GROUP 2316 – Study of a Restricted Boltzmann Machine

Andrea De Vita, Manfredi Miranda, Enrico Lupi, and Francesco Zane
(Dated: April 2, 2023)

Training a restricted Boltzmann machine (RBM) can prove to be a rather complex task, as there are many parameters to fine tune and choices to make in order to obtain the desired results. In this work we will first focus on the effects of employing different types of data encodings and optimizators on the speed and effectiveness of the training, as well as the effects of increasing the number of constrastive divergence steps. Finally, we use the best hyperparametrs to train a RBM and discuss how the pattern in the weights correlates to the structure of the input data.

CONTENTS

INTRODUCTION

The core idea of generative models is to learn a parametric model of the probability distribution from which the data was drawn.[4][5] Typically, they belong to the unsupervised learning approach and therefore it is not possible to evaluate the effectiveness of the model through a cost function based on a comparison of outputs and labels. Energy-based generative models try to overcome this problem through an approach that has many affinities with problem-solving methods in physics: the training goal is to minimize a function representing the total energy of the system.[4] One of the most common methods is entropy maximization (MaxEnt): according to this principle it is possible to evaluate the probability of generating the observed data from the model using the Likelihood function $\mathcal{L}(X, \boldsymbol{\theta})$ or its logarithm.[4] Therefore we can use the negative log-Likelihood as a cost function and minimize it in order to get the best possible set of parameters $\boldsymbol{\theta}$.[4]
In order to perform cost function minimization, it is necessary to evaluate the gradient with respect to the parameters. It turns out that this is the difference of two contributions, the first is data-dependent (positive phase), while the second is model-dependent (negative phase). The interpretation of this result is that the goal of training is to reduce the energy of configurations close to the observed data and raise it for configurations far from them. In this way it is possible to reduce any bias introduced by the model, which represents only an approximation of the unknown probability distribution.[4]
In particular, the evaluation of the negative contribution can be challenging. For *intractable likelihoods* it can be evaluated numerically through Markov chain Monte Carlo (MCMC) methods, such as *Gibbs Sampling*.[4] They are based on the extraction from the model of N fantasy samples, which represent the dataset on which the negative contribution will be calculated, similarly to the positive contribution, which, however, is calculated on the visible data.[4]
In energy-based models, parameters represent the coupling constants between the variables, leading to complex interactions that would be difficult to get during the training. For this reason, some algorithms introduce hidden variables which lead to simpler interactions by increasing the number of degrees of freedom of the system.[4] This reduces the complexity of training and opens up the possibility of imposing restrictions on the type of interactions between real and hidden variables: for example, interactions can be allowed only between variables of different type (hidden-visible links). An example of this type of approach is the Restricted Boltzmann Machine, where *restricted* means the system just described.[4]
In this work a performance study of this type of algorithms is proposed using a dataset represented by 20 features. Each datum is divided into 5 blocks, made of 4 pins that can assume two values (in general up and down): in each block there is only one pin up and thus the physical system has 4 possible energy levels per block, one for each possible pin up position. Moreover, blocks are distributed so that each polar block (where the up pin is one of the first two) is followed by a non-polar block (where the up pin is one of the last two) and viceversa.

METHODS

More formally, we can define an RBM as a Markov random field with pairwise interactions defined on a bipartite graph of two non-interacting layers of variables and refer to visible variables as $\mathbf{v} = v_i$, $i = 1, ..., N_v = 10.000$, and to hidden variables as $\mathbf{h} = h_j$, $j = 1, ..., M = 3$. In this work we use Bernoulli layers, i.e. with each node represented as a binary value (both {0;1} and {-1;1} encodings are discussed). Using $\mathbf{w}$ to indicate the weight matrix, $\mathbf{a}$ for the visible biases and $\mathbf{b}$ for the hidden ones, we can

compute the energy function as

$$E_{\mathbf{w},\mathbf{a},\mathbf{b}}[\mathbf{v},\mathbf{h}] = -\sum_i a_i v_i - \sum_\mu b_\mu h_\mu - \sum_{i\mu} v_i w_{i\mu} h_\mu \quad (1)$$

The Boltzmann distribution is given by

$$p[\mathbf{v},\mathbf{h}]_{\mathbf{w},\mathbf{a},\mathbf{b}} = \frac{exp(-E_{\mathbf{w},\mathbf{a},\mathbf{b}}[\mathbf{v},\mathbf{h}])}{Z}, \quad (2)$$

with $Z = \sum_{\{\mathbf{v},\mathbf{h}\}} exp(-E_{\mathbf{w},\mathbf{a},\mathbf{b}}[\mathbf{v},\mathbf{h}])$ being the partition function.
Expressing the approximated probability density function produced by the RBM as $p_{\mathbf{w},\mathbf{a},\mathbf{b}}(\mathbf{v})$, we can define the Log-Likelihood (LL) of the learnt model with respect to a dataset $\mathcal{D} = \{\mathbf{v}^{(1)}, ..., \mathbf{v}^{(N)}\}$ as

$$\mathcal{L}_\mathcal{D}(\mathbf{w},\mathbf{a},\mathbf{b}) = \langle \log p_{\mathbf{w},\mathbf{a},\mathbf{b}}(\mathbf{v}) \rangle_\mathcal{D} \quad (3)$$
$$= -\langle E_{\mathbf{w},\mathbf{a},\mathbf{b}}(\mathbf{v}) \rangle_\mathcal{D} - \log Z_{\mathbf{w},\mathbf{a},\mathbf{b}} \quad (4)$$

This quantity will be maximized during the RBM training by the chosen gradient ascent algorithm; in order to do so, we need to compute the following derivatives:

$$\frac{\partial \mathcal{L}}{\partial w_{i\mu}} = \langle v_i h_\mu \rangle_\mathcal{D} - \langle v_i h_\mu \rangle_\mathcal{H} \quad (5)$$
$$\frac{\partial \mathcal{L}}{\partial a_i} = \langle v_i \rangle_\mathcal{D} - \langle v_i \rangle_\mathcal{H} \quad (6)$$
$$\frac{\partial \mathcal{L}}{\partial b_\mu} = \langle h_\mu \rangle_\mathcal{D} - \langle h_\mu \rangle_\mathcal{H} \quad (7)$$

where $\langle \cdot \rangle_\mathcal{D}$ denotes the average over the dataset $\mathcal{D}$, i.e. the original "authentic" data, while $\langle \cdot \rangle_\mathcal{H}$ denotes the average over the Boltzmann measure in eq. 2.[4] In practice, the first terms of the gradient are calculated on a randomized mini-batch of $N_{mini} = 500$ samples, while the second terms are calculated as the averages over a fantasy dataset, obtained from the mini-batch by means of the Contrastive Divergence (CD) technique by Hinton, with varying number of steps k.[3] It is best to implement a backward step of contrastive divergence so as to preserve the one-hot encoding structure of the original data. In order to do so, for each block I of size $A = 4$ we compute the energy of the configuration ξ as

$$E_\xi = \sum_{k\in I} [a_k + \sum_\mu w_{k\mu} h_\mu] v_k^\xi, \quad \xi = 1, ..., A \quad (8)$$

to which corresponds the probability

$$p_\xi = \frac{e^{-E_\xi}}{\sum_{\lambda=1}^A e^{-E_\lambda}} \quad (9)$$

and then extract randomly one configuration based on the probabilities we just computed.

Another feature that can be exploited during the training phase of RBM is the so called *centering trick*.[1] It consists in shifting the visible and hidden variables by some offset parameters $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$. This model was introduced by Montavon and Müller with $\boldsymbol{\mu} =< \boldsymbol{v}_{data} >$ and $\boldsymbol{\lambda} =< \boldsymbol{h}_{data} >$ and it leads to the following expression of the energy for the corresponding *centered binary RBM*:

$$E_{\mathbf{w},\mathbf{a},\mathbf{b}}[\mathbf{v},\mathbf{h}] = -\sum_i a_i(v_i - \mu_i) - \sum_\mu b_\mu(h_\mu - \lambda_\mu) - \\ - \sum_{i\mu} (v_i - \mu_i) w_{i\mu} (h_\mu - \lambda_\mu) \quad (10)$$

Also the derivatives of the LL will depend on the new shifted variables $\boldsymbol{v} - \boldsymbol{\mu}$ and $\boldsymbol{h} - \boldsymbol{\lambda}$.[1]
In practice the offset parameters are updated on a given mini-batch through a convex combination of a term calculated in the current mini-batch and a term given by the previous step:

$$\boldsymbol{\mu} \leftarrow (1 - \xi_\mu)\boldsymbol{\mu} + \xi_\mu \boldsymbol{\mu}_{batch} \quad (11)$$
$$\boldsymbol{\lambda} \leftarrow (1 - \xi_\lambda)\boldsymbol{\lambda} + \xi_\lambda \boldsymbol{\lambda}_{batch} \quad (12)$$

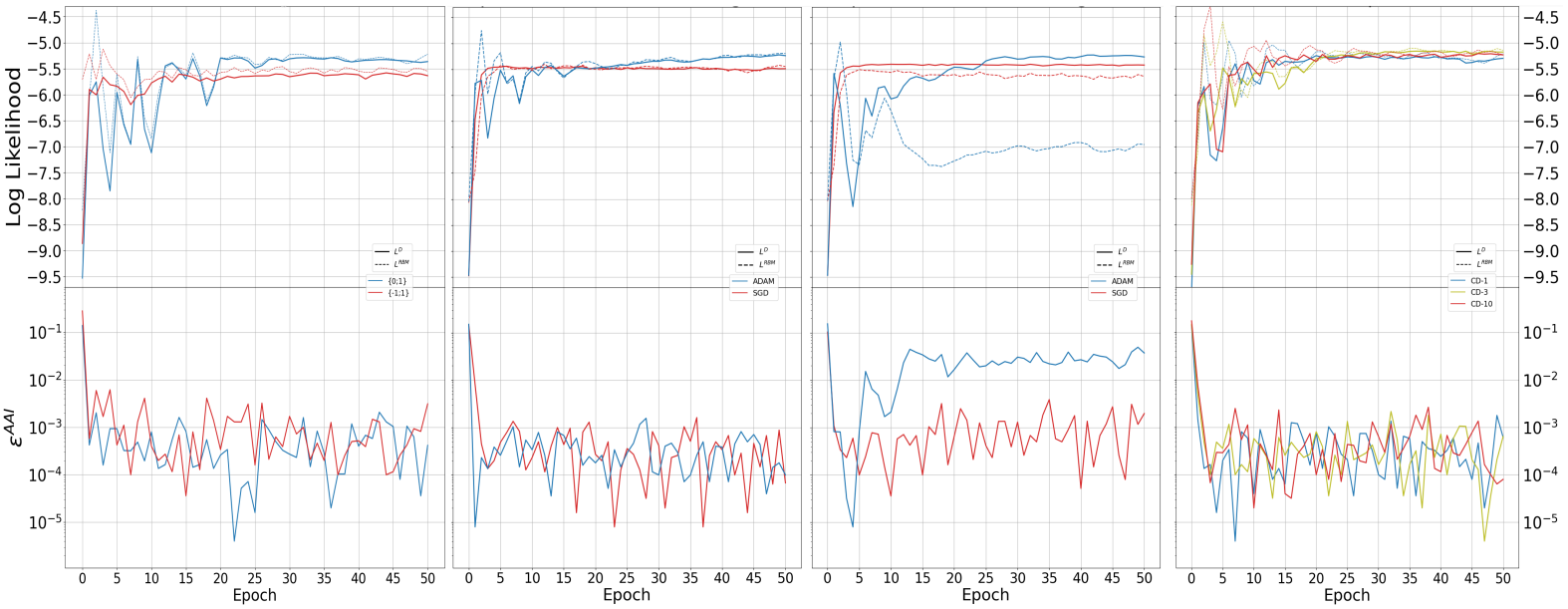
with $\xi_\mu \approx \xi_\lambda \approx 0.01$.[1]

Two different observables are used in this work to evaluate and quantify the performance of the RBM. First, we employ the Log-Likelihood, as defined in eq. 4; notice that, given the low complexity of our model, all $A^G \cdot 2^M \sim 10^5$ combinations necessary to exactly compute the partition function can be explored. Both the LL of the whole original dataset ($\mathcal{L}^\mathcal{D}$) and of a dataset of equal size of generated samples ($\mathcal{L}^{\mathcal{RBM}}$) are computed: the ideal outcome would be that these values are both high and similar to each other. Second, we use the error in the Adversarial Accuracy Indicator (AAI) ε^{AAI}, introduced in [6]. We first define two sets of N_s samples: a 'target' set containing only generated samples $T \equiv \{v_{RBM}^{(m)}\}_{m=1}^{N_s}$ and a 'source' with samples from the dataset $S \equiv \{v_\mathcal{D}^{(m)}\}_{m=1}^{N_s}$; both sets contain $N_S = 250$ samples. Then, for each sample m in groups $\{T, S\}$, we compute distance with the closest sample (using the taxicab distance) to the set groups $\{T, S\}$, and define this value $d_{TS}(m)$, $d_{TT}(m)$, $d_{ST}(m)$ or $d_{SS}(m)$ based on the pair of sets chosen.[2] We can estimate the frequency at which we find the nearest neighbors of a data point in the same set that already contained that point as

$$\mathcal{A}_S = \frac{1}{N_S} \sum_m^{N_S} f(d_{SS}(m), d_{ST}(m)) \quad (13)$$

$$\mathcal{A}_T = \frac{1}{N_S} \sum_m^{N_S} f(d_{TT}(m), d_{TS}(m)) \quad (14)$$

where $f(a, b)$ is equal to 1 if $a < b$, 0.5 if $a = b$ and 0 otherwise. The optimal result is achieved when both values converge to 0.5, as the generated samples would be indistinguishable from the real ones: we can thus condense



(a) Encoding comparison

(b) Optimizers comparison, without (left) and with (right) centering trick

(c) CD-k comparison

FIG. 1: LL and ϵ^{AAI} evolution starting from the standard configuration ({0;1}, ADAM, no centering, CD-1) and varying the following parameters: (a) encodings {0;1} and {-1;1}, (b) optimizers and centering combinations, (c) number of contrastive divergence steps.

this information in the error:

$$\varepsilon^{AAI} = (\mathcal{A}_S - 0.5)^2 + (\mathcal{A}_T - 0.5)^2 \tag{15}$$

For the analysis we run various machines with different parameters and settings for 50 epochs in order to study their evolution. The standard configuration is with {0;1} encoding, using ADAM as the optimizer with learning rate of $\eta^{ADAM} = 0.1$, no centering trick and CD-1; we also experiment with {-1;1} encoding, using stochastic gradient descent (SGD) with adaptive learning rate of $\eta_{t+1}^{SGD} = \eta_t^{SGD}/(0.01 \cdot \eta_t^{SGD} + 1)$ with $\eta_0^{SGD} = 1.0$, centering trick and a higher number of CD steps (3 and 10).

RESULTS

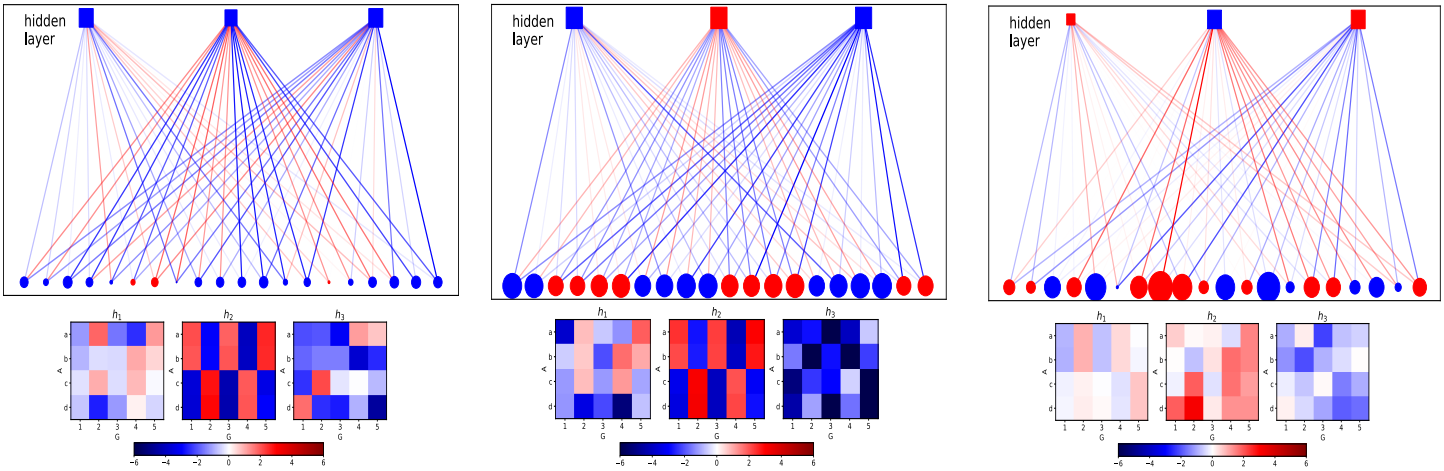
We study the behaviour of the LL and of ε^{AAI} with respect to the various hyper-parameters; in particular we will confront the effects of:

1. using {0;1} or {-1;1} encoding (fig. 1a)
2. exploiting the centering trick with different optimization methods (fig. 1b)
3. varying the number of CD steps (fig. 1c)

We may notice that the ideal behaviour of LL and of ε^{AAI} is shown in all configurations and that the evaluators become stable just after few epochs, justifying *a posteriori* our choice of stopping after only 50 epochs. These two pieces of information suggest that the problem at hand is indeed simple and that the hyper-parameters do not influence the performance of the machine much. We also notice that the value of ε^{AAI} is very small, possibly due to the fact that the visible variables are simple so there is not much variability in the distances.

The only case where the behaviours of the evaluators differ from the norm is when using the centering trick: ADAM becomes more unstable and takes more epochs to stabilize, the likelihood of the fantasy data set is much lower than that of the original data and ε^{AAI} is higher. Another interesting observation can be made for the number of CD steps. We expect from theory that an increase in this parameter would make the fantasy data less reliant on the original data and thus a better simulation for the model. In fact, we observe that the performances are not affected by the number of steps, and CD-1 is already enough to obtain good results.

We now study in more details the statistical fluctuations of the evaluators during the training in the standard configuration, which is observed to be the optimal configuration as well. In fig. 3, obtained by averaging and calculating the errors over 10 randomized



(a) {0;1} encoding, centering (b) {0;1} encoding, no centering (c) {-1;1} encoding, no centering

FIG. 2: Plots of the RBM structure after 50 epochs of training, highlighting the pattern of the weights. All three are obtained using ADAM an CD-1.

runs, we observe that the likelihood varies greatly in the first ~ 25 epochs due to the randomness of the training algorithm, while afterwards it stabilizes and the errors become small. A similar point can be made for ε^{AAI}, even though the relative errors are slightly greater. Given the small standard error in the later epochs, we are justified in only studying the performances over one run in the previous analysis.

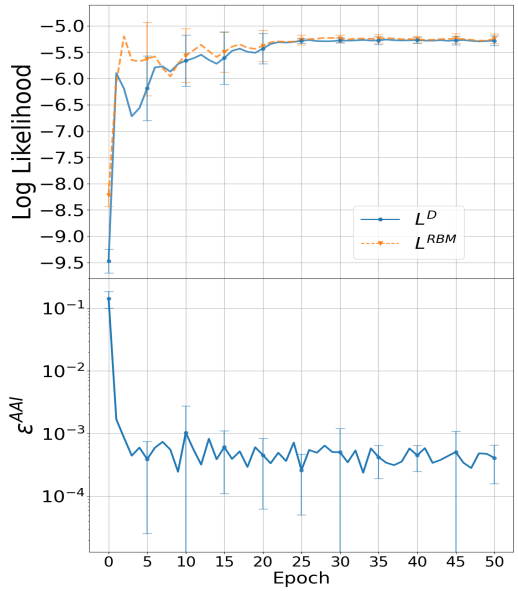


FIG. 3: Evolution of LL and ε^{AAI} as functions of the epoch, obtained by averaging and computing standard deviation across 10 different training runs, using the standard configuration: {0;1}, ADAM, no centering, CD-1

Next we focus on the structure of the graphs and weights themselves. In figs. 2a and 2b we notice that the couplings between weights and bias of the visible units changes whether or not we apply the centering technique. In the latter case the weights and the bias are coupled in the same pattern, while in the former such relations are not present anymore. This result is what we expect *a priori*, as no configuration of visible variables is preferred over the others and thus the bias structure should be random. As we can see in fig. 2c, {-1;1} encoding removes this coupling as well.

Another feature that is taken into account is the readability of the weights, meaning that they should show some pattern that reproduces the general structure of the dataset. In figs. 2b and 2c we may notice that the choice of the encoding can make that pattern more or less obvious. The results highlight that {0;1} has more expressive weights since the colour pattern manifestly shows the expected polar-non polar (or viceversa) alternation as in the original data.
Due to the simplicity of the dataset structure we can see that ADAM reduces the number of useful hidden variables to one, as the weights for two out of three hidden variables are either evanescent or show no visible pattern (see fig. 2).

CONCLUSIONS

In this work we study how the choice of hyper-parameters influences the performance and the interpretability of the results of a Restricted Boltzmann Machine given a particularly structured dataset. We

show that, due to the simplicity of the original dataset, the performances remain mostly unchanged at optimal levels for every configuration, with a slight lead by the combination of {0;1} encoding, ADAM optimizer, no centering trick and CD-1. In particular, the trends of the estimators for different CD steps appear to be very similar. In theory, increasing the CD steps should lead to better performances; however, different indicators in different configurations show that we are in a local maximum of the likelihood, so we cannot improve the performance anymore.

We further show that the choice of encoding has great repercussions on the readability of the weights. Contrary to the {-1;1} case, observing their pattern with {0;1} encoding let us correctly reconstruct the polar-non polar (or viceversa) alternation of the original data, as specified in the introduction of the paper. In addition, the use of the *centering trick* allows this pattern between weights and biases to be decoupled, since without this trick the bias and weights structures result identical.

As for the optimizer, the data show that the use of ADAM promotes pattern learning. In particular, its use shows that the number of degrees of freedom chosen, which in this case is represented by M=3, can be reduced. This suggests that further tests may be done with ADAM and M=1.

[1] M. Bortoletto. Study of performances for restricted boltzmann machines. Master's thesis, University of Padua - Physics of Data, 2021.

[2] A. Decelle et al. Equilibrium and non-equilibrium regimes in the learning of restricted boltzmann machines. *Journal of Statistical Mechanics: Theory and Experiment*, 2022(11), Nov 2022.

[3] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8), Aug 2002.

[4] P. Mehta et al. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810, 2019. A high-bias, low-variance introduction to Machine Learning for physicists.

[5] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014.

[6] A. Yale et al. Generation and evaluation of privacy preserving synthetic health data. *Neurocomputing*, 416, Apr 2020.