Bayesian Model for Prediction of Protein Residue-Residue Contacts

Susann Vorberg

15.10.2017

Dissertation zur Erlangung des Doktorgrades der Fakultt fr
Chemie und Pharmazie der Ludwig-Maximilians-Universitt
Mnchen

Bayesian Model for Prediction of Protein Residue-Residue Contacts

vorgelegt von
Susann Vorberg
geboren in Leipzig, Germany

Mnchen, den 15.10.2017

Erklrung

Diese Dissertation wurde im Sinne von 7 der Promotionsordnung vom 28. November 2011 von Dr. Johannes Soeding betreut.

Eidesstattliche Versicherung

Diese Dissertation wurde eigenstndig und ohne unerlaubte Hilfe erarbeitet.

. . . .
Ort, Datum   Susann Vorberg

Dissertation eingereicht am: 15.10.2017

Erstgutachter: Dr. Johannes Soeding . . . .

Zweitgutachter: Prof. Dr. Julien Gagneur . . . .

Tag der mndlichen Prfung: 15.12.2017

Summary

Awesome contact prediction project abstract

Acknowledgements

I thank the world.

Table of Contents

1

Interpretation of Coupling Matrices

Contact prediction methods learning a *Potts model* for the MSA of a protein familiy, map the inferred 20 x 20 dimensional coupling matrices w_{ij} onto scalar values to obtain contact scores for each residue pair as outlined in section **??**. As a result, the full information contained in coupling matrices is lost, such as the contribution of individual couplings w_{ijab}, whether a coupling is positive or negative, higher order dependencies between couplings or possibly biological meaningful signals. The following sections give some intuition for the information contained in coupling matrices.

1.1 Single Coupling Values Carry Evidence of Contacts

Given the success of DCA methods, it is clear that the inferred couplings $\mathbf{w}_{ij}$ are good indicators of spatial proximity for residue pairs. As described in section **??**, a contact score $C_{i,j}$ for a residue pair (i, j) is commonly computed as the Frobenius norm over the coupling matrix, $C_{i,j} = ||\mathbf{w}_{ij}||_2 = \sqrt{\sum_{a,b=1}^{20} w_{ijab}}$.

The left plot in Figure 1.1 shows the correlation between squared coupling values $(w_{ijab})^2$ and binary contact class (contact=1, non-contact=0) for approximately 100.000 residue pairs per class (for details see methods section 3.7.1). All couplings have a positive class correlation, meaning the stronger the squared coupling value, the more likely a contact can be inferred. Generally, couplings that involve an aliphatic amino acid such as isoleucine (I), leucine (L), valine (V) or an alanine (A) express the strongest class correlation. In contrast, cysteine pairs (C-C) or pairs involving only the charged residus arginine (R), glutamic acid (E), lysine (K) or aspartic acid (D) correlate only weakly with contact class. Interestingly, C-C and couplings involving charged residues have the highest standard-deviation among all couplings as can be seen in the right plot in Figure 1.1. It can be hypothesized that these couplings considerably contribute to false positive predictions when

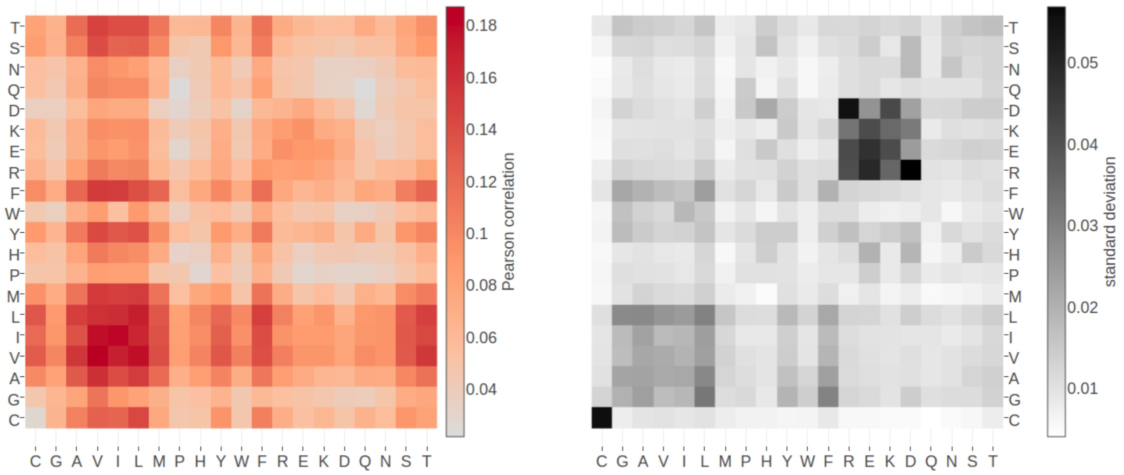

Figure 1.1: **Left** Pearson correlation of squared coupling values $(w_{ijab})^2$ with contact class (contact=1, non-contact=0). **Right** Standard deviation of squared coupling values. Dataset contains 100.000 residue pairs per class (for details see methods section 3.7.1). Amino acids are abbreviated with one-letter code and they are broadly grouped with respect to physico-chemical properties listed in Appendix A.1.

using the Frobenius norm as a contact score because they can have high squared values (high standard deviation) that do not correlate well with being a contact.

Different couplings are of varying importance for contact inference and have distinct characteristics. When looking at the raw coupling values (without squaring), these charateristics become even more pronounced. The left plot in Figure 1.2 shows the correlation of raw coupling values w_{ijab} with contact class. Interestingly, in contrast to the findings for squared coupling values, couplings for charged residue pairs, involving arginine (R), glutamic acid (E), lysine (K) and aspartic acid (D), have the strongest class correlation (positive and negative), whereas aliphatic coupling pairs correlate to a much lesser extent. This implies that squared coupling value is a better indicator of a contact than the raw signed coupling value for aliphatic couplings. On the contrary, the raw signed coupling values for charged residue pairs are much more indicative of a contact than the magnitude of their squared values. Raw couplings for cysteine (C-C) pairs, proline (P) and tryptophane (W) correlate only weakly with contact class. For these pairs neither a squared coupling value nor the raw coupling value seems to be a good indicator for a contact.

Looking only at correlations can be misleading if there are non-linear patterns in the data, for example higher order dependencies between couplings. For this reason it is advisable to take a more detailed view at coupling matrices and the distributions of their values.

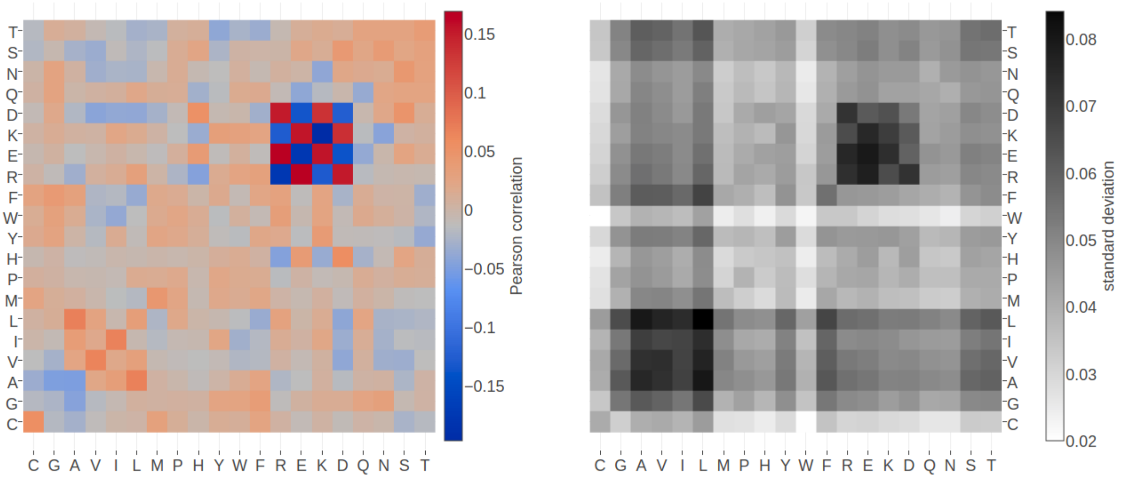

Figure 1.2: **Left** Pearson correlation of raw signed coupling values w_{ijab} with contact class (contact=1, non-contact=0). **Right** Standard deviation of coupling values. Dataset contains 100.000 residue pairs per class (for details see section 3.7.1). Amino acids are abbreviated with one-letter code and they are broadly grouped with respect to physico-chemical properties listed in Appendix A.1.

1.2 Physico-Chemical Fingerprints in Coupling Matrices

The correlation analysis of coupling matrices in the last section revealed that certain couplings are more indicative of a contact than others. Individual coupling matrices for a residue pair that is in physcial contact often display striking patterns that agree with the previous findings. These patterns allow a biological interpreation of the coupling values that reveal details of the physico-chemical interdependency between both residues.

Figure 1.3 visualizes the inferred coupling matrix for a residue pair using the pseudo-likelihood method. A cluster of strong coupling values can be observed for the couplings between the charged residues glutamic acid (E), aspartic acid (D), lysine (K) and arginine (R) and the polar residue glutamine (Q). Positive coupling values arise between positively charged residues (K, R) and negatively charged residues (E, D), whereas couplings between equally charged residues have negative values. These exemplary couplings (E-R, E-K, K-D) perfectly reflect the interaction preference for residues forming salt bridges. Indeed, in the protein structure the first residue (E) forms a salt bridge with the second residue (R) as can be seen in the left plot in Figure 1.5.

Figure 1.4 visualizes the coupling matrix for a pair of hydrophobic residues. Hydrophobic pairings, such as alanine (A) - isoleucine (I), or glycine (G) - isoleucine (I) have strong coupling values but the couplings also reflect a sterical constraint. Alanine is a small hydrophobic residue and it is favoured at both residue positions because it has strong positive couplings with isoleucine (I), leucine (L) and methionine (M). But alanine is disfavoured to appear at both positions at the same time as the A-A coupling is negative. Figure 1.5 illustrates the location of the two residues in the protein core. Here, hydrophobic residues are densely packed and

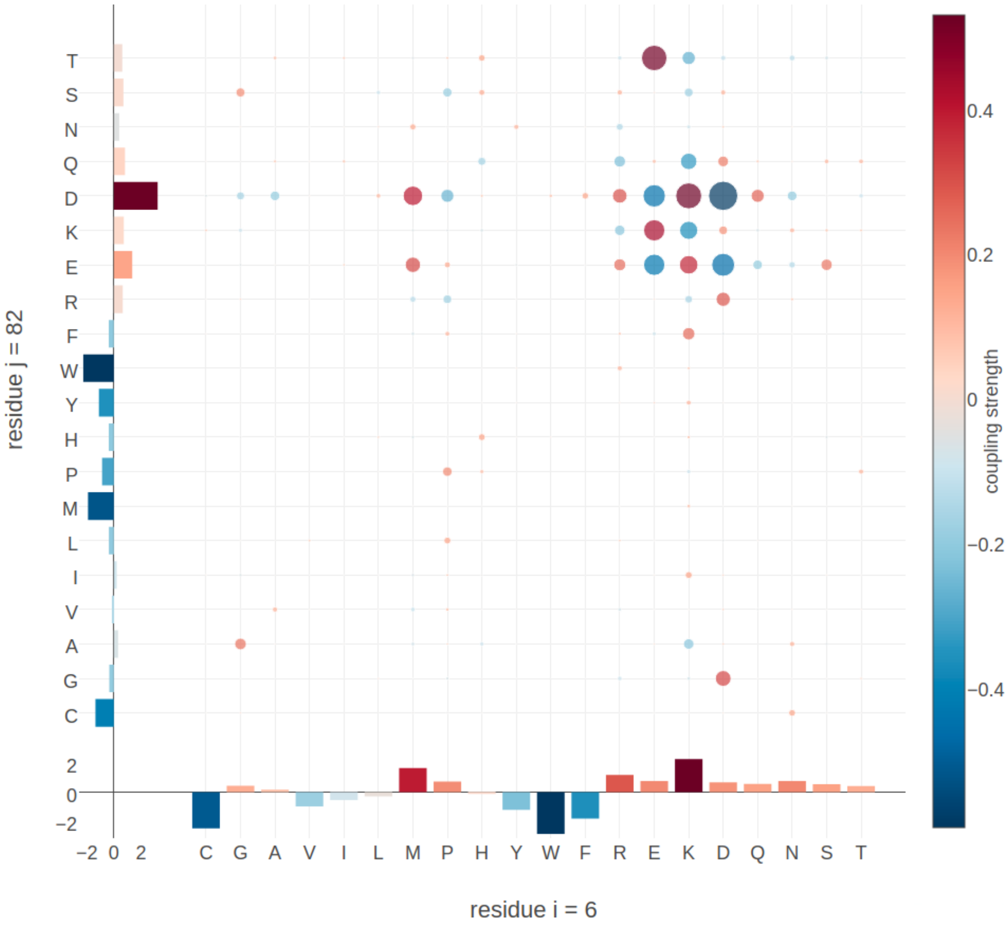

Figure 1.3: Coupling matrix computed with pseudo-likelihood for residues 6 and 82 in protein chain 1a9x_A_05. Color represents coupling strength and direction (red = positive coupling value, blue = negative coupling value) and diameter of bubbles represents absolute coupling value $|w_{ijab}|$. Bars at the x-axis and y-axis correspond to the *Potts model* single potentials v_i and v_j. Amino acids are abbreviated with one-letter code and they are broadly grouped with respect to physico-chemical properties listed in Appendix A.1.

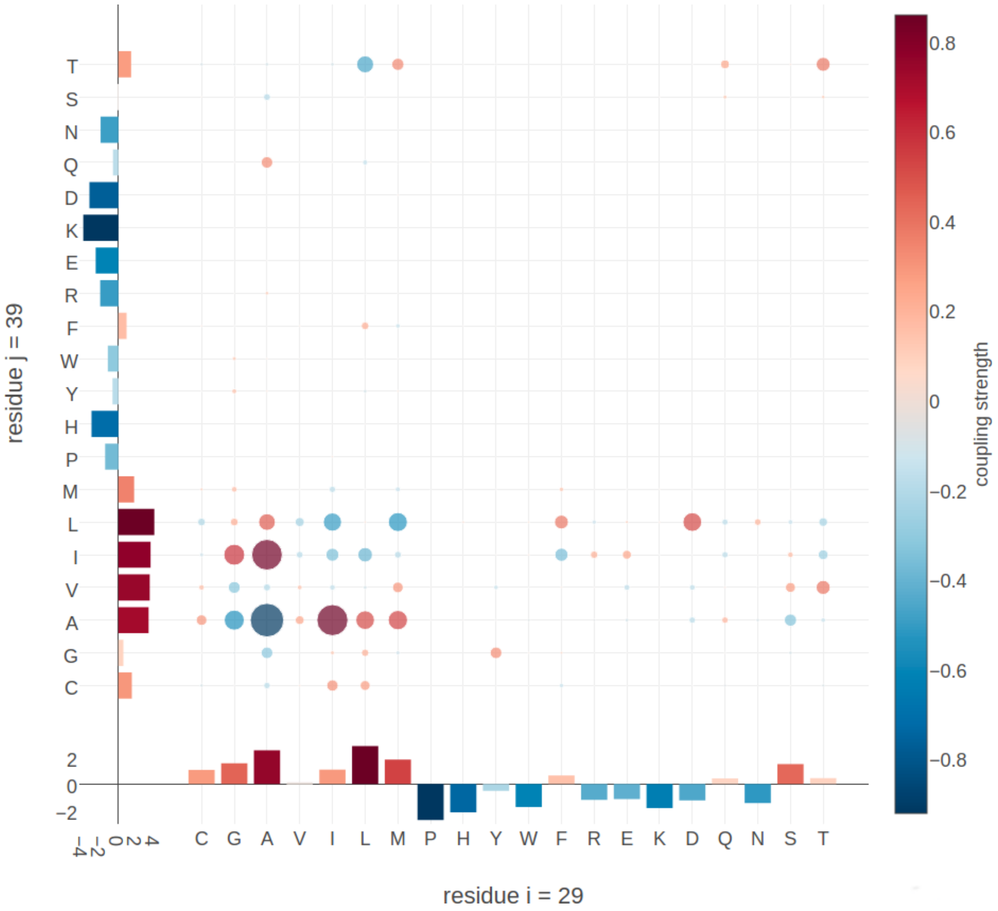

Figure 1.4: Coupling matrix computed with pseudo-likelihood for residues 29 and 39 in protein chain 1ae9_A_00. Color represents coupling strength and direction (red = positive coupling value, blue = negative coupling value) and diameter of bubbles represents absolute coupling value $|w_{ijab}|$. Bars at the x-axis and y-axis correspond to the *Potts model* single potentials v_i and v_j. Amino acids are abbreviated with one-letter code and they are broadly grouped with respect to physico-chemical properties listed in Appendix A.1.

the limited space allows for only small hydrophobic residues.

Many more biological interpretable signals can be identified from coupling matrices, including pi-cation interactions (see Appendix C.1), aromatic-proline interactions (see Appendix C.3), sulfur-aromatic interactions or disulphide bonds (see Appendix C.2).

Coucke and collegues performed a thorough quantitative analysis of coupling matrices selected from confidently predicted residue pairs [1]. They showed that eigenmodes obtained from a spectral analysis of averaged coupling matrices are closely related to physico-chemical properties of amino acid interactions, like electrostaticity, hydrophobicity, steric interactions or disulphide bonds. By looking at specific populations of residues, like buried and exposed residues or residues from specific protein classes (small, mainly α, etc), the eigenmodes of corresponding coupling matrices are found to capture very characteristic interactions for each class, e.g. rare disulfide contacts within small proteins and hydrophilic contacts between exposed residues. Their study confirms the qualitative observations pre-

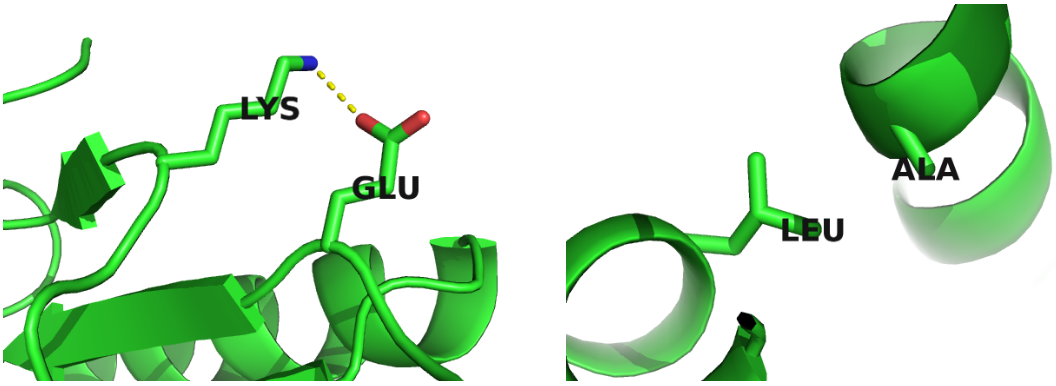

Figure 1.5: Interactions between protein side chains. **Left**: residue 6 (E) forms a salt bridge with residue 82 (R) in protein chain 1a9x_A_05. **Right**: residue 29 (A) and residue 39 (L) within the hydrophobic core of protein chain 1ae9_A_00.

sented above that amino acid interactions can leave characteristic physico-chemical fingerprints in coupling matrices.

1.3 Coupling Profiles Vary with Distance

Analyses in the previous sections showed that certain coupling values correlate more or less strong with contact class and that coupling matrices for contacts express biological meaningfull patterns.

More insights can be obtained by looking at the distribution of distinct coupling values for contacts, non-contacts and arbitrary populations of residue pairs. Figure 1.6 shows the distribution of selected couplings for filtered residue pairs with $C_\beta - C_\beta$ distances $< 5\mathring{A}$ (see methods section 3.7.2 for details). The distribution of R-E and E-E coupling values is shifted and skewed towards positive and negative values respectively. This is in accordance with attracting electrostatic interactions between the positively charged side chain of arginine and the negatively charged side chain of gluatamic acid and also with repulsive interactions between the two negatively charged glutamic acid side chains. Coupling values for cysteine pairs (C-C) have a broad distribution that is skewed towards positive values, reflecting the strong signals obtained from covalent disulphide bonds. The broad distribution for C-C, R-E and E-E agrees with the observation in section 1.1 that these specific coupling values have large standard deviations and that for charged residue pairings the signed coupling value is a strong indicator of a contact.

Hydrophobic pairs like V-I have an almost symmetric coupling distribution, confirming the finding that the direction of coupling is not indicative of a true contact whereas the strength of the coupling is. The hydrophobic effect that determines hydrophobic interactions is not specific or directed. Therefore, hydrophobic interaction partners can commonly be substituted by other hydrophobic residues, which explains the not very pronounced positive coupling signal compared to more specific interactions, e.g ionic interactions. The distribution of aromatic coupling values like F-W is slightly skewed towards negative values, accounting for steric hindrance of their large side chains at small distances.

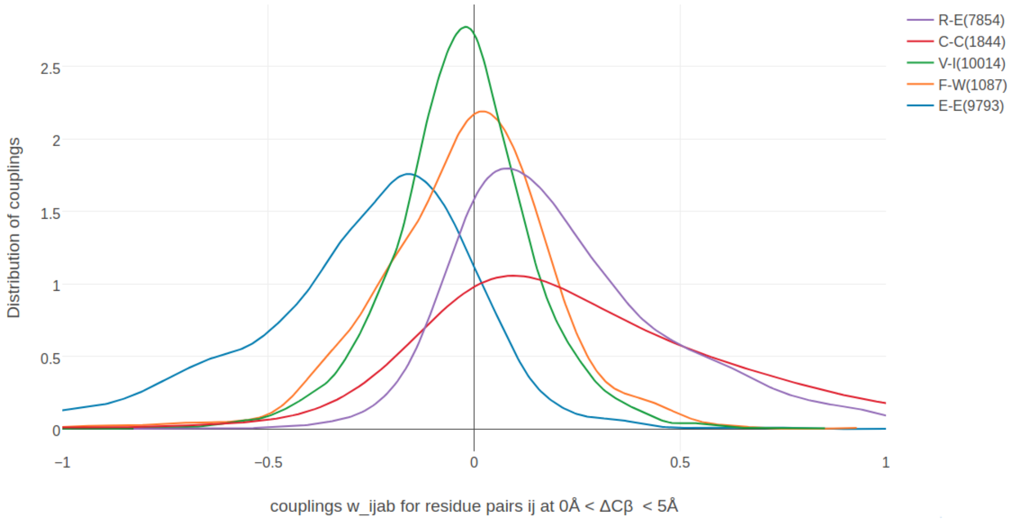

Figure 1.6: Distribution of selected couplings for filtered residue pairs with $C_\beta - C_\beta$ distances $< 5\AA$ (see methods section 3.7.2 for details). Number of coupling values used to determine the distribution is given in brackets in the legend. R-E = couplings for arginine and glutamic acid pairs, C-C = coupling for cystein residue pairs, V-I = coupling for valine and isoleucine pairs, F-W = coupling for phenylalanine and tryptophane pairs, E-E = coupling for glutamic acid residue pairs.

In an intermediate C_β distance range between $8\AA$ and $12\AA$ the distributions for all coupling values are centered close to zero and are less broad. The distributions are still shifted and skewed, but less pronounced compared to the distributions at $C_\beta - C_\beta$ distances $< 5\AA$. For aromatic pairs like F-W, the distribution of coupling values has very long tails, suggesting rare but strong couplings for aromatic side chains at this distance.

Figure 1.8 shows the distribution of selected couplings for residue pairs far apart in the protein structure ($C_\beta - C_\beta$ distances $> 20\AA$).
The distribution for all couplings is centered at zero and has small variance. Only for C-C coupling values, the distribution has a long tail for positve values, presumably arising from the fact that the maximum entropy model cannot distuinguish highly conserved signals of multiple disulphide bonds within a protein. This observation also agrees with the previous finding in section 1.1 that C-C coupling values, albeit having large standard-deviations, correlate only weakly with contact class. The same arguments apply to couplings of aromatic pairs that have a comparably broad distribution and do not correlate strongly with the contact class. The strong coevolution signals for aromatic pairs even at high distance ranges might result from transitive effects that could not be completely resolved by the *Potts model*. Aromatic residues are known to form network-like structures in the protein core that stabilize protein structure and can lead to transitive effects (see Figure C.7 in Appendix)[2].

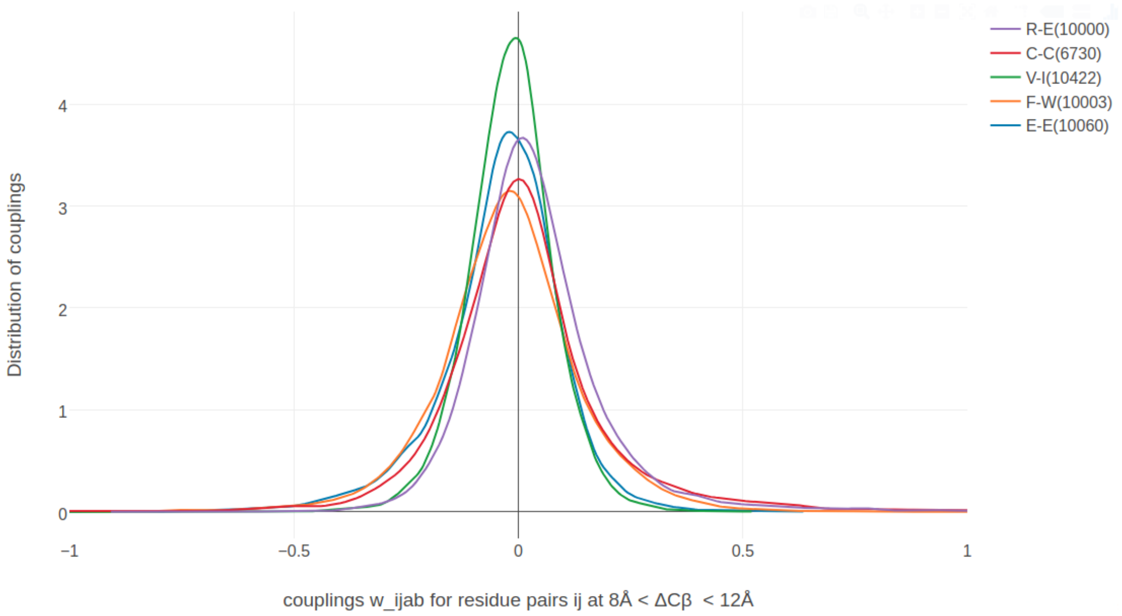

Figure 1.7: Distribution of selected couplings for filtered residue pairs with $C_\beta - C_\beta$ distances between $8\mathring{A}$ and $12\mathring{A}$ (see methods section 3.7.2 for details). Number of coupling values used to determine the distribution is given in brackets in the legend. Couplings are the same as in Figure 1.6.

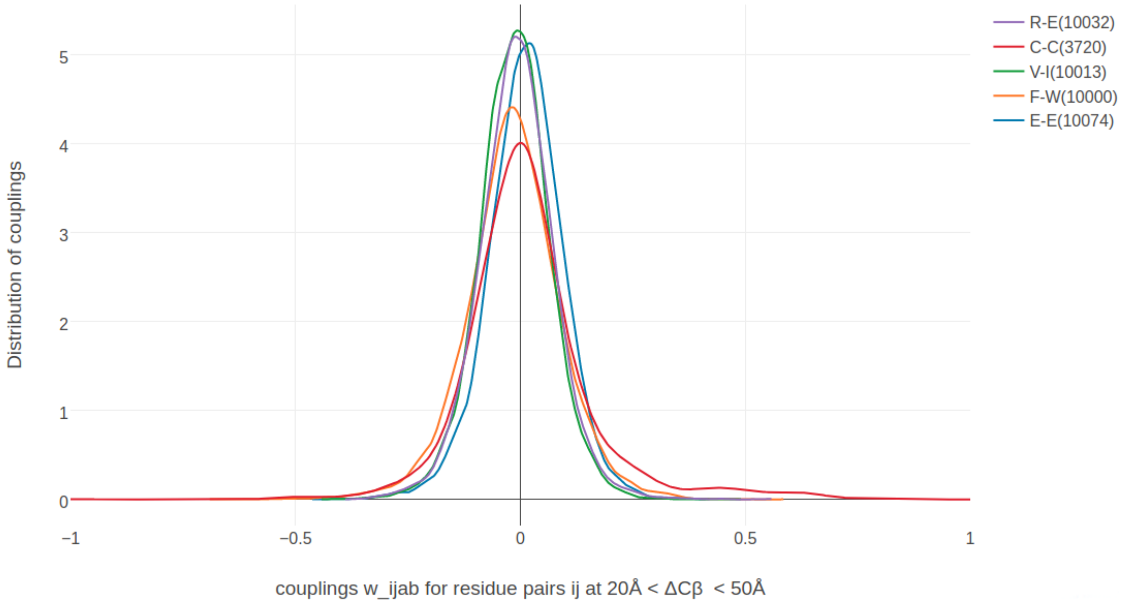

Figure 1.8: Distribution of selected couplings for filtered residue pairs with $C_\beta - C_\beta$ distances between $20\mathring{A}$ and $50\mathring{A}$ (see methods section 3.7.2 for details). Number of coupling values used to determine the distribution is given in brackets in the legend. Couplings are the same as in Figure 1.6.

1.4 Higher Order Dependencies Between Couplings

The analyses in the previous sections focused on single coupling values picked from the 20×20-dimensional coupling matrices $\mathbf{w}_{ij}$. As mentioned before, analysing only single dimensions might be misleading when variables are dependent on each other and further insights might be concealed in higher order relationships. Unfortunately, it is not possible to reasonably visualize high dimensional coupling matrices.

Exploring two dimensional coupling scatter plots strengthens the observation that couplings matrices contain signals that reflect biological relevant amino acid interactions. The plots in the top row in Figure 1.9 show the distribution of couplings for filtered residue pairs with $C_\beta - C_\beta$ distances $< 8\mathring{A}$ between the ionic pairings of E-R and R-E and between the ionic pairing R-E and the equally charged residues E-E, respectively. Coupling values for R-E and E-R are positively correlated with predominantly positive values. This means when the amino acid pair R-E is frequently observed at two positions i and j, then it also likely that the amino acid pair E-R can be frequently observed. This situation indicates an important ionic interaction whereby the location of the positively and negatively charged residue at position i or j is irrelevant.

On the contrary, coupling values for R-E and E-E are negatively correlated, with positive values for R-E and negative values for E-E. This distribution can be interpreted with frequently occuring amino acid pairs R-E at two positions i and j while at the same time the amino acid pair E-E cannot be observed. Again, this situation coincides with amino acid pairings that would be expected for an ionic interaction.

The bottom left plot in Figure 1.9 shows the distribution between couplings for the hydrophobic pairings I-L and V-I that is almost symmetric and broadly centered around zero. Coupling distributions for residue pairs that are not physically interacting ($C_\beta \gg 8\mathring{A}$) resemble the distribution for hydrophobic pairings in that there is no correlation, but at high distance the distributions are much tighter centered around zero (bottom right plot in Figure 1.9).

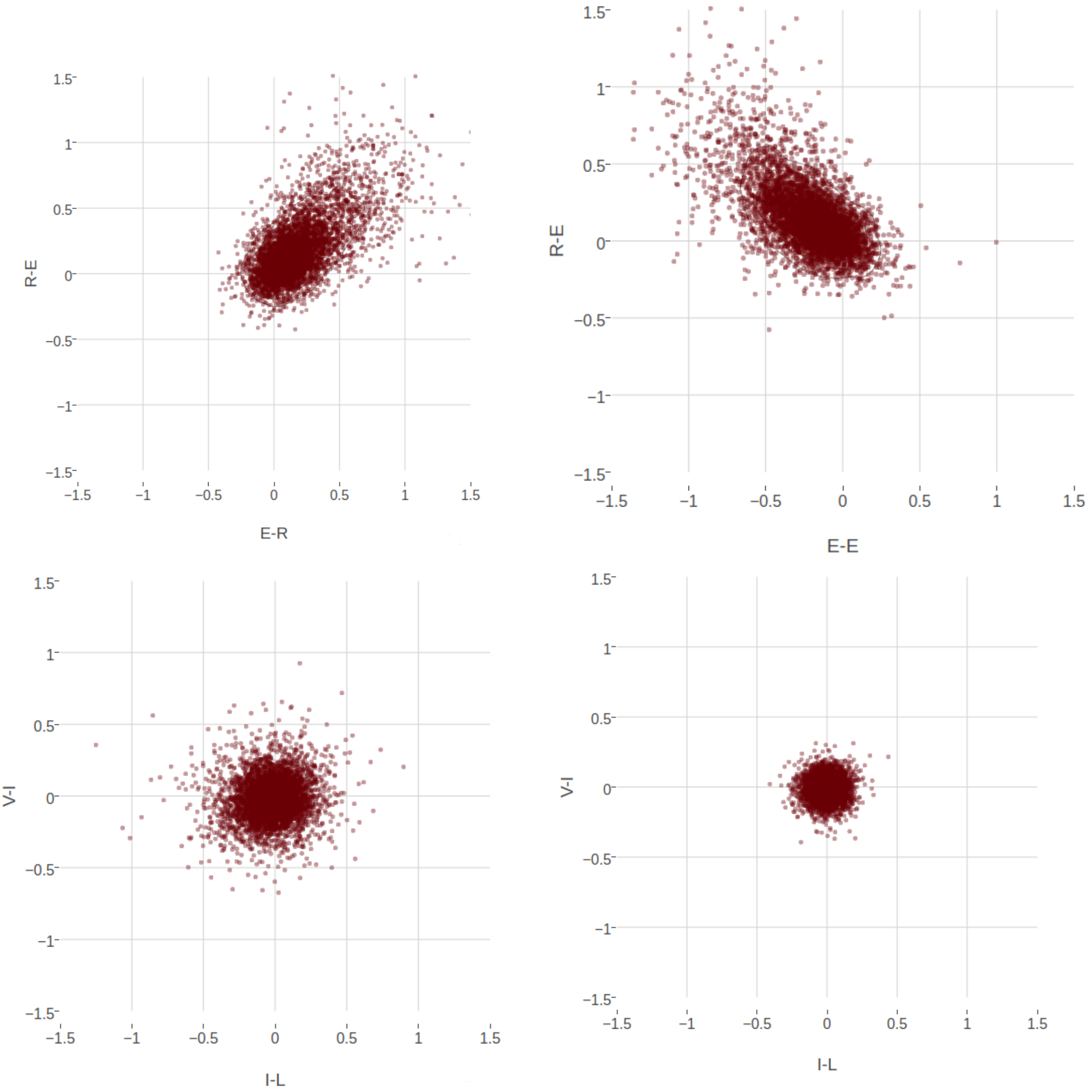

Figure 1.9: Two-dimensional distribution of approximately 10000 coupling values computed with pseudo-likelihood. **Top Left** The 2-dimensional distribution of couplings E-R and R-E for residue pairs with $C_\beta - C_\beta$ distances $< 8\mathring{A}$ is almost symmetric and the coupling values are positively correlated. **Top Right** The 2-dimensional distribution of couplings E-R and E-E for residue pairs with $C_\beta - C_\beta$ distances $< 8\mathring{A}$ is almost symmetric and the coupling values are negatively correlated. **Bottom Left** The 2-dimensional distribution of couplings I-L and V-I for residue pairs with $C_\beta - C_\beta$ distances $< 8\mathring{A}$ is symmetrically distributed around zero without visible correlation. **Bottom Right** The 2-dimensional distribution of couplings I-L and V-I for residue pairs with $C_\beta - C_\beta$ distances $> 20\mathring{A}$ is tighly distributed around zero. .

2
Contact Prior

The wealth of successful meta-predictors presented in section **??** highlights the importance to exploit other sources of information apart from coevolution statistics. Much information about residue interactions is typically contained in single position features that can be predicted from local sequence profiles, such as secondary structure, solvent accessibility or contact number, and in pairwise features such as the contact prediction scores for residue pairs (i, j) from a simple local statistical methods as presented in section **??**.

For example, predictions of secondary structure elements and solvent accessibility are used by almost all modern machine learning predictors, such as MetaPsicov [3], NeBCon [4], EPSILON-CP [5], PconsC3 [6]. Other frequently used sequence derived features include pairwise contact potentials, sequence separation and conservation measures such as column entropy [3,4,7].

In the following sections I present a random forest classifier that uses sequence derived features to distinguish contacts from non-contacts. Methods section 3.13.1 lists all features used to train the classifier including the aforementioned standard features as well as some novel features.

The probabilistic predictions of the random forest model can be introduced directly as prior information into the Bayesian statistical model presented in the last section **??** to improve the overall prediction accuracy in terms of posterior probabilities. Furthermore, contact scores from coevolution methods can be added as additional feature to the random forest model in order to elucidate how much the combined information improves prediction accuracy over the single methods.

2.1 Random Forest Classifiers

Random Forests are supervised machine learning methods that belong to the class of ensemble methods [8–10]. They are easy to implement, fast to train and can handle large numbers of features due to implicit feature selection [11].

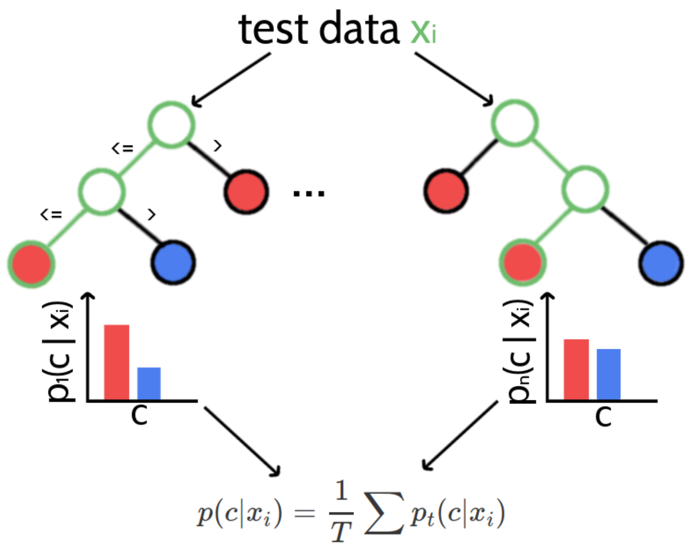

Figure 2.1: Classifying new data with random forests. A new data sample is run down every tree in the forest until it ends up in a leaf node. Every leaf node has associated class probabilities $p(c)$ reflecting the fraction of training samples belonging to every class c. The color of the leaf nodes reflects the class with highest probability. The predictions from all trees in form of the class probabilties are averaged over all trees and yield the final prediction.

Ensemble methods combine the predictions of several independent base estimators with the goal to improve generalizability over a single estimator. Random forests are ensembles of decision trees where randomness is introduced in two ways:

1. every tree is build on a random sample that is drawn with replacement from the training set and has the same size as the training set (i.e., a bootstrap sample)
2. every split of a node is evaluated on a random subset of features

A single decision tree, especially when it is grown very deep is highly susceptible to noise in the training set and therefore prone to overfitting which results in poor generalization ability. As a consequence of randomness and averaging over many decision trees, the variance of a random forest predictor decreases and therefore the risk of overfitting [12]. It is still advisable to restrict the depth of single trees in a random forest, not only to counteract overfitting but also to reduce model complexity and to speedup the algorithm.

Random forests are capable of regression and classification tasks. For classification, predictions for new data are obtained by running each data sample down every tree in the forest and then either apply majority voting over single class votes or averaging the probabilistic class predictions. Probabilistic class predictions of single trees are computed as the fraction of training set samples of the same class in a leaf whereas the single class vote refers to the majority class in a leaf. Figure 2.1 visualizes the procedure of classifying a new data sample.

Typically, *Gini impurity*, which is a computationally efficient approximation to the entropy, is used as a split criterion to evaluate the quality of a split. It measures the degree of purity in a data set regarding class labels as $GI = (1 - \sum_{k=1}^{K} p_k^2)$, where p_k is the proportion of class k in the data set. For every feature f in the random subset that is considered for splitting a particular node N, the *decrease in Gini impurity* ΔGI_f will be computed as,

$$\Delta GI_f(N_{\text{parent}}) = GI_f(N_{\text{parent}}) - p_{\text{left}} GI_f(N_{\text{left}}) - p_{\text{right}} GI_f(N_{\text{left}})$$

where p_{left} and p_{right} refers to the fraction of samples ending up in the left and right child node respectively [11]. The feature f with highest ΔGI_f over the two resulting child node subsets will be used to split the data set at the given node N.

Summing the *decrease in Gini impurity* for a feature f over all trees whenever f was used for a split yields the *Gini importance* measure, which can be used as an estimate of general feature relevance. Random forests therefore are popular methods for feature selection and it is common practice to remove the least important features from a data set to reduce the complexity of the model. However, feature importance measured with respect to *Gini importance* needs to be interpreted with care. The random forest model cannot distinguish between correlated features and it will choose any of the correlated features for a split, thereby reducing the importance of the other features and introducing bias. Furthermore, it has been found that feature selection based on *Gini importance* is biased towards selecting features with more categories as they will be chosen more often for splits and therefore tend to obtain higher scores [13].

2.2 Evaluating Random Forest Model as Contact Predictor

I trained a random forest classifier on the feature set described in methods section 3.13.1 and optimized model hyperparameters as well as some data set specific settings (e.g window size and class ratios) with 5-fold cross-validation as described in methods section 3.13.2.

Figure 2.2 shows the ranking of the ten most important features according to *Gini importance*. Both local statistical contact scores, *OMES* [14] and MI (mutual information between amino acid counts), constitute the most important features besides the mean pair potentials according to Miyazawa & Jernigan [15] and Li&Fang[16]. Further important features include the relative solvent accessibility at both pair positions, the total percentage of gaps at both positions, the correlation between mean isoelectric point property at both positions, sequence separation and the beta-sheet propensity in a window of size five around position i.

Many features have low *Gini importance* scores which means they are rarely considered for splitting a node and can likely be removed from the dataset. Removing irrelevant features from the dataset is a convenient procedure to reduce model complexity. As described in methods section 3.13.3, I performed feature selection by evaluating model performance on subsets of features of decreasing importance. Most models trained on subsets of the total feature space perform nearly identical compared to the model trained on all features, as can be seen in Figure 2.3.

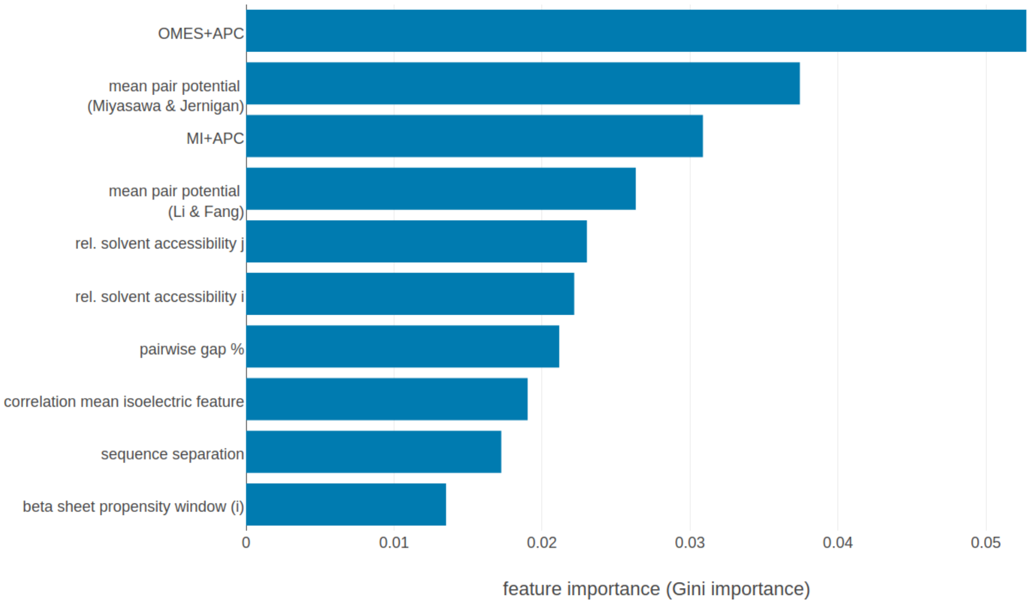

Figure 2.2: Top ten features ranked according to *Gini importance*. **OMES+APC**: APC corrected OMES score according to Fodor&Aldrich [14]. **mean pair potential (Miyasawa & Jernigan)**: average quasi-chemical energy of transfer of amino acids from water to the protein environment [15]. **MI+APC**: APC corrected mutual information between amino acid counts (using pseudo-counts). **mean pair potential (Li&Fang)**: average general contact potential by Li & Fang [16]. **rel. solvent accessibilty i(j)**: RSA score computed with Netsurfp (v1.0) [17] for position i(j). **pairwise gap%**: percentage of gapped sequences at either position i and j. **correlation mean isoelectric feature**: Pearson correlation between the mean isoelectric point feature (according to Zimmermann et al., 1968) for positions i and j. **sequence separation**: |j-i|. **beta sheet propensity window(i)**: beta-sheet propensity according to Psipred [18] computed within a window of five positions around i. eatures are described in detail in methods section 3.13.1.

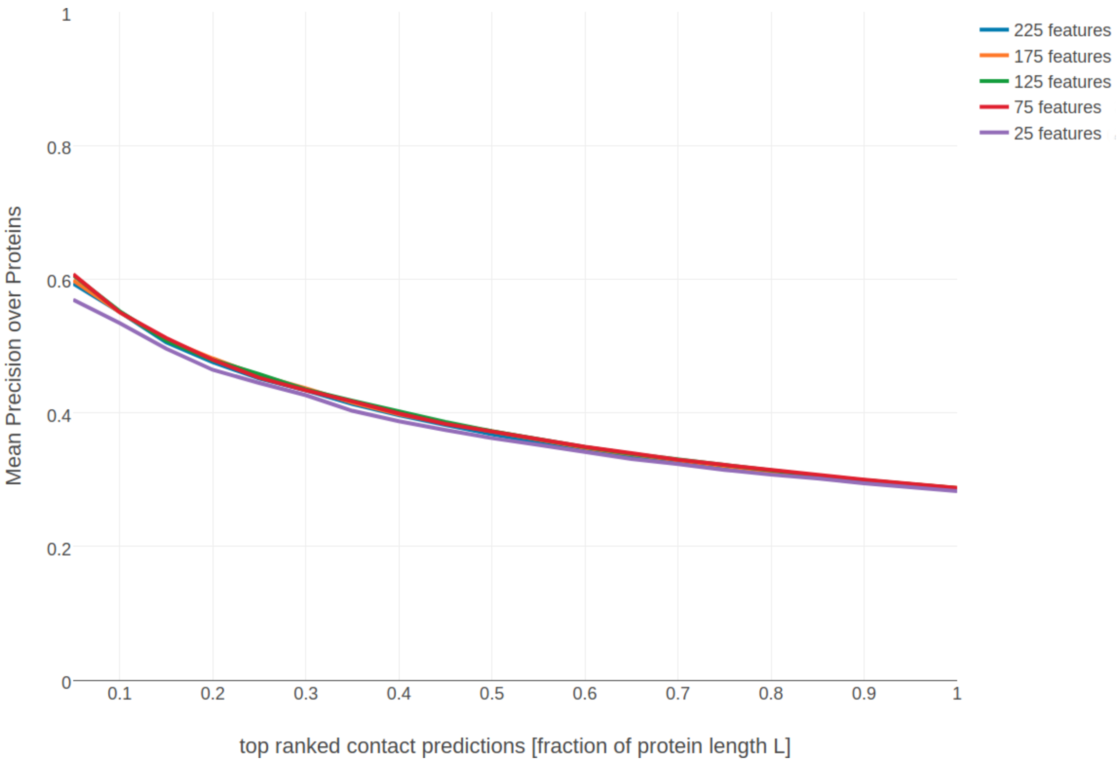

Figure 2.3: Mean precision of top ranked predictions over 200 proteins for random forest models trained on subsets of features of decreasing importance. Subsets of features have been selected as described in methods section 3.13.3.

Performance of the random forest models drops noticeably when using only the 25 most important features. For the further analysis I am using the random forest model trained on the 75 most important features as this model constitutes the smallest set of features while performing nearly identical compared to the model trained on the complete feature set.

Figure 2.4 shows the mean precision for the random forest model trained on the 75 most important features. The random forest model has a mean precision of 0.33 for the top $0.5 \cdot L$ contacts compared to a precision of 0.47 for pseudo-likelihood. Furthermore, the random forest model improves approximately ten percentage points in precision over the local statistical contact scores, *OMES* and mutual information (MI). Both methods comprise important features of the random forest model as can be seen in Figure 2.2.

When analysing performance with respect to alignment size it can be found that the random forest model outperforms the pseudo-likelihood score for small alignments (see Figure 2.5).
Both, local statistial models *OMES* and MI also perform weak on small alignments, leading to the conclusion that the remaining sequence derived features are highly relevant when the alignment contains only few sequences. This finding is expected, as it is well known that models trained on simple sequence features perform almost independent of alignment size [5].

Figure 2.5 showed that the random forest predictor improves over the pseudo-likelihood coevolution method when the alignment consists of only few sequences. In order to assess this improvement in a more direct manner, it is possible to

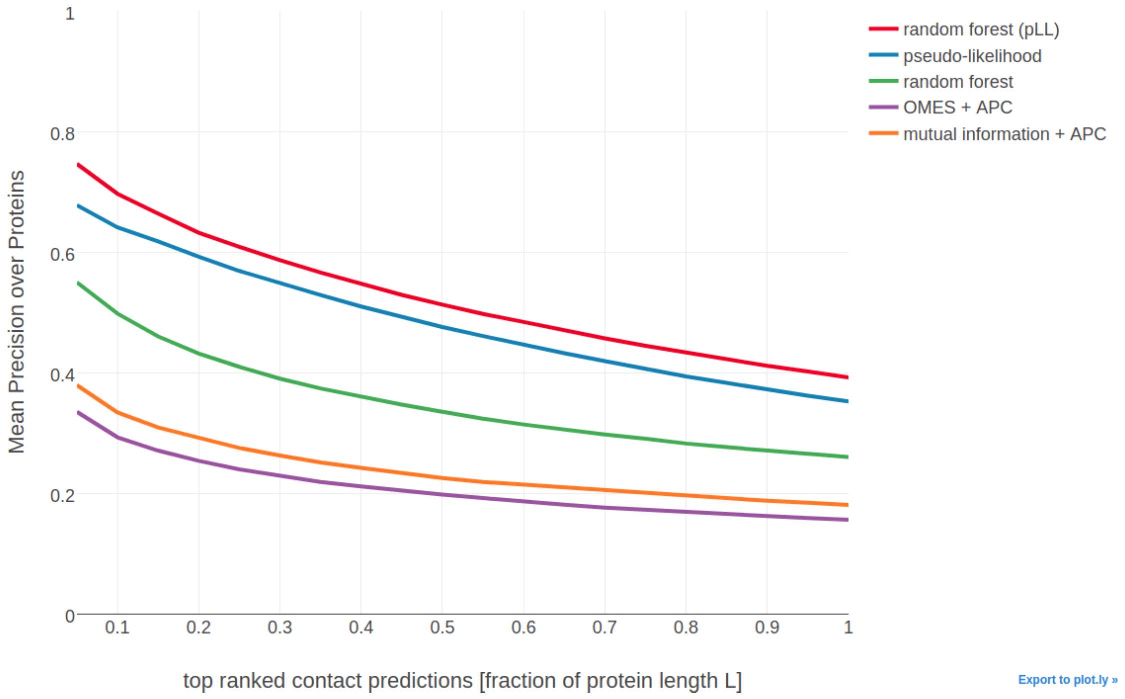

Figure 2.4: Mean precision for top ranked contacts on a test set of 774 proteins. **random forest (pLL)** = random forest model using sequence derived features and pseudo-likelihood contact score (APC corrected Frobenius norm of couplings). **pseudo-likelihood** = APC corrected Frobenius norm of couplings computed with pseudo-likelihood. **random forest** = random forest model trained on 75 sequence derived features. **OMES** = APC corrected *OMES* contact score according to Fodor&Aldrich [14]. **mutual information** = APC corrected mutual information between amino acid counts (using pseudo-counts).

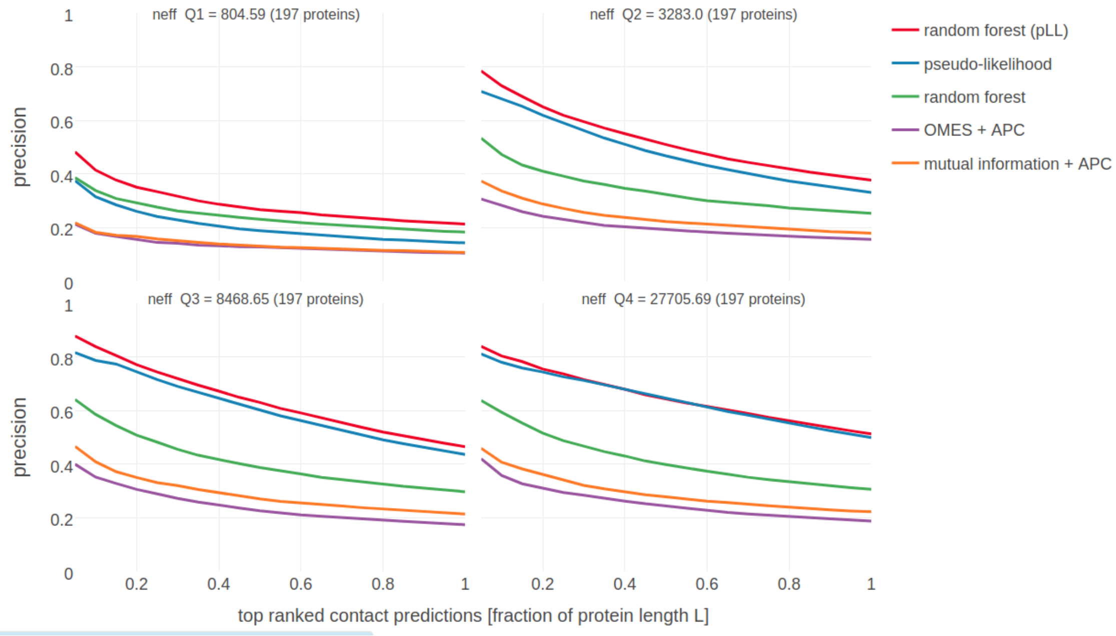

Figure 2.5: Mean precision for top ranked contacts on a test set of 774 proteins splitted into four equally sized subsets with respect to Neff. Subsets are defined according to quantiles of Neff values. Upper left: Subset of proteins with Neff < Q1. Upper right: Subset of proteins with Q1 <= Neff < Q2. Lower left: Subset of proteins with Q2 <= Neff < Q3. Lower right: Subset of proteins with Q3 <= Neff < Q4. **random forest (pLL)** = random forest model using sequence derived features and pseudo-likelihood contact score (APC corrected Frobenius norm of couplings). **pseudo-likelihood** = APC corrected Frobenius norm of couplings computed with pseudo-likelihood. **random forest** = random forest model trained on 75 sequence derived features. **OMES** = APC corrected *OMES* contact score according to Fodor&Aldrich [14]. **mutual information** = APC corrected mutual information between amino acid counts (using pseudo-counts).

build a combined random forest predictor that is not only trained on the sequence derived features but also on the pseudo-likelihood contact score as an additional feature (see methods section 3.13.4 for details). As expected, the pseudo-likelihood score comprises the most important feature in the model (Figure 3.21 in methods section) followed by the same sequence features that were found in the previous analysis in Figure 2.2.

Finally, comparing the random forest model trained on sequence features and pseudo-likelihood contact score to the pseudo-likelihood score in Figure 2.4 reveals that combining both types of information indeed improves predictive power over both single approaches. Especially for small alignments, the improvement is substantial as can be seen in in the left upper plot in Figure 2.5. In contrast, the improvement on large alignments (right lower plot in Figure 2.5) is small, as the gain from simple sequence features compared to the much more powerful coevolution signals is neglectable.

3

Methods

all you need to know

3.1 Dataset

A protein dataset has been constructed from the CATH (v4.1) [19] database for classification of protein domains. All CATH domains from classes 1(mainly α), 2(mainly β), 3($\alpha + \beta$) have been selected and filtered for internal redundancy at the sequence level using the `pdbfilter` script from the HH-suite[20] with an E-value cutoff=0.1. The dataset has been split into ten subsets aiming at the best possible balance between CATH classes 1,2,3 in the subsets. All domains from a given CATH topology (=fold) go into the same subsets, so that any two subsets are non-redundant at the fold level. Some overrepresented folds (e.g. Rossman Fold) have been subsampled ensuring that in every subset each class contains at max 50% domains of the same fold. Consequently, a fold is not allowed to dominate a subset or even a class in a subset. In total there are 6741 domains in the dataset.

Multiple sequence alignments were built from the CATH domain sequences (COMBS) using HHblits [20] with parameters to maximize the detection of homologous sequences:

```
hhblits -maxfilt 100000 -realign_max 100000 -B 100000 -Z 100000 -n 5
-e 0.1 -all hhfilter -id 90 -neff 15 -qsc -30
```

The COMBS sequences are derived from the SEQRES records of the PDB file and sometimes contain extra residues that are not resolved in the structure. Therefore, residues in PDB files have been renumbered to match the COMBS sequences. The process of renumbering residues in PDB files yielded ambigious solutions for 293 proteins, that were removed from the dataset. Another filtering step was applied to remove 80 proteins that do not hold the following properties:

- more than 10 sequences in the multiple sequence alignment ($N > 10$)
- protein length between 30 and 600 residues ($30 \leq L \leq 600$)

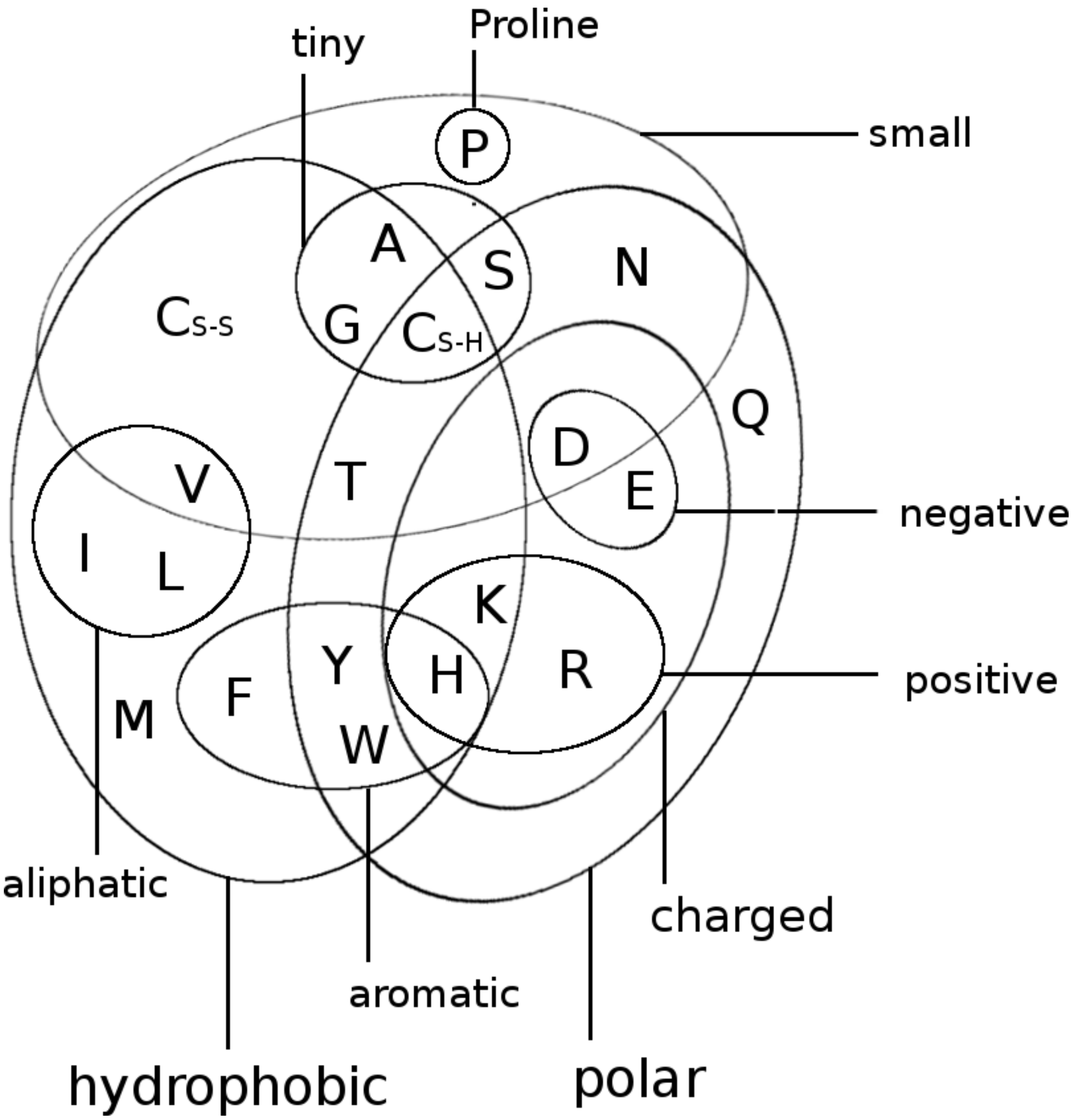

Figure 3.1: Distribution of CATH classes (1=mainly α, 2=mainly β, 3=$\alpha - \beta$) in the dataset and the ten subsets.

- less than 80% gaps in the multiple sequence alignment (percent gaps < 0.8)
- at least one residue-pair in contact at $C_\beta < 8\mathring{A}$ and minimum sequence separation of 6 positions

The final dataset is comprised of **6368** proteins with almost evenly distributed CATH classes over the ten subsets (Figure 3.1).

3.2 Computing Pseudo-Likelihood Couplings

Dr Stefan Seemayer has reimplementated the open-source software CCMpred [21] in Python. CCMpred optimizes the regularized negative pseudo-log-likelihood using a conjugate gradients optimizer. Based on a fork of his private github repository I continued development and extended the software, which is now called

CCMpredPy. It will soon be available at https://github.com/soedinglab/CCMpredPy. All computations in this thesis are performed with CCMpredPy unless stated otherwise.

3.2.1 Differences between CCMpred and CCMpredpy

CCMpredPy differs from CCMpred [21] which is available at https://github.com/soedinglab/CCMpred in several details:

- Initialization of potentials **v** and **w**
  - CCMpred initializes single potentials $\mathbf{v}_i(a) = \log f_i(a) - \log f_i(a = "-")$ with $f_i(a)$ being the frequency of amino acid a at position i and $a = "-"$ representing a gap. A single pseudo-count has been added before computing the frequencies. Pair potentials **w** are intialized at 0.
  - CCMpredPy initializes single potentials **v** with the ML estimate of single potentials (see section **??**) using amino acid frequencies computed as described in section 3.4. Pair potentials **w** are initialized at 0.
- Regularization
  - CCMpred uses a Gaussian regularization prior centered at zero for both single and pair potentials. The regularization coefficient for single potentials $\lambda_v = 0.01$ and for pair potentials $\lambda_w = 0.2 * (L - 1)$ with L being protein length.
  - CCMpredPy uses a Gaussian regularization prior centered at zero for the pair potentials. For the single potentials the Gaussian regularization prior is centered at the ML estimate of single potentials (see section **??**) using amino acid frequencies computed as described in section 3.4. The regularization coefficient for single potentials $\lambda_v = 10$ and for pair potentials $\lambda_w = 0.2 * (L - 1)$ with L being protein length.

Default settings for CCMpredPy have been chosen to best reproduce CCMpred results. A benchmark over a subset of approximately 3000 proteins confirms that performance measured as PPV for both methods is almost identical (see Figure 3.2).

The benchmark in Figure 3.2 as well as all contacts predicted with CCMpred and CCMPredPy (using pseudo-likelihood) in my thesis have been computed using the following flags:

CCMpredPy

```
--maxit 250                       # Compute a maximum of MAXIT operations
--center-v                        # Use a Gaussian prior for single potentials centere
--reg-l2-lambda-single 10         # regularization coefficient for single potentials
--reg-l2-lambda-pair-factor 0.2   # regularization coefficient for pairwise potential
--pc-uniform                      # use uniform pseudocounts (1/21 for 20 amino acids
--pc-count 1                      # defining pseudo count admixture coefficient rho =
```

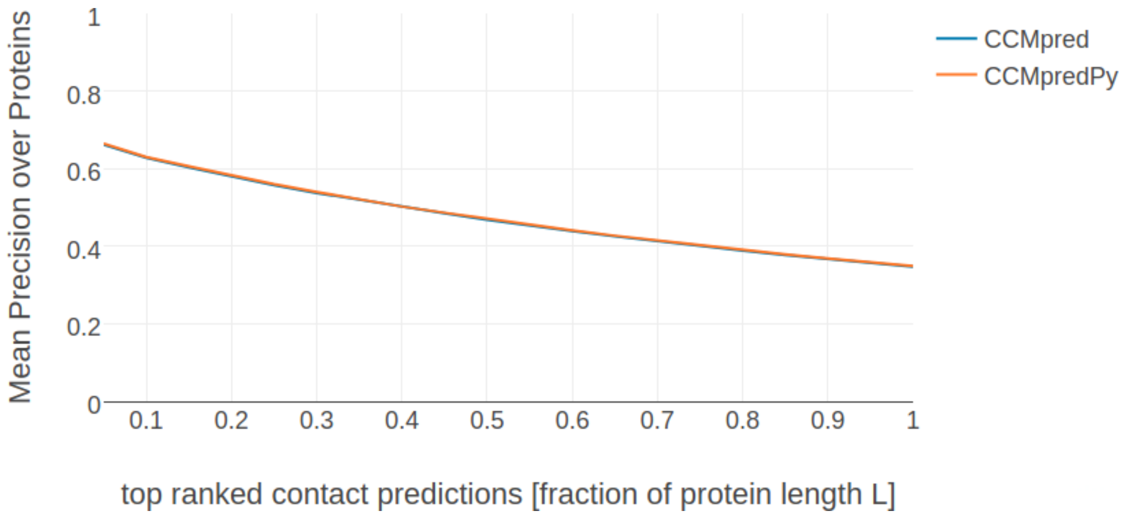

Figure 3.2: Mean precision over 3124 proteins of top ranked contacts computed as APC corrected Frobenius norm of couplings. Couplings have been computed with CCMpred [21] and CCMpredPy as specified in the legend. Specific flags that have been used to run both methods are described in detail in the text (see section 3.2.1).

```
--epsilon 1e-5                    # convergence criterion for minimum decrease i
--ofn-pll                         # using pseudo-likelihood as objective functio
--alg-cg                          # using conjugate gradient to optimize objecti
```

CCMpred

```
-n 250     # NUMITER:  Compute a maximum of NUMITER operations
-l 0.2     # LFACTOR:  Set pairwise regularization coefficients to LFACTOR * (L-1
-w 0.8     # IDTHRES:  Set sequence reweighting identity threshold to IDTHRES
-e 1e-5    # EPSILON:  Set convergence criterion for minimum decrease in the last
```

3.3 Sequence Reweighting

As discussed in section **??**, sequences in a MSA do not represent independent draws from a probabilistic model. To reduce the effects of overrepresented sequences, typically a simple weighting strategy is applied that assigns a weight to each sequence that is the inverse of the number of similar sequences according to an identity threshold [22]. It has been found that reweighting improves contact prediction performance [23–25] significantly but results are robust against the choice of the identity threshold in a range between 0.7 and 0.9 [24]. An identity threshold of 0.8 has been used for all analyses in this thesis.

Every sequence x_n of length L in an alignment with N sequences has an associated weight $w_n = 1/m_n$, where m_n represents the number of similar sequences:

$$w_n = \frac{1}{m_n}, m_n = \sum_{m=1}^{N} I\left(ID(x_n, x_m) \geq 0.8\right) ID(x_n, x_m) = \frac{1}{L} \sum_{i=1}^{L} I(x_n^i = x_m^i) \quad (3.1)$$

The number of effective sequences $\mathbf{N}_{\text{eff}}$ of an alignment is then the number of sequence clusters computed as:

$$N_{\text{eff}} = \sum_{n=1}^{N} w_n \tag{3.2}$$

TODO: Plot Performance for Seq weighting

3.4 Computing Amino Acid Frequencies

Single and pairwise amino acid frequencies are computed from amino acid counts of weighted sequences as described in the last section 3.3 and additional pseudocounts that are added to improve numerical stability.

Let $a, b \in \{1, \ldots, 20\}$ be amino acids and $q_0(x_i = a), q_0(x_i = a, x_j = b)$ be the empirical single and pair frequencies without pseudocounts. The empirical single and pair frequencies with pseudocounts, $q(x_i = a), q(x_i = a, x_j = b)$, are defined

$$\begin{align} q(x_i\!=\!a) :=& (1-\tau)\; q_0(x_i\!=\!a) + \tau \tilde{q}(x_i\!=\!a) \tag{3.3} \\ q(x_i\!=\!a, x_j\!=\!b) :=& (1-\tau)^2 \; [q_0(x_i\!=\!a, x_j\!=\!b) - q_0(x_i\!=\!a) q_0(x_j\!=\!b)] + \tag{3.4} \\ & q(x_i\!=\!a)\; q(x_j\!=\!b) \tag{3.5} \end{align}$$

with $\tilde{q}(x_i\!=\!a) := f(a)$ being background amino acid frequencies and $\tau \in [0, 1]$ is a pseudocount admixture coefficient, which is a function of the diversity of the multiple sequence alignment:

$$\tau = \frac{N_{\text{pc}}}{(N_{\text{eff}} + N_{\text{pc}})} \tag{3.6}$$

where $N_{pc} > 0$.

The formula for $q(x_i\!=\!a, x_j\!=\!b)$ in the second line in eq (3.5) was chosen such that for $\tau\!=\!0$ we obtain $q(x_i\!=\!a, x_j\!=\!b) = q_0(x_i\!=\!a, x_j\!=\!b)$, and furthermore $q(x_i = a, x_j\!=\!b) = q(x_i\!=\!a) q(x_j\!=\!b)$ exactly if $q_0(x_i\!=\!a, x_j\!=\!b) = q_0(x_i\!=\!a) q_0(x_j\!=\!b)$.

3.5 Regularization

CCMpredPy offers an L2-regularization that pushes the single and pairwise terms smoothly towards zero and is equivalent to the logarithm of a zero-centered Gaussian prior,

$$\begin{align} R(\mathbf{v}, \mathbf{w}) &= \log \left[\mathcal{N}(\mathbf{v} | \mathbf{v}^*, \lambda_v^{-1} I) \mathcal{N}(\mathbf{w} | \mathbf{w}^*, \lambda_w^{-1} I) \right] \\ &= -\frac{\lambda_v}{2} ||\mathbf{v} - \mathbf{v}^*||_2^2 - \frac{\lambda_w}{2} ||\mathbf{w} - w^*||_2^2 + \text{const.}\,, \tag{3.7} \end{align}$$

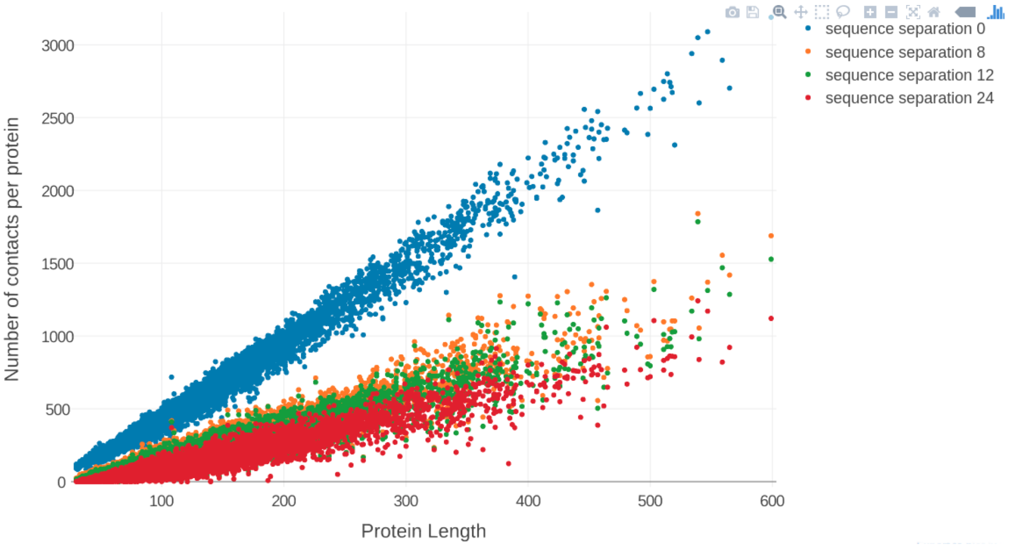

Figure 3.3: Number of contacts ($C_\beta < 8\mathring{A}$) with respect to protein length and sequence separation has a linear relationship.

where the regularization coefficients λ_v and λ_w determine the strength of regularization. Most previous pseudo-likelihood approaches using L2-regularization for pseudo-likelihood optimization set $\mathbf{v}^* = \mathbf{w}^* = \mathbf{0}$ [21,26,27]. A different choice for v^* is discussed in section 3.6.3 that is is used with *CCMpredPy* for pseudo-likelihood optimization. The single potentials will not be optimized with CD but will be fixed at the v^* given in eq. (3.43). Furthermore, *CCMpredPy* uses regularization coefficients $\lambda_v = 10$ and $\lambda_w = 0.2 \cdot (L-1)$ for pseudo-likelihood optimization and the choice for λ_w used with CD is discussed in section 3.9.

The regularization coefficient λ_w for couplings $\mathbf{w}$ is defined with respect to protein length L owing to the fact that the number of possible contacts in a protein increases quadratically with L whereas the number of observed contacts only increases linearly as can be seen in Figure 3.3.

3.6 The Potts Model

The N sequences of the MSA $\mathbf{X}$ of a protein family are denoted as $\mathbf{x}_1, ..., \mathbf{x}_N$. Each sequence $\mathbf{x}_n = (\mathbf{x}_{n1}, ..., \mathbf{x}_{nL})$ is a string of L letters from an alphabet indexed by $\{0, ..., 20\}$, where 0 stands for a gap and $\{1, ..., 20\}$ stand for the 20 types of amino acids. The likelihood of the sequences in the MSA of the protein family is modelled with a *Potts Model*, as described in detail in section **??**:

$$\begin{aligned} p(\mathbf{X}|\mathbf{v},\mathbf{w}) &= \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{v},\mathbf{w}) \\ &= \prod_{n=1}^{N} \frac{1}{Z(\mathbf{v},\mathbf{w})} \exp\left(\sum_{i=1}^{L} v_i(x_{ni}) \sum_{1 \leq i < j \leq L} w_{ij}(x_{ni}, x_{nj}) \right) \end{aligned} \tag{3.8}$$

The coefficients v_{ia} and w_{ijab} are referred to as single potentials and couplings, respectively that describe the tendency of an amino acid a (and b) to (co-)occur at the respective positions in the MSA. $Z(\mathbf{v}, \mathbf{w})$ is the partition function that normalizes the probability distribution $p(\mathbf{x}_n|\mathbf{v}, \mathbf{w})$:

$$Z(\mathbf{v}, \mathbf{w}) = \sum_{y_1,\ldots,y_L=1}^{20} \exp\left(\sum_{i=1}^{L} v_i(y_i) \sum_{1\leq i<j\leq L} w_{ij}(y_i, y_j)\right) \tag{3.9}$$

The log likelihood is

$$\begin{aligned} LL(\mathbf{v}, \mathbf{w}) &= \log p(\mathbf{X}|\mathbf{v}, \mathbf{w}) && (3.10)\\ &= \sum_{n=1}^{N}\left[\sum_{i=1}^{L} v_i(x_{ni}) \sum_{1\leq i<j\leq L} w_{ij}(x_{ni}, x_{nj})\right] - N \log Z(\mathbf{v}, \mathbf{w}). && (3.11)\end{aligned}$$

The gradient of the log likelihood has single components

$$\begin{aligned} \frac{\partial LL(\mathbf{v}, \mathbf{w})}{\partial v_{ia}} &= \sum_{n=1}^{N} I(x_{ni} = a) - N \frac{\partial}{\partial v_{ia}} \log Z(\mathbf{v}, \mathbf{w}) && (3.12)\\ &= \sum_{n=1}^{N} I(x_{ni}{=}a) - N \sum_{y_1,\ldots,y_L=1}^{20} \frac{\exp\left(\sum_{i=1}^{L} v_i(y_i) + \sum_{1\leq i<j\leq L} w_{ij}(y_i, y_j)\right)}{Z(\mathbf{v}, \mathbf{w})} I(y_i{=}a) && (3.13)\\ &= Nq(x_i{=}a) - Np(x_i{=}a|\mathbf{v}, \mathbf{w}) && (3.14)\end{aligned}$$

and pair components

$$\begin{aligned} \frac{\partial LL(\mathbf{v}, \mathbf{w})}{\partial w_{ijab}} &= \sum_{n=1}^{N} I(x_{ni} = a, x_{nj} = b) - N \frac{\partial}{\partial w_{ijab}} \log Z(\mathbf{v}, \mathbf{w}) && (3.15)\\ &= \sum_{n=1}^{N} I(x_{ni}{=}a, x_{nj}{=}b) && (3.16)\\ &\quad - N \sum_{y_1,\ldots,y_L=1}^{20} \frac{\exp\left(\sum_{i=1}^{L} v_i(y_i) + \sum_{1\leq i<j\leq L} w_{ij}(y_i, y_j)\right)}{Z(\mathbf{v}, \mathbf{w})} I(y_i{=}a, y_j{=}b) && (3.17)\\ &= Nq(x_i{=}a, x_j{=}b) - N \sum_{y_1,\ldots,y_L=1}^{20} p(y_1, \ldots, y_L|\mathbf{v}, \mathbf{w})\, I(y_i{=}a, y_j{=}b) && (3.18)\\ &= Nq(x_i{=}a, x_j{=}b) - Np(x_i{=}a, x_j{=}b|\mathbf{v}, \mathbf{w}) && (3.19)\end{aligned}$$

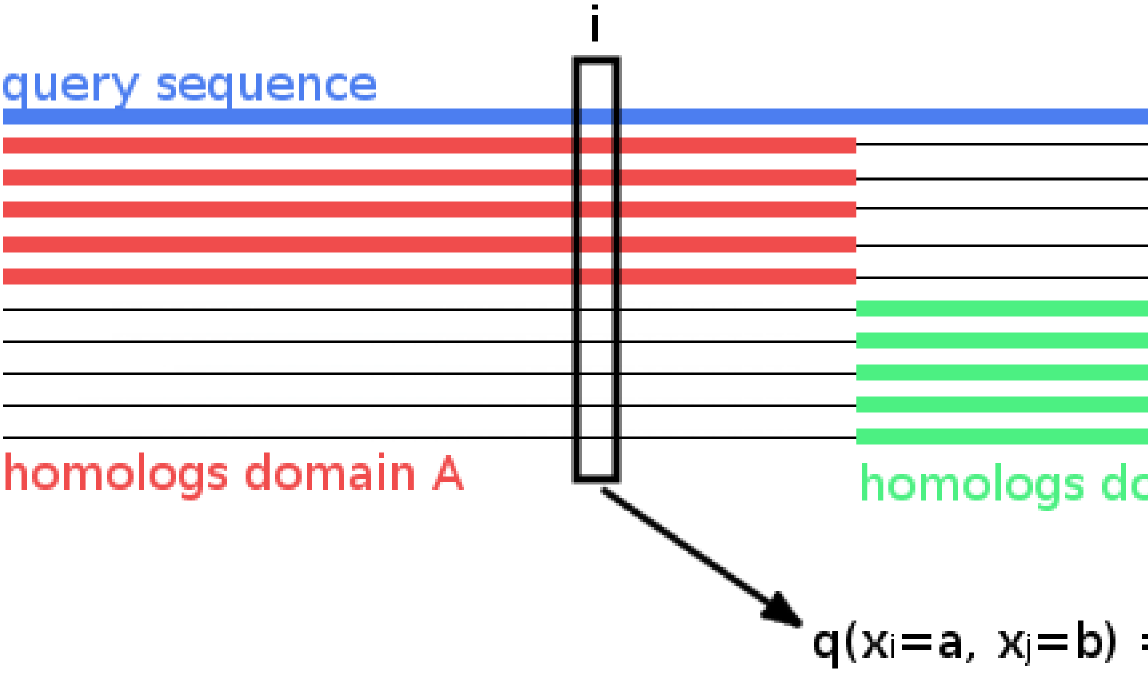

Figure 3.4: Hypothetical MSA consisting of two sets of sequences: the first set has sequences covering only the left half of columns, while the second set has sequences covering only the right half of columns. The two blocks could correspond to protein domains that were aligned to a single query sequence. Empirical amino acid pair frequencies $q(x_i = a, x_j = b)$ will vanish for positions i from the left half and j from the right half of the alignment.

3.6.1 Treating Gaps as Missing Information

Treating gaps explicitly as 0'th letter of the alphabet will lead to couplings between columns that are not in physical contact. To see why, imagine a hypothetical alignment consisting of two sets of sequences as it is illustrated in Figure 3.4. The first set has sequences covering only the left half of columns in the MSA, while the second set has sequences covering only the right half of columns. The two blocks could correspond to protein domains that were aligned to a single query sequence. Now consider couplings between a pair of columns i, j with i from the left half and j from the right half. Since no sequence (except the single query sequence) overlaps both domains, the empirical amino acid pair frequencies $q(x_i = a, x_j = b)$ will vanish for all $a, b \in \{1, ..., L\}$.

According to the gradient of the log likelihood for couplings w_{ijab} given in eq (3.19), the empirical frequencies $q(x_i = a, x_j = b)$ are equal to the model probabilities $p(x_i = a, x_j = b | \mathbf{v}, \mathbf{w})$ at the maximum of the likelihood when the gradient vanishes. Therefore, $p(x_i = a, x_j = b | \mathbf{v}, \mathbf{w})$ would have to be zero at the optimum when the empirical amino acid frequencies $q(x_i = a, x_j = b)$ vanish for pairs of columns as described above. However, $p(x_i = a, x_j = b | \mathbf{v}, \mathbf{w})$ can only become zero, when the exponential term is zero, which would only be possible if w_{ijab} goes to ∞. This is clearly undesirable, as physical contacts will be deduced from the size of the couplings.

The solution is to treat gaps as missing information. This means that the normalisation of $p(\mathbf{x}_n|\mathbf{v},\mathbf{w})$ should not run over all positions $i \in \{1,...,L\}$ but only over those i that are not gaps in $\mathbf{x}_n$. Therefore, the set of sequences S_n used for normalization of $p(\mathbf{x}_n|\mathbf{v},\mathbf{w})$ in the partition function will be defined as:

$$S_n := \{(y_1,...,y_L) : 0 \leq y_i \leq 20 \wedge (y_i\!=\!0 \text{ iff } x_{ni}\!=\!0)\} \tag{3.20}$$

and the partition function becomes:

$$Z_n(\mathbf{v},\mathbf{w}) = \sum_{\mathbf{y}\in S_n} \exp\left(\sum_{i=1}^{L} v_i(y_i) \sum_{1\leq i<j\leq L} w_{ij}(y_i,y_j)\right) \tag{3.21}$$

To ensure that the gaps in $y \in S_n$ do not contribute anything to the sums, the parameters associated with a gap will be fixed to 0

$$v_i(0) = \mathbf{w}_{ij}(0,b) = \mathbf{w}_{ij}(a,0) = 0 \ ,$$

for all $i,j \in \{1,...,L\}$ and $a,b \in \{0,...,20\}$.

Furthermore, the empirical amino acid frequencies q_{ia} and q_{ijab} need to be redefined such that they are normalised over $\{1,...,20\}$,

$$N_i := \sum_{n=1}^{N} w_n I(x_{ni}\!\neq\!0) \qquad q_{ia} = q(x_i\!=\!a) := \frac{1}{N_i}\sum_{n=1}^{N} w_n I(x_{ni}\!=\!a) \tag{3.22}$$

$$N_{ij} := \sum_{n=1}^{N} w_n I(x_{ni}\!\neq\!0, x_{nj}\!\neq\!0) \quad q_{ijab} = q(x_i\!=\!a, x_j\!=\!b) := \frac{1}{N_{ij}}\sum_{n=1}^{N} w_n I(x_{ni}\!=\!a, x_{nj}\!=\!b) \tag{3.23}$$

with w_n being sequence weights calculated as described in methods section 3.3. With this definition, empirical amino acid frequencies are normalized without gaps, so that

$$\sum_{a=1}^{20} q_{ia} = 1 \ , \ \sum_{a,b=1}^{20} q_{ijab} = 1. \tag{3.24}$$

3.6.2 The Regularized Full Log Likelihood and its Gradient With Gap Treatment

In pseudo-likelihood based methods, a regularisation is commonly used that can be interpreted to arise from a prior probability. The same treatment will be applied to the full likelihood. Gaussian priors $\mathcal{N}(\mathbf{v}|\mathbf{v}^*, \lambda_v^{-1}\mathbf{I})$ and $\mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda_w^{-1}\mathbf{I})$ will be used to constrain the parameters $\mathbf{v}$ and $\mathbf{w}$ and to fix the gauge. The choice of v^* is discussed in section 3.6.3. By including the logarithm of this prior into the log likelihood the regularised log likelihood is obtained,

$$LL_{\text{reg}}(\mathbf{v}, \mathbf{w}) = \log\left[p(\mathbf{X}|\mathbf{v}, \mathbf{w})\, \mathcal{N}(\mathbf{v}|\mathbf{v}^*, \lambda_v^{-1}\mathbf{I})\, \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda_w^{-1}\mathbf{I})\right] \tag{3.25}$$

or explicitely,

$$LL_{\text{reg}}(\mathbf{v}, \mathbf{w}) = \sum_{n=1}^{N}\left[\sum_{i=1}^{L} v_i(x_{ni}) + \sum_{1\leq i<j\leq L} w_{ij}(x_{ni}, x_{nj}) - \log Z_n(\mathbf{v}, \mathbf{w})\right] \tag{3.26}$$

$$-\frac{\lambda_v}{2}\sum_{i=1}^{L}\sum_{a=1}^{20}(v_{ia} - v_{ia}^*)^2 - \frac{\lambda_w}{2}\sum_{1\leq i<j\leq L}\sum_{a,b=1}^{20} w_{ijab}^2. \tag{3.27}$$

The gradient of the regularized log likelihood has single components

$$\frac{\partial LL_{\text{reg}}}{\partial v_{ia}} = \sum_{n=1}^{N} I(x_{ni} = a) - \sum_{n=1}^{N} \frac{\partial}{\partial v_{ia}} \log Z_n(\mathbf{v}, \mathbf{w}) - \lambda_v(v_{ia} - v_{ia}^*) \tag{3.28}$$

$$= N_i q(x_i\!=\!a) \tag{3.29}$$

$$-\sum_{n=1}^{N}\sum_{\mathbf{y}\in S_n} \frac{\exp\left(\sum_{i=1}^{L} v_i(y_i) + \sum_{1\leq i<j\leq L}^{L} w_{ij}(y_i, y_j)\right)}{Z_n(\mathbf{v}, \mathbf{w})} I(y_i = a) \tag{3.30}$$

$$-\lambda_v(v_{ia} - v_{ia}^*) \tag{3.31}$$

and pair components

$$\frac{\partial LL_{\text{reg}}}{\partial w_{ijab}} = \sum_{n=1}^{N} I(x_{ni}\!=\!a, x_{nj}\!=\!b) - \sum_{n=1}^{N} \frac{\partial}{\partial w_{ijab}} \log Z_n(\mathbf{v}, \mathbf{w}) - \lambda_w w_{ijab} \tag{3.32}$$

$$= N_{ij} q(x_i\!=\!a, x_j = b) \tag{3.33}$$

$$-\sum_{n=1}^{N}\sum_{\mathbf{y}\in S_n} \frac{\exp\left(\sum_{i=1}^{L} v_i(y_i) + \sum_{1\leq i<j\leq L}^{L} w_{ij}(y_i, y_j)\right)}{Z_n(\mathbf{v}, \mathbf{w})} I(y_i\!=\!a, y_j\!=\!b) \tag{3.34}$$

$$-\lambda_w w_{ijab} \tag{3.35}$$

Note that (without regulariation $\lambda_v = \lambda_w = 0$) the empirical frequencies $q(x_i\!=\!a)$ and $q(x_i\!=\!a, x_j = b)$ are equal to the model probabilities at the maximum of the likelihood when the gradient becomes zero.

If the proportion of gap positions in $\mathbf{X}$ is small (e.g. $< 5\%$, also compare percentage of gaps in dataset in Appendix Figure B.2), the sums over $\mathbf{y} \in S_n$ in eqs. (3.31) and (3.35) can be approximated by $p(x_i = a|\mathbf{v}, \mathbf{w}) I(x_{ni} \neq 0)$ and $p(x_i = a, x_j = b|\mathbf{v}, \mathbf{w}) I(x_{ni} \neq 0, x_{nj} \neq 0)$, respectively, and the partial derivatives become

$$\frac{\partial LL_{\text{reg}}}{\partial v_{ia}} = N_i q(x_i{=}a) - N_i \, p(x_i{=}a|\mathbf{v}, \mathbf{w}) - \lambda_v (v_{ia} - v^*_{ia}) \tag{3.36}$$

$$\frac{\partial LL_{\text{reg}}}{\partial w_{ijab}} = N_{ij} q(x_i{=}a, x_j = b) - N_{ij} \, p(x_i{=}a, x_j{=}b|\mathbf{v}, \mathbf{w}) - \lambda_w w_{ijab} \tag{3.37}$$

Note that the couplings between columns i and j in the hypothetical MSA presented in the last section 3.6.1 will now vanish since $N_{ij}{=}0$ and the gradient with respect to w_{ijab} is equal to $-\lambda_w w_{ijab}$.

3.6.3 The prior on v

Most previous approaches chose a prior around the origin, $p(\mathbf{v}) = \mathcal{N}(\mathbf{v}|\mathbf{0}, \lambda_v^{-1}\mathbf{I})$, i.e., $v^*_{ia} = 0$. It can be shown that the choice $v^*_{ia} = 0$ leads to undesirable results. Taking the sum over $b = 1, \ldots, 20$ at the optimum of the gradient of couplings in eq. (3.37), yields

$$0 = N_{ij} \, q(x_i{=}a, x_j \neq 0) - N_{ij} \, p(x_i{=}a|\mathbf{v}, \mathbf{w}) - \lambda_w \sum_{b=1}^{20} w_{ijab} \;, \tag{3.38}$$

for all $i, j \in \{1, \ldots, L\}$ and all $a \in \{1, \ldots, 20\}$.

Note, that by taking the sum over $a = 1, \ldots, 20$ it follows that,

$$\sum_{a,b=1}^{20} w_{ijab} = 0. \tag{3.39}$$

At the optimum the gradient with respect to v_{ia} vanishes and according to eq. (3.36), $p(x_i = a|\mathbf{v}, \mathbf{w}) = q(x_i = a) - \lambda_v (v_{ia} - v^*_{ia})/N_i$. This term can be substituted into equation (3.38), yielding

$$0 = N_{ij} \, q(x_i{=}a, x_j \neq 0) - N_{ij} \, q(x_i = a) + \frac{N_{ij}}{N_i} \lambda_v (v_{ia} - v^*_{ia}) - \lambda_w \sum_{b=1}^{20} w_{ijab} \;. \tag{3.40}$$

Considering a MSA without gaps, the terms $N_{ij} \, q(x_i{=}a, x_j \neq 0) - N_{ij} \, q(x_i = a)$ cancel out, leaving

$$0 = \lambda_v (v_{ia} - v^*_{ia}) - \lambda_w \sum_{b=1}^{20} w_{ijab}. \tag{3.41}$$

Now, consider a column i that is not coupled to any other and assume that amino acid a was frequent in column i and therefore v_{ia} would be large and positive.

Then according to eq. (3.41), for any other column j the 20 coefficients w_{ijab} for $b \in \{1, \dots, 20\}$ would have to take up the bill and deviate from zero! This unwanted behaviour can be corrected by instead choosing a Gaussian prior centered around $\mathbf{v}^*$ obeying

$$\frac{\exp(v^*_{ia})}{\sum_{a'=1}^{20} \exp(v^*_{ia'})} = q(x_i = a). \tag{3.42}$$

This choice ensures that if no columns are coupled, i.e. $p(\mathbf{x}|\mathbf{v}, \mathbf{w}) = \prod_{i=1}^{L} p(x_i)$, $\mathbf{v} = \mathbf{v}^*$ and $\mathbf{w} = \mathbf{0}$ gives the correct probability model for the sequences in the MSA. Furthermore imposing the restraint $\sum_{a=1}^{20} v_{ia} = 0$ to fix the gauge of the v_{ia} (i.e. to remove the indeterminacy), yields

$$v^*_{ia} = \log q(x_i = a) - \frac{1}{20} \sum_{a'=1}^{20} \log q(x_i = a'). \tag{3.43}$$

For this choice, $v_{ia} - v^*_{ia}$ will be approximately zero and will certainly be much smaller than v_{ia}, hence the sum over coupling coefficients in eq. (3.41) will be close to zero, as it should be.

3.7 Analysis of Coupling Matrices

3.7.1 Correlation of Couplings with Contact Class

Approximately 100000 residue pairs have been filtered for contacts and non-contacts respectively according to the following criteria:

- consider only residue pairs separated by at least 10 positions in sequence
- minimal diversity ($= \frac{\sqrt{N}}{L}$) of alignment = 0.3
- minimal number of non-gapped sequences = 1000
- C_β distance threshold for contact: $< 8\mathring{A}$
- C_β distance threshold for noncontact: $> 25\mathring{A}$

3.7.2 Coupling Distribution Plots

For one-dimensional coupling distribution plots the residue pairs and respective pseudo-log-likelihood coupling values w_{ijab} have been selected as follows:

- consider only residue pairs separated by at least 10 positions in sequence
- discard residues that have more than 30% gaps in the alignment
- discard residue pairs that have insufficient evidence in the alignment: $N_{ij} \cdot q_i(a) \cdot q_j(b) < 100$ with:
  - N_{ij} is the number of sequences with neither a gap at position i nor at position j

- $q_i(a)$ and $q_j(b)$ are the frequencies of amino acids a and b at positions i and j (computed as described in section 3.4)

These criteria ensure that uninformative couplings are neglected, e.g. sequence neighbors albeit being contacts according to the C_β contact definition cannot be assumed to express biological meaningful coupling patterns, or couplings for amino acid pairings that do not have statistical relevant counts in the alignment.

The same criteria have been applied for selecting couplings for the two-dimensional distribution plots with the difference that evidence for a single coupling term has to be $N_{ij} \cdot q_i(a) \cdot q_j(b) < 80$.

3.8 Optimizing Contrastive Divergence with Stochastic Gradient Descent

This section describes hyperparameter tuning for the stochastic gradient descent optimization of CD.

The couplings w_{ijab} are initialized at 0 and single potentials v_i will not be optimized but rather kept fixed at their maximum-likleihood estimate v_i^* as described in methods section 3.6.3. The gradient of the full likelihood is approximated with CD which involves Gibbs sampling of protein sequences according to the current model parametrization and is described in detail in methods section 3.10. Zero centered L2-regularization is used to constrain the coupling parameters $\mathbf{w}$ using the regularization coefficient $\lambda_w = 0.2L$ which is the default setting for optimizing the pseudo-likelihood with *CCMpredPy*. Performance will be evaluated by the mean precision of top ranked contact predictions over a benchmark set of 300 proteins, that is a subset of the data set described in methods section 3.1. Contact scores for couplings are computed as the APC corrected Frobenius norm as explained in section **??**. Pseudo-likelihood couplings are computed with the tool *CCMpredPy* that is introduced in methods section 3.2.1 and the pseudo-likelihood contact score will serve as general reference method for tuning the hyperparameters.

3.8.1 Convergence Criterion for Stochastic Gradient Descent

In theory the gradient descent algorithm has converged and the optimum of the objective function has been reached when the gradient becomes zero. In practice the gradients will never be exactly zero, especially due to the stochasticity of the gradient estimates when using stochastic gradient descent with CD. For this reason, it is crucial to define a suitable convergence criterion that can be tested during optimization and once the criterion is met, convergence is assumed and the algorithm is stopped. Typically, the objective function (or a related loss function) is periodically evaluated on a validation set and the optimizer is halted whenever the function value saturates or starts to increase. This technique is called early stopping and additionally prevents overfitting [28,29]. Unfortunately, we cannot compute the full likelihood function due to its complexity and need to define a different convergence criterion.

One possibility is to stop learning when the norm of the gradients is close to zero [30]. As described earlier, only a subset of alignment sequences is used to estimate the gradient with CD which increases stochasticity of the gradient estimate and speeds up the algorithm. As a result of stochasticity, the norm of gradients does not converge to zero but saturates at a certain offset as it is described in section 3.10. Only by using large number of sequences to estimate the gradients with CD, the norm over gradients will converge towards a value close to zero. However, runtine increases linearly in the number of sequences used for estimating the gradient with CD by Gibbs sampling. Convergence can also be monitored as the relative change of the norm of gradients between a particular number of iterations and optimization can be stopped when the norm of gradients has reached a certain plateau. As gradient estimates are generally very noisy with stochastic gradient descent, gradient fluctuations complicate the proper assessment of this criterion.

Instead of the gradients, it is also possible to observe the relative change over the norm of parameter estimates $||\mathbf{w}||_2$ and stop learning when it falls below a small threshold ϵ,

$$\frac{||\mathbf{w}_{t-1}||_2 - ||\mathbf{w}_t||_2}{||\mathbf{w}_{t-1}||_2} < \epsilon \ . \tag{3.44}$$

This measure is less noisy than subsequent gradient estimates because the magnitude of parameter updates is bounded by the learning rate.

Another idea is to monitor the direction of gradients, since the assumption goes that gradients will start oscillating when approaching the optimum. However, this theoretical assumption is complicated by the fact that gradient oscillations are also typically observed when the parameter surface contains narrow valleys or when the learning rate is too big, as it is visualized in the right plot in Figure **??**. Another interfering factor is momentum, as it is used in the *ADAM* optimizer. Parameters will be updated into the direction of a smoothed historical gradient and oscillations, regardless of which origin, will be dampened. It is therefore hard to define a general convergence criteria that uses gradient directions and can distinguish these different scenarios.

Of course, the simplest strategy to assume convergence is to specify a maximum number of iterations for the optimization procedure, which also ensures that the algorithm will stop eventually if none of the other convergence criteria is met.

A necessary but not sufficient criterion for convergence for the full likelihood is given by $\sum_{a,b=1}^{20} w_{ijab} = 0$. This requirement is derived in section 3.6.3. When using plain stochastic gradient descent without momentum and without adaptive learning rates, this criterion is never violated when parameters are initialized uniformly. This is due to the fact that the 400 gradients w_{ijab} for $a, b \in \{1, \ldots, 20\}$ are not independent because the sum over the 400 pairwise amino acid counts at positions i and j is identical for the observed and the sampled alignment and amounts to,

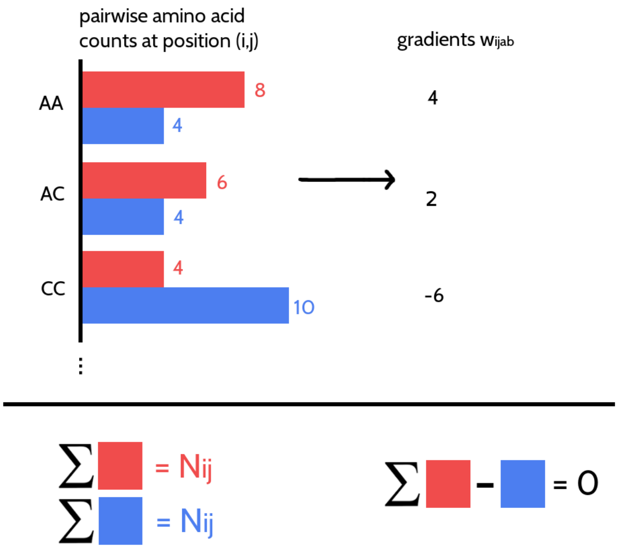

Figure 3.5: The 400 gradients $\nabla_{w_{ijab}} LL_{\text{reg}}(\mathbf{v}, \mathbf{w})$ at position (i, j) for $a, b \in \{1, \dots, 20\}$ are not independent. Red bars represent pairwise amino acid counts at position (i, j) for the sampled alignment. Blue bars represent pairwise amino acid counts at position (i, j) for the input alignment. The sum over pairwise amino acid counts at position (i, j) for both alignments is N_{ij}, which is the number of ungapped sequences. The gradient $\nabla_{w_{ijab}} LL_{\text{reg}}(\mathbf{v}, \mathbf{w})$ is computed as the difference of pairwise amino acid counts for amino acids a and b at position (i, j). The sum over gradients $\nabla_{w_{ijab}} LL_{\text{reg}}(\mathbf{v}, \mathbf{w})$ at position (i, j) for all $a, b \in \{1, \dots, 20\}$ is zero.

$$\sum_{a,b=1}^{20} N_{ij} q(x_i = a, q_j = b) = N_{ij} \; . \tag{3.45}$$

Considering a residue pair (i,j) and assuming amino acid pair (a,b) has higher counts in the sampled alignment than in the observed input alignment, then this difference in counts must be compensated by other amino acid pairs (c,d) having less counts in the sampled alignment compared to the true alignment (see Figure 3.5). Therefore it holds, $\sum_{a,b=1}^{20} \nabla_{w_{ijab}} LL_{\text{reg}}(\mathbf{v}, \mathbf{w}) = 0$. This symmetry is translated into parameter updates as long as the same learning rate is used to update all parameters. However, when using adaptive learning rates, this symmetry is broken and the condition $\sum_{a,b=1}^{20} w_{ijab} = 0$ can be violated during the optimization processs. It is therefore interesting to monitor $\sum_{1 \leq 1 < j \leq L} \sum_{a,b=1}^{20} w_{ijab}$.

I set the maximum number of iterations to 5000 and stop the optimization algorithm when the relative change over the L2-norm of parameter estimates $||\mathbf{w}||_2$ falls below the threshold of $\epsilon = 1e-8$.

3.8.2 Tuning Hyperparameters of Stochastic Gradient Descent Optimizer

The coupling parameters $\mathbf{w}$ will be updated at each time step t by taking a step of size α along the direction of the negative gradient of the regularized full log

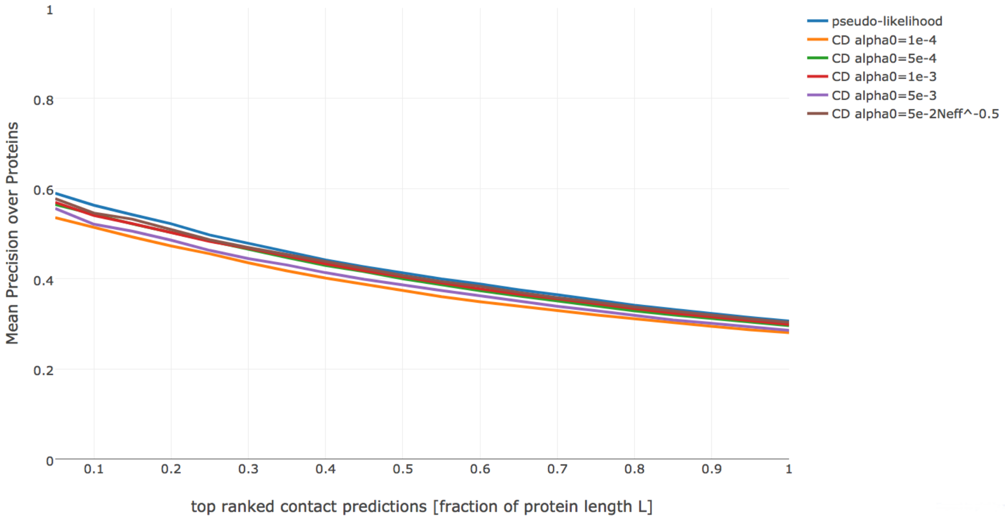

Figure 3.6: Mean precision for top ranked contact predictions over 286 proteins. Contact scores are computed as the APC corrected Frobenius norm of the couplings $\mathbf{w}_{ij}$. **pseudo-likelihood**: couplings computed with pseudo-likelihood. **CD**: couplings computed with CD using stochastic gradient descent with different initial learning rates α_0 as specified in the legend.

likelihood, $-\nabla_w LL_{\text{reg}}(\mathbf{v}, \mathbf{w})$, that has been approximated with CD,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \nabla_w LL_{\text{reg}}(\mathbf{v}, \mathbf{w}) \ . \tag{3.46}$$

In order to get a first intuition of the optimization problem, I tested initial learning rates $\alpha_0 \in \{1\text{e}{-4}, 5\text{e}{-4}, 1\text{e}{-3}, 5\text{e}{-3}\}$ with a standard learning rate annealing schedule, $\alpha = \frac{\alpha_0}{1+\gamma \cdot t}$ where t is the time step and γ is the decay rate that is set to 0.01[31].

Figure 3.6 shows the mean precision for top ranked contacts computed from pseudo-likelihood couplings and from CD couplings optimized with stochastic gradient descent using the four different learning rates. Overall, mean precision for CD contacts is lower than for pseudo-likelihood contacts, especially when using the smallest ($\alpha_0 = 1\text{e}{-4}$) and biggest ($\alpha_0 = 5\text{e}{-3}$) learning rate.

Looking at individual proteins it turns out that the optimal learning rate depends on alignment size. The left plot in Figure 3.7 shows a convergence plot of SGD optimization using different learning rates for a protein with a small alignment. With a small initial learning rate $\alpha_0 = 1\text{e} - 4$ the optimization runs very slowly and does not reach convergence within 5000 iterations. Using a large initial learning rate $\alpha_0 = 5\text{e} - 3$ will result in slighly overshooting the optimum at the beginning of the optimization but with the learning rate decaying over time the parameter estimates converge. In contrast, for a protein with a big alignment (right plot in Figure 3.7) the choice of learning rate has a more pronounced effect. With a small initial learning rate $\alpha_0 = 1\text{e} - 4$ the optimization runs slowly but almost converges within 5000 iterations. A large initial learning rate $\alpha_0 = 5\text{e} - 3$ lets the parameters diverge quickly and the optimum cannot be revovered. With learning rates $\alpha_0 = 5\text{e} - 4$ and $\alpha_0 = 1\text{e} - 3$, the optimum is well overshot at the beginning of

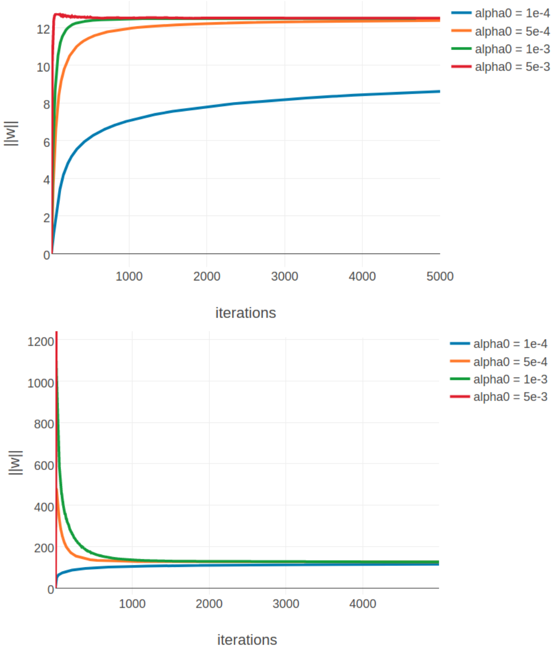

Figure 3.7: Convergence plots for two proteins during SGD optimization with different learning rates and convergence measured as L2-norm of the coupling parameters $||\mathbf{w}||_2$. Linear learning rate annealing schedule has been used with decay rate $\gamma = 0.01$ and initial learning rates α_0 have been set as specified in the legend. **Left** Convergence plot for protein 1mkc_A_00 having protein length L=43 and 142 sequences in the alignment (Neff=96). **Right** Convergence plot for protein 1c75_A_00 having protein length L=71 and 28078 sequences in the alignment (Neff=16808). Figure is cut at the yaxis at $||\mathbf{w}||_2 = 1500$, but learning rate of $5\text{e}{-3}$ reaches $||\mathbf{w}||_2 \approx 13000$.

the optimization but the parameter estimates eventually converge as the learning rate decreases over time.

These observations can be explained by the fact that the magnitude of the gradient scales with the number of sequences in the alignment. The gradient is computed from amino acid counts as explained in section **??**. Therefore, alignments with many sequences will generally produce larger gradients than alignments with few sequences, especially at the beginning of the optimization procedure when the difference in amino acid counts between sampled and observed sequences is largest. Following these observations, I defined the initial learning rate α_0 as a function of Neff, aiming at values for α_0 around 5e-3 for small Neff and values for α_0 around 1e-4 for large Neff,

$$\alpha_0 = \frac{5\text{e}{-2}}{\sqrt{N_{\text{eff}}}} \; . \tag{3.47}$$

For small Neff $\approx$ 50 this definition of the learning rate yields $\alpha_0 \approx 7\text{e}{-3}$ and for big Neff $\approx$ 20000 this yields $\alpha_0 \approx 3.5\text{e}{-4}$. Using this learning rate defined as a function of Neff, precision improves over the previous fixed learning rates (see Figure 3.6). All following analyses are conducted using the Neff-dependent learning rate.

In a next step, I evaluated the following learning rate annealing schedules and decay rates using the Neff-dependent initial learning rate given in eq. (3.47):

- default linear learning rate schedule $\alpha = \frac{\alpha_0}{1+\gamma t}$ with $\gamma \in \{1\mathrm{e}{-3}, 1\mathrm{e}{-2}, 1\mathrm{e}{-1}, 1\}$
- square root learning rate schedule $\alpha = \frac{\alpha_0}{\sqrt{1+\gamma t}}$ with $\gamma \in \{1\mathrm{e}{-2}, 1\mathrm{e}{-1}, 1\}$
- sigmoidal learning rate schedule $\alpha_{t+1} = \frac{\alpha_t}{1+\gamma t}$ with $\gamma \in \{1\mathrm{e}{-6}, 1\mathrm{e}{-5}, 1\mathrm{e}{-4}, 1\mathrm{e}{-3}\}$
- exponential learning rate schedule $\alpha_{t+1} = \alpha_0 \cdot \exp(-\gamma t)$ with $\gamma \in \{5\mathrm{e}{-4}, 1\mathrm{e}{-4}, 5\mathrm{e}{-3}\}$

The learning rate annealing schedules are visualized for different decay rates in Appendix Figure D.1 and the respective benchmark plots can be found in Appendix D.2. Optimizing CD with SGD using any of the learning rate schedules listed above yields on average lower precision for the top ranked contacts than the pseudo-likelihood contact score. Several learning rate schedules perform almost equally well as can be seen in Figure 3.8. The highest precision, being one to two percentage points below the mean precision for the pseudo-likelihood contact score, is obtained with a linear learning rate schedule and decay rate $\gamma = 1\mathrm{e}{-2}$, with a sigmoidal learning rate schedule and decay rates $\gamma = 1\mathrm{e}{-5}$ and $\gamma = 1\mathrm{e}{-6}$ and with an exponential learning rate schedule and decay rate $\gamma = 1\mathrm{e}{-3}$ and $\gamma = 1\mathrm{e}{-5}$. The square root learning rate schedule gives ovarally bad results and does not lead to convergence because the learning rate decays slowly at later time steps.

In contrast to the findings regarding the initial learning rate earlier, an optimal decay rate can be defined independent of the alignment size. Figure 3.9 shows convergence plots for the same two exemplary proteins as before. Proteins with low Neff are robust against the particular choice of learning rate schedule and decay rate (see left plot in Figure 3.9). The presumed optimum at $||w||_2 \approx 12.5$ is almost always reached. Proteins with high Neff are stronger adversely affected by quickly decaying learning rates and the optimum at $||w||_2 \approx 125$ cannot be reached before the learning rate diminishes which effectively prevents further optimization progress. Less quickly decaying learning rates, such as $\gamma = 1\mathrm{e}{-2}$ with a linear schedule or $\gamma = 1\mathrm{e}{-6}$ with a sigmoidal schedule, guide the parameter estimates close to the expected optimum and can be used with proteins having low Neffs as well as having high Neffs.

Several different learning rate annealing schedules yield almost identical mean precision for top ranked contacts, as was shown earlier (see Figure 3.8). But it can be found that they differ in convergence speed. Figure 3.10 shows the distribution over the number of iterations until convergence for SGD optimizations with five different learning rate schedules that yield similar performance. The optimization converges on average within less than 2000 iterations only when using either a sigmoidal learning rate annealing schedule with decay rate $\gamma = 1\mathrm{e}{-5}$ or an exponential learning rate annealing schedule with decay rate $\gamma = 5\mathrm{e}{-3}$, On the contrary, the distribution of iterations until convergence has a median of 5000 when using a linear learning rate annealing schedule with $\gamma = 1\mathrm{e}{-2}$ or an exponential schedule with decay rate $\gamma = 1\mathrm{e}{-3}$. Under these considerations, I chose a sigmoidal learning rate schedule with $\gamma = 5\mathrm{e}{-6}$ for all further analysis.

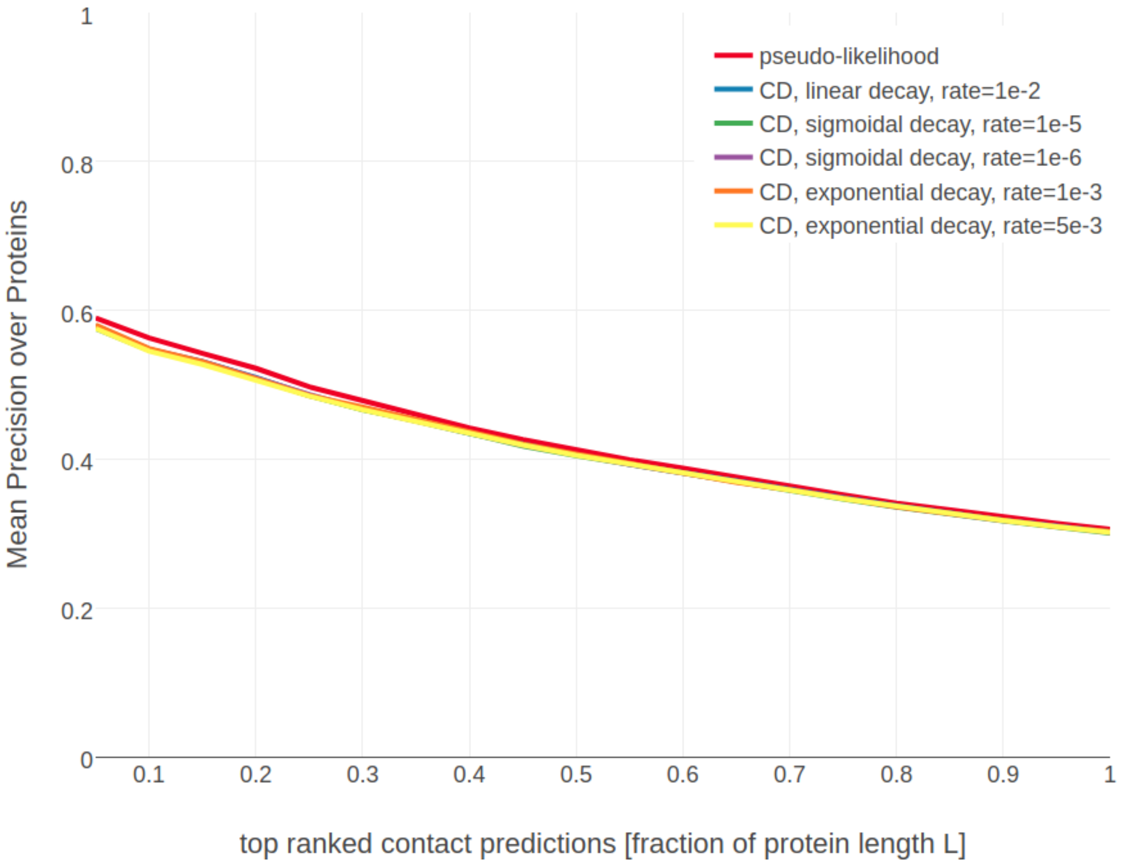

Figure 3.8: Mean precision for top ranked contact predictions over 288 proteins. Contact scores are computed as the APC corrected Frobenius norm of the couplings $\mathbf{w}_{ij}$. **pseudo-likelihood**: couplings computed with pseudo-likelihood. **CD**: couplings computed with CD using stochastic gradient descent with an initial learning rate defined with respect to Neff. Learning rate annealing schedules and decay rates as specified in the legend.

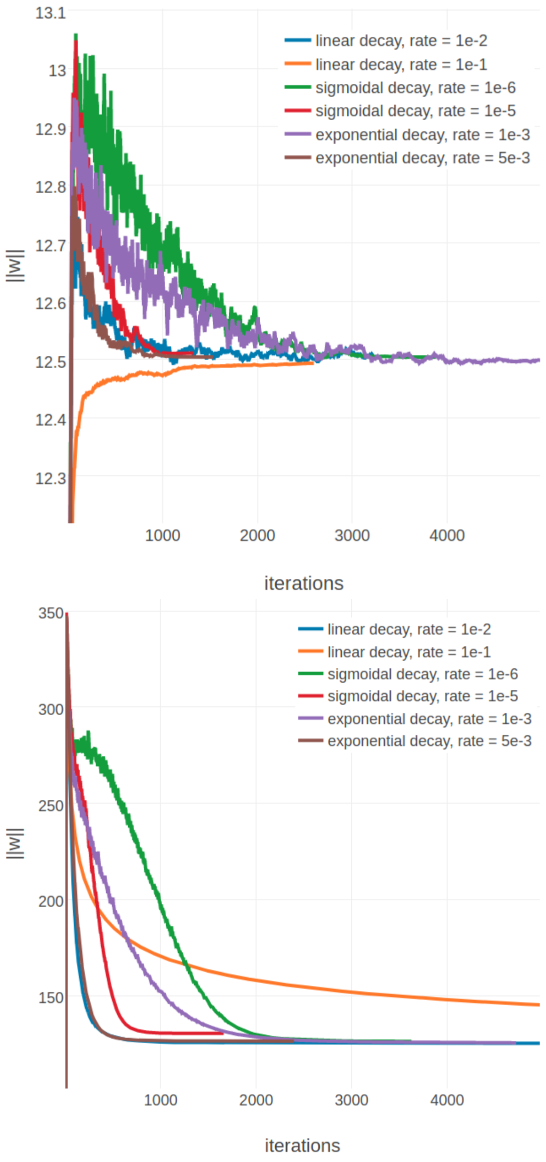

Figure 3.9: L2-norm of the coupling parameters $||\mathbf{w}||_2$ during stochastic gradient descent optimization with different learning rates schedules. The initial learning rate α_0 is defined with respect to Neff as given in eq. (3.47). Learning rate schedules and decay rates are used according to the legend. **Left** Convergence plot for protein 1mkc_A_00 having protein length L=43 and 142 sequences in the alignment (Neff=96). **Right** Convergence plot for protein 1c75_A_00 having protein length L=71 and 28078 sequences in the alignment (Neff=16808).

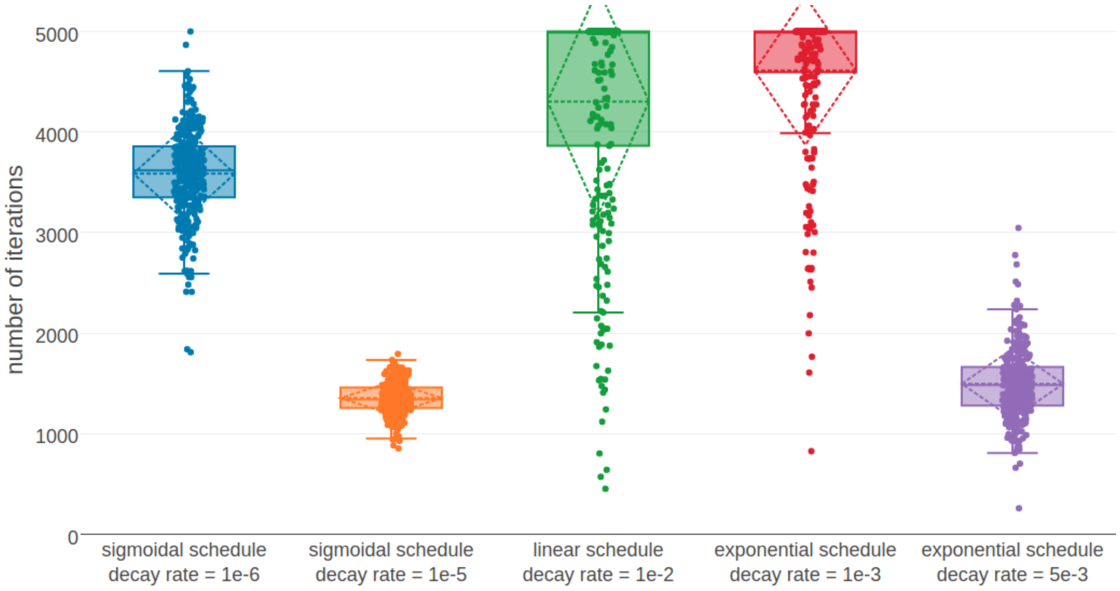

Figure 3.10: Distribution of the number of iterations until convergence for SGD optimizations of the full likelihood for different learning rate schedules. Convergence is reached when the relative difference of parameter norms $||\mathbf{w}||_2$ falls below $\epsilon = 1e - 8$. Initial learning rate α_0 is defined with respect to Neff as given in eq. (3.47) and maximum number of iterations is set to 5000. Learning rate schedules and decay rates are used as specified in the legend.

3.8.3 Tuning Hyperparameters of *ADAM* Optimizer

ADAM [32] stores an exponentially decaying average of past gradients and squared gradients,

$$m_t = \beta_1 m_{t1} + (1\beta_1)g \tag{3.48}$$

$$v_t = \beta_2 v_{t1} + (1\beta_2)g^2 \ , \tag{3.49}$$

with $g = \nabla_w LL_{\text{reg}}(\mathbf{v}, \mathbf{w})$ and the rate of decay being determined by hyperparameters β_1 and β_2. Both terms m_t and v_t represent estimates of the first and second moments of the gradient, respectively. The following bias correction terms compensates for the fact that the vectors m_t and v_t are both initialized at zero and therefore are biased towards zero especially at the beginning of optimization,

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.50}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \ . \tag{3.51}$$

Parameters are then updated using step size α, a small noise term ϵ and the corrected moment estimates $\hat{m}_t$, $\hat{v}_t$, according to

$$x_{t+1} = x_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.52}$$

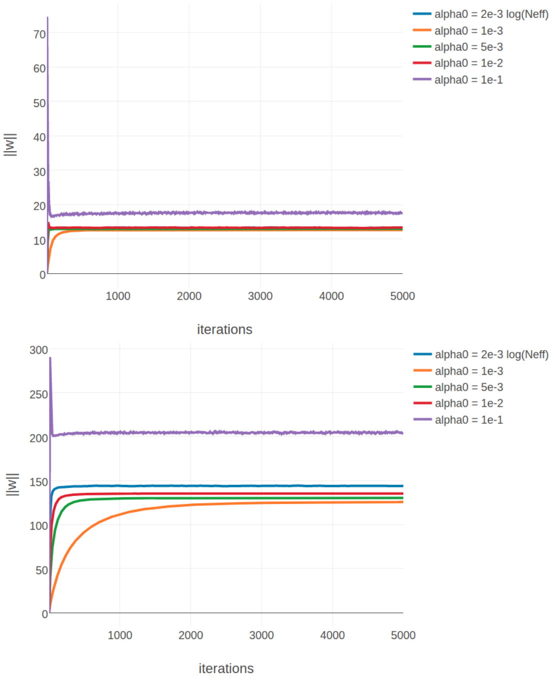

Figure 3.11: L2-norm of the coupling parameters $||\mathbf{w}||_2$ during optimization with $ADAM$ and different learning rates without annealing. The learning rate α is specified in the legend. **Left** Convergence plot for protein 1mkc_A_00 having protein length L=43 and 142 sequences in the alignment (Neff=96). **Right** Convergence plot for protein 1c75_A_00 having protein length L=71 and 28078 sequences in the alignment (Neff=16808).

Kingma et al. proposed the default values $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e8$ and a constant learning rate $\alpha = 1e - 3$.

For the two protein chains 1mkc_A_00 and 1c75_A_00, having 142 (Neff=96) and 28078 (Neff=16808) aligned sequences respectively, I analysed the convergence for SGD with different learning rates α (see Figure 3.11). In contrast to plain stochastic gradient descent, with $ADAM$ it is possible to use larger learning rates for proteins having big alignments, because the learning rate will be adapted to the magnitude of the gradient for every parameter individually. For protein 1mkc_A_00 having a small alignment, a learning rate of 5e-3 quickly leads to convergence whereas for protein 1c75_A_00 a larger learning rate can be chosen to obtain quick convergence. As a consequence, I defined the learning rate α as a function of Neff,

$$\alpha = 2\text{e}{-3} \log(N_{\text{eff}}) \ , \tag{3.53}$$

such that it will take values ˜5e-3 for proteins with small alignments and values ˜1e-2 for proteins with large alignments.

It is interesting to note in Figure 3.11, that the norm of the coupling parameters $||\mathbf{w}||_2$ converges towards different values depending on the choice of the learning rate α. This inidicates that it is necessary to decrease the learning rate over time. By default, $ADAM$ uses a constant learning rate, because the algorithm performs a kind of step size annealing by nature. However, popular implementations of $ADAM$ in the Keras [33] and Lasagne [34] packages allow the use of an annealing schedule.

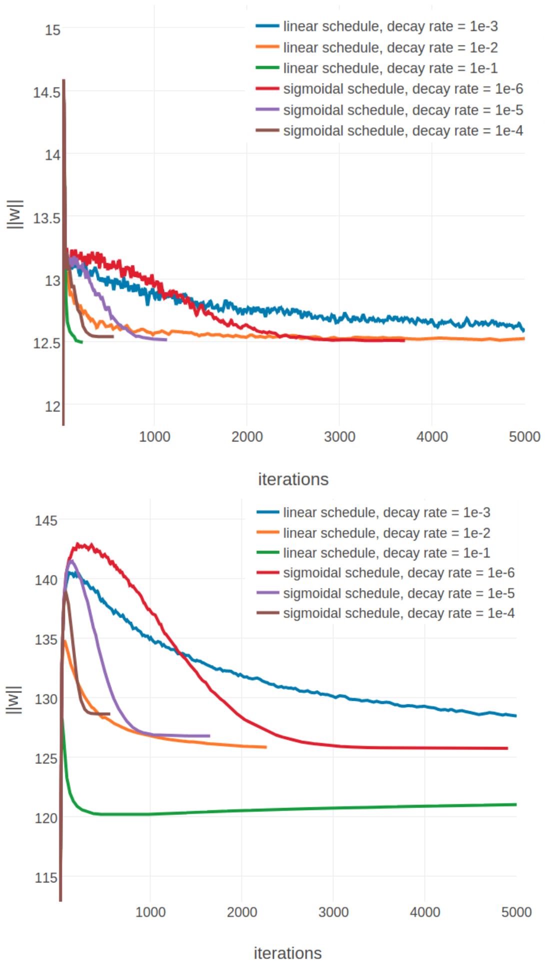

Figure 3.12: L2-norm of the coupling parameters $||\mathbf{w}||_2$ during optimization with *ADAM* and different learning rate annealing schedules. The learning rate α is specified with respect to Neff as $\alpha = 2\mathrm{e}{-3}\log(N_{\text{eff}})$. The learning rate annealing schedule is specified in the legend. **Left** Convergence plot for protein 1mkc_A_00 having protein length L=43 and 142 sequences in the alignment (Neff=96). **Right** Convergence plot for protein 1c75_A_00 having protein length L=71 and 28078 sequences in the alignment (Neff=16808).

I therefore tested different learning rate annealing schedules for *ADAM* assuming that with decreasing learning rates the L2-norm of the coupling parameters $||\mathbf{w}||_2$ will converge towards a consistent value. Indeed, as can be seen in Figure 3.12, when using a linear or sigmoidal learning rate annealing schedule with *ADAM*, the L2-norm of the coupling parameters $||\mathbf{w}||_2$ converges roughly towards the same value that has been obtained with plain SGD shown in Figure 3.9.

3.9 Tuning Regularization Coefficients for Contrastive Divergence

For tuning the hyperparameters of the stochastic gradient descent optimizer in the last section 3.8.2, the coupling parameters $\mathbf{w}$ were constrained by a Gaussian

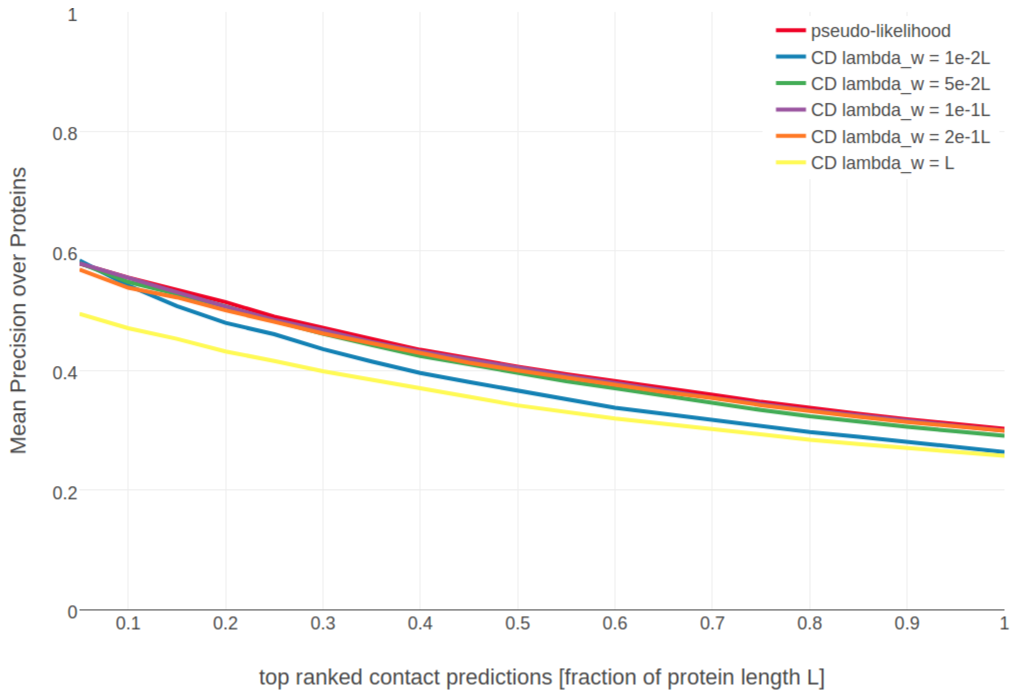

Figure 3.13: Mean precision for top ranked contact predictions over 288 proteins. Contact scores are computed as the APC corrected Frobenius norm of the couplings $\mathbf{w}_{ij}$. **pseudo-likelihood**: couplings computed with pseudo-likelihood. **CD lambda_w = X**: couplings computed with CD using L2-regularization on the couplings $\mathbf{w}$ with regularization coefficient λ_w chosen as specified in the legend and keeping the single potentials v_i fixed at their MLE optimum v_i^* given in eq. (3.43).

prior $\mathcal{N}(\mathbf{w}|0, \lambda_w^{-1} I)$ using the default pseudo-likelihood regularization coefficient $\lambda_w = 1\text{e}{-}2L$ as decscribed in methods section 3.5. It is conceivable that CD achieves optimal performance using stronger or weaker regularization than used for pseudo-likelihood optimization. Therefore, I evaluated performance of CD using the previously identified hyperparamters for SGD and different regularization coefficients $\lambda_w \in \{1\text{e}{-}2L, 5\text{e}{-}2L, 1\text{e}{-}1L, 1\text{e}{-}2L, L\}$. The single potentials $\mathbf{v}$ are not subject to optimization and are kept fixed at their maximum-likelihood estimate v^* given in eq. (3.43).

As can be seen in Figure 3.13, using strong regularization for the couplings, $\lambda_w = L$, results in a drastic drop of mean precision. Using weaker regularization such as $\lambda_w = 1\text{e}{-}2L$ or $\lambda_w = 5\text{e}{-}2L$ improves precision for the top $L/10$ and $L/5$ predicted contacts but decreases precision when including lower ranked predictions. As a matter of fact, a slightly weaker regularization $\lambda_w = 1\text{e}{-}1L$ than the default $\lambda_w = 1\text{e}{-}2L$ improves mean precision especially for the top $L/2$ contacts in such a way, that it is comparable to the pseudo-likelihood performance.

As mentioned before, a difference compared to pseudo-likelihood optimization is that the single potentials $\mathbf{v}$ are not optimized with CD but rather set to their maximum-likelihood estimate as it is obtained in a single position model that is discussed in methods section (3.43). When the single potentials $\mathbf{v}$ are optimized with CD using the same regularization coefficient $\lambda_v = 10$ as it is used when optimizing the pseudo-likelihood, performance is almost indistinguishable compared to keeping the single potentials $\mathbf{v}$ fixed as can be seen in appendix Figure D.10.

3.10 Tuning the Gibbs Sampling Scheme for Contrastive Divergence

This section describes the default Gibbs sampling scheme that is used to approximate the gradients with CD.

The gradient of the full log likelihood with respect to the couplings $\mathbf{w}$ is computed as the difference of paiwise amino acid counts between the input alignment and a sampled alignment plus an additional regularization term as given in eq. (**??**). Pairwise amino acid counts for the input alignment are computed accounting for sequence weights (described in methods section 3.3) and including pseudo counts (described in methods section 3.4). Pairwise amino acid counts for the sampled alignment are computed in the same way using the same sequence weights that have been computed for the input alignment. A subset of sequences of size $S = \min(10L, N)$, with L being the length of sequences and N the number of sequences in the input alignment, is selected from the input alignment and used to initialize the Markov chains for the Gibbs sampling procedure. Because the input MSA is typically bigger than the sampled MSA (sampled alignment has $\min(10L, N)$ sequences), the weighted pairwise amino acid counts of the sampled alignment need to be rescaled such that the total sample counts match the total counts from the input alignment.

The default implementation of the Gibbs sampler will sample new sequences by performing one full step of Gibbs sampling on each sequence as follows:

```
# Input: multiple sequence alignment X  with N sequences of length L
# Input: model parameters v and w

N = dim(X)[0]     # number of sequences in alignment
L = dim(X)[1]     # length of sequences in alignment
S = min(10L, N)   # number of sequences that will be sampled
K = 1             # number of Gibbs steps

# randomly select S sequences from the input alignment X without replacement
sequences = random.select.rows(X, size=S, replace=False)
for seq in sequences:
    # perform K steps of Gibbs sampling
    for step in range(K):
        # iterate over permuted sequence positions i in {1, ..., L}
        for i in shuffle(range(L)):
            # ignore gap positions
            if seq[i] == gap:
              continue
            # compute conditional probabilities for every amino acid a in {1, ..., 20
            for a in range(20):
```

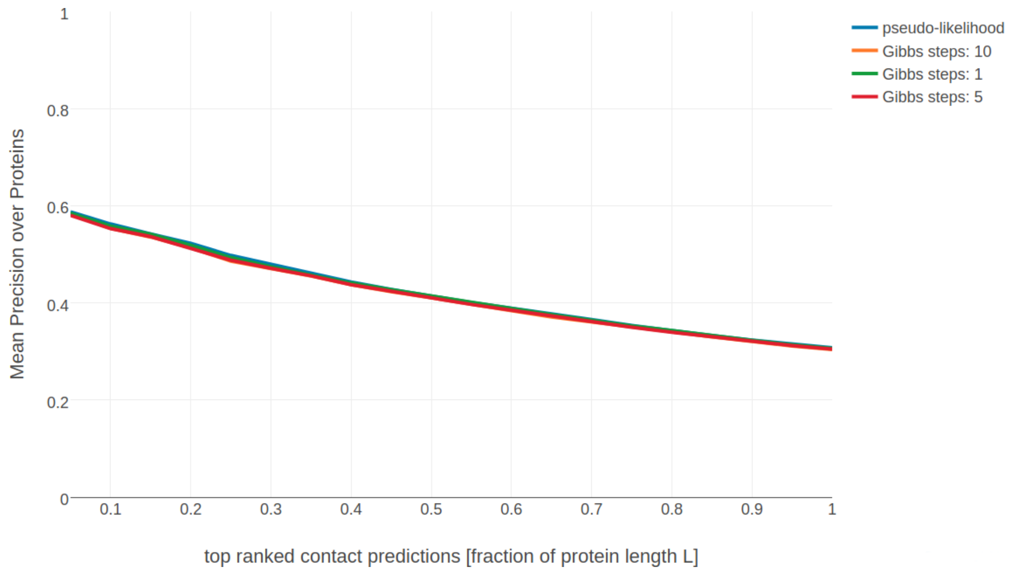

Figure 3.14: Performance of contrastive divergence optimization of the full likelihood with different number of Gibbs steps compared to pseudo-likelihood (blue) for 287 proteins. Contact scores are computed as the APC corrected Frobenius norm of the couplings $\mathbf{w}_{ij}$. pseudo-likelihood: contact scores computed from pseudo-likelihood. The other methods derive contact scores from couplings computed from CD with different number of Gibbs sampling steps.

```
       p_cond[a] = p(seq[i]=a | (seq[1], ..., seq[i-1], seq[i+1], ..., se
      # randomly select a new amino acid a in {1, ..., 20} for position i
      # according to conditional probabilities
      seq[i] = random.integer({1, ...,20}, p_cond)

# sequences will now contain S newly sampled sequences
return sequences
```

3.10.1 Varying the number of Gibbs Steps

The default CD algorithm as described by Hinton in 2002 applies only one full step of Gibbs sampling on each data sample to generate a sampled data set that will be used to approximate the gradient [35]. One full step of Gibbs sampling corresponds to sampling each position in a protein sequence according to the conditional probabilities computed from the current model probabilties as described in ??. The sampled sequences obtained after only one step of Gibbs sampling will be very similar to the input sequences. It has been shown that sampling with $n > 1$ steps gives more precise results but increases computational cost per gradient evaluation [36,37].

In the following I analysed the impact on performance when Gibbs sampling each sequence with 1, 5 and 10 full steps. As can be seen, there is hardly an impact on precision while having much longer runtimes (by a factor or 5 and 10).

It is sometimes convenient to block some of the variables to improve mixing (Jensen, Kong, & Kjrulff, 1995;Wilkinson & Yeung, 2002). Mixing is slow if consecutive samples are highly dependent. Improve mixing by sampling sets of variables [38] Sample a percentage of all positions: - 100% - 90% - 65% - 50% - 1 = pLL

Another variant of CD that has been suggested by Tieleman in 2008 is *PCD*[37] that does not reset the Markov Chains at every iteration. The reason being that when using small learning rates, the model changes only slightly between iterations and the true data distribution can be better approximated However, subsequent samples of *PCD* will be highly correlated creating a kind of momentum effect. Furthermore is has been found that *PCD* should be used with smaller learning rates and higher minibatch sizes.

As PCD might require samller update steps and larger minibatches, I analysed the performance of PCD for the default settings of CD and additionally for smaller learning and decay rates and larger minibatches. Note that one Markov chain is kept for every sequence of the input alignment. At each iteration a subset $N' < N$ of the Markov chains is randomly selected (without replacement) and used to for another round of Gibss sampling at the current iteration.

PLOT PCD for different LEARNIGN RATEWS and SAMPLE SIZES

Discussion: - as could be seen: improved gradients and different solutions do not translate into improved precision of top ranked contacts - APC corrected l2norm might not be an appropriate measure for CD couplings: look at correlation plot of couplings (pll vs cd) and l2norm (pll vs cd) and apc l2norm (pll vs cd) –> differences vanish - ranking of residues might not be influenced by subtle changes in parameters when crude l2norm is computed –> rank plot: merged list of top ranked contacts from both methods what can we see: - generally pll has stronger scores (see also boxplot over all proteins? statistic?) - ranking is very similar, especially for top ranked contacts (thats why benchmark plots so similar)

-

3.11 Bayesian Model for Residue-Resdiue Contact Prediction

3.11.1 Efficiently Computing the negative Hessian of the regularized log-likelihood

Surprisingly, the elements of the Hessian at the mode $\mathbf{w}^*$ are easy to compute. Let $i, j, k, l \in \{1, \ldots, L\}$ be columns in the MSA and let $a, b, c, d \in \{1, \ldots, 20\}$ represent amino acids.

The partial derivative $\partial / \partial \mathbf{w}_{klcd}$ of the second term in the gradient of the couplings in eq. (3.35) is

$$\frac{\partial^2 LL_{\text{reg}}(\mathbf{v}^*, \mathbf{w})}{\partial w_{klcd}\, \partial w_{ijab}} = -\sum_{n=1}^{N} \sum_{\mathbf{y} \in S_n} \frac{\partial \left(\frac{\exp\left(\sum_{i=1}^{L} v_i(y_i) + \sum_{1 \le i < j \le L}^{L} w_{ij}(y_i, y_j)\right)}{Z_n(\mathbf{v}, \mathbf{w})} \right)}{\partial w_{klcd}} I(y_i \!=\! a, y_j \!=\! b) \tag{3.54}$$

$$-\lambda_w \delta_{ijab,klcd}\,, \tag{3.55}$$

where $\delta_{ijab,klcd} = I(ijab = klcd)$ is the Kronecker delta. Applying the product rule, we find

$$\frac{\partial^2 LL_{\text{reg}}(\mathbf{v}^*, \mathbf{w})}{\partial w_{klcd}\, \partial w_{ijab}} = -\sum_{n=1}^{N} \sum_{\mathbf{y} \in S_n} \frac{\exp\left(\sum_{i=1}^{L} v_i(y_i) + \sum_{1 \le i < j \le L}^{L} w_{ij}(y_i, y_j)\right)}{Z_n(\mathbf{v}, \mathbf{w})} I(y_i \!=\! a, y_j \!=\! b) \tag{3.56}$$

$$\times \left[\frac{\partial}{\partial w_{klcd}} \left(\sum_{i=1}^{L} v_i(y_i) + \sum_{1 \le i < j \le L} w_{ij}(y_i, y_j) \right) - \frac{1}{Z_n(\mathbf{v}, \mathbf{w})} \frac{\partial Z_n(\mathbf{v}, \mathbf{w})}{\partial w_{klcd}} \right] \tag{3.57}$$

$$- \lambda_w \delta_{ijab,klcd} \tag{3.58}$$

$$\frac{\partial^2 LL_{\text{reg}}(\mathbf{v}^*, \mathbf{w})}{\partial w_{klcd}\, \partial w_{ijab}} = -\sum_{n=1}^{N} \sum_{\mathbf{y} \in S_n} \frac{\exp\left(\sum_{i=1}^{L} v_i(y_i) + \sum_{1 \le i < j \le L}^{L} w_{ij}(y_i, y_j)\right)}{Z_n(\mathbf{v}, \mathbf{w})} I(y_i \!=\! a, y_j \!=\! b) \tag{3.59}$$

$$\times \left[I(y_k \!=\! c, y_l \!=\! d) - \frac{\partial}{\partial w_{klcd}} \log Z_n(\mathbf{v}, \mathbf{w}) \right] \tag{3.60}$$

$$- \lambda_w \delta_{ijab,klcd}\,. \tag{3.61}$$

We simplify this expression using

$$p(\mathbf{y}|\mathbf{v}, \mathbf{w}) = \frac{\exp\left(\sum_{i=1}^{L} v_i(y_i) + \sum_{1 \le i < j \le L} w_{ij}(y_i, y_j)\right)}{Z_n(\mathbf{v}, \mathbf{w})}, \tag{3.62}$$

yielding

$$\frac{\partial^2 LL_{\text{reg}}(\mathbf{v}^*, \mathbf{w})}{\partial w_{klcd}\, \partial w_{ijab}} = -\sum_{n=1}^{N} \sum_{\mathbf{y} \in S_n} p(\mathbf{y}|\mathbf{v}, \mathbf{w})\, I(y_i \!=\! a, y_j \!=\! b, y_k \!=\! c, y_l \!=\! d) \tag{3.63}$$

$$+ \sum_{n=1}^{N} \sum_{\mathbf{y} \in \mathcal{S}_n} p(\mathbf{y}|\mathbf{v}, \mathbf{w})\, I(y_i \!=\! a, y_j \!=\! b) \sum_{\mathbf{y} \in S_n} p(\mathbf{y}|\mathbf{v}, \mathbf{w}) I(y_k \!=\! c, y_l \!=\! d) \tag{3.64}$$

$$- \lambda_w \delta_{ijab,klcd}\,. \tag{3.65}$$

If $\mathbf{X}$ does not contain too many gaps, this expression can be approximated by

$$\frac{\partial^2 LL_{\text{reg}}(\mathbf{v}^*, \mathbf{w})}{\partial w_{klcd}\, \partial w_{ijab}} = -N_{ijkl}\, p(x_i \!=\! a, x_j \!=\! b, x_k \!=\! c, x_l \!=\! d|\mathbf{v}, \mathbf{w})$$

$$+N_{ijkl}\, p(x_i \!=\! a, x_j \!=\! b|\mathbf{v}, \mathbf{w})\, p(x_k \!=\! c, x_l \!=\! d|\mathbf{v}, \mathbf{w}) - \lambda_w \delta_{ijab,klcd}, \tag{3.66}$$

where N_{ijkl} is the number of sequences that have a residue in i, j, k and l. Looking at three cases separately:

- case 1: $(k,l) = (i,j)$ and $(c,d) = (a,b)$
- case 2: $(k,l) = (i,j)$ and $(c,d) \neq (a,b)$
- case 3: $(k,l) \neq (i,j)$ and $(c,d) \neq (a,b)$,

the elements of $\mathbf{H}$, which are the negative second partial derivatives of $LL_{\text{reg}}(\mathbf{v}^*, \mathbf{w})$ with respect to the components of $\mathbf{w}$, are

$$\begin{aligned}
\text{case 1 :}(\mathbf{H})_{ijab,ijab} &= N_{ij}\, p(x_i\!=\!a, x_j\!=\!b|\mathbf{v}^*, \mathbf{w}^*)\,(1 - p(x_i\!=\!a, x_j\!=\!b|\mathbf{v}^*, \mathbf{w}^*)) && (3.67)\\
&\quad +\lambda_w && (3.68)\\
\text{case 2 :}(\mathbf{H})_{ijcd,ijab} &= -N_{ij}\, p(x_i\!=\!a, x_j\!=\!b|\mathbf{v}^*, \mathbf{w}^*)\, p(x_i\!=\!c, x_j\!=\!d|\mathbf{v}^*, \mathbf{w}^*) && (3.69)\\
\text{case 3 :}(\mathbf{H})_{klcd,ijab} &= N_{ijkl}\, p(x_i\!=\!a, x_j\!=\!b, x_k\!=\!c, x_l\!=\!d|\mathbf{v}^*, \mathbf{w}^*)\\
&\quad -N_{ijkl}\, p(x_i\!=\!a, x_j\!=\!b|\mathbf{v}^*, \mathbf{w}^*)\, p(x_k\!=\!c, x_l\!=\!d|\mathbf{v}^*, \mathbf{w}^*) && (3.70)
\end{aligned}$$

We know from eq. (**??**) that at the mode $\mathbf{w}^*$ the model probabilities match the empirical frequencies up to a small regularization term,

$$p(x_i\!=\!a, x_j\!=\!b|\mathbf{v}^*, \mathbf{w}^*) = q(x_i\!=\!a, x_j\!=\!b) - \frac{\lambda_w}{N_{ij}} w^*_{ijab} \,, \tag{3.71}$$

and therefore the negative Hessian elements in cases 1 and 2 can be expressed as

$$\begin{aligned}
(\mathbf{H})_{ijab,ijab} =& N_{ij} \left(q(x_i\!=\!a, x_j\!=\!b) - \frac{\lambda_w}{N_{ij}} w^*_{ijab} \right) \left(1 - q(x_i\!=\!a, x_j\!=\!b) + \frac{\lambda_w}{N_{ij}} w^*_{ijab} \right) && (3.72)\\
&+ \lambda_w && (3.73)\\
(\mathbf{H})_{ijcd,ijab} =& - N_{ij} \left(q(x_i\!=\!a, x_j\!=\!b) - \frac{\lambda_w}{N_{ij}} w^*_{ijab} \right) \left(q(x_i\!=\!c, x_j\!=\!d) - \frac{\lambda_w}{N_{ij}} w^*_{ijcd} \right) . && (3.74)
\end{aligned}$$

In order to write the previous eq. (3.74) in matrix form, the *regularised* empirical frequencies $\mathbf{q\prime}_{ij}$ will be defined as

$$(\mathbf{q\prime}_{ij})_{ab} = q'_{ijab} := q(x_i\!=\!a, x_j\!=\!b) - \lambda_w w^*_{ijab} / N_{ij} \,, \tag{3.75}$$

and the 400 × 400 diagonal matrix $\mathbf{Q}_{ij}$ will be defined as

$$\mathbf{Q}_{ij} := \operatorname{diag}(\mathbf{q\prime}_{ij}) \,. \tag{3.76}$$

Now eq. (3.74) can be written in matrix form

$$\mathbf{H}_{ij} = N_{ij} \left(\mathbf{Q}_{ij} - \mathbf{q\prime}_{ij} \mathbf{q\prime}_{ij}^{\mathrm{T}} \right) + \lambda_w \mathbf{I} \,. \tag{3.77}$$

3.11.2 Efficiently Computing the Inverse of Matrix $\mathbf{\Lambda}_{ij,k}$

It is possible to efficiently invert the matrix $\mathbf{\Lambda}_{ij,k} = \mathbf{H}_{ij} - \lambda_w \mathbf{I} + \Lambda_k$, that is introduced in **??** where $\mathbf{H}_{ij}$ is the 400×400 diagonal block submatrix $(\mathbf{H}_{ij})_{ab,cd} := (\mathbf{H})_{ijab,ijcd}$ and Λ_k is an invertible diagonal precision matrix that is introduced in section **??**.

Equation (3.77) can be used to write $\mathbf{\Lambda}_{ij,k}$ in matrix form as

$$\mathbf{\Lambda}_{ij,k} = \mathbf{H}_{ij} - \lambda_w \mathbf{I} + \mathbf{\Lambda}_k = N_{ij}\mathbf{Q}_{ij} - N_{ij}\mathbf{q\prime}_{ij}\mathbf{q\prime}_{ij}^{\mathrm{T}} + \mathbf{\Lambda}_k \,. \tag{3.78}$$

Owing to eqs. (3.24) and (3.39), $\sum_{a,b=1}^{20} q'_{ijab} = 1$. The previous equation (3.78) facilitates the calculation of the inverse of this matrix using the *Woodbury identity* for matrices

$$(\mathbf{A} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} \,. \tag{3.79}$$

by setting

$$\begin{align}
\mathbf{A} &= N_{ij}\mathbf{Q}_{ij} + \mathbf{\Lambda}_k \tag{3.80}\\
\mathbf{B} &= \mathbf{q\prime}_{ij} \tag{3.81}\\
\mathbf{C} &= \mathbf{q\prime}_{ij}^{\mathrm{T}} \tag{3.82}\\
\mathbf{D} &= -N_{ij}^{-1} \tag{3.83}\\
& \tag{3.84}
\end{align}$$

$$\begin{align}
(\mathbf{H}_{ij} - \lambda_w \mathbf{I} + \mathbf{\Lambda}_k)^{-1} &= \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{q\prime}_{ij}\left(-N_{ij}^{-1} + \mathbf{q\prime}_{ij}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{q\prime}_{ij}\right)^{-1}\mathbf{q\prime}_{ij}^{\mathrm{T}}\mathbf{A}^{-1} \tag{3.85}\\
&= \mathbf{A}^{-1} + \frac{(\mathbf{A}^{-1}\mathbf{q\prime}_{ij})(\mathbf{A}^{-1}\mathbf{q\prime}_{ij})^{\mathrm{T}}}{N_{ij}^{-1} - \mathbf{q\prime}_{ij}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{q\prime}_{ij}} \,. \tag{3.86}
\end{align}$$

Note that $\mathbf{A}$ is diagonal as $\mathbf{Q}_{ij}$ and $\mathbf{\Lambda}_k$ are diagonal matrices: $\mathbf{A} = \mathrm{diag}(N_{ij}q'_{ijab} + (\mathbf{\Lambda}_k)_{ab,ab})$. Moreover, $\mathbf{A}$ has only positive diagonal elements, because $\mathbf{\Lambda}_k$ is invertible and has only positive diagonal elements and because $q'_{ijab} = p(x_i = a, x_j = b|\mathbf{v}^*, \mathbf{w}^*) \geq 0$.

Therefore $\mathbf{A}$ is invertible: $\mathbf{A}^{-1} = \mathrm{diag}(N_{ij}q'_{ijab} + (\mathbf{\Lambda}_k)_{ab,ab})^{-1}$.

Because $\sum_{a,b=1}^{20} q'_{ijab} = 1$, the denominator of the second term is

$$N_{ij}^{-1} - \sum_{a,b=1}^{20} \frac{q'^2_{ijab}}{N_{ij}q'_{ijab} + (\mathbf{\Lambda}_k)_{ab,ab}} > N_{ij}^{-1} - \sum_{a,b=1}^{20} \frac{q'^2_{ijab}}{N_{ij}q'_{ijab}} = 0 \tag{3.87}$$

and therefore the inverse of $\mathbf{\Lambda}_{ij,k}$ in eq. (3.86) is well defined.

The log determinant of $\mathbf{\Lambda}_{ij,k}$ is necessary to compute the ratio of Gaussians (see equation (**??**)) and can be computed using the matrix determinant lemma:

$$\det(\mathbf{A} + \mathbf{u}\mathbf{v}^{\mathrm{T}}) = (1 + \mathbf{v}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{u}) \det(\mathbf{A}) \tag{3.88}$$

Setting $\mathbf{A} = N_{ij}\mathbf{Q}_{ij} + \mathbf{\Lambda}_k$ and $\mathbf{v} = \mathbf{q\prime}_{ij}$ and $\mathbf{u} = -N_{ij}\mathbf{q\prime}_{ij}$ yields

$$\det(\mathbf{\Lambda}_{ij,k}) = \det(\mathbf{H}_{ij} - \lambda_w \mathbf{I} + \mathbf{\Lambda}_k) = (1 - N_{ij}\mathbf{q\prime}_{ij}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{q\prime}_{ij}) \det(\mathbf{A}) \,. \tag{3.89}$$

$\mathbf{A}$ is diagonal and has only positive diagonal elements so that $\log(\det(\mathbf{A})) = \sum \log(\mathrm{diag}(\mathbf{A}))$.

3.11.3 Training the Hyperparameters μ_k, $\mathbf{\Lambda}_k$ and γ_k

The model parameters $\mu = (\mu_1, \ldots, \mu_K)$, $\mathbf{\Lambda} = (\mathbf{\Lambda}_1, \ldots, \mathbf{\Lambda}_K)$ and $\gamma = (\gamma_1, \ldots, \gamma_K)$ will be trained by maximizing the logarithm of the full likelihood over a set of training MSAs $\mathbf{X}^1, \ldots, \mathbf{X}^N$ and associated structures with distance vectors $\mathbf{r}^1, \ldots, \mathbf{r}^N$ plus a regularizer $R(\mu, \mathbf{\Lambda})$:

$$LL(\mu, \mathbf{\Lambda}, \gamma) + R(\mu, \mathbf{\Lambda}) = \sum_{n=1}^{N} \log p(\mathbf{X}^n | \mathbf{r}^n, \mu, \mathbf{\Lambda}, \gamma) + R(\mu, \mathbf{\Lambda}) \rightarrow \max \,. \tag{3.90}$$

The regulariser penalizes values of μ_k and $\mathbf{\Lambda}_k$ that deviate too far from zero:

$$R(\mu, \mathbf{\Lambda}) = -\frac{1}{2\sigma_\mu^2} \sum_{k=1}^{K} \sum_{ab=1}^{400} \mu_{k,ab}^2 - \frac{1}{2\sigma_{\text{diag}}^2} \sum_{k=1}^{K} \sum_{ab=1}^{400} \Lambda_{k,ab,ab}^2 \tag{3.91}$$

Reasonable values are $\sigma_\mu = 0.1$, $\sigma_{\text{diag}} = 100$.

The log likelihood can be optimized using LBFG-S-B[**???**], which requires the computation of the gradient of the log likelihood. For simplicity of notation, the following calculations consider the contribution of the log likelihood for just one protein, which allows to drop the index n in r_{ij}^n, $(\mathbf{w}_{ij}^n)^*$ and $\mathbf{H}_{ij}^n$.

From eq. (**??**) the log likelihood for a single protein is

$$LL(\mu, \mathbf{\Lambda}, \gamma_k) = \sum_{1 \le i < j \le L} \log \sum_{k=0}^{K} g_k(r_{ij}) \frac{\mathcal{N}(\mathbf{0} | \mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0} | \mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})} + R(\mu, \mathbf{\Lambda}) + \text{const.} \,. \tag{3.92}$$

3.11.4 The gradient of the log likelihood with respect to μ

By applying the formula $df(x)/dx = f(x)\, d \log f(x)/dx$ to compute the gradient of eq. (3.92) (neglecting the regularization term) with respect to $\mu_{k,ab}$, one obtains

$$\frac{\partial}{\partial \mu_{k,ab}} LL(\mu, \mathbf{\Lambda}, \gamma_k) = \sum_{1 \leq i < j \leq L} \frac{g_k(r_{ij}) \frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})} \frac{\partial}{\partial \mu_{k,ab}} \log \left(\frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})} \right)}{\sum_{k'=0}^{K} g_{k'}(r_{ij}) \frac{\mathcal{N}(\mathbf{0}|\mu'_k, \mathbf{\Lambda}'^{-1}_k)}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})}} . \tag{3.93}$$

To simplify this expression, we define the responsibility of component k for the posterior distribution of $\mathbf{w}_{ij}$, the probability that $\mathbf{w}_{ij}$ has been generated by component k:

$$p(k|ij) = \frac{g_k(r_{ij}) \frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})}}{\sum_{k'=0}^{K} g_{k'}(r_{ij}) \frac{\mathcal{N}(\mathbf{0}|\mu'_k, \mathbf{\Lambda}'^{-1}_k)}{\mathcal{N}(\mathbf{0}|\mu'_{ij,k}, \mathbf{\Lambda}'^{-1}_{ij,k})}} . \tag{3.94}$$

By substituting the definition for responsibility, (3.93) simplifies

$$\frac{\partial}{\partial \mu_{k,ab}} LL(\mu, \mathbf{\Lambda}, \gamma_k) = \sum_{1 \leq i < j \leq L} p(k|ij) \frac{\partial}{\partial \mu_{k,ab}} \log \left(\frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})} \right) , \tag{3.95}$$

and analogously for partial derivatives with respect to $\Lambda_{k,ab,cd}$.

The partial derivative inside the sum can be written

$$\frac{\partial}{\partial \mu_{k,ab}} \log \left(\frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})} \right) = \frac{1}{2} \frac{\partial}{\partial \mu_{k,ab}} \left(\log |\mathbf{\Lambda}_k| - \mu_k^{\mathrm{T}} \mathbf{\Lambda}_k \mu_k - \log |\mathbf{\Lambda}_{ij,k}| + \mu_{ij,k}^{\mathrm{T}} \mathbf{\Lambda}_{ij,k} \mu_{ij,k} \right) . \tag{3.96}$$

Using the following formula for a matrix $\mathbf{A}$, a real variable x and a vector $\mathbf{y}$ that depends on x,

$$\frac{\partial}{\partial x} \left(\mathbf{y}^{\mathrm{T}} \mathbf{A} \mathbf{y} \right) = \frac{\partial \mathbf{y}^{\mathrm{T}}}{\partial x} \mathbf{A} \mathbf{y} + \mathbf{y}^{\mathrm{T}} \mathbf{A} \frac{\partial \mathbf{y}}{\partial x} = \mathbf{y}^{\mathrm{T}} (\mathbf{A} + \mathbf{A}^{\mathrm{T}}) \frac{\partial \mathbf{y}}{\partial x} \tag{3.97}$$

the partial derivative therefore becomes

$$\begin{aligned} \frac{\partial}{\partial \mu_{k,ab}} \log \left(\frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})} \right) &= \left(-\mu_k^{\mathrm{T}} \mathbf{\Lambda}_k \mathbf{e}_{ab} + \mu_{ij,k}^{\mathrm{T}} \mathbf{\Lambda}_{ij,k} \mathbf{\Lambda}_{ij,k}^{-1} \mathbf{\Lambda}_k \mathbf{e}_{ab} \right) && (3.98) \\ &= \mathbf{e}_{ab}^{\mathrm{T}} \mathbf{\Lambda}_k (\mu_{ij,k} - \mu_k) \, . && (3.99) \end{aligned}$$

Finally, the gradient of the log likelihood with respect to μ becomes

$$\nabla_{\mu_k} LL(\mu, \mathbf{\Lambda}, \gamma_k) = \sum_{1 \leq i < j \leq L} p(k|ij) \, \mathbf{\Lambda}_k \left(\mu_{ij,k} - \mu_k \right) \, . \tag{3.100}$$

3.11.5 The gradient of the log likelihood with respect to $\mathbf{\Lambda}_k$

Analogously to eq. (3.95) one first needs to solve

$$\begin{aligned}
&\frac{\partial}{\partial \Lambda_{k,ab,cd}} \log \frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})} = \tag{3.101}\\
&\frac{1}{2}\frac{\partial}{\partial \Lambda_{k,ab,cd}} \left(\log|\mathbf{\Lambda}_k| - \mu_k^{\mathrm{T}}\mathbf{\Lambda}_k\mu_k - \log|\mathbf{\Lambda}_{ij,k}| + \mu_{ij,k}^{\mathrm{T}}\mathbf{\Lambda}_{ij,k}\mu_{ij,k}\right) \;, \tag{3.102}
\end{aligned}$$

by applying eq. (3.97) as before as well as the formulas

$$\begin{aligned}
\frac{\partial}{\partial x}\log|\mathbf{A}| &= \mathrm{Tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\right), \tag{3.103}\\
\frac{\partial \mathbf{A}^{-1}}{\partial x} &= -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{A}^{-1}\,. \tag{3.104}
\end{aligned}$$

This yields

$$\begin{aligned}
\frac{\partial}{\partial \Lambda_{k,ab,cd}}\log|\mathbf{\Lambda}_k| &= \mathrm{Tr}\left(\mathbf{\Lambda}_k^{-1}\frac{\partial \mathbf{\Lambda}_k}{\partial \Lambda_{k,ab,cd}}\right) = \mathrm{Tr}\left(\mathbf{\Lambda}_k^{-1}\mathbf{e}_{ab}\mathbf{e}_{cd}^{\mathrm{T}}\right) = \Lambda^{-1}_{k,cd,ab} \tag{3.105}\\
\frac{\partial}{\partial \Lambda_{k,ab,cd}}\log|\mathbf{\Lambda}_{ij,k}| &= \mathrm{Tr}\left(\mathbf{\Lambda}_{ij,k}^{-1}\frac{\partial(\mathbf{H}_{ij} - \lambda_w\mathbf{I} + \mathbf{\Lambda}_k)}{\partial \Lambda_{k,ab,cd}}\right) = \Lambda^{-1}_{ij,k,cd,ab} \tag{3.106}\\
\frac{\partial(\mu_k^{\mathrm{T}}\mathbf{\Lambda}_k\mu_k)}{\partial \Lambda_{k,ab,cd}} &= \mu_k^{\mathrm{T}}\mathbf{e}_{ab}\mathbf{e}_{cd}^{\mathrm{T}}\mu_k = \mathbf{e}_{ab}^{\mathrm{T}}\mu_k\mu_k^{\mathrm{T}}\mathbf{e}_{cd} = (\mu_k\mu_k^{\mathrm{T}})_{ab,cd} \tag{3.107}\\
\frac{\partial(\mu_{ij,k}^{\mathrm{T}}\mathbf{\Lambda}_{ij,k}\mu_{ij,k})}{\partial \Lambda_{k,ab,cd}} &= \mu_{ij,k}^{\mathrm{T}}\frac{\partial \mathbf{\Lambda}_{ij,k}}{\partial \Lambda_{k,ab,cd}}\mu_{ij,k} + 2\mu_{ij,k}^{\mathrm{T}}\mathbf{\Lambda}_{ij,k}\frac{\partial \mathbf{\Lambda}_{ij,k}^{-1}}{\partial \Lambda_{k,ab,cd}}(\mathbf{H}_{ij}\mathbf{w}_{ij}^{*} + \mathbf{\Lambda}_k\mu_k) + 2\mu_{ij,k}^{\mathrm{T}}\frac{\partial \mathbf{\Lambda}_k}{\partial \Lambda_{k,ab,cd}}\mu_k\\
&= (\mu_{ij,k}\mu_{ij,k}^{\mathrm{T}} + 2\mu_{ij,k}\mu_k^{\mathrm{T}})_{ab,cd} - 2\mu_{ij,k}^{\mathrm{T}}\mathbf{\Lambda}_{ij,k}\mathbf{\Lambda}_{ij,k}^{-1}\frac{\partial \mathbf{\Lambda}_{ij,k}}{\partial \Lambda_{k,ab,cd}}\mathbf{\Lambda}_{ij,k}^{-1}(\mathbf{H}_{ij}\mathbf{w}_{ij}^{*} + \mathbf{\Lambda}_k\mu_k) \tag{3.108}\\
&= (\mu_{ij,k}\mu_{ij,k}^{\mathrm{T}} + 2\mu_{ij,k}\mu_k^{\mathrm{T}})_{ab,cd} - 2\mu_{ij,k}^{\mathrm{T}}\frac{\partial \mathbf{\Lambda}_{ij,k}}{\partial \Lambda_{k,ab,cd}}\mu_{ij,k} \tag{3.109}\\
&= (-\mu_{ij,k}\mu_{ij,k}^{\mathrm{T}} + 2\mu_{ij,k}\mu_k^{\mathrm{T}})_{ab,cd}\,. \tag{3.110}
\end{aligned}$$

Inserting these results into eq. (3.102) yields

$$\frac{\partial}{\partial \Lambda_{k,ab,cd}} \log \frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})} = \frac{1}{2}\left(\mathbf{\Lambda}_k^{-1} - \mathbf{\Lambda}_{ij,k}^{-1} - (\mu_{ij,k} - \mu_k)(\mu_{ij,k} - \mu_k)^{\mathrm{T}}\right)_{ab,cd}\,. \tag{3.111}$$

Substituting this expression into the equation (3.95) analogous to the derivation of gradient for $\mu_{k,ab}$ yields the equation

$$\nabla_{\mathbf{\Lambda}_k} LL(\mu, \mathbf{\Lambda}, \gamma_k) = \frac{1}{2} \sum_{1 \le i < j \le L} p(k|ij) \left(\mathbf{\Lambda}_k^{-1} - \mathbf{\Lambda}_{ij,k}^{-1} - (\mu_{ij,k} - \mu_k)(\mu_{ij,k} - \mu_k)^{\mathrm{T}} \right) . \tag{3.112}$$

3.11.6 The gradient of the log likelihood with respect to γ_k

With $r_{ij} \in \{0, 1\}$ defining a residue pair in physical contact or not in contact, the mixing weights can be modelled as a softmax function according to eq. (**??**). The derivative of the mixing weights $g_k(r_{ij})$ is:

$$\frac{\partial g_{k'}(r_{ij})}{\partial \gamma_k} = \left\{ \begin{array}{ll} g_k(r_{ij})(1 - g_k(r_{ij})) & : k' = k \\ g_{k'}(r_{ij}) - g_k(r_{ij}) & : k' \neq k \end{array} \right. \tag{3.113}$$

The partial derivative of the likelihood function with respect to γ_k is:

$$\frac{\partial}{\partial \gamma_k} LL(\mu, \mathbf{\Lambda}, \gamma_k) = \sum_{1 \le i < j \le L} \frac{\sum_{k'=0}^{K} \frac{\partial}{\partial \gamma_k} g_{k'}(r_{ij}) \frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(0|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})}}{\sum_{k'=0}^{K} g_{k'}(r_{ij}) \frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})}} \tag{3.114}$$

$$= \sum_{1 \le i < j \le L} \frac{\sum_{k'=0}^{K} g_{k'}(r_{ij}) \frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})} \cdot \begin{cases} 1 - g_k(r_{ij}) & \text{if } k' = k \\ -g_k(r_{ij}) & \text{if } k' \neq k \end{cases}}{\sum_{k'=0}^{K} g_{k'}(r_{ij}) \frac{\mathcal{N}(\mathbf{0}|\mu_k, \mathbf{\Lambda}_k^{-1})}{\mathcal{N}(\mathbf{0}|\mu_{ij,k}, \mathbf{\Lambda}_{ij,k}^{-1})}} \tag{3.115}$$

$$= \sum_{1 \le i < j \le L} \sum_{k'=0}^{K} p(k'|ij) \begin{cases} 1 - g_k(r_{ij}) & \text{if } k' = k \\ -g_k(r_{ij}) & \text{if } k' \neq k \end{cases} \tag{3.116}$$

$$= \sum_{1 \le i < j \le L} p(k|ij) - g_k(r_{ij}) \sum_{k'=0}^{K} p(k'|ij)$$

$$= \sum_{1 \le i < j \le L} p(k|ij) - g_k(r_{ij}) \tag{3.117}$$

3.12 Bayesian Statistical Model for Prediction of Protein Residue-Residue Distances

3.12.1 Modelling the dependence of $\mathbf{w}_{ij}$ on distance

It is straightforward to extend the model presented in **??** for distances.

The mixture weights $g_k(r_{ij})$ in eq. (**??**) are modelled as softmax over linear functions $\gamma_k(r_{ij})$ (Figure **??**fig:softmax-linear-fct):

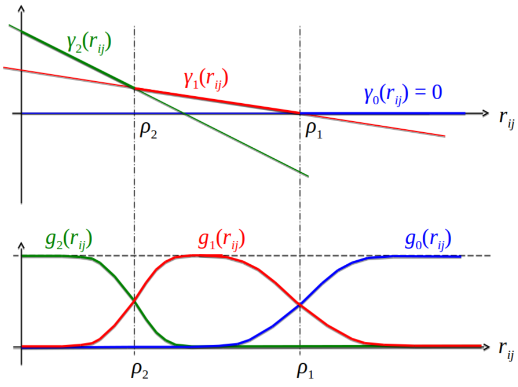

Figure 3.15: The Gaussian mixture coefficients $g_k(r_{ij})$ of $p(\mathbf{w}_{ij}|r_{ij})$ are modelled as softmax over linear functions $\gamma_k(r_{ij})$. ρ_k sets the transition point between neighbouring components $g_{k-1}(r_{ij})$ and $g_k(r_{ij})$, while α_k quantifies the abruptness of the transition between $g_{k-1}(r_{ij})$ and $g_k(r_{ij})$.

$$g_k(r_{ij}) = \frac{\exp \gamma_k(r_{ij})}{\sum_{k'=0}^{K} \exp \gamma_{k'}(r_{ij})}\,, \tag{3.118}$$

$$\gamma_k(r_{ij}) = -\sum_{k'=0}^{k} \alpha_{k'}(r_{ij} - \rho_{k'}). \tag{3.119}$$

The functions $g_k(r_{ij})$ remain invariant when adding an offset to all $\gamma_k(r_{ij})$. This degeneracy can be removed by setting $\gamma_0(r_{ij}) = 0$ (i.e., $\alpha_0 = 0$ and $\rho_0 = 0$). Further, the components are ordered, $\rho_1 > \ldots > \rho_K$ and it is demanded that $\alpha_k > 0$ for all k. This ensures that for $r_{ij} \to \infty$ we will obtain $g_0(r_{ij}) \to 1$ and hence $p(\mathbf{w}|\mathbf{X}) \to \mathcal{N}(0, \sigma_0^2 \mathbf{I})$.

The parameters ρ_k mark the transition points between the two Gaussian mixture components $k-1$ and k, i.e., the points at which the two components obtain equal weights. This follows from $\gamma_k(r_{ij}) - \gamma_{k-1}(r) = \alpha_t(r_{ij} - \rho_t)$ and hence $\gamma_{k-1}(\rho_k) = \gamma_k(\rho_k)$. A change in ρ_k or α_k only changes the behaviour of $g_{k-1}(r_{ij})$ and $g_k(r_{ij})$ in the transition region around ρ_k. Therefore, this particular definition of $\gamma_k(r_{ij})$ makes the parameters α_k and ρ_k as independent of each other as possible, rendering the optimisation of these parameters more efficient.

3.12.2 Training the Hyperparameters ρ_k and α_k for distance-dependent prior

3.13 Training Random Forest Contat Prior

3.13.1 Sequence Derived Features

Given a multiple sequence alignment of a protein family, various sequence features can be derived that have been found to be informative of a residue-residue contact.

In total there are **250** features that can be divided into global, single position and pairwise features and are described in the following sections. If not stated otherwise, *weighted* features have been computed using amino acid counts or amino acid frequencies based on weighted sequences as described in section 3.3.

3.13.1.1 Global Features

These features describe alignment characteristics. Every pair of residues (i, j) from the same protein will be attributed the same feature.

Table 3.1: Features characterizing the total alignment

Feature	Description	No. Features per residue pair (i, j)
L	log of protein length	1
N	number of sequences	1
Neff	number of effective sequences Neff computed as the sum over sequence weights (see section 3.3)	1
gaps	average percentage of gaps over all positions	1
diversity	$\frac{\sqrt{N}}{L}$, N=number of sequences, L=protein length	1
amino acid composition	weighted amino acid frequencies in alignment	20
secondary structure prediction	average three state propensities PSIPRED (v4.0)[18]	3
secondary structure prediction	average three state propensities Netsurfp (v1.0)[17]	3
contact prior protein length	simple contact predictor based on expected number of contacts per protein with respect to protein length (see next subsection 3.13.1.4)	1

There are in total **32** global alignment features.

3.13.1.2 Single Position Features

These features describe characteristics of a single alignment column. Every residue pair (i, j) will be described by two features, once for each position.

Table 3.2: Single Position Sequence Features

Feature	Description	No. Features per residue pair (i, j)
shannon entropy (20 states)	$-\sum_{a=1}^{20} p_a \log p_a$	2
shannon entropy (21 states)	$-\sum_{a=1}^{21} p_a \log p_a$	2
kullback leibler divergence	between weighted observed and background amino acid frequencies [39]	2
jennson shannon divergence	between weighted observed and background amino acid frequencies [39]	2
PSSM	log odds ratio of weighted observed and background amino acid frequencies [39]	40
secondary structure prediction	three state propensities PSIPRED (v4.0) [18]	6
secondary structure prediction	three state propensities Netsurfp (v1.0) [17]	6
solvent accessibility prediction	RSA and RSA Z-score Netsurfp (v1.0) [17]	4
relative position in sequence	$\frac{i}{L}$ for a protien of length L	2
number of ungapped sequences	$\sum_n w_n I(x_{ni} \neq 20)$ for sequences x_n and sequence weights w_n	2
percentage of gaps	$\frac{\sum_n w_n I(x_{ni}=20)}{N_{\text{eff}}}$ for sequences x_n and sequence weights w_n	2
Average physico-chemical properties	Atchley Factors 1-5 [40]	10
Average physico-chemical properties	Polarity according to Grantham, 1974. Data taken from AAindex Database [41].	2
Average physico-chemical properties	Polarity according to Zimmermann et al., 1986. Data taken from AAindex Database [41].	2
Average physico-chemical properties	Isoelectric point according to Zimmermann et al., 1968. Data taken from AAindex Database [41].	2
Average physico-chemical properties	Hydrophobicity scale according to Wimley & White, 1996. Data taken from UCSF Chimera [42].	2

Feature	Description	No. Features per residue pair (i, j)
Average physico-chemical properties	Hydrophobicity index according to Kyte & Doolittle, 1982. Data taken from AAindex Database [41].	2
Average physico-chemical properties	Hydrophobicity according to Cornette [43].	2
Average physico-chemical properties	Bulkiness according to Zimmerman et al., 1968. Data taken from AAindex Database [41].	2
Average physico-chemical properties	Average volumes of residues according to Pontius et al., 1996. Data taken from AAindex Database [41].	2

There are in total **96** single sequence features.

Additionally, all single features will be computed within a window of size 5. The window feature for center residue i will be computed as the mean feature over residues $[i-2, \ldots, i, \ldots, i+2]$. Whenever the window extends the range of the sequence (for $i < 2$ and $i > (L-2)$), the window feature will be computed only for valid sequence positions. This results in additional **96** window features.

3.13.1.3 Pairwise Features

These features are computed for every pair of columns (i, j) in the alignment with $i < j$.

Table 3.3: Pairwise Sequence Features

Feature	Description	No. Features per residue pair (i, j)
sequence separation	$j - i$	1
gaps	pairwise percentage of gaps using weighted sequences	1
number of ungapped sequences	$\sum_n w_n I(x_{ni} \neq 20, x_{nj} \neq 20)$ for sequences x_n and sequence weights w_n	1
correlation physico-chemical features	pairwise correlation of all physico-chemical properties listed in 3.13.1.2	13
pairwise potential	Average quasi-chemical energy of interactions in an average buried environment. Data taken from AAindex Database [41].	1

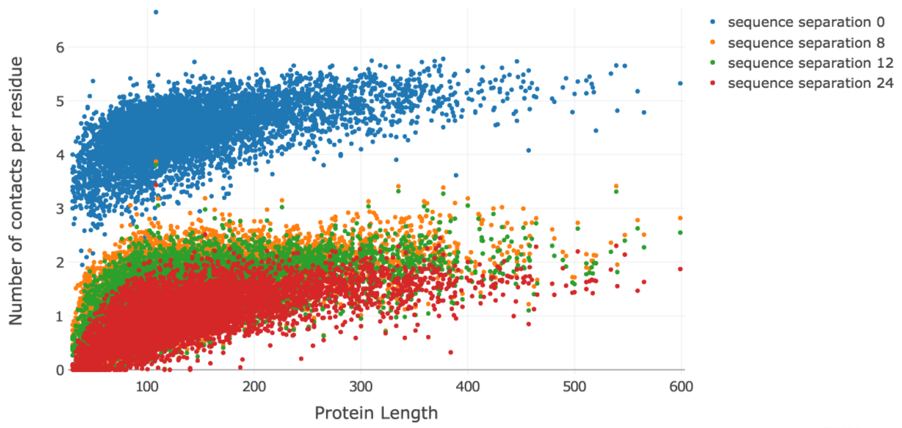

Figure 3.16: Observed number of contacts per residue has a non-linear relationship with protein length. Distribution is shown for several thresholds of sequence separation.

Feature	Description	No. Features per residue pair (i,j)
pairwise potential	Average quasi-chemical energy of transfer of amino acids from water to the protein environment. Data taken from AAindex Database [41].	1
pairwise potential	Average general contact potential by Li&Fang [16]	1
pairwise potential	Average statistical potential from residue pairs in beta-sheets by Zhu&Braun [44]	1
joint_shannon_entropy (20 state)	$-\sum_{a=1}^{20}\sum_{b=1}^{20} p(a,b)\log p(a,b)$	1
joint_shannon_entropy (21 state)	$-\sum_{a=1}^{21}\sum_{b=1}^{21} p(a,b)\log p(a,b)$	1
mutual information (MI)	mutual information of amino acid counts at two positions; several variants: MI with pseudo-counts, MI with pseudo-counts + APC, normalized MI	3
OMES	according to Fodor&Aldrich [14] with and without APC	2

There are in total **26** pairwise sequence features.

3.13.1.4 Protein length dependent Contact Prior

The average number of contats per residue, computed as the observed number of contacts divided by protein length L, has a non-linear relationship with protein length L as can be seen in Figure 3.16.

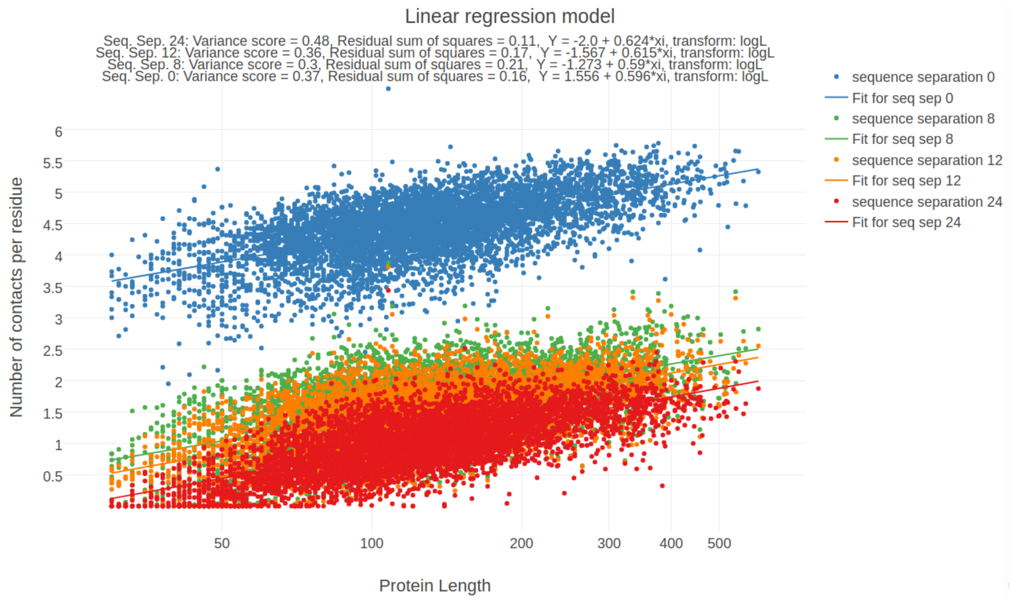

Figure 3.17: (ref:caption-avg-nr-contacts-per-residue-vs-log-protein-length-linfit)

In log space, the average number of contats per residue can be fitted with a linear regression (see Figure 3.17) and yields the following functions:

- $f(L) = 1.556 + 0.596\log(L)$ for sequence separation of 0 positions
- $f(L) = -1.273 + 0.59\log(L)$ for sequence separation of 8 positions
- $f(L) = -1.567 + 0.615\log(L)$ for sequence separation of 12 positions
- $f(L) = -2.0 + 0.624\log(L)$ for sequence separation of 24 positions

A simple contact predictor can be formulated as the ratio of the expected number of contacts per residue, given by $f(L)$, and the possible number of contacts per residue which is $L-1$,

$$p(r_{ij} = 1|L) = \frac{f(L)}{L-1} \; ,$$

with $r_{ij} = 1$ representing a contact between residue i and j.

(ref:caption-avg-nr-contacts-per-residue-vs-log-protein-length-linfit) Linear regression fits for average number of contats per residue on logarithm of protein length. Distribution and linear regression fits are shown for different sequence separation thresholds.

3.13.2 Hyperparameter Optimization for Random Forest Prior

There are several hyperparameters in a random forest model that need to be tuned to achieve best balance between predictive power and runtime. While more trees in the random forest generally improve performance of the model, they will slow down training and prediction. A crucial hyperparamter is the number of features

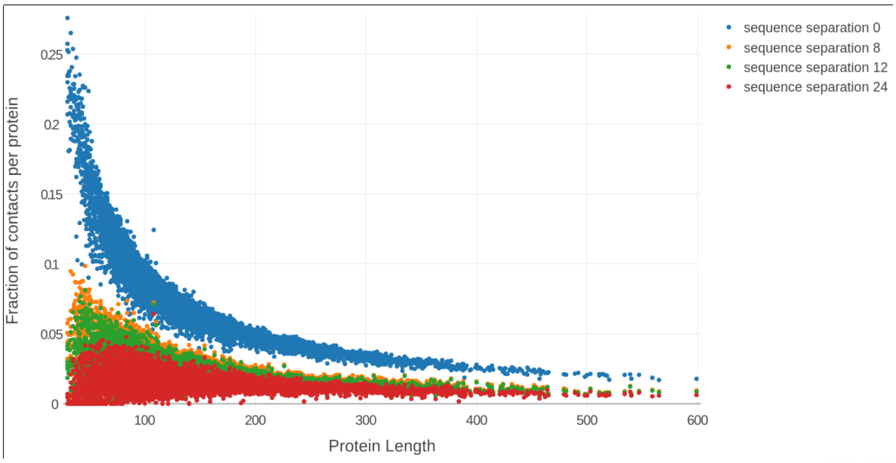

Figure 3.18: Fraction of contacts among all possible contacts ($\frac{L(L-1)}{2}$) in a protein against protein length L. The distribution has a non-linear relationship. At a sequence separation threshold >8 positions the fraction of contacts for intermediate size proteins with length >100 is approximately 2%.

that is randomly selected for a split at each node in a tree [45]. Stochasticity introduced by the random selection of features is a key characteristic of random forests as it reduces correlation between the trees and thus the variance of the predictor. Selecting many features typically increases performance as more options can be considered for each split, but at the same time increases risk of overfitting and decreases speed of the algorithm. In general, random forests are robust to overfitting, as long as there are enough trees in the ensemble and the selection of features for splitting a node introduces sufficient stochasticity. Overfitting can furthermore be prevented by restricting the depth of the trees, which is known as pruning or by enforcing a minimal node size with respect to the number of features per node. A positive side-effect of taking these measures is a speedup of the algorithm. [12]

In the following, I use 5-fold cross-validation to identify the optimal architecture of the random forest. I used the module RandomForestClassifier in the Python package `sklearn (v. 0.19)` [46] and trained the models on sequence features extracted from MSAs as described in section 3.13.1. Single position features are computed with a window of size five as described in section 3.13.1.2.

Proteins constitute highly imbalanced datasets with respect to the number of residue pairs that form and form not physical contacts. As can be seen in Figure 3.18, depending on the enforced sequence separation threshold the percentage of contacts varies between approximately 1% and 5%.

Most studies applying machine learning algorithms to the problem of predicting residue-residue contacts, chose the standard approach of rebalancing the data set by undersampling of the majority class.

Study	Proportion of Contacts	Proportion of Non-contacts
Wu et al. (2008) [47]	1	4
Li et al. (2011) [16]	1	1, 2
Wang et al. (2011) [48]	1	4

Study	Proportion of Contacts	Proportion of Non-contacts
DiLena et al. (2012) [49]	1	˜4
Wang et al. (2013) [50]	1	˜4

I followed the same strategy and undersampled residue pairs that are not physical contacts with a proportion of contacts to non-contacts of 1:5. The training set is comprised of 50.000 residue pairs $< 8\mathring{A}$ (“contacts”“) and 250.000 residue pairs $> 8\mathring{A}$ (”non-contacts“) so that each of the five cross-validation models will be trained on 40.000 contacts and 200.000 non-contacts. As the training set has been undersampled for non-contacts, it is not representative of real world proteins and the models should be validated on a more realistic validation set. Each of the five models is therefore cross-validated on an own independent dataset of residue pairs extracted from 40 proteins by means of the standard contact prediction benchmark (mean precision against top ranked contacts).

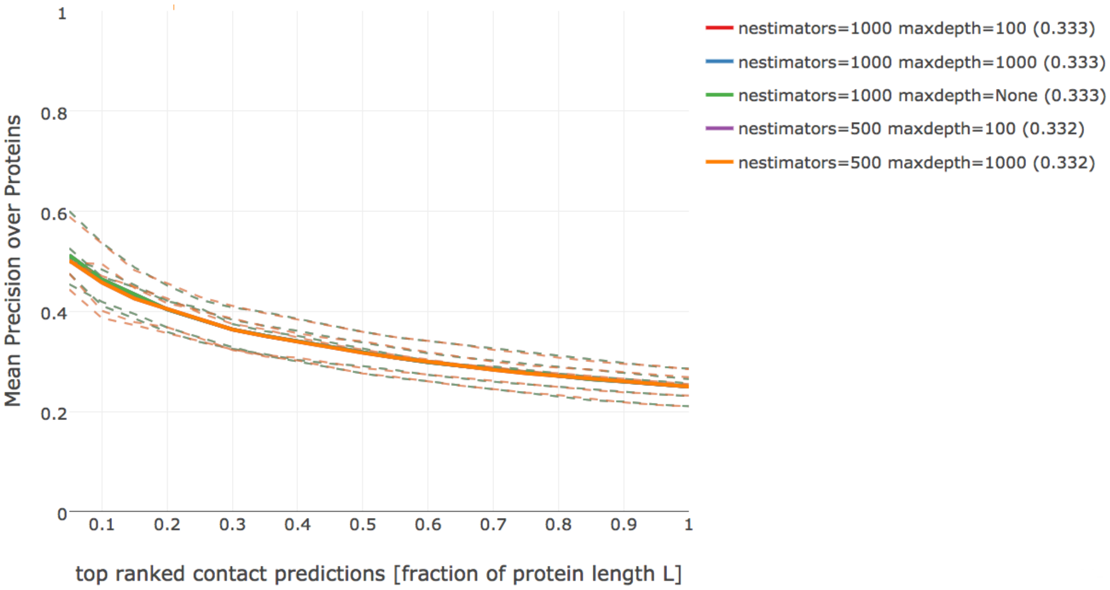

Figure 3.19: Mean precision over 200 proteins against highest scoring contact predictions from random forest models for different settings of *n_estimators* and *max_depth*. Dashed lines show the performance of models that have been learned on the five different subsets of training data. Solid lines give the mean precision over the five models. Only those models are shown that yielded the five highest mean precision values (given in parantheses in the legend). Random forest models with 1000 trees and maximum depth of trees of either 100, 1000 or unrestricted tree depth perform nearly identical (lines overlap). Random forest models with 500 trees and *max_depth*=10 or *max_depth*=100 perform slightly worse.

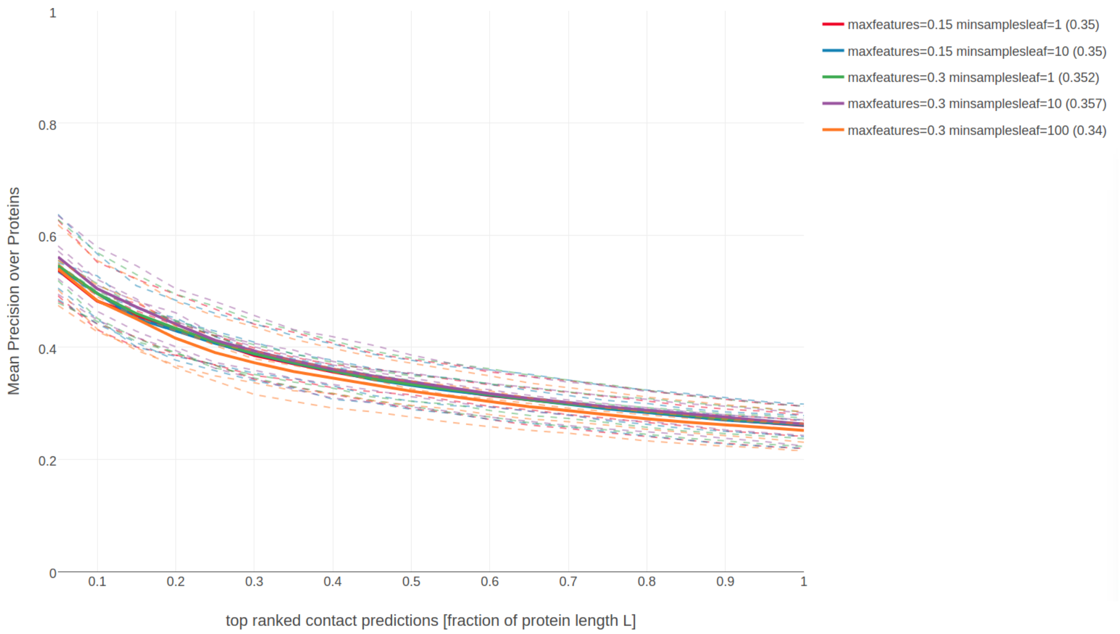

Figure 3.20: Mean precision over 200 proteins against highest scoring contact predictions from random forest models with different settings of *min_samples_leaf* and *max_features*. Dashed lines show the performance of models that have been learned on the five different subsets of training data. Solid lines give the mean precision over the five models. Only those models are shown that yielded the five best mean precision values (given in parantheses in the legend).

First I assessed performance of models for combinations of the parameter *n_estimators*, defining the number of trees in the forest and the parameter *max_depth* defining the maximum depth of the trees:

- *n_estimators* $\in \{100, 500, 1000\}$
- *max_depth* $\in \{10, 100, 1000, None\}$

Figure 3.19 shows that the top five parameter combinations perform nearly identical. Random forests with 1000 trees perform slightly better than models constituting 500 trees, irrespective of the depth of the trees. In order to keep model complexity small, `n_estimators=1000` and `max_depth=100` for further analysis.

Next, I optimized the parameters *min_samples_leaf*, defining the minimum number of samples required to be at a leaf node and *max_features*, defining the number of randomly selected features considered for each split using the following settings:

- *min_samples_leaf* $\in \{1, 10, 100\}$
- *max_features* $\in \{\text{sqrt}, \text{log2}, 0.15, 0.3\}$

Randomly selecting 39% of features (=75 features) and requiring at least 10 samples per leaf gives highest mean precision as can be seen in Figure 3.20. I chose `max_features = 0.30` and `min_samples_leaf = 10` for further analysis. Tuning the hyperparameters in a different order or on a larger dataset gives similar results.

In a next step I assessed dataset specific settings, such as the window size over which single positions features will be computed, the distance threshold to define non-contacts and the optimal proportions of contacts and non-contacts in the training set. I used the previously identified settings of random forest hyperparameters (`n_estimators=1000, min_samples_leaf=10, max_depth=100, max_features=0.30`).

- ratio of non-contacts/contacts $\in \{2, 5, 10, 20\}$ within a fixed total dataset size of 300 000 residue pairs
- window size: $\in \{5, 7, 9, 11\}$
- non-contact threshold $\in \{8, 15, 20\}$

As can be seen in appendix E.1 and E.2, the default choice of using a window size of five positions and the non-contact threshold of $8 \mathring{A}$ proves to be the optimal setting. Furthermore, using five-times as many non-contacts as contacts in the training set results in highest mean precision as can be seen in appendix E.3. These estimates might be biased in a way since the random forest hyperparameters have been optimized on a dataset using exactly these optimal settings.

3.13.3 Feature Selection

Many features obtain low *Gini importance* scores and can most likely be removed from the data set which will also reduce model complexity. It has been found, that prediction performance might even increase after removing the most irrelevant features [11]. For example, during the development of *EPSILON-CP*, a deep neural network method for contact prediction, the authors performed feature selection using boosted trees. By removing 75% of the most non-informative features (mostly features related to amino acid composition), the performance of their predictor increased slightly [5]. Other studies have also emphasized the importance of feature selection to improve performance and reduce model complexity [16,51].

I therefore developed a feature selection pipeline that retrains the random forest model on subsets of features. The subsets are composed of those features having *Gini importance* larger than the $\{10, 30, 50, 70, 90\}$-percentile of the distribution of *Gini importance* values obtained by training a model on all features. Performance of the models trained on these subsets of features is evaluated on a validation set.

3.13.4 Using Pseudo-likelihood Coevolution Score as Additional Feature

Besides the 250 sequence derived features, the pseudo-likelihood contact score (APC corrected Frobenius norm of couplings) is used as an additional feature. The random forest was trained on 100.000 residue pairs in physical contact ($\Delta C_\beta < 8 \mathring{A}$) and 500.000 residue pairs not in physical contact ($\Delta C_\beta > 8 \mathring{A}$) using the cross-validated hyperparameters as described earlier.

The pseudo-likelihood contact score comprises by far the most important feature as can be seen in the Figure 3.21. Other important features include the local

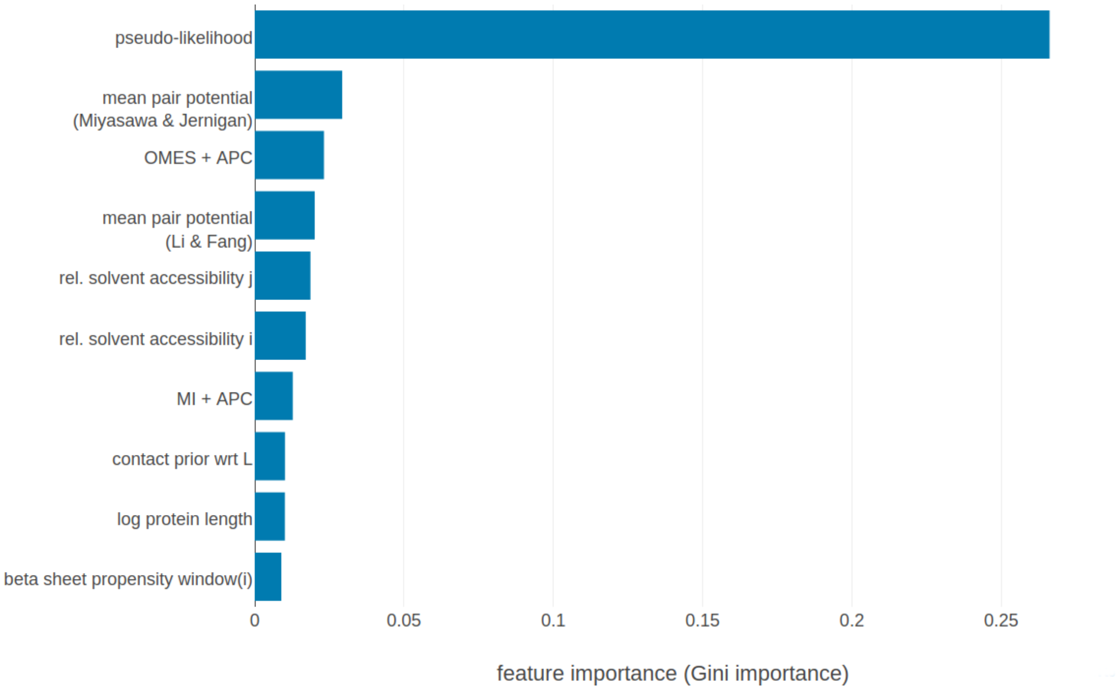

Figure 3.21: Top ten features ranked according to *Gini importance*. **pseudo-likelihood**: APC corrected Frobenius norm of couplings computed with pseudo-likelihood. **mean pair potential (Miyasawa & Jernigan)**: average quasi-chemical energy of transfer of amino acids from water to the protein environment [15]. **OMES+APC**: APC corrected OMES score according to Fodor&Aldrich [14]. **mean pair potential (Li&Fang)**: average general contact potential by Li & Fang [16]. **rel. solvent accessibilty i(j)**: RSA score computed with Netsurfp (v1.0) [17] for position i(j). **MI+APC**: APC corrected mutual information between amino acid counts (using pseudo-counts). **contact prior wrt L**: simple contact prior based on expected number of contacts wrt protein length (see methods section 3.13.1.4). **log protein length**: logarithm of protein length. **beta sheet propensity window(i)**: beta-sheet propensity according to Psipred [18] computed within a window of five positions around i. Features are described in detail in methods section 3.13.1.

statistical contact scores OMES and mutual information, the mean pairwise potentials according to Miyasawa & Jernigan [15] and Li & Fang [16], relative solvent accessibilty predictions (with NetsurfP [17]). The most important features apart from the pseudo-likelihood contact score, are the same features that are highly relevant for the basic random forest model (see Figure 2.2).

Models that have been trained on subsets of features, comprising 226, 176, 126 or 76 of the most important features with respect to *Gini importance*, perform equally well as the model trained on the complete set of features (see Figure 3.22). Only the model trained on the 26 most important features has slighlty decreased precision for the top L/10 ranked contacts.

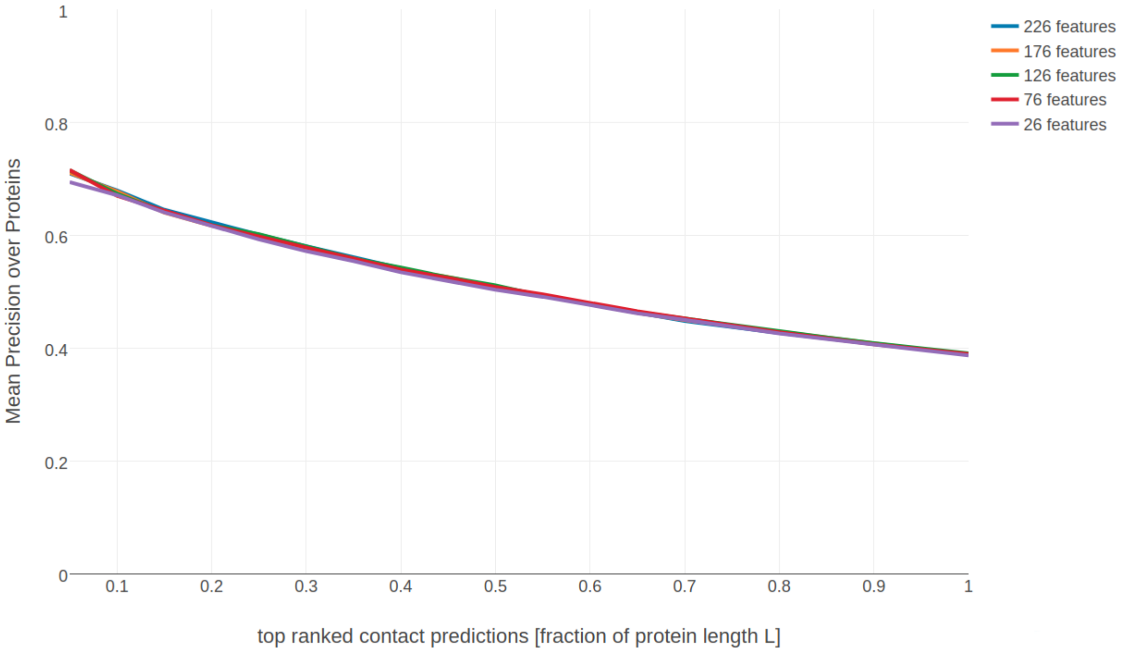

Figure 3.22: Mean precision for top ranked contacts over 200 proteins for variaous random forest models trained on subsets of features. Subsets of features have been selected as described in section 3.13.3.

A

Abbreviations

APC Avarage Product Correction

CASP Critical Assessment of protein Structure Prediction

CD Contrastive Divergence

DCA Direct Coupling Analysis

DI Direct Information

EM electron microscopy

IDP intrinsically disordered proteins

MAP Maximum a posteriori

MCMC Markov Chain Monte Carlo

MI mutual information

ML Maximum-Likelihood

MLE Maximum-Likelihood Estimate

MRF Markov-Random Field

MSA Multiple Sequence Alignment

Neff Number of effective sequences

PCD Persistent Contrastive Divergence

PDB protein data bank

SGD stochastic gradient descent

A.1 Amino Acid Alphabet

One-letter Code	Three-letter Code	Amino Acid	Physico-chemical properties
A	Ala	**A**lanine	tiny, hydrophobic
C	Cys	**C**ysteine	small, hydrophobic, polar (C_{S-H})
D	Asp	Aspartic Aci**D**	small, negatively charged, polar
E	Glu	Glutamic Acid	negatively charged, polar
F	Phe	Phenylalanine	aromatic, hydrophobic
G	Gly	**G**lycine	tiny, hydrophobic
H	His	**H**istidine	hydrophobic, aromatic, polar, (positivel
I	Ile	**I**soleucine	aliphatic, hydrophobic
K	Lys	Lysine	positively charged, polar
L	Leu	**L**eucine	aliphatic, hydrophobic
M	Met	**M**ethionine	hydrophobic
N	Asn	Asparagi**N**e	small, polar
P	Pro	**P**roline	small
Q	Gln	Glutamine	tiny, hydrophobic
R	Arg	A**R**ginine	positively charged, polar
S	Ser	**S**erine	tiny, polar
T	Thr	**T**hreonine	hydrophobic, polar
V	Val	**V**aline	small, aliphatic
W	Trp	**T**ryptophan	aromatic, hydrophobic, polar
Y	Tyr	T**Y**rosine	aromatic, hydrophobic, polar

B
Dataset Properties

The following figures display various statistics about the dataset used throughout this thesis. See section 3.1 for information on how this dataset has been generated.

B.1 Alignment Diversity

B.2 Proportion of Gaps in Alignment

B.3 Alignment Size (number of sequences)

B.4 Protein Length

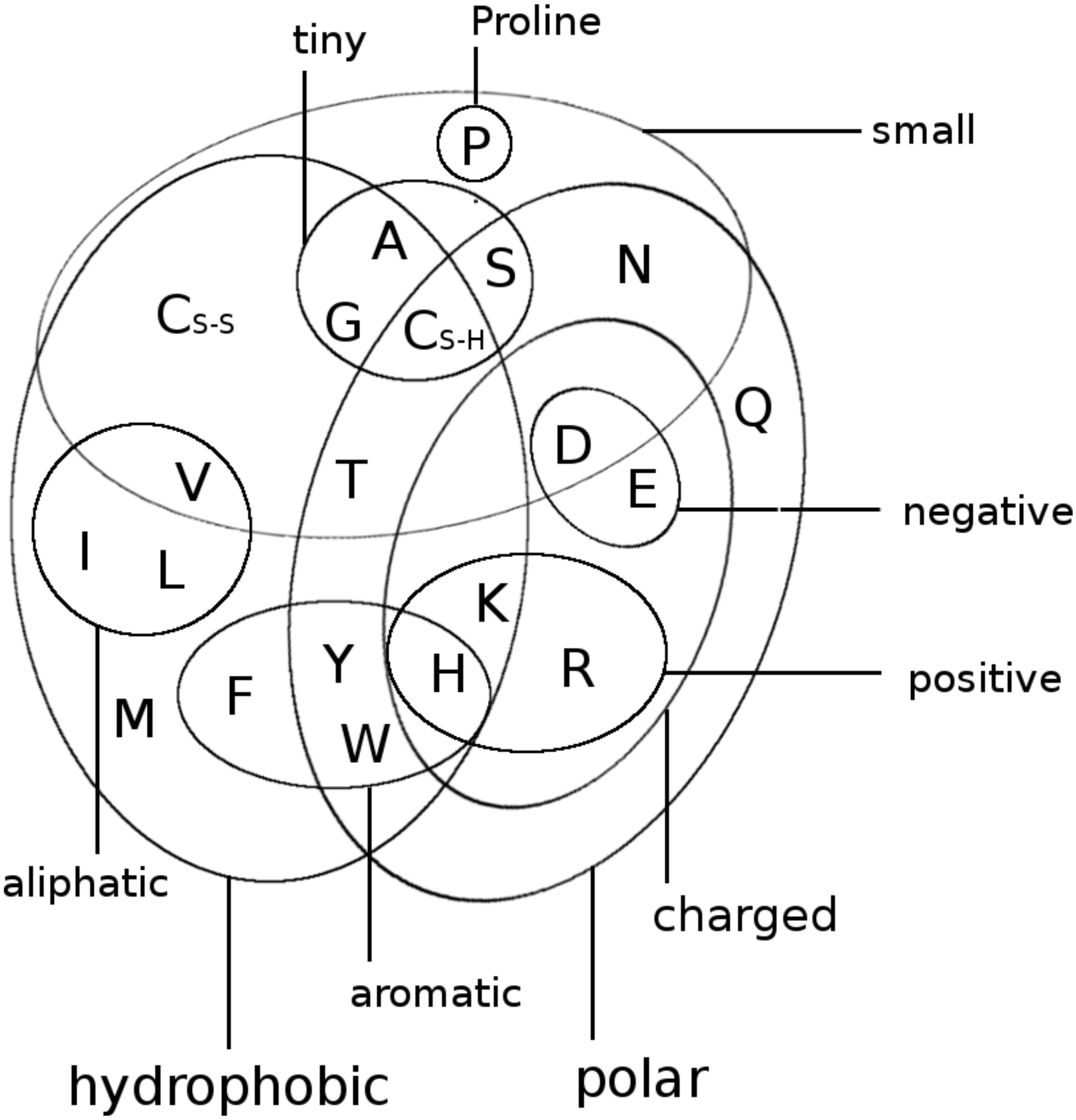

Figure B.1: Distribution of alignment diversity ($= \sqrt{(\frac{N}{L})}$) in the dataset an its ten subsets.

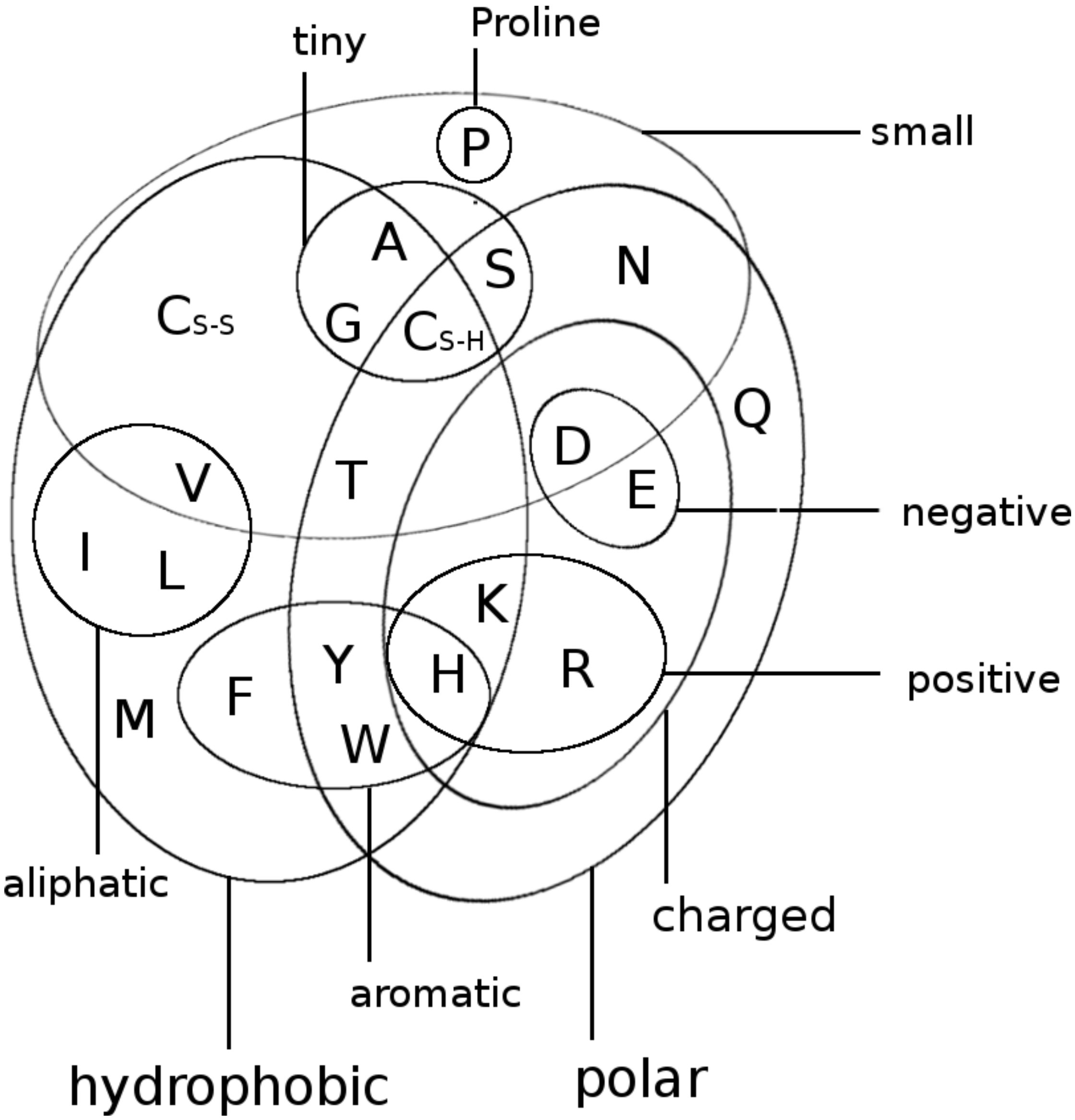

Figure B.2: Distribution of gap percentage of alignments in the dataset an its ten subsets.

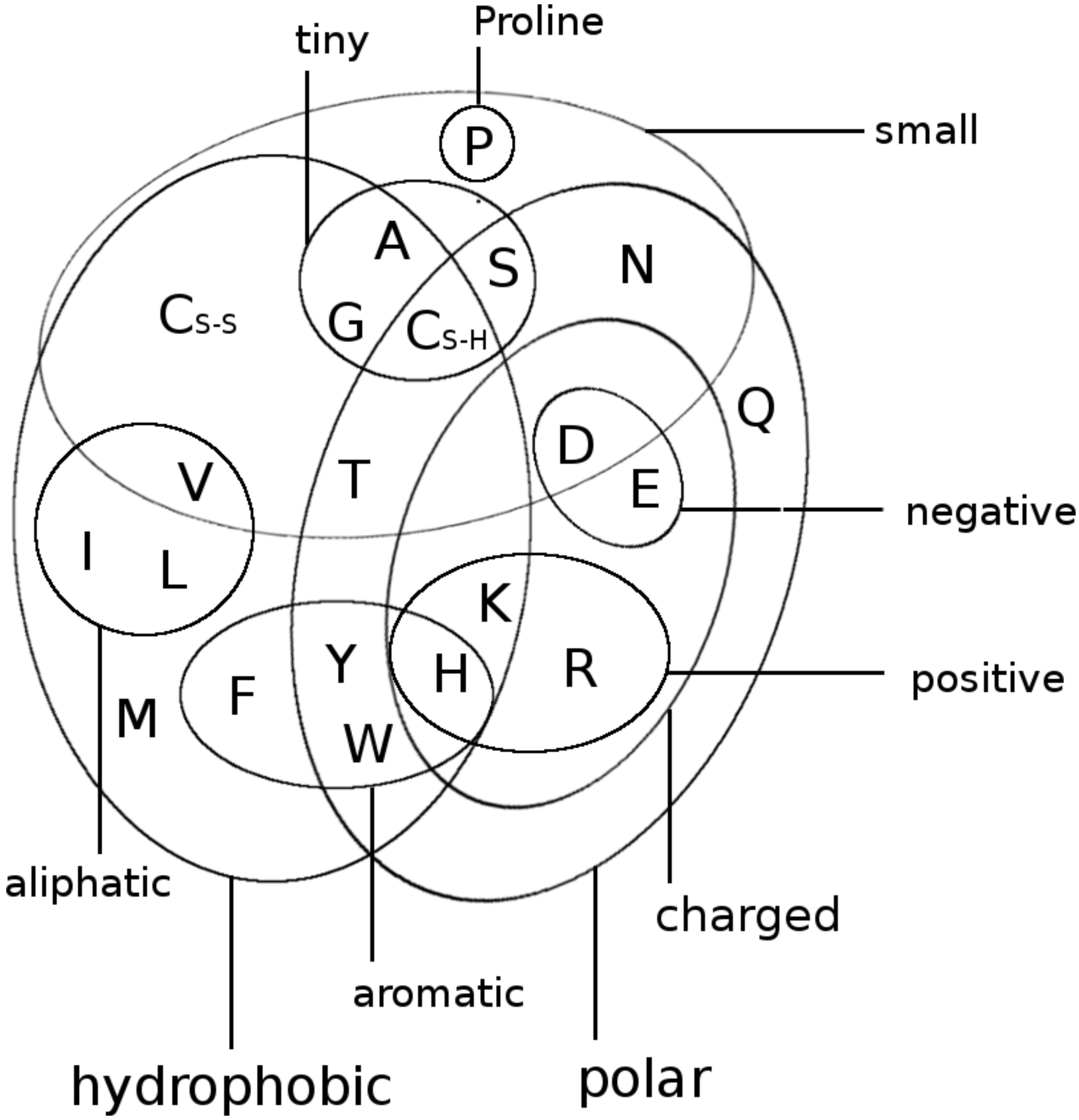

Figure B.3: Distribution of alignment size (number of sequences N) in the dataset an its ten subsets.

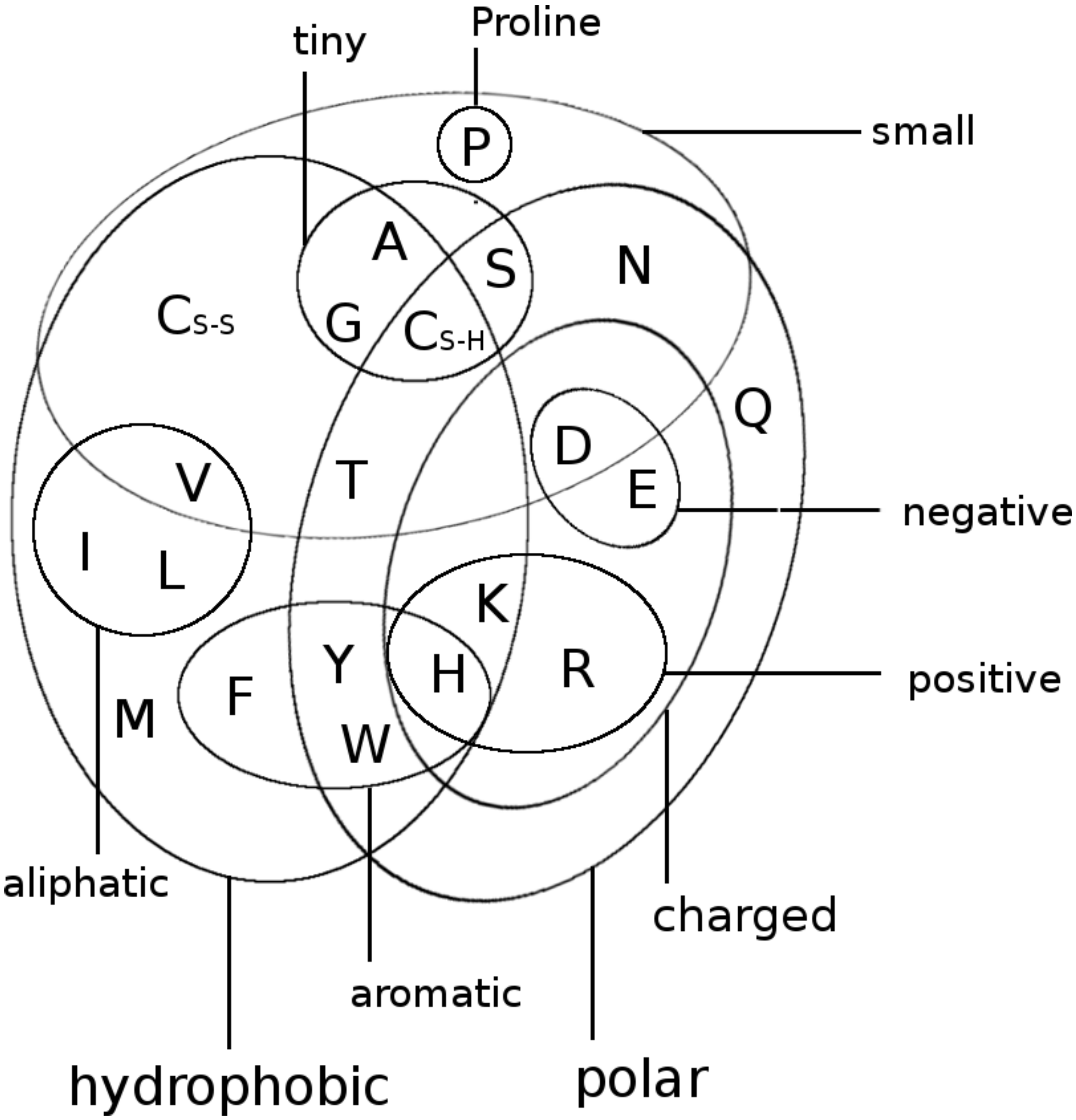

Figure B.4: Distribution of protein length L in the dataset an its ten subsets.

C

Amino Acid Interaction Preferences Reflected in Coupling Matrices

C.1 Pi-Cation interactions

Figure C.1 shows a Tyrosine and a Lysine residue forming a cation-π interaction in protein 2ayd. The corresponding coupling matrix in figure C.2 reflects the strong interaction preference.

C.2 Disulfide Bonds

Figure C.3 shows two cysteine residues forming a covalent disulfide bond in protein 1alu. The corresponding coupling matrix in figure C.4 reflects the strong interaction preference of cysteines.

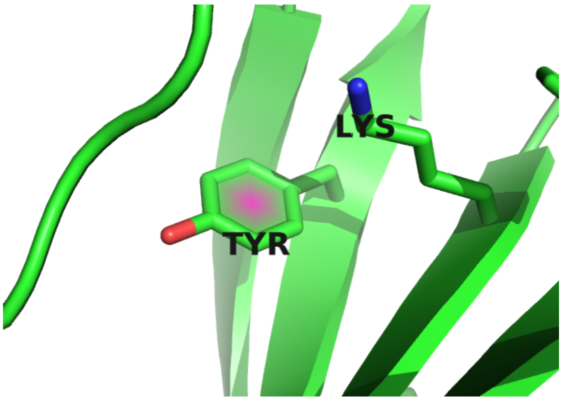

Figure C.1: Tyrosing (residue 37) and Lysine (residue 48) forming a cation-π interaction in protein 2ayd.

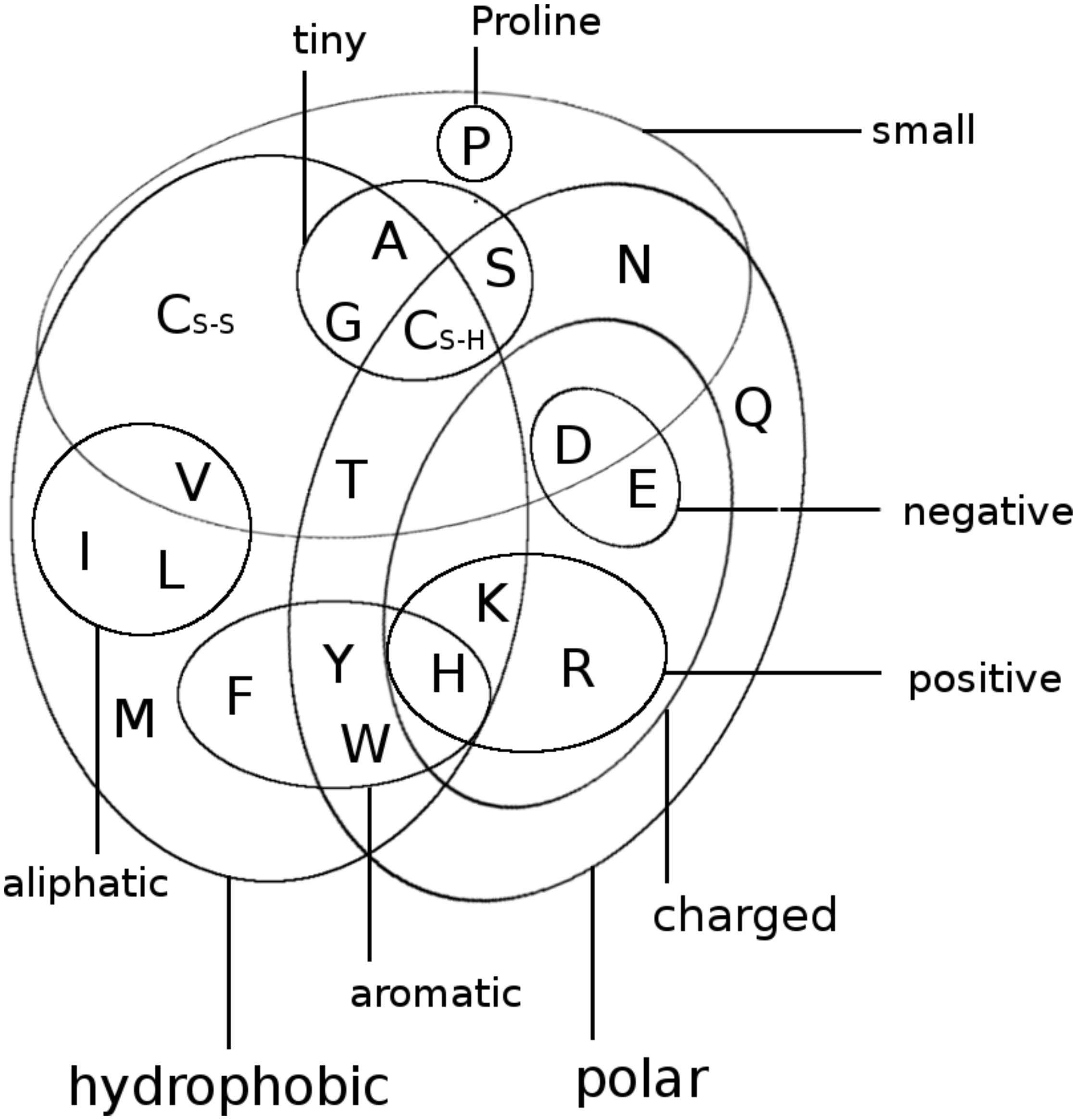

Figure C.2: Coupling Matrix for residue pair i=37 and j=48 of PDB 2ayd chain A domain 1. Size of the bubbles represents coupling strength and color represents the direction of coupling: red = positive coupling, blue = negative coupling. Bars at the x-axis represent single potentials for residue i=37 and bars at the y-axis represent single potentials for residue j=48. Height of the bars represents potential strength and color represents positive (red) and negative (blue) values.

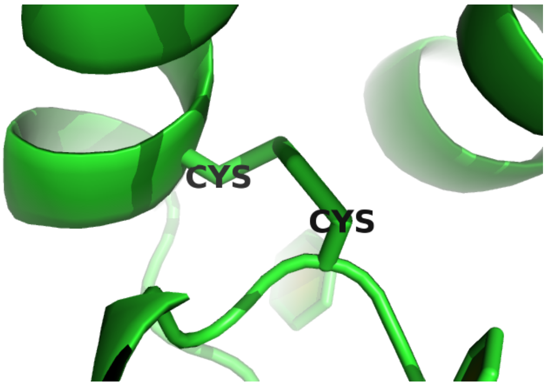

Figure C.3: Two cystein residues (residues 54 and 64) forming a covalent disulfide bond in protein 1alu.

C.3 Aromatic-Proline Interactions

Figure @ref(fig:coupling-matrix-aromatic-proline-pymol)shows a proline and a tryptophan residue forming such a CH/π interaction in protein 1aol. The corresponding coupling matrix in figure C.6 reflects this interaction with strong positive coupling between proline and tryptophan.

C.4 Network-like structure of aromatic residues

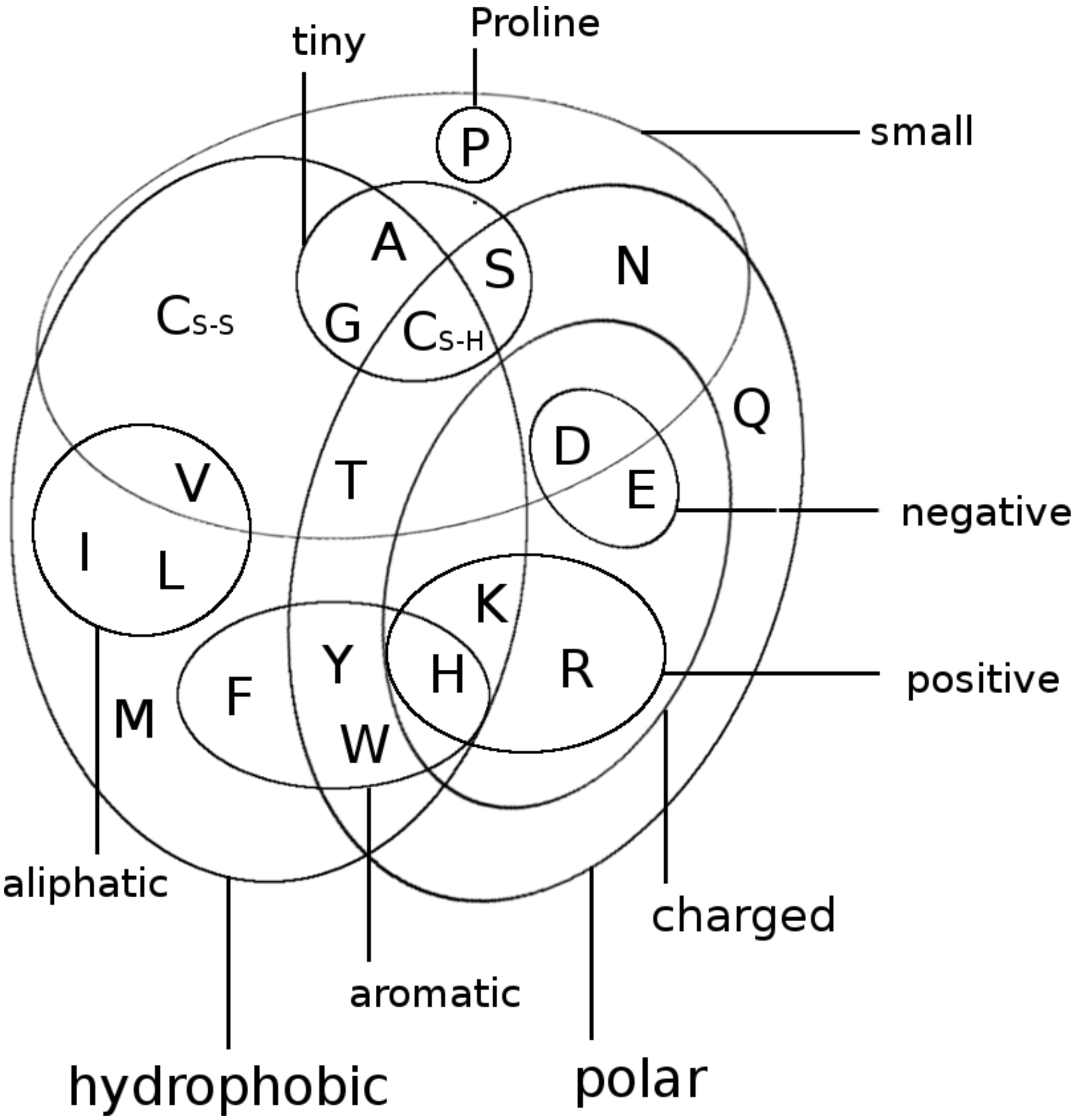

Figure C.4: Coupling Matrix for residue pair i=54 and j=64 of PDB 1alu chain A. Size of the bubbles represents coupling strength and color represents the direction of coupling: red = positive coupling, blue = negative coupling. Bars at the x-axis represent single potentials for residue i=54 and bars at the y-axis represent single potentials for residue j=64. Height of the bars represents potential strength and color represents positive (red) and negative (blue) values.

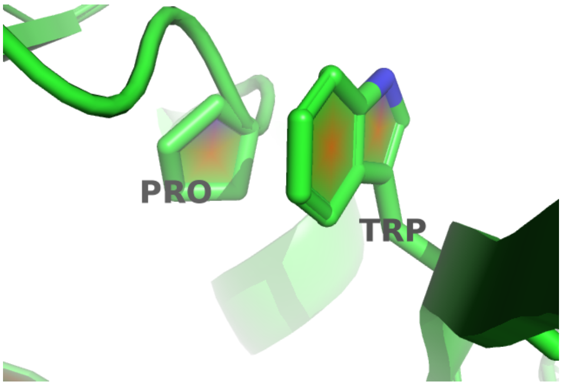

Figure C.5: Proline and tryptophan (residues 17 and 34) stacked on top of each otherengaging in a CH/π interaction in protein 1alu.

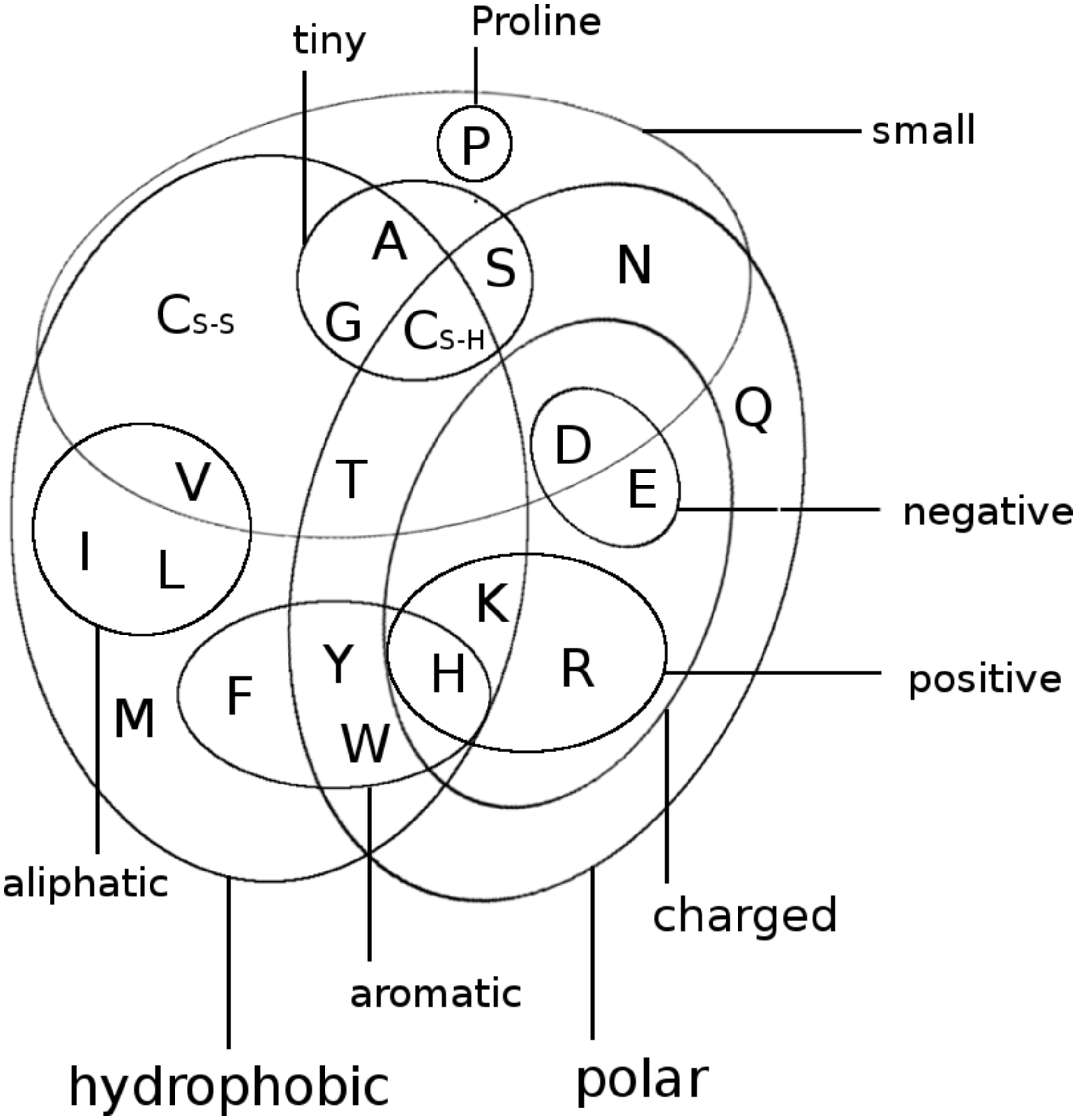

Figure C.6: Coupling Matrix for residue pair i=17 and j=34 of PDB 1aol chain A. Size of the bubbles represents coupling strength and color represents the direction of coupling: red = positive coupling, blue = negative coupling. Bars at the x-axis represent single potentials for residue i=17 and bars at the y-axis represent single potentials for residue j=34. Height of the bars represents potential strength and color represents positive (red) and negative (blue) values.

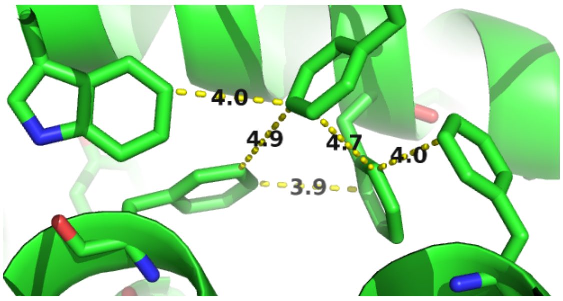

Figure C.7: Network-like structure of aromatic residues in the protein core. 80% of aromatic residues are involved in such networks that are important for protein stability [2].

D

Optimizing Full Likelihood with Gradient Descent

D.1 Visualisation of learning rate schedules

D.2 Benchmarking learning rate schedules

D.2.1 Linear learning rate schedule

D.2.2 Sigmoidal learning rate schedule

D.2.3 Square root learning rate schedule

D.2.4 Exponential learning rate schedule

D.3 Number of iterations until convergence for different learning rate schedules

D.3.1 Linear learning rate schedule

(ref:caption-full-likelihood-opt-numit-lin-learning-rate-schedule) Distribution of the number of iterations until convergence for gradient descent optimizations of the full likelihood using different decay rates with a **linear** learning rate schedule $\alpha = \alpha_0/(1 + \gamma \cdot t)$ with t being the iteration number and the decay rate γ as specified in the legend. Initial learning rate α_0 defined with respect to Neff and maximum number of iterations is set to 5000.

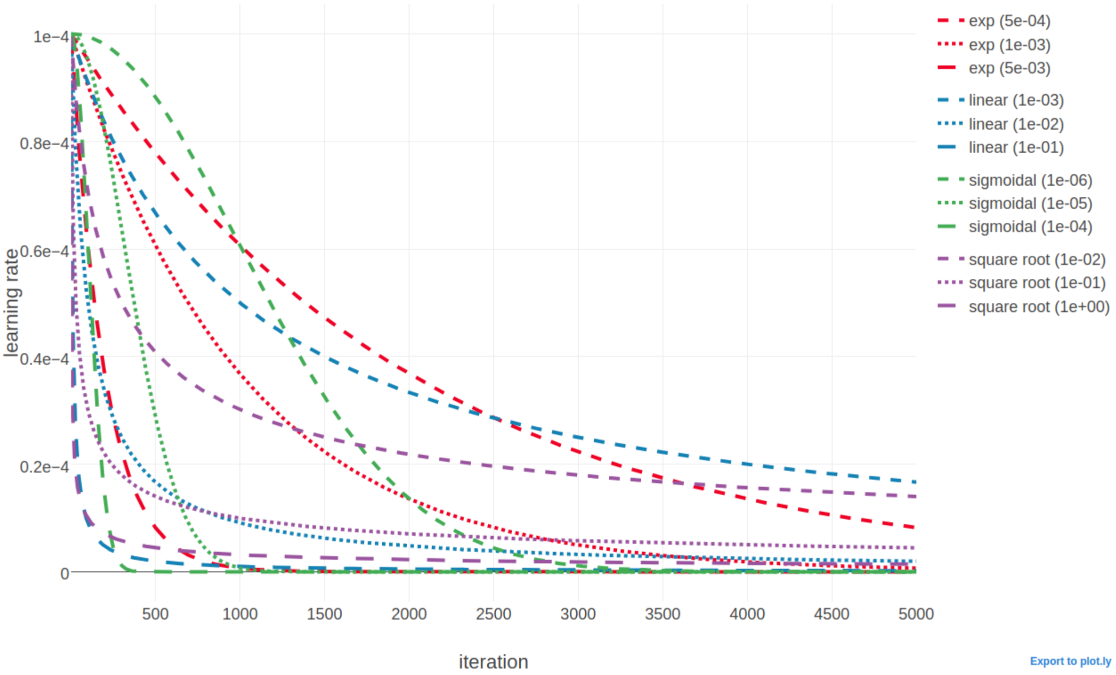

Figure D.1: Value of learning rate against the number of iterations for different learning rate schedules. Red legend group represents the **exponential** learning rate schedule $\alpha_{t+1} = \alpha_0 \cdot \exp(-\gamma t)$. Blue legend group represents the **linear** learning rate schedule $\alpha = \alpha_0/(1 + \gamma \cdot t)$. Green legend group represents the **sigmoidal** learning rate schedule $\alpha_{t+1} = \alpha_t/(1 + \gamma \cdot t)$ with γ. Purple legend group represents the **square root** learning rate schedule $\alpha = \alpha_0/\sqrt{1 + \gamma \cdot t}$. The iteration number is given by t. Initial learning rate α_0 is set to 1e-4 and γ is the decay rate and its value is given in brackets in the legend.

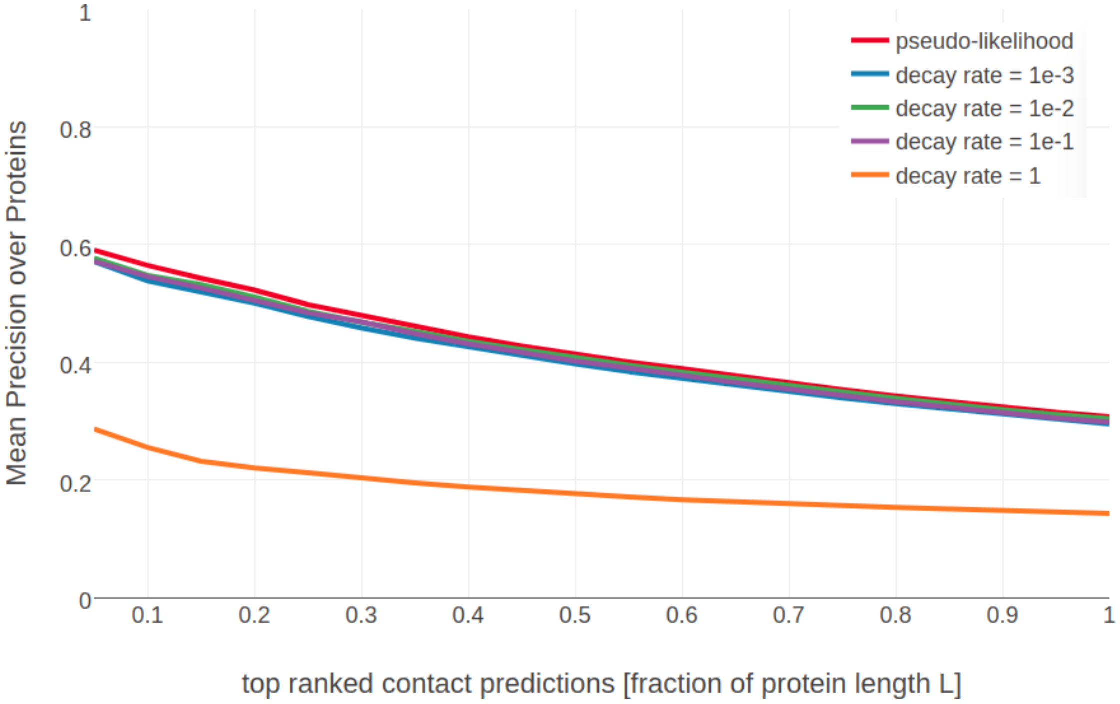

Figure D.2: Mean precision for top ranked contact predictions over 288 proteins. Contact scores are computed as the APC corrected Frobenius norm of the couplings $\mathbf{w}_{ij}$. pseudo-likelihood: Contact scores computed from pseudo-likelihood. The other methods derive contact scores from couplings computed from CD using stochastic gradient descent with an initial learning rate defined with respect to Neff and a *linear* learning rate annealing schedule $\alpha = \frac{\alpha_0}{1+\gamma t}$ with decay rate γ as specified in the legend.

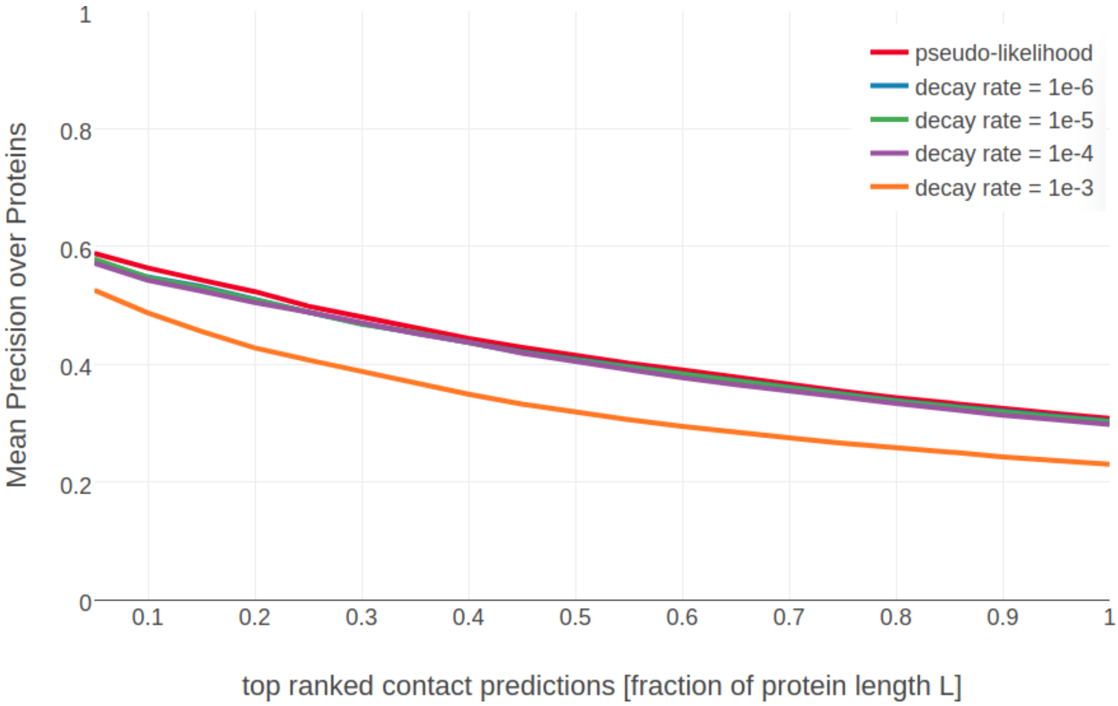

Figure D.3: Mean precision for top ranked contact predictions over 288 proteins. Contact scores are computed as the APC corrected Frobenius norm of the couplings $\mathbf{w}_{ij}$. pseudo-likelihood: Contact scores computed from pseudo-likelihood. The other methods derive contact scores from couplings computed from CD using stochastic gradient descent with an initial learning rate defined with respect to Neff and a *sigmoidal* learning rate annealing schedule $\alpha_{t+1} = \frac{\alpha_t}{1+\gamma t}$ with t being the iteration number and decay rate γ as specified in the legend.

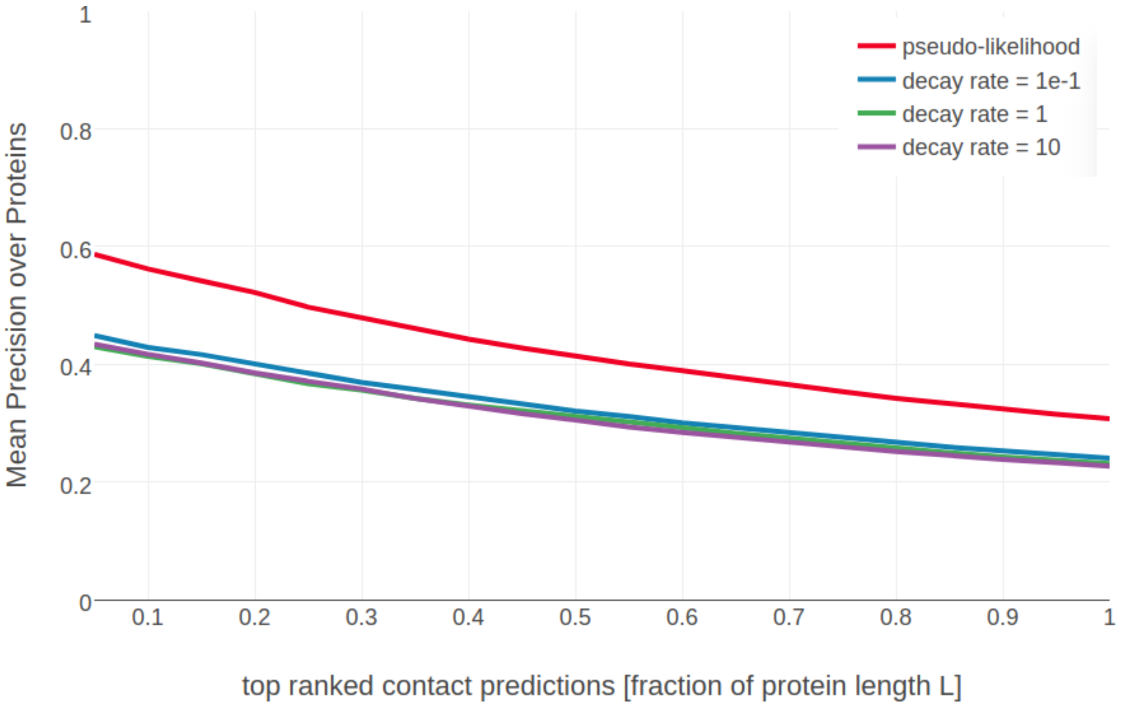

Figure D.4: Mean precision for top ranked contact predictions over 288 proteins. Contact scores are computed as the APC corrected Frobenius norm of the couplings $\mathbf{w}_{ij}$. pseudo-likelihood: Contact scores computed from pseudo-likelihood. The other methods derive contact scores from couplings computed from CD using stochastic gradient descent with an initial learning rate defined with respect to Neff and a *square root* learning rate annealing schedule $\alpha = \frac{\alpha_0}{\sqrt{1+\gamma t}}$ with t being the iteration number and decay rate γ as specified in the legend.

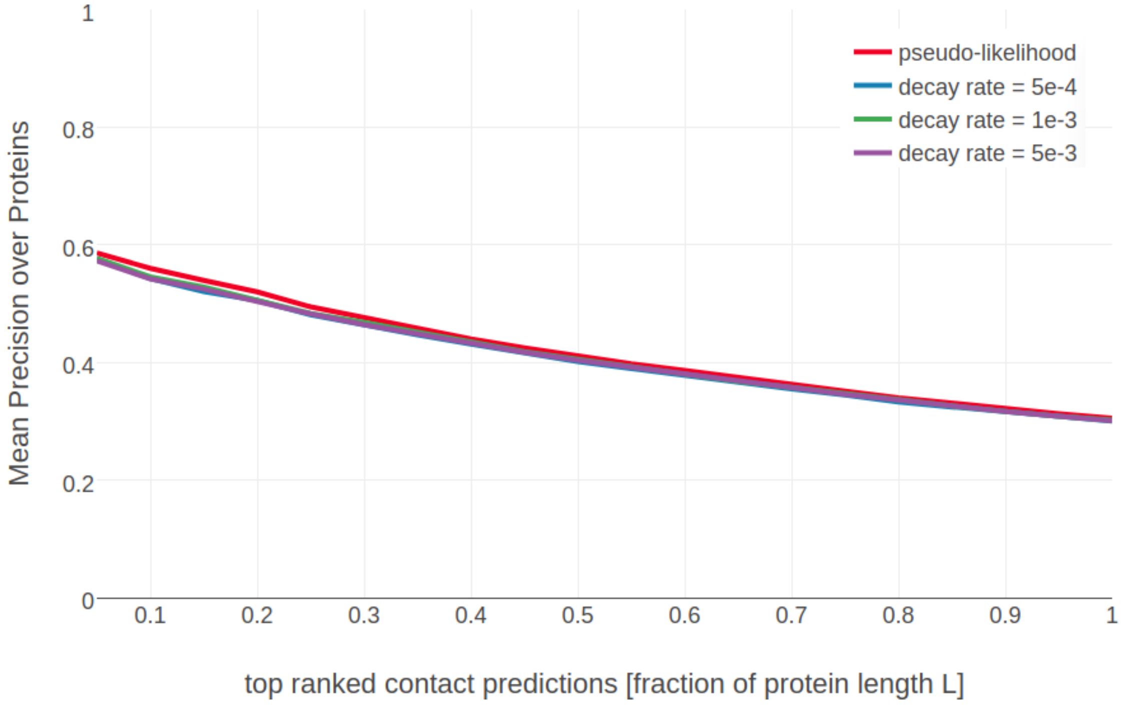

Figure D.5: Mean precision for top ranked contact predictions over 288 proteins. Contact scores are computed as the APC corrected Frobenius norm of the couplings $\mathbf{w}_{ij}$. pseudo-likelihood: Contact scores computed from pseudo-likelihood. The other methods derive contact scores from couplings computed from CD using stochastic gradient descent with an initial learning rate defined with respect to Neff and a *exponential* learning rate annealing schedule $\alpha = \alpha_0 \cdot \exp(-\gamma t)$ with t being the iteration number and decay rate γ as specified in the legend.

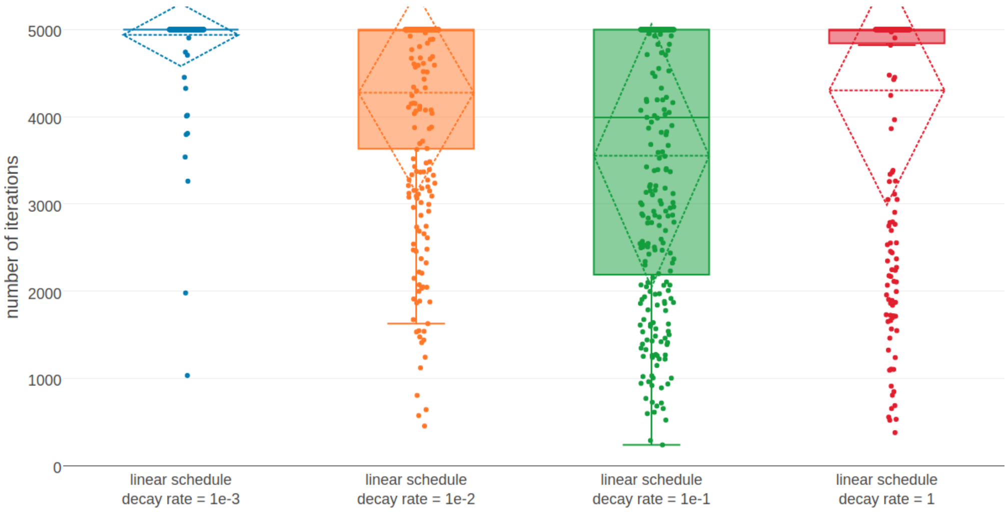

Figure D.6: (ref:caption-full-likelihood-opt-numit-lin-learning-rate-schedule)

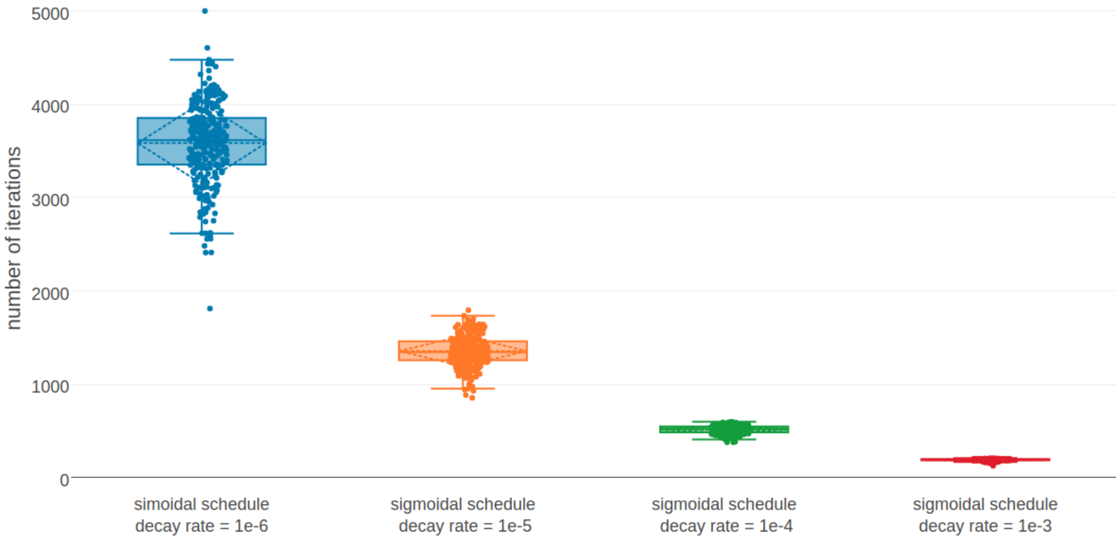

Figure D.7: Distribution of the number of iterations until convergence for gradient descent optimizations of the full likelihood using different decay rates with a **sigmoidal** learning rate schedule $\alpha_{t+1} = \alpha_t/(1 + \gamma \cdot t)$ with t being the iteration number and the decay rate γ as specified in the legend. Initial learning rate α_0 defined with respect to Neff and maximum number of iterations is set to 5000.

D.3.2 Sigmoidal learning rate schedule

D.3.3 Square root learning rate schedule

(ref:caption-full-likelihood-opt-numit-sqrt-learning-rate-schedule) Distribution of the number of iterations until convergence for gradient descent optimizations of the full likelihood using different decay rates with a **square root** learning rate schedule $\alpha = \alpha_0/\sqrt{1 + \gamma t}$ with t being the iteration number and the decay rate γ as specified in the legend. Initial learning rate α_0 defined with respect to Neff and maximum number of iterations is set to 5000.

D.3.4 Exponential learning rate schedule

(ref:caption-full-likelihood-opt-numit-exp-learning-rate-schedule) Distribution of the number of iterations until convergence for gradient descent optimizations of the full likelihood using different decay rates with an **exponential** learning rate schedule $\alpha = \alpha_0 \cdot \exp(-\gamma t)$ with t being the iteration number and the decay rate γ as specified in the legend. Initial learning rate α_0 defined with respect to Neff and maximum number of iterations is set to 5000.

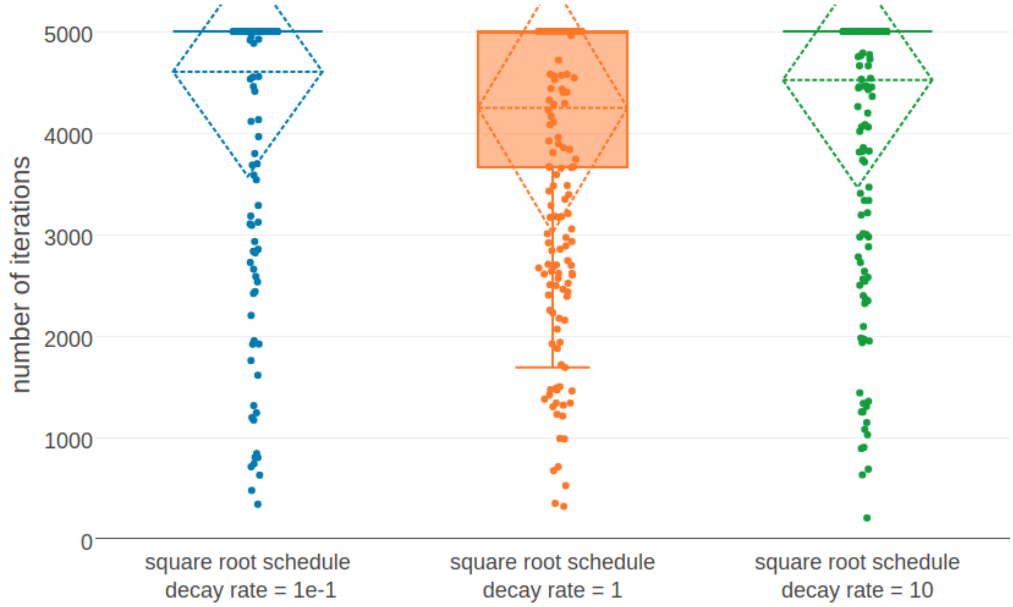

Figure D.8: (ref:caption-full-likelihood-opt-numit-sqrt-learning-rate-schedule)

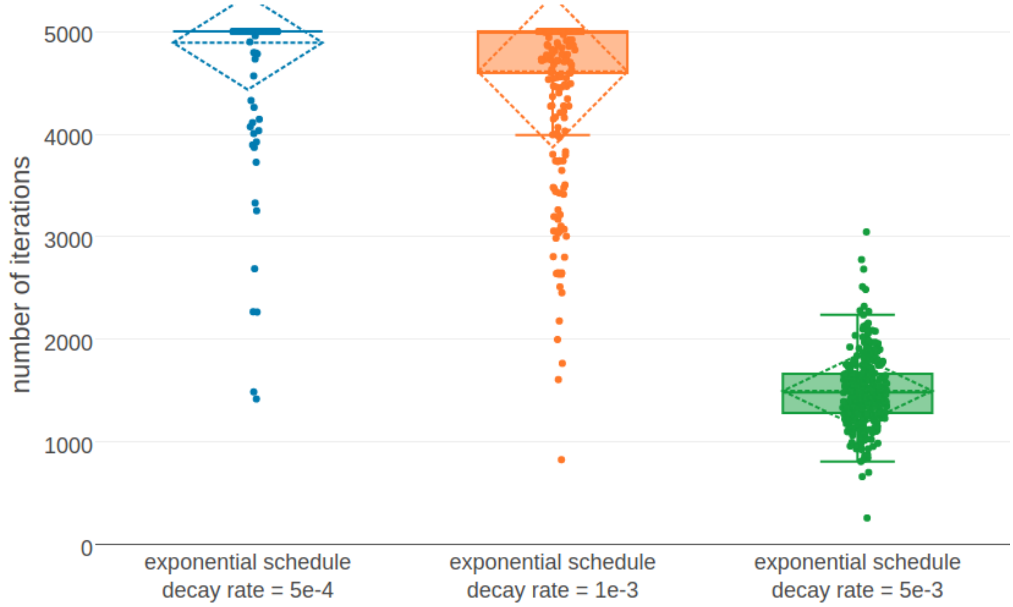

Figure D.9: (ref:caption-full-likelihood-opt-numit-exp-learning-rate-schedule)

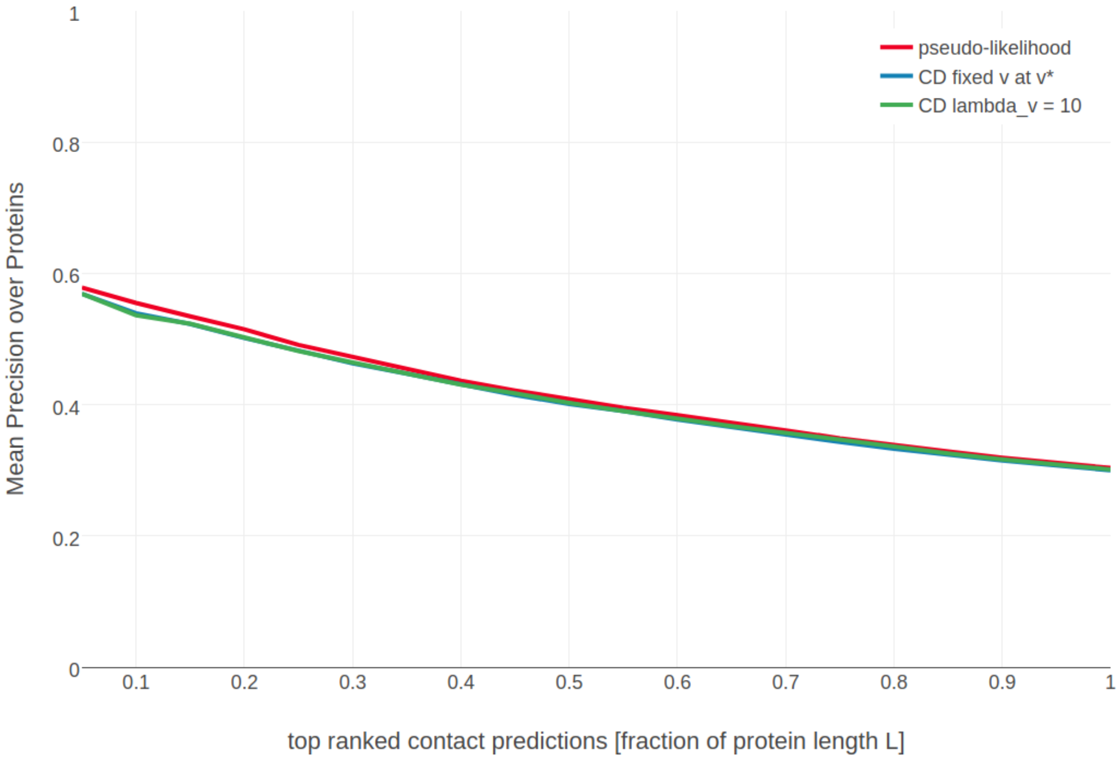

Figure D.10: Mean precision for top ranked contact predictions over 300 proteins. Contact scores are computed as the APC corrected Frobenius norm of the couplings $\mathbf{w}_{ij}$. SGD settings for CD optimization are as follows: sigmoidal learning rate schedule with decay rate $\gamma = 5\text{e}-6$ and initial learning rate $\alpha_0 = 5\text{e}-2/N_{\text{eff}}$. **pseudo-likelihood**: contact scores computed from pseudo-likelihood. **CD fixed v at v**: contact scores computed from CD with SGD and single potentials $\mathbf{v}$ are not optimized but fixed at $\mathbf{v}^*$ as given in eq. (3.43). **CD lambda_v = 10**: contact scores computed from CD with SGD and single potentials $\mathbf{v}$ are subject to optimization using L2-reglarization with $\lambda_v = 10$.

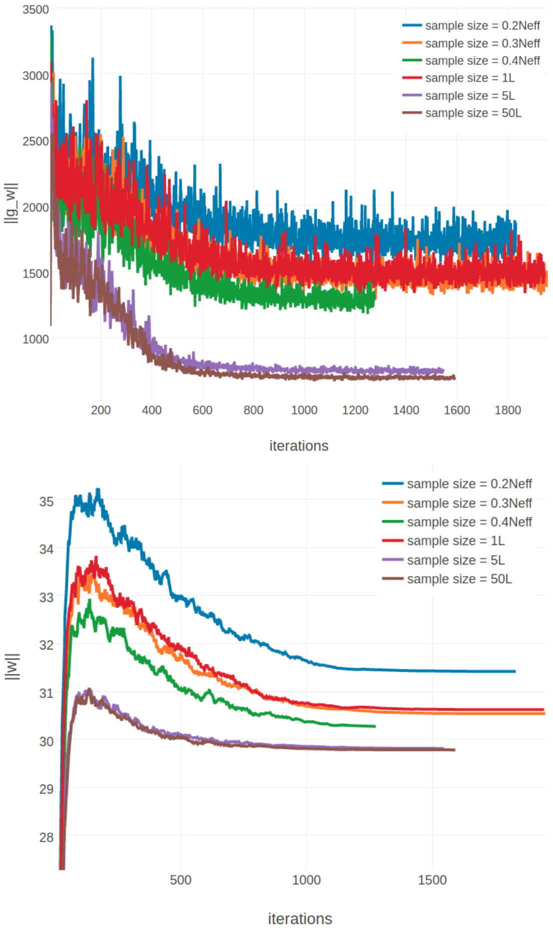

Figure D.11: Monitoring parameter norm and gradient norm for protein 1aho_A_00 during SGD using different sample sizes. Protein 1c75_A1aho_A_00_00 has length L=64 and 378 sequences in the alignment (Neff=229) **Left** L2-norm of the gradients for coupling parameters $||\mathbf{w}||_2$ (without contribution of regularizer). The number of sequences, that is used for Gibbs sampling to approximate the gradient, is given in the legend. **Right** L2-norm of the coupling parameters $||\mathbf{w}||_2$. The number of sequences, that is used for Gibbs sampling to approximate the gradient, is given in the legend.

D.4 Fix single potentials at maximum-likelihood estimate v*

D.5 Monitoring Optimization for different Sample Sizes

E

Training of the Random Forest Contact Prior

E.1 Evaluating window size with 5-fold Cross-validation

E.2 Evaluating non-contact threshold with 5-fold Cross-validation

E.3 Evaluating ratio of non-contacts and contacts in the training set with 5-fold Cross-validation

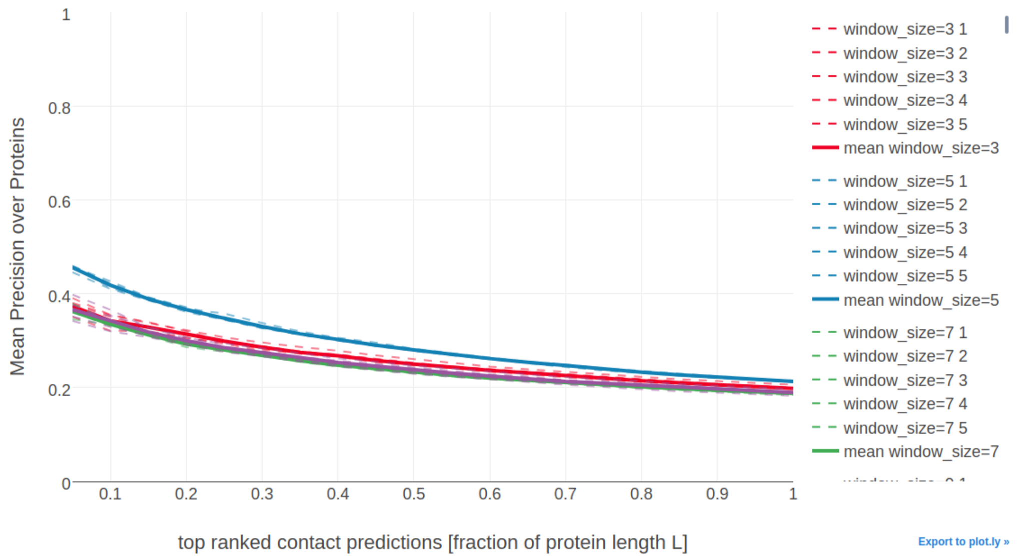

Figure E.1: Mean precision over validation set of 200 proteins for top ranked contact predictions for different choices of window size for single position features. Dashed lines represent the models trained on four subsets of the training data according to the 5-fold cross-validation scheme. Solid lines represent the mean over the five cross-validation models.

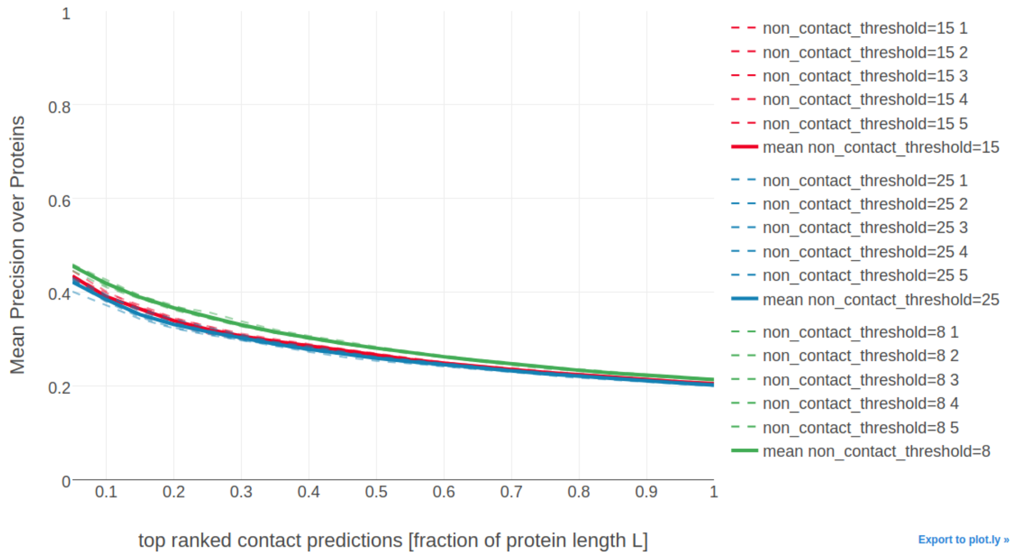

Figure E.2: Mean precision over validation set of 200 proteins for top ranked contact predictions for different choices of the non-contact threshold to define non-contacts. Dashed lines represent the models trained on four subsets of the training data according to the 5-fold cross-validation scheme. Solid lines represent the mean over the five cross-validation models.

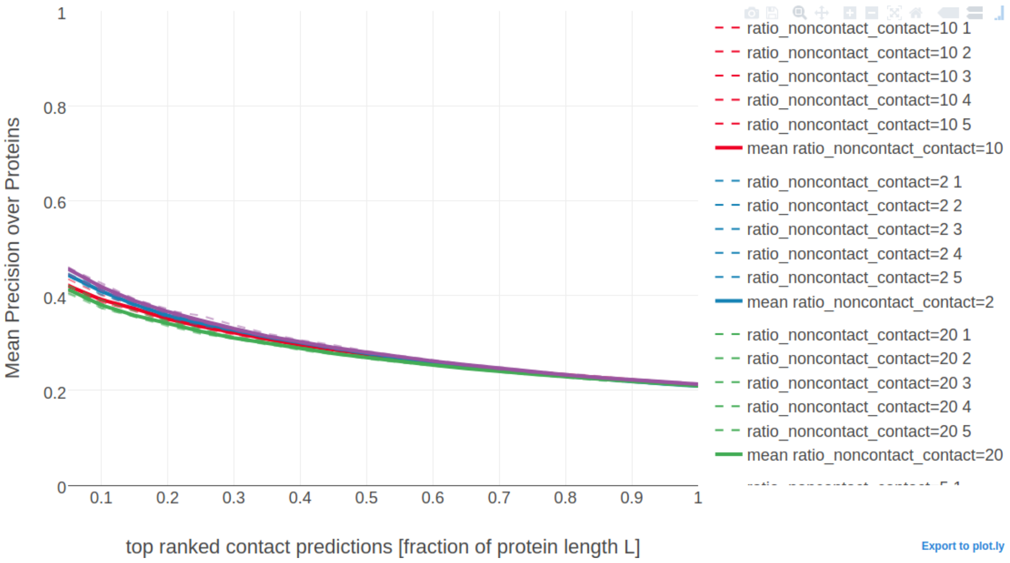

Figure E.3: Mean precision over validation set of 200 proteins for top ranked contact predictions for different choices of dataset composition with respect to the ratio of contacts and non-contacts. Dashed lines represent the models trained on four subsets of the training data according to the 5-fold cross-validation scheme. Solid lines represent the mean over the five cross-validation models.

List of Figures

List of Tables

References

1. Coucke, A., Uguzzoni, G., Oteri, F., Cocco, S., Monasson, R., and Weigt, M. (2016). Direct coevolutionary couplings reflect biophysical residue interactions in proteins. J. Chem. Phys. *145*, 174102. Available at: http://scitation.aip.org/content/aip/journal/jcp/145/17/10.1063/1.4966156.

2. Burley, S., and Petsko, G. (1985). Aromatic-aromatic interaction: a mechanism of protein structure stabilization. Science (80-.). *229*, 23–28. Available at: http://www.sciencemag.org/cgi/doi/10.1126/science.3892686.

3. Jones, D.T., Singh, T., Kosciolek, T., and Tetchner, S. (2015). MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. Bioinformatics *31*, 999–1006. Available at: http://bioinformatics.oxfordjournals.org/content/31/7/999.short.

4. He, B., Mortuza, S.M., Wang, Y., Shen, H.-B., and Zhang, Y. (2017). NeBcon: Protein contact map prediction using neural network training coupled with nave Bayes classifiers. Bioinformatics. Available at: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btx164.

5. Stahl, K., Schneider, M., and Brock, O. (2017). EPSILON-CP: using deep learning to combine information from multiple sources for protein contact prediction. BMC Bioinformatics *18*, 303. Available at: http://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-1713-x.

6. Skwark, M.J., Michel, M., Menendez Hurtado, D., Ekeberg, M., and Elofsson, A. (2016). Accurate contact predictions for thousands of protein families using PconsC3. bioRxiv.

7. Ma, J., Wang, S., Wang, Z., and Xu, J. (2015). Protein contact prediction by integrating joint evolutionary coupling analysis and supervised learning. Bioinformatics, btv472. Available at: http://bioinformatics.oxfordjournals.org/content/early/2015/09/04/bioinformatics.btv472.

8. Ho, T.K. (1998). The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. *20*, 832–844. Available at: http://ieeexplore.ieee.org/document/709601/.

9. Tin Kam Ho (1995). Random decision forests. In Proc. 3rd int. conf. doc. anal. recognit. (IEEE Comput. Soc. Press), pp. 278–282. Available at: http://ieeexplore.ieee.org/document/598994/.

10. Breiman, L. (2001). Random Forests. Mach. Learn. *45*, 5–32. Available at: http://link.springer.com/10.1023/A:1010933404324.

11. Menze, B.H., Kelm, B.M., Masuch, R., Himmelreich, U., Bachert, P.,

Petrich, W., and Hamprecht, F.A. (2009). A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. BMC Bioinformatics *10*, 213. Available at: http://www.ncbi.nlm.nih.gov/pubmed/19591666 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2724423.

12. Louppe, G. (2014). Understanding Random Forests: From Theory to Practice. Available at: http://arxiv.org/abs/1407.7502.

13. Strobl, C., Boulesteix, A.-L., Zeileis, A., and Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics *8*, 25. Available at: http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-25.

14. Fodor, A.A., and Aldrich, R.W. (2004). Influence of conservation on calculations of amino acid covariance in multiple sequence alignments. Proteins *56*, 211–21. Available at: http://www.ncbi.nlm.nih.gov/pubmed/15211506.

15. Miyazawa, S., and Jernigan, R.L. (1999). Self-consistent estimation of inter-residue protein contact energies based on an equilibrium mixture approximation of residues. Proteins *34*, 49–68. Available at: http://www.ncbi.nlm.nih.gov/pubmed/10336383.

16. Li, Y., Fang, Y., and Fang, J. (2011). Predicting residue-residue contacts using random forest models. Bioinformatics *27*, 3379–84. Available at: http://bioinformatics.oxfordjournals.org/content/27/24/3379.long.

17. Petersen, B., Petersen, T.N., Andersen, P., Nielsen, M., and Lundegaard, C. (2009). BMC Structural Biology A generic method for assignment of reliability scores applied to solvent accessibility predictions. BMC Struct. Biol. *9*. Available at: http://www.biomedcentral.com/1472-6807/9/51.

18. Jones, D.T. (1999). Protein secondary structure prediction based on position-specific scoring matrices 1 1Edited by G. Von Heijne. J. Mol. Biol. *292*, 195–202. Available at: http://www.ncbi.nlm.nih.gov/pubmed/10493868 http://linkinghub.elsevier.com/retrieve/pii/S0022283699930917.

19. Sillitoe, I., Lewis, T.E., Cuff, A., Das, S., Ashford, P., Dawson, N.L., Furnham, N., Laskowski, R.A., Lee, D., and Lees, J.G. *et al.* (2015). CATH: comprehensive structural and functional annotations for genome sequences. Nucleic Acids Res. *43*, D376–D381. Available at: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gku947.

20. Remmert, M., Biegert, A., Hauser, A., and Sding, J. (2012). HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. Nat. Methods *9*, 173–5. Available at: http://dx.doi.org/10.1038/nmeth.1818.

21. Seemayer, S., Gruber, M., and Sding, J. (2014). CCMpred-fast and precise prediction of protein residue-residue contacts from correlated mutations. Bioinformatics, btu500. Available at: http://bioinformatics.oxfordjournals.org/content/early/2014/08/12/bioinformatics.btu500.

22. Stein, R.R., Marks, D.S., and Sander, C. (2015). Inferring Pairwise Interactions from Biological Data Using Maximum-Entropy Prob-

ability Models. PLOS Comput. Biol. *11*, e1004182. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=4520494{\&}tool=pmcentrez{\&}rendert

23. Buslje, C.M., Santos, J., Delfino, J.M., and Nielsen, M. (2009). Correction for phylogeny, small number of observations and data redundancy improves the identification of coevolving amino acid pairs using mutual information. Bioinformatics *25*, 1125–31. Available at: http://www.ncbi.nlm.nih.gov/pubmed/19276150 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2672635.

24. Morcos, F., Pagnani, A., Lunt, B., Bertolino, A., Marks, D.S., Sander, C., Zecchina, R., Onuchic, J.N., Hwa, T., and Weigt, M. (2011). Direct-coupling analysis of residue coevolution captures native contacts across many protein families. Proc. Natl. Acad. Sci. U. S. A. *108*, E1293–301. Available at: http://www.pnas.org/content/108/49/E1293.full.

25. Jones, D.T., Buchan, D.W.A., Cozzetto, D., and Pontil, M. (2012). PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. Bioinformatics *28*, 184–90. Available at: http://bioinformatics.oxfordjournals.org/content/28/2/184.full.

26. Ekeberg, M., Hartonen, T., and Aurell, E. (2014). Fast pseudolikelihood maximization for direct-coupling analysis of protein structure from many homologous amino-acid sequences. J. Comput. Phys. *276*, 341–356. Available at: http://www.sciencedirect.com/science/article/pii/S0021999114005178.

27. Kamisetty, H., Ovchinnikov, S., and Baker, D. (2013). Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era. Proc. Natl. Acad. Sci. U. S. A. *110*, 15674–9. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3785744{\&}tool=pmcentrez{\&}rendert

28. Bengio, Y. (2012). Practical Recommendations for Gradient-Based Training of Deep Architectures. In Neural networks: Tricks of the trade (Springer Berlin Heidelberg), pp. 437–478. Available at: https://arxiv.org/pdf/1206.5533v2.pdf.

29. Mahsereci, M., Balles, L., Lassner, C., and Hennig, P. (2017). Early Stopping without a Validation Set. arXiv. Available at: http://arxiv.org/abs/1703.09580.

30. Carreira-Perpin, M. a, and Hinton, G.E. (2005). On Contrastive Divergence Learning. Artif. Intell. Stat. *0*, 17. Available at: http://learning.cs.toronto.edu/{˜}hinton/absps/cdmi

31. Bottou, L. (2012). Stochastic Gradient Descent Tricks. In Neural networks: Tricks of the trade (Springer, Berlin, Heidelberg), pp. 421–436. Available at: http://link.springer.com/10.1007/978-3-642-35289-8{_}25.

32. Kingma, D., and Ba, J. (2014). Adam: A Method for Stochastic Optimization. Available at: http://arxiv.org/abs/1412.6980.

33. Chollet, F. and others (2015). Keras. Available at: https://github.com/fchollet/keras.

34. Dieleman, S., Schlter, J., Raffel, C., Olson, E., Snderby, S.K., Nouri, D., Maturana, D., Thoma, M., Battenberg, E., and Kelly, J. *et al.* (2015). Lasagne: First release. Available at: https://zenodo.org/record/27878.

35. Hinton, G.E. (2002). Training Products of Experts by Minimizing Contrastive

Divergence. Neural Comput. *14*, 1771–1800. Available at: http://www.gatsby.ucl.ac.uk/publications/tr/tr00-004.pdf.

36. Bengio, Y., and Delalleau, O. (2009). Justifying and Generalizing Contrastive Divergence. Neural Comput. *21*, 1601–21. Available at: http://www.iro.umontreal.ca/{~}lisa/publications2/index.php/attachments/single/105.

37. Tieleman, T. (2008). Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. Proc. 25th Int. Conf. Mach. Learn. *307*, 7.

38. Jensen, C.S., Kjrulff, U., and Kong, A. (1995). Blocking Gibbs sampling in very large probabilistic expert systems. Int. J. Hum. Comput. Stud. *42*, 647–666. Available at: http://www.sciencedirect.com/science/article/pii/S1071581985710294.

39. Robinson, A.B., and Robinson, L.R. (1991). Distribution of glutamine and asparagine residues and their near neighbors in peptides and proteins. Proc. Natl. Acad. Sci. U. S. A. *88*, 8880–4. Available at: http://www.ncbi.nlm.nih.gov/pubmed/1924347 http://www.pubmedcentral.nih.gov/articlerend

40. Atchley, W.R., Zhao, J., Fernandes, A.D., and Drke, T. (2005). Solving the protein sequence metric problem. Proc. Natl. Acad. Sci. U. S. A. *102*, 6395–400. Available at: http://www.pnas.org/content/102/18/6395.abstract.

41. Kawashima, S., Pokarowski, P., Pokarowska, M., Kolinski, A., Katayama, T., and Kanehisa, M. (2008). AAindex: amino acid index database, progress report 2008. Nucleic Acids Res. *36*, D202–5. Available at: http://www.ncbi.nlm.nih.gov/pubmed/17998252 http://www.pubmedcentral.nih.gov/articleren

42. Wimley, W.C., and White, S.H. (1996). Experimentally determined hydrophobicity scale for proteins at membrane interfaces. Nat. Struct. Biol. *3*, 842–8. Available at: http://www.ncbi.nlm.nih.gov/pubmed/8836100.

43. Cornette, J.L., Cease, K.B., Margalit, H., Spouge, J.L., Berzofsky, J.A., and DeLisi, C. (1987). Hydrophobicity scales and computational techniques for detecting amphipathic structures in proteins. J. Mol. Biol. *195*, 659–685. Available at: http://linkinghub.elsevier.com/retrieve/pii/0022283687901896.

44. Zhu, H., and Braun, W. (1999). Sequence specificity, statistical potentials, and three-dimensional structure prediction with self-correcting distance geometry calculations of beta-sheet formation in proteins. Protein Sci. *8*, 326–42. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2144259{\&}tool=pmcentrez{\&}

45. Bernard, S., Heutte, L., and Adam, S. (2009). Influence of Hyperparameters on Random Forest Accuracy. In (Springer, Berlin, Heidelberg), pp. 171–180. Available at: http://link.springer.com/10.1007/978-3-642-02326-2{_}18.

46. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., and Dubourg, V. *et al.* (2011). Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. *12*, 2825–2830. Available at: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

47. Wu, S., and Zhang, Y. (2008). A comprehensive assessment of sequence-based and template-based methods for protein contact prediction. Bioinformatics *24*,

924–31. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2648832{\&}tool=p

48. Wang, X.-F., Chen, Z., Wang, C., Yan, R.-X., Zhang, Z., and Song, J. (2011). Predicting residue-residue contacts and helix-helix interactions in transmembrane proteins using an integrative feature-based random forest approach. PLoS One *6*, e26767. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3203928{\&}tool=p

49. Di Lena, P., Nagata, K., and Baldi, P. (2012). Deep architectures for protein contact map prediction. Bioinformatics *28*, 2449–57. Available at: http://bioinformatics.oxfordjournals.org/content/28/19/2449.full{\#}sec-14.

50. Wang, Z., and Xu, J. (2013). Predicting protein contact map using evolutionary and physical constraints by integer programming. Bioinformatics *29*, i266–73. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3694661{\&}tool=pmcentre

51. Cheng, J., and Baldi, P. (2007). Improved residue contact prediction using support vector machines and a large feature set. BMC Bioinformatics *8*, 113. Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1852326{\&}tool=pmcentrez{\&