PROJECT IN MOLECULAR LIFE SCIENCE REPORT

--Stride 3 State Secondary Structure Predictor Building

Xueqing Wang 950124-4942

Abstract:
Protein secondary structure prediction is of great importance to a broad range of studies. Here in this project we built a 3-state Stride protein secondary structure predictor using python programming with the help of scikit-learn (sklearn). The database we used are preprocessed to get better results, and we also added evolutionary information by running psi-blast on each sequence. We tested the performance of our predictor by doing 5-fold cross validation under different window sizes as well as three different classification methods. The overall accuracy, confusion matrix, precision and recall, as well as operation time are used to evaluate how good the predictor performs. In our situation, the combination of random forest classification, with frequency matrix input under window size 11 is confirmed to be the best one, with Q3 accuracy of 74.3%. We also analyzed the reason why the accuracy of our predictor is lower than the state of art ones.

Introduction:

Protein structure prediction is of great importance to a huge fields of researchers, as it's the fundamental step of a large variety of studies such as drug design, protein evolutionary study and is useful for understanding the functions of different protein chains. The majority of protein 3D structure information is stored in the amino acid sequence. From the sequence one can infer where the secondary structure units lie in, which depict the interactions among neighboring amino acids and constitute the subunits of protein 3D structure. In other words, the structure of a protein chain can be considered as an array of secondary structure units. [1] Thus, precisely predicting protein secondary structure is a crucial issue for bioinformatics or biotechnology studies. Precise define and classification of different secondary structure elements are essential for the training and assessment of secondary structure prediction methods. Several algorithms for distinguishing secondary structure categories have been released over the last several decades and some of them are widely admitted. For example, DSSP [2] is one of the most used secondary structure assigning algorithm, which assigns the secondary structure by identifying the protein backbone hydrogen bonds using pure electrostatic definition. This algorithm assigns protein sequence to 8 types of secondary structures (8 states), which can usually be grouped into 3 larger groups, helix, strand and loop. The Stride [3] assignment criteria includes dihedral angle potentials on the basis of the hydrogen bond criteria of DSSP. The structures are divided into the same groups as DSSP with slight difference in the assignment. Besides the original amino acid data, Stride also includes probability information from solved sructures on PDB.

Scikit-learn (sklearn) is a machine learning software library which is free to use and mainly consists of python language [4]. There are many built-in algorithms including ones that are used for classification, regression, clustering and so on. It is initially created by David Cournapeau and is later developed by others. We used the resources (python codes for certain functions like cross validation and confusion matrix) in sklearn in our project and modified them to make them suitable for our cases. Support vector machine (svm) is a supervised learning method [5] which can be used to classify the training set by calculating a hyperplane which can divide data with different labels to the maximum extent with the gap between different groups as wide as possible. It utilizes a subset of training points in support vectors, which are decision functions, and is effective in predicting data in high dimensional spaces.

We applied svm for the classification of training dataset, to build a 3 state predictor model based on Stride, and tested the accuracy got out of cross validation for different kernel (linear, sigmoid, polynomial and rbf) in

different window sizes to compare the performance and optimized the kernel and window size, under the combination of which the accuracy can reach the maximum. Evolutionary information is added by running psi-blast of the original sequence database. Under the optimized condition we also calculated the confusion matrix, precision as well as f1-score of the established predictor, and compared our results with state-of-the-art predictors.

We also compared the performance of our method based on svm with two other sklearn built-in methods: decision tree and random forests. Decision tree classification maps the training data by a tree, with leaves representing different labels and branches depicting the connections of features. Random forest classification constructs a multitude of decision trees within training set and then outputs the mode of the classes. [6,7] In most cases it can avoid the feature of overfitting. [8]

Here we used the svm function in sklearn to build a 3 state predictor model based on Stride, and evaluated its performance. The 3 states are divided as helix, strand and coil, represented by H, S and C, separately. We also compared the performance of our method based on svm with two other built-in methods in sklearn—random forests and decision tree. After comparing the performance of model under different window sizes and classification methods, we built the predictor under the best conditions, and reported the overall performance of it. We also compared our predictor performance with that of the state of art predictors.

Methods:

1. **Raw-data processing**

   The database I got contains 399 sequences, each with label H(Helix), S(Strand) or C(Coil). I excluded two sequences whose length is less than 30 amino acids, and then did homology reduction by CD-HIT to reduce homological sequences. The cutoff of CD-HIT was set to be 0.7, leaving the remaining sequences with difference more than 30%. To do the homology reduction, I first extracted the sequences with only the headers but no amino acid features(labels), building a fasta file, and after the reduction I got another fasta file containing 332 sequence, all of which come from the original ones. Then I added the labels behind each extracted sequence. The reason I did homology reduction is that a dataset containing sequences with similar fractions may causes bias for the prediction: if the training set itself contains too many homologous sequences it may overpredict some features, while if the training set and testing set contain similar sequences the accuracy we observed will be higher than its real value, which is called overfitting.

   To see if these pre-processing procedures can improve the overall performance, we also conserved the raw data set and did identical data extraction and evaluation procedures to compare the performance of the predictor got out of raw sequences.

2. **Data extraction**

   To make the raw data suitable for inputting into svm, we have to extract useful information from the database and rewrite them to meet the standard format of training set input. We substituted the amino acids(features) and labels by 3 different integers and 20-dimension vectors, separately, and created a program to read the amino acids dynamically by sliding through the sequence one by one. We chose 20-dimensional vectors basing on the fact that there are only 20 essential amino acids in my database, thus 20 dimension is efficient for representing them individually. To take the affect of neighboring amino acids into account we used the concept of sliding window and read the sequence by the unit of window. Different window sizes are tested on the same database for optimizing. Each window corresponds to the label of the center amino acid in this window, thus the size of it must be an odd number. On the N- and C- termini of each sequence we introduced 20-dimension vectors only consisting of zeros to include the feature at the ends of the sequences, avoided the loss of information at the termini due to the use of sliding window.

3. **Evolutionary information adding**

As the performance of predictor got out of single sequence data is not good enough (the accuracy is not high) we tried to introduce evolutionary information at the input level. This is achieved by locally running psi-blast on the raw sequences. We chose uniprot90 database to search and used 3 iterations and 0.001 as e-value threshold. All the pssm files are processed in the similar way as the above section to make proper input format of training set. However, instead of using binary code vectors as before, the input after running psi-blast is the frequency vectors, which are still 20-dimensional but consists of floats ranging from 0 to 1.

4. **Evaluation procedure: sklearn built-in functions**

   To see whether the predictor we built can properly predict the outcome, we did 5-fold cross validation on our predictor, thus the rate of training and testing dataset is 4:1. Using the classification method of support vector machine(svm), we tried different window sizes and different kernel types to optimize the accuracy. As the time consumed is considerably much we changed several parameters of svm from their default value to speed up. We used one vs rest decision function of shape, reported the time of each round of testing as well as added integer seed to get the solid grouping pattern of each round of cross validation, which made the comparison among the accuracy of different window sizes more credible. Meanwhile we added "n_jobs=-1" for cross validation to make use of all the cores of computer to fully utilize the calculation power. However, the operation time is still high for svm classification and thus should be considered as a crucial factor for the evaluation. Linear Support Vector Classification (LinearSVC) takes much less time than the situations when using SVC and defined kernels by ourselves. As a result, we tested a large range of window sizes under LinearSVC and chose a relatively smaller range of window sizes to test the performance under SVC with different kernels.

   The cross validation function within sklearn automatically splits the raw dataset randomly with each sliding window as the smallest unit, so the training set and the test set may not be divided at the protein sequence level, thus they may contain the segments from the same sequence. But as our goal is predicting unlabeled protein sequences, all we need to concern is the residue itself and its neighboring effect, thus the whole sequence doesn't make much sense. From this point of view, manual cross validation is not of great importance and thus we didn't take it into account in this project.

   The use of accuracy alone is not enough to precisely depict the overall performance of our predictor. We introduced confusion matrix, precision as well as f1-score as complimentary approaches to evaluate our results. The specific procedures are built with the help of resources on sklearn website. We also compared our results with state of art results in similar area.

5. **Comparison with other built-in methods in svm**

   Besides svm we also utilized other classification methods to compare the performance and get the one with the most optimized prediction ability. We chose random forest and decision tree classifiers and tested the performance under different window sizes. The accuracy and precision of different methods vary but the operation time of the latter two approaches are much lower than that of svm.

Results:

1. **The performance of predictor based on single sequence dataset without homology reduction**

   We first used the whole dataset without any processing to get the original result for the comparison afterwards. As stated above, the operation time of cross validation is rather high so we did LinearSVC at first to shrink the window size range. As we can see from Figure 1a, the accuracy increases fast with the window size when window is small, but after window size 15-19 the accuracy doesn't change much with the increasing window and nearly saturates. As larger window size requires more operating time and occupies more internal storage we considered 15 as the potential optimal window size and did SVC around the range of size 15-19 using four kernels: linear, sigmoid, polynomial and rbf (radial basis function). The

results in terms of accuracy and operating time are also showed in Figure 1b. As we can see, the polynomial kernel gives out results with accuracy much poorer than all the other kernels so we can discard it.

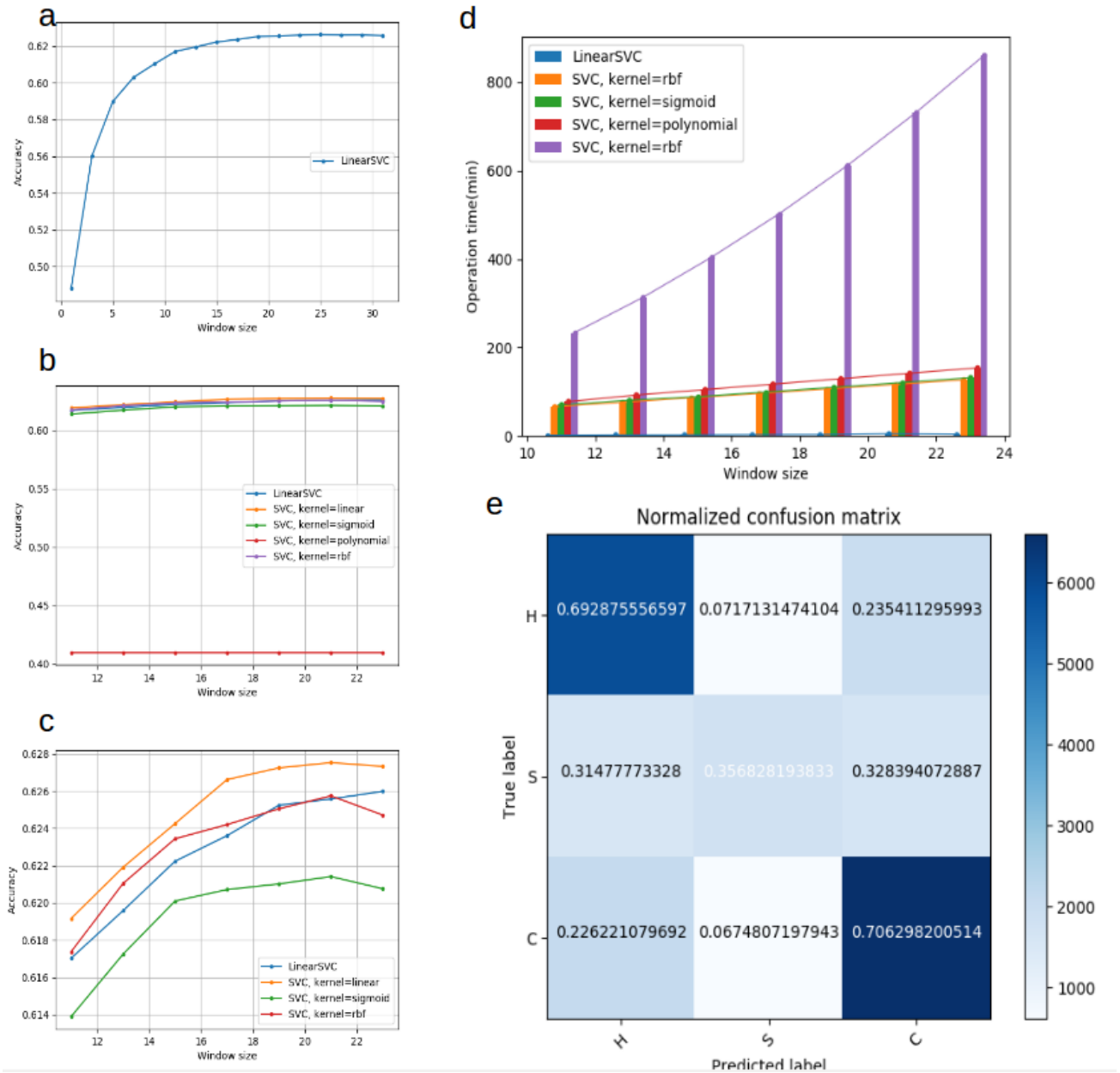

Fig. 1 a-c. The accuracy under different window sizes of LinearSVC as well as linear kernel, sigmoid kernel, rbf kernel in svm. d. The operation time of different kernels under different window sizes. e. The normalized confusion matrix of LinearSVC, window size 19, as these is the optimized condition considering both performance and operation time.

As the accuracy got out of polynomial kernel is much lower than others we can discard it and compare the outcomes of other kernels. According to Figure 1c, LinearSVC, SVC(kernel=linear) and SVC(kernel=sigmoid) have similar performance but the operation time of LinearSVC is much less than the other two. Thus LinearSVC at window size 19 can be used for the predictor, with accuracy of 0.625.
The time of running cross validation on SVC with each of the four different kernels is rather long and this hugely interfere with our research process. In the endeavor of solving this problem we introduced the "verbose" parameter to visualize the operating time of each round in the 5-fold validation, as well as "n_jobs=-1" to make full use of all the cores within a PC, as stated in the Method part. We also used "decision_function_shape='ovr'" in the data fitting step to speed up the procedure. But as we can see from Figure 1d, the optimized operating time is still long and has become an obstruction to the project. One

thing needed to be noticed is that, as we are not clear with the status of the computer we used and the difference of configuration of different computers, the time listed here may has some flexibility. From the confusion matrix in Figure 1e we can see that, in terms of the prediction of H and C, the accuracy is higher than the overall performance, but when it comes to S, the predicted outcome is only slightly better than a random state (33.3%). Although the prediction of strands is harder than the other two labels by nature, in this case for S the number of true positive is even worse than false negative, thus we hope that in the following steps it can be improved.

2. **The performance of predictor based on single sequence dataset with 70% homology reduction**

To increase the accuracy, we did homology reduction as stated above and tested the overall performance of window size 1-33 (Figure 2a) to find the relatively better window sizes for further exploration. After that we did SVC with different kernels at window sizes 9-19, as shown below in Figure 2b. Combining the consideration of accuracy and operating time, LinearSVC at 15-19 is the best choice. The operation time of LinearSVC here is also much better than that with all the other kernels, and as the amount and variation trend are quite similar as those in the former section, we didn't include a bar plot about time as before. The confusion matrix at window size 19 under LinearSVC is included as Figure 2c. The measurement of operation time has the same drawback of ambiguity as in the former section and thus should only be take into consideration but not as a rigid impact parameter.

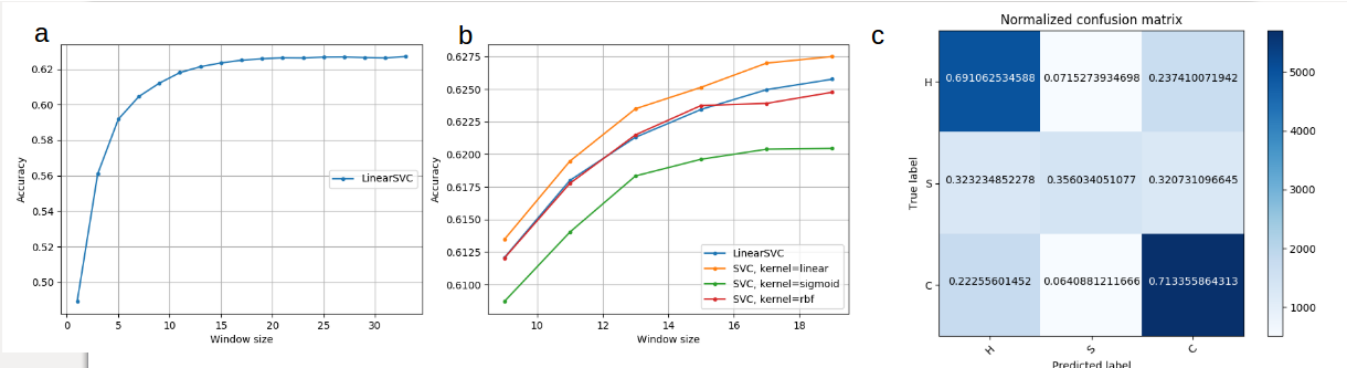

Fig. 2 a. The accuracy got in window sizes 1-33 under LinearSVC. b. The accuracy got in window sizes 9-19 under three different SVC kernels as well as LinearSVC. c. The confusion matrix in LinearSVC, window size=19.

In terms of performance, some other measurements can also be done such as precision and recall. More detailed description about this will be provided in section 4.

From the results we can see that, with the homology reduction done the accuracy doesn't show much enhance. As for the confusion matrix, the accuracy of label "C" is enhanced a little, but the other two labels didn't change much. The time consumed doesn't show much improvement. In other words, time is still a limit of the cross validation process.

It seems that, with homology reduction done, the result in terms of overall accuracy, operating time and the prediction for H and S have no or no much improvement, the only enhancement lies in the prediction for C. However, after homology reduction the size dataset do shrink into 90% of the original one, which can make the time of running psi-blast shorter in the following parts. Besides, the generated dataset corresponds to the database features of the state of art researches better, so we decided to still to it in the following steps. The fact that the improvement is not significant might because that the database we use it not large enough.

3. **The performance of predictor based on pssm files**

We added evolutionary information by running psi-blast locally as described above. The data in frequency matrix was extracted and used as the training set for the predictor. In this case, instead of using binary code vector consisting of "0" and "1" to depict the amino acids, we used the frequency within limit 0-1 to present each of the different positions. The evaluation procedures are similar to the ones above: doing

LinearSVC first at a huge range of window sizes and test the performance of these window sizes with different kernels of SVC. The accuracy of each kind of classification method is enhanced in this case, indicating the advantage of adding evolutionary information at the input level. The time is shrunk but still long.

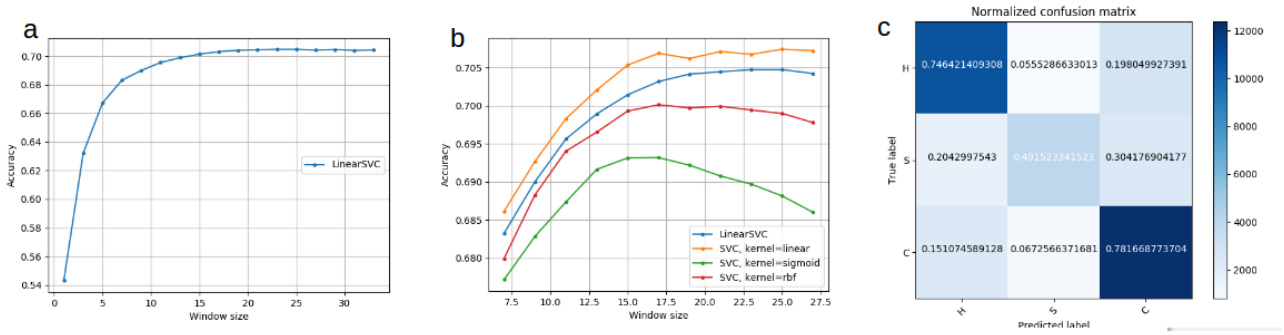

Fig 3 a. The accuracy of window size 1-33 in LinearSVC. b. The accuracy of window size 7-27 in three different SVC kernels as well as LinearSVC. c. The confusion matrix in LinearSVC, window size=19. All the figures are done with pssm files as inputs.

The predictor with pssm files as input has better performance in terms of accuracy compared with that with single sequence input. As our goal in this project is to get a predictor with the performance as good as possible, we chose pssm input instead of single sequence input for the following steps. The normalized confusion matrix with the best condition (LinearSVC, window size=19) is also shown in Fig 3c, with the sensitivity of the three labels increased by 0.05, 0.14, 0.07, separately, suggesting the effectiveness of adding evolutionary information. The prediction of label S is much better than before, the sensitivity of which is nearly 50 percent.

In terms of the operation time, when using pssm files as input, the time of LinearSVC doesn't show much difference, that is to say, although the time seems to be half of that when using original single sequence data as input, the absolute values of LinearSVC when using whole single sequences, single sequences after homology reduction and pssm files are all less than 5 mins in window size, thus need not to be considered as an important impact factor of the total performance. However, the operation time of SVC with different kernels do decreases, and as we want to test the performance under all different kernels it is a notable point for pssm input. We only show the operation time of sigmoid kernel below in Figure 4, but basically the time of other kernels were also lowered down compared to single sequence inputs.

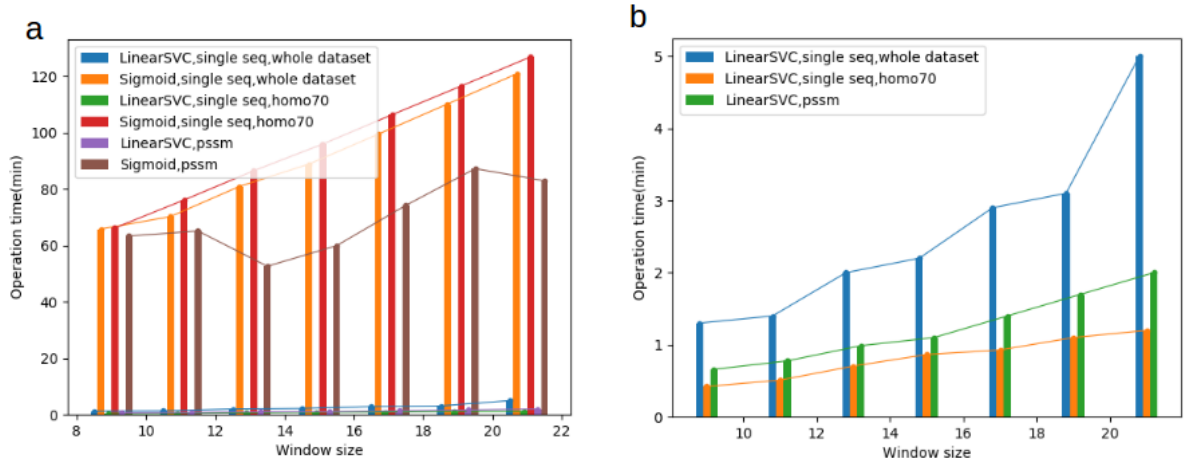

Fig 4 a. The operation time of LinearSVC and sigmoid kernel under different window sizes when using original single sequences, reduced single sequences or pssm files as input. b. The operation time of LinearSVC under different window sizes when using original single sequences, reduced single sequences or pssm files as input.

4. The comparison of performance of predictors using two other classification methods and state of

art predictors

We also took a look into other classification methods, random forest and decision tree, after focusing on svm. The overall accuracy of these two methods using pssm input are shown below. The random forest classification is done using a series of different numbers of forest trees from 10 to 80 with the step 10, while the decision tree classifier was done by default settings.

From Figure 5a we can see that, the more trees we use in random forest, the higher accuracy we will get. Meanwhile, the accuracy increases rapidly with the window size in the range of 1-5, after that it nearly reaches saturation with slight increase at around 5-11, and after 7-11 it slowly goes down. The fewer trees the predictor has, the lower accuracy we get, and the maximum shows at smaller window size. For instance, when 10 trees (the default number) are used the maximum is reached at window size 7, but when 80 are used it emerges at window size 11. The operating time of these two methods is similarly short as LinearSVC (Figure 5d), as shown below, and much shorter than SVC with other kernels. However, as we can see, with the number of trees increasing, the accuracy enhances slower and slower, with less than 0.9% growth from 50 to 80 trees. Besides, the time consumed also gets longer, which is, although not a factor to be considered much in this case, still make some sense to the overall performance of the predictor. To sum up, we chose 70 trees under window size 11 as the optimized parameter for random forest predictor.

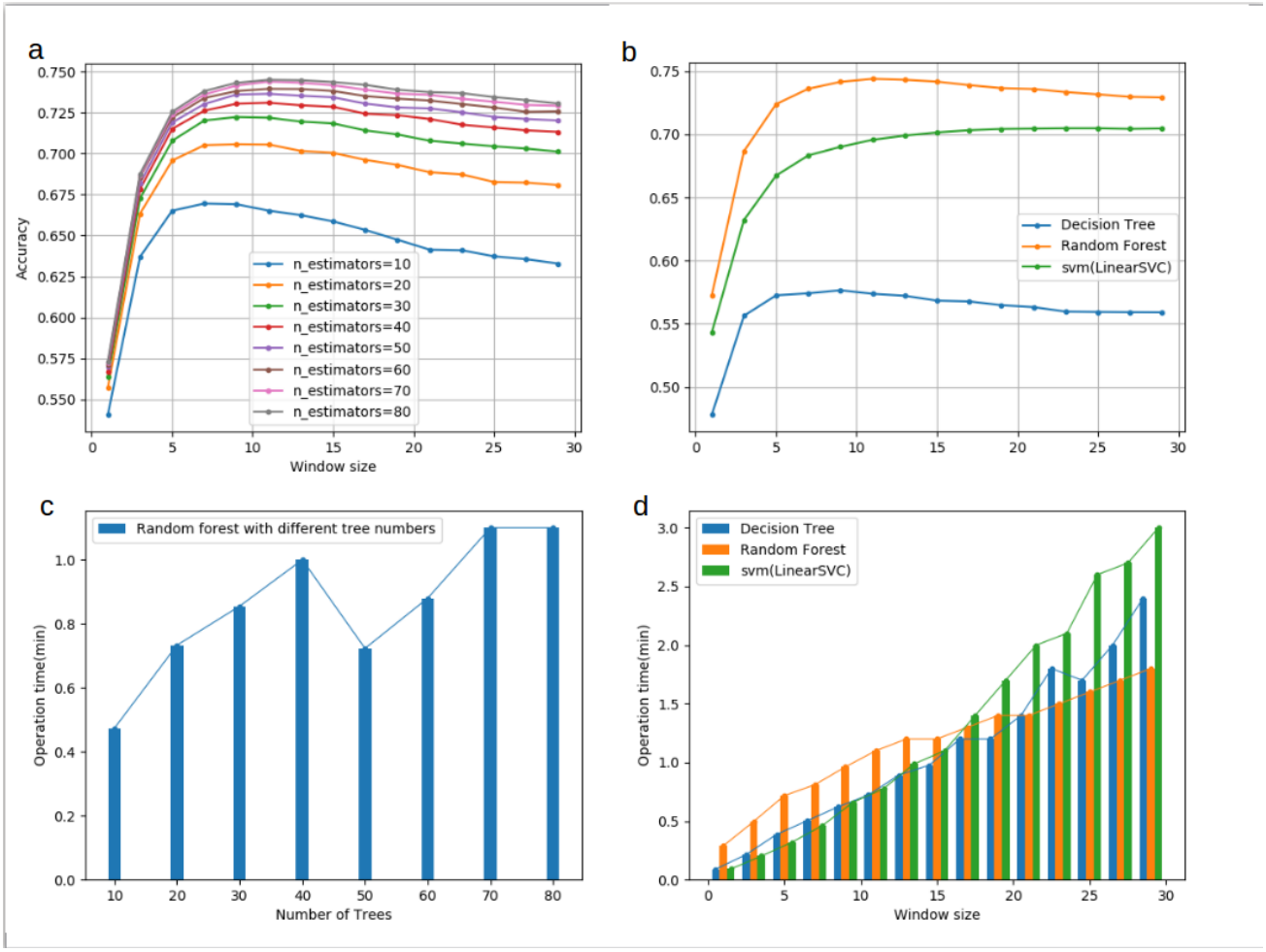

Fig 5 a. The accuracy of random forest classifier under different window sizes with different number of trees (presented by n_estimators). b. The comparison of three different classifiers in different window sizes. Random forest classifier has the best accuracy in all the window sizes. c. The operation time of random forest classifier under different number of trees. d. The operation time of three different classifiers in the optimized parameters under different window sizes.

The comparison among the accurancy of the three predictors with different classification as well as how they change according to the window size are shown in Figure 5b, each of the data in one classification method is got out of the best optimized condition. The precision, recall and f1-score of the three methods

are also shown in Table1, so does the confusion matrix. As we can see, considering either of these evaluation measurements, the performance of label "S" (Strand) is always the lowest, with the performance nearly identical to the random state in terms of accuracy and recall, and its performance in precision and f1-score is also poorer than that of "H" (Helix) and "C"(Coil).

Table 1 Precision, recall and f1-score of predictors under three different classification methods, after optimized

		precision	recall	f1-score
window size=19, pssm, svm.LinearSVC	H	0.72	0.74	0.73
	S	0.68	0.5	0.57
	C	0.69	0.78	0.73
	average	0.7	0.7	0.7
window size=11,pssm,random forest	H	0.8	0.78	0.79
	S	0.81	0.49	0.61
	C	0.69	0.85	0.76
	average	0.76	0.75	0.74
window size=9,pssm,decision tree	H	0.61	0.61	0.61
	S	0.45	0.45	0.45
	C	0.65	0.61	0.61
	average	0.57	0.57	0.57

We also searched for the state of art secondary structure predictors, and compared our results with their performance in terms of Q3 accuracy. The results are shown in Table 2. As we are not totally sure about where our dataset comes from, the comparison might not be accurate, based on the fact that when testing on different database, the accuracy got out of the same predictor can varies.

From the result we can see that, in terms of input data, the predictor with pssm files (evolutionary information) as input has better performance, while in terms of classification methods the quality of performance has the order random forest, svm, decision tree from high to low. Thus, we build our predictor with pssm input and random forest method, which has two versions to take in two different kinds of inputs, single sequence and fasta files. The python script can be find in supplement 1 and 2. We also attached the predictors using svm as supplement 3 for comparison.

Table 2 Q3 accuracy of the state of art predictors and our predictor

Predictor	SSpro (without template)	PSIPRED	JPRED	DeepCNF-SS	3 State Stride Predictor in Our Project
Q3 Accuracy	79.5	82.5	82.9	85.4	74.3

Discussion and Conclusion:

From our results we can clearly see that, under the procedures of homology reduction and evolutionary information adding by running psi-blast, the performance of predictor is enhanced. This can be interpreted as that, homology reduction removes considerable portion of sequences with identity above the threshold, thus reduces the possibility of overfitting or training on similar data; meanwhile, with the additional information of multiple sequence alignment, the accuracy of secondary structure prediction would improve than the cases when only single sequence [9] as structure information is largely restored in sequence information. [10]

When random forest is used, the accuracy we got is much better than a single decision tree, and also better than svm. This can be interpreted as, random forest creates a multitude of decision trees at the training step,

and since it returns the mean prediction of multiple trees, the results it gives make more sense than when only one decision tree is used as well as when a single hyperplane is used.

The operation time of cross validation can be super long when using different SVC kernels, which becomes one of the major parameters to be concerned in the optimization step. As a result, when using the svm classifier, we didn't choose larger window sizes (for example, the window sizes more than 20), although they can lead to a slightly higher accuracy. Despite the fact that, according to the classifier we finally chose in svm—LinearSVC, the operation time is dramatically reduced and need not to be considered as an refining factor of the overall performance, we cannot compare the performance of LinearSVC and SVC in other kernels in larger window sizes due to the limited time of the project. Thus, we stuck to the optimized window size of 19 in svm, although when the window size is increased not much more time is consumed. Besides, as the accuracy is close to saturation in large window sizes, it doesn't matter much whether to use larger window sizes or not.

From the confusion matrixes we can see that, for label "S", the number of false negative is nearly equal to that of true positive, and it is even slightly higher than true positive. This corresponds to the fact that the recall (sensitivity) of "S" is low in all the classification methods we have used, either equal or less than a random state(<=0.5). This may due to the fact that β-sheet structure need to be stabilized by hydrogen bonds formed with other β-sheet far from itself, thus it's not easy to predict it by just taking into account the nearby residues provided by the sliding window. [1] However, this is a universal problem confronting most of the secondary structure predictions.

The performance of my predictor is not so good as the state of art predictors, which can reach the accuracy of nearly 80%. This may due to three reasons: a) The datasets we used only contain 399 sequences before and 322 sequences after homology reduction, rather small compared to the databases which other researchers used. For example, one of the best secondary structure predictors, DeepCNF, developed by Wang et al in 2015, utilized ~5600 proteins from CullPDB as training set and ~500 PDB proteins as the test data, reaching Q3 accuracy of 85.4%.[11] b) Many of the state of art predictors use neural network as the core of machine learning procedure, which may fit better with the training set than our method using svm or random forest. c) The cross validation sets we used were automatically created by sklearn built-in function, without taking the impact of residues far away from the amino acid being predicted into account. However, actually residues far away can interact with each other to a considerable extent, such that the secondary structure can be influenced by hydrophobic effect and hydrogen bonds.

Our predictor is still a primary one and much parameters within the procedure have not been considered to the greatest extent due to the time limit. However, it does work well on predicting some sequences extracted from pdbcull database and some predicting results we got are quite close to the true secondary structures. To enhance its performance and make it more suitable for different test sets, further work such that checking and testing more parameters in the classifiers we chose need to be done.

References:

1. Zhou, J. & Troyanskaya, O. Deep Supervised and Convolutional Generative Stochastic Network for Protein Secondary Structure Prediction. Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. JMLR Proceedings 32, 745-753 (2014).
2. Kabsch W, Sander C. "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features". Biopolymers. 22 (12): 2577–637 (1983).
3. Frishman D, Argos P. Knowledge-based protein secondary structure assignment. Proteins 23(4):566-79 (1995).
4. Fabian Pedregosa; Gaël Varoquaux; Alexandre Gramfort; Vincent Michel; Bertrand Thirion; Olivier Grisel; Mathieu Blondel; Peter Prettenhofer; Ron Weiss; Vincent Dubourg; Jake Vanderplas; Alexandre Passos; David Cournapeau. "Scikit-learn: Machine Learning in Python". Journal of Machine Learning Research. 12: 2825–2830 (2011).
5. Rost, B. & Sander, C. Prediction of protein secondary structure at better than 70% accuracy. J. Mol. Biol. 232, 584-599 (1993).
6. Ho, Tin Kam. Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282 (1995).
7. Ho, Tin Kam. "The Random Subspace Method for Constructing Decision Forests" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 20 (8): 832–844 (1998).
8. Cortes, C., Vapnik, V. "Support-vector networks". Machine Learning. 20 (3): 273–297 (1995).
9. Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome. The Elements of Statistical Learning (2nd ed.). Springer. ISBN 0-387-95284-5 (2008).
10. Spencer, M., Eickholt, J., and Cheng, J. A deep learning network approach to ab initio protein secondary structure. IEEE/ACM Transactions on Computational Biology & Bioinformatics. 12, 103-112 (2015).
11. Wang,S., Peng,J., Ma,J. and Xu,J. Protein secondary structure prediction using deep convolutional neural fields. Scientic Rep., 6, 18962 (2016).