Direct Coupling Analysis for protein contact prediction using neural networks
TPIV - Laboratory of statistical biophysics

Aude Maier

June 8, 2022

1 Introduction

The aim of this work is to combine a Direct Coupling Analysis method for protein structure prediction, pseudo-likelihood maximization, with a machine learning approach. Pseudolikelihood maximization is a statistical method to infer interaction coefficients between residues of a protein family from a Multiple Sequence Alignment. Pseudo-likelihood will be reformulated as a linear (i.e. without hidden layers) neural network and then be generalized to take into account interactions between more than two residues by introducing a hidden layer in the network. Two architectures will be explored for the neural network and two activation functions will be considered, a custom one that squares the output of the hidden layer and a regular tanh activation. The performance of these networks will then be evaluated on the DnaJ domain, as well as their capacity to accurately predict contacts between residues, and compared to the results obtained with the linear model.

2 Theoretical background

2.1 Pseudolikelihood approximation

Multiple Sequence Alignments (MSA) are tables of amino acids whose rows are amino acid sequences belonging to the same protein family aligned to be as similar as possible [1]. The alignment procedure involves the introduction of gaps in the sequences and hence the cells of an MSA take value in a $K = 21$ letters alphabet, 20 symbols for the different types of amino acids and an additional symbol for gaps. Formally an MSA containing N sequences of length L can be written as a matrix $\left(\tilde{X}_{nl}\right) \in \{0, ..., K-1\}^{N\times L}$ where the gaps are attributed the value 0 and each amino acid type is designated by a number between 1 and $K-1$. An alternative representation of an MSA is its one-hot encoded version $(X_{nlk}) \in \{0,1\}^{N\times L\times K}$ defined in the following manner:

$$X_{nlk} = \begin{cases} 1 & \text{if } k = \tilde{X}_{nl} \\ 0 & \text{else} \end{cases} \tag{1}$$

The idea behind Direct Coupling Analysis (DCA) is to modelize the interactions of the residues in a protein with a statistical model and to infer the parameters of the model by maximizing the likelihood of the MSA. Each residue is treated as a spin that can take K different values and that interacts with other residues according to interaction coefficients. The key assumption of DCA is that these coefficients are correlated with the physical distance between the residues (i.e. residues that are close in the 3D-conformation of the protein will interact more). Hence, if one can learn these coefficients, one might be able to predict which residues are or are not in contact in the 3D-conformation.

The statistical model that allows to modelize such a system while maximizing the entropy is the Potts model [1], a generalization of the Ising model where the spins can take more than 2 values. The ernegy corresponding to a sequence $(x_{lk}) \in \{0,1\}^{L\times K}$ (where (x_{lk}) is a short hand notation for $(x_{lk})_{l=1,k=1}^{L,K}$)according to the Potts model is given by:

$$E\left((x_{lk})\right) = -\sum_{l,k} H_{lk}x_{lk} - \sum_{l,k,l',k'} C_{lkl'k'}x_{lk}x_{l'k'} \tag{2}$$

where $(H_{lk}) \in \mathbb{R}^{L\times K}$, the field coefficients, can be seen as a bias for the residue l to be in state k whereas $(C_{lkl'k'}) \in \mathbb{R}^{L\times K\times L\times K}$, the coupling coefficients, represent the influence of the residue l' when it is in state k' on

the residue l to be in state k. Additional terms corresponding to interactions involving more than two residues can be similarly added. The probability of a sequence (x_{lk}) is given by the Boltzmann distribution:

$$p\left((x_{lk})\right) = \frac{e^{-E((x_{lk}))}}{\sum_{\{x_{lk}\}} e^{-E((x_{lk}))}} = \frac{e^{\sum_{l,k} H_{lk}x_{lk} + \sum_{l,k,l'\neq l,k'} C_{lkl'k'}x_{lk}x_{l'k'}}}{\sum_{\{(x_{lk})\}} e^{\sum_{l,k} H_{lk}x_{lk} + \sum_{l,k,l'\neq l,k'} C_{lkl'k'}x_{lk}x_{l'k'}}} \tag{3}$$

The denominator of this expression, which is called the partition function and can be seen as a normalization of the probability, is a sum over all possible configurations $\{(x_{lk})\}$. The number of possible configurations can be counted as L^K which can easily become in the order of 10^{40} for a sequence of length 60, and therefore evaluating the partition function is not tractable computationally. However, if one considers the conditional probability of an amino acid position $(x_{l=i,k})_{k=1}^K$ for some $i = 1, ..., L$ given all the other positions $(x_{lk})_{l\neq i,k}$, then the partition function is cancelled and we are left with :

$$\begin{aligned} p\left((x_{l=i,k})|(x_{lk})_{l\neq i}\right) &= \frac{p\left((x_{l=i,k}), (x_{lk})_{l\neq i}\right)}{\sum_{\{(\tilde{x}_{l=i,k})\}} p\left((\tilde{x}_{l=i,k}), (x_{lk})_{l\neq i}\right)} = \frac{e^{\sum_k x_{ik}\left(H_{ik} + \sum_{l'\neq i,k'} C_{ikl'k'}x_{l'k'}\right)}}{\sum_{\{(\tilde{x}_{l=i,k})\}} e^{\sum_k \tilde{x}_{ik}\left(H_{ik} + \sum_{l'\neq i,k'} C_{ikl'k'}x_{l'k'}\right)}} \\ &= \frac{e^{\sum_k x_{ik}\left(H_{ik} + \sum_{l'\neq i,k'} C_{ikl'k'}x_{l'k'}\right)}}{\sum_{q=1}^K e^{H_{iq} + \sum_{l'\neq i,k'} C_{iql'k'}x_{l'k'}}} \end{aligned} \tag{4}$$

where this time the denominator is a sum over only K terms.

The aim of the pseudolikelihood approximation is to get rid of the partition function in (3) by approximating the probability of a configuration in the following way: [1]

$$p\left((x_{lk})\right) \approx \prod_{i=1}^{L} p\left((x_{l=i,k})|(x_{lk})_{l\neq i}\right) \tag{5}$$

Hence the negative log likelihood that has to be minimized is:

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{l=1}^{L} \log\left(\frac{e^{\sum_k X_{nlk}\left(H_{lk} + \sum_{l'\neq l,k'} C_{lkl'k'}X_{nl'k'}\right)}}{\sum_{q=1}^K e^{H_{lq} + \sum_{l'\neq l,k'} C_{lql'k'}X_{l'k'}}}\right) \tag{6}$$

with (H_{lk}) and $(C_{lkl'k'})$ the parameters to be learned.

As previously introduced, the assumption made in DCA is that residues that have a great influence on each other are likely to be in contact. Therefore the quantity that will be used to predict the contacts in a protein is the couplings matrix $(C_{lkl'k'})$.

2.2 Pseudolikelihood and cross entropy

Using that $\sum_{k=1}^K X_{nik} = 1$, $\forall i = 1, ..., L$, $n = 1, ..., N$, (6) can be rewritten: [2]

$$\begin{aligned} \mathcal{L} &= -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{L} \log\left(\left(\frac{e^{H_{lk} + \sum_{l'\neq l,k'} C_{lkl'k'}X_{nl'k'}}}{\sum_{q=1}^K e^{H_{lq} + \sum_{l'\neq l,k'} C_{lql'k'}X_{l'k'}}}\right)^{\sum_k X_{nlk}}\right) \\ &= -\frac{1}{N}\sum_{n,l,k} X_{nlk} \log\left(\text{softmax}\left(H_{lk} + \sum_{l'\neq l,k'} C_{lkl'k'}X_{nl'k'}\right)\right) \end{aligned} \tag{7}$$

One can observe that this expression is the function of a linear model with cross entropy loss whose architecture is illustrated on Fig. [1] and who takes the MSA (X_{nlk}) as both input and labels. Hence the pseudolikelihood maximization algorithm is equivalent to a linear model trained with softmax activation and cross entropy loss; (H_{lk}) and $(C_{lkl'k'})$ being the bias and weights of the model [2]. As shown in Fig. [1], the architecture consists in an input layer and an output layer of same size containing K neurons for each residue. Each residue in the input layer is fully connected with every other residues in the output layer but are disconnected from their corresponding residue in the output layer, meaning that the K input neurons of the residues are fully connected with the K output neurons of any other residue.

This architecture can be seen as L different classifiers that have to predict the value of a specific residue given all the others. The weight $C_{lkl'k'}$ is then the influence of the residue l' when it takes the value k' on the prediction for the residue l to take the value k.

2.3 Sequence reweighting

In the Potts model, configurations are drawn independently from the Boltzmann distribution (3). In consequence, inference of the coefficients $(C_{lkl'k'})$using the Potts model relies on the assumption that the MSA sequences are statistically independent which, due to phylogeny, bias in the selection of sequenced species, etc., is not true [1]. An approach to mitigate this issue is sequence reweighting; very similar sequences are made less important in the inference of the Potts model by means of weights to avoid their overrepresentation. This method consists to attribute a weight to each sequences of the MSA equal to the inverse number of sequences whose similarity exceeds a defined threshold $0 < x < 1$: [1]

$$w_n = \frac{1}{\sum_{\tilde{n}} \delta_{\{\text{similarity}(X_n, X_{\tilde{n}}) > x\}}} \tag{8}$$

where similarity $(X_n, X_{\tilde{n}}) = \sum_l \delta_{\{X_{nl} = X_{\tilde{n}l}\}}$.

Taking into account the sequence reweighting, the loss for pseudolikelihood maximization in (7) becomes:

$$\mathcal{L} = -\frac{1}{N} \sum_{n,l,k} \frac{1}{w_n} X_{nlk} \log \left(\text{softmax} \left(H_{lk} + \sum_{l' \neq l, k'} C_{lkl'k'} X_{nl'k'} \right) \right) \tag{9}$$

Hence pseudolikelihood maximization with sequence reweighting is still equivalent to a linear model with cross entropy loss but whose labels are the one-hot encoded sequences multiplied by their weights $\frac{1}{w_n} X_{nlk}$ (i.e. the sequences with their 1's replaced by w_n) instead of simply the sequences.

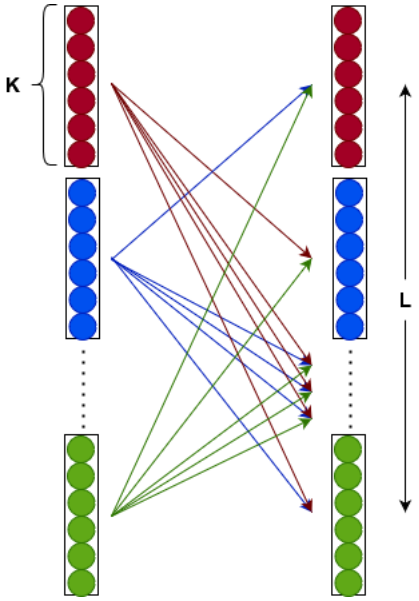


Figure 1: Architecture of the model corresponding to (7). The input layer is a one-hot encoded sequence, each input bloc corresponds to a residue and each bloc contains K neurons with K the number of categories in which a residue can be.

2.4 Interpretability of neural networks

Since pseudolikelihood maximization has been shown equivalent to a linear model, one can wonder if this method could be deepened by adding hidden layers to the architecture of the model. Pseudolikelihood only takes into account bias and interactions between two residues. Inducing non-linearity in the model by adding a hidden layer with a non-linear activation would allow to include the influence of higher order interactions which might make the model more accurate. Including higher order terms would also allow to identify the most correlated triplets, quadruplets, etc. in the sequence and would give keys to investigate further the roles and interactions of the residues.

However, the use of neural networks lead to interpretability issues. Indeed, in the linear case the interaction coefficients were trivially the weights learned by the network whereas when hidden layers and activations are added the meaning of the weights learned by the model becomes much less clear.

Keeping in mind that (X_{nlk}) is filled with 0's and 1's which implies that $X^m_{nlk} = X_{nlk}$ for any $m \in \mathbb{N}$, the function of a model taking a sequence (x_{lk}) as input and returning a sequence (z_{lk}) as output (before softmax activation) can be decomposed in its Taylor expansion as follows: [3]

$$z_{\lambda\kappa}\left((x_{lk})\right) = W^{(1)}_{\lambda\kappa} + \sum_{l \neq \lambda, k} W^{(2)}_{\lambda\kappa lk} x_{lk} + \sum_{l \neq \lambda, k} \sum_{(l' \neq \lambda, l), k'} W^{(3)}_{\lambda\kappa lkl'k'} x_{lk} x_{l'k'} + \ldots \tag{10}$$

The n-th term of (10) describes the contribution of n-th order interactions (i.e. interactions between n residues) to the prediction of z_{lk}. Hence the maximal number of terms in the expansion is L since we cannot have interactions between more than L residues in a sequence of length L [3]. Hence if one approximates this expansion by keeping only the first two terms, then the linear model is recovered and one can identify $H_{\lambda\kappa} = W^{(1)}_{\lambda\kappa}$, $C_{\lambda\kappa lk} = W^{(2)}_{\lambda\kappa lk}$.

It also follows from (10) that $z_{\lambda\kappa}(\mathbf{0}) = W^{(1)}_{\lambda\kappa}$ and $z_{\lambda\kappa}\left((\delta_{l,r}\delta_{k,p})\right) = W^{(1)}_{\lambda\kappa} + W^{(2)}_{\lambda\kappa rp}$ (where $\mathbf{0}$ is a tensor of size $L \cdot K$ filled with zeros and $(\delta_{l,r}\delta_{k,p})$ is a tensor of size $L \cdot K$ filled with zeros except for $x_{rp} = 1$). Consequently, once the model has been trained, the coupling coefficients $C_{\lambda\kappa lk} = W^{(2)}_{\lambda\kappa lk}$ can be obtained using : [3]

$$C_{\lambda\kappa rp} = z_{\lambda\kappa}\left((\delta_{l,r}\delta_{k,p})\right) - z_{\lambda\kappa}(\mathbf{0}) \tag{11}$$

The same reasoning can be applied to obtain the interaction coefficients of any order.

2.5 Ising gauge

The decomposition shown in (10) is not unique. Indeed, due to the property $\sum_k x_{lk} = 1$, it is possible to move a part of the weights from one term to another and consequently there are infinitely many possible sets of weights that satisfy (10) [3]. It is thus necessary when extracting the coupling coefficients $C_{\lambda\kappa lk}$ to fix the gauge that will be used to interpret and compare the results.

For the sake of parsimony, it is preferable to explain the observations by the lowest possible order of interactions. Hence the higher order weights should be reduced as much as possible to favour lower order weights. This is achieved by the Ising gauge, which consists of making the average of the weights over all k zero for each order and each position indices [3]. For instance for the third order weights, the Ising gauge results in the following changes:

$$W^{(3),new}_{\lambda\kappa lkl'k'} = W^{(3)}_{\lambda\kappa lkl'k'} - \frac{1}{K}\sum_k W^{(3)}_{\lambda\kappa lkl'k'} - \frac{1}{K}\sum_{k'} W^{(3)}_{\lambda\kappa lkl'k'} + \frac{1}{K^2}\sum_{k,k'} W^{(3)}_{\lambda\kappa lkl'k'}$$

$$W^{(2),new}_{\lambda\kappa lk} = W^{(2)}_{\lambda\kappa lk} + \sum_{l'\neq l}\frac{1}{K}\sum_{k'}\left(W^{(3)}_{\lambda\kappa lkl'k'} + W^{(3)}_{\lambda\kappa l'k'lk}\right) = W^{(2)}_{\lambda\kappa lk} + 2\sum_{l'\neq l}\frac{1}{K}\sum_{k'} W^{(3)}_{\lambda\kappa lkl'k'}$$

$$W^{(1),new}_{\lambda\kappa} = W^{(1)}_{\lambda\kappa} - \sum_{l,l'\neq l}\frac{1}{K^2}\sum_{k,k'} W^{(3)}_{\lambda\kappa lkl'k'} \tag{12}$$

and similarly for second order corrections:

$$W^{(2),new}_{\lambda\kappa lk} = W^{(2)}_{\lambda\kappa lk} - \frac{1}{K}\sum_k W^{(2)}_{\lambda\kappa lk}$$

$$W^{(1),new}_{\lambda\kappa} = W^{(1)}_{\lambda\kappa} + \sum_l \frac{1}{K}\sum_k W^{(2)}_{\lambda\kappa lk} \tag{13}$$

One can see that for each order, the Ising gauge requires a cascade of corrections on all smaller orders, therefore the higher order terms should be corrected before the lower order ones [3].

3 Methods

The aim of this work is to implement pseudolikelihood maximization for protein contact prediction using a linear model as described in Sec. 2.2 and evaluate its performance on the DnaJ domain. In a second step, the model will be complexified by adding a hidden layer and following the approach developped in Sec. 2.4 to extract the couplings, two architectures and two activation functions will be tested and compared.

Dataset

The MSA of the DnaJ domain (PF00226) has been dowloaded on Pfam with its hidden Markov model. The MSA originally contained 92755 sequences from which the ones contaning more that 10% gaps (after removal of the inserts) have been discarded, reducing the number of sequences to $N = 75518$ of length $L = 63$. The weights of the preprocessed sequences have been computed as described by (8) with a similarity threshold of $x = 80\%$.

In addition to the 20 categories of amino acids and gaps, multiple sequence alignments contain an additional symbole, X. This letter means that the amino acid at this position is unknown or unusual. In general, this symbole appears very few times in the MSA and is therefore not statistically relevant. In this work, the X's will be treated as an additional amino acid type which results in $K = 22$ rather than 21.

Linear model

The first part of this work is to implement and evaluate the linear model whose function is given by (9). The coupling coefficients can be straightforwardly extracted from the trained model since in the linear case they are simply the weights learned. The Ising gauge is then applied to the couplings coefficients according to Sec. 2.5, one can note that only the second order correction is needed since all higher order weights are zero in the linear case.

The pseudolikelihood maximization has no constraint to generate symmetrical couplings (i.e. $C_{\lambda\kappa lk} \neq C_{lk\lambda\kappa}$ in general). It will therefore be worthy of interest to observe how symmetrical or asymmetrical the coupling coefficients learned by the model are. The coefficients will then need to be averaged to extract a unique coupling

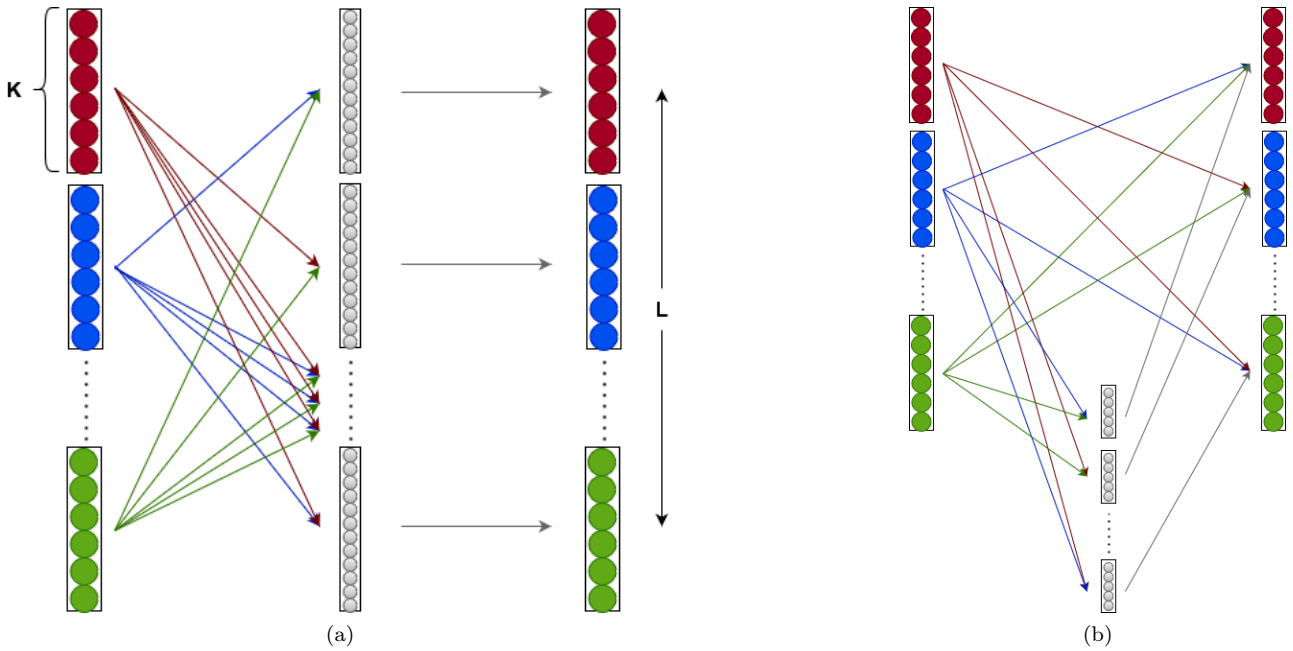


Figure 2: Architecture of two networks generalizing pseudolikelihood maximization by adding non-linearity. The input layer is a one-hot encoded sequence, each input or output bloc corresponds to residue and contains K neurons with K the number of possible categories for the residues. (a) Inputs and outputs are connected via a hidden layer designed such that an output residue is never connected to the corresponding input residue. (b) Inputs and outputs are connected linearly and via a hidden layer still ensuring that residues are not connected to themselves.

between positions and categories $\lambda\kappa$ and lk. Then, to obtain a coefficient that represents how residues λ and l are correlated all categories combined, the Frobenius norm is applied over κ and k:

$$C_{\lambda l} = \sum_{\kappa,k} \mid C_{\lambda\kappa lk} \mid^2 \tag{14}$$

Finally, after average product correction has been applied on the $C_{\lambda l}$'s, contacts in the protein 3D conformation can be predicted by taking the N_{pred} largest coefficients, with N_{pred} a chosen parameter.

To evaluate the quality of contact prediction using the coupling coefficients, the predictions must be compared to the experimentally determined structure of the protein. The PDB file of the DnaJ domain can be dowloaded from the Protein Data Bank (PDB) and mapped to the hidden Markov model of the MSA in order to generate a contact map and identify which of the predicted contacts are effectively close in space.

Non-linear networks

As developped in Sec. 2.4, it is possible to extract the coupling coefficients from a non-linear network, which could enable to take into account higher order interactions between the residues. A second goal of this work will be to introduce a hidden layer in the network. As for the linear case, the K input neurons of residue l must be entirely disconnected from the same K output neurons, otherwise the model would only learn a trivial identity between the input and the output layer. To achieve this, the hidden layer must be structured in L blocs of neurons such that the l-th bloc of the hidden layer is connected to every blocs of the input layer except for the l-th one. The hidden blocs are then connected to their corresponding output bloc (i.e. the l-th hidden bloc is connected solely to the l-th output bloc). This architecture is shown on Fig. 2a. Each bloc of the hidden layer contains the same number of neurons and this number must necessarily be greater than K so as not to induce a loss of information when passing through the hidden layer. In this work we chose 32 neurons per hidden bloc.

To introduce non-linearity in the network, an activation function for the hidden layer must be chosen. The network will first be restricted to interactions up to the third order (i.e. interactions between at most 3 residues) by choosing an activation function that simply squares the neurons of the hidden layer. This way it will be possible to observe what the third order interactions bring to the model compared to the linear case. For this model, both the third and second order Ising gauge will be needed on the coupling coefficients. The performance of the model

with square activation function will also be compared to a regular tanh activation which is not restricted to third order interactions. For this latter case, the Ising gauge should in theory be applied to all orders which generates a number L of corrections on the coupling coefficients. Moreover, the number of weights in the n-th term of the Taylor exapansion in (10) is equal to L^n, which becomes quickly computationally intractable for $n > 3$. In consequence, the Ising gauge will only be applied up to third order corrections even when tanh activation will be used.

As said earlier in this section, the number of hidden neurons for each bloc needs to be greater that K to avoid loss of information. An alternative neural network architercture can be designed to escape from this constraint where the input and output layers would be connected both linearly and via hidden blocs. This architecture is shown on Fig. 2b. This network will be evaluated with square and tanh activation function, similarly as for the previous model, but with only 20 neurons per hidden bloc.

4 Results and discussion

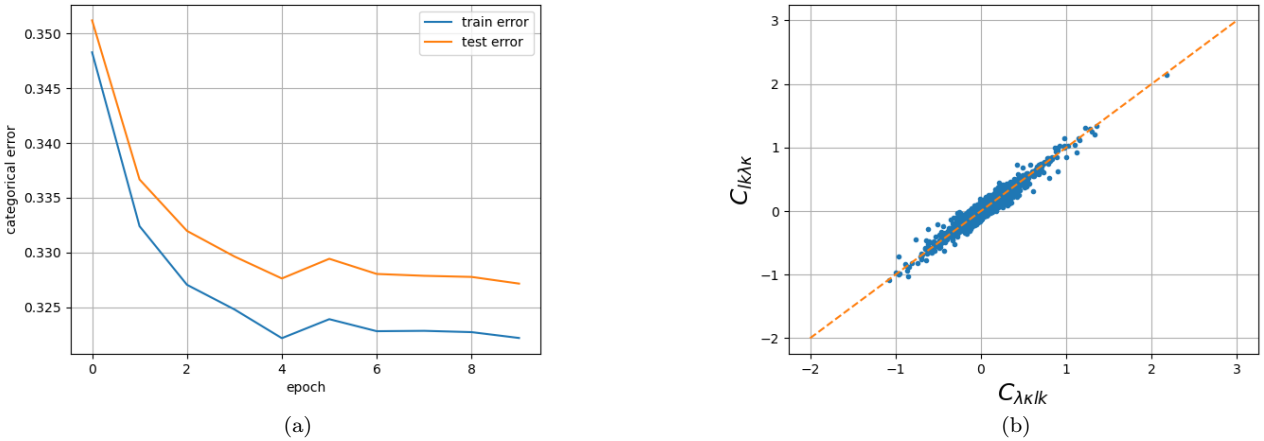
All the results presented in this section have been generated by the scripts that can be found at `https://gitlab.com/LBS-EPFL/neural-network-dca-aude-maier/-/tree/AudeMaier-main-patch-60277`.

4.1 Linear model

First the linear model as presented in Sec. 3 will be evaluated to then be used as a comparison for the performance of non-linear networks.

Performance of the model

Fig. [3a] shows the learning curve of the linear model defined by (9) and Fig. [3b] shows the symmetry of the coupling coefficients learned by the model by plotting $C_{\lambda\kappa lk}$ against $C_{lk\lambda\kappa}$.



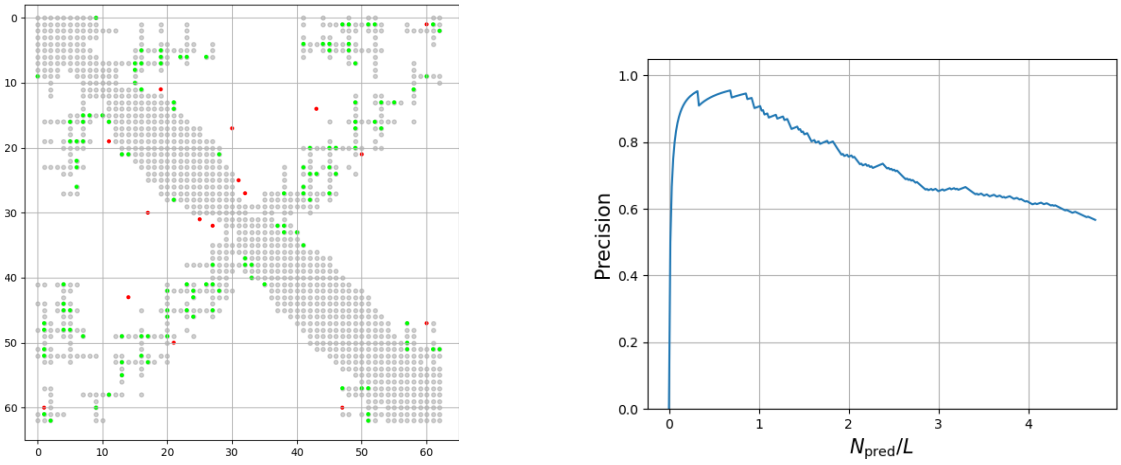
(a) (b)

Figure 3: (a) Categorical error achieved by the linear model (i.e. rate of wrong classifications) after each epoch of training. The training parameters are : learning rate=1e-3, weight decay=1e-1, optimizer=Adam, number of epochs=10. (b) Symmetry of the coupling coefficients $C_{\lambda\kappa lk}$ ($C_{lk\lambda\kappa}$) (before Ising gauge) extracted from the model after training. The dotted line is the identity.

On Fig. [3a], one can see the performance of the model in predicting the value of a residue given all other residues of the sequence. The model converges in less than 10 epochs and has a prediction accuracy of 0.33. Fig [3b] shows that the coupling coefficients are visually close to being symmetrical, since $C_{\lambda\kappa lk}$ ($C_{lk\lambda\kappa}$) closely follows the identity. This means that, according to this model, the influence two residues l and λ have on each other is symmetric. This is coherent with the assumption of DCA stating that the influence of a residue on another is mainly linked to their distance, which is a symmetric quantity.

Contact prediction

Using the coefficients learned by the linear model, Fig. [4a] shows a contact map of the DnaJ domain on which 70 predictions made from the coupling coefficients are displayed in green (for correct predictions) and red (for wrong predictions). Fig. [4b] shows how the rate of correct predictions decays as a function of the number of predictions.



(a) (b)

Figure 4: (a) In grey : contact map of the DnaJ domain where contacts are defined with a threshold of 8.5Å, in colour : 70 predictions made from the coupling coefficients (corresponding to the 70 largest $C_{\lambda l}$), the correct predictions are displayed in green and the incorrect ones in red. (b) Rate of correct predictions as a function of the number of predictions made divided by the length of the sequences L.

On Fig. [4b], one can observe that the very first prediction is incorrect which indicates that the model found a strong link between the residues corresponding to this prediction but that this link was not caused by a physical contact between them. Except for this prediction, the precision is very high for the first 60 contacts (i.e. $\frac{N_{pred}}{L} \approx 1$), above 90%, and then slowly decays.

4.2 Non-linear networks

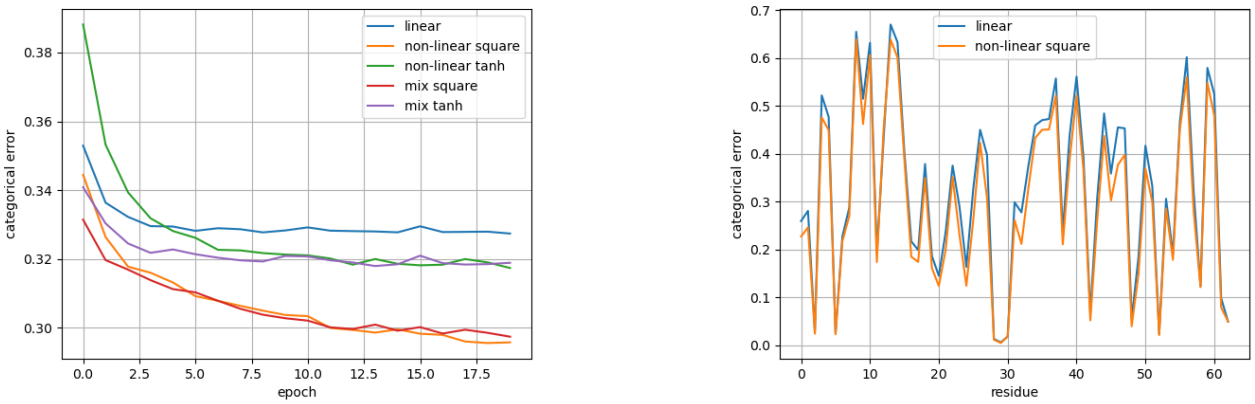
In this section the non-linear networks presented in Sec. 3 will be compared to the linear network. The network where the input and output layer are connected via a hidden layer (Fig. 2a) will be referred to as the non-linear model whereas the networks that connects the inputs and outputs linearly and via a hidden layer will be called mix model. The two models will be analysed with each of the activation functions square and tanh.

Performances

Fig. [5a] shows the categorical test error of each of the five tested networks after each epoch of training and Fig. [5b] shows the test error for each amino acid position of the linear model and the non-linear model with square activation (the worst and best models according to Fig. [5a]) after training.

One can see that the 4 non-linear networks achieve better performance than the linear model. Hence the models are able to incorporate additional properties of the MSA when they can take into account terms of larger orders. However the models with tanh activation have a higher error rate than the ones with square activation even though in theory tanh activation should allow the model to adapt better to the MSA by not being limited in the order of the interactions that are taken into account. One hypothesis for this result could be that the complexity of the tanh function prevents the model from easily learning the simple interactions of the residues which could cause the model to get stuck in local minima or to need more hidden neurons to be optimally exploited. Moreover, the mix models show similar performance to non-linear models while they are trained with significantly less hidden neurons. Therefore, the mix architecture shown on Fig. [2b] allows to use less neurons in the hidden layer whose effect is, among other properties, to speed up the execution time.

The architecture of the mix model with square activation is reminiscent of the structure of (2.4), with the weights of the linear part representing the two first terms of the Taylor expansion and the hidden part with square activation that allows the model to addition the influence of third order interactions to the linear term. Hence, since we saw



(a) (b)

Figure 5: (a) Test error achieved by the networks after each epoch of training. Linear: linear model shown on Fig. [1], non-linear square and tanh: network shown on Fig. [2a] with square and tanh activation function for the hidden layer and 32 neurons per hidden bloc, mix square and tanh: network shown on Fig. [2b] with square and tanh activation and 20 neurons per hidden bloc. (b) Test error per amino acid position after training for the linear model and the non-linear model with square activation. Training parameters : learning rate=1e-3, weight decay=1e-1, optimizer=Adam, number of epochs=20.

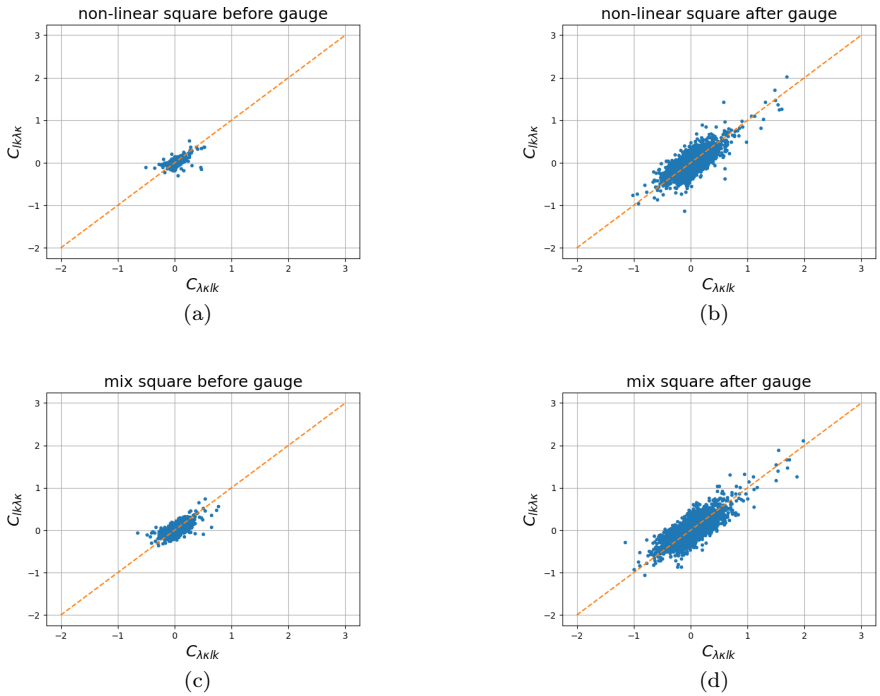
a complex function like tanh can create an overly complicated search space preventing the model's potential from being optimally used, the mix model could be generalized to include an additional set of hidden neurons with cubic activation and so on.

Fig. [5b] shows that there is a large disparity in the predictive ability of the models depending on the residues. One can also see that the improvement brought about by the non-linear model concerns the residues where the error rate is larger whereas at the positions where the error achieved by the linear model is already under 0.1 the non-linear model does not perform better.

Symmetry of coefficients

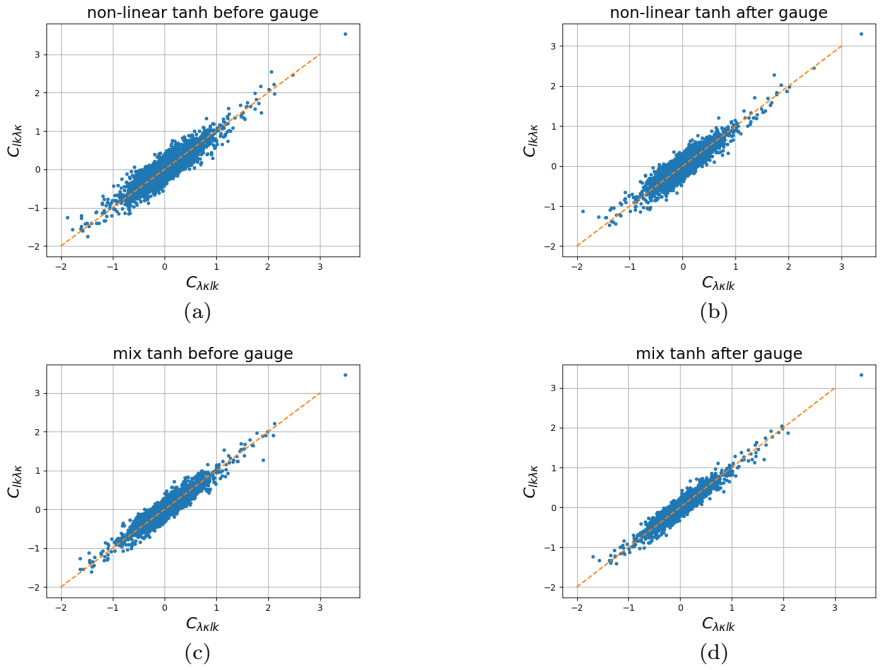
Fig. [6] and [7] show the symmetry of coupling coefficients for all four models (non-linear or mix and tanh or square).

Compared to the linear case on Fig. [3b], the dispertion around the identity is larger for the non-linear and mix models. Moreover, the sprawl is more pronounced for the non-linear models than for the mix models, which can be easily explained by the linear part in the mix models. The shape of the coefficients distribution is less elongated for square activation compared to tanh activation. As it could have been expected, the Ising gauge contributes to reduce the dispertion of the weights for all the models.



(a) (b) (c) (d)

Figure 6: Symmetry of the coupling coefficients $C_{\lambda\kappa lk}$ ($C_{lk\lambda\kappa}$), before (left pannel) and after (right pannel) Ising gauge, extracted from the non-linear (a,b) and mix (c,d) models with square activation after training. The dotted line is the identity.



(a) (b) (c) (d)

Figure 7: Symmetry of the coupling coefficients $C_{\lambda\kappa lk}$ ($C_{lk\lambda\kappa}$), before (left pannel) and after (right pannel) Ising gauge, extracted from the non-linear (a,b) and mix (c,d) models with tanh activation after training. The dotted line is the identity.

Contact prediction

Fig. [8] compares the rate of correct predictions as a function of the number of predictions for all five models that have been implemented in this work.

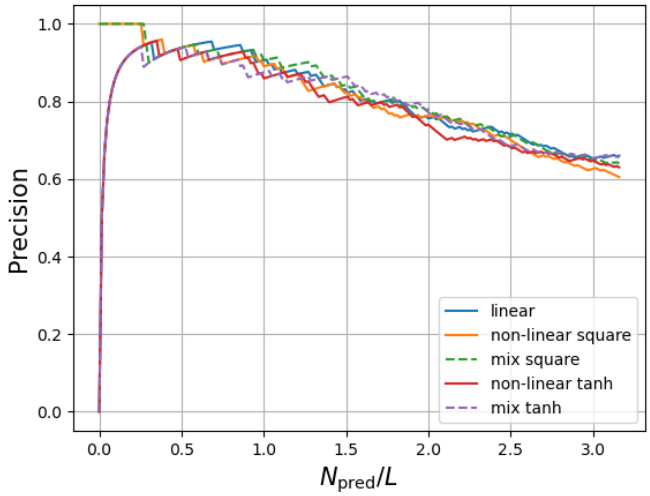


Figure 8: Rate of correct predictions for each of the five models defined in Sec. 3 as a function of the number of predictions made divided by the length of the sequences L. Threshold for the contacts: 8.5Å

One can see on Fig. [8] that from approximately 30 predictions (i.e. $\frac{N_{pred}}{L} \approx 0.5$), the precision of all models is similar. In the linear case, we had identified that the very first prediction was incorrect. This observation is the same for the two models with tanh activation but not with the square activation. Therefore, it means that the two models with square activation managed to identify that this strong link was not related to a contact between the two residues. Except from this observation, these results suggest that the additional information acquired by the non-linear and mix models is not relevant for predicting contacts more accurately. However, Fig. [5a] showed that these models were able to classify more accurately than the linear model and hence that these models had learned more about the statistical model that describes the sequences of the DnaJ domain. This additional knowledge might be useful for other applications than protein contact prediction such as sequence generation methods.

5 Conclusion

In this work, we modelized pseudolikelihood maximization for Direct Coupling Analysis by using a linear neural network. Taking advantage of the one-hot-encoded form of the data, the model was then generalized to take into account interactions involving more than two residues by exploring two distinct architectures, a first one, the non-linear model, where the input and output neurons are connected via a hidden layer and an alternative one, the mix model, where the neurons are connected both linearly and via a hidden layer. Two activation functions have been considered for the hidden layer of the two models, a custom square activation and tanh. The performance of these networks were then evaluated on the DnaJ domain and compared to the results of the linear model. It was found that the use of tanh as an activation function made the performance worse than with the square activation, although still better than the linear case. It has been hypothesized that tanh might make the search space overly complex which could result in the model getting stuck in local minima or requiering more neurons in the hidden layer. The mix model with $20 \cdot L$ neurons in the hidden layer, with L the length of the sequences, were shown to perform similarly as the non-linear model with $32 \cdot L$ hidden neurons. A possible continuation of this work was brought to light by proposing a generalization of the mix model, consisting in adding blocs of neurons to the hidden layer representing interactions between 3 residues or even more. Finally, the ability of the models to predict contacts in the DnaJ domain was analysed and it was observed that the non-linear and mixed models did not show significant improvement in contact prediction compared to the linear model. This last observation suggests that the additional knowledge learned by the non-linear and mix models is not usefull for protein contact prediction. However, the additional information about the statistical model behind the MSA acquiered by the networks might be relevant for other applications, as for instance sequence generation methods.

References

[1] M. Ekeberg, C. Lövkvist, Y. Lan, M. Weigt, E. Aurell, "Improved contact prediction in proteins: Using pseudolikelihoods to infer Potts models," *Physical Review E*, vol. 87, no. 1, Jan. 2013. Accessed: Jun. 5, 2022. doi: 10.1103/physreve.87.012707. [Online]. Available: https://doi.org/10.1103%2Fphysreve.87.012707

[2] J. Dauparas, H. Wang, A. Swartz, P. Koo, M. Nitzan, et S. Ovchinnikov, "Unified framework for modeling multivariate distributions in biological sequences," 2019. Accessed: Jun. 5, 2022. doi: 10.48550/ARXIV.1906.02598. [Online]. Available: https://arxiv.org/abs/1906.02598

[3] S. Zamuner et P. De Los Rios, "Interpretable Neural Networks based classifiers for categorical inputs," 2021. Accessed: Jun. 5, 2022. doi: 10.48550/ARXIV.2102.03202. [Online]. Available: https://arxiv.org/abs/2102.03202