# Trajectory Methods for Neural Network Training

Y. G. Petalas[(1)], D. K. Tasoulis[(1,2)], M. N. Vrahatis[(1,2)]
[(1)] Department of Mathematics, University of Patras, GR–26110 Patras, Greece,
[(2)] University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR–26110 Patras, Greece
Email: {petalas, dtas, vrahatis}@math.upatras.gr

**ABSTRACT**
A new class of methods for training multilayer feedforward neural networks is proposed. The proposed class of methods draws from methods for solving initial value problems of ordinary differential equations, and belong to the subclass of trajectory methods. The training of a multilayer feedforward neural network is equivalent to the minimization of the network's error function with respect to the weights of the network. To address this problem we solve the differential equation $\dot{x} = -\nabla E(x)$, where $x$ is the vector of network weights and $\nabla E$ is the gradient of the error function of the network. The solution of the above system of ordinary differential equations corresponds to the solution of the aforementioned minimization problem.

**KEY WORDS**
Neural Networks, Training Algorithms, Ordinary Differential Equations, Trajectory Methods

## 1 Introduction

Artificial Feedforward Neural Networks (FNNs) are parallel computational models comprised of densely interconnected, simple, adaptive processing units, characterized by an inherent propensity for storing experiential knowledge and rendering it available for use. FNNs resemble the human brain in two fundamental respects; firstly, knowledge is acquired by the network from its environment through a learning process, and secondly, inter-neuron connection strengths, known as synaptic weights are employed to store the acquired knowledge [3].

Two critical parameters for the successful application of FNNs are the appropriate selection of network architecture and training algorithm. The problem of identifying the optimal network architecture for a specific task remains up to date an open and challenging problem.

The efficient supervised training of FNNs is the subject of considerable ongoing research and numerous algorithms have been proposed to this end. Supervised training amounts to the global minimization of the network error function $E$. The rapid computation of a set of weights that minimizes this error is a rather difficult task since, in general, the number of network weights is large and the resulting error function generates a complex surface in the weight space, characterized by multiple local minima and broad flat regions adjoined to narrow steep ones.

Considerable effort has been spent in developing local minimization methods. Thus, modifying these methods for the case of global minimization is important. One way to accomplish this task is to modify the differential equation that describes the local descent trajectory. The methods based on this modification constitute the subclass of *trajectory* methods [13]. A crucial property of the equations under consideration is that the trajectory passes through the neighborhood of the majority of stationary points of the objective function.

The trajectory that emerges by the solution of the system of differential equations:

$$\frac{dx}{dt} = -\nabla E(x), \tag{1}$$

corresponds to the solution of the unconstrained minimization of the objective function $E(x)$ with respect to $x$, where $x \in \mathbb{R}^n$. The above equation represents a set of first order differential equations that has to be solved in order to detect the minima of the function $E(x)$.

Besides the above system of first order differential equations, a modification of Eq. (1) can also be interpreted, through a mechanical analogy, as an assignment of mass to a point moving in a field of forces [4, 5, 15]. Mathematically this modification means introducing a second order term into the equation of motion. Let the mass of the particle be $m(t)$. The particle is moving in a field of forces defined by the potential $E$ subject to the dissipative force $-n(t)\dot{x}(t)$. The trajectory may then be described by a system of differential equations:

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

where $m(t) \geqslant 0$ and $n(t) \leqslant 0$. Under a set of conditions the trajectory converges to a local minimum of the objective function $E$. It is obvious that the efficiency of the algorithm heavily depends on the parameters $m(t)$ and $n(t)$.

A similar idea has been suggested in [10], where the motion of a point with unit mass is described in a field with potential $E$ without friction. The system of equations of the motion of the particle is then a special case of the above equation.

Furthermore, since the system is autonomous, a theorem due to Poincare states that the trajectory passes through the neighborhood of all the stationary points of $E$. The application of this theorem to the case of neural network training will be very interesting since all the minima of the function $E$ can be detected. We intend to pursue this line of research in a future correspondence.

Some other methods that constitute a special case of the introduced second order differential equation are:

- The method by Griewank [2],

  $$m(t) = (E(x(t)) - c)/e,$$

  and

  $$n(t) = -\nabla E(x(t))\dot{x}(t)$$

  where $c$ is the target level that has to be set somewhat higher than the global minimum of the the objective function. If a point $x_g$ is found such that $E(x_g) < c$ then the global search defined by the second order differential equation is terminated and a local descent is started from the point $x_g$. In [2] the choice of the parameter values for $e$ and $c$ is discussed and examples of successful minimization are provided.

- The method by Snyman and Fatti [12] where $m(t) = 1$ and $n(t) = 0$ i.e. the trajectory is determined by

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

  This gives the following energy conservation relationship:

  $$\frac{1}{2}\left\|\dot{x}(t)\right\|^2 + E(x(t)) = E(x_0).$$

  This allows the particle to continue past a local minimum (the minimum is recorded) and surmount a ridge of height less than $E(x_0)$, continuing further along a path that may lead to an even lower value of $E$. Using many starting points the probability that the global minimum will be found increases. It is worth noting, that for the method to be efficient the trajectory is terminated before it retraces, or approximately retraces, itself in an indefinite motion.

In the next Section a more detailed description of the proposed method is provided, while Section 3 presents experimental results. The paper ends with concluding remarks and ideas for future research.

## 2 The proposed class of methods

There is a variety of efficient and effective methods for the numerical solution of ordinary differential equations of the form $y' = f(x, y)$, with initial value $y_0 = y(x_0)$ [1, 9]. Through our approach each one of these methods corresponds to a Neural Network training algorithm. In this paper we focus on the application of Runge–Kutta methods. In particular we will consider here the second order Runge–Kutta methods. It is evident that for the differential Eq. (1) these methods assume the form:

$$\begin{aligned} x_{n+1} &= x_n + \alpha_1 k_1 + \alpha_2 k_2, \\ k_1 &= h\nabla E(x_n), \\ k_2 &= h\nabla E(x_n + \beta_2 k_1), \end{aligned}$$

where $h > 0$ determines the stepsize. If the coefficients $\alpha_1, \alpha_2, \beta_2$ satisfy a specific system of algebraic equations then the corresponding method constitutes a second order Runge–Kutta method [1]. It is well–known that there is an infinite number of solution sets [1].

We have tried some well–known combinations of the values $\alpha_1, \alpha_2, \beta_2$ and produced the following three Runge Kutta methods which we have used in the test problems considered:

- RK1: $\alpha_1 = 0, \alpha_2 = 1, \beta_2 = \frac{1}{2}$,
- RK2: $\alpha_1 = \frac{1}{2}, \alpha_2 = \frac{1}{2}, \beta_2 = 1$,
- RK3: $\alpha_1 = \frac{1}{4}, \alpha_2 = \frac{2}{3}, \beta_2 = \frac{2}{3}$.

The above sets of parameters correspond to well–known and widely used methods, and in particular the third one (RK3) is considered optimal since it obtains the smallest possible local truncation error among all the second order Runge–Kutta methods.

## 3 Experimental Results

Primarily, we briefly present the test problems considered. For these problems we have applied and compared some well known and widely used variations of the Backpropagation (BP) method; namely:

- Backpropagation with Momentum (MBP) [6, 7],
- Second Order Momentum (SMBP), and
- Adaptive Backpropagation (ABP) using the adaptive scheme suggested by Vogl [6, 14].

Furthermore, we also applied the Parallel Tangents method (PARTAN) [11].

### 3.1 Description of the Problems

The problems used were the classical XOR, the Coder-Decoder, and Cancer1 a classification problem from the proben1 [8] dataset. In more detail:

- **XOR**: This is a classical test problem which is characterized by a multitude of local minima. The performance of training algorithms on this problem is very sensitive on the initial weights. The network topology used was the traditional one, namely, 2-2-1. The stopping error criterion for training was an error goal of 0.04 within 2000 function evaluations (including gradient evaluations). The parameter values used by the Backpropagation family methods were: stepsize equal to 0.75 and the momentum term was set to 0.9. For Vogl's method the error ratio factor was 1.04, the stepsize increment factor was equal to 1.05 while the stepsize decrease factor was 0.7. For the Runge Kutta methods a stepsize equal to 40 was selected. The final choice of the parameters was done after experimentation.

- **Coder-Decoder**: This is a problem which has one hidden layer that includes a number of neurons equal to the logarithm of the input units. The desired output of the network is identical to the input. The architecture used was a 4–2–4 feedforward neural network. Four training input patterns are presented to the network. Each pattern has four bits. Three of them are zero and the remaining one has the value of one. The stopping error criterion for training was 0.1 within 600 function evaluations (including gradient evaluations). The values for the parameters for the Backpropagation family methods and the Runge-Kutta methods were set to the values used in XOR.
- **Cancer1**: This is a classification problem taken from the proben1 [8] dataset. The architecture used was a 9–4–2–2 feedforward neural network (best architecture for cancer1 in proben1). The stopping error criterion for training was an error goal of 0.05 within 2000 function evaluations (as before, including gradient evaluations). The parameter configuration for the previous two problems was also applied in this case.

## 3.2 Presentation of the Results

We performed 100 simulations for each problem. For the first two problems, XOR and Coder-Decoder, we measured the number of successes, the mean number of function evaluations (gradient evaluations were also included), the minimum (min) and the maximum (max) number of function evaluations, and the standard deviation. For problem Cancer1, in addition to the above measures we computed the aforementioned statistics for misclassification.

As it is exhibited in Table 1 Backpropagation never converged within less than 894 Function Evaluations. The proposed RK1 exhibits the smallest mean value. MBP and ABP had a performance similar to RK1, but RK1 exhibited smaller deviation. Finally, MBP achieved a 95% convergence percent. The smallest deviation was attained by RK3.

On the other hand, as it is exhibited in Table 2 in the Coder-Decoder problem RK1, RK2 and RK3 performed the minimum function evaluations and only ABP exhibited a similar performance.

Regarding the Cancer1 problem as it is shown in Tables 3 and 4 the overall less computationally demanding method is RK2. On the other hand, the best classification error (CE) is obtained by RK3. Again, ABP had the most comparable results with the proposed methods. In conclusion in all the test cases we examined the proposed methods outperformed the backpropagation family methods.

Table 1. XOR problem

| Algorithm | Mean | Stdev | Max | Min | Suc |
|---|---|---|---|---|---|
| BP | 1586.03 | 293.598 | 1994 | 894 | 36 |
| PARTAN | 441.92 | 92.47 | 591 | 25 | 100 |
| MBP | 148.51 | 90.21 | 356 | 28 | 95 |
| SMBP | 470.52 | 131.59 | 598 | 118 | 100 |
| ABP | 161.08 | 62.52 | 510 | 89 | 100 |
| RK1 | 143.4 | 53.61 | 294 | 64 | 100 |
| RK2 | 249.9 | 145.68 | 902 | 70 | 100 |
| RK3 | 163 | 48.89 | 388 | 70 | 100 |

Table 2. Coder-Decoder problem

| Algorithm | Mean | Stdev | Max | Min | Suc |
|---|---|---|---|---|---|
| BP | 522.09 | 60.35 | 595 | 400 | 100 |
| PARTAN | 366.03 | 101.23 | 579 | 217 | 100 |
| MBP | 354.01 | 159.14 | 564 | 84 | 100 |
| SMBP | 409.84 | 82.67 | 581 | 221 | 100 |
| ABP | 78.47 | 12.37 | 134 | 65 | 100 |
| RK1 | 67.34 | 32.83 | 163 | 30 | 100 |
| RK2 | 71.56 | 39.77 | 214 | 26 | 100 |
| RK3 | 78.42 | 48.53 | 296 | 28 | 100 |

Table 3. Cancer1 problem

| Algorithm | Mean | Stdev | Max | Min | Suc |
|---|---|---|---|---|---|
| BP | 527.36 | 133.98 | 865 | 330 | 100 |
| PARTAN | 164.6 | 53.03 | 327 | 89 | 100 |
| MBP | 352.96 | 253.92 | 955 | 25 | 100 |
| SMBP | 345.03 | 99.09 | 548 | 148 | 100 |
| ABP | 74.26 | 8.87 | 95 | 63 | 100 |
| RK1 | 110.96 | 203.36 | 1182 | 43 | 100 |
| RK2 | 50.13 | 8.27 | 68 | 36 | 100 |
| RK3 | 70.46 | 24.76 | 148 | 32 | 100 |

## 4 Concluding Remarks

In this paper we propose a class of methods for training feedforward multilayer neural networks. The proposed methods are based on methods for solving initial value problems of ordinary differential equations. We have tested the proposed approach on some well known test problems for neural network training and the obtained results are promising. Here we have focused the trajectory obtained

Table 4. Cancer1 problem CE

| Algorithm | Mean | Stdev | Max | Min |
|---|---|---|---|---|
| BP | 2.45 | 0.87 | 6.32 | 1.14 |
| PARTAN | 2.16 | 0.51 | 2.87 | 1.14 |
| MBP | 17.27 | 19.40 | 62.64 | 1.72 |
| SMBP | 2.06 | 0.63 | 2.87 | 0.57 |
| ABP | 1.93 | 0.83 | 3.44 | 0.57 |
| RK1 | 1.78 | 0.81 | 3.44 | 0.57 |
| RK2 | 1.99 | 0.61 | 2.87 | 0.57 |
| RK3 | 1.64 | 0.71 | 3.44 | 0.57 |

by $\dot{x} = -\nabla E(x)$.

In a future correspondence we are intend to apply our approach on second order differential equation, as mentioned in Section 1. Besides the second order Runge-Kutta methods applied in the present paper the performance of various other methods for the numerical solution of initial value problems will be investigated.

## References

[1] J. Butcher. *Numerical Analysis of Ordinary differential equations*. Wiley London, 1987.

[2] A.O. Griewank. Generalized descnet for global optimization. *JOTA*, 34(3,8):11–39, 1981.

[3] S. Haykin. *Neural Networks: A Comprehensive Foundation*. New York: Macmillan College Publishing Company, 1999.

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

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

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

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

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

[9] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

[10] B.N. Pshenichnyi and D.I. Marchenko. On one approach to the search for the global minimum. *Optimal Decision theory*, 2(3):3–12, 1967.

[11] S.S. Rao. *Optimization theory and Applications*. Wiley Eastern Limited, 1992.

[12] J.A. Snyman and L.P. Fatti. A multi-start global optimization algorithm with dynamic search trajectories. *JOTA*, 54(3,8):121–141, 1987.

[13] A. Törn and A. Žilinskas. Global Optimization, *Lecture Notes in computer Science*, 350, Springer Verlag, 1987.

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

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