Computational experience with pseudoinversion-based training of neural networks using random projection matrices

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

[1] Università di Torino, Department of Computer Science
Turin, Italy
{luca.rubini,rossella.cancelliere,andrea.grosso}@unito.it
253081@studenti.unito.it
[2] Laboratory of Computer Sciences, LIP6, Université Pierre et Marie Curie
Paris, France
patrick.gallinari@lip6.fr

Abstract. Recently some novel strategies have been 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; such non-iterative strategies are appealing since they allow fast learning. Aim of this study is to investigate the performance variability when random projections are used for convenient setting of the input weights: we compare them with state of the art setting i.e. weights randomly chosen according to a continuous uniform distribution. We compare the solutions obtained by different methods testing this approach on some UCI datasets for both regression and classification tasks; this results in a significant performance improvement with respect to conventional method.

Keywords: random projections, weights setting, pseudoinverse matrix

1 Introduction

Methods based on gradient descent (and among them the large family of techniques based on backpropagation [1]) have largely been used for training of one of the most common neural architecture, 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 according to the error gradient steepest descent direction. The main critics about gradient descent-based learning are concerned with high computational cost because of slow convergence and zigzagging behavior showed by such

methods, and relevant risk of converging to poor local minima on the landscape of the error function [2].

The reduction of computational efforts in training is of great interest and may become imperative for learning the kind of complicated high-level relations required e.g. in vision [3, 4], natural language processing [5, 6], and other typical artificial intelligence tasks.

A wave of interest has recently grown around some non-iterative procedures based on the evaluation of generalized pseudoinverse matrices. The idea of using these appealing techniques, usually employed to train radial basis function neural networks [7], also for different neural architectures was suggested e.g. in [8]. The work by Huang et al. [9] gave rise to a great interest in neural network community, originating many application-oriented studies in the last years devoted to the use of these single-pass techniques, easy to implement and computationally fast; some are described e.g. in [10, 11, 12, 13]. A yearly conference is currently being held on the subject, the International Conference on Extreme Learning Machines (ELM), and the method is currently dealt with in some journal special issue, e.g. Soft Computing [14] and the International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems [15].

In the pseudoinverse framework input weights and hidden neurons biases are selected randomly, usually according to a uniform distribution in the interval $[-1, 1]$, and no longer modified, while output weights are analytically determined by a single computation of the Moore-Penrose (MP) generalized inverse. Since incremental adjustment of weights is completely avoided these techniques turn out to be very fast when compared to classical gradient descent approaches; 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.

This paper proposes an improvement to the state-of-the-art and focuses in initializing input weights and hidden neurons biases with "special" random structures — specifically, random projection matrices. The theoretical rationale for this approach can be found in many studies, showing random projections as a powerful method for dimensionality treatment [16, 17, 18] thanks to their property to be *almost orthogonal* projections. This feature makes them a potentially useful tool in order to improve performace when dealing with input data relevant features. This argument will be deepened in section 3.

The paper is organized as follows. We recall main ideas on SLFN learning by pseudoinversion in section 2; in section 3 we present funda-

mentals ideas on random projection and finally in section 4 we report results comparing weights setting.

2 Training by Pseudoinversion

In this section we introduce notation and we recall basic idea concerning the use of generalized inverse for neural training.

Fig. 1 shows a standard SLFN with P input neurons, M hidden neurons and Q output neurons, non-linear activation functions ϕ in the hidden layer and linear activation functions in the output layer.

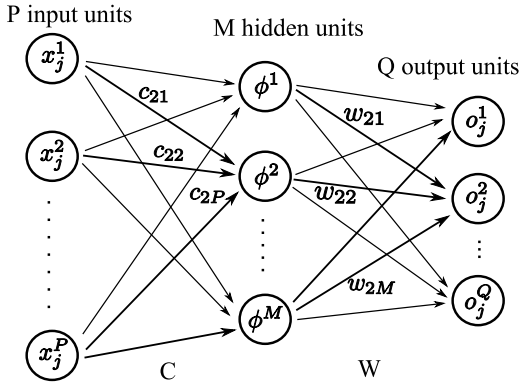

Fig. 1. A Single Layer Feedforward Neural Network.

Considering a dataset of N distinct training samples of (input, output) pairs $(\mathbf{x}_j, \mathbf{t}_j)$, where $\mathbf{x}_j \in \mathbb{R}^P$ and $\mathbf{t}_j \in \mathbb{R}^Q$, the learning process for a SLFN aims at producing the matrix of desired outputs $T \in \mathbb{R}^{N \times Q}$ when the matrix of all input instances $X \in \mathbb{R}^{N \times P}$ is presented as input.

As stated in the introduction, in the state of the art pseudoinverse approach input weights c_{ij} (and hidden neurons biases) are randomly sampled from a uniform distribution in a fixed interval and no longer modified.

After having fixed input weights C, the use of linear output units allows to determine output weights w_{ij} as the solution of the linear system $H\,W\ = T$, where $H \in \mathbb{R}^{N \times M}$ is the hidden layer output matrix of the neural network, $H = \Phi(X\,C)$.

Since H is a rectangular matrix, the least square solution W^* that minimises the cost functional $E_D = ||HW - T||_2^2$, as shown e.g. in [19, 20]

is:

$$W^* = H^+ T. \tag{1}$$

H^+ is the Moore-Penrose generalised inverse (or pseudoinverse) of matrix H.

Direct use of expression (1) is not anyway the best choice because most learning problems are ill-posed; regularisation methods have to be used [21,22] to turn the original problem into a well-posed one, i.e. roughly speaking into a problem insensitive to small changes in initial conditions. Among them, Tikhonov regularisation is one of the most common [23,24]: it minimises the error functional

$$E \equiv E_D + E_R = ||HW - T||_2^2 + \lambda ||W||_2^2. \tag{2}$$

With regularisation we introduce a penalty term that not only improves on stability, but also contains model complexity avoiding overfitting, as largely discussed in [25]. Applications to different neural network models are discussed for instance in [26, 27, 28].

If we consider the singular value decomposition (SVD) of H

$$H = U \Sigma V^T \,, \tag{3}$$

the regularised solution $\hat{W}$ that minimises the error functional (2) has the form (see e.g. [29]):

$$\hat{W} = V D U^T T \,. \tag{4}$$

$U \in \mathbb{R}^{N \times N}$ and $V \in \mathbb{R}^{M \times M}$ are orthogonal matrices and $D \in \mathbb{R}^{M \times N}$ is a rectangular diagonal matrix whose elements, built using the singular values σ_i of matrix Σ, are:

$$D_i = \frac{\sigma_i}{\sigma_i^2 + \lambda}. \tag{5}$$

Therefore in our work we always utilise regularised pseudoinversion. Input weights setting is discussed in next section.

3 Basic Ideas on Random Projections

If $X_{N \times P}$ is the original set of N P-dimensional observations,

$$X_{N \times K}^{RP} = X_{N \times P} C_{P \times K} \tag{6}$$

is the projection of the data onto the new K-dimensional space.

Strictly speaking, a linear mapping such as (6) is not a projection because C is generally not orthogonal and it can cause significant distortions in the data set. However, and unfortunately, orthogonalizing C is computationally expensive. Instead, we can rely on a result presented by Hecht-Nielsen [30]: in a high-dimensional space, there exists a much larger number of *almost orthogonal* than strictly orthogonal directions. Besides, Bingham and Mannila [31] performed an extensive experimentation which allows them to claim that vectors having random directions might be sufficiently close to orthogonality and equivalently that $C^T C$ would approximate an identity matrix. They estimate the mean squared difference between $C^T C$ and the identity matrix is about $1/K$ per element.

This key idea is confirmed also by the Johnson-Lindenstrauss lemma [32]: if a set of points in a vector space is randomly projected onto a selected space of suitable dimension, then the original distances between the points are approximately preserved in the new space, with only minimal distortions. For a simple proof of this result, see [33]. This property appears to be really appealing because suggests the possibility to preserve the topological structure of the initial input space while allowing the creation of a new optimal data representation in the hidden layer space, able to easy the classification/diagnosis task and to increase performance.

Therefore we can use random projections to project the original P-dimensional data into a K-dimensional space, using a random entries matrix $C_{K \times P}$ whose columns have unit norm.

Besides, random projection is very simple from a computational standpoint: the process of forming the random matrix C and projecting the data matrix X into K dimensions has complexity of order $\mathrm{O}(PKN)$; moreover, if the data matrix X is sparse with about G nonzero entries per column, the complexity is of order $\mathrm{O}(GKN)$.

Actually, a large variety of zero mean, unit variance distributions of elements c_{ij} result in a mapping that still satisfies the Johnson-Lindenstrauss lemma: among them, entries of C can be randomly sampled from a gaussian distribution. Another appealing possibility is using sparse random projections which have only a small fraction of nonzero elements. For example, Achlioptas [34] shows that generating random entries c_{ij} by

$$c_{ij} = \sqrt{3} \cdot \begin{cases} +1 & \text{with probability } 1/6 \\ 0 & \text{with probability } 2/3 \\ -1 & \text{with probability } 1/6 \end{cases} \tag{7}$$

one obtains a valid random projection with (expected) density 33%.

A 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 random projections matrices for the setting of input weights while the subsequent processing by hidden nodes nonlinear activation function will account for the non-linearity of the problem.

4 Experimental Investigation

In this section we report results of some numerical experiments performed on the eight benchmark datasets from the UCI repository [35] listed in Table 1, and investigate neural networks with the architecture shown in Fig. 1 and sigmoidal hidden neuron activation functions. The number of input and output neurons is determined by dataset features.

Table 1. UCI datasets characteristics.

Dataset	Type	N. Instances	N. Attributes	N. Classes
Abalone	Regression	4177	8	-
Cpu	Regression	209	6	-
Delta Ailerons	Regression	7129	5	-
Housing	Regression	506	13	-
Iris	Classification	150	4	3
Wine	Classification	178	13	3
Diabetes	Classification	768	8	2
Landsat	Classification	4435	36	7

For the sake of comparison input weights are selected according to i) the conventional strategy, where c_{ij} is sampled from a uniform random distribution in the interval $[-1, 1]$, that in the following will be referred to as Unif. or ii) using random projection matrices with elements c_{ij} gaussian distributed, with mean value 0 and variance 1 (in the following referred to as Gauss.), or iii) using sparse random projection matrices with 33% average density. All simulations are carried out in Matlab 7.10 environment.

4.1 Regularisation Parameter Calibration

To determine the regularisation parameter value for the three cases, for each dataset we gradually increase the number of hidden nodes by unit

steps in an unregularised framework (eq. (2), $\lambda = 0$); for each selected hidden layer size, average RMSE (for regression tasks), or average misclassification rate (for classification tasks) were computed over 100 different initial trials for each input weight setting, i.e. uniform and gaussian.

All datasets show, after an initial steep decrease, a fast error growth as a function of the hidden layer size, opposite to the monotonically decreasing training error.

This effect is typically caused by overfitting, arising when a large amount of free parameters is available to reproduce almost exactly training data.

The best performance is associated to an interval of hidden neurons, that we name critical dimension, in which we decided to look for, according to a cross validation scheme, the value of λ resulting in the best score: its determination concludes the calibration phase.

4.2 Computational Results

Comparison of the relative strengths of the approaches studied in this work is assessed by evaluation of the mean test error resulting from 100 trials for each fixed size of SLFN in the regularised framework: the test performance is reported in Table 2.

We underline that the regularised test error features a monotonic decrease as a function of hidden neurons number, proving that regularisation is necessary to provide overfitting control, and to allow optimal exploitation of the superior potential of larger architectures.

In the "Error" columns we report the average value (of 100 trials) and standard deviation of the RMSE for regression datasets; for classification datasets, we report average value (of 100 trials) and standard deviation for the percentage missclassification error. On each row, the lowest average error figure is highlighted in bold whenever we can prove a statistically significant dominance of the random-projection initialization over the random-uniform initialization, assessed with at least a confidence level of 95%in the Student's test.

We also report the number of hidden neurons N_H and the value of λ emerged from the calibration phase.

As far as the testing performance is concerned, we can claim a substantial dominance of the random projections based approach over the classical uniform initialization.

We then compared the test performance of networks with initialization based on random projections and trained by pseudoinversion against

the test performances of networks trained with a classical backpropagation method. The comparison is shown in Table 3; for each dataset, the "PINV" columns report the error statistics for the winner observed in Table 2. Statistics for backpropagation are taken from `tunedit.org`, except for the Wine dataset, for which we got better results than `tunedit`'s ones by running the backpropagation method on our own under WEKA. For all datasets in the table we can claim dominance of the pseudoinversion based approach with a 99% confidence level.

In our experiments, the running times of all the pseudoinversion-based approaches are substantially equivalent, hence we base the comparison only on the average error. As far as the comparison with backpropagation is concerned, pseudoinversion based methods save a relevant amount of time, being up to 10 times faster than backpropagation. For example, 10 runs of pseudoinversion-based training on the Wine dataset require 0.078 seconds on average whereas backpropagation requires on average 0.721 seconds (times on a laptop with Pentium CPU, 2 GHz clock, 4 GB RAM); other tests gave roughly similar results.

5 Conclusions

We considered pseudoinversion-based techniques for training of neural networks feeding them by random projections (gaussian and sparse) matrices of input weights and biases instead of the classical uniform-random initialization. We believe that the computational results presented in this paper assess initialization by random projection matrices as a useful tool for improving performances

Future steps in the research will consider hybridizing the pseudoinversion-based training technique with basic descent techniques. The rationale behind this is that pseudoinversion-based techniques mostly rely on a pure random sampling of input weights and biases, whereas it could make sense trying to profit also from some local exploration of the error landscape.

6 Acknowledgment

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).

Table 2. Random projections vs. random-uniform setting. For Delta Ailerons, the average errors and standard deviations are multiplied by 10^{-4}.

Dataset	**Unif.**				**Gauss.**				**Sparse**			
	Error				Error				Error			
	Avg	StD	N_H	λ	Avg	StD	N_H	λ	Avg	StD	N_H	λ
Abalone	2.165	0.004	128	$3 \cdot 10^{-2}$	2.169	0.009	129	$3 \cdot 10^{-1}$	**2.162**	0.006	118	$3 \cdot 10^{-2}$
Mach. Cpu	57.35	1.7	98	$4 \cdot 10^{-2}$	**56.85**	2.8	61	$8 \cdot 10^{-1}$	57.86	1.6	89	$5 \cdot 10^{-1}$
Delta Ail.(10^{-4})	1.636	$2 \cdot 10^{-3}$	244	$3 \cdot 10^{-3}$	**1.630**	$4 \cdot 10^{-3}$	272	$3 \cdot 10^{-2}$	1.636	$2 \cdot 10^{-3}$	225	$3 \cdot 10^{-3}$
Housing	3.61	0.21	130	$8 \cdot 10^{-3}$	3.58	0.19	200	$5 \cdot 10^{-2}$	3.64	0.18	180	$7 \cdot 10^{-2}$
Iris	1.00	1.1	102	$3 \cdot 10^{-4}$	1.88	1.1	120	$3 \cdot 10^{-2}$	1.08	1.0	266	$3 \cdot 10^{-3}$
Diabetes	20.312	0.8	266	$3 \cdot 10^{-3}$	20.430	1.0	173	$3 \cdot 10^{-2}$	**20.086**	1.0	192	$3 \cdot 10^{-3}$
Landsat	10.438	0.32	579	$3 \cdot 10^{-3}$	**9.848**	0.30	600	$3 \cdot 10^{-2}$	10.394	0.32	600	$3 \cdot 10^{-3}$
Wine	2.2542	1.5246	60	$3 \cdot 10^{-2}$	2.0847	1.6313	70	$2 \cdot 10^{-1}$	2.5593	1.5704	80	$8 \cdot 10^{-2}$

Table 3. Random projections based training (pseudoinversion) vs. backpropagation.

Dataset	**PINV**		**Backprop.**		
	Avg	StDev	Avg	(N_{tests})	StDev
Abalone	**2.162**	0.006	2.3044	(35)	0.1908
Mach. Cpu	**56.85**	2.8	28.6673	(5)	27.3535
Delta Ail.	$\mathbf{1.630 \cdot 10^{-4}}$	$4 \cdot 10^{-7}$	$2 \cdot 10^{-3}$	(10)	0.0
Housing	**3.58**	0.19	4.5492	(35)	0.9517
Iris	**1.00**	1.1	1.73	(10)	0.85
Diabetes	**20.086**	1.0	26.52	(31)	2.38
Landsat	**9.848**	0.30	13.03	(5)	0.63
Wine	**2.0847**	1.6313	3.77	(10)	0

References

1. Rumellhart, D. E., Hinton, G. E., Williams, R. J.: Learning internal representations by error propagation. In: Parallel Distrib. Process.: Exploration in the Microstructure of Cognition, MIT Press, Cambridge, Massachussetts, vol. 1, pp. 318-362 (1986)
2. LeCun, Y., Bottou, L., Orr, G. B., Müller, K.-R.: Efficient backprop, Lecture Notes Comput. Sci., vol. 1524, pp. 9-50 (1998)
3. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: 24th ICML (2007)
4. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A.: Extracting and composing robust features with denoising autoencoders. In: 25th ICML (2008)
5. Collobert, R., Weston, J.: A unified architecture for language processing: Deep neural networks with multitask learning. In: 25th ICML (2008)
6. Mnih, A., Hinton, G. E.: A scalable hierarchical distributed language model. In: 23rd NIPS, pp. 1081-1088 (2009)
7. Poggio, T., Girosi, F.: Networks for approximation and learning. In: IEEE 78, n. 9, 1481 - 1497 (1990)
8. Cancelliere, R.: A High Parallel Procedure to initialize the Output Weights of a Radial Basis Function or BP Neural Network. In: Sorevik, T., Manne, F., Moe, R., Gebremedhin, A. H., (eds.) PARA 2000. LNCS, vol. 1947, pp. 384-390, Springer Verlag(2001)
9. Huang, G.-B. , Zhu, Q.-Y., Siew, C.-K.: Extreme Learning Machine: Theory and applications. Neurocomputing 70, pp. 489-501 (2006)
10. Halawa, K.: 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 34, pp. 293-303 (2011)
11. Nguyen, T. D., Pham, H. T. B., Dang, V. H.: An efficient Pseudo Inverse matrix-based solution for secure auditing. In: IEEE International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (2010)
12. Kohno, K., Kawamoto, M., Inouye, Y.: A Matrix Pseudoinversion Lemma and Its Application to Block-Based Adaptive Blind Deconvolution for MIMO Systems. Circuits and Systems I: Regular Papers, IEEE Transactions on, 57, no.7, pp.1449-1462 (2010)
13. Ajorloo, H., Manzuri-Shalmani, M. T., Lakdashti, A. : Restoration of damaged slices in images using matrix pseudo inversion. In: 22nd International Symposium on Computer and Information Sciences (2007)
14. Wang, X.-Z., Wang D., Huang, G.-B.: Special Issue on Extreme Learning Machines. Editorial. Soft Comput. 16(9), pp. 1461-1463 (2012).
15. Wang, X.: Special Issue on Extreme Learning Machine with Uncertainty. Editorial. Int. J. Unc. Fuzz. Knowl. Based Syst. 21(supp02), pp. v-vi (2013).
16. Arriaga, R. I., Vempala, S.: An algorithmic theory of learning: robust concepts and random projection. In: 40th Annual Symp. on Foundations of Computer Science, pp. 616-623. IEEE Computer Society Press (1999)
17. Vempala, S.: Random projection: a new approach to VLSI layout. In: 39th Annual Symp. on Foundations of Computer Science. IEEE Computer Society Press (1998)
18. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: 30th Symp. on Theory of Computing, pp. 604-613. ACM (1998)

19. Penrose, R.: On best approximate solution of linear matrix equations. Proceedings of the Cambridge Philosophical Society 52, pp. 17-19 (1956)
20. Bishop, C. M.: Pattern Recognition and Machine Learning. Springer, Berlin (2006)
21. Badeva, V., Morosov, V.: Problemes incorrectements posès, thèorie et applications, (in French). Masson, Paris (1991)
22. Cancelliere, R., De Luca, R., Gai, M., Gallinari, P., Artires, T.: Pseudoinversion for neural training: tuning the regularisation parameter. Technical report n. 149/13, Dep. of Computer Science, University of Turin (2013)
23. Tikhonov, A.N., Arsenin, V.Y.: Solutions of Ill-Posed Problems. Winston, Washington, DC (1977)
24. Tikhonov, A.N.: Solution of incorrectly formulated problems and the regularization method. Soviet Mathematics 4, pp. 1035-1038 (1963)
25. Gallinari, P. , Cibas, T.: Practical complexity control in multilayer perceptrons. Signal Processing 74, pp. 29-46 (1999)
26. Poggio, T., Girosi, F.: Regularization algorithms that are equivalent to multilayer networks. Science 247, pp. 978-982 (1990)
27. Girosi, F., Jones, M., Poggio, T.: Regularization theory and neural networks architectures. Neural Computation 7 (2), pp. 219-269 (1995)
28. Haykin, S.: Neural Networks, a comprehensive foundation. Prentice Hall, U.S.A (1999)
29. Fuhry, M., Reichel, L.: A new Tikhonov regularization method. Numerical Algorithms 59, pp. 433-445 (2012)
30. Hecht-Nielsen, R.: Context vectors: general purpose approximate meaning representations self-organized from raw data. In: Zurada J.M., Marks II R.J., Robinson C.J. (eds.), Computational Intelligence: Imitating Life, pp. 43-56. IEEE Press (1994)
31. Bingham, E. , Mannila, H.: Random projection in dimensionality reduction: Applications to image and text data. In: Conference on Knowledge Discovery and Data Mining KDD 2001, San Francisco CA, USA (2001)
32. Johnson, W.B., Lindenstrauss, J.: 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)
33. Dasgupta, S., Gupta, A.: An elementary proof of the Johnson-Lindenstrauss lemma. Technical report TR-99-006, International Computer Science Institute, Berkeley, California, USA (1999)
34. Achlioptas, D.: Database-friendly random projections. In: ACM Symp. on the Principles of Database Systems, pp. 274-281 (2001)
35. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, `http://www.ics.uci.edu/~mlearn/MLRepository.html` (2007)