2023 20th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE) | 979-8-3503-0676-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCE60043.2023.10332840

Computing Pseudoinverse Matrices with Single-Layer Neural Networks

Felipe Gómez Castañeda
Electrical Engineering
Department, CINVESTAV-IPN,
Mexico City, Mexico
fgomez@cinvestav.mx

Luis Martín Flores Nava
Electrical Engineering
Department, CINVESTAV-IPN,
Mexico City, Mexico
lmflores@cinvestav.mx

Alvaro Anzueto Ríos
Bionic Academic, UPIITA-IPN,
Mexico City, Mexico
aanzuetor@ipn.mx

José Antonio Moreno Cadenas
Electrical Engineering
Department, CINVESTAV-IPN,
Mexico City, Mexico
jmoreno@cinvestav.mx

***Abstract*— This work evaluates numerical results related to single-layer neural networks, which compute pseudoinverse matrices. This is possible due to the gradient descendent algorithms available in the machine learning TensorFlow framework, which optimizes training processes. Linear algebra principles support our formulations, which are proved at the end of the training stage.**

Keywords— Pseudoinverse, Neural Network, Machine Learning, Adamax.

I. Introduction

Currently, machine learning systems based on neural networks and linear algebra converge to provide efficient solutions to diverse and complex problems in engineering, science, and other knowledge fields.

This is the case of processing raw data as a dimensionality reduction task, where an autoencoder neural network can be used. Its simulation and analysis are done with TensorFlow for efficient numerical performance. Considering this machine learning framework i.e. using the training algorithms based on the gradient descendent method, we propose computing the pseudoinverse matrix of full-rank matrices as an approach supported by a single-layer neural network (SLNN). To this end, a linear algebra formulation follows before this computation; namely, basic formulae relate linearly output with input matrices in this neural technique, from which after training the pseudoinverse matrix can be observed with acceptable precision.

II. Pseudoinverse matrix concept

The pseudoinverse matrix X of matrix Y, denoted as Y^{+}, was first stated by Moore and later it was completed by Penrose [1]. The Moore-Penrose pseudoinverse is recognized as such, satisfying the equations below.

$$YXY = Y \tag{1}$$

$$XYX = X \tag{2}$$

$$(YX)^T = YX \tag{3}$$

$$(XY)^T = XY \tag{4}$$

The superscript T means transpose. From these 4 equations, if only one is satisfied, then it is called the generalized inverse; this is the case in this work. If Y is a full-rank matrix, where: $Y \in \mathbb{R}^{mxn}$, then the Moore-Penrose pseudoinverse Y^{+} can be calculated numerically using either (5) or (7) below, where $(\bullet)^{-1}$ indicates inverse.

$$(Y^TY)^{-1}Y^T \quad if \quad m > n \tag{5}$$

$$Y^{-1} \quad if \quad m = n \tag{6}$$

$$Y^T(YY^T)^{-1} \quad if \quad m < n \tag{7}$$

In this work, the numerical value of the Moore-Penrose pseudoinverse is taken as a reference for comparison with the computed pseudoinverse matrix by SLNNs. Additionally, because many actual cases deal with data matrices whose dimension is m>n, we consider only this type of rectangular matrix.

III. Proposed methodology

Neural network models were used earlier [2] as parallel processes for computing pseudoinverse matrices, where their timing was the main issue. At present, with both computer technology and machine learning techniques neural models become a means for efficiently dealing with complex data in real-time. This fact occurs mainly because the existing training algorithms are based on optimal gradient descent methods [3]. Observing this advantage, we benefit by applying its potential in training single-layer neural networks to compute pseudoinverse matrices.

This methodology codes in Python linear neural networks in a TensorFlow/Spyder framework and uses Adamax from the Keras library for their training. The architecture of these networks is shown in Figure 1, where A and B are the input and output matrices, respectively. The weighting matrix W connects the input and output layers and it is the object of training. The pseudoinverse matrix X of matrix Y can be computed with

979-8-3503-0676-7/23/$31.00 ©2023 IEEE

appropriate accuracy by proposing different formulations for matrices A and B.

We present two formulations in Table I that support relevant results from other rejected initial ones. The loss functions are evaluated as Frobenious norms. Matrices O and I mean orthogonal and identity matrices, respectively. We describe the origin of these formulations below based on linear algebra principles.

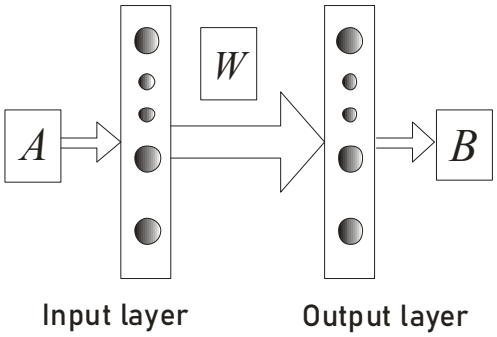

Fig. 1. The architecture of these networks.

TABLE I. FORMULATIONS: 1 AND 2

		After training	Loss function
1	$A = Y$	$B = X^{\mathsf{T}}$, $W = X X^{\mathsf{T}}$	$\mathrm{L} = \mid Y^{\mathsf{T}} B - I \mid^2_{\mathrm{F}}$
2	$A = O Y^{\mathsf{T}} Y$	$B = O$, $X = W Y^{\mathsf{T}}$	$\mathrm{L} = \mid O - B \mid^2_{\mathrm{F}}$

The first formulation computes the pseudoinverse X of Y via the transpose of the output matrix. This comes from relating A=Y and B=X^{T} with W=XX^{T} i.e. $Y(XX^{\mathsf{T}}) = X^{\mathsf{T}}$, which gets high accuracy after training.

Using the second formulation, the pseudoinverse X is obtained by multiplying W by the transpose of Y. In this case, the original input and output matrices are A=$OY^{\mathsf{T}}Y$ and B=O. Relating both A and B with W, it gets $OY^{\mathsf{T}}Y\,W = O$. Working out this equation as with the 8 equations below, the pseudoinverse X is obtained.

$$OY^TYW = O \tag{8}$$

$$(OY^TY)^{-1}OY^TYW = (OY^TY)^{-1}O \tag{9}$$

$$W = (OY^TY)^{-1}O \tag{10}$$

$$W = (Y^TY)^{-1}O^{-1}O \tag{11}$$

$$W = (Y^TY)^{-1} \tag{12}$$

$$W = Y^{-1}(Y^T)^{-1} \tag{13}$$

$$WY^T = Y^{-1} \tag{14}$$

$$WY^T = X \tag{15}$$

Notably, the loss functions in Table I are responsible for the end values of matrices X that approach pseudoinverse matrices with acceptable accuracy.

IV. EVALUATION OF RESULTS

The performance of both Formulation 1 and Formulation 2 is evaluated by counting the number of digits that match between the elements of the pseudoinverse matrix using the numerical Moore-Penrose algorithm and the one given by the neural network. This evaluation is presented visually as whisker graphs. Figures 2 and 6 compare the effect of using either a uniform or a normal distribution in the entries of the matrices for computing their pseudoinverse, where the uniform one presents the best performance. Figures 2-5 show the results for the first formulation meanwhile, Figures 6-9 present results for the second formulation.

For original matrices Y, whose entries are in [0, 1], this neural method does not converge. The solution is to transform them into matrices with entries in [-1, 1]. The following equations sequence (16-19), demonstrates the supporting transformation which helps to find Y^{+}. Where, Z is an auxiliary and arbitrary matrix, whose entries are random in [-1, 1].

$$Y = YZ(Z)^{+} \tag{16}$$

$$(Y)^{+} = (YZ(Z)^{+})^{+} \tag{17}$$

$$(Y)^{+} = ((Z)^{+})^{+}(YZ)^{+} \tag{18}$$

$$Y^{+} = Z(YZ)^{+} \tag{19}$$

Using the new matrix YZ, whose entries are now in [-1, 1], its pseudoinverse $(YZ)^{+}$ can be computed by the neural network. Therefore, the pseudoinverse matrix Y^{+} is obtained by multiplying $(YZ)^{+}$ by Z.

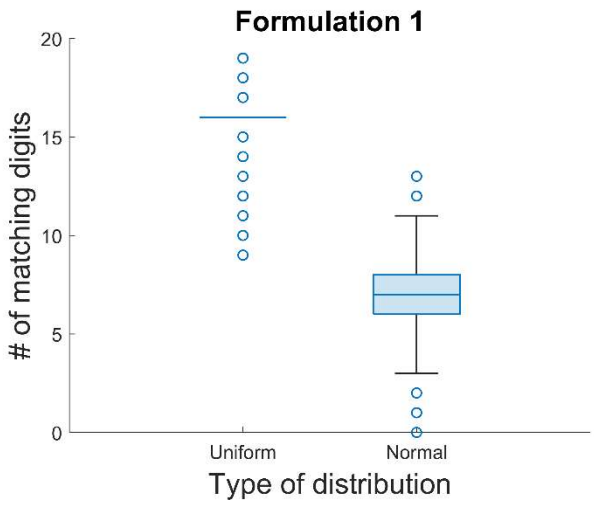

Fig. 2. Effect of Uniform and Normal distribution in the entries of matrices with a 200x50 dimension. Entries are in [-1, 1].

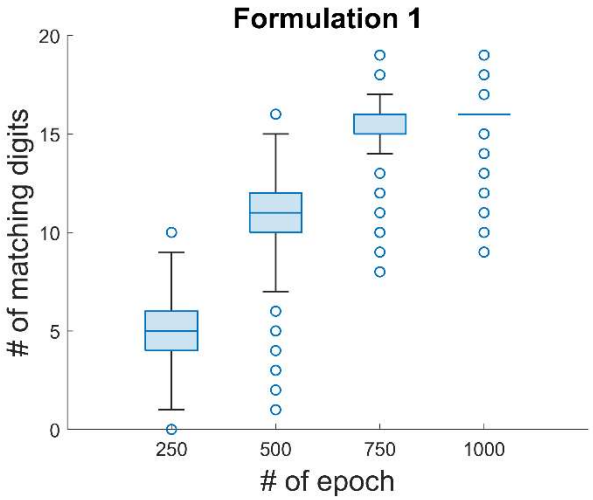

Fig. 3. Effect of the number of epochs in a matrix with a 200x50 dimension. Entries are random in [-1, 1].

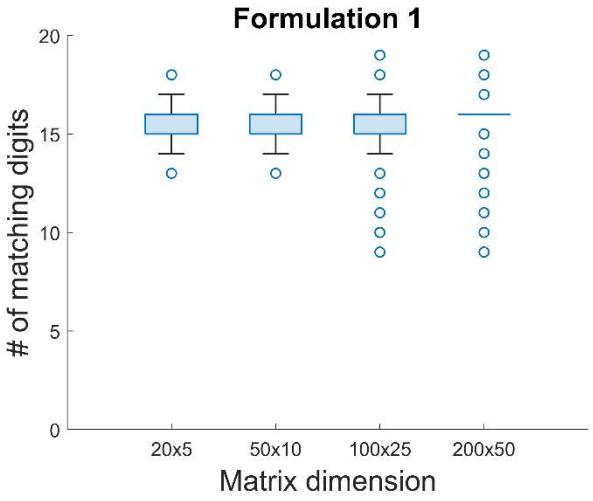

Fig. 4. Effect of the matrix dimension. Entries are random in [-1, 1].

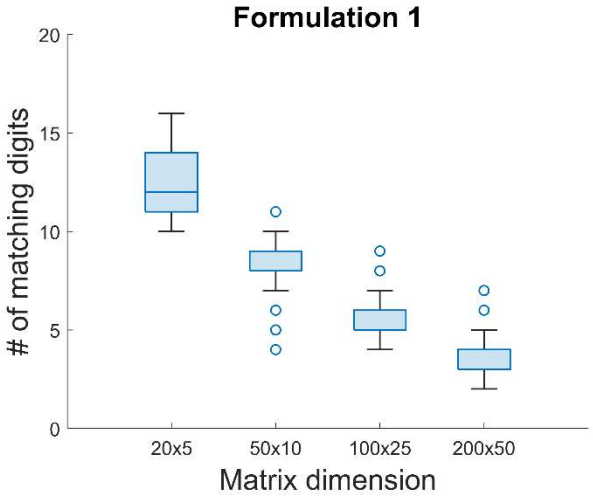

Fig. 5. Effect of the matrix dimension. Entries are random in [0, 1].

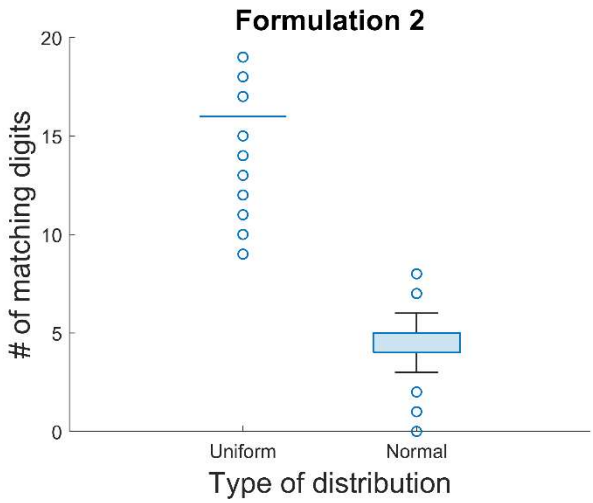

Fig. 6. Effect of Uniform and Normal distribution in the entries of matrices with a 200x50 dimension. Entries are in [-1, 1].

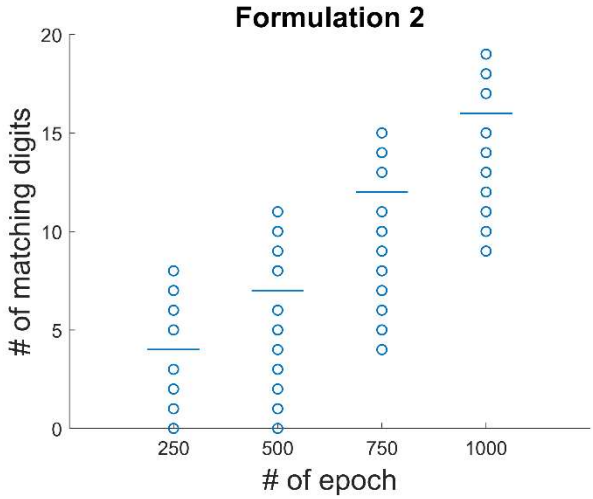

Fig. 7. Effect of the number of epochs in matrix with a 200x50 dimension. Entries are random in [-1, 1].

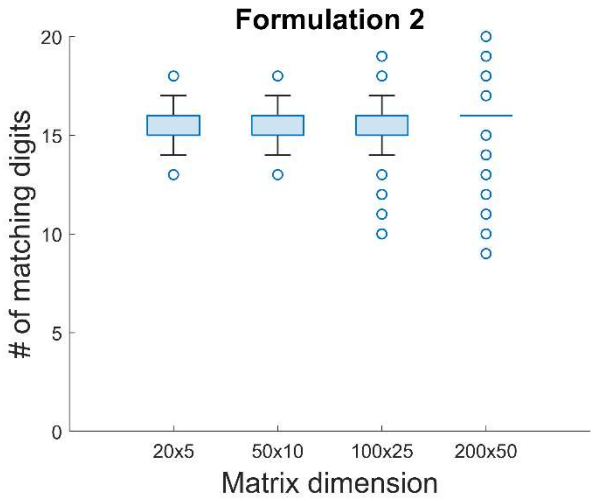

Fig. 8. Effect of the matrix dimension. Entries are random in [-1, 1].

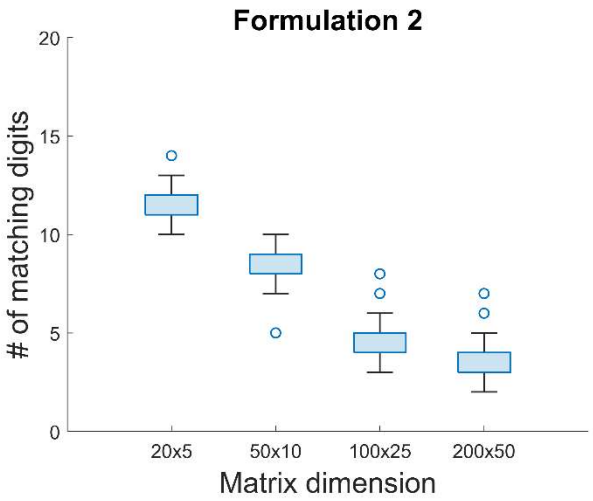

Fig. 9. Effect of the matrix dimension. Entries are random in [0, 1].

Some extra details of the above results follow.

Figures 2 and 6 deal with two types of distribution: random uniform and Normal. Both generate the entries of the matrices in [-1, 1] and, the media and the standard deviation are 0 and 0.3, respectively for the Normal case.
Figures 3 and 7 show the effect of the number of epochs for a 200x50 dimension matrix, whose entries are random and uniform in [-1, 1].

Figures 4 and 8 present the same number of matching digits i.e. 15, which is obtained with 2500, 2000, 1500, and 1000 epochs for matrices whose dimension is 20x5, 50x10, 100x25, and 200x50, respectively.

Figures 5 and 9 were obtained with 500 epochs for matrices whose entries are random and uniform in [0, 1].

The graphical results in all Figures indicate that the number of matching digits is in the range between 12 and 15, which is equivalent to a Frobenious value lower than $1x10^{-30}$. The obtained accuracy is acceptable for a wide set of applications, which might include the implementation of both the training algorithm and the neural network with FPGA technology.

V. Discussion

The objective of this work was to present a series of numerical results from the training of single-layer linear neural networks for approaching pseudoinverse matrices with good accuracy relative to their numerical representation with standard computing systems. This objective was reached due to the training algorithms available from machine learning frameworks. The strategy was established via linear algebra principles that relate matrices, in this case proposing 2 formulations, which are arbitrary but certain. Moreover, to evaluate possible trends in the final accuracy, the effect of the following factors was considered.

1. A normal o Gaussian distribution against a uniform distribution of the entry values of the matrices.
2. Number of training epochs.
3. Matrix dimension.
4. The combination of the above.

And, the Figures 2-9 are a graphical presentation of these 4 points. From these Figures we can choose those ones whose # of matching digits is large and combine them properly as a criterion to get the best performance. This task is left open to further research.
.

VI. Conclusions

The authors presented an exploration of the usefulness of the gradient descendent training method Adamax, on single-layer neural networks for computing pseudoinverse matrices. Two formulations were established by linear algebra principles and proofed numerically leading to acceptable accuracies. Moreover, the TensorFlow framework with Keras library reduces time analysis for large dimension matrices.

References

[1] R. Penrose, “A Generalized Inverse for Matrices”, Mathematical Proceedings of the Cambridge Philosophical Society, pp. 406-413, 1954.

[2] M. M. Polycarpou, P.A. Ioannou, “A Neural-Type Parallel Algorithm for Fast Matrix Inversion”, Proceedings of the Fifth International Parallel Processing Symposium, pp. 108-113, 1991.

[3] Saad Hikmat Haji, Adnan Mohsin Abdulazeez, “Comparison of Optimization Techniques Based on Gradient Descent Algorithm: A Review”, PalArch's Journal of Archaeology of Egypt / Egyptology, pp. 2715-2743, 2021.