TRAJECTORY METHODS FOR SUPERVISED LEARNING

Yannis G. Petalas and Michael N. Vrahatis
Department of Mathematics, University of Patras, GR–26110 Patras, Greece,
University of Patras Artificial Intelligence Research Center(UPAIRC)
email:{petalas,vrahatis}@math.upatras.gr

Keywords: Neural Networks, Trajectory Methods, Ordinary Differential Equations

Abstract. *The task of supervised learning in Artificial Neural Networks reduces to the minimization of the network error function subject to the weights of the network. Trajectory Methods are global optimization methods that formulate the optimization problem into a set of ordinary differential equations, the equilibrium points of which, correspond to local minima of the objective function. In this work, we apply Trajectory methods to address the Neural Network training problem. The reported experimental results indicate that this is a promising approach.*

1 INTRODUCTION

A trajectory method[1] defines a finite set of curves (paths, trajectories) constructed in a way that ensures that many of the solutions of the optimization problem lie on them. In many cases these curves are created by solving ordinary differential equations of first, or second order. Various papers describe the use of second order differential equations to create search trajectories[2, 3, 4, 5, 6, 7]. In analogy with classical mechanics these search trajectories are based on Newton's law for a particle of mass $m(t)$ in a potential F subject to a dissipative force $-u(t)\dot{x}(t)$,

$$m(t)\ddot{x}(t) + u(t)\dot{x}(t) = -\nabla f(x(t)). \tag{1}$$

Artificial neural networks (ANNs) are computational paradigms which implement simplified models of their biological counterparts, biological neural networks. Biological neural networks are the local assemblages of neurons and their dendritic connections that form the (human) brain. Accordingly, ANNs are characterized by local processing at the level of artificial neurons; massively parallel processing, implemented by rich connection patterns between neurons; the ability to acquire knowledge via learning from experience; and knowledge storage in distributed memory through the synaptic connections among neurons.

In supervised training, both the inputs and the outputs are provided for a number of patterns which comprise the training set. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are propagated backward through the system, in order to adjust the weights which in turn control the network. The goal is to minimize the error function of the network with respect to the weights of the network.

In this work, we propose the application of the trajectory methods to the optimization problem that arises in the training of a neural network. The paper is organized as follows: In Section 2 we briefly review some well–known trajectory methods and the proposed approach is exposed. In Section 3 the experimental results are presented, and finally in Section 4 we give the conclusions of our work.

2 TRAJECTORY METHODS

In this Section, we give a short description of the most popular trajectory methods, namely the methods due to Griewank[4], Snyman and Fatti[3], and Branin[2].

Griewank method: Griewank[4] states five desirable properties that a search method must satisfy using a target value c which is slightly larger than the global minimum f_*.

1 The trajectory cannot converge to minima with values greater than c.

2 As long as $f \gg c$, the trajectory is little affected by the perturbation of u and tries to follow a descent direction with respect to $-\nabla u(t)$.

3 As f tends to c, the trajectory minimizes more thoroughly and finally reduces to a local minimization technique when $f \leqslant c$.

4 The trajectory does not explicitly depend on the Hessian of f.

5 The trajectory is invariant with respect to translations and multiplication of $f-c$ by a positive scalar.

Griewank has also shown that these requirements are satisfied by the solutions of the following second order differential equation,

$$\ddot{x}(t) = -e(I - \dot{x}\dot{x}^\top)\frac{\nabla f(x)}{f(x)-c}, \quad e > 0, \tag{2}$$

from any initial point $x_0, \dot{x}_0$ with $f(x_0) > c$ and $\|\dot{x}_0\| = 1$. The values of the parameters e and c play a significant role in the performance of the method.

Snyman and Fatti method: In[3] Snyman and Fatti use the following initial value problem,

$$\begin{aligned} \ddot{x} &= -\nabla f(x), \\ x(0) &= x_0, \\ \dot{x}(0) &= 0. \end{aligned} \tag{3}$$

The multiplication of the above equation with $\dot{x}$ and its integration give the energy conservation relationship,

$$\frac{1}{2}\|\dot{x}\|^2 + f(x) = f(x_0). \tag{4}$$

A particle will move in the direction of steepest descent and his kinetic energy will increase as long as it moves downhill. The energy conservation relationship will continue past a local minimum and surmount a ridge of height less than $f(x_0)$, possibly finding an even lower value of f. The trajectory must be terminated before it approximately retraces itself. Snyman and Fatti use for the integration of the equation the leap–frog method which has been proven efficient besides its simplicity. Finally they combine this method with a multistart global optimization algorithm. The starting points are taken randomly and a Bayesian approach is used to terminate the procedure. This method has been applied in numerous test functions with very good results.

Branin's method: Branin[2] studied the problem of finding all the solutions of the equation $F(x) = 0$ in $\mathbb{R}^n$. He introduced the following first order differential equation,

$$\frac{d}{dt}F(x) + F(x) = 0, \quad x(0) = x_0, \tag{5}$$

which gives,

$$\dot{x} = -DF(x)^{-1}F(x), \quad x(0) = x_0, \tag{6}$$

as long as the Jacobian $DF(x)$ is nosingular. Applying Euler's method to the above equation we get the usual Newton method with stepsizes h_k,

$$x_{k+1} = x_k + h_k DF(x_k)^{-1}F(x_k). \tag{7}$$

This is the reason these trajectories are also known as Newton trajectories. From the analytical solution,

$$F(x(t)) = F(x_0)e^{-t}, \tag{8}$$

it follows that the trajectories tend to zero if the integration can be continued to $t \to \infty$. Furthermore, we note that F has constant direction on the trajectories, that is, $\frac{F(x)}{\|F(x)\|}$ stays constant on any trajectory $x(t)$. However, the right hand side of the equation is not defined in these points where $DF(x)$ becomes singular.

Here we experimentally show that the application of Stoermer's rule[8] to the initial value problem of Eq. (3) produces better results than the backpropagation family methods. Assume the following initial value problem,

$$\begin{aligned} \ddot{y} &= f(x,y), \\ y(x_0) &= y_0, \\ \dot{y}(x_0) &= z_0. \end{aligned} \tag{9}$$

$$\tag{10}$$

Stoermer's rule is described by the equations,

$$\begin{aligned} y_1 &= y_0 + h\left[z_0 + \frac{1}{2}hf(x_0,y_0)\right] \\ y_{k+1} &= 2y_k - y_{k-1} + h^2 f(x_0 + kh, y_k) \quad k = 1,\ldots,m-1 \\ z_m &= y_m - y_{m-1}/h + \frac{1}{2}hf(x_0 + H, y_m) \end{aligned} \tag{11}$$

where H is the total step to be taken in m substeps. Thus, each substep is of length $h = H/m$. The final value, z_m, is $y(x_0 + H)$. Henrici[9] showed how to rewrite the above equations reducing the roundoff error using the quantities $\Delta_k = y_{k+1} - y_k$. The new formulation of the method assumes the form,

$$\begin{aligned} \Delta_0 &= h\left[z_0 + \frac{1}{2}hf(x_0, y_0)\right], \\ y_1 &= y_0 + \Delta_0, \\ \Delta_k &= \Delta_{k-1} + h^2 f(x_0 + kh, y_k) \quad k = 1, \ldots, m-1, \\ y_{k+1} &= y_k + \Delta_k \quad k = 1, \ldots, m-1, \\ z_m &= \Delta_{m-1}/h + \frac{1}{2}hf(x_0 + H, y_m). \end{aligned} \tag{12}$$

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], as well as, Scaled Conjugated Gradient (SCG)[14].

3.1 Description of the problems

The problems we used were Cancer1, Diabetes1, and Heart1. All three are classification problems from the proben1[15] 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. The parameter setup for the proposed method was $H = 12$ and $m = 4$.

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. As for the Cancer1 problem, the parameter setup for the proposed method was $H = 12$ and $m = 4$.

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. For the proposed method, H was set to 12 and m was set to 4.

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.

In Tables 1, 2 the results for the cancer1 problem are presented. The proposed method required the minimum function evaluations. From the other methods considered, only ABP exhibited a comparable performance. With respect to misclassification error ABP achieved the minimum while the proposed method exhibited almost the same performance.

In Tables 3, 4 the results for the heart1 problem are exhibited. Concerning the number of function evaluations, the proposed method achieved the best performance followed by ABP. Relative to the classification performance, MBP attained the minimum error but overall all methods exhibited a very similar performance.

The results for the diabetes1 problem are reported in Tables 5, 6. The proposed method required the minimum number of function evaluations to converge. Concerning the classification error, the proposed method achieved the second lowest misclassification percentage.

Algorithm	Mean	Stdev	Max	Min	Suc.
BP	583.25	154.72	1001	347	98
MBP	300.79	95.59	664	158	**100**
SMBP	323.98	102.93	773	157	**100**
ABP	74.26	**8.87**	**95**	63	**100**
PARTAN	159.12	39.99	273	87	**100**
SCG	259.36	807.77	1001	**16**	92
Stoermer	**60.34**	19.86	139	31	**100**

Table 1: FE Cancer1 problem

Algorithm	Mean	Stdev	Max	Min
BP	2.2	0.57	3.44	**0.57**
MBP	2.01	0.63	3.65	**0.57**
SMBP	2.02	0.60	**3.44**	**0.57**
ABP	**1.93**	0.83	3.44	**0.57**
PARTAN	2.15	**0.48**	3.45	1.15
SCG	4.51	9.76	37.36	**0.57**
Stoermer	1.95	0.61	3.45	**0.57**

Table 2: CE Cancer1 problem

Algorithm	Mean	Stdev	Max	Min	Suc.
BP	1001.00	0.00	1001	1001	0
MBP	631.09	136.06	984	367	**100**
SMBP	638.17	161.63	1001	303	98
ABP	159.13	26.77	217	95	**100**
PARTAN	327.69	47.95	427	213	**100**
SCG	339.37	841.67	1001	**46**	91
Stoermer	**123.40**	**19.03**	**181**	85	**100**

Table 3: FE Heart1 problem

Algorithm	Mean	Stdev	Max	Min
BP	20.75	0.81	23.04	18.70
MBP	**20.62**	1.10	22.60	**17.82**
SMBP	20.93	1.18	24.35	**17.82**
ABP	20.76	1.01	22.60	**17.82**
PARTAN	20.64	**0.66**	22.60	19.13
SCG	21.60	4.76	47.82	**17.82**
Stoermer	20.71	0.78	**22.17**	18.7

Table 4: CE Heart1 problem

Algorithm	Mean	Stdev	Max	Min	Suc.
BP	1001.00	0.00	1001	1000	0
MBP	863.75	186.39	1001	354	46
SMBP	865.45	190.57	1001	256	46
ABP	664.85	153.80	965	385	**100**
PARTAN	796.35	145.54	1001	549	81
SCG	499.69	973.11	1001	**64**	87
Stoermer	**355.42**	**115.79**	**889**	181	**100**

Table 5: FE Diabetes1 problem

4 CONCLUSIONS

In this work we propose a trajectory method for neural network training. To this end, the optimization problem that appears in Neural Networks was addressed using a second order differential equation. We solved numerically

Algorithm	Mean	Stdev	Max	Min
BP	36.45	36.45	36.45	36.45
MBP	28.71	4.80	36.45	23.43
SMBP	28.70	5.12	36.45	21.88
ABP	**24.54**	0.94	29.17	22.92
Partan	26.44	3.01	36.45	23.96
SCG	26.48	3.98	36.45	**21.87**
Stoermer	25.19	**0.65**	**26.56**	22.40

Table 6: CE Diabetes1 problem

this equation using Stoermer's rule and tested it on three classification problems. The results showed that the proposed method exhibits a significantly better performance compared to the considered backpropagation family methods.

Acknowledgment

We acknowledge the partial support of the "Pythagoras" research grant awarded by the Greek Ministry of Education and Religious Affairs and the European Union.

References

[1] Diener, I. (1995). "Trajectory methods in global optimization". editors Horst, R. and Pardalos, P. M., In *Handbook of Global Optimization II*, Vol., pp. 649–668, pp. 649–668. Dordrecht, Netherlands: Kluwer.

[2] Branin, F. H. (1972). "A widely convergent method for finding multiple solutions of simultaneous nonlinear equations". *I.B.M Journal of Research and Development*, Vol. 16, pp. 504–522.

[3] Snyman, J.A. and Fatti, L.P. (1987). "A multi–start global minimization algorithm with dynamic search trajectories". *JOTA*, Vol. 54(3,8), pp. 121–141.

[4] Griewank, A.O. (1981). "Generalized descnet for global optimization". *JOTA*, Vol. 34(3,8), pp. 11–39.

[5] Incerti, S., Parisi, V., and Zirilli, F. (1979). "A new method for solving nonlinear simultaneous equations". *SIAM J. Num. Anal*, Vol. 16(3), pp. 779–789.

[6] Zhidkov, N. and Shchdrin, B. (1978). "On the search of minimum of a function of several variables". *Computing methods and Programming*, Vol. 10(3,7), pp. 203–210.

[7] Inomata, S. and Cumada, M. (1964). "On the golf method". *Bulletin of the Electronical Laboratory*, Vol. 25(3), pp. 495–512.

[8] Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1988). "Numerical recipes in C: The art of scientific computing". Cambridge University Press.

[9] Henrici, P. (1962). "Discrete variable methods in ordinary differential equations". John Wiley, New York.

[10] Magoulas, G.D., Vrahatis, M.N., and Androulakis, G.S. (1997). "Effective backpropagation training with variable stepsize". *Neural Networks*, Vol. 10(1), pp. 69–82.

[11] Magoulas, G.D., Vrahatis, M.N., and Androulakis, G.S. (1999). "Increasing the convergence rate of the error backpropagation algorithm by learning rate adaptation methods". *Neural Computation*, Vol. 11(7), pp. 1769–1796.

[12] Vogl, T.P., Mangis, J.K., Rigler, A.K., Zink, W.T., and Alkon, D.L. (1988). "Accelerating the convergence of the back-propagation method". *Biol. Cybern.*, Vol. 59, pp. pp. 257–263.

[13] Rao, S.S. (1992). "Optimization theory and applications". Wiley Eastern Limited.

[14] Moller, M. (1993). "A scaled conjugate gradient algorithm for fast supervised learning". *Neural Networks*, Vol. 6, pp. 525–533.

[15] Prechelt, L. (1994). "Proben1: A set of neural network benchmark problems and benchmarking rules". Technical Report 21/94.