GROUP 9 – Restricted Boltzmann Machines to Learn Patterns in Bit Sequences

Davide Bacilieri, Lorenzo Barbiero, Guglielmo Bordin, and Alessio Pitteri
(Dated: 26th March 2023)

In the context of unsupervised learning Restricted Boltzmann Machines (RBMs) excel in the fields of pattern recognition and denoising, this leads to application in various fields such as Information Theory or, in the case of this paper, biological datasets. In this paper a RBM is built from an Ising-like model, implementing a proper partition function, energy function and log likelihood function that allows learning. Combining it with more proper data analysis methods such as clever gradiend descents the effectiveness of the framework is evaluated on plausible datasets for amino acids recognition, both for known and unknown sequences.

INTRODUCTION

Restricted Boltzmann Machines belong to the class of energy-based generative models. They are *generative* in the sense that they attempt to reproduce the underlying probability distribution that governs the generation of the training data, in order to generate new samples that could have been drawn from the same dataset [1].

One use-case of this kind of models is in biology, with protein sequences [2, 3]: RBMs can efficiently and reliably learn complex and recurring patterns hidden in the sequences of amino acids. Indeed, we worked on a very rudimentary implementation of this concept: our dataset consisted of several short sequences of 1 and 0 bits, encoding patterns of polar and non-polar amino acids. The idea is to filter out the noise and inconsistencies from the sequence, and grasp the underlying pattern by having the RBM generate a "clean" sequence after training. Of course, this has no pretence of actual resemblance to reality; it rather serves as a proof of concept to show the capabilities of the learning model.

The learning framework of RBMs strongly resembles many Ising-like models of statistical physics. Taking the same equations, the problem is reframed as the iterative learning of the parameters of a variational distribution that should approximate the true distribution of the data.

Another thing that we borrow from physics is the concept of *hidden variables*. In Ising-like models, one usually performs a Hubbard–Stratonovich transformation to avoid having to deal with the complex quadratic interactions between spins, by shifting to the description of a system of non-interacting spins immersed in a Gaussian field [4]. Similarly, in the context of Boltzmann learning, we can simplify the complex relationships between the variables in the training data by making them interact in a "controlled" manner with an additional layer of fictitious variables.

METHODS

We will borrow the notation from the 2019 review on Machine Learning by Mehta et al. [1]. Like we have said, Restricted Boltzmann Machines are trained with the goal of best approximating a joint probability distribution p of the visible variables $\boldsymbol{v}$ and the hidden variables $\boldsymbol{h}$, which, carrying on the analogy with statistical physics models, is written as a Boltzmann distribution

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z}, \tag{1}$$

where the energy function $E(\boldsymbol{v}, \boldsymbol{h})$ takes the form

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{N} a_i v_i - \sum_{\mu=1}^{M} b_\mu h_\mu - \sum_{i=1}^{N}\sum_{\mu=1}^{M} W_{i\mu} v_i h_\mu. \tag{2}$$

We will refer to the variables $W_{i\mu}$ as *weights*, and to the coefficients a_i and b_μ as *biases*. These are the parameters to learn, and we will collectively denote them with $\boldsymbol{\vartheta}$.

The actual training, that is, the iterative approximation of the true p with a parametrized $p_{\boldsymbol{\vartheta}}$, is performed through the Maximum Likelihood Estimation procedure. This involves the maximization – or minimization of the negative – of the *log-likelihood* $\mathcal{L}(\boldsymbol{\vartheta})$ of the model:

$$\mathcal{L}(\boldsymbol{\vartheta}) = -\langle E(\boldsymbol{\vartheta})\rangle_{\text{data}} - \log(Z(\boldsymbol{\vartheta})). \tag{3}$$

In practice, this translates to the application of (stochastic) gradient descent methods to the following set of equations:

$$\frac{\partial \mathcal{L}(\boldsymbol{\vartheta})}{\partial W_{i\mu}} = \langle v_i h_\mu\rangle_{\text{data}} - \langle v_i h_\mu\rangle_{\text{model}}, \tag{4}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\vartheta})}{\partial a_i} = \langle v_i\rangle_{\text{data}} - \langle v_i\rangle_{\text{model}}, \tag{5}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{\vartheta})}{\partial b_\mu} = \langle h_\mu\rangle_{\text{data}} - \langle h_\mu\rangle_{\text{model}}. \tag{6}$$

Here the expectation values with respect to the data are the empirical averages on the training samples, while the expectation values with respect to the model need to be estimated by drawing samples from the parametrized distribution $p_{\boldsymbol{\vartheta}}$.

In our case, the gradient descent procedure was conducted on mini-batches of 500 samples with the *RMSProp* algorithm (as described in [1]). To calculate the expectation values in Eqs. (4)–(6), we performed three steps of

block Gibbs sampling (so, a *three-step Contrastive Divergence*), each step being a forward sample from the visible layer to the hidden layer, plus a backward sample in the opposite direction.

We should now specify that our training data consisted of many sequences of one-hot encoded blocks of six bits. These blocks were interpreted as encodings for the different amino acids chaining to form a "protein". So, while the hidden layer posed no sampling problems – the h_μ's are simple binary variables – the backward sample on the visible layer required some special attention due to the one-hot encoding.

Let us define the possible values of each data block in $\boldsymbol{v}$ as $\boldsymbol{v}^{(k)}$, with $k = 1, \dots, 6$ denoting the position of the positive bit. During the backward sampling, we have to clamp the variables in the visible layer to every possible configuration and compute their respective probability. Therefore, a given block has six different probabilities associated to it:

$$p(\boldsymbol{v}^{(k)} \mid \boldsymbol{h}) = \frac{1}{Z} \exp\left[a_k + \sum_{\mu=1}^{M} (b_\mu + W_{k\mu}) h_\mu\right]. \quad (7)$$

The partition function Z is the sum of the six probabilities, so the term with $\boldsymbol{b}$ cancels out. In practice, for each block of 6 bits in $\boldsymbol{v}$, we computed the cumulative probabilities C_k as $\tilde{C}_k/\tilde{C}_6$ with

$$\tilde{C}_k = \sum_{i=1}^{k} \exp\left(a_i + \sum_{\mu=1}^{M} W_{i\mu} h_\mu\right) \quad (8)$$

Then, we generated a random number r, chose ℓ such that $C_{\ell-1} < r < C_\ell$, and replaced the block with $\boldsymbol{v}^{(\ell)}$.

We also explored the possibility to work with "spins" instead of bits, i.e. $+1/-1$ binary variables. Eq. (7) has to be reworked a bit, giving

$$p(\boldsymbol{v}^{(k)} \mid \boldsymbol{h}) = \frac{1}{Z'} \exp\left[-\sum_{j=1}^{6} (-1)^{\delta_{jk}} \left(a_j + \sum_{\mu=1}^{M} W_{j\mu} h_\mu\right)\right], \quad (9)$$

where Z' is the partition function without the terms with $\boldsymbol{b}$ that cancel out. We can use a similar expression as Eq. (8) but with a factor of 2 multiplying the argument of the exponentials. The reason for this becomes clear if we factor out

$$\exp\left[-\sum_{j=1}^{6} \left(a_j + \sum_{\mu=1}^{M} W_{j\mu} h_\mu\right)\right] \quad (10)$$

from Eq. (9).

Before moving on to the results discussion, we should mention that we implemented the Adversarial Accuracy Indicator from [5] to provide a meaningful rating for our trained model. This indicator helps gauge the level of similarity between the generated model and the original distribution [6]. Looking at two sample sets, one from the original dataset and one generated by the model, we counted how many times a given data point's nearest neighbour came from the same set and how many times it came from the other set. The idea is that if the sets came from the same probability distribution, the ratio of nearest neighbours of the same type to the total number of samples would approach 1/2. We will discuss this further later on.

RESULTS

Moreover, in this study case, the most useful quantity to visualize are the weights beacuse they suggest us an underlying pattern. Just by looking at the following heatmap we can suppose that the amino acids are of two different types: first type if there is $\boldsymbol{v}^{(k)}$ with $k = 1, 2, 3$, second type if $k = 4, 5, 6$. Using this notation we can visualize the denoised original data in a compact table. Denoised sequence are computed using the biases and the weights of the very last epoch with a single divergent step introducing a parameter β which multiply rescale the energy. This is of course the analogous of the temperature in an Ising like system. In particular this system can be mapped in an Hopfield model and so, setting to a very low value the temperature, we expect to obtain a high overlap between the generated spins and the misterious patterns. Looking at the table we can guess the unknown distribution of the blocks in a sequence 11221122 etc

[1] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, A high-bias, low-variance introduction to machine learning for physicists, Physics Reports **810**, 1 (2019).

[2] J. Tubiana, S. Cocco, and R. Monasson, Learning protein constitutive motifs from sequence data, eLife **8**, e39397 (2019).

[3] J. Tubiana, S. Cocco, and R. Monasson, Learning Compositional Representations of Interacting Systems with Restricted Boltzmann Machines: Comparative Study of Lattice Proteins, Neural Computation **31**, 1671 (2019), https://direct.mit.edu/neco/article-pdf/31/8/1671/1053381/neco_a_01210.pdf.

[4] J. Hubbard, Calculation of partition functions, Phys. Rev. Lett. **3**, 77 (1959).

[5] A. Yale, S. Dash, R. Dutta, I. Guyon, A. Pavao, and K. P. Bennett, Generation and evaluation of privacy preserving synthetic health data, Neurocomputing **416**, 244 (2020).

[6] A. Decelle, C. Furtlehner, and B. Seoane, Equilibrium and non-equilibrium regimes in the learning of restricted boltzmann machines, Journal of Statistical Mechanics: Theory and Experiment **2022**, 114009 (2022).