CONTENTS

1. INTRODUCTION

Proteins are biomolecules that are fundamental to nearly all biological processes. Their diverse roles include transporting nutrients, catalyzing chemical reactions, providing structural support, and more. The function of a protein is determined by the composition of its primary sequence created from amino acids. A single protein sequence can vary greatly in length and order of its amino acids, leading to a very large number of possible configurations. The massive size of the protein sequence space makes exhaustive experimental exploration infeasible.

Advances in sequencing technologies has ushered in a rapid growth of biological databases, transforming biology into a data-rich discipline. Large repositories such as UniProt, Pfam, and the Protein Data Bank store millions of sequences and structures, providing crucial resources for computational approaches [1]. This wealth of data has enabled the development of advanced statistical and machine learning models capable of simulating protein sequence evolution, predicting structural conformations, and generating novel sequences with desired properties. In addition, breakthroughs in protein structure prediction–most notably the Nobel Prize winning AlphaFold [2]– show that computational methods can rival experimental accuracy, in a much cheaper manner.

In this work, we will implement arDCA, an efficient autoregressive network for protein sequence generation leveraging the Direct Coupling Analysis (DCA) method in Python. The biological background will be introduced in Section 2. The mathematical foundations will be laid out in Section 3. Section 4 will describe previous iterations of Direct Coupling Analysis methods and arDCA, while Section 5 will explore our implementation.

2. Biological Background

2.1. Proteins and their structure

Proteins are essential biological molecules responsible for a wide range of functions in living organisms. Despite their functional diversity, all proteins are polymers of the same set of standard building blocks: the 20 canonical amino acids. Amino acids share a common core structure consisting of a central carbon atom bonded to a hydrogen atom, an amino group, a carboxyl group, and a variable side chain. The side chain is the defining feature of an amino acid, giving rise to differences in size, shape, chemical reactivity, and polarity. Proteins, or polypeptides, are formed through peptide bonds of distinct amino acids. Certain atoms are lost in this process, thus within a protein sequence, individual amino acids are typically referred to as residues. Generally, protein sequences are made up of between 50 and 2000 amino acids. The ordering of the amino acids dictates how a protein folds into its three-dimensional structure, known as its conformation. Although each conformation is unique, two common folding patterns occur in many proteins: the α helix and the β sheet [3].
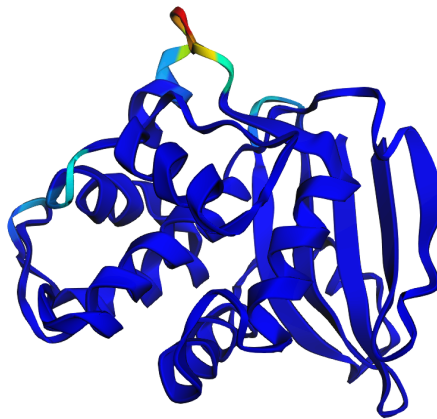

Figure 1: Protein from PF13354 folded using `ColabFold` [4]

Each amino acid is chemically distinct and can occur at any position in a protein chain, giving rise to 20^n possible polypeptide sequences of length n. For a typical protein of 300 amino acids, the number of possible sequences is astronomically large ($\approx 2 \cdot 10^{390}$).

However, only a small fraction of these sequences are capable of folding into a stable three-dimensional conformation.

2.2. Protein families and evolutionary information

Proteins do not evolve in isolation; they often belong to protein families, groups of proteins that share a common evolutionary origin, therefore exhibit similar sequence features and functional properties [5]. Over evolutionary timescales, mutations accumulate in these families: some are beneficial, altering the protein activity in ways that give rise to new functions, while many others are neutral and have no effect on stability or activity. Harmful mutations, by contrast, disrupt protein folding or its function. These destructive changes are eliminated through natural selection. The result is a collection of homologous proteins that retain overall structural and functional characteristics, but also display sequence variability that encodes the evolutionary history of the family [3].

The study of evolutionary history and relationships among biological entities is referred to as phylogenetics. Protein families provide the domain for phylogenetic analysis, as examining families provides insight that cannot be obtained from a single sequence [6]. From homologous proteins, we can detect important correlated mutations between amino acid positions which represents constraints enforced to maintain protein integrity by evolution. These statistical patterns are crucial for computational approaches which learn them to predict three-dimensional structure and understand protein function.

2.3. Multiple sequence alignments

To extract important statistical information from protein families, the homologous sequences need to be organized in a systematic way. This is done through a multiple sequence alignment (MSA), in which three or more sequences are arranged so that homologous sites are placed in the same column [7]. Alignment gaps are introduced throughout the MSA to maximize the positional correspondence of sequences and enable them to have the same fixed length.
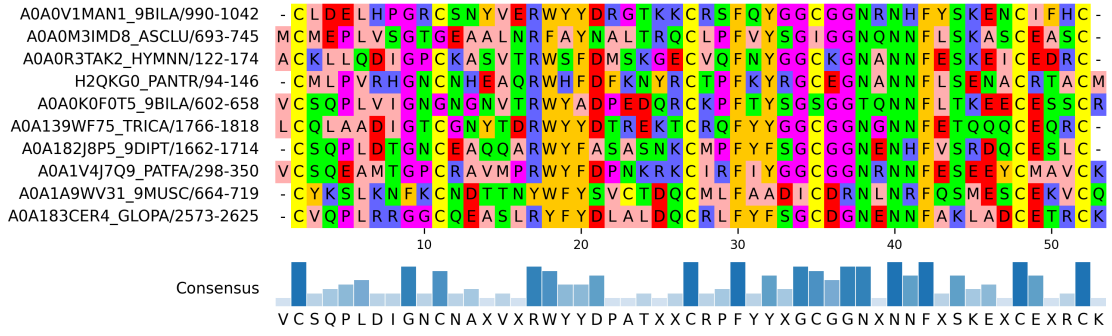
Figure 2: MSA of first 10 proteins in PF00014, created using `pyMSAviz` [8]

Aligning sequences in this manner, patterns of conservation and variation are revealed across the family. Conserved positions, those which are unchanged in multiple sequences, represent sites that are critical for maintaining structure or function, while variable positions indicate sites that can tolerate mutations without disruption to the protein. Beyond conservation, MSAs also capture covariation: pairs of positions that mutate in a correlated way across sequences. These covariation signals reflect couplings, where a mutation at one site requires compensation at another to maintain protein integrity [9].

2.4. PROTEIN CONTACTS

One of the earliest challenges for researchers in computational biology–historically encouraged by the CASP (Critical Assessment of Structure Prediction) [10] competition–was understanding how the linear sequence of amino acids folds into its conformation. A key insight was the role of contacts, pairs of residues that, while possibly far apart in the linear sequence, end up spatially close in the folded three-dimensional structure. Since structures can be represented in Cartesian coordinates (x, y, z), contacts are usually defined by distance thresholds, with two residues considered in contact if the distance between the selected atoms is below a set threshold, usually 8 Ångströms (Å). These contacts are further categorized into short, medium, and long range, with long-range contacts playing a disproportionately important role in determining the fold. Importantly, this illustrates that correlations in protein sequences are not limited to

6

neighboring sites along the chain, many of the most informative dependencies span large separations [11].

2.5. The problem of inference

The central challenge in protein sequence analysis is disentangling the true structural and functional constraints of protein families from spurious correlations introduced by other factors. Correlations can arise indirectly, for example, if residue A correlates with B, and B with C, then A and C may appear correlated without a direct interaction. Moreover, shared evolutionary history (phylogeny) and biases in sequence databases can further obscure the true couplings that underlie protein folding and function [12]. Addressing this problem requires a mathematical framework capable of distinguishing direct from indirect correlations in the data.

3. Mathematical Foundations

3.1. Proteins as statistical systems

Protein sequences can be thought of as random variables produced by a certain distribution. Each sequence of length L can be written as:

$$\boldsymbol{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_L), \quad \sigma_i \in \mathcal{A}, \tag{1}$$

where $\mathcal{A}$ is the alphabet of size $q = 21$ (20 amino acids and the alignment gap) and σ_i are the residue sites. We can organize these sequences into multiple sequence alignments, a table $\left\{\boldsymbol{\sigma}^{(m)}\right\}_{m=1}^{M}$ of M empirical samples. These samples are aligned to have a common length L. Each row in the MSA represents a protein, and each column a position, or site, in the sequence. From the alignments, we can define single and pairwise frequency counts for the columns. The single-site frequency for MSA column i can be computed as:

$$f_i(A) = \frac{1}{M} \sum_{m=1}^{M} \delta\left(\sigma_i^{(m)}, A\right), \quad A \in \mathcal{A}, \quad \text{where } \delta \text{ is the Kronecker delta}[1] \tag{2}$$

---

[1] $\delta(x, y) := \begin{cases} 0 \text{ if } x \neq y \\ 1 \text{ if } x = y \end{cases}$

while the pairwise frequency of MSA columns i, j is computed as:

$$f_{ij}(A, B) = \frac{1}{M} \sum_{m=1}^{M} \delta\left(\sigma_i^{(m)}, A\right) \delta\left(\sigma_j^{(m)}, B\right) \tag{3}$$

The empirical frequencies will serve as constraints for the distribution we want to infer.

3.2. Maximum Entropy Principle

To find the probability distribution $P(\boldsymbol{\sigma})$ that will satisfy constraints, in other words reproduce the empirical marginals $f_i(A)$ and $f_{ij}(A, B)$, we can use the maximum entropy principle [13], [14]. The first step to set up the MEP is extracting information from the system. Usually, this information is given in the form of averages of functions $\langle f_k(x) \rangle$. For example, in a physical system, one could compute average magnetization or energy of the observed system. We also need to define a probability of occupancy of states, $p(x)$, which runs over all the possible states. This distribution satisfies the standard properties of a probability distribution: for all states x, $0 \leq p(x) \leq 1$, and the distribution is normalized, $\sum_x p(x) = 1$ [15]. Our uncertainty on the system is expressed quantitatively through Shannon entropy S [16]:

$$S = -\sum_x p(x) \ln p(x) \tag{4}$$

In general, the distribution that maximizes entropy under no constraints is the uniform distribution. However, once constraints are imposed, the uniform is no longer suitable. Our system is constrained by the probability distribution's normalization condition and the expectation constraints:

$$\sum_x p(x) = 1, \quad \text{and} \quad \sum_x p(x) f_k(x) = \langle f_k \rangle \quad (k = 1, ..., m), \tag{5}$$

where each $f_k(x)$ represents a feature, and $\langle f_k(x) \rangle$ its empirical average. By selecting Shannon entropy as our measure of information, we can introduce Lagrange Multipliers $\lambda_0, \lambda_1, ... \lambda_m$ to maximize Eq. 4 subject to the constraints defined in Eq. 5, yielding:

$$p(x) = \exp\left(-\lambda_0 - \sum_{k=1}^{m} \lambda_k f_k(x)\right). \tag{6}$$

Define the partition function Z as:

$$Z(\lambda) = \sum_{x} \exp\left(-\sum_{k=1}^{m} \lambda_k f_k(x)\right). \tag{7}$$

With normalization $\lambda_0 = \ln Z$, we can write the moments as:

$$\langle f_k \rangle = -\frac{\partial}{\partial \lambda_k} \ln Z(\lambda) \tag{8}$$

The maximum entropy of the distribution then reduces to[2]:

$$S_{\text{max}} = \lambda_0 + \sum_{k=1}^{m} \lambda_k \langle f_k \rangle \tag{9}$$

When the constraints are chosen to match the empirical single-site and pairwise frequencies of the MSA, the maximum-entropy (MaxEnt) distribution reduces to the Potts model.

3.3. CONNECTION TO STATISTICAL MECHANICS

The Potts model is rooted in statistical mechanics, where it was introduced as a generalization of the Ising spin model. Both models were developed to study ferromagnetism and phase transitions.

---

[2]The full derivation can be found in Appendix A.

3.3.1. ISING MODEL

The Ising model [17] describes a system of spins σ_i arranged on a lattice. Each spin can take one of two possible values:

$$\sigma_i \in \{+1, -1\}. \tag{10}$$

The Hamiltonian function, which represents the energy, of a spin configuration $\{\sigma\}$ is

$$H_{\text{Ising}}(\{\sigma\}) = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j - h \sum_i \sigma_i, \tag{11}$$

where J is the coupling constant calculated over nearest-neighbor pairs (represented by $\langle i, j \rangle$) that defines the strength between paired interactions. The h term is the external magnetic field acting on the spins. At thermal equilibrium, the probability of a configuration is given by the Boltzmann distribution:

$$P(\{\sigma\}) = \frac{1}{Z} \exp\big(-\beta H_{\text{Ising}}(\{\sigma\})\big), \tag{12}$$

with $\beta = 1/(k_b T)$ and partition function

$$Z = \sum_{\{\sigma\}} \exp\big(-\beta H_{\text{Ising}}(\{\sigma\})\big). \tag{13}$$

3.3.2. POTTS MODEL

The Potts model [18] generalizes the Ising model by allowing each spin to take on q possible states. Its Hamiltonian is

$$H_{\text{Potts}}(\{\sigma\}) = -J \sum_{\langle i,j \rangle} \delta\big(\sigma_i, \sigma_j\big). \tag{14}$$

where J is the ferromagnetic coupling that encourages neighboring spins to align in the same state. While the original formulation did not include external fields, they can be

incorporated into the Hamiltonian to allow for site-dependent biases. The corresponding partition function is

$$Z = \sum_{\{\sigma\}} \exp(-\beta H_{\text{Potts}}(\{\sigma\})). \tag{15}$$

For $q = 2$, the Potts model reduces to the Ising model. The probability distribution over spin configurations is then given by

$$P(\sigma) = \frac{1}{Z} \exp(-\beta H_{\text{Potts}}(\{\sigma\})). \tag{16}$$

It is important to note that the Potts model is over-parametrized: multiple choices of (h, J) yield the same probability distribution. As a result, the parameters are not uniquely identifiable without fixing a gauge. This ambiguity can make the norms of J misleading and complicate optimization. Several gauge conventions are commonly used–such as the reference-state gauge, the zero-sum gauge, or implicit definitions through regularization–which will be discussed in detail in the following sections. In the context of protein sequence analysis, the Potts model

3.4. DIRECT COUPLING ANALYSIS

The statistical dependencies observed in MSAs encode important information about protein structure and function. However, naive measures such as covariance cannot distinguish between direct correlations, which arise from true physical or functional contacts, and indirect correlations, propagated via other residues. Direct Coupling Analysis (DCA) addresses this problem by introducing a method to separate direct correlations.

DCA assumes that sequences in the MSA are independent realizations of a Potts model distribution:

$$P(\boldsymbol{\sigma}) = \frac{1}{Z} \exp \left(\sum_i h_i(\sigma_i) + \sum_{i<j} J_{ij}(\sigma_i, \sigma_j) \right) \tag{17}$$

where h_i are site-specific fields and J_{ij} are coupling parameters between residue pairs. The central goal of DCA is to infer the interaction parameters J_{ij} that best explain the observed single- and pairwise frequencies. In practice, direct inference of these parameters is computationally challenging. Evaluating the partition function Z is intractable for realistic proteins, as it requires summing over q^L possible sequences. Thus, distinct implementations of DCA introduce approximations to bypass this computation. The original message passing implementation [19], directly attempted to solve for couplings through a slowly converging iterative scheme, making it inapplicable in practice. In this paper, we will explore some important algorithmic innovations in DCA following its conception.

4. EVOLUTION OF DCA METHODS

4.1. MEAN-FIELD DCA

The mean-field Direct Coupling Analysis (mfDCA) algorithm, introduced by Morcos et al. [20], provided the first computationally feasible approximation of the Potts model. The idea is to approximate a small-coupling expansion, valid when correlations between sites are weak which reduces the inference problem to the inversion of a correlation matrix.

4.1.1. METHOD

To begin, raw frequency counts in an MSA are affected by sampling bias, since closely related sequences contribute redundant information. To mitigate this, a sequence reweighting procedure is applied. Each sequence $\sigma^{(a)}$ is assigned a weight

$$w_m = \frac{1}{\left| \{ b \in \{1, ..., M\} : \mathrm{sim}(\sigma^{(a)}, \sigma^{(b)}) \geq x \} \right|}, \tag{18}$$

where M is the number of sequences in the MSA, and $\text{sim}(\cdot, \cdot)$ denotes the pairwise sequence similarity[3]. Intuitively, if a sequence is similar to many others, with a similarity greater than the threshold x, its contribution is down-weighted; if it is unique, it retains full weight. Through empirical tests $x = 0.8$ was found to be most effective. The weight of a sequence m is w_m, and the effective number of sequences is

$$M_{\text{eff}} = \sum_{m=1}^{M} w_m. \tag{19}$$

From these weights, the new regularized empirical single and pairwise frequencies are computed as

$$f_i(A) = \frac{1}{M_{\text{eff}} + \lambda} \left(\frac{\lambda}{q} + \sum_{m=1}^{M} w_m \delta\left(A, \sigma_i^{(m)}\right) \right)$$

$$f_{ij}(A, B) = \frac{1}{M_{\text{eff}} + \lambda} \left(\frac{\lambda}{q^2} + \sum_{m=1}^{M} w_m \delta\left(A, \sigma_i^{(m)}\right) \delta\left(B, \sigma_j^{(m)}\right) \right) \tag{20}$$

where $q = 21$ is the amino acid alphabet with the gap symbol, and λ is a pseudocount parameter used for regularization. To address the overparametrization of the Potts, mfDCA employs a reference-state gauge, which removes redundancy among parameters while preserving all physically relevant couplings. Typically, it is set as the last amino acid $A = q$. Implementing this gauge gives us

$$\forall i, j : J_{ij}(a, q) = J_{ij}(q, a) = h_i(q) = 0 \tag{21}$$

Coupling Inference: Mean-field Approximation

To circumvent the intractable computation of the partition function Z, mfDCA assumes that the correlations between sites are weak. Under this approximation, the exponential

---

[3]The original paper used "seqid" for percentage identity. We adopt the notation "sim" as the future methods used this notation.

of the Potts defined in Eq. 17 can be expanded to first order in a Taylor series [21], [22]. This yields a relation between the couplings and the connected correlation matrix:

$$C_{ij}(A, B) = f_{ij}(A, B) - f_i(A)f_j(B). \tag{22}$$

In this framework, the couplings can be approximated through a naive mean-field inversion,

$$e_{ij}(A, B) = -\left(C^{-1}\right)_{ij}(A, B), \tag{23}$$

where C is treated as a $((q-1)L) \times ((q-1)L)$ matrix, and the pair (i, A) is regarded as a single index[4]. In practice, however, the correlation matrix is often singular or ill-conditioned, making the inversion unstable. To address this, mfDCA opts to use a strong pseudocount $(\lambda \approx M_{\text{eff}})$ ensuring numerical stability and preventing spurious large couplings.

Pair Scoring: Direct Information

Once the pairwise couplings $e_{ij}(A, B)$ are inferred, residue pairs must be ranked by their interaction strength. The $(q-1) \times (q-1)$-dimensional coupling matrices need to be mapped to a single scalar parameter. A natural starting point is mutual information (MI), which measures statistical dependence between two MSA columns. However, MI is not able to disentangle direct correlations. Therefore a new measure, direct information (DI) [19] was introduced. To begin, for each pair (i, j) a two-site model is constructed:

$$P_{ij}^{(\text{dir})}(A, B) = \frac{1}{Z_{ij}} \exp\left(e_{ij}(A, B) + \tilde{h}_i(A) + \tilde{h}_j(B)\right), \tag{24}$$

with auxiliary fields $\tilde{h}_i, \tilde{h}_j$ chosen such that the single-site marginals are preserved:

---

[4]A detailed derivation is provided in Appendix B

$$\sum_{B=1}^{q} P_{ij}^{(\mathrm{dir})}(A, B) = f_i(A), \quad \sum_{A=1}^{q} P_{ij}^{(\mathrm{dir})}(A, B) = f_j(B). \tag{25}$$

The direct information is then defined as the mutual information of this two-site model:

$$\mathcal{S}_{ij}^{\mathrm{DI}} = \sum_{A,B=1}^{q} P_{ij}^{(\mathrm{dir})}(A, B) \ln \frac{P_{ij}^{(\mathrm{dir})}(A, B)}{f_i(A) f_j(B)} \tag{26}$$

Residue pairs with the highest DI scores are predicted to be structural contacts.

4.1.2. LIMITATIONS

While mfDCA represented a breakthrough in usability of DCA, the simplifying approximations impose limitations on the model:

- Weak coupling assumptions: the small-coupling expasion assumes nearly linear correlations, which can underestimate strong epistatic effects in proteins [23].

- Computational scaling: the inversion of the correlation matrix scales as $\mathcal{O}(L^3)$, which is costly for very large MSAs.

- Pseudocount dependence: due to the algorithm's vast parameter size (in the order of $400N^2$) strong regularization is required. The choice of pseudocount λ can significantly affect performance.

4.2. PSEUDO-LIKELIHOOD MAXIMIZATION DCA

While mfDCA provided a fast approximation, it relied on strong assumptions. Pseudolikelihood maximization (plmDCA) relaxed these by fitting sitewise conditional probabilities directly. Concretely, the inverse Potts problem on an alignment of length L becomes L coupled multinomial logistic regressions rather than a single optimization over the global partition function.

4.2.1. METHOD

For plmDCA, the Potts parametrization of fields and couplings introduced in Eq. 17 is retained. Similar to mfDCA, sequences are reweighted using w_m defined in Eq. 18 and the effective sample size M_{eff} defined in Eq. 19.

Coupling Inference: Pseudolikelihood Maximization

The pivotal idea is to optimize pseudolikelihoods. For site r, the conditional distribution given all other sites $\sigma_{\backslash r}$ is

$$P\big(\sigma_r = A \mid \sigma_{\backslash r}\big) = \frac{\exp\Big(h_r(A) + \sum_{i \neq r} J_{ri}(A, \sigma_i)\Big)}{\sum_{B=1}^{q} \exp\Big(h_r(B) + \sum_{i \neq r} J_{ri}(B, \sigma_i)\Big)}). \tag{27}$$

The weighted sitewise negative log-pseudolikelihood is

$$g_r(h_r, J_r) = -\frac{1}{M_{\text{eff}}} \sum_{m=1}^{M} w_m \log P\Big(\sigma_r^{(m)} \mid \sigma_{\backslash r}^{(m)}\Big). \tag{28}$$

and the global objective aggregates these points:

$$\mathcal{L}_{\text{pseudo}}(h, J) = \sum_{r=1}^{L} g_r(h_r, J_r). \tag{29}$$

To curtail overfitting, we add convex l_2 penalties,

$$R_{\ell_2} = \lambda_h \sum_{r=1}^{L} \|h_r\|_2^2 + \lambda_J \sum_{1 \leq i < j \leq L} \|J_{ij}\|_2^2 \tag{30}$$

and minimize

$$\{h^{\text{PLM}}, J^{\text{PLM}}\} = \arg\min_{h,J}\Big\{\ell_{\text{pseudo}}(h, J) + R_{\ell_2}(h, J)\Big\} \tag{31}$$

The ℓ_2 penalty fixes the gauge implicitly by selecting a unique representative among gauge-equivalent parameters. Each g_r is precisely the loss of a multinomial logistic regression (softmax): classes are the q states of σ_r; features are one-hot encodings of $\{\sigma_i\}_{i \neq r}$ with $(L-1)(q-1)$ degrees of freedom after dropping a reference state. Hence (asymmetric) plmDCA is implemented as L independent, weighted softmax problems with ℓ_2 penalty. Two variants are used in practice: asymmetric PLM, which fits each independently and then symmetrizes averaging:

$$\hat{J}_{ij} \leftarrow \frac{1}{2}\left(J_{ij}^{(i)} + J_{ij}^{(j)}\right) \tag{32}$$

and symmetric (joint) PLM, which minimizes $\mathcal{L}_{\text{pseudo}}$ over all parameters at once. Results are typically similar, but optimization in the symmetric variant can be heavier.

Pair Scoring: Corrected Frobenius Norm

For pair scoring, plmDCA avoids the previously defined Direct Information (DI) as it would introduce a pseudocount regularization parameter, on top of the existing λ_h and λ_J. Instead, a new score is defined starting from the Frobenius Norm (FN)

$$\left\| J_{ij} \right\|_2 = \sqrt{\sum_{A,B=1}^{q} J_{ij}(A,B)^2}. \tag{33}$$

Unlike the DI score, the FN is not independent of gauge choice so a direct decision must be made. As noted in [19], the zero-sum gauge minimizes the FN, making it the most appropriate gauge choice available. The procedure is hence (i) convert to zero-sum gauge J'_{ij}:

$$J'_{ij}(A,B) = J_{ij}(A,B) - J_{ij}(\cdot,B) - J_{ij}(A,\cdot) + J_{ij}(\cdot,\cdot), \tag{34}$$

where "$\cdot$" denotes a simple average over the q states at that position; (ii) compute the Frobenius norm

$$\mathcal{S}_{ij}^{\mathrm{FN}} = \left\| J_{ij}' \right\|_2 = \sqrt{\sum_{A,B=1}^{q} \left(J_{ij}'(A,B) \right)^2}; \tag{35}$$

(iii) apply Average Product Correction (APC) adjusted from its use in [24] to reduce phylogenetic bias,

$$\mathcal{S}_{ij}^{\mathrm{CFN}} = \mathcal{S}_{ij}^{\mathrm{FN}} - \frac{\mathcal{S}_{i\cdot}^{\mathrm{FN}} \mathcal{S}_{\cdot j}^{\mathrm{FN}}}{\mathcal{S}_{\cdot\cdot}^{\mathrm{FN}}}, \tag{36}$$

where $\mathcal{S}_{i\cdot}^{\mathrm{FN}}$ and $\mathcal{S}_{\cdot j}^{\mathrm{FN}}$ are row/column means and $\mathcal{S}_{\cdot\cdot}^{\mathrm{FN}}$ is the grand mean. Residue pairs (i,j) are ranked by $\mathcal{S}_{ij}^{\mathrm{CN}}$ to predict structural contacts.

4.2.2. LIMITATIONS

Despite its impact for practical applications, plmDCA remains sensitive to sampling: accurate contact recovery still requires large M_{eff}, and sparse or biased MSAs degrade estimates. Phylogenetic and positional bias persist (reweighting and APC help but do not eliminate them), which can inflate false positives.

4.2.3. FAST PLMDCA

To make plmDCA deployable at scale, two of the original authors and a collaborator revisited its optimization strategy [25]. The revised method adopts an asymmetric variant, instead of optimizing a single global pseudolikelihood, it solves L independent, weighted softmax regressions–one per site. The resulting couplings are then symmetrized by averaging,

$$J_{ij} = \frac{1}{2} \left(J_{ij}^{(i)} + J_{ij}^{(j)} \right). \tag{37}$$

This decomposition reduces the dimensionality of each optimization problem and, crucially, enables trivial parallelization across CPU cores or nodes–the primary source of runtime gain. Since each J_{ij} is regularized twice (once in the regression for i and once for j), the coupling penalty parameter must be halved relative to the symmetric

formulation. Before averaging, the two coupling blocks are shifted into the zero-sum gauge to ensure consistency. For contact prediction, the method adopts the Corrected Frobenius Norm (CFN) scoring function defined above. The combination of asymmetric regression and symmetrization achieves accuracy comparable to the original plmDCA, but at a fraction of the runtime, making large protein families and long sequences tractable in practice.

4.3. Boltzmann Machine DCA

In mfDCA accuracy was traded for speed via the small-coupling inversion; in plmDCA, a product of sitewise conditionals was optimized, avoiding the global partition function. bmDCA [26] instead proposes a solution closer to the statistical ideal: fitting the full Potts model by maximizing the true likelihood, using Monte Carlo Markov Chains to estimate intractable expectations. The result is a generative model that (by construction) matches the empirical one- and two-site statistics of the reweighted MSA.

4.3.1. Method

bmDCA aims to infer the full Potts model defined as Eq. 17 that reproduces the single- and pair-wise frequencies of the MSA. The training procedure follows the classical Boltzmann machine learning algorithm:

1. Sampling step. For current parameters $\{J, h\}$, estimate model frequencies p_i, p_{ij} by MCMC

2. Update step. Adjust parameters when the estimated frequencies deviate from empirical ones: $\Delta h_i \propto f_i - p_i, \quad \Delta J_{ij} \propto f_{ij} - p_{ij}$

3. Iterate until empirical and model moments match within tolerance.

Regularization is imposed to stabilize the fit and avoid overfitting.

4.3.2. Limitations

- Computational cost: Direct Boltzmann machine learning with MCMC is extremely slow for realistic protein lengths ($10^7 - 10^9$ parameters).

- Sampling difficulty: Accurate moment estimation requires long chains and careful equilibration, making the approach impractical without approximations.
- Scaling: While bmDCA is theoretically optimal (full likelihood optimization), it is limited in practice to small systems where large-scale computation is feasible.

4.3.3. ADABMDCA

adabmDCA [27] the same goal of fitting the full Potts by likelihood maximization, but introduces an adaptive MCMC procedure to make the approach more practical. Instead of relying on fixed sampling parameters, it dynamically tunes equilibration, waiting times, and chain persistence. adabmDCA pushes the bmDCA approach closer to practical usability, at the cost of a more complex implementation and still significant compute requirements.

Coupling Infernce: Full Likelihood and Moment Matching

The maximum-entropy distribution that reproduces chosen moments is the Potts model defined in Eq. 17. To reduce phylogenetic redundancy, each sequence is assigned a weight w_m as in Eq. 18. Reweighted empirical frequencies, smoothed pseudocount λ are defined as

$$f_i(A) = (1 - \lambda)f_i^{\text{data}}(A) + \frac{\lambda}{q}, \quad f_{ij}(A, B) = (1 - \lambda)f_{ij}^{\text{data}}(A, B) + \frac{\lambda}{q^2}$$
$$(38)$$
$$f_i^{\text{data}}(A) = \frac{1}{M_{\text{eff}}} \sum_m w_m \delta\left(A, \sigma_i^{(m)}\right), \quad f_{ij}^{\text{data}}(A, B) = \frac{1}{M_{\text{eff}}} \sum_m w_m \delta\left(A, \sigma_i^{(m)}\right)\left(B, \sigma_j^{(m)}\right)$$

The average log-likelihood of the MSA under $P(\cdot \,|\, J, h)$ is

$$\mathcal{L}(J, h) = \frac{1}{M} \sum_{m=1}^{M} \left[\sum_i h_i\left(\sigma_i^{(m)}\right) + \sum_{i<j} J_{ij}\left(\sigma_i^{(m)}, \sigma_j^{(m)}\right) \right] - \log Z(J, h). \qquad (39)$$

As an exponential-family model, $\mathcal{L}$ is concave in the natural parameters, so gradient ascent converges to the unique optimum. The gradients are moment gaps:

$$\frac{\partial \mathcal{L}}{\partial h_i(A)} = f_i(A) - p_i(A), \quad \frac{\partial \mathcal{L}}{\partial J_{ij}(A,B)} = f_{ij}(A,B) - p_{ij}(A,B), \qquad (40)$$

where p_i and p_{ij} are model marginals under current (J,h). Hence the updates

$$h_i^{t+1}(A) \leftarrow h_i^t(A) + \eta_h \left[f_i(A) - p_i^{(t)}(A) \right],$$

$$J_{ij}^{t+1}(A,B) \leftarrow J_{ij}^t(A,B) + \eta_J \left[f_{ij}(A,B) - p_{ij}^{(t)}(A,B) \right]. \qquad (41)$$

drive the model toward the maximum likelihood-solution where $f = p$. The obstacle is that the marginals p_i and p_{ij} are not analytically computable at scale; adabmDCA estimates them by MCMC at each epoch.

Estimating Model Expectations via Adaptive MCMC

At training epoch t, run N_s independent Markov chains using Metropolis-Hastings [28], [29] (Gibbs sampling strategy may also be used [30]), each producing N_c samples after an equilibration period T_{eq} and with an inter-sample waiting time T_{wait}. The Monte Carlo estimators are

$$p_i^{(t)}(A) = \frac{1}{N_s N_c} \sum_{m=1}^{N_s N_c} \delta\left(A, \sigma_i^{(m)}(t)\right),$$

$$p_{ij}^{(t)}(A,B) = \frac{1}{N_s N_c} \sum_{\nu=1}^{N_s N_c} \delta\left(A, \sigma_i^{(m)}(t)\right) \delta\left(B, \sigma_j^{(m)}(t)\right). \qquad (42)$$

Chains may be transient, reinitialized every epoch, or persistent, initialized only at the first epoch. Equilibration is often sped up through persistence of chains. For more details on the adaptive scheme, refer to Appendix C.

A practical convergence proxy is the difference between the empirical and the model two-site connected correlations:

$$\varepsilon_c = \max_{i,j,a,B} \left| c_{ij}^{\text{model}}(A,B) - c_{ij}^{\text{emp}}(A,B) \right| \quad \text{where}$$

$$c_{ij}^{\text{model}} = p_{ij} - p_i p_j, \quad c_{ij}^{\text{emp}} = f_{ij} - f_i f_j \tag{43}$$

with a target $\varepsilon_c \approx 10^{-2}$ In addition to this, some other commonly used diagnostics is the Pearson correlation between c^{model} and c^{emp}, one- and two-site fitting errors, and optionally, a third connected correlation on a subset of triples is used to assess generative fidelity beyond pairwise constraints.

Regularization

A fully connected Potts model contains $\sim \frac{L(L-1)}{2}q^2 + Lq$ parameters, in the order of 10^7 -10^9 for realistic L. Due to the finite sample size of MSAs, they rarely contain enough independent information to estimate all of the parameters robustly. Not controlling this uncertainty could lead to overfitting, high variance or instability, and bad conditioning in the model. adabmDCA addresses this with two strategies: regularization and sparsity priors. First, parameter penalties are added via ℓ_1 and ℓ_2 norms:

$$R_{\ell_1}(J,h) = \theta_{1,h} \sum_i \|h_i\|_1 + \theta_{1,J} \sum_{i<j} \|J_{ij}\|_1$$

$$R_{\ell_2}(J,h) = \theta_{2,h} \sum_i \|h_i\|_2^2 + \theta_{2,J} \sum_{i<j} \|J_{ij}\|_2^2 \tag{44}$$

ℓ_2 reduces variance and improves conditioning by smoothly shrinking all parameters. It selects a unique gauge and tends to preserve the relative ordering of strong couplings. On the other hand, ℓ_1 induces sparsity by zeroing weak parameters, also reducing overfitting, though at a cost of biasing small effects downward. Generally, in stochastic settings, elastic-net is used (a combination of both parameters), that stabilizes training near zero. In addition, a separate parameter is used for fields and couplings, typically regularizing J more strongly than h.

Second, structural sparsity is imposed, reflecting that only a small fraction of residue pairs form contacts. Encoding a structural prior can reduce variance and speed up learning. There are two approaches:

1. A priori topology. Reduce the number of parameters by starting from a restricted edge set, for example pairs with high MI, and learn only those J_{ij}, omitting the rest.

2. Information-based decimation. Starting dense and iteratively removing the least informative couplings until target sparsity. This directly controls overfitting by only keeping the parameters that actually affect the model's predictions.

In short, pruning and decimation prevent overfitting by removing parameters that don't materially alter the model, and have the added benefit of aligning with the biological prior of contact sparsity.

4.4. Autoregressive Network DCA

Building on the pseudolikelihood maximization of plmDCA, and Boltzmann-machine parametrization of bmDCA, arDCA was introduced as a faster and more efficient alternative [31]. While plmDCA is limited in scope and bmDCA suffers from slow MCMC sampling, reformulating the problem as an autoregressive network provides some key improvements. The idea is to decompose the joint sequence probability distribution into its conditionals, thereby turning the problem of inference into a supervised learning task. In this way, arDCA emerges as a powerful model for extracting structural and functional information from the rapidly growing protein databases.

4.4.1. Method

In arDCA, the exponential-family MaxEnt distribution underlying previous DCA methods is factorized into conditional probabilities, predicting each residue from its predecessors. This follows directly from the chain rule of probability:

$$P(\boldsymbol{\sigma}) = \prod_{i=1}^{L} P(\sigma_i | \sigma_1, ..., \sigma_{i-1}) \tag{45}$$

Inspired by approaches in classical [32] and quantum [33] statistical mechanics, the conditional distribution is parametrized as:

$$P(\sigma_i \mid \sigma_{i-1}, ..., \sigma_1) = \frac{\exp\left(h_i(\sigma_i) + \sum_{j=1}^{i-1} J_{ij}(\sigma_i, \sigma_j)\right)}{\sum_{\sigma_i} \exp\left(h_i(\sigma_i) + \sum_{j=1}^{i-1} J_{ij}(\sigma_i, \sigma_j)\right)} \tag{46}$$

This is a multiclass softmax regression [34], generalizing logistic regression to multiple residue states. The model is defined by site-specific fields $h_i(A)$ and directed couplings $J_{ij}(A, B)$, and was therefore termed arDCA. For comparison, the authors also introduced a simpler profile model (independent-site model), which only includes field terms. Its joint probability factorizes across sites,

$$P(\sigma_1, ..., \sigma_L) = \prod_{i=1...L} f_i(\sigma_i) \tag{47}$$

. making it a useful baseline.

Although arDCA shares the same number of parameters as standard DCA, their interpretation differs. In the Potts formulation used by standard DCA, couplings are symmetric, $J_{ij}(a, b) = J_{ji}(b, a)$. In contrast, arDCA employs directed couplings, describing the influence of site j on i only for $j < i$. Thus, only the lower-triangular part of the coupling matrix is populated. Inference is carried out via pseudo-likelihood maximization [35], allowing exact gradients to be computed from data–unlike bmDCA, which requires costly MCMC sampling. Moreover, plmDCA conditions each residue on all others in the sequence and then symmetrizes the couplings to align with Potts models. This symmetrization shifts parameters away from their maximum-likelihood values and reduces generative accuracy. arDCA avoids this step, preserving likelihood consistency.

The chain rule decomposition is valid under any site ordering, but once the conditional parametrization of Eq. 46 is adopted, the likelihood becomes order dependent. Opti-

mizing over $L!$ permutations is infeasible in practice, so the original authors proposed entropic ordering as a principled heuristic, ranking sites from lowest entropy (most conserved position) to the highest entropy (most variable). This choice reflects a natural interpretation: conserved positions provide little additional information if occupied by their dominant residue, whereas deviations must be compensated downstream by more variable sites. By placing these variable positions later in the order, the model can capture compensatory effects in a structured way. Although we do not optimize or reorder sites in our implementation, entropic ordering illustrates how positional dependencies can be learned by arDCA.

Coupling Inference: Likelihood Maximization

The inference of the parameters is done through likelihood maximization. Following a Bayesian setting with a uniform prior, the optimal parameters are those that maximize the probability of the data:

$$
\begin{aligned}
\{J^*, h^*\} &= \arg\max_{\{J,h\}} P(\mathcal{M}|\{J,h\}) \\
&= \arg\max_{\{J,h\}} \log P(\mathcal{M}|\{J,h\}) \\
&= \arg\max_{\{J,h\}} \sum_{m=1}^{M} \log \prod_{i=1}^{L} P(\sigma_i^m|\sigma_{i-1}^m, ..., \sigma_1^m) \\
&= \arg\max_{\{J,h\}} \sum_{m=1}^{M} \sum_{i=1}^{L} \log P(\sigma_i^m|\sigma_{i-1}^m, ..., \sigma_1^m)
\end{aligned}
\tag{48}
$$

Each $h_i(A)$ and $J_{ij}(A, B)$ is present in only one conditional probability $P(\sigma_i|\sigma_{i-1}, ..., \sigma_1)$, thus we can maximize each conditional probability independently in Eq. 48:

$$
\{J_{ij}^*, h_i^*\} = \arg\max_{\{J_{ij}, h_i\}} \sum_{m=1}^{M} \left[h_i(\sigma_i^m) + \sum_{j=1}^{i-1} J_{ij}(\sigma_i^m, \sigma_j^m) - \log z_i(a_{i-1}^m, ..., a_1^m) \right] \tag{49}
$$

where

$$z_i(\sigma_{i-1}, .., \sigma_1) = \sum_{\sigma_i} \exp\left(h_i(\sigma_i) + \sum_{j=1}^{i-1} J_{ij}(\sigma_i, \sigma_j)\right) \tag{50}$$

is the normalization factor of the conditional probability of σ_i. Taking the derivative with respect to $h_i(A)$ or $J_{ij}(A, B)$, with $j = 1, ..., i-1$, we get:

$$0 = \frac{1}{M} \sum_{m=1}^{M} \left[\delta(A, \sigma_i^m) - \frac{\partial \log z_i(\sigma_{i-1}^m, .., \sigma_1^m)}{\partial h_i(A)}\right]$$

$$0 = \frac{1}{M} \sum_{m=1}^{M} \left[\delta(A, \sigma_i^m)\delta\left(B, \sigma_j^m\right) - \frac{\partial \log z_i(\sigma_{i-1}^m, .., \sigma_1^m)}{\partial J_{ij}(A, B)}\right]. \tag{51}$$

Using Eq. 50, we find

$$\frac{\partial \log z_i(\sigma_{i-1}^m, .., \sigma_1^m)}{\partial h_i(A)} = P(\sigma_i = A \mid \sigma_{i-1}^m, ..., \sigma_1^m)$$

$$\frac{\partial \log z_i(\sigma_{i-1}^m, .., \sigma_1^m)}{\partial J_{ij}(A, B)} = P(\sigma_i = A \mid \sigma_{i-1}^m, ..., \sigma_1^m)\delta\left(\sigma_j^m, B\right). \tag{52}$$

The set of equations reduces to a simple form:

$$f_i(A) = \langle P(\sigma_i = A \mid \sigma_{i-1}^m, ..., \sigma_1^m)\rangle_{\mathcal{M}},$$

$$f_{ij}(A, B) = \langle P(\sigma_i = A \mid \sigma_{i-1}^m, ..., \sigma_1^m)\delta\left(\sigma_j^m, b\right)\rangle_{\mathcal{M}}, \tag{53}$$

where $\langle \cdot \rangle_{\mathcal{M}} = \frac{1}{M} \sum_{m=1}^{M} \cdot^m$ denotes the empirical average. The first variable, $i = 1$, is unconditioned, therefore the coupling equation is $J \equiv 0$ and the equation for the field is the profile model, $h_1(a) = \log f_1(a) + \text{const}$.

Unlike bmDCA, the equations do not enforce exact matching between model marginals and empirical frequencies. Thus the ability to reproduce the frequencies is a good proxy for the generative properties of the model, on top of the fitting quality of current

parameters. In practice, the parameters are updated using gradient descent on the likelihood. These gradients are exact as we can take the expectations directly over the MSA. For regularization, arDCA makes use of ℓ_2 penalties, with different strengths depending on whether the task is generating sequences or contact prediction. It was noted that small regularization values improved the generative quality while larger were beneficial for contact prediction.

4.4.2. Key Contributions

arDCA represents a major advance over previous DCA methods by enabling exact gradient computations from the data. This eliminates the need for costly MCMC sampling, leading to training speeds that are two to three orders of magnitude faster while maintaining, or even improving, predictive accuracy [31]. The resulting model is therefore lightweight and computationally scalable, making it feasible for application on large protein families. For the largest families tested, although arDCA was most time efficient, the deep learning model DeepSequence [36] prevailed in prediction accuracy.

A second key contribution is that arDCA allows for the exact calculation of sequence probabilities, a task intractable for previous models. In bmDCA, only unnormalized sequence weights could be computed, with the partition function requiring expensive thermodynamic integration. By contrast, each conditional probability in arDCA is normalized locally. This reduces the computational burden from summing over q^L possible sequences to only L sums over q residue states. This feature enables direct sequence-level comparisons across models, essential for applications such as homology detection, protein family classification, and model-based sequence evaluation.

5. Implementation and Exploration

The original arDCA model was developed in Julia, a high-performance language designed for numerical and scientific computing. Julia's just-in-time (JIT) compilation, efficient handling of linear algebra, and native support for parallelism, make it well-suited for implementing large-scale statistical models. For broader accessibility and

integration with standard machine-learning workflows, we re-implemented the model in Python.

Important Differences Between Julia and Python

Several structural differences complicate a direct translation between Julia and Python. Julia is column-major and 1-indexed, while Python is row-major and 0-indexed; these distinctions require careful adjustment of array reshaping, indexing, and loop boundaries to maintain correctness and efficiency. Julia loops are compiled to fast machine code and can avoid bounds checking with the `@inbounds` macro, whereas Python loops are inherently slow and require vectorization through NumPy or broadcasting for comparable performance. Finally, Julia's JIT compilation introduces an initial overhead but accelerates subsequent calls, while Python relies on interpretation and delegates heavy computation to optimized C/C++ or Fortran backends.

Julia favors low-level efficiency with straightforward loop-based implementations while for Python, a vectorized style is encouraged, relying on well-developed numerical libraries. Our Python re-implementation therefore required significant restructuring of the original Julia code to align with Python's capabilities without sacrificing performance.

5.1. IMPLEMENTATION DETAILS

We implement the autoregressive network within the PyTorch ecosystem, using the `nn.Module` interface to leverage its training and optimization capabilities. The central operation of the model is the computation of the autoregressive logits, which define the conditional distribution at each sequence position.

Logit Computation

For site i, the logit vector is given by

$$z[m, i, a] = h[i, a] + \sum_{j<i} \sum_{b} J[i, j, a, b] X[m, j, b] \tag{54}$$

where h are the local biases and J the pairwise couplings, and X is a tensor holding the one-hot encoding of the symbol b observed at site j in sequence m.

To compute the autoregressive logits, our initial implementation employed a masked matrix multiplication, performed in a single step using the `einsum` operator. While correct, and more efficient than naive loops, calculating the full interaction matrix is computationally inefficient because the upper-triangular entries are all masked to zero. To address this, we devised a more efficient formulation that exploits the block-sparse structure of the coupling matrix. Specifically, we collect all lower-triangular index pairs (i, j) with $j < i$, extract the corresponding coupling blocks $J[i, j]$, and accumulate their contributions through an `einsum` operation. The resulting implementation only computes terms required by the autoregressive factorization, removing unnecessary computation.

```python
def compute_ar_logits(self, X_oh: torch.Tensor):
    """z[m,i,a] = h[i,a] + sum_{j<i} sum_b J[i,j,a,b] * X[m,j,b]"""
    X_oh = X_oh.to(self.J.dtype)
    M, L, q = X_oh.shape
    logits = self.h_pos.unsqueeze(0).expand(M, -1, -1).clone()  # (M,L,q)

    # collect lower-triangular index pairs
    i_idx, j_idx = torch.tril_indices(L, L, offset=-1)
    J_blocks = self.J[i_idx, j_idx]    # (n_pairs, q, q)
    X_blocks = X_oh[:, j_idx]          # (M, n_pairs, q)

    # compute pairwise contributions and accumulate into logits
    contrib = torch.einsum("mpq,pqr->mpr", X_blocks, J_blocks)
    contrib = torch.einsum("mpb,pab->mpa", X_blocks, J_blocks)
    logits = logits.index_add(1, i_idx, contrib)
    return logits
```

With this formulation, the computational complexity scales with the number of lower-triangular pairs rather than the full $L \times L$ coupling matrix. For long sequences, this is especially beneficial. In addition, by avoiding the construction of the full masked interaction matrix, the memory footprint is roughly halved, from L^2 couplings to $\frac{L(L-1)}{2}$.

Throughout the model, an explicit binary mask J_mask is maintained that encodes the lower-triangular structure, to ensure consistency in the autoregressive aspect. After each optimization step, unused entries of J are clamped to zero.

Ancestral Sampling

Once model parameters (h, J) are inferred, sequences can be generated by ancestral sampling. The autoregressive factorization in Eq. 45 allows residues to be sampled sequentially, one site at a time. The procedure is:

1. Draw the first residue from $P(\sigma_1)$

2. For each subsequent site i, compute the conditional distribution

$$P(\sigma_i | \sigma_1, ..., \sigma_i - 1) \tag{55}$$

given the residues already sampled.

3. Sample σ_i from this distribution.

4. Repeat until all L positions are generated.

Since each conditional distribution only has $q = 21$ states, the sampling step is fast. The entire sequence generation scales linearly with sequence length L.

```python
def sample_sequences(self, n_samples: int = 1) -> torch.Tensor:
    self.eval()
    L, q = self.L, self.q
    samples = torch.zeros((n_samples, L), dtype=torch.long)
    for pos in range(L):
        if pos == 0:
            logits = self.h_pos[0].unsqueeze(0).expand(n_samples, -1)
```

```
else:
    partial_oh = torch.zeros((n_samples, L, q), device=device)
    for i in range(pos):
        partial_oh[torch.arange(n_samples), i, samples[:, i]] = 1.0
    full_logits = self.compute_ar_logits(partial_oh) # (n_samples, L, q)
    logits = full_logits[:, pos, :]
probs = F.softmax(logits, dim=-1)
samples[:, pos] = torch.multinomial(probs, 1).squeeze(-1)
return samples
```

The python implementation above iteratively constructs one-hot encodings of previously sampled positions, computes logits for the current site via the previously defined autoregressive logits function, converts them to probabilities through a softmax, and samples the next residue via multinomial sampling.

Training and Evaluation Details

The model is initialized with the local field of the first position, $h[0]$, estimated from the empirical frequencies observed at site 0. The rest of the parameters are drawn from small random distributions. Training incorporates sequence reweighting to correct for redundancy in MSAs. The weights for each sequence are computed before model training, with parameter `theta` controlling the sequence similarity threshold. The model can be optimized with either L-BFGS [37], suited for energy-based models, or AdamW [38], offering a more scalable and robust option for large datasets. Regularization is applied separately to the field and couplings via ℓ_2-norm penalties with hyperparameters λ_h and λ_J.

For evaluation, the model reports the negative log-likelihood (NLL) per position,

$$\mathrm{nll}_{m,i} = -w_m \log p_\theta\big(X_{m,i}|X_{m,<i}\big) \tag{56}$$

with the average NLL per position

31

$$\overline{\text{NLL}} = \frac{1}{ML} \sum_{m=1}^{M} \sum_{i=1}^{L} \text{nll}_{m,i}. \tag{57}$$

Another metric used for evaluation is the perplexity,

$$\text{Perplexity} = \exp\left(\overline{\text{NLL}}\right). \tag{58}$$

Perplexity is widely used in Natural Language Processing and represents the average number of equally likely choices the model is making per residue [39]. Lower perplexity means the model is more confident and better fits the data. The effective sample size is also reported, to make the results more easily comparable across different datasets.

5.2. Experiment on the effect of gaps in multiple sequence alignments

When constructing MSAs, gaps are introduced to better align homologous positions across sequences. These gaps serve two main purposes: (i) ensuring residues align correctly at homologous sites, and (ii) extending sequences to a specific length for models like ArDCA, that require fixed length input. Since the presence of gaps can influence the statistical properties of the alignment, we examined how restricting the fraction of gaps affects results. Specifically, we introduced two filtering parameters:

- `max_gap_fraction`: sets the maximum allowed fraction of gaps per sequence.
- `max_col_gap_fraction`: sets the maximum allowed fraction of gaps in a column.

We will observe how the M_{eff} is affected by removing rows and columns, as well as some key metrics like average per-site Perplexity, and average per-site train and validation NLL. For the first experiment, we varied `max_gap_fraction`, while holding the other fixed, with three thresholds: $0.05, 0.1,$ and 1. The last threshold does not exclude any sequences, therefore is used as a baseline for comparison. The difference in training time of the models was also recorded, but given the small nature of the change, no conclusive results were observed.

version	max_gap_fraction	M	M_{eff}	Perplexity	Train NLL	Val NLL
0	1	13600	**4248.8**	1.72	0.32	0.54
1	0.1 (-1.6%)	13376	4145.4	**1.58**	0.32	**0.46**
2	0.05 (-6%)	12784	3876.5	1.68	0.3	0.51

Table 1: Perplexity and NLL improve when moderate gap filtering (0.1) is applied

Restricting the allowed fraction of gaps removed sequences therefore reduced the effective number of sequences M_{eff}. The strictest filtering criterion (0.05), removed approximately 6% of sequences relative to the baseline, yielding a smaller dataset and just slightly better perplexity value. The lowest perplexity was observed when 10% of gaps were excluded. A perplexity of 1.58 means the model is, on average, choosing between fewer than 2 equally likely residues. Although the full dataset had the highest M_{eff}, higher perplexity suggests that allowing too many gaps may negatively impact the quality of the alignment. A similar exploration was performed for the column gaps with values $0.025, 0.1,$ and 1. Again, 1 represents the baseline with no columns removed.

version	max_col_gap_fraction	L	M_{eff}	Perplexity	Train NLL	Val NLL
0	1	53	4248.8	1.72	0.32	0.54
3	0.1 (-3.7%)	51	4208.5	**1.71**	0.32	**0.53**
4	0.025 (-7.5%)	49	**4363.0**	1.75	0.34	0.56

Table 2: Similar perplexity and NLL throughout the experiments

In this experiment, the change in perplexity and M_eff wasn't as drastic, though again we observe the best perplexity value in the second threshold. It is interesting that the effective sample size increased when removing the four columns of version 4. This occurred because of the change in the weight computation of the sequences. Both sequence-level and site-level filtering yield a better result when using moderate thresholds, where excluding only the data with the most gaps improves model performance. Overly aggressive filtering discards useful information and hurts perplexity, while using the full dataset also doesn't reach the best performance.

6. Results and Discussion

6.1. Training Behavior

We trained arDCA with L-BFGS, using exact gradients of the negative log-likelihood (NLL). A 10% validation split was held out, and early stopping prevented overfitting. Across all settings, the training NLL decreased smoothly and the validation NLL tracked it closely, indicating stable optimization and good generalization. A few values of the λ_J and λ_h parameters were tested but the final choices aligned with the papers recommendations of `1e-4` and `1e-6` respectively. Figure 3 shows the runs in the fraction gap study, the NLL drops from around 1.7 to under 1.1. In practice. we observed little sensitivity to random seed for the reported settings, consistent with convex, per-site softmax problems and well-behaved ℓ_2 regularization. Exact gradients and the autoregressive factorization yield smooth, monotonic learning curves and minimal overfitting under regularization.
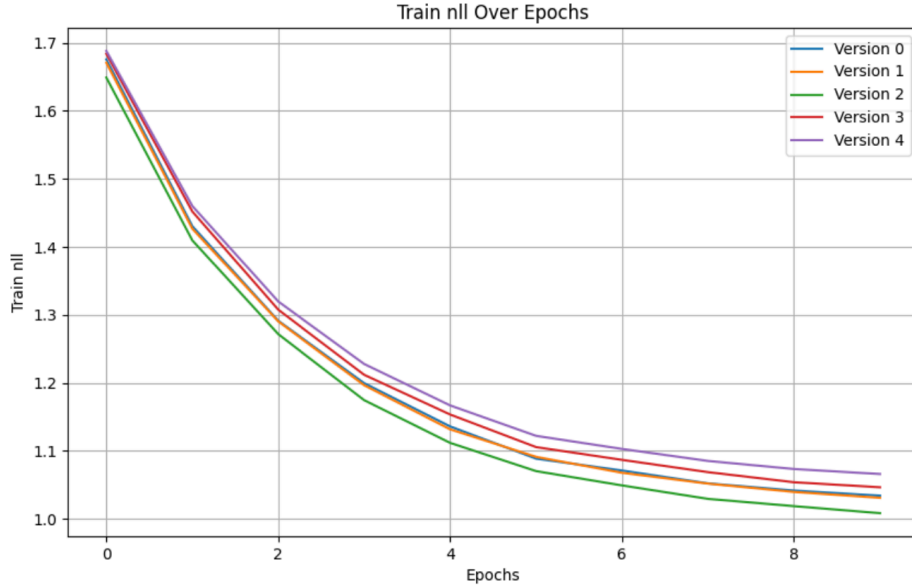

Figure 3: Training NLL plotted for baseline and gap fraction experiments

6.2. Generative Model Quality

To assess the structural plausibility of sequences generated by the model, we employed AlphaFold's predicted Local Distance Difference Test (pLDDT) score [40]. The pLDDT provides a residue-level confidence measure ranging from 0 to 100, reflecting the

34

predicted accuracy of local environments. High pLDDT values (>90) correspond to very reliable backbone and side-chain placement, values between 70-90 indicate generally well-resolved domain-level structures, while values below 70 are associated with uncertain or potentially disordered regions. By evaluating generated sequences with this metric, we obtain an intrinsic estimate of whether the model captures the structural constraints encoded in the protein family. pLDDT represents a quantitative proxy for generative quality, complementing the evaluation metrics used previously such as negative log-likelihood and perplexity. It serves to provide a bridge between sequence statistics and three-dimensional structure accuracy.
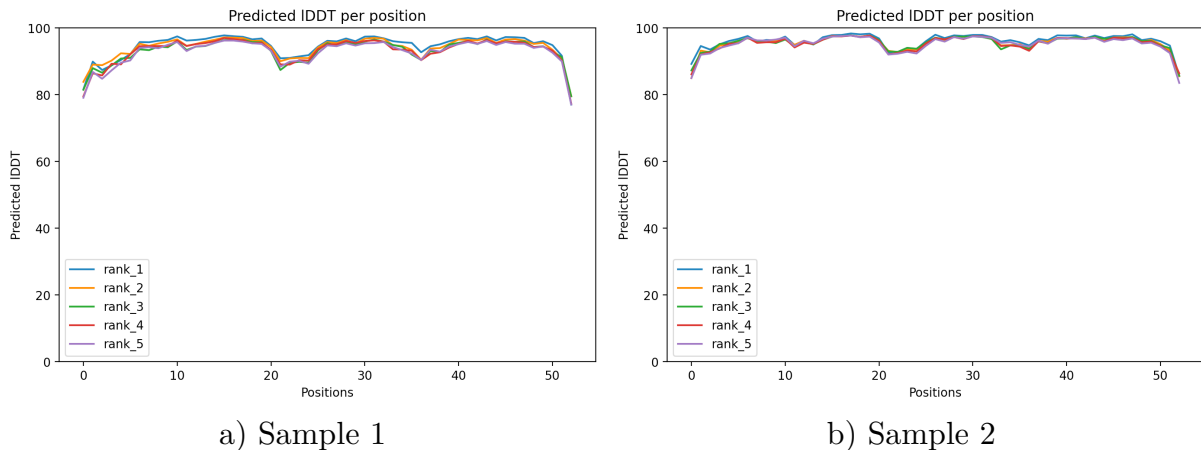

a) Sample 1   b) Sample 2

Figure 4: pLDDT per site for model-generated samples

Figure 4 shows the pLDDT profiles for two representative model-generated samples. Both sequences display consistently high pLDDT values across most positions, indicating that the model generates foldable and structurally coherent proteins. The slight drop in the end of a sequence is likely as these sites often correspond to poorly aligned or frequently gapped positions in the alignment. These results complement our previously observed statistical metrics by linking generative performance to structural integrity.

6.3. LIMITATIONS AND FUTURE WORK

The Python implementation, while optimized through a lower-triangular masked multiplication, remains less efficient than the original Julia version due to the differences in performance between the languages. GPU acceleration could substantially improve

speed and scalability, enabling the model to handle longer, or more populated, protein families. From an architectural perspective, arDCA is restricted to fixed length sequences, limiting its ability to model families with variable domain lengths. Furthermore, our study used the natural site ordering exclusively; prior work has demonstrated that entropy-based permutations can yield superior likelihood estimates that improve the quality of the model. Exploring alternative orderings could yield more expressive models with better structural accuracy. Another important limitation is the strong dependence of arDCA on the quality of the multiple sequence alignment. Families with limited diversity or strong phylogenetic bias may provide insufficient or misleading signals. In such cases, the model risks poor generalization, generating sequences that fail to capture the true structural and functional constraints of the protein family. Careful sequence reweighting and gap filtering could mitigate these effects. Our analysis was constrained by evaluation throughput. The pLDDT metric from AlphaFold provides valuable structural insight but running it for each sample is computationally expensive, limiting us to a small number of generated samples. A more rigorous statistical analysis would require folding a larger set of generated sequences and comparing them to natural ones.

Future work should thus focus on shifting the heavy computations to be done on the GPU, investigating alternative site orderings, such as the entropic order heuristic, and exploring more expressive generative architectures such as transformer-based models [41]. Finally, the study could benefit from a large-scale structural benchmarking through AlphaFold's pLDDT. Collectively, these improvements would make a more scalable, flexible, and biologically accurate model, further bridging the gap between statistical models and natural protein sequences.

7. CONCLUSION

In this work, we demonstrated that arDCA, originally implemented in Julia, can be successfully adapted in Python as a practical tool for protein sequence analysis. By

leveraging PyTorch and introducing an efficient block-sparse formulation for computing autoregressive logits, our implementation reduces computational cost and memory usage. Empirical evaluation shows smooth optimization dynamics, improvements in negative log-likelihood and perplexity, and structural plausibility as measured by pLDDT. Our experiments on gap filtering highlight the importance of sequence quality control, where moderate gap filtering improves model confidence. In the future, this work could benefit from GPU acceleration to improve its speed and allow it to work more efficiently with large datasets. Overall, this work adapted arDCA beyond its Julia origin and produced a powerful protein-generating model, opening the doors for downstream tasks like novel protein generation and structural modeling.

Code availability The full Python code is available at https://shorturl.at/5ufHk.

BIBLIOGRAPHY

[1]   M. Weigt, "Coevolutionary Analysis of Protein-Protein Interactions." [Online].
      Available: https://www.youtube.com/watch?v=IYA8WEsUcG0

[2]   J. Jumper *et al.*, "Highly accurate protein structure prediction with AlphaFold,"
      Nature, vol. 596, no. 7873, pp. 583–589, 2021, doi: 10.1038/s41586-021-03819-2.

[3]   "The Shape and Structure of Proteins," in *Molecular Biology of the Cell*, 4th ed.,
      New York: Garland Science, 2002. [Online]. Available: https://www.ncbi.nlm.nih.
      gov/books/NBK26830/

[4]   M. Mirdita, K. Schütze, Y. Moriwaki, L. Heo, S. Ovchinnikov, and M. Steinegger,
      "ColabFold: making protein folding accessible to all," *Nature Methods*, vol. 19, no.
      6, pp. 679–682, 2022, doi: 10.1038/s41592-022-01488-1.

[5]   European Bioinformatics Institute (EMBL-EBI), "What are protein fami-
      lies?." [Online]. Available: https://www.ebi.ac.uk/training/online/courses/protein-
      classification-intro-ebi-resources/protein-classification/what-are-protein-families/

[6]   E.-E. Training, "What is phylogenetics?." [Online]. Available: https://www.ebi.ac.
      uk/training/online/courses/introduction-to-phylogenetics/what-is-phylogenetics/

[7]   M. Wiltgen, "Algorithms for Structure Comparison and Analysis: Homology Mod-
      elling of Proteins," *Encyclopedia of Bioinformatics and Computational Biology*.
      Academic Press, pp. 38–61, 2019. doi: https://doi.org/10.1016/B978-0-12-809633-
      8.20484-6.

[8]   moshi4, "pyMSAviz: MSA (Multiple Sequence Alignment) visualization Python
      package." 2024.

[9] C. Zhang *et al.*, "The Historical Evolution and Significance of Multiple Sequence Alignment in Molecular Structure and Function Prediction," *Biomolecules*, vol. 14, no. 12, 2024, doi: 10.3390/biom14121531.

[10] "Prediction Center."

[11] B. Adhikari and J. Cheng, "Protein Residue Contacts and Prediction Methods," *Methods in Molecular Biology*, vol. 1415, pp. 463–476, 2016, doi: 10.1007/978-1-4939-3572-7_27.

[12] N. Dietler, U. Lupo, and A.-F. Bitbol, "Impact of phylogeny on structural contact inference from protein sequence data," *Journal of the Royal Society, Interface*, vol. 20, no. 199, 2023, doi: 10.1098/rsif.2022.0707.

[13] E. T. Jaynes, "Information Theory and Statistical Mechanics," *Physical Review*, vol. 106, no. 4, pp. 620–630, May 1957, doi: 10.1103/PhysRev.106.620.

[14] E. T. Jaynes, "Information Theory and Statistical Mechanics. II," *Physical Review*, vol. 108, no. 2, pp. 171–190, 1957, doi: 10.1103/PhysRev.108.171.

[15] J. Paul Penfield, "Lecture Notes for Information and Entropy." 2003.

[16] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423623–656, 1948.

[17] E. Ising, "Beitrag zur Theorie des Ferromagnetismus," *Zeitschrift für Physik*, vol. 31, no. 1, pp. 253–258, 1925, doi: 10.1007/BF02980577.

[18] R. B. Potts, "Some generalized order-disorder transformations," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 48, no. 1, pp. 106–109, 1952, doi: 10.1017/S0305004100027419.

[19] S. H. H. J. H. T. Weigt M White RA, "Identification of direct residue contacts in protein-protein interaction by message passing," *Proc Natl Acad Sci USA*, vol. 106, pp. 67–72, 2009, doi: 10.1073/pnas.0805923106.

[20] F. Morcos *et al.*, "Direct-coupling analysis of residue coevolution captures native contacts across many protein families," *Proceedings of the National Academy of Sciences*, vol. 108, no. 49, pp. E1293–E1301, 2011, doi: 10.1073/pnas.1111471108.

[21] T. Plefka, "Convergence condition of the TAP equation for the infinite-ranged Ising spin glass model," *Journal of Physics A: Mathematical and General*, vol. 15, no. 6, pp. 1971–1978, 1982, doi: 10.1088/0305-4470/15/6/035.

[22] A. Georges and J. S. Yedidia, "How to expand around mean-field theory using high-temperature expansions," *Journal of Physics A: Mathematical and General*, vol. 24, no. 9, pp. 2173–2192, 1991, doi: 10.1088/0305-4470/24/9/018.

[23] A. Haldane and R. M. Levy, "Influence of multiple-sequence-alignment depth on Potts statistical models of protein covariation," *Physical Review E*, vol. 99, no. 3, Mar. 2019, doi: 10.1103/physreve.99.032405.

[24] D. T. Jones, D. W. A. Buchan, D. Cozzetto, and M. Pontil, "PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments," *Bioinformatics*, vol. 28, no. 2, pp. 184–190, 2012, doi: 10.1093/bioinformatics/btr638.

[25] 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, 2014, doi: https://doi.org/10.1016/j.jcp.2014.07.024.

[26] M. Figliuzzi, P. Barrat-Charlaix, and M. Weigt, "How Pairwise Coevolutionary Models Capture the Collective Residue Variability in Proteins," *Molecular Biology and Evolution*, vol. 35, no. 4, pp. 1018–1027, Apr. 2018, doi: 10.1093/molbev/msy007.

[27] A. P. Muntoni, A. Pagnani, M. Weigt, and F. Zamponi, "adabmDCA: adaptive Boltzmann machine learning for biological sequences," *BMC Bioinformatics*, vol. 22, no. 528, 2021, doi: 10.1186/s12859-021-04441-9.

[28] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953, doi: 10.1063/1.1699114.

[29] W. K. Hastings, "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970, [Online]. Available: http://www.jstor.org/stable/2334940

[30] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 721–741, 1984, doi: 10.1109/TPAMI.1984.4767596.

[31] J. Trinquier, G. Uguzzoni, A. Pagnani, F. Zamponi, and M. Weigt, "Efficient generative modeling of protein sequences using simple autoregressive models," *Nature Communications*, vol. 12, no. 1, p. 5800, 2021, doi: 10.1038/s41467-021-25756-4.

[32] D. Wu, L. Wang, and P. Zhang, "Solving statistical mechanics using variational autoregressive networks," *Physical Review Letters*, vol. 122, no. 8, p. 80602, 2019, doi: 10.1103/PhysRevLett.122.080602.

[33] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua, "Deep autoregressive models for the efficient variational simulation of many-body quantum systems,"

Physical Review Letters, vol. 124, no. 2, p. 20503, 2020, doi: 10.1103/Phys-RevLett.124.020503.

[34] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer Science & Business Media, 2009.

[35] S. Balakrishnan, H. Kamisetty, J. G. Carbonell, S.-I. Lee, and C. J. Langmead, "Learning generative models for protein fold families," *Proteins: Structure, Function, and Bioinformatics*, vol. 79, no. 4, pp. 1061–1078, 2011, doi: 10.1002/prot.22934.

[36] A. J. Riesselman, J. B. Ingraham, and D. S. Marks, "Deep generative models of genetic variation capture the effects of mutations," *Nature Methods*, vol. 15, no. 10, pp. 816–822, 2018.

[37] D. C. Liu and J. Nocedal, "On the Limited Memory Method for Large Scale Optimization," *Mathematical Programming*, vol. 45, no. 1–3, pp. 503–528, 1989, doi: 10.1007/BF01589116.

[38] I. Loshchilov and F. Hutter, "Fixing Weight Decay Regularization in Adam," *CoRR*, 2017, [Online]. Available: http://arxiv.org/abs/1711.05101

[39] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, and P. Koehn, "One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling," *CoRR*, 2013, [Online]. Available: http://arxiv.org/abs/1312.3005

[40] E.-E. Training, "pLDDT: Understanding local confidence." [Online]. Available: https://www.ebi.ac.uk/training/online/courses/alphafold/inputs-and-outputs/evaluating-alphafolds-predicted-structures-using-confidence-scores/plddt-understanding-local-confidence/

[41] F. Caredda and A. Pagnani, "Direct Coupling Analysis and the Attention Mechanism," *bioRxiv*, 2024, doi: 10.1101/2024.02.06.579080.

A LANGRAGE MULTIPLIERS CALCULATION FOR MAXIMUM ENTROPY PRINCIPLE

We seek the probability distribution $\{p_i\}$ over stattes x_i that maximizes the Shannon entropy subject to the normalization and expectation value constraints. Introduce multipliers λ and μ:

$$\Phi = -\sum_i p_i \ln p_i - \lambda \left(\sum_i p_i - 1 \right) - \mu \left(\sum_i p_i f(x_i) - \langle f \rangle \right) \tag{59}$$

From here, we can set the stationarity condition

$$\frac{\partial \Phi}{\partial p_i} = -(\ln p_i + 1) - \lambda - \mu f(x_i) = 0, \tag{60}$$

rearranging:

$$\ln p_i = -1 - \lambda - \mu f(x_i). \tag{61}$$

We then solve for p_i, absorbing the -1 constant in λ,

$$p_i = \exp(-\lambda - \mu f(x_i)) \tag{62}$$

Impose the normalization condition:

$$1 = \sum_i \exp(-\lambda - \mu f(x_i)), \quad \text{and factor our } e^{-\lambda}$$

$$1 = e^{-\lambda} \sum_i e^{-\mu f(x_i)}. \tag{63}$$

Define the partition function:

$$Z(\mu) = \sum_i e^{-\mu f(x_i)}, \text{ thus}$$

$$e^{-\lambda} Z(\mu) = 1 \quad \Rightarrow \quad \lambda = \ln Z(\mu) \tag{64}$$

Thus, the maximum-entropy distribution is

$$p_i = \frac{1}{Z(\mu)} \exp(-\mu f(x_i)). \qquad (65)$$

The multiplier μ is determined by the constraint:

$$\langle f \rangle = \sum_i p_i f(x_i) = -\frac{\partial}{\partial \mu} \ln Z(\mu). \qquad (66)$$

Finally, inserting this into the entropy formula,

$$S_{\text{max}} = \ln Z(\mu) + \mu \langle f \rangle \qquad (67)$$

giving us the canonical exponential-family distribution.

B Small Coupling Mean-Field Derivation

Start with the perturbed Hamiltonian

$$\mathcal{H}(\alpha) = -\alpha \sum_{1 \leq i < j \leq L} e_{ij}(A_i, A_j) - \sum_{i=1}^{L} h_i(A_i) \tag{1}$$

which allows interpolation between independent couplings, $\alpha = 0$, and the original model, $\alpha = 1$. We also define the Gibbs potential

$$-\mathcal{G}(\alpha) = \ln \left[\sum_{\{A_i | i=1,\dots,L\}} e^{-\mathcal{H}(\alpha)} \right] - \sum_{i=1}^{L} \sum_{B=1}^{q-1} h_i(B) P_i(B) \tag{2}$$

as the Legendre transform of the free energy $\mathcal{F} = -\ln Z$. The fields can be found via

$$h_i(A) = \frac{\partial \mathcal{G}(\alpha)}{\partial P_i(A)}, \tag{3}$$

and

$$\left(C^{-1} \right)_{ij}(A, B) = \frac{\partial h_i(A)}{\partial P_j(B)} = \frac{\partial^2 \mathcal{G}(\alpha)}{\partial P_i(A) \partial P_j(B)}. \tag{4}$$

Our aim is to expand the Gibbs potential up to first order around the independent-site case $\alpha = 0$,

$$\mathcal{G}(\alpha) = \mathcal{G}(0) + \left. \frac{d\mathcal{G}(\alpha)}{d\alpha} \right|_{\alpha=0} \alpha + \mathcal{O}(\alpha^2) \tag{5}$$

Independent-site approximation

To start, let us consider the Gibbs potential in $\alpha = 0$. In this case, the Gibbs potential equals the negative entropy of an ensemble of L uncoupled Potts spins,

$$\mathcal{G}(0) = \sum_{i=1}^{L} \sum_{A=1}^{q} P_i(A) \ln P_i(a)$$

$$= \sum_{i=1}^{L} \sum_{A=1}^{q-1} P_i(A) \ln P_i(a) + \sum_{i=1}^{L} \left[1 - \sum_{A=1}^{q-1} P_i(A) \right] \ln \left[1 - \sum_{A=1}^{q-1} P_i(A) \right] \quad (6)$$

Mean-field approximation

To get the first order in Eq. 5, we have to determine the derivative at $\alpha = 0$. Recalling the definition of Gibbs potential in Eq. 2,

$$\frac{d\mathcal{G}(\alpha)}{d\alpha} = \frac{-d}{d\alpha} \ln Z(\alpha) - \sum_{i=1}^{L} \sum_{A=1}^{q-1} \frac{dh_i(A)}{d\alpha} P_i(A)$$

$$= - \sum_{\{A_i\}} \left[\sum_{i<j} e_{ij}(A_i, A_j) + \sum_{i} \frac{dh_i(A)}{d\alpha} \right] \frac{e^{-\mathcal{H}(\alpha)}}{Z(\alpha)} - \sum_{i=1}^{L} \sum_{A=1}^{q-1} \frac{dh_i(A)}{d\alpha} P_i(A) \quad (7)$$

$$= - \left\langle \sum_{i<j} e_{ij}(A_i, A_j) \right\rangle_{\alpha}$$

The first derivative of the Gibbs potential with respect to α this is the average of the coupling term in the Hamiltonian. At $\alpha = 0$, this average can be done easily due to the joint distribution of all variables becoming factorized over the single sites,

$$\frac{d\mathcal{G}(\alpha)}{d\alpha}\bigg|_{\alpha=0} = - \sum_{i<j} \sum_{A,B} e_{ij}(A_i, A_j) P_i(A) P_j(B). \quad (8)$$

Plugging this and Eq. 6 into Eq. 5, we find the first-order approximation of the Gibbs potential. The first and second partial derivatives with respect to $P_i(A)$ provide self-consistent equations for the local fields,

$$\frac{P_i(A)}{P_i(q)} = \exp \left(h_i(A) + \sum_{\{j|j\neq i\}} \sum_{B=1}^{q-1} e_{ij}(A, B) P_j(B) \right) \quad (9)$$

and the inverse of the connected correlation matrix,

$$\left. \left(C^{-1}\right)_{ij}(A,B)\right|_{\alpha=0} = \begin{cases} -e_{ij}(A,B) \text{ for } i \neq j \\ \frac{\delta(A,B)}{P_i(A)} + \frac{1}{P_i(q)} \text{ for } i = j \end{cases} . \tag{10}$$

This equation allows us to solve the original inference in the mean-field approximation in a single step. To determine the marginals of the empirical frequencies, we just need to determine the empirical connected correlation matrix and invert it to get the couplings e_{ij}.

C Adaptive MCMC Sampling in adabmDCA

To ensure accurate estimation of the marginals, adabmDCA monitors both equilibration and decorrelation of the Markov chains through sequence overlaps. For two sampled sequences $s_n^i, s_m^k \in \mathcal{A}^L$, the normalized overlap is defined as

$$O\big(s_n^i, s_m^k\big) = \frac{1}{L} \sum_{j=1}^{L} \delta\big(s_n^i(j), s_m^k(j)\big) \tag{11}$$

Three types of overlaps are used:

- External overlap (O_{ext}): between samples from different chains at the same time n.
- Internal-1 overlap (O_{int1}): between consecutive samples of the same chain at times n and $n+1$.
- Internal-2 overlap (O_{int2}): between samples of the same chain at times n and $n+2$.

At each epoch, adabmDCA computes averages μ_α and standard errors σ_α ($\alpha \in \{\text{ext}, \text{int1}, \text{int2}\}$). In equilibrium, all three overlaps should agree within statistical error.

Adaptive update of T_{wait}

To control correlation between successive samples, T_{wait} is adjusted dynamically:

Increase T_{wait} (double it) if:

$$|\mu_{\text{ext}} - \mu_{\text{int2}}| > 5\sqrt{\sigma_{\text{ext}}^2 + \sigma_{\text{int2}}^2} \tag{12}$$

i.e. samples at lag $2T_{\text{wait}}$ are still too correlated.

Reduce T_{wait} (average with last pre-increase value) if:

$$|\mu_{\text{ext}} - \mu_{\text{int1}}| < 5\sqrt{\sigma_{\text{ext}}^2 + \sigma_{\text{int1}}^2} \tag{13}$$

i.e. decorrelation already occurs at lag T_{wait}. Equilibration time is then set to $T_{\text{eq}} = 2T_{\text{wait}}$.

This adaptive scheme ensures that chain samples are equilibrated, initial bias is removed after T_{eq} steps, and decorrelated, independence between samples at lag $T_{\mathrm{wait}} - 2T_{\mathrm{wait}}$. As a result, the estimated marginals match those of the equilibrium Potts distribution, guaranteeing stable gradient estimates and convergence of the Boltzmann machine.