Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Structural bioinformatics

Comparing the Garnier-Osguthorpe-Robson method with Support Vector Machines for the prediction of protein secondary structure from primary sequence

Immanuela Antigone Engländer[1]

[1]Department of Pharmacy and Biotechnology, Alma Mater Studiorum Università di Bologna, Bologna 40126, Italy

Abstract

Motivation: Reliable and fast functional annotation of protein sequences is one of the most important topics in bioinformatics. Protein sequence data is generated at a constantly increasing pace. Experiments that can determine their corresponding structures at atomic resolution are, however, still costly and time consuming. Machine learning based methods can be adopted to tackle the ever growing amount of sequence data in a time and cost efficient manner. Therefore, we implement and compare two computational methods for secondary structure prediction: the Garnier-Osguthorpe-Robson and a Support Vector Machine. First a five-fold cross-validation was carried out on both methods. In a second and final step hold-out validation was carried out on both models by training each of them on the entire jPRED4 dataset and subsequent testing on the blind test set to evaluate their capability of generalizing to never-seen-before data. The methods were then compared in terms of their performance for predicting secondary structures of protein sequences of known structure.

Results: Both implementations successfully produced two models. The Garnier-Osguthorpe-Robson model achieved a three class accuracy of 63.7%. While the Support Vector machine surpassed the previous method with an accuracy of 69.3%. Despite the SVMs higher ability for predicting helix and coil conformations the performance on predicting strands remained low.

Availability:

Contact: immanuela.englander@studio.unibo.it

Supplementary information: Supplementary data are available at *Github: https://github.com/ilante/LB-2-project-GOR-vs-SVM*

1 Introduction

Determining a protein's three dimensional structure is fundamental to understanding protein function and thus its role in protein protein interactions (PPI). Structural biologists can experimentally solve protein structures by determining the relative position of each of its atoms in space. As of October 2020 almost 90% of proteins deposited Protein Data Bank (PDB) have been solved by X-ray crystallography [1] which is still considered being the gold standard. Predicting secondary structure is useful in homology modeling in absence of a good template for our protein with any standard alignment model as well as *ab initio* methods, where inferred secondary structure can be used to produce contact map predictions to enforce local constraints in structure formation. The disparity between experimentally solved protein structures ranges between 150 916 in the PDB [1], ~564 000 manually curated proteins deposited in the UniProtKB/Swiss-Prot and ~209 000 000 records in TrEMBL that have yet to be reviewed [2, Dec. 2020]. Annotating all these sequences with a structure by using experimental methods only would be a time consuming and very costly endeavor. Given the sheer numbers of protein sequences in databases, developing and refining methods for reliably predicting three dimensional protein structure from primary sequences remains a major challenge in bioinformatics that has been studied by scientists around the world since the 1960's. First generation prediction methods, such as the Chou-Fasman scale, were based on the propensities of single amino acids and statistics. It uses the relative frequency of individual amino acids found in each secondary structure which was derived from protein

structures solved by X-ray crystallography [3]. That way scale takes into account merely the probability of each individual residue being part of a coil, strand or a helix. Second generation methods such as the Garnier-Osguthorpe-Robson (GOR) [4] evaluate the propensities of segments of neighboring residues including the contribution of the context. That way the GOR method includes conditional probabilities of residues being part of a particular secondary structure given adjacent amino acids being in that secondary structure in contrast to the Chou-Fasman method. GOR simplifies the computation with the assumption of statistical independence of the residues in the window which is not valid from a biological point of view. This is achieved by the use of information theory and and Bayesian statistics while implementing a sliding window to capture adjacent residues and reached an accuracy of ~60%.

Third generation methods such as PSIPRED implement neural networks for predicting protein secondary structure based on position specific scoring matrices generated by PSI-BLAST [5]. PSIPRED has reported a multi-class accuracy (Q_3) of 78.3% [5]. By including evolutionary information, contained in multiple alignments and sequence profiles, both the variation of the sequence and the structure are accounted for. These methods reached around 70% of accuracy by using complex algorithms trained on large databases such as implemented by Rost & Sander in the 1994 [6].

Even if knowing the secondary structure is not as informative as knowing the entire three-dimensional structure, secondary structure predictions produce important constraints for techniques such as homology modelling, *ab initio*, fold-recognition techniques, constraint-based tertiary structure prediction methods, and for the identification of functional domains [7]. The fact that sequence is less conserved than structure which in turn is less conserved than function can be exploited by computational methods. The purpose of developing such methods is to manage the colossal amount of data in a more efficient manner and consequently rely less heavily on experimental data.

2 Approach

The objective of the present study is to compare the performance of the GOR method and Support Vector Machines (SVMs) based approach for the prediction of protein secondary structure. Even though algorithms such as the Define Secondary Structure of Proteins algorithm (DSSP) [8] distinguish eight different classes of secondary structure conformations, we limited our methods to distinguish only the following three super classes: $\alpha-$helix, $\beta-$sheet and coil. Both methods were trained and tested on the JPred4 dataset in five-fold cross-validation and then subjected to a hold-out validation consisting in final a training on the entire dataset and subsequent validation on a blind test set. The blind test set was generated to be as different from the training set as possible, to prove that generalization of respective performances is achieved.

3 Methods

3.1 Generating the training dataset

The training set was generated from the JPred4 dataset which has been used by the authors of JPred4 to train their predictor [7]. The initial SCOPe ASTRAL set comprised 1987 representative domain sequences from each superfamily in SCOPe v.2.04. The set was further filtered by removing domain sequences having a resolution above 2.5 Å, to ensure good quality. Structures of less than 30 or more than 800 residues were excluded, as the first kind is unlikely to represent protein domains, while the upper bound was set for computational reasons. Domain sequences that were missing more than 9 adjacent residues obtained from the Define Secondary Structure of Proteins algorithm (DSSP) [8] were filtered out as well as fragments. Sequences with mapping inconsistencies or sequences that failed to generate a profile in Position-Specific Iterated BLAST (PSI-BLAST) [9] were excluded as well. After these filtering steps we were left with 1348 domain sequences in the training set.

3.2 Generating the blind test set

For the final performance evaluation in the hold-out validation of the GOR and the SVM method we generated an independent blind test set. It serves the purpose of unbiased comparison which is not susceptible to overfitting of models to the training data.

All data was accessed and downloaded via the PDB advanced search [10, 11]. Considering that JPred4 was released in 2014 we chose domain sequence structures that were deposited after January 2015. The criteria for resolution remained the same as for the training set; we included only structures of a resolution below 2.5 Å. The lower bound of chain length was set to 50 residues while the upper bound was limited to 300 residues. These search criteria returned 22636 structures which were further refined. In a first filtering step internal redundancy was limited to 30% sequence identity which was achieved by employing the BLASTClust algorithm [12, 13] with a coverage set to 50% ensuring a high level of heterogeneity within the blind set. One representative was taken from each of the best clusters and then run against the training set employing the BLASTP algorithm [12] ensuring that all pairs have less than 30% sequence identity. This step serves the goal of creating a highly heterogeneous blind test set that is dissimilar to the training set. Out of all sequences satisfying the described criteria 160 were picked at random and their corresponding PDB structures were downloaded. The final blind set comprised however only 150 domain sequences after running the DSSP program as explained below.

3.2.1 The Define Secondary Structure of Proteins algorithm

The DSSP algorithms was used to map each residue in each of the selected PDB structures to one of eight secondary structure labels [8]. The DSSP output distinguishes between $\alpha-$helix (H), $3_{10}-$helix (G), $\pi-$helix (I), isolated $\beta-$bridge (B), extended strand (E), hydrogen bonded turn (T) and bend (S) for residues lacking any assignment are denoted with an empty string (" "). For our purposes we remapped those eight classes into three super classes; All three types of helices (H, G and I) were mapped to the label helix (H) and both types of $\beta-$ (B and E) were mapped to the label strand (E). The third label, coil (C) was assigned to turns and bends but also to any residue that has not been endowed with a structural label by the DSSP (see Figure 6).

3.3 Highlighting basic statistics the training dataset

We analyzed and compared our training set with the UniProtKB/Swiss-Prot [14, 15] to ensure that it is a representative sample of the protein space. We distinguish three super-classes of secondary structure conformations: helix (H), strand (E) and coil (C). Relative abundance of secondary structure conformations is 35.6% H, 42.2% C and 22.2% E (Figure 6) and is comparable to the one in UniProtKB/Swiss-Prot [16, Oct-07, 2020] (see Supplementary material) when considering abundance of $\alpha-$helices. The class of $\beta-$sheets in the training set is 12% smaller while the class of coils is 12.7% larger. The Structural Classification of Proteins (SCOP) class [17] is shown in figure 1. The distribution of superkingdoms and the top 10 species is highlighted in Figure 2 and Figure **??** and compares well to the UniprotKB/Swiss-Prot entries per taxonomic group [16, Oct-07, 2020]. We computed the comparative amino-acid composition of the dataset considering the frequency of each residue (individually) and each residue in each of the conformations (H, E and C) to be classified as reported in Figure 3.

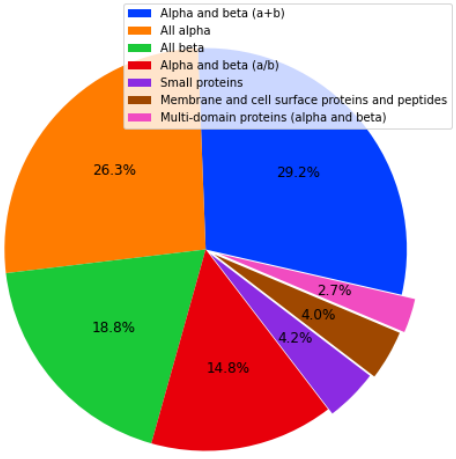

Fig. 1. The figure shows the distribution of SCOP classes within the dataset. The three largest classes are 'alpha and beta', 'all alpha' and 'all beta'

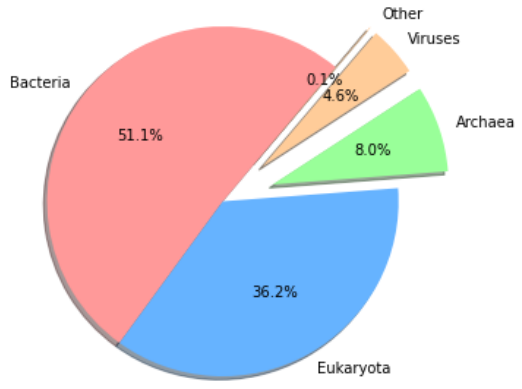

Fig. 2. The figure shows the distribution of superkingdoms within the dataset. The two largest groups are bacteria with 51.1% followed by eukaryota constituting 36.3%.

3.4 Generating sequence profiles

Sequence profiles can be used for storing the probabilities of a residue being found in a certain position. Using sequence profiles rather than single sequences for training the GOR and the SVM allowed us to capture the notion of a protein family by including evolutionary information to improve predictions as discussed in the introduction. Aligned positions of a multiple sequence alignment represent structural and thus functional regions which show a certain level of sequence variability. We used sequence profiles because they are condensed representations of a multiple sequence alignments thus allowing us to generate an input encoding that captures variability going beyond the mere propensity of each amino acid to form part of a certain secondary structure.

To obtain sequence profiles the PSI-BLAST algorithm was used [9]. The program builds a multiple sequence alignment and therefrom a sequence profile capturing evolutionary information of each domain sequence. The PSI-BLAST algorithm iteratively searches a sequence database for sequences similar to the queries[9]. In the first iteration it runs a standard BLAST [12] where it performs a pairwise comparison using a BLOSUM or PAM substitution matrix. Then it computes a multiple sequence alignment from the matches by stacking pairwise local alignments. From the generated multiple sequence alignment it computes a position specific substitution matrix (PSSM). In all consecutive iterations it performs searches while using the PSSM generated in each previous step, from which it generates a new multiple sequence alignment and a corresponding PSSM.

As our database, we indexed the entire UniprotKB/Swiss-Prot containing 56 3082 sequences as of September 2020 [16]. PSI-BLAST was run with an E-value set to 0.01, three iterations and both the number of alignments, and the number of descriptions set to 10000. By parsing each PSSM file we extracted sequence profiles dividing each value by 100 for normalization in the range of 0 and 1. Not all inputs yielded a valid profile in output when running a PSI-BLAST against the UniprotKB/Swiss-Prot [16], as for some sequences PSI-BLAST did not find any satisfactory matches within the database with the given parameters. For other sequences PSI-BLAST generated profiles containing zeros only which were removed manually. This left us with a final dataset comprised of 1204 sequences in the training set and 150 in the blind test set.

3.5 The principles of the Garnier–Osguthorpe–Robson method

Garnier, Osguthorpe and Robson first introduced this second generation method for secondary structure inference from primary sequence in 1987 [4]. It combines Bayesian statistics and information theory while taking into account evolutionary information and the influence of adjacent amino acids of a given residue for assigning secondary structure. Originally it distinguished between the four secondary structures helix, sheet, coil and turn (H, E, C, and T respectively).

Due to the difficulty of discriminating different turn types using the DSSP program [8] GOR was however modified to infer only H, E and C and it was further improved by incorporating a larger database [18]. GOR acquires propensity measures from training data consisting of protein sequences of known secondary structure. The parameters are stored in one matrix per conformation (H,E and C) following training. These matrices are then used for inferring the conformation that achieves the highest propensity score for each residue of a query sequence. The information function was originally employed in electronic transmission of information and is used here in single residue context [18].

$$I(S;R) = log\frac{P(S|R)}{P(S)} \tag{1}$$

$$= logP(S|R) - logP(S) \tag{2}$$

Where S is one of the three possible conformations (H,E or C), R is a given residue out of 20 amino acids and $P(S|R)$ is the conditional probability of observing the conformation S given the residue R. $P(S)$ is the marginal probability of observing the conformation S. By employing the chain rule the equation may be rewritten as:

$$I(S;R) = log\frac{P(R,S)}{P(S)P(R)} \tag{3}$$

The following parameters are estimated from training data: P(R,S) is the joint probability of observing residue R in conformation S; P(R) is the marginal probability of observing a residue type R; and P(S) is the marginal probability of observing a residue in conformation S. If the log ratio is > 0 there is a dependency between the residue and the structure. Given the

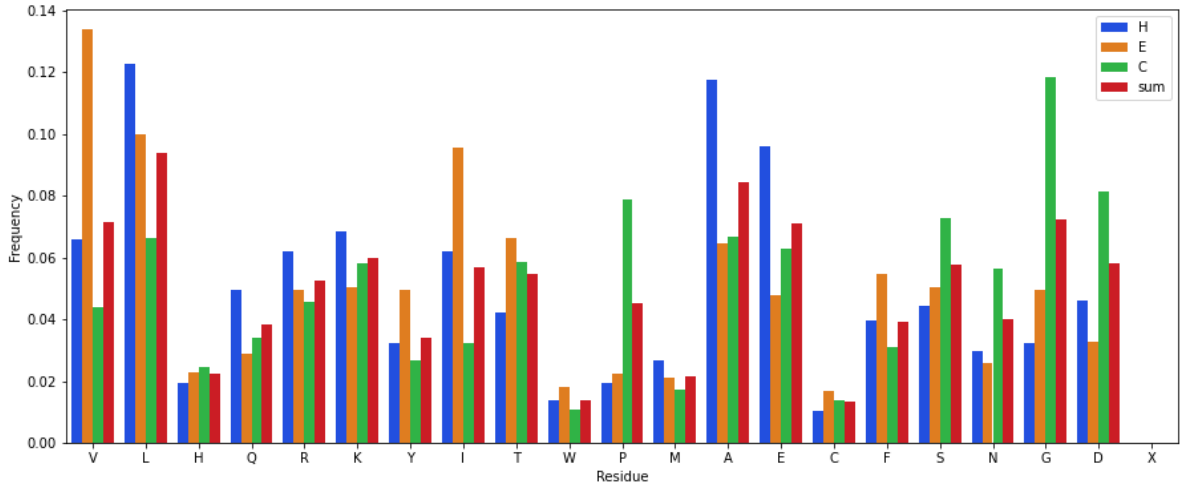

Fig. 3. The figure shows the frequency of each residue in each conformation where H, E and C denote helix, strand and coil and the abundance of each residue with respect to the entire dataset and in each conformation.

definition of conditional probability we obtain

$$P(S|R) = \frac{P(S,R)}{P(R)} \tag{4}$$

Where $P(S,R)$ denotes the joint probability of observing the conformation S and residue R and $P(R)$ is the probability of observing residue R. $I(S,R)$ is then estimated from a database of known sequences and their corresponding observed secondary structures given that the joint probability of observing conformation S and residue R is:

$$P(S,R) = \frac{f_{S,R}}{N} \tag{5}$$

$P(R)$ and $P(S)$ are the probability of observing residue R and the probability of observing conformation S:

$$P(R) = \frac{f_R}{N} \tag{6}$$

$$P(S) = \frac{f_S}{N} \tag{7}$$

Where N is the total number of residues in the database $f_{S,R}$ the number of residues R observed in conformation S in that same data base, f_R the total number of residues R and f_S being the total number of residues observed in the conformation S, again in that same data base. Thus the information function can be expressed as follows provided that the database is large enough [18]:

$$I(S,R) = log\frac{(\frac{f_{S,R}}{f_R})}{(\frac{f_S}{N})} \tag{8}$$

3.5.1 Sequence windows to include information from neighboring residues

To incorporate the influence of neighboring residues on the conformation of a given amino acid the information function I can be extended by considering a sequence window that spans d residues before and after the central residue R_0.

$$I(S; R_{-d}, ..., R_0, ..., R_d) = log\frac{P(S|R_{-d}, ..., R_d)}{P(S)} \tag{9}$$

$$= log\frac{P(S, R_{-d}, ..., R_d)}{P(S)P(R_{-d}, ..., R_d)} \tag{10}$$

The value of d is set to 8 resulting in a window of size 17 based on studies of information content [18]. It is important to note that GOR implements one simplifying assumption: Calculating the joint probabilities of S and all possible residues in all possible positions within the window is computationally expensive and would require much larger databases to render reliable probabilities. To overcome this limitation we simplify the computation by assuming statistical independence of the residues $R_{-d}, ..., R_d$ within the window. This allows us to factorize the joint probability of the full context into the product of marginal probabilities of the residues in the context yielding the following equation:

$$P(R_{-d}, ..., R_d) = \prod_{k=-d}^{d} P(R_k) \tag{11}$$

This independence assumption allows us to reformulate the information function I as follows:

$$I(S; R_{-d}, ..., R_d) = log\frac{P(S, R_{-d}, ..., R_d)}{P(S)P(R_{-d}, ..., R_d)} \tag{12}$$

$$= log\frac{P(S)\prod_{k=-d}^{d} P(R_k|S)}{P(S)P(R_{-d}, ..., R_d)} = log\prod_{k=-d}^{d}\frac{P(R_k, S)}{P(S)P(R_k)} \tag{13}$$

$$= \sum_{k=-d}^{d} log\frac{P(R_k, S)}{P(S)P(R_k)} = \sum_{k=-d}^{d} I(S; R_k) \tag{14}$$

Where $P(R_k)$ is the probability of residue R in position k and $P(R_k, S)$ is the probability of residue R observed in position k with the central residue R_0 being in conformation S. The information function I as written in 14 allows us to calculate the sum of each information function I as shown in equation 3 using the parameters derived from the training data.

We have implemented the GOR method using sequence profiles to weigh the contribution from different residues thus rendering the information

function as follows:

$$I(S, PW) = \sum_{k=-d}^{d} \sum_{R_k} [PW_{R_k}] * I(S; R_k) \tag{15}$$

Where PW is the profile window and k is index of Residue R. The prediction is carried out by assigning the conformation associated to the highest scoring value of the window-based information function I.

3.5.2 Implementation of the GOR - method

We implemented the GOR method via two python scripts. One dedicated to generating the trained model in the form of five matrices: The matrices for joint probabilities of residues and conformation H, E, C, marginal probabilities of the residues and marginal probabilities of the secondary structure conformations. As suggested by the authors of GOR, the window size was set to 17 ($d = 8$). Five fold cross-validation, generating the final model, and final testing on the blind test set was automated with help of a bash script.

3.6 The principles of support vector machines

Support vector machines (SVMs) are supervised learning algorithms that were first published under the name of *support vector networks* by Corinna Cortes and Vladimir Vapnik in 1995 [19]. These models work by analyzing labeled training data belonging to one of two classes (e.g. +1 and -1) and are able to perform reliable binary and multi-class classification and regression. For the scope of this work we have only used SVM for classification purposes. The SVM algorithm finds an *optimal hyperplane*, that separates the two classes, by maximizing the margin. The maximum margin hyperplane is used as a decision boundary for classifying new data points either to class +1 or -1 (see Figure 4). The properties of the decision boundary ensure high generalization ability of this method [19]. The *kernel trick* enables SVMs to tackle even non-linear classification by mapping the input into a (usually) higher-dimensional feature space [20]. The same approach has been exploited for secondary structure prediction in 2001 by Hua et al. who reported a per-residue accuracy Q_3 of 73.% [21]. We identify a set of *support vectors (SV)*, defined by the points that are closest to the maximum-margin-hyperplane. The maximum-margin-hyperplane is defined as the set of vectors $\vec{x}$, that have a certain constant projection (positive or negative) on a defining vector $\vec{w}$. We can define a hyperplane for the training data x_i $(i = 1, 2, 3, ..., n)$ as:

$$\vec{w}^T \vec{x} + b = 0 \tag{16}$$

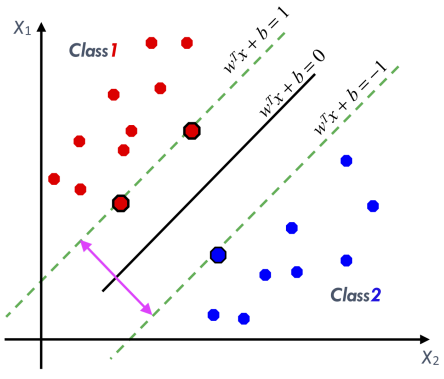

Fig. 4. SVM are algorithms allowing for separation of two classes with the help of a hyperplane accommodating the largest margin indicated by the pink arrow. The margin is the distance between the two nearest examples. The bold datapoints indicate the support vectors. The image was adapted from Support Vector Machine: Principles, Parameters, and Applications, Gholami and Fakhari [22]

The margin m of the separator is the distance between the support vectors of the 2 different classes. By employing the optimality criterion for choosing the hyperplane which maximizes the margin we imply that only the support vectors matter while other training examples can be ignored. This approach also aides in reducing the likelihood of misclassification of new data. Given these two properties the data points of each class y^i can only lie on opposite sides of the hyperplane such that the positive class is $y_i = +1$ and the negative class is $y_i = -1$. We can choose $\vec{w}$ and b such that the decision boundary is defined as in Equation 16 and the margins are defined as:

$$\left.\begin{array}{ll} \vec{w}^T \vec{x} + b \leq +1 & \text{for} \quad y_i = +1 \\ \vec{w}^T \vec{x} + b \leq +1 & \text{for} \quad y_i = -1 \end{array}\right\} \Rightarrow y_i(\vec{w}^T \vec{x} + b \leq +1) \quad \forall \vec{x^i} \tag{17}$$

Note that the two parallel hyperplanes share the same $\vec{w}$, thus the projection can be defined as:

$$p = \frac{b}{\|\vec{w}\|} \tag{18}$$

The margin can be computed as the difference between the projections of the hyperplane onto $\vec{w}$. By choosing $\vec{w}$ and b such that the decision boundary is equal to zero, as defined in Equation 16, and the margins are defined to equate to either ± 1 as in Equation 17 the margin is:

$$m = \frac{|b_2 - b_1|}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|} \tag{19}$$

Maximizing the margin is equivalent to minimizing $\|\vec{w}\|$. Since the case $\|\vec{w}\| = 0$ has to be excluded we have to minimize the square norm of $\vec{w}$, as it is derivable in all points.

$$\begin{array}{l} \text{Max} \quad m = \dfrac{2}{\|\vec{w}\|} \quad \Leftrightarrow \quad \text{Min} \quad \dfrac{1}{2}\|\vec{w}\|^2 \quad \text{over} \quad \vec{w} \\ \text{s.t.} \quad y_i(\vec{w}^T \vec{x} + b) \leq 1 \quad \forall \vec{x^i} \end{array} \tag{20}$$

This is known as the *Quadratic Optimization Problem* where a quadratic function subject to linear constraints is optimized. Given the problem being subject to an *inequality constraint* we have to expand the *Lagrangian* to obtain the *Dual Lagrangian* (indicated by $\bar{\mathcal{L}}$). For every inequality constraint one non-zero *Lagrange multiplier* α_i is introduced:

$$\begin{array}{l} \text{Max} \quad \bar{\mathcal{L}}(\alpha_i) = \sum_{i=1}^{n} \alpha_i - \dfrac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \quad \text{over} \quad \alpha \\ \left.\begin{array}{l} \text{s.t.} \quad \alpha_i \leq 0 \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{array}\right\} \forall i \end{array} \tag{21}$$

The *Dual Lagrangian* shown in Equation 21 is a *Quadratic Programming problem* this implies that a global maximum can always be found. Once we obtain an optimal value for each α_i we can obtain $\vec{w}$. All x_i with an α_i that is non-zero are support vectors. Since only the support vectors influence the solution $\vec{w}$ can be expressed just in terms of the support vectors S:

$$\vec{w} = \sum_{S} \alpha_i y_i x_s \tag{22}$$

And b can be obtained by:

$$b = y_k - \sum_{s} \alpha_s y_s \langle x_s, x_k \rangle \quad \forall \quad k \quad \text{subject to} \quad \alpha_k \leq 0 \tag{23}$$

Thus the function for classification of new data points becomes:

$$f(z) = \sum_s \alpha_s y_s \langle x_s, z \rangle + b \tag{24}$$

Where z is any new data point and the sign of the function indicates which class z is assigned to. Equation 30 allows only for linearly-separable training sets. For data that is non-linearly separable a different approach is needed.

3.7 Soft margin classification support vector machine

For situations where the training data cannot be separated without error a linear SVM may still provide a good solution for the problem by introducing a penalty function. This function is defined by the the distance ζ_i between the margin of a class and wrongly classified data is measured and then minimized (Figure 5). By introducing ζ_i, the *slack variable*, we account for noisy data thus allowing for a *soft margin* [19]. Equation 20 is expanded to incorporate the sum of the slack variables and multiplied by the hyper-parameter C:

$$\begin{aligned} &\text{Min} \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\zeta_i \quad \text{over} \quad \vec{w} \\ &\text{s.t.} \quad y_i(\vec{w}^T\vec{x_i} + b) \leq 1 - \zeta_i \quad \zeta_i \leq 0 \quad \forall i \end{aligned} \tag{25}$$

Hyper parameter C, the "trade-off" parameter controls the relative importance of maximizing the margin and minimizing the classification error in training data. The higher the value of C the smaller the margin and the lower the training error and vice versa. The solution of the soft margin is found by solving the dual problem which is expanded from Equation 21 by including an upper bound for α_i as: $0 \leq \alpha_i \leq C \; \forall i$ That way α_i must be less or equal to the 'trade-off' hyper-parameter C. In this case b is obtained by modifying Equation 23 to:

$$b = y_k(1 - \zeta_k) - \sum_s \alpha_s y_s \langle x_s, x_k \rangle \quad \forall k \quad \text{subject to} \quad \alpha_k \leq 0 \tag{26}$$

The discrimination function (30) remains however unchanged.

3.8 Non-linear support vector machine

For some data the hyperplane constructed in the input space as described by hard and soft margin (in the previous sections) don't allow for good generalization. This problem can be solved by transforming the data into a (usually) higher dimensional feature space, where the data is linearly separable. This is accomplished by choosing a by transforming the vector $\vec{x}$:

$$x \Rightarrow \phi(x) \tag{27}$$

With help of a kernel function we can exploit the fact that the linear classifier relies of the inner product between vectors. The kernel function is equivalent to an inner product in some feature space and it implicitly maps data to a high-dimensional space making an explicit calculation of each $\phi(\vec{x})$ obsolete.

$$K(\vec{x}_1, \vec{x}_2) = \langle \phi(\vec{x}_i), \phi(\vec{x}_j) \rangle \tag{28}$$

For non-linear SVMs the quadratic programming problem is solved by adapting Equation 21 to constrain the Lagrange multiplier to an upper bound:

$$\begin{aligned} &\text{Max} \quad \bar{\mathcal{L}}(\alpha_i) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j) \quad \text{over} \quad \alpha \\ &\left.\begin{aligned} &\text{s.t.} \quad 0 \leq \alpha_i \leq C \\ &\sum_{i=1}^{n}\alpha_i y_i = 0 \end{aligned}\right\} \forall i \end{aligned} \tag{29}$$

The discrimination function (30) has to be adapted as follows:

$$f(z) = \sum_s \alpha_s y_s K(x_s, z) + b \tag{30}$$

While the techniques for finding and optimizing α stay unchanged.

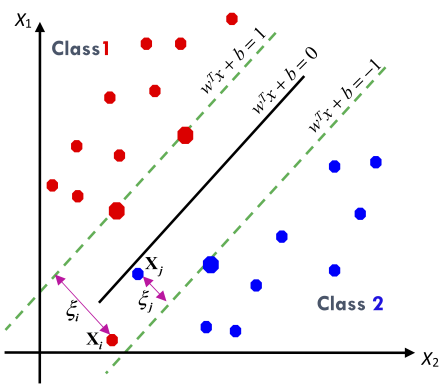

Fig. 5. Soft margin classification for SVM are algorithms allowing is adopted for non-linearly separable training data. The hyperplane is chosen by finding the largest margin (m) and minimizing the distance (ζ_i) between the margin and wrongly classified data. The bold data points indicate the support vectors. The image was adapted from Support Vector Machine: Principles, Parameters, and Applications, Gholami and Fakhari [22].

3.9 Implementation of SVM

The *LIBSVM* Version 3.24 [23] is a library for Support Vector Machines supporting multi-class classification. The LIBSVM programs take input data in the format of numerical vectors for both training and prediction. Thus we developed a python script that transforms the data from the profile windows described in section 3.5.1 into a 20 (amino acids) by 17 (window size) vector. These 340 feature vectors were subsequently matched to the class of the central residue (H, E or C) as obtained in section 3.2.1. The secondary structure was encoded as 1 for H, 2 for E and 3 for C. As we were dealing with three-class classification the one versus all others approach was implemented such that prediction is performed comparing the values of the three binary discrimination functions.

For an informed choice of the best hyperparameters C (cost) and γ of the Radial-Basis Function (RBF) kernel, we performed a minimal grid search of four different settings: (1) $c = 2, \gamma = 0.5$, $c = 2, \gamma = 2$, $c = 4, \gamma = 0.5$ and $c = 4, \gamma = 2$. We carried out 5-fold cross-validation for each of the four value pairs and calculated average Matthews Correlations Coefficient ($\overline{MCC}$) over the three classes as shown in Equation 36 described in section 4. The value pair yielding the highest $\overline{MCC}$ was then selected for training on the entire training set generating the final SVM model which was then tested on the blind set.

4 Measuring performance

For the scope of this work secondary structure prediction is carried out as a three class classification problem. We evaluated both methods by comparing the predicted secondary structures to the known labels obtained by the DSSP algorithm (see section 3.2.1).

		predicted		
		H	E	C
observed	H	p_{HH}	p_{HE}	p_{HC}
	E	p_{EH}	p_{EE}	p_{EC}
	C	p_{CH}	p_{CE}	p_{CC}

Table 1. Three-class confusion matrix where p_{ij} is the object in class i predicted in class j H, E and C indicate helix, strand and coil respectively. The correct predictions (true positives) indicated by the diagonal values of the matrix are used to calculate the three-class accuracy (Q_3 Equation 31).

		predicted	
		S	not$_s$
observed	S	c_s	u_s
	not$_s$	o_s	n_s

Table 2. Binary confusion matrix derived from Table 1. Particular secondary structure conformation are indicated by the letter S, c_s is a correct positive, o_s an over-prediction, u_s an under-prediction and n_s is a true negative.

For this purpose we implemented a python script that first generates a three-class confusion matrix from our labeled data and the predictions of each model.

Three-class accuracy is a measure of the proportion of correct secondary structure predictions with respect to the total predictions. It is computed by adding all correct predictions (p_{ss}) of each class along the diagonal of the matrix and dividing it by the total number of calls (N) given by the sum of the values of all cells of the three-class confusion matrix (see Table 1 and Equation 31).

$$Q_3 = \frac{p_{HH} + p_{EE} + p_{CC}}{N} \tag{31}$$

From the three-class confusion matrix (Table 1) we extract three binary matrices (one for each class) where we only distinguish between one particular secondary structure and all remaining classes combined, known as the "one-vs-all-other" approach. From this matrix we then calculated all binary scoring indices. Given that GOR uses an information function and the SVM trains three binary classifiers to discriminate between the three conformations we expect them to differ in their performance. For each of the three conformations we calculated Sensitivity (SEN), Positive Predictive Value (PPV), Accuracy (ACC) and Matthews Correlations Coefficient (MCC) as shown in equations 32, 33, 34 and 35.

Sensitivity, also called true positive rate, is a measure of the proportion of correct predictions (c_s) with respect to the sum of correct predictions (c_s) and under-predictions (u_s) found by the method. It is worth pointing out that this measure does not take into account false positives. The positive predictive value also known as precision, on the other hand, measures the proportion of observed positives over all positive calls (correct predictions c_p and overpredictions o_p respectively). Both indices range from 0-1.

$$SEN_s = \frac{c_s}{c_s + u_s} \tag{32}$$

$$PPV_s = \frac{c_s}{c_s + o_s} \tag{33}$$

Accuracy measures the proportion of all the correct predictions with respect to the total number of predictions thus ranging from 0-1 as well. It is however, sensitive to class imbalance.

$$ACC = \frac{c_s + n_s}{c_s + o_s + n_s + u_s} \tag{34}$$

The Matthew's correlation coefficient is a balanced measure of the observed and predicted classifications ranging between -1 (inverse prediction) and 1 (perfect prediction), where a value 0 indicates that the classifier is no better than a random predictor.

$$MCC_s = \frac{c_s * n_s - o_s * u_s}{\sqrt{(c_s + o_s) * (c_s + u_s) * (n_s + o_s) * (n_s + u_s)}} \tag{35}$$

$$\overline{MCC}_{C,\gamma} = \frac{MCC_H + MCC_E + MCC_C}{3} \tag{36}$$

As mentioned in section 2, five-fold cross-validation was carried out for both methods to assess each method. In a final step we generated a model from the entire training set which was then tested on our blind set to reproduce a "never-seen-before" condition. By doing so we were able to asses both methods' abilities for generalizing to an independent and unknown set of data. This is useful not only to assess how well the model generalizes but also serves to detect overfitting.

The entire training set of 1204 sequences was split for each validation run such that 80% was used for training while the remaining 20% was used for testing. For the hold-out validation both methods were used for training on all 1204 sequences, while the blindest (150 sequences) was used for testing. Then we computed the standard error for each of the averaged scoring indices mentioned in equations 32, 33, 35 and 31.

$$SE = \frac{\sigma}{\sqrt{n}} \tag{37}$$

Where σ is the standard deviation and n is the number of samples.

5 Results

Cross-validation of both methods and minimal grid search for choosing hyperparameters of the SVM

Five-fold cross-validation (CV) was carried out on both GOR and SVM methods where we evaluated each model's ability to correctly predict the data by comparing the predictions to the labels previously obtained from the DSSP algorithm (section 3.2.1). Analyzing Table 3 we can appreciate how lowering the γ parameter, thus increasing the number of support vectors, improves the performance of the model. Changes in C while keeping the γ at 0.5, on the contrary yielded comparable outcomes in particular when looking at the MCC and Q_3. This is justified by a higher number of support vectors rendering a more precise model.

When evaluating the performance measures of the grid search we found that the SVM model with cost parameter C = 2 and a γ = 0.5 performed best across the scoring indexes and achieved a three-class accuracy (Q_3) of 0.706 $\pm$ 0.003 (Table 3 column 1). These parameters were then selected to train the final SVM model on the entire JPred4 training set and perform a subsequent hold-out validation on the blind set. As the GOR method had no hyperparameters to be trained we performed no grid search but simply five-fold cross-validation which achieved a three class accuracy of

	SVM grid search							
	C = 2 γ = 0.5		**C = 2 γ = 2**		**C = 4 γ = 0.5**		**C = 4 γ = 2**	
H	ACC	0.846 ± 0.004	ACC	0.685 ± 0.006	ACC	0.843 ± 0.004	ACC	0.684 ± 0.006
	SEN	0.698 ± 0.01	SEN	0.129 ± 0.005	SEN	0.691 ± 0.01	SEN	0.127 ± 0.005
	PPV	0.842 ± 0.005	PPV	0.878 ± 0.013	PPV	0.838 ± 0.005	PPV	0.878 ± 0.012
	MCC	0.657 ± 0.007	MCC	0.256 ± 0.008	MCC	0.65 ± 0.007	MCC	0.255 ± 0.007
E	ACC	0.843 ± 0.004	ACC	0.783 ± 0.006	ACC	0.841 ± 0.004	ACC	0.782 ± 0.006
	SEN	0.4 ± 0.01	SEN	0.02 ± 0.004	SEN	0.389 ± 0.01	SEN	0.02 ± 0.004
	PPV	0.785 ± 0.009	PPV	0.813 ± 0.033	PPV	0.783 ± 0.009	PPV	0.807 ± 0.034
	MCC	0.484 ± 0.007	MCC	0.105 ± 0.01	MCC	0.475 ± 0.006	MCC	0.103 ± 0.01
C	ACC	0.723 ± 0.003	ACC	0.469 ± 0.004	ACC	0.717 ± 0.003	ACC	0.468 ± 0.003
	SEN	0.873 ± 0.003	SEN	0.985 ± 0.001	SEN	0.871 ± 0.003	SEN	0.985 ± 0.001
	PPV	0.624 ± 0.004	PPV	0.443 ± 0.003	PPV	0.618 ± 0.004	PPV	0.443 ± 0.003
	MCC	0.488 ± 0.004	MCC	0.157 ± 0.005	MCC	0.478 ± 0.004	MCC	0.156 ± 0.005
	Q3	0.706 ± 0.003	Q3	0.468 ± 0.004	Q3	0.7 ± 0.003	Q3	0.467 ± 0.003

Table 3. Summary of averaged performance measures of grid search. Five fold cross-validation was done with each set of parameters C and γ note that best performance was achieved with C = 2 and γ = 0.5.

Q_3 0.647 ± 0.001 (Table 4). The standard error was between 0.001 and 0.012.

Hold-out validation

For hold-out validation one GOR and one SVM model were trained on the entire JPred4 set and subsequently tested on our curated blind test set. This way we were able to assess how well each model generalizes to unknown data. Comparing the scores (Table 5) we found that SVM overall outperforms GOR with a three-class accuracy (Q_3) of 0.693 while GOR achieved only 0.637. It is however worth noting that the MCC_E 0.477 was very similar to its counterpart in GOR (0.425) and that SVM's SEN_E was only 0.45 while GOR achieved 0.658. While both models do well in predicting helices, SVM showed a much higher MCC_H with 0.628 while GOR had reached only 0.486. Overall the results of the blind test compare well to the ones obtained by cross-validation indicating that both models are able to generalize to data that is dissimilar to the training set.

6 Discussion

Experimental methods for solving protein structure are the gold standard for precise and reliable protein annotation. However, they are costly and time consuming compared to the relatively cheap sequencing methods. Hence, developing and improving methods for *in silico* secondary structure prediction is necessary. Here we have implemented two methods, namely GOR and SVM and analyzed their performance to uncover their strengths and limitations in tackling this problem. GOR is based on Bayesian statistics and information theory while it also considers the influence of local residue context on the secondary structure. It relies however on an assumption of statistical independence of the residues in the window which is not valid from a biological point of view. The method has a short training time of only a few minutes, it is how ever less accurate than SVM due to the aforementioned limitation. The SVM, a third generation method has the ability to consider large feature spaces by making use of kernel functions. It uses more complex computations for finding a decision boundary which leads to longer training times but also better performance.

We found in both methods that predictions of coils and strands scored lower compared to helices. This might be due to coils and strands being subject to interactions with distant atoms which cannot be captured by a window of 17 residues only. In addition, the slightly skewed distribution of classes within the training set where strands make up only 22.2% of the data (Figure 6) may influenced the low performance in this class. Another limitation was addressed in 3.2.1; eight secondary structure motives where mapped into only three classes (H,C and E) rendering the categories less biologically meaningful. By implementing both methods again while including constraints that e.g. dictate a minimum number of four residues to form a helix, and using a more balanced dataset may achieve better results.

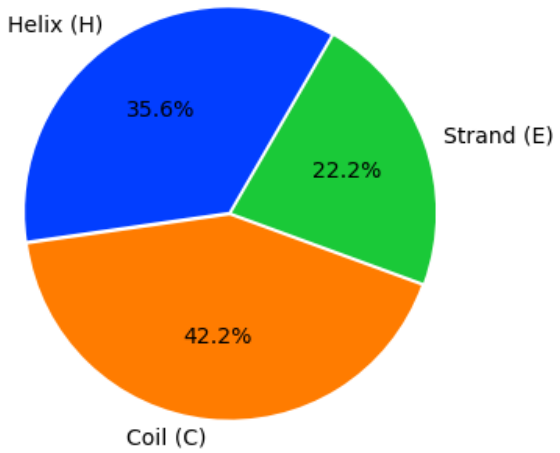

Fig. 6. The figure shows the percentage abundance of helices (H), strands (E) and coils (C) in the training dataset.

7 Conclusion

GOR was outperformed by the SVM in both cross-validation and hold-out validation where GOR yielded a three class accuracy of only Q_3 0.647 ± 0.001 and Q_3 0.637 respectively, while SVM was Q_3 of 0.706 ± 0.003 and 0.693 ± 0.003. The comparison of the overall results for cross-validation and final model of both methods resulted in a low standard error when compared among each other. The highest SE was 0.0012, indicating that the JPred4 is a balanced set. and it performed better across all scoring indices. Five-fold cross-validation was carried out on both methods then a last validation was performed by training of both methods on the entire dataset (JPred4 with 1204 sequences) and testing each of the models on a blind testing set holding 150 sequences. Five-fold cross-validation and hold-out testing were comparable, suggesting that the models generalize well to unseen data. Overall prediction of SVM was better with a Q_3 of

0.693 compared to the Q_3 of 0.637 achieved by GOR. While the SVM yielded high sensitivity in detecting helix (H) and coil (C) it scored lower for β-strands E which may be explained by β-strands being subject to long-range residue interactions which may pose limitations to accurate secondary structure prediction [24]. When looking at the results for α-helices, governed by short range interactions we can see that the predictions score higher across both methods and indices.

In this project we implemented and scrutinized two methods for secondary structure prediction. We included evolutionary information in the form of sequence profiles and made use of profile windows for taking account of the influence of the local residue context on secondary structure conformation. The GOR method makes use of information theory and Bayesian statistics while the support vector machine (SVM) is a Both methods showed the ability to generalize to unseen datasets, as we confirmed with testing against a blind dataset whose labels were previously obtained by the DSSP algorithm (section 3.2.1).

	Cross-validation			
	GOR		**SVM**	
H	ACC	0.776 ± 0.001	ACC	0.846 ± 0.004
	SEN	0.783 ± 0.003	SEN	0.698 ± 0.01
	PPV	0.66 ± 0.007	PPV	0.842 ± 0.005
	MCC	0.539 ± 0.001	MCC	0.657 ± 0.007
E	ACC	0.786 ± 0.001	ACC	0.843 ± 0.004
	SEN	0.676 ± 0.001	SEN	0.4 ± 0.01
	PPV	0.505 ± 0.012	PPV	0.785 ± 0.009
	MCC	0.447 ± 0.005	MCC	0.484 ± 0.007
C	ACC	0.732 ± 0.001	ACC	0.723 ± 0.003
	SEN	0.517 ± 0.003	SEN	0.873 ± 0.003
	PPV	0.774 ± 0.002	PPV	0.624 ± 0.004
	MCC	0.446 ± 0.002	MCC	0.488 ± 0.004
	Q_3	0.647 ± 0.001	Q_3	0.706 ± 0.03

Table 4. Comparison of averaged scoring indices of five-fold cross-validation of GOR model and highest scoring SVM model (C = 2, γ = 0.5).

	Hold-out validation			
	GOR		**SVM**	
H	ACC	0.749	ACC	0.825
	SEN	0.734	SEN	0.669
	PPV	0.66	PPV	0.85
	MCC	0.486	MCC	0.628
C	ACC	0.738	ACC	0.715
	SEN	0.526	SEN	0.865
	PPV	0.71	PPV	0.584
	MCC	0.425	MCC	0.477
E	ACC	0.787	ACC	0.846
	SEN	0.658	SEN	0.45
	PPV	0.53	PPV	0.793
	MCC	0.45	MCC	0.518
	Q_3	0.637	Q_3	0.693

Table 5. Hold-out testing: Comparison of scoring indices obtained from validation on blind set after training both models on the entire JPred4 dataset parameters used for training SVM: C = 2, γ = 0.5

References

[1]RCSB PDB - Holdings Report. URL: http://www.rcsb.org/pdb/statistics/holdings.do.

[2]UniProtKB/TrEMBL 2020_06. URL: https://www.uniprot.org/statistics/TrEMBL.

[3]Peter Y. Chou and Gerald D. Fasman. Prediction of protein conformation. *Biochemistry*, 13(2):222–245, January 1974. Publisher: American Chemical Society. doi:10.1021/bi00699a002.

[4]J. Garnier, D. J. Osguthorpe, and B. Robson. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *Journal of Molecular Biology*, 120(1):97–120, March 1978. URL: http://www.sciencedirect.com/science/article/pii/0022283678902978, doi:10.1016/0022-2836(78)90297-8.

[5]D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195–202, September 1999. doi:10.1006/jmbi.1999.3091.

[6]Burkhard Rost, Chris Sander, and Reinhard Schneider. Redefining the goals of protein secondary structure prediction. *Journal of Molecular Biology*, 235(1):13–26, January 1994. URL: http://www.sciencedirect.com/science/article/pii/S0022283605800075, doi:10.1016/S0022-2836(05)80007-5.

[7]Alexey Drozdetskiy, Christian Cole, James Procter, and Geoffrey J. Barton. JPred4: a protein secondary structure prediction server. *Nucleic Acids Research*, 43(W1):W389–W394, July 2015. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkv332, doi:10.1093/nar/gkv332.

[8]Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, December 1983. URL: http://doi.wiley.com/10.1002/bip.360221211, doi:10.1002/bip.360221211.

[9]Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, September 1997. URL: https://academic.oup.com/nar/article/25/17/3389/1061651, doi:10.1093/nar/25.17.3389.

[10]Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, January 2000. URL: https://academic.oup.com/nar/article/28/1/235/2384399, doi:10.1093/nar/28.1.235.

[11]RCSB Protein Data Bank. RCSB PDB. Library Catalog: www.rcsb.org. URL: https://www.rcsb.org/.

[12]Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990. URL: http://www.sciencedirect.com/science/article/pii/S0022283605803602, doi:10.1016/S0022-2836(05)80360-2.

[13]National Center for Biotechnology Information (NCBI) Documentation of the BLASTCLUST-algorithm. BlastClust. URL: https://ftp.ncbi.nih.gov/blast/documents/blastclust.html.

[14]UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515, January 2019. URL: https://academic.oup.com/nar/article/47/D1/D506/5160987, doi:10.1093/nar/gky1049.

[15]UniProt. URL: https://www.uniprot.org/.

[16]UniProtKB/Swiss-Prot Release 2020_06 statistics. URL: https://web.expasy.org/docs/relnotes/relstat.html.

[17]Antonina Andreeva, Eugene Kulesha, Julian Gough, and Alexey G Murzin. The SCOP database in 2020: expanded classification of representative family and superfamily domains of known protein structures. *Nucleic Acids Research*, 48(D1):D376–D382, January 2020. doi:10.1093/nar/gkz1064.

[18]Jean Garnier, Jean-François Gibrat, and Barry Robson. [32] GOR method for predicting protein secondary structure from amino acid sequence. In *Methods in Enzymology*, volume 266 of *Computer Methods for Macromolecular Sequence Analysis*, pages 540–553. Academic Press, January 1996. URL: http://www.sciencedirect.com/science/article/pii/S0076687996660340, doi:10.1016/S0076-6879(96)66034-0.

[19]Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995. doi:10.1007/BF00994018.

[20]Support-vector machine, December 2020. Page Version ID: 997327362. URL: https://en.wikipedia.org/w/index.php?title=Support-vector$_machineoldid$ = 997327362.

[21]S. Hua and Z. Sun. A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. *Journal of Molecular Biology*, 308(2):397–407, April 2001. doi:10.1006/jmbi.2001.4580.

[22]Raoof Gholami and Nikoo Fakhari. Chapter 27 - Support Vector Machine: Principles, Parameters, and Applications. In Pijush Samui, Sanjiban Sekhar, and Valentina E. Balas, editors, *Handbook of Neural Computation*, pages 515–535. Academic Press, January 2017. URL: http://www.sciencedirect.com/science/article/pii/B9780128113189000272, doi:10.1016/B978-0-12-811318-9.00027-2.

[23]Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

[24]Daisuke Kihara. The effect of long-range interactions on the secondary structure formation of proteins. *Protein Science: A Publication of the Protein Society*, 14(8):1955–1963, August 2005. doi:10.1110/ps.051479505.