GROUP 05 – Restricted Boltzmann Machines as a tool for pattern discovery in short aminoacid sequences

Tommaso Bertola, Giacomo Di Prima, Giuseppe Viterbo, and Marco Zenari
All authors contributed equally to the project
(Dated: April 2, 2023)

Nowadays, model explainability in Machine Learning algorithms is an issue often overlooked. However, in some circumstances, a human-understandable explanation of algorithm decision-making behavior is required. Typical examples concern the understanding of patterns hidden in the data, where a standard statistical analysis would fail. In order to showcase one model, we test Restricted Boltzmann Machines as a tool to discover already known correlations in simplified protein structures. We test the explainability of RBMs by checking the recurring patterns appearing in the trained weights. We also verify the RBM susceptibility to the tuning of its hyperparameters: the number of hidden units, the encoding representation, CD-n steps, training and generating amplitudes.

INTRODUCTION

The aim of this paper is to show how an Unsupervised Learning model can be used to recognise and generalise patterns in a given data set of protein strands made of 5 amino acids.

More specifically, the number of different types of amino acids is 4 and each of them is encoded as a sequence of 4 digits, 1s and 0s. The position of digit 1 defines the type of amino acid: $A_1 = \{1000\}, A_2 = \{0100\}, A_3 = \{0010\}, A_4 = \{0001\}$.

A possible approach involves Restricted Boltzmann Machines, RBMs, a tool that takes inspiration from Statistical Mechanics. The objective is to maximise the LogLikelihood $\mathcal{L}$ function that encodes the similarity between the empirical distribution of data and the one generated by the model. This procedure leverages the Maximum Entropy principle first defined by Jaynes [1].

In practise, the model consists of iteratively updating the weights of the edges of a bipartite network made up of visible and hidden units. The network structure is as follows: the number of visible units corresponds to the number of digits that each sample given as input contains, while the number of hidden units can be tuned. The relationships between visible $\{v_i\}_{i=1\dots V}$ and hidden $\{h_\mu\}_{\mu=1\dots H}$ units are encoded in the weight matrix $W_{i\mu}$. In addition to the matrix entries W, each component of the bias vectors $\{a_i\}_{i=1\dots V}$ and $\{b_\mu\}_{\mu=1\dots H}$ is also iteratively updated to encapsulate the patterns in the data.

Similarly to the Hopfiled model, the energy function takes the following form, where $\boldsymbol{v}$ and $\boldsymbol{h}$ will take values in $\{0, 1\}$.

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_i a_i v_i - \sum_\mu b_\mu v_\mu - \sum_{i,\mu} v_i W_{i\mu} h_\mu \qquad (1)$$

The energy E is used to assign values to $\boldsymbol{h}_\mu$, according to the Boltzmann weights in the positive phase. Each h_μ is set to 1 or 0 according to the probability:

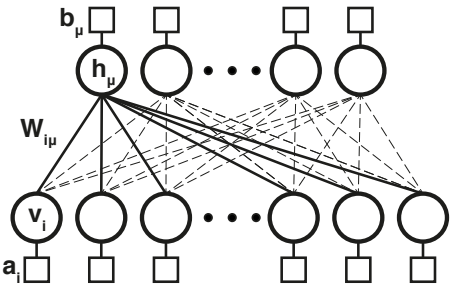

FIG. 1. Network structure. Visible and hidden units are circles, bias values are squares. Connections from $h_{\mu=1}$ to all visible units are highlighted

$$\begin{aligned} p(h_\mu = 1, 0|\boldsymbol{v}, \gamma) =& \frac{e^{-\gamma E(\boldsymbol{v}, h_\mu=1,0)}}{e^{-\gamma E(\boldsymbol{v}, h_\mu=1,0)} + e^{-\gamma E(\boldsymbol{v}, h_\mu=0,1)}} \\ =& \sigma\left(\gamma(b_\mu + \sum_i W_{i\mu} v_i)\right) \end{aligned} \qquad (2)$$

$$\gamma = \text{GAP} \cdot \text{AMP} \qquad (3)$$

Where GAP = 1 is the energy separation level and the amplitude AMP influences the temperature-related noise contribution of the system.

Similarly, in the negative phase, a properly constructed probability is used to create a new set of $\boldsymbol{v}$. These new $\boldsymbol{v}$ are called *Fantasy Particles* and allow the computation of $\mathcal{L}$ and eventually its gradient:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \langle O_i(\boldsymbol{v}, \boldsymbol{h}) \rangle_{\text{data}} - \langle O_i(\boldsymbol{v}, \boldsymbol{h}) \rangle_{\text{model}} \qquad (4)$$

The quality of the learnt distribution is assessed by computing the Jensen-Shannon divergence between the original data distribution and the Fantasy one.

METHODS

Architecture and Weights initialisation – The number of visible units V is set to 20, the size of each

input protein. The number of hidden units H is set to 3. For comparison reasons, the results with $H = 6$ are also reported in the following section. $W_{i\mu}$ and a_i entries are randomly initialised using the Glorot algorithm with $\sigma = 2\sqrt{1/(V+M)}$. b_μ are set to 0.

Data representation – The model can be trained by interpreting the digits with $\{0, 1\}$ or $\{-1, 1\}$ encodings. Changing the representation from $\{0, 1\}$ to $\{-1, 1\}$ implies the need to modify the energy separation between states, namely GAP, which goes from 1 to 2. In computing the probability as in Equation 6, we must therefore multiply the argument of the exponential by GAP. Whenever the $\{-1, 1\}$ encoding is used, $A_{j=1,2,3,4}$, h_μ and v_i are redefined accordingly.

One-hot encoding – A key feature that the model implements is the *one-hot* encoding which forces the generation of Fantasy Particles in such a way as to respect the amino acid encodings. This is achieved by splitting each protein strand into $K = \{1, ..., 5\} \ni k$ fragments, one for each amino acid site, and deciding according to the probability $p_j^k(\boldsymbol{A_j}|\boldsymbol{h})$. The probability $p_j^k(\boldsymbol{A_j}|\boldsymbol{h})$ is defined as follows, with k being the site index that restricts the sum to the kth fragment. j is the index related to all four possible cases.

$$E_j^k = \left(\sum_{l=4k-3}^{4k} \sum_{\mu} W_{l\mu} h_\mu + a_l \right) A_j \tag{5}$$

$$p_j^k(\boldsymbol{A_j}|\boldsymbol{h}) = \frac{e^{-\gamma E_j^k}}{\sum_{j'} e^{-\gamma E_{j'}^k}} \tag{6}$$

Algorithm 1 One-hot Encoding

```
Chose representation: {0,1} or {−1,1}
A_j ← one-hot encoded states
E_j^k ← as Eq 5
p_j^k ← as Eq 6
Compute cumulative probability from p_j^k
K ← sites in protein strand
n_k ← uniform random number k ∈ K
for k in K do
    Find within which jth cumulative step n_k falls
    v_{f_k} ← A_j corresponding to jth step
end for
v_f ← Concatenate v_{f_k}
```

Contrastive divergence – The computation of the $\mathcal{L}$ gradient requires RBMs to produce fantasy particles independently from the input data through Gibbs sampling. In reality, the sampling is stopped at the first iteration, one positive and one negative phase, as soon as the gradient computation becomes possible, as suggested by the Contrastive Divergence technique (CD-1)[2]. We use the best CD-1 results to train the model again with $n > 1$ CD steps to check if the learning capabilities of the model improve.

Optimisation algorithm – The optimisation algorithm used is *RMSprop*. It enables automatic adjustment of the learning rate for each model parameter, namely a_i, b_μ and $W_{i\mu}$. As suggested by [2], we set the algorithm parameters to $\beta = 0.9$ and $\epsilon = 1 \times 10^{-8}$. For further reference on our implementation, see [4].

Training – Model training is achieved in $N_{\text{epochs}} = 100$ epochs and uses minibatches of size $N_{\text{mini}} = 500$ samples over a dataset of $N_{\text{samples}} = 10000$ entries.

Algorithm 2 RBM Training

```
N_mini ← samples in minibatch
cdn ← number of contrastive divergence steps
l_rate, ε ← 0.001, 10^−8
for epoch = 1, ..., 100 do
    for v in minitbatch do
        for cd = 1, ..., cdn do
            h ← Positive phase(v)
            v_f ← Negative phase(h)
        end for
        h_f ← Final positive phase(v_f)
        Compute Eq 4 averages
        W, a, b ← update by RMSprop
    end for
end for
```

Generation – To better understand the capabilities of the model, we implement an additional Generation procedure. We force the RBM to produce a new set of Fantasy Particles, starting from the trained weights, the original $\boldsymbol{v}$, and independently chosen amplitudes AMP.

Quality metrics – To assess the quality of the model, we calculate the Jensen-Shannon divergence (JSD) [5] between the original and generated data distribution. JSD is a bounded measurement of the divergence of probabilities distribution [3]. It is symmetric, defined everywhere, and is computed from the Kullback–Leibler (KL) divergence as follows.

$$JSD(P||Q) = \frac{1}{2} D(P||M) + \frac{1}{2} D(Q||M) \tag{7}$$

$$\text{with } M := \frac{1}{2}(P + Q) \tag{8}$$

To bound the JSD between 0 and 1, we choose 2 as the logarithm base to compute the KL. Such probabilities distributions are computed by normalising the countings of distinct occurrences of possible protein sequences in the original or generated datasets. These assessment criteria are discussed in the following section. Furthermore, we also check that $\mathcal{L}$ keeps increasing during training, as expected.

RESULTS

We perform training and generating procedures at different amplitudes ranging from 0.05 to 0.5, with increments of 0.05, both for the $\{-1, 1\}$ and $\{0, 1\}$ encodings.

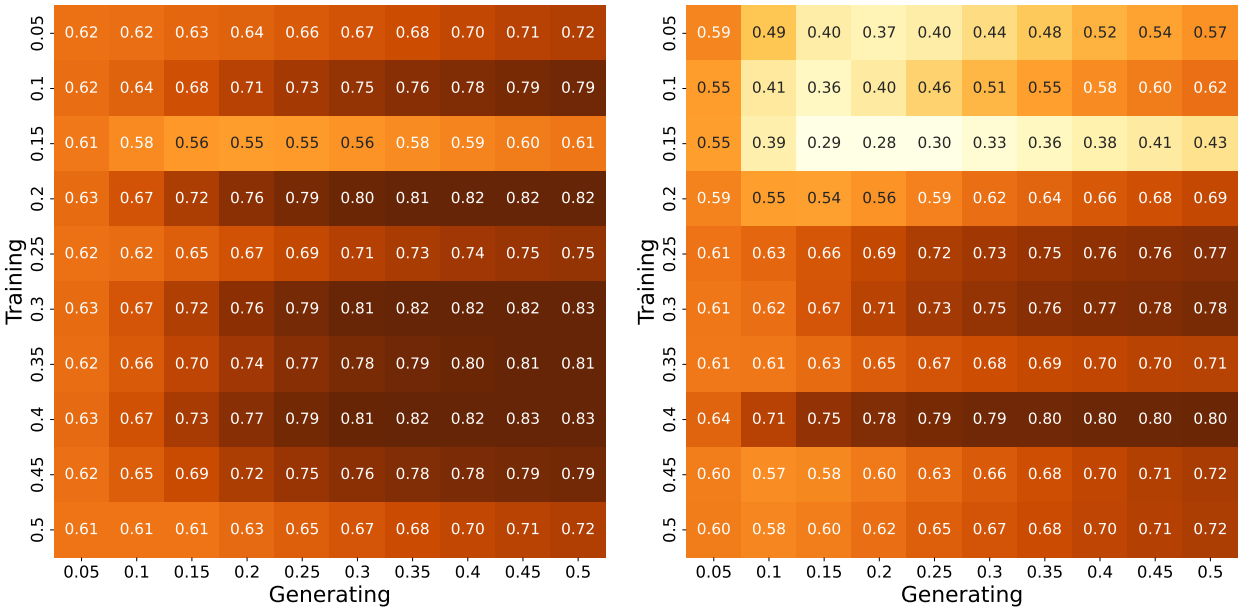

FIG. 2. On the left: heatmap showing the JSD as a function of generating and training amplitude for $\{0, 1\}$ representation. On the right: heatmap showing the JSD as a function of generating and training amplitude for $\{-1, 1\}$ representation.

For each pair of chosen amplitudes, we calculate the JS divergence between the original and the generated distribution, as shown in Figure 2.

Comparing the two encodings, we see that $\{-1, 1\}$ produces the lowest overall JS divergence with training and generating amplitudes of 0.15 for both. We speculate that changing amplitudes in the training corresponds to varying the σ of Glorot weights initialisation. However, changing the amplitude in the generating procedure alters the expressive and generalisation power of the model, as can be seen from the JSD values plotted horizontally in the heatmaps.

Limiting our analyses to only the best pairs of amplitudes yielding lower JS divergences, we show the weight matrices. For each hidden unit μ, we highlight in matrix form the weights $W_{i\mu}$, as shown in Figure 3 and 4.

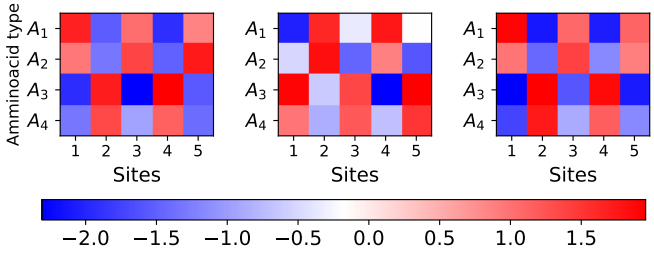

FIG. 3. Weight matrices grouped by hidden units for $\{-1, 1\}$ representation and $H = 3$ and training amplitude of 0.15. The columns represent the five k sites, while the rows represent the four i digits encoding for each amino acid. Higher values in matrix entry i, k imply higher probabilities for the A_i amino acid type to occupy site k.

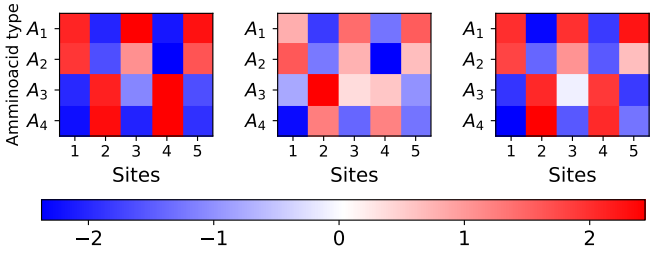

FIG. 4. Weight matrices grouped by hidden units for $\{0, 1\}$ representation and $H = 3$ and training amplitude of 0.15. The columns represent the five k sites, while the rows represent the four i digits encoding for each aminoacid. Higher values in matrix entry i, k imply higher probabilities for the A_i aminoacid type to occupy site k.

It is possible to verify the emergence of alternating patterns in the weights, where correlations are speculated to be stored. More specifically, higher positive weights alternate site-wise between positions referring to A_0, A_1 and A_2, A_3. No significant differences are found between the patterns of the two encodings. However, taking into account the JSD measurement, we hypothesise that the $\{-1, 1\}$ encoding performs better as the patterns are visible and the JS divergence is simultaneously minimised.

Increasing the number of hidden units to $H = 6$ and using the same amplitudes found above, we plot the same weight matrices, Figure 5. Patterns are still recognisable in three out of the six hidden units. We also notice that the absolute values of the weights are smaller with respect to the $H = 3$ cases. These observations suggest that using more than $H = 3$ hidden units is redundant to encode correlations in the data.

To check the convergence of the training procedure, we show in Figure 6 $\mathcal{L}$ calculated at the end of each epoch. As expected, there is a general monotonic increase over the epochs. The initial plateau is due to the initial lack of momentum of the maximisation algorithm.

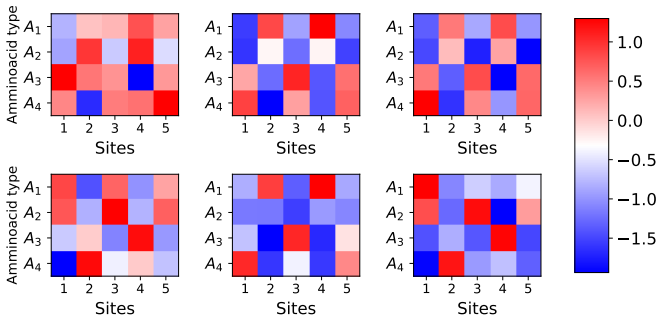

FIG. 5. Weight matrices grouped by hidden units for $\{-1, 1\}$ representation and $H = 6$ and training amplitude of 0.15. The columns represent the five k sites, while the rows represent the four i digits encoding for each aminoacid. Higher values in matrix entry i, k imply higher probabilities for the A_i aminoacid type to occupy site k.

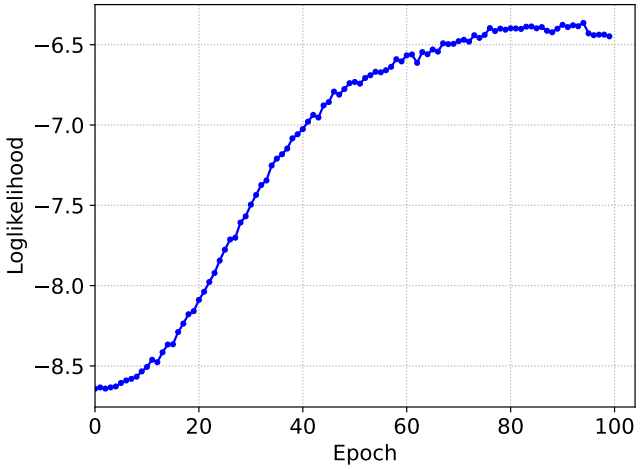

FIG. 6. Loglikelihood computation over the epochs during the model's training procedure with amplitude 0.15

We try to improve the training results by increasing the number of contrastive divergence steps (CD-2 and CD-3), but the resulting values of the JS divergence get worse, proving its ineffectiveness, as shown in Table 1.

	CD-1	CD-2	CD-3
JSD	0.28	0.77	0.80

TABLE I. JS divergence computation for different CD-n

CONCLUSIONS

The models yield different JSD values depending on the representation choice suggesting its importance, as shown in Figure 2, although patterns are still recognisable in both representations.

Increasing the number of hidden units suggests the existence of an upper limit for the useful value of H: any additional hidden unit does not seem to learn valuable information about the data structure, as we infer from Figure 5. Future studies could focus on a lower limit of the number of hidden units for which the model is unable to learn the data structure, depending on the complexity of the distribution.

Increasing the number of contrastive divergence steps, in order to enhance the performance of the algorithm, seems ineffective according to our metric, as reported in Table I.

RBMs are powerful tools that allow us to infer correlations in data through the model weights. The architecture is fairly susceptible to the initialisation of its parameters, but can be optimally tuned to learn the data distribution and to produce synthetic samples.

[1] E.T. Jaynes, Phyical Review **106**, 620–630 (1957).
[2] Mehta, Pankaj, et al. "A high-bias, low-variance introduction to machine learning for physicists." Physics reports 810 (2019): 1-124.
[3] Briët, Jop, and Peter Harremoës. "Properties of classical and quantum Jensen-Shannon divergence." Physical review A 79.5 (2009): 052311.
[4] GitHub Repository - threeblueonebrowneyes.
[5] JS Scipy documentation