Secondary Structure prediction using GOR method and Support Vector Machines

Giuili Edoardo
Department of Pharmacy and Biochemistry
Master's degree in Bioinformatics

Abstract

Motivation: Secondary structure prediction is a set of different tools that are able to predict the secondary structure conformation of a protein. In general, protein secondary structure prediction provides a significant first step toward tertiary structure prediction, as well as offering information about protein activity, relationships, and functions. Under these motivations, I implemented and compared two methods able to predict the secondary structure composition: a statistical approach method named Garnier-Osguthorpe-Robson (GOR) method (Garnier *et al.*, 1978) and a machine learning approach using SVMs.
Results: The GOR approach has demonstrated to reach 59.8% of accuracy, while SVM has reached an accuracy of 63.9%. Both methods were compared to the secondary structures inferred by the DSSP program.
Contact: edoardo.giuili@studio.unibo.it

1. Introduction

Protein secondary structure is an important characteristic of proteins. It represents the protein backbone conformation and it can be used to predict the tertiary structure of a protein. Thus, predicting the secondary structure can offer information about the function, the localization, and the relationships of a protein. Under these motivations, several tools have been developed in the last decades which try to predict the secondary structure conformation starting from the primary sequence. One of the first algorithms developed was the Chou-Fasman (Chou and Fasman, 1974) method which relies on frequencies of each amino acid's appearance in each type of secondary structure. The accuracy reached was around 50%. The GOR method was developed in 1978 (Garnier *et al.*, 1978) and showed an accuracy of around 65%: it takes into account not only the frequencies of each amino acid in each type of secondary structure, but also a window of positions around each amino acid (conditional probability). However, a big step forward in the secondary structure prediction has been reached using machine learning methods, such as artificial neural networks and support vector machines, which have demonstrated to be able to reach up to 80% of accuracy. In our work, I implemented the GOR method and SVMs, using a training set composed of 1348 protein sequences and a testing set composed of 150 protein sequences. In both methods I applied a cross-validation procedure to fix the hyperparameters and I finally tested the models on the testing set. In the GOR method I reached an overall accuracy of 59.8%, while in the SVMs 63.9%.

2. Materials and Methods

2.1 Training dataset description

The training dataset was retrieved from JPred4/JNet (v.2.3.1) web page (Drozdetskiy et al., 2015): it contains 1348 protein sequences that the authors used to train the model for protein secondary structure prediction. All of the 1348 sequences were subjected to several filters. Firstly the authors started from an initial set of 1987 representative domain sequences taken from each superfamily in SCOP v2.04 (Fox *et al.*,

2014). Secondly, all those sequences with low resolution (<2.5Å), a length in a range between 30 aa and 800 aa (extremes excluded) and missing DSSP information (Touw *et al.*, 2015) for less than 9 consecutive residues were retained. Finally, they obtained 1348 final sequences.
The domain length distribution can be approximated to a Gumbel distribution, with half of the distribution set around 150 residues in length (Fig.1).

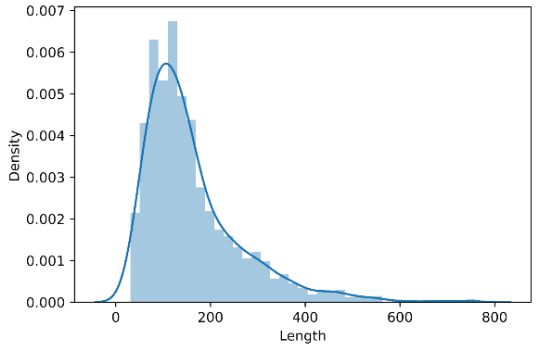

Figure 1 | Domain length distribution of the training set retrieved from JPred *(Drozdetskiy* et al.*, 2015)*.

All of these sequences were classified according to the SCOP classes. The absolute frequencies of each SCOP class has been reported in Fig.2.

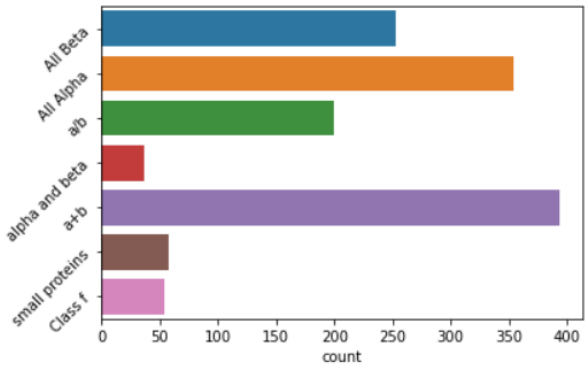

Figure 2 | The count plot of the 7 SCOP classes in the 1348 proteins sequences of the training set

Instead, the relative secondary structure conformations can be seen in the Fig.3, in which the percentages of the three main classes of secondary structure have been computed.

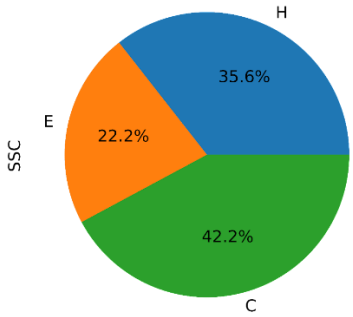

Figure 3 | The pie plot of the secondary structure conformations in the training set shows an higher percentage of helix and coil. The higher percentage of coil can be due to the fact that the authors collapsed in coil also missing information.

For each of the three main secondary structures, I computed the amino acid composition and I compared the relative frequencies of each amino acid in the different secondary structure and in the entire data set (Fig.4).

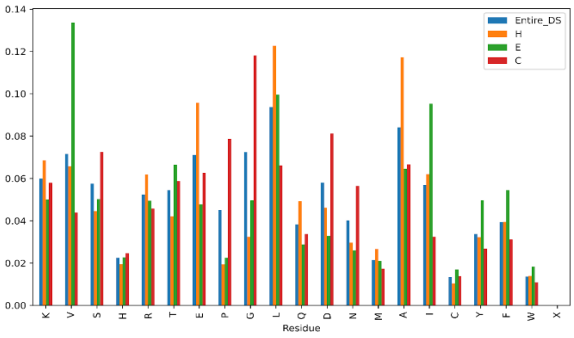

Figure 4 | The residues abundance in each class has been reported in a count plot, comparing the three classes and the entire dataset. As expected, some residues are more abundant in a secondary structure, such as V and I in strand, G in coil or A in helix.

I also performed a taxonomic classification: thus I computed the number of sequences belonging to a specific Super Kingdom and I plotted their relative frequencies into a pie chart (Fig.5).

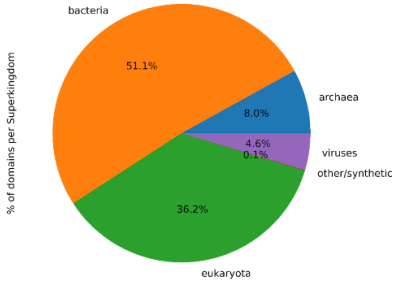

Figure 5 | Pie plot of the percentage of sequences belonging to a specific Super Kingdom.

2.2 Blind test description

The blind test set is an important additional dataset which gives information regarding the accuracy of a model without being biased by the training procedure. Indeed, the training of both GOR and SVM models are performed on the complete JPred4 training dataset, specifically under the cross-validation procedure. Once the parameters of the models have been fixed, the models must be evaluated using an independent test set, in order to understand if it is able to generalize well. The blind test set has been generated from Protein Data Bank (PDB). In particular, the extracted protein sequences had the following criteria: a deposit date after January, 2015 (since the JPred4 training set has been generated in 2014), a X-ray crystal resolution lower than 2.0 Å and a chain length between 50 and 800 residues. Obviously, only protein sequences were chosen.

After these filters, more than 20'000 of proteins were retrieved. However, I applied two more filters in order to remove the internal and external redundancy. Firstly, I clustered the sequences using *MMseqs* (Hauser *et al.*, 2016) with coverage 0.5 and minimum sequence identity of 0.3: that is, if two sequences share more than 30% of sequence identity with a coverage more than 50%, they will be grouped in the same cluster: *mmseqs easy-cluster cluster_seqs.fasta --cluster-mode 1 cluster_seqs tmp --min-seq-id 0.3 -c 0.5 --cov-mode 1* . With *--cov-mode 1* I specified the type of clustering MMseq has to follow: in this case a single-linkage clustering (remote homologs can be detected and clustered together). After the clustering procedure, I retained only the representative sequences of each cluster: in this way I reduced the internal redundancy (all the sequences retained are different). Secondly, I used blastp to reduce the external redundancy, which consists in removing all those sequences in the blind test set that have even only one match sequence in the training set. In this way I reduce the similarity between the blind and training set. In order to perform this procedure, I adopted blastp. In particular, I first created the database with the command "*makeblastdb –in training_seqs.fasta -dbtype prot"* using the training set as input sequences. Secondly, I used blastp to search for matches between the blind set and the database, with a 0.01 e-value threshold "*blastp -query cluster_seqs_rep_seq.fasta -db training_seqs.fasta -evalue 0.01 -out hits.blast.tab –outfmt 6".* Finally, I retained in the blind set only those sequences with no matches with the database (more than 4'000 of sequences). Since the number of proteins that I wanted to retrieve at the end in the blind set was 150, I randomly extracted 150 sequences. At the end of this procedure, for the final 150 sequences, I generated the corresponding dssp files using a BioPython function (Cock *et al.*, 2009).

2.3 Sequence profile generation

Once I obtained the final 150 blind test sequences, I had to generate profiles for both training and blind sets. In particular, I used PsiBlast (Altschul *et al.*, 1997) to generate a sequence profile for each sequence of the two data sets. In this way the model can generalize better in the prediction phase since a profile sequence gives much more information about, for example, strongly conservated positions. In order to run PsiBlast (Altschul *et al.*, 1997), I firstly created a database consisting of all the SwissProt (The UniProt Consortium, 2021) sequences and I specified 2 main parameters: an e-value threshold of 0.01 and a number of iterations equal to 3. In the training set, 144 sequences were found to have zero matches with the database or to have a zero only profile, while in the blind set 26 sequences: the 144 training sequences were removed from the training dataset, while for each one of the 26 blind sequences a one-hot encoding profile was generated. For the remaining sequences with matches, the profiles have been extracted in a matrix for further analysis.

2.4 GOR method

The Garnier-Osguthorpe-Robson (GOR) method (Garnier *et al.*, 1978) is one of the so-called second-generation methods for protein secondary structure prediction. The GOR

method was published for the first time in 1978 and followed the first method implemented by Chou and Fasman (Chou and Fasman, 1974) at the beginning of the 70'. The GOR was the first method to reach 60% of accuracy and it differs from the previous methods because it takes into account, together with the probability for a given residue in a given position to generate a given secondary structure, also the contribution of the residue local sequence context, that is, a window of residues before and after the central residue.

Figure 6| In the figure, the window opened at a specific position of a protein sequence

Entering into the mathematical formulation, the method is divided into a training phase, in which the parameters of the model are computed, and a prediction phase, in which it tries to predict the secondary structure of a new sequence given the model previously trained. In the training phase the method computes the probability for the residue R in a given position to generate the secondary structure as follows:

$$I(S;R) = \log\frac{P(S|R)}{P(S)} = log\frac{P(R,S)}{P(S)P(R)}$$

Where P(R,S) is the joint probability of having R and S in a given position and P(S) and P(R) are the relative abundances of R and S in the initial dataset.

This type of probability can be extended to take into account also the local context of the residue R. However, taking into account all the conditional probabilities for all the close residues would be computationally very expensive since we would get an exponential number of possible configurations. For this reason, the final mathematical formulation of I(S; R) makes the assumption that within a window the residues are independent from each other. In this way, I(S; R) can be written as:

$$I(S;R_{-d},\dots,R_d) = \log\frac{P(S,R_{-d},\dots,R_d)}{P(S)P(R_{-d},\dots,R_d)} =$$

$$\log\prod_{k=-d}^{d}\frac{P(S,R_k)}{P(S)P(R_k)} = \sum_{k=-d}^{d}\log\frac{P(S,R_k)}{P(S)P(R_k)}$$

$$\text{where d} = \frac{\text{size of the window}}{2} + 1$$

After the training phase, given a new protein sequence, for each position it generates the secondary structure with the highest value of the window-based information function:

$$S^* = argmax_s I(S;R_{-d},\dots,R_d) = argmax_s \sum_{k=-d}^{d} I(S;R_k)$$

In my case, I decided to use a window size of 17 residues (Supplementary Materials: gor_training.py, gor_prediction.py, gor_performance.py).

2.5 Support Vector Machines (SVMs)

SVMs are supervised learning models in machine learning that are able to solve classification and regression problems. Specifically, for classification tasks, given a training set of points labeled with 0 or 1 value, SVMs are able to separate these points by a hyperplane if the points are linearly separable. Since there is an infinite number of hyperplanes passing through a point (if we are in R^2 space), the method chooses the hyperplane that maximizes the margin. The margin is simply the distance between support vectors belonging to the two classes, where support vectors are the vectors that rely on the margin hyperplanes:

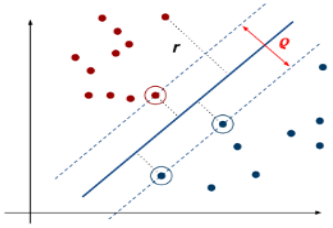

Figure 7 | Hard margin SVM with Q representing the margin.

However, in case of non-linearly separable problems it is possible to allow small errors ζ_i:

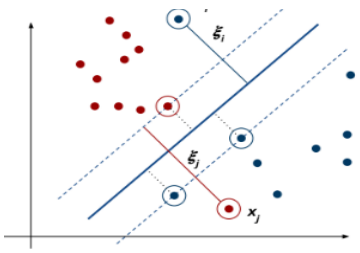

Figure 8| Soft margin SVM.

In my case, the problem is represented by correctly predicting (classifying) the secondary structure for each position of a new protein sequence. Since the possible secondary structures are three (helix, strand and coil) I had to use a Multi-Class SVM, in which an all-against-all procedure is applied and, for each position, the most probable secondary structure is generated. For the multi-class SVM I used the scikit learn python library. In order to train the SVM, I encoded the training sequence profiles into two matrices: a matrix X containing all 20 dimensional vectors representing the rows of the profiles; a matrix Y containing 1 for helix, 2 for strand and 3 for coil representing the secondary structure of all the training sequences. In particular, the matrix X is a list of lists: the lists are composed of windows of size 17 and have a length of 340 ($17\,rows \times 20\,columns$). The matrix Y instead contains as many values as the sum of the lengths of all the training sequences. Thus, the number of lists in X must be equal to the number of elements in Y. Finally, with the cross-validation procedure I fixed the two hyperparameters C and $\boldsymbol{\gamma}$ through a so called grid search procedure: that is, the algorithm chose C and $\boldsymbol{\gamma}$ that maximize the accuracy. C is a penalty parameter of the error term (high C allows low errors, low C allows high errors), while the value of $\boldsymbol{\gamma}$ controls the width of the kernel distribution, which in my case is the Gaussian RBF Kernel.

Thus, using very high $\boldsymbol{\gamma}$, only a small number of points have a kernel different from zero (risk of overfitting), while with very low $\boldsymbol{\gamma}$ you allow even points very far from the 0 to have a kernel different from 0.

In order to perform the SVM procedure I implemented two different scripts: one for the training phase and the other one for the prediction phase and the evaluation of the performances (Supplementary Materials, svm_method.py, svm_prediction.py) .

2.6 Evaluation procedure and scoring measure

The evaluation procedure is an important aspect that has to be taken into consideration when building a model. In particular, given some score indexes, we can evaluate how well the model performs in a test set which is independent from the training set: how much the model is able to generalize to new data. In my case, the evaluation procedure is composed of three main data sets: a training set, a validation set and a blind (test) set. The training set is important to build the model starting from known examples (the sequences retrieved from JPred4) while the validation set is used to fix the hyperparameters. Specifically, I divided the training set composed of 1204 protein sequences into five splits of the same size and I performed a cross-validation procedure. A cross-validation procedure consists in training the model on 4 splits and test it on the remaining one: this step is iterated over all the splits, thus in my case 5 times. The cross-validation procedure is fundamental in fixing the hyperparameters. In particular, it is proven that the validation set error at some point starts increasing (even if the training set error continues decreasing): the validation set error starts increasing because the training set starts overfitting, and the overfitting is mainly due to the hyperparameters C and $\boldsymbol{\gamma}$ (Fig.9).

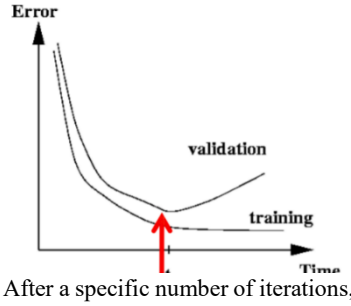

Figure 9 | After a specific number of iterations, the model starts overfitting and, even if the training set error continues decreasing, the validation set error starts increasing. This means that the model is not able to generalize on independent new examples.

Thus, after the cross-validation procedure I chose the hyperparameters C and $\boldsymbol{\gamma}$ that

maximized the scoring indexes (or minimized the validation set error).

However this method determines an overfitting in the validation set in terms of hyperparameters, and for this reason to finally validate the method (that is, to score the method in terms of error) I had to use a third set that in my case is the blind set that I have generated.

At the end of the evaluation procedure, I retrieved 4 main scoring indexes: the Matthews correlation coefficient (MCC), the three-class accuracy (Q_3), the precision (PPV) and the recall (also called sensitivity or true positive rate TPR). Mathematically, they are defined as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

$$Q_3 = \frac{TP_H + TP_E + TP_C}{N}$$

$$ACC = \frac{TN + TP}{TN + TP + FN + FP}$$

$$PPV = \frac{TP}{TP + FP} \qquad TPR = \frac{TP}{TP + FN}$$

Specifically, since I have three classes of data (helix, strand and coil), the MCC, PPV and TPR are computed for each type of class and consequently the mean and the standard deviation are retrieved. Instead, Q_3 score gives an overall accuracy, since it computes the total number of true positives over the total number of predictions (N).

3 Results

The cross-validation procedure, for both methods, was performed using the 5 splits of the training set. At the end of the procedure I computed the mean and the standard deviation of the MCC,ACC,TPR and PPV for each class (Table 1). Finally the models have been trained on the entire training set (in case of SVM using the optimal hyperparameter obtained at the end of the cross-validation procedure), tested on the blind set and evaluated. Moreover, the 26 one-hot encoded profiles of the blind set have been tested alone, in order to see the performances of the model on a set of sequences with no actual profile.

		Cross-Validation					
		GOR			SVM		
H	ACC	0.763	±	0.005	0.841	±	0.004
	MCC	0.524	±	0.008	0.655	±	0.016
	TPR	0.808	±	0.012	0.713	±	0.023
	PPV	0.629	±	0.014	0.813	±	0.006
E	ACC	0.772	±	0.003	0.835	±	0.004
	MCC	0.441	±	0.010	0.494	±	0.014
	TPR	0.706	±	0.017	0.525	±	0.007
	PPV	0.487	±	0.022	0.765	±	0.009
C	ACC	0.715	±	0.003	0.741	±	0.003
	MCC	0.420	±	0.005	0.493	±	0.009
	TPR	0.429	±	0.010	0.790	±	0.004
	PPV	0.810	±	0.008	0.615	±	0.006
	Q3	0.625	±	0.002	0.708	±	0.006

Table 1 | The results of 5-fold cross validation. For what concerns SVM, the values refer to the cross-validation performed using the optimal hyperparameters C = 2 and $\boldsymbol{\gamma}$ = 0.5.

The results of GOR showed a Q3 score equal to 0.598, which is more or less what such a method is expected to perform. Moreover, for both the entire blind set and the cross-validation procedure, the MCC for the helix shows to be higher than the MCC for strand and coil, and this difference can be due to the higher percentage, in the training set, of helix with respect to strand (35.6% for H and 22.3% for E) and to the fact that in the training set the authors of the article decided to consider coil also empty and ambiguous positions. For what concerns the one-hot encoded sequences, as expected the method performed worst compared to the performance on the sequences with a valid profile, and this is mainly due to the less provided information. Indeed, the profile gives to the model an higher and more complete picture of the evolutionary history of a protein family: this greater information resulted in more accurate predictions (Table 2).

		Hold-out validation					
		Entire Blind Set		Match Seqs		No Match Seqs	
		GOR	SVM	GOR	SVM	GOR	SVM
H	ACC	0.748	0.802	0.755	0.825	0.714	0.643
	MCC	0.486	0.550	0.503	0.613	0.396	0.021
	TPR	0.738	0.648	0.752	0.751	0.656	0.014
	PPV	0.667	0.795	0.679	0.834	0.596	0.446
E	ACC	0.748	0.798	0.753	0.821	0.722	0.751
	MCC	0.38	0.440	0.386	0.496	0.351	0.027
	TPR	0.664	0.501	0.666	0.53	0.656	0.012
	PPV	0.437	0.745	0.438	0.782	0.434	0.368
C	ACC	0.701	0.721	0.7	0.736	0.706	0.41
	MCC	0.344	0.383	0.338	0.442	0.373	0.004
	TPR	0.42	0.742	0.414	0.788	0.446	0.003
	PPV	0.691	0.596	0.684	0.65	0.723	0.411
	Q3	0.598	0.639	0.604	0.685	0.571	0.407

Table 2 | In the table the results for the entire blind set, for the sequences with a valid profile (Match_seqs, 124 in total) and for the protein sequences which were one-hot encoded (No_Match_seqs, 26 in total).

For what concerns the SVM method, I used the same 5 splits for the cross-validation procedure and I decided to use as hyperparameters for the blind testing set C = 2 and Gamma = 0.5, since they maximized the average MCC (Table 3).

SVM Grid Search						
C=2,Gamma=0.5						
	cv0	cv1	cv2	cv3	cv4	Mean
MCC_H	0.637	0.642	0.667	0.676	0.655	
MCC_E	0.503	0.482	0.514	0.481	0.491	**0.548 ± 0.07**
MCC_C	0.479	0.491	0.498	0.499	0.500	
Q3	0.699	0.706	0.717	0.709	0.709	**0.708 ± 0.01**
C=2,Gamma=2						
	cv0	cv1	cv2	cv3	cv4	Mean
MCC_H	0.228	0.254	0.276	0.254	0.257	
MCC_E	0.094	0.093	0.106	0.145	0.093	0.172 ± 0.07
MCC_C	0.138	0.160	0.170	0.156	0.155	
Q3	0.455	0.470	0.477	0.470	0.466	0.467 ± 0.01
C=4,Gamma=0.5						
	cv0	cv1	cv2	cv3	cv4	Mean
MCC_H	0.628	0.632	0.658	0.668	0.646	
MCC_E	0.493	0.476	0.503	0.474	0.481	0.538 ± 0.08
MCC_C	0.469	0.480	0.487	0.489	0.488	
Q3	0.693	0.700	0.711	0.704	0.702	0.702 ± 0.01
C=4,Gamma=2						
	cv0	cv1	cv2	cv3	cv4	Mean
MCC_H	0.226	0.252	0.272	0.250	0.254	
MCC_E	0.090	0.090	0.104	0.142	0.092	0.169 ± 0.07
MCC_C	0.136	0.157	0.167	0.152	0.153	
Q3	0.454	0.469	0.476	0.469	0.465	0.466 ± 0.01

Table 3 | The results of the SVM Grid Search. C = 2 and Gamma = 0.5 have shown to be the optimal hyperparameters.

Thus, I tested the model on the independent set using the above hyperparameters and I computed the performance of the model on the entire blind set, on the sequences with a valid profile and on the sequences with no valid profiles. As expected and according to the results obtained with the GOR method, the performances were higher on the sequences with a valid profile (Table 2).

4 Conclusion

Protein secondary structure prediction is an important aspect of the structural bioinformatics since it can result in preliminary information about the function, the localization and the putative relationships of a new protein. For this reason, two different models were trained and tested: the GOR and the SVM methods. The first one showed an overall accuracy on entire blind set lower than the overall accuracy of the SVM method (Q_3^{GOR}= 0.598 , Q_3^{SVM} = 0.639), as well as all the others score indexes such as the MCC, the PPV and the TPR. In both methods, the presence of a one-hot encoded profile determined decreased performances, demonstrating the importance of sequence profiles in the secondary structure prediction.

5 References

Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Chou,P.Y. and Fasman,G.D. (1974) Prediction of protein conformation. *Biochemistry*, **13**, 222–245.

Cock,P.J.A. *et al.* (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.

Drozdetskiy,A. *et al.* (2015) JPred4: a protein secondary structure prediction server. *Nucleic Acids Res.*, **43**, W389–W394.

Fox,N.K. *et al.* (2014) SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Res.*, **42**, D304–D309.

Garnier,J. *et al.* (1978) Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *J. Mol. Biol.*, **120**, 97–120.

Hauser,M. *et al.* (2016) MMseqs software suite for fast and deep clustering and searching

of large protein sequence sets. *Bioinforma. Oxf. Engl.*, **32**, 1323–1330.
The UniProt Consortium (2021) UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.*, **49**, D480–D489.
Touw,W.G. *et al.* (2015) A series of PDB-related databanks for everyday needs. *Nucleic Acids Res.*, **43**, D364–D368.