Pseudoinversion for neural training: tuning the regularisation parameter

Noname manuscript No.
(will be inserted by the editor)

Random projections and matrix pseudoinversion for data neural processing

Rossella Cancelliere Roberta de Luca MarioGai Patrick Gallinari Thierry Artières

Received: date / Accepted: date

Abstract Recently some novel strategies have been proposed for training of Single Layer Feedforward Networks, 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, to minimise the mean square error on training set. Such non-iterative strategies are appealing since they allow fast learning, but many choices have to be made using such approach, mainly concerning the initialization of input weights and the procedure for determining output weights. One aim of this study is to investigate the performance variability with weight choice and number of hidden neurons. We explore the use of various random projections for convenient setting of the input weights. Then we evidence that the method application may require some care to achieve good results, mainly concerning the procedure used for matrix pseudoinversion. We show that this key step suffers from numerical problems related to matrix invertibility, and we propose a heuristic procedure for bringing more robustness to the method. We report results on a difficult astronomical regression problem of chromaticity diagnosis to illustrate the various points under study.

Keywords random projections · weights initialization · unsupervised learning · pseudoinverse matrix

R. Cancelliere
Department of Computer Sciences, Universitá di Torino, Turin, Italy
Tel.: +39 011 6706737
Fax: +39 011 751603
E-mail: rossella.cancelliere@unito.it

M. Gai
Astronomical Observatory of Torino, National Institute of Astrophysics, Turin, Italy

T. Artières, P. Gallinari
Laboratory of Computer Sciences, LIP6, Université Pierre et Marie Curie, Paris, France

1 Introduction

In the past two decades, single hidden layer feedforward neural networks (SLFNs) have been one of the most important subject of study and discussion among neural researchers [1–3]. Their main applications are in the field of supervised learning tasks, where adjustment of parameters is required in order to approximate a functional relationship between given inputs and outputs. Typically, this relationship is not known analytically and is defined by the set of examples in the training set, while performance is evaluated on the deviations between network outputs and targets in test set.

Over a few decades, methods based on gradient descent have mainly been used, although defining different learning algorithms; among them there is the large family of techniques based on backpropagation, widely studied in its variations [4]. The start-up of these techniques assigns random values to the weights connecting input, hidden and output nodes, these weights are then iteratively adjusted.

Anyway, gradient descent-based learning methods are typically slow, frequently require small learning steps, and are subject to convergence to local minima. Therefore, many iterations may be required by such algorithms in order to achieve an adequate learning performance, and many trials are required for avoiding poor local minima.

This computational cost of gradient based techniques is even higher for deep architectures (e.g. deep neural networks with multiple hidden layers), proposed in the recent years [5] for learning the kind of complicated functions which might represent high-level abstractions (e.g. in vision [6, 7], natural language processing (NLP) [8,9], and other typical AI-level tasks). Hence, it is subject of debate today which could be the more effective inizialization for weights in such deep networks; the most common strategy is that of using unsupervised learning at each stage of a deep network, as it was recently put forward by Hinton et al. [10] followed by a second and final phase of fine-tuning of all the parameters of the network using backpropagation and gradient descent on a global supervised cost function.

The reduction of computational efforts for NN learning is therefore of great importance when dealing with deep architectures, because of the presence of multiple hidden layers whose weights have to be adequately trained, and may become imperative in the case of large datasets, as required for instance in text, image, speech and handwritten processing. As key aspects addressed by the present paper we propose therefore some possible alternatives to the standard weight initialization, on one side, and to iterative methods for solution, on the other.

Some non iterative procedures based on the evaluation of generalized pseudoinverse matrices were proposed recently as novel learning algorithms for SLFNs, among them a method to improve performance of multilayer perceptron by Halawa [11] and the Extreme Learning Machine (ELM) [12]. In ELM, the input weights (linking input and hidden layers) and hidden biases are randomly chosen, and the output weights (linking hidden and output layers) are analytically determined by the Moore-Penrose (MP) generalized inverse. This theoretically appealing method has many interesting features among which its non iterative feature, but it needs some care in the determination of the pseudoinverse of the hidden to output layer weight matrix, because of the eventual presence of singular or almost singular matrices.

The aim of this paper is a deep investigation on a few major issues of pseudoinversion-based learning for SLFN models. On one side, we propose a new approach through the application of a preprocessing to the input data by randomly projecting them in appropriately chosen spaces, to simplify the following step of the neural analysis. The theoretical rationale for this approach can be found in many recent studies, showing random projections as a powerful method for dimensionality treatment [13–15], so that they appear to be a potentially useful tool for weight initialization by early extraction of relevant features in the input data.

Besides, we investigate a few ways for determining output weights and the associated numerical problems. The most commonly used method involves direct matrix inversion, or evaluation of the pseudoinverse matrix, by means of Singular Value Decomposition (SVD), but this technique needs to be deeply analyzed, because of eventual numerical problems arising when dealing with nearly singular matrices.

We first recall the main ideas on SLFN learning and on ELM in section 2, and we describe fundamentals results on random projection in section 3. Next we detail in section 4 our proposition for the two main issues in ELM-like learning, namely how to initialize hidden weights, and how to compute output weights. Finally in section 5 we report results on a difficult astronomical problem of chromaticity diagnosis, related to the relevance of various weights initialization strategies and of pseudoinverse evaluation.

2 How to train weights by pseudoinversion

In this section we first recall basic notations regarding SLFN then we explain the motivation and principle of ELM.

2.1 Single Layer Feedforward Neural Networks

A standard SLFN with P inputs, M hidden neurons, Q output neurons, non-linear activation function ϕ on the hidden layer and linear activation function otherwise, computes an output vector $o = (o_1, ..., o_Q)$ from an input vector $x = (x_1, ..., x_P)$ according to:

$$o_k = b_k^O + \sum_{i=1}^{M} w_{k,i}\phi(\mathbf{c}_i \cdot \mathbf{x} + b_i^H) \qquad k = 1, \cdots, Q \tag{1}$$

where $\mathbf{c}_{i,j}$ denote weights from input to hidden neurons, $w_{k,i}$ denote weights from hidden to output neurons, b_i^O and b_i^H denote biases for Output and Hidden neurons. The typical architecture of a SLFN is shown in Fig.1.

Considering a dataset of N distinct training samples of (input, output) pairs $(\mathbf{x}_j, \mathbf{t}_j)$, where $\mathbf{x}_j \in \Re^P$ and $\mathbf{t}_j \in \Re^Q$, learning a SLFN aims at producing the desired output $\mathbf{t}_j$ when $\mathbf{x}_j$ is presented as input. Training aims at determining weights w, c, and biases b such that:

$$\forall k = 1, \cdots, Q, \quad \forall j = 1, \cdots, N, \qquad t_j^k = b_k^O + \sum_{i=1}^{M} w_{ki}\phi(\mathbf{c}_i \cdot \mathbf{x}_j + b_i) \tag{2}$$

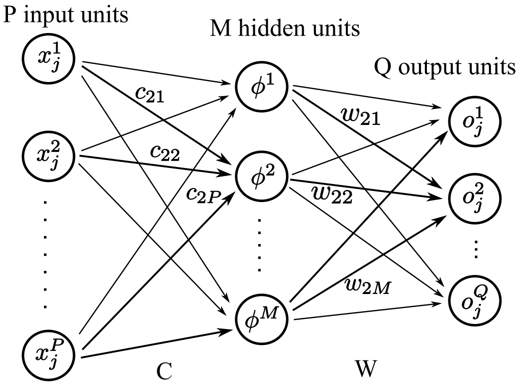

Fig. 1 A Single Layer Feedforward Neural Network

The above N equations can be written compactly in matrix form as

$$\mathbf{T} = \mathbf{H}\mathbf{w}, \tag{3}$$

where:

$$\mathbf{w} = \begin{vmatrix} \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_{_M}^T \end{vmatrix}_{M \times Q}, \quad \mathbf{T} = \begin{vmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{vmatrix}_{N \times Q},$$

$$\mathbf{H} = \begin{vmatrix} \phi\left(\mathbf{c}_1 \cdot \mathbf{x}_1 + b_1\right) & \cdots & \phi\left(\mathbf{c}_M \cdot \mathbf{x}_1 + b_M\right) \\ \vdots & \ddots & \vdots \\ \phi\left(\mathbf{c}_1 \cdot \mathbf{x}_N + b_1\right) & \cdots & \phi\left(\mathbf{c}_M \cdot \mathbf{x}_N + b_M\right) \end{vmatrix}_{N \times M} \tag{4}$$

$\mathbf{H}$ is the hidden layer output matrix of the neural network; the i-th column of $\mathbf{H}$ is the i-th hidden node output with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_N$. Traditionally in order to train a SLFN, a least square solution is searched, determining $\mathbf{c}, b, w$ such that the following cost functional is minimized:

$$E_D = \sum_{j=1}^{N} \sum_{k=1}^{Q} \left(t_j^k - \sum_{i=1}^{M} w_{ki} \phi(\mathbf{c}_i \cdot \mathbf{x}_j + b_i) \right)^2 = ||\mathbf{H}\mathbf{w} - \mathbf{T}||^2. \tag{5}$$

As stated above, gradient-based learning algorithms, that require to adjust input weights and hidden layer biases, are generally used to search the minimum of $||\mathbf{H}\mathbf{w} - \mathbf{T}||^2$.

2.2 The Extreme Learning Machine algorithm

Huang and Babri [16] evidenced that a SLFN performing function approximation with a large class of non-linear activation functions on a finite training set can exactly learn N distinct patterns using N hidden nodes. Huang et al. [12] also proved that, if $M \leq N$ hidden nodes are used, the discrepancy between actual outputs and targets can be reduced, with probability one, below a value ϵ positive and arbitrarily small.

They also demonstrated that, contrarily to what happens in traditional training algorithms, input weights and hidden layer biases of a SLFN do not need to be adjusted, but they can be randomly assigned. After the input weights and the hidden layer biases are chosen randomly, SLFNs can be simply considered as a linear system, and the output weights (linking the hidden layer to the output layer) can be analytically determined through simple generalized inverse operation on the hidden layer output matrix, if the activation functions of hidden neurons are infinitely differentiable.

These ideas gave rise to the ELM algorithm, whose main idea, as rigorously proved in [12], builds on a random choice of input weights and hidden layer biases of a SLFN, under the only assumption that the activation functions are infinitely differentiable. Hence the hidden layer output matrix $\mathbf{H}$ in (4) can actually remain unchanged, retaining the initial values associated to the assigned random values for input weights $\mathbf{c}_i$ and hidden layer biases b_i. The SLFN training reduces then to an easy step, i.e. finding the solution $\mathbf{w}$ that minimizes the cost functional (5).

In most cases of interest, the number of hidden nodes is much lower than the number of distinct training samples, i.e. $M << N$, so that $\mathbf{H}$ is a non-square matrix; in this case, as is demonstrated in [17], we search for the *smallest norm least-squares solution* w^* of the linear system:

$$\mathbf{w}^* = \mathbf{H}^+\mathbf{T}, \tag{6}$$

where $\mathbf{H}^+$ is the Moore-Penrose generalized inverse (or pseudoinverse) of matrix $\mathbf{H}$.

The solution $\mathbf{w}^*$ has some important properties:

- It is one of the least-squares solutions of the general linear system (3), hence it reaches the smallest training error.
- It has the smallest norm among all least-squares solutions
- It is unique.

The ELM algorithm is appealing since it can be easily implemented, it runs extremely fast because it works in a single pass, and not only it tends to reach the smallest training error, but also it provides the *smallest norm* solution. The last point is quite relevant, since, according to Bartlett's theory [18] on the generalization performance of feedforward neural networks, the smaller is the norm of weights, the better generalization performance is usually achieved by the network; therefore, this learning algorithm is expected to reach good generalization performance.

3 Main ideas concerning random projections

Random projections are used to project the original d-dimensional data into a k-dimensional subspace, using a random $k \times d$ matrix R whose columns have unit norm. Using matrix notation, if $X_{d \times N}$ is the original set of N d-dimensional observations,

$$X^{RP}_{k\times N} = R_{k\times d}X_{d\times N} \tag{7}$$

is the projection of the data onto the new k-dimensional subspace. Random projection is very simple from a computational standpoint: the process of forming the random matrix R and projecting the data matrix X into k dimensions has complexity of order $\mathrm{O}(dkN)$; moreover, if the data matrix X is sparse with about g nonzero entries per column, the complexity is of order $\mathrm{O}(gkN)$.

Random Projections are usually employed for dimensionality reduction. To this end, the key idea of random mapping arises from the Johnson-Lindenstrauss lemma [19]: if a set of points in a vector space is projected onto a randomly selected subspace of suitable dimension, then the original distances between the points are approximately preserved in the new space. For a simple proof of this result, see [20].

In our work we rather use Random Projections for dimensionality expansion. In such a setting, a linear mapping such as (7) can cause significant distortions in the data set if R is not orthogonal. However, and unfortunately, orthogonalizing R is computationally expensive. Instead, we can rely on a result presented by Hecht-Nielsen [21]: in a high-dimensional space, there exists a much larger number of *almost orthogonal* than strictly orthogonal directions. Therefore, vectors having random directions might be sufficiently close to orthogonality; equivalently, R^TR would approximate an identity matrix.

Few random projections have been proposed. The elements r_{ij} of R are often gaussian distributed, but this is by no means a constraint of the method. For instance, Achlioptas [22] has recently shown that the gaussian distribution can be replaced by a much simpler distribution, such as:

$$r_{ij} = \sqrt{3}\cdot\begin{cases} +1 & \text{with probability } 1/6 \\ 0 & \text{with probability } 2/3 \\ -1 & \text{with probability } 1/6 \end{cases} \tag{8}$$

or

$$r_{ij} = \begin{cases} +1 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/2 \end{cases} \tag{9}$$

Actually, a large variety of zero mean, unit variance distributions of elements r_{ij} result in a mapping that still satisfies the Johnson-Lindenstrauss lemma.

Achlioptas result means further computational savings in database applications, as the computations can be performed using integer arithmetic. Bingham and Mannila [23], in their experiments, showed that Achlioptas' theoretical result has practical significance, using both gaussian distributed random matrices and matrices like (8). The latter are named sparse in that context; for simplicity, in the following we retain the same term, although usually matrices are defined sparse when they have a much larger fraction of zero elements.

4 Discussion

In this section we wish to analyze in more details interesting and often critical aspects for learning SLFN with a ELM-like procedure. Following the ELM principle we aim at simplifying the learning by fixing input to hidden weights through random projection.

A first difficulty arises because random projections are mainly used for linearly separable tasks although many real world problems are not linearly separable. Neural networks feature among the tools available to deal with the latter class of problems, so we propose to join these techniques using a random projection to initialize the input weights, while the subsequent processing by hidden nodes nonlinear activation function accounts for the non-linearity of the problem.

As shown in [24] the use of sigmoidal functions, especially in deep neural networks, has recently been subject to debate because they seem to be more easily driven towards saturation because of their non-zero mean value; hyperbolic tangent seems to be less sensible to this problem, therefore we utilize this one. In the perspective of mitigating saturation issues, we also adopt a normalisation factor $1/\sqrt{M}$, where M is the number of hidden nodes. This reduces the typical size of the input values, thus exploiting the almost linear central part of the activation function, that for the hyperbolic tangent is centred on zero. The presence of this factor, when exploring the performance as a function of an increasing number of nodes, contributes to decrease neuron outputs.

So in our work the sampling intervals of all used distributions vary during training as a function of the number of hidden neurons, while the ELM technique usually maintains them fixed.

Hereafter we address the main critical issue which concerns learning of hidden to output layers' weights by pseudoinversion.

4.1 Pseudoinverse computation

Probably the main difficulty, when using an ELM strategy, concerns the computation of hidden to output layers' weights by pseudoinversion, because the evaluation of the Moore-Penrose psudoinverse matrix requires some caution, since singular and *almost singular* matrices may be found.

Several methods are indeed available, e.g. in [25], to evaluate the Moore-Penrose pseudoinverse matrix: in particular, if $\mathbf{H}$ has maximum rank the orthogonal projection method can be used when $\mathbf{H}^T\mathbf{H}$ is nonsingular, so that

$$\mathbf{H}^+ = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T \tag{10}$$

but it is known that this method is affected by severe limitations when the matrix $\mathbf{H}^T\mathbf{H}$ is almost singular, i.e. its determinant is almost zero. In this case, the computation of the inverse is highly unstable, and consequently the product $\mathbf{H}^+\mathbf{H}$ is potentially much different from the unit matrix. A possible solution consists in the addition of a regularization term to the cost functional (5)

$$E = E_D + \lambda E_W \tag{11}$$

where λ is the regularization coefficient that controls the relative strength of the data-dependent error E_D and the regularization term E_W. For L_2 regularization the term E_W usually takes the form

$$E_W = \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{12}$$

so that the cost functional (5) becomes:

$$E = E_D + \lambda E_W = \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{Q}\left(\left(t_j^k - \sum_{i=1}^{M} w_{ki}\phi(\mathbf{c}_i \cdot \mathbf{x}_j + b_i)\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{M}|w_{ki}|^2\right). \tag{13}$$

With this approach, the solution (6) becomes:

$$\mathbf{w}^* = (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T. \tag{14}$$

A different approach consists in evaluating the *singular value decomposition* (SVD) of $\mathbf{H}$:

$$\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{15}$$

where $\mathbf{U} \in \Re^{N\times N}, \mathbf{V} \in \Re^{M\times M}$ are unitary matrices and $\mathbf{S} \in \Re^{N\times M}$ has elements $\sigma_{ij} = 0$ for $i \neq j$ and $\sigma_{ii} = \sigma_i$ for $i = j$, with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$, $p = \min\ \{N, M\}$ (see for instance [26]). This method is preferable because it avoids the matrix inversion step required in (10) and (14). Moreover, because no regularization term is needed, it is possible to achieve a more effective minimization of the data dependent error.

Using this strategy, the pseudoinverse matrix $\mathbf{H}^+$ is defined as (see [27] for more details):

$$\mathbf{H}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^\mathbf{T} \tag{16}$$

where $\mathbf{\Sigma}^+ \in \Re^{\mathbf{M}\times\mathbf{N}}$ has elements $\sigma_{ij}^+ = 0$ for $i \neq j$ and $\sigma_{ii}^+ = 1/\sigma_i$ for $i = j$; the elements σ_i are the singular values of the decomposition.

If $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > \sigma_{k+1} = \cdots = \sigma_p = 0$, the rank of matrix $\mathbf{H}$ is k and the inverses of the $p - k$ zero elements are replaced by zeros.

Despite its computation advantage compared to direct matrix inversion, this method may exhibit some problem since almost singular matrices may have very small singular values σ_i, whose numerical inversion may cause instability in the algorithm. We will therefore propose a careful monitoring of such singular values, by directly replacing their inverses by zeros. As we will see in subsection 5.2, this cut-off threshold acts as a regularisation that stabilises the algorithm nevertheless producing a further approximation of the matrix of output weights.

5 Experimental results on chromaticity diagnosis

We report here a number of experimental results on a complex supervised regression learning problem in the astronomical field. We first describe the problem and detail the datasets used. Then we investigate the influence of the choice of a random projection and of the procedure used for determining the optimal set of weights in an ELM-like approach. We further provide insights on numerical aspects that may lead to poor performance and report experimental evidence that our procedure for computing the pseudoinverse does overcome these problems and indeed bring robustness to the method. Lastly we show that the random nature of the method is not a severe drawback. Although the method is sensitive to the sampled random projection, we show that only a few tries of a random projection are necessary to be sure to get a good projection.

5.1 The astronomical problem

Astrometry, i.e. the precise determination of stellar positions, distance and motion, is largely based on imaging instrumentation fed by telescopes operating in the visible range. The measured image profile of a star however depends on its spectral type, i.e. the emitted light distribution as a function of frequency, so that its measured position appears affected by an error called chromaticity. Chromaticity was identified for the first time [28], in the data analysis of the space mission *Hipparcos* of the European Space Agency (ESA).

The chromaticity issue becomes even more important for the current ESA mission Gaia [29] for global astrometry, aiming at much higher precision. The detection and correction of chromaticity in different conditions has been addressed in recent years [30,31]; to this purpose, a single-hidden layer feedforward neural network (SLFN), trained by a classical BP algorithm, was used to solve this diagnosis task.

Each image is first reduced to a one-dimensional signal $s(x)$ along the single measurement direction of Gaia, by integration on the orthogonal direction, also to reduce the data volume and telemetry rate.

With respect to our previous studies, we also adopt as input to the neural network processing a more convenient set of statistical moments M_k for image encoding:

$$M_k = \sum_n \left(x_n - x_{COG}\right)^k \cdot s\left(x_n\right) \cdot s_A\left(x_n\right) \ , \tag{17}$$

where $s\left(x_n\right)$ is the above mentioned signal, $s_A\left(x_n\right)$ is the signal from an ideal instrument, and x_{COG} is the signal barycenter:

$$x_{COG} = \frac{\sum_n x_n \cdot s\left(x_n\right)}{\sum_n s\left(x_n\right)} \, . \tag{18}$$

Chromaticity, already mentioned, can be more conveniently defined using the concept of blue (effective temperature $T = 30,000\,K$) and red ($T = 3,000\,K$) stars, in particular it is the barycenter displacement between them, and it is the neural network target. Correct diagnostics allows effective correction of this kind of error, therefore a good approximation by the neural network results in a small *residual chromaticity* after neural processing.

The 11 neural network inputs are the red barycenter and the moments of red and blue stars from Eq. 17 ($k = 1, \cdots, 5$), computed for each instance.

We have a total of 13000 instances, built according to the above prescriptions, and split in a training set of 10000 instances and a test set with 3000 instances. The data set was provided by the Astronomical Observatory of Turin of the Italian National Institute for Astrophysics.

5.2 Experimental investigation

We investigate neural networks with the architecture shown in Fig.1. They have 11 input neurons corresponding to the 11 statistical moments describing each image, 1 output neuron and a number of hidden neurons varying from 50 to 600; as already discussed the hidden neuron activation function is the hyperbolic tangent. All the simulations for the BP and ELM algorithms are carried out in Matlab 7.3 environment.

In the following we first investigate the influence of random initialization on the final performance in a ELM-like architecture, where only output weights are learned. Then, we compare the various strategies we detailed before for determining the output weights. In passing, we put in evidence a numerical problem encountered when dealing with almost singular value decomposition and propose a solution to bring some robustness to the method. Finally, we provide some hints on how a random-based ELM-like learning procedure could significantly outperform the well established gradient backpropagation learning procedure.

Yet, as a first result we evaluated the performance reached by a standard SLFN neural network (with hyperbolic tangent as hidden neuron activation function) fully trained using the backpropagation algorithm. This will the reference for other results all along the experimental section. We initialized NN weights randomly according to a uniform distribution, looking for the minimum value of RMSE when the number of hidden nodes is gradually increased from 10 to 200. Because we used a sufficiently large dataset for training and no overfitting arose, best results are gained without any regularization term, and learning consists then in minimizing the cost functional defined by eq. (5). With such a setting the minimum RMSE obtained was 3.81, with 90 hidden neurons.

We first investigate the importance of the random projection for setting the weights from input to hidden layer. We applied a ELM-like strategy, initializing the input weights by two different kinds of random projection matrices: the elements r_{ij} are respectively i) gaussian distributed, with mean value zero and variance $1/\sqrt{M}$ and ii) sparsely distributed according to eq.(8), but with multiplicative coefficient $1/\sqrt{M}$. Note that eq.(8) corresponds to restricting each hidden node to processing only a subset of the inputs. We compare these results with a more conventional initialization derived from a uniform random distribution in the interval $(-1/\sqrt{M}, 1/\sqrt{M})$. Lastly, in order to make a comparison with the original ELM method, we also investigate the uniform initialization in the interval $(-1, 1)$ with sigmoidal activation function (column ELM-like).

In all cases we determine output weights using the algorithm ELM, analyzing and comparing the different methods of evaluation of the pseudo-inverse matrix presented in section 2. For each initial setup, the number of hidden nodes has been gradually increased adding one node each time and average results are computed over 10 simulation trials for each selected size of SLFN. Finally, the number of hidden nodes related to the best performance (mean and standard deviation over the 10 trials) is reported, as well as the optimal performance, i.e. the minimum value; the corresponding number of hidden neurons is specified in brackets. In all cases the pseudoinverse is evaluated through singular value decomposition. Results are reported in Table 1.

It should be noted that different points of view may be adopted in interpreting the results of Table 1. If we consider that the learning scheme consists of drawing

Inizialization:	Sparse	Gaussian	Uniform	ELM-like
Mean RMSE	2.17 (546)	3.13 (367)	1.77 (573)	8.52 (88)
St. Dev.	0.97 (546)	1.87 (367)	0.73 (573)	2.54 (88)
Min. RMSE	0.80 (342)	0.91 (307)	0.81 (300)	1.51 (448)

Table 1 Comparison of performance on the test set; the pseudoinverse matrix is evaluated by singular value decomposition.

a single random projection then evaluating the output weights, then mean RMSE (first line) can be viewed as the average result of the method for a particular kind of random projection. The lower it is the better the method performs. In such a case a high variance means a low guarantee on the result gained when performing one try only.

Alternatively, we may consider that the learning scheme consists of attempting many random projections, evaluating the output weights for each, and keeping the best model. We adopt this second point of view: what we are more interested in is the minimum RMSE (third line) achieved with a specific random projection over the set of trials. Also, we are interested in the variance of this minimum with respect to the number of random projection drawn, and we will discuss this latter point in section 5.3.

Looking at the results in Table 1 we note first that the best performance, in terms of minimum RMSE, is achieved by both Sparse and Uniform initialization methods, while there is a significant worsening for the ELM-like method. Besides, the variance of the results is significantly less for these two methods, meaning a better behaviour with respect to over-fitting.

Finally we note that the number of hidden units required to reach the best performance is almost independent of the method; both Sparse and Uniform methods appear to be able to reach best performance and although the Sparse method requires 342 hidden units, only 2/3 of the weights in a sparse RP are non zero, yielding a lower number of effective parameters.

Next we investigated the procedure for determining output weights (results in Table 2). Using the uniform initialization, we compared the performance obtained by evaluating the pseudoinverse matrix by Singular Value Decomposition with the two cases of orthogonal projection methods described by eq.(10) (non regularized column) and eq.(14) (regularized column). It is immediate to see that singular value decomposition significantly outperforms the other two methods, it provides much better RMSE values than either simple or regularized Moore-Penrose pseudoinversion.

The SVD method is more accurate, also if it requires a larger number of hidden units to reach its optimal performance, while the two other methods appear to be unable to take advantage of a larger hidden layer.

Fig.2 and Fig.3 give more insights on the evolution of the performance with the number of hidden units. The figures show the mean RMSE (Fig.2) and the minimum RMSE (Fig.3) trends vs. number of hidden nodes for the three different kinds of random projection matrices whose best values are reported in columns 1-3 of Table 1 (note that SVD is used in every case for determining output weights). It is interesting to note that all trends show a peak for a number of hidden neu-

Pseudoinv. method:	SVD	Regularized	Non regularized
Mean RMSE	1.77 (573)	7.69 (90)	9.43 (60)
St. Dev.	0.73 (573)	2.28 (90)	3.91 (60)
Min. RMSE	0.81 (300)	4.83 (90)	6.45 (60)

Table 2 Comparison of performance on the test set with uniform random inizialization of weights.

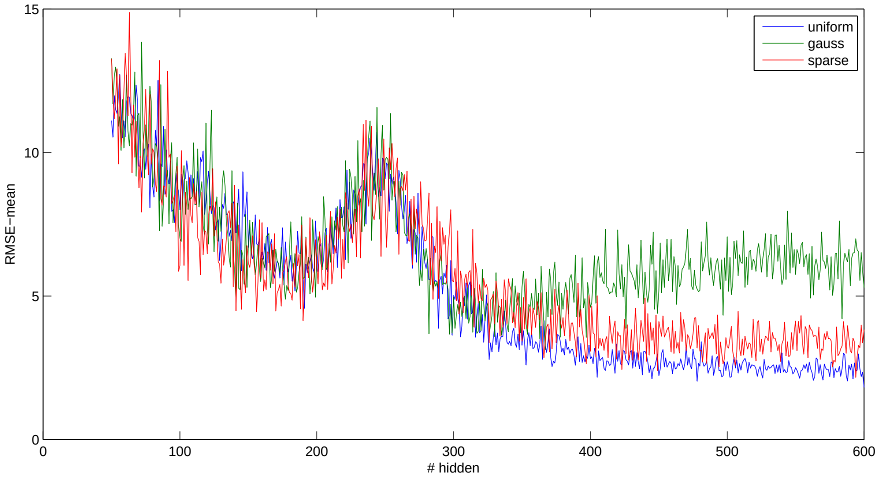

Fig. 2 Mean performance vs. projection space size

rons approximately equal to 250. For a larger number of hidden units, the RMSE stabilizes for both uniform and sparse scheme, while it starts increasing with the gaussian initialization.

This error peak is related to the presence of singular values close to zero in matrix **S**, as discussed in section 4. This correlation between the approaching of singular values to zero and the growth of RMSE is put in evidence by plotting the ratio of the minimum singular value and the Matlab default threshold (i.e. the threshold below which singular values are replaced by zeros in matrix $\mathbf{\Sigma}^+$) as shown in Fig.3, where logarithmic units are used (light blue line). This fact has potentially dramatic effects on performance, and must therefore be taken into account when SVD is used to evaluate $\mathbf{H}^+$.

One way to treat this problem is to increase the threshold. The effect of such a cut-off is a sort of regularisation that stabilises the algorithm, introducing a further approximation of the matrix of output weights. This is the strategy we used.

We selected as new threshold the mean value of the smallest singular values obtained over the 10 trials when using a hidden layer with 180 hidden neurons, because this is approximately the dimension of hidden neuron space when RMSE starts to increase.

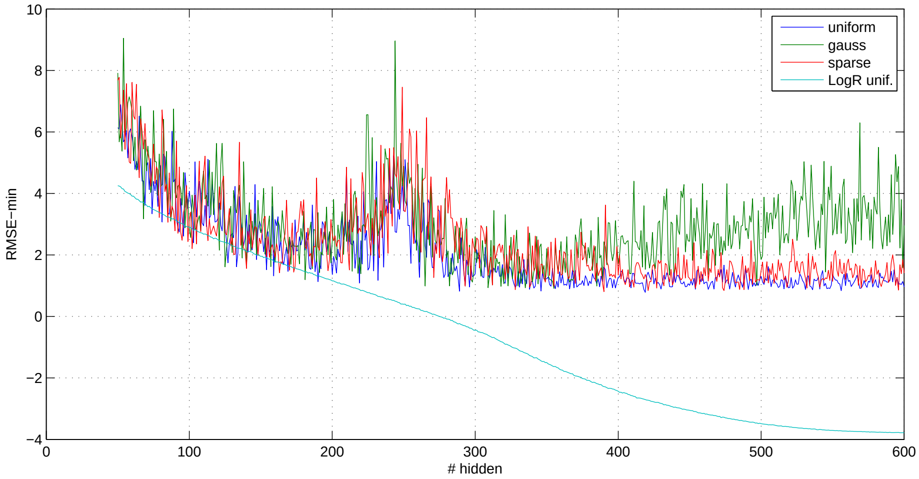

Fig. 3 Minimum performance vs. projection space size

This simple threshold choice provides a significant improvement in the method robustness, as can be seen from the relevant reduction of error peaks; this is obtained at the expense of a slight increase of RMSE values, as shown in Fig.4, where the optimal RMSE trends are shown for the three initialization cases.

Threshold tuning is therefore an interesting option since the limiting RMSE performance (which is respectively 1.15, 1.21 and 1.20 for uniform, gaussian and sparse initializations) remains quite attractive with respect to the classical backpropagation algorithm.

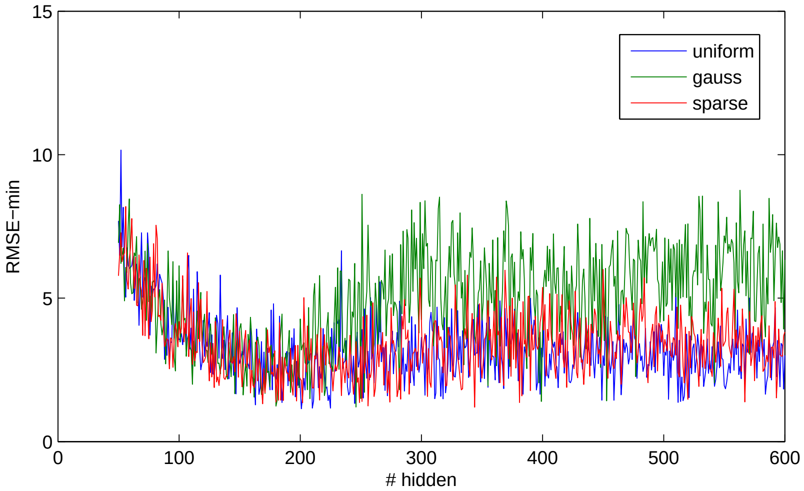

Fig. 4 Minimum performance vs. projection space size with threshold tuning

5.3 Final considerations

The above results show clear advantage of the ELM strategy over standard training for SLFN. We provide some hints for a better understanding of such a phenomenon.

When stressing the advantage of our method in previous sections, we mentioned two viewpoints to interpret the results. Our viewpoint considers that actual model learning includes many drawings of a random projection, determining the corresponding output weights, and keeping the one that yields the best results. This is the reason why we reported in Fig.3 the best results achieved over several issues of random projection (followed by output weight computation). Such a training phase, based on multiple random choices of weights, can be supposed to explore more extensively the parameter space, with respect to the trajectories followed by backpropagation algorithm, that develop continuously from a single random starting point. Yet this approach is reasonable provided that the number of trials, i.e. the number of times one has to draw a random projection then evaluate the output weights, remains limited.

We verified the presence in the parameter space of many *good solutions* that can be reached also with a relatively small number of initial random choices. We first performed a set of 1000 different training trials. Then for a given subset size s, ranging from 10 to 200, we randomly built 100 subsets each containing s trials among the initial 1000 cases, in order to achieve statistical information. We then selected in each subset the minimum value of test error, and evaluated mean values and standard deviations of the distributions of such minimums over the 100 subset instances. The test results are summarised in Fig.5, showing the best performance as a function of the number of trials only for the sparse case, because of the curve similarity for the other initialization schemes.

As expected, the mean value of the error distribution decreases with increasing number of trials; also, the range of variability decreases quickly, meaning that a limited number of trials is indeed required to attain good results, i.e. close to the best case. In our experiments, 100 trials are enough to reach an almost optimal performance, while 10 experiments already provide very good average results. This clearly demonstrates the method reliability, since good results are obtained even with a limited number of trials.

6 Conclusions

In this paper some strategies are investigated for training single layer feedforward networks using a non iterative learning scheme, where the weights from the input to the hidden layer are randomly set while the weights from the hidden to the output layer are determined to minimize a quadratic criterion on a training dataset.

We showed that best RMSE performance was achieved by using uniform and sparse initializations, whereas gaussian initialization was significantly worse.

We also noted that the performance is not smoothly decreasing for a large number of hidden units but it shows error peaks. We validated our hypothesis that this phenomenon is due to the numerical instability and we proposed a technique to cut-off singular values too close to zero; this results in a more robust learning scheme while still reaching very good results.

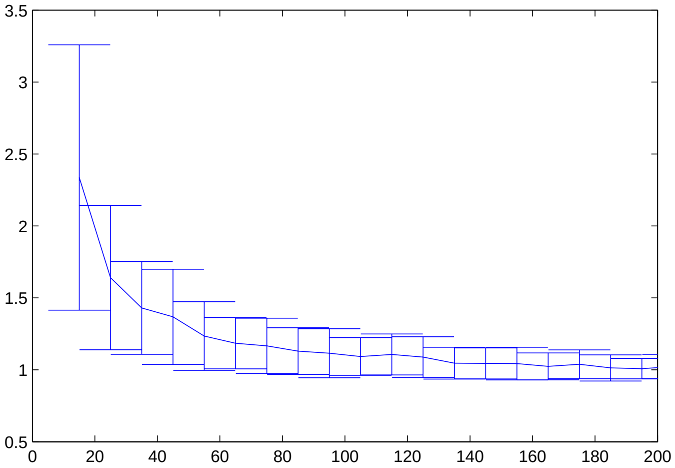

Fig. 5 Best performance vs. attempt number

We also put in evidence that in spite of this, the proposed method based on SVD provides the best performance while pseudoinversion by orthogonal projection, either with or without regularization, achieves higher residuals.

Finally we showed that, on a difficult astronomical regression problem, our proposals significantly improve performance with respect to a SLFN trained with classical backpropagation.

7 Acknowledgements

Work supported by Fondazione CRT, Torino, and Universitá degli Studi di Torino, in the context of the World Wide Style Project.

References

1. D. E. Rumellhart, G. E. Hinton, and R. J.Williams, Learning internal representations by error propagation, Parallel Distrib. Process.: Explanations Microstructure of Cognition, vol. 1, pp. 318-362, 1986.
2. C. Xiang, S. Q. Ding, and T. H. Lee, Geometrical interpretation and architecture selection of MLP, IEEE Trans. Neural Netw., vol. 16, n. 1, pp. 84-96, 2005.
3. J. Park and I. W. Sandberg, Universal approximation using radialbasis- function networks, Neural Comput., vol. 3, pp. 246-257, 1991.
4. LeCun Y., L. Bottou, G. B. Orr, and K.-R.Müller, Efficient backprop, Lecture Notes Comput. Sci., vol. 1524, pp. 9-50, 1998.
5. Bengio Y., Learning deep architectures for AI. Foundations and Trends in Machine Learning, vol. 2, n. 1, pp. 1-127, 2009.
6. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. , An empirical evaluation of deep architectures on problems with many factors of variation. ICML 2007.
7. Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A., Extracting and composing robust features with denoising autoencoders, ICML 2008.

8. Collobert, R., and Weston, J., A unified architecture for language processing: Deep neural networks with multitask learning. ICML 2008.
9. Mnih, A., and Hinton, G. E., A scalable hierarchical distributed language model. NIPS 21, pp. 1081-1088, 2009.
10. G. E. Hinton, S. Osindero, and Y. Teh, A fast learning algorithm for deep belief neets, Neural Computation, vol.18, pp. 1527-1554, 2006.
11. K. Halawa, 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. 293303, 2011.
12. G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, Extreme Learning Machine: Theory and applications, Neurocomputing, vol. 70, pp. 489-501, 2006.
13. R. I. Arriaga and S. Vempala. An algorithmic theory of learning: robust concepts and random projection. In Proc. 40th Annual Symp. on Foundations of Computer Science, pp. 616-623. IEEE Computer Society Press, 1999.
14. S. Vempala. Random projection: a new approach to VLSI layout. In Proc. 39th Annual Symp. on Foundations of Computer Science. IEEE Computer Society Press, 1998.
15. P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In Proc. 30th Symp. on Theory of Computing, pp. 604-613. ACM, 1998.
16. G.-B. Huang, and H. A. Babri, Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions, IEEE Trans. Neural Networks, vol. 9, n. 1, pp. 224-229, 1998.
17. C. M. Bishop, Pattern Recognition and Machine Learning, ed. Springer, Berlin, 2006.
18. P. L. Bartlett, The sample complexity of pattern classification with neural networks: the size of the weights is more important that the size of the network, IEEE Trans. Inf. Theory, vol. 44, n. 2, pp. 525-536, 1998.
19. W.B. Johnson and J. Lindenstrauss, Extensions of Lipshitz mapping into Hilbert space. In Conference in modern analysis and probability, vol. 26 of Contemporary Mathematics, pp. 189-206. Amer. Math. Soc., 1984.
20. S. Dasgupta and A. Gupta, An elementary proof of the Johnson-Lindenstrauss lemma. Technical Report TR-99-006, International Computer Science Institute, Berkeley, California, USA, 1999.
21. R. Hecht-Nielsen, Context vectors: general purpose approximate meaning representations self-organized from raw data. In J.M. Zurada, R.J. Marks II, and C.J. Robinson, editors, Computational Intelligence: Imitating Life, pp. 43-56. IEEE Press, 1994.
22. D. Achlioptas, Database-friendly random projections. In Proc. ACM Symp. on the Principles of Database Systems, pp. 274-281, 2001.
23. E. Bingham and H. Mannila, Random projection in dimensionality reduction: Applications to image and text data. In Proc. of the conference Knowledge Discovery and Data Mining KDD 2001, San Francisco CA, USA, 2001.
24. X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks. Proc. 13-th International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia, Italy, 2010.
25. J.M. Ortega, Matrix Theory, Plenum Press, New York, London, 1987.
26. G. Golub and C. van Loan, Matrix computations, The Johns Hopkins University Press, London, 1996.
27. D. Bini, M. Capovani, O. Menchi, Metodi numerici per l'algebra lineare, ed. Zanichelli, Bologna, Italy, 1988.
28. J.Y. Le Gall, M. Saisse, Chromatic aberration of an all-reflective telescope, Proc. SPIE, vol. 445, pp. 497-504, 1984.
29. Perryman M.A.C. et al., GAIA - Composition, formation and evolution of the galaxy. Concept and technology study, Rep. and Exec. Summary, ESA-SCI(2000)4, European Space Agency, Munich, Germany, 2000.
30. M. Gai, R. Cancelliere, Neural network correction of astrometric chromaticity, MNRAS, vol. 362, n. 4, pp.1483-1488, 2005.
31. R. Cancelliere, M. Gai, Efficient computation and Neural Processing of Astrometric Images, Computing and Informatics, vol. 28, pp. 711-727, 2009.