An Efficient Improvement of the Rprop Algorithm

Aristoklis D. Anastasiadis
Department of Information Systems and Computing, Brunel University Uxbridge UB8 3PH, United Kingdom Email: cspgada@brunel.ac.uk

George D. Magoulas
Department of Information Systems and Computing, Brunel University Uxbridge UB8 3PH, United Kingdom Email: George.Magoulas@brunel.ac.uk

Michael N. Vrahatis
Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece Email: vrahatis@math.upatras.gr

Abstract

This paper introduces an efficient modification of the Rprop algorithm for training neural networks. The convergence of the new algorithm can be justified theoretically, and its performance is investigated empirically through simulation experiments using some pattern classification benchmarks. Numerical evidence shows that the algorithm exhibits improved learning speed in all cases, and compares favorably against the Rprop and a recently proposed modification, the iRprop.

1. Introduction

Gradient descent is the most widely used class of algorithms for supervised learning of neural networks. The most popular training algorithm of this category is the batch Back-Propagation (BP) [1]. It is a first order method that minimizes the error function by updating the weights w using the steepest descent method [1]:

$$w^{t+1} = w^t - \eta \nabla E\left(w^t\right) \qquad (1)$$

where E is the batch error measure defined as the Sum of Squared differences Error function (SSE) over the entire training set. The parameter η is a heuristic, which is known as learning rate. Proper learning rate values help to avoid convergence to a saddle point or a maximum. In order to secure the convergence of the BP training algorithm and avoid oscillations in a steep direction of the error surface a small learning rate is chosen ($0<\eta<1$). However, it is well known that this approach tends to be inefficient.

Adaptive gradient-based algorithms with individual step-sizes try to overcome the inherent difficulty of choosing the right learning rates. This is done by controlling the weight update for each weight in order to minimize oscillations and maximize the length of the step-size. One of the best algorithms of this class, in terms of convergence speed, accuracy and robustness with respect to its parameters, is the *Resilient Backpropagation* (Rprop) algorithm [2]. Recently a modification of the Rprop, the so-called *Improved Rprop* (iRprop) has been proposed [3]. Empirical evaluations of iRprop gave good results, showing that iRprop outperforms in several cases the Quickprop and Conjugate gradient algorithms [3].

One problem inherent with gradient descent methods relates to convergence to local minima. These techniques use only gradient information, e.g. the partial derivative of the error with respect to the weights, to adapt weight-specific parameters. While some local minima can provide acceptable solutions, they often result in poor performance. This problem can be overcome through the use of global optimization. Various algorithms of this category have been employed, including simulated annealing (SA) [4], evolutionary methods [5], random methods, and deterministic searches [6]. Global optimization, however, is considered computationally expensive, which could be a significant problem particularly for large networks [7].

In this paper, we propose to combine a quick and computationally cheap gradient descent algorithm, Rprop, with more 'global' information like the magnitude of the network batch error, in order to improve the learning speed. This paper is organized as follows. First, we describe the Rprop algorithm and its parameters. Next, the new algorithm is described and a mathematical justification for its behavior is presented. Then the new algorithm is empirically evaluated and compared with the classic Rprop and iRprop. Finally our results are discussed and conclusions are drawn.

2. The Resilient Propagation Algorithm

The basic principle of Rprop is to eliminate the harmful influence of the size of the partial derivative on the weight step. As a consequence, only the sign of the derivative is considered to indicate the direction of the weight update. The size of the weight change is exclusively determined by a weight-specific, so called ‘update-value’

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}(t), & if \ \dfrac{\partial E(t)}{\partial w_{ij}} > 0 \\ +\Delta_{ij}(t), & if \ \dfrac{\partial E(t)}{\partial w_{ij}} < 0 \\ 0, & else \end{cases} \quad (2)$$

where $\vartheta E(t)/\vartheta w_{ij}$ denotes the partial derivative with respect to each weight. The second step of Rprop learning is to determine the new update-values.

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^{+} \cdot \Delta_{ij}(t), & if \ \dfrac{\partial E(t-1)}{\partial w_{ij}} \cdot \dfrac{\partial E\ (t)}{\partial w_{ij}} > 0 \\ \eta^{-} \cdot \Delta_{ij}(t), & if \ \dfrac{\partial E\ (t-1)}{\partial w_{ij}} \cdot \dfrac{\partial E(t)}{\partial w_{ij}} < 0 \\ \Delta_{ij}(t-1), & else \end{cases} \quad (3)$$

where $0 < \eta^{-} < 1 < \eta^{+}$. Thus every time the partial derivative of the corresponding weight $w_{ij}(t)$ changes its sign, which indicates that the last update was too big and the algorithm has jumped over the local minimum, the update-value $\Delta_{ij}(t)$ is decreased by the factor η^{-}. If the derivative retains its sign, the update value is slightly increased in order to accelerate convergence in shallow regions. Additionally, in case of a change in sign, there should be no adaptation in the succeeding learning step. In practice this can be achieved by setting $\vartheta E(t)/\vartheta w_{ij} = 0$ in the adaptation rule. Finally the weight update and the adaptation are performed after the gradient information of the whole pattern set is computed.

The Rprop algorithm requires setting the following parameters: (i) the increase factor is set to $\eta^{+} = 1.2$; (ii) the decrease factor is set to $\eta^{-} = 0.5$; (iii) the initial update-value is set to $\Delta_0 = 0.1$; (iv) the maximum weight step, which is used in order to prevent the weights from becoming too large, is $\Delta_{\max} = 50$ [2].

3. Description of a new method

Rprop is based on the assumption that a sign change of the partial derivative implies a jump over a local minimum along the coordinate direction $w_{ij}(t)$, but it does not take into account whether the weight update has caused an increase or decrease of the error. Also in Rprop, if the derivatives retain their signs, the update value is slightly increased in order to accelerate convergence in shallow regions. This strategy helps to speed up convergence when the derivative is negative but may be inefficient when the two derivatives are positive, as in this case the weight updates may lead the weight trajectory far away from the minimum or in regions with higher error function values. In an attempt to alleviate these situations Rprop employs a heuristic parameter $\Delta_{\max}$, which constraints the size of the update step.

The idea behind the new algorithm is to take into account the evolution of the error. We try to combine ‘individual’ information about the error surface, i.e. the sign of the partial derivative of the error function with respect to a weight, with more ‘global’ information, i.e. the magnitude of the network learning error, in order to decide for each weight individually whether or not to revert/reduce a step. When the gradient of the error function is available at the endpoints of an interval of uncertainty, it is necessary to evaluate function information at an interior point in order to reduce this interval [8]. This is because it is possible to decide whether the corresponding interval brackets a local minimum simply by looking the function values between two successive epochs *t* and *t*-1 (i.e. *E*(*t*-1) and *E*(*t*), respectively), and the gradient values $\vartheta E(t-1)/\vartheta w_{ij}$, $\vartheta E(t)/\vartheta w_{ij}$ at the endpoints of the considered interval. The conditions that have to be satisfied are the following:

$$\frac{\partial E(V1)}{\partial w_{ij}} < 0 \text{ and } \frac{\partial E(V2)}{\partial w_{ij}} > 0 \quad (4)$$

$$\frac{\partial E(V1)}{\partial w_{ij}} < 0 \text{ and } \frac{\partial E(V2)}{\partial w_{ij}} < 0 \text{ and } E(V1) < E(V2) \quad (5)$$

$$\frac{\partial E(V1)}{\partial w_{ij}} > 0 \text{ and } \frac{\partial E(V2)}{\partial w_{ij}} > 0 \text{ and } E(V1) > E(V2) \quad (6)$$

where, $V1$ and $V2$ determine the sets of weights for which the coordinate that corresponds to the weight w_{ij} is replaced by $a = \min\{w_{ij}(t-1), w_{ij}(t)\}$, and $b = \max\{w_{ij}(t-1), w_{ij}(t)\}$ correspondingly. Notice that, at this instance, between two successive epochs, *t*-1 and *t*, all the other coordinates remain the same. The above three conditions lead to the conclusion that the interval $[a,b]$ includes a local subminimizer along the direction, which corresponds to the weight w_{ij}. A robust method of interval reduction called bisection can now be used. By computing the midpoint $m = \frac{1}{2}(a+b)$ of the interval $[a,b]$ we take as the next interval whichever of $[a,m]$ and $[m,b]$ is the one that still brackets a minimizer according to the criteria mentioned above.

The bisection method always converges with certainty within the given interval $[a,b]$ and it is a global convergence method. Moreover it has a great advantage since it is optimal, i.e. it possesses asymptotically the best possible rate of convergence [9]. Also, the number of iterations of the bisection method that are required for the attainment of an approximate minimizer within the interval $[a,b]$ to a predetermined accuracy ε is known beforehand and is given by $\lceil \log_2[(b-a)\varepsilon^{-1}] \rceil$. Finally, it requires the algebraic signs of the values of the gradient to be computed, and proceeds solely by comparing the relative sizes of the functions values. Next, we give a theorem that ensures that the training method that updates the weights according to the criteria mentioned above converges to a local minimizer. The objective is to show that there is a neighborhood of a minimizer of the error function for which convergence to the local minimizer can be guaranteed.

Theorem: Suppose that the error function is twice continuously differentiable in an open neighborhood $S_0 \subset D$ of a local minimizer $w^* \in D$ for which the Hessian is positive definite with property A^{π}. Then there exists an open ball S, with center w^* and radius r, in S_0 such that the sequence generated by the training method that updates the weights according to the criteria mentioned above converges to w^*.

Proof: It can be shown that the proof of the above theorem follows from the proof of Theorem 1 of [10] by observing that the training method forms a nonlinear

```
**Initialise**: t=0; set the maximum number of epochs T, calculate E(0); set q=1; for all weights $w_{ij}$
($i,j$=1,…N) set $\forall i,j : \Delta_{ij}(t) = \Delta_0 = \Delta_{\min}$, $\Delta_{\max}$, $0 < \eta^- < \eta^+ < 1$, $\frac{\partial E(t-1)}{\partial w_{ij}} = 0$, and calculate
$\Delta w_{ij}(0) = -sign\frac{\partial E(0)}{\partial w_{ij}} \cdot \Delta_{ij}(0)$
**Repeat** for $t = 1 \ldots T$
        Compute the gradients $\nabla E(t)$
        **if** E(t) ≤ E(t-1) **then**
            for all weights and biases (i,j=1,…N)
            **if** $\left(\frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} > 0\right)$ **then** {
                $\Delta_{ij}(t) = \min(\Delta_{ij}(t-1) \cdot \eta^+, \Delta_{\max})$
                $\Delta w_{ij}(t) = -sign\frac{\partial E\ (t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$
                $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
                $\frac{\partial E(t-1)}{\partial w_{ij}} = \frac{\partial E(t)}{\partial w_{ij}}$
                $\Delta w_{ij}(t-1) = \Delta w_{ij}(t)$
            }
            **else if** $\left(\frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} < 0\right)$ **then** {
                $\Delta_{ij}(t) = \max(\Delta_{ij}(t-1) \cdot \eta^-, \Delta_{\min})$
                $\frac{\partial E(t)}{\partial w_{ij}} = 0$
            }
            **else if** $\left(\frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} = 0\right)$ **then** {
                $\Delta w_{ij}(t) = -sign\frac{\partial E\ (t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$
                $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
                $\frac{\partial E(t-1)}{\partial w_{ij}} = \frac{\partial E(t)}{\partial w_{ij}}$
                $\Delta w_{ij}(t-1) = \Delta w_{ij}(t)$
            }
        q = 1
        **if** E(t) > E(t-1) **then** {
                $w_{ij}(t+1) = w_{ij}(t) + \frac{1}{2^q}\Delta w_{ij}(t-1)$
                q=q+1
        }
**Terminate**: Get the final weights and the corresponding error value E(w)
```

Figure 1. Pseudocode of the modified Rprop

Jacobi scheme, which fulfils the assumptions of Theorem 1 of [10]. A detailed description is outside the scope of this paper.

Remark : *The Property A^{π}:* Young [11] has discovered a class of matrices described as having property A^{π} that can be partitioned into block tridiagonal form, possibly after a suitable permutation [12]. An algorithmic procedure for transforming a symmetric matrix to a tridiagonal form is presented in [13].

In case the initial points are far from the neighborhood of a local minimizer, then it is possible to equip algorithms with local learning rates (like the one proposed here) with a strategy for adapting the direction of search to a descent one. In this way, a decrease of the function value can be ensured at each iteration; and convergence to a local minimizer of the objective function from remote initial points can be achieved [14] (see the detailed description of [14] on how this can be applied to any adaptive gradient-based algorithms with individual step-sizes).

Based on the above theoretical discussion we propose in Figure 1 a simplified version of the algorithm that only depends on $E(t-1) < E(t)$; q is a reduction factor that is used to update the midpoint of the considered interval. The choice of q has an influence on the number of error function evaluations required to obtain an acceptable weight vector [15]. In practice, one iteration of the bisection method along each weight direction is recommended, as exact subminimization to obtain accurate approximations of the subminimizer in each direction requires significant computational efforts [10].

4. Experimental study

In this section, we evaluate the performance of the new method (ARPROP) on four pattern classification problems and compare it with the original Rprop [2] and the Improved Rprop (iRprop) proposed recently by Igel and Husken [3]. We have used well-studied problems from the UCI Repository of Machine Learning Databases [16], as well as problems studied extensively by other researchers in an attempt to reduce as much as possible biases introduced by the size of the weights space and the quality of the training data. We decided not to enhance the algorithms tested with add-on techniques for improving the classification success in the testing phase (i.e. the generalization ability of the trained neural network) as this would require introducing, and fine tuning or optimizing additional heuristics depending on the learning task.

4.1. Problems description

We have used four pattern classification problems from the UCI repository of machine learning database, namely cancer1, diabetes1, thyroid1, genes2, as described in the PROBEN1 benchmark collection [17]. In all experiments, we use the same parameters as suggested in [2], and feed-forward neural networks with sigmoid hidden and output nodes. The notation I-H-O is used to denote network architecture with I inputs, H hidden layer nodes and O outputs nodes. All results are based on 100 independent trials for each algorithm.

Moreover, the 100 random weight initializations are the same for all the learning algorithms. In all cases, the training and testing sets were created according to the guidelines of PROBEN1 (see [17] for details).

In order to analyze the statistical significance, we have implemented the Wilcoxon test as suggested in [18]. This is a nonparametric method that is considered an alternative to the *paired t*-test. This test assumes that there is information in the magnitudes of the differences between paired observations, as well as the signs. Firstly, we take the paired observations, we calculate the differences and then we rank them from smallest to largest by their absolute value. After adding all the ranks associated with positive and negative differences giving T_+ and T_- statistic respectively. Finally, the probability value associated with this statistic is found from the appropriate table. In our experiments, all statements refer to a significance level of 5%, which corresponds to $Z_c > 1.95$ for the convergence speed and $Z_{cr} > 1.95$ for the classification success.

4.1.1. The Cancer1 problem. This is a breast cancer diagnosis problem based on 9 inputs describing a tumour as a benign or malignant. The data set consists of 350 patterns. We have used a feed-forward neural network with 9-4-2-2 nodes as suggested in the PROBEN1 benchmark collection, [17], and in [3].

4.1.2. The Diabetes1 problem. The aim of this real-world classification task is to decide whether a Pima Indian individual is diabetes positive or not. We have 8 inputs representing personal data and results from a medical examination. The data set consists of 384 patterns. The PROBEN1 collection proposes several architectures for this problem, including one with 8-2-2-2 nodes. We decided to use this architecture as it was also suggested by others [3].

4.1.3. The Genes2 problem. It is a binary problem. The goal of this classification task is to decide, from a window of 60 DNA sequence elements (nucleotides), whether the middle is either an intron/exon boundary (a donor), or an exon/intron boundary (an acceptor), or none of these. Each nucleotide is encoded binary by two bipolar inputs (i.e -1 and +1). The data set consists of 1588 patterns. This data set was created based on the 'splice junction' problem dataset from the UCI repository of machine learning database. We have used a network with 120-4-2-3 nodes as suggested in the PROBEN1 benchmark collection.

4.1.4. The Thyroid1 problem. This problem is based on patient query data and patient examination data. The task is to decide whether the patient's thyroid has over function, normal function, or under function. The data set consists of 3600 patterns. We use the thyroid1, which is not a permutation of the original data, but retains the original order instead. We have used a network with 21-4-3 nodes, as suggested by [7].

4.2. Presentation of results

Convergence speed (learning speed) is a critical factor when we decide which algorithm to use. Classification success in testing (generalization performance) is another crucial factor. Below we present experimental results for both factors.

4.2.1. Cancer1. The results for this pattern classification problem are summarized in Table 1. The new algorithm performs significantly better than the other two methods. The differences between iRprop and Rprop are not important. In Table 1, we represent the average time ("Time", measured in secs spends to train a network), the classification success in testing ("Classification Success" or "Success", measured by the percentage of testing patterns that were classified correctly), and the convergence success in the training phase ("Convergence", measured by the percentage of simulation runs that converged to the error goal) for all algorithms. For each comparison we apply the Wilcoxon rank test to calculate the significance of the results; when $Z_c > 1.95$ or $Z_{cr} > 1.95$ then there is a remarkable improvement.

Judging from Table 1, we can notice that all methods converge to the error goal and there is no significant difference in the generalization performance. Nevertheless, the new algorithm is faster than the other first order methods. The percentage of improvement that the new algorithm achieved over Rprop and iRprop in terms of learning speed is 38.2% and 37.8% respectively. The results show slightly less classification success but, as discussed above, this is not statistically significant.

Cancer	**Average performace**			**Wilcoxon Test**	
Algorithm	Time (sec)	Classification Success (%)	Converge (%)	Time (Zc)	Success (Zcr)
Rprop	1.15	97.64	100	6.55	1.75
iRprop	1.13	97.63	100	6.75	1.54
ARPROP	0.70	97.60	100		

Table 1. Performance in the cancer1 problem

4.2.2. Diabetes1. In the diabetes classification problem both Rprop and iRprop behave similarly.

Cancer	**Average performace**			**Wilcoxon Test**	
Algorithm	Time (sec)	Classification Success (%)	Converge (%)	Time (Zc)	Success (Zcr)
Rprop	3.37	75.64	99	3.37	2.32
iRprop	3.32	75.67	99	4.20	2.30
ARPROP	1.09	75.78	100		

Table 2. Performance in the diabetes1 problem

Table 2 gives the average convergence speed and success (%) of the algorithms. The table also includes the results of the Wilcoxon Rank test. The increased convergence speed does not seem to affect the classification success of the ARPROP in testing. ARPROP is 67.5% and 67% faster than Rprop and iRprop, respectively.

4.2.3. Genes2. In Table 3, the performance of ARPROP is statistically significant when compared to Rprop and iRprop with respect to both learning time and classification success in testing. The overall improvement achieved by the ARPROP over Rprop and iRprop is 25% and 23.8% in terms of convergence speed.

Genes	Average performance			Wilcoxon Test	
Algorithm	Time (sec)	Classification Success (%)	Converge (%)	Time (Zc)	Success (Zcr)
Rprop	53.3	98.76	95	3.11	7.32
iRprop	52.4	98.76	94	2.52	7.32
ARPROP	39.9	100	100		

Table 3. Performance in the genes2 problem

4.2.4. Thyroid1. The comparative results of the three algorithms using an architecture 21-4-3 with one hidden layer are given in Table 4. The table gives the average training time, classification success in testing, and converge success, as well as the result of the Wilcoxon Rank Test for each algorithm. The new algorithm outperforms to other two algorithms both in time and testing success (the average improvement in learning speed achieved by ARPROP over Rprop and iRprop is 55.4% and 54.8% respectively).

Thyroid	Average			Wilcoxon Test	
Algorithm	Time (sec)	Classification Success(%)	Converge (%)	Time (Zc)	Success (Zcr)
Rprop	26.2	98.12	100	7.18	3.5
iRprop	25.8	98.12	100	6.84	3.5
ARPROP	11.6	98.23	100		

Table 4. Mean values for thyroid problem

5. Concluding remarks

It is widely accepted that the Rprop algorithm is one of the best performing first order learning algorithms for multilayer neural networks. In this paper we introduced an efficient improvement of the Rprop algorithm that is built on a firm theoretical basis. Furthermore we proposed three conditions that should be fulfilled by any algorithm that employs information of the sign of partial derivative to update the weights. The third condition (see Eq. 6) requires special treatment as it may lead the algorithm to converge to an undesired local minimum and was not considered in the algorithm model we presented in this paper. This issue will be addressed in a future work. We tested the new algorithm in four pattern classification problems from the PROBEN1 repository. It exhibits significantly better convergence speed than the Rprop and iRprop, a recently introduced modification of the Rprop algorithm. We currently test the performance of our method in other pattern recognition problems to fully explore its advantages and identify possible limitations.

6. References

[1] Rumelhart, D. E., Hinton, G.E, and Williams, R. J., "Learning internal representations by error propagation", In D. E Rumelhart and J. L. McClellend (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition*. Cambridge, MIT Press, 1986, pp. 318-362.

[2] M. Riedmiller, and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", Proceedings International Conference on Neural Networks, San Francisco, CA, 1993, pp. 586-591.

[3] C. Igel, and M. Husken, "Empirical evaluation of the improved Rprop learning algorithms", Neurocomputing, 50, 2003, pp. 105-123.

[4] L. Fang, and T. Li, "A globally optimal annealing learning algorithm for multilayer perceptrons with applications", in proc. AI'90: Austrtalian Joint Conf. Artificial Intell. Perth Australia: World Scientific, 1990, pp. 201-206.

[5] B.D. Fogel, J.L. Fogel, and W.V. Porto, "Evolving neural networks", Biol. cybern, 63, 1990, pp. 487-493.

[6] Z. Tang, and J.G. Koehler, "Deterministic global optimal FNN training algorithms", Neural Networks, 7, 1994, pp. 1405-1412.

[7] N.K. Treadgold, and T.D. Gedeon, "Simulated Annealing and Weight Decay in Adaptive Learning: The SARPROP Algorithm," in IEEE Transactions on Neural Networks,1998, pp. 662-668.

[8] Scales L.E. *Introduction to non-linear optimization*, MacMillan Publishers LTD, 1985, pp. 34-35.

[9] K. Sikorski, "Bisection is optimal", Numer. Math, 40, 1982, pp. 111-117.

[10] M.N. Vrahatis, G.D. Magoulas, and V.P. Plagianakos, "From linear to nonlinear iterative methods", Applied Numerical Mathematics, 45, 1, 2003, pp. 59-77.

[11] D. Young, Iterative methods for solving partial difference equations of elliptic type, Trans. Amer. Math. Soc.,1954, pp.92-111.

[12] Axelsson, O., *Iterative Solution Methods*, Cambridge Univ. Press, New York, 1996.

[13] Stewart, G.W., *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[14] G.D Magoulas, V.P. Plagianakos, and M.N. Vrahatis, "Globally convergent algorithms with local learning rates", IEEE Tr. Neural Networks, vol. 13, no. 3, 2002, pp. 774-779.

[15] G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis, "Improving the Convergence of the Backpropagation Algorithm Using Learning Rate Adaptation Methods," Neural Computation, 11, 1999, pp. 1769-1796.

[16] P.M. Murphy, and D.W. Aha, UCI Repository of machine learning databases Irvine, CA: University of California, Department of Information and Computer Science, 1994. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

[17] L. Prechelt, "PROBEN1-A set of benchmarks and benchmarking rules for neural network training algorithms", Technical report 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.

[18] Snedecor, G., and Cochran, W., *Statistical Methods*, Iowa State University Press, 8th edition. , 1989.