Politecnico di Torino

Master Degree in Physics of Complex Systems

Master Degree's Thesis

Attention Based Direct Coupling Analysis for Protein Structure Prediction

Supervisor:
Prof. Andrea PAGNANI

Candidate:
Francesco CAREDDA

July 2022

Abstract

Proteins are at the base of every biological function within the cell, ranging through a variety of transport, signaling and enzymatic tasks. Their functionalities heavily rely on their three-dimensional structure which is extremely difficult, time consuming and expensive to determine. In this thesis we discuss Direct Coupling Analysis (DCA), the state-of-the-art statistical physics model used to learn structural information about co-evolving proteins based on their amino-acid sequence. Phylogenetically related homologous sequences can be considered as belonging to a unique protein family with specific structural properties defining their functionality. For our purposes such sequences, aligned and collected in a data structure called Multiple Sequence Alignment (MSA), can be thought of as samples drawn from a probability distribution encoding the fundamental structural traits of the protein family they belong to. The form of the distribution is obtained by applying a Maximum Entropy Principle imposing as empirical constraints the single and pairwise frequency counts of the amino-acids in the MSA. The resulting probability distribution is a Potts model whose parameters, to be inferred, represent the direct-interaction tensor for any two given residues and the local biases for each position in the sequence. More precisely, DCA represents an inverse Potts problem aimed at inferring the set of parameters which better describes the direct residue-residue interactions for a specific protein family. Among the possible methods that can be used to solve the inference problem, we consider the state-of-the-art architecture for contact-prediction, PlmDCA, a maximum likelihood estimate of the parameters

by means of a gradient-ascent of a pseudo-loglikelihood function depending on the specific MSA. In particular, the purpose of this thesis is to develop a possible improvement of PlmDCA inspired by the Attention Mechanism, a deep learning technique developed in the context of Natural Language Processing. Attentions is gaining popularity in the computational biology community after the recent exploit of AlphaFold 2 by DeepMind which used it in its deep learning architecture for protein structure prediction at the 2020 CASP competition. In this new model, the interaction tensor of the Potts model is written as a non-linear low-rank decomposition whose aim is to share amino-acid features, effectively reproducing the fact that different positions may be in contact due to similar chemical interactions. The validity of the Attention-Based PlmDCA is tested against the standard PlmDCA architecture using three MSA whose structural data are fully available through the Pfam database.

Acknowledgments

I really wish to thank my supervisor Andrea Pagnani for the opportunity of working with him and for his constant help and support. Andrea's guidance, advice and ideas have been fundamental for this thesis.
A big thanks to all the members of the Complex Systems group at Politecnico di Torino. They made me feel welcome in these months of work among them and for that I am truly grateful. A special mention goes to Anna Paola Muntoni for all the crucial advice she gave me and for finding a major bug in my code which really changed the outcome of this thesis!
Finally, I deeply thank all my friends who took the time to read this thesis and helped me correcting mistakes. In particular, to Chiara for being there for me since the very beginning.

Contents

List of Figures

Acronyms

APC Average-Product Correction. 26

BERT Bidirectional Encoder Representation from Transformers. 62

CASP Critical Assessment of Protein Structure Prediction. 2, 11

DCA Direct Coupling Analysis. 1, 13, 20

GPT-3 Generative Pre-Trained Transformer 3. 62

LSTM Long Short-Term Memory. 45

MEP Maximum Entropy Principle. 18, 19

MLE Maximum Likelihood Estimate. 27

MSA Multiple Sequence Alignment. 1, 15, 17, 31, 32

NLG Natural-Language Generation. 38

NLI Natural-Language Interpretation. 38

NLP Natural-Language Processing. 38, 39

PlmDCA Pseudo-Loglikelihood Maximisation DCA. 1, 27, 62, 64

PPI Protein-Protein Interactions. 78

PPV Positive Predictive Value. 33

RNN Recurrent Neural Network. 39, 41

VGP Vanishing Gradient Problem. 44

Chapter 1

Introduction

This introductory chapter is aimed at presenting the fundamental biological aspects at the base of the problem of protein folding and the techniques that have been developed in an attempt to solve it. Furthermore, we discuss the connection between the Protein Folding Problem and the Protein Design Problem, along with the scientific and industrial relevance of the two. Finally, we give an overview of the recent developments in statistical inference models for protein analysis, which are at very the core of this thesis and will be discussed in more details in the following chapters.

1.1 The Protein Folding Problem

The problem of protein folding is among the hardest and most challenging issues in modern biological chemistry, biophysics and statistical learning. Since the first theoretical studies by Linus Pauling [1] and the development of X-ray crystallography, the question of what determines the three-dimensional conformation of a protein eluded researchers for almost 70 years. In order to adequately understand the scale of the problem and the implications of solving it, it is necessary to first describe the chemical constituents of proteins, along with the interactions that play the fundamental role in their composition.

Proteins are a broad class of biological macromolecules involved in virtually every activity within living cells [2]. Out of their many tasks, proteins act as enzymes for metabolic reactions, DNA replication, transcription and translation. They constitute the dynamical scaffolding of the cell in the form of the cytoskeleton, an arrangement of protein filaments capable of extending or contracting depending on the cell's needs. They even serve as molecular motors, converting energy into motion or mechanical work that can be used to move cargo inside the cell or to propel micro-organisms through viscous fluids by means of flagella [3].

From a biochemical point of view, proteins are polymers composed of subunits called amino-acids. An amino-acid is a compound formed of an amino and a carboxyl functional group, as well as a side chain residue that uniquely identifies a specific amino-acid. Out of the hundreds possible amino-acids occurring in nature, in a biological genome only 20 of them are codified and used as building blocks for proteins. Amino-acids form chains by means of condensation reactions producing peptide bonds. Given two amino-acids, a peptide bond is a covalent linkage between the carbon of the carboxyl group of the one amino-acid and the nitrogen in the amino group of the other amino-acid. The rotational degrees of freedom of the resulting dipeptide are the dihedral angles Φ and Ψ which characterise its spatial conformation [4].

In particular, proteins are biological polypeptides constituted of amino-acids synthesised inside the cell. According to the Central Dogma of molecular biology, proteins are the result of a two-part process: *transcription* and *translation*. At the level of the nucleus in eukaryotes and of the cytoplasm in prokaryotes, the genomic information stored inside DNA double strands is transcribed into mRNA filaments . The actual protein synthesis is performed within ribosomes, where the mRNA information is translated into a sequence of amino-acids. This sequence constitutes the primary structure, i.e. the most fundamental bit of information to describe the protein. However, as soon as the amino-acids emerge from the ribosome, the sequence starts folding at first into local complexes defining a secondary structure and eventually into a global three-dimensional

configuration, the native state or tertiary structure. Although there might be some environmental influences, the native state strongly characterises the biochemical functionality of the protein in a specific organism, so its knowledge can be highly valuable for medical and industrial purposes. However, unfortunately structure analysis is extremely expensive and time consuming compared to modern protein sequencing methods, i.e. techniques used to unveil the amino-acid sequence underlying a specific protein. As a result of this, the quantity of structural data is largely outnumbered by the amount of sequenced proteins and the only way to put to concrete use this information would require a full knowledge of the folding mechanism. Indeed, if this were understood, primary and tertiary structures would be virtually equivalent with each other.

1.1.1 Protein Folding Models

From a thermodynamics point of view, the native state of a protein represents a stable state or global minimum in the energy landscape associated to the amino-acid conformation space. How this state is reached constitutes the actual question of the problem. In order to be in the folded state each amino-acid must minimise the electrostatic interactions with any other amino-acid. This corresponds to finding the optimal conformation of the dihedral angles (Φ, Ψ) for each peptide bond. Since each of these has a varying number of stable configurations, the number of possible structures a polypeptide can fold into grows exponentially with the length of the backbone of the chain. In 1969, Cyrus Levinthal [5] pointed out that if a polypeptide had to randomly sample through all possible configurations, then reaching the native state would take it a time longer than the age of the universe, assuming a sampling rate of the order of nanoseconds or picoseconds. Undoubtedly, nature does not work this way and a protein folds into its native state in a time which is on average of the order of milliseconds, following pathways and short-cuts which are still not entirely understood. This is the main challenge behind the problem of protein folding: it is theoretically and computationally extremely hard to understand

and predict how a given amino-acid sequence will fold into its native state.

From a pure theoretical point of view, the standard way to find the structure of a molecule would be to write down its Hamiltonian and apply Schrodinger's equation in order to find the ground state, which would correspond to the native state of the protein. This method guarantees the most accurate results, as demonstrated in the field of Quantum Chemistry. However, for macromolecules the complexity of the interactions at play would make it impossible to diagonalise the system's Hamiltonian, making any attempt essentially vain.

Since a quantum mechanics approach is practically not feasible, the physics community focused its efforts in developing effective descriptions aimed at understanding the thermodynamics of the folding process and statistical inference methods to predict structural properties starting from amino-acid sequences. During the last 70 years, biophysicists presented different models which treated the folding process as a phase transition between an unfolded and a native state. An example of these is the 2-state model, which correctly predicted the phenomenon of cold unfolding. Other examples may include the diffusion and 2-state kinetics models, which determined the temperature dependence of the escape rate from the unfolded to the native state and their relaxation dynamics [6].

Even if these models give some insights about the nature of the folding transition, they do not allow us to understand the actual mechanism that a protein spontaneously follows to reach its native state. Therefore, the ability to predict structural information about an amino-acid sequence and generate new sequences with specifically selected features still eludes us. However, the recent development in statistical and machine learning techniques, combined with the ever increasing amount of sequenced data available to train the models, constituted a huge leap forward in protein structure prediction. In 2020 the Critical Assessment of Protein Structure Prediction (CASP), a worldwide biennial community experiment to determine the state of the art in modeling protein structure, reached an astonishing result with Google's architecture AlphaFold2 [7] which scored a prediction accuracy measure higher than 90%. Up

to this day, this constitutes the best result in determining the structure of a protein given its amino-acid sequence and effectively solves the prediction problem. Nonetheless, any deep learning architecture and in particular the one involved in AlphaFold2 poses a problem of interpretability: a human observer cannot consistently predict the model's result or simply understand the reasons behind the prediction. In other words, even if AlphaFold2 does solve the predictive aspect of the Protein Folding Problem, it does not convey any information on the actual folding mechanism. In this sense, the problem is still wide open.

1.2 The Protein Design Problem

Complementary to the problem of protein folding which focuses on understanding the mechanism that produces the native state of an amino-acid sequence, the Protein Design Problem aims at developing a systematic method to synthesise artificial amino-acid sequences which spontaneously and accurately fold into a given target three-dimensional structure. As a consequence, this is often referred to as the Inverse Folding Problem and its relevance is deeply rooted in the fact that the biological functionality of a protein can be engineered by acting directly on its spatial conformation. Therefore, being able to design amino-acid sequences capable of folding into desired structures would allow researchers to create proteins with specifically selected functionalities useful in a plethora of fields ranging from medical to environmental or industrial purposes [8], [9].

Even though the Protein Design Problem could be tackled using deep learning in an analogous way to AlphaFold2, the available amount of data paves the way to alternative inference methods based on statistical physics models. These do not bring about the problem of interpretability due to their reduced architectural complexity. The general idea behind statistical physics models for protein structure analysis relies in the assumption that there exists a set of potential amino-acid sequence which fold into specific structures related to particular biological functions. Then, the actual observed proteins can be thought of as samples drawn from this set according to some probability distribution that

captures the characteristics of those specific structures. The goal of statistical physics is to model such probability distributions so that to learn from sequenced data the largest amount of information relative to the hidden, artificial protein space.

In the following chapters we describe in details the general methods of statistical physics modelling and the current state-of-the-art techniques, Direct Coupling Analysis, used to extrapolate structural information starting from evolutionary-related amino-acid sequences. Then, we explore the attention mechanism, a recent machine learning development which had a primary role in AlphaFold2 and that is gaining more and more popularity in the protein community, as well as in the natural language processing community where it was actually conceived. Finally, we present a possible development of DCA inspired by the attention mechanism.

Chapter 2

Statistical Inference

In this chapter we set the main statistical framework starting from the evolutionary evidence that make possible this particular modellisation. Then, we develop the mathematical treatment leading to the definition of the Potts model. Finally, we explore possible solutions to the Inverse Potts Problem and the techniques used to extract structural information from the model itself.

2.1 Co-evolution and genetic compensation

The reasons behind the recent uprise of statistical physics models in the field of protein analysis are to be sought in two main factors. From a purely technical point of view, the last decade has seen significant improvements in protein sequencing methods, resulting in a vast increment in the amount of available data suitable to deep statistical analysis. Moreover, from a biological point of view, researchers were able to recognise the natural tendency of proteins to conserve their structural conformation throughout the course of evolution. During long periods of time, random mutations in the genetic code of individuals may lead to the synthesis of dysfunctional proteins in which one or more amino-acids differ from the original sequence. This phenomenon could occasionally produce beneficial alteration of the phenotype, but most likely it is the cause of unfavourable,

deleterious diseases. When this is the case, on average natural selection prevents the mutation from being passed on to future generations by implicitly applying the principle of the survival of the fittest. Therefore, only those mutations which preserve the biological functionality of the mutated protein are likely to be preserved. The spatial conformation, on which the functionality depends, can be conserved whenever the presence of a new amino-acid in the chain is counterbalanced by another mutated amino-acid such that their mutual interaction mimics the one of the original pair. The positions of such amino-acids in the sequence are said to have *co-evolved*, producing a new protein which is phylogenetically related to the original one. Phylogenetically or evolutionary related proteins constitute a protein family, characterised by a certain biological functionality which is expressed across different species [10].

The ability to collect and categorise amino-acid sequences into different protein families can be turned into a very powerful statistical tool as it converts the myriads of sequenced proteins into a finite clustering system which can be used to selectively learn specific functionalities originated from repeated structures within the family. Indeed, this is the road we will take.

2.2 Multiple Sequence Alignments

Phylogenetically related sequences belonging to a protein family can be collected into data structures called Multiple Sequence Alignments (MSAs), constructed in such a way that homologous positions in the amino-acid chains are aligned in the MSA. Since the length of the various sequences may vary across the family and their mutual similarity can be limited, the task of constructing a Multiple Sequence Alignment can be computationally demanding. Before the final result is reached, sequences may be altered by inserting or deleting specific positions or adding gaps in order to produce a more meaningful alignment. Among the various techniques employed to this end, a common solution is the use of Hidden Markov Models [11] which determine the most likely MSA by assigning a probability to all possible combinations of gaps, matches and mismatches. Once

it is determined, a MSA identifies a protein family and can be used to produce a summary statistics on which a model can infer relevant patterns. In particular, under the assumption that the MSA is a sample from a hidden probability distribution, the goal of inverse statistical physics is that of inferring structural information of the protein family by learning the most compatible distribution with the summary statistics.

Mathematically, a MSA can be modelled as a matrix $\mathbf{A} = (A_i^a)$ with $i = 1, 2, \ldots, N$ and $a = 1, 2, \ldots, M$, where N is the length of each sequence, while M is the depth of the alignment, i.e. the number of sequences stored in it. Given the a^{th} sequence, element A_i^a represents the specific amino-acid at position i and it is encoded as an integer number between 1 and 21, representing the alphabet of cardinality $q = 21$ containing the 20 amino-acids and the gap sign "$-$" used during the alignment procedure to represent insertions and deletions.

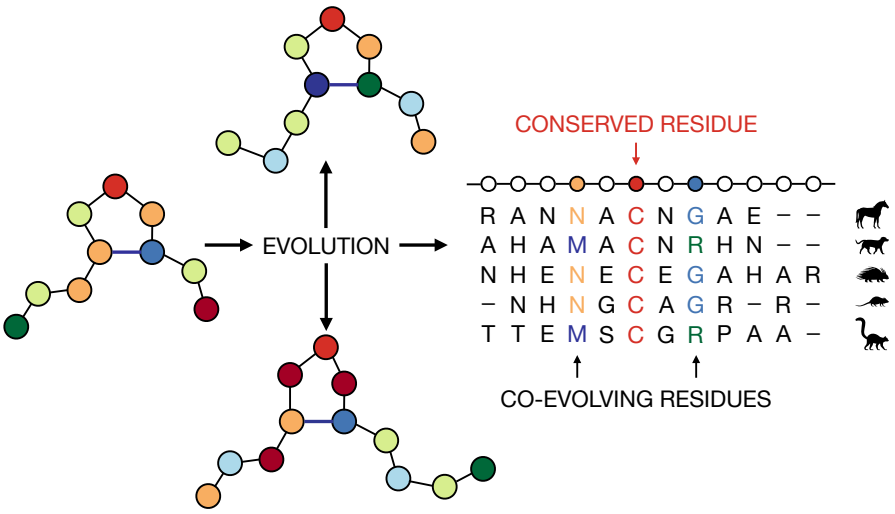


Figure 2.1: Natural selection constraints on the structure and biological functionality of proteins impose different levels of conservation along the evolutionary tree of a protein family. Constraints on specific site lead amino-acid conservation, while constraints on specific contacts lead to amino-acid co-evolution

Considering the average depth of actual MSAs, the most basic and yet in-

formative statistics we can extract out of them are the single and pair-wise frequency counts of the amino-acids in the alignments:

$$\begin{aligned} f_i(A) &= \frac{1}{M}\sum_{a=1}^{M}\delta_{A,A_i^a}, \\ f_{i,j}(A,B) &= \frac{1}{M}\sum_{a=1}^{M}\delta_{A,A_i^a}\delta_{B,A_j^a}. \end{aligned} \tag{2.1}$$

From a biological point of view, the phylogenetic character of the MSA brings about a certain degree of homology in the sequence pool which can be interpreted as a probabilistic bias towards a specific set of sequences. This may lead to false correlations and under-representation of biologically meaningful but under-sampled sequences. Because of this, empirical frequency counts have to be corrected by introducing a weighting factor for each sequence, in order to account for the fact that they are not independently drawn from a probability distribution. For a given sequence $\mathbf{A}^a$, its weight can be computed as the inverse of the number of sequences with Hamming distance $d^H(\mathbf{A}^a, \mathbf{A}^b)$ smaller than zN, where $z \in \{0, 1\}$ is a similarity threshold which usually is taken around 0.2-0.3. The sum of all these weights acts as the effective sequence number M_{eff} of the MSA:

$$\begin{aligned} m^a &= \Big|\big\{b|1 \leq b \leq M, d^H(\mathbf{A}^a, \mathbf{A}^b) \leq xN\big\}\Big|, \\ M_{eff} &= \sum_{a=1}^{M}\frac{1}{m^a}. \end{aligned}$$

Therefore, the frequency counts can be re-written as:

$$\begin{aligned} f_i(A) &= \frac{1}{M_{eff}}\sum_{a=1}^{M}\frac{1}{m^a}\delta_{A,A_i^a}, \\ f_{i,j}(A,B) &= \frac{1}{M_{eff}}\sum_{a=1}^{M}\frac{1}{m^a}\delta_{A,A_i^a}\delta_{B,A_j^a}. \end{aligned} \tag{2.2}$$

Regardless of the definition of the frequency counts, it is possible to evaluate the correlations between amino-acid position occupancy introducing the mutual

information

$$MI_{ij} = \sum_{A,B} f_{i,j}(A,B) \log \frac{f_{i,j}(A,B)}{f_i(A) f_j(B)}.$$

However, this measure cannot be used as a proxy for residue interactions in the three-dimensional structure of the protein. This is due to the fact that pairwise frequency counts are not representative of the direct interactions between two sites [12]. Indeed, even if residue contacts give rise to high correlations in position occupancy, this can be affected by indirect intermediate interactions as well. In order to disentangle these direct and indirect correlations, it is necessary to develop a model which explicitly only accounts for two-bodies interactions.

2.3 Principle of Maximum Entropy

Since the fundamental idea behind statistical physics approaches to protein analysis is that the observed sequences are samples drawn from a hidden probability distribution, we are interested in finding the optimal distribution consistent with the observables represented by the summary statistics extracted from the MSA.

In general, when presented with some observations of a given stochastic process, finding the most compatible probability distribution with those data is a classical problem in statistical inference. In particular, the desirable distribution is the one which better "explains" the observations by making the least amount of assumptions. In mathematical terms, this is equivalent to finding the least constrained or flattest distribution consistent with the given empirical observations. Given a discrete random variable $x \in X$ and some partial observation $\tilde{f}$, the Maximum Entropy Principle (MEP) states that the least constrained probability distribution compatible with the observation is the one which maximises the Shannon entropy

$$S[P] = -\sum_{x \in X} P(x) \log P(x), \tag{2.3}$$

under the constraints

$$\begin{aligned} \sum_{x\in X} f(x)P(x) &= \tilde{f}, \\ \sum_{x\in X} P(x) &= 1. \end{aligned} \tag{2.4}$$

This is a classical problem in information theory and can be straightforwardly thought of as a constrained functional maximisation problem [13]. The standard way to solve it is by means of Lagrange multipliers. In particular the functional to be maximised is written in the form

$$\mathcal{L}[P] = -\sum_{y\in X} P(y)\log P(y) - \lambda\big(\tilde{f} - \sum_{y\in X} f(y)P(y)\big) - \mu\big(1 - \sum_{y\in X} P(y)\big), \tag{2.5}$$

where λ and μ are the multipliers. The optimal distribution is computed by imposing the functional derivative equal to zero as in

$$\frac{\delta\mathcal{L}}{\delta P(x)} = -\log P(x) + \lambda f(x) + \text{const} = 0,$$

which implies a distribution of the form

$$\begin{aligned} P(x) \propto e^{\lambda f(x)} &= \frac{1}{Z} e^{\lambda f(x)}, \\ Z &= \sum_{x\in X} e^{\lambda f(x)} \end{aligned} \tag{2.6}$$

In the statistical physics jargon, the normalisation constant Z has the role of the partition function of the described system.

2.4 Potts Model and Direct Coupling Analysis

Given the single site and pair frequency counts define in equation 2.2, applying the MEP to infer the probability density $P(\mathbf{A}) = P(A_1, \ldots, A_N)$ defined over the space of protein sequences, we get:

$$P(\mathbf{A}) = \frac{1}{Z}\exp\Big\{\sum_{i<j} J_{ij}(A_i, A_j) + \sum_{i=1}^{N} h_i(A_i)\Big\}, \tag{2.7}$$

where $\{J_{ij}(A_i, A_j)\}$ and $\{h_i(A_i)\}$ act as the Lagrange multipliers used to impose respectively the constraints on the single and pair-wise frequency counts. In a

statistical physics context, this is the probability distribution of a paradigmatic model called Potts model, whose Hamiltonian reads

$$\mathcal{H} = -\sum_{i<j} J_{ij}(A_i, A_j) - \sum_{i=1}^{N} h_i(A_i) \tag{2.8}$$

The Potts model is a generalisation of the Ising model for interacting degrees of freedom whose states, often referred to as *colours*, are discrete values taken from a finite palette of q elements. In general, the degrees of freedom can be collocated at the vertices of a graph $\boldsymbol{G}$, specific of the system which is being described. In the context of modelling a protein family, the corresponding graph is fully-connected and the colours are to be intended as the 21 possible amino-acids with which sequences can be constructed. For what concerns the coupling tensor $\boldsymbol{J}$ and the field $\boldsymbol{h}$, their biological interpretation is that $\{h_i(A_i)\}$ represent the local amino-acid biases, while $\{J_{ij}(A_i, A_j)\}$ are the direct couplings between amino-acids A_i and A_j, respectively at position i and j.

Once the form of the distribution has been established, in order to obtain actual information about the interactions within the protein family, the parameters of the model have to be inferred from the observables, effectively solving the Inverse Potts Problem. In statistical physics, solving the Potts model amounts to computing the partition function of the system which is then used to determine its properties. The inverse problem corresponds to finding the optimal parameters which fit the model and return an accurate description of the system. The solutions of the Inverse Potts Problem is at the core of all statistical physics techniques which go by the name of Direct Coupling Analysis (DCA) [14]. These techniques may present different approaches to the solution of the inference problem, either analytical or computational ones. In the following, we discuss the independent-site and mean-field approximation which can be thought of as approximated analytical methods, while next chapter is devoted to a purely computational method based on the principle of maximum likelihood.

2.4.1 Gauge Invariance

In order to solve the inverse problem, it is necessary to determine $\binom{N}{2}q^2 + Nq$ parameters corresponding to $\{J_{ij}(A_i, A_j)\}$ and $\{h_i(A_i)\}$. However, the consistency conditions

$$\begin{aligned} P_i(A_i) &= \sum_{\mathbf{A}\setminus A_i} P(\mathbf{A}) = f_i(A_i), \\ P_{i,j}(A_i, A_j) &= \sum_{\mathbf{A}\setminus\{A_i,A_j\}} P(\mathbf{A}) = f_{i,j}(A_i, A_j), \end{aligned} \tag{2.9}$$

only account for $\binom{N}{2}(q-1)^2 + N(q-1)$ independent equations. This has to do with the fact that both single and pair-wise empirical frequency counts must follow the normalisation and marginalisation conditions

$$1 = \sum_{A=1}^{q} f_i(A), \tag{2.10}$$

$$f_i(A) = \sum_{B=1}^{q} f_{ij}(A, B) \tag{2.11}$$

for all $i = 1, \ldots, N$. This implies a redundancy in the set of parameters which gives rise to a gauge invariance: the probability distribution $P(\mathbf{A})$ does not change under the transformation

$$\begin{cases} \tilde{J}_{ij}(A, B) = J_{ij}(A, B) + K_{ij}(A) + K_{ji}(B) \\ \tilde{h}_i(A) = h_i(A) + g_i - \sum_{j\neq i}(K_{ij}(A) + K_{ji}(A)) \end{cases}, \tag{2.12}$$

where $\{K_{ij}(A)\}$ and g_i are arbitrary quantities. In the following we shall use the *lattice gas gauge* in which

$$J_{i,j}(A, q) = J_{i,j}(q, A) = h_i(q) = 0$$

for all $i, j = 1, \ldots, N$ and $A = 1, \ldots, q$. This choice, which does not influence the following results, denotes the case where couplings and biases are measured with respect to the gap state q.

2.5 Solutions to the Inverse Potts Problem

Once the gauge is fixed, a naive solution to the inference problem would be that of directly computing the single site and pair marginal distributions $P_i(A)$ and $P_{ij}(A, B)$ with the goal of using them to single out the parameters $\{J_{ij}(A_i, A_j)\}$ and $\{h_i(A_i)\}$ in terms of the empirical frequencies $f_i(A_i)$ and $f_{i,j}(A_i, A_j)$. However, computing the marginals would require tracing out all other variable in a computationally unfeasible procedure which would take an exponential time of order $\mathcal{O}(q^N)$.

A more promising technique to solve the problem is based on the fact that the partition function contains all the information we may need about marginals. Indeed, from statistical physics we know the relationships:

$$\begin{aligned} P_i(A) &= -\frac{\partial \log Z}{\partial h_i(A)}, \\ C_{ij}(A, B) = P_{i,j}(A, B) - P_i(A)P_j(B) &= -\frac{\partial^2 \log Z}{\partial h_i(A)\partial h_j(B)}, \end{aligned} \tag{2.13}$$

where C_{ij} is the connected correlation function. These results would be extremely useful if there were a way to compute the partition function, which is another computational problem that can only be solved in exponential time. Approximate solutions make this feasible for small couplings. In particular, introducing the modified model

$$\mathcal{H}(\alpha) = -\alpha \sum_{i<j} J_{ij}(A_i, A_j) - \sum_{i=1}^{N} h_i(A_i),$$

it is possible to compute the corresponding Gibbs potential, defined as the Legendre transform of the free energy $\mathcal{F} = -\log Z$:

$$\mathcal{G}(\alpha) = -\log Z(\alpha) + \sum_{i=1}^{N} \sum_{X=1}^{q-1} h_i(X) P_i(X). \tag{2.14}$$

This is useful in that, due to the functional form of the Legendre transform, for any given value of α biases and correlations can be computed as

$$h_i(A) = \frac{\partial \mathcal{G}(\alpha)}{\partial P_i(A)}, \tag{2.15}$$

$$(C^{-1})_{ij}(A, B) = \frac{\partial h_i(A)}{\partial P_j(B)} = \frac{\partial^2 \mathcal{G}(\alpha)}{\partial P_i(A)\partial P_j(B)}. \tag{2.16}$$

These computations can be performed by means of a small-coupling expansion [15] around the independent-site condition, i.e. a Taylor expansion around $\alpha = 0$

$$\mathcal{G}(\alpha) = \mathcal{G}(0) + \frac{\partial \mathcal{G}(\alpha)}{\partial \alpha}\Big|_{\alpha=0} \alpha + \mathcal{O}(\alpha^2) \tag{2.17}$$

2.5.1 Independent-site approximation

Sites are independent when there is no interaction between the degrees of freedom of the system. In our case, this can be obtained setting $\alpha = 0$. In this particular situation the Gibbs potential is equivalent to the opposite of the entropy. Indeed,

$$\begin{aligned} \mathcal{G}(\alpha = 0) &= \langle E \rangle(0) - S(0) + \sum_{i=1}^{N} \sum_{B=1}^{q} h_i(B) P_i(B) \\ &= -\sum_{i=1}^{N} \sum_{A=i}^{q} h_i(B) P_i(B) - S(0) + \sum_{i=1}^{N} \sum_{B=1}^{q} h_i(B) P_i(B) = -S(\alpha = 0) \end{aligned} \tag{2.18}$$

Writing everything as

$$\mathcal{G}(\alpha = 0) = \sum_{i=1}^{N} \sum_{A=1}^{q-1} P_i(A) \log P_i(A) + \sum_{i=1}^{N} \Big[1 - \sum_{A=1}^{q-1} P_i(A)\Big] \log \Big[1 - \sum_{A=1}^{q-1} P_i(A)\Big] \tag{2.19}$$

we can use equation 2.15 to compute the biases:

$$h_i(A)\Big|_{\alpha=0} = \log \frac{P_i(A)}{P_i(q)} = \log \frac{f_i(A)}{f_i(q)} \tag{2.20}$$

for all $A = 1, \ldots, q-1$. Analogously, using equation 2.16 and inverting the resulting matrix we get

$$C_{ij}(A,B)\Big|_{\alpha=0} = \Big[P_i(A)\delta_{A,B} - P_i(A)P_i(B)\Big]\delta_{i,j} = \Big[f_i(A)\delta_{A,B} - f_i(A)f_i(B)\Big]\delta_{i,j} \tag{2.21}$$

As expected, different sites do not present any correlation, while different amino-acids in the same sites are anti-correlated. The independent-site approximation fits all the single-site frequency counts to the biases, but it does not have enough parameters to fit the two-point correlations as well. For that, it is necessary to move to a higher term in α.

2.5.2 Mean-field approximation

In order to compute the first order term, we need to evaluate the derivative of the Gibbs potential at $\alpha = 0$. Starting from equation 2.14, we get:

$$\begin{aligned}\frac{d\mathcal{G}(\alpha)}{d\alpha} &= -\frac{d\log Z(\alpha)}{d\alpha} - \sum_{i=1}^{N}\sum_{A=1}^{q-1}\frac{dh_i(A)}{d\alpha}P_i(A) \\ &= -\frac{1}{Z(\alpha)}\sum_{\mathbf{A}}\Big[\sum_{i<j}J_{i,j}(A_i,A_j) + \sum_{i=1}^{N}\frac{dh_i(A_i)}{d\alpha}\Big]e^{-\mathcal{H}(\alpha)} - \sum_{i=1}^{N}\sum_{A=1}^{q-1}\frac{dh_i(A)}{d\alpha}P_i(A) \\ &= -\Big\langle\sum_{i<j}J_{i,j}(A_i,A_j)\Big\rangle_{\alpha}. \end{aligned} \tag{2.22}$$

Since $\alpha = 0$ coincides with the independent-site case, the probability distribution can be factorised into single-site marginals resulting in a Mean-Field approximation. Then, the expectation value of the coupling tensor can be computed as:

$$\frac{d\mathcal{G}(\alpha)}{d\alpha}\Big|_{\alpha=0} = -\sum_{i<j}\sum_{A,B}J_{i,j}(A,B)P_i(A)P_j(B). \tag{2.23}$$

Then, up to first order the Gibbs potential can be approximated by

$$\mathcal{G}(\alpha) = \sum_{i=1}^{N}\sum_{A=1}^{q}P_i(A)\log P_i(A) - \alpha\sum_{i<j}\sum_{A,B}J_{i,j}(A,B)P_i(A)P_j(B) + \mathcal{O}(\alpha^2). \tag{2.24}$$

Once more, using equations 2.15 and 2.16 we determine the self-consistent equations for local biases and connected correlations:

$$\frac{P_i(A)}{P_i(q)} = \exp\Big\{h_i(A) + \sum_{j\neq i}\sum_{B=1}^{q-1}J_{i,j}(A,B)P_j(B)\Big\}, \tag{2.25}$$

$$(C^{-1})_{i,j}(A,B)\Big|_{\alpha=0} = \begin{cases} -J_{i,j}(A,B) & \text{for } i\neq j \\ \frac{\delta_{A,B}}{P_i(A)} + \frac{1}{P_i(q)} & \text{for } i = j \end{cases}. \tag{2.26}$$

Within the mean-field approximation, equations 2.25 and 2.26 solve the original problem of inferring biases $\{h_i(A_i)\}$ and couplings $\{J_{i,j}(A_i,A_j)\}$ by fitting single site and pair empirical frequency counts $f_i(A)$ and $f_{ij}(A_i,A_j)$ to single- and two-site marginals.

2.6 Interaction score and contact prediction

Regardless of the method used to tackle it, once the inverse problem has been solved and all the parameters have been determined, the relevant information regarding the residue-residue interactions is contained inside the coupling tensor. However, since $\mathbf{J}$ is a 4-order tensor, it associates a $q \times q$ matrix with any given position pair (i, j) making it not straightforward to rank the likelihood that two positions are in contact within the three-dimensional structure of the protein family. In order to do so, it is necessary to devise a mapping $\boldsymbol{J_{ij}} \to s \in \mathbb{R}$ returning a score associated with any given position pair. This is core behind

Historically, the first proxy to be introduced was a direct information measure [16] defined as the mutual information of the two-side model

$$P_{ij}^{dir}(A, B) = \frac{1}{Z_{ij}} \exp\Big\{ J_{ij}(A, B) + \tilde{h}_i(A) + \tilde{h}_j(B) \Big\}, \tag{2.27}$$

where $\{J_{ij}(A, B)\}$ are the already inferred couplings, while $\{\tilde{h}_i(A)\}$ are new biases obtained by imposing the single-site empirical frequencies

$$f_i(A) = \sum_{B=1}^{q} P_{ij}^{dir}(A, B) \tag{2.28}$$

and Z_{ij} is the corresponding normalisation constant. Then, the direct information is defined as

$$D_{ij} = \sum_{A,B} P_{ij}^{dir}(A, B) \log \frac{P_{ij}^{dir}(A, B)}{f_i(A) f_j(B)}. \tag{2.29}$$

Alternatively, a popular choice is to consider the *zero-sum gauge*:

$$\sum_{A=1}^{q} J_i(A) = \sum_{A=1}^{q} J_{ij}(A, B) = 0 \tag{2.30}$$

for all $i, j = 1, \ldots, N$ and $b = 1, \ldots, q$. Unlike the lattice-gas gauge, the zero-sum gauge does not break the symmetry between the states of the Potts model. It also minimises the Frobenius norm, which can be used as a reasonable measure of the interaction strength:

$$F_{ij} = \sqrt{\sum_{A,B=1}^{q-1} J_{ij}^2(A, B)}. \tag{2.31}$$

Moreover, trial and error [10] showed that the best results are achieved by implementing an Average-Product Correction (APC), for which

$$F_{ij}^{APC} = F_{ij} - \frac{F_i F_j}{F},$$

where

$$F_i = \frac{1}{N} \sum_{k \neq i} F_{ik},$$
$$F = \frac{1}{N(N-1)} \sum_{k,l \neq i,j} F_{kl}.$$

What this correction does is to subtract from the Frobenius norm a null-model contribution for the pair (i, j), due to single site properties of i and j. This is needed because it can be seen empirically that some residues are more prone to establish interactions, acting as a confounding factor which would increase the score of all the pairs containing those residues. Average-Product Correction effectively diminishes this effect.

Ultimately, contact prediction is implemented by sorting from higher to lower the Frobenius norm score for each of the $\binom{N}{2}$ possible position pairs and considering the first $L = \mathcal{O}(N)$ terms.

Chapter 3

Pseudo-loglikelihood Direct Coupling Analysis

This chapter is devoted to the discussion of PlmDCA, the current state-of-the-art DCA method based on a likelihood maximisation principle. After the theoretical and computational framework is established, we also present some results obtained by applying the model to three protein family MSA.

3.1 Maximum Likelihood Estimation

Section 2.3 was focused on the discussion of the problem of finding the least constrained probability distribution which is compatible to a given set of empirical observations. Another problem, very much related to this, is that of inferring the parameters of a known distribution starting from set of samples independently drawn from the distribution.

In the context of Bayesian statistics, a common solution to this inference problem is given by the Maximum Likelihood Estimate (MLE) [17]. Consider a probability distribution $P(\boldsymbol{x}, \{\boldsymbol{\theta}\})$ characterised by parameters $\boldsymbol{\theta}$ and a dataset of M independent samples $\mathcal{D} = \{\boldsymbol{x}^\mu\}_{\mu=1,\dots,M}$ drawn from it. The likelihood

function is defined as the joint probability of the parameters conditioned to the observation of the dataset: $P(\boldsymbol{\theta}|\mathcal{D})$. According to the Maximum Likelihood Principle, the parameters that better explain the observations are those which maximise the likelihood, i.e.

$$\boldsymbol{\theta}^{MLE} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\mathcal{D}). \tag{3.1}$$

Usually, for reasons of computational efficiency the best the likelihood is redefined as the opposite of the loglikelihood

$$\mathcal{L} = -\log P(\mathcal{D}|\boldsymbol{\theta}) \tag{3.2}$$

which is then minimised with respect to the parameters. This method is particularly useful when the analytical form of the function is particularly tractable and its convexity is easy to prove, so that the existence of a global minimum is guaranteed.

3.2 PlmDCA

For our purposes, the probability distribution is the one associated to the Potts model, described by equation 2.7 and characterised by the coupling tensor $\boldsymbol{J}$ and the biases $\boldsymbol{h}$, while the dataset is the Multiple Sequence Alignment $\boldsymbol{A} = (A_i^a)$. As discussed in section 2.2, after the reweighting implemented to mitigate possible phylogenetic biases in the alignment, all instances of the MSA can be thought to be independent with one another. Therefore, the conditional probability $P(\boldsymbol{A}|\boldsymbol{J},\boldsymbol{h})$ factorises into

$$P(\boldsymbol{A}|\boldsymbol{J},\boldsymbol{h}) = \prod_{a=1}^{M} P(\boldsymbol{A}^a|\boldsymbol{J},\boldsymbol{h}) \tag{3.3}$$

and the loglikelihood function becomes:

$$\begin{aligned}\mathcal{L}(\boldsymbol{A},\boldsymbol{J},\boldsymbol{h}) &= -\frac{1}{M}\sum_{a=1}^{M}\log P(\boldsymbol{A}^a|\boldsymbol{J},\boldsymbol{h}) \\ &= -\frac{1}{M}\sum_{a=1}^{M}\log\Big[\frac{1}{Z}\exp\Big(\sum_{i<j}J_{ij}(A_i^a,A_j^a)+\sum_{i=1}^{N}h_i(A_i^a)\Big)\Big].\end{aligned} \tag{3.4}$$

Although this is an analytic function, the computation of the partition function Z is extremely demanding and makes the problem of minimising $\mathcal{L}$ essentially unfeasible. A practical solution to this problem is that of considering the conditional probability $P(A_i|\boldsymbol{A}_{\setminus i})$ instead of the whole distribution, so that:

$$P(A_i|\boldsymbol{A}_{\setminus i}) = \frac{1}{Z_i}\exp\Big\{\sum_{j\neq i} J_{ij}(A_i, A_j) + h_i(A_i)\Big\}, \tag{3.5}$$

with

$$Z_i = \sum_{A=1}^{q}\exp\Big\{\sum_{j\neq i} J_{ij}(A, A_j) + h_i(A)\Big\}. \tag{3.6}$$

Being a sum only over the alphabet $\{1, 2, \ldots, q\}$, the partial partition function Z_i is easier to compute and we can define a pseudo-loglikelihood function [18] for any position in the alignment $i = 1, \ldots, N$ as

$$\begin{aligned} g_i(\boldsymbol{A}, \boldsymbol{J}, \boldsymbol{h}) &= -\frac{1}{M}\sum_{a=1}^{M}\log P(A_i^a|\boldsymbol{A}_{\setminus i}^a) \\ &= -\frac{1}{M}\sum_{a=1}^{M}\log\frac{\exp\Big\{\sum_{j\neq i} J_{ij}(A_i, A_j) + h_i(A_i)\Big\}}{\sum_{A=1}^{q}\exp\Big\{\sum_{j\neq i} J_{ij}(A, A_j) + h_i(A)\Big\}} \\ = -\frac{1}{M}\sum_{a=1}^{M}&\left\{\sum_{j\neq i} J_{ij}(A_i, A_j) + h_i(A_i) - \log\Big[\sum_{A=1}^{q}\exp\Big\{\sum_{j\neq i} J_{ij}(A, A_j) + h_i(A)\Big\}\Big]\right\} \end{aligned} \tag{3.7}$$

The parameters of the model can now be inferred by minimising equation 3.7 for each possible position in the sequence . The stationary point is found by means of a gradient-based minimisation algorithm which requires the explicit form of the gradient in order to optimise the computation of each update step. The gradient is given by

$$\frac{\partial g_i}{\partial J_{xy}(\alpha,\beta)} = -\frac{1}{M}\sum_{a=1}^{M}\mathbb{I}[A_y^a = \beta]\Big(\mathbb{I}[A_x^a = \alpha] - P(A_x = \alpha|\boldsymbol{A}_{\setminus x} = \boldsymbol{A}_{\setminus x}^a)\Big)\delta_{i,x}, \tag{3.8}$$

$$\frac{\partial g_i}{\partial h_x(\alpha)} = -\frac{1}{M}\sum_{a=1}^{M}\Big(\mathbb{I}[A_x^a = \alpha] - P(A_x = \alpha|\boldsymbol{A}_{\setminus x} = \boldsymbol{A}_{\setminus x}^a)\Big)\delta_{i,x}. \tag{3.9}$$

Finally, one last adjustment that can be made is the addition of a regularisation term. This is usually included in order to avoid overfitting the model on the

available data, mitigating the finite sampling problem. In this case, the standard choice is to use the L2 regularisation

$$R_i(\boldsymbol{J}, \boldsymbol{h}) = \lambda_J \sum_{i<j} \sum_{A,B} J_{ij}^2(A,B) + \lambda_h \sum_{A=1}^{q} h_i^2(A), \tag{3.10}$$

which penalises arbitrarily large coupling terms. Empirically, a good choice for the regularisation parameters is $\lambda_J = \lambda_h \approx 0.01$ for most of the MSAs we deal with.

At this point , the optimisation consists in N parallel minimisations for each $g_i^{reg} = g_i + R_i$ with $i = 1, \ldots, N$. Each of these minimisations will return the set of optimal parameters $J_{i,*}^{MLE}$ and h_i^{MLE}. The final step is to enforce by hand the symmetrisation $J_{i,j}(A,B) = J_{j,i}(B,A)$ by imposing

$$J_{ij}(A,B) = J_{ji}(B,A) \longleftarrow \frac{1}{2}[J_{ij}(A,B) + J_{ji}(B,A)].$$

Once the coupling tensor has been inferred, residue-residue contacts are predicted by by computing the Frobenius norm of all $\binom{N}{2}$ possible contacts, sorting the resulting score from higher to lower.

3.3 Results from PlmDCA

The accuracy of PlmDCA can be investigated by applying it to protein families of which the structure has already been established by experimental means. In this is way it possible to compare the predicted residue-residue contacts with the actual contacts in the three-dimensional structure of the family. In general, for a given protein family, the associated structure is given as the relative positions of all heavy elements constituting the amino-acids. This can be referred either to the structure of a single sequence or to the average of the structures of multiple sequences within the family.

Upon defining a threshold below which residues are considered in contact, structural information can be used to extrapolate the effective contact positions which serve as a benchmark to validate the contacts predicted by the model. In the following, two amino-acids are considered in contact if they are within 8Å

in the folded structure, but lie apart by at least 6 positions in the underlying sequence.

We considered three protein families of different dimensions taken from the Pfam database[1]. The first family is the PF00014 which contains active domains of protease catalyzers, called Kunitz domains. The corresponding MSA contains 8871 non-identical sequences of 53 amino-acids. Figure 3.1 shows a specific instance of this family, the domain 1KTH found in *homo sapiens*.

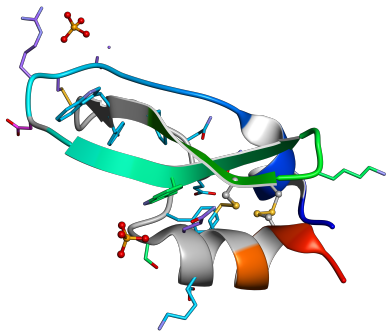


Figure 3.1: Kunitz domain 1KTH of the Protein Family PF00014

The second family is the PF00595 which contains PDZ domains, common structural domains found in signaling proteins of bacteria, yeast, plants and animals. Its corresponding MSA contains 15299 sequences of 82 residues. Figure 3.2 shows structure 1B8Q, a PDZ found in *Rattus norvegicus*.

[1] https://pfam.xfam.org/

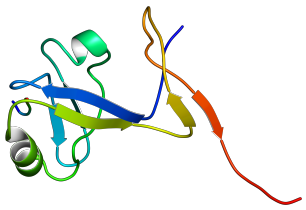


Figure 3.2: PDZ domain 1B8Q of the Protein Family PF00595

The third family, PF13354, consists of catalytic domains of class A of beta-lactamases and its MSA is composed of 7515 sequences of 202 amino-acids. Figure 3.3 shows structure 4R3B, a Beta-lactamase SHV-1 domain found in *Klebsiella pneumonia*e, an anaerobic, rod-shaped bacterium.

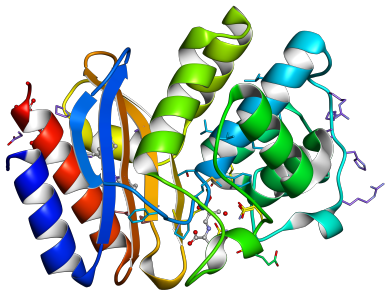


Figure 3.3: Beta-lactamase domain 4R3B of the Protein Family PF00595

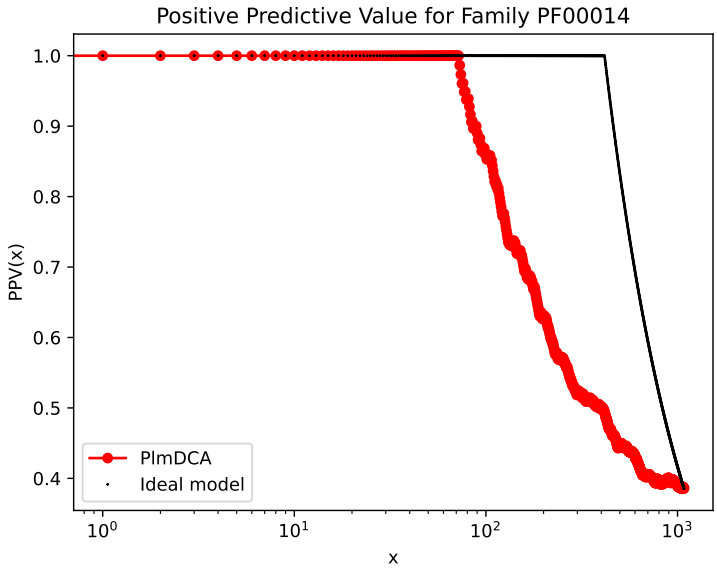
Using the data provided by the Pfam database, the accuracy of the model

has been evaluated by computing the Positive Predictive Value (PPV) curve. Given the Frobenius norm of the $\binom{N}{2}$ possible contacts, ranked from higher to lower, the PPV at $x \in (1, 2, \ldots, \binom{N}{2})$ returns the value

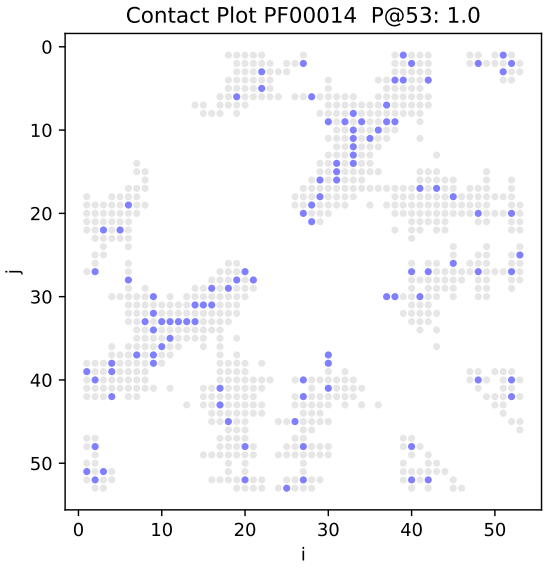
$$\mathrm{PPV}(x) = \frac{\mathrm{TP(x)}}{x}, \tag{3.11}$$

where $\mathrm{TP}(x)$ represents the number of true contacts predicted among the first x predictions. Ideally, for a perfect inference model the PPV curve should be constant for the first N_c predictions and then it should decrease as N_c/x, where N_c is the number of actual contacts in the structure.

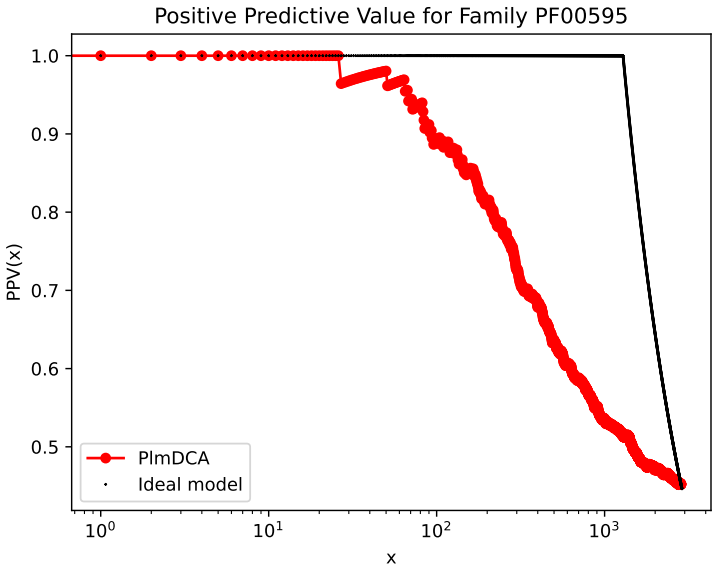
Moreover, for each family we included a graphical representation of the contact map, a binary matrix indicating whether positions i and j are in contact. In our graphs, the actual contact map is superimposed with the predicted contact map defined using the first N predictions. Positive predictions are shown as blue dots, while negative ones are in red. Grey dots indicate the effective structure obtained by the information provided by the Pfam database.



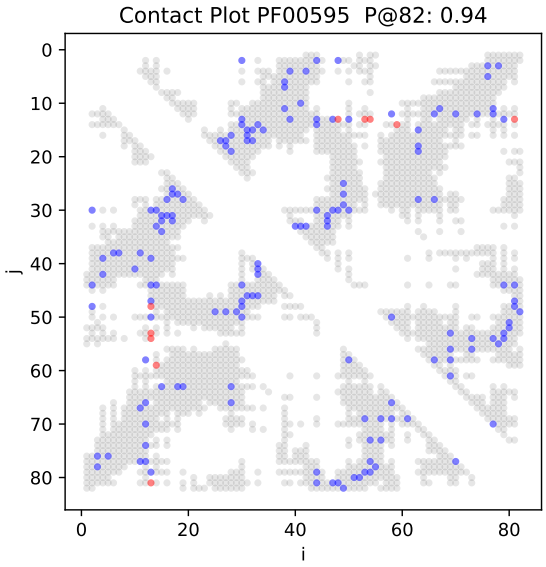
(a) PPV Family PF00014



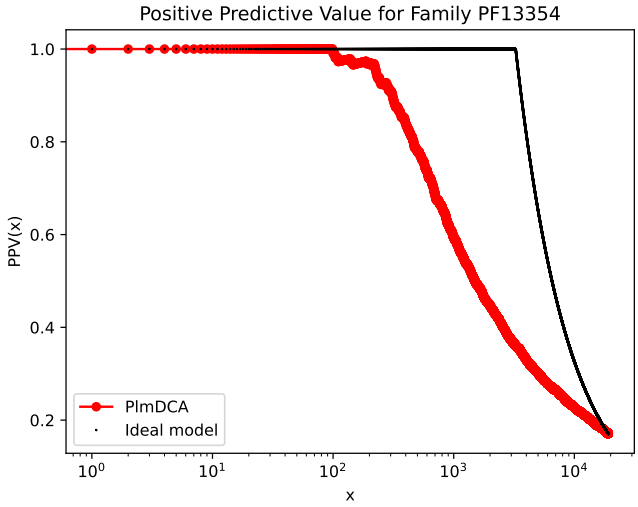
(b) Contact Plot Family PF00014



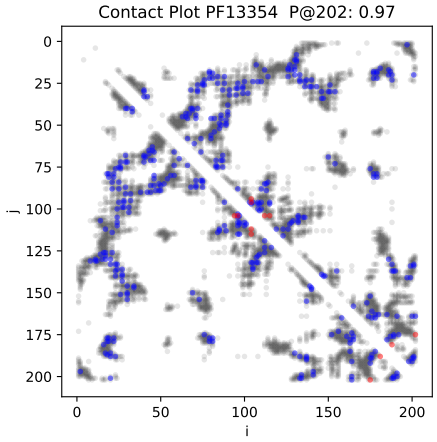
(c) PPV Family PF00595



(d) Contact Plot Family PF00595



(e) PPV Family PF13354



(f) Contact Plot Family PF13354

Figure 3.4: (a), (c), (e): Positive Predictive Value as a function of the number of predicted contacts. The red and black curve represent respectively the PPV of PlmDCA and that of an ideal model that correctly predicts all and only true contacts. (b), (d), (f): Graphical representation of the symmetric Contact Matrix, where grey dots correspond to the known structure of the family, while blue and red dots correspond to true and false predictions by PlmDCA respectively. P@N represents the PPV evaluated at N.

Far from being a perfect model, as shown by the PPV curve, nonetheless PlmDCA is able to capture hundreds of contacts with little computational effort. Indeed, the minimisation can be easily parallelised by writing the total likelihood as a disjoint sum of single-site likelihood functions as seen in equation 3.2. Each of these functions g_i depends only on a set of parameters $(\boldsymbol{J}_{i,*}, \boldsymbol{h}_i)$ which can be learned in parallel [19].

Chapter 4

Attention Mechanism

In this chapter we give an overview of the main concepts and models developed in the context of Natural-Language Processing (NLP). In particular, we introduce the Attention Mechanism as a general method for emulating human cognitive attention and its different implementations. Finally, we present the Transformer, a attention-based encoder-decoder architecture introduced by a team at Google Brain in 2017.

4.1 Natural Language Processing

As the name suggests, a natural language is any language used in human societies which has developed and evolved naturally without any intentional planning or premeditation. Natural Language Processing is a particular field of artificial intelligence mainly concerned with two broad tasks: Natural-Language Generation (NLG) and Natural-Language Interpretation (NLI) [20], [21]. Generally speaking, NLI deals with machine reading comprehension, i.e. the ability to process text and understand its meaning by recognising words and their interplay in the construction of sentences. In particular, examples of this may include speech recognition, text classification or sentiment analysis. NLG is concerned with the construction of specific architectures that can produce human-readable

(or -understandable) text, taking as input textual or non-textual information. Possible application of NLG are image captioning, text summarisation and text emulation. Many more NLP applications can be classified both as interpretation and generation tasks. Among these, two of the most iconic applications are machine translation programs and speech-emulating software, also known as chatbox. In the first case, the machine must understand the given text in a specific language and generate a textual output which conveys the same information as the input, but in another language. In the second case, the software must reply to the user input emulating a conversation, therefore it must be able to discriminate between assertions and questions producing a meaningful comeback with respect to the general context of the discussion.

4.1.1 Sequential Processing

Among the different techniques and architectures developed by the NLP community, a first breakthrough was represented by Google's *Seq2seq* [22] in 2014. *Seq2seq* is a multi-purpose family of machine learning approaches to NLP based on the sequential transformation of an input sequence, representing textual or non textual information, into an output sequence by means of an encoder-decoder structure made of Recurrent Neural Networks (RNNs). A Recurrent Neural Network is any neural network which presents feedback connections, effectively introducing loops in the topology of the model. This defines a non-linear dynamical architecture particularly suited for sequential processing as in the case of NLP [23]. For a basic RNN, given an input sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ at time-step t the model computes a hidden state $\boldsymbol{h}_t$ as a function of the current input $\boldsymbol{x}_t$ and the previous state $\boldsymbol{h}_{t-1}$ which are concatenated and passed to a feedforward network with weight matrix $\boldsymbol{W}$, biases $\boldsymbol{b}$ and a sigmoid activation function:

$$\boldsymbol{h}_t = \sigma\Big(\boldsymbol{W} \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{h}_{t-1} \end{bmatrix} + \boldsymbol{b}\Big). \tag{4.1}$$

The hidden state is simultaneously stored for computing the next input's hidden state and passed to another feedforward layer to return the final output:

$$\boldsymbol{y}_t = f(\boldsymbol{V}\boldsymbol{h}_t + \boldsymbol{c}) \tag{4.2}$$

Usually the final layer contains a non-linear function f depending on the form of the expected output. As an example, categorical output may be preceded by a softmax layer which returns the probability distribution of the result among all the possible categories. Figure 4.1 shows the internal architecture of a basic recurrent layer.

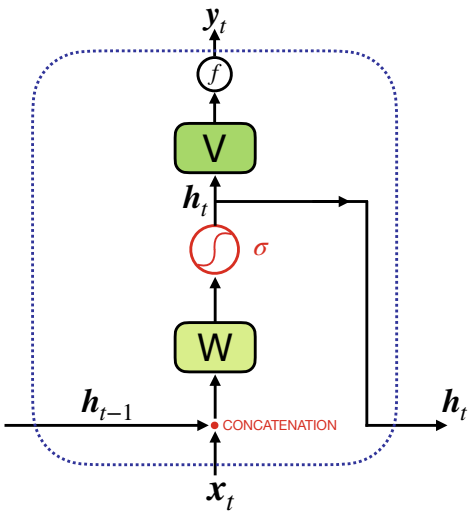


Figure 4.1: Schematic representation of the internal architecture of a Recurrent Neural Network (RNN). Current input $\boldsymbol{x}_t$ and previous hidden state $\boldsymbol{h}_{t-1}$ are concatenated and fed to feedforward layer with weights $\boldsymbol{W}$ and sigmoid activation. The result is the current hidden state of the RNN, which is fed to a new feedforward layer with weights $\boldsymbol{V}$ and activation f depending of the specific task of the model. This returns the current output of the network. Bias terms in feedforward layers are omitted for a lighter graphical representation.

In the context of NLP, in order to be processed, each element of an input sentence has to be mapped into a unique mathematical representation suitable for the machine to understand. A simple solution to this problem can be a

dictionary ID lookup, in which an integer number is assigned to each word of a given dictionary. Another possibility is to implement a one-hot encoding so that words are mapped into finite-size vectors whose components are all zero except for a single element, different for each word, which takes value 1. However, even though these solutions have their field of applications, they fail to represent a fundamental aspect of natural languages which is the fact that words have a semantic and morphological relationship with one another. Word embedding solves this problem by introducing real-valued vectors that encode the meaning of a word in such a way that if two words have similar meaning their vector representations will be close in the vector space. In this way, the semantic structure of a language and its dictionary can be fully represented into a mathematical form.

In order to understand the fundamental idea underneath the *Seq2seq* algorithm, consider an input sequence representing a sentence $I = [I_1, I_2, \dots, I_N]$, where I_i can either be a word or a punctuation mark. Each element of the sentence is mapped into the embedding space producing the actual input $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N)$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ is the embedding of the i^{th} word and D is the dimensionality of the embedding space. Each embedding is then processed sequentially by a RNN which encodes the input information into a fixed-length context vector $\boldsymbol{c}$, given by [23]:

$$\boldsymbol{c} = f(\{\boldsymbol{h}_1, \boldsymbol{h}_2, \dots, \boldsymbol{h}_N\}), \tag{4.3}$$

where $\boldsymbol{h}_t = g(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$ is the hidden state of the RNN at time t. Functions f and g depend on the specific implementation of the network. A common choice is to take as context vector the final hidden state of the sequence, so that $\boldsymbol{c} = \boldsymbol{h}_N$. The encoded information stored in the context vector is then passed into a RNN decoder architecture which is trained to sequentially predict the next output word $\boldsymbol{y}_t$ as a function of the previous predictions $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{t-1}\}$. Practically, the starting input of the decoder are $\boldsymbol{c}$ and a <START> token which tells the model when to initiate the prediction procedure. Eventually, it stops when the <END> token is generated. Finally, the output sequence $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_M\}$

is mapped into the word representation producing the actual output sentence $O = [O_1, O_2, \ldots, O_M]$. From a mathematical point of view, the decoder implements a time-step-based factorisation of the probability distribution of the output sequence:

$$P(\boldsymbol{y}) = \prod_{t=1}^{M} P(\boldsymbol{y}_t | \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\}, \boldsymbol{c}), \tag{4.4}$$

with

$$P(\boldsymbol{y}_t | \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\}, \boldsymbol{c}) = F(\boldsymbol{y}_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}), \tag{4.5}$$

where F is a non-linear function, specific of the choice of the RNN architecture, while $\boldsymbol{s}_t$ is the hidden state of the decoder at time t. Figure 4.2 shows a schematic implementation of the encoder-decoder architecture. For clarity's sake, it should be mentioned that the RNNs used in *Seq2seq* are variations of the generic model shown in figure 4.1. In particular, in the encoder's RNNs all outputs are entirely neglected and only hidden states are passed to the decoder. Also, decoder's RNNs modified in order to include an extra input which corresponds to the context vector $\boldsymbol{c}$, which is fundamental in the prediction of the output $\boldsymbol{y}_t$.

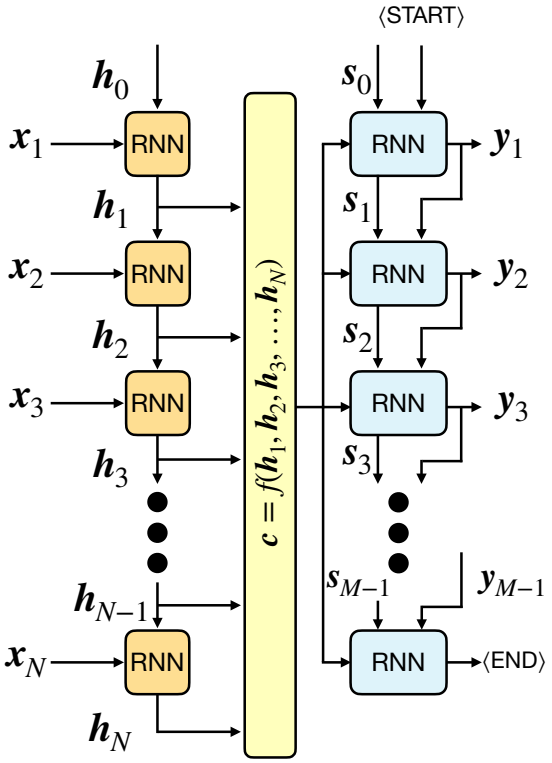


Figure 4.2: General representation of a *Seq2seq* architecture. At each iteration during the encoding process, a word embedding is passed as input to a Recurrent Neural Network (RNN) which produces a hidden state as a function of the previous hidden state and the current input. The encoder's hidden states are then used to construct a fixed-length context vector which is passed to the decoder. At each step of decoding phase, a RNN predicts a new output as a function of the context vector and both hidden state and output from the previous step. The decoding process initiates when a ⟨START⟩ token is passed to the first RNN and terminates when the ⟨END⟩ token is predicted.

The encoding mechanism implemented by *Seq2seq* constituted a great step forward in areas such as machine translation and text summarisation in that it provided a model that, at each time-step, is potentially capable of unbounded memory of previous words, regardless of their distance relative to the present input. This makes possible a better understanding of longer sentences, where words with a strong semantic relation might be far away from each other. However, from a practical point of view the actual memory of the model turns out to be very short due to the Vanishing Gradient Problem (VGP). In general, during the supervised training process, a backpropagation algorithm efficiently computes the gradient of a loss function with respect to the weights and biases of the model and recursively updates the parameters accordingly. Training stops when the loss function is minimised, i.e. when the model has learned the optimal weights and biases. The efficiency of this type of algorithms, based on the chain rule, puts backpropagation at the foundation of machine learning. Its main drawback depends on the presence of non-linear activations, such as hyperbolic tangents or sigmoid functions, whose gradient is a number in $(0, 1]$. As a consequence of the use of the chain rule, the backpropagation produces smaller and smaller gradients which make the update irrelevant, effectively halting the learning procedure. This problem grows more and more relevant for deep networks with a large number of layers.

Another criticality of the encoder-decoder RNN model is its sequential nature. During the training process, inputs must be fed sequentially into the architecture and because of this it cannot be parallelised. This makes such models extremely slow if compared with current state-of-the-art set models, where the input is fed as a single block, allowing for a better parallelisation.

4.1.2 Long Short-Term Memory

The Vanishing Gradient Problem is a common issue in deep neural networks and because of this different techniques have been devised to counteract its effect of stopping the learning procedure. Among these, the most frequently used

solutions include batch normalisation, residual connections or the adoption of different activation functions. Since standard activation functions such as hyperbolic tangents and sigmoid functions are almost flat for large inputs, resulting in vanishing derivatives, batch normalisation rescales the input so that it does not reach the outer edges of these functions. Another solution is that of implementing an additive connection between the two ends of an activation block. This residual connection is not flattened by the activation function, resulting in a higher overall derivative. Finally, a more straightforward solution is that of introducing different activation functions whose derivative do not tend to vanish. An increasingly common class of activation functions that suit this issues are called rectifiers and their prototype is the rectified Linear Unit (ReLU).

In the context of Natural Language Processing, a popular solution to VGP is represented by the Long Short-Term Memory unit [24], a recurrent network that can be used in encoder-decoder architectures as a more advanced replacement for the basic recurrent layer discussed in the previous section. At the core of this block there are gates specifically designed to implement a selective *forget & remember* process. Figure 4.3 shows the internal architecture of a LSTM block inside an encoder.

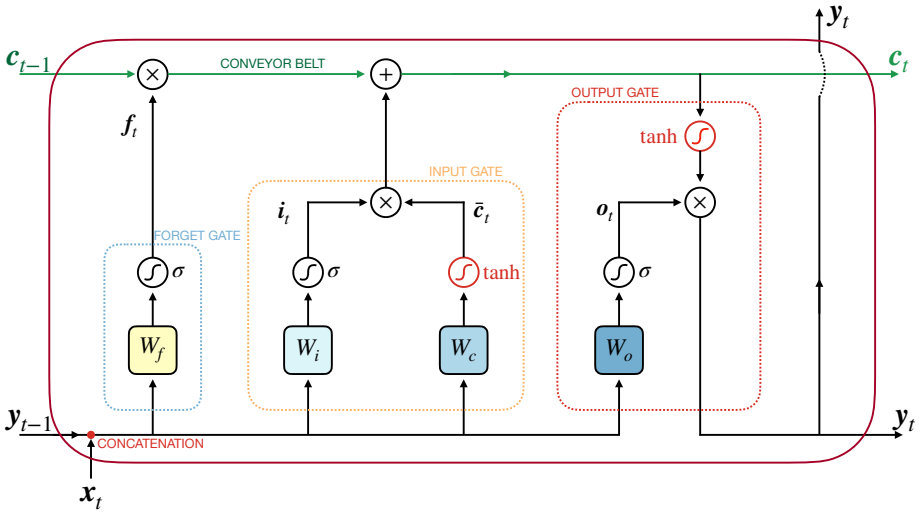


Figure 4.3: Internal representation of a Long Short-Term Memory layer. When used in encoder-decoder models, at each iteration the current input $\boldsymbol{x}_t$ and the previous output $\boldsymbol{y}_{t-1}$ are concatenated and activated by two non-linear gates, the forget gate and the input gate, with the goal of updating the cell state of the layer, $\boldsymbol{c}_{t-1}$. An output gate combines the input and the updated state to produce the new output $\boldsymbol{y}_t$ which is fed to the next iteration of the encoding or decoding process, along with the cell state $\boldsymbol{c}_t$. Bias terms in feedforward layers are omitted for a lighter graphical representation.

At each time-step, the LSTM encoder updates the cell state $\boldsymbol{c}_{t-1}$ and computes a new output vector $\boldsymbol{y}_t$ as a function of internal hidden states of the encoder given by:

$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma\Big(\boldsymbol{W}_f \boldsymbol{x}'_t + \boldsymbol{b}_f\Big), \\
\boldsymbol{i}_t &= \sigma\Big(\boldsymbol{W}_i \boldsymbol{x}'_t + \boldsymbol{b}_i\Big), \\
\bar{\boldsymbol{c}}_t &= \tanh\Big(\boldsymbol{W}_c \boldsymbol{x}'_t + \boldsymbol{b}_c\Big), \\
\boldsymbol{o}_t &= \sigma\Big(\boldsymbol{W}_o \boldsymbol{x}'_t + \boldsymbol{b}_o\Big),
\end{aligned}
$$

where $\boldsymbol{x}'_t$ is the concatenation of output $\boldsymbol{y}_{t-1}$ and the current input $\boldsymbol{x}_t$. The

new output and cell state are computed as:

$$\boldsymbol{c}_t = (\boldsymbol{f}_t \otimes \boldsymbol{c}_{t-1}) \oplus (\boldsymbol{i}_t \otimes \bar{\boldsymbol{c}}_t) \tag{4.6}$$

$$\boldsymbol{y}_t = \boldsymbol{o}_t \otimes \tanh \boldsymbol{c}_t, \tag{4.7}$$

where $\otimes$ and $\oplus$ represent element-wise multiplication and addition. In reference to figure 4.3, it possible to distinguish the three main components of the architecture: a forget gate, an input gate and an output gate. Each gate is modelled as a feedforward network characterised by a specific activation function and a set of weights and biases to be learned during the supervised training process. The forget gate erases part of the memory inside the cell state which is then updated by the input gate. Finally, the resulting state is fed to the output gate which produces the current output to be passed to the next iteration, along with the cell state itself.

The strength of this architecture is that the outputs are obtained combining linear and non-linear operations. This produces an additive gradient which has linear access to the forget gate's activations. The fact that the forget matrix can be learned to not vanish, along with the additivity property make the gradient much more likely to not vanish, making the training process actually feasible. However, even if replacing a standard RNN with a LSTM does mitigate the Vanishing Gradient Problem resulting in a longer memory of the encoder-decoder model, it does not improve its learning time. This is due to the fact that models such as *Seq2seq* remain intrinsically sequential and therefore not prone to parallelisation.

The first non-sequential model which solved this problem was presented in 2017 by Vaswani et al. [25] with the introduction of the Transformer architecture based on the Attention Mechanism. In the following sections we explore the concept of Attention in machine learning and how the Transformer drastically modified the field of NLP.

4.2 Attention and self-attention

In a RNN encoder-decoder, whether it is based on simple recurrent layers or LSTMs, the main fragility lies in the fact that the encoder has to represent all the information enclosed in the input sequence within a fixed-length context vector. If this fails to fully represent the input, then the decoder's output will be poorly defined.

Instead of having a fixed vector, it would be reasonable to introduce a model in which the decoder can look back at the entire input sequence, focusing on the parts that are more important for the prediction of the next output. The main idea behind attention is to enforce this mechanism in a convenient and meaningful way.

The attention mechanism was devised in an attempt to emulate biological cognitive attention, i.e. the ability to concentrate on relevant stimuli, blocking out those which are not necessary in producing a response to a particular input. In machine learning this can be implemented in different forms. A particularly useful form of attention for language interpretation is self- or intra-attention. Self-attention relates different positions of the same sequence in order to produce a representation which highlights the relative importance of each element with respect to the others. Consider an input sequence $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]$, where $\boldsymbol{x}_i \in \mathbb{R}^{D,1}$. A self-attention layer computes a row-wise normalised matrix $\boldsymbol{A}$ whose components A_{ij} express how much the i^{th} element attends to the j^{th} one, i.e. how much attention word $\boldsymbol{x}_i$ should pay to word $\boldsymbol{x}_j$. Using this attention matrix, the initial sequence can be mapped into a new representation which encodes the self-attention information:

$$\boldsymbol{y}_i = \sum_{j=1}^{N} \boldsymbol{A}_{ij} \boldsymbol{x}_j.$$

In order to compute the attention matrix, it is necessary to have a way to evaluate the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. In the case of simple self-attention,

this is done using a dot product or a scaled dot product, so that:

$$\boldsymbol{A} = \text{softmax}\Big(\frac{\boldsymbol{X}^T\boldsymbol{X}}{\sqrt{D}}\Big).$$

In this form simple self-attention has no learnable parameters and therefore its result is purely deterministic. To avoid this, we can introduce a matrix $W \in \mathbb{R}^{D,D}$ and define the attention matrix as:

$$\boldsymbol{A} = \text{softmax}\Big(\frac{\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}}{\sqrt{D}}\Big) \in \mathbb{R}^{N,N}.$$

In this way the model has more flexibility as it can learn the parameters W by means of backpropagation, also giving a deeper meaning to the similarity measure of the attention matrix.

4.2.1 Additive attention

In the context of encoder-decoder RNNs, the attention mechanism was introduced by Bahdanau et al. in 2014 [26] in order to tackle the problem relative to the fixed-length encoding vector. In particular, since the encoder always maps the input information into a fixed context vector, then for longer sequences the decoder has a limited access to the whole information provided by the input. This causes a bottleneck which lowers the performance of the model when applied to complex inputs. The solution proposed by Bahdanau involves an attention layer following the encoder and which provides the decoder with a different context vector at every sequential iteration of the decoding process. In this new architecture, the probability distribution of the output sequence is factorised into

$$P(\boldsymbol{y}) = \prod_{t=1}^{M} P(\boldsymbol{y}_t|\{\boldsymbol{y}_1,\dots,\boldsymbol{y}_{t-1}\},\boldsymbol{X}) = \prod_{t=1}^{M} g(\boldsymbol{y}_{t-1},\boldsymbol{s}_t,\boldsymbol{c}_t), \tag{4.8}$$

where $\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1},\boldsymbol{y}_{t-1},\boldsymbol{c}_t)$ is the hidden state of the decoder at time t and $\boldsymbol{c}_t$ is the specific context vector for the prediction of output $\boldsymbol{y}_t$. The computation of the context vector, which depends on a set of annotations $(\boldsymbol{h}_1,\dots,\boldsymbol{h}_N)$ representing the hidden states of the encoder, enforces the attention mechanism.

This is not to be intended as self-attention as the similarity score, or alignment score, is computed between the current hidden state of the decoder and all the annotations of the encoder. In particular, the annotations are mapped into the context vector as in

$$\boldsymbol{c}_i = \sum_{j=1}^{N} \alpha_{ij} h_j \tag{4.9}$$

where $\boldsymbol{\alpha} = (\alpha_{ij}) \in \mathbb{R}^{M,N}$ is the attention matrix. The main difference between the present attention mechanism and self-attention is that in this case the similarity score refers to elements of different vector spaces. A given row $\boldsymbol{\alpha_{i,*}}$ specifies how much hidden state $\boldsymbol{s}_{i-1}$ attends to every annotation $\{\boldsymbol{h}_j\}$. In particular, for each annotation-hidden state pair the alignment score is determined as in

$$\boldsymbol{e_{ij}} = a(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j), \tag{4.10}$$

where function a can be a linear or non-linear mapping, specific of the chosen implementation. The attention matrix is then defined as the row-wise softmax of the corresponding alignment matrix $\boldsymbol{e}$:

$$\boldsymbol{\alpha} = \text{softmax}(\boldsymbol{e}), \tag{4.11}$$

that is

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{N} \exp(e_{ik})}. \tag{4.12}$$

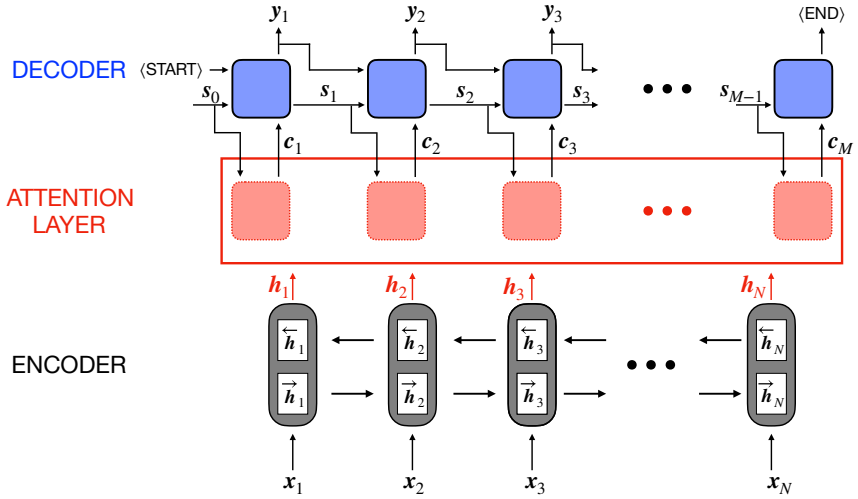
Being positive definite and normalised, α_{ij} can be thought of as a probability indicating the importance of annotation $\boldsymbol{h}_j$ with respect to the previous hidden state $\boldsymbol{s}_{i-1}$ in constructing the next state $\boldsymbol{s}_i$ and predicting the output $\boldsymbol{y}_i$.

The specific architecture proposed by Bahdanau consisted in a bidirectional-RNN encoder with LSTM or Hidden Gate units, which for each input produces two hidden states $\overrightarrow{\boldsymbol{h}}_i$ and $\overleftarrow{\boldsymbol{h}}_i$. These are computed sequentially from start to end and back, in order to capture the relations a word has with preceding and following words so that to provide a better representation of the semantic and

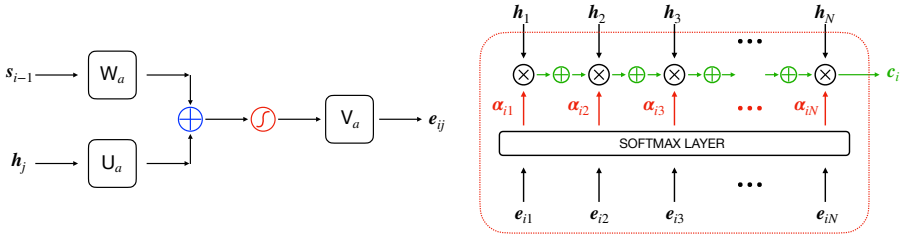
logical structure of the sentence. These two hidden states are concatenated and constitute the annotation for input $\boldsymbol{x}_i$. Then, for every iteration of the decoder, annotations are passed to the attention layer which is implemented as a feedforward network with the alignment score being defined as

$$e_{ij} = a(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j) = \boldsymbol{v}_a^T \tanh(\boldsymbol{W}_a \boldsymbol{s}_{i-1} + \boldsymbol{U}_a \boldsymbol{h}_j), \tag{4.13}$$

where $\boldsymbol{v}_a$, $\boldsymbol{W}_a$ and $\boldsymbol{U}_a$ are model parameters to be learned during the supervised training process. A softmax layer returns the weights α_{ij} which are used along with the annotations to compute the context vector, as in equation 4.9. This is then passed to the decoder which outputs the predicted result $\boldsymbol{y}_i$. The attention mechanism proposed by Bahdanau is known as *additive attention* as the similarity score between hidden states and annotations is computed by adding them together, as shown in equation 4.13 and graphically in figure 4.4b.



(a) RNN Encoder-Decoder with Additive Attention



(b) Feedforward network (c) Attention layer

Figure 4.4: (a): Bahdanau et al.'s architecture based on the encoder-decoder model with intermediate attention layer. For each sequential input, a bi-directional encoder computes the hidden state which is then passed to the attention layer. Iteratively, the decoder predicts an output $\boldsymbol{y}_t$ based on the context vector $\boldsymbol{c}_t$ computed in the attention layer. (b): Internal representation of the feedforward layer used to compute the alignment score $\boldsymbol{e}_{ij}$ relative to the encoder hidden state $\boldsymbol{h}_j$ and decoder hidden state $\boldsymbol{s}_{i-1}$. (c): Schematic representation of the attention layer producing context vector $\boldsymbol{c}_i$. The input $\boldsymbol{s}_i$ used in computing the alignment scores $\{\boldsymbol{e}_{i*}\}$ is omitted.

Figure 4.4a shows the encoder-decoder sequential model proposed by Bahdanau [26], while figures 4.4b and 4.4c show the internal architecture of the additive attention layer.

Every time the decoder produces an output, it soft-searches for relevant information in a sequence of vectors each of which encodes an element of the input sequence, effectively enforcing the attention mechanism. This way the encoder does not have the constraint of squeezing the entire input into a single fixed-length context vector. Because of this, Bahdanau's model represent an important improvement over the standard encoder-decoder approach. This is especially evident for long sequences, but it can be observed with sequences of any length. However, even if it implements the attention mechanism, this architecture remains a sequential model as the inputs are fed one at a time into the encoder. Therefore, as for standard encoder-decoder RNNs, the model cannot be easily parallelised and the training process remains slow.

4.3 Attention Is All You Need

4.3.1 Attention as a soft dictionary

A common trait between simple self-attention described in section 4.2 and additive attention seen in section 4.2.1 is that a set of inputs $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N)$ is mapped into a new representation $(\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_N)$ through a linear mapping $\boldsymbol{W}$ constructed by computing an attention or similarity score between elements $\{\boldsymbol{a}_i\}$ and $\{\boldsymbol{b}_j\}$ of two possibly different vector spaces , both associated with the initial input through some sort of relationship. Mathematically:

$$\boldsymbol{W} = (W_{ij}) = \Big(\text{score}(\boldsymbol{a}_i, \boldsymbol{b}_j)\Big), \tag{4.14}$$

$$\boldsymbol{z}'_i = \sum_j W_{ij} \boldsymbol{z}_j. \tag{4.15}$$

In dot product self-attention, every element involved is taken from the same pool, corresponding to the input sentence, and the similarity score is computed simply by means of a dot product. In Bahdanau's additive attention the score is given by a feedforward network and it involves hidden states of the encoder and the decoder.

The thread between these two versions is that the attention mechanism can be seen as a smart implementation of a soft-dictionary retrieval algorithm. Generally, in programming languages a hard dictionary is an associative array composed of keys paired to a unique value. In order to retrieve a value, we pass a query which is compared with all the keys in the dictionary until the matching one is found. Then, the dictionary returns the value corresponding to that key. If there is no matching between query and keys, the algorithm throws an error. Figure 4.5 shows as an example the implementation of a hard dictionary in Julia. In the example the query is the string "strawberry" and the retrieval system returns its value which is 5.

```
julia> my_dictionary = Dict("Apple"=>1, "Banana"=>2, "Kiwi"=>3, "Watermelon"=>4, "Strawberry"=>5)
Dict{String, Int64} with 5 entries:
  "Strawberry" => 5
  "Apple"      => 1
  "Watermelon" => 4
  "Kiwi"       => 3
  "Banana"     => 2

julia> my_dictionary["Strawberry"]
5
```

Figure 4.5: Example of a hard dictionary in Julia. Keys and values are in one-to-one correspondence and when a query is passed to the retrieval system, this returns the value associated to the unique key matching the query.

In contrast to this, in a soft dictionary the result is given by a weighted mixture of all the possible values with the weights depending on the compatibility between the given query and the key of each value. Consider a dictionary with a set of keys and values $\{\boldsymbol{k}_j, \boldsymbol{v}_j\}_{j=1,\dots,N}$, for a specific query $\boldsymbol{q_i}$ the soft dictionary algorithm will return

$$\boldsymbol{y}_i = \sum_{j=1}^{N} f_{score}(\boldsymbol{q}, \boldsymbol{k}_j)\boldsymbol{v}_j, \tag{4.16}$$

where f_{score} is the particular compatibility score of the algorithm.

The query, key and value formalism in the context of attention was introduced by Vaswani et al. in a paper of 2017 with the captivating title *Attention Is All You Need* [25]. They presented the Transformer, an encoder-decoder architecture based on this particular implementation of the attention and self-attention mechanism which allows for an efficient in-block reading of the input sentence. The Transformer effectively solved the last remaining issue in encoder-decoder NLP-oriented architectures, i.e. the long training time due to the sequentiality of the models. Indeed, in contrast with the standard sequential encoder-decoder, Vaswani et al.'s architecture can be conveniently parallelised making the training process remarkably faster.

In their architecture, an attention layer is characterised by the following operations. Consider a query vector $\boldsymbol{q} \in \mathbb{R}^d$ and a set of keys and value $\{\boldsymbol{k}_i, \boldsymbol{v}_i\}_{j=1,\dots,N}$ with $\boldsymbol{k}_i \in \mathbb{R}^d$ and $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$. The attention relative to that

specific query is given by:

$$\text{attention}\big(\boldsymbol{q}, \{\boldsymbol{k}\}, \{\boldsymbol{v}\}\big) = \sum_{i=1}^{N} \frac{\exp(\frac{\boldsymbol{q}\cdot\boldsymbol{k}_i}{\sqrt{d}})}{\sum_{l=1}^{N}\exp(\frac{\boldsymbol{q}\cdot\boldsymbol{k}_l}{\sqrt{d}})}\boldsymbol{v}_i. \tag{4.17}$$

Thus, this particular implementation of the attention mechanism relies on a scaled dot product similarity score, activated by a softmax function which normalises the attention weights and increases the sparsity of the model. For a set of queries $(\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N)$, the attention function can be efficiently computed simultaneously on each of them by combining all queries, keys and values in corresponding matrices $\boldsymbol{Q} \in \mathbb{R}^{N,d}$, $\boldsymbol{K} \in \mathbb{R}^{N,d}$ and $\boldsymbol{V} \in \mathbb{R}^{N,d_v}$. Using linear algebra, which is highly optimisable from a computational point of view, the attention matrix is given by:

$$\boldsymbol{A} = \text{attention}\big(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\big) = \text{softmax}\Big(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\Big)\boldsymbol{V}, \tag{4.18}$$

so that each row $\boldsymbol{A}_i$ corresponds to the attention vector relative to query $\boldsymbol{q}_i$.

4.3.2 Self-attention and Multi-Head attention

The nature of queries, keys and values depends on the particular model to which the attention mechanism is applied. For instance, in the case of self-attention all the elements depend on the initial input $\boldsymbol{X} \in \mathbb{R}^{N,d_x}$ which is used to generate $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ by means of linear mappings:

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{X}\boldsymbol{W}_Q, \\ \boldsymbol{K} &= \boldsymbol{X}\boldsymbol{W}_K, \\ \boldsymbol{V} &= \boldsymbol{X}\boldsymbol{W}_V. \end{aligned} \tag{4.19}$$

Matrices $\boldsymbol{W}_Q \in \mathbb{R}^{d_x,d}$, $\boldsymbol{W}_K \in \mathbb{R}^{d_x,d}$ and $\boldsymbol{W}_V \in \mathbb{R}^{d_x,d_v}$ constitute the parameters of the model and can be learned with the goal of highlighting specific intra-relationships within the input data. However, in many cases there are different kinds of such relationships. For example, consider the sentence "the restaurant was not too terrible". A NLP sentiment analysis algorithm designed to classify users' reviews should be able to understand that the word "terrible"

is a property of "restaurant", but also that "not" and "too" are both referred to "terrible", respectively inverting and moderating its meaning. Therefore, in this particular case there are three different kinds of relationships within the elements of the input sentence.

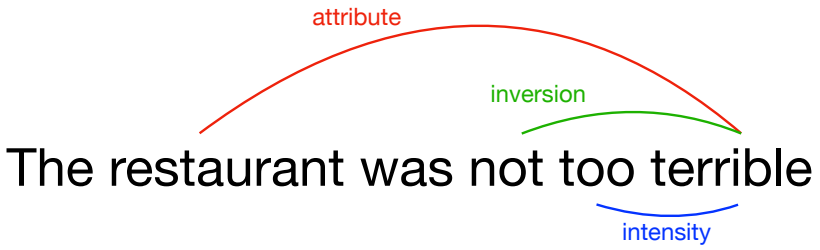


Figure 4.6: Example of the possible semantic relations in a sentence.

In order to deal with this fundamental aspect of natural languages, Vaswani et al. [25] proposed an in-parallel implementation of H self-attention mechanisms by mapping $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ with linear projections $(\boldsymbol{W}_Q^h, \boldsymbol{W}_K^h, \boldsymbol{W}_V^h)$, one for each *attention head* $h = 1, \ldots, H$. The results of each head are then concatenated and projected once more into $\boldsymbol{W}_O \in \mathbb{R}^{d_o}$ which gives the final output. Figure 4.7 shows the architecture of a multi-head layer.

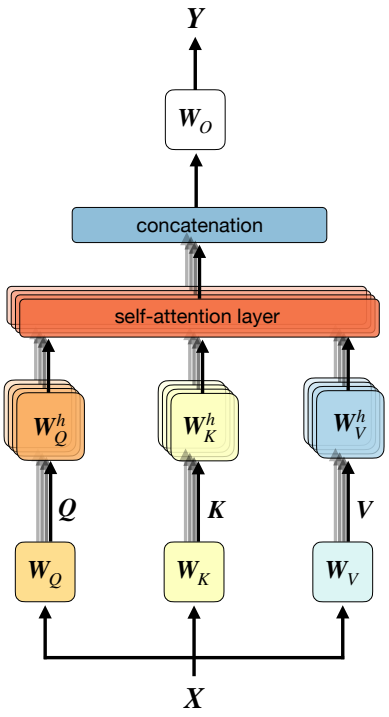


Figure 4.7: Multi-Head Attention layer. The matrix input $\boldsymbol{X}$ is mapped into Query, Key and Value matrices $(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$. These are re-mapped and fed in parallel to H self-attention layers. The results of these are concatenated and finally mapped into output matrix $\boldsymbol{Y}$.

4.3.3 Transformers

The main claim of Vaswani et al. is that the old RNN encoder-decoder architecture used for machine translation and other NLP tasks could be entirely replaced by a multi-head attention encoder-decoder model: the Transformer. As already mentioned, this development replaces the slow sequential model with a much faster block architecture, easy to parallelise and train. The only recurrence in the model happens in the decoder which predicts the next target output as a

function of the output predicted in earlier steps.

The transformer's encoder is composed of a multi-head self-attention followed by a feedforward layer. Then, its output is fed to the multi-head attention layer inside the decoder, acting both as queries and keys. The values, updated at each iteration of the decoding process, are provided by a multi-head *masked* self-attention computed on the word embeddings of all the previous outputs of the decoder itself. The mask consists of a matrix $\boldsymbol{M}$, such that

$$M_{ij} = \begin{cases} 0 & i \geq j \\ -\infty & i < j \end{cases}, \tag{4.20}$$

to be added to $\boldsymbol{QK}^T$, so that the resulting softmax matrix has null elements for $i < j$. This device is used to avoid word embeddings from attending to words in subsequent positions. This choice is implemented in order to ensure only causal relations among words, i.e. that the predictions for position i can only depend on the known outputs at positions less that i. Consider at time t the predicted sentence $(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_t)$. When computing the self attention for each element of this sentence, we want to preserve the fact that when word $\boldsymbol{y}_i$ was predicted there was no information regarding words $\boldsymbol{y}_{i+1}, \boldsymbol{y}_{i+2}, \ldots, \boldsymbol{y}_t$, so word $\boldsymbol{y}_i$ could only attend to previous predictions.

At each iteration, the result of the decoder's self-attention mechanism is fed to the multi-head attention layer which takes as queries and keys the output of the encoder attention layer, which is computed only once. Finally, the result is passed to a feedforward network.

A crucial point to be remarked is that, since this attention architecture is no longer sequential in the reading of the input, the model must include a device which introduces a notion of ordering in the input sentence. This is crucial in NLP as words may have very different meaning and very different relations with one another depending on their position in the sentence. In their article Vaswani et al. proposed a positional encoding to be added to the word embeddings. In particular, for a d-dimensional word embedding $\boldsymbol{x}_{\text{pos}}$, its positional encoding

$\boldsymbol{p}_{\text{pos}} = (p_1^{\text{pos}}, \ldots, p_d^{\text{pos}})$ is given by

$$p_{\text{pos},i} = \begin{cases} \sin(\text{pos}/10000^{2i/d}) & \text{i even} \\ \cos(\text{pos}/10000^{2i/d}) & \text{i odd} \end{cases}.$$

Finally, in order to reduce the possibility of having small gradients, which would impact the training process, each attention and feedforward layer is equipped with a residual connection and a normalisation layer. Figure 4.8 shows the Transformer architecture as depicted in Vaswani et al.'s paper.

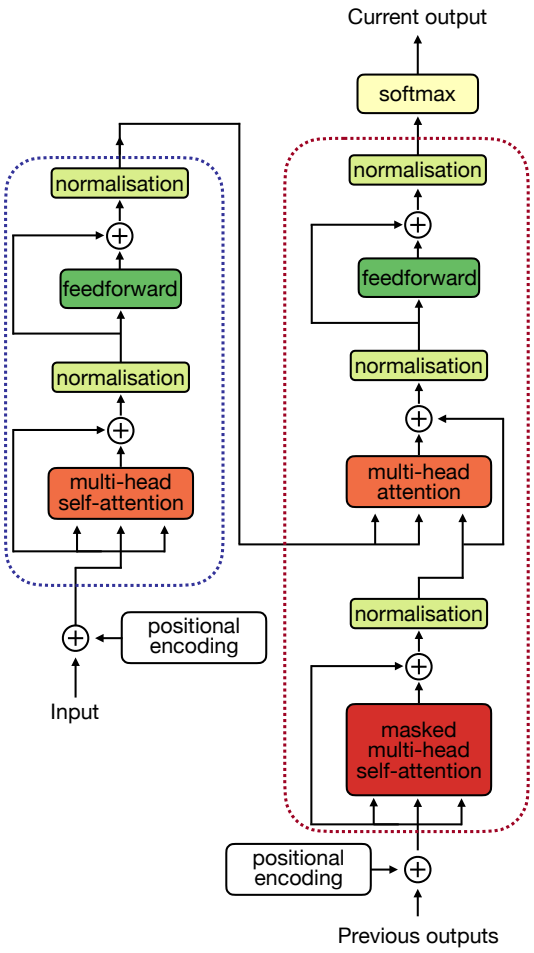


Figure 4.8: Schematic representation of the Transformer architecture. Positional encodings are added to the input sequence which is passed to a multi-head self attention layer. After a feedforward layer, the result acts as Query and Key in the decoder's multi-head attention. Iteratively, the Value matrix is given by a masked multi-head self attention evaluated on the outputs predicted on previous steps of the decoding procedure. The result of the decoder's multi-head attention is passed to a feedforward layer and activated by a softmax resulting in the current prediction. Throughout the architecture, residual connections and normalisation layers are used to mitigate the Vanishing Gradient Problem.

Chapter 5

Attention Based Direct Coupling Analysis

In this chapter we discuss how protein analysis can be modelled as a Language Processing task, implying that attention based architectures can be tailored specifically for this problem. Moreover, we introduce a variation of the standard PlmDCA method inspired by the attention mechanism and in particular by the multi-head attention discussed in section 4.3.2. Finally, we present a comparison between the results of this novel architecture and PlmDCA.

5.1 Protein language Models

The attention mechanism was conceived and developed in the context of natural language processing, giving excellent results in many tasks such as machine translation, sentiment analysis, text summarisation and many more. Nevertheless, the range of applicability of this mechanism is much wider than NLP, as it can be applied virtually to any machine learning task [27]. In particular, a straightforward implementation of the attention can be thought for those problems which can be modelled as a natural language. Protein Structure Analysis

belongs to this specific set of problems.

From a linguistic point of view, any natural language can be characterised by a morphology, defining the distinctive vocabulary of the language, a syntax which governs how words can be combined in a sentence and a semantic which determines its meaning. In parallel to this, a protein family can be modelled as a language whose dictionary is composed of the 20 naturally occurring amino-acids and with a morphology corresponding to the physical laws governing the residue-residue interactions which shape the three-dimensional structure. Each instance of the protein family represents a possible sentence in that language, with its semantic being conveyed by the biological functionality of the protein within an organism [28]. Alternatively, the set of amino-acids can be thought of as the alphabet of the language, where common structural motifs constituting the secondary structure or functional domains are treated as words of a vocabulary. However, this vocabulary would be ill-defined as in general it is not clear whether a sequence of amino-acids is part of a functional domain or not. This is in contrast with human languages which include punctuation to clearly and univocally separate structures such as words or sentences [29].

Regardless of the specific modellisation, protein families can be thought of as languages. Therefore, machine learning and statistical learning approaches to Protein Structure Analysis can be enhanced by applying attention-based concepts and architectures. In particular, after the introduction of the Transformer and various techniques based on that, such as GPT-3 or BERT [30], the computational biology community developed different implementation of these, specifically adapted to protein language models and usually pre-trained on protein sequence databases. Examples of these are ProtBERT, ProtXLNet and ProtTrans [31].

In contrast to these supervised models, we are interested in a unsupervised attention-based approach which leverages the evolutionary constraints of protein families. In the next sections we present *AttentionBasedDCA*, a version of PlmDCA inspired by the attention mechanism in its Query-Key-Value formulation.

5.2 AttentionBasedDCA

Following the work of Bhattacharya et al. [32], we propose to modify the Potts model defined in equation 2.7 by discarding the local bias term and introducing the following non-linear low-rank decomposition of the interaction tensor:

$$J_{i,j}(A_i, A_j) = \sum_{h=1}^{H} \text{softmax}(\boldsymbol{W}^h)_{ij} \boldsymbol{V}^h_{A_i,A_j} \tag{5.1}$$

$$= \sum_{h=1}^{H} \text{softmax}(\boldsymbol{Q}^h \boldsymbol{K}^{h^T})_{i,j} \boldsymbol{V}^h_{A_i,A_j}, \tag{5.2}$$

The main idea behind this new model, *AttentionBasedDCA*, is that we are interested in decoupling the information relative to the specific family under investigation and the information depending on the amino-acid pool which is common to all proteins. The form of equation 5.1 is inspired by the self-attention mechanism, such as in equation 4.18. In particular, for a protein family described by a MSA with sequences of length N, matrix $\boldsymbol{W} \in \mathbb{R}^{N,N}$ is meant to encode the positional information relative to two-body interactions specific of that family. Since it can be observed that the number of contacts in a protein grows linearly with the length of the sequence [32], it is reasonable to assume that matrix $\boldsymbol{W}$ is rather sparse. Therefore, we can lower its rank by decomposing it into matrices $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{N,d}$, where d is a hyper-parameter of the model. In this way, the number of independent parameters lowers from N^2 to $2Nd$, resulting in a linear model with respect to the MSA length. The row-wise softmax function is applied for normalisation purposes and in order to increase sparsity.

For protein sequence alignments composed of $q = 21$ possible amino-acids (20 natural amino-acids and the gap sign), matrix $\boldsymbol{V} \in \mathbb{R}^{q,q}$ is meant to capture the information relative to cross-family protein properties. Indeed, due to the universality of the amino-acid alphabet, proteins present local three-dimensional complexes which are repeated throughout their backbone, determining their secondary structure. Among these common structural motifs, two paradigmatic examples are the alpha helices and the beta sheets.

Finally, as in multi-head attention, the information is split in different triplets

or *heads* $\{\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\}_{h=1,\dots,H}$ each of which should ideally capture different properties of the protein family. Such properties may include the presence of specific patterns of amino-acids, their formal charge and their electrostatic interplay, as well as their three-dimensional relative arrangement or functional relation. From an attention point of view, the interaction term $J_{ij}(A_i, A_j)$ is built as the tensor product between the attention score $\text{softmax}(\boldsymbol{Q}^h \boldsymbol{K}^{h^T})_{ij}$ measuring the self-compatibility of position-dependent features and the amino-acid features $\boldsymbol{V}^h_{A_i,A_j}$. Therefore, we are assuming that the self-attention score is evaluated over the matrix representation of the information relative to the specific protein family.

5.2.1 Implementation

As in the case of PlmDCA, we are interested in inferring the $2HNd + Hq^2$ model parameters corresponding to matrices $\{\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\}_{h=1,\dots,H}$. This is equivalent to a unsupervised learning which is performed by minimising the pseudo-loglikelihood function built on the MSA $\boldsymbol{A} = (A_i^a)_{i=1,\dots,N}^{a=1,\dots,M}$

$$\begin{aligned} g(\boldsymbol{A}, \{\boldsymbol{Q}^h, \boldsymbol{K}^h, \boldsymbol{V}^h\}) &= -\frac{1}{M}\sum_{i=1}^{N}\sum_{a=1}^{M} \log P(A_i^a | \boldsymbol{A}^a_{\setminus i}) \\ &= -\frac{1}{M}\sum_{i=1}^{N}\sum_{a=1}^{M} \log \frac{\exp\Big\{\sum_{j\neq i} J_{ij}(A_i, A_j)\Big\}}{\sum_{c=1}^{q}\exp\Big\{\sum_{j\neq i} J_{ij}(c, A_j)\Big\}} \\ = -\frac{1}{M}\sum_{i=1}^{N}\sum_{a=1}^{M} &\left\{ \sum_{j\neq i} J_{ij}(A_i, A_j) - \log\Big[\sum_{c=1}^{q}\exp\Big\{\sum_{j\neq i} J_{ij}(c, A_j)\Big\}\Big]\right\}, \end{aligned} \tag{5.3}$$

where $J_{ij}(A_i, A_j)$ is given by equation 5.1. The optimisation of the parameters is performed by means of gradient-based minimisation algorithms requiring the

analytic form of the gradient, which is given by:

$$\frac{\partial g}{\partial Q^h_{xy}} = \frac{1}{M}\sum_{a=1}^{M}\sum_{j=1}^{N}\sum_{\alpha,\beta=1}^{q}\mathbb{I}[A^a_j=\beta]\Big(\mathbb{I}[A^a_x=\alpha]-P(A_x=\alpha|\boldsymbol{A}^a_{\setminus x})\Big)\times$$
$$\times\,[K^T_{yj}\ \mathrm{sf}(\boldsymbol{Q}^h\boldsymbol{K}^{h^T})_{xj}-\ \mathrm{sf}(\boldsymbol{Q}^h\boldsymbol{K}^{h^T})_{xj}\sum_{l=1}^{N}K^T_{yl}\ \mathrm{sf}(\boldsymbol{Q}^h\boldsymbol{K}^{h^T})_{xl}], \tag{5.4}$$

$$\frac{\partial g}{\partial K^h_{xy}} = \frac{1}{M}\sum_{a=1}^{M}\sum_{i,j=1}^{N}\sum_{\alpha,\beta=1}^{q}\mathbb{I}[A^a_j=\beta]\Big(\mathbb{I}[A^a_i=\alpha]-P(A_i=\alpha|\boldsymbol{A}^a_{\setminus i})\Big)\boldsymbol{V}^h_{\alpha,\beta}\boldsymbol{Q}^h_{iy}$$
$$\times\,[\mathrm{sf}(\boldsymbol{Q}^h\boldsymbol{K}^{h^T})_{ij}\delta_{xj}-\ \mathrm{sf}(\boldsymbol{Q}^h\boldsymbol{K}^{h^T})_{ij}\ \mathrm{sf}(\boldsymbol{Q}^h\boldsymbol{K}^{h^T})_{ix}], \tag{5.5}$$

$$\frac{\partial g}{\partial V^h_{\alpha\beta}} = \frac{1}{M}\sum_{a=1}^{M}\sum_{i,j=1}^{N}\mathbb{I}[A^a_j=\beta]\Big(\mathbb{I}[A^a_i=\alpha]-P(A_i=\alpha|\boldsymbol{A}^a_{\setminus i})\Big)\ \mathrm{sf}(\boldsymbol{Q}^h\boldsymbol{K}^{h^T})_{ij}]. \tag{5.6}$$

Finally, a coupling-wise L2 regularisation term is added to the total likelihood in order to reduce the possibility of overfitting the parameters:

$$R(\boldsymbol{J}) = \lambda\sum_{i,j}\sum_{A,B}J^2_{ij}(A,B), \tag{5.7}$$

where λ is the regularisation rate which we fixed to ~ 0.005 after some trial and error.

The actual implementation of the code has been carried out *de novo* using the programming language Julia. The main challenge was that of speeding up the computation of the gradient whose structure is more complex than PlmDCA's one. This was a crucial aspect since the optimisation procedure was enforced by a gradient descent algorithm. Moreover, the new form of the coupling tensor in equation 5.1 does not guarantee the proper convexity of the pseudo-loglikelihood function and the existence of an absolute minimum. Therefore, the gradient descent may be even more computationally demanding because a higher number of iterations are needed to converge. Finally, in contrast to what happens for PlmDCA, matrices $\{\boldsymbol{V}^h\}$ are shared across all positions $i = 1, \ldots, N$ resulting in single site pseudo-likelihood functions which depend on a common set of parameters. Therefore, a complete parallelisation of the minimisation procedure is not allowed and the optimisation time is much longer if compared to that of

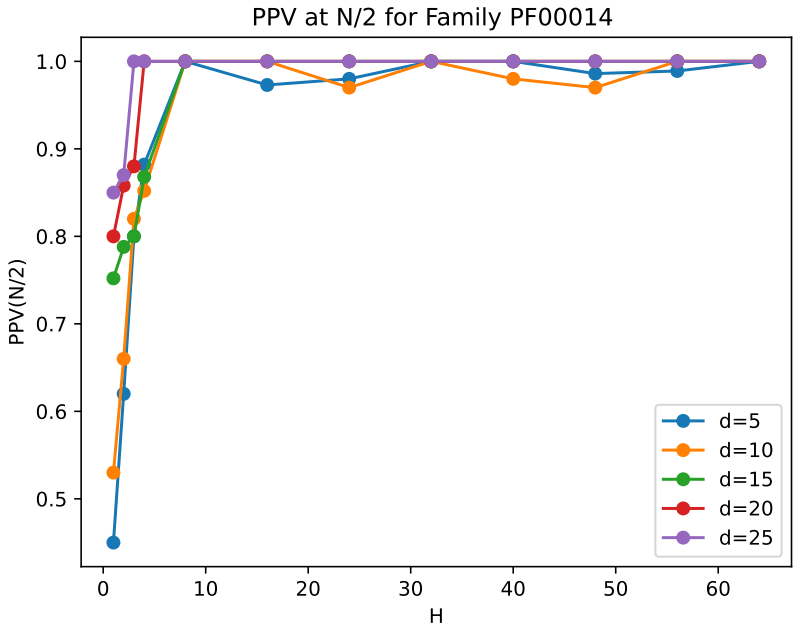
PlmDCA. A possible way we used to exploit the power of parallel computation was that of implementing a site-depending update of the gradient relative to each row of matrices $\{\boldsymbol{Q}^h, \boldsymbol{K}^h\}$. Future developments may focus on this important issue. The current code is stored in a private GitHub repository which will be released to the public as soon as few adjustments are made.

5.2.2 Results

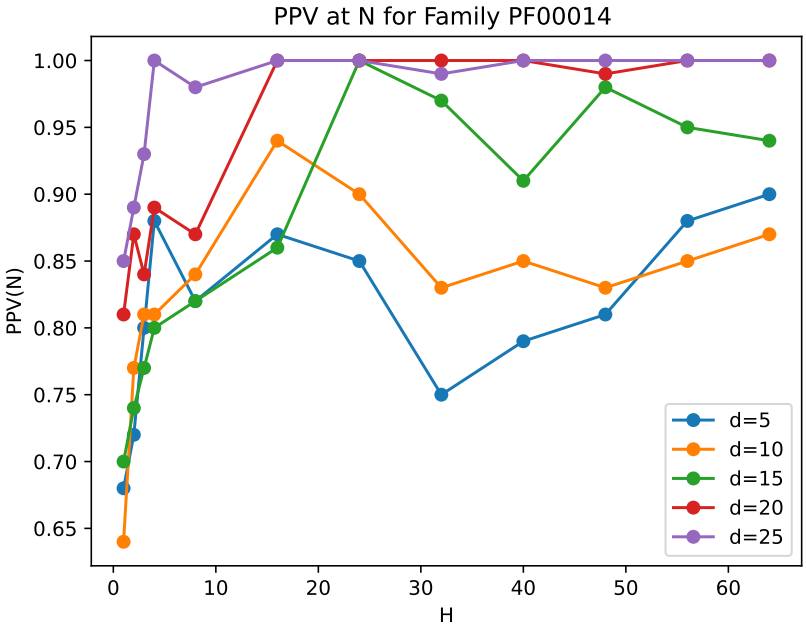
In the same way as in PlmDCA, once the parameters are optimised position-position contacts are predicted by computing a score which associates each of the $\binom{N}{2}$ position-pair (i, j) with the Frobenius norm of the corresponding $q \times q$ interaction matrix $\boldsymbol{J}_{ij}$ given by equation 5.1. Higher scores correspond to stronger interactions. The accuracy of the model can be evaluated by comparing predicted contacts either with known structural information belonging to protein families or with other state-of-the-art models such as PlmDCA itself. In the following the present the results obtained by applying the model to the same three protein families used for PlmDCA in section 3.3: PF00014, composed of 8871 Kunitz domains of 53 amino-acids; PF00595, composed of 15299 PDZ domains of 82 amino-acids; PF13354, composed of 7515 β-lactamase domains of 202 amino-acids.

The hyper-parameters of the model have been trained on family PF00014 for reasons of computational efficiency. In particular, in order to determine the optimal value of d and H, the lower dimension of the rectangular matrices $(\boldsymbol{Q}^h, \boldsymbol{K}^h)$ and the number of attention heads respectively, we evaluated the performance of the model in predicting the first $N/2$, N and $2N$ couplings, where $N = 53$ is the length of sequences in family PF00014. Figures 5.1a, 5.1b and 5.1c represent the Positive Predictive Value respectively at $N/2$, N and $2N$ as a function of the number of attention heads H and for different values of d. It can be noticed that for small values of H the model performs consistently badly, however it reaches a plateau around $H = 20$. For what concerns the lower dimension d, the best performances are reached for $d \gtrsim 15$. As expected,

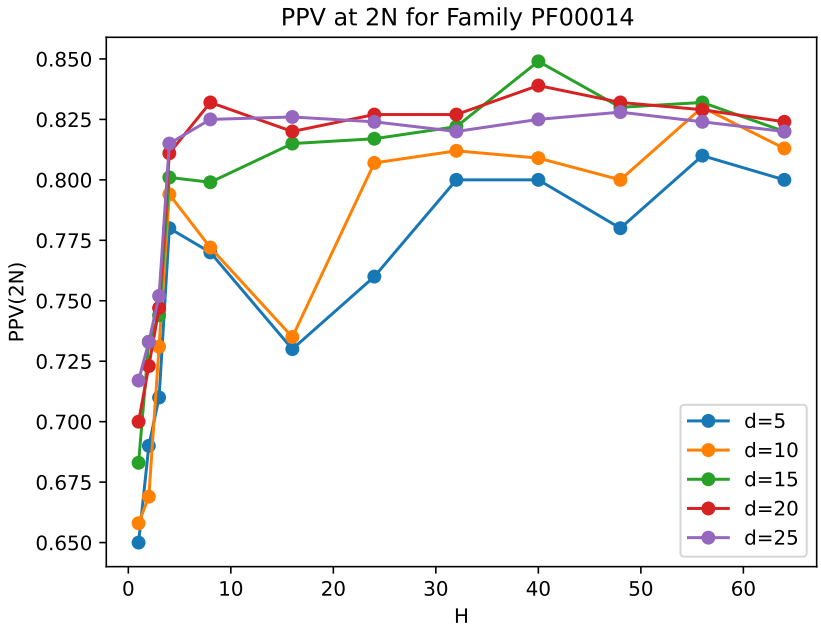
the model is quite accurate in predicting the first $N/2 = 26$ contacts and it gets worse at $N = 53$ and especially at $2N = 106$.


PPV at N/2 for Family PF00014
PPV(N/2)
H
d=5
d=10
d=15
d=20
d=25

(a)


PPV at N for Family PF00014
PPV(N)
H
d=5
d=10
d=15
d=20
d=25

(b)



(c)

Figure 5.1: Positive Predictive Value evaluated at (a) $N/2$, (b) N and (c) $2N$ for increasing values of d as a function of the number of attention heads H for protein family PF00014.

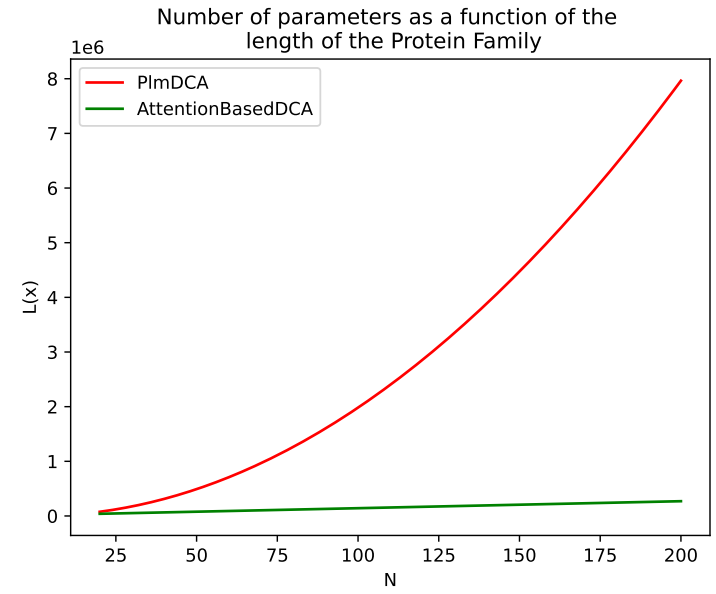
In the following we shall fix the number of attention heads and the lower dimension to $H = 32$ and $d = 20$. This combination produces a significant compression in the number of parameters with respect to PlmDCA. Figure 5.2a shows how the number of parameters in *AttentionBasedDCA* grows linearly with the length of the protein family, in contrast with PlmDCA where the growth is quadratic. In order to quantify the parameter reduction of the two models, we define a *compression ratio*

$$C_{ratio}(N) = \frac{L_{Att}(N)}{L_{Plm}(N)}, \tag{5.8}$$

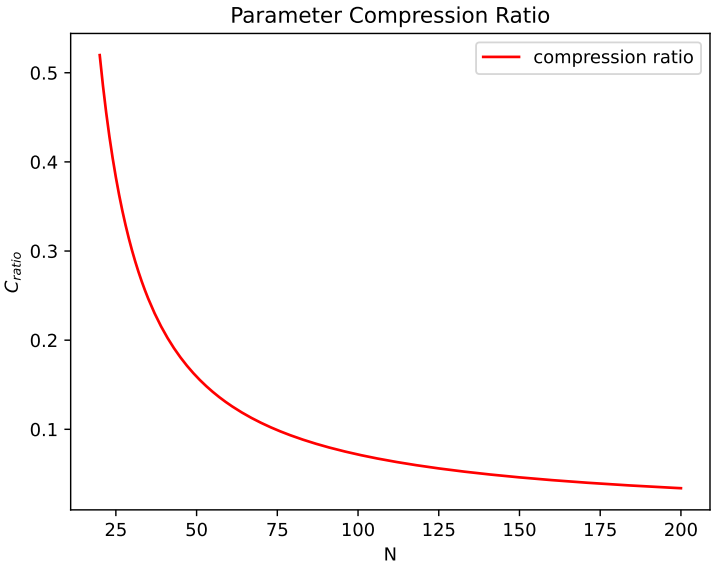
where $L_{Att}(N)$ and $L_{Plm}(N)$ are the number of parameters in *AttentionBasedDCA* and PlmDCA respectively. In particular, for $N = 53$ we have $C_{ratio} =$

0.15, which means that only 15% of the number of the parameters of PlmDCA are used in *AttentionBasedDCA*. For higher values of N, the ratio decreases even more. Figure 5.2b shows the compression ratio as a function of the length of the MSA.

Finally, following what we did in section 3.3, we present the PPV curve and the Contact Plot evaluated by *AttentionBasedDCA* on protein families PF00014, PF00595 and PF13354. In particular, for each family the PPV curve of the model is compared to that of PlmDCA and to an ideal model which predicts all and only true contacts. Figures 5.3a, 5.3c, 5.3e show the PPV curves, while figures 5.3b, 5.3d and 5.3f the contact maps evaluated at $x = N$. The known structure of the family is represented by grey dots, while positive and negative contacts are represented by blue and red dots respectively. Even though PlmDCA performs systematically better than *AttentionBasedDCA*, it is clear that the general predictive power of the latter is of the same order of magnitude of the former's. This is particularly true for protein family PF00014 on which the two models are essentially indistinguishable. However, the accuracy drops of about 20% in the cases of PF00595 and PF13354. This may be due a poor evaluation of the hyper-parameters. In particular, the lower dimension $d = 20$ might be too small to encode the whole information of protein family significantly longer than PF00014. In the future, this effect could be corrected by learning a mapping between the length of the sequences and the optimal value of the lower dimension d. Nevertheless, this must be done maintaining the desired linear scaling law of the parameters with N, which is the actual fundamental strength of the present architecture.

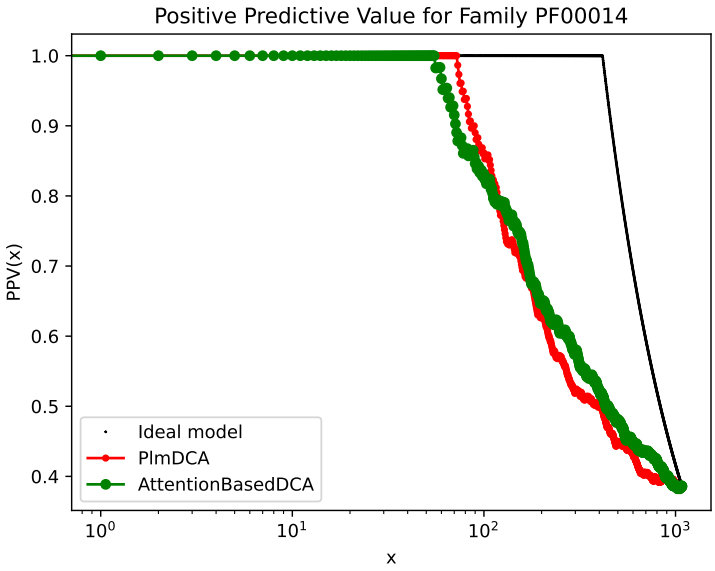


(a)

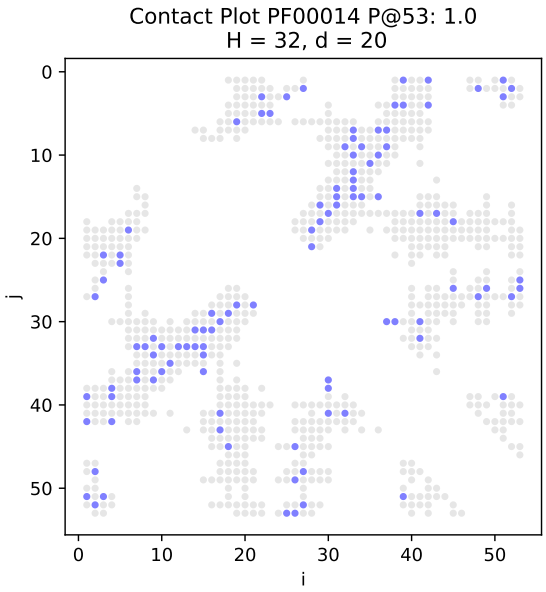


(b)

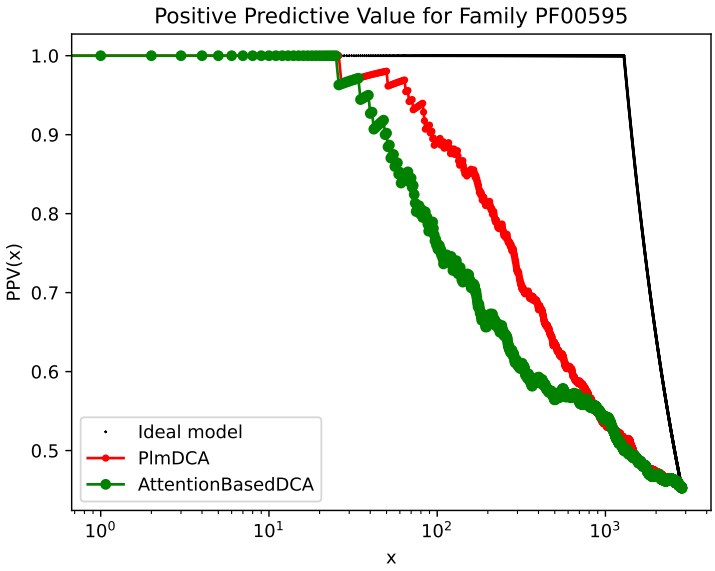
Figure 5.2: Number of parameter in AttentionBasedDCA and PlmDCA (a) and their ratio (b) as a function of the length of the protein family MSA.



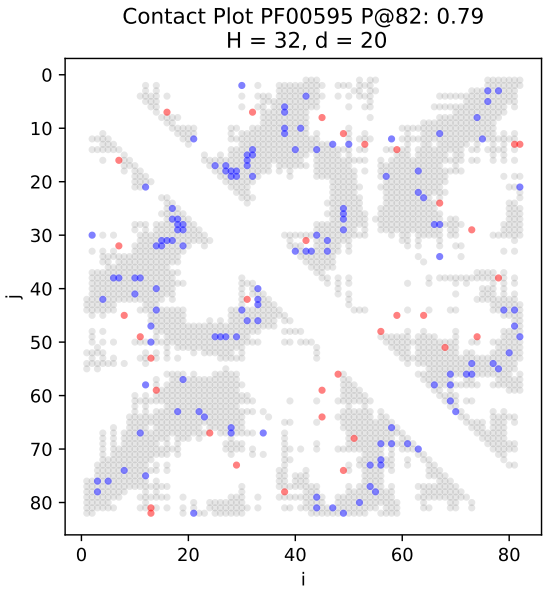
(a) PPV Family PF00014



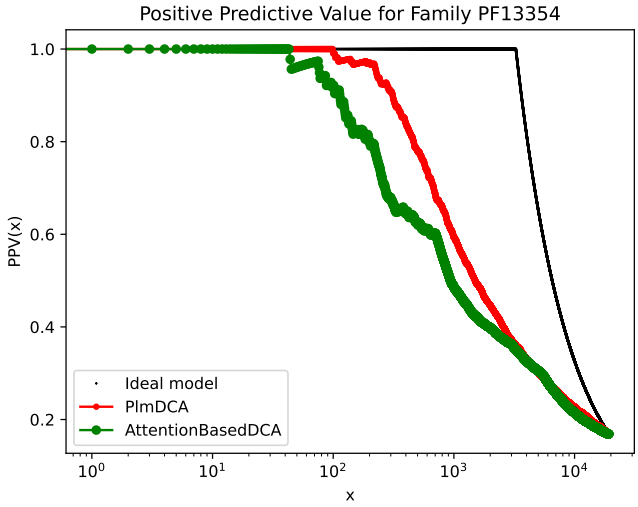
(b) Contact Plot Family PF00014



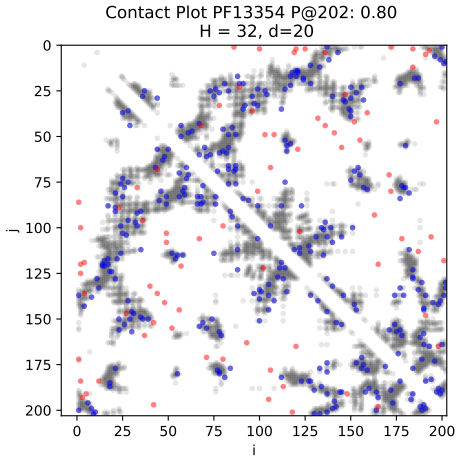
(c) PPV Family PF00595



(d) Contact Plot Family PF00595



(e) PPV Family PF13354



(f) Contact Plot Family PF13354

Figure 5.3: (a), (c), (e): Positive Predictive Value as a function of the number of predicted contacts. The green, red and black curve represent respectively the PPV of AttentionBasedDCA, PlmDCA and that of an ideal model that correctly predicts all and only true contacts. (b), (d), (f): Graphical representation of the symmetric Contact Matrix, where grey dots correspond to the known structure of the family, while blue and red dots correspond to true and false predictions by AttentionBasedDCA respectively. P@N represents the PPV evaluated at N.

Chapter 6

Conclusions

In this thesis we discussed the fundamental aspects of Direct Coupling Analysis, the state-of-the-art statistical model for protein contact predictions and structural analysis based on the conservation of particular patterns in phylogenetically related amino-acid sequences constituting a protein family. Furthermore, we presented the Attention Mechanism in the various forms in which it was conceived and developed in the context of Natural Language Processing, focusing on the implementation of the Transformer and the Query-Key-Value formalism. Inspired by this, we developed *AttentionBasedDCA*, a particular implementation of a pseudo-loglikelihood maximisation DCA where the coupling tensor is written as a non-linear low-rank decomposition. This choice is motivated by biological aspects such as the necessity to lower the number of parameters in the interaction tensor in order to mimic the linear scaling of the contacts with the length of the sequence [32]. In particular, using a small fraction of the parameters used in standard PlmDCA, our model reaches an accuracy on contact predictions which is comparable to that of PlmDCA for medium-small sequences.

6.1 Future Developments

6.1.1 Optimisation

Compared to PlmDCA, the computations needed for maximising the pseudo-likelihood for the total length of the sequence is several order of magnitude slower. Apart from a possible sub-optimal implementation of the code, this slowness depends on the increased computational complexity of the gradients along with the impossibility of a complete parallelisation over the positions of the sequence. A possible development of this work could be that of investigating alternative variations of this architecture, more suitable to parallelisation. An example of this is replacing the Value matrix $\boldsymbol{V} \in \mathbb{R}^{q,q}$ with a position-dependent matrix $\tilde{\boldsymbol{V}} = (V_{i,\alpha,\beta})^{i=1,\dots,N}_{\alpha,\beta=1,\dots,q}$. This would solve the parallelisation problem, maintaining at the same time the number of parameters of the order of $\mathcal{O}(N)$.

6.1.2 Auto-regressive Generative Model

Following a recent line of research [33], a further improvement of the architecture would be that of turning it into an auto-regressive generative model. In general, given a set of observation $\boldsymbol{X} = (X_{ij})^{j=1,\dots,M}_{i=1,\dots,N}$, a generative model learns a probability distribution $P(\boldsymbol{x}|\boldsymbol{X})$ over the space $(x_1, x_2, \dots, x_N)$, with the goal of using it to sample new points which are statistically indistinguishable from the observed ones [34]. In the context of protein analysis, this would correspond to generating artificial sequences which carry the relevant information of the specific protein family used to train the model. Remarkably, the ability to sample new sequences with determined characteristics possibly originating from different protein families suggests the potential to solve the Protein Design Problem, with limitless implications in both research and medicine. Moreover, the generation of artificial sequences would provide a new method to test the accuracy of the model. Indeed, looking for ways to discriminate between natural and artificial sequences would provide a benchmark for the effectiveness of the

statistical modellisation.

The path to make our architecture a generative model is quite straightforward as it is already designed to learn the parameters of the probability distribution over the space of protein sequences

$$P(\mathbf{A}) = \frac{1}{Z} \exp\Big\{\sum_{i<j} J_{ij}(A_i, A_j)\Big\}, \tag{6.1}$$

where

$$J_{ij}(A_i, A_j) = \sum_{h=1}^{H} \text{softmax}(\boldsymbol{Q}^h \boldsymbol{K}^{h^T})_{ij} \boldsymbol{V}^h_{A_i, A_j} \tag{6.2}$$

and Z is the normalisation constant. The generative character of the model depends on the particular technique used to sample over $P(\boldsymbol{A})$. An efficient approach exploits Bayes' theorem and the chain rule for probability [34] in order to write the N-dimensional distribution using the exact factorisation

$$P(\boldsymbol{A}) = \prod_{i=1}^{N} P(A_i | A_{i-1}, A_{i-2}, \ldots, A_1), \tag{6.3}$$

where $P(A_i | A_{i-1}, A_{i-2}, \ldots, A_1)$ is the probability distribution for amino-acid in position i, conditioned to all previous positions. Such a model is define an *auto-regressive* model, adopting a terminology originating from the literature on time-series models where previous time-steps observations of a system are used to predict its current state. In our case, we fix an ordering for the variables so that the distribution of the i^{th} random variable depends on the values of all the preceding ones in the chosen ordering. In particular, we may use a parameterisation of the conditional distributions which follows that of equation 6.1:

$$\begin{aligned} P(A_i | A_{i-1}, A_{i-2}, \ldots, A_1) &= \frac{\exp\Big\{\sum_{j=1}^{i-1} J_{ij}(A_i, A_j)\Big\}}{z_i(A_{i-1}, A_{i-2}, \ldots, A_1)} \\ &= \frac{\exp\Big\{\sum_{j=1}^{i-1} J_{ij}(A_i, A_j)\Big\}}{\sum_{A_i=1}^{q} \exp\Big\{\sum_{j=1}^{i-1} J_{ij}(A_i, A_j)\Big\}}, \end{aligned} \tag{6.4}$$

where at each step the normalisation factor z_i can be computed in polynomial time $\mathcal{O}(q)$, in contrast with the normalisation constant of the full distribution which can only be compute in exponential time $\mathcal{O}(q^N)$.

Finally, using an auto-regressive model of this kind turns the unsupervised task of learning the full probability distribution from a Multiple Sequence Alignment into a task of supervised learning, where $(A_{i-1}, \ldots, A_1)$ is the input feature vector and A_i represents the output label. This opens the way to the implementation of possible new techniques coming from the much wider literature on supervised learning [33].

6.1.3 Inter-Attention for Protein-Protein Interactions

In this thesis we presented the Attention Mechanism as a set of techniques used to learn relationships of various nature depending on the specific input data under investigation. In particular, we focused on general-purpose attention and self- or intra-attention which is the version on which we based *AttentionBasedDCA*. In the context of proteins, self-attention is used to learn the specific *language* of a family, i.e. the patterns that characterise phylogenetically related homologous sequences.

A possible future line of research is that of implementing an inter-attention based statistical modelling aimed at studying Protein-Protein Interactions (PPI). In order to carry out their biological function, proteins tend to aggregate in networks by means of physical interactions [35], [36]. As in residue-residue contacts, protein-protein interaction mechanisms are conserved across different species due to the high degree of co-evolution between residues at the interface of partner proteins. Following a linguistic analogy, an inter-attention model for the statistical analysis of PPI would be equivalent to a translation process where proteins of a given family are associated with their partner proteins from another family. In our architecture, this could be implemented by allowing the Value matrix $\boldsymbol{V}$ to learn across different families and introducing family-specific Queries and Keys matrices.

Bibliography

[1] L. Pauling, R. B. Corey, and H. R. Branson, "The structure of proteins: Two hydrogen-bonded helical configurations of the polypeptide chain," *Proceedings of the National Academy of Sciences*, vol. 37, pp. 205–211, Apr. 1951. Publisher: Proceedings of the National Academy of Sciences.

[2] J. Boyle, "Lehninger principles of biochemistry (4th ed.): Nelson, D., and Cox, M.," *Biochemistry and Molecular Biology Education*, vol. 33, no. 1, pp. 74–75, 2005. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bmb.2005.494033010419.

[3] S. L. Bardy, S. Y. M. Ng, and K. F. Jarrell, "Prokaryotic motility structures," *Microbiology*, vol. 149, pp. 295–304, Feb. 2003.

[4] P. C. Nelson, M. Radosavljevic, and S. Bromberg, *Biological physics: energy, information, life*. New York: W.H. Freeman and Co., 2008. OCLC: 181591287.

[5] C. Levinthal, "Are there pathways for protein folding?," *Journal de Chimie Physique*, vol. 65, pp. 44–45, Jan. 1968. ADS Bibcode: 1968JCP....65...44L.

[6] K. Sneppen and G. Zocchi, *Physics in molecular biology*. Cambridge, UK; New York: Cambridge University Press, 2006. OCLC: 76785875.

[7] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure

prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, Aug. 2021. Number: 7873 Publisher: Nature Publishing Group.

[8] D. Baker, "An exciting but challenging road ahead for computational enzyme design," *Protein Sci*, vol. 19, pp. 1817–1819, Oct. 2010.

[9] J. Karanicolas and B. Kuhlman, "Computational Design of Affinity and Specificity at Protein-Protein Interfaces," *Curr Opin Struct Biol*, vol. 19, pp. 458–463, Aug. 2009.

[10] S. Cocco, C. Feinauer, M. Figliuzzi, R. Monasson, and M. Weigt, "Inverse statistical physics of protein sequences: a key issues review," *Rep. Prog. Phys.*, vol. 81, p. 032601, Jan. 2018. Publisher: IOP Publishing.

[11] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1 ed., Apr. 1998.

[12] J. Tubiana, S. Cocco, and R. Monasson, "Learning protein constitutive motifs from sequence data," *eLife*, vol. 8, p. e39397, Mar. 2019. Publisher: eLife Sciences Publications, Ltd.

[13] E. T. Jaynes, "Information Theory and Statistical Mechanics," *Phys. Rev.*, vol. 106, pp. 620–630, May 1957. Publisher: American Physical Society.

[14] F. Morcos, A. Pagnani, B. Lunt, A. Bertolino, D. S. Marks, C. Sander, R. Zecchina, J. N. Onuchic, T. Hwa, and M. Weigt, "Direct-coupling analysis of residue coevolution captures native contacts across many protein families," *Proc Natl Acad Sci U S A*, vol. 108, pp. E1293–E1301, Dec. 2011.

[15] V. Sessak and R. Monasson, "Small-correlation expansions for the inverse Ising problem," *J. Phys. A: Math. Theor.*, vol. 42, p. 055001, Feb. 2009. arXiv:0811.3574 [cond-mat].

[16] D. S. Marks, L. J. Colwell, R. Sheridan, T. A. Hopf, A. Pagnani, R. Zecchina, and C. Sander, "Protein 3D Structure Computed from Evolutionary Sequence Variation," *PLoS One*, vol. 6, p. e28766, Dec. 2011.

[17] R. J. Rossi, *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons Inc, 2018. OCLC: 1027728548.

[18] M. Ekeberg, C. Lövkvist, Y. Lan, M. Weigt, and E. Aurell, "Improved contact prediction in proteins: Using pseudolikelihoods to infer Potts models," *Phys. Rev.*

E, vol. 87, p. 012707, Jan. 2013. arXiv:1211.1281 [cond-mat, physics:physics, q-bio].

[19] M. Ekeberg, T. Hartonen, and E. Aurell, "Fast pseudolikelihood maximization for direct-coupling analysis of protein structure from many homologous amino-acid sequences," *Journal of Computational Physics*, vol. 276, pp. 341–356, Nov. 2014. arXiv:1401.4832 [physics, q-bio].

[20] D. W. Otter, J. R. Medina, and J. K. Kalita, "A Survey of the Usages of Deep Learning in Natural Language Processing," Tech. Rep. arXiv:1807.10854, arXiv, Dec. 2019. arXiv:1807.10854 [cs] type: article.

[21] P. Semaan, "Natural Language Generation: An Overview," *Journal of Computer Science*, vol. 1, no. 3, p. 8, 2012.

[22] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," Tech. Rep. arXiv:1409.3215, arXiv, Dec. 2014. arXiv:1409.3215 [cs] type: article.

[23] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," Tech. Rep. arXiv:1406.1078, arXiv, Sept. 2014. arXiv:1406.1078 [cs, stat] type: article.

[24] J. Cheng, L. Dong, and M. Lapata, "Long Short-Term Memory-Networks for Machine Reading," Sept. 2016. arXiv:1601.06733 [cs].

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Tech. Rep. arXiv:1706.03762, arXiv, Dec. 2017. arXiv:1706.03762 [cs] type: article.

[26] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," Tech. Rep. arXiv:1409.0473, arXiv, May 2016. arXiv:1409.0473 [cs, stat] type: article.

[27] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, Sept. 2021.

[28] J. Vig, A. Madani, L. R. Varshney, C. Xiong, R. Socher, and N. F. Rajani, "BERTology Meets Biology: Interpreting Attention in Protein Language Models," Tech. Rep. arXiv:2006.15222, arXiv, Mar. 2021. arXiv:2006.15222 [cs, q-bio] type: article.

[29] D. Ofer, N. Brandes, and M. Linial, "The language of proteins: NLP, machine learning & protein sequences," *Comput Struct Biotechnol J*, vol. 19, pp. 1750–1758, 2021.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Tech. Rep. arXiv:1810.04805, arXiv, May 2019. arXiv:1810.04805 [cs] type: article.

[31] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik, and B. Rost, "ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[32] N. Bhattacharya, N. Thomas, R. Rao, J. Dauparas, P. K. Koo, D. Baker, Y. S. Song, and S. Ovchinnikov, "Single Layers of Attention Suffice to Predict Protein Contacts," tech. rep., bioRxiv, Dec. 2020. Section: New Results Type: article.

[33] J. Trinquier, G. Uguzzoni, A. Pagnani, F. Zamponi, and M. Weigt, "Efficient generative modeling of protein sequences using simple autoregressive models," *Nat Commun*, vol. 12, p. 5800, Oct. 2021. Number: 1 Publisher: Nature Publishing Group.

[34] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009. Google-Books-ID: eBSgoAEACAAJ.

[35] C. Feinauer, H. Szurmant, M. Weigt, and A. Pagnani, "Inferring protein-protein interaction networks from inter-protein sequence co-evolution," preprint, Bioinformatics, Dec. 2015.

[36] C. Feinauer, H. Szurmant, M. Weigt, and A. Pagnani, "Inter-Protein Sequence Co-Evolution Predicts Known Physical Interactions in Bacterial Ribosomes and the Trp Operon," *PLOS ONE*, vol. 11, p. e0149166, Feb. 2016. Publisher: Public Library of Science.