Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Project Report

Project Report

Secondary structure prediction: a probabilistic and machine learning showdown

Roncelli Stefano [1,*]

[1] International Master in Bioinformatics, Univerisity of Bologna, Bologna

[*] To whom correspondence should be addressed.

Abstract

Motivation: Protein structure prediction is of paramount importance in many fields. Through the decades many breakthougts allowed to extend the boundaries of sequence based prediction further than the generation before, but still nowhere near structure based methods. Analysis of the most significant innovations is essential in deriving a knowledgebase to devise even powerful methods.
Results: Powerful models allow to model non-linearity. However, the risk of overfitting is always plaguing complex predictors. More simple probabilistic models can yield comparable results if used in an information-rich context.
Contact:stefano.roncelli@studio.unibo.it
Supplementary information: Supplementary data are available at github.

1 Introduction

Prediction and pattern recognition is one of the core topics in all bioinformatic fields. Specifically, secondary structure prediction was first attempted around 60 years ago with analysis on the Helix-Coil transition (Zimm and Bragg, 1959) (Lifson and Roig, 1961), but the lack of available structures hindered the progress. It was more than a decade later that the first probabilistic models set the start of what we now know as 'first generation' of sequence-based secondary structure prediction. Their somewhat respectable performances (if we frame the epoque) were limited by the fact that they modeled only the statical propensities on the single residues for the prediction of the secondary structure. It was only with the second generation that the context (Garnier *et al.*, 1978) was taken into account for more accurate predictions, which were verging 60% accuracy. This context takes the form of a sliding window centered on the residue for which the effort of predicting its structure resides, but the lack of available known sequences resulted again in difficult estimation of the parameters. Only when the numbers finally caught up, the third generation models were able to harness the valuable data provided by evolution, in the form of sequence profiles derived from multiple sequence alignments, and breached the 70% accuracy threshold. Today, state-of-the-art pushed the threshold even higher (**?**), even though the theoretical limit of 88% has not yet been achieved (Rost, 2001)(Russell and Barton, 1993). Thanks to the independent advancements in machine learning, most notably neural networks (Drozdetskiy *et al.*, 2015)(Heffernan *et al.*, 2018) and Support Vector Machines (Kieslich *et al.*, 2016), the boundary stalls at 82% accuracy.

Here, we propose a side by side comparison between the old and the new: a performance review between a model based on the time-proven Garnier-Osguthorpe-Robson method (Garnier *et al.*, 1978) and a much more modern estimator based on Support Vector Machines (Cortes and Vapnik, 1995). All steps from dataset preparation, to profile generation, to scoring have been devised in order to compare their performances on equal grounds. Such datasets are designed to represent as accurately as possible the protein space, albeit with some simplifications. This review is motivated by the curiosity to asses the peculiarities and differences of the two approaches.

2 Methods

2.1 Training set preparation

The training set contains 1348 protein primary and secondary sequences in fasta format as available from JPred4 (Drozdetskiy *et al.*, 2015). For correct class identification, the *de facto* golden standard is the labeling of residues based on the three dimensional structure, in the form of a structure file such as a macro molecular Christallographic Information File (mmCIF). This information retrieval of the "true" classes of a chain is usually achieved by Define Secondary Structure of Proteins (DSSP)(Kabsch and Sander, 1983) or STRuctural IDEntification (STRIDE)(Frishman and Argos, 1995) algorithms.

Proteins form a wide variety of secondary structures, however, JPred remaps the eight DSSP's categories with a three state notation: α-helices (H) are labeled as H, beta strands (E) and residues in β-bridge (B) as E, and everything else as C. Hence, all the possible structures can be either labeled as helix (H), extended strand (E) or coil (C). It is important to note that whereas the H and E classes maintain a certain degree of homogeneity, the C category is more spurious in nature, since it groups together many, widely different, secondary structure conformations. To be precise, such mapping is a methodological mistake, because afterwards it is no longer possible to distinguish well defined structures (the turns, the 3_{10} helixes and so on) from the errors and limitations of the program (the " " blanks), since they are all collapsed into the C class. Odd as it may seem, this remapping has the benefit of denoising the two most emblematic classes, alas, at the expense of the Coil category. In this framework we relate the C category as "not H or E": whatever is not helix or strand is swept under the rug and labeled C. We will see later how this influences the estimators classification performance.

2.2 Test set

The JPred dataset comes with a test set, but we decided to generate a new one *de novo*, following the approach of the original JPred test set. The RCSB Protein Data Bank (PDB)(rcsb.org) (Berman, 2000) was queried in order to retrieve all the proteins that met the following criteria:

- Deposit date after Jan, 2015, *i.e.* after the release of JPred4.
- Resolution above 2.5Å: high quality data is important in secondary structure determination when using atomic coodinates.
- Chain length between 50 and 300 residues.
- No mutations or artifacts as they may bias the estimators performance.

This raw dataset, however, is highly redundant. Many entries are similar to others in this set or to ones already present in the training set and must be removed. As a cutoff threshold, we kept only sequences that share less than 30% sequence identity to other proteins in the test set. This internal redundancy filtering was achieved by clusterization with MMseqsS (Steinegger and Söding, 2017, 2018), with a coverage threshold of 50%. To identify the similar proteins between train and test set, the latter was aligned with the former with BLASTp (Altschul *et al.*, 1990). To avoid non-significant alignments, an E-value threshold of 0.01 was enforced. Again, protein recognized to share more than 30% sequence identity to the training set were purged from the test. For the generation of the secondary structure annotation we used the same standard as JPred: DSSP. To be consinstent with JPred's pipeline, the same reduced mapping was used as previously discussed. The sole difference with the original test set was the elimination of entries whose secondary structure, as determined by DSSP, was shorter than their respective primary structure. Whereas JPred's approach allows for no more than 9 mismatched between primary and secondary structure, we opted for a no tolerance rule.

2.3 Profile generation

Using only plain sequences is possible, but complementing it with evolutionary information allows us to achieve much more accurate predictions at virtually no cost. Following this rationale, each entry in both sets was converted to its relative sequence profile by means of Psiblast (Altschul, 1997) against the SwissProt as of `December 2020` (The UniProt Consortium, 2018) with the following parameters: e-value set to 0.01, number of descriptions and alignments both set to 10000 and number of iterations equal to 3. Since the GOR model uses a probabilistic framework to operate, the probabilities in the sequence profiles were normalized accordingly. For each position in a profile, two windows were extracted, one of size 1 and another of size 17, both centered on the residue of interest. To avoid inconsistent window length sizes, for the positions at both ends of a profile an adequate amount of padding was introduced, with all values set to 0. Since all the estimators implemented, by design, work on a vectorized input, each window was then reshaped accordingly and appended to the final dataset. Through the filtering process, the candidate test set was enormously reduced in size: from more than 12000 candidates we obtained a total of 453 sequence profiles. To mimic the size of the original test set from JPred, 150 profiles were randomly sampled until their residue and secondary structure composition was comparable to the training set with a tolerance threshold of less than 2% for secondary structure differences. Subsamples that did not met the tolerance criteria were discarded and the whole test set resampled. This ensures that the test set is a valid representative of both the protein space and the training set, so that the performances of the estimators are as unbiased as possible.

2.4 GOR method

Theorical backround

The Garnier-Osguthorpe-Robson (Garnier *et al.*, 1978) method is a information theory model for secondary structure prediction, and is based on the statical propensities of residues for secondary structure conformations. It is possible to devise an information function I that calculates the probability a residue R to be in the conformation S, such that:

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

Where $P(S|R)$ is the conditional probability of observing residue R in conformation S, and $P(S)$ is the probability of conformation S. Using the chain rule:

$$I(S;R) = \log \frac{P(R,S)}{P(S)P(R)} \tag{2}$$

If we want (and we should) also take into consideration the *context* of each residue, that is, how much nearby residues influence the structural conformation of a given R:

$$I(S,R_{-d},...,R_d) = \log \frac{P(S|R_{-d},...,R_d)}{P(S)P(R_{-d},...,R_d)} \tag{3}$$

Where d is the number of residues taken into consideration before and after the central residue. Training the model therefore consist in the correct estimation of the probabilities involved: $P(S)$ and $P(R_{-d},...,R_d)$ are easy to estimate, since for a large enough traning set they can be approximated to their correspondent frequencies. $P(S|R_{-d},...,R_d)$, however, is much harder to estimate reliably. First and foremost, because it is very computationally challenging due to the exponential increase for wider window sizes. Second, a good approximation of the true probabilities is very depending of the size of the training set. For these reasons two simplifying assumptions have to be introduced: the limitation of the window size and the statistical independence of nearby residues. It is important to note that these assumption are not biologically justifiable, but they help to reshape an untractable problem into a more approachable one. The size of the window is usually 17, which means:

$$P(S|R_{-8},...,R_8) = \prod_{k=-8}^{8} P(R_k) \tag{4}$$

So we can rewrite (3) as:

$$I(S,R_{-8},...,R_8) = \prod_{k=-8}^{8} \log \frac{P(R_k,S)}{P(S)P(R_k)} \tag{5}$$

For a large enough training set the probabilities can be approximated to their correspondent frequencies. Usually the trained model takes the form of an information table or matrix, which is filled according to the frequencies derived from the training set. In this probabilistic framework, predicting the secondary structure of an unknown sequence S^* means labeling each position with the highest class probability, which is another level over-simplification of the model:

$$S^* = \arg\max_S I(S, R_{-8}, ..., R_8) \tag{6}$$

Implementation

To test for the context contribution, two models were trained: one with the window size set to 1 (*i.e.* no context) and another with the window set to 17. As previously discussed, our practical implementation revolves around the use of sequence profiles to boost the estimator performance. Especially for the GOR, which makes some hindering simplifying assumptions, this proves to be a significant boon to the information content of the training set, which somewhat counterbalances the information loss caused by the model assumptions. Our implementation of the GOR method starts from the appropriate reshaping of the vectorized sample of the training set. The information is then stored in an intuitive manner: specifically, a dictionary of NumPy arrays (Harris *et al.*, 2020). Training the model proceeds in a top-down fashion, iteratively reshaping all the windows of the training dataset. After that, the appropriate parameters are estimated following the just discussed rationale. Prediction is as straight-forward as fitting the estimator: for each profile window the estimation of the probabilities is computed from the fitted estimator. In the event of ties, such as when a specific window contains no signal, the estimator defaults to C class prediction. This bias is justified by the reduced mapping introduced for the classes.

2.5 SVM model

Theoretical background

Support vector machines (Cortes and Vapnik, 1995) are a class of supervised learning models used both for classification and regression tasks. At their core reside the computation of a maximum margin hyperplane that is able to separate two classes. An hyperplane is a N-1 dimensional manifold, where N is the dimension of the feature space, an can be formulated as:

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{7}$$

Where $\mathbf{w}$ is a vector perpendicular to the hyperplane, $\mathbf{x}$ any point that lies on the hyperplane, and b is a bias to unbound hyperplanes from passing through the origin. By definition, such hyperplane divides the space in two subspaces, one for each class. Formally:

$$t_n(\mathbf{w}^T\mathbf{x} + b) \geq 1 \tag{8}$$

Where t_n is the target variable (the class). Equation (8) states all the points must lie on correct side of the subspace. The margin ρ can be seen as the distance between two parallel hyperplanes and is:

$$\rho = \frac{2}{\|\mathbf{w}\|} \tag{9}$$

The problem of minimazation of the margin hence can be expressed as:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{10}$$

We can therefore formulate the original problem as the minimization of (9), subject to (8), which is a constrained optimization problem. The classical approach involves the use Lagrange multipliers in conjunction with Karush-Kuhn-Tucker (Karush, 1939) conditions:

$$\mathcal{L}(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_n a_n[1 - t_n(\mathbf{w}^T\mathbf{x} + b)] \tag{11}$$

(11) is function of $\mathbf{w}$, b and $\mathbf{a}$, but it can be rewritten as a function of $\mathbf{a}$, the *dual lagrangian*:

$$\widetilde{\mathcal{L}}(\mathbf{a}) = \sum_n a_n - \frac{1}{2}\sum_n\sum_m a_n a_m t_n t_m \langle \mathbf{x}_n \mathbf{x}_m \rangle \tag{12}$$

subject to

$$\sum_n a_n t_n = 0 \qquad\qquad a_n \geq 0 \tag{13}$$

Which is a quadratic optimization problem of a convex function, and therefore allows to find a global optimum. So far, this framework allows only to tackle linearly separable problems. To extend SVMs beyond linear separation, two methods have been successfully implemented. The first, easier, approach is to allow some degree of misclassification of the model by relaxing the margins. This is achieved with the introduction of ξ, a *slack variable*, that changes (8) and (9) in:

$$t_n y(\mathbf{x}) \geq 1 - \xi_n \qquad\qquad C\sum_n \xi_n + \frac{1}{2}\|\mathbf{w}\|^2 \tag{14}$$

This means that now it is possible to allow for some n points of the sample to fall within the margin, with a distance equal to ξ_n. The regularization parameter C controls the trade-off between misclassification rate and classification error.
Secondly, we can apply a fixed space transformation $\phi(\mathbf{X})$ that remaps the original, non-linearly separable, space into a new feature space where the classes are linearly separable. The insight that two input vectors appear in (12) as a scalar product however avoids the explicit computation of $\phi(\mathbf{X})$. We can define a *kernel function* K such that:

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}') \tag{15}$$

A linearly inseparable problem in the original input space can be linearly separated in a higher dimension feature space, with an appropriate Kernel function:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}' \tag{16}$$
$$K(\mathbf{x}, \mathbf{x}') = (\gamma\mathbf{x}^T\mathbf{x}' + c)^d \tag{17}$$
$$K(\mathbf{x}, \mathbf{x}') = \exp\Big(-\gamma\big\|\mathbf{x} - \mathbf{x}'\big\|^2\Big) \tag{18}$$
$$K(\mathbf{x}, \mathbf{x}') = \tanh\Big(\gamma\mathbf{x}^T\mathbf{x}' + c\Big) \tag{19}$$

Trough time, several kernel functions have been successfully devised; the most famous are the linear kernel (16), Radial Basis Function (RBF) (18), sigmoid (19) and polynomial (17). The choice of the kernel, along with the value of all the other hyperparameters directly influences the generalization performance of the model.

By definition, SVMs are unable to perform multiclass classification, but it is still possible to solve such problem by training multiple classifiers that jointly perform the task. The decision function of this ensemble of classifiers can be either One-vs-One (OvO) or One-vs-Rest (OvR), depending on whether each classifier performs the separation between two distinct classes, ignoring all the others (OvO) or merges all the classes but one in a single one (OvR). The choice of the decision function shape is also very problem dependent, but it is worth noting that the OvR involves computation of a number of hyperplanes equal to the number of classes, whereas the OvO scales quadratically.

Evaluation of models

Training the GOR model is not very computationally expensive, since only 1023 parameters have to be estimated and their calculation involve only basic operations. Estimation of the optimal parameters proceeds in a top-down fashion: once any 17 by 20 window has updated the information table, it's never looked back at again. Albeit unnecessary for the purpose of hyperparameters optimization, the GOR model was evaluated with a 5-fold cross validation. The validation procedure did however provide useful insights on the model's sensitivity with respect to the training subsample.

On the other hand, training the SVM on the staggering size of the training set, which comprises of 340 features and 197610 samples proved to be somewhat problematic in a practical amount of time. Unfortunately, the nature of SVMs doesn't allow a single training or testing instance to work in a multi-threading environment, which means that the training time is constrained by the single core performance of the machine. To be fair, cross validation could, -in principle- be conducted in a parallelized pipeline. However, the necessity of load in memory each split of the training data, along with the need to have at least half as much cores as training folds to significantly reduce training time, crossed out this possibility. Machines equipped with core-dense processors are hard to come by, and even when they are, they are seriously expensive. Fortunately, it is still possible to accelerate the workload using a Graphical Processor Unit (GPU). Our approach is based on the computational Python library thundersvm (Wen *et al.*, 2018), which efficiently harnesses the computational workforce of the CUDA (Nickolls *et al.*, 2008) cores available, decreasing the time needed for both fitting and predicting by orders of magnitude.

This allowed us to test a wide variety of hyper-parameters with the whole training set:

- C, the regularization parameter was either 4 or 2
- γ, the coefficient of some kernels, either 2 or 0.5.
- All the available kernels: linear, RBF, polynomial and sigmoid.

For a total of 14 possible combinations. In principle, a solution involving training on subsample of the set to conduct a deeper grid search may sound promising; but the knowledge about state-of-the-art performances along with the importance of the training size made us head for the GPU acceleration pipeline. The complete grid search was hence conducted on a Microsoft Azure Virtual Machine type NV6, that comes equipped with half a NVIDIA Tesla M60, and took less than 20 hours. The best found parameters were used to retrain the estimator on the whole dataset, which was then evaluated on the test set.

Evaluation metrics

For the cross validation and grid search procedures the scoring metric used was the Matthews Correlation Coefficient (Matthews, 1975):

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{20}$$

The MCC is a solid score, resilient to class imbalance, that takes into account all classifications an misclassifications of an estimator: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

For evaluation of the multiclass problem, we used the average of the three OvR MCCs:

$$\text{MCC}_{tot} = \frac{\text{MCC}_H \text{MCC}_E \text{MCC}_C}{3} \tag{21}$$

For the grid search, the best estimator therefore was the one that maximized (21).

For the generalization capabilities of both the GOR model and the SVM we used the following metrics: accuracy (ACC), precision (PRE), recall (REC), as well as (21).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{22}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{23}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{24}$$

To address the performances on a per-class basis we included also all the OvR averages for each class. Finally, we reported also the multiclass accuracy (Q_3), which is the *de facto* standard for secondary structure prediction:

$$Q_3 = \frac{\sum_i n_{ii}}{\sum_i \sum_{j \neq i} n_{ij} + \sum_i n_{ii}} \tag{25}$$

Where relevant, average scores reported also the standard error, calculated as follows:

$$\text{SE} = \frac{\sigma}{\sqrt{n}} \tag{26}$$

Where σ is the standard deviation.

3 Results

Dataset composition

Great effort was taken in order to mimic the features of the original test size from JPred as close as possible. The distribution of sequence length (Figure S1) along with the relative abundance of residues per secondary structure (Figure 1) were consistent between training, testing and Uniprot Swiss-Prot. Most notably, the three distributions all have the same shape, with a peak at around 100 residues and a more pronounced decay starting just before 300 residues. Figure 1 shows how, besides a modest sampling noise, train and test sets are comparable to one another. The proportions of classes Figure 3 between train and test sets are very comparable, thanks to the introduction of iterative resampling discussed above.

GOR

Both models performed as expected, with a respective Q_3 accuracy of 0.50 and 0.63 (page 6). The difference between the results with when no context was given (Table S4) and when the window was set to 17 (Table S5) clarifies the importance and influence of nearby residues in determining the secondary structure. The 5-fold cross validation results of the contextualized model (Table S1) clearly dismisses the possibility of any particular skew that may have been introduced by the training subsampling. Performance-wise, the MCC reported both during cross validation and testing (Table S5) once again confirms the adequate capabilities of the GOR method to predict secondary structures despite its simplifying assumptions. All the other metrics tell a similar story, with the notable exception of the precision of the Coil conformation. That is, when predicting the C class the estimator has the highest confidence (high precision), even though it rarely does so (low recall). This can be ascribed to a multitude of variables: perhaps, a fraction of the collapsed secondary structures have a very distinctive signal that triggers the estimator, either by lowering the probabilities for H and E or by incresing the one for C (or both). This hypothesis is in line with the nature of the hybrid nature of the C class. Without additional data we can only speculate.

The more structured nature of the helix is reflected also in the scores: it is consistently better predicted among all three. In particular, the high recall and precision for the helix is a direct consequence of this structure's signature. The fact that most of the information for the helix tends to fall within the window, again, is quite evident. The same applies for the Elongated class, but in reverse. It is known the β-bridges are more than

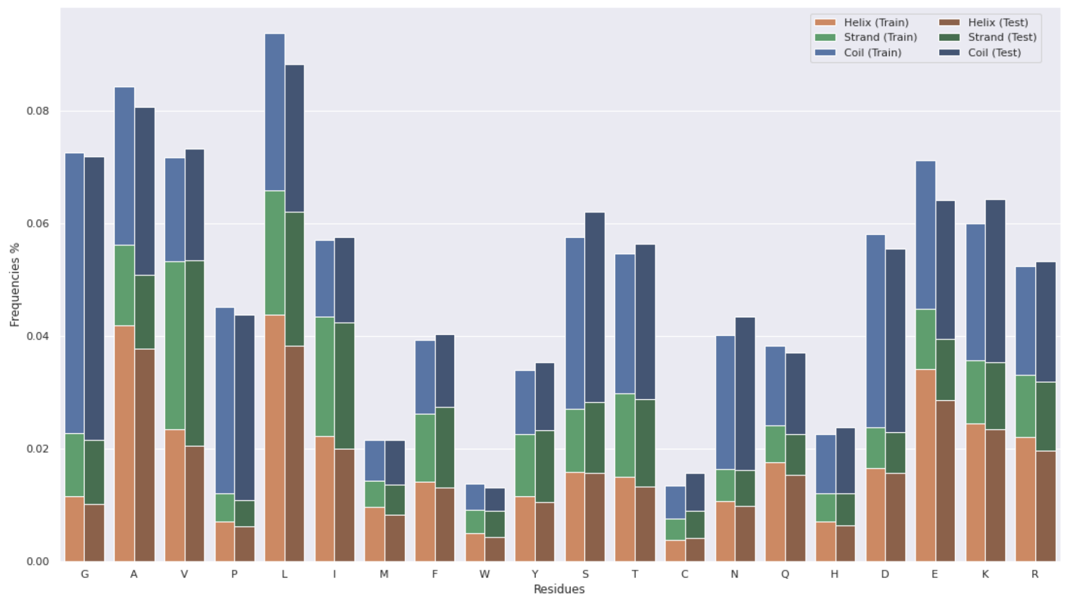

Fig. 1. Secondary structure composition on a per-residue level between train and test.

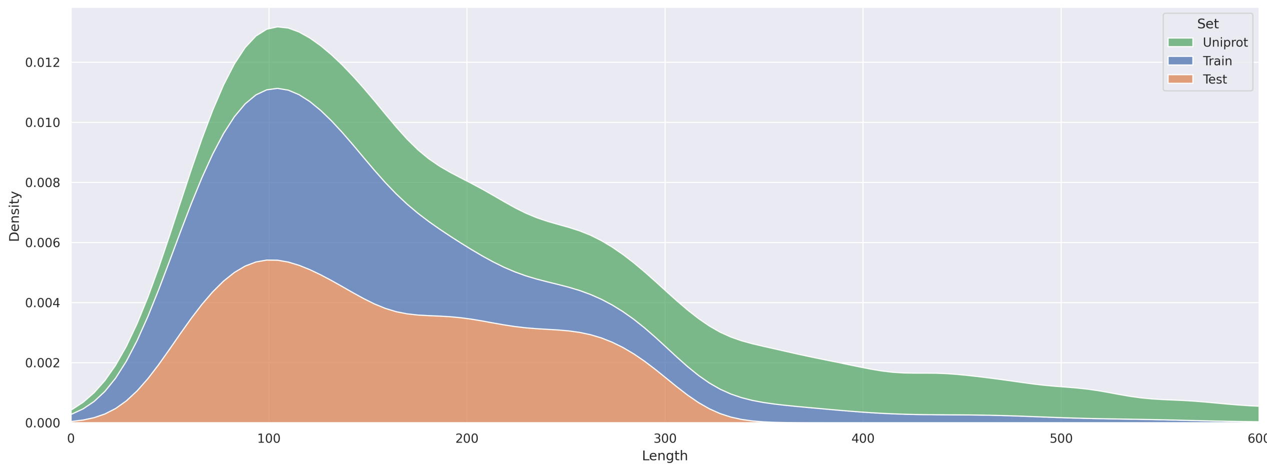

Fig. 2. Sequence length comparison between train, test and Uniprot Swiss-Prot datasets.

often determined by residues quite far apart, which clearly tend not to be represented inside the window.

To no surprise, the worst overall performance belongs to the Coil class. The reason for this outcome can be pinpointed to both the reduced mapping representation, which contributes to increase noise of its information matrix; and to the intrinsic unstructured nature of this class, which together lead wrong parameter estimation.

SVM

Train (Table S2) and test (Table S3) scores for the grid search give valuable insights of the nuances between different kernels. The polynomial and RBF kernels manage to achieve train scores up to 99% across all classes, only to be significantly downsized when presented with new samples. Nevertheless, their overfitted performance still managed to surpass the GOR model. On the other hand, the linear and sigmoidal kernels show little differences between training and testing performance. The latter, however, performed significantly more poorly than all the others; even worse, both train and test performances fell well below what was showed for the GOR model. Moreover, a significant drop in score was observed for higher γ values, which once more hints that a non-linear approach may not the right way to proceed. All in all, it is difficult to elect a clear winner, since all the non-dismal kernels scored around 0.55 MCC. But looking solely at the numbers, the best estimator chosen to be retrained on the whole dataset was the linear kernel.

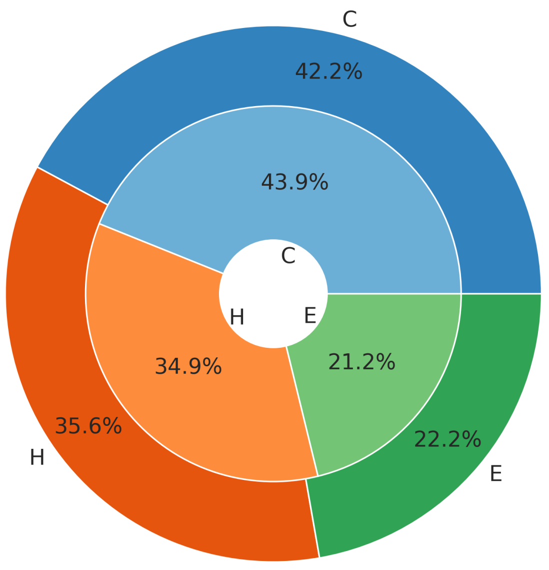

Fig. 3. Secondary structure comparison between train and test sets.

Test

The results of the generalization error of the best estimator were largely expected. For an out-of-the-box SVM, out classifier managed to achieve a score of 0.73 (Table 1), which is quite impressive given the minimal amount of optimization involed. The SVM model consistently outperformed the GOR model in all scores, if not for the Helix recall and Coil precision (table Table S6). It seems that the SVM traded some of the precision for a significant boost in Coil recall. Almost the same trend discussed before was observed now: the Helix class still is the easier to predict, but the Elongated and Coil classes frequently exchange places according to the metric used. Coil precision and recall are better that Elongated, but for accuracy the situation is reversed. However, the is a clear difference on a per-class level between the probabilistic approach of the GOR and the less transparent one of the SVM. If before a standard error of more than 0.1 was frequently observed, right now the OvR mean show a remarkable decrease of variability between the classes. We can suspect that this is due to the notion, or better lack thereof, that SVMs treats all numbers equally in spite of the clear biological differences that exist between them.

Table 1. Q_3 test accuracy summary.

	GOR-1	GOR-17	SVM
Q_3	0.504	0.627	0.730

4 Discussion

One of the most striking result is the cross validation of the second GOR model. It clearly indicates how the model, even with the modest size of our training set, is saturated with information: no conceivable difference was noted across all five validation folds. It's clear that the modest performances noted are, without a shadow of a doubt, direct implications of the limitations induced by the three simplifications previously discussed. The underlying available probabilities are correctly estimated, but the lack of predictive power resides in the approximations used to extract the information from the trained parameters. In other words, the lack of complexity constrains the model to pick up only the most evident patterns: helix and strands. Moreover, the paramount importance of the sequence context is confirmed by the sub-par performance of the context-deprived estimator ()Table 1). In synthesis, the gor model trained is as good as it can get (* GOR V accounts for residues pairs and triplets, decision constants in the final prediction, 13 residues window); to achieve higher performances there is the need to look for other approaches. In this regard, the SVM classifier consistently outperformed the GOR method, but this has to be expected since it encapsulates decades of machine learning advancements. The possibility to model non-linearity, introduced by non-linear functions, was a great achievement. Unfortunately, our results show that this feature of SVMs was not necessary to achieve satisfactory performances. Both the influence of the γ parameter and, perhaps surprisingly, the use of non linear kernels clearly were not rewarded. To be fair, the performance differences between the linear and non-linear kernels were not that obvious, but the noticeable tendency to overfitting, for the non-linear kernels, should come at least as suspicious. In summary, SVMs, when properly optimized, can offer a consistent boost in performance with respect to their probabilistic counterparts. However, careful hyperparameter optimization is needed to achieve satisfactory results; a not so trivial use-case given their problem dependence. Moreover, the fact that they are more or less "black boxes" makes it difficult to derive biological knowledge, even from well fitted estimators. This remark can be juxtaposed to the notion that a deep understanding of the biological reality is perhaps even more important than the sheer power of the model. Such consideration inspired by the stunning effect of the sequence context in the accuracy for the model performance. More powerful and efficient models (i.e. neural networks) have been proven to work nearby the theoretical upper limit. As of today, it is difficult to recommend support vector machines for secondary structure prediction, at least with their naive implementation.

References

Altschul, S. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, **25**(17), 3389–3402.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3), 403–410.

Berman, H. M. (2000). The protein data bank. *Nucleic Acids Research*, **28**(1), 235–242.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, **20**(3), 273–297.

Drozdetskiy, A., Cole, C., Procter, J., and Barton, G. J. (2015). JPred4: a protein secondary structure prediction server. *Nucleic Acids Research*, **43**(W1), W389–W394.

Frishman, D. and Argos, P. (1995). Knowledge-based protein secondary structure assignment. *Proteins: Structure, Function, and Bioinformatics*, **23**(4), 566–579.

Garnier, J., Osguthorpe, D., and Robson, B. (1978). 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.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant,

T. E. (2020). Array programming with NumPy. *Nature*, **585**(7825), 357–362.

Heffernan, R., Paliwal, K., Lyons, J., Singh, J., Yang, Y., and Zhou, Y. (2018). Single-sequence-based prediction of protein secondary structures and solvent accessibility by deep whole-sequence learning. *Journal of Computational Chemistry*, **39**(26), 2210–2216.

Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, **22**(12), 2577–2637.

Karush, W. (1939). *Minima of functions of several variables with inequalities as side conditions*. Master's thesis, University of Chicago.

Kieslich, C. A., Smadbeck, J., Khoury, G. A., and Floudas, C. A. (2016). conSSert: Consensus SVM model for accurate prediction of ordered secondary structure. *Journal of Chemical Information and Modeling*, **56**(3), 455–461.

Lifson, S. and Roig, A. (1961). On the theory of helix—coil transition in polypeptides. *The Journal of Chemical Physics*, **34**(6), 1963–1974.

Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, **405**(2), 442–451.

Nickolls, J., Buck, I., Garland, M., and Skadron, K. (2008). Scalable parallel programming with cuda. *Queue*, **6**(2), 40–53.

Rost, B. (2001). Review: Protein secondary structure prediction continues to rise. *Journal of Structural Biology*, **134**(2-3), 204–218.

Russell, R. B. and Barton, G. J. (1993). The limits of protein secondary structure prediction accuracy from multiple sequence alignment. *Journal of Molecular Biology*, **234**(4), 951–957.

Steinegger, M. and Söding, J. (2017). MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, **35**(11), 1026–1028.

Steinegger, M. and Söding, J. (2018). Clustering huge protein sequence sets in linear time. *Nature Communications*, **9**(1).

The UniProt Consortium (2018). UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, **47**(D1), D506–D515.

Wen, Z., Shi, J., Li, Q., He, B., and Chen, J. (2018). ThunderSVM: A fast SVM library on GPUs and CPUs. *Journal of Machine Learning Research*, **19**, 797–801.

Zimm, B. H. and Bragg, J. K. (1959). Theory of the phase transition between helix and random coil in polypeptide chains. *The Journal of Chemical Physics*, **31**(2), 526–535.

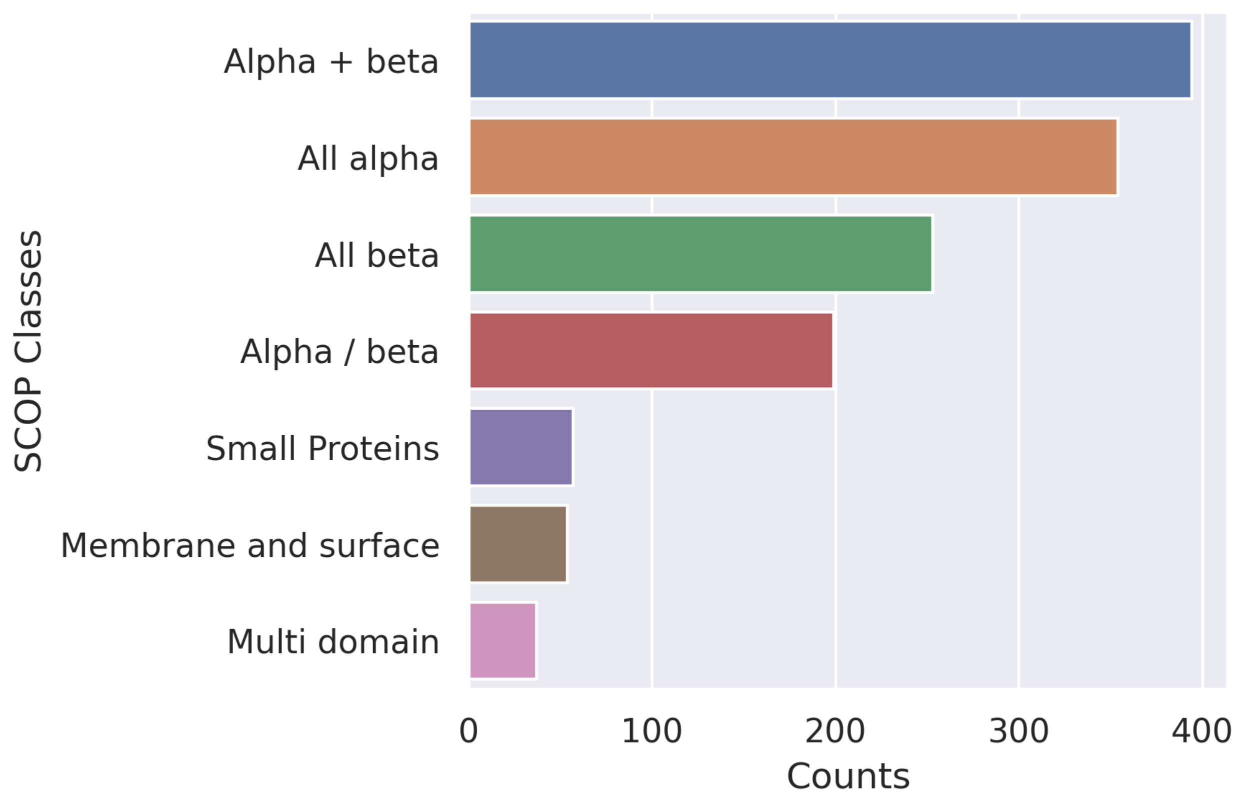

Fig. S1. Sequence length comparison between train, test and Uniprot Swiss-Prot datasets.

Supplementary material

Table S1. GOR method train and test MCC.

Fold	1	2	3	4	5	Average
Train score	0.467	0.468	0.476	0.460	0.463	0.467±0.003
Test score	0.472	0.470	0.464	0.472	0.466	0.469±0.002

Table S2. MCC train scores for the grid search of the SVM.

Fold			1	2	3	4	5	Average
Kernel	C	γ						
linear	2	NA	0.556	0.554	0.552	0.558	0.558	0.555±0.001
	4	NA	0.556	0.554	0.552	0.558	0.558	0.555±0.001
polynomial	2	0.5	0.991	0.991	0.991	0.992	0.991	0.991±0.000
		2.0	0.995	0.995	0.995	0.996	0.996	0.995±0.000
	4	0.5	0.993	0.993	0.994	0.994	0.994	0.994±0.000
		2.0	0.995	0.995	0.995	0.996	0.996	0.995±0.000
rbf	2	0.5	0.978	0.979	0.979	0.979	0.979	0.979±0.000
		2.0	0.994	0.994	0.995	0.995	0.995	0.995±0.000
	4	0.5	0.990	0.990	0.991	0.991	0.991	0.991±0.000
		2.0	0.995	0.995	0.995	0.996	0.996	0.995±0.000
sigmoid	2	0.5	0.329	0.300	0.292	0.301	0.298	0.304±0.006
		2.0	0.153	0.084	0.099	0.000	0.062	0.080±0.025
	4	0.5	0.298	0.300	0.293	0.302	0.299	0.298±0.002
		2.0	0.000	0.105	0.099	0.066	0.062	0.066±0.019

ciaociao

Table S3. MCC test scores for the grid search of the SVM.

Fold			1	2	3	4	5	Average
Kernel	C	γ						
linear	2	NA	0.548	0.551	0.566	0.543	0.544	0.551±0.004
	4	NA	0.548	0.551	0.567	0.543	0.544	0.551±0.004
polynomial	2	0.5	0.543	0.544	0.556	0.539	0.534	0.543±0.004
		2.0	0.496	0.504	0.514	0.497	0.489	0.500±0.004
	4	0.5	0.526	0.528	0.542	0.523	0.519	0.528±0.004
		2.0	0.497	0.504	0.514	0.497	0.488	0.500±0.004
rbf	2	0.5	0.549	0.552	0.560	0.539	0.541	0.548±0.004
		2.0	0.168	0.185	0.184	0.169	0.153	0.172±0.006
	4	0.5	0.539	0.545	0.550	0.530	0.531	0.539±0.004
		2.0	0.167	0.182	0.181	0.166	0.151	0.169±0.006
sigmoid	2	0.5	0.323	0.302	0.324	0.299	0.302	0.310±0.006
		2.0	0.148	0.064	0.108	0.000	0.069	0.078±0.025
	4	0.5	0.307	0.302	0.324	0.298	0.302	0.307±0.005
		2.0	0.000	0.087	0.108	0.048	0.069	0.063±0.018

Table S4. GOR estimato metrics with window size set to 1.

	Helix	Elongated	Coil	OvR
Precision	0.492	0.363	0.631	0.495±0.078
Recall	0.468	0.506	0.531	0.502±0.018
Accuracy	0.646	0.701	0.661	0.669±0.016
Matthews Correlation Coefficient	0.212	0.234	0.300	0.249±0.026

Table S5. GOR estimator blind set metrics with window size set to 17.

Score	Helix	Elongated	Coil	OvR
Precision	0.641	0.470	0.826	0.646±0.103
Recall	0.804	0.734	0.436	0.658±0.113
Accuracy	0.775	0.768	0.712	0.752±0.020
Matthews Correlation Coefficient	0.541	0.444	0.428	0.471±0.035

Table S6. SVM estimator blind set metrics

	Helix	Elongated	Coil	OvR
Precision	0.767	0.682	0.723	0.724±0.025
Recall	0.740	0.571	0.800	0.703±0.068
Accuracy	0.831	0.853	0.777	0.820±0.022
Matthews Correlation Coefficient	0.624	0.534	0.555	0.571±0.027

The results refer to an estimator trained on the whole training set with a linear kernel, C set to 2 and γ set to 0.5.