Dynamic Search Trajectory Methods for Neural Network Training*

Y.G. Petalas[1,2], D.K. Tasoulis[1,2], M.N. Vrahatis[1,2]

[1] Department of Mathematics, University of Patras, GR–26110 Patras, Greece
{petalas,dtas,vrahatis}@math.upatras.gr
[2] University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR–26110 Patras, Greece

Abstract. Training multilayer feedforward neural networks corresponds to the global minimization of the network error function. To address this problem we utilize the Snyman and Fatti [1] approach by considering a system of second order differential equations of the form, $\ddot{x} = -\nabla E(x)$, where x is the vector of network weights and ∇E is the gradient of the network error function E. Equilibrium points of the above system of differential equations correspond to optimizers of the network error function. The proposed approach is described and experimental results are discussed.

Keywords: Trajectory Methods, Neural Networks Training, Ordinary Differential Equations

1 Introduction

Training multilayer feedforward neural networks is equivalent to the global minimization of the network error function with respect to the weights of the network. *Trajectory* methods constitute a class of global optimization methods [2]. An important property of these methods is that the generated trajectory passes through the neighborhood of many of the stationary points of the objective function. The simplest trajectory method is described by means of the following initial value problem:

$$\dot{x} = -\nabla E(x), \quad x(0) = x_0. \tag{1}$$

Trajectory methods can also be viewed, through a mechanical analogy, as an assignment of mass to a particle moving in a field of forces [3–5]. Let the mass of the particle be $m(t)$. Further, assume that the particle is moving in a field of forces defined by the potential E, subject to the dissipative force $-n(t)\dot{x}(t)$. Then the trajectory generated by the motion of the particle can be described by the following system of differential equations:

$$m(t)\,\ddot{x}(t) - n(t)\,\dot{x}(t) = -\nabla E(x(t)), \tag{2}$$

* We thank the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and particulary the Program PYTHAGORAS, for funding the above work.

where $m(t) \geqslant 0$ and $n(t) \leqslant 0$. Under certain conditions, the trajectory can determine a local minimum of the objective function E. A similar approach has been proposed in [6], where the motion of a particle with unit mass is described in a field with potential E without friction. We would like to note at this point that since the system is autonomous an interesting theorem due to Poincaré, which states that the trajectory passes through the neighborhood of all the stationary points of E, is applicable [6].

The efficient global optimization methods of Griewank [7] and Snyman and Fatti [1] can be derived from (2). The method due to Griewank [7] assumes $m(t) = (E(x(t)) - c)/e$ and $n(t) = -\nabla E(x(t))\dot{x}(t)$, where c is the target level, to be set somewhat higher than the global minimum of the the objective function. Snyman and Fatti's method [1] considers the case where $m(t) = 1$ and $n(t) = 0$, i.e. the trajectory is determined by,

$$\ddot{x}(t) = -\nabla E(x(t)), \quad x(0) = x_0, \quad \dot{x}(0) = 0.$$

The paper is organized as follows: Section 2 presents the proposed class of neural network training methods; Section 3 is devoted to the presentation of the experimental results; the paper ends with concluding remarks and ideas for future research in Section 4.

2 The proposed method

Through this study each method for the numerical solution of ordinary differential equations corresponds to a Neural Network training algorithm. In previous work [8] we investigated the effectiveness of numerical methods for the solution of first order differential equations (1) on the task of neural network training. Here we consider the effectiveness of both the formulation and the local minimizer procedure proposed in [1]. Moreover, we propose a simple modification of this local minimizer algorithm which seems to be more efficient in Neural Network training applications. Our modification operates as follows. Following a trajectory, if the value of the error function decreases, the step of integration is increased by a factor ζ to speed–up convergence; while, if the ratio of the current function value to the previous function value exceeds a pre–specified factor, $\beta > 1$, the previously visited point is retained and a new trajectory starts from it. At the beginning of a new trajectory, the local minimizer procedure requires a reduction in the function value to render a step–size acceptable; otherwise the step–size is halved. Our experience indicates that when it holds that,

$$\frac{E(x_{t+1})}{E(x_t)} > \beta,$$

$\|x_t\|$ is small. Thus, in practice, the condition imposed by the local minimization procedure operates similar to the Armijo line search algorithm [9]. The modified local minimization procedure of [1] is presented below in pseudocode. The proposed modifications are indicated with the comment *Additional Step* in parentheses.

Modified Snyman–Fatti Algorithm (MSF)

1. Initial values for parameters $\alpha, \gamma, \epsilon, k_m, \Delta t, x_0$ are given. Δt is the time step of integration method and x_0 is the starting point.
2. Set $j \leftarrow 0, \quad k \leftarrow 0, \quad F_t \leftarrow \infty$.
3. Set $x_m \leftarrow x_k, \quad x_s \leftarrow x_k, \quad x_b \leftarrow x_k$ and
$\dot{x}_k \leftarrow -\nabla F_k \Delta t/\gamma, \;\; x_{k+1} \leftarrow x_k + \dot{x}_k \Delta t$
if $F_{k+1} < F_k$ set $F_b \leftarrow F_{k+1}, \quad x_b \leftarrow x_{k+1}$ and **goto 4**
else set $\Delta t \leftarrow \Delta t/2$ and **goto 3**.
4. Set $F_s \leftarrow F_k, \quad F_m \leftarrow F_k$.
5. Set $xprev_k \leftarrow x_k$, (**Additional Step**)

$$x_{k+1} \leftarrow x_k + \dot{x}_k \Delta t, \quad t_1 \leftarrow -\dot{x}_k^T \nabla F_{k+1}, \quad T = 0.5\|\dot{x}_k\|^2,$$
$$\dot{x}_{k+1} \leftarrow \dot{x}_k - \nabla F_{k+1} \Delta t, \quad k \leftarrow k+1.$$

if $F_k \leqslant Es$ **goto 16**. (**Additional Step**)(Es stopping error)
if $\|\nabla F_k\| \geqslant \epsilon$ **goto 6** else if $F_k < F_b$ and $F_k \leqslant F_m$ **goto 16.**
else if $F_m < F_b$ set $x_b \leftarrow x_m, \quad x_k \leftarrow x_m$ and **goto 3**
else set $x_k \leftarrow x_b$ and **goto 3**.
6. if $k > k_m$ **goto 16**.
if $F_k/F_m > \beta$ set $x_k \leftarrow xprev_k$ and **goto 3**. (**Additional Step**)
if $F_k \leqslant F_m$ set $F_m \leftarrow F_k, \quad x_m \leftarrow x_k, \quad \dot{x}_m \leftarrow \dot{x}_k$
$\Delta t = \Delta t * \zeta$ (**Additional Step**)
7. if $t_1 < 0$ **goto 8** else set $j \leftarrow 0$ and **goto 5**.
8. if $F_T > F_m$ set $F_T \leftarrow F_m$.
if $(F_k - F_T) > \alpha(F_s - F_T)$ **goto 9**
else if $T < (1-a)(F_s - F_T)$ **goto 9**
else set $j \leftarrow 0$ and **goto 5**.
9. if $F_m < F_b$ set $x_b \leftarrow x_m, \quad F_b \leftarrow F_m$ and **goto 10.**
else set $x_b \leftarrow x_m$ and continue
10. Set $x_k \leftarrow \frac{1}{2}(x_m + x_s), \quad x_s \leftarrow x_k, \quad x_m \leftarrow x_k, \quad \dot{x}_k \leftarrow \frac{1}{2}\dot{x}_m$.
11. if $j \leftarrow 0$ **goto 14**.
12. Set $t_2 \leftarrow -\dot{x}_k^T \nabla F_k$.
13. if $t_2 < 0$ set $j \leftarrow 0$ and **goto 3** else $\dot{x}_k \leftarrow \dot{x}_k/2^j$.
14. $j \leftarrow j+1$.
15. if $F_b \neq F_m$ **goto 4** else set $\dot{x}_m \leftarrow \frac{1}{2}\dot{x}_m$ and **goto 4**
16. Set $x_f \leftarrow x_k, \quad F_f \leftarrow F(x_f)$ and terminate.

3 Experimental Results

The performance of the proposed method has been compared with that of well–known and widely used variations of the Backpropagation (BP) method, namely: Backpropagation with Momentum (MBP) [10, 11], Second Order Momentum (SMBP) and Adaptive Backpropagation (ABP), using the adaptive scheme suggested by Vogl [10, 12], Parallel Tangents method (PARTAN) [13], Scaled Conjugated Gradient(SCG), Resilient Back Propagation [14] (RPROP) and Improved Resilient Back Propagation [15] (iRPROP).

3.1 Description of the Problems

The problems we used were Cancer1, Diabetes1, and Heart1. All three are classification problems from the proben1 [16] dataset with fixed training and test sets. A brief description of each problem follows.

Cancer1: The architecture used was 9–4–2–2. The stopping error criterion for training was an error goal of 0.05 within 1000 function (including gradient) evaluations. In the experiments the best results for the methods were given with the following parameters: For BP the step–size was set to 0.9, for PARTAN it was 0.9, for MBP and SMBP the step–size was 0.9 and the momentum term was 0.7. For ABP, the error ratio factor was 1.04, the stepsize increment factor was equal to 1.05, while the stepsize decrease factor was 0.7.

Diabetes1: The architecture used was an 8–2–2–2 feedforward neural network. The stopping error criterion for training was an error goal of 0.15 within 1000 function (also counting gradient) evaluations. In the experiments the best results for the methods were given with the following parameters: For BP the step–size was 0.6, for PARTAN it was 0.9, for MBP the step–size was 0.9 and the momentum term 0.4, for SMBP the stepsize was 0.9 while the momentum was 0.6. For ABP, the error ratio factor was 1.04, the step–size increment factor was equal to 1.05 while the step–size decrease factor was 0.7.

Heart1: The architecture used was 35–8–2. The stopping error criterion for training was an error goal of 0.1 within 1000 function (also counting gradient) evaluations. The parameter configuration for the previous two problems was also applied in this case. In the experiments the best results for the methods were given with the following parameters: For BP the step–size was 0.9, for PARTAN it was 0.9. For MBP the step–size was 0.9 and the momentum term was 0.6. For SMBP the step–size was 0.9 and the momentum term was 0.7. For ABP, the error ratio factor was 1.04, the step–size increment factor was equal to 1.05 while the step–size decrease factor was 0.7.

3.2 Presentation of the Results

The parameter setup for the proposed method was the default setup suggested in [1]: $\alpha = 0.95$, $\gamma = 2$, $\Delta t = 2$, $\epsilon = 10^{-4}$, $k_m = 1000$. The values of ζ and β, required for the modification of the algorithm, were set to 1.05 and 1.04, respectively. We performed 100 simulations for each problem. The evaluation measures we used are, the number of successes(suc), the mean, the standard deviation (stdev), the minimum (min), and the maximum (max), number of function evaluations (also counting gradient evaluations). In addition to the above measures we computed these statistics for the percentage of misclassification on the test set. Regarding the Cancer1 problem as it is shown in Table 1 the less computationally demanding method is iRPROP. MSF ranked third after iRPROP and RPROP. In the test set MSF exhibited the second best classification error after

Table 1. Cancer1 problem

Algorithm	Training Set Mean	Stdev	Max	Min	Suc.	Test Set Mean	Stdev	Max	Min
BP	583.25	154.72	1001	347	98	2.2	0.57	3.44	0.57
MBP	300.79	95.59	664	158	100	2.01	0.63	3.65	0.57
SMBP	323.98	102.93	773	157	100	2.02	0.60	3.44	0.57
ABP	74.26	8.87	95	63	100	1.93	0.83	3.44	0.57
PARTAN	159.12	39.99	273	87	100	2.15	0.48	3.45	1.15
SCG	259.36	807.77	1001	16	92	4.51	9.76	37.36	0.57
RPROP	16.26	2.93	24	9	100	**1.74**	0.67	4.02	0.00
iRPROP	**15.25**	4.66	38	9	100	1.89	0.78	3.44	0.57
MSF	52.25	14.05	109	33	100	1.82	0.55	2.87	0.57

Table 2. Diabetes1 problem

Algorithm	Training Set Mean	Stdev	Max	Min	Suc.	Test Set Mean	Stdev	Max	Min
BP	1001.00	0.00	1001	1000	0	36.45	36.45	36.45	36.45
MBP	863.75	186.39	1001	354	46	28.71	4.80	36.45	23.43
SMBP	865.45	190.57	1001	256	46	28.70	5.12	36.45	21.88
ABP	664.85	153.80	965	385	100	**24.54**	0.94	29.17	22.92
PARTAN	796.35	145.54	1001	549	81	26.44	3.01	36.45	23.96
SCG	499.69	973.11	1001	64	87	26.48	3.98	36.45	21.87
RPROP	90.11	31.62	206	47	100	25.94	1.54	30.20	21.87
iRPROP	**59.27**	20.91	159	35	100	25.00	1.56	28.12	20.31
MSF	112.87	92.80	1002	73	99	25.35	1.43	36.45	23.44

Table 3. Heart1 problem

Algorithm	Training Set Mean	Stdev	Max	Min	Suc.	Test Set Mean	Stdev	Max	Min
BP	1001.00	0.00	1001	1001	0	20.75	0.81	23.04	18.70
MBP	631.09	136.06	984	367	100	20.62	1.10	22.60	17.82
SMBP	638.17	161.63	1001	303	98	20.93	1.18	24.35	17.82
ABP	159.13	26.77	217	95	100	20.76	1.01	22.60	17.82
PARTAN	327.69	47.95	427	213	100	20.64	0.66	22.60	19.13
SCG	339.37	841.67	1001	46	91	21.60	4.76	47.82	17.82
RPROP	20.49	3.18	30	14	100	20.69	0.99	23.47	18.26
iRPROP	**16.89**	2.47	24	12	100	**20.34**	1.00	22.17	17.39
MSF	42.55	8.22	73	30	100	20.59	1.30	24.34	16.95

RPROP. The results for the problem Diabetes1 are reported in Table 2. Again RPROP and iRPROP performed less function evaluations than the other methods, while MSF ranked third. Concerning test set performance, ABP obtained the best classification error while MSF was the third best method. On the Heart1 problem, as shown in Table 3, iRPROP required the least function evaluations and produced the minimum classification error in the test set. MSF had the second best classification error and the third place with respect to function evaluations.

4 Conclusions

This paper presents a new scheme for feedforward multilayer neural networks training. We utilize the trajectory that results from the solution of the second–order ordinary differential equation $\ddot{x} = -\nabla E(x)$, using a modification of the method proposed in [1] to obtain minima of the network's error function. The proposed method clearly outperforms the well known and widely used family of BP and conjugate gradient methods. It is less efficient in terms of function evaluations than RPROP and iRPROP but with respect to generalization it exhibits almost the same performance. In a future correspondence we intend to apply our approach on the general second order differential equation (2). We will also study the implications that arise from the application of the Theorem due to Poincaré in the present setting.

References

1. Snyman, J., Fatti, L.: A multi–start global minimization algorithm with dynamic search trajectories. JOTA **54** (1987) 121–141
2. Törn, A., Žilinskas, A.: Global optimization. In Goos, G., Hartmans, J., eds.: Lecture Notes in computer Science. Volume 350. Springer Verlag (1987)
3. Incerti, S., Parisi, V., Zirilli, F.: A new method for solving nonlinear simultaneous equations. SIAM J.Num.Anal **16** (1979) 779–789
4. Inomata, S., Cumada, M.: On the golf method. Bulletin of the Electronical Laboratory **25** (1964) 495–512
5. Zhidkov, N., Shchdrin, B.: On the search of minimum of a function of several variables. Computing methods and Programming **10** (1978) 203–210
6. Pshenichnyi, B., Marchenko, D.: On one approach to the search for the global minimum. Optimal Decision theory **2** (1967) 3–12
7. Griewank, A.: Generalized descnet for global optimization. JOTA **34** (1981) 11–39
8. Petalas, Y.G., Tasoulis, D.K., Vrahatis, M.N.: Trajectory methods for neural network training. In: Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA'04), vol. 1, ACTA press (2004) 400–408.
9. Armijo, L.: Minimization of function having lipschitz continuous first partial derivatives. Pac J. Math. **16** (1966) 1–3
10. Magoulas, G., Vrahatis, M.N., Androulakis, G.: Effective backpropagation training with variable stepsize. Neural Networks **10** (1997) 69–82
11. Magoulas, G., Vrahatis, M.N., Androulakis, G.: Increasing the convergence rate of the error backpropagation algorithm by learning rate adaptation methods. Neural Computation **11** (1999) 1769–1796
12. Vogl, T., Mangis, J., Rigler, A., Zink, W., Alkon, D.: Accelerating the convergence of the back-propagation method. Biol. Cybern. **59** (1988) 257–263
13. Rao, S.: Optimization theory and Applications. Wiley Eastern Limited (1992)
14. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The rprop algorithm. In: Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA. (1993) 586–591
15. Igel, C., Hüsken, M.: Improving the Rprop learning algorithm. In Bothe, H., Rojas, R., eds.: Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000), ICSC Academic Press (2000) 115–121
16. Prechelt, L.: Proben1: A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94 (1994)