POLYTECHNIC UNIVERSITY OF TURIN

MASTER THESIS

Generative statistical models for RNA sequences

Author:
Francesco CALVANESE

Supervisors:
Martin WEIGT
Andrea PAGNANI

A thesis submitted in fulfillment of the requirements for the Master degree in

Physics of Complex Systems

July 13, 2021

Abstract

The design of artificial biomolecules with given biological functions has become one of the main interests of biotechnology and bio-engineering in recent years. One of the goals in this field is the to improve natural molecules. Tha aim is to design artificial molecules that have the functionality of natural ones while being more stable/efficient/resistant.
The recent advances in sequencing technology have significantly speeded up and increased the amount of biological data available. Now it is finally possible to apply data-driven approaches to address this issue.
Generative models are tools in Machine or Statistical Learning used to generate artificial molecules that mimics the statistical features of natural ones, in the hope to also reproduce their biological functionality. There are several examples in the literature where these tools have been already applied successfully to proteins. In this thesis we apply them to RNA either designing new model architectures or adapting already existing ones.
The generative models treated are inspired from statistical physics. We use inverse statistical physics to build Potts models from which we sample artificial data. We test and compare several models having a special consideration for interpretability, since by analyzing the parameters of a good model we can deepen our understanding of the biophysics of RNA sequences.
As compared to proteins, the study of RNA has the advantage that the information on RNA secondary structure is easily accessible and there are efficient and precise algorithms for its prediction. We used secondary structure tests on our artificial sequences as an indicator for correct biological functionality. Furthermore, we used the information obtained from our models to build/improve structure prediction algorithms.
We conclude that, possibly after refinements based on experimental tests, that generative sequence models are good candidates for the design of artificial RNA sequences.

Contents

List of Abbreviations

DCA	**D**irect **C**oupling **A**nalysis
MC	**M**onte **C**arlo
MCMC	**M**arkov **C**hain **M**onte **C**arlo
NAT	**NAT**ural sequences
ART	**ART**ificial sequences
EDA	**E**dge **A**ddition **A**lgorithm
MEP	**M**aximum **E**ntropy **P**rinciple

Chapter 1

Introduction

1.1 Aim of the Thesis

The aim of this thesis is to find ways to generate artificial RNA molecules that are able to replicate structure and functionality of natural ones (these techniques are called "generative models"). This kind of problem is not new: the design and the generation of bio-molecules with given functions has been one of the main interests of biotechnology and bio-engineering in the last 20 years. The majority of the work in this direction has been done on proteins [1]: generative models are used as an effective tool in the design of antibodies and artificial sequences have been shown to be able to effectively substitute natural ones in microorganisms [17]. Besides these clear direct applications there are also several additional benefits: the theoretical apparatus developed for the construction of effective generative models can be (and has been) used to study mutations, folding dynamics and native contact prediction of bio-molecules [11] and there are many more possibilities, for example the work done on RNA in this thesis serves as a basis for a PhD project that aims to study the plausibility of the "RNA World" theory that is one of the leading hypothesis for the origin of life on this planet [16].

1.2 Generative models

Generative models are Machine Learning tools with the aim of generating artificial data that reproduces the important features of natural data.
A possible definition is the following [8]:

A generative model describes how a data-set is generated, in terms of a probabilistic model. By sampling from this model, we are able to generate new data

A naif but effective example is the following: suppose we want to generate artificial images of kittens. The first thing we have to do is to explain to our generative model how they look like. To do so we feed our models with a large number of kitten images so that he can learn the general rules of appearance of a kitten. This phase in Machine Learning is called "training" and the images fed to the model are called "training set". After this, if the model is well made, it should be able to generate brand new images of kittens and we should not be able to tell if an image is artificial or it is an actual photo. This phase is called "sampling" and the generated pictures are called "artificial data" (as opposed to "natural data" that is data coming from real-life settings).

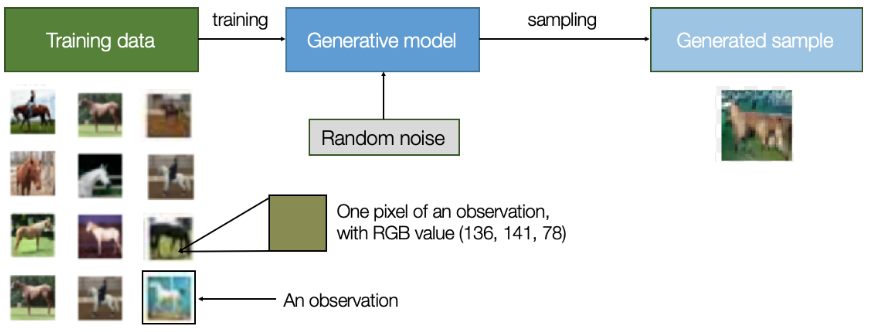


FIGURE 1.1: Representation of the process (with horses) [8]

There are two main issues with generative models:
The first problem is when the artificial data is too different from the natural data of the same kind. If the generated images do not resemble kittens of course our model is failing.
The second problem is when the model samples slight variations (or exact copies) of the training set. In this case it is obvious that we are not able to differentiate between natural and artificial data but we are also not gaining any new information nor exploring any new variability (for example if we want to find a protein with a certain antibody function that is more efficient than the ones present in nature copying natural data will not bring us very far).

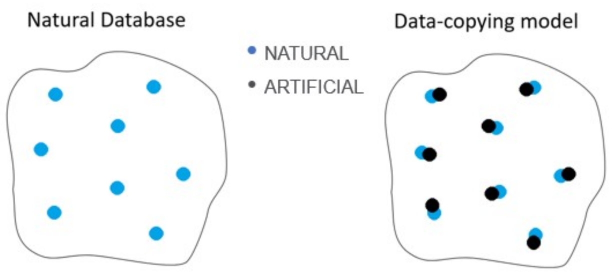
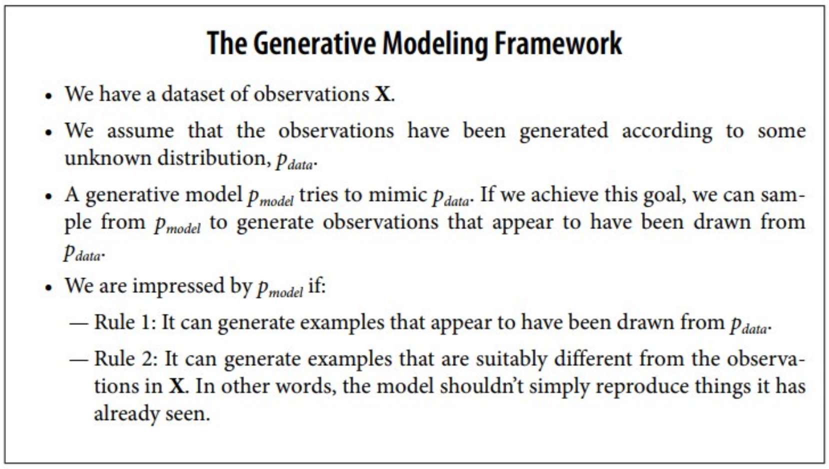


FIGURE 1.2: Data is represented by points, the model is clearly falling under the second issue. This is called "data copying prolem"

If the generative model operates with probabilities (which will be the case throughout all the thesis) it is called a "probabilistic generative model". For this kind of models all the observation done till now can be summarized with the "generative models framework" [8]:


The Generative Modeling Framework

- We have a dataset of observations **X**.
- We assume that the observations have been generated according to some unknown distribution, p_{data}.
- A generative model p_{model} tries to mimic p_{data}. If we achieve this goal, we can sample from p_{model} to generate observations that appear to have been drawn from p_{data}.
- We are impressed by p_{model} if:
  — Rule 1: It can generate examples that appear to have been drawn from p_{data}.
  — Rule 2: It can generate examples that are suitably different from the observations in **X**. In other words, the model shouldn't simply reproduce things it has already seen.

FIGURE 1.3: Probabilistic generative models framework

In this thesis we will study several model in the context of RNA, either adapting already existent protein models or designing new architectures.

1.3 RNA introduction

Ribonucleic acid (RNA) is a polymeric bio-molecule essential in many biological processes. It is most famous because it can carry, code and decode genetic information (mainly in the protein synthesis process) but it can also have catalytic (Rybozimes) and structural functions. RNA molecules rivals protein for their versatility.
Each RNA is primarly a one dimensional polymer composed by nucleotides that folds into convoluted shapes. Along with the nucleotide sequence the correct folding is necessary for the functionality of the molecule.
The three-dimensional structure of RNA is deeply connected with the functionality of the sequence [19]. The knowledge of the first is key to understand (and potentially engineer) the latter.

1.3.1 RNA structure

The building monomers of RNA molecules are nucletides. There are four possible nucleotides (for RNA): Adenine, Guanine (purine bases), Cytosine and Uracil (pyrimidine bases). As in DNA they can form base pairs (Cytosine-Guanine and Adenine-Uracil). The presence of self-complementary segments in the RNA strand leads to intrachain base-pairing and folding into complex forms consisting of bulges and helices. The correct folding of RNA is critical to its stability and function. In addition to the canonical base-pairing it's also possible to form a wobble base-pair between (Adenine-Guanine) even though it's less stable than the others.

The structure of RNA molecules can be easily understood considering three different levels:

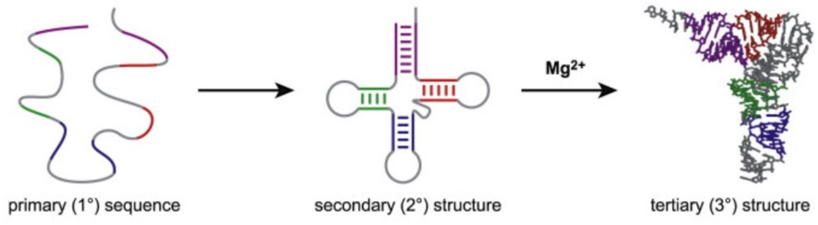


FIGURE 1.4: Intuitive representation of the hierarchical level of structural organisation in RNA

- Primary structure: monodimensional sequence of nucleotides
- Secondary structure: 2D structure driven by the intrachain base-pairing, it divides the sequences in several ends/domains
- Tertiary structure: 3D structure driven by the interaction of the various ends/domains of the molecules

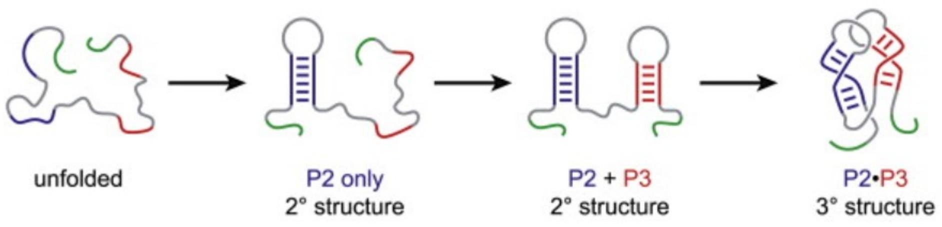


FIGURE 1.5: Illustration of how secondary structure domains can interact and construct the shape of the sequence

In this thesis we highlighted the importance of secondary structure (there is a whole chapter dedicated to it). Algorithms for RNA secondary structure prediction are really efficient and accurate. Since we were not able to do experimental tests we used secondary structure reproduction as a form of "biological functionality test" for our artificial sequences.

1.3.2 Data: RNA families

The true protagonist of every Machine Learning problem is the data. Looking at the data we can answer two important questions that justify the approach taken in this thesis:

Is it necessary to use Machine Learning?

Often Machine Learning models and approaches lack of interpretability. It is true that our goal is to generate functional artificial RNA sequences but we would also like to gain insight and knowledge on the underlying science during the process. This not a point in favour of ML. The motivations behind the choice become clear looking at the data. Let's suppose we are dealing with sequences that are 200 nucleotides long (realistic order of magnitude). Since each nucleotide has four possible options we see that the sequence space of our problem contains $4^{200} \sim 2.6 \times 10^{120}$ to put this number in prospective the estimated number of atoms in the known universe is 10^{78}. At the moment there exists no better technique to explore these huge spaces than a computer aided statistical approach.

Is it possible to use Machine Learning?

Machine Learning techniques are very data hungry and just a decade ago it was impossible to utilize this approaches to RNA sequence data. Thanks to the recent revolution in sequencing technology a lot of sequences have become available and are accumulating at exponential speed. Now we have the possibility to use these methods. Clearly we have to be really careful since the available data is not yet at the level of the more classical Machine Learning techniques.

RNA sequence data is organized in RNA families [10]. In the course of organisms evolution RNA evolves as well. Even if function and structure do not change much during evolution different organisms can accumulate mutations and now present analogous RNA molecules that are very different from each other.
It is natural to group molecules which have similar structure and function and come from the same evolutionary branch (homologous sequences) into labelled collections and consider them as variants of the same sequence. The set of such sequences makes an RNA family. The Rfam database is a collection of RNA families and to date it contains data for 3940 families. Each family contains $\sim 10^3$ sequences.

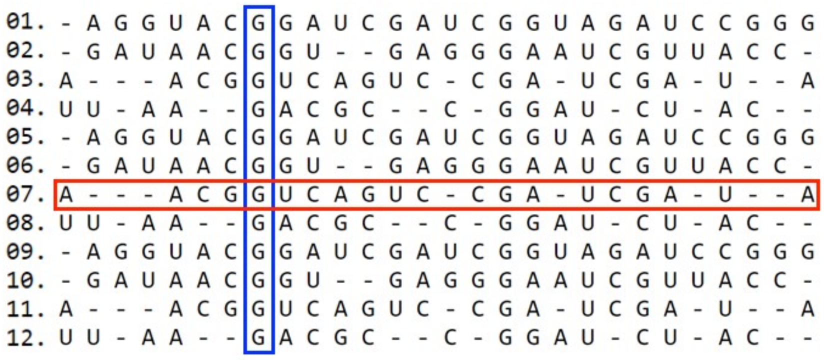
The effect of evolution on RNA sequences is not only to modify the type of nucleotide but also their number, it's not uncommon that in a a family we have a variability of more than ± 20 nucleotides. Since we want to do a statistical analysis we would like to know what are the corresponding sites/regions on different molecules so the variable length is a huge problem. This issue corresponds to a whole field of bioinformatics called "sequence alignment" [7]. In this thesis we will not deal with this problem since all the families are taken form Rfam and are already aligned with the most up to date techniques.

Chapter 2

RNA Generative Models

2.1 Introduction

As discussed in the introduction, generative models want to generate artificial RNA sequences that mimics the important statistical features of the natural ones. The underlying assumption is that the biological information (about structure and functionality) is hidden in the statistical proprieties of the data. From the replication of the latter should come also the replication of the first.



```
01. - A G G U A C G G A U C G A U C G G U A G A U C C G G G
02. - G A U A A C G G U - - G A G G G A A U C G U U A C C -
03. A - - - A C G G U C A G U C - C G A - U C G A - U - - A
04. U U - A A - - G A C G C - - C - G G A U - C U - A C - -
05. - A G G U A C G G A U C G A U C G G U A G A U C C G G G
06. - G A U A A C G G U - - G A G G G A A U C G U U A C C -
07. A - - - A C G G U C A G U C - C G A - U C G A - U - - A
08. U U - A A - - G A C G C - - C - G G A U - C U - A C - -
09. - A G G U A C G G A U C G A U C G G U A G A U C C G G G
10. - G A U A A C G G U - - G A G G G A A U C G U U A C C -
11. A - - - A C G G U C A G U C - C G A - U C G A - U - - A
12. U U - A A - - G A C G C - - C - G G A U - C U - A C - -
```

FIGURE 2.1: outline of an RNA family: a row corresponds to the primary sequence of a molecule and a column represents the corresponding site across all the family. "-" represents gaps (absence of nucleotide, necessary for the alignment of different length sequences)

A probabilistic generative models looks at the natural data and tries to infer an approximation of the probability distribution $P(a_1, a_2, \ldots, a_L)$ that generated it. From this approximation we can sample artificial sequences that replicate the features of the natural ones.

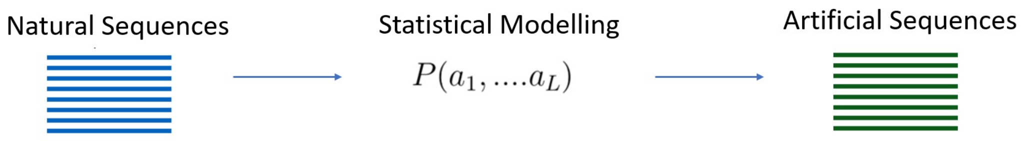


FIGURE 2.2: Functioning of a generative model

But what are the statistical features contained in the RNA family that we would like to replicate?

- One-point frequencies $f_i(a_i)$ (frequency of nucleotide a_i in site i across the family)
- Two-points frequnecies $f_{ij}(a_i, a_j)$ (frequency of the pair (a_i, a_j) in site i and j respectively)

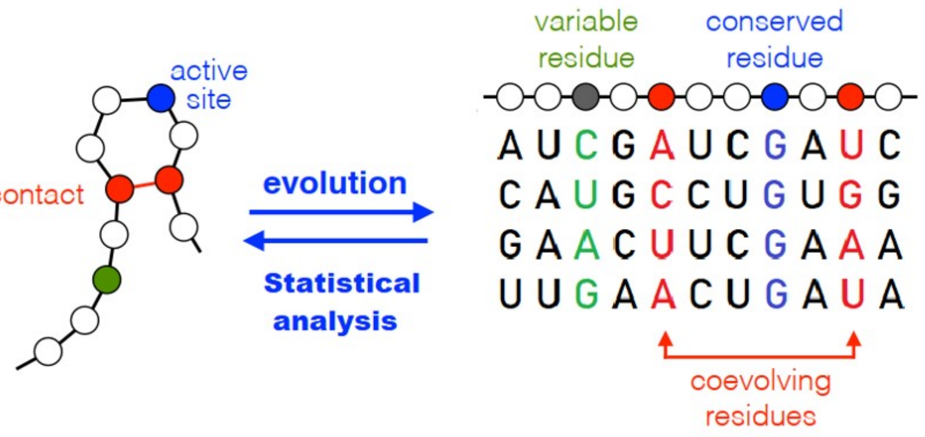


FIGURE 2.3: biological interpretation of one and two points frequencies

These two quantities have an easy biological interpretation.
One point frequencies $f_i(a_i)$ effectively represent the evolutionary phenomenon of conservation. Let's suppose that a site is directly involved in the functionality of the molecule (for example it has to bind to a precise substrate or has to posses precise biochemical proprieties), then a mutation that changes the nucleotide on this site is very likely to be deleterious. In this case the nucleotide type on this site is *conserved* among the family. When a nucleotide is conserved the value of $f_i(a_i)$ is ~ 1 for a nucleotide type and very small for all the others.
Two points frequencies $f_{ij}(a_i, a_j)$ effectively represent the evolutionary phenomenon of co-evolution. Let's suppose that a pair of sites is in contact in the folded state of the molecule. Then, if one of the two changes due to a mutation the other wants to change too to maintain the ability to form the bond. Across the family these two sites appear most often as complementary pair. This is the co-evolution and its effect can be seen on the two point frequencies $f_{ij}(a_i, a_j)$.

In all the proposed models we tried to replicate these two quantities. Of course they do not correspond to all the statistical information contained in the RNA family because there are also all the higher order frequencies. We have to consider some things:

- The $f_{ijk}(a_i, a_j, a_k)$ consists of $5 \times 5 \times 5 = 125$ terms. A database of length ~ 1000 sequences [10] is not adequate to have a fair representation of it. Models that try to fit higher order frequencies will most likely be fitting the noise in the training data (overfitting in Machine Learning terminology)

- There are several examples in the literature that suggest that one and two point frequencies are sufficient to obtain functional artificial sequences [17] [11]
- Models that try to replicate one and two point frequencies will often have fairly good representation of higher order frequencies without trying to fit them directly [18] [17]

For all these reason we will just concentrate on the first two orders of the frequencies.

2.1.1 Sampling Bias

The main assumption of probabilistic generative models is that there exists a probability distribution $P(a_1, a_2, \dots a_L)$ for each RNA family and that all the sequences in a given family are i.i.d. variables drawn from it.
The problem is that the sequences in a RNA families often do not respect this assumption. The aim of the scientists is not always to fairly explore the sequence space so they collect data in a biased way [14]. For example if they are studying a certain RNA molecule in rodents they will sequence it for several different species of rodents and so they will add to the RNA family a lot of sequences similar to each other. Looking at the family one could be misled and think that the $P(a_1, a_2, \dots a_L)$ has a sharp peak in the region of those sequences.
Also, due to phylogenetic effects, natural sequences can not be considered independent from each other. RNA sequences in a family are not obtained from independent *sampling* because they descend from each other through evolutionary branches built from the accumulation of mutations.
This issue is called sampling bias. Addressing it is an important problem in bioinformatics. In this thesis we have taken a simple approach. For the computation of the $f_i(a_i)$ and $f_{ij}(a_i, a_j)$ we assigned a weight that reduces the relative importance of too similar sequences:

$$f_i(a) = \frac{\sum_{k=1}^{N} \omega_k * \delta_{a_i^k, a}}{\sum_{k=1}^{N} \omega_k} \tag{2.1}$$

$$f_{ij}(a,b) = \frac{\sum_{k=1}^{N} \omega_k * \delta_{a_i^k, a} \delta_{a_j^k, b}}{\sum_{k=1}^{N} \omega_k} \tag{2.2}$$

with N being the number of sequences in the family. ω_k is the weight associated to the k^{th} sequence of the family and it is equal to $\frac{1}{n_k}$ where n_k is the number of sequences that share more than 80% common nucleotides with sequence k.
If we set all the weights equal to one we retrieve the usual definition of the frequencies.

The fact that not all sequences have the same importance changes the effective size of the RNA family.
We define N_{eff} as:

$$N_{eff} = \sum_{i=1}^{N} \omega_i \tag{2.3}$$

$N_{eff} = N$ if all the weights ω_i are equal to 1 (when there are no sequences that are too similar) otherwise it is smaller.

2.1.2 Model testing

When we generate artificial sequences with a generative model we have to test if they are valid, in particular we have to check some things:

- Do artificial sequences replicate the statistics of the natural data?
- Are generated sequences just a copy of the training data?
- Do artificial sequences replicate the structure and functionality of the natural data?

To answer these questions we decided a set of tests.

For the statistical part we checked the correct reproduction of conservation and co-evolution. All our models almost perfectly replicate the $f_i(a_i)$ (conservation statistics). For the co-evolution test we did not use $f_{ij}(a_i, a_j)$ but we preferred to use the connected two point correlations $C_{ij}(a_i, a_j) = f_{ij}(a_i, a_j) - f_i(a_i)f_j(a_j)$. This because this quantity isolates the co-evolutionary information contained in the $f_{ij}(a_i, a_j)$ form the effect of conservation. To be more precise we can write $f_{ij}(a_i, a_j)$ as:

$$f_{ij}(a_i, a_j) = f_i(a_i)f_j(a_j) + \epsilon_{ij}(a_i, a_j) \tag{2.4}$$

where :

- $\epsilon = 0$ if the two sites are independent (do not co-evolve)
- $\epsilon > 0$ if the pair (a_i,a_j) is favoured by co-evolution
- $\epsilon < 0$ if the pair (a_i,a_j) is disadvantaged by co-evoluton

We can see that all the co-evolutionary information is contained in ϵ and if we look at the definition we have that:

$$\epsilon_{ij}(a_i, a_j) = f_{ij}(a_i, a_j) - f_i(a_i)fj(a_j) \tag{2.5}$$

$$\epsilon_{ij}(a_i, a_j) = C_{ij}(a_i, a_j) \tag{2.6}$$

So it's natural to use the correlations as a test for the correct reproduction of co-evolution statistics.

After we have ensured the correct reproduction of the statistical properties we have to be sure that our generated data is not falling under the data copying problem. When we are dealing with low dimensional data it is immediate to say if our generative model is data copying or not. This is clear looking at the following example:

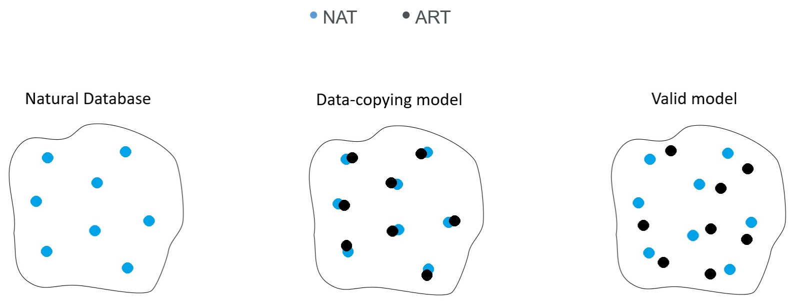


FIGURE 2.4: Blue dots are natural data (NAT) black dots are artificial data (ART)

We can immediately say that the model in the middle is data copying whereas the model on the left is more fairly exploring the sequence space.
The problem of applying this method to RNA is that sequences are very high dimensional and the visualization becomes tricky. If we are dealing with molecules that are 100 nucleotides long (and this is the shorter side of length scales present in nature) there is no way in which we can visualize two 100 dimensional data-sets and say if they are too similar. To solve this issue we can use a popular technique in Machine Learning called PCA (Principal Component Analysis) [12]. PCA is a technique of linear dimensional reduction. It finds the direction with maximal variance (principal directions) and uses them as coordinates to have a low dimensional representation of high dimensional data.

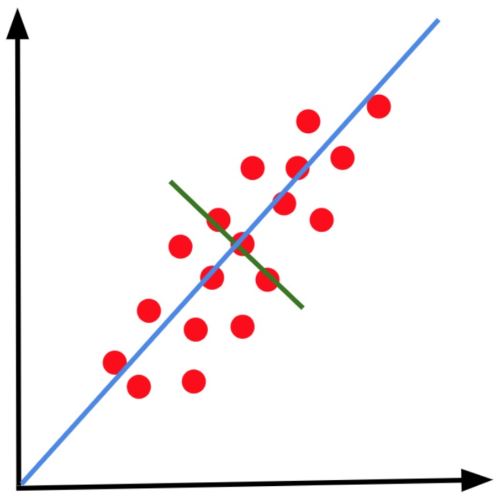


FIGURE 2.5: Example of PCA: the blue line represent the first principal direction; if we want a one dimensional representation of this 2-D data we can use its coordinate on the blue line losing the least amount of information

This techniques are very suited to sequence data and often the first two principal directions can explain up to the 10% of the total variance of the data-set.

Giving the projection on the first two principal directions we are able to say if the generated data is introducing diversity or if it is just copying the training set.

Lastly we have to check if our generative model is able to replicate biological structure and functionality. Sadly during this thesis we were not able to do experimental tests. Instead of this we did secondary structure test with prediction algorithms and saw if they predicted the same secondary structure for natural and artificial sequance.

2.1.3 RNAalifold

RNAalifold is a RNA secondary structure prediction algorithm [4]. It integrates a classical biophysical approach with the statistical information contained in RNA families. From a sequence alignment it predicts the consensus secondary structure of a family with more than 90% accuracy. It is one of the algorithms contained in the Vienna package, a set of tools used to study the RNA.
For more information on secondary structure prediction see section 3.

2.1.4 Potts Model

All the generative models in this thesis (except the Autoregressive) will be based on a Potts Model Hamiltonian.
The Potts model, a generalization of the Ising model, is a model of interacting spins on a lattice. Each spin is a discrete degree of freedom and can assume q different value. In the RNA case $q = 5$ (four values for the nucleotides and one for the gap).
The lattice consists in a set of vertex V and a set of edges between vertexes E. In our case V will be the set of sites and E the set of interacting pairs which will change depending on the model.

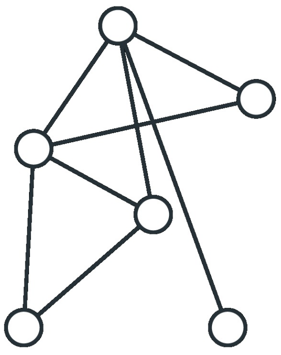


FIGURE 2.6: Exemple of vertex and edges for a Potts model on a graph

In the Potts model we have:

$$H(a_1, a_2, \dots, a_L) = -\sum_{i \in V} h_i(a_i) - \sum_{(i,j) \in E} J_{ij}(a_i, a_j) \tag{2.7}$$

$$P(a_1, a_2, \dots, a_L) \propto e^{-H} \tag{2.8}$$

The $h_i(a_i)$ represents the local field term used to tune the single site frequencies. The higher is $h_i(a_i)$ the higher becomes $f_i(a_i)$. This term models conservation.
The $J_{ij}(a_i, a_j)$ represents the pair interaction between site i and site j and it is used to tune $f_{i,j}(a_i, a_j)$. the The higher is $J_{i,j}(a_i, a_j)$ the more often (a_i, a_j) appear as a pair in site i and j in our generated sequences. This term models co-evolution.

The complexity of a Potts model architecture depends on how many interaction edges we have in the set E.
For a RNA family of length L we always have qL parameters for the local fields $h_i(a_i)$ (q parameters for each field). Then we have $q^2|E|$ parameters for the interactions $J_{ij}(a_i, a_j)$ since each one of them is a $q \times q$ matrix. $|E|$ can go from 0 (independent site model with no interactions) to $\frac{L(L-1)}{2}$ (fully connected model with all possible interactions).

2.2 Reference models

To see if we should be pleasantly surprised with our generative models we should have something to compare them with. We will build two really simple reference models that contain a minimal amount of information. Both models represent a widely used standard in bioinformatics. If our more complex models do not perform substantially better than them it means that they are not working properly.

2.2.1 Independent sites model

The independent sites model (also known as "profile model") is the simplest non-trivial generative model and doing better than it is the bare minimum. It is based on a Potts model Hamiltonian without pairwise interaction:

$$H(a_1, a_2, \dots, a_L) = -\sum_{i \in V} h_i(a_i) \tag{2.9}$$

$$P(a_1, a_2, \dots, a_L) \propto e^{-H} \tag{2.10}$$

The probability is factorizable as follows

$$P(a_1, a_2, \dots, a_L) = P_1(a_1)P_2(a_2)\dots P_L(a_L) \tag{2.11}$$

$$P_i(a_i) = \frac{e^{h_i(a_i)}}{Z_i} \tag{2.12}$$

where

$$Z_i = \sum_{a_i=1}^{5} e^{h_i(a_i)} \tag{2.13}$$

The parameters $h_i(a_i)$ are tuned in order to reproduce the single site frequencies $f_i(a_i)$.
Looking at the sampling method it's evident that we don't actually have to learn any parameter. In order to reproduce single sites frequencies $f_i(a_i)$ we can just draw the type of nucleotide on each site from it's frequency.

$$P_i(a_i) = f_i(a_i) \tag{2.14}$$

setting $Z_i = 1$ (which is always possible) we have

$$P_i(a_i) = f_i(a_i) = e^{h_i(a_i)} \tag{2.15}$$

$$h_i(a_i) = log(f_i(a_i)) \tag{2.16}$$

Here are the results of the tests done on the RF00010 RNA family of the Rfam database (N=6479):

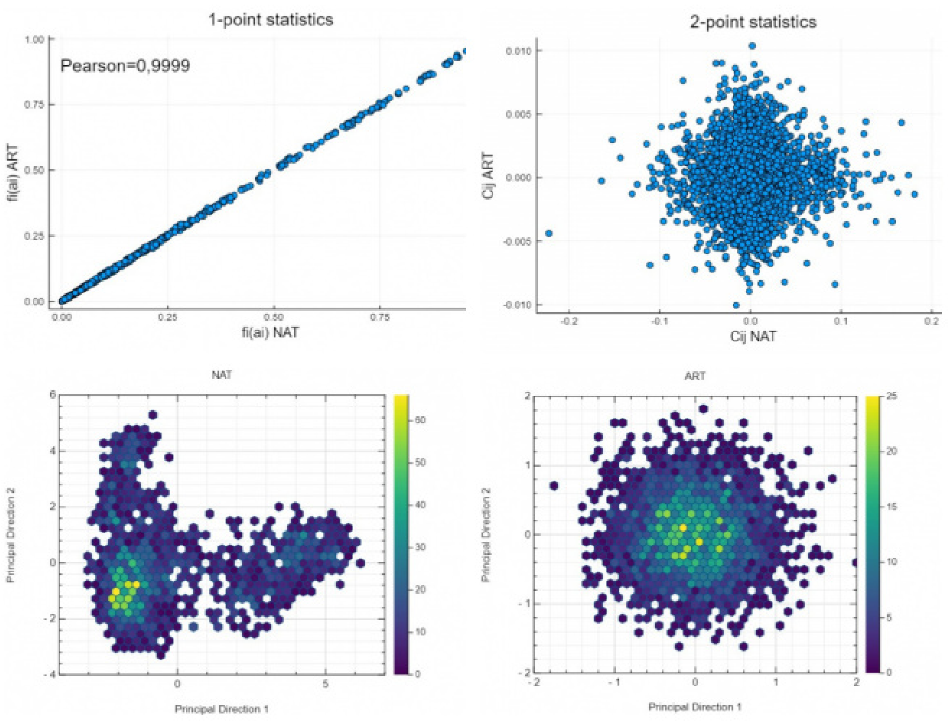


FIGURE 2.7: One point frequencies, two point correlation and PCA projection (6000 artificial sequences)

One point frequencies are correctly reproduced, two point correlation and PCA projection are a failure.

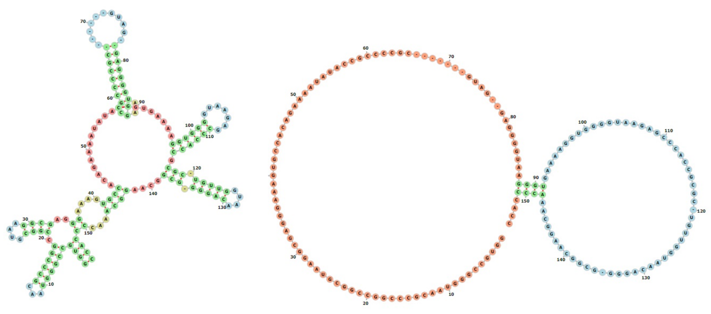


FIGURE 2.8: Left: RNAalifold prediction for 300 natural sequences, Right: RNAalifold prediction for 300 artificial sequences

Artificial sequences do not have the right secondary structure.

2.2.2 Secondary Structure model

Since the information about the secondary structure of an RNA family is easily available, [10] [4] it can be implemented to build our reference model. The secondary structure model (also called "covariance model" in bioinformatics) is obtained adding this information to the local field model discussed before. We impose the replication of single site frequencies $P_i(a_i) = f_i(a_i)$ for all sites and the replication of two site frequencies $P_{ij}(a_i, a_j) = f_{ij}(a_i, a_j)$ for the pairs involved in secondary structure contacts.
In the Potts model point of view the system consists in a set of independent sites and a set of interacting pairs, all of which are independent from each other.

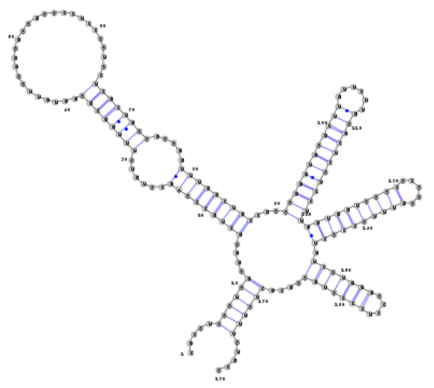


FIGURE 2.9: Interacting pairs are represented with blue edges

V contains all the sites while E now corresponds to the secondary structure pairs. The probability distribution can be written as follows:

$$P(a_1, a_2, \ldots, a_L) = \prod_{i \in V} f_i(a_i) \prod_{i,j \in E} \frac{f_{ij}(a_i, a_j)}{f_i(a_i) f_j(a_j)} \tag{2.17}$$

This expression is valid in general when the pairs of which we want to fit the $f_{ij}(a_i, a_j)$ do not form a cycle (of course secondary structure interacting pairs can not form a cycle since each site interacts at most only with another one).
We can also find the value of the local fields and interactions:

$$H(a_1, a_2, \ldots, a_L) = -\sum_{i \in V} h_i(a_i) - \sum_{i,j \in E} J_{ij}(a_i, a_j) \tag{2.18}$$

$$h_i(a_i) = log(f_i(a_i)) \tag{2.19}$$

$$J_{ij}(a_i, a_j) = log\left(\frac{f_{ij}(a_i, a_j)}{f_i(a_i) f_j(a_j)}\right) \tag{2.20}$$

Doing e^{-H} with these choices we retrieve expression 2.17.
This is not the only possible choice, there are several gauges for the choice of local field h_i and interaction terms J_{ij}. This particular choice is aimed to assign all the conservation representation to the local fields and all co-evolution representation to the interactions.
As in the previous case we do not really have to do any learning since the sampling is straightforward: we chose the nucleotide type on all sites not involved in

the secondary structure drawing from the frequencies $f_i(a_i)$ and the nucleotide pair on interacting couples drawing from the respective $f_{ij}(a_i, a_j)$.
Here are the results of the tests done on the RF00010 RNA family of the Rfam database (N=6479):

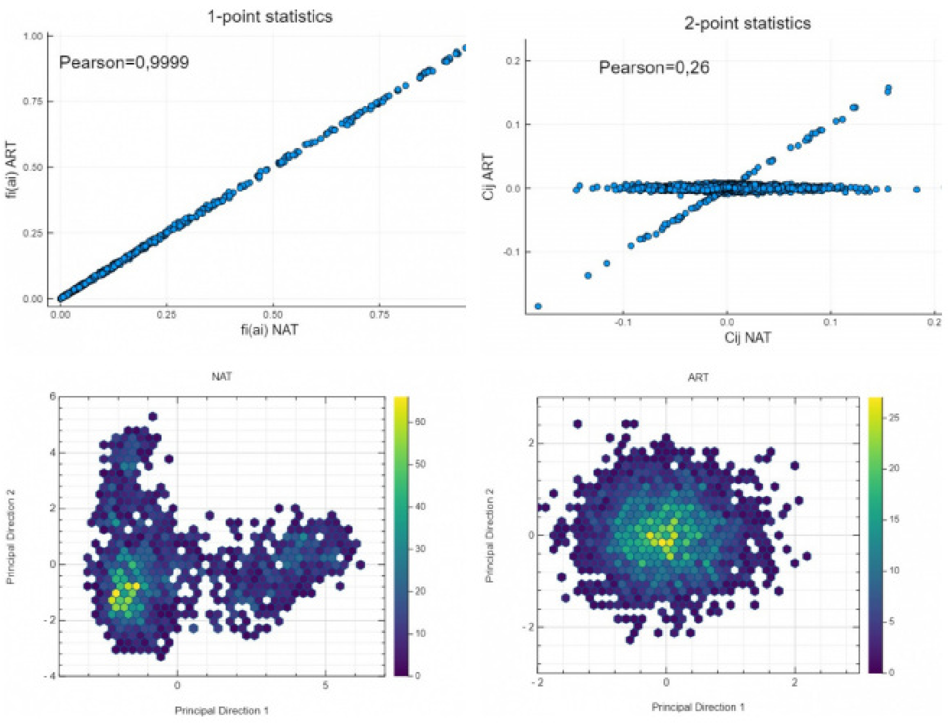


FIGURE 2.10: One point frequencies, two point correlation and PCA projection (6000 artificial sequences)

The model fits perfectly all the $f_i(a_i)$ for all sites and the $C_{ij}(a_i, a_j)$ for all 45 secondary structure pairs. It fails for all the others. PCA projection evidences that the model fails to capture essential statistical features of the distribution.

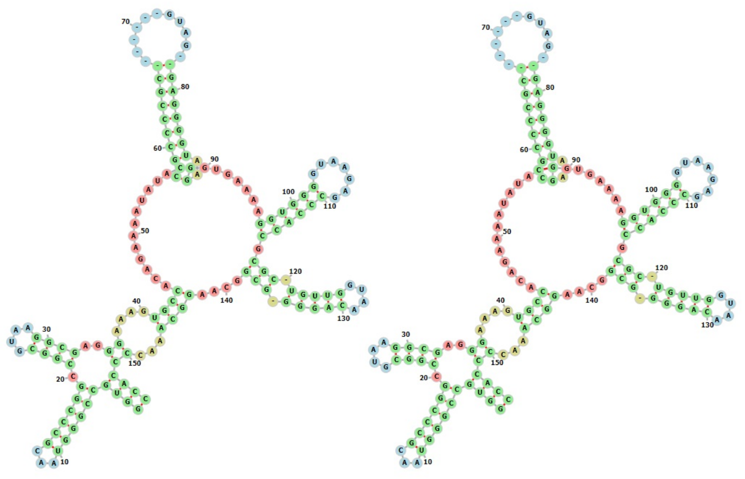


FIGURE 2.11: Correct reproduction of secondary structure

RNAalifold predicts the same secondary structure for natural and artificial database.

Both natural and artificial database need ~ 5 sequences to let the structure appear from the prediction algorithm.

2.3 Best Tree Model (Bayesian Network)

If the set of edges E does not contain cycles using the formula 2.17 as $P(a_1, a_2, \ldots, a_L)$ imposes that the marginals $P_i(a_i)$ are equal to $f_i(a_i)$ for all sites and the $P_{ij}(a_i, a_j)$ are equal to the $f_{ij}(a_i, a_j)$ for all pairs contained in E. When this is the case it is possible to show that there is a really convenient method for sampling [5] that will be briefly described in the following.
Since there are no cycles, from a graph theory point of view, our systems consists either in a tree (connected acyclic graph) or in forest (set of disconnected trees).

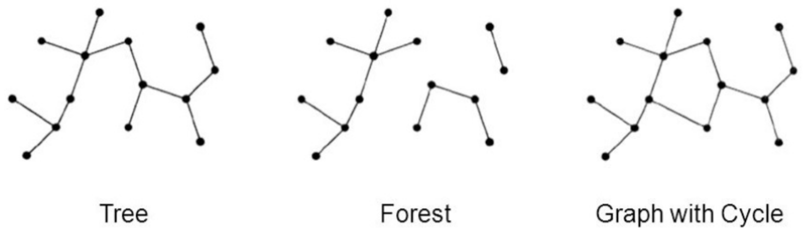


FIGURE 2.12: Examples of a tree a forest and a graph with cycles

We will explain the sampling procedure on a tree using an example (in the forest case all the disconnected trees are independent from each other and they can be sampled independently following the procedure)

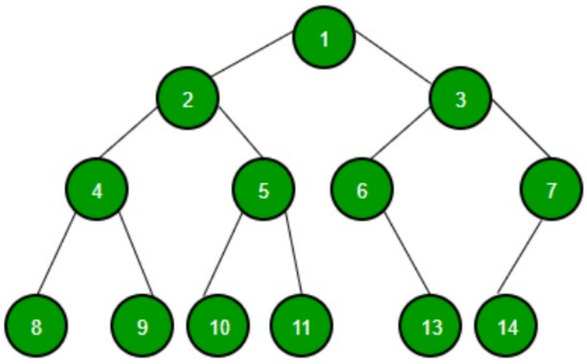


FIGURE 2.13: Example tree for the explanation

For the sampling of an artificial sequence we start deciding arbitrarily a root for the interaction graph (in this case site 1). We draw the type of the nucleotide on this site from $f_1(a_i)$. Supposing we got nucleotide U our artificial chain has Uracil on site 1. Then we move to the neighbours of the root (so sites 2 and 3 in the example). We draw the nucleotide on these from the two point frequencies conditioned to the nucleotide drawn on site 1 $\frac{f_{1,3}(a_1,a_3)}{f_1(a_1)}$. We draw the nucleotide on site 2 from $\frac{f_{1,2}(U,a_2)}{f_1(U)}$ and

the one on site 3 from $\frac{f_{1,3}(U,a_3)}{f_1(U)}$. We repeat this procedure descending all the branches of the tree till we complete the artificial sequence.
This procedure is really efficient since the generation of an artificial sequence becomes as hard as the generation of L random numbers.
If we could generate good artificial sequences using an interaction tree model we would have also several other benefits in addition to the already discussed simplicity of sampling. A model of this type has $L-1$ edges so it falls on the lower range of complexity for Potts models (for instance a fully connected model has $\frac{L(L-1)}{2}$ edges). A relatively simple model has the advantage of being more interpretable from a physical point of view and it also reduce the the risk of overfitting [18].

To test how good a Potts model can get without interaction loops we have to answer this question:

What is the best interaction tree that can describe our natural data?

Essentially we are trying to find the interaction tree for which the formula 2.17 maximizes the likelihood of the natural database

$$E^* = \underset{E}{argmax}\left\{\mathcal{L}\left(DATA|P = \prod_{i\in V} f_i(a_i) \prod_{E\in trees} \frac{f_{ij}(a_i,a_j)}{f_i(a_i)f_j(a_j)}\right)\right\} \tag{2.21}$$

This kind of problem is called "inverse problem". This particular case is widely treated in literature and the solution is finding the maximal spanning tree that maximizes the sum of the mutual information of the sites connected by the edges [5]. The model obtained is called "tree model" and it is the first original model treated in this thesis.

For the values of the $h_i(a_i)$ and the $J_{ij}(a_i,a_j)$ equations 2.19 and 2.20 still holds.

Here are the results of the tests done on the RF00010 RNA family of the Rfam database:

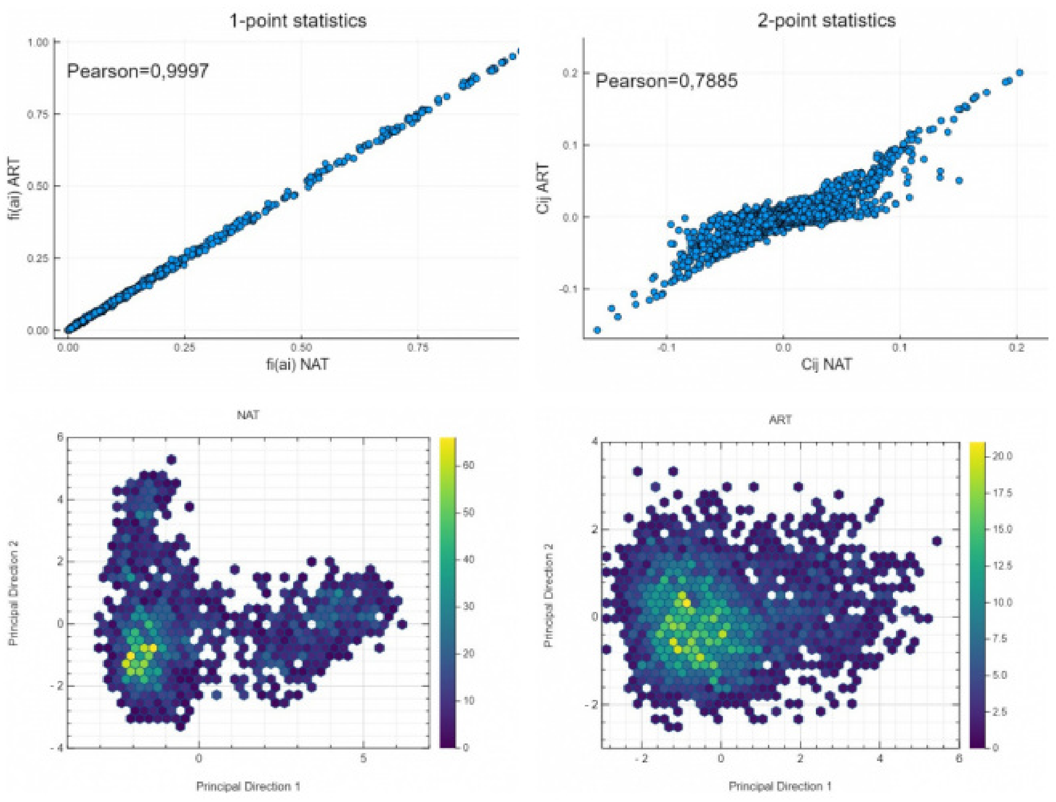


FIGURE 2.14: One point frequencies, two point correlation and PCA projection

One point frequencies are correctly reproduced, two point correlation and PCA projection are a failure.

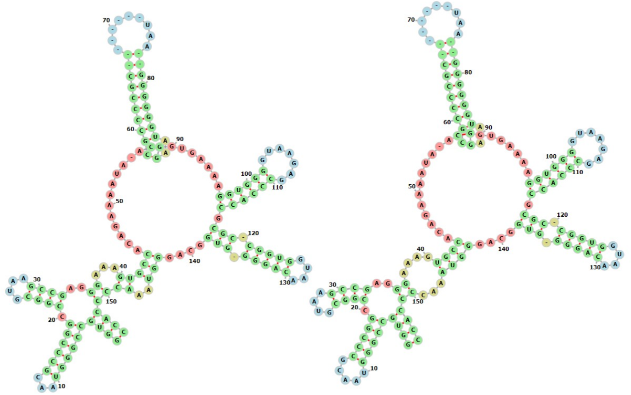


FIGURE 2.15: Right: RNAalifold prediction for natural sequences, Left: RNAalifold prediction for artificial sequences

Artificial sequences do not have the right secondary structure although it is similar. RNAalifold need ~ 5 natural sequences to consistently predict the consensus structure for NAT database. For the generated sequences it needs ~ 7.

The failure of this models means that interaction cycles are necessary in E in order to have a valid generative model.

2.4 Boltzmann Machine Direct Coupling Analysis

2.4.1 Introduction

The bmDCA (Boltzmann Machine Direct Coupling Analysis) [18] algorithm is one of the generative models successfully applied to proteins. It is capable of generating functional artificial sequences [17] and the theoretical apparatus built around it has been implemented to improve folding algorithm and to predict the effects of mutations [11].
In this thesis we did not test bmDCA on RNA but we will mention it anyway because it is the "father" of the application of Potts generative models to bio-molecules. It is based on the Maximum Entropy Principle.

2.4.2 Maximum Entropy Principle

A classical problem in information theory is to try to acquire information on an unknown probability distribution $P(\vec{x})$ by looking at a set of data-points sampled from it (observation data-set or observations). Depending on how large is the number of observation compared to the phase space there are different techniques to address this issue. When the amount of observation is limited a possible strategy is to apply the Maximum Entropy Principle. This principle consists in deciding some features of the observation data-set and finding the "least constrained" probability distribution $P_{MEP}(\vec{x})$ that reproduce the selected features. Usually those features are mean value of observables. The idea is that we do not want $P_{MEP}(\vec{x})$ to fit the noise in the observations so we impose our approximation to be minimally constrained (that means gaining the least amount of information possible from the observations).

The procedure is the following:

Suppose we have N observations:

$$\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N \in X \text{ observed data-points} \tag{2.22}$$

We select M observables $O_\alpha(\vec{x})$ and compute their average on the observations:

$$\widetilde{O_\alpha} = \frac{\sum_{k=1}^N O_\alpha(\vec{x}_k)}{N} \tag{2.23}$$

$$\alpha = 1, 2, \dots, M$$

A possible observable could be $O_\alpha(\vec{x}_k) = x_k^i$ in this case $\widetilde{O_\alpha} = \frac{\sum_{k=1}^N x_k^i}{N}$ corresponds to the mean value of the i^{th} component of the observations.
We have to find the probability distribution $P_{MEP}(\vec{X})$ with maximal entropy that respects:

$$< O_\alpha(\vec{x}) >_{P_{MEP}} = \widetilde{O_\alpha} \quad \text{for } \alpha = 1, 2, \dots, M \tag{2.24}$$

This means we are matching average on observations and ensemble mean value of the selected observables.

To do so we can use Lagrange multiplier technique:

$$P_{MEP}(\vec{x}) \;\propto\; argmax \;\; \left\{ -\sum_{\vec{x}} P(\vec{x}) log(P(\vec{X})) - \sum_{\alpha=1}^{M} \lambda_\alpha \left(\sum_{\vec{x}} P(\vec{x}) O_\alpha(\vec{x}) - \widetilde{O_\alpha} \right) \right\} \tag{2.25}$$

The first term represents the entropy we are trying to maximize (the less constrained is a probability distribution the higher is its entropy). The second term represent the Lagrange multiplier constrains for the M chosen observables. The solution is:

$$P_{MEP}(\vec{x}) \;\propto\; exp\left\{ \sum_{\alpha=1}^{M} \lambda_\alpha O_\alpha(\vec{x}) \right\} \tag{2.26}$$

The values of the λ_α have to be tuned according to equation 2.24.
How good $P_{MEP}(\vec{x})$ approximate the real $P(\vec{x})$ depends on the number/quality of the observations and on the choice of observables.
Once we get $P_{MEP}(\vec{x})$ we can use it to sample artificial sequences. This kind of generative models are called "maximum entropy generative models".

2.4.3 bmDCA generative model

The bmDCA generative model is a maximum entropy generative model used to generate artificial biomolecules [18]. We want to replicate the conservation and the co-evolution statistics. We want our model to replicate the single site frequencies $f_i(a)$ and the two sites frequencies $f_{ij}(a,b)$ of the training natural data (observations). We can write:

$$f_i(a) = \frac{\sum_{k=1}^{N} \delta(a_i^k, a)}{N} \tag{2.27}$$

Where $\delta(a_i^k, a)$ is 1 if the nucleotide on the i^{th} site of the k^{th} sequence in the training set is of type a. It is 0 otherwise. Similarly we can write:

$$f_{ij}(a,b) = \frac{\sum_{k=1}^{N} \delta(a_i^k, a)\delta(a_j^k, b)}{N} \tag{2.28}$$

From 2.23 and 2.24 the observables of which we want to match average on training data and mean value on P_{MEP} are $\delta(a_i, a)$ and $\delta(a_i, a)\delta(a_j, b)$.

Explicitating the mean values we obtain:

$$< \delta(a_i, a) >_{P_{MEP}} = \sum_{a_1, a_2, \ldots, a_L} P_{MEP}(a_1, a_2, \ldots, a_L)\delta(a_i, a) = P^i_{MEP}(a) \tag{2.29}$$

$$< \delta(a_i, a)\delta(a_j, b) >_{P_{MEP}} = \sum_{a_1, a_2, \ldots, a_L} P_{MEP}(a_1, a_2, \ldots, a_L)\delta(a_i, a)\delta(a_j, b) = P^{ij}_{MEP}(a, b)$$

So the aim is to find the probability distribution with maximal entropy for which the one and two point marginals respect

$$P^i_{MEP}(a_i) = f_i(a_i) \tag{2.30}$$

$$P^{ij}_{MEP}(a_i, a_j) = f_{ij}(a_i, a_j)$$

Using equation 2.26 we can infer the form of P_{MEP} that is:

$$P_{MEP}(a_1, a_2, \dots, a_l) \propto exp\Big\{ \sum_i^L \sum_{a=1}^q h_i(a)\delta(a_i, a) + \sum_{i<j} \sum_{a=1}^q \sum_{b=1}^q J_{ij}(a,b)\delta(a_i,a)\delta(a_j,b) \Big\} \tag{2.31}$$

Where $q = 1, \dots, 21$ in the case of proteins (20 aminoacids and one gap) and $q = 1, \dots, 5$ in the case of RNA.
We can rewrite 2.31 in the more compact form (summing over a and b):

$$P_{MEP}(a_1, a_2, \dots, a_l) = \frac{1}{Z} exp\Big\{ \sum_{i=1}^L h_i(a_i) + \sum_{i<j} J_{ij}(a_i, a_j) \Big\} \tag{2.32}$$

$$Z = \sum_{a_1, a_2, \dots, a_L} exp\Big\{ \sum_{i=1}^L h_i(a_i) + \sum_{i<j} J_{ij}(a_i, a_j) \Big\}$$

It becomes evident that bmDCA is equivalent to a Potts model. Specifically a fully connected Potts model with Hamiltonian:

$$H(a_1, a_2, \dots, a_L) = -\sum_{i=1}^L h_i(a_i) - \sum_{i<j} J_{ij}(a_i, a_j) \tag{2.33}$$

$$P_{MEP}(a_1, a_2, \dots, a_L) = e^{-H}$$

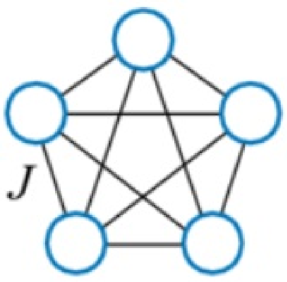


FIGURE 2.16: In a fully connected Potts model all possible pairs of sites are connected by and edge

Now that we know what is the functional form of the probability distribution we have to tune all the $h_i(a_i)$ and $J_{ij}(a_i, a_j)$ according to equation 2.30. The exact computation of these quantities implies a sum over all the sequence space. The number of configurations is q^L and the summation becomes rapidly intractable at the increasing of L.
We can take an approximated approach:

We initialize all the parameters to random values $J_{ij}^0(a_i, a_j)$ $h_i^0(a_i)$ and do the following procedure:

$$J_{ij}^{t+1}(a_i, a_j) = J_{ij}^t(a_i, a_j) + \nu\Big\{ f_{ij}(a_i, a_j) - P_{ij}^t(a_i, a_j) \Big\} \tag{2.34}$$

$$h_i^{t+1}(a_i) = h_i^t(a_i) + \nu\Big\{ f_i(a_i) - P_i^t(a_i) \Big\}$$

Where $P^t(a_1, a_2, \ldots, a_L)$ is 2.32 with parameters $J^t_{ij}(a_i, a_j)$ $h^t_i(a_i)$ and P^t_{ij} $P^t_i(a_i)$ are its one and two point marginals. This procedure comes from likelihood maximization [18] but has an intuitive explanation: the idea is that the higher is a parameter the higher is its associated marginal. At each step we increase the parameter associated with marginals that are smaller than empirical frequencies and decrease the one which are higher. The steady state of the algorithm is reached when condition 2.30 is satisfied.
This approach still does not solve the scaling problem since the marginals in 2.34 require the summation on all the sequence space. The solution is to to sample sequences from the $P^t(a_1, a_2, \ldots, a_L)$ and use the one and two point frequencies on the sampled data-set as an approximation of P^t_{ij} and $P^t_i(a_i)$. The set of interaction pairs E now contains loop (fully connected) and we can not use the fast sampling technique described before. In any case is still possible to sample using Monte Carlo [2] techniques .Since we have to do a Monte Carlo sampling for each step of the 2.34, the algorithm becomes slow (compared to previous model).

Once we are satisfied with the model obtained we can take the Monte Carlo sampled sequences as our artificially generated data.

2.5 Edge addition algorithm

All the models that we studied are based on the Potts model (except the Autoregressive model). The only difference between them is the set E of the interacting pairs. We go from the $E = \emptyset$ in the local field model (least complex Potts model possible) to the bmDCA which contains all the possible edges (most complex Potts model possible).
The main goal of our original work in this thesis is to find the least complex Potts model able to generate valid artificial sequences.
The motivation of our research lies on the answer of this question:

If we have two generative models that perform comparably, which one should we chose?

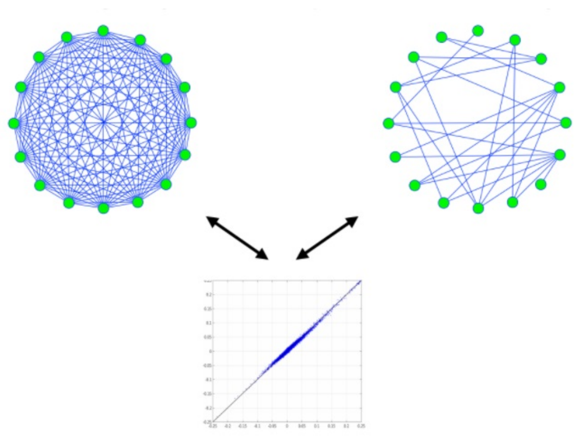


FIGURE 2.17: If those two models are statistically indistinguishable, which one should we chose?

The answer is the less complex one. [18]
There are several reasons for this:

- A less complex model has the advantage of being more interpretable. Interpretable parameters can be used to improve or build folding/mutation algorithm and deepen our knowledge on RNA biophysics.
- In ML, when two model perform comparably, the less complex one is always preferred to prevent overfitting. This is done to make the model less data hungry.

In generative models overfitting usually happens in the form of data-copying. We have to specify that in the case of Potts models applied to bio-molecules even the most complex one (that is the fully connected bmDCA) seems to not fall under this issue. Nevertheless it is better to have relatively simpler generative model especially in the case of smaller data-sets.

An approach present in literature is the "interaction decimation" of the bmDCA [18]. Simplifying, this technique starts from a fully connected model and starts to remove edges with the weakest interactions (and re-learning the remaining ones) gradually decreasing the number of interaction pairs contained in E.
The Edge Addition Algorithm is the inverse approach to the issue. We start from a local field model ($E = \emptyset$) and gradually add edges till we reach a model that we deem as valid.

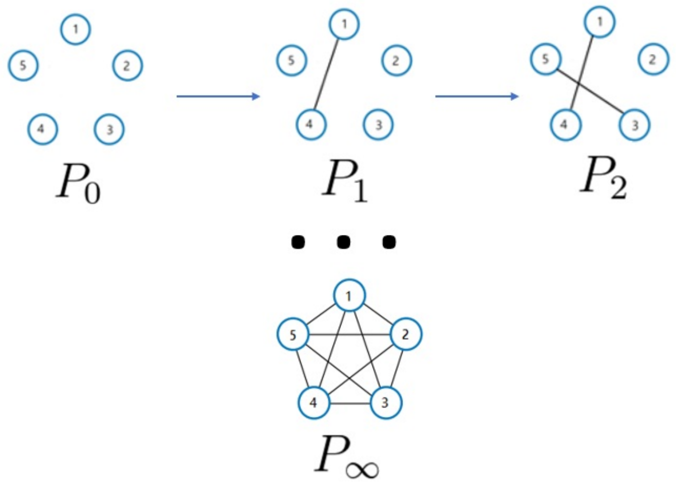


FIGURE 2.18: At each iteration of the edge addition algorithm we get a more complex model, eventually we arrive to a fully connected model that we will show to be equivalent to bm-DCA

This algorithm has the advantage that is analytically possible to find, at each iteration, which edge is better to activate/update in order to maximize the likelihood of the natural data-set.

Suppose we start from an Hamiltonian

$$H^t(a_1, a_2, \dots, a_L) = - \sum_{(m,n)\in E} J_{mn}(a_m, a_n) \tag{2.35}$$

$$P^t(\vec{a}) = \frac{e^{-H^t(\vec{a})}}{Z_t}$$

We do not consider local fields since they are never updated by the algorithm, this corresponds to a possible Gauge choice.
We want to find another Hamiltonian in the form

$$H^{t+1}(\vec{a}) = H^t(\vec{a}) - J_{ij}(a_i, a_j) \tag{2.36}$$

It is important to notice that $J_{ij}(a_i, a_j)$ is not necessarily a new interaction but it can also be an interaction-update if edge (i, j) is already in E.

The quantity $J_{ij}(a, b)$ is a $q \times q$ matrix that has to be optimized over *i,j* and all its 25 values in order to maximize the gain in likelihood of $P^{t+1}(\vec{a})$ respect to $P^t(\vec{a})$. It is important to notice that J_{ij} is not necessarily a new interaction but it can also be an interaction-update

$$l_t = \frac{1}{N} log \left\{ \prod_{\vec{a_k} \ in \ \text{Data}} \frac{e^{-H^t(\vec{a_k})}}{Z_t} \right\} \tag{2.37}$$

$$l_t = \frac{1}{N} \sum_{a_k \ in \ \text{Data}} \sum_{(m,n)\in E} J_{mn}(a^k_m, a^k_n) - log(Z_t) \tag{2.38}$$

Exploiting the summation on the natural data we have

$$l_t = \sum_{(m,n)\in E} \sum_{a=1}^{q} \sum_{b=1}^{q} J_{mn}(a, b) f_{mn}(a, b) - log(Z_t) \tag{2.39}$$

From 2.37 we get that:

$$l_{t+1} = \sum_{(m,n)\in E} \sum_{a=1}^{q} \sum_{b=1}^{q} J_{mn}(a, b) f_{mn}(a, b) + \sum_{a=1}^{q} \sum_{b=1}^{q} J_{ij}(a, b) f_{ij}(a, b) - log(Z_{t+1}) \tag{2.40}$$

So Δl is:

$$\Delta l = \sum_{a,b} J_{ij}(a, b) f_{ij}(a, b) - log\left(\frac{Z_{t+1}}{Z_t}\right) \tag{2.41}$$

For the term $\frac{Z_{t+1}}{Z_t}$ can be written as:

$$\frac{Z_{t+1}}{Z_t} = \frac{\sum_{a_1,a_2,\dots,a_L} e^{-H_t(a_1,a_2,\dots,a_L)} \times e^{J_{ij}(a_i,a_j)}}{\sum_{a_1,a_2,\dots,a_L} e^{-H_t(a_1,a_2,\dots,a_L)}} \tag{2.42}$$

$$\frac{Z_{t+1}}{Z_t} =< e^{J_{ij}(a_i,a_j)} >_{P^t(\vec{a})} \tag{2.43}$$

$$\frac{Z_{t+1}}{Z_t} = \sum_{a,b} e^{J_{ij}(a,b)} P^t_{ij}(a, b) \tag{2.44}$$

Equation 2.41 can be witten as:

$$\Delta l = \sum_{a,b} J_{ij}(a,b) f_{ij}(a,b) - log\bigg(\sum_{a,b} e^{J_{ij}(a,b)} P^t_{ij}(a,b) \bigg) \tag{2.45}$$

When optimizing Δl over $J_{ij}(a,b)$ we get:

$$0 = f_{ij}(a,b) - \frac{e^{J_{ij}(a,b)} P^t_{ij}(a,b)}{\sum_{a',b'} e^{J_{ij}(a',b')} P^t_{ij}(a',b')} \tag{2.46}$$

This equation is solved by:

$$J_{ij}(a,b) = log\bigg(\frac{f_{ij}(a,b)}{P^t_{ij}(a,b)} \bigg) \tag{2.47}$$

$$\Delta l = \sum_{a,b} f_{ij}(a,b) log\bigg(\frac{f_{ij}(a,b)}{P^t_{ij}(a,b)} \bigg) = D_{kl}(f_{ij}|P^t_{ij})$$

So to have the maximum likelihood gain possible with a single edge addition/update (the selected (i,j) could also already be in E) we have to see for what edge we have the highest Kullback Leibler divergence $D_{kl}(f_{ij}|P^t_{ij})$ and add interaction 2.47 to our Hamitlonian H^t.

We can finally describe the iterative procedure of the Edge Addition Algorithm:

- Start with a model $H_t(a_1, a_2, \dots, a_L)$
- Find all the marginals $P^t_{ij}(a,b)$
- Find $(i,j) = argmax\ \ D_{kl}(f_{ij}|P^t_{ij})$
- Obtain a new model $H^{t+1}(\vec{a}) = H_t(\vec{a}) - J_{ij}(a_i, a_j)$ where $J_{ij}(a,b)$ are obtained from 2.47
- Iterate

Iteration after iteration we add (or update) new edges and obtain generative models at increasing order of complexity. Looking at 2.47 it is evident that the steady state model of the algorithm is equivalent to bmDCA.The procedure stops only when $P^t_{ij}(a,b) = f_{ij}(a,b)$ for all (i,j), exactly the condition that we impose for bmDCA.

Now that we have the algorithm we should answer two main questions:

1. How do we find marginals P^t_{ij}?
2. When do we stop adding edges?

Addressing the first question: we cannot hope to have a good generative model without interaction loops in E. We have already tried the best possible tree model and it failed. Because of this we are sure that the Edge Addition algorithm can not reach a good generative model without creating loops in E. This rules out the possibility to use the fast sampling described before. To estimate $P^t_{ij}(a,b)$ we have to use Monte Carlo techniques (Metropolis, Gibbs) to sample artificial sequences and use their one and two point frequencies as approximation for P_i^t and P_{ij}^t. This slows significantly the algorithm because you have to wait thermalization at each MCMC step. H^t and H^{t+1} differs only by an interaction edge so we can use sequences sampled from P^t as a starting point for the Mont Ccarlo run to estimate the marginals of P^{t+1}. We will see that this significantly reduces thermalization time ad accelerate the algorithm.

Addressing the second question: after a bunch of iteration we end with a set of generative models from which we have to choose the "best" one.

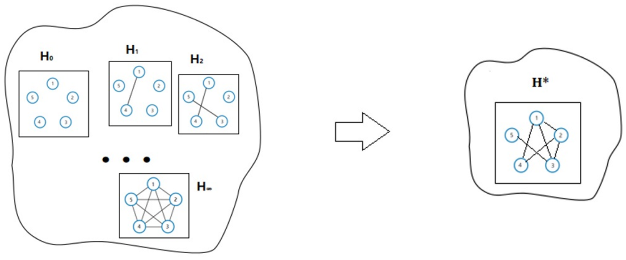


FIGURE 2.19: For each step of the EAA we have a new model. How we can select the right model to generate artificial sequences? [18]

We investigated two criterion for the selection: the first is a Machine Learning like train/test score. As score we used the Pearson correlation between the $C_{ij}(a_i, a_j)$ between the natural and artificial data-set. This choice is motivated by the fact that the algorithm steady state is equivalent to the bmDCA which aims to a perfect reproduction of this statistics. The second method is the Bayesian Information Criterion. The BIC consists in assigning a score Y to each of our model with a positive term for the likelihood of the training data-set and a penalty for the complexity [3].

$$Y = 2log(\mathcal{L}) - log(N_{eff})K \tag{2.48}$$

Where $\mathcal{L}$ is the likelihood of the training data-set and K is the number of parameters of the model. This quantity can be estimated at each step using eq. 2.47 but the estimation results too noisy. To solve this issue we used the pseudo-likelihood approximation [15].

We tested the algorithm on the RF00010 RNA family of the Rfam database. For the model selection we implemented the two criterion described before:
We see that both training and test scores have saturating behaviours. Since we are not gaining anything in performance we should select the model at the beginning of the saturation at around iteration 300. This is done to not add unnecessary complexity to our generative model. BIC score has a peak around the same iteration. The two selection criterion are in good agreement. The selected model has around 5% of the complexity of bmDCA.

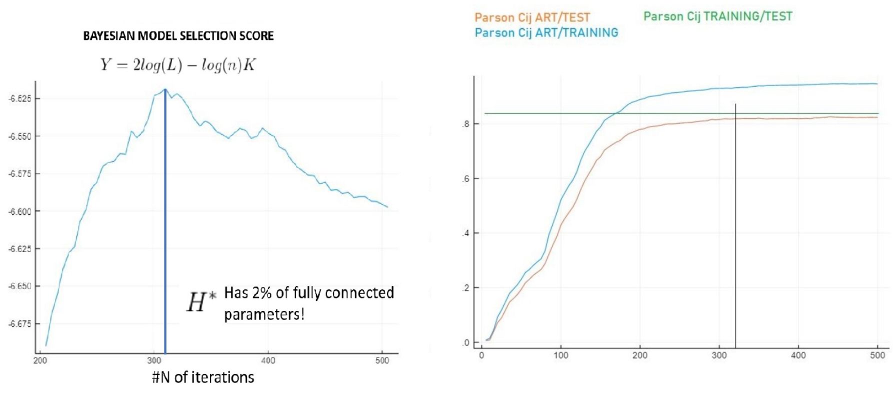


FIGURE 2.20: BIC score (left), ML train/test score (right)

Here are the test of the selected model:

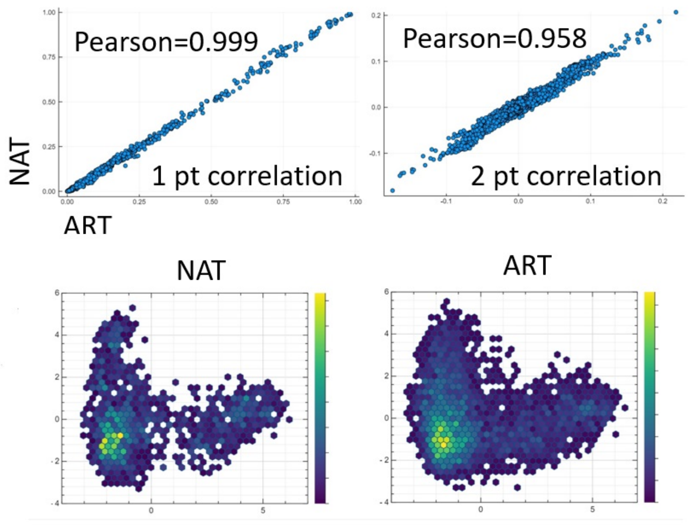


FIGURE 2.21: Conservation and coevolution statistics are well represented. Looking at PCA projection we see that the model is not data copying

For the secondary structure we see that RNAalifold predicts the same structure for natural and artificial sequences.

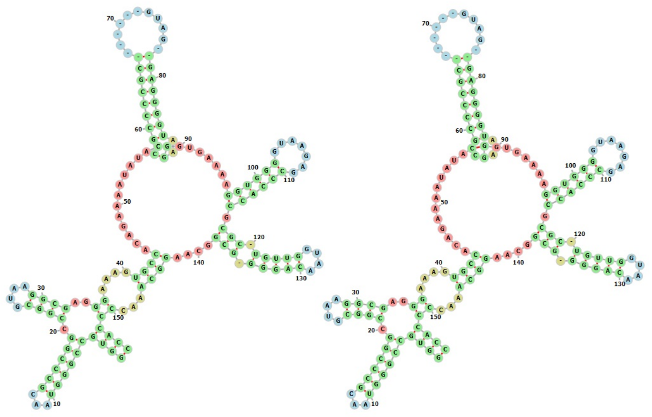


FIGURE 2.22: RNAalifold predicted structure for: natural sequences (left) artificial sequences (right)

The only difference is that RNAalifold needs only ~ 5 natural sequences on average to come up with the consensus secondary structure while it needs ~ 7 for artificial ones. This could be caused by the the failing of the algorithm to generate valid sequences in certain regions of the sequence space.

The results are promising. We were able to reduce the complexity to 5% of bmDCA while obtaining comparable results. (comparing with bmDCA performance on proteins).

2.6 Autoregressive DCA

Autoregressive DCA is the last generative model studied in this thesis. Unlike the previous models this is not based on a Potts model architecture. This model has already been applied to proteins and during the thesis work we adapted and tested the algorithm with RNA.
This model is based on the following exact decomposition of the probability of a sequence (Bayes theorem):

$$P(a_1, a_2, \dots, a_L) = P(a_1)P(a_2|a_1)P(a_3|a_2, a_1)\dots P(a_L|a_{L-1}, \dots, a_2, a_1) \tag{2.49}$$

This decomposes the probability on a product of L single sites conditional probabilities. The main idea of the model is to give an approximate form for each of those factors $P(a_i|a_{i-1}, a_{i-2}, \dots, a_1)$ and then sample artificial sequences site by site from them.
This approach is very interesting since it transforms the generation of an artificial sequence from an unsupervised learning task in a self-supervised learning one. Deciding the nucleotide on site i of the chain knowing all the other in positions $i-1, i-2, \dots, 2, 1$ is equivalent to assign a label (with values $1, 2, \dots, q$) to the vector $(a_1, a_2, \dots, a_{i-1})$. This is a great advantage since the supervised learning branch of machine learning is much more developed than the unsupervised one.
The chosen parametrization is a popular tool of supervised learning: the softmax regression [6]. It is a generalization to multi-class labels of the logistic regression.

$$P(a_i|a_{i-1}, a_{i-2}, \dots, a_2, a_1) = \frac{e^{h_i(a_i)+\sum_{j<i} J_{ij}(a_i,a_j)}}{z_i(a_1, a_2, \dots, a_{i-1})} \tag{2.50}$$

$$z_i(a_1, a_2, \dots, a_{i-1}) = \sum_{a_i} e^{h_i(a_i)+\sum_{j<i} J_{ij}(a_i,a_j)} \tag{2.51}$$

Where z_i is the normalization factor for each of the conditional probabilities.
The model parameters are $h_i(a_i)$ and $J_{ij}(a_i, a_j)$.

Equation 2.50 has several similarities with bmDCA. The number of parameters is the same and they are also in the same form (local field and pairwise interactions) . The difference is that while in the Potts generative models J_{ij} is a symmetric interaction in the Autoregressive it is directed and represent the influence on site i of all previous sites j. In fact in bmDCA $J_{ij} = J_{ji}$ instead in this case $J_{ji} = 0$ if $i > j$.

A great advantage of this model is that each of the eq. 2.50 marginals is normalized. This means that the whole probability distribution $P(a_1, a_2, \dots, a_L)$ is normalized too. In Potts models instead we can only have *weights* for the sequences since the calculation of the normalization factor Z requires a summation on all the sequence space and is prohibitive. When we are working within an RNA family having normalized probabilities has not many advantages over having not-normalized weights. The situation is different when we want to do infra-families studies. A common issue is the family assignment problem that is when we want to decide whether a certain RNA or Protein molecules belongs to a family or another. Comparing the weights of the molecule for two different families is not possible since the weights could have different scales. If we have normalized probabilities a comparison is possible and we can assign the molecule to the family for which it has the highest probability.

In eq. 2.49 we did the decomposition following the site order of the molecule chain. However this is not necessary and the decomposition can be done with any of the possible $L!$ ordered permutation of the the sites. Each order leads to a different performance for the generative model so we not only have to do an optimization over the parameters $h_i(a_i)$ and $J_{ij}(a_i, a_j)$ but also on the order of the factorization in 2.49.

The optimization of the parameters does not pose a great problem and is based on likelihood maximization of the natural data-set, a classical problem in Machine Learning [6].
The optimization over all the orders is more complicated. It's not possible to compare the performance of the model for all the $L!$ ordering. For instance the number of permutation for a family with $\sim$ 150 nucleotides is $\sim 10^{263}$, way more than the extimated number of atoms in the known universe ($\sim 10^{81}$).
The entropic ordering (from the least variable to the most variable site) is alluring for its interpretability: we start from the most conserved sites because if we draw a low probability nucleotide type on them this has a great impact on the chain and the model has to be able to compensate adjusting the nucleotide on the less conserved sites. In addition to being easily interpretable this approach also has a good performance, the quality of the generated sequences is comparable with the bmDCA.

The last issue we have to address before testing is the sampling. It can be easily done with the following procedure:

- Sample the nucleotide on the first site from $P(a_1)$
- Sample the nucleotide on the second site from $P(a_2|a_1)$ where a_1 assumes the value sampled in the previous step
- ....Sample the last nucleotide from $P(a_L|a_{L-1}, \ldots, a_1)$ where $a_1, \ldots, a_L$ assume the value sampled in the previous steps

It is important to notice that now the index i in a_i does not indicate the position of the site on the chain but indicate its position in the entropic ordering.
Each step is very fast because there are only 5 possible values for each probability. On the contrary, sampling from the bmDCA model is slower because it requires several MCMC runs and we have to wait thermalization for all of them.

Here are the results of the tests done on the RF00010 RNA family of the Rfam-database:

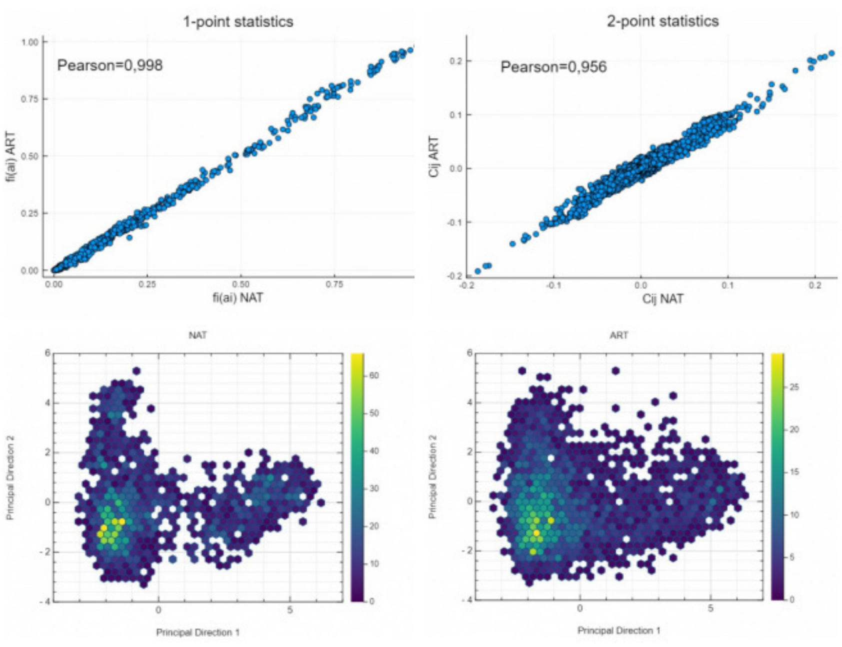


FIGURE 2.23: Conservation and coevolution statistics are well represented. Looking at PCA projection we see that the model is not data copying

For the secondary structure we see that RNAalifold predicts the same structure for natural and artificial sequences.

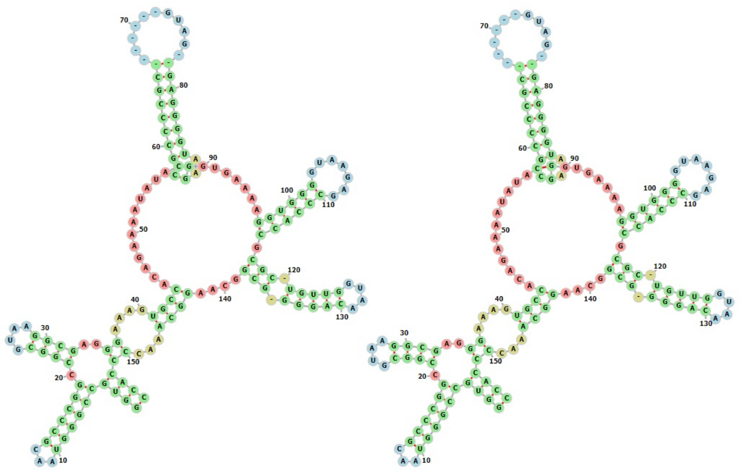


FIGURE 2.24: RNAalifold predicted structure for: natural sequences (left) artificial sequences (right)

We have the same issue as before. RNAalifold needs only ~ 5 natural sequences on average to come up with the consensus secondary structure while it needs ~ 7 for artificial ones.

We are very happy with the results of the model. The performance is comparable with bmDCA (performance on proteins) but the sampling is much faster and we have normalized probabilities.

Chapter 3

Secondary Structure prediction

3.1 Introduction

Secondary structure prediction algorithms vary widely. The most sophisticated ones are based on Free Energy minimization. The major limit of these models is that they need precise experimental estimate of the magnitude of interaction between base pairs and between sequence domains with the environment. Often this data is not precise enough to get a realistic secondary structure.
There is much that can be gleaned from a statistical approach. sequences in the same family can be very different from each other but since they have the same biological function they also have very similar structures (as discussed before there is a deep connection between the two) [19]. Simply speaking if two sites are very correlated (coevolution) they are more likely to form a bond in the secondary structure than two sites with low correlation.

3.2 RNAalifold

The RNAalifold secondary structure prediction algorithm combines the Free Energy minimization approach with the statistical one [4]. It is part of the Vienna Package that is a set of programs to study RNA. It predicts consensus structure for RNA families and it is one of the most reliable algorithm currently available. Throughout the following paragraphs the predictions done with RNAalifold will be taken as a benchmark to judge the effectiveness of other algorithms.

3.3 Nussinov

The Nussinov algorithms uses dynamic programming principles to find the secondary structure of an RNA molecule [13]. Originally it was a single-sequence algorithm but it can be easily improved with statistical information contained in the multi-sequence alignment of a RNA family [9].
The algorithm starts with the assignment of a score S_{ij} for all site pairs, here are two examples:

$$\begin{cases} S_{ij} = 1 & \text{if } (i,j) \text{ complementary} \\ S_{ij} = 0 & \text{otherwise} \end{cases} \qquad \begin{cases} S_{ij} = 3 & \text{if } (i,j) = \text{G-C} \\ S_{ij} = 3 & \text{if } (i,j) = \text{A-U} \\ S_{ij} = 1 & \text{if } (i,j) = \text{G-U} \\ S_{ij} = 0 & \text{otherwise} \end{cases}$$

The first one takes only in account the possibility for the pair to bond, the second also takes in account the strength of the contact (G-C forms three hydrogen bonds, A-U forms two, G-U forms wobble bond).
The next step is to find the nested configuration of contacts that maximizes the total score

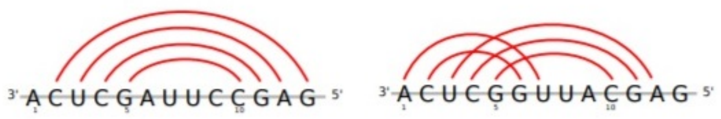


FIGURE 3.1: Nested configuration are like the one on the left, we do not want configurations like the one on the right because they lead to the formation of pseudoknots, highly unstable structures

We indicate with $\gamma(i,j)$ (for $j > i$) the score associated to the optimal configuration of an hypothetical chain that starts with site i and ends with site j. This quantity can be easily found iteratively as follows:

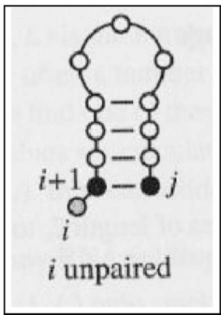
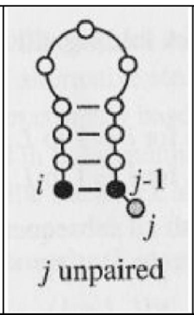
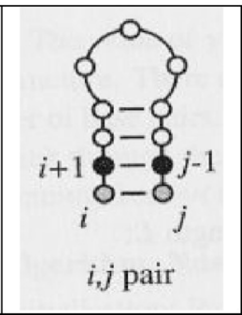
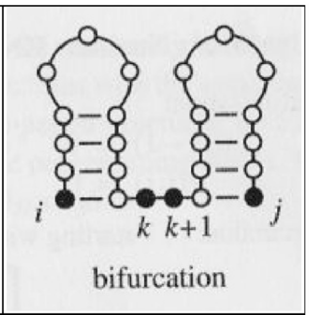
			
$\gamma(i+1,j)$	$\gamma(i,j-1)$	$\gamma(i+1,j-1)+\delta(i,j)$	$\max_{i<k<j}[\gamma(i,k)+\gamma(k+1,j)]$

FIGURE 3.2: $\gamma(i,j)$ is the max out of the four possible cases of the picture

with initialization $\gamma(i,i) = 0$. At this point we have not only calculated the score of the optimal configuration for the whole chain $\gamma(1,N)$ but also the one for all the possible "sub-chains" $\gamma(i,j)$. Once we have all the $\gamma(i,j)$ Nussinov algorithm makes use of a traceback procedure to get all the contacts present in the score maximizing configuration.
The Nussinov algorithm works with single sequences so it does not take advantage of the huge amount of statistical information contained in RNA families. In addition it maximizes contact number/energy but does not take in account the energy contribution of the various molecule domains and their interaction with the environment. As we can see if we try to predict the secondary structure of several RNA molecules belonging to the same family we get very diverse results instead of getting similar structures:

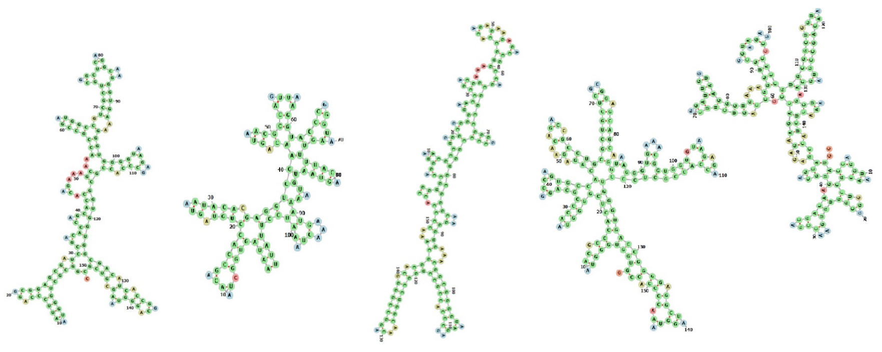


FIGURE 3.3: Nussinov prediction on 4 sequences of the RF00010 RNA family of the Rfm database (3-2-1 scoring system)

Even tough the predictions are pretty inconsistent the "architecture" of the algorithm is very good and can be improved adding statistical information to the mix [9]. This is done changing the scoring system.
We can compute the Mutual Information MI_{ij} between all site pairs in the multiple sequence alignment and use this as contact score

$$S_{ij} = MI_{ij} = \sum_{A_i,A_j} f_{ij}(A_i,A_j) log \frac{f_{ij}(A_i,A_j)}{f_i(A_i)f_j(A_j)} \tag{3.1}$$

MI between two sites corresponds on the amount of information that I gain on site j (or i) once i know the nucleotide on site i (or j).
If two sites are in contact in the secondary structure they will surely have a big Mutual information since, due to co-evolution, if one of the two changes the other has to change too because they want to conserve the ability to form a bond.
We have to be careful about a couple of things: MI can only be positive so there is the risk that the score maximizing configuration will try to maximize the number of contacts since each one of them gives a positive contribution to the final score, in addition two sites can have a relatively high mutual information even if they are not in contact. For example if they are spatially close in the folded structure of the molecule and their contact would compromise the biological functionality, in this case knowing the nucleotide on one of them we gain a bit of information on the other because we know it can not be the complementary Watson-Crick pair. Usually in this situation the MI is not as high as when they form a contact simply because the options for not being complementary are more than the ones for being complementary.

Both issues can be solved using a threshold scoring:

$$S_{ij} = MI_{ij} \quad \text{if} \quad MI_{ij} > T$$

Many contacts will now have zero score preventing fake contacts. Setting the T sufficiently high will also solve the *not want to be in contact* problem. Looking at the MI_{ij} of the sites in contact for different RNA families with known secondary structure we decided $T = 0.4$. In addition we set an ∞ penalty for neighbouring sites contacts (separated by less than 4 sites). That is because neighbouring sites will often have high mutual information but they rarely form a bond in secondary stucture.

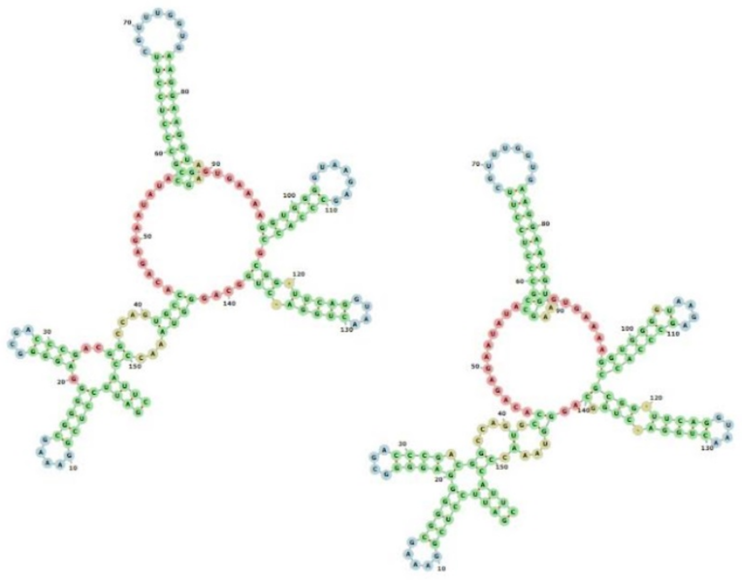


FIGURE 3.4: MI-Nussinov (right) RNAalifold (left) predicted structure are almost identical (RF0010)

3.4 Edge addition algorithm and secondary structure

As we discussed before at each iteration of the Edge Addition Algorithm we add (or update) an edge interaction. After several iterations we not only end up with many possible generative models to choose form but also with an ordered list of edges. Since the starting point of our algorithm is a local filed model the conservation statistics (single site frequencies) is already well represented so each one of the added edge is trying to fix the co-evolution statistics.
It's natural to ask if there is a link between the obtained list of edges and the consensus secondary structure of the RNA family since we know that co-evolutionary effects are strongest between sites that for a contact
Here we analyzed the first 300 added edge for the RF00010 family dividing them in the following categories: contacts (links two sites that are in contact in the RNAalifold predicted structure), neighbours (if they are less than 4 nucleotides distant) and false contacts:

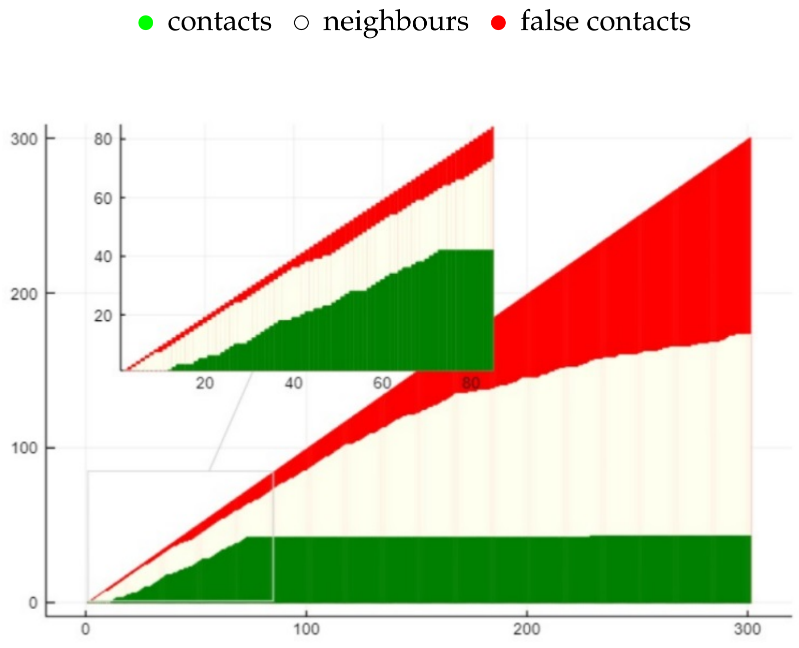
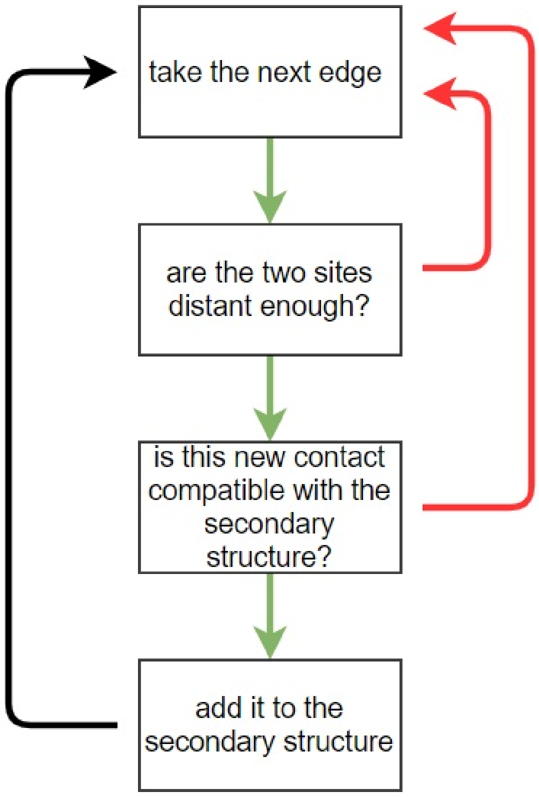


FIGURE 3.5: 43/45 predicted contacts are taken in the first 100 iterations

We can clearly see the hypothesized relationship between contact and added edges. The first iterations take either neighbouring sites or predicted contacts (except for a few false contacts). After around $\sim$ 90 iterations the number of contact taken saturates and the algorithm starts to take more and more neighbouring sites and false contacts.

3.4.1 Toy algorithm for secondary structure

We can think of using this information to build prediction algorithms for secondary structure. A first naive approach is the following algorithm: we start form the Edge Addition Algorithm obtained ordered list of edges and we do as follows:


take the next edge
are the two sites distant enough?
is this new contact compatible with the secondary structure?
add it to the secondary structure

Where checking the compatibility of a contact with the secondary structure means to check if it causes the appearence of any pseudoknots.
Of course we have also to specify a termination condition for the algorithm, taking inspiration from the previous models we set the termination condition when the MI_{ij} of the two sites connected by the added edge is < 0.4
This algorithm is only a starting point (hence the toy model name) but the results are already satisfactory:

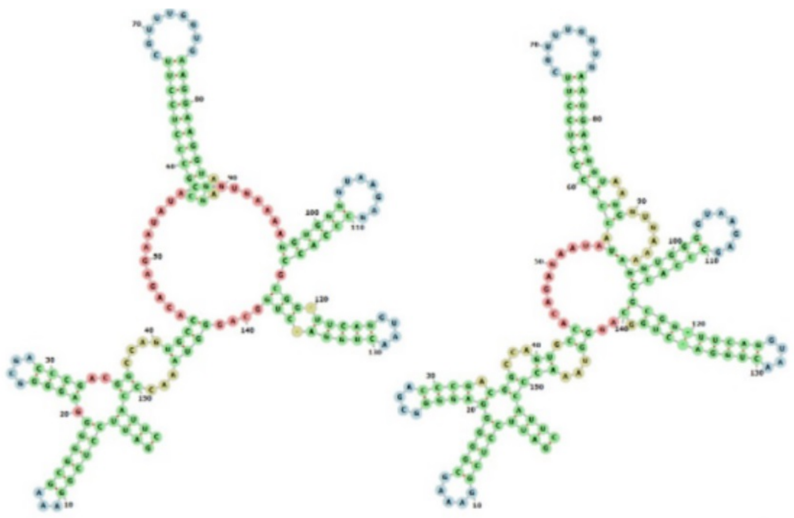


FIGURE 3.6: RNAalifold (right) Toy model (left)

Chapter 4

Conclusion and future research

In this last chapter we will make some comments and give some hints on possible future improvements and applications of the topics covered in this thesis.
First of all, we should address the main question of the thesis:

What generative model should we use to generate artificial sequences?

Of all the generative models treated the one obtained with the Edge Addition Algorithm seems (at least in theory) to be the best compromise: it has an excellent reproduction of natural statistics while having very few parameters.

The next step is to further improve the algorithm.
We could try to improve its efficiency by adding/updating more than an interaction at a time. Instead of modifying only the interaction of the pair with the worst represented $f_{ij}(a_i, a_j)$ we could modify the fist two/three for example.
Another possible improvement is trying to better estimate the $P^t_{ij}(a_i, a_j)$ at each step of the algorithm. The saturating behaviour in the the two-point statistics (evident from Fig. 2.20) descends from the fact that errors in the estimation of the $P^t_{ij}(a_i, a_j)$ start to be comparable to their differences and the algorithm begins to be dominated by noise.

Regarding the secondary structure: the toy model proposed seems to be a promising starting point to develop more complex and precise prediction algorithms. It confirms that the edge list obtained from the EDA contains a lot of information on the secondary structure. A possible improvement could be using it to develop a Nussinov score.

Another important things to do is to extend these studies to a large number of RNA families in order to test the robustness of the generative models and of the techniques developed.

Lastly we will experimentally test the generated sequences to have the final confirmation of the good functioning of the models. In case of affirmative results we can use the information obtained from the experiments to further improve the generative models and to finally generate artificial sequences for practical purposes (that is the true aim of this work).

Bibliography

[1] Dror Baran et al. "Principles for computational design of binding antibodies". In: *Proceedings of the National Academy of Sciences* 114.41 (2017), pp. 10900–10905. ISSN: 0027-8424. DOI: 10.1073/pnas.1707171114. eprint: https://www.pnas.org/content/114/41/10900.full.pdf. URL: https://www.pnas.org/content/114/41/10900.

[2] Adrian Barbu and Song-Chun Zhu. *Monte Carlo Methods*. Springer Singapore, 2020. DOI: 10.1007/978-981-13-2971-5. URL: https://doi.org/10.1007/978-981-13-2971-5.

[3] "Bayesian Information Criteria". In: *Springer Series in Statistics*. Springer New York, 2008, pp. 211–237. DOI: 10.1007/978-0-387-71887-3_9. URL: https://doi.org/10.1007/978-0-387-71887-3_9.

[4] Stephan H. Bernhart et al. "RNAalifold: improved consensus structure prediction for RNA alignments". In: *BMC Bioinformatics* 9.1 (2008), p. 474. ISSN: 1471-2105. DOI: 10.1186/1471-2105-9-474. URL: https://doi.org/10.1186/1471-2105-9-474.

[5] Arnab Bhattacharyya et al. "Near-optimal learning of tree-structured distributions by Chow-Liu". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. ACM, June 2021. DOI: 10.1145/3406325.3451066. URL: https://doi.org/10.1145/3406325.3451066.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[7] James W. Brown et al. "The RNA structure alignment ontology". eng. In: *RNA (New York, N.Y.)* 15.9 (2009). 19622678[pmid], pp. 1623–1631. ISSN: 1469-9001. DOI: 10.1261/rna.1601409. URL: https://pubmed.ncbi.nlm.nih.gov/19622678.

[8] D. Foster. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. O'Reilly Media, 2019. ISBN: 9781492041894. URL: https://books.google.co.za/books?id=RqegDwAAQBAJ.

[9] Eva Freyhult, Vincent Moulton, and Paul Gardner. "Predicting RNA Structure Using Mutual Information". In: *Applied bioinformatics* 4 (2005), pp. 53–9. DOI: 10.2165/00822942-200504010-00006. URL: https://doi.org/10.2165/00822942-200504010-00006.

[10] Sam Griffiths-Jones et al. "Rfam: an RNA family database". eng. In: *Nucleic acids research* 31.1 (2003). 12520045[pmid], pp. 439–441. ISSN: 1362-4962. DOI: 10.1093/nar/gkg006. URL: https://pubmed.ncbi.nlm.nih.gov/12520045.

[11] Thomas Gueudré et al. "Simultaneous identification of specifically interacting paralogs and interprotein contacts by direct coupling analysis". In: *Proceedings of the National Academy of Sciences* 113.43 (2016), pp. 12186–12191. ISSN: 0027-8424. DOI: 10.1073/pnas.1607570113. eprint: https://www.pnas.org/

content/113/43/12186.full.pdf. URL: https://www.pnas.org/content/113/43/12186.

[12] I. T. Jolliffe. "Principal Component Analysis and Factor Analysis". In: *Principal Component Analysis*. Springer New York, 1986, pp. 115–128. DOI: 10.1007/978-1-4757-1904-8_7. URL: https://doi.org/10.1007/978-1-4757-1904-8_7.

[13] R Nussinov and A B Jacobson. "Fast algorithm for predicting the secondary structure of single-stranded RNA". In: *Proceedings of the National Academy of Sciences* 77.11 (1980), pp. 6309–6313. ISSN: 0027-8424. DOI: 10.1073/pnas.77.11.6309. eprint: https://www.pnas.org/content/77/11/6309.full.pdf. URL: https://www.pnas.org/content/77/11/6309.

[14] S. Panzeri, C. Magri, and L. Carraro. "Sampling bias". In: *Scholarpedia* 3.9 (2008). revision #148550, p. 4258. DOI: 10.4249/scholarpedia.4258.

[15] "Pseudo-Likelihood". In: *Models for Discrete Longitudinal Data*. New York, NY: Springer New York, 2005, pp. 189–202. ISBN: 978-0-387-28980-9. DOI: 10.1007/0-387-28980-1_9. URL: https://doi.org/10.1007/0-387-28980-1_9.

[16] Michael P. Robertson and Gerald F. Joyce. "The origins of the RNA world". eng. In: *Cold Spring Harbor perspectives in biology* 4.5 (2012). 20739415[pmid], a003608. ISSN: 1943-0264. DOI: 10.1101/cshperspect.a003608. URL: https://pubmed.ncbi.nlm.nih.gov/20739415.

[17] William P. Russ et al. "An evolution-based model for designing chorismate mutase enzymes". In: *Science* 369.6502 (2020), pp. 440–445. ISSN: 0036-8075. DOI: 10.1126/science.aba3304. eprint: https://science.sciencemag.org/content/369/6502/440.full.pdf. URL: https://science.sciencemag.org/content/369/6502/440.

[18] Kai Shimagaki and Martin Weigt. "Collective-variable selection and generative Hopfield-Potts models for protein-sequence families". In: *bioRxiv* (2019). DOI: 10.1101/652784. eprint: https://www.biorxiv.org/content/early/2019/05/29/652784.full.pdf. URL: https://www.biorxiv.org/content/early/2019/05/29/652784.

[19] Lee E. Vandivier et al. "The Conservation and Function of RNA Secondary Structure in Plants". eng. In: *Annual review of plant biology* 67 (2016). 26865341[pmid], pp. 463–488. ISSN: 1545-2123. DOI: 10.1146/annurev-arplant-043015-111754. URL: https://pubmed.ncbi.nlm.nih.gov/26865341.