Decomposition Techniques for Multilayer Perceptron Training

Luigi Grippo, Andrea Manno, and Marco Sciandrone

***Abstract*— In this paper, we consider the learning problem of multilayer perceptrons (MLPs) formulated as the problem of minimizing a smooth error function. As well known, the learning problem of MLPs can be a difficult nonlinear nonconvex optimization problem. Typical difficulties can be the presence of extensive flat regions and steep sided valleys in the error surface, and the possible large number of training data and of free network parameters. We define a wide class of batch learning algorithms for MLP, based on the use of block decomposition techniques in the minimization of the error function. The learning problem is decomposed into a sequence of smaller and structured minimization problems in order to advantageously exploit the structure of the objective function. Theoretical convergence results are established, and a specific algorithm is constructed and evaluated through an extensive numerical experimentation. The comparisons with the state-of-the-art learning algorithms show the effectiveness of the proposed techniques.**

***Index Terms*— Decomposition techniques, extreme learning machine (ELM), feedforward neural networks, multilayer perceptrons (MLPs), optimization methods for supervised learning.**

I. Introduction

MULTILAYER feedforward neural networks, also known as multilayer perceptrons (MLPs), do constitute one of the most popular classes of neural networks for classification and regression (see [12]). As it is known, an MLP consists of an input layer that simply receives the external inputs, and of a set of neural units organized into $L \geq 1$ hidden layers and one output layer. To each neural unit in the hidden layer or in the output layer, we associate an activation function and a vector of weights of the inputs to the unit, augmented with a threshold parameter.

We are interested in supervised learning techniques for MLPs, which determine the free parameters of the neural model, on the basis of a training set of input–output patterns. We will refer, in particular, to single-hidden layer feedforward networks (SLFNs) with linear output unit. However, the case of multiple hidden layers can also be considered and will be briefly discussed in the concluding remarks. To simplify our analysis, we also assume that the network has a scalar output; the extension of our results to the case of multiple outputs is conceptually immediate, but notation becomes more complicated.

It is well known that an SLFN network already possesses the interpolation capability and the universal approximation capability in the space of continuous functions on compacta (see [24]), if and only if the activation function is not a polynomial. Supervised learning for an SLFN consists in choosing with some rules the number N of hidden units and in determining the parameters associated with each hidden unit (weights, thresholds) and the output weights.

For a fixed N, the learning problem has been traditionally formulated as the problem of minimizing a suitable error function that measures the error on the outputs, in correspondence to a given set of training pairs, possibly augmented with some regularization terms.

The minimization of the training error can be performed either by a batch or by an incremental (possibly online) strategy [12]. In this paper, we assume that the training set is available *a priori*, and hence, we focus on batch training algorithms. We denote the error function by $E(w, \lambda)$, where w is the vector of input weights related to hidden units and λ is the vector of output weights. The problem of minimizing E with respect to (w, λ) can be a difficult nonlinear nonconvex optimization problem.

Typical difficulties can be [26]: 1) the presence of extensive flat regions and steep sided valleys in the error surface, which may cause a poor performance of some algorithms; 2) the presence of multiple local and global minima; and 3) the possible occurrence of phenomena of over-fitting when the number of free parameters is large in relation to the size of the training set. Moreover, for a very large data set, each function evaluation can be quite expensive from a computational point of view.

Although most of these difficulties can be faced to some extent through an appropriate definition of the error function and a convenient choice of the minimization algorithm, the training problem remains a difficult time-consuming task.

In recent years, considerable attention has been devoted to the learning technology known as extreme learning machine (ELM) (see [14]–[19], [27]). The essence of ELM, originally developed for SLFNs, is that the hidden layer of ELM is not tuned and the parameter vector w associated with hidden units is randomly chosen at the start. If the function E is defined as a sum-of-squares error function, the output weights can be evaluated by solving a linear least-squares problem, holding fixed the parameter vector w. In particular, the vector λ can be computed as the smallest norm least-squares solution, by estimating the Moore–Penrose generalized inverse of the coefficient matrix.

Manuscript received September 12, 2014; revised April 2, 2015 and August 27, 2015; accepted August 29, 2015. Date of publication September 22, 2015; date of current version October 17, 2016.

L. Grippo is with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale Antonio Ruberti, Sapienza Università di Roma, Rome 00185, Italy (e-mail: grippo@dis.uniroma1.it).

A. Manno and M. Sciandrone are with the Dipartimento di Ingegneria dell'Informazione, Università di Firenze, Florence 50121, Italy (e-mail: manno@dis.uniroma1.it; sciandro@dsi.unifi.it).

Digital Object Identifier 10.1109/TNNLS.2015.2475621

From a theoretical point of view, it has been established that this technique has both the interpolation capability and the universal approximation capability in a probabilistic sense [15]. Various extensions and modifications of the basic approach have been considered until recently (see [14], [18], [27]).

The computational experience reported in several papers indicates that ELM produces a better generalization performance at a much faster learning speed, in comparison with the traditional techniques, such as the backpropagation method [17].

If we compare the ELM technique with some efficient unconstrained optimization algorithms that operate on both w and λ (such as, for instance, the Limited Memory Broyden Fletcher Goldfarb Shanno (L-BFGS) method [21] or some nonmonotone spectral gradient methods [10], [25]), we may expect that in many large dimensional problems, the optimization approach can yield better generalization results with a smaller number of hidden nodes but with much larger computing times. In other cases, however, the ELM technique reaches even smaller training errors. This can be explained by considering that the local gradient-based algorithm can be attracted toward irrelevant stationary points in (w, λ).

In this paper, we introduce a new class of learning algorithms, based on decomposition techniques for unconstrained optimization, by extending in several directions, some decomposition algorithms introduced for learning in radial basis functions neural networks [7]. Our main objective is that of retaining both the advantages of the optimization approach and those of the ELM techniques. Indeed, the supervised learning problem with respect to all the network parameters (optimization approach) is decomposed into a sequence of simpler problems by partitioning the problem variables into different blocks considering the structure of the objective function. As already observed in connection with an ELM strategy, once the vector w is fixed, the resulting subproblem in the block component λ is a linear least-squares problem, which can be solved efficiently. Then, alternate partial minimizations with respect to the different blocks can be performed by suitable block-descent techniques. Thus, the adoption of the decomposition approach permits to exploit the structure of the objective function, obtaining advantages in terms of both computing times and error values.

From a theoretical standpoint, it can be shown that, under the assumptions stated on E, the block minimization algorithms defined in the sequel guarantee that every accumulation point is a stationary point of E. Moreover, at each major iteration, we can ensure that the new point generated by the algorithm is a global minimizer in λ, and this may help escaping from irrelevant local minimizers. This could not be true in the optimization approach without decomposition.

The computational experience reported in this paper indicates that computation can be organized in a way that a good generalization performance can be obtained, with much less computational effort than that required in the optimization approach without decomposition. At the same time, we can improve in many instances the ELM results, by reaching a better generalization performance with a smaller number of hidden neurons.

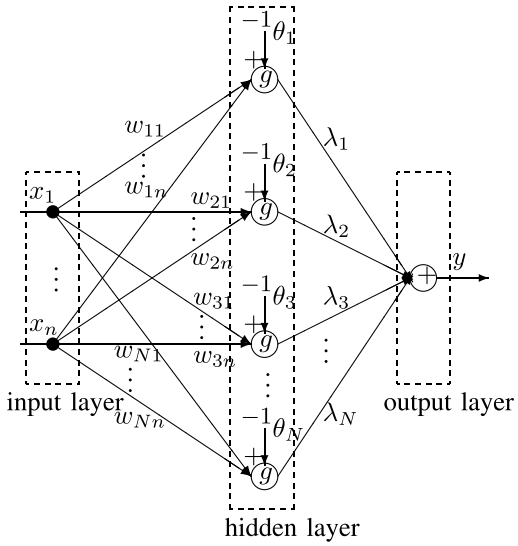

Fig. 1. Single-layer feedforward network.

This paper is organized as follows. In Section II, we introduce our notation and illustrate the main features of the error function employed in this paper. In Section III, we recall some basic results on the decomposition of optimization problems. In Section IV, an algorithm model based on block decomposition is formally stated, and the main convergence results are established. A specific learning algorithm, which can include ELM as a special case, is introduced and analyzed in Section V. In Section VI, we report the numerical results obtained with our algorithm in a set of test problems, and we show comparisons with the basic ELM technique and the optimization approach without decomposition. Finally, the conclusion is drawn in Section VII. In Appendixes A and B, we give the proofs of several results stated in Sections IV and V.

II. Notation and Preliminary Background

In the sequel, all vectors are intended as column vectors in real multidimensional linear spaces; however, we often use the notation (u, v) for denoting the ordered pair of two column vectors u, v. The norm $\|\cdot\|$ is intended as the Euclidean norm.

Given an MLP with one hidden layer, N hidden units and one linear output unit, the output $y \in R$ of the network can be expressed in the form

$$y = \sum_{j=1}^{N} \lambda_j g\left(w_j^T x - \theta_j\right) \tag{1}$$

where $w_j \in R^n$, $j = 1, \ldots, N$ are the *input weights*; $\theta_j \in R$, $j = 1, \ldots, N$ are the *thresholds* of the units; $\lambda_j \in R$, $j = 1, \ldots, N$ are the *output weights*; and $g : R \to R$, $g \in C^1$ is the *activation function*, as shown in Fig. 1.

For all $j = 1, \ldots, N$, we define $w_j = (w_{j1}\ w_{j2} \ldots w_{jn})^T$, where each component w_{ji} is the weight of the connection between input i and neuron j. Vector $w \in R^{Nn}$ is defined as $w = (w_1^T w_2^T \ldots w_N^T)^T$.

We have assumed that the output function is linear (as in ELM strategies), since, once fixed the vector w of input weights, the error function results to be a convex,

quadratic function of the output weights λ (see later), and this can be theoretically and computationally useful from an optimization point of view.

To simplify notation, we can include θ_j in the vector w_j by adding a fictitious component $x_0 = -1$ to the input vector and by redefining n, x, and w_j for all j. Then, we can construct the vector $w \in R^{Nn}$ with block components $w_j \in R^n$ and the vector λ with components $\lambda_j \in R$ for $j = 1, \ldots, N$.

We can also introduce a real function ϕ defined as

$$y = \phi(x; w, \lambda) = \sum_{j=1}^{N} \lambda_j g\left(w_j^T x\right). \tag{2}$$

The training set for the MLP is indicated by

$$TS = \{(x^p, y^p) : x^p \in R^n, y^p \in R, p = 1, \ldots, P\} \tag{3}$$

where (x^p, y^p) are the input–output data pairs. Then, the square error on a single sample can be expressed as

$$E_p(\lambda, w) = (\phi(x^p; \lambda, w) - y^p)^2. \tag{4}$$

Hence, the overall error function can be written into the form

$$E(\lambda, w) = \frac{1}{2} \sum_{p=1}^{P} E_p(\lambda, w) + \frac{\tau_\lambda}{2} \|\lambda\|^2 + \frac{\tau_w}{2} \|w\|^2 \tag{5}$$

where τ_λ and $\tau_w > 0$ are the regularization parameters. The last two terms of $E(w, \lambda)$ penalize large weights; they are introduced following a standard regularization technique (see [5]) to constrain the complexity of the network in order to prevent overfitting.

Let us introduce the $P \times N$ matrix $H(w)$ with elements $(H(w))_{p,j} = g(w_j^T x^p)$, and the $(P + N) \times N$ matrix $\tilde{H}(w)$

$$\tilde{H}(w) = \begin{bmatrix} H(w) \\ \sqrt{\tau_\lambda} I \end{bmatrix} \tag{6}$$

where I is the $N \times N$ identity matrix. We can rewrite the error function $E(\lambda, w)$ as a sum of a squared norm and a regularization term on the input weights

$$E(\lambda, w) = \frac{1}{2} \|\tilde{H}(w)\lambda - u\|^2 + \frac{1}{2} \tau_w \|w\|^2 \tag{7}$$

where $u = (y^1, \ldots, y^P, 0_N^T)^T$ and 0_N denotes the zero vector with N components. Thus, the training problem can be expressed as

$$\min_{\lambda \in R^N, w \in R^{Nn}} E(\lambda, w). \tag{8}$$

The partial gradients $\nabla_\lambda E(\lambda, w) \in R^N$, $\nabla_{w_j} E(\lambda, w) \in R^n$, and $\nabla_w E(\lambda, w) \in R^{Nn}$, of $E(\lambda, w)$ with respect to λ, w_j and w are the column vectors

$$\nabla_\lambda E(\lambda, w) = \tilde{H}(w)^T (\tilde{H}(w)\lambda - u) \tag{9}$$

$$\nabla_{w_j} E(\lambda, w) = \lambda_j \left[\sum_{p=1}^{P} (\phi(x^p; \lambda, w) - y^p) g'\left(w_j^T x^p\right) x^p \right] + \tau_w \; w_j \tag{10}$$

$$\nabla_w E(\lambda, w) = [\nabla_{w_1} E^T(\lambda, w) \quad \ldots \quad \nabla_{w_N} E^T(\lambda, w)]^T \tag{11}$$

where g' is the derivative of the function g.

It is important to notice that for a fixed value $\bar{w}$ of the input weights, the function $E(\lambda, \bar{w})$ with $\tau_\lambda > 0$ is a quadratic strictly convex function in λ. Furthermore, if $\tau_\lambda, \tau_w > 0$, the regularization terms guarantee that for a given $(\lambda^0, w^0) \in R^N \times R^{Nn}$, the level set

$$\mathcal{L}_0 = \{(\lambda, w) \in R^N \times R^{Nn} : E(\lambda, w) \leq E(\lambda^0, w^0)\} \tag{12}$$

is compact, so that the function $E(\lambda, w)$ admits a global minimum on $R^N \times R^{Nn}$.

In the context of decomposition methods, when we want to point out in the error function E, the blocks of input weights associated with each hidden unit, we use the notation $E(\lambda, w_1, \ldots, w_j, \ldots, w_N)$. When appropriate, a similar notation is also used for the arguments of the partial gradients $\nabla_\lambda E$ and $\nabla_{w_j} E$.

III. Decomposition Methods for Unconstrained Optimization

Let us consider the problem

$$\min_{z \in R^d} f(z) \tag{13}$$

where $f : R^d \to R$ is a continuously differentiable function.

In some cases, in dependence on the structure of the objective function and on the problem dimension, it could be convenient to partition the vector of variables as

$$z = \left(z_1^T, \ldots, z_s^T, \ldots, z_q^T\right)^T$$

where $z_s \in \mathbb{R}^{d_s}$ and $s \in \{1, \ldots, q\}$ is the sth block component, and to operate (sequentially or in parallel) on each block component.

Setting $z_1 = w$ and $z_2 = \lambda$, the MLP training problem described in Section II can be rewritten in the form

$$\min_{z_1, z_2} f(z_1, z_2) = \frac{1}{2} \|\tilde{H}(z_1) z_2 - u\|^2 + \frac{\tau_1}{2} \|z_1\|^2. \tag{14}$$

We may observe that, once z_1 is fixed, the resulting subproblem in the block component z_2 is a linear least-squares problem, which can be solved efficiently.

The basic ELM algorithms randomly select the block component z_1 and compute the subvector z_2 as the solution of the linear least-squares problem, without performing minimizations with respect to z_1.

In this paper, we focus on the class of descent methods that make use of an alternating minimization approach with respect to the block components. In this context, the block version of the Gauss–Seidel algorithm is one of the most popular methods. It can be described as a sequence of main steps, such that each of these steps consists of q iterations that update, in sequence, a single component z_s by minimizing f with respect to z_s, holding the remaining components fixed. The same set of iterations is then repeated in the subsequent main steps.

It is known that, in general, the Gauss–Seidel method may not converge, in the sense that it may produce a sequence with limit points that are not stationary points of the objective function. Convergence results for the Gauss–Seidel method have been given under suitable generalized convexity assumptions on the objective function (see [4], [11], [22]). In the particular

case of two blocks, i.e., $q = 2$, convergence results have been established without convexity assumptions on f [11]. Note that the computation of the exact solutions of the subproblems can be quite expensive whenever the subproblems do not have a favorable structure. This has motivated studies on inexact decomposition methods [6], [8], [11], [23].

In a Gauss–Seidel scheme, the partition into block components of z is prefixed and does not vary during the iterations. In order to overcome this limitation, we can consider a more general decomposition framework where, at the generic iteration k, the components of the current point z^k are partitioned into two subsets. The first, usually called working set, is defined through the index set W^k. The second set is defined through the complement set $\overline{W}^k = \{1, \ldots, q\}/W^k$.

The variables corresponding to $\overline{W}^k$ are unchanged at the new iteration, while the components corresponding to W^k are determined by minimizing (possibly inexactly) the objective function. After a suitable reordering, this subproblem can put into the form

$$\min_{z_{W^k}} \; f\left(z_{W^k}, z^k_{\overline{W}^k}\right). \tag{15}$$

Then, if the solution of problem (15) is denoted by $z^{k+1}_{W^k}$, the new point z^{k+1} can be defined by letting

$$z^{k+1}_{\overline{W}^k} = z^k_{\overline{W}^k}. \tag{16}$$

Decomposition methods may differ in the following:

1) the choice of the working set W^k at each k;
2) the kind of solution (exact or approximate) of (15);
3) the minimization algorithm employed.

In order to guarantee the global convergence of a decomposition method, both the working set selection and the solution algorithms for the subproblems cannot be arbitrary but must satisfy suitable conditions.

For instance, a well-known convergent decomposition algorithm is based on the Gauss–Southwell rule for selection of the working set [9], [22]. According to the Gauss-Southwell rule, the working set W^k must include the index i^k corresponding to the block variable that mostly violates the first-order optimality condition, that is, the index i^k such that $|\nabla_{i^k} f(z^k)| \geq |\nabla_j f(z^k)|$ for $j = 1, \ldots, d$.

In the case of approximate Gauss–Seidel algorithms, based on a sequence of inexact minimizations with respect to the different blocks, it is required that each index j is selected sufficiently often in W^k, that the distance $\|z^{k+1} - z^k\|$ goes to zero for $k \to \infty$, and that globally convergent minimization algorithms are employed for solving the subproblems.

In Section IV, we present a decomposition framework for MLP training where the variables are decomposed into output (λ) and input (w) weights in order to exploit the structure of the objective function. Furthermore, both λ and w may be (possibly) partitioned, and the partitioning may vary with the iterations.

IV. Algorithm Model Based on Decomposition

In this section, we define a class of block decomposition algorithms for the minimization of the error function E, and we establish some basic convergence results.

Algorithm 1 Decomposition Algorithm Model (DAM)

Initialization : determine an initial starting point

$$w^0 = ({w_1^0}^T, \ldots, {w_N^0}^T)^T \in R^{nN}$$
$$\lambda^0 = (\lambda_1, \ldots, \lambda_N)^T \in R^N.$$

Set $k = 0$.

While the *termination test* is not satisfied, choose

$$J^k, L^k \subseteq \{1, 2, \ldots, N\}$$

and perform the following steps.

Step 1 *(Minimization with respect to output weights)* If $J^k = \emptyset$ set $\lambda^{k+1} = \lambda^k$ and terminate Step 1.
Otherwise, starting from (λ^k, w^k), minimize (exactly or approximately) the function E with respect to all λ_j, for $j \in J^k$ and determine the new updated output weights λ_j^{k+1}, $j \in J^k$, Then define λ^{k+1} by letting $\lambda_j^{k+1} = \lambda_j^k$, $j \notin J^k$.

Step 2 *(Minimization with respect to input weights)*
If $L^k = \emptyset$ set $w^{k+1} = w^k$ and terminate Step 2.
Otherwise, starting from (λ^{k+1}, w^k) minimize approximately E with respect to all w_j, $j \in L^k$ and determine the updated vectors w_j^{k+1}, $j \in L^k$.
Define w^{k+1} by letting $w_j^{k+1} = w_j^k$, $j \notin L^k$.

Set $k = k + 1$.

End While

An algorithm in this class can be described, in general, as a sequence of main iterations, indexed by k, such that at each iteration, we update only a specific set of parameters of the network, through a few elementary steps.

In particular, we suppose that, at each k, two index sets J^k and L^k are chosen, which define two sets of hidden units. We admit that, possibly, we have $J^k = L^k$ and that one of these index sets can be empty. Then, we update only the output weights $\lambda_j \in R$, with $j \in J^k$ and the input weights $w_j \in R^n$, with $j \in L^k$.

More specifically, we can define the following algorithm model, where we have introduced a termination test that can impose, for instance, a maximum number of iterations or a bound on a norm of the gradient ∇E.

Algorithm 1 defined above, referred to as DAM, may include different learning algorithms, which can be defined by specifying: the termination test, the definition of the index sets J^k and L^k at each k, and the algorithms used at Step 1 and Step 2.

In any case, however, we require that a decomposition of the problem variables into output and input weights is retained, and this preserves the possibility of obtaining global minimizers with respect to the output weights in J^k.

In fact, we can observe that the problem at Step 1 is a strictly convex quadratic problem in the variables λ_j, for $j \in J^k$, which can be solved either exactly, through a direct method for least-squares problems, or, approximately, by means of an iterative method. We can make use, for instance, of the conjugate gradient method, and we can require that the

problem is solved with greater precision for increasing values of k.

At Step 2, we consider a nonconvex problem, and therefore, only an approximate minimization of the objective function can be performed.

We can define various specific choices for the index sets J^k, L^k and for the termination criterion, which correspond to different training techniques.

1) A first possibility is that of choosing, for all k, the index sets: $J^k = \{1, 2, \ldots, N\}$, $L^k = \emptyset$. In this case, only the output weights are updated, and the input weights remain fixed at some initial random values. If only one iteration is admitted and we set $\tau_w = 0, \tau_\lambda = 0$, the algorithm can be identified with the basic version of the ELM.
2) When N and P are very large, then also the problem of computing the output weights could be decomposed into different blocks, by letting again $L^k = \emptyset$ for all k, but defining appropriately J^k for each k and admitting a sufficiently large number of main iterations. This can be viewed as an iterative version of an ELM technique, based on a decomposition approach.
3) A decomposition technique in which all weights are updated in each major step can be defined by choosing the index sets $J^k = L^k = \{1, 2, \ldots, N\}$. If Step 2 is implemented according to this strategy, for each k, then the training problem is decomposed in two blocks (input weights and output weights).
4) We can define a decomposition scheme where, for a set of $N + 1$ consecutive iterations, first we minimize with respect λ, and then we perform N approximate minimizations with respect to each w_j, for $j = 1, \ldots, N$. When w_N has been updated, a new cycle of iterations is started with the same block variables.
5) An extension of Algorithm 1, referred to as Algorithm DEC, will be analyzed in detail in Section V. As in case 4) above, it consists in performing a first minimization with respect to λ and then N approximate minimizations with respect to each w_j, for $j = 1, \ldots, N$. However, now each component λ_j is analytically updated, each time that w_j has been updated, by solving a 1-D optimization problem in λ_j. As in case 4), at the end of these steps, we can perform a new sequence. We can also, in principle, omit the step corresponding to the minimization with respect to λ and then perform a sequence of alternate (approximate) minimizations in λ_j and w_j.

It will be shown that Algorithm DEC can be recast into the model DAM, through an appropriate definition of the index sets.

Various other schemes can be defined in the same framework introduced with the algorithm model DAM. More general structures than DAM could also be considered if we admit block minimizations with respect to pairs of input and output weights. However, these possibilities will not be discussed here.

If DAM generates an infinite sequence, in order to establish formally convergence results toward stationary points of E, but also for realizing a practically significant learning scheme, we must impose suitable conditions on the choice of the index sets J^k and L^k and on the minimizations performed at each step.

The first condition stated below requires that each block component of λ and w is periodically considered within a prefixed maximum number of iterations.

Condition 1: Suppose that DAM generates an infinite sequence, then there exists an integer $M > 0$ such that for every $j \in \{1, 2, \ldots, N\}$ and $k \geq 0$, and there exist iteration indices h and ℓ, with $k \leq h \leq k + M$ and $k \leq \ell \leq k + M$ such that $j \in J^h$ and $j \in L^\ell$. ■

As regards Step 1, we require that, when J^k is nonempty and $j \in J^k$ a convergent minimization algorithm is employed.

Condition 2: For all k, we have

$$E(\lambda^{k+1}, w^k) \leq E(\lambda^k, w^k). \tag{17}$$

Moreover, for every infinite subsequence such that $j \in J^k$ for all $k \in K$, we have

$$\lim_{k \in K, k \to \infty} \nabla_{\lambda_j} E(\lambda^{k+1}, w^k) = 0. \tag{18}$$

■

At Step 2, we have a nonconvex problem, and hence, as already discussed in Section III, more restrictive conditions must be imposed. In particular, we must guarantee that the distance between successive iterates tends to zero, and that, at least in the limit, the optimality conditions are satisfied.

Condition 3: We have, for all k

$$E(\lambda^{k+1}, w^{k+1}) \leq E(\lambda^{k+1}, w^k). \tag{19}$$

Moreover, for every infinite subsequence such that $j \in L^k$ for all $k \in K$, we have

$$\lim_{k \in K, k \to \infty} \left\| w_j^{k+1} - w_j^k \right\| = 0 \tag{20}$$

$$\lim_{k \in K, k \to \infty} \nabla_{w_j} E(\lambda^{k+1}, w^k) = 0. \tag{21}$$

■

The convergence of DAM is established in Proposition 1, which is proved in Appendix A.

Proposition 1: Suppose that DAM generates an infinite sequence $\{(\lambda^k, w^k)\}$ and that Conditions 1–3 are satisfied. Then, the following holds.

1) The sequence $\{(\lambda^k, w^k)\}$ has limit points.
2) The sequence $\{E(\lambda^k, w^k)\}$ converges to a limit.
3) $\lim_{k \to \infty} \|\lambda^{k+1} - \lambda^k\| = 0$.
4) $\lim_{k \to \infty} \|w^{k+1} - w^k\| = 0$.
5) Every limit point of $\{(\lambda^k, w^k)\}$ is a stationary point of E. ■

V. Decomposition Algorithm

In this section, we define more precisely a block decomposition algorithm, already introduced in the Section IV as Algorithm DEC, and we establish convergence results on the basis of Proposition 1 of Section IV.

In the algorithm, which corresponds to example 5) in Section IV, starting from an initial random choice of starting point, we perform a sequence of major iterations, by alternating a minimization with respect to $\lambda \in R^N$ with a set of $2N$

Algorithm 2 Armijo's Method

Data: $a > 0$, $\gamma \in (0, 1)$, $\delta \in (0, 1)$.
Set $\alpha = a$.
While

$$E(\lambda^{k+1}, w_1^k, \ldots, w_j^k + \alpha d_j^k, \ldots, w_N^k) > E(\lambda^{k+1}, w^k) - \gamma\,\alpha \|\nabla_{w_j} E(\lambda^{k+1}, w^k)\|^2$$

set $\alpha = \delta\alpha$
End While
Set $\alpha_j^k = \alpha$ and terminate.

■

iterations that update in sequence each w_j and then again the scalar output weight λ_j for $j = 1, \ldots, N$.

At the current point (λ^k, w^k), the minimization with respect to $\lambda \in R^N$ can be performed exactly either by means of a direct method for least-squares problems or by employing an iterative method for minimizing a convex quadratic function, such as the conjugate gradient method, with a suitable termination criterion. At the end of this phase, we obtain the pair (λ^{k+1}, w^k).

The approximate minimization with respect to a block $w_j \in R^n$ can be performed by running an unconstrained minimization algorithm, for a finite number of inner iterations, holding fixed the remaining variables. This determines an updated point w_j^{k+1}. In particular, we can employ an efficient gradient-based method, such as, for instance, a reduced memory quasi-Newton method [21], or a nonmonotone spectral gradient method [10], [25] or a nonlinear conjugate gradient method [13].

The minimization with respect to λ_j can be performed analytically by solving a 1-D quadratic optimization problem in λ_j.

In the minimization with respect to $w_j \in R^n$, as imposed by Condition 3, suitable safeguards must be introduced for preventing an oscillatory behavior and for establishing convergence results. The simplest choice that satisfies the requirements of Condition 3 can be that of performing for each j a step along the (partial) steepest descent direction

$$d_j^k = -\nabla_{w_j} E(\lambda^{k+1}, w^k)$$

by employing a line search technique with appropriate acceptability conditions for the stepsize.

In particular, we can make use of Algorithm 2 and based on the well-known Armijo's method, where we compute the stepsize α_j^k along d_j^k, starting from an initial constant estimate $a > 0$.

The convergence properties of Armijo's method in our formulation are reported in Proposition 2, which is proved in Appendix B.

Proposition 2: Let j be a fixed index in $\{1 \ldots N\}$ and let $\{(\lambda^{k+1}, w^k)\}_K$ be an infinite subsequence such that

$$\nabla_{w_j} E(\lambda^{k+1}, w^k) \neq 0$$

and assume that, for all $k \in K$, the steplength α_j^k is computed along d_j^k by means of Armijo's method.

Then, the algorithm terminates yielding a stepsize $\alpha_j^k > 0$ such that the following holds.

1) For all $k \in K$, we have

$$E\big(\lambda^{k+1}, w_1^k, \ldots, w_j^k + \alpha_j^k d_j^k, \ldots, w_N^k\big) < E(\lambda^{k+1}, w^k). \tag{22}$$

2) If the following limit is valid:

$$\lim_{k \in K, k \to \infty} E(\lambda^{k+1}, w^k) - E\big(\lambda^{k+1}, w_1^k, \ldots, w_j^k + \alpha d_j^k, \ldots, w_N^k\big) = 0 \tag{23}$$

then

a) $\lim_{k \in K, k \to \infty} \alpha_j^k \|d_j^k\| = 0.$

b) If $\{(\lambda^{k+1}, w^k)\}_{K_1}, K_1 \subseteq K$ is a convergent subsequence, we have

$$\lim_{k \in K_1, k \to \infty} \nabla_{w_j} E(\lambda^{k+1}, w^k) = 0. \tag{24}$$

■

By Proposition 2, if we set at each step of a subsequence

$$w_j^{k+1} = w_j^k + \alpha_j^k d_j^k$$

then, as the stepsize is bounded above, the convergence of the function values also implies that the condition $\|w_j^{k+1} - w_j^k\| \to 0$ is satisfied in the subsequence. The point $w_j^k + \alpha_j^k d_j^k$ and the corresponding function value

$$E\big(\lambda^{k+1}, w_1^k, \ldots, w_j^k + \alpha_j^k d_j^k, \ldots, w_N^k\big)$$

computed through Armijo's method also yield a reference point and a reference value for the objective function, if we use a different minimization technique for computing w_j^{k+1}. More specifically, using any technique, we can compute a tentative point $\hat{w}_j^k$ such that

$$E\big(\lambda^{k+1}, w_1^k, \ldots, \hat{w}_j^k \ldots w_N^k\big) \leq E\big(\lambda^{k+1}, w_1^k \ldots w_j^k + \alpha_j^k d_j^k, \ldots, w_N^k\big). \tag{25}$$

Then, we can check whether, for a $\tau > 0$, we have

$$E\big(\lambda^{k+1}, w_1^k, \ldots \hat{w}_j^k, \ldots, w_N^k\big) \leq E(\lambda^{k+1}, w^k) - \tau \big\|\hat{w}_j^k - w_j^k\big\|^2 \tag{26}$$

and we can set $w_j^{k+1} = \hat{w}_j^k$, if this condition is satisfied.

Algorithm 3 (DEC) that we will define can be described as a sequence of two major steps, which perform a sequence of $2N + 1$ block minimizations with respect to different blocks of variables. In order to describe a major step, we introduce an index h such that $h = 0, 1, \ldots, N$, and we distinguish two possible cases: 1) Step A, where only output weights are modified and 2) Step B, where only input weights are modified.

In Algorithm 3, we can also introduce suitable tests for deciding whether it could be convenient to update each block of weights and whether we can terminate the inner steps and the main iterations. In particular, we indicate by maxiter, with $1 \leq \text{maxiter} \leq \infty$, the maximum number of times Step A has to be repeated. If maxiter $< \infty$, this also implies a finite termination condition for the whole algorithm.

Algorithm 3 Decomposition Algorithm (DEC)

Data. $w^0 = ({w_1^0}^T, \dots {w_N^0}^T)^T \in R^{nN}$, $\lambda^0 \in R^N$, $1 > \xi \geq 0, \theta \in (0,1)$, $\tau > 0$. $maxiter \geq 1$.
Set $k = 0$, $iter = 1$ and $h = 0$.
While $\nabla E(\lambda^k, w^k) \neq 0$ do
Step A *(Only output weights are modified)*
If $h = 0$ then
compute λ^{k+1} such that

$$E(\lambda^{k+1}, w^k) \leq E(\lambda^k, w^k)$$
$$\|\nabla_\lambda E(\lambda^{k+1}, w^k)\| \leq (\xi)^k.$$

set $w^{k+1} = w^k$, $k = k+1$,
Else
set $j = h$, $\lambda_s^{k+1} = \lambda_s^k$ for all $s \neq j$ and then compute λ_j^{k+1} by solving a 1-D problem in λ_j,
in a way that

$$E(\lambda^{k+1}, w^k) \leq E(\lambda^k, w^k)$$
$$\nabla_{\lambda_j} E(\lambda^{k+1}, w^k) = 0.$$

Endif
If $iter \geq maxiter$ terminate.
Step B *(Only input weights are modified)*
If $h = 0$ set $\lambda^{k+1} = \lambda^k$
Set $h = h+1$.
If $h < N+1$ then:
set $j = h$
if $\|\nabla_{w_j} E(\lambda^{k+1}, w^k)\| \leq (\theta)^k$
set $w^{k+1} = w^k$ and $k = k+1$
else:
(i) set $d_j^k = -\nabla_{w_j} E(\lambda^{k+1}, w^k)$
(ii) compute α_j^k along d_j^k, by employing Armijo's method
(iii) using any technique, compute a point $\hat{w}_j$ such that (25) is satisfied.
(iv) if (26) holds set $w_j^{k+1} = \hat{w}_j$; otherwise set $w_j^{k+1} = w_j^k + \alpha_j^k d_j^k$;
(v) set $w_s^{k+1} = w_s^k$, for all $s \neq j$
set $k = k+1$.
endif
Else
set $h = 0$
Endif
iter=iter+1
End While ■

If we set maxiter = 1, then only Step A is performed, and hence, if the minimization with respect to λ is performed exactly, the algorithm corresponds, in essence, to an ELM.

It can be easily verified that Algorithm DEC can be recast into the structure DAM, for an appropriate definition of the indices J^k and L^k. In fact, at each given iteration index k, we can distinguish the cases $h = 0$ and $h \in \{1, \dots, N\}$.

If $h = 0$, then only Step A is performed, and Step 1 of DAM corresponds to $J^k = \{1, \dots, N\}$, while Step 2 of DAM corresponds to $L^k = \emptyset$.

If $h > 0$, then both Step A and Step B are performed, we have that Step 1 of DAM corresponds to $J^k = \{h\}$ and Step 2 of DAM corresponds to the choice $L^k = \{h\}$.

An important point is that, during Step B of the kth iteration, computations can be efficiently organized in order to reduce the cost of evaluating the objective function E, each time that a block component w_j is changed. In order to explain this point, let us denote by $\phi(x^p; \lambda^{k+1}, w^k)$, the output corresponding to the input x^p at the current point (λ^{k+1}, w^k). Now, in the line search performed at Step B, if E must be evaluated in correspondence to a new tentative value w_j, we must recompute the network output in correspondence to w_j. However, we can write

$$\phi\left(x^p; \lambda^{k+1}, w_1^k, \dots, w_j, \dots, w_N^k\right) = \phi(x^p; \lambda^{k+1}, w^k) - \lambda_j^{k+1} g\left(\left(w_j^k\right)^T x^p\right) + \lambda_j^{k+1} g((w_j)^T x^p). \quad (27)$$

Therefore, if we store the current output at each k for every x^p, the new output can be obtained, for each j, with few elementary operations.

Similarly, we can recompute efficiently the network output each time that the weight λ_j is updated at Step A for $h > 0$. In fact, in this case, we can write

$$\phi(x^p; \lambda^{k+1}, w^k) = \phi(x^p; \lambda^k, w^k) + \left(\lambda_j^{k+1} - \lambda_j^k\right) g\left(\left(w_j^k\right)^T x^p\right). \quad (28)$$

We note also that the global minimizer, say λ_j^{k+1}, of the 1-D subproblem considered at Step A, can be analytically obtained by imposing that λ_j^{k+1} is the solution of a linear equation in λ_j, such that $\nabla E_{\lambda_j}(\lambda^{k+1}, w^k) = 0$.

The convergence of Algorithm DEC (with $maxiter = \infty$) is established in Proposition 3. We show that Conditions 1–3 are satisfied, so that the assertion follows from Proposition 1.

Proposition 3: Suppose that Algorithm DEC generates an infinite sequence $\{(\lambda^k, w^k)\}$. Then, the following holds.

1) $\{(\lambda^k, w^k)\}$ has limit points.
2) The sequence $\{E(\lambda^k, w^k)\}$ converges to a limit.
3) We have $\lim_{k\to\infty} \|\lambda^{k+1} - \lambda^k\| = 0$.
4) For $j = 1, \dots, N$, we have

$$\lim_{k\to\infty} \left\|w_j^{k+1} - w_j^k\right\| = 0.$$

5) Every limit point of $\{(\lambda^k, w^k)\}$ is a stationary point of E.

Proof: We show that Conditions 1–3 of Section IV hold, so that, the assertions follow from Proposition 1.

If the algorithm does not terminate, then it produces infinite sets of iterations for $h = 0, 1, \dots N$. During each of the sets, all block variables are updated in sequence, and hence, Condition 1 is obviously satisfied with $M = 2N+1$.

Consider now Condition 2. Given (λ^k, w^k), in correspondence to the given k, we may have either that $h = 0$ or that

$h \in \{1, \ldots, N\}$. In both cases, inequality (17) is satisfied. If $h = 0$ for some infinite subsequence, Condition 2 follows immediately from the instructions of Step A. In fact, as $\xi \in (0, 1)$, the termination condition at Step A implies that the limit (18) of Condition 2 holds.

If $h \in \{1, \ldots, N\}$, then $J^k = \{h\}$, $j = h$, and the instructions at Step A imply that $\nabla_{\lambda_j} E(\lambda^{k+1}, w^k) = 0$, and hence, if this equality holds for an infinite subsequence, say $k \in K$, Condition 2 must be satisfied.

As regards Condition 3, in correspondence to a given k, we can distinguish again the case $h = 0$ and the case $h \in \{1, \ldots, N\}$. In both cases, we have that (19) holds, and we must prove that conditions (20) and (21) are satisfied.

If $h = 0$, the instructions of the algorithm imply that $L^k = \emptyset$, and hence, for establishing conditions (20) and (21), we can assume that $h > 0$ and that Step B is performed.

The instructions of Step B imply that $\lambda^{k+1} = \lambda^k$; moreover, for every $j \in L^k$, we have either that $w_j^{k+1} = w_j^k$ or that (26) holds, or else that $w_j^{k+1} = w_j^k + \alpha_j^k d_j^k$, where $d_j^k = -\nabla_{w_j} E(\lambda^{k+1}, w^k)$. In the latter case, as $\alpha_j^k \leq a$, recalling the acceptance condition of Armijo's method, we can write

$$E(\lambda^{k+1}, w^{k+1}) \leq E(\lambda^{k+1}, w^k) - \frac{\gamma}{a} \left\| w_j^{k+1} - w_j^k \right\|^2. \quad (29)$$

Then, in any case, for every $j \in L^k$, we have that

$$E(\lambda^{k+1}, w^{k+1}) \leq E(\lambda^{k+1}, w^k) - \hat{\tau} \left\| w_j^{k+1} - w_j^k \right\|^2 \quad (30)$$

with $\hat{\tau} = \min\{\tau, \frac{\gamma}{a}\}$. Since we have

$$E(\lambda^{k+1}, w^{k+1}) \leq E(\lambda^{k+1}, w^k) \leq E(\lambda^k, w^k) \quad (31)$$

and $\mathcal{L}^0$ is compact, the sequences of function values $\{E(\lambda^k, w^k)\}$ and $\{E(\lambda^{k+1}, w^k)\}$ converge to the same limit. Suppose now that for an infinite subsequence, say for $k \in K$, we have that $j \in L^k$. Then, considering the instructions of Step B and the preceding discussion, for $k \in K$, we have either that $w_j^{k+1} = w_j^k$ or that (30) holds. This implies that

$$\lim_{k \in K, k \to \infty} \left\| w_j^{k+1} - w_j^k \right\| = 0 \quad (32)$$

which establishes (20).

Finally, in order to complete the proof that Condition 3 holds, assume, by contradiction, that there exists an infinite subset $K_1 \subseteq K$ of the subsequence considered, such that for an index $j \in L^k$, we have

$$\|\nabla_{w_j} E(\lambda^{k+1}, w^k)\| \geq \eta > 0 \quad \forall k \in K_1. \quad (33)$$

By (31) and the compactness of $\mathcal{L}^0$, there exists a further subsequence $\{(\lambda^{k+1}, w^k)\}_{K_2}$, with $K_2 \subseteq K_1$ converging to a point (λ, w), so that from (33), we get

$$\|\nabla_{w_j} E(\lambda, w)\| \geq \eta. \quad (34)$$

Because of the instructions at Step 2, (33) implies that, for all sufficiently large $k \in K_2$, Armijo's method is employed, and a stepsize α_j^k is computed along d_j^k. Then, for all large $k \in K_2$, we have

$$\begin{aligned} E(\lambda^{k+1}, w^{k+1}) &\leq E\big(\lambda^{k+1}, w_1^k, \ldots, w_j^k + \alpha_j^k d_j^k, \ldots, w_N^k\big) \\ &\leq E(\lambda^{k+1}, w^k) \leq E(\lambda^k w^k). \end{aligned} \quad (35)$$

TABLE I
DATASET INFORMATION

name	#features	#training data	#test data
cod-rna [2] (C)	8	59535	271617
magic [20](C)	10	17118	1902
ringnorm [20](C)	20	6500	900
satellite [19](MC)	36	4435	2000
diabetes [19](C)	8	576	192
mv [20](R)	10	32615	8153
california [20] (R)	8	16512	4128
census [20](R)	16	18228	4556
deltaelv [20](R)	6	7614	1903
elevators [20](R)	18	13280	3319
ailerons [20](R)	40	11000	2750

Moreover, as the sequence of function values is convergent, from (35), we also get the limit

$$\begin{aligned} \lim_{k \in K_2, k \to \infty} \big[& E\big(\lambda^{k+1}, w_1^k, \ldots, w_j^k \ldots, w_N^k\big) \\ & - E\big(\lambda^{k+1}, w_1^k, \ldots, w_j^k + \alpha_j^k d_j^k, \ldots, w_N^k\big)\big] = 0. \end{aligned} \quad (36)$$

Then, recalling Proposition 2, we have

$$\lim_{k \in K_2, k \to \infty} \nabla_{w_j} E(\lambda^{k+1}, w^k) = 0 \quad (37)$$

and hence, by continuity of the gradient, we obtain $\nabla_{w_j} E(\lambda, w) = 0$, which contradicts (34). This concludes the proof. ■

VI. COMPUTATIONAL EXPERIMENTS

A. Test Problems

We have considered 11 freely available data sets taken from the literature. Four of these data sets concern binary classification problems, one is a multiclass problem, and the remaining six data sets refer to regression problems (for the regression problems, we linearly scaled the output labels to $[0, 1]$). Information about the problem sizes is reported in Table I, where (C) indicates a classification problem, (MC) denotes a multiclass problem, and (R) denotes a regression problem.

B. Implementation Details

We have chosen as activation function g of the hidden units the sigmoid function, that is, we set $g(t) = 1/1 + e^{-t}$. The regularization parameters τ_λ and τ_w in the objective function could be selected by a standard cross-validation technique. However, for our experiments, we have set, for all problems, $\tau_\lambda = 10^{-3}$ and $\tau_w = 10^{-2}/(n \times N)$, which appeared reasonable values, considering that $\lambda \in R^N$ and $w \in R^{nN}$, with N in the range $[20, 500]$. The components of w^0 are randomly chosen in the interval $[-0.5, 0.5]$.

We have implemented Algorithm DEC in Fortran90. All the tests of Algorithm DEC have been performed on an Intel Core2 duo 3.16×2 GHz CPU with 4-GB RAM.

Step A: The vector λ^{k+1} is computed by minimizing the function $E(., w^k)$ with respect to $\lambda \in R^N$, using the conjugate gradient method and satisfies the stopping criterion

$$\|\tilde{H}(w^k)\lambda^{k+1} - u\| / \|u\| \leq 10^{-6}. \quad (38)$$

TABLE II
COD-RNA

algorithm	neurons	tr_time	ts_time	%tr	%ts
DEC(2)	20	3.22	0.40	92.04	93.55
	50	8.83	1.01	92.86	94.57
	100	21.03	2.02	93.42	94.93
DEC(10)	20	25.98	0.40	94.36	95.95
lbfgs(500)	20	84.15	0.38	91.03	91.61
lbfgs(1000)	20	172.55	0.38	91.12	91.91
elm	100	1.98	2.04	84.81	86.06
	200	4.55	4.06	87.24	88.62
	400	17.56	8.14	87.89	89.16

TABLE III
MAGIC

algorithm	neurons	tr_time	ts_time	%tr	%ts
DEC(2)	20	1.65	0.003	81.28	81.80
	50	4.85	0.007	82.93	82.88
	100	10.64	0.015	84.26	83.30
DEC(10)	20	10.81	0.003	86.71	87.53
lbfgs(500)	20	29.32	0.004	81.33	82.11
lbfgs(1000)	20	57.29	0.004	82.14	83.07
elm	100	0.50	0.015	74.59	74.65
	500	5.40	0.075	78.56	77.77
	1500	57.06	0.231	81.75	78.88

TABLE IV
RINGNORM

algorithm	neurons	tr_time	ts_time	%tr	%ts
DEC(2)	100	1.27	0.008	98.40	94.27
	200	3.67	0.01	99.41	94.92
	500	15.82	0.04	99.94	95.13
DEC(10)	100	10.13	0.008	99.63	96.00
lbfgs(500)	100	85.04	0.008	99.45	96.79
lbfgs(1000)	100	164.79	0.008	99.72	96.88
elm	500	3.01	0.04	91.88	88.40
	1000	15.99	0.09	95.94	91.23
	2000	105.39	0.18	98.98	91.97

TABLE V
SATELLITE

algorithm	neurons	tr_time	ts_time	%tr	%ts
DEC(2)	100	4.17	0.02	83.97	79.77
	200	20.88	0.05	86.00	80.91
	500	119.78	0.13	90.04	82.31
DEC(10)	100	45.94	0.02	92.76	85.00
lbfgs(500)	100	81.46	0.02	75.85	73.59
lbfgs(1000)	100	221.14	0.02	82.43	77.45
elm	500	1.72	0.13	79.40	76.40
	1000	7.56	0.27	83.71	78.35
	2000	170.74	0.54	87.25	78.19

TABLE VI
DIABETES

algorithm	neurons	tr_time	ts_time	%tr	%ts
DEC(2)	20	0.014	0.0004	76.97	77.44
	50	0.037	0.0007	78.43	78.90
	100	0.078	0.001	78.73	79.06
DEC(10)	20	1.56	0.0004	79.34	81.25
lbfgs (500)	20	0.90	0.0003	80.93	76.71
lbfgs (1000)	20	1.77	0.0003	84.20	75.00
elm	20	0.004	0.0004	77.96	77.34
	100	0.020	0.001	82.22	76.45
	200	0.084	0.002	88.22	74.58

When $h \neq 0$, the scalar λ_j^{k+1} is analytically computed by solving the 1-D quadratic problem.

Step B: The point $\hat{w}_j$ is computed by employing the L-BFGS method [21] whose Fortran code is available at http://www.ece.northwestern.edu/~nocedal/lbfgs.html. The maximum number of inner iterations of L-BFGS method has been set equal to 100, and the parameter that determines the accuracy with which the solution is to be found has been set equal 10^{-2}.

C. Comparisons With L-BFGS and ELM

Two versions of Algorithm DEC have been defined and evaluated. The first one, indicated as DEC(2), is defined as setting maxiter $= 2$, and the second one, indicated as DEC(10), is defined as setting maxiter $= 10$.

A first set of experiments has been performed in order to assess the effectiveness of the proposed decomposition-based strategy in comparison with a standard approach employing an efficient algorithm for unconstrained optimization. To this aim, we have compared the performance of Algorithms DEC(2) and DEC(10) with those of L-BFGS applied to the unconstrained minimization problem $\min_{z\in R^{nN+N}} E(z)$, where $z = (\lambda^T \quad w^T)^T$. After some preliminary experiments, we considered two different values for the maximum number of iterations of the L-BFGS method by setting maxiter $= 500$ and maxiter $= 1000$, and the termination condition was defined in a way that termination occurs only in correspondence to the maximum number of iterations.

In order to evaluate the influence of considering as free parameters to be optimized both the output weights λ and the input weights w, we have also performed comparisons with the basic ELM strategy, where only the output weights λ are considered in the minimization of the training error. We have implemented the simplest ELM algorithm: the components of w are randomly chosen in the interval $[-0.5, 0.5]$, and λ is computed by solving the linear least-squares problem $\min_{\lambda\in R^N} E(\lambda, w^0)$ (considering again $\tau_w = 0, \tau_\lambda = 0$), using the generalized inverse routine DGELSS of the Lapack library.

The results of this comparison are shown in Tables II–VI for the classification problems and in Tables VII–XII for the regression problems. For each test, the results concern the average over ten runs corresponding to ten different starting points.

In Tables II–VI, we report the number N of hidden neurons, the train time (tr_time), the test time (ts_time), the classification accuracy on the training set (%tr), and the classification accuracy on the test set (%ts). The train and test times are reported in seconds. The classification accuracies on the two sets are defined as follows:

$$\%\text{tr} = \frac{N_c^{\text{tr}}}{N^{\text{tr}}} \times 100 \quad \%\text{ts} = \frac{N_c^{\text{ts}}}{N^{\text{ts}}} \times 100$$

where N^{tr} is the number of training data, N_c^{tr} is the number of right classified samples in the training data, N^{ts} is the number of test data, and N_c^{ts} is the number of right classified samples in the test data.

In Tables VII–XII, we report the number N of hidden neurons, the train time (tr_time), the test time (ts_time),

TABLE VII
MV

algorithm	neurons	tr_time	ts_time	RMSEtr	RMSEts
DEC(2)	20	1.05	0.006	0.096	0.126
	50	3.16	0.15	0.074	0.098
	100	8.01	0.030	0.019	0.090
DEC(10)	20	19.57	0.006	0.014	0.019
lbfgs(500)	20	13.38	0.006	0.142	0.142
lbfgs(1000)	20	26.02	0.006	0.127	0.127
elm	100	1.02	0.031	0.220	0.219
	200	2.98	0.061	0.199	0.199
	500	9.14	0.154	0.178	0.177

TABLE VIII
CALIFORNIA

algorithm	neurons	tr_time	ts_time	RMSEtr	RMSEts
DEC(2)	20	2.983	0.006	0.145	0.191
	50	8.090	0.15	0.134	0.178
	100	16.387	0.030	0.125	0.169
DEC(10)	20	20.76	0.006	0.102	0.142
lbfgs(500)	20	142.29	0.006	0.195	0.198
lbfgs(1000)	20	546.95	0.006	0.162	0.164
elm	100	0.54	0.031	0.203	0.205
	200	1.47	0.061	0.197	0.200
	500	7.14	0.154	0.189	0.198

TABLE IX
CENSUS

algorithm	neurons	tr_time	ts_time	RMSEtr	RMSEts
DEC(2)	20	6.146	0.008	0.056	0.092
	50	14.905	0.020	0.058	0.095
	100	33.689	0.042	0.055	0.092
DEC(10)	20	71.478	0.008	0.048	0.083
lbfgs(500)	20	214.78	0.008	0.094	0.089
lbfgs(1000)	20	802.26	0.008	0.093	0.088
elm	100	0.65	0.041	0.100	0.101
	200	1.68	0.082	0.100	0.105
	500	7.78	0.207	0.098	0.106

TABLE X
DELTAELV

algorithm	neurons	tr_time	ts_time	RMSEtr	RMSEts
DEC(2)	20	0.616	0.001	0.041	0.057
	50	1.533	0.006	0.040	0.056
	100	3.093	0.013	0.039	0.055
DEC(10)	20	3.135	0.001	0.040	0.055
lbfgs(500)	20	10.417	0.001	0.053	0.055
lbfgs(1000)	20	20.574	0.001	0.053	0.055
elm	100	0.200	0.013	0.052	0.056
	200	0.639	0.027	0.051	0.061
	500	3.518	0.068	0.049	0.305

the root mean square error on the training data (RMSEtr), and the root mean square error on the testing data (RMSEts). Again, the train and test times are reported in seconds. The root mean square errors on a set $\bar{T} = \{(\bar{x}^s, \bar{y}^s) : \bar{x}^s \in R^n,\ \bar{y}^s \in R,\ s = 1, 2, \ldots, S\}$ are defined as follows:

$$\mathrm{RMSE}_T = \sqrt{\sum_{s \in \bar{T}} (\phi(\bar{x}^s, \lambda, w) - \bar{y}^s)^2 / |\bar{T}|}$$

where $\bar{y}^s$ is the actual output, $\phi(\bar{x}^s, \lambda, w)$ is the predicted output for the input pattern $\bar{x}^s$, and $|\bar{T}|$ denotes the cardinality of T.

TABLE XI
ELEVATORS

algorithm	neurons	tr_time	ts_time	RMSEtr	RMSEts
DEC(2)	20	6.60	0.008	0.092	0.092
	50	17.06	0.019	0.091	0.093
	100	34.58	0.038	0.089	0.095
DEC(10)	20	65.93	0.008	0.085	0.087
lbfgs(500)	20	32.19	0.008	0.092	0.092
lbfgs(1000)	20	63.81	0.008	0.092	0.092
elm	100	0.448	0.038	0.097	0.097
	200	1.252	0.075	0.097	0.098
	500	6.070	0.188	0.094	0.098

TABLE XII
AILERONS

algorithm	neurons	tr_time	ts_time	RMSEtr	RMSEts
DEC(2)	20	5.671	0.008	0.066	0.065
	50	9.376	0.019	0.062	0.062
	100	18.085	0.038	0.059	0.061
DEC(10)	20	1.605	0.008	0.048	0.048
lbfgs(500)	20	47.21	0.008	0.060	0.060
lbfgs(1000)	20	94.59	0.008	0.052	0.052
elm	100	0.401	0.038	0.105	0.105
	200	1.084	0.075	0.104	0.105
	500	5.389	0.188	0.09	0.102

As regards the comparison between the two versions of the proposed algorithm, DEC(2) and DEC(10), we may observe that this latter shows better performance in terms of accuracy at expense of an increase, as we may expect, of the computing time in the training process. The alternating minimization with respect to input and output weights performed for ten main iterations reaches suitable levels of accuracy with a relatively limited number of hidden neurons, and this yields benefits in terms of test time. For instance, we can see in Table II that the network trained by DEC(10), with 20 neurons, has a classification accuracy on the test set of 95.95% and a test time of 0.40 s, while the network trained by DEC(2), with 100 neurons, has a classification accuracy on the test set of 94.93% and a test time of 2.02 s.

From the comparison between DEC(10) and L-BFGS, we get that the former outperforms both versions of the latter in 9 cases over 11 in terms of training time, and in 9 cases over 11 in terms of accuracy level. L-BFGS outperforms DEC(10) in 2 cases over 11 in terms of training time, and in 1 case over 11 in terms of accuracy level. Note that in 7 cases over 11 DEC(10) outperforms L-BFGS both in terms of training time and of accuracy level. It is also to be noted that the differences in terms of computing times are quite relevant.

On the whole, the comparison between DEC(10) and L-BFGS shows the advantages of a decomposition-based strategy with respect to a standard optimization approach.

Finally, concerning the comparison between DEC(10) and a simple ELM, we can note that in all cases, the level of accuracy of networks trained by DEC(10) is substantially better than that obtained by a network trained by ELM (evaluated with different numbers of hidden neurons). As well known, this latter is very efficient in terms of training time, in particular whenever the number of neurons is not too high. However, to attain suitable performance in terms of level of accuracy, in some cases, it is necessary to increase the number of neurons,

at expense of a significant increase of the training time, of the testing time, and of the memory requirement. Summarizing, from the comparison between DEC(10) and a simple ELM, we get that the former is suitable whenever a high level of accuracy is required, and the latter can be advantageously employed whenever the training process must be very fast (for instance, in applications real time).

VII. Conclusion

In this paper, we have introduced a class of algorithms for training SLFNs, based on the decomposition of the learning problem into a sequence of smaller and structured minimization problems. The main feature of these techniques can be that of providing, in most of cases, good generalization performances, with much smaller training times, in comparison with efficient standard algorithms without decomposition. In some instances, the decomposition algorithms adopted here can also provide smaller training and testing errors by escaping from the attraction of irrelevant local minimizers, through the global minimization with respect to the output weights, performed at each major step.

Our approach can also include, in particular, extensions of the basic ELM technique, where input parameters can also be updated for a few cycles. This can be useful whenever a greater accuracy is requested, and the number of neural units must be kept not very large, for guaranteeing small evaluation times in the recall phase.

The results established here can be further extended along several directions. A first possibility can be that of experimenting also block decompositions of the output weights, in association with block minimizations with respect to input parameters, by generalizing the scheme defined in Algorithm DEC.

A second extension can be that of employing decomposition techniques for training MLPs with more than one hidden layer. In this case, in the same spirit of the extensions proposed for ELMs [17], we can retain, in principle, the distinction between output weights and weights associated with hidden layers. The parameters associated with hidden layers can be further decomposed into different blocks, possibly associated with the different layers.

Finally, the algorithms experimented here in the simplest form, for permitting easier comparisons, can be realized by introducing validation tests for choosing the number of neurons and the weighting parameters. Other implementations can also be considered by replacing the reduced memory quasi-Newton method L-BFGS with other minimization techniques, such as some nonmonotone implementations of the spectral gradient method [10], [25].

Appendix A

In this appendix, we report in detail the convergence proof of Algorithm DAM of Section IV.

First, we introduce some additional notations. Given an index set $J \subseteq \{1, 2, \ldots, N\}$, if $J \neq \emptyset$, we denote the vector components of λ by letting (after reordering) $\lambda = (\lambda_J^T, \lambda_{\bar{J}}^T)^T$, where λ_J is the subvector with components λ_j, $j \in J$ and $\lambda_{\bar{J}}$ is formed with the remaining components.

Similarly, given $L \subseteq \{1, 2, \ldots, N\}$, if $L \neq \emptyset$ (after reordering), we set, $w = (w_L^T, w_{\bar{L}}^T)^T$, where w_L is the subvector with components w_j, $j \in L$ and $w_{\bar{L}}$ has the remaining components.

Then, the partial gradient and the partial Hessian matrix of E relative to λ_J are denoted, respectively, by $\nabla_J E$ and $\nabla_J^2 E$.

We start by establishing the following lemma, which is a consequence of the strict convexity of E with respect to λ, when w is fixed.

Lemma 1: Let $\{(\lambda^k, w^k)\}$ be an infinite sequence of points in $\mathcal{L}_0$ and assume that we have, for all k

$$E(\lambda^{k+1}, w^{k+1}) \leq E(\lambda^{k+1}, w^k) \leq E(\lambda^k, w^k). \tag{39}$$

Suppose there exist a subsequence $\{(\lambda^k, w^k)\}_K$ and an index set $J \subseteq \{1, 2, \ldots, N\}$ such that, for all $k \in K$ we have

$$\lambda^{k+1} = \left(\left(\lambda_J^{k+1}\right)^T, \left(\lambda_{\bar{J}}^{k+1}\right)^T\right)^T \quad \text{with } \lambda_{\bar{J}}^{k+1} = \lambda_{\bar{J}}^k. \tag{40}$$

Suppose also that

$$\lim_{k \in K, k \to \infty} \nabla_J E(\lambda^{k+1}, w^k) = 0. \tag{41}$$

Then, we have

$$\lim_{k \in K, k \to \infty} \left\| \lambda_J^{k+1} - \lambda_J^k \right\| = 0. \tag{42}$$

Proof: By (39) and the compactness of $\mathcal{L}_0$, we have that $\{E(\lambda^k, w^k)\}$ converges to a limit and also that

$$\lim_{k \to \infty} \{E(\lambda^k, w^k) - E(\lambda^{k+1}, w^k)\} = 0. \tag{43}$$

In order to simplify notation, let us define

$$g_k = \nabla_J E(\lambda^{k+1}, w^k) \quad Q_k = \nabla_J^2 E(\lambda^{k+1}, w^k) \tag{44}$$

where Q_k does not depend on λ^{k+1}. Considering the expression of E and the assumption (40), which implies, for all $k \in K$, that $\lambda_{\bar{J}}^{k+1} - \lambda_{\bar{J}}^k = 0$, we can write, for all $k \in K$

$$\begin{aligned} E(\lambda^k, w^k) = {} & E(\lambda^{k+1}, w^k) + g_k^T \left(\lambda_J^k - \lambda_J^{k+1}\right) \\ & + 1/2 \left(\lambda_J^k - \lambda_J^{k+1}\right)^T Q_k \left(\lambda_J^k - \lambda_J^{k+1}\right) \end{aligned} \tag{45}$$

where the partial Hessian matrix Q_k, relative to λ_J, is definite positive with the smallest eigenvalue $\sigma_{\min}(Q_k)$ that satisfies $\sigma_{\min}(Q_k) \geq \rho_1$, for all k and a $\rho_1 > 0$. Then, from (45), we obtain

$$\begin{aligned} E(\lambda^k, w^k) - E(\lambda^{k+1}, w^k) \geq {} & -\|g_k\| \left\| \lambda_J^k - \lambda_J^k \right\| \\ & + 1/2 \rho_1 \left\| \lambda_J^k - \lambda_J^{k+1} \right\|^2. \end{aligned} \tag{46}$$

Taking limits and recalling (41) and (43), we obtain (42), and this concludes the proof. ∎

Next, we state as a lemma a property of limit points of a sequence of points defined in an Euclidean space R^d. To avoid confusion with the variables of our problems, we denote by $z^k \in R^d$ a point of this sequence.

Lemma 2: Let $\{z^k\}$, with $z^k \in R^d$, be a sequence such that

$$\lim_{k \to \infty} \|z^{k+1} - z^k\| = 0 \tag{47}$$

and let $\bar{z}$ be a limit point, so that there exists a subsequence $\{z^k\}_K$ converging to $\bar{z}$. Then, for every fixed integer $M > 0$, we have that every subsequence $\{z^{k+j}\}_K$, for $j = 1, \ldots, M$ converges to $\bar{z}$.

Proof: First, we note that, for each $j \in \{1, \ldots, M\}$, we can write: $z^{k+j} - z^k = \sum_{h=0}^{j-1}(z^{k+h+1} - z^{k+h})$, whence it follows, for $j = 1, \ldots, M$ that:

$$\|z^{k+j} - z^k\| \leq \sum_{h=0}^{j-1} \|z^{k+h+1} - z^{k+h}\| \tag{48}$$

where, by assumption, we have

$$\lim_{k\to\infty} \|z^{k+h+1} - z^{k+h}\| = 0, \quad h = 0, \ldots, j-1. \tag{49}$$

As the sum in (48) has a finite number of terms, bounded above by M, the limit (49) also implies that

$$\lim_{k\to\infty} \|z^{k+j} - z^k\| = 0, \quad j = 1, \ldots, M. \tag{50}$$

On the other hand, for every $j = 1, \ldots, M$, we can write

$$\|z^{k+j} - \bar{z}\| = \|z^{k+j} - z^k + z^k - \bar{z}\| \leq \|z^{k+j} - z^k\| + \|z^k - \bar{z}\|$$

and therefore, recalling that, by assumption, we have that $\|z^k - \bar{z}\| \to 0$, for $k \in K, k \to \infty$ and considering (50), we can establish our thesis. ∎

Now, we prove the convergence result stated in Proposition 1.

Proof of Proposition 1:

Proof: First, we remark that when $J^k = \emptyset$ we have $\lambda^{k+1} = \lambda^k$ and similarly that $L^k = \emptyset$ implies $w^{k+1} = w^k$. Considering this, from Conditions 2 and 3 and the instructions of Algorithm DAM, it follows that the following inequality holds for all k:

$$E(\lambda^{k+1}, w^{k+1}) \leq E(\lambda^{k+1}, w^k) \leq E(\lambda^k w^k). \tag{51}$$

This implies, by induction, that all points of the sequence $\{(\lambda^k, w^k)\}$ and also all points (λ^{k+1}, w^k) remain in the compact level set $\mathcal{L}_0$. Then, the sequence $\{E(\lambda^k, w^k)\}$ converges to a limit, which is also the limit of $\{E(\lambda^{k+1}, w^k)\}$. so that assertions 1) and 2) must hold.

We now establish assertion 3). By contradiction, let us assume that assertion 3) is false. This implies that there exist a subsequence $\{(\lambda^k, w^k)\}_K$, and a number $\varepsilon > 0$ such that, for all sufficiently large $k \in K$, say $k \geq \hat{k}$, we have

$$\|\lambda^{k+1} - \lambda^k\| \geq \varepsilon \quad \forall k \in K, \quad k \geq \hat{k} \tag{52}$$

so that, for all $k \in K$, $k \geq \hat{k}$, the index set J^k considered in Algorithm DAM must be nonempty. Moreover, as the number of components of λ is obviously finite, we can find a further subsequence $\{(\lambda^k, w^k)\}_{K_1}$, with $K_1 \subseteq K$, and an index set J^*, such that the set J^k considered in Algorithm DAM satisfies $J^k = J^*$, and hence from (52), we get

$$\left\|\lambda_{J^*}^{k+1} - \lambda_{J^*}^k\right\| \geq \varepsilon \quad \forall k \in K_1, \quad k \geq \hat{k}. \tag{53}$$

By the instructions at Step 1, recalling Condition 2, we have that $\lim_{k\in K_1, k\to\infty} \nabla_{\lambda_{j^*}} E(\lambda^{k+1}, w^k) = 0$, and therefore, from Lemma 1, where we identify J with J^*, we get a contradiction to (53), and this implies that assertion 3) must hold.

Consider now assertion 4). Assume there exist a subsequence, which we call again $\{(\lambda^k, w^k)\}_K$, and a number $\varepsilon > 0$ such that, for all $k \in K, k \geq \hat{k}$, we have

$$\|w^{k+1} - w^k\| \geq \varepsilon \quad \forall k \in K, \quad k \geq \hat{k}. \tag{54}$$

Reasoning as in case 3), we can find a further subsequence $\{(\lambda^k, w^k)\}_{K_1}$, with $K_1 \subseteq K$, and an index set L^*, with $L^k = L^*$, such that from (54), we get

$$\left\|w_{L^*}^{k+1} - w_{L^*}^k\right\| \geq \varepsilon \quad \forall k \in K_1, \quad k \geq \hat{k}. \tag{55}$$

On the other hand, as L^* is nonempty, Condition 3 and assertion 2) imply that $\lim_{k\in K_1, k\to\infty} \|w_{L^*}^{k+1} - w_{L^*}^k\| = 0$, which contradicts (55). We can conclude that also assertion 4) must hold.

Finally, we prove that assertion 5) holds true. Reasoning by contradiction, let us assume that there exists a subsequence $\{(\lambda^k, w^k)\}_K$ converging to a point $(\bar{\lambda}, \bar{w})$ and such that $\nabla E(\bar{\lambda}, \bar{w}) \neq 0$. Suppose first, there exist an index, say $j^* \in \{1, \ldots, N\}$, and a number $\eta > 0$, such that

$$|\nabla_{\lambda_{j^*}} E(\bar{\lambda}, \bar{w})| \geq \eta. \tag{56}$$

By Condition 1, we can construct another subsequence, which we denote by $\{(\lambda^{h_k}, w^{h_k})\}_K$, such that $j^* \in J^{h_k}$, with $k \leq h_k \leq k+M$, with an integer $M > 0$. Now, as $j^* \in J^{h_k}$, by Condition 2, we have $\lim_{k\in K, k\to\infty} \nabla_{\lambda_{j^*}} E(\lambda^{h_k+1}, w^{h_k}) = 0$. Recalling assertions 3) and 4), and Lemma 2, we have that $\{(\lambda^{h_k+1}, w^{h_k})\}_K$ must converge to the same limit point $(\bar{\lambda}, \bar{w})$. But then, taking limits for $k \in K, k \to \infty$, we have $\nabla_{\lambda_{j^*}} E(\bar{\lambda}, \bar{w}) = 0$, which contradicts our assumption.

Assume now that there exist $j^* \in \{1, \ldots, N\}$ and $\eta > 0$, such that

$$\|\nabla_{w_{j^*}} E(\bar{\lambda}, \bar{w})\| \geq \eta. \tag{57}$$

Recalling again Condition 1, we can construct another subsequence, which we denote again by $\{(\lambda^{h_k}, w^{h_k})\}_K$, such that $j^* \in L^{h_k}$, with

$$k \leq h_k \leq k + M \tag{58}$$

for an integer $M > 0$.

As $j^* \in L^{h_k}$, recalling Condition 3, suppose that for an infinite subsequence $K_1 \subseteq K$, we have

$$\lim_{k\in K_1, k\to\infty} \nabla_{w_{j^*}} E(\lambda^{h_k+1}, w^{h_k}) = 0. \tag{59}$$

Recalling assertions 3) and 4), and Lemma 2, we have that $\{(\lambda^{h_k+1}, w^{h_k})\}_{K_1}$ must converge to the same limit point $(\bar{\lambda}, \bar{w})$ and hence, taking limits for $k \in K_1, k \to \infty$, we have

$$\nabla_{w_{j^*}} E(\bar{\lambda}, \bar{w}) = 0 \tag{60}$$

which contradicts our assumption. This concludes the proof. ∎

APPENDIX B

We establish here the convergence properties of the Armijo-type method defined in Section V. Preliminarily, in order to simplify notation, let us introduce, for $j = 1 \ldots N$, the 1-D function ϕ_j^k defined as

$$\phi_j^k(\alpha_j) = E\left(\lambda^{k+1}, w_1^k, \ldots, w_j^k + \alpha_j d_j^k, \ldots, w_N^k\right) \tag{61}$$

so that $\phi_j^k(0) = E(\lambda^{k+1}, w^k)$ and

$$\dot{\phi}_j^k(0) = \nabla_{w_j} E(\lambda^{k+1}, w^k)^T d_j^k = -\|\nabla_{w_j} E(\lambda^{k+1}, w^k)\|^2$$

where $\dot{\phi}_j^k(0) \leq 0$ is the directional derivative along d_j^k with respect to w_j when the remaining variables are fixed.

Then, the sufficient descent condition in Armijo's method can be put into the form: $\phi_j^k(\alpha) \leq \phi_j^k(0) + \gamma\, \alpha \dot{\phi}_j^k(0)$. The proof of Proposition 2 is given below.

Proof of Proposition 2:

Proof: A finite termination of Armijo's method for fixed j and k follows immediately from the assumptions that E is continuously differentiable and that $0 < \gamma < 1$ (see [3]). At termination, we have

$$\phi_j^k\left(\alpha_j^k\right) \leq \phi_j^k(0) + \gamma\, \alpha_j^k \dot{\phi}_j^k(0) < \phi_j^k(0) \tag{62}$$

and this implies (i). We have also that

$$\phi_j^k(0) - \phi_j^k\left(\alpha_j^k\right) \geq \gamma\, \alpha_j^k \left\| d_j^k \right\|^2 \geq (\gamma/a) \left\| \alpha_j^k d_j^k \right\|^2 \tag{63}$$

so that (ii)-(a) must hold. Finally, assume that a subsequence $\{(\lambda^{k+1}, w^k)\}_{K_1}$ converges to a point $(\bar{\lambda}, \bar{w})$, so that, by continuity, we have

$$\lim_{k \in K_1, k \to \infty} d_j^k = -\nabla_{w_j} E(\bar{\lambda}, \bar{w}). \tag{64}$$

Now suppose, by contradiction, that (ii)-(b) is false. This implies that

$$\left\| \nabla_{w_j} E(\bar{\lambda}, \bar{w}) \right\| > 0. \tag{65}$$

Now, there are two possibilities: the first is that $\alpha_j^k = a$ for an infinite subsequence, say $k \in K_2 \subseteq K_1$; the second is that for all sufficiently large $k \in K_1$, we have $\alpha_j^k < \alpha_j^k/\delta \leq a$. In the first case, it easily seen from (63) that $\lim_{k \in K_2, k \to \infty} d_j^k = 0$, which yields a contradiction to (65). In the second case, for all sufficiently large $k \in K_1$, the initial tentative stepsize has been reduced and hence, for $k \in K_1$, we have necessarily that

$$\phi_j^k\left(\frac{\alpha_j^k}{\delta}\right) - \phi_j^k(0) > \gamma\, \frac{\alpha_j^k}{\delta} \dot{\phi}_j^k(0). \tag{66}$$

By the theorem of the mean, we can write

$$\phi_j^k\left(\frac{\alpha_j^k}{\delta}\right) = \phi_j^k(0) + \frac{\alpha_j^k}{\delta} \dot{\phi}_j^k\left(\xi_j^k\right) \tag{67}$$

where $\xi_j^k = t_j^k(\alpha_j^k/\delta)$ for $t_j^k \in (0, 1)$. Therefore, we can write $\dot{\phi}_j^k(\xi_j^k) > \gamma\, \dot{\phi}_j^k(0)$, and this is equivalent, by definition, to

$$\begin{aligned} \nabla_{w_j} E\left(\lambda^{k+1}, w_1^k, \ldots, w_j^k + t_j^k \frac{\alpha_j^k}{\delta} d_j^k, \ldots, w_N^k\right)^T d_j^k \\ > \gamma\, \nabla_{w_j} E\left(\lambda^{k+1}, w_1^k, \ldots, w_j^k, \ldots, w_N^k\right)\Big)^T d_j^k. \end{aligned} \tag{68}$$

Taking limits for $k \in K_1$ and recalling (ii)-(a) and (64), we obtain $-\|\nabla_{w_j} E(\bar{\lambda}, \bar{w})\|^2 \geq -\gamma \|\nabla_{w_j} E(\bar{\lambda}, \bar{w})\|^2$, and this contradicts (65), as $\gamma \in (0, 1)$. This concludes the proof. ■

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers whose careful analysis of the first version of this paper have greatly contributed to correct and to improve their work.

REFERENCES

[1] E. Anderson *et al.*, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA, USA: SIAM, 1999.
[2] A. V. Uzilov, J. M. Keegan, and D. H. Mathews, "Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change," *BMC Bioinform.*, vol. 7, p. 173, Dec. 2006.
[3] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA, USA: Athena Scientific, 1999.
[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, 1989.
[5] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Clarendon, 1995.
[6] S. Bonettini, "Inexact block coordinate descent methods with application to non-negative matrix factorization," *IMA J. Numer. Anal.*, vol. 31, no. 4, pp. 1431–1452, 2011.
[7] C. Buzzi, L. Grippo, and M. Sciandrone, "Convergent decomposition techniques for training RBF neural networks," *Neural Comput.*, vol. 13, no. 8, pp. 1891–1920, 2001.
[8] A. Cassioli, D. Di Lorenzo, and M. Sciandrone, "On the convergence of inexact block coordinate descent methods for constrained optimization," *Eur. J. Oper. Res.*, vol. 231, no. 2, pp. 274–281, 2013.
[9] A. Cassioli and M. Sciandrone, "A convergent decomposition method for box-constrained optimization problems," *Optim. Lett.*, vol. 3, no. 3, pp. 397–409, 2009.
[10] L. Grippo and M. Sciandrone, "Nonmonotone globalization techniques for the Barzilai–Borwein gradient method," *Comput. Optim. Appl.*, vol. 23, no. 2, pp. 143–169, 2002.
[11] L. Grippo and M. Sciandrone, "Globally convergent block-coordinate techniques for unconstrained optimization," *Optim. Methods Softw.*, vol. 10, no. 4, pp. 587–637, 1999.
[12] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1999.
[13] M. R. Hestenes, *Conjugate Direction Methods in Optimization*. New York, NY, USA: Springer-Verlag, 1980.
[14] G.-B. Huang, "An insight into extreme learning machines: Random neurons, random features and kernels," *Cognit. Comput.*, vol. 6, no. 3, pp. 376–390, 2014.
[15] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.
[16] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, nos. 4–6, pp. 576–583, 2008.
[17] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 107–122, 2011.
[18] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.
[19] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.
[20] J. Alcalá-Fdez *et al.*, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Logic Soft Comput.*, vol. 17, nos. 2–3, pp. 255–287, 2011.
[21] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, no. 1, pp. 503–528, 1989.
[22] Z. Q. Luo and P. Tseng, "On the convergence of the coordinate descent method for convex differentiable minimization," *J. Optim. Theory Appl.*, vol. 72, no. 1, pp. 7–35, 1992.
[23] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, 2012.
[24] A. Pinkus, "Approximation theory of the MLP model in neural networks," *Acta Numer.*, vol. 8, pp. 143–195, Jan. 1999.
[25] M. Raydan, "The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem," *SIAM J. Optim.*, vol. 7, no. 1, pp. 26–33, 1997.
[26] A. J. Shepherd, *Second-Order Methods for Neural Networks*. London, U.K.: Springer-Verlag, 1997.
[27] W. Zong, G.-B. Huang, and L. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, Feb. 2013.

Luigi Grippo was a Full Professor of Operations Research with the Sapienza, Università di Roma, Rome, Italy, from 1990 to 2012. Since 2012, he has been appointed as Professor of Optimization Algorithms with the Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti of the same university.

Andrea Manno was born in Rome, Italy, in 1985. He received the bachelor's and master's degrees in management engineering and the Ph.D. degree in operations research from the Sapienza Università di Roma, Rome, in 2008, 2011, and 2015, respectively.

He is currently a Research Fellow with the Università di Firenze, Florence, Italy.

Marco Sciandrone was born in Veroli, Italy, in 1965. He received the M.Sc. degree in electrical engineering and the Ph.D. degree in systems engineering from the Sapienza Universitá di Roma, Rome, Italy, in 1991 and 1997, respectively.

He has been an Associate Professor with the Università di Firenze, Florence, Italy, since 2006. He has authored or co-authored about 40 papers in international journals. His current research interests include operations research, nonlinear optimization, and machine learning.

Prof. Sciandrone has been an Associate Editor of the *Optimization Methods and Software* journal since 2009.