Politecnico di Torino

Manipulating specificity in biological sequences with representation learning

Author: Jacopo Boccato

Supervisors: Jorge Fernandez de Cossio Diaz
Andrea Pagnani

Academic Year: 2024–2025

Contents

1 Introduction **3**
1.1 Unsupervised learning and generative models 3
1.2 Biological sequence design 3
1.3 Inverse Statistical Physics of Biological Sequences 4
1.4 Disentangled representations 4
1.5 Our goal 5

2 Methods and datasets **6**
2.1 Restricted Boltzmann Machines 6
2.2 Non-adversarial constrained training 6
2.3 Generalization to Multi-Class Labels 7
2.4 Lattice Proteins 8
2.5 WW domain 9

3 Results **10**
3.1 Electrostatic modes in Lattice Proteins 10
3.1.1 Dataset 11
3.1.2 RBM setup 11
3.1.3 Results 11
3.1.4 Characterizing the transition 12
3.1.5 Second order RBM 14
3.1.6 Classifiers on hidden units activations 14
3.2 WW domain 15
3.2.1 The dataset 15
3.2.2 RBM setup 16
3.2.3 WW domain partial results 16
3.2.4 Energy landscape 17
3.2.5 Trajectories in classifier RBM 17
3.2.6 Final results 18

4 Conclusions **21**
4.1 Results 21
4.2 Future Perspectives 21

1 Introduction

1.1 Unsupervised learning and generative models

A foundational problem in machine learning is inferring the statistical properties of a dataset $\mathcal{D} = \{\mathbf{x}\}_{i=1}^{\mathcal{N}}$. This task, known as unsupervised learning, is the precursor to a variety of objectives, including clustering, dimensionality reduction and data generation. In this thesis, we will focus on the latter. Our aim is to learn a probability distribution that approximates the true data distribution $p(\mathbf{x})$ so that we generate new samples $\widetilde{\mathbf{x}} \sim \hat{p}(\mathbf{x};\theta)$ whose statistics match those of $\mathcal{D}$. We will start with an unlabeled dataset, which we will assume to be

$$\mathbf{x}_i \overset{\text{i.i.d.}}{\sim} p(\mathbf{x}), \quad i = 1, \ldots, \mathcal{N}$$

Since in modern machine learning, as well as in biological application, the dimension of the dataset D is large, making nonparametric estimation mathematically and computationally intractable, we have to assume a parametric structure for the probability distribution, which we will indicate as:

$$p(\mathbf{x};\boldsymbol{\theta})$$

With these assumptions learning the distribution amounts to choosing parameters $\boldsymbol{\theta}$ that maximize the data log-likelihood or minimize the Kullback-Leibler divergence from the true distribution. The problem of choosing a fitting form for the probability distribution is extremely complicated and far from being solved, leading to the proposal of many approaches such as variational autoencoders, energy-based models and generative adversarial networks (GANs).

1.2 Biological sequence design

Proteins and RNA play essential roles in all living systems. Proteins are responsible for most biological functions, including catalyzing chemical reactions, transporting molecules, and maintaining cellular structure. RNA, beyond serving as a messenger of genetic information, is also involved in crucial regulatory processes. A central paradigm in molecular biology is that sequence determines function, this mapping however is highly complex and context-dependent and very poorly understood.

Designing novel proteins and RNA molecules—i.e., solving the inverse problem of identifying sequences that perform a given phenotypic task—is an open challenge. Solving this problem, known as sequence design, has significant applications in medicine, biotechnology, and fundamental research. The difficulty lies in the intricate and nonlinear relationship between sequence and biological activity. A common source of data in this field is the multi-sequence alignment (MSA) of protein domains, linear strings of amino acids that perform similar functions across different species. Evolution provides a natural source of functional sequence diversity: over time, nature samples a wide range of sequences that carry out similar tasks, embedding signatures of evolutionary constraints into sequence variability. Modern sequencing technologies allow us to access large datasets of homologous sequences, from which statistical patterns can be extracted to infer properties of the sequence-function relationship.

Due to the cost and difficulty of experimentally characterizing individual biological functions, machine learning, particularly unsupervised methods, are extensively used to analyze MSAs. These models aim to uncover underlying biological information without relying on labeled data, making them especially valuable for interpreting sequence variability.

1.3 Inverse Statistical Physics of Biological Sequences

One of the approaches that has led to notable advancements in the understanding and in the generation of biological sequences is the use of MSAs of protein families as the basis for the inference of Boltzmann-like distributions. Approximating an unknown distribution with a Boltzmann distribution is an assumption which strongly simplifies the biological reality but allows us to work within a well-defined mathematical framework and has shown very good results [1]. This approach, however, while providing good results, has some shortcomings, in particular a Boltzmann Machine (BM) learns a distribution on the visible units (the dataset) and thus cannot be used to generate sequences with a given function.

Restricted Boltzmann Machines (RBMs) are another generative model rooted in statistical physics that learn a Boltzmann-like distribution through representational learning, introducing a latent layer to capture complex dependencies among visible units. This approach enables the model to retain higher-order statistics and has been successfully applied to RNA sequence design [2].

1.4 Disentangled representations

Many generative model architectures use hidden or latent variables to encode the statistical relationship between data points. These latent variables learn during training a way to represent data points and their relationships in the latent space. This representation is a complex, non-linear function which translates the data points used for training in activation patterns of these latent variables. The representation encoded in the latent variables is often complex and distributed, meaning that each attribute of the dataset is represented among many hidden units, making it impossible to manipulate the generation of new attribute-specific data points. This property of generative models is very undesirable, as in principle we would like to control the generation of our sample. In order to have generative models where a latent variable encodes information about a specific attribute, one has to "force" the model to learn a disentangled representation of the features. In a disentangled representation, information about a given attribute of the data is "concentrated" on a few latent variables, while the remaining latent variables are uncorrelated with the attribute of interest, as can be seen in Figure 1.1. To pursue this objective, many approaches have been proposed in literature; these algorithms rely on an adversarial framework, which is very difficult to tune correctly as it involves a double maximization-minimization constraint and it's not easily interpretable[3].

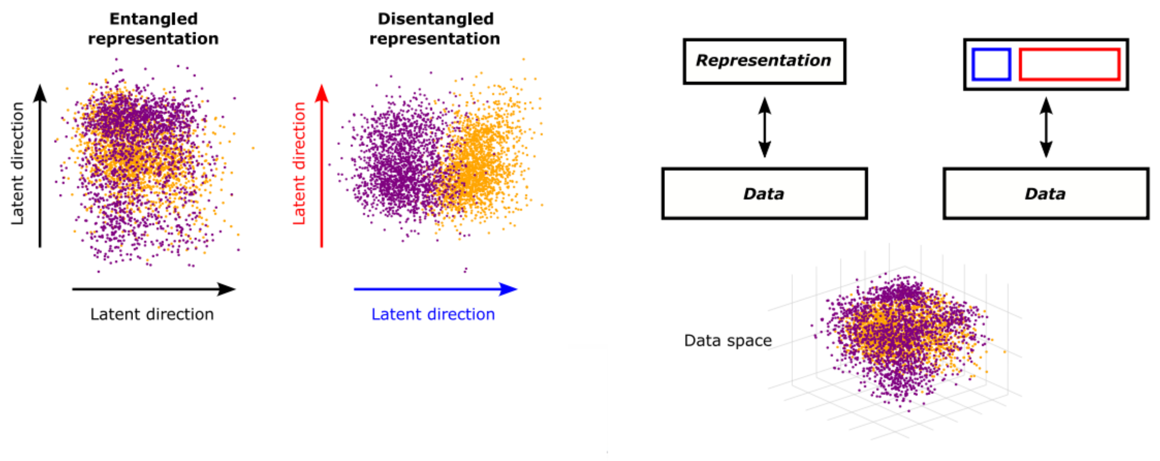

Figure 1.1: Entangled vs disentangled representations. **(Bottom right)** A set of high-dimensional data points is mapped through unsupervised learning onto a latent representation **(left)**. Data are colored in purple and orange according to a binary-valued attribute, e.g., being an odd or even number for MNIST images of handwritten digits. **(Top right)**: When representations are entangled, the separation of data classes is not aligned with a single latent direction, when representations are disentangled, one or few directions in latent space (blue) separate the labeled classes, while other directions are not correlated with the label (red).

1.5 Our goal

In this report, we will utilize a novel approach proposed in [4], which will be discussed thoroughly in Section 2. This algorithm, which is a semi-unsupervised method, uses classified data points to compute constraints that are applied to some of the hidden units of a Restricted Boltzmann Machine (RBM) in order to obtain a disentangled representation. These constraints, which can be of first or second order, effectively render the latent units on which they are applied uninformative about the attribute of interest. These constraints, during the usual max-likelihood training, also have the indirect effect of concentrating information about the given attribute on the few hidden units that are released. This algorithm, after training, allows easily to sample the RBM to obtain data points that live in a desired sub-region of data space by tuning the values of the released units.

In particular in this report our goal is to generalize this algorithm to multi-class attributes, as well as studying the effect of applying a low rank approximation of the quadratic constraint, effectively reducing the number of total constraints to which the RBM is subject to. We will benchmark this approach with protein data, in particular focusing on the Lattice protein model-a simple solvable model that models protein folding-and the WW domain, a real world domain involved in protein protein interactions that is known to form clusters.

2 Methods and datasets

2.1 Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMs) are an energy-based generative model. They are bipartite undirected graphical models defined over N visible units $\mathbf{v} = \{v_1, \ldots, v_N\}$ and M hidden units $\mathbf{h} = \{h_1, \ldots, h_M\}$, each of which may assume values in a finite discrete alphabet or over a continuous interval, with interactions parameterized by weights $w_{i\mu}$. In the context of our work, we will always assume that each hidden unit takes values $h = \{0, 1\}$, whereas each visible unit can take on one of the $L = 20$ amino acids plus the gap symbol, that we represent as a one-hot encoded vector. An RBM defines a joint Boltzmann-like probability over the visible and hidden units as:

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

Where Z is a normalization constant and $E(\mathbf{v}, \mathbf{h})$ can be seen as an energy function defined as:

$$E(\mathbf{v}, \mathbf{h}) = -\Sigma_{i=1}^{N} g_i(v_i) - \Sigma_{\mu=1}^{M} \theta_\mu h_\mu - \Sigma_{i=1}^{N} \Sigma_{\mu=1}^{M} w_{i\mu}(v_i) h_\mu$$

The goal of an RBM is to reproduce the statistical properties of the dataset on which it's trained. To do so it must find the set of parameters $\{g_i\}_{i=1}^{N}, \{\theta_\mu\}_{\mu=1}^{M}, \{w_{i\mu}\}_{i,\mu=1}^{N,M}$ for which the Boltzmann measure defined above is the closest to the empirical distribution. This occurs during training, where one maximizes the likelihood over the dataset. Maximizing the likelihood involves computing the partition function, which requires summing over an exponentially large configuration space and is thus computationally hard [5]. To overcome this, approximate methods based on Gibbs sampling are employed.

In this report, the training is performed using Persistent Contrastive Divergence (PCD) [6], an algorithm that estimates the model distribution by maintaining a set of persistent Markov chains across updates rather than initializing from the data at each step, thereby improving sampling efficiency and convergence stability.

2.2 Non-adversarial constrained training

The central idea of the algorithm proposed in [4] is to concentrate information about a specific attribute into a small subset of hidden units of the RBM, by constraining the remaining units to be uninformative about that attribute. The original procedure requires the attribute to be binary. In this case, one would ideally impose that the mutual information (MI) between the attribute and the hidden unit activations is zero. However, computing MI is difficult in practice, so one instead enforces the vanishing of correlations up to various orders.
By defining $u(\mathbf{v})$ as the label associated to a visible configuration $\mathbf{v}$, the vanishing of first-order correlations leads to:

$$\sum_{i=1}^{N} q_i^{(1)} w_{i\mu} = 0, \quad \mu \in \text{constrained units} \quad q_i^{(1)} = \langle u(\mathbf{v})\, v_i \rangle_D - \langle u(\mathbf{v}) \rangle_D \langle v_i \rangle_D \tag{2.1}$$

Similarly, the second-order constraint reads:

$$\sum_{i,j=1}^{N} q_{ij}^{(2)}\, w_{i\mu} w_{j\nu} = 0, \quad \mu, \nu \in \text{constrained units} \quad q_{ij}^{(2)} = \langle u(\mathbf{v})\, v_i v_j \rangle_D - \langle u(\mathbf{v}) \rangle_D \langle v_i v_j \rangle_D \tag{2.2}$$

This constraint is computationally heavy, as it introduces $N^2 \times L^2$ equations for each weight. To mitigate this, we adopt a low-rank approximation of the second-order matrix $\mathbf{q}^{(2)}$, justified by the fact that only a few directions in the covariance space carry meaningful class-related information. Empirically, most directions are compatible with noise and have eigenvalues comparable to a null model based on random amino acid arrangements.
In practice in this report, we will approximate $\mathbf{q}^{(2)}$ using its biggest and smallest eigenvectors:

$$\mathbf{q}^{(2)} \approx \lambda_+ q_+ q_+^\top - \lambda_- q_- q_-^\top,$$

with $\lambda_\pm > 0$ and $q_\pm \in \mathbb{R}^N$. The constraint $W^\top \mathbf{q}^{(2)} W = 0$ then implies that the weights must lie in one of two orthogonal subspaces:

$$w_\mu^\top \left(\sqrt{\lambda_+} q_+ \pm \sqrt{\lambda_-} q_- \right) = 0, \quad \forall \mu \in \text{constrained units}$$

The RBM makes the choice of which subspace to inhabit during training by maximizing the likelihood.

During training, the first-order constraint is enforced by projecting the weights $\{w_{i\mu}\}_{i,\mu=1}^{N,M}$ after each step of likelihood maximization, while the second-order constraint is implemented as a strong penalty in the loss function. The first-order constraint implies that a linear classifier trained on the constrained hidden unit activations cannot recover the labels; under the second-order constraint, a quadratic classifier fails as well. Only more complex models, such as deep neural networks (DNNs), can partially recover label information, confirming that the constrained units carry negligible class-relevant information.

This algorithm also enables controlled manipulation of samples. One can sample a hidden configuration from a visible input, flip the value of the released hidden unit, and resample to obtain a configuration associated with the opposite class. To ensure the output reflects the equilibrium distribution, additional equilibration steps can be performed while keeping the flipped value fixed. Another strength of this method lies in the analytical tractability of RBMs, which allows the utilization of existing theoretical tools to study training dynamics and interpret model behavior, allowing for a better understanding of the theoretical limitations of this approach in the case of multiple and closely related attributes.

2.3 Generalization to Multi-Class Labels

We are interested in extending the algorithm proposed in Equation 2.1 to a dataset composed of an arbitrary number of clusters K. To this end, we have developed a new Julia package that trains a restricted Boltzmann machine (RBM) with $M-1$ binary hidden units and one Potts unit whose alphabet has K symbols. Our key idea—naturally

generalizing the two-cluster procedure—is to enforce that the weight vectors of all binary hidden units remain perpendicular to the hyperplane spanned by the K cluster centers. In practice, this orthogonality constraint causes the Potts unit to learn the data distribution projected onto that hyperplane, and, provided the clusters are sufficiently well separated and represented, each symbol of the Potts unit becomes unambiguously associated with one of the K classes.
To formalize this disentanglement in a multi-class setting, we introduce for each data point a one-hot label vector

$$\mathbf{u} \in \{0,1\}^K,$$

where $u_d = 1$ if and only if the example belongs to class d. We then impose the first-order vanishing correlation constraint

$$\langle u_d\, \phi(\mathbf{I})\rangle_D - \langle u_d\rangle_D \langle \phi(\mathbf{I})\rangle_D = 0, \qquad d = 1, \dots, K, \tag{2.3}$$

where $\mathbf{I} = W^\top \mathbf{v}$ denotes the vector of inputs to the hidden layer and $\phi(\cdot)$ is a (possibly nonlinear) feature map. Equation 2.3 ensures that, in expectation over the empirical data distribution D, no component of the hidden representation in the feature space $\phi(\mathbf{I})$ linearly correlates with any class label.
In the special case of a linear feature map, $\phi(\mathbf{I}) = \mathbf{I}$, the constraint simplifies to the orthogonality condition

$$\langle u_d\, \mathbf{I}\rangle_D - \langle u_d\rangle_D \langle \mathbf{I}\rangle_D = 0, \qquad d = 1, \dots, K. \tag{2.4}$$

Together, these constraints guarantee that no linear classifier—or, more generally, no classifier acting on the feature space $\phi(\mathbf{I})$—can recover the class label from the constrained hidden activations, thus achieving the desired disentanglement across K classes.

2.4 Lattice Proteins

In this report to benchmark the performance of the algorithm in Section 2.2 in application to protein sequences we will use the Lattice Proteins (LP) model. In this model $L = 27$ amino acids are arranged on a 1D backbone structure, occupying the sites of a $3 \times 3 \times 3$ cubic lattice in 1 out of the $\mathcal{N} = 103,406$ possible structures, characterized by a contact map **c**. These amino acids interact with their nearest neighbors on the structure (but not on the backbone) through the Miyazawa-Jernigan (MJ) energy matrix. This model allows us to define the energy of a sequence **A** of amino acids folded in a structure S as:

$$\mathcal{E}(\mathbf{A}|S) = \Sigma_{i<j} c_{ij}^{(S)} E(a_i, a_j)$$

From this definition, we can easily define the Boltzmann probability that a sequence **A** folds into a given structure S as:

$$P_{nat}(S|\mathbf{A}) = \frac{e^{-\mathcal{E}(\mathbf{A},S)}}{\Sigma_{s\in\mathcal{S}} e^{-\mathcal{E}(\mathbf{A},s)}}$$

This model allows us to test the quality of our generated data points by computing the $P_{nat}(S|\mathbf{A})$ of each sequence. These analytical possibilities, together with a non-trivial data structure, make the LP model a perfect benchmarking dataset [7] [8]. To generate our dataset we use a Metropolis algorithm, in particular to obtain sequences with an average $P_{nat}(S|\mathbf{A}) = 0.9994$ we set the inverse temperature of our Metropolis algorithm $\beta = 10^4$.

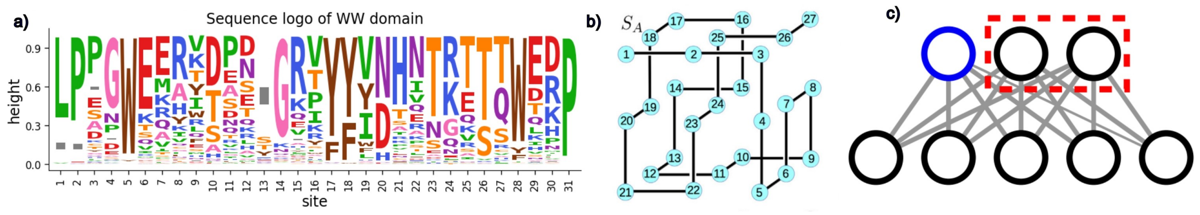

Figure 2.1: **a)**: Sequence Logo of data of the WW domain used for training the RBM. We can observe the two highly conserved tryptophans in positions 5 and 28. **b)** Structure of the Lattice Protein used in the report. **c)** Structure of an RBM, highlighted in red the hidden units that will be subject to the constraint. This structure will learn a disentangled representation like the one in Figure 1.1.

2.5 WW domain

The WW domain is a small protein module of 30–40 amino acids, named for its two conserved tryptophan (W) residues. It functions as a versatile interaction platform by recognizing short proline-rich peptide motifs [9]. The WW domain interacts with a vast array of peptides, but the binding affinity with these partners is highly variable inside the domain. To account for this diversity, the WW domain has been classified into four groups to account for the different binding preferences of each protein [9] to the various classes of peptides. This cluster structure is the perfect testing ground for our extension in Equation 2.4. In this report in particular, we will consider a three-cluster adaptation, uniting classes II and III as presented in [10]. This choice, which is often done in literature [11], is due to the many similarities between classes II and III, which have always been known [10], as many sequences in literature are classified as class II/III. To avoid confusion in this report, we will use Roman numbers to describe the 4 clusters classification as used in literature and Arab numbers to talk about our 3 clusters division.

3 Results

3.1 Electrostatic modes in Lattice Proteins

It has been reported in literature [8] that a specific structure in the LP model can correspond to two markedly different amino acid configurations. These configurations form two distinct clusters, which we will refer to as electrostatic modes, as their main difference lies in the distribution of charged residues: one mode is characterized by positively charged amino acids (H, K, R) at positions $[1, 25]$ and negatively charged amino acids (E, D) at positions $[2, 16, 18, 20]$, whereas the other mode exhibits the opposite charge distribution. This physical situation is particularly suited to benchmark our algorithm as it allows to create a dataset with a clear cluster structure, as reported in Figure 3.1, and the distinctive characteristics of the classes are concentrated on a few amino acids.

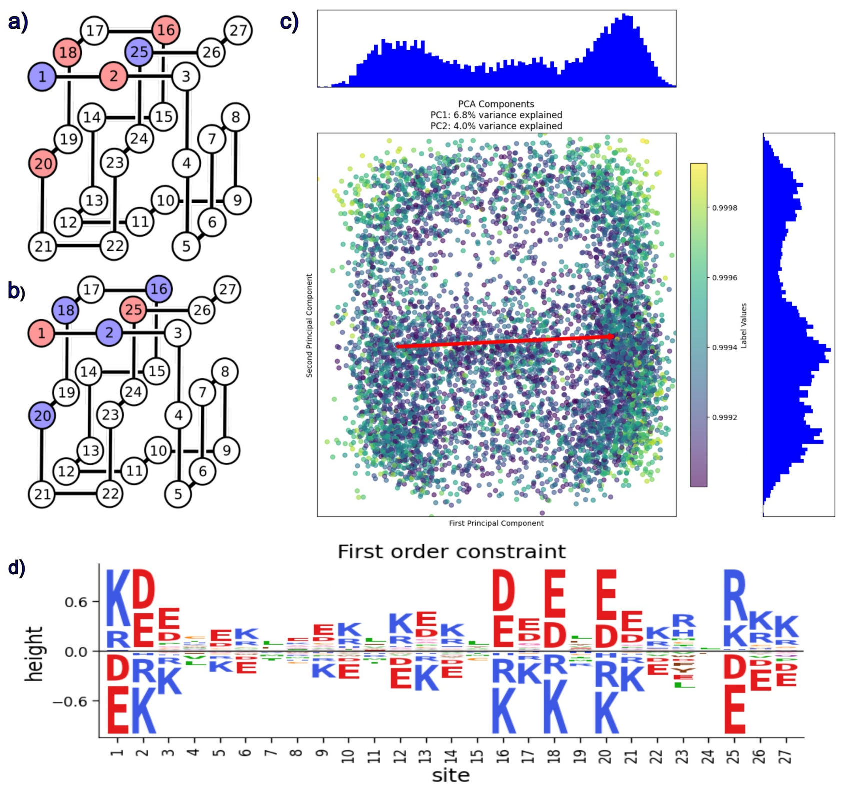

Figure 3.1: **a) b)**: Lattice protein structure highlighting the two electrostatic modes of structure A (red: negatively charged amino acids; blue: positively charged). **c)**: PCA projection of training data with overlaid density histograms along PC1 and PC2, revealing cluster structure, in red the projection of the vector used in training reported in **d)**. **d)**: Sequence logo of the first order constraint used in the training of the RBMs. The size of the letters is proportional to the times that a given amino acid appears in that position in the vector/sequence.

3.1.1 Dataset

To train our RBM, we generate a dataset of sequences characterized by an average $P_{nat}(S|\mathbf{A}) = 0.994$ as explained in Section 2.4, In this dataset the clusterization is evident Figure 3.1. To obtain the labels needed to compute the first-order constraint for the training of the RBMs we select out the sequences that belong to one of the two electrostatic modes and compute the firs- order constraint, which in this case amounts to the difference of the centroid of the two clusters, as detailed in Equation 2.1. For a complete explanation, see Appendix 4.2. The first order constraint, as defined in Section 2.2, is reported in Figure 3.1, showing clearly how the main difference in the clusters is concentrated in the electrostatic position. For the training, we use the entirety of the generated dataset.

3.1.2 RBM setup

To benchmark the performance of the algorithm proposed in Section 2.2 we train three RBMs using the same dataset and architecture, composed of $M = 150$ binary hidden units. One RBM was trained with the classic PCD algorithm, another one was trained with the PCD algorithm and the first-order constraint applied to all but one hidden unit and the last one was trained with the same algorithm but a first and second order constraint to all but one released unit, in particular as reported in Chapter 2 we choose a rank 2 approximation for $\mathbf{q}^{(2)}$.

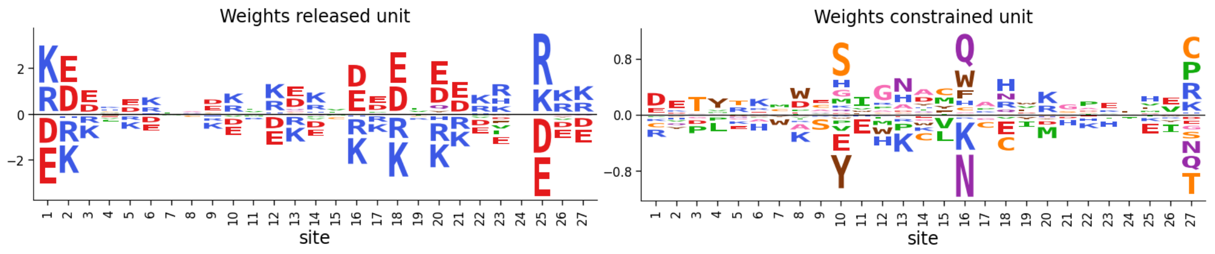

Figure 3.2: We report the Sequence Logo of the weights of the RBM constrained at the first order after completing training. **Left**: sequence logo of the released unit. **Right**: sequence logo of a random constrained unit. We can clearly see the resemblance between the released weight learned by the RBM and Figure 3.1. The constrained weight, on the other hand, carries no information about the separation between the two clusters and little information about the variability of each cluster (Note position [1,25] where positive and negative amino acids vary in the opposite way with respect to the constraint).

3.1.3 Results

After an RBM has converged one can look at the weights $\mathbf{w}_\mu$, to understand their role in generating new sequences. In particular we can look at the sequence logo of the weights of the unconstrained unit to verify if its activation can discriminate the two clusters. We can see how Figure 3.2 clearly resembles Figure 3.1, this is a first indication that our model has indeed achieved label concentration, as switching this unit changes the generated sequence along the electrostatic amino acids. One can also observe how the other weights-here we report only one on the right of Figure 3.2-is completely uninformative about the two structures that characterize the dataset. After having observed

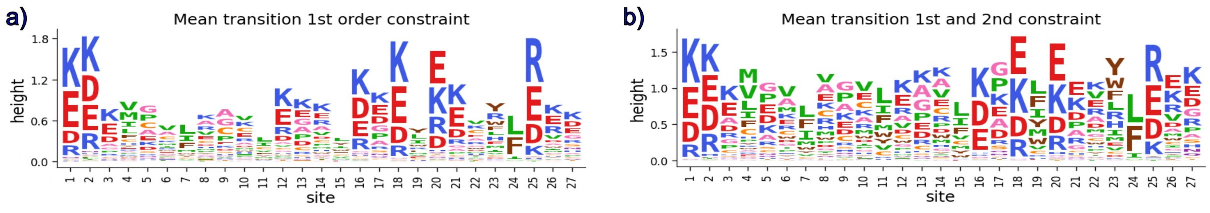

Figure 3.3: Average sequence-space transition after released hidden unit flip: **a)**: Mean transition after forceful flip of released unit in RBM constrained only at the first order **b)**: Mean transition after forceful flip of released unit in RBM constrained at first and second-order. We can observe how the transition on the Left includes almost only transitions in the electrostatic amino acids. The right image appears to be more "noisy".

the sequence logo of the unconstrained weight as a check, we want now to understand how the released hidden unit influences the regions sampled by the model and whether, as it seems from the sequence logo of the weight, the released unit encodes the electrostatic structure's position, indicating information concentration. To do so we study the average behavior of the transition induced by flipping the released hidden unit. In Figure 3.3, we report for each constrained RBM the sequence logo of the vector:

$$< \vec{v} >= \frac{1}{|\mathcal{D}|}\Sigma_{i\in\mathcal{D}}|\vec{v}_{bt}-\vec{v}_{at}| \tag{3.1}$$

where $\vec{v}_{bt}$, $\vec{v}_{at}$ are the one-hot encoded vectors describing the sequence before and after the forced transition. The choice of the absolute value in this average vector is to avoid that "opposite" transitions cancels out. By observing Figure 3.3 we can conclude that the transition of the first order constrained model is cleaner, as the electrostatic amino acids are almost always changed, while the remaining positions remain unchanged. The RBM constrained at both orders instead shows a more "noisy" transition. This is due to the many constraints imposed on the model that impede learning, the only released unit is not able to fully encode all the information of the dataset. This reduced expressivity of the model can also be observed by looking at the average $P_{nat}(S|\mathbf{A})$ of the generated sequences, which drops from 0.995 of the first order constraint RBM to 0.91. In Appendix 4.2 we report the equivalent transition obtained by flipping a random weight of the unconstrained RBM as a comparison. The question of how to constrain the RBM to obtain an expressive yet controllable generative model is a completely open and non-trivial problem, as it is very dataset and architecture-dependent. We will try to address the question on this particular example in the following subsections.

3.1.4 Characterizing the transition

An important question is whether changing the released unit leads to completely different sequences or causes only minimal modifications necessary to switch between clusters, while keeping other features of the sequence unchanged. Having a model that is able to perform a parsimonious transition would allow us to manipulate single sequences and modify one of their function without impeding the other ones. Looking at the left plot in Figure 3.4, we see that the conserved structure pattern reveals how the transition mainly affects the electrostatic positions, leaving the rest of the sequence mostly unchanged.

The right plot in Figure 3.4 provides further insight into the physical behavior of the model. It shows that the conditional frequency of change (used as a proxy for conditional probability) reflects information about the contact map: amino acids that are in physical contact tend to show higher frequency of co-variation. For example, there is a notable co-variation between positions 3 and 26. The constant yellow columns in correspondence with the six electrostatic positions indicate that these amino acids tend to change with high probability, regardless of changes in the other amino acids. This is another proof of how the released unit's flip acts mainly on the electrostatic amino acids, keeping the others relatively unchanged.

To further characterize the transition, we also analyze the Hamming distance, which counts how many positions differ between two sequences. Specifically, we calculate the Hamming distance between each sequence and its corresponding version after the transition. The average distance between sequences from different clusters in the training dataset is 24.6, while the average distance between a sequence and its post-transition version is 12.8, in Appendix 4.2 the complete histogram. These results confirm that the constrained RBM behaves as expected, producing minimal and localized changes during the transition (while maintaining information about the physical model behind, as shown by the information about the contact map present in the pattern of co-conservation).

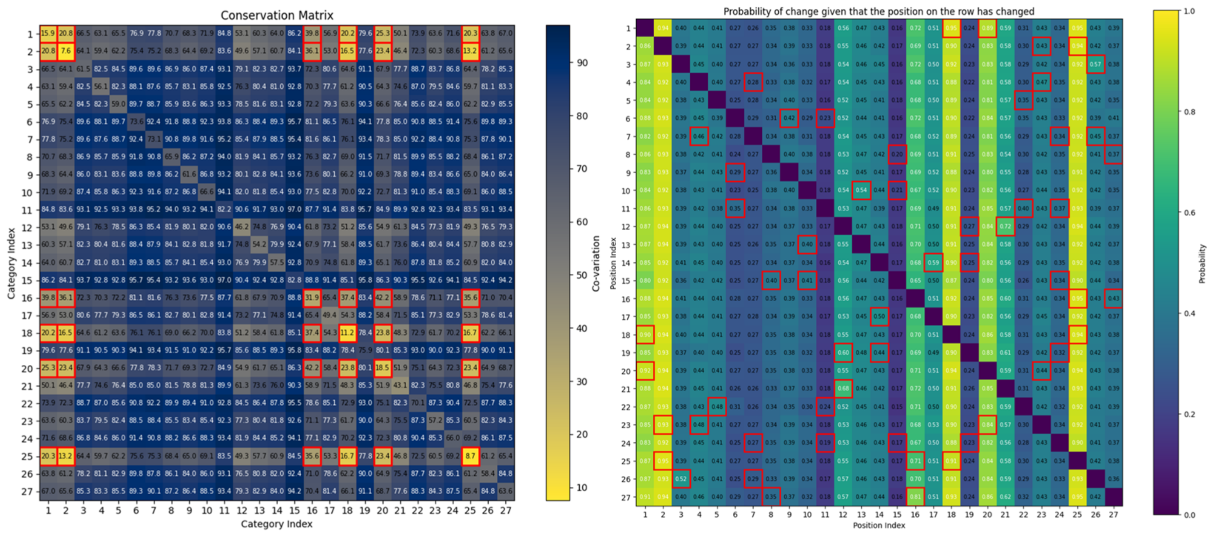

Figure 3.4: **Left**: We take a large sample of sequences with an average $P_{nat}(S|\mathbf{A})$ of 0.996 as initial condition and we forcefully flip the released unit. Above the matrix showing the percentage of transitions in which a couple of positions was not modified by this transition. In red, we see the pairs corresponding to electrostatic amino acids. **Right**: Matrix that shows the probability that a position underwent a change given that the position on the row has undergone a transition. In red, we highlight the contact between positions.

3.1.5 Second order RBM

We have discussed above the difficulty of balancing constraints to train a model which is controllable yet is able to reproduce the complexity of the dataset. To partially answer this question, we train a series of RBMs, with the constraint at the first order applied to all but one hidden unit and the low-rank approximation applied to different sets of hidden units, starting from all but one released unit. Assessing if a given generative model is expressive enough is a hard problem, in particular for RBM. On e could compute exactly the likelihood, but this is complex, as it involves computing the partition function. With Lattice Proteins, we can use the average $P_{nat}(S|\mathbf{A})$ to assess the quality of our RBM. To do so, we generate 5000 sequences for each of these RBMs and compare their statistics:

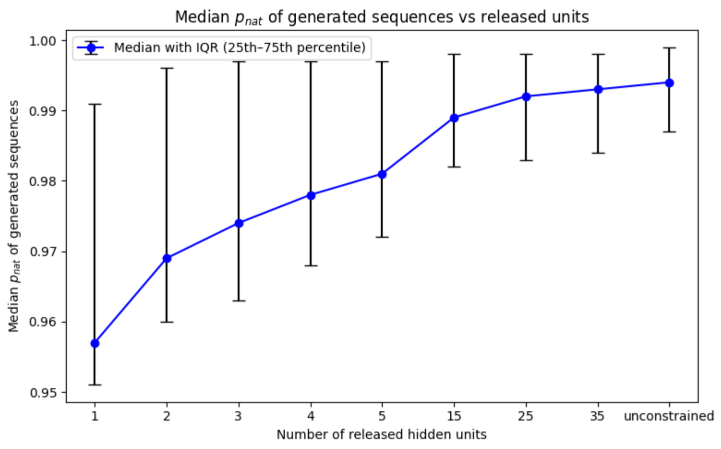

Figure 3.5: Value of the median $P_{nat}(S|\mathbf{A})$ of various RBM trained with different numbers of released units.The architecture of the RBM is the same, with $M = 150$ hidden binary units.

We can see how releasing more hidden units causes an increase in the median $P_{nat}(S|\mathbf{A})$.

3.1.6 Classifiers on hidden units activations

To conclude our analysis of the RBMs trained on the electrostatic modes dataset, we assess whether information about the class can be extracted from the hidden unit activations of the RBM. The activation of a hidden unit is defined as $I_\mu(\mathbf{v}) = \sum_{i=1}^{N} w_{i\mu} v_i$. The choice to use activations I_μ instead of the values of the latent variables h_μ is motivated by the fact that given a visible configuration $\mathbf{v}$, the hidden variables h_μ are stochastic and conditioned on I_μ. This implied that by the data processing inequality, the mutual information between the class and I_μ provides an upper bound to that with h_μ, justifying the use of activations for analysis. Specifically, we train both a linear classifier and a neural network on the activations of the constrained hidden units of the RBM subject to the first-order constraint. Given the nature of the constraint, we expect the linear classifier to perform poorly, as it cannot capture higher-order statistical dependencies. In contrast, the neural network is expected to succeed, as it can recover information associated with higher-order correlations that go beyond first-order statistics. This behavior is consistent with our observations from dimensionality reduction techniques: while principal component analysis (PCA) shows that the two classes have nearly identical mean positions and similar variances, suggesting no separability at the linear level, a t-SNE embedding reveals the emergence of class-specific structure, indicating the presence of non-linear features in the data, as we can observe in Figure 3.6. As expected, the linear classifier is completely unable to distinguish class information, performing with a precision of 55%, comparable with the relative abundance of one class. On the other hand, as expected, a relatively shallow neural network composed of just one hidden layer of 16 ReLU units performs perfectly on the dataset.

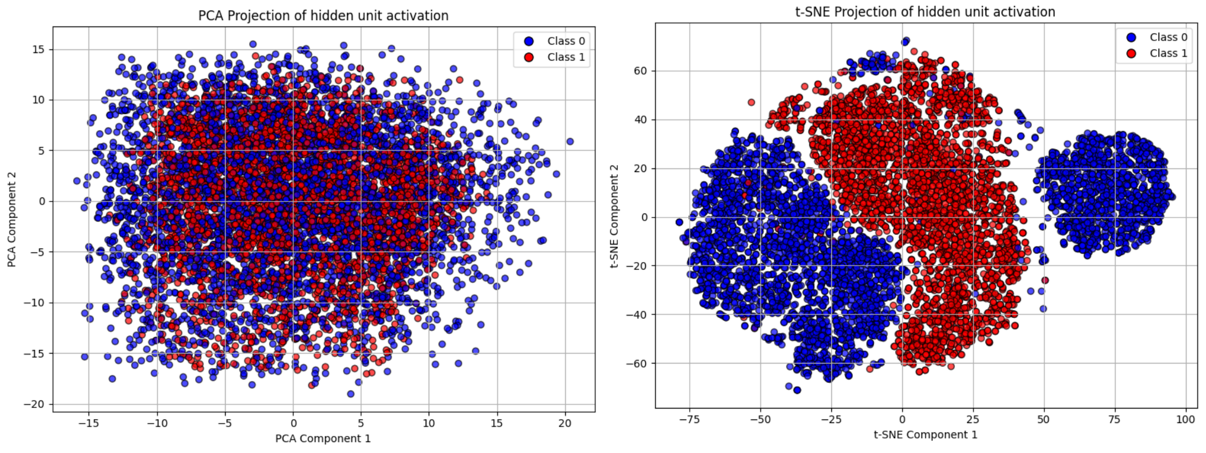

Figure 3.6: Comparison of dimensionality reduction techniques applied to the activations of the constrained hidden units (first-order constraint). PCA **Left**: shows significant overlap between classes, with similar means and variances, indicating the absence of linear separability. In contrast, t-SNE **Right**: reveals emergent structure and class-specific clustering, suggesting the presence of non-linear features not captured by first-order statistics.

3.2 WW domain

3.2.1 The dataset

The dataset used to train the RBM is a MSA of WW domains obtained from the PFAM database (PFAM ID PF00397). The data can be found in [11] and is composed of more than 60k sequences coming from 3700 different species. In [11], one can find a procedure to train an unconstrained RBM with a dual rectified linear unit (dRELu) potential, which allows to train an RBM which is able to predict the class (with a linear decision boundary) of the experimentally validated sequences by looking at the activation of two hidden units. In the following report, we will use the classification of such RBM as an additional and independent proof of the quality of the sequences generated by the constrained RBM trained by us. The constraints have been computed using sequences that were characterized by experiments [10]. In particular, class 1 is characterized by a highly conserved Histidine (H) in position 21, together with a Threonine (T) in position 26. Class 2 is characterized by aromatic amino acids (W, Y, F) in position 19 and basic amino acids (K, R) in site 21. To conclude class 3 is characterized by the absence of the gap in position 13 and the presence of Arginine (R) in positions 11 and 13 [9], [11]. These characteristics can be observed in the two sequence logos reported in Figure 3.7.

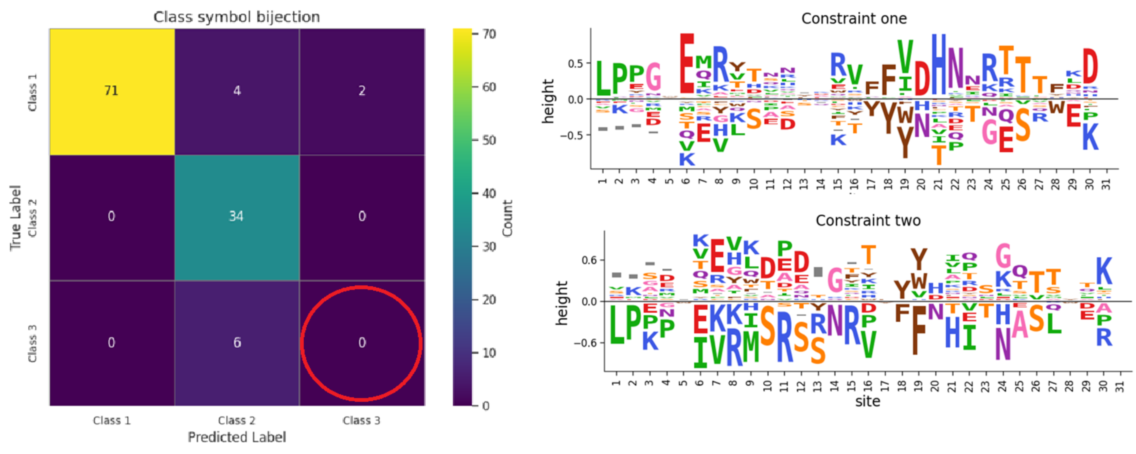

Figure 3.7: **Left**:Confusion matrix of the classified sequences in the first RBM trained. Predicted label is the non-zero element inside the Potts unit configuration. 1 (1,0,0); 2 (0,1,0); 3 (0,0,1). **Right**: We report the two sequence logo of the vectors $\mathbf{q}_a^{(1)}$, $\mathbf{q}_b^{(1)}$ used as constraints. Constraint one is the vector connecting the center of class 1 and class 2, Constraint two is the vector connecting the centers of class 2 and 3.

3.2.2 RBM setup

We train an RBM constrained at the first order with 150 constrained hidden units and 1 released Potts unit with $M = 3$. To compute the two constraints $\mathbf{q}_a^{(1)}$, $\mathbf{q}_b^{(1)}$ we use sequences that are already classified from experiments [10]. As reported in Section 2.5, we consider a 3-cluster adaptation, meaning that the constraints create a plane in weight space to which the binary unit weights have to be perpendicular.

3.2.3 WW domain partial results

It is known from literature that class 3 [10] is significantly underrepresented in natural sequences. Therefore, our first objective is to verify whether the model correctly assigns a distinct Potts unit symbol to each class in the dataset. This step is important because our algorithm does not explicitly enforce a one-to-one correspondence between classes and symbols—i.e., it does not impose information concentration. Instead, the model learns this mapping through likelihood maximization, which requires that each class has sufficient representation in the dataset. We can observe in Figure 3.7 that the class with the fewest elements (class 3) is completely not represented. This is, as mentioned above, due to the class imbalance of our dataset. To fix this problem, the natural solution is to develop an augmented dataset, where the data points classified by the classifier RBM get repeated 5 times to obtain balanced classes. In this way, the augmented data is composed of roughly 75k sequences, 50% classified as class 1, 33% classified as class 2, and the remaining belonging to class 3. Another possible option could have been to use sequences generated by the classifier RBM, filtering them to retain only those belonging to class 3.

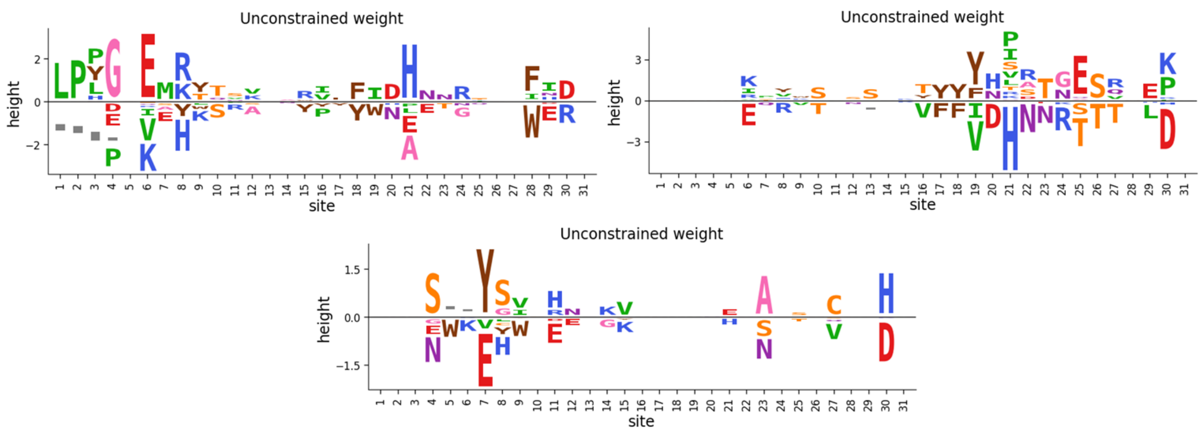

Figure 3.8: Sequence logo of the unconstrained Potts unit. We can recognize how the model has learned some important distinctions between classes described in Section 3.2.1. In particular we can observe the prominent H in position 21 accompanied by the T in position 26, as well as the aromatic amino acid in position 19. Other crucial distinction however were not captured by the model in particular the gap in position 13.

3.2.4 Energy landscape

The confusion matrix illustrating the bijective mapping established by the RBM between Potts unit symbols and sequence classes is reported in Appendix 4.3. We also report the weights of the trained RBM, which reflect how the Potts unit learned the distribution in the plane connecting the centroids Figure 3.8. Having confirmed that the RBM trained on the augmented dataset successfully learns a bijective mapping, we now turn our attention to analyzing the path followed by sampled sequences after a forced transition. Specifically, we aim to characterize the trajectories in the input space of the classified RBM and examine the energetic properties of these sequences.

As an initial step, we investigate the energy landscape associated with the classified sequences. As shown in Figure 3.9, different clusters are characterized by different typical energies. This observation provides an additional tool for assessing the quality of the transitions induced by the model.

3.2.5 Trajectories in classifier RBM

To assess the ability of our model to transition between classes we will now look at the trajectories of our generated sequences in the space of the two hidden unit used to classify sequences in the classifier RBM. In Figure 3.10 we can see two trajectory of different labeled sequences of class 1 being forcefully transformed into a sequence of class 3 (above) and class 2 (below) by modifying the value of the released unit. We can observe in the first transition (above) how while the Potts variable of the sequence correspond to class 3 the RBM [11] classifies it as belonging to class 2 most of the time. On the right we can see the free energies of the sequences sampled during the trajectories, by comparing with Figure 3.9 we can see how the transition was able to shift the energy from the typical energies of class 1 to energies that are well represented in class 3. Observing the second plot (below) we can see how the transitions from class

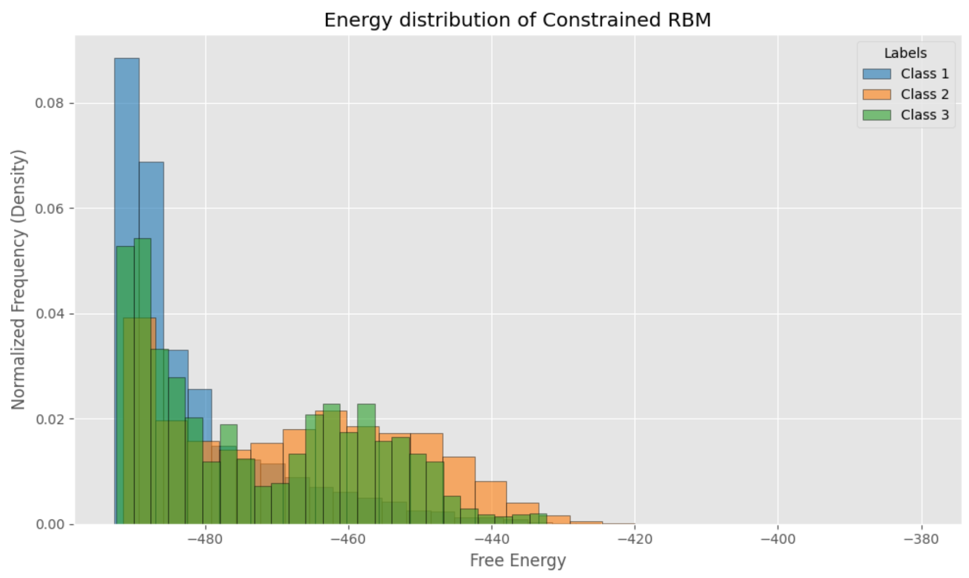

Figure 3.9: Histogram of the energies of the classified sequences. We can observe the three different classes with their relative typical energies.

1 to class 2 is very well reproduced by our model, in fact the free energy plot suggest that the sequences have energy typical of class 2 and the RBM classifier agrees with the value of the Potts unit.

3.2.6 Final results

As in Section 3.1.3, we study transitions induced by forcefully flipping the released unit. Specifically, we take each classified sequence and change its Potts unit value to enforce a class switch. To avoid sampling unlikely configurations, we run 5 Gibbs sampling steps before saving the resulting sequence, for a detailed plot see Appendix 4.3. While this number is somewhat arbitrary and dataset-dependent, especially since the LP model required no such “conversion” step, it was chosen based on the typical energy trajectories observed after transition.

In Figure 3.11, we show the sequence logos of transitions from labeled sequences. Panel (a) highlights transitions between Classes 1 and 2, with changes at positions 25–27 (Threonine) and position 21 (Histidine), in agreement with the characteristics of Class 1. These transitions closely follow the applied constraint and appear well captured by the model. In contrast, transitions in panels (b) and (c) show weaker agreement, likely due to class imbalance in the dataset, which may cause the constraint to represent a biased center, rather than the true centroid of the natural cluster. Nonetheless, key features such as the Class 3-specific gap at position 13, as well as Arginine (R) in position 11 and 13 are preserved [9, 10].

Overall, the results are encouraging given the data limitations. Future improvements could involve computing constraints utilizing the classifier RBM or incorporating biological information that is not represented in the dataset.

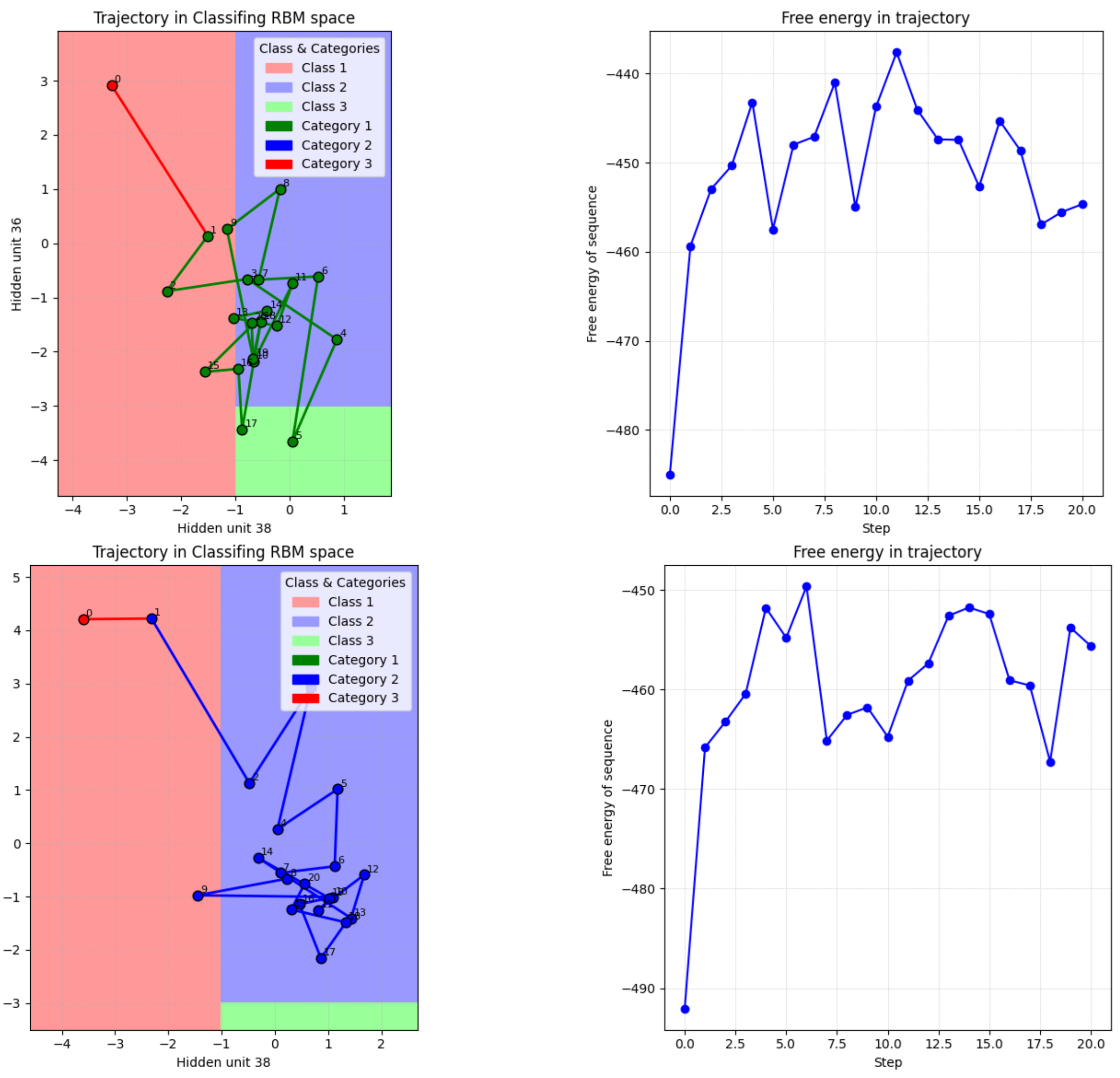

Figure 3.10: **Top**: Left: Trajectory of a natural sequence forcefully transitioned from class 1 to class 3, shown in the space of the two hidden-unit inputs of the classifier RBM. A discrepancy emerges between the Potts unit value (assigned to class 3, see Appendix 4.3) and the RBM classification in [11]. Right: The free energies along the trajectory show values typical of class 3, consistent with Figure 3.9. **Bottom**: Left: Trajectory of a class 1 sequence transformed into class 2. Here, the RBM classification agrees with the released unit. Right: Sequence energies are again characteristic of class 2.

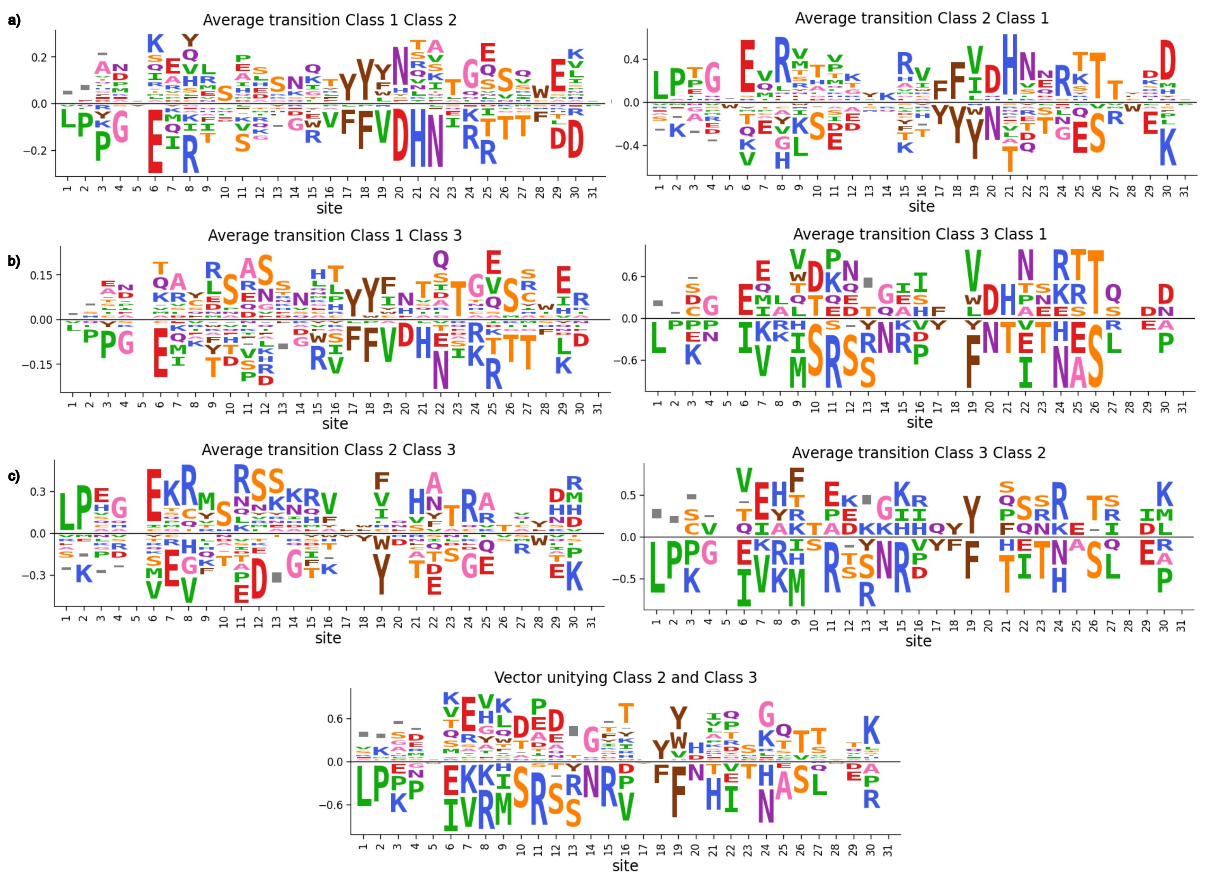

Figure 3.11: Sequence logo of the average transitions of the labeled sequences. **In panel a)** we see that the two transitions closely resembles the vector uniting the center of the clusters Figure 3.7 (constraint 1). In particular we can see very well the transition of the three Threonine (T) in positions 25-26-27 as well as the Histidine (H) in position 21. **In panel b)** we observe the transitions involving class 1 and 3, this transitions are a bit more noisy, in particular the transition from class 1 to class 3 [Figure 3.7 (constraint 2)]. **In panel c)** we see transitions of class 2 into class 3 and viceversa, we can observe the prominent gap in position 13, which is characteristic of class 1 and 2, as well as an R appearing in position 11 and 13. Note that the the transition in this panel are less precise due to the under representation of class 3 in both the total dataset (according to the RBM classifier) and in the labeled dataset.

4 Conclusions

4.1 Results

In this report, we have benchmarked the algorithm presented in [4] on an exactly solvable model describing proteins, obtaining non-trivial results which confirm the theoretically expected predictions. We showed how this algorithm allows to train RBM which concentrates the information about the attribute of interest on one hidden unit, allowing to easily modify one sequence to be part of one or the other cluster. We characterized how these transitions are parsimonious, in the sense that during the transition, only the class-relevant positions vary in the sequence. We explored analytically and computationally the effect of implementing a low-rank approximation of the second-order constraint, discussing how the number of constraints must be commensurate with the number of released units.

In the second part of the report, we extended the algorithm in [4] to multiclass attributes. To this end, we developed a publicly available Julia package that implements the training and sampling procedures of an RBM composed of 1 released Potts unit with K classes and M hidden binary unit. We investigated the model's performance on a real-world protein domain, characterizing the transitions between classes and highlighting the RBM's capability to learn the bijective mapping between class and label. We showed the ability of the constrained RBM to transition from one class to the other by flipping the released unit. We showed the difficulties of transitioning to classes that are less represented in the dataset, highlighting the role of dataset structure and labeled data to compute constraints that are representative of the entire dataset.

4.2 Future Perspectives

In this work, we have benchmarked and generalized to multiclass attributes the algorithm presented in [4]. The results obtained in this report show promising results, and the natural continuation of this work is the collaboration with an experimental team to verify the prediction and the sequences generated by the RBM.

The generalization presented in this report is, however only the beginning, as this algorithm applied to RBM can be further extended to broader and more biologically relevant contexts. A particularly promising direction concerns the ligand-receptor binding problem, which is a fundamental question in biology. This scenario is characterized by a family of receptors and a set of ligands that may or may not bind to each receptor. One is interested in disentangling the attribute associated with the ability or not of a given sequence to bind to a given receptor. The hope is that, as we saw in Chapter 3, disentangling this attribute could allow us to create new sequences that are specifically designed to bind to a given ligand while maintaining similarities to the starting sequence. This problem could be tackled by imposing a second-order soft constraint on the RBM that is being trained on the family of receptors. The idea is to penalize covariance between elements of the receptor and of the ligand. Once training has converged, one could combine the hidden units representation of the released units of a given sequence A with the hidden units representation of the constrained one coming from another sequence B to obtain a new sequence with the binding partner of A and the biological functions of B.

During this internship, we laid the groundwork for this extension by curating a dataset composed of proteins belonging to the PDZ domain family. This dataset, which was developed starting from data gathered by [12], can serve as a basis for future developments aimed at applying a variation of this disentangling algorithm to the receptor-ligand problem.

Another interesting direction to build on this project is to explore similar algorithms that can be applied to other types of generative model architectures. This is an active area of research, and new algorithms have been proposed in the literature. For example, disentangled representations have been obtained using encoder-decoder architectures [13]. A final promising direction is the use of advanced techniques from statistical mechanics to analytically study the problem of disentangling representations in RBMs. This raises important questions, such as the "cost" of achieving disentanglement and the theoretical limits of this approach when applying it to scenarios like multiple attributes at the same time or correlated attributes.

Bibliography

[1] Simona Cocco, Christoph Feinauer, Matteo Figliuzzi, Rémi Monasson, and Martin Weigt. Inverse statistical physics of protein sequences: a key issues review. *Reports on Progress in Physics*, 81(3):032601, Jan 2018.

[2] Jorge Fernandez-de Cossio-Diaz, Pierre Hardouin, Francois-Xavier Lyonnet du Moutier, Andrea Di Gioacchino, Bertrand Marchand, Yann Ponty, Bruno Sargueil, Rémi Monasson, and Simona Cocco. Designing molecular RNA switches with Restricted Boltzmann Machines. *bioRxiv*, 2023.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[4] Jorge Fernandez-de Cossio-Diaz, Simona Cocco, and Rémi Monasson. Disentangling representations in Restricted Boltzmann Machines without adversaries. *Phys. Rev. X*, 13:021003, Apr 2023.

[5] Asja Fischer and Christian Igel. An introduction to Restricted Boltzmann Machines. pages 14–36, 01 2012.

[6] Tijmen Tieleman and Geoffrey E. Hinton. Using fast weights to improve persistent contrastive divergence. In *International Conference on Machine Learning*, 2009.

[7] Hugo Jacquin, Amy Gilson, Eugene Shakhnovich, Simona Cocco, and Rémi Monasson. Benchmarking inverse statistical approaches for protein structure and design with exactly solvable models. Oct 2015.

[8] Eugenio Mauri, Simona Cocco, and Rémi Monasson. Mutational paths with sequence-based models of proteins: From sampling to mean-field characterization. *Phys. Rev. Lett.*, 130:158402, Apr 2023.

[9] Hunter T. Sudol M. NeW wrinkles for an old domain. *Cell.*, pages 1001–1004, 2000.

[10] Hai Hu. A map of WW domain family interactions. *Proteomics*, 4:643–55, 03 2004.

[11] Jérôme Tubiana, Simona Cocco, and Rémi Monasson. Learning protein constitutive motifs from sequence data. *eLife*, 8:e39397, mar 2019.

[12] Michael Stiffler. Pdz domain binding selectivity is optimized across the mouse proteome. *Science (New York, N.Y.)*, 317:364–9, 08 2007.

[13] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. Fader networks: Manipulating images by sliding attributes, 2018.

Appendix

In this section, we will present some graphs that allow one to better understand the discussion presented in Section 3.2 and Section 3.1.

Results on Electrostatic modes

Computing the first-order constraint

To compute the first order constraint we filter out from the dataset only the sequences that belong perfectly in one of the two structures. To do so we compute a scores, which assigns a value +1 for a positive amino acids and -1 for a negative in position [1,25] and the opposite for positions [2,16,18,20]. We can observe the histogram of the scores reported in red on the left in Figure 4.1. On the right we plot the two clusters obtained with this procedure on the first to PCs.

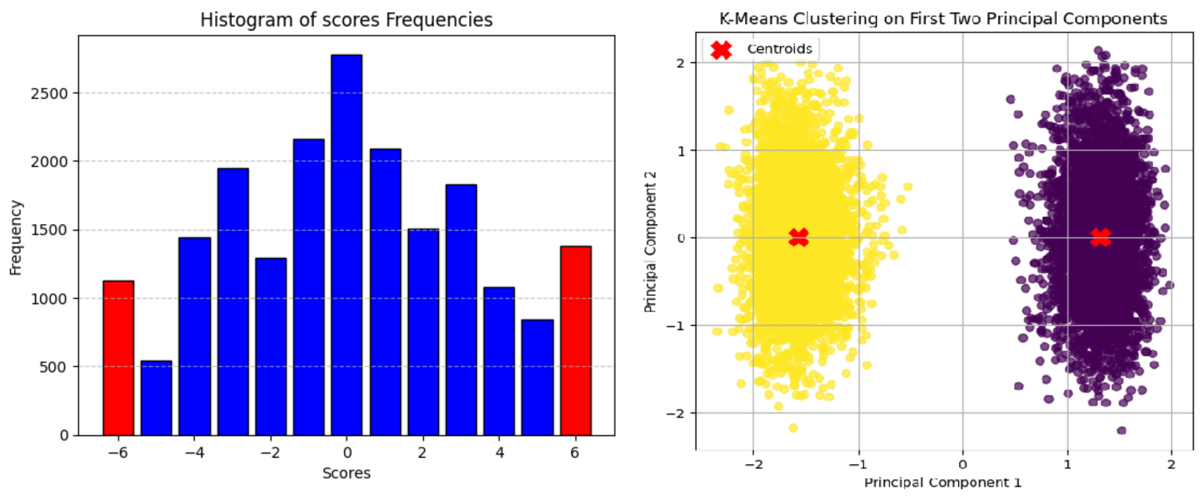

Figure 4.1: **Left**: histogram of the scores of the dataset, in red highlighted the sequences belonging to one of the two structures. **Right**: plot of the two clusters used to compute the constraints. In red the two centroids $\vec{c}_1, \vec{c}_0$. The vector $\mathbf{q}^{(1)}$ is computed as $\mathbf{q}^{(1)} = \vec{c}_1 - \vec{c}_0$

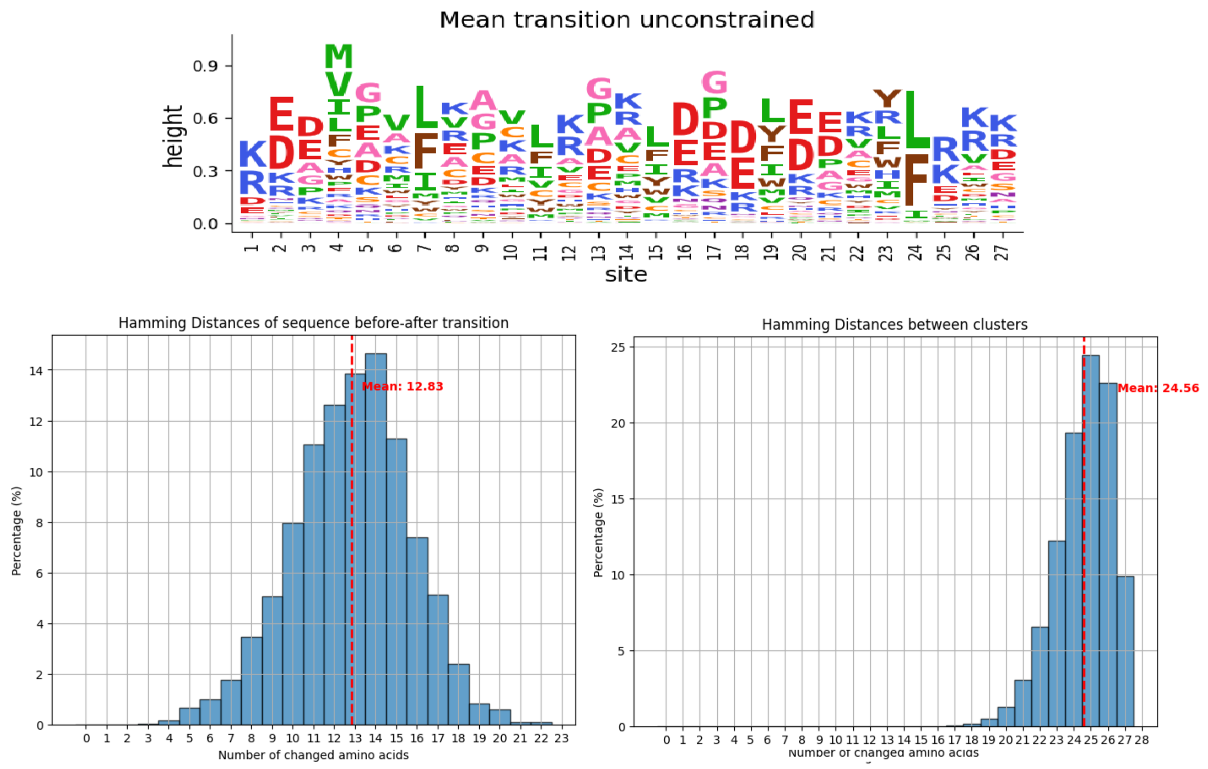

Figure 4.2: **Top**: Average transition, as defined in Section 3.1 of a random unit in the unconstrained RBM. One can us this transition as a baseline to judge the "quality" of transition in Figure 3.3. We can see how the changes in the electrostatic modes are lost inside the other changes. **Bottom** Left: Histogram of hamming distance of the sequence before and after transition. Right: Histogram of hamming distance of the sequences that compose the dataset, taking only into account distances of elements in opposite clusters. We can see how the sequences after the transition closely resemble the one at the beginning when we compare their hamming distance with the average hamming distance between classes.

Results WW domain

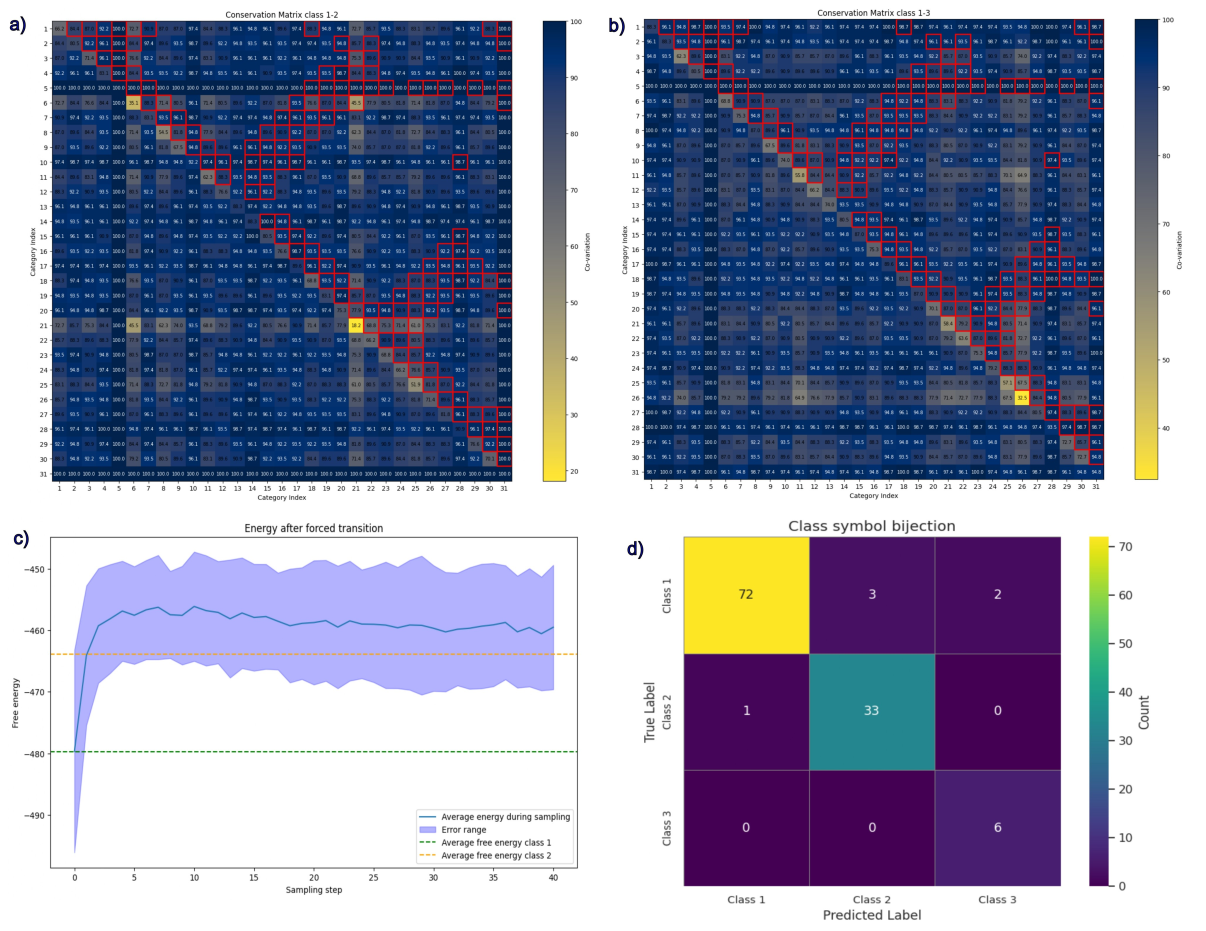

Figure 4.3: **a)**: Conservation matrix of the transition forced transition from class 1 to class 2. **b)**: Conservation matrix of the forced transition from class 1 to class 3. Circled in red we report the contact of the WW structure. **c)** Average energy of sequences of class 1 after a forceful transition to class 2, in particular we are interested in the time needed to reach equilibration in the sampling of class 2. **d)** Confusion matrix of the constrained RBM trained on the augmented dataset. We can see how the model was able to learn the one-to-one mapping between class and Potts symbols