A Comprehensive Evaluation of Weight Growth and Weight Elimination Methods Using the Tangent Plane Algorithm

P May
K College,
Brook Street, Tonbridge,
Kent, UK

E Zhou
Engineering, Sports and Sciences
Academic Group,
University of Bolton, UK

C. W. Lee
Engineering, Sports and Sciences
Academic Group
University of Bolton, UK

***Abstract*—The tangent plane algorithm is a fast sequential learning method for multilayered feedforward neural networks that accepts almost zero initial conditions for the connection weights with the expectation that only the minimum number of weights will be activated. However, the inclusion of a tendency to move away from the origin in weight space can lead to large weights that are harmful to generalization. This paper evaluates two techniques used to limit the size of the weights, weight growing and weight elimination, in the tangent plane algorithm. Comparative tests were carried out using the Extreme Learning Machine which is a fast global minimiser giving good generalization. Experimental results show that the generalization performance of the tangent plane algorithm with weight elimination is at least as good as the ELM algorithm making it a suitable alternative for problems that involve time varying data such as EEG and ECG signals.**

***Keywords*—neural networks; backpropagation; generalization; tangent plane; weight elimination; extreme learning machine**

I. Introduction

In Lee [1] an algorithm was described for supervised training in multilayered feedforward neural networks giving faster convergence and improved generalization relative to the gradient descent backpropagation algorithm. This tangent plane algorithm starts the training with the connection weights set to values close to zero in the expectation that the minimum weights necessary will be activated.

The results based on two real world datasets indicated that the tangent plane algorithm gives improved generalization over a range of network sizes and that it is robust with respect to the choice of its internal parameters.

Despite the success of the tangent plane algorithm there is, however, strong evidence to suggest that growing the weights to assume large values can actually hurt generalization in different ways. Excessively large weights feeding into output units can cause wild outputs far beyond the range of the data if an output activation function is not included. To put it another way, large weights can cause excessively large variances in the output. According to Bartlett [2], the size of the weights is more important than the number of weights in determining good generalization. This poses the following question: can we modify this algorithm so that it discourages the formation of weights with large values? Further, can the algorithm encourage weights with small values to decay rapidly to zero thus producing a network having the optimum size for good generalization?

Weight decay is a subset of regularization methods. The principal idea of weight decay is to penalize connection weights with small values so that the network removes the superfluous weights itself. The simplest method is to subtract a small proportion of a weight after it has been updated [3]. This is equivalent to adding a penalty term $\sum_{ji} w_{ji}^2$ to the objective function and performing gradient descent on the resulting total error. Unfortunately this method penalizes more of the w_{ji}'s than necessary whilst keeping the relative importance of the weights unchanged. This can be cured by using a different penalty term, $\sum_{ji} w_{ji}^2 / (1 + w_{ji}^2)$, so that the small w_{ji}'s decay faster than the larger ones [4]. Williams [5] proposed yet another type of penalty function which is proportional to the logarithm of the l_1 norm of the weights,

$\sum_{ji} | w_{ji} |$. It was shown in [5] that using this penalty term is more appropriate for internal weights than weight decay. Hoyer [6] proposed a sparseness measure based on the l_1 norm and l_2 norm of weights. Experiments with Hoyer's method indicate that it performs well in comparison with weight decay and weight elimination. A further refinement involves using a mixed norm penalty term [7]. In this procedure the l_1 norm of the weight vector is minimised subject to the constraint that the l_2 norm equals unity.

II. Objectives

The principal objective of this paper is to describe an alternative strategy for improving generalization in neural networks trained using the tangent plane algorithm. In the newly developed algorithm, the training is started from arbitrary initial conditions and the inactive weights in the network encouraged decaying to zero by using the weight elimination procedure.

Unlike other implementations of weight elimination procedures [4, 6, 8 - 9], the method used here is built into the geometry used in the derivation of the algorithm. A secondary objective is to compare the newly developed algorithm with the extreme learning machine [10], which obtains the least squares solution with the minimum training error and minimum norm of weights.

III. Derivation of the Algorithm

In Lee [1] an algorithm is described that accepts almost zero staring conditions for the connection weights, and which moves away from the origin in a direction indicated by the training data with the expectation that only the minimum weights would be activated. This tangent plane algorithm uses the target values of the training data to define a $(n-1)$ surface in weight space R^n. The weights are adjusted by moving from the current position to a point near to the foot of the perpendicular to the tangent plane to this surface, but displaced somewhat in the direction away from the origin, on the expectation that the smaller the distance moved from the foot of the perpendicular the less disturbance there will be to the previous learning.

Two enhancements are made to the tangent plane algorithm to obtain the improved tangent plane algorithm referred to as iTPA. Firstly, a directional movement vector is introduced into the training process to push the movement in weight space towards the origin.

This movement vector simulates weight decay which is known to have a beneficial effect on generalization in backpropagation learning. Secondly, the directional vector is further modified to give a heavier weighting to weights with small weight values to avoid penalizing more of the weights than necessary; one large weight costs much more than many smaller ones. A high degree polynomial term is used to select the proportion of weights for pruning. This term can be adjusted so that a weight decay procedure is implemented or refined in a way that specific weights are removed by causing them to decay more rapidly to zero.

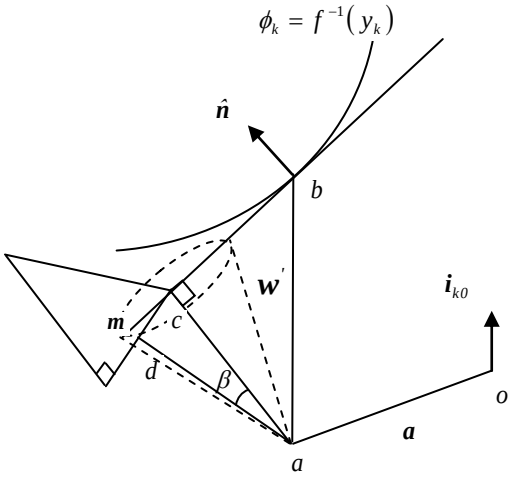

Fig. 1 Movement from the present position a to the point d inclined at an angle β to the perpendicular from a to the tangent plane to the constraint surface $\phi_k = f^{-1}(y_k)$ at point b in the weight space R^n. The vector m represents the orthogonal projection of the weight elimination vector w' orthogonally onto the normal n to the constraint surface at point b

The method assumes a feed-forward neural network of units $\{u_j\}$, where the connection between u_i and u_j is mediated by w_{ji}. ϕ_j and θ_j denote the input and output of u_j, so that $\theta_j = f(\phi_j)$, and $\phi_j = \sum_i w_{ji}\theta_i$. The single output unit u_k is trained to mimic the target value y_k. Let n denote the number of weights in the network. For a given set of inputs we can consider ϕ_k to be a function of the weights $\phi_k : R \rightarrow R^n$. The tangent plane algorithm adjusts the weights by moving along the line at an angle β to the perpendicular from the current position a to the tangent plane to the surface $\phi_k = f^{-1}(y_k)$, on the side of the perpendicular away from the origin (see Fig. 1).

Let $a = \sum_{ji} w'_{ji}\, i_{ji}$ be the current values of the weights, where i_{ji} is a unit vector in the direction of the w_{ji} axis. Use the equation $f^{-1}(y_k) = w_{k0} + \sum_{i\neq 0} w_{ki}\theta_i$ to find a value, w''_{k0}, for the bias weight w_{k0} from the values w_{ji} of the other weights, so that the surface $\phi_k = f^{-1}(y_k)$ contains the point

$b = w''_{k0}\, i_{k0} + \sum_{j,i\neq k,0} w'_{ji}\, i_{ji}$. Now, if we use the equation $f^{-1}(y_k) = w''_{k0} + \sum_{i\neq 0} w'_{ki}\theta_i$ and $f^{-1}(\theta_k) = w'_{k0} + \sum_{i\neq 0} w'_{ki}\theta_i$, and note that b differs from a only in the value of w_{k0}, we get

$$\begin{aligned} \mathbf{b} - \mathbf{a} &= (w''_{k0} - w'_{k0})\,\mathbf{i}_{k0} \\ &= (f^{-1}(y_k) - f^{-1}(\theta_k))\,\mathbf{i}_{k0} \end{aligned} \quad (1)$$

Let $\hat{n}$ be the unit normal to the surface at b, so $\hat{n} = \nabla\phi_k / \|\nabla\phi_k\|$. The length of the perpendicular from a to the tangent plane at b is $(b-a).\hat{n}$. If c is the foot of the perpendicular from a to the tangent plane at b,

$$\begin{aligned} c - a &= (f^{-1}(y_k) - f^{-1}(\theta_k))\,(i_{k0}.\hat{n})\,\hat{n} \\ &= \frac{f^{-1}(y_k) - f^{-1}(\theta_k)}{\|\nabla\phi_k\|}\,\frac{\nabla\phi_k}{\|\nabla\phi_k\|} \end{aligned} \quad (2)$$

And

$$\|c - a\| = \frac{f^{-1}(y_k) - f^{-1}(\theta_k)}{\|\nabla\phi_k\|} \quad (3)$$

The vector that is directed towards the origin and biased along the axes of the weights w_{ji} that have small weight values relative to some small positive constant w_a is

$w' = -\sum_{j,i} (w_{ji}/w_a)\, i_{ji} / (1 + w^2_{ji}/w^2_a)$. The projection of w' onto the tangent plane is given by

$$\begin{aligned} m &= w' - (w'.\hat{n})\,\hat{n} \\ &= w' - \frac{1}{\|\nabla\phi_k\|}\sum_{l,m}\left(w'_{lm}\frac{\partial\phi_k}{\partial w_{lm}}\right)\frac{\nabla\phi_k}{\|\nabla\phi_k\|} \end{aligned} \quad (4)$$

Where

$$w' = -\sum_{j,i}\frac{(w_{ji}/w_a)}{1 + (w^2_{ji}/w^2_a)}\, i_{j,i} \quad (5)$$

Thus, if d is the point of intersection with the tangent plane of a line from a inclined at angle β to the perpendicular, then

$$\begin{aligned} d - a &= (d - c) + (c - a) \\ &= \|c - a\|\, \tan\beta\, \frac{m}{\|m\|} + (c - a) \end{aligned} \quad (6)$$

Let $\delta = f^{-1}(y_k) - f^{-1}(\theta_k)$ be the error in the input to final unit. Hence using equations (2), (3) and (4) in (5) yields

$$d - a = \frac{1}{\|\nabla \phi_k\|^2} \delta \nabla \phi_k + \frac{|\delta|}{\|\nabla \phi_k\|} tan\beta \frac{1}{\|m\|} \left(w' - \frac{1}{\|\nabla \phi_k\|} \times \sum_{lm} w'_{lm} \frac{\partial \phi_k}{\partial w_{lm}} \frac{\nabla \phi_k}{\|\nabla \phi_k\|} \right) \quad (7)$$

Thus, to adjust a given weight w_{ji}

$$\Delta w_{ji} = \frac{1}{\|\nabla \phi_k\|^2} \delta \frac{\partial \phi_k}{\partial w_{ji}} + \frac{|\delta|}{\|\nabla \phi_k\|} tan\beta \frac{1}{\|m\|} \left(w'_{ji} - \frac{1}{\|\nabla \phi_k\|^2} \times \sum_{lm} w'_{lm} \frac{\partial \phi_k}{\partial w_{lm}} \frac{\partial \phi_k}{\partial w_{ji}} \right) \quad (8)$$

where $\|m\|^2 = \sum_{j,i} \left(w'_{ji} - \frac{1}{\|\nabla \phi_k\|^2} \sum_{l,m} \left(w'_{lm} \frac{\partial \phi_k}{\partial w_{lm}} \right) \frac{\partial \phi_k}{\partial w_{ji}} \right)^2$ (9)

The term $\partial \phi_k / \partial w_{ji}$ is the partial derivative of the net input to the output unit. The treatment of this term follows from Lee [1]

$$\frac{\partial \phi_k}{\partial w_{ji}} = \frac{\partial \phi_k}{\partial \phi_j} \theta_i$$

and

$$\frac{\partial \phi_k}{\partial \phi_j} = \begin{cases} 1, & if \ \ j = k \\ f'_j(\phi_j) \sum_{m \in M_j} \frac{\partial \phi_k}{\partial \phi_m} w_{mj}, & if \ \ j \neq k \end{cases}$$

Where M_j is the set of units to which u_j passes its output.

The new iTPA algorithm requires two parameters that need to be set manually. First parameter is the angle parameter ***tanβ***, which gives the angle between the movement vector and the perpendicular from the current position to the tangent plane. Its value is usually chosen to be small, typically 0.05, so that movement is to a point nearby the foot of the perpendicular. Second parameter is the weight sensitivity parameter w_a, which gives the value an individual weight w_{ji} receives a large push towards the origin. w_a is preferred to be small, typically 0.5, so that weights with small values are selected for removal from the network. This will produce the required separation of active and inactive weights in the network.

An individual term $w'_{ji} = -(w_{ji}/w_a) / (1 + (w_{ji}/w_a)^2)$ in the directional vector w' varies according to (w_{ji}/w_a) in an anti-symmetric fashion. This permits the necessary sign changes in w'_{ji} so that the movement along a weight dimension is always directed towards the origin. When $|w_{ji}| < w_a$ the directional term for that weight is approximately linear. On the other hand, when $|w_{ji}| > w_a$ the directional term approaches to zero. Thus a weight will receive a large push when w_{ji} equals w_a. A potential difficulty arises when both w_{ji} and w_a are less than 0.5. If $|w_{ji}| = w_a$, the resulting push may be large enough to overshoot the origin. This situation can be avoided through the appropriate choice of ***tanβ***.

The approach of the new iTPA algorithm is reminiscent of the Newton-Raphson method of first degree [10] used to find the zero points of functions that depend on one value. An important difference is that it provides a whole R^n plane of suitable points to move towards. Any method that does a zero-point search of a function cannot get trapped in a local minimum unless it hits one by accident. The new iTPA algorithm uses the vector $\nabla \phi_k$ to do a linear extrapolation of the surface $\phi_k = f^{-1}(y_k)$ in order to gain a new weight vector that is hoped to be on, or at least close, to this surface.

A simple cost saving can be made by replacing the term $\|m\| = \|w' - (w' . \hat{n}) \hat{n}\|$ in the algorithm with $w' . \|w'\|$ is greater than or equal to $\|w' - (w' . \hat{n}) \hat{n}\|$ with equality holding when w' is perpendicular to $\hat{n}$. Its use will result in a reduction in the size of m, but this term is scaled by ***tanβ*** anyway. This reduction is greatest when w' is perpendicular to the tangent plane. According to equation (9) $\|m\|$ involves adding n products of the terms w'_{lm} and $\partial \phi_k / \partial w_{lm}$, and then using this result to scale n partial derivatives, $\partial \phi_k / \partial w_{ji}$. Thus the total computational saving is $2n$ operations per weight update.

The algorithm can be further improved by using a high degree polynomial in the denominator of each term w_{ji} in the directional vector w', so that $w'_{ji} = -(w_{ji}/w_a)/(1 + (w_{ji}/w_a)^n)$ where n is a positive constant. The curves of an individual directional term w'_{ji} for three values of parameter n are shown in Fig. 2. Examination of the curves for $|w_{ji}| > w_a$ yield the following observation. When n is large (typically > 6), the directional term w'_{ji} for that weight decays rapidly to zero. Thus the proper choice of the parameter n will permit some weights in the network to assume values that are larger than with $n = 2$.

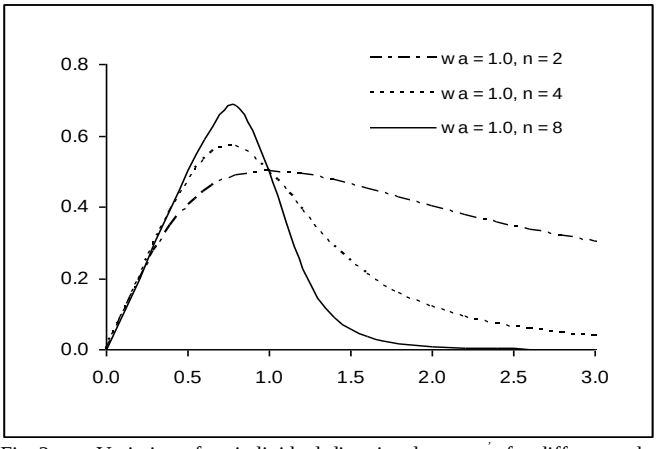

Fig. 2 Variation of an individual directional term w'_{ji} for different values of the parameter n

IV. Estimating the Weight Sensitivity

The rationale of pruning is to reduce the number of free parameters in the network by removing dispensable ones. If applied properly, this approach often reduces overfitting and improves generalization. At the same time it produces a smaller network. The approach adopted in this paper is to automatically prune superfluous weights by using the method of weight elimination [4]. But how do we know whether the method of weight elimination actually produces the required separation of active and inactive weights? One approach might be to measure the significance or importance of each weight, as the magnitude of the weights is not the best measure of their contribution to the training process [11].

There are several methods suggested for calculating the importance of connection weights. Karnin [12] measures the sensitivity s_{ji} of each weight by monitoring the sum of all the changes to the weights during training. Thus the saliency of a weight is given as $s_{ji} = \sum_t \partial \varepsilon^{(t)}_k / \partial w_{ji} \Delta w^{(t)}_{ji} w^f_{ji} / (w^f_{ji} - w^0_{ji})$, where t is the number of epochs trained, w^f_{ji} and w^0_{ji} are the final and initial values of weight w_{ji}. LeCun et al [13] measure the saliency of a weight by estimating the second derivative of the error. They also reduce the network complexity by constraining certain weights to be equal. Low saliency means low importance of the weights. A more sophisticated approach avoids the drawbacks of approximating the second derivatives by computing them exactly [14].

The last two methods have the disadvantage of requiring training down to the error minimum. The autoprune method [11] avoids this problem. It uses a statistic t to allocate an importance coefficient to each weight based upon the assumption that a weight becomes zero during the training process

$$t(w_{ji}) = log \left(\frac{\left| \sum_t w_{ji} - \Delta w^{(t)}_{ji} \right|}{\sum_t (\Delta w^{(t)}_{ji} - (\Delta \overline{w}_{ji}))^2} \right) \qquad (10)$$

In the above formula, sums are over all training examples t of the training set, and the overline means arithmetic mean over all examples. A large value of t_{ji} indicates high importance of weight w_{ji}.

V. Simulations And Results

The convergence behaviour of the new iTPA algorithm was evaluated and compared with the gradient descent backpropagation algorithm. The dataset used was the two spiral problem [15 – 16]. Like most published work classifying the two spiral problem [17], a network with three hidden layers was used. 10 trials were performed with the classification error on the training and test sets, mean number of epochs to converge, and number of successful trials recorded. Network training was terminated when all the training patterns were learned correctly or 5,000 epochs or presentations of the entire dataset.

Next, the ability of the new iTPA and original tangent plane algorithms to generalise from a given set of training data was evaluated and compared with the Extreme Learning Machine [18]. The Extreme Learning Machine (ELM) is a fast learning algorithm that obtains the least squares solution with the minimum training error and minimum norm of the weights. The benchmark datasets used were the Henon map [19] and the non-linear dynamic plant [20]. A standard feedforward neural network with two hidden layers was utilised with the number of hidden units determined by grid search. For each test, 10 trials were performed with the normalised mean square error on the training and test sets recorded together with the number of successful trials. Network training was terminated after 5,000 epochs or presentations of the entire training set.

Finally, the evolution and development of the weights in the new iTPA algorithm was evaluated and compared with the original tangent plane algorithm. The benchmark datasets used were the Henon map [19] and the non-linear dynamic plant [20]. The method used to estimate the sensitivity of the weights was autoprune [11]. Histograms of weight sensitivities were plotted after 100, 300 and 500 epochs.

A. Network initialization

The algorithms used in the study require manually set parameters. Preliminary tests showed that the best results were obtained with the parameters set as follows. First test is the new iTPA algorithm. For the two spiral problem, $tan\beta$ = 0.01. The weight sensitivity parameter w_a and n were varied according to a grid search. The input weights were set to random values in the range [-2, 2]. For the Henon map and non-linear dynamic plant, $tan\beta$ = 0.01, w_a = 0.5, and n = 4.0. The input weights were set to random values in the range [-0.5, 0.5]. Next test is the original tangent plane algorithm. The angle parameter $tan\beta$ = 0.01. The input weights were set to random values in the range [-0.01, 0.01]. Finally test is the standard back-propagation algorithm. For the two spiral problem, the learning rate η = 0.01, and momentum coefficient α = 0.3. The input weights were set to random values in the range [-2, 2].

B. Simulation problems

The two spiral problem consists of two interlocking spirals, each made up of 97 data points. The network must learn to discriminate the two spirals. Traditionally this is known to be a very difficult problem for the back-propagation algorithm to solve. There are two inputs and one output. The inputs are the x and y co-ordinates, and the output notifies which spiral the point belongs to. For the points in the first spiral the output is set to +1, and for points on the other spiral the output is set to -1. The number of training samples is 194. A test set of 192 samples was generated by rotating the two spirals by a small angle.

The Henon map problem is a chaotic time-series prediction problem. The time series is computed by

$$x^{(t+1)} = 1 - c\,[x^{(t)}]^2 + b\,x^{(t-1)} \qquad (11)$$

Where $x^{(t)}$ is the value at taken time t, and the parameters b = 0.3, and c = 1.4. Initial values for the time series are $x^{(1)} = x^{(0)}$ = 0.63133545. This point is called the fixed point of the time series. In neural network simulations, four successive values of the time series are used in predicting the next value.

Thus, the number of inputs is four and the number of output is one. Data values were taken from the range [31,230] as given in Lahnajärvi et al [19]. The number of training samples is 100, and testing samples is 100.

The non-linear dynamic plant problem is a high order non-linear system introduced in Narendra and Parthasarathy [20]. It is modelled by the following discrete time equation

$$y^{(t)} = \frac{y^{(t-1)}y^{(t-2)}y^{(t-3)}u^{(t-1)}[y^{(t-3)}-1]+u^{(t)}}{1+[y^{(t-2)}]^2+[y^{(t-3)}]^2} \quad (12)$$

Where $\boldsymbol{y^{(t)}}$ is the model output at time $\boldsymbol{t}$. Like Narendra and Parthasarathy [20], training data was generated using a random input signal uniformly distributed over the interval [-1, 1]. Five hundred data points were generated, the first three hundred used as training data whilst the remaining used as test data.

C. Error metrics used to determine convergence

The error metrics used in the simulations were CERR (Classification ERRor) for classification problems and NMSE (Normalized Mean Square Error) for regression problems. The CERR was calculated using the 40-20-40 criteria e.g. the actual output does not differ from the target output by more than 0.4 [15 – 16]. The NMSE was calculated by dividing the MSE by the variance of the target output.

D. Discussion of results

Two spiral problem. The first test is a difficult non-linearly separable problem where a set of co-ordinates (***x,y***) is classified as belonging to one of two interwoven spirals. A 2-100-100-100-1 network topology was chosen as given in [16]. For the new iTPA algorithm, 10 trails gave no failures and a mean number of steps to converge of 28 with standard deviation 9. The classification error on the test set was 1.3×10^{-2} (e.g. % test set learned = 98.7) with all of the points on the training set correctly classified. Using the standard backpropagation algorithm, there was one failure and a mean number of steps to converge of 736 with standard deviation 462. The classification error on the test set was 1.5×10^{-2} (e.g. % test set learned = 98.5). The results compare very favourably with those given in Linder et al [16] (Aprop: epochs = 67, % test set learned = 96.6; Rprop: epochs = 246, % test set learned = 65.6).

Table 1 demonstrates the effects of changing the weight sensitivity parameters $\boldsymbol{w_a}$ and $\boldsymbol{n}$ of the new iTPA algorithm. A 2-20-20-20-1 architecture was chosen to determine the degree to which the iTPA algorithm could generalize in a large network with many free parameters. It was found that the classification error improved slightly when large values were chosen for the weight sensitivity parameter $\boldsymbol{w_a}$.

Further, increasing the value of the parameter $\boldsymbol{n}$ caused the classification error to dip to a clearly defined minimum. When $\boldsymbol{w_a}$ is large (e.g. typically > 0.5) the directional vector $\boldsymbol{w'}$ will push more of the network weights to small values close to zero thus implementing a weight decay procedure. This suggests that weight decay is a far more effective strategy for improving generalization than weight elimination in large neural networks trained using the tangent plane algorithm.

TABLE I. CONVERGENCE SPEED AND CLASSIFICATION ACCURACY FOR DIFFERENT PARAMETERS IN THE ITPA ALGORITHM

w_a	n	Cerr	Cerr*	Steps	Succ
0.10	4.0	0.46	1.35	497	10
0.20	4.0	0.36	1.56	634	10
0.50	4.0	0.41	0.78	563	10
1.00	4.0	0.36	0.78	460	10
0.50	2.0	0.31	0.89	435	10
0.50	4.0	0.41	0.78	563	10
0.50	6.0	0.52	0.99	532	10
0.50	8.0	0.41	0.89	559	10

Note: the columns in Table 1 refer to the classification error on the training set (Cerr) and test set (Cerr*), the mean number of steps to converge, and number of success trials for different values of the weight sensitivity parameter $\boldsymbol{w_a}$ and $\boldsymbol{n}$.

Fig 3 and 4 show some typical test curves for both algorithms on the two spiral problems. Different sets of initial weights were used in each test. The test curves of the new iTPA algorithm show some variation (Fig 3). In many of the curves generated the test error was found to diminish slowly at the start of the training run with intermittent rises in the test error fairly typical (test 2). When the new algorithm was close to a solution, the convergence was usually rapid (test 1 and 3). The test curves of the standard back-propagation algorithm also show wide variation in the test error. Some curves exhibited turbulent behaviour similar to the new iTPA algorithm (test 3). Other curves got trapped in local minima of the error landscape resulting in very long runtimes (test 1). Generally speaking the new iTPA algorithm is prone to problems with stability. Introducing a 50% staged reduction in the step size resulted in faster convergence speeds.

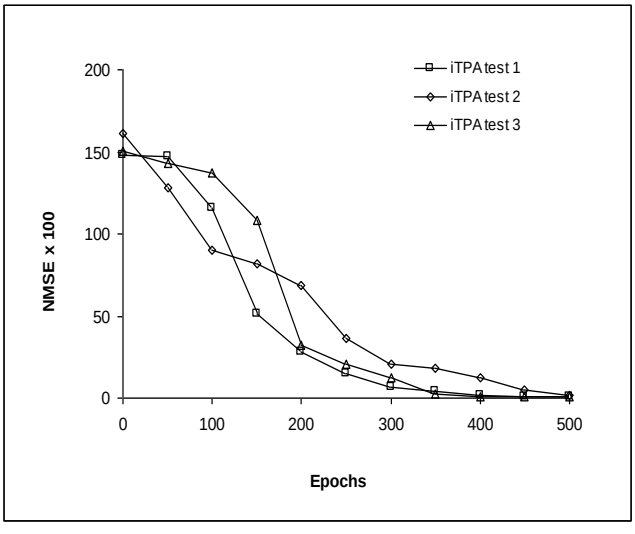

Fig. 3 Typical generalization behaviour of the new iTPA algorithm on the two spiral problem

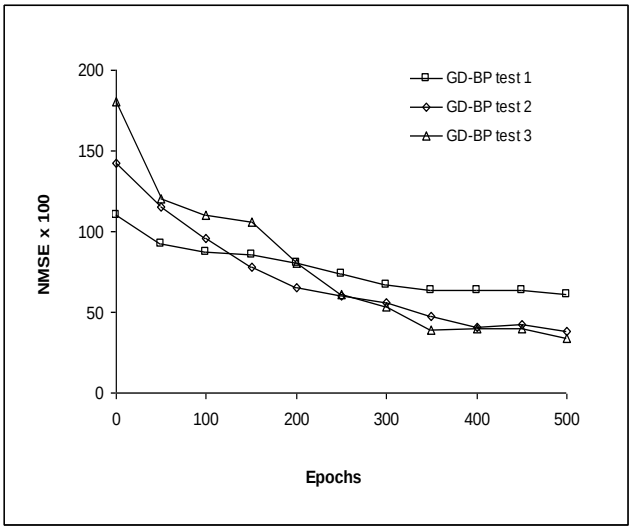

Fig. 4 Typical generalization behaviour of the gradient descent backpropagation algorithm on the two spiral problem

Henon map time series. The second test is a classical deterministic one-step-ahead prediction problem. Preliminary tests showed that the best results for both tangent plane algorithms were obtained using a 4-15-15-1 architecture. Network training was terminated after 5,000 epochs or presentations of the entire dataset. For the new iTPA algorithm, 10 trials gave a normalised mean square error on the training set and test set of 0.00007 and 0.00008 respectively. Using the original tangent plane algorithm, 10 trials gave (training set = 0.00005, test set = 0.00009). There was little evidence of overtraining. The performance of both tangent plane algorithms compare favourably with the Extreme Learning Machine. For ELM a single hidden layer feedforward neural net with 80 hidden units gave (training set = 0.00001, test set = 0.00011).

Fig 5 and 6 show histograms of the importance coefficients of the weights for both algorithms on the Henon map problem. The importance coefficients were recorded from the same trial at epochs 100, 300 and 500. The coefficient sizes were grouped in classes of width one and histograms plotted to show the distribution of the t_{ji} values at three different stages of training. The new iTPA algorithm gave average coefficient sizes at 100, 300, 500 epochs of 1.82, 1.95, and 1.95 respectively. The original algorithm gave 1.70, 1.96, and 2.25. Notice the right skewness of the histograms produced by the new algorithm (see Fig 5). After 500 epochs the histogram is dominated by a single high peak and long right tail. This suggests that many of the weights have taken on equally important roles in the network. Thus are likely to be fewer outlier weights with extreme values that are known to produce overtraining in neural networks.

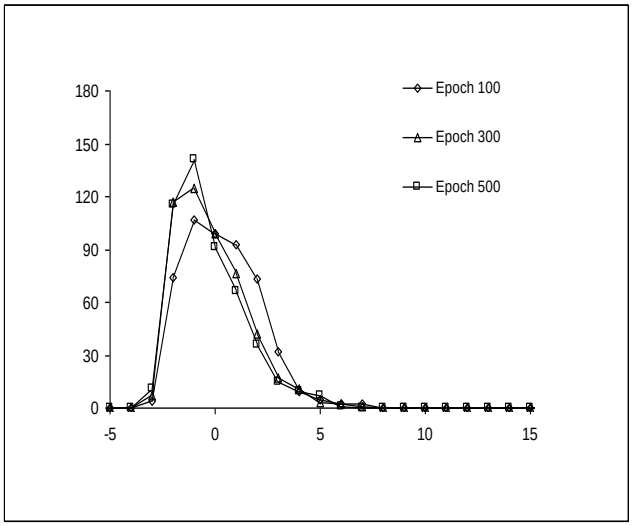

Fig. 5 Importance coefficient histograms for the new iTPA algorithm (Henon map problem). Horizontal axis: coefficient size grouped in classes of width 1. Vertical axis: absolute frequency of weights with this coefficient size.

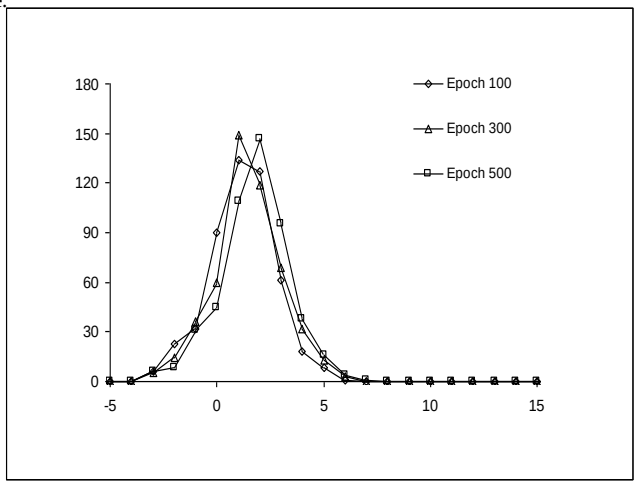

Fig. 6 Importance coefficient histograms for the original tangent plane algorithm (Henon map problem). Horizontal axis: coefficient size grouped in classes of width 1. Vertical axis: absolute frequency of weights with this coefficient size

Non-linear dynamic plant. The third test is a high order non-linear discrete time system. Preliminary tests showed that the best results for both tangent plane algorithms were obtained using a 2-10-10-1 architecture. Network training was terminated after 5,000 epochs or presentations of the entire dataset. For the new iTPA algorithm, 10 trials gave an average normalised mean square error on the training and test sets of 0.00045 and 0.00086 respectively. Using the original tangent plane algorithm, 10 trials gave (training set = 0.00142, test set = 0.00393).

Once again the performance of both tangent plane algorithms compare favourably with the Extreme Learning Machine. For ELM a single hidden layer neural net with 200 hidden units gave (training set = 0.00007, test set = 0.00568). The results on the training set suggest that the new iTPA algorithm is an effective global minimizer capable of reaching the smallest training error.

Fig 7 and 8 show histograms of the importance coefficients of the weights for both algorithms on the non-linear dynamic plant problem. The importance coefficients were recorded from the same trial at epochs 100, 300 and 500. The coefficient sizes were grouped in classes of width one and histograms plotted to show the distribution of the t_{ji} values at three different stages of training. The new iTPA algorithm gave average coefficient sizes at 100, 300, 500 epochs of 2.24, 2.68, and 2.79 respectively. The original algorithm gave 2.56, 3.33, and 3.81. Notice the left drift of the histograms produced by the new algorithm (see Fig 7). In contrast the histograms produced by the original algorithm tend to drift right, as expected (Fig 8). Further, these histograms have a long right tail which suggests that some weights are taking on a far more active role in the network. This might account for the worse generalization of the original algorithm on this problem.

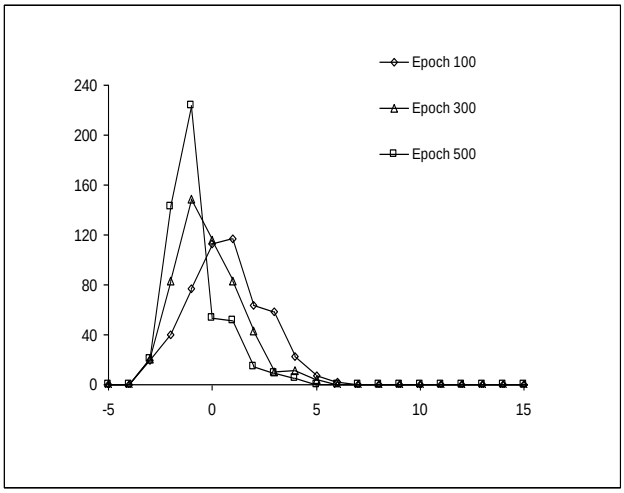

Fig. 7 Importance coefficient histograms for the new iTPA algorithm (non-linear dynamic plant). Horizontal axis: coefficient size grouped in classes of width 1. Vertical axis: absolute frequency of weights with this coefficient size

VI. Comparison Of The Different Algorithms

In order to determine whether the difference in the results is statistically significant, we perform some hypothesis tests. The test used was a standard *t*-test with the sample of test errors from the iTPA algorithm compared with the corresponding sample from the original tangent plane algorithm for each dataset used in the study. A second test was carried out by comparing these test results with the ELM algorithm on the same set of problems.

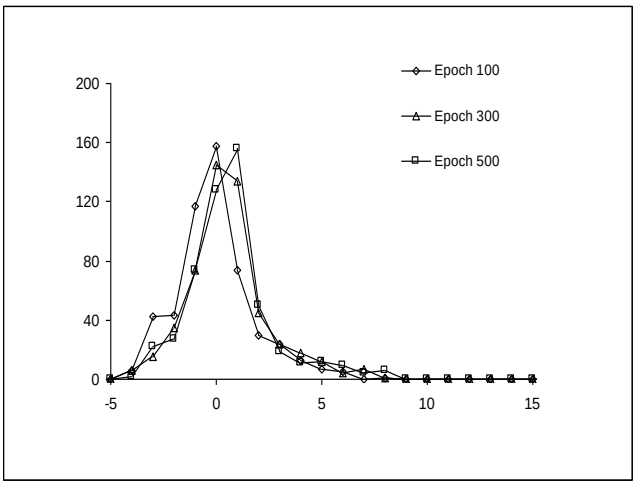

Fig. 8 Importance coefficient histograms for the original tangent plane algorithm (non-linear dynamic plant). Horizontal axis: coefficient size grouped in classes of width 1. Vertical axis: absolute frequency of weights with this coefficient size

The ELM algorithm requires the number of hidden units to be set which was found be grid search. For the correct application of the *t*-test, it was necessary to take the logarithm of the test errors (since the test errors have log-normal distribution) and remove any outliers, following the same procedure in [21]. The resulting samples were tested for normality using the Kolmogorov-Smirnov test.

TABLE II. Results of a T-test comparing the mean test errors of the different algorithms

Problem	Training samples	Test samples	Inputs	(a)	(b)	(c)
Spiral	194	192	2	L 6.51	-	E 7.03
Henon	100	100	4	-	-	-
Plant	150	103	13	L 5.37	L 11.98	-

Note: The entries show differences that are statistically significant on a 10% level and dashes mean no significance found. Column (a): iTPA algorithm ("L") vs. original tangent plane algorithm ("T"). Column (b): iTPA algorithm vs. ELM algorithm ("E"). Column. (c): original tangent plane algorithm vs. ELM algorithm.

The results are tabulated in Table 2. Dashes mean differences that are not significant at the 10% level i.e. the probability that the differences are purely accidental. Other entries indicate the superior algorithm (e.g. iTPA algorithm - L, original tangent plane algorithm – T, ELM algorithm - E), and the value of the t statistic. Column (a) gives a comparison between the new iTPA algorithm and the original tangent plane algorithm. The results show two times L is better (spiral and non-linear dynamic plant) and once T is better (Henon map).

The new iTPA Algorithm performed better on the datasets that were more difficult to learn and so convergence speeds tended to be slower. Where convergence occurred quickly the original algorithm was the better method. Column (b) and (c) give comparisons between the new iTPA and original tangent plane algorithms, and the ELM algorithm. The results show 4 times no statistical difference, once L is better and twice E is better.

This suggests that the generalization performance of the tangent plane method is at least as good as the ELM algorithm, which is one of the best neural network classifiers. In situations where time varying signals are required, such as EEG and ECG signals, the sequential learning ability of the tangent plane algorithm might be the preferred method.

VII. CONCLUSIONS

A new variant of the tangent plane algorithm referred to as iTPA is proposed for feed-forward neural networks. This new algorithm includes two modifications to the existing algorithm. Firstly, a directional movement vector is introduced into the training process to push the movement in weight space towards the origin. This directional vector is built into the geometry of the tangent plane algorithm and implements a weight elimination procedure. Secondly, a high degree polynomial term is utilised to adjust the proportion of weights that receive an inwards push. Thus the algorithm can be tuned to decay specific weights to zero (which can help generalization).

Comparative tests were carried out using the new iTPA and original tangent plane algorithms, the gradient descent back-propagation algorithm and the Extreme Learning Machine. The results indicate that the new iTPA algorithm retains the fast convergence speed of the original method. However, the new iTPA algorithm is prone to problems with stability. Including a 50% reduction in step size often improves convergence behaviour without any diminution in learning speed. The results also show that the new algorithm gives improved generalization relative to the original algorithm in some problems, and has comparable generalization performance in yet others. Further, the generalization performance of the tangent plane method is at least as good as the Extreme Learning Machine, which is one of the best neural network classifiers.

VIII. FUTURE WORK

This paper shows that the newly developed improved tangent plane algorithm (iTPA) is at least as good as the extreme learning machine, which is one of the best neural network classifiers. In situations where time varying signals are required, such as EEG and ECG signals, the sequential learning ability of the improved tangent plane algorithm might be the preferred method.

REFERENCES

[1] C.W. Lee, "Training feedforward neural networks: an algorithm giving improved generalization," *Neural Networks*, vol. 10. 1997 pp. 61-68.

[2] P. Bartlett, "For valid generalization, the size of the weights is more important than the size of the network," Advances in Neural Information Processing Systems 9, Cambridge, MA: The MIT Press, 1997, pp. 134-140.

[3] S. J. Nowlan, and G. E. Hinton, "Simplifying neural networks by soft weight sharing," *Neural Computation*, vol. 4, no. 4, pp. 473-493, 1992

[4] A.S. Weigend, D.E. Rumelhart and B.A. Huberman, "Generalization by weight elimination with application to forecasting," *Advances in neural information processing* (3), 1991, pp. 875-882.

[5] P.M. Williams, "Baysian regularisation and pruning using a Laplacian prior," *Technical report*, (312), 1994

[6] P.O. Hoyer, "Non-negative matrix factorisation with sparseness constraints," *Journal of machine learning research*, (5): 1457 – 1469. 2004

[7] Huiwen Zeng, "Dimensionality reduction using a mixed term penalty reduction," *IEEE workshop on machine learning for signal processing*, 2005

[8] C.M. Ennett and M. Frize, "Weight elimination neural networks applied to coronary surgery morality prediction." IEEE Trans Inf Technol Biomed, 2003, 7(2):86-92.

[9] R.M. Zur, Yulei Jiang, L. Pesce, and K. Drukker, "Noise injection for training neural networks: a comparison with weight decay and early stopping," *Med. Phys*. 2009, 36(10): 4810-4818.

[10] J. Stoer, "Einführung in die numerische Mathematik, ", Springer, Vol. 1. S, 1976

[11] W. Finnoff, F. Hergert, and H.G. Zimmermann, "Improving model selection by non-convergent methods," *Neural Networks*, vol.6, 1993, 771-783.

[12] E.D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE trans neural networks*, vol. 1, no. 2, 1990, pp. 239-242.

[13] Y.L, LeCun, J.S., Denker,and S.A. Solla, "Optimal brain damage," *Advances in neural information processing systems*, vol.2, 1990, pp. 598-605.

[14] B. Hassibi, and D.G. Stork, "Second order derivatives for network pruning," *Advances in neural information processing systems*, vol.5, 1993, pp. 164-171.

[15] S. Fahlman, and C. Lebiere, "The cascade correlation learning architecture," *Advances in neural information processing systems*, vol. 2. 1990, pp. 524-532.

[16] R. Linder, S. Wirtz, and S.J. Poppl, "Speeding up backpropagation learning by the APROP algorithm," *Proceedings of the second international ICSC symposium on neural computation*, ICSC Academic Press, 2000, pp. 122-128.

[17] K.L. Lang, and M.J. Witbrock, "Learning to Tell Two Spirals Apart," *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann 1988.

[18] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Slew, "Extreme learning machine: Theory and applications," *Neurocomputing*, 70, 489-501, 2006

[19] J.J.T. Lahnajärvi, M.I. Lehtokangas, and J.P.P. Saarinen, "Evaluation of constructive neural networks with cascade architectures," *Neurocomputing*, vol. 48, 2002, pp. 573-607.

[20] K.S Narandra, and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE transactions on neural networks*, 1 (1), 4 1990.

[21] L. Prechelt, "Connection pruning with static and adaptive pruning schedules," *Neurocomputing*, Volume 16, Issue 1, 1997, pp. 49-61