Hybrid Methods Using Evolutionary Algorithms for On–line Training

G.D. Magoulas[1,3], V.P. Plagianakos[2,3], and M.N. Vrahatis[2,3]

[1]Department of Information Systems and Computing, Brunel University,
Uxbridge UB8 3PH, UK. e–mail: `George.Magoulas@brunel.ac.uk`

[2]Department of Mathematics, University of Patras,
GR–261.10 Patras, Greece. e–mail: {`vpp,vrahatis`}`@math.upatras.gr`

[3]University of Patras Artificial Intelligence Research Center–UPAIRC.

Abstract

A novel hybrid evolutionary approach is presented in this paper for improving the performance of neural network classifiers in slowly varying environments. For this purpose, we investigate a coupling of Differential Evolution Strategy and Stochastic Gradient Descent, using both the global search capabilities of Evolutionary Strategies and the effectiveness of on–line gradient descent. The use of Differential Evolution Strategy is related to the concept of evolution of a number of individuals from generation to generation and that of on–line gradient descent to the concept of adaptation to the environment by learning. The hybrid algorithm is tested in two real-life image processing applications. Experimental results suggest that the hybrid strategy is capable to train on–line effectively leading to networks with increased generalization capability.

1 Introduction

Learning in Artificial neural Networks (ANNs) is usually achieved by minimizing the network's error, which is a measure of its performance, and is defined as the difference between the actual output vector of the network and the desired one. This approach is very popular for training ANNs and includes training algorithms that can be divided in two categories: batch, also called off–line, and stochastic, also called on–line.

The batch training of ANNs is considered as the classical machine learning approach: a set of examples is used for learning a good approximating function, i.e. train the ANN, before the network is used in the application. Batch training is consistent with the theory of unconstrained optimization and can be viewed as the minimization of the function E; that is to find a set of weights $w^* = (w_1^*, w_2^*, \ldots, w_n^*) \in \mathbb{R}^n$, such that:

$$w^* = \min_{w \in \mathbb{R}^n} E(w), \qquad (1)$$

where E is the batch error measure defined as the sum–of–squared–differences error function over the entire training set.

The rapid computation of such a minimizer is a rather difficult task since, in general, the dimension of parameter space is high and the error function generates a complicated surface in this space, possessing multitudes of local minima and having broad flat regions adjoined to narrow steep ones that need to be searched to locate an "optimal" weight set.

On the other hand, in on–line training, the function E is pattern-based and is defined as the instantaneous squared–differences error function with respect to the currently presented training pattern. In this case, the ANN weights are updated after the presentation of each training example, which may be sampled with or without repetition. On–line training may be the appropriate choice for learning a task, either because of the very large (or even redundant) training set, or because of the slowly time–varying nature of the task. Moreover, it helps escaping local minima and provides a more natural approach for learning time varying functions and continuously adapt in a changing environment. As Sutton pointed out, [24], "on–line learning is essential if we want to obtain learning systems as opposed to merely

learned ones".

In practice, on–line methods seem to be more robust than batch methods as errors, omissions or redundant data in the training set can be corrected or ejected during the training phase. Additionally, training data can often be generated easily and in great quantities when the system is in operation, whereas they are usually scarce and precious before. Furthermore, on–line training, and/or on–line retraining, of ANNs is very important in many real–time reactive environments. For example, when we require to control the steering direction of a autonomous vehicle system under various road conditions [5], or recognize, detect and extract objects in images and video sequences under variable perceptual conditions (shading, shadows, lighting conditions, and reflections) [4, 7, 13, 14, 27].

Despite the abundance of methods for learning from examples, there are only few that can be used effectively for on–line learning. For example, the classic batch training algorithms cannot straightforwardly handle nonstationary data. Even when some of them are used in on–line training there exists the problem of "catastrophic interference", in which training on new examples interferes excessively with previously learned examples leading to saturation and slow convergence [25]. Note that in this context it is not possible to use advanced optimization methods, such as conjugate gradient, variable metric, simulated annealing etc., as these methods rely on a fixed error surface [19]. Consequently, given the inherent efficiency of stochastic gradient descent, various schemes have been recently proposed based on this idea, [2, 19, 21, 22, 24]. However, these schemes suffer from several drawbacks, such as sensitivity to learning parameters [19].

This paper proposes a new Hybrid Evolutionary Algorithm (HEA) for on–line training. The HEA could conceptually be split-up into two stages. In the first stage, on–line training is adopted using a recently proposed stochastic gradient descent with adaptive stepsize [11]. In the second stage, a Differential Evolution (DE) Strategy, [23], is used for on–line retraining. The usage of DE Strategy is based on the assumption that the first stage has produced a "good" solution that can be incorporated directly into the genes and inherited by offspring.

The rest of the paper is organized as follows: the use of Evolutionary Algorithms in ANNs training is discussed in Section 2. In Section 3, the new hybrid on–line training algorithm is introduced. Section 4 presents details on the application of the hybrid method in training on–line ANNs for real-life image recognition problems and outlines the implementation results. Finally, in Section 5, conclusions and a short discussion of future work are presented.

2 Evolutionary Algorithms in ANN Training

Evolutionary Algorithms (EAs) are stochastic search methods that mimic the metaphor of natural biological evolution. They operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics [3]. Many attempts have been made within the artificial intelligence community to integrate EAs and ANNs. A number of attempts has concentrated on applying evolutionary principles to improve the generalization of ANNs, discover the appropriate network topology, and the best available set of weights (see [18]).

The majority of approaches in which evolutionary principles are used in conjunction with ANN training formulates the problem of finding the weights of a fixed neural architecture, when the whole set of examples is available, as an optimization problem. EAs are global search methods and, thus, less susceptible to local minima [16]. Nevertheless, EAs remain, in certain cases, more computationally expensive than training by a variant of the backpropagation method [17]. This is one of the reasons that narrow the applicability of EAs to off–line ANN training.

In previous work, we demonstrated the efficiency of a special class of EAs, called *Differential Evolution* (DE) strategies, [12, 23], in off–line training [15]. DE strategies can handle non differentiable, nonlinear and multimodal objective functions efficiently, and require few easily chosen control parameters. Experimental results have shown that DE strategies have good convergence properties and outperform other evolutionary algorithms [15].

To apply DE strategies to ANN training we start with a specific number (NP) of n–dimensional weight vectors, as an initial weight population, and evolve them over time; NP is fixed throughout the training process and the weight population is initialized randomly following a uniform probability distribution. At each iteration, called *generation*, new weight vectors are generated by the combination of weight vectors randomly chosen

from the population. This operation is called *mutation*. The outcoming weight vectors are then mixed with another predetermined weight vector – the target vector – and this operation is called *crossover*. This operation yields the so–called trial vector. The trial vector is accepted for the next generation if and only if it reduces the value of the error function E. This last operation is called *selection*. The above mentioned operations introduce diversity in the population and are used to help the algorithm escape the local minima in the weight space. The combined action of mutation and crossover is responsible for much of the effectiveness of DE's search, and allows them to act as parallel, noise–tolerant hill–climbing algorithms, which efficiently search the whole weight space.

As in this work we focus on the on–line training and retraining of ANNs, we adopt a formulation of this problem which is based on tracking the changing location of the minimum of a pattern-based, and, thus, dynamically changing, error function. This approach coincides with the way adaptation in the evolutionary time scale is considered [20], and allows us to explore and expand further research on the tracking performance of evolution strategies and genetic algorithms [1, 20, 26].

3 The Hybrid Evolutionary Algorithm

In this section, we present a Lamarck–inspired combination of Differential Evolution strategy and Stochastic Gradient Descent (SGD). The DE strategy works on the termination point of the SGD. Thus, the method consists of a SGD–based on–line training stage and an Evolutionary strategy–based on–line retraining stage.

A generic description of the proposed hybrid algorithm, is given in Algorithm 1. First, the SGD is outlined in the Stage 1 of Algorithm 1, where η is the stepsize, K is the meta–stepsize and $\langle \cdot, \cdot \rangle$ stands for the usual inner product in $\mathbb{R}^n$. The memory–based calculation of the stepsize, in Step 4a, takes into consideration previously computed pieces of information to adapt the stepsize for the next pattern presentation. This provides some kind of stabilization in the calculated values of the stepsize, and helps the stochastic gradient descent to exhibit fast convergence and high success rate. Note that the classification error, an upper limit to the error function evaluations, or a pattern-based error measure can be used as the termination condition in Step 5a. The key features of the SGD method are the low storage requirements and the inexpensive computations. Moreover, in order to calculate the stepsize to be used at the next iteration, this on–line algorithm

uses information from the current, as well as the previous iteration.

In Stage 2 of Algorithm 1, the DE strategy, responsible for the on–line retraining is outlined. Steps 3b and 4b implement the mutation and crossover operators, respectively, while Step 5b is the selection operator.

The first DE operator used is the mutation operator. Specifically, for each weight vector w_i^p, a new vector called mutant vector is generated according to the following relation:

$$\text{Mutant\_Vector} = w_i^p + \xi(w_{best} - w_i^p) + \xi(w_{r_1} - w_{r_2}),$$

where w_{best} is the best member of the previous generation, $\xi > 0$ is a real parameter called mutation constant and controls the amplification of the difference between two weight vectors, and w_{r_1} and w_{r_2} are two randomly chosen weight vectors, different from w_i^p.

To increase further the diversity of the mutant weight vector, the crossover operator is applied. Specifically, for each component $j, (j = 1, 2, \ldots, n)$, of the mutant weight vector, we randomly choose a real number r from the interval $[0, 1]$. Then, we compare this number with $\rho > 0$ (crossover constant), and if $r \leqslant \rho$ we select, as the j-th component of the trial vector, the corresponding component j of the mutant vector. Otherwise, we pick the j-th component of the target vector.

4 Experiments and Results

We have tested the proposed hybrid algorithm in two real–life classification tasks. The first experiment concerns training on–line an ANN classifier to discriminate among 12 texture images, and the second one, training an ANN to detect suspicious regions in colonoscopic video sequences. In all cases, no operation for tuning the mutation and crossover constants was carried out; default fixed values $\xi = 0.5$ and $\rho = 0.7$ have been used.

4.1 The texture classification problem
A total of 12 Brodatz texture images of size 512×512, [6], as shown in Figure 1, was acquired by a scanner at 150dpi. From each texture image, 10 subimages of size 128×128 were randomly selected, and the co–occurrence method, [8], was applied. In the co–occurrence method, the relative frequencies of gray–level pairs of pixels at certain relative displacements are computed and stored in a matrix. The combination of the nearest neighbor pairs at orientations 0º, 45º, 90º and 135º is used in the experiment. A set

Stage 1 - "Learning"	
Step 0a:	Initialize the weights w^0, η^0 and the meta–stepsize K.
Step 1a:	**Repeat** for each pattern p.
Step 2a:	Calculate $E(w^p)$ and then $\nabla E(w^p)$.
Step 3a:	Update the weights: $$w^{p+1} = w^p - \eta^p \nabla E(w^p).$$
Step 4a:	Calculate the stepsize to be used with the next pattern $p+1$: $$\eta^{p+1} = \eta^p + K \left\langle \nabla E(w^{p-1}), \nabla E(w^p) \right\rangle.$$
Step 5a:	**Until** the termination condition is met.
Step 6a:	**Return** the final weights w^{p+1} to the Stage 2.
Stage 2 - "Evolution"	
Step 0b:	Initialize the DE population in the neighborhood of w^{p+1}.
Step 1b:	**Repeat** for each input pattern p.
Step 2b:	**For** $i = 1$ to NP
Step 3b:	MUTATION(w_i^p) $\rightarrow$ Mutant_Vector.
Step 4b:	CROSSOVER(Mutant_Vector) $\rightarrow$ Trial_Vector.
Step 5b:	**If** E(Trial_Vector) $\leqslant E(w_i^p)$, accept Trial_Vector for the next generation.
Step 6b:	**EndFor**
Step 7b:	**Until** the termination condition is met.

Algorithm 1: Generic Model of the Hybrid On–line Training Algorithm

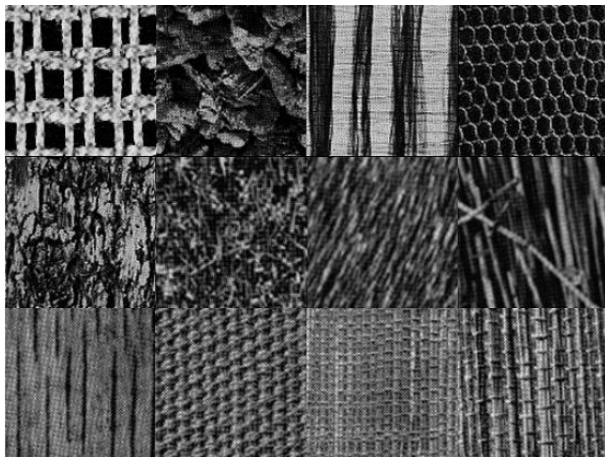of 10 sixteenth–dimensional training patterns was extracted from each image.

Figure 1: The twelve texture images.

A 16–8–12 ANN (224 weights, 20 biases) was trained on–line to classify the patterns into the 12 texture types. The network used neurons of logistic activations with biases, and the weights and biases were initialized with random numbers from the interval $(-1, 1)$. The termination condition for the first stage was a classification error $CE \leqslant 3\%$. Then, the second stage was executed for on–line retaining using new patterns from the training set. At the end, the generalization capability of the trained network was tested on the test set, which consisted of 320 patterns (20 patterns from each image extracted from randomly selected subimages). The ANN correctly classified 304 out of 320 patterns. Thus, it exhibited 95% generalization success. In the same task, the performance of the SGD alone, i.e. without using the evolution stage of the algorithm for the on–line retraining, was 93%, while the performance of the batch backpropagation algorithm with variable stepsize, [10], was 90%.

4.2 Abnormalities detection by colonoscopy

Colonoscopy is a minimal invasive technique for the production of medical images. A narrow pipe like structure, an endoscope, is passed into the patient's body. Video endoscopes have small cameras in their tips. When passed into a body, what the camera observes is displayed on a television monitor (see Figure 2 for a sample frame of the video sequence). The physician controls the endoscope's direction using wheels and buttons. An important stage of the implementation is the feature extraction process [9]. In our experiments we have used the co–occurrence matrices to generate features. More specifically, each frame of the endoscopic video sequence was separated into windows of size 16 pixels by 16 pixels. Then, the co–occurrence matrices algorithm was used to gather information regarding each pixel in an image window, and to generate feature vectors that contain sixteen elements each.

A 16–30–2 ANN (540 weights, 32 biases) with logistic

activations was trained on–line to discriminate between normal and suspicious image regions using 300 randomly selected patterns from the first frame. On–line training stopped when the ANN exhibited 3% misclassifications on the training set. It must be noted that the first stage was extremely fast; approximately 40 training epochs were needed. For on–line retraining, the DE population has been initialized with weight vectors in the neighborhood of the weight vector obtained from the first stage. In order to test the tracking performance of the hybrid algorithm, we introduced in the training set patterns from other frames of the same video sequence, which exhibited resolution change, different perceptual direction of the physician, different diffused light conditions. Thus, on–line retraining was performed using a training set of 1200 patterns. The DE algorithm was allowed to perform only two iteration with each pattern. This was necessary to prevent the "catastrophic interference" among patterns of different frames. To test the performance of the trained ANN approximately 4000 patterns have been extracted from each frame. The 4000 patterns cover the whole image region of a frame and contain normal and suspicious samples.
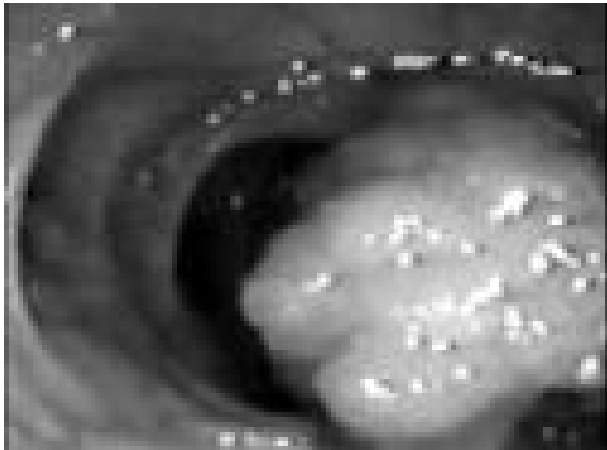

Figure 2: Frame of the colonoscopic video sequence.

The generalization results with and without the evolution stage of the algorithm, are exhibited in Table 1. The first column of Table 1 exhibits results from training a special ANN for each frame and, then, testing it using data from the same frame without on–line retraining. From example, let us observe Frame 1. The corresponding ANN was trained using data extracted from Frame 1 and achieved a recognition success of 83.77% when tested with the whole frame. Similarly, a percentage of success of 87.60% was achieved by the ANN that was trained on Frame 4. On the other hand, the hybrid method by applying on–line retraining man-

aged to locate weights that are eminently suitable for all of the four frames. Thus, the ANN trained with the hybrid method provides higher percentage of generalization in all cases when compared with the four specially trained ANNs.

	Without Evolution	With Evolution
Frame 1	83.77%	91.91%
Frame 2	77.18%	83.57%
Frame 3	82.84%	93.09%
Frame 4	87.60%	89.24%

Table 1: Results from the interpretation of images.

5 Conclusions

A new hybrid method for on–line neural network training has been developed, tested and applied to a texture classification problem and a tumor detection problem in colonoscopic video sequences. Simulation results suggest that the new method exhibits fast and stable learning, good generalization and therefore a great possibility of good performance. The proposed algorithm is able to train large networks on–line, and seems better suited for tasks with large, redundant or slowly time–varying training sets, such as those of image analysis. Further work is needed to optimize the hybrid algorithm performance, as well as to test it on even bigger training sets.

References

[1] P. Angeline, "Tracking extrema in dynamic environments". In: *Proc. of the Sixth Annual conference on Evolutionary Programming VI*, 335-345, Springer, (1997).

[2] L.B. Almeida, T. Langlois, J.D. Amaral, and A. Plankhov, "Parameter adaptation in Stochastic Optimization". In: *On–line Learning in Neural Networks*, D. Saad, (ed.), 111–134, Cambridge University Press, (1998).

[3] T. Bäck and H.P. Schwefel, "An overview of evolutionary algorithms for parameter optimization", *Evolutionary Computation*, **1**, 1–23, (1993).

[4] S.W. Baik and P. Pachowicz, "Adaptive object recognition based on the radial basis function paradigm". In:*Proc. of the IEEE Int. Joint Conf. on Neural Networks (IJCNN'99)*, Washington, U.S.A., (1999). CD-ROM Proceedings, Paper No.215, Session 9.4.

[5] S. Baluja, "Evolution of an artificial neural network based autonomous land vehicle controller", *IEEE Transactions on System, Man and Cybernetics-Part B*, **26**, 450–463, (1996).

[6] P. Brodatz, *Textures – a Photographic Album for Artists and Designers*. New York, Dover, (1966).

[7] A.D. Doulamis, N.D. Doulamis, S.D. Kollias, "On–line retrainable neural networks: improving the performance of neural networks in image analysis problems", *IEEE Transactions on Neural Networks*, **11**, 137–155, (2000).

[8] RM. Haralick, "Statistical and structural approaches to texture". *Proc. IEEE*, **67**, 786–804, (1979).

[9] S. Karkanis, G.D. Magoulas, and N. Theofanous, "Image recognition and neuronal networks: Intelligent systems for the improvement of imaging information", *Minimal Invasive Therapy & Allied Technologies*, **9**, 225–230, (2000).

[10] G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis, "Effective back-propagation training with variable stepsize", *Neural Networks*, **10**, 69-82, (1997).

[11] G.D. Magoulas, V.P. Plagianakos, and M.N. Vrahatis, "Adaptive stepsize algorithms for on–line training of neural networks", *Nonlinear Analysis: Theory, Methods and Applications*, in press, (2001).

[12] Z. Michalewicz and D.B. Fogel, *How to solve it: Modern heuristics*, Springer, 2000.

[13] P.W. Pachowicz and S.W. Baik, "Adaptive RBF classifier for object recognition in images sequences". In:*Proc. of the IEEE Int. Joint Conf. on Neural Networks (IJCNN'2000)*, Como, Italy, vol. VI-600, (2000).

[14] N.G. Panagiotidis, D. Kalogeras, S.D. Kollias, and A. Stafylopatis, "Neural network-assisted effective lossy compression of medical images", *Proc. IEEE*, **84**, 1474–1487, (1996).

[15] V.P. Plagianakos and M.N. Vrahatis, "Training Neural Networks with Threshold Activation Functions and Constrained Integer Weights". In: *Proc.*

of the IEEE Int. Joint Conf. on Neural Networks (IJCNN'2000), Italy, (2000).

[16] V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis, "Learning in multilayer perceptrons using global optimization strategies", *Nonlinear Analysis: Theory, Methods and Applications*, in press, (2001).

[17] V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis, "Improved learning of neural nets through global search". In: *Global Optimization - Selected Case Studies*, J.D. Pintér (ed.), Kluwer Academic Publishers, to appear, (2001).

[18] J.C.F. Pujol, and R. Poli, "Evolving the topology and the weights of neural networks using a dualreprsentation", *Applied Intelligence*, **8**, 73–84, (1998).

[19] D. Saad, *On–line learning in neural networks*, Cambridge University Press, (1998).

[20] R. Salomon and P. Eggenberger, "Adaptation on the evolutionary time scale: a working hypothesis and basic experiments". In:*Proc. of the Third European Conference on Artificial Evolution (AE'97)*, Nimes, France, Lecture Notes in Computer Science vol. 1363, Springer, (1998).

[21] N.N. Schraudolph, "Online local gain adaptation for multi–layer perceptrons". Technical Report, IDSIA–09–98, IDSIA, Lugano, Switzerland, (1998).

[22] N.N. Schraudolph, "Local gain adaptation in stochastic gradient descend", Technical Report, IDSIA–09–99, IDSIA, Lugano, Switzerland, (1999).

[23] R. Storn and K. Price, "Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, **11**, 341–359, (1997).

[24] R.S. Sutton, "Adapting bias by gradient descent: an incremental version of delta–bar–delta". In: *Proc. of the Tenth National Conference on Artificial Intelligence*, MIT Press, 171–176, (1992).

[25] R.S. Sutton and S.D. Whitehead, "Online learning with random representations". In: *Proc. of the Tenth International Conference on Machine Learning*, Morgan Kaufmann, 314–321, (1993).

[26] F. Vavak and T.C. Fogarty, "A comparative stady of steady state and generational genetic algorithms". In: *Proceedings of Evolutionary Computing: AISB Workshop*, Lecture Notes in Computer Science vol. 1143, Springer, (1996).

[27] G.G. Wilknson, "Open questions in neurocomputing for earth observation". In: *Proc. of the first COMPARES Workshop*, York, U.K., 1996.