Global Feedforward Neural Network Learning For Classification And Regression

Kar-Ann Toh, Juwei Lu and Wei-Yun Yau

Centre for Signal Processing
School of Electrical & Electronic Engineering
Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798
Email: ekatoh@ntu.edu.sg

Abstract. This paper addresses the issues of global optimality and training of a Feedforward Neural Network (FNN) error funtion incorporating the weight decay regularizer. A network with a single hidden-layer and a single output-unit is considered. Explicit vector and matrix canonical forms for the Jacobian and Hessian of the network are presented. Convexity analysis is then performed utilizing the known canonical structure of the Hessian. Next, global optimality characterization of the FNN error function is attempted utilizing the results of convex characterization and a convex monotonic transformation. Based on this global optimality characterization, an iterative algorithm is proposed for global FNN learning. Numerical experiments with benchmark examples show better convergence of our network learning as compared to many existing methods in the literature. The network is also shown to generalize well for a face recognition problem.

1 Introduction

Backpropagation of error gradients has proven to be useful in layered feedforward neural network learning. However, a large number of iterations is usually needed for adapting the weights. The problem becomes more severe especially when a high level of accuracy is required. It has been an active area of research, between late 1980s and early 1990s, to derive fast training algorithms to circumvent the problem of slow training rate as seen in the error backpropagation algorithm. Among the various methods proposed (see e.g. [6, 14]), significant improvement to the training speed is seen through the application of nonlinear optimization techniques in network training (see e.g. [1, 2, 22]). Very often, this is achieved at the expense of heavier computational requirement.[1] Here, we note that most of them are local methods and training results are very much dependent on the choices of initial estimates.

In light of efficient training algorithm development and network pruning (see [5], page 150-151), exact calculation of the second derivatives (Hessian) and its

[1] For example, the complexity of each step in Newton's method is $O(n^3)$ as compared to most first-order methods which are $O(n)$.

multiplied forms were studied [4, 16]. In [4], it was shown that the elements of the Hessian matrix can be evaluated exactly using multiple forward propagation through the network, followed by multiple backward propagation, for a feedforward network without intra-layer connections. In [16], a product form of the Hessian which took about as much computation as a gradient evaluation was presented. The result was then applied to a one pass gradient algorithm, a relaxation gradient algorithm and two stochastic gradient calculation algorithms to train the network. From the review in [7] specifically on the computation of second derivatives of feedforward networks, no explicit expression for the eigenvalues was seen. Here we note that eigenvalues are important for convexity analysis.

In view of the lack of an optimality criterion for global network learning in classification and regression applications, we shall look into the following issues in this paper: (i) characterization of global optimality of a FNN learning objective incorporating the weight decay regularizer, and (ii) derivation of an efficient search algorithm based on results of (i). We shall provide extensive numerical studies to support our claims.

The paper is organized as follows. In section 2, the layered feedforward neural network is introduced. This is followed by explicit Jacobian and Hessian formulations for forthcoming analysis. The FNN learning error function is then regularized using the weight decay method for good generalization capability. Then convexity analysis is performed in section 3 for local solution set characterization. This paves the way for a new approach on global optimality characterization in section 4. In section 5, the analysis results are used to derive a search algorithm to locate the global minima. Several benchmark examples are compared in section 6 in terms of convergence properties. The learning algorithm is also applied to a face recognition problem with good convergence as well as good generalization. Finally, some concluding remarks are drawn.

Unless otherwise stated, vector and matrix quantities are denoted using bold lowercase characters and bold uppercase characters respectively to distinguish from those scalar quantities. The superscript 'T' on the respective character is used to denote matrix transposition. Also, if not otherwise stated, $\|\cdot\|$ is taken to be the l_2-norm.

2 Multilayer Feedforward Neural Network

2.1 Neural feedforward computation

The neural network being considered is the familiar multilayer feedforward network with no recurrent or intra-layer connections. The forward calculation of a strictly 2-layer network with one output-node, i.e. network with one *direct* input-layer, one hidden-layer, and one output-layer consisting a single node (i.e. network with (N_i-N_j-1) structure) can be written as:

$$y(\mathbf{x}, \boldsymbol{w}) = g\left[\sum_{j=0}^{N_j} g\left(\sum_{i=0}^{N_i} w_{ji}^h x_i\right) \cdot w_{kj}^o\right], \quad g(z_0^h) = g(w_{0i}^h x_i) = 1,\ x_0 = 1,\ k = 1 \tag{1}$$

where $\mathbf{x} = \{x_i, i = 1, 2, ..., N_i\}$ denotes the network input vector and $\boldsymbol{w} = \left\{[w^o_{kj}]_{j=0,...,N_j;k=1}, [w^h_{ji}]_{i=0,...,N_i;j=1,...,N_j}\right\}$ denotes the weight parameters to be adjusted. $g(\cdot)$ is a sigmoidal function given by

$$g(\cdot) = \frac{1}{1 + e^{-(\cdot)}} \ . \tag{2}$$

The superscripts '^{o}' and '^{h}' denote weights that are connected to output-nodes and hidden-nodes respectively.

For n number of data points, input $\mathbf{x}$ becomes a n-tuple vector (i.e. $\mathbf{x} \in \mathcal{R}^{(N_i+1)\times n}$ for N_i number of input-nodes plus a constant bias term) and so is the output $\mathbf{y}$ (i.e. $\mathbf{y} \in \mathcal{R}^n$).

2.2 Explicit Jacobian and Hessian for a two-layer network with single output

Here, we stack the network weights as a parameter vector $\boldsymbol{w} \in \mathcal{R}^p$ ($p = (N_j + 1) + (N_i + 1)N_j$) as follows:

$$\boldsymbol{w} = [w^o_{k,0}, ..., w^o_{k,N_j}, \ w^h_{1,0}, ..., w^h_{1,N_i}, \ \cdots, \ w^h_{N_j,0}, ..., w^h_{N_j,N_i}]^T. \tag{3}$$

Consider a single data point, the first derivative of network output $y(\mathbf{x}, \boldsymbol{w})$ (1) with respect to the weights vector can be written as

$$\mathbf{J}(\boldsymbol{w}) = \nabla^T y(\mathbf{x}, \boldsymbol{w}) = \dot{g}(z^o_k)\mathbf{r}^T, \tag{4}$$

where

$$\mathbf{r} = \left[1, g(z^h_1), ..., g(z^h_{N_j}), \ \dot{g}(z^h_1)w^o_{k,1}\mathbf{u}^T, \ \cdots \ \dot{g}(z^h_{N_j})w^o_{k,N_j}\mathbf{u}^T\right]^T \ \in \mathcal{R}^p, \tag{5}$$

$$\mathbf{u} = [1, x_1, ..., x_{N_i}]^T \ \in \mathcal{R}^{N_i+1}, \tag{6}$$

$$\dot{g}(\cdot) = \frac{e^{-(\cdot)}}{(1 + e^{-(\cdot)})^2}, \tag{7}$$

$$z^o_k = \sum_{j=0}^{N_j} g(z^h_j) \cdot w^o_{kj}, \quad g(z^h_0) = 1, \ k = 1, \tag{8}$$

$$z^h_j = \sum_{i=0}^{N_i} w^h_{ji}x_i, \quad g(z^h_0) = g(w^h_{0i}x_i) = 1, \ x_0 = 1, \ j = 1, ..., N_j \ . \tag{9}$$

The second derivative which is also termed the Hessian for the network function $y(\mathbf{x}, \boldsymbol{w})$ (1) is then

$$\mathbf{Y} = \nabla^2 y(\mathbf{x}, \boldsymbol{w}) = \ddot{g}(z^o_k)\mathbf{P} + \dot{g}(z^o_k)\mathbf{Q}, \tag{10}$$

where

$$\ddot{g}(\cdot) = \frac{\left(e^{-(\cdot)} - 1\right)e^{-(\cdot)}}{(1 + e^{-(\cdot)})^3}, \tag{11}$$

$$\mathbf{P} = \mathbf{r}\mathbf{r}^T, \tag{12}$$

$$\mathbf{Q} = \begin{bmatrix} 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots & \dot{g}(z_1^h)\mathbf{u}^T & & 0 \\ \vdots & & \ddots & \vdots & & \ddots & \\ 0 & \cdots & \cdots & 0 & 0 & & \dot{g}(z_{N_j}^h)\mathbf{u}^T \\ 0 & \dot{g}(z_1^h)\mathbf{u} & & 0 & \ddot{g}(z_1^h)w_{k,1}^o\mathbf{u}\mathbf{u}^T & & 0 \\ \vdots & & \ddots & & & \ddots & \\ 0 & 0 & & \dot{g}(z_{N_j}^h)\mathbf{u} & 0 & & \ddot{g}(z_{N_j}^h)w_{k,N_j}^o\mathbf{u}\mathbf{u}^T \end{bmatrix}. \tag{13}$$

2.3 Learning and generalization

Since the goal of network training is not just to learn the given sample training data, here we adopt the *weight decay* (see e.g. [5, 9]) regularization method to provide some degree of network generalization.

Consider a target learning vector given by $y_i^t, i = 1, ..., n$, the following learning objective for FNN $y(\mathbf{x}_i, \boldsymbol{w})$ is considered:

$$s(\boldsymbol{w}) = s_1(\boldsymbol{w}) + b\, s_2(\boldsymbol{w}), \quad b \geqslant 0 \tag{14}$$

where

$$s_1(\boldsymbol{w}) = \sum_{i=1}^{n} \left(y_i^t - y(\mathbf{x}_i, \boldsymbol{w})\right)^2 = \sum_{i=1}^{n} \epsilon_i^2(\boldsymbol{w}) = \boldsymbol{\epsilon}^T(\boldsymbol{w})\boldsymbol{\epsilon}(\boldsymbol{w}), \tag{15}$$

and

$$s_2(\boldsymbol{w}) = \|w\|_2^2 = \boldsymbol{w}^T\boldsymbol{w}. \tag{16}$$

In what follows, we shall retain s_1 as the minimization objective for regression applications (or by setting $b = 0$ for s in (14)) since only fitting accuracy is required. We shall observe how the learning objective in (14) can influence generalization of FNN learning for classification in a face recognition problem.

3 Convexity Analysis

In this section, we shall analyze the convexity of the more general s such that local optimal solutions can be found. The convexity of s_1 can be directly obtained by setting $b = 0$ from that of s. According to ([17],Theorem 4.5), convexity of a twice continuously differentiable function is conditioned by the positive semi-definiteness of its Hessian matrix.

To determine explicit conditions for our application, consider the l_2-norm training objective given n training data in (14). The Hessian of $s(\boldsymbol{w})$ can be written as

$$\nabla^2 s(\boldsymbol{w}) = 2\left[\mathbf{E}^T(\boldsymbol{w})\mathbf{E}(\boldsymbol{w}) + \mathbf{R}(\boldsymbol{w}) + b\mathbf{I}\right], \tag{17}$$

where $\mathbf{E}(\boldsymbol{w})$ is the Jacobian of $\boldsymbol{\epsilon}(\boldsymbol{w})$ and

$$\begin{aligned} \mathbf{E}^T(\boldsymbol{w})\mathbf{E}(\boldsymbol{w}) &= (-\mathbf{J})^T(\boldsymbol{w})(-\mathbf{J})(\boldsymbol{w}) = \mathbf{J}^T(\boldsymbol{w})\mathbf{J}(\boldsymbol{w}), \\ \mathbf{R}(\boldsymbol{w}) &= \left[-\sum_{i=1}^{n} \frac{\partial^2 y(\mathbf{x}_i, \boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^T} \epsilon_i(\boldsymbol{w})\right] \in \mathcal{R}^{p \times p}. \end{aligned} \tag{18}$$

The positive semi-definiteness of $\nabla^2 s(\boldsymbol{w})$ is thus dependent on the matrices $\mathbf{J}^T(\boldsymbol{w})\mathbf{J}(\boldsymbol{w})$, $b\mathbf{I}$ and $\mathbf{R}(\boldsymbol{w})$. By exploring the structure of the above Hessian, we present our convexity result as follows. We shall prove two lemmas before we proceed further:

Lemma 1. *Consider* $\mathbf{v} = [v_1, v_2, ..., v_n]^T \in \mathcal{R}^n$ *for some* $v_i \neq 0, i = 1, 2, ..., n$. *Then, the eigenvalues of* $\mathbf{v}\mathbf{v}^T$ *are given by*

$$\lambda(\mathbf{v}\mathbf{v}^T) = \left\{\sum_{i=1}^n v_i^2, 0, ..., 0\right\}. \tag{19}$$

Proof: Since $\mathbf{v}\mathbf{v}^T$ is symmetric with rank one, we have one and only one real non-zero eigenvalue which is the trace of $\mathbf{v}\mathbf{v}^T$. This completes the proof. ■

Lemma 2. *Consider* $\mathbf{v} = [v_1, v_2, ..., v_n]^T$ *for some* $v_i \neq 0, i = 1, 2, ..., n$ *and*

$$\mathbf{A} = \begin{bmatrix} 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots & a_1\mathbf{v}^T & & 0 \\ \vdots & & \ddots & \vdots & & \ddots & \\ 0 & \cdots & \cdots & 0 & 0 & & a_{n-1}\mathbf{v}^T \\ 0 & a_1\mathbf{v} & & 0 & b_1\mathbf{v}\mathbf{v}^T & \cdots & 0 \\ \vdots & & \ddots & & \vdots & \ddots & \vdots \\ 0 & 0 & & a_{n-1}\mathbf{v} & 0 & \cdots & b_{n-1}\mathbf{v}\mathbf{v}^T \end{bmatrix} \in \mathcal{R}^{n^2 \times n^2}. \tag{20}$$

Then, the eigenvalues of $\mathbf{A}$ *are given by* $\lambda(\mathbf{A}) = \left\{\lambda_1, ..., \lambda_{2(n-1)}, 0, ..., 0\right\}$, *where* $\lambda_j = \frac{1}{2}\left[b_j V \pm \sqrt{(b_j V)^2 + 4(a_j^2 V)}\right]$, $j = 1, 2, ..., n-1$, *and* $V = \sum_{i=1}^n v_i^2$.

Proof: Solve for the eigenvalues block by block will yield the above result. ■

Now, we are ready to perform convexity analysis for local solution set characterization. The FNN learning problem addressed, in a more precise manner, is stated as:

Problem 1 *The problem of FNN learning is defined by the* l_2*-norm minimization objective given by (14) where* $y(\mathbf{x}_i, \boldsymbol{w})$ *is a (*N_i*-*N_j*-1) network defined by (1) with sigmoidal activation functions given by (2). The Jacobian of* $y(\mathbf{x}_i, \boldsymbol{w})$, $i = 1, 2, ..., n$ *is denoted by* $\mathbf{J}^T(\boldsymbol{w}) = [\nabla y(\mathbf{x}_1, \boldsymbol{w}), \nabla y(\mathbf{x}_2, \boldsymbol{w}), ..., \nabla y(\mathbf{x}_n, \boldsymbol{w})]$ *where each of* $\nabla y(\mathbf{x}_i, \boldsymbol{w}), i = 1, 2, ..., n$ *is evaluated using (4)-(9).*

First we present a first-order necessary condition for network learning as follows:

Proposition 1. *Given Problem 1. Then, the least squares estimate* $\hat{\boldsymbol{w}}$ *of* $\boldsymbol{w} = [w_1, w_2, ..., w_p]^T$, $p = (N_j + 1) + (N_i + 1)N_j$ *in the sense of minimizing* $s(\boldsymbol{w})$ *of (14) satisfies*

$$\hat{\boldsymbol{w}} - \mathbf{J}^T(\hat{\boldsymbol{w}})[\mathbf{y}^t - \mathbf{y}(\mathbf{x}, \hat{\boldsymbol{w}})] = \mathbf{0}. \tag{21}$$

Proof: Minimizing $s(\boldsymbol{w})$ with respect to $\boldsymbol{w}$ by setting its first derivative to zero yields the normal equation given by (21). $\hat{\boldsymbol{w}}$ is the point satisfying (21). Hence the proof. ■

The convexity result is presented as follows:

Theorem 1 *Given Problem 1. Then $s(\boldsymbol{w})$ is convex on $\boldsymbol{w} \in \mathcal{W}_c$ if*

$$\sum_{i=1}^{n}\left[\left(\ddot{g}(z_k^o)\lambda_m(\mathbf{P})\right)_i + \left(\dot{g}(z_k^o)\lambda_m(\mathbf{Q})\right)_i\right]\epsilon_i \leqslant b, \tag{22}$$

where

$$\lambda_m(\mathbf{P}) = \begin{cases} \sum_{k=1}^{p} r_k^2, & \epsilon_i \geqslant 0 \\ 0 & \epsilon_i < 0 \end{cases}, \quad i = 1, 2, ..., n\ , \tag{23}$$

$$[r_1, r_2, ..., r_p] = \left[1, g(z_1^h), ..., g(z_{N_j}^h),\ \dot{g}(z_1^h)w_{k,1}^o\mathbf{u}^T,\ \cdots\ \dot{g}(z_{N_j}^h)w_{k,N_j}^o\mathbf{u}^T\right], \tag{24}$$

$$\lambda_m(\mathbf{Q}) = \begin{cases} \max_j \lambda_j, & \epsilon_i \geqslant 0 \\ \min_j \lambda_j, & \epsilon_i < 0 \end{cases}, \quad j = 1, 2, ..., N_j\ , \tag{25}$$

$$\lambda_j = \frac{1}{2}\left[\ddot{g}\left(z_j^h\right)w_{k,j}^o V \pm \sqrt{\left(\ddot{g}\left(z_j^h\right)w_{k,j}^o V\right)^2 + 4\dot{g}^2\left(z_j^h\right)V}\right], j = 1, ..., N_j, \tag{26}$$

$$V = 1 + \sum_{i=1}^{N_i} x_i^2, \tag{27}$$

with $\dot{g}(\cdot)$, $\ddot{g}(\cdot)$, z_k^o and z_j^h being given by (7) through (9) and (11). Moreover, when the inequality in (22) is strict, strict convexity results.

Proof: For convexity(strict convexity) of $s(\boldsymbol{w})$, we need its Hessian to be positive semidefinite(definite). From (10), (17)-(18), we know that matrices $\mathbf{J}^T\mathbf{J}$, $\mathbf{R}$, $\mathbf{I}$ are symmetric. Hence, for the Hessian $[\mathbf{J}^T\mathbf{J} + \mathbf{R} + b\mathbf{I}]$ to be positive semidefinite (definite), it is sufficient to have

$$\lambda_{min}(\mathbf{J}^T\mathbf{J}) + \lambda_{min}(\mathbf{R}) + b \geqslant 0, \tag{28}$$

by Weyl's theorem (see [10], p.181). Substitute $\lambda_{min}(\mathbf{J}^T\mathbf{J}) = 0$, (10), (12) and (13) into above, and apply Weyl's theorem again, we further have

$$b - \sum_{i=1}^{n}\lambda_m\left(\ddot{g}(z_k^o)\mathbf{P}\right)_i\epsilon_i - \sum_{i=1}^{n}\lambda_m\left(\dot{g}(z_k^o)\mathbf{Q}\right)_i\epsilon_i \geqslant 0 \tag{29}$$

where $\lambda_m(\cdot) = \lambda_{max}(\cdot)$ when $\epsilon_i \geqslant 0$ and $\lambda_m(\cdot) = \lambda_{min}(\cdot)$ when $\epsilon_i < 0$. Denote the trace of a matrix $\mathbf{A}$ by $\mathrm{tr}(\mathbf{A})$. According to Lemma 1, we have for $\epsilon_i < 0$

$$\lambda_m\left(\ddot{g}(z_k^o)\mathbf{P}\right)_i = 0,\ i = 1, 2, ..., n, \tag{30}$$

and for $\epsilon_i \geqslant 0$

$$\lambda_m\left(\ddot{g}(z_k^o)\mathbf{P}\right)_i = \left(\ddot{g}(z_k^o)\mathrm{tr}(\mathbf{r}\mathbf{r}^T)\right)_i = \left(\ddot{g}(z_k^o)\sum_{k=1}^{p} r_k^2\right)_i, \quad i = 1, 2, ..., n, \tag{31}$$

where $\mathbf{r}$ is given in (5). By Lemma 2 with adaptation of $\mathbf{A} \in \mathcal{R}^{n^2 \times n^2}$ to $\mathbf{Q} \in \mathcal{R}^{p \times p}$, $p = (N_j + 1) + (N_i + 1)N_j$, we have $2N_j$ number of non-zero eigenvalues which are identified as in (26) and (27). This completes the proof. ■

Remark 1:- By exploiting the known canonical structure of the Hessian of the FNN, Theorem 1 presents an explicit convexity condition using eigenvalue characterization. The characterization is general since it can be applied to non-sigmoidal activation functions by replacing g and its derivatives with other activation functions. Here, we note that local optimal solution set can be characterized by Proposition 1 and Theorem 1. These results will be incorporated into global optimality characterization in the following section. □

4 Global Optimality

In this context, we refer to the solution of a minimization problem as:

Definition 1 *Let f be a function to be minimized from $\mathcal{D}$ to $\mathcal{R}$ where $\mathcal{D} \subseteq \mathcal{R}^p$ is non-empty and compact. A point $\boldsymbol{\theta} \in \mathcal{D}$ is called a feasible solution to the minimization problem. If $\boldsymbol{\theta}_g^* \in \mathcal{D}$ and $f(\boldsymbol{\theta}) \geqslant f(\boldsymbol{\theta}_g^*)$ for each $\boldsymbol{\theta} \in \mathcal{D}$, then $\boldsymbol{\theta}_g^*$ is called a global optimal solution (global minimum) to the problem. If $\boldsymbol{\theta}^* \in \mathcal{D}$ and if there exists an ε-neighborhood $N_\varepsilon(\boldsymbol{\theta}^*)$ around $\boldsymbol{\theta}^*$ such that $f(\boldsymbol{\theta}) \geqslant f(\boldsymbol{\theta}^*)$ for each $\boldsymbol{\theta} \in \mathcal{D} \cap N_\varepsilon(\boldsymbol{\theta}^*)$, then $\boldsymbol{\theta}^*$ is called a local optimal solution (local minimum). The set which contains both local optimal solutions and global optimal solutions is called a solution set (denoted by Θ^*).*

4.1 Mathematical construct

Denote $\mathcal{R}^+ = (0, \infty)$. Consider a strictly decreasing transformation ϕ on the function to be minimized. We shall use the following result (see [20] for more details) for global optimality characterization of a FNN error function.

Proposition 2. *Let $f : \mathcal{D} \to \mathcal{R}$ be a continuous function where $\mathcal{D} \subseteq \mathcal{R}^p$ is compact. Let $\phi : \mathcal{R} \to \mathcal{R}^+$ be a strictly decreasing function. Suppose $\boldsymbol{\theta}^* \in \mathcal{D}$. Then $\boldsymbol{\theta}^*$ is a global minimizer of f if and only if*

$$\lim_{\gamma \to \infty} \frac{\phi^\gamma(f(\boldsymbol{\theta}))}{\phi^\gamma(f(\boldsymbol{\theta}^*))} = 0, \quad \forall\, \boldsymbol{\theta} \in \mathcal{D},\, f(\boldsymbol{\theta}) \neq f(\boldsymbol{\theta}^*). \tag{32}$$

Proof: This follows from the fact that $\frac{\phi^\gamma(f(\boldsymbol{\theta}))}{\phi^\gamma(f(\boldsymbol{\theta}^*))} > 0$ so that $\lim_{\gamma \to \infty} \frac{\phi^\gamma(f(\boldsymbol{\theta}))}{\phi^\gamma(f(\boldsymbol{\theta}^*))} = 0$ is equivalent to $\frac{\phi(f(\boldsymbol{\theta}))}{\phi(f(\boldsymbol{\theta}^*))} < 1$. ■

Consider the solution set given by Definition 1, and using a more structured convex transformation ϕ, the following proposition is a straightforward consequence.

Proposition 3. *Let $f : \mathcal{D} \to \mathcal{R}$ be a continuous function where $\mathcal{D} \subseteq \mathcal{R}^p$ is compact. Let $\phi : \mathcal{R} \to \mathcal{R}^+$ be a strictly decreasing and convex function. Denote by Θ^* the solution set given by Definition 1. Suppose $\boldsymbol{\theta}^* \in \Theta^*$. Then $\boldsymbol{\theta}^*$ is a global minimizer of f if and only if*

$$\lim_{\gamma \to \infty} \frac{\phi^\gamma(f(\boldsymbol{\theta}))}{\phi^\gamma(f(\boldsymbol{\theta}^*))} = 0, \quad \forall\, \boldsymbol{\theta} \in \Theta^*,\, f(\boldsymbol{\theta}) \neq f(\boldsymbol{\theta}^*). \tag{33}$$

4.2 Global optimality of the modified FNN error function

Consider a feedforward neural network (FNN) with the training objective given by (14). Let $v(\boldsymbol{w})$ be a convex monotonic transformation function given by:

$$v(\boldsymbol{w}) = \phi^\gamma(s(\boldsymbol{w})) = \rho e^{-\gamma s(\boldsymbol{w})}, \quad \boldsymbol{w} \in \mathcal{R}^p,\ \rho > 0,\ \gamma > 1. \tag{34}$$

Notice that $v(\boldsymbol{w}) \in \mathcal{R}^+ = (0, \infty)$ for all finite values of ρ, γ and $s(\boldsymbol{w})$. If a lower bound or the value of global minimum of $s(\boldsymbol{w})$ is known, then we can multiply $s(\boldsymbol{w})$ by $\rho = e^{\gamma s_L}$ for scaling purpose:

$$v(\boldsymbol{w}) = e^{-\gamma(s(\boldsymbol{w}) - s_L)}. \tag{35}$$

This means that the maximum value of $v(\boldsymbol{w})$ can be pivot near to or at 1 while all other local minima can be "flattened" (relative to global minima) using a sufficiently high value of γ. For FNN training adopting a l_2-norm error objective, the ultimate lower bound of $s(\boldsymbol{w})$ is zero. For network with good approximation capability, the global minimum value should be a small value where this zero bound provides a good natural scaling.

Noting that the FNN considered is a continuous function mapping on a compact set of weight space $\boldsymbol{w} \in \mathcal{W}$, characterization of global optimality for the FNN training problem is presented as follows:

Theorem 2 *Consider Problem 1. Denote by $\mathcal{W}^*$ the solution set which satisfies Proposition 1 and strict convexity in Theorem 1. Let $v(s(\boldsymbol{w})) = \rho e^{-\gamma s(\boldsymbol{w})}$, $\rho > 0$. If there exists $\gamma_o > 1$ such that*

$$\frac{v(s(\boldsymbol{w}))}{v(s(\boldsymbol{w}^*))} \leqslant \frac{1}{2}, \quad s(\boldsymbol{w}) \neq s(\boldsymbol{w}^*), \tag{36}$$

for all $\gamma \geqslant \gamma_o$ and for all $\boldsymbol{w} \in \mathcal{W}^*$, then $\boldsymbol{w}^*$ is a global minimizer of $s(\boldsymbol{w})$.

Proof: The solution set $\mathcal{W}^*$ which satisfies Proposition 1 and strict convexity in Theorem 1 defines sufficiency for local optimality of the FNN error function (14). Let $\boldsymbol{w} \in \mathcal{W}^*$ where $s(\boldsymbol{w}) \neq s(\boldsymbol{w}^*)$. By the hypothesis in the theorem, there exists $\gamma_o > 1$ such that (36) holds for all $\gamma \geqslant \gamma_o$. Thus $\frac{v(s(\boldsymbol{w}))}{v(s(\boldsymbol{w}^*))} \leqslant \frac{1}{2} < 1$ for $\gamma > \gamma_o$. Passing to limit, we have $\lim_{\gamma \to \infty} \frac{v(s(\boldsymbol{w}))}{v(s(\boldsymbol{w}^*))} = 0$. By Proposition 3, $\boldsymbol{w}^*$ is a global minimizer. ■

Remark 2:- Theorem 2 shows that if we can find a $\gamma \geqslant \gamma_o$ such that (36) is satisfied, then a level $\zeta > 0$ can be found to segregate the global minima from all

other local minima on the transformed error function v. For FNN training problems adopting a l_2-norm error function s_1, the global optimal value is expected to approach zero if network approximation capability is assumed (i.e. with sufficient layers and neurons). Hence, we can simply choose a cutting level ζ to be slightly less than 1 (since $v(0) = 1$ when $\rho = 1$) on a transformed function v with sufficiently high value of γ. We shall utilize this observation for FNN training in both regression and classification problems. □

5 Network Training

5.1 Global descent search

In this section, we show that the results of global optimality characterization can be directly applied to network training problem. Here, we treat network training as a nonlinear minimization problem. To achieve global optimality, the minimization is subjected to optimality conditions defined by Theorem 2. Mathematically, the network training can be written as:

$$\min_{\boldsymbol{w}} s(\boldsymbol{w}) \quad \text{subject to} \quad \boldsymbol{w} \in \mathcal{W}_g^*, \tag{37}$$

where $\mathcal{W}_g^*$ defines the solution set containing the global minima according to Theorem 2. Here, the condition given by Proposition 1 is used for iterative search direction design (e.g. Gauss-Newton search). While the convex characterization can be used for verification purpose, the global condition in Theorem 2 is used as a constraint to search for $\boldsymbol{w}$ over a high cutting level on the transformed function:

$$v(s(\boldsymbol{w})) > \zeta \quad or \quad h(\boldsymbol{w}) = \zeta - v(s(\boldsymbol{w})) < 0, \tag{38}$$

where ζ can be chosen to be any value within $(\frac{1}{2}, 1)$.

5.2 Penalty function method

Suppose there are l constraint functions[2] which are put into the following vector form: $\mathbf{h}(\boldsymbol{w}) = [h_1(\boldsymbol{w}), h_2(\boldsymbol{w}), \cdots, h_l(\boldsymbol{w})]^T \leqslant \mathbf{0}$. Let $\bar{h}_j(\boldsymbol{w}) = \max\{0, h_j(\boldsymbol{w})\}$, $j = 1, 2, ..., l$ and define for $j = 1, 2, ..., l$,

$$\nabla \bar{h}_j(\boldsymbol{w}) = \begin{cases} \nabla h_j(\boldsymbol{w}) \text{ if } h_j(\boldsymbol{w}) \geqslant 0 \\ \mathbf{0} \quad \text{if } h_j(\boldsymbol{w}) < 0 \ . \end{cases} \tag{39}$$

Using the more compact matrix notation, (39) can be packed for $j = 1, 2, ..., l$ as $\bar{\mathbf{H}}^T(\boldsymbol{w}) = \nabla \bar{\mathbf{h}}(\boldsymbol{w}) = [\nabla \bar{h}_1(\boldsymbol{w}), \nabla \bar{h}_2(\boldsymbol{w}), \cdots, \nabla \bar{h}_l(\boldsymbol{w})] \in \mathcal{R}^{p \times l}$. By the penalty function method [13], the constrained minimization problem of (37) can be rewritten as:

$$(P_c): \ \min \ q(c, \boldsymbol{w}) = s(\boldsymbol{w}) + cP(\boldsymbol{w}) \tag{40}$$

[2] Apart from constraint arising from (36), the boundaries of the domain of interest can also be included as constraints.

where

$$P(\boldsymbol{w}) = \psi(\bar{h}_j(\boldsymbol{w})) = \sum_{j=1}^{l} \left(\bar{h}_j(\boldsymbol{w})\right)^2, \tag{41}$$

$$\bar{h}_j(\boldsymbol{w}) = \max\{0,\, h_j(\boldsymbol{w})\}, \quad j = 1, 2, ..., l, \tag{42}$$

and $c > 0$ is a large penalty coefficient.

In [13], it has been shown that as $c \to \infty$, the solution to the minimization problem (P_c) given by (40) will converge to a solution of the original constrained problem given by (37). In the sequel, we shall concentrate on finding the solution of (P_c) (40)-(42).

For the unconstrained objective function given by $s(\boldsymbol{w})$ and the penalty function given by $P(\boldsymbol{w})$ in (40), we note that their first- and second-order partial derivatives are written as:

$$\nabla s(\boldsymbol{w}) = -2\mathbf{J}^T\boldsymbol{\epsilon} + 2b\boldsymbol{w}, \qquad \nabla^2 s(\boldsymbol{w}) = 2(\mathbf{J}^T\mathbf{J} + \mathbf{R}_y) + 2b\mathbf{I},$$
$$\nabla P(\boldsymbol{w}) = 2\bar{\mathbf{H}}^T\bar{\mathbf{h}}, \qquad \nabla^2 P(\boldsymbol{w}) = 2(\bar{\mathbf{H}}^T\bar{\mathbf{H}} + \mathbf{R}_h).$$

The functional dependency on $\boldsymbol{w}$ (and so in the subsequent derivations) are omitted when clarity is not affected. The following algorithms are derived to search for the global minima.

5.3 Algorithm

It can be further assumed that the first-order partial derivatives of $P(\boldsymbol{w})$ are continuous including those points where $\bar{h}_j(\boldsymbol{w}) = 0$, $j = 1, 2, ..., l$ following [13]. Hence, by taking the quadratic approximation of $q(\boldsymbol{w})$ (40) about $\boldsymbol{w}_o$ and set the first derivative to zero, we have

$$\boldsymbol{w} = \boldsymbol{w}_o + \left[\mathbf{J}^T\mathbf{J} + \mathbf{R}_y + b\mathbf{I} + c\bar{\mathbf{H}}^T\bar{\mathbf{H}} + \mathbf{R}_h\right]^{-1}\left(\mathbf{J}^T\boldsymbol{\epsilon} - b\boldsymbol{w} - c\bar{\mathbf{H}}^T\bar{\mathbf{h}}\right). \tag{43}$$

If we drop the second-order partial derivatives of network ($\mathbf{R}_y$) and the second-order partial derivatives of the constraint function ($\mathbf{R}_h$), we can formulate a search algorithm as follows:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \left[\mathbf{J}_i^T\mathbf{J}_i + b\mathbf{I} + c\bar{\mathbf{H}}_i^T\bar{\mathbf{H}}_i\right]^{-1}\left(\mathbf{J}_i^T\boldsymbol{\epsilon}_i - b\boldsymbol{w}_i - c\bar{\mathbf{H}}_i^T\bar{\mathbf{h}}_i\right). \tag{44}$$

By including a weighted parameter norm in the error objective function (14), we note that this has resulted in having a weighted identity matrix ($b\mathbf{I}$) included for the term in (44) which requires matrix inversion. This provides a mechanism to avoid the search from ill-conditioning which is analogous to that of the Levenberg-Marquardt's method.

To further improve numerical properties, the widely distributed eigenvalues of the penalty term can be normalized as shown:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \beta\left[\mathbf{J}_i^T\mathbf{J}_i + b\mathbf{I} + c\bar{\mathbf{H}}_i^T\mathbf{A}\bar{\mathbf{H}}_i\right]^{-1}\left(\mathbf{J}_i^T\boldsymbol{\epsilon}_i - b\boldsymbol{w} - c\bar{\mathbf{H}}_i^T\bar{\mathbf{h}}_i\right) \tag{45}$$

where $\mathbf{A} = (\bar{\mathbf{H}}_i\bar{\mathbf{H}}_i^T)^{-1}$. Here, we use the Line Search procedure (with β chosen to minimize the objective function) for iterative search.

In the following, we shall compare the global descent algorithm given by (45) (denoted as LSGO) with its local counter part, the local line search (denoted as LS) obtained from setting $c = 0$ and $b\boldsymbol{w} = 0$ in (45).

6 Numerical Experiments

6.1 Benchmark problems

For the following experiments in this subsection, the FNN learning objective is chosen to be $s_1(\boldsymbol{w})$ rather than $s(\boldsymbol{w})$ since only training accuracy for regression is needed. Here the $b\boldsymbol{w}$ term in algorithm (45) is set to zero while the $b\mathbf{I}$ term is retained for numerical stability. For all examples, 100 trials using random initial points within the box $[0, 1]^p$ were carried out. Training results in terms of the number of trials reaching a neighborhood of the desired global minimum and the mean number of iterations for these trials are presented.

As the number of iterations required to reach a desired error goal only provides a partial picture of the training algorithm, numerical results on computational aspects are also provided. All the experiments are conducted using an IBM-PC/Pentium compatible machine with 266Hz clock speed. In the following, we tabulate the average CPU time required to run 10 iterations for each algorithm.

For ease of comparison, recent results (mean number of iterations, percentage of trials attaining near global solution) for the XOR problem from [22] are listed: (i) Standard error backpropagation: (332, 91.3%); (ii) Error backpropagation with line minimization: (915.4, 38%); (iii) Davidon-Fletcher-Powell quasi-Newton minimization: (2141.1, 34.1%); (iv) Fletcher-Reeves conjugate gradient minimization: (523, 81.5%); (v) Conjugate gradient minimization with Powell restarts: (79.2, 82.1%). The reader is referred to [22] for more results on $f(x) = \sin(x)\cos(2x)$ fitting problem which requires much more than 600 mean iterations for the above search methods.

As for existing global optimization algorithms, similar statistical comparisons for these examples are not available. We note that the particular training example for XOR given in [8] (TRUST) used about 1000 training iterations to reach the global optimal solution. As for the global algorithm proposed by [19] (GOTA), the convergence speed is reported to be comparable to the backpropagation algorithm for the XOR example.

Example 1: XOR pattern

In this example, 4 samples of the XOR input-output patterns were used for network training. The network chosen was similar to that in [22] where 2 hidden-units were used. The target sum of the squared error was set to be less than 0.025 which was sufficiently closed to the global optimal solution. For both local and global methods, b was chosen to be a fixed value of 0.0001 which was sufficient to provide a stable numerical conditioning. As for other parameters of the global

descent algorithm (LSGO), the following settings were used: $\gamma = 4$, $\rho = 1$, $\zeta = 0.9999$, $c = 10$ (γ and ρ appear in $v(\boldsymbol{w})$, ζ is the cutting level on $v(\boldsymbol{w})$ as shown in (38) and c is a penalty constant).

Training results comprising of 100 trials for each of the algorithms are shown in Table 1. The respective statistics (i.e. min: minimum value, max: maximum value, mean: mean value, std dev: standard deviation, and GOP: percentage of trials achieving the desired global error objective within 500 iterations) are also included in Table 1. In order to show the core distribution for trials which took less than 500 iterations, the statistics shown exclude those trials above 500 iterations.

Table 1. Results for the Examples 1,2 and 3

Ex. 1		CPU	Number of iterations				GOP
Case	Algo.	(sec)	min	max	mean	std dev.	(%)
(a)	LS	0.66	5	> 500	13.73	22.46	33
(b)	LSGO	0.77	28	132	39.40	15.76	100
Ex. 2		CPU	Number of iterations				GOP
Case	Algo.	(sec)	min	max	mean	std dev.	(%)
(a)	LS	0.93	21	> 5000	94.70	94.76	50
(b)	LSGO	1.15	29	3191	72.34	26.78	100
Ex. 3		CPU	Number of iterations				GOP
Case	Algo.	(sec)	min	max	mean	std dev.	(%)
(a)	LS	5.50	–	> 500	–	–	0
(b)	LSGO	8.41	49	> 500	73.16	18.85	99

As shown in Table 1, the global descent algorithm (LSGO) have succeeded in locating the approximate global minima within 132 iterations for all the 100 trials using different initial values. This as compared to the local line search algorithm (LS), which scores only 33%, is a remarkable improvement. In terms of computational cost, the global constrained method is found to take slightly higher CPU time than that using unconstrained method.

Example 2: 1-D curve $f(x) = \sin(x)\cos(2x)$

In this example, 20 input-output patterns were uniformly chosen on $0 \leqslant x \leqslant 2\pi$ for network training. Similar to the first example, the sum of the squared error was set to be less than 0.025 which was sufficiently close to the global optimal solution. A single-output network with 10-hidden units was chosen according to [22]. As in previous example, b was set to be 0.0001 throughout. For the global descent algorithm (LSGO), the following settings were chosen: $\gamma = 4$, $\rho = 10$, $\zeta = 9.9999$, $c = 10$. Training results for 100 trials are shown in Table 1, with respective statistics and CPU times. For this example, GOP refers to the percentage of trials achieving the desired error goal within 5000 iterations.

In this example, the global method (LSGO) has achieved a 100% GOP which is much better than the 50% for local method (LS). The largest iteration number for the case in LSGO was found to be 3191. As for the CPU time, the global constrained algorithm takes longer time than its local counterpart in each iteration.

Example 3: 2-D shape

For this example, the network is to learn a two-dimensional *sinc* function. The network size chosen was (2-15-1) and the error goal was set at 0.8 with 289 input-output training sets. Similar to the above examples, b was chosen to be 0.0001 throughout. For the global descent algorithm (LSGO), the following settings were chosen: LSGO: $\gamma = 2$, $\rho = 10$, $\zeta = 9.999999$, $c = 1000$. The training results for 100 trials are shown in Table 1. Here we note that the GOP for this example indicates the percentage of trials reaching the error goal within 500 iterations.

From Table 1, we see that for all 100 trials, the LS method was unable to descent towards the error goal within 500 iterations. In fact, we observed that most of these trials had landed on local minima which are much higher than the error goal. The LSGO had improved the situation with only one trial resulted in SSE slightly greater than the error goal at the end of 500th iteration.

Remark 3:- Despite the remarkable convergence using random initial estimates for all the examples, it is noted that when the initial point was chosen at some of those local minima, the penalty based algorithms converge with extremely slow speed and were unable to locate the global minima within 5000 iterations. □

6.2 Face recognition

Face recognition represents a difficult classification problem since it has to deal with large amount of variations in face images due to viewpoint, illumination and expression difference even for similar person. As such, many recognition conditions are ill-posed because "the variations between the images of the same face due to illumination and viewing direction are almost always larger than image variation due to change in face identity" [15].

Here we use the ORL Cambridge database [18] for classification. The ORL database contains 40 distinct persons, each having 10 different images taken under different conditions: different times, varying lighting (slightly), facial expression (open/closed eyes, smiling/non-smiling) and facial details (glasses/no-glasses). All the images are taken against a dark homogeneous background and all persons are in up-right, frontal position except for some tolerance in side movement. The face recognition procedure is performed in two stages, namely:

1. *Feature extraction*: the training set and query set are derived in the same way as in [12] where 10 images of each of the 40 persons are randomly partitioned into two sets, resulting in 200 images for training and 200 images for testing with no overlapping between them. Each original image is then projected onto the feature spaces derived from Eigenface [21], Fisherface [3] and D-LDA [23] methods.
2. *Classification*: conventional nearest centre (NC), nearest neighbour (NN) and our proposed FNN method are used for classification. The error rates

are taken only from the averages of test errors obtained from 8 different runs.[3]

The network chosen for this application consists of 40 separate FNN (one network per person), each with (N_i-1-1) structure considering network dilution for good generalization [11]. The number of inputs $N_i = 39$ is set according to the feature dimension obtained from projections using Eigenface, Fisherface and D-LDA. During the training phase, each of the 40 FNN outputs was set to '1' for one corresponding class (person) while the others set to '0'. The global FNN was tested for 5 cases: FNN(a)-(e), each for 500 learning iterations, all using $\gamma = 4$, $\rho = 10$, $\zeta = 9.9999$. Among the 40 individual outputs of the FNNs, the output with the highest value is assigned to the corresponding class. Training and testing error rates were obtained from the number of mis-classified persons divided by 200. The results corresponding to other different FNN settings and various feature extraction methods are shown in Table 2.

Table 2. FNN settings and corresponding error rates

	Settings			Classification error rate (%)					
Classification Method	FNN settings			Eigenface		D-LDA		Fisherface	
	$\boldsymbol{w}_o$	c	b	Train	Test	Train	Test	Train	Test
NC	–	–	–	–	12.25	–	5.56	–	8.12
NN	–	–	–	–	6.50	–	5.38	–	8.81
FNN(a)	$k \times 10^{-4}$	0	10^{-4}	3.69	25.88	34.44	45.31	69.25*	79.25*
FNN(b)	$k \times 10^{-4}$	10^3	10^{-4}	3.56	26.75	34.25	47.06	69.25*	79.38*
FNN(c)	$k \times 10^{-4}$	10^3	0.05	0	5.38	0	5.62	0	12.56
FNN(d)	$\bar{k} \times 10^{-8}$	10^3	0.25	0	5.06	0	5.31	0	7.81
FNN(e)	$R \times 10^{-8}$	10^3	0.05-0.5	0	3.88	0	4.63	0	7.81

* : encounter singularity of matrix for some cases during training.

From Table 2, we see that poor results were obtained for both the conventional FNN (FNN(a): unconstrained minimization on s_1 with $c = 0$) and the global descent FNN for regression (FNN(b): constrained minimization on s_1 with $c = 10^3$). However, remarkable improvement was observed when the weightage b was set at 0.05 as seen in FNN(c). With smaller initial values ($\bar{k}$ indicates $k = 1, ..., p$ offset by its mean) and a higher b, the results were seen to improve further in FNN(d). In FNN(e), we provide our best achievable results from random initial estimates ($R \times 10^{-8}$) and variations of b value. We see that our simple training method can provide good generalization capability as compared to best known network based method that incorporated several ideas: local receptive fields, shared weights, and spatial subsampling [12]. In short, for this case of using half the data set for training, our best FNN provides a good error rate as compared to best known methods reported in [12]: Top-down HMM (13%), Eigenfaces (10.5%), Pseudo 2D-HMM (5%) and SOM+CN (3.8%).

[3] Here we note that only 3 runs were tested in [12].

7 Conclusion

In this paper, we propose to train a regularized FNN using a global search method. We have presented explicit vector and matrix canonical forms for the Jacobian and the Hessian of the FNN prior to convexity analysis of the weighted l_2-norm error function. The sufficient conditions for such convex characterization are derived. This permits direct means to analyze the network in aspects of network training and possibly network pruning. Results from the convex characterization are utilized in an attempt to characterize the global optimality of the FNN error function which is suitable for regression and classification applications. By means of convex monotonic transformation, a sufficient condition for the FNN training to attain global optimality is proposed. The theoretical results are applied directly to network training using a simple constrained search. Several numerical examples show remarkable improvement in terms of convergence of our network training as compared to available local methods. The network learning is also shown to possess good generalization property in a face recognition problem. It is our immediate task to generalize these results to a network with multiple outputs. Design of more robust constrained search remains an issue to guarantee global convergence.

Acknowledgement

The authors are thankful to Dr Geok-Choo Tan and Dr Zhongke Wu for their comments and assistance on various aspects of the paper. The authors are also thankful to the anonymous reviewers for providing additional references. Last, but not least, the authors are thankful to A/P Wee Ser for his kind support.

References

1. Etienne Barnard. Optimization for training neural nets. *IEEE Trans. on Neural Networks*, 3(2):232–240, 1992.
2. Roberto Battiti. First- and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4:141–166, 1992.
3. P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, July 1997.
4. Chris Bishop. Exact calculation of the Hessian matrix for the multilayer perceptron. *Neural Computation*, 4:494–501, 1992.
5. Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press Inc., New York, 1995.
6. Richard P. Brent. Fast training algorithms for multilayer neural networks. *IEEE Trans. on Neural Networks*, 2(3):346–354, 1991.
7. Wray L. Buntine and Andreas S. Weigend. Computing second derivatives in feed-forward networks: A review. *IEEE Tran. Neural Networks*, 5(3):480–488, 1994.
8. B. C. Cetin, J. W. Burdick, and J. Barhen. Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks. In *IEEE Int. Conf. Neural Networks*, 1993.

9. John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, New York, 1991.
10. Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, 1992.
11. Peter Kuhlmann and Klaus-Robert Müller. On the generalisation ability of diluted perceptrons. *J. Phys. A: Math. Gen.*, 27:3759–3774, 1994.
12. Steve Lawrence, C. Lee Giles, Ah Chung Tsoi, and Andrew D. Back. Face recognition: A convolutional neural-network approach. *IEEE Tran. Neural Networks*, 8(1):98–113, 1997.
13. David G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, Inc., Massachusetts, 1984.
14. Martin Fodslette Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.
15. Y. Moses, Y. Adini, and S. Ullman. "Face recognition: The problem of compensating for changes in illumination direction". In *Proceedings of the European Conference on Computer Vision*, volume A, pages 286–296, 1994.
16. Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 4:147–160, 1994.
17. R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1972.
18. Web site of ORL Cambridge face database:. "http://www.cam-orl.co.uk/facedatabase.html". *AT&T Laboratories Cambridge*, 1994.
19. Zaiyong Tang and Gary J. Koehler. Deterministic global optimal FNN training algorithms. *Neural Networks*, 7(2):301–311, 1994.
20. K. A. Toh. Global energy minimization: A transformation approach. In *Proceedings of Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*. Springer Verlag, Sophia-Antipolis, France, September 2001. (Lecture Notes in Computer Science).
21. Matthew A. Turk and Alex P. Pentland. "Eigenfaces for recognition". *Journal of Cognitive Neuroscience*, 3(1):71–86, March 1991.
22. P. Patrick van der Smagt. Minimisation methods for training feedforward neural networks. *Neural Networks*, 7(1):1–11, 1994.
23. Jie Yang, Hua Yu, and William Kunz. "An efficient LDA algorithm for face recognition". In *Proceedings of Sixth International Conference on Control, Automation, Robotics and Vision*, Singapore, December 2000.