TRAINING OF PERCEPTRON NEURAL NETWORK USING PIECEWISE LINEAR ACTIVATION FUNCTION

Maria P. Barbarosou[1] and Nicholas G. Maratos[2]

[1]M. P. Barbarosou is with the Technological Educational Institute of Piraeus, 250 Thivon and P. Ralli Ave., GR 12244, Egaleo, Greece, and also with the Hellenic Air-Force Academy, Dekelia Air Force Base, GR 1010, Tatoi, Greece, Fax:+302105450962, E-mail: mbarbar@teipir.gr.

[2]N.G. Maratos is with the School of Electrical and Computer Engineering, National Technical University of Athens, 9 Iroon Polytechniou St., GR 15773, Athens, Greece, Fax:+302107722281, E-mail: maratos@ece.ntua.gr.

Abstract

A new Perceptron training algorithm is presented, which employs the piecewise linear activation function and the sum of squared differences error function over the entire training set.

The most commonly used activation functions are continuously differentiable such as the logistic sigmoid function, the hyperbolic-tangent and the arc-tangent. The differentiable activation functions allow gradient-based optimization algorithms to be applied to the minimization of the error. This algorithm is based on the following approach: the activation function is approximated by its linearization near the current point, hence the error function becomes quadratic and the corresponding constraint quadratic program is solved by an active set method.

The performance of the new algorithm was compared with recently reported methods. Numerical results indicate that the proposed algorithm is more efficient in terms of both, its convergence properties and the residual value of the error function.

1. INTRODUCTION

Neural network (NN) models are of board interest to researchers in the recent years, as its applications have flooded many areas.

The activation function is a key factor in the NN structure [1]. The most fuzzy applications use a piecewise linear function (PLF) [2] for activation of neurons, because of its easy handling from their limited computational resources. NNs that use PLFs as activation function is known as piecewise linear NNs (PWL NNs). These cannot be trained by a gradient based optimization method because the lack of continuous derivatives. Hence several training algorithms of PWL NNs have been developed. For example the algorithm presented in [3] is used to NNs that employs the absolute value as activation function. In addition, in [4] is proposed a basis exchange algorithm.

In this paper a new algorithm for training a PWL NN is proposed. The main stage of the method is the modification of the training problem to a quadratic programming. This process is briefly described in Section 2. The main steps of the algorithm are presented in Section 3. Section 4 contains numerical results. The paper is concluded in Section 5.

2. MODIFICATION OF PERCEPTRON TRAINING TO A CONSTRAINED QUADRATIC OPTIMIZATION PROBLEM

A graphical representation of a single hidden layer Perceptron with a single output is shown in Figure 1. The hidden layer consists of *n* neurons.

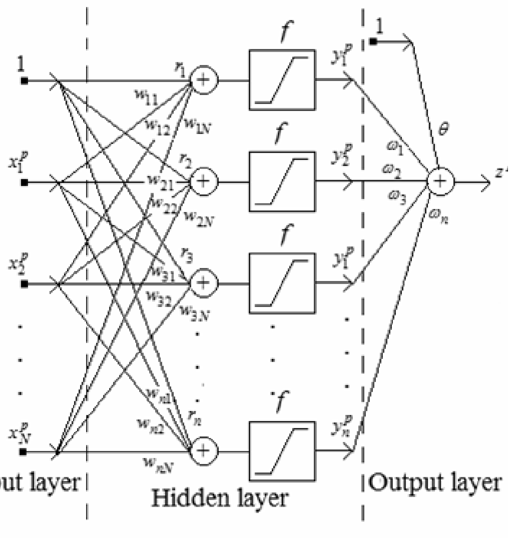

Fig. 1. Graphical depiction of a single hidden layer Peceptron

In Fig. 1 each neuron passes its input which is the weighted sum of the inputs of the network plus the input bias term, through its activation function f and presents the result to its output. The proposed method adopts the 1-dim piecewise linear function (PLF) as activation of neurons:

$$f: R \to [-L\, L],\ f(y,\, L) = \begin{cases} L & y \ge L \\ y & |y| < L \\ -L & y \le -L \end{cases} \quad (1)$$

where $L = 1$, its threshold. Obviously, it is nondefferential and bounded between -1 and 1. Its graph is given in Fig. 2. It has three linear pieces which are locally differentiable and two corners. This form of the activation function may be viewed as an approximation to a nonlinear amplifier.

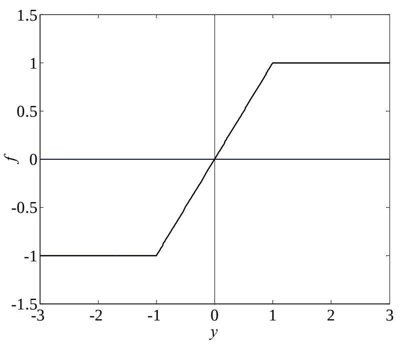

Fig. 2. 1-dim PLF bounded between -1 and 1

Hence the output of the *i* th neuron, y_i^p, corresponding to the input of the *p*th training data x^p, is computed as:

$$y_i^p = f\left(\sum_{j=1}^{N} w_{ij} x_j^p + r_i,\ 1\right), \text{ for } i = [1, \dots, n] \quad (2)$$

where *n* is the number of neurons, x_j^p is the *j* th element of the N-dim vector x^p, that is $x^p = [x_1^p, \dots, x_j^p, \dots x_N^p]^{\mathrm{T}}$, r_i the input bias of the *i* th neuron and w_{ij} the weight connecting the *j* th input to the *i* th neuron. By considering $w_i = [w_{i1}, \dots, w_{iN}]^{\mathrm{T}}$, (2) is written equivalently as: $y_i^p = f\left(w_i^{\mathrm{T}} x^p + r_i,\ 1\right)$ for $i = [1, \dots, n]$.

The output of the network, z^p, corresponding to the x^p, is the weighted linear combination of the outputs of the neurons plus the output bias:

$$z^p = \sum_{i=1}^{n} \omega_i f\left(w_i^{\mathrm{T}} x^p + r_i,\ 1\right) + \theta = \sum_{i=1}^{n} \omega_i y_i^p + \theta \quad (3)$$

where θ is the output bias and ω_i the weight connecting the *i* th neuron to the output layer.

For every given training sample, let the *p* th one, the output of the network, z^p, differs from the target (desired) value, t^p, by $(t^p - z^p)$. The purpose of the proposed method is to determine the coefficients w_{ij}, ω_i, r_i, and θ in such a way that the summed over all training samples squared error, between the actual and the target output, to be minimized. Hence the total error is selected to be $E = \frac{1}{2}\sum_{p=1}^{M}\left(t^p - z^p\right)^2$ which since (3) becomes:

$$E(\theta, r, W, \omega) = \frac{1}{2}\sum_{p=1}^{M}\left(t^p - \theta - \sum_{i=1}^{n} \omega_i f\left(w_i^{\mathrm{T}} x^p + r_i,\ 1\right)\right)^2 \quad (4)$$

where M is the number of training samples, r is the vector of input bias, that is $r = [r_1, \dots, r_n]$, W is the $n \times N$ matrix of the weights connecting the inputs to the neurons that is $W = [w_{ij}]_{i=1,\dots,n}^{j=1,\dots,N}$ and ω is the vector of weights connecting the outputs of neurons and the output layer of the network, that is $\omega = [\omega_1, \dots, \omega_n]$.

The following property of the PLF is an easy consequence of its definition given by (1):

$$af(y, L) = f(ay, |a|L),\ \forall a \in R \quad (5)$$

Take into account (5), (4) becomes:

$$E(\theta, r, W, \omega) = \frac{1}{2}\sum_{p=1}^{m}\left(t^p - \theta - \sum_{i=1}^{n} f\left(\omega_i w_i^T x^p + \omega_i r_i,\ |\omega_i|\right)\right)^2 \quad (6)$$

To simplify the formula (6), the following transformation is applied:

$$\zeta_i = |\omega_i|,\ b_i^{\mathrm{T}} = \omega_i w_i^{\mathrm{T}},\ q_i = \omega_i r_i,\ \forall i = 1, \dots, n \quad (7)$$

Hence the error function (6) is written:

$$\hat{E}(\theta, q, B, \zeta) = \frac{1}{2}\sum_{p=1}^{m}\left(t^p - \theta - \sum_{i=1}^{n} f\left(b_i^{\mathrm{T}} x^p + q_i,\ \zeta_i\right)\right)^2 \quad (8)$$

where $\zeta = [\zeta_1, \dots, \zeta_n]$, $B = [b_1, \dots, b_n]$ and $q = [q_1, \dots, q_n]$. So the training of NN whose layout is pictured in Figure 1, is modified to the following optimization problem with inequalities constraints:

$$\min_{\theta,q,B,\zeta} \left\{ \hat{E}(\theta, q, B, \zeta): \quad \zeta_i \geq 0, i = 1,\dots,n \right\} \quad (9)$$

The basic idea of the proposed approach to carry out the optimization process for the problem (9) is the following: At each algorithm iteration and $\forall(i,p)$ the PWF in problem (9) is approximated by its linearization at the current point. In doing so the problem (9) is modified to a constrained quadratic optimization problem. It is remarked that, to be valid the linearization has to be restricted within the linear piece of the PWF where its current value belongs. The validity of the linearization, is assured by setting extra constrains. The resulting quadratic problem is solved via an active set method [5]. The original weights and bias of NN can be obtained by applying the inverse of the transformation (7) on the solution.

3. OVERVIEW OF THE ALGORITHM

The outline of the proposed algorithm is as follows:

Initialization

Input:

1) $\{(x^1, t^1), \dots (x^M, t^M)\}$, the training set,

2) n : the number of neurons,

3) ε : the threshold for stopping criterion,

4) $u^o \overset{def}{=} (\theta^o, q^o, B^o, \zeta^o)$, the initial point such that $b_i^{o\,\mathrm{T}} x^p + q_i^o \neq \pm\zeta_i^o, \forall\{i,p\}$.

k iteration

Step 1. Let $u^k = (\theta^k, q^k, B^k, \zeta^k)$ be the current point. Compute $f\left(b_i^{k\,\mathrm{T}} x^p + q_i^k,\ \zeta_i^k \right), \forall\{i,p\}$.

Step 2. Modify the problem (9) to the corresponding quadratic substituting $\forall(i,p)$ the PLF for its linearization and setting the proper extra constraints, so that the linearizations to be valid.

Step 3. Apply the active set quadratic programming algorithm. Firstly, determine the feasible descent direction, d^*. If $\|d^*\| < \varepsilon$ the algorithm ends, otherwise determine the maximum step length a, in this direction, for point $u^{k+1} = u^k + a \times d^*$ to be feasible.

Step 4. Set $k = k+1$ and go back to Step 1. (For some $\{i,p\}$ at the new point u^k the PLF attains a corner of its graph. In these cases the linearization consider the other linear piece. As a result both the objective function and the extra constraints of the quadratic problem change at each iteration of the algorithm).

[6] provides a detailed description of both the modification process and the relative algorithm.

4. SIMULATION RESULTS

Two benchmark problems were selected from [7]. The simulation results confirm the effectiveness of the algorithm in terms of both accuracy and speed. The algorithm was developed using Matlab.

4.1. 1-dim Function approximation

The desired function is the following:

$$g(x) = 0.5 \sin \frac{2\pi\, x}{10} + 0.5 \left(\sin \frac{2\pi\, x}{10} \right)^2$$

As in [7], it was used 20 $(x, f(x))$-samples with x in $(0, 1)$ randomly chosen, as the training set of two neurons NN. The new algorithm run 50 times using each time different random starting weights and bias in $(0, 1)$. All times the algorithm was reaching the termination after 4 iterations and the final sum squared error $SSE = \sum_{p=1}^{20} (g(x^p) - z^p)^2$ was 6×10^{-4}. These are better performances than most in [7].

Furthermore, the algorithm is tested to approximate g over a larger domain. So, the algorithm used 20 random samples within $(0, 9)$, as the training set of 3 neurons NN and is tested for 50 different random starting weights and bias, in $(0, 1)$. The final SSE fluctuated between 1.9×10^{-4} and 2.3×10^{-2} after 19 and 39 iterations correspondingly. In Figure 1 it is shown the approximation of g from the output of NN with the best performance.

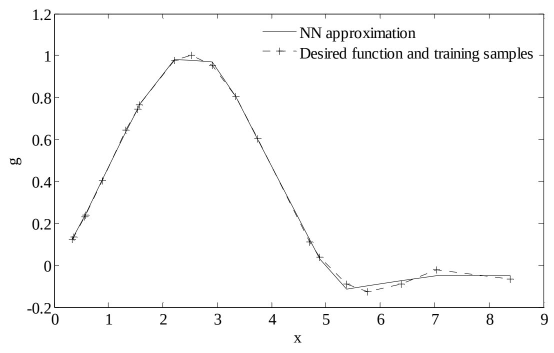

Fig. 3. The output of NN after training with the new algorithm

4.2. 2-dim Exclusive OR problem

In this example a two neurons NN is trained to output a 1, when its input is (0,0) or (1,1), and a 0, when its input is (0,1) or (1,0). The new algorithm run 50 times using each time different random starting weights and bias in $(0,1)$. All times the algorithm was reaching the termination after 10 iterations and the final SSE was 2.5×10^{-5}. These are by far better performances than most in [7].

5. CONCLUSION

A new training algorithm for a single layer NN with a single layer output is introduced in this work. Making use of PLF as activation of neurons, it modifies the training problem to a constrained quadratic optimization problem. Numerical results confirm its effectiveness compared to other algorithms.

References

[1] A. Ngaopitakaul and A. Kunakorn "Selection of proper activation functions in backpropagation neural networks algorithm for transformer internal fault locations", International Journal of Computer and Network Security (IJCNS), vol. 1, 2009, pp. 47-55.

[2] W. Eppler, "Piecewise linear networks (PLN) for process control, Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference, 2001.

[3] J. Lin and R. Unbehauen, "Canonical piecewise-linear networks", IEEE Trans. On Neural Networks, vol.6, 1995,pp. 43-50.

[4] E. F. Gad, et al., "A new Algorithm for learning in piecewise-linear neural network", Neural Networks, vol. 13, 2000, pp. 485-505.

[5] D. G. Luenberger, "Linear and Nonlinear Programming", Addison Wesley Publishing Company, Canada, 1989.

[6] Μ. Π. Μπαρμπαρόσου, "Νέα αναλογικά νευρωνικά δίκτυα και νέες μέθοδοι εκπαίδευσης νευρωνικών δικτύων», Διδακτορική Διατριβή, ΕΜΠ, 2005.

[7] A. Bortoletti, et al. "A new class of quasi-Newtonian methods for optimal learning in MLP-networks", IEEE Trans. On Neural Networks, vol. 14, 2003, pp. 263-273.