International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:1, No:7, 2007

Advanced Neural Network Learning Applied to Pulping Modeling

Z. Zainuddin, W. D. Wan Rosli, R. Lanouette, and S. Sathasivam

Abstract**—** This paper reports work done to improve the modeling of complex processes when only small experimental data sets are available. Neural networks are used to capture the nonlinear underlying phenomena contained in the data set and to partly eliminate the burden of having to specify completely the structure of the model. Two different types of neural networks were used for the application of pulping problem. A three layer feed forward neural networks, using the Preconditioned Conjugate Gradient (PCG) methods were used in this investigation. Preconditioning is a method to improve convergence by lowering the condition number and increasing the eigenvalues clustering. The idea is to solve the modified problem $M^{-1}Ax = M^{-1}b$ where M is a positive-definite preconditioner that is closely related to A. We mainly focused on Preconditioned Conjugate Gradient- based training methods which originated from optimization theory, namely Preconditioned Conjugate Gradient with Fletcher-Reeves Update (PCGF), Preconditioned Conjugate Gradient with Polak-Ribiere Update (PCGP) and Preconditioned Conjugate Gradient with Powell-Beale Restarts (PCGB). The behavior of the PCG methods in the simulations proved to be robust against phenomenon such as oscillations due to large step size.

Keywords**—**Convergence, Pulping Modeling, Neural Networks, Preconditioned Conjugate Gradient

I. Introduction

Pulping is a process that liberates fibers from the wood matrix which can be used for the production of paper. This can be done via mechanical, chemical or a combination of the two processes. This paper deals with the later process where wood chips are digested in closed vessel filled with a solution of sodium hydroxide and (or) sodium sulfite at a particular temperature. After for a period of time, the pressure from the vessel is relieved, the partially cooked chips are removed and subjected to a mechanical process in a refiner to obtain the final pulp which was used to make laboratory paper sheets, followed by evaluation using the Canadian Pulp and Paper Association Standards (CPPA).

Because of the highly competitive nature of the pulp and paper industry, it is important that all processes are operated as close as possible to the optimal conditions. In this respect, a reliable mechanistic or empirical model is required. Compounding the problem is the fact that it is usually very expensive to derive a comprehensive model and hence one has to rely only on a limited number of experiments. The challenge is thus to derive a good model out of this small number of data sets [1]. One methodology to circumvent this problem is to use neural network.

Neural networks are most widely used to solve problems because it resembles more closely to what scientists are used to manipulate since it is simply a nonlinear model that maps the input vector onto an output vector. Neural networks are more flexible. It can be manipulated using standard analysis tools such as post-training analysis (postreg) to test the generalization capability. Furthermore, neural networks can be developed using software packages compared with other models which are developed using mathematical analysis mainly.

II. Algorithms

The backpropagation method consists of three main layers-input layers, output layers and hidden layers. The input nodes constitute the first layer and the output nodes constitute the output layer while the remaining nodes constitute hidden layers of the network. The input vector is presented to the input layer and the signals are propagated forward to the first hidden layer; the resulting outputs of the first hidden layer are in turn applied to the next hidden layer and the same procedure continues for the rest of the network.

The error signal $e_j(n)$ at the output of neuron j at iteration n is defined by

$$e_j(n) = d_j(n) - y_j(n) \tag{1}$$

where $d_j(n)$ and $y_j(n)$ is the desired and the actual response of neuron j at iteration n respectively.

Hence the instantaneous value $\xi(n)$ of the sum of squared error over all neurons can be written as

$$\xi(n) = \frac{1}{2}\sum_{j\in C} e_j^{\,2}\ (n) \tag{2}$$

where C indicates all the neurons in the output layer of the network. The average squared error over the total number of patterns N is given

Manuscript received December 6, 2004. This work is supported in part by the Malaysian IRPA Grant No. 190/9606/3104.

Z. Zainuddin and S. Sathasivam are with the School of Mathematical Sciences, Universiti Sains Malaysia, 11800, Penang, Malaysia (e-mail: zarita@cs.usm.my; saratha_sathasivam@hotmail.com).

W. D. Wan Rosli is with the School of Industrial Technology, Universiti Sains Malaysia, 11800, Penang, Malaysia (phone: (6)04-6533813; fax: (6)04-6573678; e-mail: wanrosli@ usm.my).

R. Lanouette is with the Centre de Recherche en Pates et Papiers, Universite du Quebec a Trois-Rivieres, P.O. Box 500, Trois-Rivieres, Quebec, Canada G9A 5H7 (robert_lanouette@uqtr.uquebec.ca).

$$\xi_{av} = \frac{1}{N}\sum_{n=1}^{N}\xi(n) \tag{3}$$

The objective of the learning process is to adjust the free parameters (i.e. synaptic weights and thresholds) of the network so as to minimize ξ_{av} [2].

A. Conjugate Gradient (CG) Methods

In optimization theory, the conjugate gradient method has been known since Fletcher and Reeves [3]. Leonard and Kramer [4] introduced the original Fletcher – Reeves algorithm in the field of neural network research. The conjugate direction is very effective. We can minimize a quadratic function exactly by searching along the eigenvectors of the Hessian matrix, since they form the principal axes of the function contours.

Let us assume that the error function is quadratic in **w**, that is, it can be approximated to a quadratic function as

$$E(\mathbf{w}) = c - \mathbf{b}^{\mathbf{T}}\mathbf{w} + \frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{A}\mathbf{w} \tag{4}$$

where **A** is a symmetric positive definite matrix. Let **p**(n) denotes the direction vector at iteration n of the algorithm. Then the weight vector of the network is updated in accordance with the rule

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{p}(n) \tag{5}$$

where $\eta(n)$ is the learning-rate parameter.

Suppose the initial direction of minimization, which is started at **w**(0) is **p**(0) which is set equal to the negative gradient vector **g**(n) at the initial point n=0; that is,

$$\mathbf{p}(0) = -\mathbf{g}(0) \tag{6}$$

A line minimization in the direction of **p**(0) results in a gradient at **w**(1) perpendicular to **p**(0). In general,

$$\mathbf{p}^{\mathbf{T}}(n)\mathbf{g}(n+1) = 0 \tag{7}$$

Because we do not want to spoil this minimization step in subsequent minimizations, the gradient of subsequent points of minimization must also be perpendicular to **p**(n):

$$\mathbf{p}^{\mathbf{T}}(n)\mathbf{g}(n+2) = 0 \tag{8}$$

Therefore, with Eq. (7) and Eq. (8),

$$\mathbf{p}^{\mathbf{T}}(n)(\mathbf{g}(n+2) - \mathbf{g}(n+1)) = 0 \tag{9}$$

Now, **g**(n+2) – **g**(n+1) is the change in the gradient as we move from **w**(n+1) to **w**(n+2). From Eq. (7), the gradient of E at **w**(n) can be found to be

$$\mathbf{g}(n) = \mathbf{A}\mathbf{w}(n) - \mathbf{b} \tag{10}$$

Therefore,

$$0 = \mathbf{p}^{T}(n)\mathbf{A}\mathbf{p}(n+1)\eta(n+1) \tag{11}$$

or

$$\mathbf{p}^{T}(n)\mathbf{A}\mathbf{p}(n+1) = 0 \tag{12}$$

When Eq. (11) holds for two vectors **p**(n) and **p**(n+1), these vectors are said to be conjugate.

There are various rules to determine $\beta(n)$ in order to ensure conjugacy of **p**(n) and **p**(n+1); two alternate rules are the following:

- The Fletcher-Reeves formula [3]

$$\beta(n) = \frac{\mathbf{g}^{T}(n+1)\mathbf{g}(n+1)}{\mathbf{g}^{T}(n)\mathbf{g}(n)} \tag{13}$$

- The Polak-Ribiere formula [5]:

$$\beta(n) = \frac{\mathbf{g}^{T}(n+1)[\mathbf{g}(n+1) - \mathbf{g}(n)]}{\mathbf{g}^{T}(n)\mathbf{g}(n)} \tag{14}$$

B. Powell-Beale Restart Procedure

The CG method can be improved by periodically resetting the search direction to the negative of the gradient. Since this procedure is ineffective, a restarting method that does not abandon the second derivative information is needed. One such reset method has been proposed by [6]. For this technique, we will restart if there is very little orthogonality left between the current gradient and the previous gradient. This is tested with the following inequality

$$\left\|\mathbf{g}^{T}(n-1)\mathbf{g}(n)\right\| \geq 0.2\left\|\mathbf{g}(n)\right\|^{2} \tag{15}$$

If this condition is satisfied, the search direction is reset to the negative of the gradient.

III. Preconditioned Conjugate Gradient (PCG) Methods

Performance of the CG method is generally very sensitive to round off in the computations that may destroy the mutual conjugacy property. Preconditioning is a technique for improving the condition number of a matrix. Intuitively, preconditioning is an attempt to stretch the quadratic form to make it appear more spherical so that the eigenvalues are closer to each other. If the eigenvalues are clustered, the condition number of a matrix will be less than 1[7]. Suppose that **M** is a symmetric, positive-definite matrix that approximates **A** . We can solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ indirectly by solving

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} \tag{16}$$

If the eigenvalues of $\mathbf{M}^{-1}\mathbf{A}$ are better clustered than those of **A**, we can iteratively solve equation (16) more quickly than the original problem. Due to the fact that most of the data sets are rectangular matrices, the QR factorization has been chosen to define the preconditioner.

A preconditioned Conjugate Gradient algorithm consists of following steps:

1. Select the first search direction to be the negative of the gradient,

$$\mathbf{p}(0) = -\mathbf{M}^{-1}\mathbf{g}(0) \tag{17}$$

2. Take a step selected learning rate η_n to minimize the function along the search direction.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{p}(n) \tag{18}$$

3. Select the next search direction according to

$$\mathbf{p}(n+1) = \mathbf{M}^{-1} * [-\mathbf{g}(n+1)] + \beta(n)\mathbf{p}(n) \tag{19}$$

where

$$\beta(n) = \frac{\mathbf{g}^T(n+1)\mathbf{M}^{-1}\mathbf{g}(n+1)}{\mathbf{g}^T(n)\mathbf{M}^{-1}\mathbf{g}(n)} \quad (20)$$

due to Preconditioned Fletcher and Reeves, and

$$\beta(n) = \frac{\mathbf{g}^T(n+1)\mathbf{M}^{-1}[\mathbf{g}(n+1)-\mathbf{g}(n)]}{\mathbf{g}^T(n)\mathbf{M}^{-1}\mathbf{g}(n)} \quad (21)$$

due to Preconditioned Polak Ribiere.

4. If the algorithm has not converged, return to step (2)

Preconditioned Powell-Beale Restart Procedure

The Preconditioned Conjugate Gradient method can be improved by periodically resetting the search direction to the negative of the gradient.

For this technique, we will restart if there is very little orthogonality left between the current gradient and the previous gradient. This is tested with the following inequality

$$\left\|\mathbf{g}^T(n-1)*\mathbf{M}^{-1}*\mathbf{g}(n)\right\| \geq 0.2\left\|\mathbf{M}^{-1}*\mathbf{g}(n)\right\|^2 \quad (22)$$

If this condition is satisfied, the search direction is reset to the negative of the gradient.

IV. Application to the Pulping Problem

The Neural Network Toolbox in Matlab R12 version 6 [8] was used to simulate the training of this data set. The number of hidden nodes is determined by using the Blum's "rules of thumb" [9]. Due to the very limited number of experimental data points, the choice of the architecture of the network becomes important. Two different types of neural networks were used for modeling this problem:

4-6-5 network with 15 training vectors, 10 testing vectors

5-6-4 network with 15 training vectors, 10 testing vectors

Four inputs/outputs correspond to temperature, sodium hydroxide change, sodium sulfite change and plate gap. The five output / input nodes correspond to specific energy, freeness, breaking length, tear index and ISO brightness [1]. The training has been continued until 30,000 epochs reached within 0.01. This is because pulping is actually a functional approximation problem. Due to this, the network needs more time to learn and adapt to the data sets.

Discussion of Results

As can be observed, the PCG algorithms were able to provide enhanced convergence and accelerated training. From the experimental results for network I from Table I, it can be seen that an improvement of over 90% was observed for the PCG methods when compared with the BP, which is significantly better than the CG methods. Conversely, for network II (Table II), we observed that the PCG methods improved the convergence rate by a speed-up of between 75% and 94% compared with the BP method. Evaluating the performance of the network with new data from the testing set, that were not used in the training, the correlation coefficient, R, between the network output and the corresponding targets (Table III and Table IV), is near to 1 for the PCG methods, indicating good correlation of the data. This indicates that the corresponding targets explain the variation in the network outputs very well.

Significant improvements over BP method are provided by CG and PCG methods. It is observed that the convergence of CG is smaller than PCG, even after 30,000 of the learning iterations, which takes nearly the same computation time. It appears that, when only a few points (15 training vectors) are available to model the network, the solution seems to get stuck at local minima values for the CG algorithms. This implies that, the choice of weights for the CG methods, are not appropriate.

V. Conclusions

The fast and robust convergence of PCG, and the failure of CG methods to converge within 30,000 epochs, demonstrates the ability of the preconditioner to solve very complex learning tasks in large networks, where a suitable solution in weight space is difficult to find.

Extensive computer simulations and performance comparisons with BP algorithm and CG methods have demonstrated the fast learning speed and high convergence accuracy of the considered learning algorithms. The corresponding targets explain the variation in the network outputs very well. The generalization properties using PCG methods for all trained neural networks have been found to be satisfactory from the application point of view. Overall, it can be concluded that modeling with neural networks can greatly improve the study of pulping problem for which experimental data sets are too expensive to obtain.

References

[1] Lanoutte, R. and Laperriere, L. Evaluation of Sugar Maple (Acer Saccharum) in high yield pulping processes. *Pulp and Paper Research Center, University du Quebec*, 2002.

[2] Haykin, S. Neural Networks: *A Comprehensive Foundation (Second Edition),* New Jersey, Prentice Hall, 1999.

[3] Fletcher, R. and Reeves, C.M. Function minimization by conjugate gradients. *Computer Journal*, 1964, 7**,** 149-154.

[4] Leonard, J. and Kramer, M.A. Improvement to the back-propagation algorithm for training neural networks, *Computers and Chemical Engineering*, 1990, 14(3), 337-341.

[5] Polak, E. *Computational methods in optimization*, New York, Academic press, 1971.

[6] Powell, M. J. D. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 1977, 12, 241-254.

[7] Sathasivam, S. Optimization Methods In Training Neural Networks**.** Master Thesis, University Science Malaysia, 2003.

[8] Demuth, H. and Bearle, M. *Neural Network Toolbox*, New York, Mathworks Inc, 2002.

[9] Blum, A. *Neural Networks in C++,* New York, John Wiley & Sons, 1992.

TABLE I
THE SIMULATION RESULTS FOR THE PULPING OF SUGAR MAPLE USING CONJUGATE GRADIENT, PRECONDITIONED CONJUGATE GRADIENT AND BACKPROPAGATION (BP) FOR NETWORK I

Algorithm		No. of iterations	Mean squared error (MSE)	% improvement over BP (%)	No. of cases not converged
Backpropagation	TRAINGD	387,000	0.105507	-	0
		593,000	0.0780839		0
		471,000	0.0860886		0
		407,000	0.0927855		0
		303,000	0.132488		0
		439,000	0.0902982		0
Conjugate Gradient	TRAINCGF	30,000	0.105507	92.25	3
	TRAINCGP	30,000	0.0780839	94.94	3
	TRAINCGB	30,000	0.0860886	93.63	0
Preconditioned Conjugate Gradient	TRAINPCGF	30,000	0.0927855	92.63	0
	TRAINPCGP	30,000	0.132488	90.10	0
	TRAINPCGB	30,000	0.0902982	93.17	0

TABLE II
THE SIMULATION RESULTS FOR THE PULPING OF SUGAR MAPLE USING CONJUGATE GRADIENT, PRECONDITIONED CONJUGATE GRADIENT AND BACKPROPAGATION (BP) FOR NETWORK II

Algorithm		No. of iterations	Mean squared error (MSE)	% improvement over BP (%)	No. of cases not converged
Backpropagation	TRAINGD	281,000	0.150492	-	0
		139,000	0.242895		0
		317,000	0.137458		0
		379,000	0.133812		0
		121,000	0.275036		0
		490,000	0.117163		0
Conjugate Gradient	TRAINCGF	30,000	0.150492	89.32	4
	TRAINCGP	30,000	0.242895	78.42	7
	TRAINCGB	30,000	0.137458	90.53	0
Preconditioned Conjugate Gradient	TRAINPCGF	30,000	0.133812	92.08	0
	TRAINPCGP	30,000	0.275036	75.21	0
	TRAINPCGB	30,000	0.117163	93.68	0

TABLE III
THE REGRESSION RESULTS FOR THE PULPING OF SUGAR MAPLE PROBLEM (NETWORK I)

Algorithm	m	b	R
Fletcher Reeves	0.72	0.103	0.849
Polak Ribiere	0.795	-0.0178	0.917
Powell Beale	0.802	0.00738	0.954
Preconditioned Fletcher Reeves	0.786	-0.0527	0.949
Preconditioned Polak Ribiere	0.744	-0.0319	0.945
Preconditioned Powell Beale	0.73	-0.0336	0.966

TABLE IV
THE REGRESSION RESULTS FOR THE PULPING OF SUGAR MAPLE PROBLEM (NETWORK II)

Algorithm	m	b	R
Fletcher Reeves	0.00046	0.999	0.85
Polak Ribiere	0.567	0.267	0.71
Powell Beale	0.741	0.133	0.903
Preconditioned Fletcher Reeves	1.87	-0.873	0.918
Preconditioned Polak Ribiere	0.714	-0.00576	0.885
Preconditioned Powell Beale	0.0666	0.756	0.946