Fine-Tuning Infinity Restricted Boltzmann Machines

L. A. Passos
Department of Computing
Federal University of São Carlos
São Carlos, Brazil
E-mail:leandropassosjr@gmail.com

J. P. Papa
Department of Computing
São Paulo State University
Bauru, Brazil
E-mail:papa@fc.unesp.br

***Abstract*—Restricted Boltzmann Machines (RBMs) have received special attention in the last decade due to their outstanding results in number of applications, such as face and human motion recognition, and collaborative filtering, among others. However, one of the main concerns about RBMs is related to the number of hidden units, which is application-dependent. Infinite RBM (iRBM) was proposed as an alternative to the regular RBM, where the number of units in the hidden layer grows as long as it is necessary, dropping out the need for selecting a proper number of hidden units. However, a less sensitive regularization parameter is introduced as well. This paper proposes to fine-tune iRBM hyper-parameters by means of meta-heuristic techniques such as Particle Swarm Optimization, Bat Algorithm, Cuckoo Search, and the Firefly Algorithm. The proposed approach is validated in the context of binary image reconstruction over two well-known datasets. Furthermore, the experimental results compare the robustness of the iRBM against the RBM and Ordered RBM (oRBM) using two different learning algorithms, showing the suitability in using meta-heuristics for hyper-parameter fine-tuning in RBM-based models.**

***Keywords*-Deep Learning; Infinity Restricted Boltzmann Machines; Meta-heuristics**

I. Introduction

Restricted Boltzmann Machines (RBMs) are two-layered undirected graphical models that use a layer of hidden units to model the distribution over a set of inputs, thus compounding a generative stochastic neural network [1], [2]. RBMs have been highlighted in the scientific community over the last years, as well as some variants concerning deep learning models, e.g., Deep Belief Networks (DBNs) [3] and Deep Boltzmann Machines (DBMs) [4], due to their outstanding results in a number of domains, such as human motion [5], classification [1], spam [6] and anomaly detection [7], and collaborative filtering [8], just to cite a few.

However, one of the main concerns related to RBMs is associated with the number of hidden units, which is application-dependent and has a great impact in the final results. Montufar and Ay [9] showed that an RBM with $2^{m-1}-1$ hidden units is a universal approximator, where m stands for the number of visible (input) units. Moreover, such a big representation may not be efficient in practice, which motivated researchers to study models that can automatically increase their capacity during learning.

Cotê and Larochelle [10] proposed an extension of the RBM that does not require specifying the number of hidden units, and it can increase its capacity (i.e., number of hidden units) during training, hereinafter called infinite RBM (iRBM). In this work, they also presented an extension of the RBM that is sensitive to the position of each unit in the hidden layer, named ordered Restricted Boltzmann Machines (oRBM), which can be interpreted as a special case of an implicit mixture of RBMs [11]. This is achieved by adding new units in the hidden layer, where each one is trained gradually from left to right. Effectively, the model is growing in capacity during training until it reaches the maximum capacity defined previously.

Based on the aforementioned assumption, it turns out to be possible to devise a model where the number of hidden units increases automatically to a capacity that is similar to the universal approximator (i.e., when the number of hidden units tends to infinite), though being much smaller. Such model is possible due to the following assumptions: (i) that a finite number of hidden units has non-zero weights and biases, and (ii) the parametrization of the per-unit energy penalty (β) ensures the infinite sums during probability computation will converge. Since the role of this energy penalty is to ensure the iRBM is properly defined only, the penalty imposed in the energy function can be compensated by the learned parameters (weight decay). Therefore, we can remove one the RBM hyper-parameters from its project, i.e., the number of hidden units.

Despite dropping out the number of hidden units that is usually required beforehand, the iRBM still demands the selection of the remaining hyper-parameters, such as the learning rate, momentum and weight decay. Furthermore, its formulation incorporates the β hyper-parameter, which is less sensitive than the number of hidden units, but still requires its fine-tuning. In this paper, we propose to find suitable hyper-parameters concerning the iRBM model by means of meta-heuristic optimization techniques, such as the Particle Swarm Optimization (PSO) [12], Bat Algorithm (BA) [13], Cuckoo Search (CS [14], and the Firefly Algorithm (FFA) [15]. Although one can use any other optimization technique, we opted to use these ones mainly because they are well recognized in the literature, and they do not require computing derivatives as usually demanded by standard optimization techniques.

Recently, Papa et al. [16], [17], [18], Rosa et al. [19], [20] and Rodrigues et al. [21] demonstrated the robustness of these algorithms to the optimization of RBMs and DBNs, but to the best of our knowledge, we have not observed any work that dealt with the problem of iRBM fine-tuning by means of meta-

heuristic techniques to date. Therefore, the main contributions of this paper are twofold: (i) to foster the scientific literature concerning iRBMs, and (ii) to deal with the problem of iRBM hyper-parameter optimization. Additionally, we also considered both standard RBMs and oRBM for comparison purposes concerning the task of binary image reconstruction over two public datasets. The remainder of this paper is organized as follows. Sections II and III present theoretical details about the iRBM and the proposed fine-tuning process, respectively. Section IV discusses the methodology and Section V presents the experimental results. Finally, Section VI states conclusions and future works.

II. Theoretical Background

In this section, we briefly explain the theoretical background related to RBMs, oRBMs and iRBMs.

A. Restricted Boltzmann Machines

Restricted Boltzmann Machines are energy-based stochastic neural networks composed of two layers of neurons (visible and hidden), in which the learning phase is conducted by means of an unsupervised fashion. A naïve architecture of a Restricted Boltzmann Machine comprises a visible layer $\mathbf{v}$ with m units and a hidden layer $\mathbf{h}$ with n units. Additionally, a real-valued matrix $\mathbf{W}_{m\times n}$ models the weights between the visible and hidden neurons, where w_{ij} stands for the weight between the visible unit v_i and the hidden unit h_j.

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

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

where $\mathbf{a}$ e $\mathbf{b}$ stand for the biases of visible and hidden units, respectively.

The probability of a joint configuration $(\mathbf{v},\mathbf{h})$ is computed as follows:

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}, \quad (2)$$

where Z stands for the so-called partition function, which is basically a normalization factor computed over all possible configurations involving the visible and hidden units. Similarly, the marginal probability of a visible (input) vector is given by:

$$P(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}. \quad (3)$$

Since the RBM is a bipartite graph, the activations of both visible and hidden units are mutually independent, thus leading to the following conditional probabilities:

$$P(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{m} P(v_i|\mathbf{h}), \quad (4)$$

and

$$P(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{n} P(h_j|\mathbf{v}), \quad (5)$$

where

$$P(v_i = 1|\mathbf{h}) = \phi\left(\sum_{j=1}^{n} w_{ij} h_j + a_i\right), \quad (6)$$

and

$$P(h_j = 1|\mathbf{v}) = \phi\left(\sum_{i=1}^{m} w_{ij} v_i + b_j\right). \quad (7)$$

Note that $\phi(\cdot)$ stands for the logistic-sigmoid function.

Let $\theta = (\mathbf{W},\mathbf{a},\mathbf{b})$ be the set of parameters of an RBM, which can be learned through a training algorithm that aims at maximizing the product of probabilities given all the available training data $\mathcal{V}$, as follows:

$$\arg\max_{\Theta} \prod_{\mathbf{v}\in\mathcal{V}} P(\mathbf{v}). \quad (8)$$

One can solve the aforementioned equation using the following derivatives over the matrix of weights $\mathbf{W}$, and biases $\mathbf{a}$ and $\mathbf{b}$ at iteration t as follows:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \eta(P(\mathbf{h}|\mathbf{v})\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T), \quad (9)$$

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \eta(\mathbf{v} - \tilde{\mathbf{v}}) \quad (10)$$

and

$$\mathbf{b}^{t+1} = \mathbf{b}^t + \eta(P(\mathbf{h}|\mathbf{v}) - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}}), \quad (11)$$

where η stands for the learning rate. Notice the terms $P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})$ and $\tilde{\mathbf{v}}$ can be obtained by means of the Contrastive Divergence [22] technique, which basically ends up performing Gibbs sampling using the training data as the visible units. Roughly speaking, Equations 22, 23 and 24 employ the well-known Gradient Descent as the optimization algorithm.

B. Ordered Restricted Boltzmann Machines

The ordered Restricted Boltzmann Machine is a variant of the RBM such that the hidden units are trained sequentially, from the left to the right. The current number of trained units at a given time step is represented by the variable $z \leq n$, as depicted in Figure 1.

Given a number z of hidden units, one can compute the energy of the current model as follows:

$$\mathbf{E}(\mathbf{v},\mathbf{h},z) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{z} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{z} (v_i h_j w_{ij} - \beta_j), \quad (12)$$

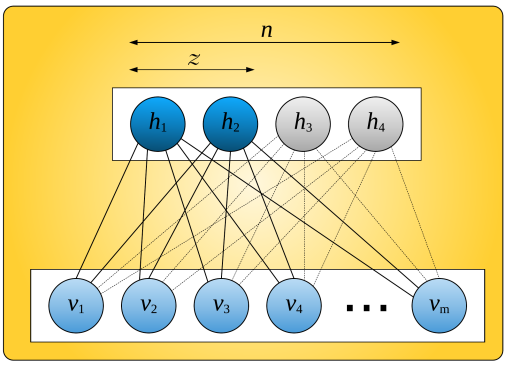


Fig. 1. An oRBM with $z = 2$ and $n = 4$.

where β_j represents the energy penalty associated to the hidden unit h_j. Actually, β_j forces the model to avoid using more hidden units than needed, thus generating smaller networks.

Therefore, the joint probability over $\mathbf{v}$, $\mathbf{h}$ and z is given as follows:

$$P(\mathbf{v}, \mathbf{h}, z) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h},z)}. \tag{13}$$

and the marginal probability is given by:

$$P(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h},z)}. \tag{14}$$

Similarly to the RBM, since Z is intractable in the above equation, the probabilities over $\mathbf{v}$ and $\mathbf{h}$ are estimated by means of Gibbs sampling:

$$P(h_j = 1|\mathbf{v}, z) = \begin{cases} \phi\left(\sum_{i=1}^{m} w_{ij} v_i + b_j\right) & \text{if } j \leq z \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

and

$$P(v_i = 1|\mathbf{h}, z) = \phi\left(\sum_{j=1}^{z} w_{ij} h_j + a_i\right). \tag{16}$$

However, oRBM has an additional information that concerns the maximum number of hidden units that is going to be used, i.e., variable z. Given an input data $\mathbf{v}$, the conditional distribution over the value of z is given as follows:

$$P(z|\mathbf{v}) = \frac{\exp(-F(\mathbf{v}, z))}{\sum_{z'=1}^{n} \exp(-F(\mathbf{v}, z'))}, \tag{17}$$

where $F(\mathbf{v}, z)$ is the so-called "free energy", being computed as follows:

$$F(\mathbf{v}, z) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{z} \left(\psi\left(\sum_{i=1}^{m} w_{ij} v_i + b_j\right) - \beta_j\right), \tag{18}$$

where $\psi(x) = \ln(1 + e^x)$.

Equation 17 tells us we need to consider sampling z from the Markov chain as well. In this case, Gibbs steps alternate between sampling $(h, z) \sim P(\mathbf{h}, z|\mathbf{v})$ and $\mathbf{v} \sim P(\mathbf{v}|\mathbf{h}, z)$. Notice the sampling from $P(\mathbf{h}, z|\mathbf{v})$ can be performed in two steps: $z \sim P(z|\mathbf{v})$ followed by $\mathbf{h} \sim P(\mathbf{h}|\mathbf{v}, z)$.

Finally, the weight matrix $\mathbf{W}$ and the biases $\mathbf{a}$ and $\mathbf{b}$ in the oRBM model are than updated by the following equations:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \eta(\boldsymbol{\xi}\mathbf{v}^T - \tilde{\boldsymbol{\xi}}\tilde{\mathbf{v}}^T), \tag{19}$$

where $\boldsymbol{\xi} = P(\mathbf{h}|\mathbf{v}) \odot (1 - \rho(z|\mathbf{v}))$ and $\tilde{\boldsymbol{\xi}} = P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}}) \odot (1 - \rho(z|\tilde{\mathbf{v}}))$. Notice the operator $\odot$ stands for the element-wise product, and $\rho(z|\mathbf{v}) = [P(z < 1|\mathbf{v}), P(z < 2|\mathbf{v}), \ldots, P(z < n|\mathbf{v})]^T$.

The biases can be updated as follows:

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \eta(\mathbf{v} - \tilde{\mathbf{v}}) \tag{20}$$

and

$$\mathbf{b}^{t+1} = \mathbf{b}^t + \eta(\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}), \tag{21}$$

where $\boldsymbol{\lambda} = (P(\mathbf{h}|\mathbf{v}) - \boldsymbol{\beta}\boldsymbol{\phi}(\mathbf{b})) \odot (1 - \rho(z|\mathbf{v}))$ and $\tilde{\boldsymbol{\lambda}} = (P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}}) - \boldsymbol{\beta}\boldsymbol{\phi}(\mathbf{b})) \odot (1 - \rho(z|\tilde{\mathbf{v}}))$. Notice that $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_z]$, and $\boldsymbol{\phi}$ is the same sigmoid-logistic function as before, but now applied to the array $\mathbf{b}$.

Roughly speaking, the rationale of oRBMs is to perform the training step adding one hidden unit at time, from the left to the right. Since $P(z|\mathbf{v})$ usually increases according to greater values of z (i.e., we have more complex models), the term $(1 - \rho(z|\mathbf{v}))$ decreases monotonically from the left to the right, thus forcing the model using less hidden units (i.e., smaller values of z).

C. Infinity Restricted Boltzmann Machines

The infinity RBM mimics the same growing behavior of the oRBM, but the maximum number of hidden units is not specified. This number increases automatically until its capacity is sufficiently high, which is possible by taking the limit of $n \to \infty$. The model is presented in Figure 2.

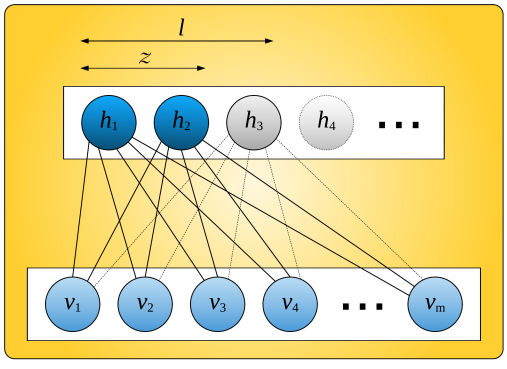


Fig. 2. An iRBM trained previously with $z = 2$ units. There are some non-zero (dashed lines) values connecting the third unit ($l = 3$) that is going to be used for training. All remaining hidden units (i.e., $l > 3$) have zero-valued weights.

The updating equations concerning iRBM are given as follows:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \eta(P(\mathbf{h}|\mathbf{v}, z)\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}}, \tilde{z})\tilde{\mathbf{v}}^T), \quad (22)$$

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \eta(\mathbf{v} - \tilde{\mathbf{v}}) \quad (23)$$

and

$$\mathbf{b}^{t+1} = \mathbf{b}^t + \eta(\boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}}), \quad (24)$$

where $\boldsymbol{\alpha} = (P(\mathbf{h}|\mathbf{v}) - \boldsymbol{\beta}\boldsymbol{\phi}(\mathbf{b})) \odot \boldsymbol{I}_z$ and $\tilde{\boldsymbol{\alpha}} = (P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}}) - \boldsymbol{\beta}\boldsymbol{\phi}(\mathbf{b})) \odot \boldsymbol{I}_z$, and $\boldsymbol{I}_z = [\underbrace{1, \ldots, 1}_{z}, \underbrace{0, \ldots, 0}_{n-z}]^T$.

III. Infinity RBM Fine-Tuning as an Optimization Problem

The proposed approach requires the optimization of three hyper-parameters for both RBM and iRBM, and four parameters for the oRBM, as follows:

- RBM: the learning rate η, the L_1 regularization parameter, and the number of hidden units n;
- oRBM: the learning rate η, the L_1 regularization parameter, the number of hidden units n, and the energy penalty parameter $\boldsymbol{\beta} \in \Re^n$ for each hidden unit; and
- iRBM: the learning rate η, the L_1 regularization parameter, and the energy penalty parameter $\boldsymbol{\beta} \in \Re^n$ for each hidden unit.

Notice the regular learning rate (i.e., $\eta \in \Re$) is used to update the RBM, and the ADAGRAD stochastic gradient technique is used for both oRBM and iRBM [23]. In this case, we have a per-dimension learning rate method, i.e., $\boldsymbol{\eta} \in \Re^n$, with $\epsilon = 10^{-6}$ [10]. This latter parameter stands for a small number to avoid numerical instabilities. Cotê and Larochelle [10] claims that one can throw away the parameter n, thus replacing the RBM model by the iRBM one. However, $\boldsymbol{\beta}$ is still a hyper-parameter to be optimized[1].

Figure 3 depicts the proposed approach to optimize the RBM, oRBM and iRBM models. Roughly speaking, the idea is to initialize all decision variables techniques at random, and then the optimization algorithm takes place. In this work, we used the following ranges concerning the parameters: $n \in [5, 500]$, $\boldsymbol{\eta} \in [0.01, 0.5]$, $\boldsymbol{\beta} \in [0.01, 1.5]$ and $L_1 \in [0.00001, 0.01]$.

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 average negative log-likelihood (NLL) over the training set considering the task of binary image reconstruction as the fitness function. Therefore, we adopted the very same methodology used by [10], but presenting the mean results obtained over 20 runs in order to provide a statistical comparison[2].

In short, the optimization technique selects the set of hyper-parameters that minimize the NLL over the training set considering a dataset of binary images as an input to the model. After learning the hyper-parameters, one can proceed to the reconstruction step concerning the testing images, whose effectiveness is assessed by the NLL method.

[1]Notice the regularization parameter $\boldsymbol{\beta}$ is way less sensitive than the number of hidden units $\mathbf{n}$

[2]Notice the work by Cotê and Larochelle [10] presents the best result over all runs only.

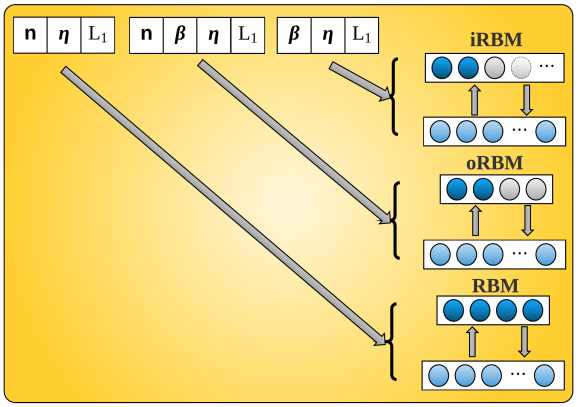


Fig. 3. Proposed approach to model the fine-tuning problem as an optimization task.

IV. Methodology

In this section, we present the methodology employed to evaluate the proposed approach, optimization techniques, datasets, and the experimental setup.

A. Optimization Techniques

Below, we present a brief description of the metaheuristic techniques employed in this paper:

- PSO: Any possible solution is represented as a particle (agent) in a swarm. Each agent has a position that represents a parameter value and velocity vector in the search space. A fitness value is associated with each position, and after some iterations iterations the global best position is selected as the best solution to the problem.
- BA: is a nature-inspired metaheuristic optimization algorithm based on the echolocation behavior of bats. Each bat flies with a randomly velocity, position and frequency. Additionally, they can vary the wavelength and loudness to search for prey/food (best solutions), adjusting the rate of pulses depending on the proximity of their target.
- CS: it is based on the obligate brood parasitic behavior of some cuckoo species in combination with the Lévy flight behavior of some birds and fruit flies. Basically, the algorithm follows three idealized rules: i) each cuckoo lays one egg at a time in randomly chosen nests, ii) the nests with best eggs will carry over to the next generations, and iii) the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0, 1]$. Furthermore, the host bird can either throw away the egg or abandon the nest and build a new one, which means that a new random solution is created.
- FFA: is derived from the fireflies' flash attractiveness when mating partners and attracting potential preys. Basically, the attractiveness of a firefly is computed by its

position related to other fireflies in the swarm, as well as its brightness is determined by the value of the objective function at that position.

Table I presents the parameters used for each aforementioned optimization technique, where five agents (initial solutions) were used for all optimization techniques during 20 iterations for convergence purposes[3]. In regard to PSO, w stands for the inertia weight, and c_1 and c_2 control the step size towards the best local and global solutions, respectively. With respect to BA, f_{min} and f_{max} bound the minimum and maximum frequency values, and A and r denote the the loudness and pulse rate values, respectively. Parameters φ and τ are used to avoid the technique getting trapped from local optima. FFA uses μ and γ, which stand for a random perturbation and the light absorption coefficient, respectively. Variable ς denotes the attractiveness of each firefly. Finally, CS uses Γ to compute the Lévy distribution, ζ for the switch probability (i.e., the probability of replacing the worst nests by new ones), and s for the step size.

TABLE I
PARAMETER CONFIGURATION FOR EACH OPTIMIZATION TECHNIQUE.

Technique	Parameters
PSO	$c_1 = 1.7$, $c_2 = 1.7$, $w = 0.7$
BA	$\varphi = 0.9$, $\tau = 0.9$ $f_{min} = 0$, $f_{max} = 100$ $A = 1.5$, $r = 0.5$
CS	$\Gamma = 1.5$, $\zeta = 0.25$, $s = 0.8$
FFA	$\gamma = 1$, $\varsigma = 1$, $\mu = 0.2$

B. Datasets

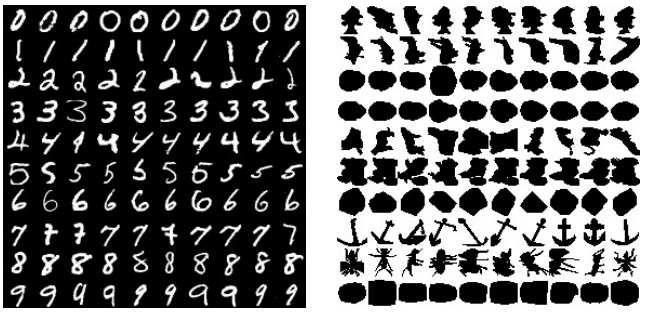
We propose to evaluate the behavior of different optimization techniques to fine-tune RBM/oRBM/iRBM in the context of binary image reconstruction using two public datasets, as described below:

- MNIST dataset[4]: it is composed of images of handwritten digits containing a training set with $70,000$ images from digits '0'-'9', which is split as follows: $50,000$ for training, $10,000$ for validation, and $10,000$ for testing according to [24].
- CalTech 101 Silhouettes Dataset[5]: it is based on the former Caltech 101 dataset, and it comprises silhouettes of images from 101 classes with resolution of 28×28. The dataset is composed of $8,671$ images, such that $4,100$ examples are used for training purpose, $2,264$ for validation, and the $2,307$ remaining for testing.

Figure 4 displays some training examples from the above datasets.

C. Experimental Setup

This work employs a cross-validation procedure with 20 runs in order to provide a statistical analysis by means of the Wilcoxon signed-rank test with significance of 0.05 [25]. The training step is conducted with $5,000$ epochs, with an Annealed Importance Sampling (AIS) evaluation every $1,000$ epochs to keep the best NLL approximation [26].



(a) (b)

Fig. 4. Some training examples from (a) MNIST and (b) Caltech 101 Silhouettes.

For the learning procedure, we used ten Gibbs sampling steps with mini-batches of size 64. In addition, we also considered two learning algorithms: Contrastive Divergence (CD) [22] and Persistent Contrastive Divergence (PCD) [27]. Furthermore, all NLL results were obtained by estimating the log-partition function using AIS with $100,000$ intermediate distributions and $5,000$ chains.

Finally, the codes used to reproduce the experiments of the paper are available on GitHub[6][7]. The experiments were conducted using a Ubuntu 16.04 Linux machine with 16Gb of RAM running an Intel® Core™i7 $-$ 4790 with a frequency of 3.60 GHz and a GPU GeForce® GTX970 with 4GB. The source-codes run on top of Python with Theano [29] and C for the RBM/oRBM/iRBM and optimization approaches, respectively.

V. EXPERIMENTAL RESULTS

In this section, we present the experimental results concerning iRBM, oRBM and RBM hyper-parameter optimization in the task of binary image reconstruction. Additionally, all techniques are compared using two different learning algorithms, i.e., Contrastive Divergence and Persistent Contrastive Divergence. In order to validate the proposed approach, we also considered a random search (RS) as a baseline for hyper-parameter optimization.

Table II presents the averaged NLL results concerning the MNIST dataset, being the values in bold the best results considering the Wilcoxon signed-rank. Although RBM achieved the best results using PSO, both iRBM and the oRBM obtained similar results according to the Wilcoxon signed-rank test, using BA and FFA techniques, respectively. This behavior is expected as it matches the results obtained in [10], which concluded that RBMs are still more accurate, but at the price of having a more sensitive parameter to be set (i.e., the number of hidden units). Also, one can clearly observe the meta-heuristic

[3]Notice these parameters were set empirically.
[4]http://yann.lecun.com/exdb/mnist/
[5]https://people.cs.umass.edu/ marlin/data.shtml
[6]iRBM: http://github.com/MarcCote/iRBM
[7]LibOPT [28]: https://github.com/jppbsi/LibOPT

techniques are able to achieve much more accurate results than the baseline provided by the random search.

It is worth mentioning that PCD has provided better results only for oRBM and iRBM. As a matter of fact, it is arguable that PCD may provide more accurate results than CD, since it does not restart the Markov chain when a new training sample is presented to the network, but it uses the last sampled data from the previous training sample to initiate the chain. However, such behavior was not observed for RBMs, and it quite reasonable to assume that PCD learning can really work well for iRBM and oRBM, since such models may not achieve results so accurate than standard RBMs due to their smaller hidden layers, which means they may have a poorer capacity for learning.

Figure 5 depicts some testing images reconstructed by RBM, oRBM and iRBM. One can observe the images are better reconstructed by RBM, with less noise as well, thus confirming the numerical results presented in Table II. Additionally, one can refer to the network's weights, as displayed in Figure 6, in which a more variety of filters can be observed for standard RBM. Such behavior evidences a greater capacity for learning, which can also be observed for oRBM as well.

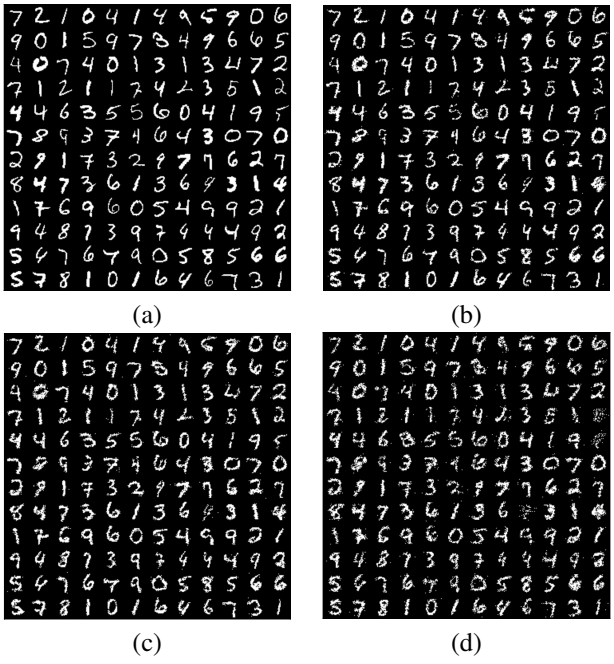


Fig. 5. Random (a) MNIST testing images reconstructed by (b) RBM fine-tuned with FFA and trained with CD, (c) oRBM fine-tuned with FFA and trained with PCD, and (d) iRBM fine-tuned with CS and trained with CD.

Finally, we also considered the computational load of each technique for comparison purposes, as presented in Table III. The fastest optimization technique has been the Cuckoo Search for RBM, oRBM and iRBM, being RBM the fastest of all since its formulation is less complex than oRBM and iRBM.

Table IV presents the average NLL results concerning Caltech 101 Silhouettes dataset. In this case, iRBM achieved the best results with all meta-heuristic techniques using CD for learning, except for CS. Additionally, oRBM obtained the best results with the FFA algorithm. Actually, iRBM trained with CD and optimized by FFA achieved the best result so far. Such results are pretty much interesting, since Caltech dataset poses a greater challenge than MNIST (greater NLL values). Although oRBM and iRBM were not proposed to outperform RBM, one can observe that more accurate models can be obtained by avoiding complex architectures. As a matter of fact, RBMs may be more prone to overfit when one does not choose the number of hidden units properly.

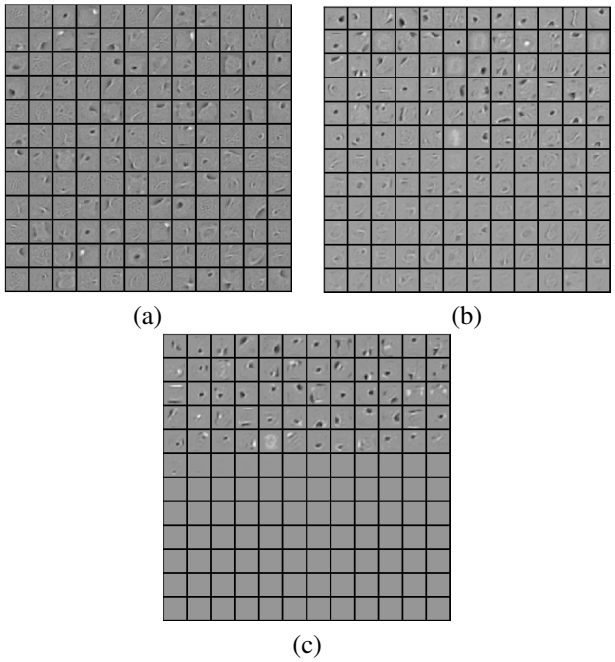


Fig. 6. "The network's mind" considering MNIST dataset: comparing the filters obtained by (a) RBM fine-tuned with FFA and trained with CD, (b) oRBM fine-tuned with FFA and trained with PCD, and (c) iRBM fine-tuned with CS and trained with CD.

Figure 7 depicts some testing images reconstructed by RBM, oRBM and iRBM concerning Caltech 101 Silhouettes dataset. In this case, it is difficult to visualize a clear difference among the techniques. Also, Caltech dataset has much more classes than MNIST, thus resulting in poorer reconstructed images. The weights of the networks are displayed in Figure 8, in which a richer representation in the iRBM's weights can be observed. One can notice a considerable number of full-gray patches, which means they did not learn so much information from the training step.

Table V presents the average computational load concerning Caltech 101 Silhouettes dataset. Considering the worst case, iRBM was around 14.75 times slower than RBM, which showed to be the fastest approach once again. As a matter of fact, both oRBM and iRBM tend to be faster than RBMs for the reconstruction step, since one has less hidden units for computation purposes.

VI. Conclusions and Future Works

This paper addressed the problem of iRBM fine-tuning by means of meta-heuristic techniques. Ordered and Infinity

TABLE II
AVERAGE NLL VALUES CONSIDERING MNIST DATASET.

	RBM		oRBM		iRBM	
	CD	PCD	CD	PCD	CD	PCD
RS	192.84±24.31	195.25±18.61	163.80±28.44	153.16±16.39	160.94±20.54	153.91±18.53
BA	188.07±66.12	220.75±24.35	179.15±37.24	166.13±46.36	184.18±42.05	**165.83±85.07**
CS	154.35±20.82	178.21±20.13	161.33±18.96	156.39±20.71	**149.40±8.34**	**150.71±23.32**
FFA	**125.39±39.59**	243.55±42.11	156.50±23.11	**133.82±40.26**	206.24±18.78	171.92±153.11
PSO	**124.60±44.96**	216.94±41.43	171.24±40.32	164.13±41.18	208.10±37.19	179.25±72.79

TABLE III
AVERAGE TIME (MINUTES) FOR LEARNING HYPER-PARAMETERS CONSIDERING MNIST DATASET.

	RBM		oRBM		iRBM	
	CD	PCD	CD	PCD	CD	PCD
RS	13.07	14.10	20.26	19.72	16.64	19.01
BA	68.34	65.73	121.36	122.48	111.00	300.96
CS	**49.49**	50.29	92.36	92.84	100.56	349.06
FFA	63.37	64.97	126.24	120.36	120.03	411.30
PSO	68.40	68.80	108.67	127.56	143.02	234.91

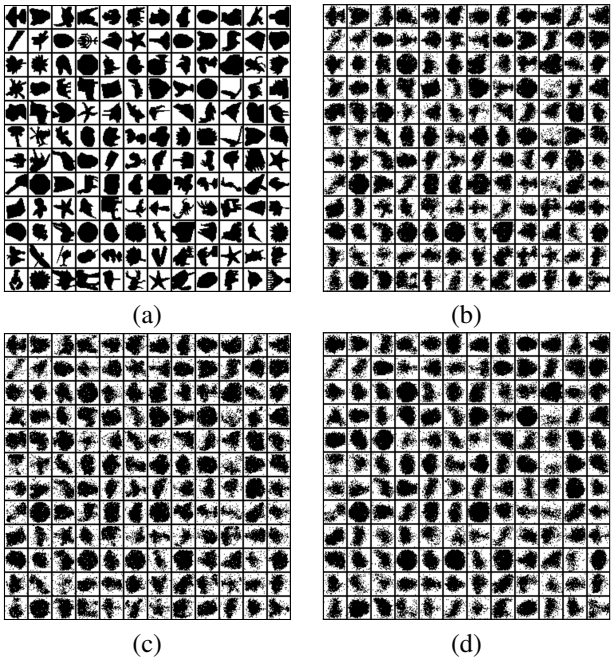


Fig. 7. Random (a) Caltech 101 Silhouettes testing images reconstructed by (b) RBM fine-tuned with FFA and trained with CD, (c) oRBM fine-tuned with FFA and trained with CD, and (d) iRBM fine-tuned with CS and trained with CD.

RBMs are very recent models that avoid choosing the number of hidden units, but at the price of introducing one more variable related to the penalty in adding one more hidden unit to the learning process. However, such parameter is way less sensitive to the number of hidden units, thus requiring less effort by the user and setting up the model.

Experiments over two public datasets concerning the task of binary image reconstruction using four meta-heuristic techniques showed they are suitable for hyper-parameter fine-tuning, being the Cuckoo Search the fastest technique, and FFA one of the most accurate.

In regard to future works, we intend to investigate the suitability of deep versions of both oRBMs and iRBMs, as well as their fine-tuning by means of meta-heuristic techniques.

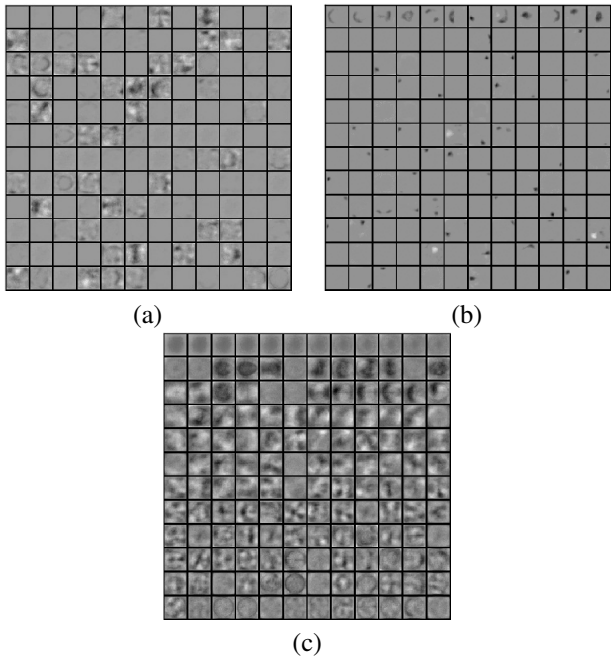


Fig. 8. "The network's mind" considering Caltech 101 Silhouettes dataset: comparing the filters obtained by (a) RBM fine-tuned with FFA and trained with CD, (b) oRBM fine-tuned with FFA and trained with CD, and (c) iRBM fine-tuned with FFA and trained with CD.

ACKNOWLEDGMENT

The authors are grateful to FAPESP grants #2014/12236-1 and #2016/19403-6, Capes, and CNPq grant #306166/2014-3.

REFERENCES

[1] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio, "Learning algorithms for the classification restricted boltzmann machine," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 643–669, 2012.

[2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[3] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[4] R. Salakhutdinov and G. E. Hinton, "An efficient learning procedure for deep boltzmann machines," *Neural Computation*, vol. 24, no. 8, pp. 1967–2006, 2012.

TABLE IV
AVERAGE NLL VALUES CONSIDERING CALTECH 101 SILHOUETTES DATASET.

	RBM		oRBM		iRBM	
	CD	PCD	CD	PCD	CD	PCD
RS	384.30±29.94	432.38±140.15	267.42±28.39	386.03±94.26	274.36±33.99	424.30±187.62
BA	292.08±77.24	609.27±170.72	243.72±24.93	458.95±216.99	**229.32±32.14**	593.33±229.98
CS	349.60±47.13	455.83±104.28	267.82±29.60	448.20±126.03	255.15±18.67	579.29±254.97
FFA	279.88±57.13	629.06±170.37	**237.85±23.63**	420.77±163.16	**218.36±28.54**	486.86±110.73
PSO	315.42±85.29	599.11±140.47	240.40±26.29	411.74±66.69	**237.83±37.83**	554.60±254.15

TABLE V
AVERAGE TIME (MINUTES) FOR LEARNING HYPER-PARAMETERS CONSIDERING CALTECH 101 SILHOUETTES DATASET.

	RBM		oRBM		iRBM	
	CD	PCD	CD	PCD	CD	PCD
RS	13.29	14.10	19.07	20.18	14.81	18.54
BA	20.20	17.68	74.44	54.45	108.00	128.25
CS	16.35	**15.61**	51.49	40.71	67.51	66.61
FFA	18.52	17.71	72.42	56.66	133.48	128.63
PSO	19.05	18.66	74.50	50.61	97.79	230.21

[5] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Advances in neural information processing systems*, 2006, pp. 1345–1352.

[6] L. A. Silva, K. A. P. Costa, P. B. Ribeiro, G. H. Rosa, and J. P. Papa, "Learning spam features using restricted boltzmann machines," *IADIS International Journal on Computer Science and Information Systems*, vol. 11, no. 1, pp. 99–114, 2016.

[7] U. Fiore, F. Palmieri, A. Castiglione, and A. Santis, "Network anomaly detection with the restricted boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, 2013, advances in cognitive and ubiquitous computingSelected papers from the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012).

[8] R. Salakhutdinov, A. Mnih, and G. E. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 791–798.

[9] G. Montufar and N. Ay, "Refinements of universal approximation results for deep belief networks and restricted boltzmann machines," *Neural Computation*, vol. 23, no. 5, pp. 1306–1319, 2011.

[10] M.-A. Côté and H. Larochelle, "An infinite restricted boltzmann machine," *Neural computation*, vol. 28, no. 7, pp. 1265–1288, 2016.

[11] V. Nair and G. E. Hinton, "Implicit mixtures of restricted boltzmann machines," in *Advances in neural information processing systems*, 2009, pp. 1145–1152.

[12] D. Rodrigues, X.-S. Yang, A. N. Souza, and J. P. Papa, *Recent Advances in Swarm Intelligence and Evolutionary Computation*. Cham: Springer International Publishing, 2015, ch. Binary Flower Pollination Algorithm and Its Application to Feature Selection, pp. 85–100.

[13] X.-S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

[14] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[15] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal Bio-Inspired Computing*, vol. 2, no. 2, pp. 78–84, 2010.

[16] 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*, ser. GECCO '15. New York, USA: ACM, 2015, pp. 1449–1450.

[17] 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*, vol. 9, pp. 14–18, 2015.

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

[19] 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*. Cham: Springer International Publishing, 2016, pp. 138–149.

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

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

[22] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[23] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[24] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator." in *AISTATS*, vol. 1, 2011, p. 2.

[25] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, Dec. 1945.

[26] R. Salakhutdinov and I. Murray, "On the quantitative analysis of deep belief networks," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 872–879.

[27] T. Tieleman, "Training restricted boltzmann machines using approximations to the likelihood gradient," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 1064–1071.

[28] J. P. Papa, G. H. Rosa, D. Rodrigues, and X.-S. Yang, "Libopt: An open-source platform for fast prototyping soft optimization techniques," *arXiv preprint arXiv:1704.05174*, 2017.

[29] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements," *arXiv preprint arXiv:1211.5590*, 2012.