >2.2622157
Vowel	1.2483 <2.2622157
Waveform20	0.9934 <2.2622157
Waveform40	0.9577 <2.2622157
Wine	0.8032 <2.2622157

Table 5: Results of K-fold validated paired t test.

For a confidence level of 95%, and 9 degrees of freedom, in most cases the null hypothesis is accepted, since there is no significant difference between the algorithms compared, both have the same performance. However, this is different for the Vehicle dataset where there is a significant difference between the algorithms compared, the algorithms have different performance, the standard training is better.

5 Conclusions

We have presented Implementation of Incremental Learning in Artificial Neural Networks, a novel Incremental Learning Algorithm for the Multilayer Perceptron. In the majority of the samples used in the experiments, the ILA here proposed presents the same performance as the standard training with all classes. The main advantage of our proposal is to add, to the Multilayer Perceptron neural network, new knowledge without forgetting what has already been learned. In addition, it reduces the time of training since it employs a sample of training with a smaller amount of elements and a smaller amount of epochs.

The ILA provides the neural network with the ability to learn a new class without forgetting the classes already learned.

6 Acknowledgments

This work was partially supported for Consejo Nacional de Ciencia y Tecnología (CONACyT) with number of CVU 740807.

References

[1] UCI Machine Learning Repository, 2017. https://archive.ics.uci.edu/ml/index.php/.

[2] Danielle Azar, Karl Fayad, and Charbel Daoud. A combined ant colony optimization and simulated annealing algorithm to assess stability and fault-proneness of classes based on internal software quality attributes. *International Journal of Artificial Intelligence*, 14(2):137–156, 2016.

[3] L. Bruzzone and D. Fernández Prieto. Incremental-learning neural network for the classification of remote-sensing images. *Pattern Recognition Letters*, 20(11-13):1241–1248, 1999.

[4] Christopher J C Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[5] Tomas G Dietterich. Approximate Statistical Test for Comparing Supervised Classification Learning Algorithms. *Neural comput*, 10(7):1895–1923, 1998.

[6] A. Engelbrecht and R. Brits. A clustering approach to incremental learning for feedforward neural networks. In *Neural Networks, Proceedings*, 2001.

[7] S Grossberg. Nonlinear neural networks: principles,mechanisms, and architectures. *Neural Networks*, 1:17–61, 1988.

[8] Steven Gutstein, Olac Fuentes, and Eric Freudenthal. Knowledge Transfer in Deep Convolutional Neural Nets. *International Journal on Artificial Intelligence Tools*, 17(03):555, 2008.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *NIPS 2014 Deep Learning Workshop*, pages 1–9, 2015.

[10] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *Journal IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[11] Estanislau Lima, Xin Sun, Junyu Dong, Hui Wang, Yuting Yang, and Lipeng Liu. Learning and Transferring Convolutional Neural Network Knowledge to Ocean Front Recognition. *IEEE Geoscience and Remote Sensing Letters*, 14(3):354–358, 2017.

[12] Xueyi Liu, Chuanhou Gao, and Ping Li. A comparative analysis of support vector machines and extreme learning machines. *Neural Networks*, 33:58–66, 2012.

[13] Peio Loubière, Astrid Jourdan, Patrick Siarry, and Rachid Chelouah. A modified sensitivity analysis method for driving a multidimensional search in the Artificial Bee Colony algorithm. *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, 41:1453–1460, 2016.

[14] Beatriz Pérez Sánchez, Oscar Fontenla Romero, and Bertha Guijarro Berdiñas. An incremental learning method for neural networks in adaptive environments. *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[15] Beatriz Pérez Sánchez, Oscar Fontenla Romero, Bertha Guijarro Berdiñas, and David Martínez Rego. An online learning algorithm for adaptable topologies of neural networks. *Expert Systems with Applications*, 40(18):7294–7304, 2013.

[16] Robi Polikar, Lalita Udpa, Satish S. Udpa, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 31(4):497–508, 2001.

[17] Udpa Satish Polikar, Robi.Udpa Lalita and Vasant Honavar. An Incremental Learning Algorithm with Confidence Estimation for Automated Identification of NDE Signals. *IEEE Transactions on ultrasonics, ferroelectrics, and frequency control*, 51(8):990–1001, 2004.

[18] L Y Pratt. Discriminability-Based Transfer between Neural Networks. *Advances in Neural Information Processing Systems 5*, pages 204–211, 1993.

[19] Radu Emil Precup, Marius Csaba Sabau, and Emil M. Petriu. Nature-inspired optimal tuning of input membership functions of Takagi-Sugeno-Kang fuzzy models for Anti-lock Braking Systems. *Applied Soft Computing Journal*, 27:575–589, 2015.

[20] Gonzalo Ramos Jiménez, Rafael Bueno Morales, and Jóse Avila Campo. IADEM-0 : Un Nuevo Algoritmo Incremental. *Tendencias de la Minería de Datos en España*, pages 91–98, 2004.

[21] Sheng Wan and Larry E Banta. Parameter Incremental Learning Algorithm for Neural Networks. *IEEE Transactions on Neural Networks*, 17(6):1424 – 1438, 2006.

[22] Dennis L. Wilson. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man and Cybernetics*, 2(3):408–421, 1972.

[23] Ying-Chun Zhang, Guo-Sheng Hu, Feng-Feng Zhu, and Jin-Lian Yu. A New Incremental Learning

Support Vector Machine. *2009 International Conference on Artificial Intelligence and Computational Intelligence*, 1:7–10, 2009.

[24] M Zribi and Y Boujelbene. The Neural Networks with an Incremental Learning Algorithm Approach for Mass Classification in Breast Cancer. *Biomedical Data Mining*, 5(118):2, 2016.