Quaternion-based Deep Belief Networks Fine-Tuning

João Paulo Papa[1], Gustavo H. Rosa[1], Danillo R. Pereira[1], Xin-She Yang[1]

[a]*Department of Computing, São Paulo State University, Av. Eng. Luiz Edmundo Carrijo Coube, 14-01, Bauru,SP, 17033-360, Brazil - Phone: +55-14-3103-6079*
papa@fc.unesp.br, gth.rosa@uol.com.br, cotonilo@gmail.com
[b]*School of Science and Technology, Middlesex University, NW4 4BT, United Kingdom - Phone: +44-020 8411-2351*
x.yang@mdx.ac.uk

Abstract

Deep learning techniques have been paramount in the last years, mainly due to their outstanding results in a number of applications. In this paper, we address the issue of fine-tuning parameters of Deep Belief Networks by means of meta-heuristics in which real-valued decision variables are described by quaternions. Such approaches essentially perform optimization in fitness landscapes that are mapped to a different representation based on hypercomplex numbers that may generate smoother surfaces. We therefore can map the optimization process onto a new space representation that is more suitable to learning parameters. Also, we proposed two approaches based on Harmony Search and quaternions that outperform the state-of-the-art results obtained so far in three public datasets for the reconstruction of binary images.

Keywords: Deep Belief Networks, Quaternion, Harmony Search

1. Introduction

In the last years, the need for a more insightful description of certain problems have fostered the research in the so-called deep learning techniques, which are based on the foundations of hierarchical brain processing. Therefore, one can stack single neural networks on top of each other to design a more complex model that can somehow learn different information

[1]Corresponding author.

Paper accepted on 22 June 2017 and available online on 1 July 2017.

at each layer. Although one can refer to a number of deep-driven related techniques out there, most of them rely on either Convolutional Neural Networks (CNNs) [17] or Restricted Boltzmann Machines (RBMs) [1]. In fact, RBMs can be stacked in several layers, and depending on the interactions among the layers, one can obtain a Deep Belief Network (DBN) [13] or a Deep Boltzmann Machine (DBM) [29].

Deep learning techniques have been often used as a feature learner, thus obtaining outstanding results in a number of computer vision- and signal processing-related applications, such as face [32] and object recognition, image reconstruction [3], speech recognition [12], and time-series modeling [16], among others [10], However, one can highlight some main shortcomings of such techniques: (i) since they usually have complex architectures, they are more prone to overfitting as well, and (ii) depending on the model, one can easily face hundreds of parameters to fine-tune, which can be an insurmountable problem. In this work, we focus on the latter issue.

Fine-tuning parameters in machine learning can be modeled as an optimization task, in which the fitness function is the effectiveness of the technique over some validating set. Albeit, to work on a high-dimensional search space computing derivatives towards the minimum of the function may not be a good choice [30]. In this context, meta-heuristic-based optimization techniques sound appealing, since most of them do not make use of derivatives in the search for better solutions, but they employ nature-inspired mechanisms instead.

Currently, the reader can refer to a very few works that deal with fine-tuning parameters in deep learning techniques, mainly because of the lack of communication among researchers from both sides. Fedorovici et al. [5], for instance, applied the Gravitational Search Algorithm to fine-tune CNNs for optical character recognition, and Papa et al. [21] employed the Harmony Search (HS) [9] to optimize the parameters of Restricted Boltzmann Machines in the context of binary image reconstruction. In the very same year, Papa et al. [22] evaluated some meta-heuristic techniques to fine-tune Discriminative Restricted Boltzmann Machines, which are basically a variant of naïve RBMs that consider the label units during the reconstruction process. Finally, Papa et al. [24], Rosa et al. [27, 28] and Rodrigues et al. [26] employed Harmony Search, Firefly Algorithm, Cuckoo Search and some of its variants to optimize Deep Belief Networks and Convolutional Neural Networks, respectively. Other works have focused on regularization-based approaches, such as the well-known Dropout [31] and DropConnect [35], which essen-

tially aim at inducing sparsity into the models, thus preventing them from overfitting.

As aforementioned, since one can find very few works that deal with both meta-heuristic optimization and deep learning, it is very easy to track them down and to state the approach proposed in this paper outperformed all previous works with respect to the task of binary image reconstruction using Deep Belief Networks. In this paper, we move the problem of parameter optimization from Euclidean-based representations to Quaternions, which are a powerful mathematical tool that extends the complex number theory and are extensively used for crystallographic texture analysis and aircraft dynamics, for instance. Inspired in the works of [7, 6], we propose two quaternion-based variants of the HS algorithm: the Quaternion Harmony Serch (QHS), and the Quaternion Improved Harmony Search (QIHS), being the former based on naïve Harmony Search, and the latter based on the Improved Harmony Search (IHS) [18]. Such variant allows HS to dynamically update its parameters, thus achieving better results. From another point of view, the approach proposed in this paper is similar to a tensor-based optimization, where each decision variable is mapped to a quaternion representation. Thus, the whole set of decision variables turns out to be a tensor.

Therefore, the main contributions of this paper are threefold: (i) to propose the Quaternion Harmony Search and (ii) the Quaternion Improved Harmony Search, and (iii) to model the task of fine-tuning parameters of Deep Belief Networks in the quaternion space. In this paper, we focused on the Harmony Search due to its efficiency during the optimization process, since it does not update all possible solutions in a single iteration, but only one.

The remainder of this paper is organized as follows. Section 2 presents theoretical details about quaternions, Harmony Search, and Deep Belief Networks, while Sections 3 and 4 discuss the methodology and experiments, respectively, and Section 5 states conclusions and future works.

2. Theoretical Background

2.1. Quaternion Algebra

Quaternions are mostly known to provide a convenient mathematical formulation to handle with orientations and rotations of objects in three-dimensional spaces. Their applications span from computer science, robotics, navigation, and flight dynamics, just to name a few. Quaternions with unit norm are also known as versors [25] hypercomplex numbers [14], which can

be used to rotate any other quaternion. Such operations of rotations are usually quite expensive by means of standard algebra, which makes quaternions even more appealing.

A quaternion q is composed of real and complex numbers, i.e., $q = x_0 + x_1 i + x_2 j + x_3 k$, where $x_0, x_1, x_2, x_3 \in \Re$ and i, j, k are imaginary numbers that follow the following set of equations:

$$ij = k, \quad jk = i, \quad ki = j, \tag{1}$$

$$ji = -k, \quad ji = -k, \quad kj = -i, \quad ik = -j, \tag{2}$$

and

$$i^2 = j^2 = k^2 = 1. \tag{3}$$

In short, a quaternion q is represented in a 4-dimensional space over the real numbers, i.e., $\Re^4$. Actually, depending on the application, we can consider the real numbers only, as the one addressed in this work.

Given two quaternions $q_1 = x_0 + x_1 i + x_2 j + x_3 k$ and $q_2 = y_0 + y_1 i + y_2 j + y_3 k$, the quaternion algebra defines a set of main operations [4]. The addition, for instance, can be defined by:

$$\begin{aligned} q_1 + q_2 &= (x_0 + x_1 i + x_2 j + x_3 k) + (y_0 + y_1 i + y_2 j + y_3 k) \\ &= (x_0 + y_0) + (x_1 + y_1) i + (x_2 + y_2) j + (x_3 + y_3) k, \end{aligned} \tag{4}$$

while the subtraction is defined as follows:

$$\begin{aligned} q_1 - q_2 &= (x_0 + x_1 i + x_2 j + x_3 k) - (y_0 + y_1 i + y_2 j + y_3 k) \\ &= (x_0 - y_0) + (x_1 - y_1) i + (x_2 - y_2) j + (x_3 - y_3) k. \end{aligned} \tag{5}$$

Another important operation is the norm, which maps a given quaternion to a real-valued number, as follows:

$$\begin{aligned} N(q_1) &= N(x_0 + x_1 i + x_2 j + x_3 k) \\ &= \sqrt{x_0^2 + x_1^2 + x_2^2 + x_3^2}. \end{aligned} \tag{6}$$

Finally, Fister et al. [7, 6] introduced two other operations, *qrand* and *qzero*. The former initializes a given quaternion with values drawn from a Gaussian distribution, and it can be defined as follows:

$$qrand() = \{x_i = \mathcal{N}(0,1) | i \in \{0,1,2,3\}\}. \quad (7)$$

The latter function initialized a quaternion with zero values, as follows:

$$qzero() = \{x_i = 0 | i \in \{0,1,2,3\}\}. \quad (8)$$

Although there are other operations, we defined only the ones employed in this work.

2.2. Harmony Search

Harmony Search is an optimization algorithm based on the way musicians create their songs. Roughly speaking, the idea is to model each decision variable as a musical instrument, and the optimization task is then reduced to find the set of instruments that generate a song with the "best" harmony as possible [8]. The notion of best is encoded as the fitness value of the function to be maximized/minimized.

Basically, the Harmony Search works by generating only one solution per iteration, the so-called "harmony", which is created based on two main principles: (i) values (musical notes/instruments) that have been used in the past, and (ii) new values generated at random that somehow encode the inspiration of the musician. In short, Harmony Search is quite simple and works well in several applications.

In this section, we first introduce the naïve HS to the reader, followed by the IHS background and the proposed QHS and QIHS.

2.2.1. Standard Harmony Search

Let $\mathcal{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_N)$ be a set of harmonies that compose the so-called "Harmony Memory", such that $\boldsymbol{h}_i \in \Re^M$. The HS algorithm generates after each iteration a new harmony vector $\hat{\boldsymbol{h}}$ based on memory considerations, pitch adjustments, and randomization (music improvisation). Further, the new harmony vector $\hat{\boldsymbol{h}}$ is evaluated in order to be accepted in the harmony memory: if $\hat{\boldsymbol{h}}$ is better than the worst harmony, the latter is then replaced by the new harmony.

In regard to the memory consideration step, the idea is to model the process of creating songs, in which the musician can use his/her memories to create a new song. This process is modelled by the Harmony Memory Considering Rate ($HMCR$) parameter, which is the probability of choosing one value from the historic values stored in the harmony memory, being

$(1 - HMCR)$ the probability of randomly choosing one feasible value, as follows:

$$\hat{h}^j \;=\; \begin{cases} h^j_A & \text{with probability HMCR} \\ \theta \in \varphi_j & \text{with probability (1-HMCR),} \end{cases} \tag{9}$$

where $A \sim \mathcal{U}(1, 2, \ldots, N)$, and $\boldsymbol{\varphi} = \{\varphi_1, \varphi_2, \ldots, \varphi_M\}$ stands for the set of feasible values for each decision variable.

Further, every component j of the new harmony vector $\hat{\boldsymbol{h}}$ is examined to determine whether it should be pitch-adjusted or not, which is controlled by the Pitch Adjusting Rate (PAR) variable, according to Equation 10:

$$\hat{h}^j \;=\; \begin{cases} \hat{h}^j \pm \beta\varrho & \text{with probability PAR} \\ \hat{h}^j & \text{with probability (1-PAR).} \end{cases} \tag{10}$$

The above equation concerns shifting the neighbouring values of some decision variable in the harmony, where ϱ is an arbitrary distance bandwidth, and $\beta \sim \mathcal{U}(0, 1)$.

2.2.2. Improved Harmony Search

The Improved Harmony Search [18] differs from traditional HS by updating the PAR and ϱ values dynamically. The PAR updating formulation at time step t is given by:

$$PAR^t = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{T} t, \tag{11}$$

where T stands for the number of iterations, and PAR_{min} and PAR_{max} denote the minimum and maximum PAR values, respectively. In regard to the bandwidth value at time step t, it is computed as follows:

$$\varrho^t = \varrho_{max} \exp\left(\frac{\ln(\varrho_{min}/\varrho_{max})}{T} t\right), \tag{12}$$

where ϱ_{min} and ϱ_{max} stand for the minimum and maximum values of ϱ, respectively.

Notice the PAR values (Equation 11) increase at a constant rate along the iterations. Essentially, the PAR variable is used to generate new solutions in the neighborhood of a given harmony (Equation 10). As the number of

iterations increases, meta-heuristic optimization algorithms tend to prevail the exploitation step, i.e. the search for better solutions are concentrated in even small (local) regions of the feature space. In case of IHS, by increasing the PAR values, one can increase the probability of generating new solutions as the number of iterations increases either. However, one should consider Equation 12 either. As the number of iterations increases, the step (variable ϱ) used to generating new solutions (Equation 10) also decreases.

2.2.3. Quaternion-based Harmony Search

The proposed approaches aim at mapping the problem of optimizing variables in the Euclidean space to the quaternions space. As aforementioned, the idea is to obtain smoother representations of the fitness landscape, thus making the problem easier to handle.

In the standard Harmony Search, each harmony $\boldsymbol{h}_i \in \Re^M$, $i = 1, 2, \ldots, N$ is modeled as an array containing M variables to be optimized, such that $h_{ij} \in \Re$. In both QHS and QIHS, each decision variable j is now represented by a quaternion $q_j \in \Re^4$, such that each harmony can be seen as a matrix $\boldsymbol{h}'_i \in \Re^{4\times N}$. Therefore, each harmony is no longer an array of decision variables, but a matrix instead.

However, we can map each quaternion to a real-valued number in order to use standard HS/IHS. Basically, one has to compute $h_{ij} = N(h'_{ij})$, $i = 1, 2 \ldots, N$ and $j = 1, 2, \ldots, M$. Further, the standard HS/IHS procedure can be executed as usual. But note the optimization process is conducted at quaternion level, which means QHS/QIHS aims finding the quaternions for each decision variable such that their norm values minimize the fitness function. An ordinary HS/IHS aims at learning values for each decision variable that minimizes the fitness function. More details about the quaternion-based optimization process can be found in Section 3.2.

2.3. Deep Belief Networks

In this section, we describe the main concepts related to Deep Belief Networks, but with a special attention to the theoretical background of RBMs, which are the basis for DBN understanding.

2.3.1. Restricted Boltzmann Machines

Restricted Boltzmann Machines are usually referred to as energy-based stochastic neural networks composed of two layers of neurons, in which the learning phase is conducted by means of an unsupervised fashion. Figure 1

depicts the architecture of a Restricted Boltzmann Machine, which comprises a visible layer $\mathbf{v}$ with m units and a hidden layer $\mathbf{h}$ with n units. The real-valued $m \times n$ matrix $\mathbf{W}$ models the weights between visible and hidden neurons, where w_{ij} stands for the weight between the visible unit v_i and the hidden unit h_j.

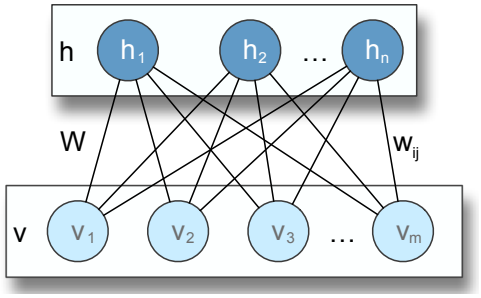


Figure 1: The RBM architecture.

Let us assume $\mathbf{v}$ and $\mathbf{h}$ as the binary visible and hidden units, respectively. In other words, $\mathbf{v} \in \{0,1\}^m$ and $\mathbf{h} \in \{0,1\}^n$. The energy function of a Restricted Boltzmann Machine is given by:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} v_i h_j w_{ij}, \tag{13}$$

where $\mathbf{a}$ and $\mathbf{b}$ stand for the biases of visible and hidden units, respectively. The probability of a configuration $(\mathbf{v},\mathbf{h})$ is computed as follows:

$$P(\mathbf{v},\mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}, \tag{14}$$

being the denominator a normalization factor that stands for all possible configurations involving the visible and hidden units. In short, the RBM learning algorithm aims at estimating $\mathbf{W}$, $\mathbf{a}$ and $\mathbf{b}$.

2.3.2. Learning Algorithm

The parameters of an RBM can be optimized by performing stochastic gradient ascent on the log-likelihood of training patterns. Given a training sample (visible unit), its probability is computed over all possible hidden vectors, as follows:

$$P(\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}. \tag{15}$$

In order to update the weights and biases, it is necessary to compute the derivatives of the logarithm of $P(v)$ with respect to w_{ij}, a_i and b_i, thus leading to the following equations:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \underbrace{\eta(P(\mathbf{h}|\mathbf{v})\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T) - \lambda\mathbf{W}^t + \alpha\Delta\mathbf{W}^{t-1}}_{=\Delta\mathbf{W}^t}, \tag{16}$$

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \underbrace{\eta(\mathbf{v} - \tilde{\mathbf{v}}) + \alpha\Delta\mathbf{a}^{t-1}}_{=\Delta\mathbf{a}^t} \tag{17}$$

and

$$\mathbf{b}^{t+1} = \mathbf{b}^t + \underbrace{\eta(P(\mathbf{h}|\mathbf{v}) - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})) + \alpha\Delta\mathbf{b}^{t-1}}_{=\Delta\mathbf{b}^t}, \tag{18}$$

where

$$P(h_j = 1|\mathbf{v}) = \sigma\left(\sum_{i=1}^{m} w_{ij}v_i + b_j\right), \tag{19}$$

and

$$P(v_i = 1|\mathbf{h}) = \sigma\left(\sum_{j=1}^{n} w_{ij}h_j + a_i\right). \tag{20}$$

In this case, $\sigma(\cdot)$ stands for the logistic sigmoid function. Notice $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{h}}$ stand for the reconstructed (sampled) versions of the input $\mathbf{v}$ and hidden units $\mathbf{h}$, respectively.

2.3.3. Stacked Restricted Boltzmann Machines

Truly speaking, DBNs are composed of a set of stacked RBMs, being each of them trained using the learning algorithm presented in Section 2.3.2 in a greedy fashion, which means an RBM at a certain layer does not consider

others during its learning procedure. Suppose we have a DBN composed of L layers, being $\mathbf{W}^i$ the weight matrix of RBM at layer i. Additionally, the hidden units at layer i become the input units to the layer $i+1$.

The approach proposed by [13] for the training step of DBNs also considers a fine-tuning as a final step after the training of each RBM. Such procedure can be performed by means of a Backpropagation or Gradient descent algorithm, for instance, in order to adjust the matrices $\mathbf{W}^i$, $i = 1, 2, \ldots, L$. The optimization algorithm aims at minimizing some error measure considering the output of an additional layer placed at the top of the DBN after its former greedy training. Such layer is often composed of softmax or logistic units, or even some supervised pattern recognition technique.

Figure 2 depicts an example of a DBN with three layers (the same used in the experimental section). In this context, we have $\mathbf{W} = \{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3\}$ as the set of weight matrices, and $\mathbf{h} = \{\mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3\}$ as the set of hidden layers. In this case, $\mathbf{W}^i$ stands for the weight matrix between layers i and $i+1$, and $\mathbf{h}^j$ denotes the j^{th} hidden layer. One can observe at every two layers we have a standard RBM, which is trained greedily, i.e. we do not consider anything else other than the two layers that compose the current RBM during the learning procedure.

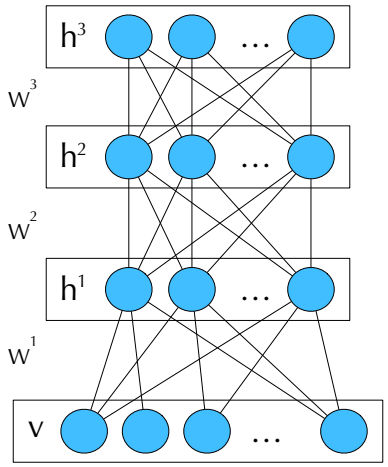


Figure 2: The DBN architecture used in this work.

3. Methodology

In this section, we describe the datasets, experimental setup and the proposed approach to model DBN fine-tuning by means of quaternions.

3.1. Datasets

We validate DBN fine-tuning in the task of binary image reconstruction over three public datasets, as described below:

- MNIST dataset[1]: it is composed of images of handwritten digits. The original version contains a training set with $60,000$ images from digits '0'-'9', as well as a test set with $10,000$ images. We preprocess the data according to the methodology used Papa et al. [24].

- CalTech 101 Silhouettes Data Set[2]: it is based on the former Caltech 101 dataset, and it comprises silhouettes of images from 101 classes with resolution of 28×28. Notice we used only the training and test sets, since our optimization model aims at minimizing the MSE error over the training set.

- Semeion Handwritten Digit Data Set[3]: this dataset contains 1,593 binary images of manuscript digits with resolution of 16×16 from around 80 persons. We employed the whole dataset in the experimental section. In regard to this dataset, we used 2% for training and the remaining 98% for testing purposes.

Figure 3 displays some training examples from the aforementioned datasets, which were partitioned in 2% for the training set and 98% to compose the test set.

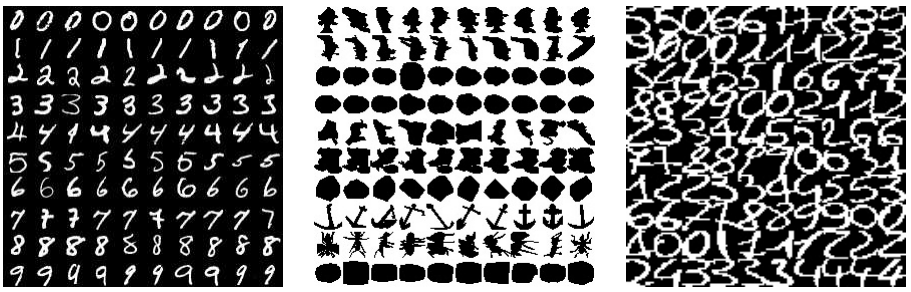
3.2. DBN Fine-Tuning as an Optimization Problem

Usually, Restricted Boltzmann Machines require the setting up of four main parameters: the learning rate η, number of hidden units n, momentum φ and weight decay λ. Since Deep Belief Nets stack RBMs on top of each

[1] `http://yann.lecun.com/exdb/mnist/`

[2] `https://people.cs.umass.edu/~marlin/data.shtml`

[3] `https://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit`



(a) (b) (c)

Figure 3: Example images from (a) MNIST, (b) CalTech 101 Silhouettes and (c) Semeion datasets.

other, if one has L layers, then each harmony encodes $4L$ variables to be optimized. However, as the training procedure of DBNs are greedy-wise, which means each layer is trained independently, only 4 variables are optimized per layer.

In this work, we initialized all harmonies with random values for the further application of the optimization techniques. In this work, we used the following ranges concerning the parameters: $n \in [5, 100]$, $\eta \in [0.1, 0.9]$, $\lambda \in [0.1, 0.9]$ and $\varphi \in [0.00001, 0.01]$. In order to fulfill the requirements of any optimization technique, one shall design a fitness function to guide the search into the best solutions. In this paper, we used the mean squared error (MSE) over the training set considering the task of binary image reconstruction as the fitness function. Therefore, we adopted the very same methodology used by Papa et al. [24] to allow a fair comparison against the works. Figure 4 depicts the model used in this paper to encode a standard optimization problem on each harmony. As aforementioned, although we have a $4L$-dimensional harmony, only 4 variables are used at time. In regard to the source-code, we used an implementation provided by LibOPT [23].

The approach proposed in this paper models each harmony as a tensor, since each decision variable is now a 4-dimensional quaternion. Therefore, the problem of optimizing each decision variable (real-valued number) is now mapped to a problem of optimizing a 4-dimensional vector in the quaternions space. Figure 5 illustrates the above situation.

Although the number of hidden units (n) is integer-valued, Harmony Search does not have equations based on velocity and other mechanisms often employed by swarm-based optimization techniques. Since each decision

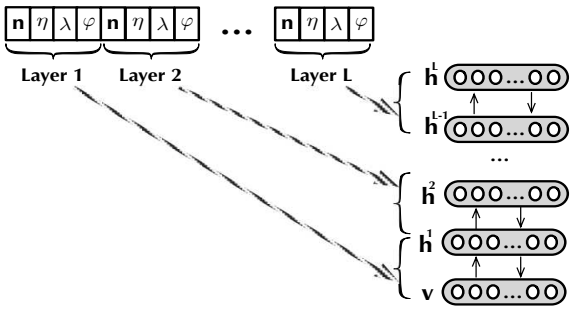


Figure 4: Proposed approach to encode the decision variables on each harmony.

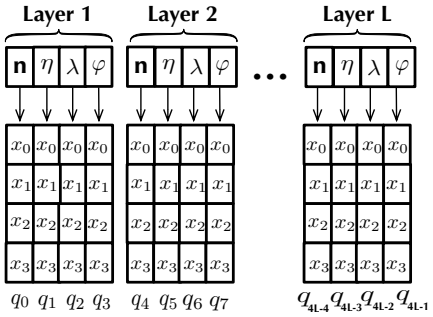


Figure 5: Quaternion-based encoding process.

variable that composes the harmony is initialized within its predefined range, when we create a new harmony we need to go through Equation 9 and further Equation 10. The first equation just generates a new random value within the range of each decision variable with probability $1 - HMCR$, or it simply takes a value from the harmony memory with probability $HMCR$. Therefore, Equation 9 does work within the range of each decision variable. With respect to Equation 10, an integer number can be mapped to a real one with probability PAR. In this case, we just map that number to its nearest integer (round operation).

3.3. Optimization Techniques

We compared the proposed QHS and QIHS against a random initialization of the parameters (RS), Particle Swarm Optimization (PSO) [15, 36], and the Hyperopt library using random search (Hyper-RS) and Tree of

Parzen Estimators (Hyper-TPE) [2]. Additionally, we evaluated five HS variants: (i) IHS, (ii) Global-best Harmony Search (GHS) [19], (iii) Novel Global Harmony Search (NGHS) [38], (iv) Self-adaptive Global Best Harmony Search (SGHS) [20] and (v) Parameter-setting-free Harmony Search (PSF-HS) [9]. We also evaluated the robustness of parameter fine-tuning using three distinct DBN models: one layer (1L), two layers (2L) and three layers (3L). Notice the 1L approach stands for the standard RBM.

Finally, Table 1 presents the parameters used for each optimization technique according to [24]. We used 5 agents (initial solutions) for all optimization techniques during 50 iterations for convergence. The p_m and LP parameters stand for the probability of mutation and learning period of NGHS and SGHS techniques, respectively.

Table 1: Parameter configuration.

Technique	Parameters
PSO	$c_1 = 1.7$, $c_2 = 1.7$, $w = 0.7$
HS/QHS	$HMCR = 0.7$, $PAR = 0.7$, $\varrho = 10$
IHS/QIHS	$HMCR = 0.7$, $PAR_{MIN} = 0.1$ $PAR_{MAX} = 0.7$, $\varrho_{MIN} = 1$ $\varrho_{MAX} = 10$
GHS	$HMCR = 0.7$, $PAR_{MIN} = 0.1$ $PAR_{MAX} = 0.7$
NGHS	$p_m = 0 \cdot t$
SGHS	$HMCR_m = 0.98$, $PAR_m = 0.9$ $\varrho_{MIN} = 0$, $\varrho_{MAX} = 0.9$, $LP = 5$

We have conducted a hold-out procedure with 20 randomly generated training and test sets, 10 iterations for the learning procedure of each RBM, and mini-batches of size 20. In addition, we also considered three learning algorithms: Contrastive Divergence (CD) [11], Persistent Contrastive Divergence (PCD) [33] and Fast Persistent Contrastive Divergence (FPCD) [34]. Finally, the Wilcoxon signed-rank test [37] with significance of 0.05 was used for statistical validation purposes.

4. Experiments

In this section, we present the experimental results concerning the task of binary image reconstruction. Tables 2, 3 and 4 present the average values of

the minimum squared errors over the MNIST, CalTech 101 Silhouettes and Semeion Handwritten Digit datasets (testing set), respectively. The values in bold stand for the best results considering the Wilcoxon test.

Clearly, one can observe QHS and QIHS obtained the best results so far for all datasets, thus outperforming the recent results obtained by [24] with IHS optimization technique. Notice no significant difference among the number of layers or the training algorithm could be observed. That is an interesting point, since it is expected that more layers would be helpful to a better description of the images. In fact, such behaviour can be observed for some HS-based techniques, such as IHS and HS-PSF considering MNIST dataset. Additionally, PSO seemed to behave similarly, since the best results were obtained with one layer only.

In regard to Semeion Handwritten Digit dataset, which poses a greater challenge (i.e., we obtained higher reconstruction errors), the more layers one has, the lower the reconstruction errors are for all optimization techniques, except for QHS and QIHS. Probably, as the proposed approaches converge faster than traditional ones, the number of layers may not be so important, since we restricted the convergence process to only 10 iterations. Regarding this dataset, the best results excluding QHS and QIHS were obtained by PSO, Hyper-RS and Hyper-TPE. As a matter of fact, it might be expected that swarm-based optimization techniques converge faster, since they share new solutions among all particles, thus updating them all. We have observed PSO results are more competitive when the problem gets more complex, such as in Caltech 101 and Semeion Handwritten Digit dataset. That leads us to think about quaternion-based versions of Particle Swarm Optimization either, which seems to be fruitful to the research community.

Although QHS and QIHS obtained the lowest errors over Caltech 101 Silhouettes dataset, IHS achieved similar results if we consider the statistical evaluation. In fact, IHS with FPCD and 3 layers obtained results quite close to the ones obtained by QHS/QIHS with CD and 1 layer only. That is an interesting point, since it shows us that QHS/QIHS can achieve similar or even better results than HS-based techniques with less layers, i.e. with a simpler neural architecture.

Usually, FPCD takes longer to converge [34]. As we have used the very same number of iterations for convergence (i.e., 10 iterations), it is expected FPCD did not obtain similar results to those achieved by both CD and PCD. Figure 6 displays the convergence curve (MSE) of QHS and QIHS during the first execution of the cross-validation procedure for the first layer

Table 2: Average MSE values considering MNIST dataset.

	1L			2L			3L		
	CD	PCD	FPCD	CD	PCD	FPCD	CD	PCD	FPCD
HS [24]	0.1059	0.1325	0.1324	0.1059	0.1061	0.1057	0.1059	0.1058	0.1057
IHS [24]	0.0903	0.0879	0.0882	0.0885	0.0886	0.0886	0.0887	0.0885	0.0886
GHS [24]	0.1063	0.1062	0.1063	0.1061	0.1063	0.1061	0.1063	0.1065	0.1062
NGHS [24]	0.1066	0.1066	0.1063	0.1065	0.1062	0.1062	0.1069	0.1064	0.1062
SGHS [24]	0.1067	0.1067	0.1062	0.1072	0.1066	0.1063	0.1068	0.1065	0.1064
PSF-HS [24]	0.1005	0.1006	0.0998	0.1032	0.0976	0.1007	0.0992	0.0995	0.0998
PSO	0.1057	0.1058	0.1057	0.1060	0.1059	0.1058	0.1058	0.1059	0.1058
RS [24]	0.1105	0.1101	0.1102	0.1105	0.1101	0.1096	0.1108	0.1099	0.1096
Hyper-RS [24]	0.1062	0.1062	0.1060	0.1062	0.1062	0.1060	0.1062	0.1061	0.1062
Hyper-TPE [24]	0.1059	0.1059	0.1058	0.1059	0.1059	0.1057	0.1050	0.1051	0.1051
QHS	**0.0876**	**0.0876**	0.0899	**0.0876**	**0.0876**	0.0901	**0.0876**	**0.0876**	0.0918
QIHS	**0.0876**	**0.0876**	0.0888	**0.0876**	**0.0876**	0.0882	0.0888	**0.0876**	0.0888

Table 3: Average MSE values considering CalTech 101 Silhouettes dataset.

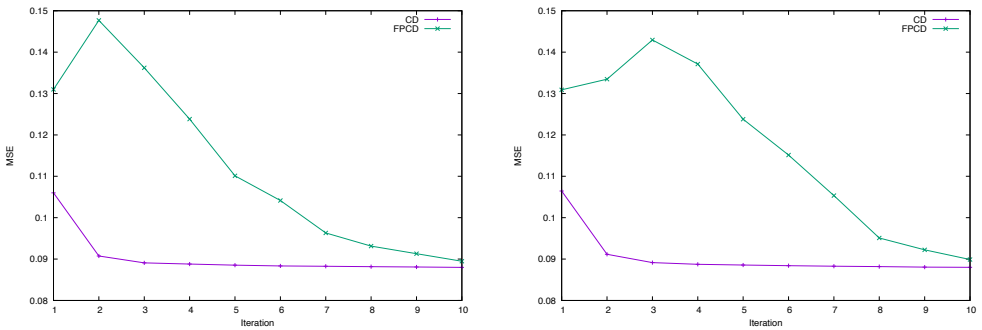
	1L			2L			3L		
	CD	PCD	FPCD	CD	PCD	FPCD	CD	PCD	FPCD
HS [24]	0.1695	0.1696	0.1691	0.1695	0.1699	0.1693	0.1694	0.1696	0.1692
IHS [24]	0.1696	0.1695	0.1693	**0.1609**	**0.1607**	**0.1612**	**0.1611**	**0.1618**	**0.1606**
GHS [24]	0.1699	0.1697	0.1692	0.1699	0.1698	0.1695	0.1697	0.1696	0.1694
NGHS [24]	0.1706	0.1703	0.1697	0.1697	0.1703	0.1694	0.1701	0.1699	0.1695
SGHS [24]	0.1703	0.1703	0.1701	0.1709	0.1706	0.1700	0.1708	0.1703	0.1701
PSF-HS [24]	0.1663	0.1670	0.1670	0.1689	0.1691	0.1681	0.1675	0.1684	0.1686
PSO	0.1691	0.1690	0.1689	0.1689	0.1691	0.1688	0.1692	0.1692	0.1690
RS [24]	0.1755	0.1759	0.1743	0.1758	0.1755	0.1748	0.1766	0.1766	0.1742
Hyper-RS [24]	0.1696	0.1697	0.1694	0.1662	0.1662	0.1695	0.1652	0.1651	0.1650
Hyper-TPE [24]	0.1694	0.1693	0.1691	0.1693	0.1693	0.1691	0.1649	0.1642	0.1642
QHS	**0.1605**	**0.1606**	0.1616	**0.1605**	**0.1605**	0.1616	**0.1605**	**0.1605**	0.1615
QIHS	**0.1605**	**0.1606**	0.1626	**0.1605**	**0.1605**	0.1624	**0.1605**	**0.1605**	0.1613

considering MNIST dataset, CD and FPCD training algorithms. Although both techniques tend to converge when one increases the number of iterations, CD clearly converges faster than FPCD for both QHS and QIHS. Notice we are not using too many learning iterations, which is often required by FPCD. Once more, the idea of this paper is to show we can fine-tune DBNs properly with a limited number of iterations.

Another way of comparing quaternion-based and standard HS techniques

Table 4: Average MSE values considering Semeion Handwritten Digit dataset.

	1L			2L			3L		
	CD	PCD	FPCD	CD	PCD	FPCD	CD	PCD	FPCD
HS [24]	0.2128	0.2128	0.2129	0.2202	0.2128	0.2128	0.2199	0.2128	0.2128
IHS [24]	0.2131	0.2130	0.2128	0.2116	0.2114	0.2121	0.2103	0.2109	0.2119
GHS [24]	0.2133	0.2129	0.2128	0.2129	0.2130	0.2129	0.2129	0.2129	0.2128
NGHS [24]	0.2134	0.2132	0.2131	0.2130	0.2131	0.2129	0.2131	0.2132	0.2130
SGHS [24]	0.2135	0.2131	0.2130	0.2131	0.2131	0.2130	0.2132	0.2132	0.2130
PSF-HS [24]	0.2137	0.2130	0.2130	0.2121	0.2120	0.2124	0.2120	0.2120	0.2121
PSO	0.2128	0.2128	0.2128	0.2128	0.2128	0.2128	0.2128	0.2128	0.2127
RS [24]	0.2146	0.2143	0.2145	0.2146	0.2144	0.2139	0.2143	0.2140	0.2140
Hyper-RS [24]	0.2127	0.2129	0.2129	0.2129	0.2129	0.2129	0.2129	0.2129	0.2128
Hyper-TPE [24]	0.2128	0.2128	0.2128	0.2128	0.2128	0.2127	0.2128	0.2128	0.2128
QHS	**0.2095**	**0.2096**	0.2143	**0.2096**	**0.2096**	0.2142	**0.2096**	**0.2096**	0.2170
QIHS	**0.2096**	**0.2096**	**0.2096**	**0.2096**	**0.2159**	0.1624	**0.2096**	**0.2096**	0.2132



(b)

Figure 6: MSE values during the convergence process considering (a) QHS and (b) IQHS optimization algorithms for MNIST dataset.

is to consider the values of the logarithm of the pseudo-likelihood (PL) when estimating the difference between the input training pattern and the output sampled by the DBN learning algorithm. Since the idea is to maximize the logarithm of PL, the higher these values, the better the technique is. Figure 7 displays a comparison among HS, IHS, QHS and QIHS considering MNIST dataset at the very first layer of the first execution of the cross-validation procedure using CD. Clearly, the proposed techniques can obtain much better values for the logarithm of pseudo-likelihood, which means they can learn

more accurate models than standard HS-based optimization techniques.

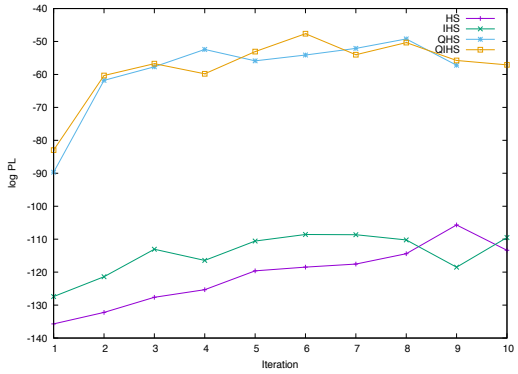


Figure 7: Logarithm of the pseudo-likelihood over MNIST dataset considering HS, IHS, QHS and QIHS optimization techniques.

Probably, one main shortcoming of quaternion-based optimization concerns with the computational load, which is usually, at least, twice more expensive than traditional techniques. The problem is related to the computation of the norm for each decision variable, which requires a loop with 4 more iterations.

5. Conclusions

This paper dealt with the problem of DBN parameter fine-tuning by means of quaternion-based optimization techniques. The idea is to map the traditional Euclidean space, commonly used by most of optimization techniques, to a probably smoother fitness landscape on the quaternions space. The two proposed approaches, i.e. QHS and QIHS, showed to be more effective concerning the task of binary image reconstruction than standard techniques based on Harmony Search, thus outperforming very recent state-of-the-art results. However, one main shortcoming concerns with their computational load, which is at least twice more expensive than traditional techniques.

In regard to future works, we intend to implement different variants of the Harmony Search to work on the quaternions space, as well as we aim at finding different search space representations. Another approach would be to consider more than one space during the optimization process in order to learn a hybrid manifold that can benefit from different representations.

Acknowledgments

The authors are grateful to FAPESP grants #2014/12236-1, #2014/16250-9 and #2015/25739-4, as well as CNPq grant #470571/2013-6 and #306166/2014-3. The authors also thank Prof. André Nunes de Souza and Prof. Danilo Sinkiti Gastaldello for their insightful comments.

References

[1] D. H. Ackley, G. E. Hinton, and T. J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] J. S. Bergstra, D. Yamins, and D. D. Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Python for Scientific Computing Conference*, pages 1–7, 2013.

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *13th European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 184–199. Springer International Publishing, 2014.

[4] D. Eberly. Quaternion algebra and calculus. Technical report, Magic Software, 2002.

[5] L. Fedorovici, R. Precup, F. Dragan, R. David, and C. Purcaru. Embedding gravitational search algorithms in convolutional neural networks for OCR applications. In *7th IEEE International Symposium on Applied Computational Intelligence and Informatics*, SACI '12, pages 125–130, 2012.

[6] I. Fister, J. Brest, I. Fister Jr., and X.-S. Yang. Modified bat algorithm with quaternion representation. In *IEEE Congress on Evolutionary Computation*, pages 491–498, 2015.

[7] I. Fister, X.-S. Yang, J. Brest, and I. Fister Jr. Modified firefly algorithm using quaternion representation. *Expert Systems with Applications*, 40(18):7220–7230, 2013.

[8] Z. W. Geem. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[9] Z. W. Geem and K.-B. Sim. Parameter-setting-free harmony search algorithm. *Applied Mathematics and Computation*, 217(8):3881–3889, 2010.

[10] X. Han and Q. Dai. Batch-normalized mlpconv-wise supervised pre-training network in network. *Applied Intelligence*, 2017.

[11] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[12] G. E. Hinton, D. Li, Y. Dong, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[13] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[14] I. L. Kantor and A. S. Solodovnikov. *Hypercomplex numbers*. Springer-Verlag, 1989.

[15] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, USA, 2001.

[16] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42(1):11–24, 2014.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2):1567–1579, 2007.

[19] M. G. H. Omran and M. Mahdavi. Global-best harmony search. *Applied Mathematics and Computation*, 198(2):643–656, 2008.

[20] Q.-K. Pan, P. N. Suganthan, M. Fatih Tasgetiren, and J. J. Liang. A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation*, 216(3):830–848, 2010.

[21] J. P. Papa, G. H. Rosa, K. A. P. Costa, A. N. Marana, W. Scheirer, and D. D. Cox. On the model selection of bernoulli restricted boltzmann machines through harmony search. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '15, pages 1449–1450, New York, USA, 2015. ACM.

[22] J. P. Papa, G. H. Rosa, A. N. Marana, W. Scheirer, and D. D. Cox. Model selection for discriminative restricted boltzmann machines through meta-heuristic techniques. *Journal of Computational Science*, 9:14–18, 2015.

[23] J. P. Papa, G. H. Rosa, D. Rodrigues, and X.-S. Yang. LibOPT: An open-source platform for fast prototyping soft optimization techniques. *ArXiv e-prints*, 2017. http://adsabs.harvard.edu/abs/2017arXiv170405174P.

[24] J. P. Papa, W. Scheirer, and D. D. Cox. Fine-tuning deep belief networks using harmony search. *Applied Soft Computing*, 46:875–885, 2016.

[25] C. Perwass. *Geometric Algebra with Applications in Engineering*. Springer, 2009.

[26] D. Rodrigues, X.-S. Yang, and J.P. Papa. Fine-tuning deep belief networks using cuckoo search. In X.-S. Yang and J.P. Papa, editors, *Bio-Inspired Computation and Applications in Image Processing*, pages 47–59. Academic Press, 2016.

[27] G. H. Rosa, J. P. Papa, K. A. P. Costa, L. A. Passos, C. R. Pereira, and X.-S. Yang. *Learning Parameters in Deep Belief Networks Through Firefly Algorithm*, pages 138–149. Springer International Publishing, Cham, 2016.

[28] G. H. Rosa, J. P. Papa, A. N. Marana, W. Scheirer, and D. D. Cox. Fine-tuning convolutional neural networks using harmony search. In A. Pardo and J. Kittler, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, volume 9423 of *Lecture Notes in Computer Science*, pages 683–690. Springer International Publishing, 2015. 20th Iberoamerican Congress on Pattern Recognition.

[29] R.Salakhutdinov and G. E. Hinton. An efficient learning procedure for deep boltzmann machines. *Neural Computation*, 24(8):1967–2006, 2012.

[30] D. Schuurmans and M. P. Wellman, editors. *Derivative-Free Optimization via Classification*. AAAI Press, 2016.

[31] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, January 2014.

[32] Y. Taigman, Y. Ming, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1701–1708, 2014.

[33] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1064–1071, New York, USA, 2008. ACM.

[34] T. Tieleman and G. E. Hinton. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1033–1040, New York, USA, 2009. ACM.

[35] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the 30th International Conference on Machine learning*, volume 28 of *ICML '13*, pages 1058–1066. JMLR Workshop and Conference Proceedings, 2013.

[36] Y. Zhang S. Wang and G. Ji. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015(2015):1–38, 2015.

[37] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[38] D. Zou, L. Gao, J. Wu, S. Li, and Y. Li. A novel global harmony search algorithm for reliability problems. *Computers & Industrial Engineering*, 58(2):307–316, 2010.