Local search and pseudoinversion: an hybrid approach to neural network training

Under consideration for publication in Knowledge and Information Systems

Local Search and Pseudoinversion: an Hybrid Approach to Neural Network Training

Luca Rubini[1], Rossella Cancelliere[1], Patrick Gallinari[2] and Andrea Grosso[1]

[1] University of Turin, Department of Computer Sciences, Turin, Italy;
[2] Laboratory of Computer Sciences, LIP6, Université Pierre et Marie Curie, Paris, France

Abstract. We consider recent successful techniques proposed for neural network training that set randomly the weights from input to hidden layer, while weights from hidden to output layer are analytically determined by Moore-Penrose generalised inverse. This study aims to analyze the impact on performances when the completely random sampling of the space of input weights is replaced by a local search procedure over a discretized set of weights. The performances of the proposed training methods are assessed through computational experience on several UCI datasets.

Keywords: Neural networks; Random projections; Local search; Pseudoinverse matrix

1. Introduction

In the past two decades methods based on gradient descent have largely been used for training of one of the most used architectures, the single hidden layer feedforward neural network (SLFN). The start-up of these techniques assigns random values to the weights connecting input, hidden and output nodes; such values are then iteratively modified, as for the traditional backpropagation algorithms (Rumelhart et al, 1996).

Recently some non-iterative procedures based on the evaluation of generalized pseudoinverse matrices (see for instance (Huang et al, 2006) with Extreme Learning Machine — ELM, (Halawa, 2011) and special issues on Soft Computing, (Wang et al, 2012) and the International Journal of Uncertainty, Fuzziness

Received Nov 12, 2014
Revised Oct 28, 2015
Accepted Feb 06, 2016

and Knowledge-Based Systems, (Wang, 2013)) have received considerable attention in computational intelligence and machine learning communities in both theoretical study and applications. Unlike the traditional algorithms in which weights are adjusted iteratively, in such framework input weights and hidden layer biases are randomly generated, usually according to a uniform distribution in the interval [-1, 1], and output weights analytically determined using Moore-Penrose generalized inverse.

Many application-oriented studies in the last years established the effectiveness of pseudoinversion-based methods; some are described for example in (Nguyen et al, 2010; Kohno et al, 2010; Ajorloo et al, 2007).

Pseudoinversion based methods are claimed to solve the well known problem concerning the slow learning speed of the traditional backpropagation methods, avoiding the necessity of tuning the network parameters. i.e. input and output weights. In (Huang et al, 2006) a pseudoinversion method (specifically, ELM) is reported to be from 10 up to 100 times faster than backpropagation. The superior speed of pseudoinversion based methods is further shown in (Deng et al, 2009; Martinez-Martinez et al, 2011). Hence we focus this study mainly on pseudoinversion based methods.

The problem of the possible convergence to poor local minima is handled by repeatedly applying the method with a number of random initializations (multistart), thereby obtaining a sampling "at large" of the landscape of the error function.

In the field of combinatorial optimization, a popular alternative to the multistart approach is the so-called "local search". Local search algorithms often work in (but are not limited to) discrete search spaces, generating an improving sequence of solution points, say $\mathbf{y}_1, \mathbf{y}_2 \ldots, \mathbf{y}_k, \ldots$ where each point $\mathbf{y}_k$ is drawn from a (finite) set of neighbour solutions $\mathcal{N}(\mathbf{y}_{k-1})$; the definition of the neighbourhood structure $\mathcal{N}(\cdot)$ basically determines the behaviour of the algorithm. For an excellent survey we refer the reader to (Aarts and Lenstra, 2003).

The aim of this paper is to study the practical effect of replacing the completely random sampling of the space of input weights used by pseudoinversion-based methods with a more sophisticated exploration, guided by a local search procedure in a discrete (or discretized) set of solutions. We experimentally show that, while keeping the computational effort substantially unchanged, the multistart technique in pseudoinversion-based methods can be effectively replaced by local search, leading to gains in performances.

The paper is organized as follows. We recall main ideas on SLFN learning by pseudoinversion in section 2; in section 3 we describe the local search technique. Finally in section 4 we report results comparing weights setting and local search strategies.

2. Training by pseudoinversion

In this section we introduce notation and we recall basic idea concerning the use of generalized inverse for neural training. In a typical setting, we consider a standard SLFN (see Fig. 1) with P input neurons, M hidden neurons, Q output neurons (labeled o in the figure) and nonlinear activation function Φ in the hidden layer. The output layer is assumed to have linear activation functions. A dataset of N training samples consisting of input instances $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^P$, along with respective desired outputs $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N \in \mathbb{R}^Q$ is given; we arrange the $\mathbf{x}_j$'s

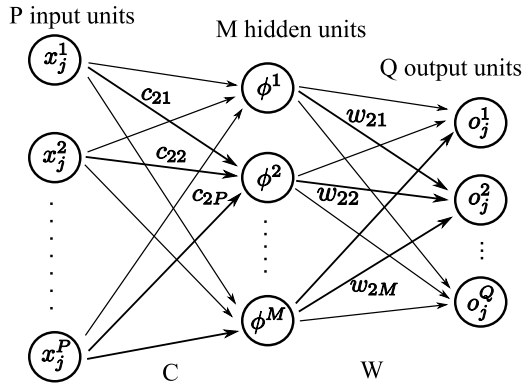

Fig. 1. A Single Layer Feedforward Neural Network

and $\mathbf{t}_j$'s in two N-row matrices $X \in \mathbb{R}^{N\times P}$ and $T \in \mathbb{R}^{N\times Q}$. The training problem consists of finding matrices of input weights $C = (c_{ij} : i = 1, \ldots, M, j = 1, \ldots, P)$ and output weights $W = (w_{ij} : i = 1, \ldots, M, j = 1, \ldots, Q)$ in order to

$$\underset{C,W}{\text{minimize}}\ E(C, W) = \|\Phi(XC)W - T\|_2^2, \tag{1}$$

which is the so-called training error function.

The classical pseudoinversion-based approach to the training problem is based on the following simple key observation. If the matrix of input weight C is fixed, problem (1) reduces to the unconstrained, convex optimization problem

$$\underset{W}{\text{minimize}}\ E_D(W) = \|HW - T\|_2^2, \tag{2}$$

where $H = \Phi(XC)$ is now fixed. For given C, problem (2) is known to be optimally solved by

$$W^* = H^+T, \tag{3}$$

where $H^+ \in \mathbb{R}^{N\times M}$ is the Moore-Penrose generalized pseudoinverse. An approximated (heuristic) solution for problem (1) is then computed by randomly generating (from a uniform distribution in $[-1, 1]$) a number of C matrices and taking among them the one that delivers the minimum E_D.

Since most learning problems are ill-posed, instead of directly using eq. (3) often *regularization methods* have to be used (Badeva and Morosov, 1991; Cancelliere et al, 2015) to turn the original problem into a well-posed one. Tikhonov regularization is one of the most common (Tikhonov and Arsenin, 1977; Tikhonov, 1963), and leads to solve

$$\underset{W}{\text{minimize}}\ E_D(W) + E_R(W) = ||HW - T||_2^2 + \lambda||W||_2^2. \tag{4}$$

The regularised solution $\hat{W}$ that minimises the error functional in (4) has the form (see e.g. (Fuhry et al, 2012)):

$$\hat{W} = (H^TH + \lambda I)^{-1}H^TT\,. \tag{5}$$

According to a vast experience, regularization not only improves on stability,

but also contains model complexity avoiding overfitting, as largely discussed in (Gallinari and Cibas, 1999). Applications to different neural network models are discussed for instance in (Poggio and Girosi, 1990; Girosi et al, 1995; Haykin, 1999).

In our work we always utilise regularised pseudoinversion; direct non-iterative application of eq. (5) for output weights determination will be referred to as PINV technique (PINV is therefore used as a shortening for pseudoinversion). Input weights setting is discussed in section 4.

3. Enhancing training by local search

We elaborated on the framework of pseudoinversion-based learning, proposing a local search procedure that generates a sequence of input weights $C^1, C^2, \ldots, C^k, \ldots$ such that $C^k \in \mathcal{N}(C^{k-1})$ and $\hat{E}(C^k) < \hat{E}(C^{k-1})$ where $\hat{E}(C)$ is the error computed on a suitable validation set V:

$$\hat{E}(C) = \|\Phi(VC)\hat{W} - T_V\|_2^2 + \lambda\|\hat{W}\|_2^2 \tag{6}$$

with

$$\hat{W} = \arg\min_W \left\{\|\Phi(XC)W - T\|_2^2 + \lambda\|W\|_2^2\right\}. \tag{7}$$

$\hat{W}$ is computed accordingly with Section 2. The structure of the neighbourhood $\mathcal{N}(\cdot)$ largely determines the behaviour of the procedure. We define a discrete neighbourhood $\mathcal{N}(C)$ made up of all matrices C' that can be obtained by permuting a pair of distinct rows of C. Other neighborhood definitions are obviously possible. It is usually reasonable to select the neighbour $C' \in \mathcal{N}(C)$ for which $\hat{E}(C')$ is minimum; this is called a *best-improve* neighbourhood exploration.

Anyway a single neighbourhood of C contains $M(M-1)/2$ matrices, meaning $O(M^2)$ pseudoinversions that need to be computed, and this can be considered computationally too heavy for many applications. We then implemented a so-called *first-improve* variant of local search where the neighbourhood exploration is terminated as soon as an improving neighbour has been generated. The routine moves through such neighbourhoods as shown in Procedure LS.

At the k-th iteration, the neighbors $C' \in \mathcal{N}(C^k)$ of the current solution C^k are generated and the corresponding value of validation error $\hat{E}(C')$ is computed. As soon as a neighbour with $\hat{E}(C') < \hat{E}(C^k)$ is found the neighbourhood exploration is interrupted and C' becomes the new current solution in the next iteration. A full neighbourhood exploration is performed only when no neighbour solution is better than the current one (i.e., C^k is locally optimal). Each neighbor C' is obtained by a "small perturbation" of C^k, still the perturbation is not taken along the gradient direction and the search can avoid the very small improvements that cramp the basic gradient-descent procedures. Instead, the landscape of the function $\hat{E}$ is sampled with a wider step, without a completely random restart. The rationale behind this is that we would like to pursue exploration of the landscape "not too far" from a point that can be perceived as a good solution, exploiting locality in the search. Anyway, similarly to what is done with the basic pseudoinversion approaches, a multistart application of the local search procedure is recommended for better performances: after a region of the landscape of $\hat{E}$ has been explored, a random restart can drive the search

towards a different region, that would not be reachable only through exploration of neighborhoods, providing what is usually called "diversification" of the search.

In order to handle the classification problems, some care has to be taken. Indeed, the landscape of the misclassification error often exhibits large plateaus, so it's very difficult to improve a solution (i.e., changing plateau) switching only two rows per time. To avoid this problem we have introduced in the LS procedure a second level error function: we measure the RMSE between the target tuple and the network output. We accept the new solution if the misclassification error is lower or if the misclassification error is equal but the RMSE is decreased.

4. Experimental investigation

In this section we report results of some numerical experiments performed on the twelve benchmark datasets from the UCI repository (Asuncion and Newman, 2007) listed in Table 1. We investigate neural networks with the architecture shown in Fig. 1 having sigmoidal hidden neuron activation function Φ. The number of input and output neurons is determined by dataset features.

All simulations are carried out in Matlab 7.3 environment.

Founding on the results obtained in (Rubini et al, 2014), we also investigated the effectivness of initializing inputs weigths C not only with the standard uniform-random setting, but also using random projections.

The random mapping is a tool developed to treat high dimensional data reducing the complexity.

The theoretical basis of this method come for from the Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss, 1984): if a set of points in a vector space is projected onto a randomly selected subspace of suitable dimension, then the distances between each points are approximately preserved in the new space.

Procedure LS$(C, X, T, V, T_V\lambda)$

input : An initial $M \times P$ matrix of input weights C, a training set $X = (\mathbf{x}_j : j = 1, \ldots, N)$ with desired output $T = (\mathbf{t}_j : j = 1, \ldots, N)$, a validation set V with desired output T_V, the regularization parameter λ.
output: An improved $M \times P$ matrix of input weights C^k and related output weights W^k

Set $k := 1$;
Set $C^k := C$, compute $W^k = \arg\min_W \{\|\Phi(XC^k)W^k - T\|_2^2 + \lambda\|W^k\|_2^2\}$ and $\hat{E}(C^k) := \|\Phi(VC^k)W^k - T_V\|_2^2 + \lambda\|W^k\|_2^2$;
repeat
  Set *Improved* := *false*;
  for $C' \in \mathcal{N}(C^k)$ **do**
    Compute $W' = \arg\min_W \{\|\Phi(XC')W - T\|_2^2 + \lambda\|W\|_2^2\}$ and $\hat{E}(C') := \|\Phi(VC') - T_V\|_2^2 + \lambda\|W'\|_2^2$;
    if $\hat{E}(C') < \hat{E}(C^k)$ **then**
      Set $C^{k+1} := C'$, $W^{k+1} := W'$;
      Set *Improved* := *true*;
      break;
  Set $k := k + 1$;
until not *Improved*;
return C^k, W^k;

Table 1. The UCI datasets and their characteristics. (a) size of regression datasets, and (b) size of classification datasets with class numbers and frequencies.

Dataset	Instances	Attributes
Abalone	4177	8
Mach. Cpu	209	6
Delta Ail.	7129	5
Housing	506	13
Concrete Compressive Strength	1030	9
Combined Cycle Power Plant	9568	4

(a)

Dataset	Instances	Attribs.	Classes num.	Classes freqs.
Iris	150	4	3	50 50 50
Diabetes	768	8	2	268 500
Landsat	4435	36	7	1072 479 961 415 470 0 1038
Statlog	2310	19	7	330 330 330 330 330 330 330
Transfusion	748	5	2	178 570
Wine	178	13	3	59 71 48

(*b*)

We can see the projection as a linear transformation as in the next equation,

$$X^{RP}_{N\times K} = X_{N\times P} C_{P\times K} \tag{8}$$

where $X_{N\times P}$ is the original set of N P-dimensional data, $C_{P\times K}$ the random entries matrix of dimensions $P \times K$ and $X^{RP}_{N\times K}$ are the mapped data onto the new K-dimensional space.

Some different random projections have been proposed in literature. For in-

stance entries of C can be randomly sampled from a gaussian distribution, but this is by no means a constraint of the method. Achlioptas (Achlioptas, 2001) has recently shown that the gaussian distribution can be replaced by a much simpler distribution, such as:

$$c_{ij} = \sqrt{3} \cdot \begin{cases} +1 & \text{with probability } 1/6 \\ 0 & \text{with probability } 2/3 \\ -1 & \text{with probability } 1/6 \end{cases} \tag{9}$$

that will be named sparse in this context.

In (Rubini et al, 2014) we performed three series of experiments where the input weights and biases are selected according to (i) a conventional strategy, where c_{ij} is sampled from a uniform random distribution in the interval $[-1, 1]$ or (ii) using random projection matrices belonging to two different types, respectively with elements c_{ij} gaussian distributed, with mean value zero and variance 1 or sparsely distributed according to eq.(9). Note that eq.(9) corresponds to restricting each hidden node to processing only a subset of the inputs.

Because the random projection approach turned out to be the winner in almost all cases, we retain this approach also for local search test.

4.1. Testing the local search

We performed a series of experiments in order to compare the relative strenghts of the PINV approach vs a local-search enhanced approach (LS). We randomly partitioned the datasets in a 2/3 training set, a 1/6 validation set and a 1/6 test set. The regularization parameter λ and the hidden layer size N_H have been calibrated through a 3-fold cross-validation scheme over the training set. We recall that the local search procedure at each neighbourhood exploration is driven by the descent of the validation error $\hat{E}$ defined by eq. (6). Referring to Procedure LS we note that, if C is a random projection, any neighbour $C' \in \mathcal{N}(C)$ generated by swapping rows is still a valid random projection. In order to limit more effectively the computation time, we anyway terminate the local search after a prefixed number of neighbours have been evaluated; we set such limit to 2000 after some preliminary experiment. The limit has been reached in less than 7% of the runs.

We tested the PINV approaches against LS ones. For a fair comparison we ran the two kinds of method as follows. We first ran the LS based learning starting from a random initial configuration of input weights, also counting the number S of neighbor solutions generated during the execution of the procedure. We then ran S executions of the PINV learning method, collecting error values obtained on the test set after each executon and averaging them over the S executions. This way both methods sample S times the landscape of the error function. Then we compute the test error for both models. In order to perform a statistically significative comparison the above procedure was repeated 15 times: we experimentally verified that in this way the sample size was large enough to obtain the statistical significance within an acceptable computational cost. We compared LS and PINV approaches with uniform, gaussian and sparse initializations. The main results are collected in Table 2 for regression datasets and Table 3 for classification datasets; average errors (either RMSE or misclassfication percentages) obtained on the test set are reported. The comparison is made evaluating the

Table 2. LS vs. PINV: regression datasets.

Dataset	**Type**	**Method**	RMSE			
			Avg	StD	N_H	λ
Abalone	Sparse	PINV	2.1624	0.0079	150	$7 \cdot 10^{-2}$
	Sparse	LS	**2.1595**	0.0051	150	$7 \cdot 10^{-2}$
Mach. Cpu	Sparse	PINV	58.312	1.1461	100	$3 \cdot 10^{-1}$
	Sparse	LS	58.695	1.5371	100	$3 \cdot 10^{-1}$
Delta Ail.	Unif.	PINV	$1.541 \cdot 10^{-4}$	$4 \cdot 10^{-7}$	100	$1 \cdot 10^{-3}$
	Unif	LS	$\mathbf{1.154 \cdot 10^{-4}}$	$5 \cdot 10^{-7}$	100	$1 \cdot 10^{-3}$
Housing	Unif.	PINV	2.8031	0.1449	150	$2 \cdot 10^{-2}$
	Unif.	LS	**2.7636**	0.0967	150	$2 \cdot 10^{-2}$
Co. Co. Strength	Gauss	PINV	6.268	0.31441	150	$9 \cdot 10^{-3}$
	Gauss	LS	6.3288	0.34324	150	$9 \cdot 10^{-3}$
CCPP	Gauss	PINV	2.0656	0.00093921	150	$7 \cdot 10^{-2}$
	Gauss	LS	2.0654	0.0006321	150	$7 \cdot 10^{-2}$

winning approach (either PINV or LS), then comparing it against its counterpart with the same initialization, requiring a confidence level of 95% unless otherwise stated.

From the table, we can make the following observations.

(i) In 6 out of 12 cases the best-performing LS approach is significantly better than the corresponding PINV approach with the same initialization. For the Abalone and Housing datasets, statistical significance is achieved with a 90% and 80% confidence level respectively. For the remaining 6 datasets LS achieves a lower error in 2 cases, whereas in 4 cases PINV delivers a lower error; anyway, because of the absence of statistical significance, these results have to be considered as draws.

(ii) In all but two cases (namely Delta Ailerons and Housing), the best performing LS method employs a random projection initialization.

The running times of both approaches were substantially equivalent.

5. Conclusions

The discussed results support our previous findings that initializing input weights by random projection matrices in a PINV based training can lead to significant performance improvements with respect to current practice, where uniformly distributed random weights are used. Taking a step further, we added a local search phase relying on a combinatorial-style neighbourhood definition that allows to locally explore the error surface along different directions with respect to the classical gradient-based algorithms. This allows to gain additional improvement over the basic approach with random projections. The local search framework is quite flexible, and several variants can be tried in order to improve performances. We recall that several types of neighbourhoods can be designed and implemented — swapping rows is just a possibility. Also, we adopted the first-improve neigh-

Table 3. LS vs. PINV: classification datasets.

Dataset	**Type**	**Method**	$\text{Err}_{\%}$			
			Avg	StD	N_H	λ
Iris	Sparse	PINV	0.8	1.6562	50	$6 \cdot 10^{-2}$
	Sparse	LS	**0.0**	0.0000	50	$6 \cdot 10^{-2}$
Diabetes	Sparse	PINV	22.552	0.7738	150	$5 \cdot 10^{-2}$
	Sparse	LS	**21.458**	1.0592	150	$5 \cdot 10^{-2}$
Landsat	Gauss.	PINV	9.9729	0.5463	500	$3 \cdot 10^{-1}$
	Gauss.	LS	10.3070	0.5004	500	$3 \cdot 10^{-1}$
Wine	Sparse	PINV	1.1494	2.1283	150	$9 \cdot 10^{-1}$
	Sparse	LS	1.6092	1.7807	150	$9 \cdot 10^{-1}$
Transfusion	Gauss	PINV	24.194	1.4933	100	$1 \cdot 10^{-5}$
	Gauss	LS	24.032	1.0647	100	$1 \cdot 10^{-5}$
Segment	Sparse	PINV	3.4978	0.35215	500	$9 \cdot 10^{-3}$
	Sparse	LS	**3.0996**	0.59284	500	$9 \cdot 10^{-3}$

Table 4. Learning times for local search

Dataset	N_H	Time (s.)
Abalone	150	8.455
Mach. Cpu	100	0.168
Delta Ail.	100	0.950
Housing	150	3.510
Concrete Compressive Strength	150	1.399
Combined Cycle Power Plant	150	2.149
Iris	50	0.040
Wine	150	0.545
Diabetes	150	1.271
Landsat	500	787.5
Transfusion	100	0.221
Segment	50	191.1

bourhood exploration strategy for reducing the computational load. For certain applications it could be viable to revert to the *best-improve* neighbourhood exploration.

We report in Table 4 the average learning times required by the tests with the local search procedure. The multistart PINV approach requires substantially equivalent times; this is a consequence of how the tests were designed, so that in order to get a fair comparison both procedures perform the same number of pseudoinversions. Reducing the number of trials (and time) for PINV makes the error figures grow significantly. We did not attempt a detailed comparison with computation times of a classical backpropagation procedure, since we rest on the results of (Huang et al, 2006; Deng et al, 2009; Martinez-Martinez et al, 2011); anyway in a number of tests performed on the largest datasets (for instance Landsat) with a traditional backpropagation procedure the learning time almost doubled while testing errors raised significantly.

Adding some search device on top of a PINV based approach is an idea that has been tested for example in (Zhu et al, 2005), where the so-called differential evolution technique is coupled with ELM. A quick comparison on the common datasets reported in (Zhu et al, 2005) shows that the local search approach is simpler and yet competitive, delivering comparable performances.

Acknowledgements. The activity has been partially carried on in the context of the Visiting Professor Program of the Gruppo Nazionale per il Calcolo Scientifico (GNCS) of the Italian Istituto Nazionale di Alta Matematica (INdAM).

References

Ajorloo H, Manzuri-Shalmani M T, Lakdashti A (2007) Restoration of damaged slices in images using matrix pseudo inversion. 22nd International Symposium on Computer and Information Sciences.

Aarts EHL, Lenstra JK (2003) Local search in combinatorial optimization. Princeton University Press.

Achlioptas D (2001) Database-friendly random projections. Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 274–281.

Asuncion A and Newman DJ (2007) UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Badeva V, Morosov V (1991) Problemes incorrectements posès, thèorie et applications. Masson, Paris.

Cancelliere R, Deluca R, Gai M, Gallinari P, Rubini L (2015) An analysis of numerical issues in neural training based on pseudoinversion. Computational and Applied Mathematics, pp. 1–11

Wanyu D, Zheng Q, Chen L (2009) Regularised Extreme Learning Machine. Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, pp. 389–395.

Fuhry M, Reichel L (2012) A new Tikhonov regularization method. Numerical Algorithms 59, pp. 433–445.

Gallinari P, Cibas T (1999) Practical complexity control in multilayer perceptrons. Signal Processing 74, pp. 29–46.

Girosi F, Jones M, Poggio T (1995) Regularization theory and neural networks architectures. Neural Computation 7 (2), pp. 219–269.

Haykin S (1999) Neural Networks, a comprehensive foundation. Ed. Prentice Hall, U.S.A.

Halawa K (2011) A method to improve the performance of multilayer perceptron by utilizing various activation functions in the last hidden layer and the least squares method. Neural Processing letters, vol. 34, pp. 293–303.

Huang GB, Zhu QY, Siew CK (2006) Extreme Learning Machine: Theory and applications. Neurocomputing, vol. 70, pp. 489–501.

Johnson WB, Lindenstrauss J (1984) Extensions of Lipshitz mapping into Hilbert space. Contemporary Mathematics, vol. 26, pp. 189–206.

Kohno K, Kawamoto M, Inouye Y (2010) A matrix pseudoinversion lemma and its application to block-based adaptive blind deconvolution for MIMO systems. IEEE Transactions on Circuits and Systems.

Martinez-Martinez J, Escandell-Montero P, Soria-Olivas E, Martn-Guerrero J, Magdalena-Benedito R, Gomez-Sanchis J (2011) Regularized extreme learning machine for regression problems. Neurocomputing, vol. 74, pp. 3716–3721.

Nguyen TD, Pham HTB, Dang VH (2010) An efficient Pseudo Inverse matrix-based solution for secure auditing. IEEE International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future.

Poggio T, Girosi F (1990) Regularization algorithms that are equivalent to multilayer networks. Science 247, pp. 978-982.

Rubini L, Cancelliere R, Gallinari P, Grosso A, Raiti A (2014) Computational Experience with Pseudoinversion-Based Training of Neural Networks Using Random Projection Matrices. Lecture Notes in Computer Science Volume 8722, pp. 236-245, 2014. Proceedings of the 16th International Conference on Artificial Intelligence: Methodology, Systems, Applications, AIMSA 2014.

Rumelhart DE, Hinton GE, Williams RJ (1996) Learning internal representations by error propagation. Parallel distributed processing: explorations in the microstructure of cognition, vol. 1, pp. 318–362. MIT Press Cambridge, MA, USA.

Tikhonov AN, Arsenin VY (1977) Solutions of Ill-Posed Problems. Winston, Washington DC.

Tikhonov AN (1963) Solution of incorrectly formulated problems and the regularization method. Soviet Mathematics 4, pp. 1035–1038.

Wang X, (2013) Special Issue on Extreme Learning Machine with Uncertainty, Editorial International Journal of Uncertainty Fuzziness and Knowledge-Based Systems, Vol. 21, pp. v-vi.

Wang XZ, Wang D, Huang GB (2012) Special Issue on Extreme Learning Machines. Ed. Soft Computing, vol. 16(9), pp. 1461–1463.

Zhu QY, Qin AK, Suganthan PN, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recognition, vol. 38, pp. 1759–1763.

Correspondence and offprint requests to: Luca Rubini, University of Turin, Department of Computer Sciences, Turin, Italy. Email: luca.rubini@unito.it