Hypercomplex-valued Neural Networks

Part 2 – Vector-Valued Matrix Computation.

Marcos Eduardo Valle
Universidade Estadual de Campinas (Unicamp)
Campinas, Brazil.

Introduction

Matrix computation is a key concept for developing efficient hypercomplex-valued network models because some fundamental building blocks, like dense and convolutional layers, compute affine transformations followed by a non-linear function.

In the following, we present some vector-valued matrix computation concepts, including the vector-valued least-squares problems.

Let $\mathbb{V}$ be an algebra over the real numbers. The product of two vector-valued matrices $\boldsymbol{A} \in \mathbb{V}^{M \times L}$ and $\boldsymbol{B} \in \mathbb{V}^{L \times N}$ results in a new matrix $\boldsymbol{C} \in \mathbb{V}^{M \times N}$ with entries defined by

$$c_{ij} = \sum_{\ell=1}^{L} a_{i\ell} b_{\ell j}, \quad \forall i = 1, \ldots, M \quad \text{and} \quad j = 1, \ldots, N.$$

To take advantage of fast scientific computing software, we compute the above operation using real-valued matrix operations.

Let $\mathcal{E} = \{e_1, \ldots, e_n\}$ be a basis for $\mathbb{V}$. Using the isomorphism $\varphi : \mathbb{V} \to \mathbb{R}^n$ defined by

$$\varphi(x) = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \forall x = \sum_{i=1}^{n} \in \mathbb{V},$$

we have

$$\varphi(c_{ij}) = \sum_{\ell=1}^{L} \varphi\left(a_{i\ell} b_{\ell j}\right) = \sum_{\ell=1}^{L} \mathcal{M}_L(a_{i\ell})\varphi(b_{\ell j}),$$

where $\mathcal{M}_L : \mathbb{H} \to \mathbb{R}^{n \times n}$ is the matrix representation of multiplication to the left by $a = \sum_{i=1}^{n} a_i e_i$.

Equivalently, using real-valued matrix operations, we have

$$\varphi(\boldsymbol{C}) = \mathcal{M}_L(\boldsymbol{A})\varphi(\boldsymbol{B}),$$

where $\mathcal{M}_L$ and φ are defined as follows for vector-valued matrices:

$$\mathcal{M}_L(\boldsymbol{A}) = \begin{bmatrix} \mathcal{M}_L(a_{11}) & \mathcal{M}_L(a_{12}) & \dots & \mathcal{M}_L(a_{1L}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{M}_L(a_{M1}) & \mathcal{M}_L(a_{M2}) & \dots & \mathcal{M}_L(a_{ML}) \end{bmatrix} \in \mathbb{R}^{nM \times nL},$$

and

$$\varphi(\boldsymbol{B}) = \begin{bmatrix} \varphi(b_{11}) & \dots & \varphi(b_{1N}) \\ \varphi(b_{21}) & \dots & \varphi(b_{2N}) \\ \vdots & \ddots & \vdots \\ \varphi(b_{L1}) & \dots & \varphi(b_{LN}) \end{bmatrix} \in \mathbb{R}^{nL \times N}.$$

Reorganizing the elements of $\varphi(\boldsymbol{C})$, we can write

$$\boldsymbol{C} = \varphi^{-1}\left(\mathcal{M}_L(\boldsymbol{A})\varphi(\boldsymbol{B})\right),$$

which allows the computation of hypercomplex-valued matrix products using the real-valued linear algebra often available in scientific computing software.

To further reduce the computing time, the real-valued matrix $\mathcal{M}_L(\boldsymbol{A}) \in \mathbb{R}^{nM \times nL}$ can be computed using the Kronecker product.

Kronecker Product

The Kronecker product between two real-valued matrices $A = (a_{ij}) \in \mathbb{R}^{N \times M}$ and $B \in \mathbb{R}^{P \times Q}$, denoted by $A \otimes B$, yields the block matrix defined by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1M}B \\ a_{21}B & a_{22}B & \dots & a_{2M}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1}B & a_{N2}B & \dots & a_{NM}B \end{bmatrix} \in \mathbb{R}^{NP \times MQ}. \quad (1)$$

Basic properties and some applications of the Kronecker product can be found in (Loan, 2000; Stenger, 1968).

Based on Zhang et al. (2021) and Grassucci et al. (2022), we use the Kronecker product to compute $\mathcal{M}_L(\boldsymbol{A})$ as follows.

Recall from the previous part that the matrix representation of the multiplication by the left by $a_{ij} = \sum_{k=1}^{n} a_{ijk} e_k$ satisfies

$$\mathcal{M}_L(a_{ij}) = \sum_{k=1}^{n} a_{ijk} P_{k:}^T, \quad \text{with} \quad P_{k:}^T = \begin{bmatrix} p_{k11} & p_{k21} & \dots & p_{kn1} \\ p_{k12} & p_{k22} & \dots & p_{kn2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k1n} & p_{k2n} & \dots & p_{knn} \end{bmatrix}.$$

Thus, we have

$$\mathcal{M}_L(\boldsymbol{A}) = \begin{bmatrix} \sum_{k=1}^{n} a_{11k} P_{k:}^T & \sum_{k=1}^{n} a_{12k} P_{k:}^T & \dots & \sum_{k=1}^{n} a_{iLk} P_{k:}^T \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} a_{M1k} P_{k:}^T & \sum_{k=1}^{n} a_{M2k} P_{k:}^T & \dots & \sum_{k=1}^{n} a_{MLk} P_{k:}^T \end{bmatrix}$$

$$= \sum_{k=1}^{n} \begin{bmatrix} a_{11k} P_{k:}^T & a_{12k} P_{k:}^T & \dots & a_{iLk} P_{k:}^T \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1k} P_{k:}^T & a_{M2k} P_{k:}^T & \dots & a_{MLk} P_{k:}^T \end{bmatrix}.$$

Let $A_k \in \mathbb{R}^{M \times L}$, for $k = 1, \ldots, n$, be the real-valued matrices such that

$$\boldsymbol{A} = \sum_{k=1}^{n} A_k e_k.$$

In words, A_k is the "matrix" component associated with the basis element e_k of $\boldsymbol{A}$.

Using $A_k \in \mathbb{R}^{M \times L}$, we conclude that

$$\mathcal{M}_L(\boldsymbol{A}) = \sum_{k=1}^{n} A_k \otimes P_{k:}^T. \tag{2}$$

Therefore, $\boldsymbol{C} = \boldsymbol{AB}$ can be efficiently computed by the equation

$$\boldsymbol{C} = \varphi^{-1}\left(\left(\sum_{k=1}^{n} A_k \otimes P_{k:}^T\right)\varphi(\boldsymbol{B})\right).$$

Alternatively, it is possible to compute $\boldsymbol{C} = \boldsymbol{AB}$ using the multiplication to the right by $b_{\ell j}$.

In this case,

$$\varphi^T(\boldsymbol{C}) = \varphi^T(\boldsymbol{A})\mathcal{M}_R(\boldsymbol{B}),$$

where

$$\varphi^T(\boldsymbol{A}) = \begin{bmatrix} \varphi(a_{11})^T & \dots & \varphi(a_{1L})^T \\ \vdots & \ddots & \vdots \\ \varphi(a_{M1})^T & \dots & \varphi(a_{ML})^T \end{bmatrix} \in \mathbb{R}^{M\times(nL)},$$

and

$$\mathcal{M}_R(\boldsymbol{B}) = \begin{bmatrix} \mathcal{M}_R(b_{11}) & \mathcal{M}_R(b_{21}) & \dots & \mathcal{M}_R(b_{1N}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{M}_R(b_{L1}) & \mathcal{M}_R(b_{L2}) & \dots & \mathcal{M}_R(b_{LN}) \end{bmatrix} \in \mathbb{R}^{(nL)\times(nN)}.$$

Moreover, if

$$\boldsymbol{B} = \sum_{k=1}^{n} B_k e_k,$$

where $B_k \in \mathbb{R}^{L \times N}$ are real-valued matrices for $k = 1, \ldots, n$, then $\mathcal{M}_R(\boldsymbol{B})$ can be efficiently computed using the Kronecker product as follows:

$$\mathcal{M}_R(\boldsymbol{B}) = \sum_{k=1}^{n} B_k \otimes P_{:k}.$$

From a computational standpoint, $\varphi(\boldsymbol{C}) = \mathcal{M}_L(\boldsymbol{A})\varphi(\boldsymbol{B})$ is faster than $\varphi^T(\boldsymbol{C}) = \varphi^T(A)\mathcal{M}_R(\boldsymbol{B})$ if the matrix $\boldsymbol{A}$ has less entries than $\boldsymbol{B}$, and vice-versa.

We suggest implementing both formulas and computing the product using the fastest one.

Example – Quaternions

Consider the quaternion-valued matrix and vector:

$$\boldsymbol{A} = \begin{bmatrix} 1 + 2\boldsymbol{i} & 3\boldsymbol{i} + 4\boldsymbol{j} & 5\boldsymbol{j} + 6\boldsymbol{k} \\ 7 + 8\boldsymbol{j} & 9 + 10\boldsymbol{k} & 11\boldsymbol{i} + 12\boldsymbol{k} \end{bmatrix} \in \mathbb{Q}^{2\times 3},$$

and the column vector

$$\boldsymbol{x} = \begin{bmatrix} 1 + 2\boldsymbol{i} + 3\boldsymbol{j} + 4\boldsymbol{k} \\ 5 + 6\boldsymbol{i} + 7\boldsymbol{j} + 8\boldsymbol{k} \\ 9 + 10\boldsymbol{i} + 11\boldsymbol{j} + 12\boldsymbol{k} \end{bmatrix} \in \mathbb{Q}^{3\times 1}.$$

Using quaternion matrix algebra, we obtain

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} -176 + 45\boldsymbol{i} + 96\boldsymbol{j} + 11\boldsymbol{k} \\ -306 - 3\boldsymbol{i} + 140\boldsymbol{j} + 363\boldsymbol{k} \end{bmatrix}.$$

Using the multiplication by the left, we compute

$$\mathcal{M}_L(\boldsymbol{A}) = \begin{bmatrix} 1 & 0 & 0 \\ 7 & 9 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \ldots + A_3 \otimes P_{3:}^T$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \end{bmatrix} + \ldots + A_3 \otimes P_{3:}^T$$

$$= \begin{bmatrix} 1 & -2 & 0 & 0 & 0 & -3 & -4 & 0 & 0 & 0 & -5 & -6 \\ 2 & 1 & 0 & 0 & 3 & 0 & 0 & 4 & 0 & 0 & -6 & 5 \\ 0 & 0 & 1 & -2 & 4 & 0 & 0 & -3 & 5 & 6 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & -4 & 3 & 0 & 6 & -5 & 0 & 0 \\ 7 & 0 & -8 & 0 & 9 & 0 & 0 & -10 & 0 & -11 & 0 & -12 \\ 0 & 7 & 0 & 8 & 0 & 9 & -10 & 0 & 11 & 0 & -12 & 0 \\ 8 & 0 & 7 & 0 & 0 & 10 & 9 & 0 & 0 & 12 & 0 & -11 \\ 0 & -8 & 0 & 7 & 10 & 0 & 0 & 9 & 12 & 0 & 11 & 0 \end{bmatrix}$$

Thus, we have

$$\varphi(\boldsymbol{y}) = \begin{bmatrix} 1 & -2 & 0 & 0 & 0 & -3 & -4 & 0 & 0 & 0 & -5 & -6 \\ 2 & 1 & 0 & 0 & 3 & 0 & 0 & 4 & 0 & 0 & -6 & 5 \\ 0 & 0 & 1 & -2 & 4 & 0 & 0 & -3 & 5 & 6 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & -4 & 3 & 0 & 6 & -5 & 0 & 0 \\ 7 & 0 & -8 & 0 & 9 & 0 & 0 & -10 & 0 & -11 & 0 & -12 \\ 0 & 7 & 0 & 8 & 0 & 9 & -10 & 0 & 11 & 0 & -12 & 0 \\ 8 & 0 & 7 & 0 & 0 & 10 & 9 & 0 & 0 & 12 & 0 & -11 \\ 0 & -8 & 0 & 7 & 10 & 0 & 0 & 9 & 12 & 0 & 11 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix}$$

$$= \begin{bmatrix} -176 & 45 & 96 & 11 & -306 & -3 & 140 & 363 \end{bmatrix}^T.$$

Recall that

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} -176 + 45\boldsymbol{i} + 96\boldsymbol{j} + 11\boldsymbol{k} \\ -306 - 3\boldsymbol{i} + 140\boldsymbol{j} + 363\boldsymbol{k} \end{bmatrix}.$$

Similarly, using the multiplication by the right, we compute

$$\mathcal{M}_R(x) = \begin{bmatrix}1\\5\\9\end{bmatrix} \otimes \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} + \ldots + \begin{bmatrix}4\\8\\12\end{bmatrix} \otimes \begin{bmatrix}0&0&0&1\\0&0&-1&0\\0&1&0&0\\-1&0&0&0\end{bmatrix}.$$

Therefore, we have

$$\varphi^T(\boldsymbol{y}) = \begin{bmatrix}1&2&0&0&0&3&4&0&0&0&5&6\\7&0&8&0&9&0&0&10&0&11&0&12\end{bmatrix} \begin{bmatrix}1&2&3&4\\-2&1&-4&3\\-3&4&1&-2\\-4&-3&2&1\\5&6&7&8\\-6&5&-8&7\\-7&8&5&-6\\-8&-7&6&5\\9&10&11&12\\-10&9&-12&11\\-11&12&9&-10\\-12&-11&10&9\end{bmatrix}$$

$$= \begin{bmatrix}-176&45&96&11\\-306&-3&140&363\end{bmatrix}.$$

Vector-valued Least Squares Problem

Least squares problems play an important role in many machine learning techniques.

For example, training an extreme learning machine (ELMs) is achieved by solving a least squares problem (Huang et al., 2006, 2011; Lv and Zhang, 2018; Vieira and Valle, 2022).

In the following, we provide a framework for solving vector-valued least squares using real-valued matrix computation.

Given an ordered basis $\mathcal{E} = \{e_1, \ldots, e_n\}$ for $\mathbb{V}$, the Frobenius norm of a vector-valued matrix $\boldsymbol{A} \in \mathbb{V}^{M \times N}$ is defined by

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} |a_{ij}|^2},$$

where $|a_{ij}|$ denotes the absolute value of a_{ij}.

Equivalently, we have

$$\|\boldsymbol{A}\|_F = \|\varphi(\boldsymbol{A})\|_F.$$

Given matrices $\boldsymbol{A} \in \mathbb{V}^{M \times L}$ and $\boldsymbol{B} \in \mathbb{V}^{M \times N}$, the vector-valued least squares problem consists of finding the minimal Frobenius norm solution to the problem

$$\min \left\{ \|\boldsymbol{AX} - \boldsymbol{B}\|_F : \boldsymbol{X} \in \mathbb{V}^{L \times N} \right\}.$$

We solve a vector-valued least square problem using the real-valued matrices as follows:

$$\|\boldsymbol{AX} - \boldsymbol{B}\|_F = \|\varphi(\boldsymbol{AX} - \boldsymbol{B})\|_F = \|\varphi(\boldsymbol{AX}) - \varphi(\boldsymbol{B})\|_F$$
$$= \|\mathcal{M}_L(\boldsymbol{A})\varphi(\boldsymbol{X}) - \varphi(\boldsymbol{B})\|_F.$$

Therefore, the vector-valued least squares problem is rewritten as a real-valued problem:

$$\min\{\|\mathcal{M}_L(\boldsymbol{A})\boldsymbol{X}^{(r)} - \varphi(\boldsymbol{B})\|_F : \boldsymbol{X}^{(r)} \in \mathbb{R}^{(n+1)L\times N}\},$$

where $\boldsymbol{X}^{(r)} = \varphi(\boldsymbol{X})$.

The real-valued least square problem can be solved using the pseudoinverse:

$$\boldsymbol{X}^{(r)} = \mathcal{M}_L(\boldsymbol{A})^\dagger \varphi(\boldsymbol{B}),$$

where $\mathcal{M}_L(\boldsymbol{A})^\dagger$ is the pseudoinverse of $\mathcal{M}_L(\boldsymbol{A})$.

Concluding, the solution of the vector-valued least squares problem is given by

$$\boldsymbol{X} = \varphi^{-1}\left(\mathcal{M}_L(\boldsymbol{A})^{\dagger}\varphi(\boldsymbol{B})\right).$$

Example – Quaternions

Consider the quaternion-valued least square problem

$$\min\{\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_F : \boldsymbol{x} \in \mathbb{Q}^2\}.$$

where

$$\boldsymbol{A} = \begin{bmatrix} 1 + 2\boldsymbol{i} & 3\boldsymbol{i} + 4\boldsymbol{j} & 5\boldsymbol{j} + 6\boldsymbol{k} \\ 7 + 8\boldsymbol{j} & 9 + 10\boldsymbol{k} & 11\boldsymbol{i} + 12\boldsymbol{k} \end{bmatrix} \in \mathbb{Q}^{2\times 3}$$

and

$$\boldsymbol{b} = \begin{bmatrix} -176 + 45\boldsymbol{i} + 96\boldsymbol{j} + 11\boldsymbol{k} \\ -306 - 3\boldsymbol{i} + 140\boldsymbol{j} + 363\boldsymbol{k} \end{bmatrix} \in \mathbb{Q}^2.$$

In this case, we have

$$\mathcal{M}_L(\boldsymbol{A}) = \begin{bmatrix} 1 & -2 & 0 & 0 & 0 & -3 & -4 & 0 & 0 & 0 & -5 & -6 \\ 2 & 1 & 0 & 0 & 3 & 0 & 0 & 4 & 0 & 0 & -6 & 5 \\ 0 & 0 & 1 & -2 & 4 & 0 & 0 & -3 & 5 & 6 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & -4 & 3 & 0 & 6 & -5 & 0 & 0 \\ 7 & 0 & -8 & 0 & 9 & 0 & 0 & -10 & 0 & -11 & 0 & -12 \\ 0 & 7 & 0 & 8 & 0 & 9 & -10 & 0 & 11 & 0 & -12 & 0 \\ 8 & 0 & 7 & 0 & 0 & 10 & 9 & 0 & 0 & 12 & 0 & -11 \\ 0 & -8 & 0 & 7 & 10 & 0 & 0 & 9 & 12 & 0 & 11 & 0 \end{bmatrix}$$

and

$$\varphi(\boldsymbol{b}) = \begin{bmatrix} -176 & 45 & 96 & 11 & -306 & -3 & 140 & 363 \end{bmatrix}^T.$$

Thus, the solution of the least square problem satisfies

$$\phi(\boldsymbol{x}) = \mathcal{M}_L(\boldsymbol{A})^\dagger \varphi(\boldsymbol{b}) = \begin{bmatrix} -0.87 & -0.08 & 4.16 & \dots & 8.92 & 10.06 & 12.06 \end{bmatrix}^T.$$

Equivalently, we have

$$\boldsymbol{x} = \begin{bmatrix} -0.87 - 0.08\boldsymbol{i} + 4.16\boldsymbol{j} + 3.08\boldsymbol{k} \\ 5.66 + 7.75\boldsymbol{i} + 7.34\boldsymbol{j} + 7.47\boldsymbol{k} \\ 8.86 + 8.92\boldsymbol{i} + 10.06\boldsymbol{j} + 12.06\boldsymbol{k} \end{bmatrix}.$$

Concluding Remarks

In this lecture, we showed how to compute a product of vector-valued matrices using real-valued matrix operations.

Let $\mathcal{E} = \{e_1, \ldots, e_n\}$ be an ordered basis for an algebra $\mathbb{V}$.

Given matrices $\boldsymbol{A} \in \mathbb{V}^{M \times L}$ and $\boldsymbol{B} \in \mathbb{V}^{L \times N}$, the vector-valued product $\boldsymbol{C} = \boldsymbol{AB} \in \mathbb{V}^{M \times N}$ can be computed by means of the equations

$$\boldsymbol{C} = \varphi^{-1}\left(\mathcal{M}_L(\boldsymbol{A})\varphi(B)\right) \quad \text{or} \quad \boldsymbol{C} = \varphi^{-T}\left(\varphi^T(A)\mathcal{M}_R(\boldsymbol{B})\right),$$

where the matrices $\mathcal{M}_L(\boldsymbol{A})$ and $\mathcal{M}_R(\boldsymbol{B})$ are computed using the Kronecker product as follows:

$$\mathcal{M}_L(\boldsymbol{A}) = \sum_{k=1}^{n} A_k \otimes P_{k:}^T \quad \text{and} \quad \mathcal{M}_R(\boldsymbol{B}) = \sum_{k=1}^{n} B_k \otimes P_{:k}.$$

Moreover, the solution to the vector-valued least square problem

$$\min_{\boldsymbol{X}} \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|_F,$$

can be computed using the equation

$$\boldsymbol{X} = \varphi^{-1}\left(\mathcal{M}_L(A)^{\dagger}\varphi(B)\right),$$

where $\mathcal{M}_L(A)^{\dagger}$ denotes the pseudo-inverse of $\mathcal{M}_L(\boldsymbol{A})$.

Note that the presented approach generalizes the one detailed by Vieira and Valle (2022) from hypercomplex to arbitrary algebras.

Thanks for your attention!

Acknowledge

These slides are part of a mini-course given during the workshop on **hypercomplex-valued neural networks**, which took place at the **Institute for Research and Applications of Fuzzy Modeling**, **University of Ostrava**, Ostrava, Checz Republic, *06-10 February 2023*, with the support of

POLISH NATIONAL AGENCY FOR ACADEMIC EXCHANGE

My research on hypercomplex-valued neural networks has been partially supported by:

FAPESP and CNPq

Conselho Nacional de Desenvolvimento Científico e Tecnológico

References (1)

E. Grassucci, A. Zhang, and D. Comminiello. PHNNs: Lightweight Neural Networks via Parameterized Hypercomplex Convolutions. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 10 2022. ISSN 2162-237X. doi: 10.1109/TNNLS.2022.3226772. URL `https://ieeexplore.ieee.org/document/9983846/`.

G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

G.-B. Huang, D. Wang, and Y. Lan. Extreme learning machines: a survey. *Int. J. Machine Learning & Cybernetics*, 2(2):107–122, 2011.

C. F. Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 123(1-2):85–100, 11 2000. ISSN 0377-0427. doi: 10.1016/S0377-0427(00)00393-9.

References (2)

H. Lv and H. Zhang. Quaternion Extreme Learning Machine. In *Proceedings of ELM-2016*, pages 27–36. Springer, 2018.

F. Stenger. Kronecker Product Extensions of Linear Operators. *SIAM Journal on Numerical Analysis*, 5(2):422–435, 6 1968. ISSN 0036-1429. doi: 10.1137/0705033.

G. Vieira and M. E. Valle. A general framework for hypercomplex-valued extreme learning machines. *Journal of Computational Mathematics and Data Science*, 3:100032, 6 2022. ISSN 2772-4158. doi: 10.1016/J.JCMDS.2022.100032. URL https://linkinghub.elsevier.com/retrieve/pii/S2772415822000062.

References (3)

A. Zhang, Y. Tay, S. Zhang, A. Chan, A. T. Luu, S. C. Hui, and J. Fu. Beyond Fully-Connected Layers with Quaternions: Parameterization of Hypercomplex Multiplications with $1/n$ Parameters. 2 2021. doi: 10.48550/arxiv.2102.08597. URL https://arxiv.org/abs/2102.08597v1.