XVIII International Conference on Water Resources
CMWR 2010
J. Carrera (Ed)
CIMNE, Barcelona 2010

ON THE USE OF RANDOM-WALK BASED TRAINING ALGORITHMS FOR NEURAL NETWORKS APPLIED IN ENVIRONMENTAL MODELING

Ioannis C. Trichakis[*], Ioannis K. Nikolos[†] and George P. Karatzas[*]

[*] Department Environmental Engineering, Technical University of Crete
University Campus, 73100, Chania, GREECE
e-mail: ioannis.trichakis@enveng.tuc.gr, karatzas@mred.tuc.gr

[†] Department of Production Engineering and Management, Technical University of Crete
University Campus, 73100, Chania, GREECE,
e-mail: jnikolo@dpem.tuc.gr

Key words: Advanced numerical methods, water resources, model predictions

Summary. Neural networks have found their way in environmental modeling and their use increases with time. The physical systems though are quite complicated and difficult to describe. This may result in poor training when a traditional method like Back-Propagation (BP) or even a more advanced like Conjugate Gradient (CG) is applied. These methods have the advantage that they converge to a minimum after a finite number of iterations, but this minimum could be a local one. In this work, a neural network that simulates the change to an aquifer's level between successive days, using hydrological and meteorological parameters as inputs, is trained using different algorithms in order to evaluate whether the conventional, widely accepted, methods may be trapped in local minima. An alternative training procedure, a random walk (RW) based training algorithm, proposed by Tan and Gu[1] is used as a better methodology to explore the solution hyperspace. The alternative algorithms are tested in two field cases, related to karstic aquifers, where adequate field measurements are available. One must never forget though that algorithms like the random-walk based one may provide better results than BP or CG but on the cost of large computational times. As a second remark these methods' convergence rate depends on the selection of the method's parameters. The use of RW based algorithms could prove to be valuable when a researcher is trying to discover if the widely used methods converge to a subpar local optimum.

1 INTRODUCTION

While many researchers have utilized neural networks to handle environmental problems, most of them use traditionally training methods like back-propagation (BP) or conjugate gradient (CG). In a review paper[2] 40 out of 56 artificial neural network (ANN) implementations used back-propagation as an optimization algorithm, and 3 used the conjugate gradient method. These methods have the advantage that they converge to a minimum after a finite number of iterations. However, in most cases the environmental problems are so complex that a very large number of local minima exist, thus increasing the

possibility the training procedure to be confined in one of them, being far from the global optimum. As Tan and Gu concluded[1] there is a possibility a very simple algorithm, based on random walk, can outperform the traditional methods reducing the neural network error by one or even more orders of magnitude. The problem complexity and the fitness function's many local minima can give global optimization algorithms an advantage over those which get trapped in local optima.

Perhaps the most important advantage that random walk based algorithms have, is that they do not get trapped in local minima. This is very important for complex problems as the fitness function value at a local minimum can be far worse than the respective value at the desired global optimum. These algorithms are great when it comes to exploration of the solution hyperspace. The algorithm tries many different combinations of decision variable values and the more it explores, the better the chances are to find the optimal solution. In addition, these algorithms are elitist, so at every step the algorithm either finds a better solution or does not worsen the already existent. Moreover, these algorithms are simple enough so that every researcher can easily implement and/or modify/fine-tune them. On the other hand, this kind of algorithms is extremely time consuming, and the more the decision variables are, the longer it takes the algorithm to find the solution. The fact that convergence cannot be guaranteed for a finite number of steps means that the researcher has no a priori knowledge of how much time or how many iterations there might be needed in order to reach an acceptable solution. Therefore, the stopping criteria selection can prove to be difficult and is usually an arbitrary decision based on trial-and-error and/or the user's experience.

2 CASE STUDIES

2.1 Attica case

Data collected at the pumping site of Mavrosouvala (in the North West part of Attica, Greece) were used to train and evaluate the ANN. An earlier hydrogeological study indicated that the recharge of the aquifer comes from the mountains in the south and that the water table is rather low, about 100 m below ground surface and 15–20 m above mean sea level[3]. Daily values were available from many stations of the region. One station near the observation wells recorded temperature and rainfall, while two other stations in the greater region also recorded rainfall data. The pumping duration and the pumping rates of the 16 wells were available as well as hydraulic heads from two observation wells (AS1 and N4). An optimization of the ANN's parameters has been conducted in a previous study[4]. The optimized neural network used henceforth, has as input variables, the day number, temperature, rainfall from three stations, pumping from 16 wells and the hydraulic head of the previous day at two observation wells. There were 37 and 39 nodes at the first and the second hidden layer respectively, and the hydraulic head change at the observation wells were the two output nodes.

2.2 Edwards Aquifer

A second field with available data series was the Edwards aquifer in Texas, USA. The

aquifer covers an area approximately 180 miles long and 5 to 40 miles wide and is the primary water source for much of this area, including the City of San Antonio, America's 7th largest city[5]. From a previous study[6] temperature, rainfall from 6 days, cumulative pumping from 4 days, peak pumping and hydraulic head of the previous day were chosen as input parameters of this ANN. The network had two hidden layers (each consisted of 40 nodes) and the output layer with the hydraulic head change of the observation well as a single output node.

3 METHODOLOGY

The wide use of BP and CG can easily be attributed to their convergence properties as well as the relatively small demands of computational power and time. Nevertheless, there are many different training algorithms, with different characteristics and each one of them might be more suitable for a specific problem than another. Back-propagation learning has emerged as the standard algorithm for the training of multilayer perceptrons, against which other learning algorithms are often benchmarked[7]. In many practical applications though, it is considered to be very slow and its training performance is sensitive to the choice of learning rate and initial values of the weight parameters[1].

3.1 Back Propagation (BP)

The back-propagation algorithm derives its name from the fact that the partial derivatives of the cost function (performance measure) with respect to the free parameters (synaptic weights and biases) of the network are determined by backpropagating the error signals (computed by the output neurons) through the network, layer by layer. In so doing, it solves the credit-assignment problem in a most elegant fashion. The computing power of the algorithm lies in its two main attributes:

- Local method for updating the synaptic weights and biases of the multilayer perceptron.
- Efficient method for computing all the partial derivatives of the cost function with respect to these free parameters.

One of the main deficiencies of back propagation is that it can get trapped in local minima[8]. Increase of the number of hidden units is used as a method to overcome this, but even this bypass, it has limitations.

3.2 Conjugate Gradient (CG)

A more advanced algorithm that accelerates the convergence rate and also ensures convergence to a local optimum after a finite number of iterations is the conjugate (or conjugated) gradient method. This algorithm is able to achieve that without the, computationally demanding, calculation of the Hessian, that is required for other advanced algorithms[7]. The method is fast and most of the time it converges to a minimum after a very small number of steps. Besides the advantages of this method, it can also be sensitive to starting point selection, and is far more complicated for a researcher to implement. There are also different methods of calculating the algorithm’s parameters, which influence the convergence rate and in some cases the success of the process[9]. Despite the fact that this

algorithm can be trapped in local minima, its fast convergence is the reason that it has found application in many cases.

3.3 Modified Random Walk (MRW)

The random walk concept is extremely simple. The algorithm randomly searches the solution space and evaluates the fitness function. If the fitness function improves, it keeps the new solution in memory, if not it ignores it and proceeds. It is fairly understandable that if the solution space (or hyperspace) is large and complex, a great amount of time and evaluation is necessary for a fair search. This is the main reason (coupled with lack of convergence guarantee) why RW based algorithms have not found many applications recently. In an attempt to address the disadvantages of RW, Tan and Gu proposed that instead of completely random, the search could be performed for one independent variable at a time and in addition, included a way to exploit the already known solution rather than just searching blindly. At each iteration, the algorithm slightly increases the parameter under optimization and evaluates the fitness function. If this doesn't improve the fitness function then it decreases the parameter by the same small value. In many cases, this increase or decrease of the parameter results in improvement of the fitness function. In case this fails, the algorithm searches randomly for a finite number of times, before it overrides the parameter and continue to the next.

With these modifications the algorithm addresses some of the generic deficiencies of RW algorithms. Firstly, it does exploit the optimal solution of the previous iteration. In doing so it also reduces the need for many random evaluations of the fitness function that would have smaller probability of success. Furthermore, the chance of improving the fitness function by adjusting each decision variable individually is more controllable than when an algorithm changes all of them at once. Also, noteworthy is that the finite number of random iterations provides a way for the algorithm to bypass a decision variable, the change of which does not seem to have a positive effect on the fitness function value. Nevertheless, this algorithm also has some issues that prevent it from being a great alternative of traditional methods, applicable to any problem. Perhaps the most difficult decision the researcher has to make is the selection of the small number the algorithm uses to increase and decrease the decision variables. If the number is very small, the improvement achieved will probably be minimal as well, and so will the optimization rate of the fitness function. On the other hand there is a threshold of this number which when crossed, results in increased failure of the exploitation section of the algorithm, something that would lead to more random iterations and increase computational time.

The algorithm that was applied in this work was further modified. Instead of having one small number, the algorithm firstly checks at ±0.5 for each weight and if unsuccessful it makes six random changes. If all random changes are unsuccessful too and before the weight is skipped and the algorithm proceeds to the next, it also checks at ±0.05 and at ±0.005 for each weight.

4 RESULTS – DISCUSSION

The algorithm was trained for each case study with the three available algorithms and the

training and evaluation error values are shown in Figure 1 and Table 1. The error values from different ANN applications are comparable to each other, since the ANN that was used throughout this work normalizes all input data, in an attempt to avoid high values of synaptic weights. The back propagation (BP) algorithm as well as the modified random walk (RW) were utilized twice; the first time for 1 000 iterations and the second for 10 000. The conjugate gradient (CG) method is quite different from the other two so the numbers of steps are not comparable. The CG can achieve high convergence rates with very little steps, while the other methods require many weight adjustments in order to reach the optimum.

Figure 1: Training and evaluation error of different algorithm in the two regions.

	Attica			**Edwards**		
	Training	*Evaluation*	*Time*	*Training*	*Evaluation*	*Time*
1 000 iterations						
BP	2.66E-04	1.73E-04	0.1 h	6.29E-05	6.49E-05	1 h
RW	1.49E-04	1.03E-04	1 h	3.97E-05	4.33E-05	5.7 h
10 000 iterations						
BP	2.01E-04	1.34E-04	1 h	4.20E-05	4.57E-05	10 h
RW	1.35E-04	1.02E-04	9.4 h	1.78E-05	2.09E-05	56.2 h
100 steps						
CG	2.59E-04	1.68E-04	0.5 h	6.98E-05	7.23E-05	0.3 h

Table 1 : Training and evaluation error and training time of different training algorithms

4.1 Attica case

In the first case study, the modified random walk (RW) seems to reach a better solution than the other two training methods and does so in as few as 1000 steps. This can be attributed to its exploration ability. On the other hand little improvement (9.4%) is achieved between the 1000th iteration and the 10 000th. This seems a little odd as it suggests that some of the weights that were randomly chosen at the beginning and have not yet been optimized have an insignificant effect on the error value. The back propagation drops the error by two

orders of magnitude in the first few steps but, due to the algorithm's construction not being elitist, there are many fluctuations. The improvement of going from 1000 to 10 000 steps is greater (24%) than in the RW case, yet it still is not very impressive considering the significant increase of computational time. Especially for the conjugate gradient algorithm, it can be said that the initial values, which are random, were not close enough to the global minimum. The algorithm had a rapid convergence to a suboptimum solution (after only 28 steps), which it wasn't able to improve. Noteworthy is that during some trial runs the CG algorithm proved to be extremely sensitive to any change at the initial values, and this suggests that a better subroutine for CG variable calculation might give better results. All the training algorithms result smaller evaluation than training error. This was expected because it has been observed that there are more outliers in the training data set.

The improvement of error in RW is accompanied by a great increase (by one order of magnitude in the computational time needed. Nevertheless, 1000 iterations of RW that require the same time as 10 000 iterations of BP result in a better error. In Figure 2, the convergence of the three algorithms is displayed, along with the final results of the algorithm with the smallest error, in this case the RW. The results of the other two calibration algorithms are similar but they are not displayed due to space limitations.

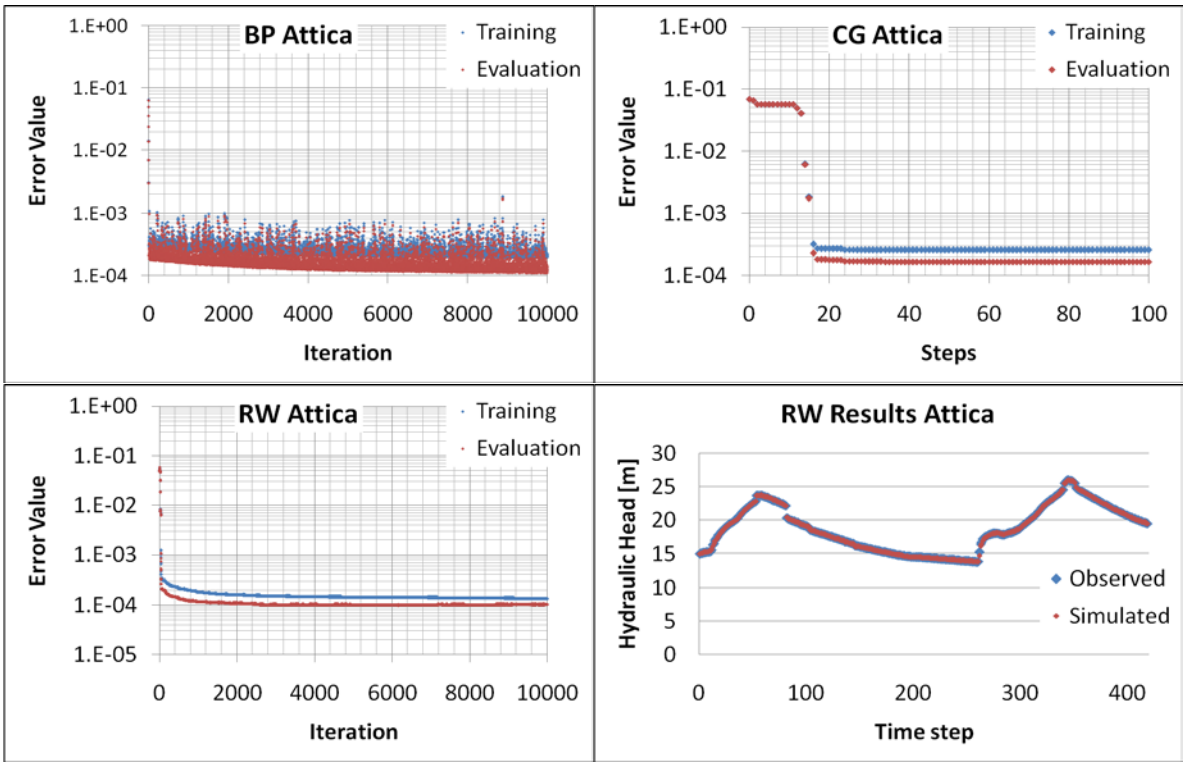

Figure 2: Convergence of the three algorithms and RW results for the Attica case study.

4.2 Edwards Aquifer

Similar to those of the first case study are also the results of the Edwards aquifer. The main

difference here is that in all three cases the evaluation is larger than the training error. This is not surprising since the objective function of the algorithms is the training error.

The random walk based algorithm yields again the smallest error, although this time the improvement of 9000 more adjustments is much greater (55%) than it was in the first case study. The conjugate gradient has once more a rapid convergence to a solution that is much worse than the one provided by the random walk based algorithm. This time it converges at the 19th step. Again the distance of the random initial point from the optimum is probably the reason of the algorithm's limited success. The back propagation fluctuates significantly in this case as well, and the improvement of the 10 000 iterations over the 1000 is 33%.

In this case study, the RW required more time (approximately 6 times) than the BP. But again the results of RW for 1000 iterations, which required 5.7 hours, are better than those of 10 000 BP iterations, which required 10 hours. In Figure 3, the convergence of the three algorithms is displayed, along with the final results of the algorithm with the smallest error, also in this case the RW. The results of the other two calibration algorithms are slightly different and they are not displayed due to space limitations.

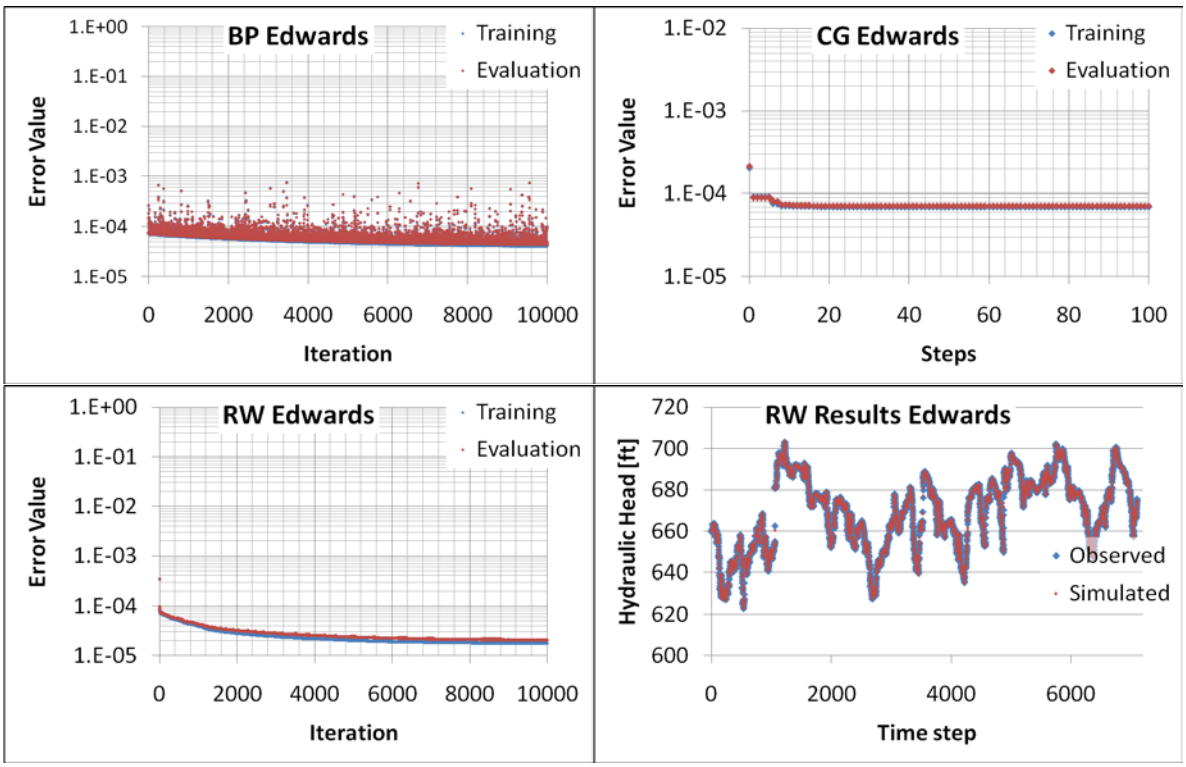

Figure 3: Convergence of the three algorithms and RW results for the Edwards case study.

5 CONCLUSIONS

Simple algorithms, based on random walk, may provide better solutions to the problem of optimizing the neural network weights than widely used algorithms like back propagation. The main reason is that the latter can be trapped in local minima. Especially when the error

function is complex and contains many local optima, these algorithms might yield a subpar solution. On the other hand random walk based algorithms have inherent exploration capabilities which, in most cases, help them discover a better solution. The price for the exploration feature is high computational time and convergence uncertainty. When the ANN is submitted to an optimization procedure and multiple trainings are necessary, high computational time is definitely a drawback that might prevent the researcher from using that specific training algorithm in favor of a faster one. It is only when even a small error improvement is consider important that RW based algorithms can find application in ANN training. However, they can prove a useful tool if the researcher wishes to test the adequacy of algorithms like back propagation to address a certain problem. For example, if a researcher wants to confirm that the solution provided by back propagation is close to the optimal, an easy way is to utilize a RW based algorithm and compare the results.

Further research could utilize a CG-RW hybrid to take advantage of the fast convergence and the global search. It could also consider the use of other stochastic optimization methods, like evolutionary algorithms, as an ANN training algorithm. Though it might not be efficient for ANNs with many nodes and hidden layers due to high computational time, it might be a faster alternative to RW methods.

REFERENCES

[1] S. Tan and J. Gu, *An Efficient Learning Algorithm for Feedforward Neural Network*, Advances in Artificial Intelligence – IBERAMIA 2004, Springer Berlin/Heidelberg, (2004).

[2] H.R. Maier and G.C. Dandy, "Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications", *Environmental Modelling and Software*, **15**, 101-124 (2000).

[3] G. Demopoulos, *Hydrogeological Study of the Mavrosouvala Area in Attica (in Greek)*, (2000).

[4] I.C. Trichakis, I.K. Nikolos and G.P. Karatzas, "Optimal selection of artificial neural network parameters for the prediction of a karstic aquifer's response", *Hydrological Processes*, **23**, 2956-2969 (2009).

[5] G.M. Schindel, J.M. Hamilton, S. Johnson, J. Mireles, R. Esquilin, C. Burgoon, G. Luevano, D. Gregory, R. Gloyd, J. Sterzenback and R. Mendoza, *Hydrologic Data Report*, Edwards Aquifer Authority, (2007).

[6] I.C. Trichakis, I.K. Nikolos and G.P. Karatzas, "Simulation of a Karstic Groundwater System Using Artificial Neural Networks (ANNs) - The Example of Edwards Aquifer", *European Water Resources Association's (EWRA) 7th International Conference*, Limassol, Cyprus, 25 - 27 June (2009).

[7] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, (1999).

[8] B. Kroese and P. van der Smagt, *An introduction to Neural Networks*, University of Amsterdam, (1996).

[9] J.R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, School of Computer Science, Carnegie Mellon University, (1994).