Iosifidis, A., Tefas, A., & Pitas, I. (2015). Single-Hidden Layer Feedforward Neual Network Training Using Class Geometric Information. In . J. J. Merelo, A. Rosa, J. M. Cadenas, A. Dourado, K. Madani, & J. Filipe (Eds.), Computational Intelligence: International Joint Conference, IJCCI 2014 Rome, Italy, October 22-24, 2014 Revised Selected Papers. (Vol. III, pp. 351-364). (Studies in Computational Intelligence; Vol. 620). Springer. DOI: 10.1007/978-3-319-26393-9_21

Single-hidden Layer Feedforward Neual network training using class geometric information

Alexandros Iosifidis, Anastasios Tefas and Ioannis Pitas

Abstract Single-hidden Layer Feedforward (SLFN) networks have been proven to be effective in many pattern classification problems. In this chapter, we provide an overview of a relatively new approach for SLFN network training that is based on Extreme Learning. Subsequently, extended versions of the Extreme Learning Machine algorithm that exploit local class data geometric information in the optimization process followed for the calculation of the network output weights are discussed. An experimental study comparing the two approaches on facial image classification problems concludes this chapter.

1 INTRODUCTION

Single-hidden Layer Feedforward (SLFN) networks have been proven to be effective in many pattern classification problems, since they are able to approximate any continuous function arbitrary well [1]. Extreme Learning Machine is a relatively new algorithm for Single-hidden Layer Feedforward Neural (SLFN) networks training [2] that leads to fast network training requiring low human supervision. Conventional SLFN network training algorithms require the input weights and the hidden layer biases to be adjusted using a parameter optimization approach, like gradient descend. However, gradient descend-based learning techniques are generally slow and may decrease the network's generalization ability, since they may lead to local minima. Unlike the popular thinking that the network's parameters need to be tuned, in ELM the input weights and the hidden layer biases are randomly assigned. The network output weights are, subsequently, analytically calculated. ELM not only tends to reach the smallest training error, but also the smallest norm of output weights. For feedforward networks reaching a small training error, the smaller

Alexandros Iosifidis, Anastasios Tefas and Ioannis Pitas
Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece,
e-mail: {aiosif,tefas,pitas}@aiia.csd.auth.gr

the norm of weights is, the better generalization performance the networks tend to have [3]. Despite the fact that the determination of the network hidden layer output is a result of randomly assigned weights, it has been shown that SLFN networks trained by using the ELM algorithm have the properties of global approximators [4]. Due to its effectiveness and its fast learning process, the ELM network has been widely adopted in many classification problems, including facial image classification [5, 6, 7, 8, 9, 10, 11, 12, 13, 14].

Despite its success in many classification problems, the ability of the original ELM algorithm to calculate the output weights is limited due to the fact that the network hidden layer output matrix is, usually, singular. In order to address this issue, the Effective ELM (EELM) algorithm has been proposed in [15], where the strictly diagonally dominant criterion for nonsingular matrices is exploited, in order to choose proper network input weights and bias values. However, the EELM algorithm has been designed only for a special case of SLFN networks employing Gaussian Radial Basis Functions (RBF) for the input layer neurons. In [9], an optimization-based regularized version of the ELM algorithm (RELM) aiming at both overcoming the full rank assumption for the network hidden layer output matrix and at enhancing the generalization properties of the ELM algorithm has been proposed. RELM has been evaluated on a large number of classification problems providing very satisfactory classification performance.

By using a sufficiently large number of hidden layer neurons, the ELM classification scheme, when approached from a Subspace Learning point of view, can be considered as a learning process formed by two processing steps [16]. The first step corresponds to a mapping process of the input space to a high-dimensional feature space preserving some properties of interest for the training data. In the second step, an optimization scheme is employed for the determination of a linear projection of the high-dimensional data to a low-dimensional feature space determined by the network target vectors, where classification is performed by a linear classifier. Based on this observation, the RELM algorithm has been extended in order to exploit subspace learning criteria in its optimization process [16, 18]. Specifically, it has been shown that the incorporation of the within-class and total scatter of the training data (represented in the feature space determined by the network hidden layer outputs) in the optimization process followed for the calculation of the network output weights enhances the network classification performance.

In this Chapter, we provide an overview of the ELM algorithm for SLFN network training [2, 9]. Extensions of the ELM algorithm exploiting subspace learning criteria on its optimization process are also described. Subsequently, an extension of the ELM algorithm which exploits local class information in the ELM optimization problem is described in detail. The so-called Local Class Variance ELM (LCVELM) algorithm aims at minimizing both the network output weights norm and the within class variance of the training data in the ELM space, expressed by employing locality constraints. An experimental study comparing the performance of ELM [2], RELM [9], MCVELM [16] and LCVELM [30] networks is facial image classification problems is finally provided.

The Chapter is structured as follows. In Section 2 we briefly describe the ELM algorithms. RELM is described in Section 2. In Section 4 ELM algorithms exploiting dispersion information in their optimization problem are described. In Section 5, we describe the LCVELM algorithm exploiting intrinsic graph structures for SLFN network training. Section 7 presents an experimental study evaluating the performance of ELM-based classification in facial image classification problems.Finally, conclusions are drawn in Section 8.

2 Extreme Learning Machine

ELM network has been proposed for SLFN network-based classification [2]. Let us denote by $\{\mathbf{x}_i, c_i\}$, $i = 1,\ldots,N$ a set of N vectors $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding class labels $c_i \in \{1,\ldots,C\}$. We employ $\{\mathbf{x}_i, c_i\}$, $i = 1,\ldots,N$ in order to train a SLFN network. Such a network consists of D input (equal to the dimensionality of $\mathbf{x}_i$), L hidden and C output (equal to the number of classes involved in the classification problem) neurons. The number of hidden layer neurons is usually selected to be much greater than the number of classes [9, 16], i.e., $L \gg C$.

The network target vectors $\mathbf{t}_i = [t_{i1},...,t_{iC}]^T$, each corresponding to a training vector $\mathbf{x}_i$, are set to $t_{ik} = 1$ for vectors belonging to class k, i.e., when $c_i = k$, and to $t_{ik} = -1$ when $c_i \neq k$. In ELMs, the network input weights $\mathbf{W}_{in} \in \mathbb{R}^{D\times L}$ and the hidden layer bias values $\mathbf{b} \in \mathbb{R}^L$ are randomly assigned, while the network output weights $\mathbf{W}_{out} \in \mathbb{R}^{L\times C}$ are analytically calculated. Let us denote by $\mathbf{v}_j$ the j-th column of $\mathbf{W}_{in}$, by $\mathbf{w}_k$ the k-th row of $\mathbf{W}_{out}$ and by w_{kj} the j-th element of $\mathbf{w}_k$. Given activation function for the network hidden layer $\Phi(\cdot)$ and by using a linear activation function for the network output layer, the output $\mathbf{o}_i = [o_1,\ldots,o_C]^T$ of the network corresponding to $\mathbf{x}_i$ is calculated by:

$$o_{ik} = \sum_{j=1}^{L} w_{kj}\, \Phi(\mathbf{v}_j, b_j, \mathbf{x}_i),\ k = 1,...,C. \tag{1}$$

It has been shown [4, 31, 9] that, almost any nonlinear piecewise continuous activation function $\Phi(\cdot)$ can be used for the calculation of the network hidden layer outputs, like the sigmoid, sine, Gaussian, hard-limiting and Radial Basis Function (RBF), Fourier series, etc. The most widely adopted choice is the sigmoid function, defined by:

$$\Phi(\mathbf{v}_j, b_j, \mathbf{x}_i) = \frac{1}{1 + e^{-(\mathbf{v}_j^T\, \mathbf{x}_i + b_j)}}. \tag{2}$$

By storing the network hidden layer outputs corresponding to the training vectors $\mathbf{x}_i$, $i = 1,\ldots,N$ in a matrix Φ:

$$\boldsymbol{\Phi} = \begin{bmatrix} \Phi(\mathbf{v}_1, b_1, \mathbf{x}_1) & \cdots & \Phi(\mathbf{v}_1, b_1, \mathbf{x}_N) \\ \cdots & \ddots & \cdots \\ \Phi(\mathbf{v}_L, b_L, \mathbf{x}_1) & \cdots & \Phi(\mathbf{v}_L, b_L, \mathbf{x}_N) \end{bmatrix}, \tag{3}$$

equation (1) can be expressed in a matrix form as:

$$\mathbf{O} = \mathbf{W}_{out}^T \boldsymbol{\Phi}. \tag{4}$$

ELM [2] assumes that the predicted network outputs are equal to the network targets, i.e., $\mathbf{o}_i = \mathbf{t}_i$, $i = 1, ..., N$, $\mathbf{W}_{out}$ can be analytically calculated by solving the following set of equation:

$$\mathbf{W}_{out}^T \boldsymbol{\Phi} = \mathbf{T} \tag{5}$$

and are given by:

$$\mathbf{W}_{out} = \boldsymbol{\Phi}^{\dagger} \mathbf{T}^T, \tag{6}$$

where $\boldsymbol{\Phi}^{\dagger} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T\right)^{-1} \boldsymbol{\Phi}$ is the Moore-Penrose generalized pseudo-inverse of $\boldsymbol{\Phi}^T$ and $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$ is a matrix containing the network target vectors.

3 Regularized Extreme Learning Machine

The ELM algorithm assumes zero training error. In cases where the training data contain outliers, this assumption may reduce its potential in generalization. In addition, since the dimensionality of the ELM space is usually high, i.e., in some cases $L > N$, the matrix $\mathbf{B} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$ is singular and, thus, the adoption of (6) for the calculation of the network output weights is inappropriate. By allowing small training errors and trying to minimize the norm of the network output weights, $\mathbf{W}_{out}$ can be calculated by minimizing [9]:

$$\mathscr{J}_{RELM} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\xi_i\|_2^2, \tag{7}$$

$$\mathbf{W}_{out}^T \boldsymbol{\phi}_i = \mathbf{t}_i - \xi_i, \; i = 1, ..., N, \tag{8}$$

where $\xi_i \in \mathbb{R}^C$ is the error vector corresponding to $\mathbf{x}_i$ and c is a parameter denoting the importance of the training error in the optimization problem. $\boldsymbol{\phi}_i$ is the i-th column of $\boldsymbol{\Phi}$, i.e., the hidden layer output corresponding $\mathbf{x}_i$. That is, $\boldsymbol{\phi}_i$ is the representation of $\mathbf{x}_i$ in $\mathbb{R}^L$. By substituting (8) in $\mathscr{J}_{RELM}$ (7) and determining the saddle point of $\mathscr{J}_{RELM}$, $\mathbf{W}_{out}$ is given by:

$$\mathbf{W}_{out} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \frac{1}{c}\mathbf{I}\right)^{-1} \boldsymbol{\Phi}\mathbf{T}^T \tag{9}$$

or

$$\mathbf{W}_{out} = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \frac{1}{c}\mathbf{I}\right)^{-1}\mathbf{T}^T \tag{10}$$

The adoption of (9) for $\mathbf{W}_{out}$ calculation, instead of (6), has the advantage that the matrices $\mathbf{B} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \frac{1}{c}\mathbf{I}\right)$ and $\tilde{\mathbf{B}} = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \frac{1}{c}\mathbf{I}\right)$ are nonsingular, for $c > 0$.

4 Extreme Learning Machine exploiting dispersion criteria

By allowing small training errors and trying to minimize both the norm of the network output weights and the within-class variance of the training vectors in the feature space determined by the network outputs, $\mathbf{W}_{out}$ can be calculated by minimizing [16]:

$$\mathscr{J}_{MCVELM} = \|\mathbf{S}_w^{\frac{1}{2}}\mathbf{W}_{out}\|_F^2 + \lambda\sum_{i=1}^{N}\|\xi_i\|_2^2, \tag{11}$$

$$\mathbf{W}_{out}^T\phi_i = \mathbf{t}_i - \xi_i,\ i = 1,...,N, \tag{12}$$

where $\mathbf{S}_w$ is the within-class scatter matrix used in Linear Discriminant Analysis (LDA) [17] describing the variance of the training classes in the ELM space and is defined by:

$$\mathbf{S}_w = \sum_{j=1}^{C}\sum_{i,c_i=j}\frac{1}{N_j}(\phi_i - \mu_j)(\phi_i - \mu_j)^T. \tag{13}$$

In (13), N_j is the number of training vectors belonging to class j and $\mu_j = \frac{1}{N_j}\sum_{i,c_i=j}\phi_i$ is the mean vector of class j. By calculating the within-class scatter matrix in the ELM space $\mathbb{R}^L$, rather than in the input space $\mathbb{R}^D$, nonlinear relationships between training vectors forming the various classes can be better described. By substituting (12) in $\mathscr{J}_{MCVELM}$ and determining the saddle point of $\mathscr{J}_{MCVELM}$, $\mathbf{W}_{out}$ is given by:

$$\mathbf{W}_{out} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \frac{1}{c}\mathbf{S}_w\right)^{-1}\boldsymbol{\Phi}\mathbf{T}^T. \tag{14}$$

Since the matrix $\mathbf{B} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \frac{1}{c}\mathbf{S}_w\right)$ is not always nonsingular, an additional dimensionality reduction processing step perfomred by applying Principal Component Analysis [17] on $\boldsymbol{\Phi}$ has been proposed in [16]. Another variants that exploiting the total scatter matrix of the entire training set and the within-class variance of multi-modal classes have been proposed in [18] and [16], respectively.

5 Extreme Learning Machine exploiting intrinsic graph structures

In this Section, we describe the an extension of the ELM algorithm exploiting local class dispersion criteria [30]. Similar to the ELM variance described in Section 2, the Local Class Variance ELM (LCVELM) algorithm exploits randomly assigned network input weights $\mathbf{W}_{in}$ and bias values $\mathbf{b}$, in order to perform a nonlinear mapping of the data in the (usually high-dimensional) ELM space $\mathbb{R}^L$. After the network hidden layer outputs calculation, we assume that the data representations in the ELM space $\boldsymbol{\phi}_i$, $i = 1,\ldots,N$ are embedded in a graph $\mathcal{G} = \{\mathcal{V},\mathcal{E},\mathbf{W}\}$, where $\mathcal{V}$ denotes the graph vertex set, i.e., $\mathcal{V} = \{\boldsymbol{\phi}_i\}_{i=1}^N$, $\mathcal{E}$ is the set of edges connecting $\boldsymbol{\phi}_i$, and $\mathbf{W} \in \mathbb{R}^{N\times N}$ is the matrix containing the weight values of the edge connections. Let us define a similarity measure $s(\cdot,\cdot)$ that will be used in order to measure the similarity between two vectors [19]. That is, $s_{ij} = s(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j)$ is a value denoting the similarity between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$. $s(\cdot,\cdot)$ may be any similarity measure providing non-negative values (usually $0 \leq s_{ij} \leq 1$). The most widely adopted choice is the heat kernel (also known as diffusion kernel) [20], defined by:

$$s(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j) = \exp\left(-\frac{\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\|_2^2}{2\sigma^2}\right), \tag{15}$$

where $\|\cdot\|_2$ denotes the (squared) l_2 norm of a vector and σ is a parameter used in order to scale the Euclidean distance between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$.

In order to express the local intra-class relationships of the training data in the ELM space, we exploit the following two choices for the determination of the weight matrix $\mathbf{W}$:

$$W_{ij}^{(1)} = \begin{cases} 1 \;\; if \; c_i = c_j \; and \; j \in \mathcal{N}_i, \\ 0, \; otherwise, \end{cases}$$

or

$$W_{ij}^{(2)} = \begin{cases} s_{ij} \; if \; c_i = c_j \; and \; j \in \mathcal{N}_i, \\ 0, \; otherwise. \end{cases}$$

In the above, $\mathcal{N}_i$ denotes the neighborhood of $\boldsymbol{\phi}_i$ (we have employed 5-NN graphs in all our experiments). $\mathbf{W}^{(1)}$ has been successfully exploited for discriminant subspace learning in Marginal Discriminant Analysis (MDA) [19], while $\mathbf{W}^{(2)}$ can be considered to be modification of $\mathbf{W}^{(1)}$, exploiting geometric information of the class data. A similar weight matrix has also been exploited in Local Fisher Discriminant Analysis (LFDA) [21]. In both MDA and LFDA cases, it has been shown that by exploiting local class information enhanced class discrimination can be achieved, when compared to the standard LDA approach exploiting global class information, by using (13).

After the calculation of the graph weight matrix $\mathbf{W}$, the graph Laplacian matrix $\mathbf{L}^{N\times N}$ is given by [22]:

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{16}$$

where $\mathbf{D}$ is a diagonal matrix with elements $D_{ii} = \sum_{j=1}^N W_{ij}$.

By exploiting $\mathbf{L}$, the network output weights $\mathbf{W}_{out}$ of the LCVELM network can be calculated by minimizing:

$$\mathscr{J}_{LCVELM} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\xi_i\|_2^2 + \frac{\lambda}{2}tr\left(\mathbf{W}_{out}^T(\Phi\mathbf{L}\Phi^T)\mathbf{W}_{out}\right), \tag{17}$$

$$\mathbf{W}_{out}^T\phi_i = \mathbf{t}_i - \xi_i,\ i = 1,...,N, \tag{18}$$

where $tr(\cdot)$ is the trace operator. By substituting the constraints (18) in $\mathscr{J}_{LCVELM}$ and determining the saddle point of $\mathscr{J}_{LCVELM}$, the network output weights $\mathbf{W}_{out}$ are given by:

$$\mathbf{W}_{out} = \left(\Phi\left(\mathbf{I} + \frac{\lambda}{c}\mathbf{L}\right)\Phi^T + \frac{1}{c}\mathbf{I}\right)^{-1}\Phi\mathbf{T}^T. \tag{19}$$

Similar to (9), the calculation of the network output weights by employing (19) has the advantage that the matrix $\mathbf{B} = \left(\Phi\left(\mathbf{I} + \frac{\lambda}{c}\mathbf{L}\right)\Phi^T + \frac{1}{c}\mathbf{I}\right)$ is nonsingular, for $c > 0$. In addition, the calculation of the graph similarity values $s(\cdot,\cdot)$ in the ELM space $\mathbb{R}^L$, rather than the input space $\mathbb{R}^D$ has the advantage that nonlinear relationships between the training vectors forming the various classes can be better expressed.

6 Data classification (test phase)

After the determination of the network output weights $\mathbf{W}_{out}$ by using (9), (10), (14) or (19), a test vector $\mathbf{x}_t \in \mathbb{R}^D$ can be introduced to the trained network and the corresponding network output is obtained:

$$\mathbf{o}_t = \mathbf{W}_{out}^T\phi_t, \tag{20}$$

where ϕ_t denotes the network hidden layer output for $\mathbf{x}_t$. $\mathbf{x}_t$ is finally classified to the class corresponding to the maximal network output:

$$c_t = \underset{k}{argmax}\, o_{tk},\ k = 1,\ldots,C. \tag{21}$$

7 Experimental study

In this section, we present experiments conducted in order to evaluate the performance of the ELM algorithms described in Sections 2, 4 and 5. We have employed six publicly available datasets to this end. These are: the ORL, AR and Extended YALE-B (face recognition) and the COHN-KANADE, BU and JAFFE (facial ex-

pression recognition). A brief description of the datasets is provided in the following subsections. Experimental results are provided in subsection 7.3.

In all the presented experiments we compare the performance of the LCVELM algorithm [30] with that of ELM [2], RELM [9] and MCVELM [16] algorithms. The number of hidden layer neurons has been set equal to $L = 1000$ for all the ELM variants, a value that has been shown to provide satisfactory performance in many classification problems [9, 16]. For fair comparison, in all the experiments, we make sure that the the same ELM space is used in all the ELM variants. That is, we first map the training data in the ELM space and, subsequently, calculate the network output weights according to each ELM algorithm. Regarding the optimal values of the regularization parameters c, λ used in the ELM-based classification schemes, they have been determined by following a grid search strategy. That is, for each classifier, multiple experiments have been performed by employing different parameter values ($c = 10^r$, $r = -3, \ldots, 3$ and $\lambda = 10^p$, $p = -3, \ldots, 3$) and the best performance is reported.

7.1 Face recognition datasets

7.1.1 The ORL dataset

It consists of 400 facial images depicting 40 persons (10 images each) [23]. The images were captured at different times and with different conditions, in terms of lighting, facial expressions (smiling/not smiling) and facial details (open/closed eyes, with/without glasses). Facial images were taken in frontal position with a tolerance for face rotation and tilting up to 20 degrees. Example images of the dataset are illustrated in Figure 1.

Fig. 1 *Facial images depicting a person from the ORL dataset.*

7.1.2 The AR dataset

It consists of over 4000 facial images depicting 70 male and 56 female faces [24]. In our experiments we have used the preprocessed (cropped) facial images provided by the database, depicting 100 persons (50 males and 50 females) having a frontal facial pose, performing several expressions (anger, smiling and screaming), in different illumination conditions (left and/or right light) and with some occlusions

(sun glasses and scarf). Each person was recorded in two sessions, separated by two weeks. Example images of the dataset are illustrated in Figure 2.

Fig. 2 *Facial images depicting a person from the AR dataset.*

7.1.3 The Extended YALE-B dataset

It consists of facial images depicting 38 persons in 9 poses, under 64 illumination conditions [25]. In our experiments we have used the frontal cropped images provided by the database. Example images of the dataset are illustrated in Figure 3.

Fig. 3 *Facial images depicting a person from the Extended YALE-B dataset.*

7.2 Facial expression recognition datasets

7.2.1 The COHN-KANADE dataset

It consists of facial images depicting 210 persons of age between 18 and 50 (69% female, 31% male, 81% Euro-American, 13% Afro-American and 6% other groups) [26]. We have randomly selected 35 images for each facial expression, i.e., anger, disgust, fear, happyness, sadness, surprise and neutral. Example images of the dataset are illustrated in Figure 4.

7.2.2 The BU dataset

It consists of facial images depicting over 100 persons (60% feamale and 40% male) with a variety of ethnic/racial background, including White, Black, East-Asian, Middle-east Asian, Hispanic Latino and others [27]. All expressions, except the

Fig. 4 *Facial images from the COHN-KANADE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

neutral one, are expressed at four intensity levels. In our experiments, we have employed the images depicting the most expressive intensity of each facial expression. Example images of the dataset are illustrated in Figure 5.

Fig. 5 *Facial images depicting a person from the BU dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

7.2.3 The JAFFE dataset

It consists of 210 facial images depicting 10 Japanese female persons [28]. Each of the persons is depicted in 3 images for each expression. Example images of the dataset are illustrated in Figure 6.

Fig. 6 *Facial images depicting a person from the JAFFE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

7.3 Results

In our first set of experiments, we have applied the competing algorithms on the face recognition datasets. Since there is not a widely adopted experimental protocol for these datasets, we randomly partition the datasets in training and test sets as follows: we randomly select a subset of the facial images depicting each of the persons in each dataset in order to form the training set and we keep the remaining facial images for evaluation. We create five such dataset partitions, each corresponding to a different training set cardinality. Experimental results obtained by applying the competing algorithms are illustrated in Tables 1, 2 and 3 for the ORL, AR and the Extended Yale-B datasets, respectively. As can be seen in these Tables, the incorporation of local class information in the optimization problem used for the determination of the network output weights, generally increases the performance of the ELM network. In all the cases the best performance is achieved by one of the two LCVELM variants. By comparing the two LCVELM algorithms, it can be seen that the one exploiting the graph weight matrix used in MDA generally outperforms the remaining choice.

Table 1 Classification rates on the ORL dataset.

	ELM	RELM	MCVELM	**LCVELM (1)**	**LCVELM (2)**
10%	30.78%	40.65%	41.01%	**41.26**%	**41.22**%
20%	20.67%	39.76%	**41.81**%	**41.81**%	**41.81**%
30%	38.17%	52.11%	55%	**55.78**%	**55.78**%
40%	38.31%	53%	57%	**57.19**%	**57.13**%
50%	47%	77.62%	75.54%	**77.69**%	**77.77**%

Table 2 Classification rates on the AR dataset.

	ELM	RELM	MCVELM	**LCVELM (1)**	**LCVELM (2)**
10%	66.47%	67.79%	68.87%	**69.19**%	**69.15**%
20%	70.49%	80.24%	80.91%	80.86%	**80.96**%
30%	65.26%	82.98%	81.81%	**83.27**%	**83.1**%
40%	75.33%	91.9%	92.94%	**93.01**%	**93.01**%
50%	80.33%	94.16%	94.65%	**94.9**%	**94.9**%

In our second set of experiments, we have applied the competing algorithms on the facial expression recognition datasets. Since there is not a widely adopted experimental protocol for these datasets too, we apply the five-fold crossvalidation procedure [29] by employing the facial expression labels. That is, we randomly split the facial images depicting the same expression in five sets and we use five splits of all the expressions for training and the remaining splits for evaluation. This process is performed five times, one for each evaluation split. Experimental results obtained

Table 3 Classification rates on the YALE-B dataset.

	ELM	RELM	MCVELM	**LCVELM (1)**	**LCVELM (2)**
10%	69.17%	**72.22**%	**72.22**%	**72.22**%	**72.22**%
20%	83.44%	84.38%	84.38%	**85**%	84.38%
30%	82.86%	85.36%	85.36%	**88.21**%	85.36%
40%	90%	92.08%	92.08%	**92.5**%	92.08%
50%	91%	93.5%	**94.5**%	**94.5**%	**94.5**%

by applying the competing algorithms are illustrated in Table 4. As can be seen in this Table, the LCVELM algorithms outperform the remaining choices in all the cases.

Table 4 Classification rates on the facial expression recognition dataset.

	ELM	RELM	MCVELM	**LCVELM (1)**	**LCVELM (2)**
COHN-KANADE	49.8%	79.59%	80%	**80.41**%	80%
BU	65%	71,57%	71,57%	**72**%	**72,86**%
JAFFE	47.62%	58.57%	59.05%	**60**%	**59.52**%

Overall, enhanced facial image classification performance can be achieved by exploiting class data geometric information in the ELM optimization process.

8 Conclusion

In this chapter an overview of Extreme Learning Machine-based Single-hidden Layer Feedforward Neural networks training has been provided. Extended versions of the ELM algorithm that exploit (local) class data geometric information in the optimization process followed for the calculation of the network output weights have been also described. An experimental study comparing the two approaches on facial image classification problems has been finally presented, showing that the exploitation of class data geometric information in the ELM optimization process enhances the performance of the ELM network.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 316564 (IMPART).

References

1. Scarselli, F and Tsoi, A.C. (1998) Universal approximation using feedforward neural networks: A survey of some existing methods and some new results. Neural Networks, 11:15-37
2. Huang, G.B., Zhu, Q.Y. and Siew, C.K. (2004) Extreme Learning Machine: a new learning scheme of feedforward neural networks. IEEE International Joint Conference on Neural Networks
3. Bartlett, P.L. (1998) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Transactions on Information Theory 44(2):525–536
4. Huang, G.B., Chen, L. and Siew, C.K. (2006) Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Nodes. IEEE Transactions on Neural Networks 17(4):879–892
5. Zong, W. and Huang, G.B. (2011) Face recognition based on Extreme Learning Machine. Neurocomputing 74(16):2541–2551
6. Rong, H.J. and Huang, G.B. and Ong, S.Y. (2008) Extreme learning machine for multicategories classification applications. IEEE International Joint Conference on Neural Networks
7. Lan, Y. and Soh, Y.C. and Huang, G.B. (2008) Extreme Learning Machine based bacterial protein subcellular localization prediction. IEEE International Joint Conference on Neural Networks
8. Helmy, T. and Sashee, Z. 2009 Multi-category bioinformatics dataset classification using extreme learning machine. IEEE Evolutionary Computation
9. Huang, G.B., Zhou, H. and DIng, X. and Zhang, R. (2012) Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42(2):513–529
10. Iosifidis, A., Tefas, A. and Pitas, I. (2013) Person Identification from Actions based on Artificial Neural Networks. IEEE Symposium Series on Computational Intelligence
11. Iosifidis, A., Tefas, A. and Pitas, I. (2013) Dynamic action recognition based on Dynemes and Extreme Learning Machine. Pattern Recognition Letters 34:1890–1989
12. Iosifidis, A., Tefas, A. and Pitas, I. (2013) Active Classification for Human Action Recognition. IEEE International Conference on Image Processing
13. Iosifidis, A., Tefas, A. and Pitas, I. (2014) Human Action Recognition based on Bag of Features and Multi-view Neural Networks. IEEE International Conference on Image Processing
14. Iosifidis, A., Tefas, A. and Pitas, I. (2014) Semi-supervised classification of human actions based on Neural Networks. IEEE International Conference on Pattern Recognition
15. Wang, Y. and Cao, F. and Yuan, Y. (2011) A study on effectiveness of extreme learning machine. Neurocomputing 74(16):2483–2490
16. Iosifidis, A., Tefas, A. and Pitas, I. (2013) Minimum Class Variance Extreme Learning Machine for Human Action Recognition. IEEE Transactions on Circuits and Systems for Video Technology 23(11):1968–1979
17. Duda, R.O., Hart, P.E. and Stork, D.G. (2000) Pattern Classification, 2nd ed. Wiley-Interscience
18. Iosifidis, A., Tefas, A. and Pitas, I. (2014) Minimum Variance Extreme Learning Machine for Human Action Recognition. IEEE International Conference on Acoustics, Speech and Signal Processing
19. Yan, S., Xu, D., Zhang, B., Zhang, H.J., Yang, Q., and Lin, S. (2007) Graph Embedding and Extensions: A General Framework for Dimensionality Reduction. IEEE Transactions on on Pattern Analysis ans Machine Intelligence 29(1):40–50
20. Kondor, R.I. and Lafferty, J.D. (2002) Diffusion kernels on graphs and other discrete input spaces. International Conference on Machine Learning
21. Sugiyama, M. (2007) Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis. Journal of Machine Learning Research 8:1027–1061

22. Belkin, M., Niyogi, P. and Sindhwani, V. (2007) Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of Machine Learning Research 7:2399–2434
23. Samaria, F. and Harter, A. (1994) Parameterisation of a stochastic model for human face identification. IEEE Workshop on Applications of Computer Vision
24. Martinez, A. and Kak, A. (2007) PCA versus LDA. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(2):228–233
25. Lee, K.C., Ho, J. and Kriegman, D. (2007) Acquiriing linear subspaces for face recognition under varialbe lighting. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(5):684–698
26. Kanade, T., Tian, Y. and Cohn, J. (2000) Comprehensive database for facial expression analysis. IEEE International Conference on Automatic Face and Gesture Recognition
27. Yin, L., Wei, X., Sun, Y., Wang, J. and Rosato, M. (2006) A 3D facial expression database for facial behavior research. IEEE International Conference on Automatic Face and Gesture Recognition
28. Lyons, M., Akamatsu, S., Kamachi, M. and Gyoba, J. (1998) Coding facial expressions with Gabor wavelets. IEEE International Conference on Automatic Face and Gesture Recognition
29. Devijver, R. and Kittler, J. (1982) Pattern Recognition: A Statistical Approach. Prentice-Hall
30. Iosifidis, A., Tefas, A. and Pitas, I. (2014) Exploiting Local Class Information in Extreme Learning Machine. International Joint Conference on Computational Intelligence
31. Huang, G. and Chen, L. (2008) Convex incremental extreme learning machine. Neurocomputing 70(16):3056–3062