Parallel Tangent Methods with Variable Stepsize

Y.G Petalas, M.N Vrahatis
Department of Mathematics, University of Patras,
University of Patras Artificial Intelligence Research Center (UPAIRC),
GR 26110 Patras, Greece
E-mail: {petalas,vrahatis}@math.upatras.gr

***Abstract*— The most widely used algorithm for training Multilayer Feedforward Neural Networks is Backpropagation. Backpropagation is an iterative gradient descent algorithm. Since its appearance various methods which modify the conventional BP have been created to improve its efficiency. One such algorithm which uses an adaptive learning rate is backpropagation with variable stepsize, proposed in [3]. Parallel Tangent methods are used in global optimization to modify and improve the simple gradient descent algorithm by using from time to time the difference between the current point and the point before two steps as a search direction, instead of using the gradient. In this study, we investigate the combination of the BPVS method with the Parallel Tangent approach for neural network training. We perform experimental results on well–known test problems to evaluate the efficiency of the method.**

I. Introduction

The Backpropagation (BP) training algorithm is an iterative gradient descent algorithm designed to minimize the mean squared error between the actual output of a feedforward neural network (FNN) and the desired output. It searches on the error surface along the direction of the gradient in order to minimize the error function of the network. Standard BP is slow in practice because it gets easily trapped in local minima. Numeral variations of BP have been proposed to improve the performance of the conventional algorithm. These methods usually include a variable learning rate or add a multiple of the previous weight update to form the new search direction.

Backpropagation with Variable Stepsize (BPVS) [3] is a BP variant that uses a variable learning rate. BPVS uses a variable learning rate based on the Armijo line search technique [1]. Following the approach of BPVS we propose two new methods for neural network training which are based on the Parallel Tangent methods, and the Armijo line search technique.

In Section II we give background material relating to the adaptation of the stepsize. In Section III we describe the proposed method. Section IV is devoted to the presentation of the experimental results, and finally, the paper ends in Section V with concluding remarks.

II. Background Material

There are two basic theorems due to Armijo [1] that underpin the learning rate adaptation scheme proposed in BPVS. The first theorem which guarantees the convergence of the method is stated immediately below.

Theorem 1: (Armijo, 1966) If $0 < \rho \leqslant 0.25K^{-1}$ then for any $w \in S(w^0)$ the set $S^*(w,\rho) = \{w_\lambda : w_\lambda = w - \lambda\nabla E(w), \quad \lambda > 0, \quad E(w_\lambda) - E(w) \leqslant -\rho\|\nabla E(w)\|^2\}$ is a non-empty set of $S(w^0)$ and any sequence of weight vectors $\{w^k\}_{k=0}^{\infty}$ such that $w^{k+1} \in S^*(w^k,\rho), \quad k = 0,1,2,\ldots$ converges to the point w^* which minimizes E.

The optimal value of the stepsize λ depends on the shape of the error function and can be obtained by the value of the Lipschitz constant K. So the weight update equation becomes:

$$w^{k+1} = w^k - 0.5K^{-1}\nabla E(w^k), \quad k = 0,1,2,\ldots$$

The second theorem gives a procedure to tune the stepsize.

Theorem 2: (Armijo 1966) Suppose that η_0 is an arbitrary assigned positive number and consider the sequence $\eta_m = \eta_0 2^{1-m}, \quad m = 1,2,\ldots$. Then the sequence of weight vectors $\{w^k\}_{k=0}^{\infty}$ defined by

$$w^{k+1} = w^k - \eta_{m_k}\nabla E(w^k), \quad k = 0,1,2,\ldots$$

where m_k is the smallest positive integer for which:

$$E(w^k - \eta_{m_k}\nabla E(w^k)) - E(w^k) \leqslant -0.5\eta_{m_k}\|\nabla E(w^k)\|^2, \quad (1)$$

converges to the point w^* which minimizes the error function E.

The application of the second theorem is a useful stepsize adaptation procedure which has been shown to be sufficient to yield an improvement over the BP algorithm with a fixed stepsize [3]. To fine–tune the size of the learning rate, the Lipschitz constant L_k is locally approximated through the following equation:

$$L_k = \frac{\|\nabla E(w^k) - \nabla E(w^{k-1})\|}{\|w^k - w^{k-1}\|}, \quad (2)$$

where w is the vector of weights and E the network error function. After computing the Lipschitz the second theorem due to Armijo can be applied to adjust the learning rate. If the stepsize $0.5L_k^{-1}$ is very long and successive steps in weight space do not satisfy Eq. (1), the stepsize adaptation procedure of Theorem 2 is used to prevent subsequent weight updates from overshooting the minimum of the error surface. On the other hand, if the stepsize $0.5L_k^{-1}$ is smaller than a specific lower bound, BPVS increases it to accelerate convergence. If the stepsize is smaller than the desired accuracy, it is not possible to reach the solution rapidly, except in situations where equation Eq. (1) is not satisfied. A simple adaptation

mechanism to increase the stepsize is to double it. Now, since BPVS satisfies Eq. (1), it converges to a minimizer of E.

In [6], [12] an alternative Armijo stepsize adaptation procedure is proposed which can be applied to any descent direction ϕ^k. This procedure uses two parameters $(a, b) \in (0, 1)$ and can be implemented in two versions depending on the input value of the parameter s. The steps of the algorithm are the following,

1. Input $\{f; x^0; (a, b) \in (0, 1); s \in \{0, 1\}; m^* \in \mathbb{Z}; MIT; \epsilon\}$.
2. Set $k = 0$.
3. If $\|\nabla f(x)\| \leqslant \epsilon$ go to step 6. Else compute a descent direction ϕ^k.
4. if $s = 0$ set $M^* = \{m \in \mathbb{Z} | m \geqslant m^*\}$ and compute the step size
   (a) $\lambda_k = \beta^{m_k} = \arg\max_{m \in M^*}\{\beta^m | f(x^k + \beta^m \phi^k) - f(x^k) \leqslant \beta^m a \langle \nabla f(x^k), \phi^k \rangle\}$.
   Else compute the stepsize $\lambda_k = \beta^{m_k}$,where $m_k \in \mathbb{Z}$ is an integer such that
   (b) $f(x^k + \beta^{m_k}\phi^k) - f(x^k) \leqslant \beta^{m_k} a \langle \nabla f(x^k), \phi^k \rangle$
   (c) $f(x^k + \beta^{m_k - 1}\phi^k) - f(x^k) > \beta^{m_k - 1} a \langle \nabla f(x^k), \phi^k \rangle$
5. Set $x^{k+1} = x^k + \lambda^k \phi^k$. If $k < MIT$ $k = k + 1$ and go to step 3; otherwise go to step 6.
6. Output $[x^k, f(x^k)]$

The selection $s = 0$ is typically used for Newton–like algorithms. For first order algorithms the best choice is $s = 1$. If the objective function is bounded from below the following subprocedure can be used to find a m_k satisfying the relations (b) and (c) of $step\,4$ of the above procedure.

The stepsize subprocedure

1. If $k = 0$, set $m' = m^*$ else set $m' = m_{k-1}$
2. if $m_k = m'$ satisfies relations (b) and (c) from $step\,4$ stop.
3. if $m_k = m'$ satisfies (b) and not (c) set $m' = m' - 1$ and go to $step\,2$.
4. if $m_k = m'$ satisfies (c) and not (b) set $m' = m' + 1$ and go to $step\,2$.

The above approach is useful when the standard Armijo line search algorithm requires many iterations to tune the stepsize.

III. Proposed Method

The Parallel Tangent method [2], [5], [8], [9], [10] is used to speed up the convergence of the steepest descent method. Instead of always taking the steepest descent direction, $-\nabla F(x_i)$, the method suggests using

$$S_i = x_i - x_{i-2}, \quad i \geqslant 2 \tag{3}$$

occasionally. This modification was suggested in [2]. An extension of the previous idea was suggested in [9]. According to the latter method the search directions are taken alternatively as the steepest descent direction and the direction given by Eq. (3). The resulting method is called a gradient based Parallel Tangent (PARTAN) method. The algorithm of this method can be described as follows,

(i) Start with an initial point x_1.
(ii) Search for the minimum along the direction $S_1 = -\nabla f(x_1)$ and set the new point as

$$x_2 = x_1 + \lambda_1^* S_1,$$

(iii) Search along the direction $S_2 = -\nabla f(x_2)$ and obtain the new point x_3.
(iv) Find the next search direction as

$$S_3 = (x_3 - x_1),$$

and obtain the point x_4.
(v) Take the new search direction as

$$S_i = \begin{cases} -\nabla f(x_i) & \text{for } i = 4, 6, 8, \ldots, 2k, \\ (x_i - x_{i-2}) & \text{for } i = 5, 7, 9, \ldots, 2k - 1, \end{cases}$$

and find the new point as

$$x_{i+1} = x_i + \lambda_i^* S_i,$$

where λ_i^* is the optimal step length in the direction S_i.

The proposed scheme, which is named PARTANVS, combines the above elements to produce a neural network training method. PARTANVS uses the adaptive learning rate employed by BPVS when the gradient of the error function is used as a search direction. This way we can reduce the computational effort for the training of the neural network.

The basic steps of our method are described immediately below.

1. Initialize the number of epochs to $k = 1$, the weights to random values, the stepsize to an arbitrary real value η_1 the stopping criterion (SC), the stepsize γ used in step 2 and a stepsize lower bound (SLB), $L_1 = 1, t = 1$.
2. If k is odd and $k > 2$ update the weights using the equation

$$w^{k+1} = w^k + \gamma(w^k - w^{k-2}),$$

and go to step 8 else go to step 3.
3. Compute the gradient vector of the error function $\nabla E(w)$ as in the conventional BP algorithm.
4. Compute the local approximation L_k of the Lipschitz constant using Eq. (2) and set $\eta = 0.5 L_k^{-1}$. If $\eta_k > SLB$ go to step 5; otherwise set $t_k = t_k + 1, \eta_k = \eta_k 2^{tk-1}$ and go to step 5.
5. If Eq. (1) holds, then set $m_k = 1$, and go to step 7; otherwise set $m_k = m_k + 1, t_k = 1$ and go to step 6.
6. Set $\eta_k = \eta_1 2^{1 - m_k}$ and return to step 5.
7. Update weights according to:

$$w^{k+1} = w^k - \eta_k \nabla E(w^k)$$

8. If $E(w^{k+1}) > SC$ set $k = k + 1$ and go to step 2; otherwise the procedure is terminated.

IV. Experimental Results

The performance of the proposed method has been compared with well–known and widely used variations of the Backpropagation (BP) method, namely: Backpropagation with Momentum (MBP) [3], [4], Second Order Momentum (SMBP) and Adaptive Backpropagation (ABP) using the adaptive scheme suggested by Vogl [3], [11], PARTAN, BPVS, PARTANVS and PARTANVS2. PARTANVS uses for the γ parameter of the proposed algorithm a fixed value, while PARTANVS2 uses the learning rate parameter η of the algorithm multiplied by a constant value c, such that the parameter $\gamma = c\eta$ is in the interval $(0, 1)$.

A. Description of the Problems

The problems we used were Cancer1, Diabetes1, and Heart1. All three are classification problems from the proben1 [7] dataset with fixed training and test sets.

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 (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 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 0.9. For ABP, the error ratio factor was 1.04, the stepsize increment factor was equal to 1.05 while the stepsize decrease factor was 0.5. The initial learning rate for BPVS, PARTANVS, and PARTANVS2 was 12.0. The parameter γ for PARTANVS (Section III) was 0.9.

Diabetes1: The architecture used was a 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 0.9, for PARTAN it was 0.9, for MBP the step–size was 0.9 and the momentum term 0.9, for SMBP the stepsize was 0.9 while the momentum was 0.9. 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. The initial learning rate for BPVS and PARTANVS was 12.0. The parameter γ mentioned in the algorithm in Section III was 0.9.

Heart1: The architecture used was a 35–8–2 feedforward neural network. 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 0.9. For SMBP the step–size was 0.9 and the momentum term was 0.9. 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 0.5. The initial learning rate for BPVS, PARTANVS, and PARTANVS2 was 12.0. The variable γ mentioned in the algorithm in Section III was 0.9.

B. Results Presentation

For each test problem we performed 100 simulations and measured the number of successes (conv), the mean number of function evaluations (gradient evaluations were also counted), the minimum (min) and the maximum (max) number of function evaluations, and the standard deviation. We also computed these statistics for the percentage of misclassification.

The results for problem cancer1 are given in Tables I and II. PARTANVS and PARTANVS2 achieve better performance than PARTAN and BPVS. In the training set PARTANVS required the fewer function evaluations to reach the error goal. In the test set PARTANVS and PARTANVS2 had the smallest classification error among all the considered methods.

TABLE I
CANCER1 PROBLEM TRAINING SET

Algorithm	Mean	Stdev	Max	Min	Conv.
BP	589.20	147.37	1001	328	97
MBP	86.75	29.43	212	46	**100**
SMBP	96.88	24.67	182	60	**100**
ABP	71.71	7.51	101	60	**100**
PARTAN	80.80	16.63	131	48	**100**
BPVS	26.63	6.84	56	16	**100**
PARTANVS	**19.28**	**4.30**	44	13	**100**
PARTANVS2	23.55	5.90	**43**	**12**	**100**

TABLE II
CANCER1 PROBLEM TESTING SET

Algorithm	Mean	Stdev	Max	Min
BP	2.52	3.54	37.36	1.15
MBP	2.10	0.58	**3.45**	0.57
SMBP	2.18	**0.53**	**3.45**	0.57
ABP	2.17	0.74	4.02	**0.00**
PARTAN	2.20	0.47	**3.45**	0.57
BPVS	2.28	0.70	4.02	0.57
PARTANVS	2.02	0.75	4.02	0.57
PARTANVS2	**2.01**	0.66	4.02	0.57

The results for the training test of Diabetes1 problem are given in Table III. BP does not manage to train networks. BPVS and PARTANVS performed significantly better than the other methods. PARTANVS was characterized by the smallest standard deviation. In Table IV the results on the test set of Diabetes1 problem are shown. Again PARTANVS had a very good generalization and it achieved the minimum classification error.

Tables V, VI present the results for the training and test set of the Heart1 problem respectively. During the training process PARTANVS required less function evaluations than the other methods while having the smallest standard deviation value. In the test set, PARTANVS2 produced the second best classification error after MBP.

TABLE III
DIABETES1 PROBLEM TRAINING SET

Algorithm	Mean	Stdev	Max	Min	Conv.
BP	1001.00	0.00	1001	1000	0
MBP	458.72	144.10	1001	229	99
SMBP	536.40	151.47	1001	307	98
ABP	365.08	53.84	542	255	100
PARTAN	430.91	158.81	1001	222	99
BPVS	153.10	32.05	224	75	100
PARTANVS	**116.92**	**23.75**	**218**	**74**	**100**
PARTANVS2	139.34	28.83	239	89	100

TABLE IV
DIABETES1 PROBLEM TESTING SET

Algorithm	Mean	Stdev	Max	Min
BP	36.45	0.00	36.45	36.45
MBP	25.40	**0.52**	**26.56**	23.96
SMBP	25.44	0.65	29.17	23.96
ABP	25.32	0.56	27.08	23.96
PARTAN	25.44	1.21	36.46	24.48
BPVS	25.35	0.55	27.08	23.96
PARTANVS	**24.30**	1.30	27.60	**21.35**
PARTANVS2	25.31	0.58	27.08	23.96

TABLE V
HEART1 PROBLEM TRAINING SET

Algorithm	Mean	Stdev	Max	Min	Conv.
BP	1001.00	0.00	1001	1001	0
MBP	164.72	25.95	260	118	**100**
SMBP	216.13	27.74	279	155	98
ABP	139.29	16.89	187	100	**100**
PARTAN	168.22	25.98	239	122	**100**
BPVS	58.90	12.60	117	37	**100**
PARTANVS	**40.73**	**9.33**	**77**	**27**	**100**
PARTANVS2	61.61	13.87	110	34	**100**

TABLE VI
HEART1 PROBLEM TESTING PROBLEM

Algorithm	Mean	Stdev	Max	Min
BP	20.75	0.81	23.04	18.70
MBP	**20.49**	0.89	**22.17**	17.83
SMBP	20.64	0.84	22.61	18.26
ABP	20.63	0.84	22.61	18.26
PARTAN	20.67	0.69	23.04	19.13
BPVS	20.55	**0.66**	22.61	19.13
PARTANVS	20.55	1.20	23.91	**16.96**
PARTANVS2	20.53	0.74	**22.17**	18.26

V. CONCLUSIONS

This paper presents two methods for training neural networks combining two methods, PARTAN and BPVS. The proposed methods draw from the PARTAN method and by following the approach of BPVS, they adapt the learning rate. The proposed methods are relatively simple, straightforward to implement, and compare favorably with PARTAN, BPVS and other backpropagation family methods on the test problems considered.

ACKNOWLEDGEMENTS

We wish to thank Professors G.S. Androulakis and G.D. Magoulas for useful discussions and the referees for their useful remarks. We also 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] L. Armijo. Minimization of function having lipschitz continuous first partial derivatives. *Pac J. Math.*, 16:1–3, 1966.

[2] G.E Forsythe and T.S Motzkin. Asymptotic properties of the optimum gradient method(abstract). *American mathematical society bulletin*, 57:183, 1951.

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

[4] 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.

[5] D.A. Pierre. Search techniques and nonlinear programming. In *Optimization theory with applications*. Wiley, New York, 1969.

[6] E. Polak. *Optimization: Algorithms and Consistent Approximations*. Springer, New York, 1997.

[7] L. Prechelt. Proben1: A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.

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

[9] B.V Shah, R.J Buehler, and O. Kempthorne. Some algorithms for minimizing a function of several variables. *SIAM journal*, 12:74, 1964.

[10] H.W Sorenson. Comparison of some conjugate direction procedures for function minimization. *Journal of the Franklin Institute*, 288:421, 1969.

[11] 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.

[12] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos, and G.D. Magoulas. A class of gradient unconstrained minimization algorithms with adaptive stepsize. *Journal of Computational and Applied Mathematics*, 114(2):367–386, 2000.