Applied Soft Computing 46 (2016) 875–885

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Fine-tuning Deep Belief Networks using Harmony Search

João Paulo Papa[a,*], Walter Scheirer[b], David Daniel Cox[b]

[a] UNESP – Univ Estadual Paulista, Department of Computing, Bauru, Brazil
[b] Harvard University, Center for Brain Science, Cambridge, USA

ARTICLE INFO

Article history:
Received 29 May 2015
Received in revised form 15 August 2015
Accepted 25 August 2015
Available online 16 September 2015

Keywords:
Restricted Boltzmann Machines
Deep Belief Networks
Harmony Search
Meta-heuristics

ABSTRACT

In this paper, we deal with the problem of Deep Belief Networks (DBNs) parameters fine-tuning by means of a fast meta-heuristic approach named Harmony Search (HS). Although such deep learning-based technique has been widely used in the last years, more detailed studies about how to set its parameters may not be observed in the literature. We have shown we can obtain more accurate results comparing HS against with several of its variants, a random search and two variants of the well-known Hyperopt library. The experimental results were carried out in two public datasets considering the task of binary image reconstruction, three DBN learning algorithms and three layers.

1. Introduction

Machine learning techniques have been actively pursued in the last years, since the number of applications that require some intelligent decision-making process is growing faster every year. An interesting branch of pattern recognition techniques related to deep learning has attracted a considerable attention in the last years [3], since their prominent results have established a hallmark for several applications, such as face and speech recognition, among others.

Roughly speaking, deep learning algorithms are modelled by means of several layers of a predefined set of operations. In regard to Convolutional Neural Networks, for instance, such operations may involve data preprocessing, convolution kernels and non-linear functions [17]. If we consider Deep Boltzmann Machines [25], which address layers of stacked Restricted Boltzmann Machines (RBMs), the input data is mapped to a set of hidden units by means of Gibbs sampling for further data reconstruction. Soon after, the output of an RBM is used to feed the upper RBM, being the sampling-reconstruction process repeated over again until the top layer.

Restricted Boltzmann Machines have attracted considerable attention in the last years due to their simplicity, high level of parallelism and strong representation ability [13]. RBMs can be interpreted as stochastic neural networks, being mainly used for image reconstruction and collaborative filtering through unsupervised learning [2]. Later on, some works attempted to develop supervised versions of Restricted Boltzmann Machines to work as sole classifiers, since RBMs have been used, essentially, for feeding supervised classifiers so far. One of such versions is the so-called Discriminative Restricted Boltzmann Machine (DRBM) [16], which now considers the label information during the learning process, which allows it to be used for classification purposes. Recent works have focused on RBMs in the context of classifier combination [32] and spectral classification in astronomy [8], as well as a very interesting review about training algorithms for RBMs has been provided by Fischer and Igel [7].

Aiming at providing a more powerful and discriminative ability for learning features with RBMs, Hinton et al. [11] presented a deep learning-oriented approach called Deep Belief Networks (DBNs), which can be seen as a set of stacked RBMs that encode a different amount of information at each layer. Soon after, a considerable amount of works have been devoted to employ DBNs in a broad range of applications, varying from natural language processing [26] to image classification [33], just to name a few.

However, one of the main shortcomings of RBMs and DBNs concerns with the proper selection of their parameters, i.e., the number of hidden units, training iterations (epochs) and learning rate, among others. Although a very useful training guide has been provided by Hinton [13], there is a need for a manual inspection of the algorithm's convergence. Yosinski and Lipson [31], for instance, highlighted some approaches for visualizing the behaviour of an RBM during its learning procedure. The authors also argued the RBM training algorithm is far from a straightforward approach.

The task of model selection in machine learning techniques aims at finding a suitable set of parameters that maximizes some fitness

* Corresponding author.
E-mail addresses: papa@fc.unesp.br (J.P. Papa), wscheirer@fas.harvard.edu (W. Scheirer), davidcox@fas.harvard.edu (D.D. Cox).

http://dx.doi.org/10.1016/j.asoc.2015.08.043
1568-4946/

function, such as the clustering quality in unsupervised approaches, or a classifier's recognition accuracy when dealing with supervised problems. Meta-heuristic techniques are among the most used for optimization problems, since they provide simple and elegant solutions in a wide range of applications. Such techniques may comprise swarm- and population-based algorithms, as well as stochastic and other nature-inspired solutions.

Although some swarm- and population-based optimization algorithms have obtained very promising results in several applications, they may suffer from a high computational burden in large-scale problems, since there is a need for optimizing all agents at each iteration. Some years ago, Geem [9] proposed the Harmony Search (HS) technique, which falls in the field of meta-heuristic optimization techniques. Basically, the idea of Harmony Search is to solve optimization problems based on the way the musicians compose a song with optimal harmony: the decision variables stand for each musician, and the problem itself corresponds to a music. The combination of musical notes that takes the music more harmonious is the one which maximizes some fitness function.

However, the reader may face just a few and very recent works that handle the problem of RBM model selection by means of meta-heuristic techniques. Huang et al. [14], for instance, employed the well-known Particle Swarm Optimization (PSO) to optimize the number of input (visible) and hidden neurons, as well as the RBM learning rate in the context of time series forecasting prediction. Later on, Liu et al. [19] applied Genetic Algorithms (GA) for RBM model selection. Additionally, Levy et al. [18] also employed RBM and GA for automatic painter classification, but the former was used only for unsupervised feature learning purposes, being GA employed to optimize a weighted nearest neighbour classifier. Very recently, Papa et al. [24] proposed to optimize DRBMs by means of Harmony Search-based techniques, being the results more accurate than some well-known optimization libraries out there.

Therefore, the main contributions of this paper are twofold: (i) to introduce HS and some of its variants to the context of DBN model selection, and (ii) to fill the lack of research regarding DBN parameter optimization by means of meta-heuristic techniques. Although one can employ any other meta-heuristic technique, we opted to use HS since is its not based on derivatives, and it can be substantially faster than some swarm-based optimization techniques (it does not update all possible solutions at each iteration, only one). However, we must highlight the proposed approach used in this paper can be used with any other optimization technique, as recently presented by Papa et al. [23]. The remainder of this paper is organized as follows. Sections 2 and 3 present some theoretical background with respect to DBNs and HS, respectively. The methodology is discussed in Section 4, while Section 5 presents the experimental results. Finally, Section 6 states conclusions and future works.

2. 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.

2.1. Restricted Boltzmann Machines

Restricted Boltzmann Machines are energy-based stochastic neural networks composed by two layers of neurons (visible and hidden), in which the learning phase is conducted by means of an unsupervised fashion. The RBM is similar to the classical Boltzmann Machine [1], except that no connections between neurons

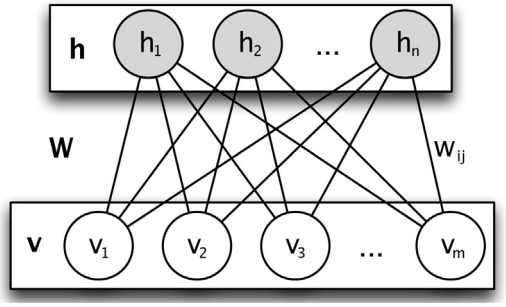

Fig. 1. The RBM architecture.

of the same layer are allowed.[1] Fig. 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.

At first, RBMs were designed using only binary visible and hidden units, the so-called Bernoulli Restricted Boltzmann Machines (BRBMs). Later on, Welling et al. [29] shed light over other types of units that can be used in an RBM, such as Gaussian and binomial units, among others. Since in this paper we are interested in BRBMs, we will introduce their main concepts, which are the basis for other generalizations of RBMs.

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 Bernoulli Restricted Boltzmann Machine is given by:

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

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:

$$P(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}, \quad (2)$$

where the denominator of above equation is a normalization factor that stands for all possible configurations involving the visible and hidden units.[2] In short, the BRBM learning algorithm aims at estimating $\mathbf{W}$, $\mathbf{a}$ and $\mathbf{b}$. The next section describes in more details this procedure.

2.2. Learning algorithm

The parameters of an BRBM 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:

$$P(\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}. \quad (3)$$

In order to update the weights and biases, it is necessary to compute the following derivatives:

$$\frac{\partial \log P(\mathbf{v})}{\partial w_{ij}} = E[h_j v_i]^{data} - E[h_j v_i]^{model}, \quad (4)$$

[1] Essentially, an RBM is modelled as a bipartite graph.

[2] Note this normalization factor is extremely hard to be computed when the number of units is too large.

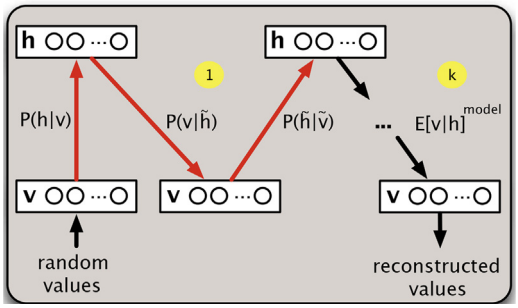

Fig. 2. Alternating Gibbs sampling. The "red" arrows denote one single iteration. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$\frac{\partial \log P(\mathbf{v})}{\partial a_i} = v_i - E[v_i]^{model}, \tag{5}$$

$$\frac{\partial \log P(\mathbf{v})}{\partial b_j} = E[h_j]^{data} - E[h_j]^{model}, \tag{6}$$

where $E[\cdot]$ stands for the expectation operation, and $E[\cdot]^{data}$ and $E[\cdot]^{model}$ correspond to the data-driven and the reconstructed-data-driven probabilities, respectively.

In practical terms, we can compute $E[h_j v_i]^{data}$ considering $\mathbf{h}$ and $\mathbf{v}$ as follows:

$$E[\mathrm{hv}]^{data} = P(\mathbf{h}|\mathbf{v})\mathbf{v}^T, \tag{7}$$

where $P(\mathbf{h}|\mathbf{v})$ stands for the probability of obtaining $\mathbf{h}$ given the visible vector (training data) $\mathbf{v}$:

$$P(h_j = 1|\mathbf{v}) = \sigma\left(\sum_{i=1}^{m} w_{ij} v_i + b_j\right), \tag{8}$$

where $\sigma(\cdot)$ stands for the logistic sigmoid function.[3] Therefore, it is straightforward to compute $E[\mathbf{hv}]^{data}$: given a training data $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ stands for a training set, we just need to set $\mathbf{v} \leftarrow \mathbf{x}$ and then employ Eq. (8) to obtain $P(\mathbf{h}|\mathbf{v})$. Further, we use Eq. (7) to finally obtain $E[\mathbf{hv}]^{data}$.

The big question now is how to obtain $E[\mathbf{hv}]^{model}$, which is the model learned by the system.[4] One possible strategy is to perform alternating Gibbs sampling starting at any random state of the visible units until a certain convergence criterion, such as k steps, for instance. The Gibbs sampling consists of updating hidden units using Eq. (8) followed by updating the visible units using $P(\mathbf{v}|\mathbf{h})$, given by:

$$P(v_i = 1|\mathbf{h}) = \sigma\left(\sum_{j=1}^{n} w_{ij} h_j + a_i\right), \tag{9}$$

and then updating the hidden units once again using Eq. (8). In short, it is possible to obtain an estimative of $E[\mathbf{hv}]^{model}$ by initializing the visible unit with random values and then performing Gibbs sampling. Fig. 2 illustrates this process, in which $E[\mathbf{hv}]^{model}$ can be approximated after k iterations. Notice a single iteration is defined by computing $P(\mathbf{h}|\mathbf{v})$, followed by computing $P(\mathbf{v}|\mathbf{h})$ and then computing $P(\mathbf{h}|\mathbf{v})$ once again.

For sake of explanation, Fig. 2 employs $P(\mathbf{v}|\tilde{\mathbf{h}})$ instead of $P(\mathbf{v}|\mathbf{h})$, and $P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})$ instead of $P(\mathbf{h}|\mathbf{v})$. Essentially, they stand for the same meaning, but $P(\mathbf{v}|\tilde{\mathbf{h}})$ is used here to denote the visible unit $\mathbf{v}$ is going to be reconstructed using $\tilde{\mathbf{h}}$, which was obtained through $P(\mathbf{h}|\mathbf{v})$. The same takes place with $P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})$, that reconstructs $\tilde{\mathbf{h}}$ using $\tilde{\mathbf{v}}$, which was obtained through $P(\mathbf{v}|\tilde{\mathbf{h}})$.

However, the procedure displayed in Fig. 2 is time-consuming, being also quite hard to establish suitable values for k.[5] Fortunately, Hinton [12] introduced a faster methodology to compute $E[\mathbf{hv}]^{model}$ based on contrastive divergence. Basically, the idea is to initialize the visible units with a training sample, to compute the states of the hidden units using Eq. (8), and then to compute the states of the visible unit (reconstruction step) using Eq. (9). Roughly speaking, this is equivalent to perform Gibbs sampling using $k = 1$.

Based on the above assumption, we can now compute $E[\mathbf{hv}]^{model}$ as follows:

$$E[\mathrm{hv}]^{model} = P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T. \tag{10}$$

Therefore, the equation below leads to a simple learning rule for updating the weight matrix $\mathbf{W}$, as follows:

$$\begin{aligned} \mathbf{W}^{t+1} &= \mathbf{W}^t + \eta(E[\mathrm{hv}]^{data} - E[\mathrm{hv}]^{model}) \\ &= \mathbf{W}^t + \eta(P(\mathbf{h}|\mathbf{v})\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T), \end{aligned} \tag{11}$$

where $\mathbf{W}^t$ stands for the weight matrix at time step t, and η corresponds to the learning rate. Additionally, we have the following formulae to update the biases of the visible and hidden units:

$$\begin{aligned} \mathbf{a}^{t+1} &= \mathbf{a}^t + \eta(\mathbf{v} - E[\mathbf{v}]^{model}) \\ &= \mathbf{a}^t + \eta(\mathbf{v} - \tilde{\mathbf{v}}), \end{aligned} \tag{12}$$

and

$$\begin{aligned} \mathbf{b}^{t+1} &= \mathbf{b}^t + \eta(E[\mathbf{h}]^{data} - E[\mathbf{h}]^{model}) \\ &= \mathbf{b}^t + \eta(P(\mathbf{h}|\mathbf{v}) - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})), \end{aligned} \tag{13}$$

where $\mathbf{a}^t$ and $\mathbf{b}^t$ stand for the visible and hidden units biases at time step t, respectively. In short, Eqs. (11)–(13) are the vanilla formulation for updating the RBM parameters.

Later on, Hinton [13] introduced a weight decay parameter λ, which penalizes weights with large magnitude,[6] as well as a momentum parameter α to control possible oscillations during the learning process. Therefore, we can rewrite Eqs. (11)–(13) as follows[7]:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \underbrace{\eta(P(\mathbf{h}|\mathbf{v})\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T) - \lambda\mathbf{W}^t + \alpha\Delta\mathbf{W}^{t-1}}_{=\Delta\mathbf{W}^t}, \tag{14}$$

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \underbrace{\eta(\mathbf{v} - \tilde{\mathbf{v}}) + \alpha\Delta\mathbf{a}^{t-1}}_{=\Delta\mathbf{a}^t} \tag{15}$$

and

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

In order to clarify the above content, Algorithm 1 presents the pseudo-code for RBM learning algorithm.

[3] The logistic sigmoid function can be computed by the following equation: $\sigma(x) = 1/(1 + \exp(-x))$.

[4] We are now writing $E[h_j v_i]^{model}$ in terms of $\mathbf{h}$ and $\mathbf{v}$.

[5] Actually, it is expected a good reconstruction of the input sample when $k \to +\infty$.

[6] The weights may increase during the convergence process.

[7] Notice when $\lambda = 0$ and $\alpha = 0$, we have the naïve gradient ascent.

Algorithm 1. Bernoulli RBM learning algorithm.

Input: A training set $\mathcal{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(l)}\}$, learning rate η, number of hidden units n, number of epochs T, weight decay λ and momentum α.
Output: Weight matrix $\mathbf{W}$ and the biases $\mathbf{a}$ and $\mathbf{b}$.

1 $\mathbf{W} \leftarrow \mathcal{N}(0, 0.01)$; $\mathbf{a} \leftarrow 0$; $\mathbf{b} \leftarrow 0$; $t = 1$; $\Delta\mathbf{W} \leftarrow 0$; $\Delta\mathbf{a} \leftarrow 0$;;$\Delta\mathbf{b} \leftarrow 0$;
2 $error = +\infty$; $\epsilon = 0.0001$;
3 **while** $(t \leq T)$ *and* $(error \geq \epsilon)$ **do**
4 $error = 0$; $\underline{\mathbf{v}} \leftarrow 0$; $\underline{\tilde{\mathbf{v}}} \leftarrow 0$; $\underline{\mathbf{h}} \leftarrow 0$; $\underline{\tilde{\mathbf{h}}} \leftarrow 0$;
5 **for** z *from* 1 *to* l **do**
6 $\mathbf{v} \leftarrow \mathbf{x}^{(z)}$; $\underline{\mathbf{v}} \leftarrow \underline{\mathbf{v}} + \mathbf{v}$;
7 **for** j *from* 1 *to* n **do**
8 Compute $P(h_j|\mathbf{v})$ according to Equation 8;
9 **if** $(P(h_j|v) \geq \mathcal{U}(0,1))$ **then**
10 $\tilde{h}_j = 1$;
11 **else**
12 $\tilde{h}_j = 0$;
13 $\underline{\mathbf{h}} \leftarrow \underline{\mathbf{h}} + P(\mathbf{h}|\mathbf{v})$;
14 **for** i *from* 1 *to* m **do**
15 Compute $P(v_i|\tilde{\mathbf{h}})$ according to Equation 9;
16 **if** $(P(v_i|\tilde{h}) \geq \mathcal{U}(0,1))$ **then**
17 $\tilde{v}_i = 1$;
18 **else**
19 $\tilde{v}_i = 0$;
20 $\underline{\tilde{\mathbf{v}}} \leftarrow \underline{\tilde{\mathbf{v}}} + \tilde{\mathbf{v}}$;
21 **for** j *from* 1 *to* n **do**
22 Compute $P(h_j|\tilde{\mathbf{v}})$ according to Equation 8;
23 **if** $(P(h_j|\tilde{v}) \geq \mathcal{U}(0,1))$ **then**
24 $\tilde{h}_j = 1$;
25 **else**
26 $\tilde{h}_j = 0$;
27 $\underline{\tilde{\mathbf{h}}} \leftarrow \underline{\tilde{\mathbf{h}}} + P(\tilde{\mathbf{h}}|\mathbf{v})$; $error = error + MSE(x^{(z)}, \tilde{\mathbf{v}})$;
28 $error = error/l$; $\underline{\mathbf{v}} \leftarrow \underline{\mathbf{v}}/l$; $\underline{\tilde{\mathbf{v}}} \leftarrow \underline{\tilde{\mathbf{v}}}/l$; $\underline{\mathbf{h}} \leftarrow \underline{\mathbf{h}}/l$; $\underline{\tilde{\mathbf{h}}} \leftarrow \underline{\tilde{\mathbf{h}}}/l$;
29 $\Delta\mathbf{W} \leftarrow \Delta\mathbf{W} - \lambda\mathbf{W} + \alpha\Delta\mathbf{W}$; $\mathbf{W} \leftarrow \mathbf{W} + \eta(\underline{\mathbf{h}}\underline{\mathbf{v}}^T - \underline{\tilde{\mathbf{h}}}\underline{\tilde{\mathbf{v}}}^T) + \Delta\mathbf{W}$;
30 $\Delta\mathbf{a} \leftarrow \Delta\mathbf{a} + \alpha\mathbf{a}$; $\mathbf{a} \leftarrow \mathbf{a} + \eta(\underline{\mathbf{v}} - \underline{\tilde{\mathbf{v}}}) + \Delta\mathbf{a}$;
31 $\Delta\mathbf{b} \leftarrow \Delta\mathbf{b} + \alpha\mathbf{b}$; $\mathbf{b} \leftarrow \mathbf{b} + \eta(\underline{\mathbf{h}} - \underline{\tilde{\mathbf{h}}}) + \Delta\mathbf{b}$;
32 $t = t + 1$;

Line 1 initializes the weight matrix $\mathbf{W}$ with a normal distribution with zero mean and variance of 0.01, as well as it initializes the biases of visible and hidden units, and the initial number of iterations.[8] The main loop in Lines 3–32 is responsible for executing the RBM learning procedure over T iterations or until the minimum error bound ϵ is reached.[9] After that, Line 4 initializes vectors $\underline{\mathbf{v}}$ and $\underline{\tilde{\mathbf{v}}}$, which are responsible for accumulating vectors $\mathbf{v}$ and $\tilde{\mathbf{v}}$ for each training data $x^{(z)}$, respectively, as well as it initializes vectors $\underline{\mathbf{h}}$ and $\underline{\tilde{\mathbf{h}}}$, which accumulate the values of vectors $\mathbf{h}$ and $\tilde{\mathbf{h}}$, respectively.[10]

The inner loop in Lines 5–27 performs the contrastive divergence algorithm for each training sample $x^{(z)}$: Line 6 sets the visible unit $\mathbf{v}$ with the current training sample, for further computation of $P(\mathbf{h}|\mathbf{v})$ in Lines 7–12 according to Eq. (8). The estimation of h_j is performed as follows: we first compute $P(h_j = 1|\mathbf{v})$, and then we compare it with a randomly generated number within the interval [0, 1] in order to assign a binary value to h_j.[11] Similarly, Lines 14–19 and Lines 21–26 compute $P(\mathbf{v}|\tilde{\mathbf{h}})$ (Eq. (9)) and $P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})$, respectively.

Line 27 computes the reconstruction error using the Minimum Squared Error (MSE), although the reader can use any other sort of error metric. After that, the error value and vectors $\underline{\mathbf{v}}$, $\underline{\tilde{\mathbf{v}}}$, $\underline{\mathbf{h}}$ and $\underline{\tilde{\mathbf{h}}}$ are then averaged by the number of training samples l (Line 28). The weights are updated in Line 29 according to Eq. (14), as well as the biases of visible and hidden units in Lines 30–31 according to Eqs. (15) and (16), respectively.

2.3. 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 2.2 in a greedy fashion, which means an RBM at a certain layer does not consider others during its learning procedure. Fig. 3 depicts such architecture, being each RBM at a certain layer represented as illustrated in Fig. 1. In this case, we have a DBN composed of L layers, being $\mathbf{W}^i$ the weight matrix of RBM at layer i. Additionally, we can observe the hidden units at layer i become the input units to the layer $i+1$. Although we did not illustrate the bias units for the visible (input) and hidden layers in Fig. 3, we also have such units for each layer.

The approach proposed by Hinton et al. [11] 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

[8] Notice the symbol "←" is used for operations with vectors and matrices.
[9] The reader can define its own bound ϵ.
[10] The summation of vectors $\mathbf{v}$, $\tilde{\mathbf{v}}$, $\mathbf{h}$ and $\tilde{\mathbf{h}}$ is required since we have a training set with l elements. Notice the theory previously presented in this section considers only one training sample.
[11] The term $\mathcal{U}(0, 1)$ stands for a uniform distribution within the interval [0, 1].

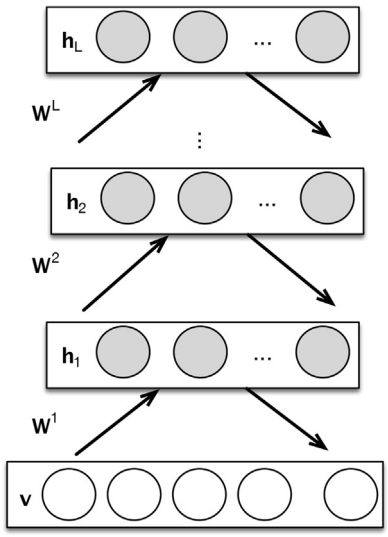

Fig. 3. The DBN architecture.

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. Zhou et al. [33], for instance, presented a Discriminative Deep Belief Network for image classification, being the top layer (additional layer) used to encode all possible labels (usually a binary vector is used for such purpose). On that work, a function that computes the classification error (loss) is then used for minimization purposes.

3. Harmony Search

Harmony Search is a meta-heuristic algorithm inspired in the improvisation process of music players. Musicians often improvise the pitches of their instruments searching for a perfect state of harmony [9]. The main idea is to use the same process adopted by musicians to create new songs to obtain a near-optimal solution according to some fitness function. Each possible solution is modelled as a harmony, and each musician corresponds to one decision variable.

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

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 of good musical notes 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,[12] as follows:

$$\hat{\phi}^j = \begin{cases} \phi_A^j & \text{with } \textit{probability } \text{HMCR} \\ \theta \in \Phi_j & \text{with } \textit{probability}(1 - \textit{HMCR}), \end{cases} \tag{17}$$

where $A \sim \mathcal{U}(1, 2, \ldots, N)$, and $\boldsymbol{\Phi} = \{\Phi_1, \Phi_2, \ldots, \Phi_M\}$ stands for the set of feasible values for each decision variable.[13]

Further, every component j of the new harmony vector $\hat{\boldsymbol{\phi}}$ is examined to determine whether it should be pitch-adjusted or not, which is controlled by the Pitch Adjusting Rate (PAR) variable, according to Eq. (18):

$$\hat{\phi}^j = \begin{cases} \hat{\phi}^j \pm \varphi_j \varrho & \text{with } \textit{probability } \text{PAR} \\ \hat{\phi}^j & \text{with } \textit{probability}(1 - \textit{PAR}). \end{cases} \tag{18}$$

The pitch adjustment is often used to improve solutions and to escape from local optima. This mechanism concerns shifting the neighbouring values of some decision variable in the harmony, where ϱ is an arbitrary distance bandwidth, and $\varphi_j \sim \mathcal{U}(0, 1)$. Below, we briefly present some variants of the vanilla Harmony Search employed in this work.

Improved Harmony Search. The Improved Harmony Search (IHS) [20] differs from traditional HS by updating the PAR and ϱ values dynamically. The PAR updating formulation at time step t is given by:

$$PAR^t = PAR_{\min} + \frac{PAR_{\max} - PAR_{\min}}{T} t, \tag{19}$$

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:

$$\varrho^t = \varrho_{\max} \exp \frac{\ln(\varrho_{\min}/\varrho_{\max})}{T} t, \tag{20}$$

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

Global-best Harmony Search. The Global-best Harmony Search (GHS) [21] employs the same modification proposed by IHS with respect to dynamic PAR values. However, it does not employ the concept of bandwidth, being Eq. (18) replaced by:

$$\hat{\phi}^j = \phi_{best}^z, \tag{21}$$

where $z \sim U(1, 2, \ldots, N)$, and ***best*** stands for the index of the best harmony.

Novel Global Harmony Search. The Novel Global Harmony Search (NGHS) [34] differs from traditional HS in three aspects: (i) the HMCR and PAR parameters are excluded, and a mutation probability ω is then used; (ii) the NGHS always replaces the worst harmony with the new one, and (iii) the improvisation footsteps are also modified, as follows:

$$R = 2\phi_{best}^j - \phi_{worst}^j, \tag{22}$$

$$\hat{\phi}^j = \phi_{worst}^j + \mu_j (R - \phi_{worst}^j), \tag{23}$$

where $\boldsymbol{\phi}_{\boldsymbol{worst}}$ stands for the worst harmony, and $\mu_j \sim U(0, 1)$. Further, another modification with respect to the mutation probability is performed in the new harmony:

$$\hat{\phi}^j = \begin{cases} L^j + \varpi_j (U^j - L^j) & \text{if } \kappa_j \leq \omega \\ \hat{\phi}^j & \text{otherwise}, \end{cases} \tag{24}$$

where $\kappa_j, \varpi_j \sim U(0, 1)$, and U^j and L^j stand for the upper and lower bounds of decision variable j, respectively.

[12] The term "feasible value" means the value that falls in the range of a given decision variable.

[13] Variable A denotes a harmony index randomly chosen from the harmony memory.

Table 1
Parameter configuration.

Technique	Parameters
HS	$HMCR = 0.7$, $PAR = 0.7$, $\varrho = 10$
IHS	$HMCR = 0.7$, $PAR_{min} = 0.1$, $PAR_{max} = 0.7$, $\varrho_{min} = 1$, $\varrho_{max} = 0.10$
GHS	$HMCR = 0.7$, $PAR_{min} = 0.1$, $PAR_{max} = 0.7$
NGHS	$\omega = 0 \cdot t$
SGHS	$HMCR_m = 0.98$, $PAR_m = 0.9$, $\varrho_{min} = 0$, $\varrho_{max} = 0.9$, $LP = 5$

Self-adaptive Global best Harmony Search. The SGHS algorithm [22] is a modification of the aforementioned GHS, which employs a new improvisation scheme and self-adaptive parameters. First of all, Eq. (21) is rewritten as follows:

$$\hat{\phi}^j = \phi^j_{best}, \tag{25}$$

and Eq. (17) can be replaced by:

$$\hat{\phi}^j = \begin{cases} \phi^j_A \pm \varphi_j \varrho & \text{with probability } HMCR \\ \theta \in \Phi_j & \text{with probability } (1 - HMCR). \end{cases} \tag{26}$$

The main difference among SGHS and the aforementioned variants concerns with the computation of *HMCR* and *PAR* values, which are estimated based on the average of their recorded values after each *LP* (learning period) iterations. Every time a new harmony is better than the worst one, the *HMCR* and *PAR* values are then recorded to be used in the estimation of their new values, which follow a Gaussian distribution, i.e., $HMCR \sim \mathcal{N}(HMCR_m, 0.01)$ and $PAR \sim \mathcal{N}(PAR_m, 0.05)$, where $HMCR_m$ and PAR_m stand for the mean values of *HMCR* and *PAR* parameters, respectively.[14] The initial values for $HMCR_m$ and PAR_m are the very same values for *HMCR* and *PAR* displayed in Table 1, respectively.

Parameter-Setting-Free Harmony Search. The PSF-HS algorithm [10] avoids using both *HMCR* and *PAR* parameters, since they are computed based on the average of times a decision variable comes from the Harmony Memory, or it has been pitched, respectively.

4. Methodology

In this section, we present the proposed approach for DBN model selection, as well as we describe the employed datasets and the experimental setup.

4.1. Modelling DBN parameter optimization

We propose to model the problem of selecting suitable parameters for DBNs by means of vanilla Harmony Search and some of its variants. As aforementioned in Section 2.2, the learning step of an RBM has four parameters: the learning rate η, weight decay λ, penalty parameter α, and the number of hidden units n. Therefore, we have a 4-dimensional search space with three real-valued variables, as well as the integer-valued number of hidden units, for each layer. As we are working with $L = \{1, 2, 3\}$ layers, the search spaces have 4, 8 and 12 decision variables, respectively. In short, the proposed approach aims at selecting the set of DBN parameters that minimizes the mean squared error (MSE) in the context of binary image reconstruction, i.e.:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (\hat{I}_i - I_i)^2, \tag{27}$$

where $\hat{I}_i$ and I_i stand for the ith reconstructed and original images, respectively. After that, the selected set of parameters is then applied to reconstruct the test images.

4.2. Datasets

In regard to the *image reconstruction* experiment, we employed three datasets, as described below:

- MNIST dataset[15]: it is composed by images of handwritten digits. The original version contains a training set with 60,000 images from digits '0' to '9', as well as a test set with 10,000 images.[16] Due to the high computational burden for RBM model selection, we decided to employ the original test set together with a reduced version of the training set.[17]
- CalTech 101 Silhouettes dataset[18]: it is based on the former Caltech 101 dataset, and it comprises silhouettes of images from 101 classes with resolution of 28×28. Since we have gray-scale images available in a training, validation and test sets, we converted them in to binary images before using Bernoulli RBMs. Additionally, 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 dataset[19]: this dataset contains 1593 binary images of manuscript digits with resolution of 16×16 from around 80 persons. We employed the whole dataset in the experimental section.

Fig. 4 displays some training examples from both datasets, which were partitioned in 2% for the training set and 98% to compose the test set.

4.3. Experimental setup

In this work, we compared the proposed HS-based DBN model selection against with a random initialization of parameters (RS) and the Hyperopt library using random search (Hyper-RS) and Tree of Parzen Estimators (Hyper-TPE) [4]. Additionally, we evaluated five HS variants: (i) IHS, (ii) GHS, (iii) NGHS, (iv) SGHS and (v) PSF-HS. We also evaluated the robustness of parameter fine-tuning in three distinct DBN models: one layer (1L), two layers (2L) and three layers (3L). Notice the 1L approach stands for the standard RBM.

In order to provide a statistical analysis by means of Wilcoxon signed-rank test [30], we conducted a cross-validation with 20 runnings. Finally, we employed 5 agents over 50 iterations for convergence considering all techniques. Therefore, this means we have 55 evaluations of the fitness function (DBN learning algorithm) for each technique, except for Random Search (RS), in which we allowed one evaluation only. However, to be fair with Hyper-RS and Hyper-TPE, we allowed them to evaluate the fitness function 55 times either. Table 1 presents the parameter configuration for each optimization technique.[20]

Finally, we have set each DBN parameter according to the following ranges: $n \in [5, 100]$, $\eta \in [0.1, 0.9]$, $\lambda \in [0.1, 0.9]$ and $\alpha \in [0.0, 0.001]$. Therefore, this means we have used such ranges to initialize the optimization techniques, as well as to conduct the baseline experiment by means of randomly set values. We also

[14] The variance values used to compute *HMCR* and *PAR* are the same as proposed by Kulluk et al. [15].

[15] http://yann.lecun.com/exdb/mnist/.

[16] The images are originally available in gray-scale with resolution of 28×28. In order to work with Bernoulli RBMs, all images were binary-scaled converted.

[17] The original training set was reduced to 2% of its former size, which corresponds to 1200 images.

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

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

[20] Notice these values have been empirically chosen.

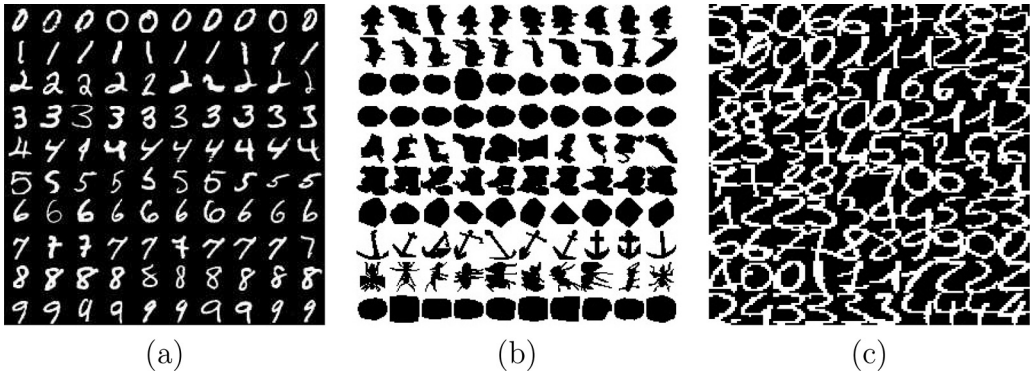

Fig. 4. Some training examples from (a) MNIST, (b) CalTech 101 Silhouettes and (c) Semeion datasets.

Table 2
Average MSE over the test set considering MNIST dataset.

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

have employed $T = 100$ as the number of epochs for DBN learning weights procedure with mini-batches of size 20. The very same of ranges were used to initialize the variables for all layers. In order to provide a more detailed experimental validation, all DBNs were trained with three different algorithms[21]: Contrastive Divergence (CD) [12], Persistent Contrastive Divergence (PCD) [27] and Fast Persistent Contrastive Divergence (FPCD) [28]. Additionally, we employed the Wilcoxon signed-rank test [30] for statistical validation purposes.

5. Experiments

In this section, we present the experimental evaluation considering each dataset individually. As aforementioned, the main idea of this work is to evaluate the robustness of HS-based techniques in the context of fine-tuning DBNs considering the task of binary image reconstruction. Besides, the proposed approach can be employed to fine-tune DBNs in the context of data classification as well.

5.1. MNIST dataset

Table 2 presents the mean MSE over the test set for each optimization technique considering DBNs with one (1L), two (2L) and three (3L) layers, as well as DBNs trained with CD, PCD and FPCD learning algorithms. The values in bold stand for the similar techniques with the lowest errors according to Wilcoxon signed-rank test.

The best mean squared errors were obtained by IHS using one layer only and PCD as the learning algorithm. Although we can notice lowest errors when increasing the number of layers for some situations, it seems PCD and FPCD are most likely to benefit from such improvement. However, it is important to consider we are using one sampling iteration for CD, PDC and FPCD only. This means we might obtain better results using more layers when applying more sampling iterations, but it is far from beyond the scope of this work, since we are interested into showing that meta-heuristic techniques can be successfully employed to fine-tune DBNs. Based on the results, we may conclude HS-based techniques, specially IHS, are suitable for the aforementioned task, since they obtained better results than a random search, as well as lowest errors when compared to a well-known optimization library (Hyperopt).

Fig. 5a shows the logarithm of the Pseudo-likelihood (PL) considering all training samples for a given execution of IHS-based DBN fine-tuning with PCD and FPCD using one layer. Additionally, Fig. 5b depicts the same information, but now considering RS technique. Clearly, we can observe better values during the convergence process among the epoch iterations for IHS (the greater the values, the better is the technique).

Obviously, the optimization techniques require much more computational effort than a simple random search, since the latter one requires only one execution of the fitness function (DBN learning algorithm) for each layer, while all other techniques require 55 executions (number of harmonies $\times$ number of iterations). Another interesting point we shall observe is related to the number of epochs, which is set to 10 in this work. Usually, we may use hundreds or even thousands of them, but with the price of a high computational burden. We did not employ such a number because even a random initialization of the parameters may converge after a long period of learning. Then, there would be not reason to employ meta-heuristics for such purpose. However, we would like to emphasize if one has time-constraints, it is possible to use the proposed approach to obtain suitable results.

5.2. CalTech 101 Silhouettes dataset

In this section, we present the reconstruction results considering Caltech 101 dataset, which is more challenging than MNIST dataset, since it has more classes and complex shapes. Once again,

[21] We used one sampling iteration for all learning algorithms.

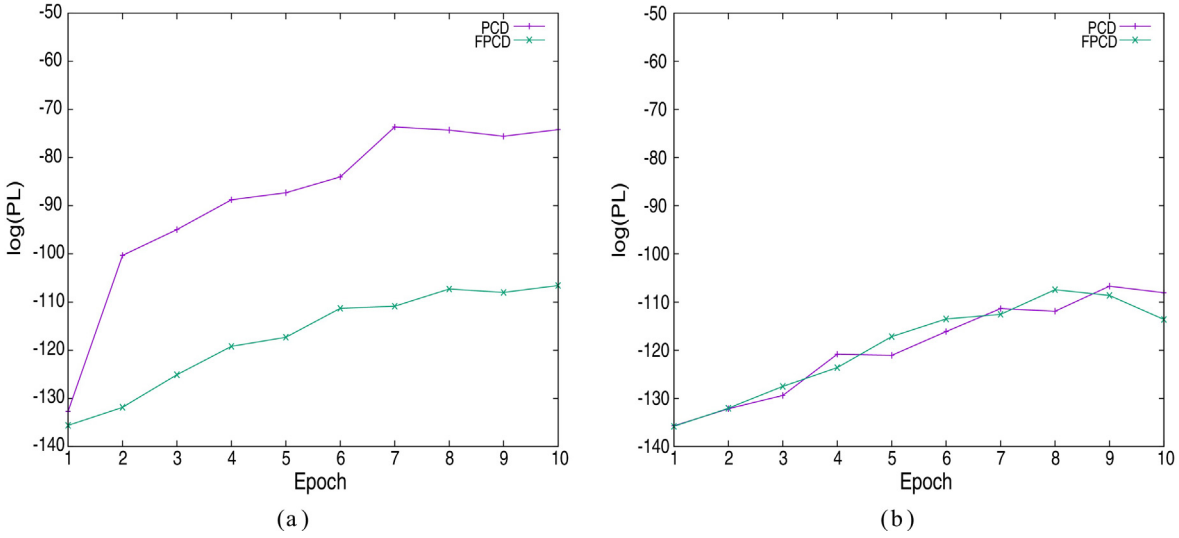

Fig. 5. Logarithm of the Pseudo-likelihood values considering (a) IHS and (b) RS for MNIST dataset.

Table 3
Average MSE over the test set considering CalTech 101 Silhouettes dataset.

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

IHS obtained the best results, but now with a DBN using three layers and FPCD as the learning algorithm. As aforementioned, it is expected a better resulting using a deeper DBN, since this dataset present more complex shapes, thus requiring better learned features. Table 3 displays such results, being the bolded values the similar techniques with the lowest errors.

Fig. 6a shows the logarithm of the Pseudo-likelihood (PL) at the second layer considering all training samples for a given execution of IHS-based DBN fine-tuning with PCD and FPCD using three layers, and Fig. 6b depicts the very same information for RS. These results follow the same pattern observed in Table 3, i.e., the greater PL values, the smaller the amount of reconstruction error.

The proposed approach performs a greedy-based optimization at each layer, as usually recommended by the literature. Therefore, we need to re-run HS (and all compared techniques) for each layer with the parameters fine-tuned in the previous one. In this work, we did not employ a post-optimization such as gradient descent or backpropagation, since we believe it would not modify the main contribution of the paper, which relies on which situations one should employ a meta-heuristic-based DBN fine-tuning.

5.3. Semeion Handwritten Digit dataset

In this section, we present the results regarding Semeion Handwritten Digit Data Set, being 30% of the dataset used for training, and the remaining 70% used for testing purposes. Table 4 displays the MSE over the test set using the very same procedure applied to the other datasets, i.e., we evaluate DBNs with 1, 2 and 3 layers, as well as CD, PCD and FPCD as the learning algorithms. Taking into account the MSE values, Semeion dataset comprises a more challenging task, since the techniques used in this work obtained the highest errors on such data. Such behaviour favoured a larger number of layers, since the best results were obtained with IHS using CD and 3 layers. Therefore, a larger number of layers allows a better description of the data.

Fig. 7a shows the logarithm of the Pseudo-likelihood (PL) at the first layer considering all training samples for a given execution of IHS-based DBN fine-tuning with CD and PCD (the second best result) using three layers, and Fig. 7b depicts the very same information for RS. Both figures depict similar PL values, since the results presented in Table 4 are close to each other with respect to the techniques considered in these figures.

Differently from the results displayed in Figs. 5 and 6, the results depicted in Fig. 7 highlighted an interesting situation we shall consider as future works: we can realize, for some iterations (e.g., iteration #6 considering Fig. 7a) the worst technique (PCD) obtained better results than CD, although the latter one achieved the best PL value so far (iteration #10). Such behaviour is very interesting to go towards approaches that combine DBN trained with different learning algorithms. Cho et al. [6], for instance, showed the suitability of Parallel Tempering (PT) for training RBMs. Roughly speaking, the idea of PT is to run several Markov chains at the same time with different temperatures each. Then, we can select the Markov chain with the best PL value, for instance. Later on, Brakel et al. [5] introduced Multi-Tempering to the same context, i.e., training RBMs. In such approach, the chains can interchange information, which does not happen with Parallel Tempering (each chain runs independently). Therefore, a possible idea would be to run parallel Markov chains with different temperatures also, but using a different sampling method on each one, i.e., we can use CD

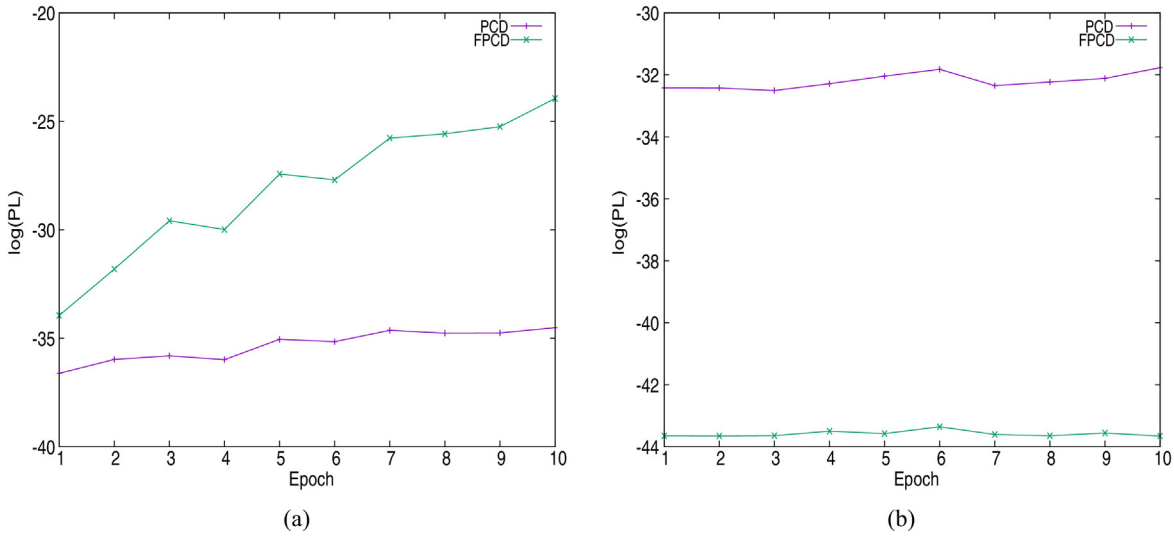

Fig. 6. Logarithm of the Pseudo-likelihood values considering (a) IHS and (b) RS for CalTech 101 Silhouettes dataset.

Table 4
Average MSE over the test set considering Semeion Handwritten Digit dataset.

	1L			2L			3L		
	CD	PCD	FPCD	CD	PCD	FPCD	CD	PCD	FPCD
HS	0.2128	0.2128	0.2129	0.2202	0.2128	0.2128	0.2199	0.2128	0.2128
IHS	0.2131	0.2130	0.2128	0.2116	0.2114	0.2121	**0.2103**	0.2109	0.2119
GHS	0.2133	0.2129	0.2128	0.2129	0.2130	0.2129	0.2129	0.2129	0.2128
NGHS	0.2134	0.2132	0.2131	0.2130	0.2131	0.2129	0.2131	0.2132	0.2130
SGHS	0.2135	0.2131	0.2130	0.2131	0.2131	0.2130	0.2132	0.2132	0.2130
PSF-HS	0.2137	0.2130	0.2130	0.2121	0.2120	0.2124	0.2120	0.2120	0.2121
RS	0.2146	0.2143	0.2145	0.2146	0.2144	0.2139	0.2143	0.2140	0.2140
Hyper-RS	0.2127	0.2129	0.2129	0.2129	0.2129	0.2129	0.2129	0.2129	0.2128
Hyper-TPE	0.2128	0.2128	0.2128	0.2128	0.2128	0.2127	0.2128	0.2128	0.2128

for one chain and PCD for another. Then, we can use the best learning algorithm considering each iteration, thus "jumping" from one chain to another.

5.4. Discussion

The experimental results over the datasets allow us to draw some important conclusions. First, the reader can benefit from any optimization technique instead of using a random search for DBN fine-tuning. Although we may pay the price of a higher computational load, we can apply meta-heuristic techniques for such purpose whenever the computational time is not a big deal. Second, IHS seemed to be the most effective technique among all HS-based variants, showing the strategy of dynamic updating PAR and bandwidth values is of great importance in the context addressed in this paper.

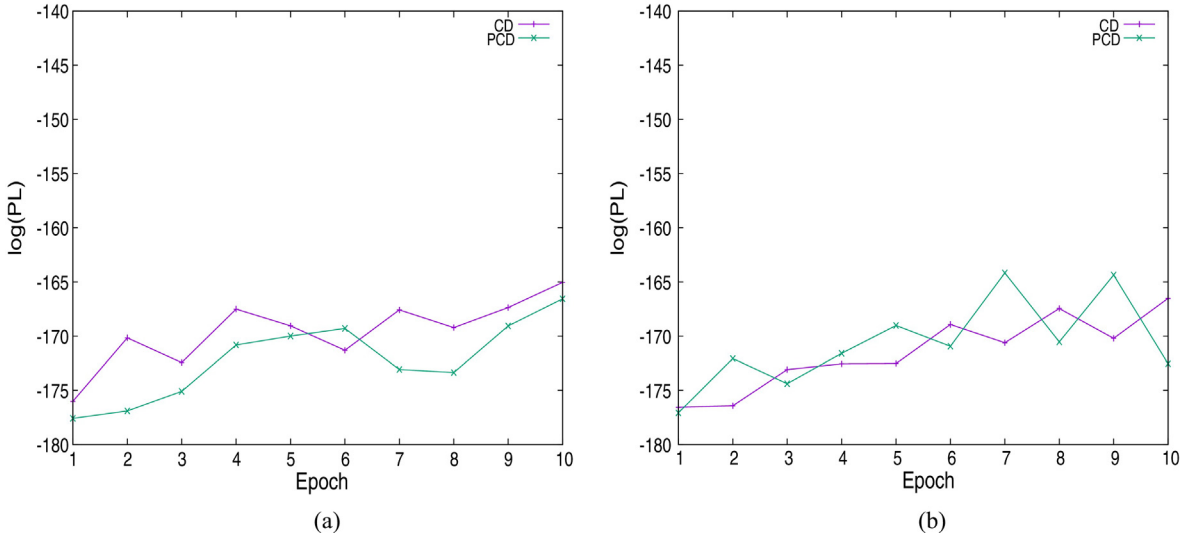

Fig. 7. Logarithm of the Pseudo-likelihood values considering (a) IHS and (b) RS for Semeion Handwritten Digit.

In regard to the DBN architecture, we have observed the more complex the datasets are, the more layers we need to better describe them. Considering MNIST dataset, the best results were achieved with one layer only, while we needed three layers to obtain the lowest reconstruction errors over Semeion dataset. Taking a look at Fig. 4a and c, we can observe the digits comprised in MNIST dataset are "well-behaved", thus imposing a less difficult problem than Semeion dataset.

Although CD obtained the best results considering Semeion dataset, PCD was the second best approach, with results very close to CD. The Persistent Contrastive Divergence also obtained the best results together with CD and FPCD over Caltech dataset, and once again the best results with FPCD with respect to MNIST data. Therefore, the idea of building a persistent chain instead of reinitializing it whenever a new training sample appears during the learning procedure seems to improve DBN results. We must highlight here we have used 10 iterations for the training step only, since we are not interested into outperforming the best results to date. If we have time enough for learning purposes (i.e., thousands of iterations), the idea is that both random search and meta-heuristic techniques may obtain very similar results, although we still have observed a slight advantage of meta-heuristic techniques over the random search.

6. Conclusions

Deep Belief Networks have been extensively used for several applications in the last years due to their capability on describing data using a number of layers that encode different source of information. However, we can observe only a few works that aim at tackling the problem of model selection for such techniques, i.e., to learn the set of parameters that lead to the best results. It is usual to find works that an employ empirical evaluation of such parameters, or even random values. While the former approach might be time-consuming, the latter one may not be the best choice. In this work, we introduced Harmony Search in the context of DBN parameters fine-tuning, as well as we compared a number of its variants against with a random search and the well-known Hyperopt library.

In order to fulfil the purpose of this work, we used three public datasets aiming at binary image reconstruction with different characteristics. Although all datasets are shape-oriented, Semeion Handwritten Digit is more challenging, since it has more complex shapes. The experimental results were carried out using cross-validation with 20 runnings and with three different learning algorithms: Contrastive Divergence, Persistent Contrastive Divergence and Fast Persistent Contrastive Divergence. Considering MNIST dataset, the best results were obtained with PCD and FPCD learning algorithms using IHS and one layer only, while for Caltech dataset, the best results were obtained with FPCD using IHS and three layers. The latter dataset required a deeper DBN than the first one, since it contains more complex shapes at different positions. Since Semeion dataset poses a more challenging task, it required more layers than the other datasets to obtain the best results, which is not surprisingly.

We believe we could achieve the main goal of this work, which is based on the hypothesis that a fast meta-heuristic technique is suitable to fine-tune DBN parameters. Although swarm-based techniques such as Particle Swarm Optimization or evolutionary approaches such as Genetic Algorithms may obtain slightly better results, we believe their high computational load may not fit to this problem. In regard to future works, we aim at fine-tuning Deep Boltzmann Machines, which have a close formulation to that one of DBNs.

Acknowledgments

The authors are grateful to FAPESP grants #2013/20387-7 and #2014/16250-9, and CNPq grants #303182/2011-3, #470571/2013-6 and #306166/2014-3.

References

[1] D.H. Ackley, G.E. Hinton, T.J. Sejnowski, A learning algorithm for Boltzmann machines, in: D. Waltz, J.A. Feldman (Eds.), Connectionist Models and Their Implications: Readings from Cognitive Science, Ablex Publishing Corp., Norwood, NJ, USA, 1988, pp. 285–307.
[2] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[3] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[4] J.S. Bergstra, D. Yamins, D.D. Cox, Hyperopt: a python library for optimizing the hyperparameters of machine learning algorithms, in: Python for Scientific Computing Conference, 2013, pp. 1–7.
[5] P. Brakel, S. Dieleman, B. Schrauwen, Training restricted Boltzmann machines with multi-tempering: harnessing parallelization, in: A.E.P. Villa, W. Duch, P. Érdi, F. Masulli, G. Palm (Eds.), in: Proceedings of the Artificial Neural Networks and Machine Learning. Lecture Notes in Computer Science, vol. 7553, Springer Berlin Heidelberg, 2012, pp. 92–99.
[6] K.-H. Cho, T. Raiko, A. Ilin, Parallel tempering is efficient for learning restricted Boltzmann machines, in: International Joint Conference on Neural Networks, 2010, pp. 1–8.
[7] A. Fischer, C. Igel, Training restricted Boltzmann machines: an introduction, Pattern Recognit. 47 (1) (2014) 25–39.
[8] C. Fuqianga, W. Yana, B. Yudea, Z. Guodonga, Spectral classification using restricted Boltzmann machine, Publ. Astron. Soc. Aust. 31 (2014) 1–7.
[9] Z.W. Geem, Music-Inspired Harmony Search Algorithm: Theory and Applications, 1st edition, Springer Publishing Company, Incorporated, 2009.
[10] Z.W. Geem, K.-B. Sim, Parameter-setting-free harmony search algorithm, Appl. Math. Comput. 217 (8) (2010) 3881–3889.
[11] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[12] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (8) (2002) 1771–1800.
[13] G.E. Hinton, A practical guide to training restricted Boltzmann machines, in: G. Montavon, G.B. Orr, K.-R. Müller (Eds.), in: Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol. 7700, Springer Berlin Heidelberg, 2012, pp. 599–619.
[14] D.-S. Huang, P. Gupta, X. Zhang, P. Premaratne, Time series forecasting using restricted Boltzmann machine, in: Emerging Intelligent Computing Technology and Applications. Communications in Computer and Information Science, Springer Berlin Heidelberg, 2012, pp. 17–22.
[15] S. Kulluk, L. Ozbakir, A. Baykasoglu, Self-adaptive global best harmony search algorithm for training neural networks, Procedia Comput. Sci. 3 (2011) 282–286, World Conference on Information Technology.
[16] H. Larochelle, M. Mandel, R. Pascanu, Y. Bengio, Learning algorithms for the classification restricted Boltzmann machine, J. Mach. Learn. Res. 13 (1) (2012) 643–669.
[17] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[18] E. Levy, O.E. David, N.S. Netanyahu, Genetic algorithms and deep learning for automatic painter classification, in: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, GECCO '14, ACM, New York, NY, USA, 2014, pp. 1143–1150.
[19] K. Liu, L.M. Zhang, Y.W. Sun, Deep Boltzmann machines aided design based on genetic algorithms, in: P. Yarlagadda, Y.-H. Kim (Eds.), Applied Mechanics and Materials, Scientific.Net, 2014, pp. 848–851, Chapter: Artificial Intelligence, Optimization Algorithms and Computational Mathematics.
[20] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2) (2007) 1567–1579.
[21] M.G.H. Omran, Mehrdad Mahdavi, Global-best harmony search, Appl. Math. Comput. 198 (2) (2008) 643–656.
[22] Q.-K. Pan, P.N. Suganthan, M. Fatih Tasgetiren, J.J. Liang, A self-adaptive global best harmony search algorithm for continuous optimization problems, Appl. Math. Comput. 216 (3) (2010) 830–848.
[23] J.P. Papa, G.H. Rosa, K.A.P. Costa, A.N. Marana, W. Scheirer, D.D. Cox, On the model selection of Bernoulli restricted Boltzmann machines through harmony search, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2015, pp. 1449–1450.
[24] J.P. Papa, G.H. Rosa, A.N. Marana, W. Scheirer, D.D. Cox, Model selection for discriminative restricted Boltzmann machines through meta-heuristic techniques, J. Comput. Sci. 9 (2015) 14–18.
[25] R. Salakhutdinov, G. Hinton, An efficient learning procedure for deep Boltzmann machines, Neural Comput. 24 (8) (2012) 1967–2006.
[26] R. Sarikaya, G.E. Hinton, A. Deoras, Application of deep belief networks for natural language understanding, IEEE/ACM Trans. Audio Speech Lang. Process. 22 (4) (2014) 778–784.

[27] T. Tieleman, Training restricted Boltzmann machines using approximations to the likelihood gradient, in: Proceedings of the 25th International Conference on Machine Learning, ACM, New York, NY, USA, 2008, pp. 1064–1071.
[28] T. Tieleman, G.E. Hinton, Using fast weights to improve persistent contrastive divergence, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, New York, NY, USA, 2009, pp. 1033–1040.
[29] M. Welling, M. Rosen-zvi, G.E. Hinton, Exponential family harmoniums with an application to information retrieval, in: L.K. Saul, Y. Weiss, L. Bottou (Eds.), in: Advances in Neural Information Processing Systems, vol. 17, MIT Press, 2005, pp. 1481–1488.
[30] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (6) (1945) 80–83.
[31] J. Yosinski, H. Lipson, Visually debugging restricted Boltzmann machine training visually debugging restricted Boltzmann machine training with a 3d example, in: Representation Learning Workshop. International Conference on Machine Learning, Edinburgh, Scotland, 2012, pp. 1–6.
[32] C.-X. Zhang, J.-S. Zhang, N.-N. Ji, G. Guo, Learning ensemble classifiers via restricted Boltzmann machines, Pattern Recognit. Lett. 36 (2014) 161–170.
[33] S. Zhou, Q. Chen, X. Wang, Discriminative deep belief networks for image classification, in: 17th IEEE International Conference on Image Processing, 2010, pp. 1561–1564.
[34] D. Zou, L. Gao, J. Wu, S. Li, Y. Li, A novel global harmony search algorithm for reliability problems, Comput. Ind. Eng. 58 (2) (2010) 307–316, Scheduling in Healthcare and Industrial Systems.