Predictor Project

Alpha helix- membrane topology prediction

by:

Sarah McComas

SciLifeLab

Science for Life Laboratory
via Stockholm University

Due: March 20th, 2016

Abstract

Protein structure prediction by means of computational biology has been a massive development in recent years. Knowing the structure or topology of the protein is important in moving forward with practices such as drug design. The most successful predictors today use different concepts laced together, including multiple sequence alignments from searching algorithms such as PSI-BLAST and using supervised machine learning. Here, I create my own predictor using the same concepts from these state of the art predictors in efforts to predict transmembrane alpha helical topology. I have created a predictor that can perform with up to 75% accuracy and I report on the fundamental concepts that lead me to creating this predictor as well as its advantages and drawbacks.

1 Introduction

Understanding protein structure is crucial to deepening our knowledge of their function. Some of the most abundant proteins are known as transmembrane proteins, which exist across various membranes and participate in many function such as cell transport and signaling. Of these transmembrane proteins, most of them are in the form of an alpha helix because their H-bonding properties allow for stability inside the hydrophobic lipid bilayer. About 27% of all proteins produced by humans are classified as an alpha helical transmembrane protein [1], and while understanding their exact structure is difficult, predicting the topology (i.e. where and in what orientation the protein is located in the membrane) gives enough information about the protein's function and potential targets for diseases or drug delivery. Gaining this information with traditional research is difficult and costly, which lead to methods to predict the protein's structure given the properties of its amino acids. The first method used to do this was known as the Chou Fasman method [2], which analyzed the frequencies of the amino acids' presence in alpha helices and other structures to predict a protein structure with about a 50% accuracy [6]. Since the development of machine learning algorithms and incorporation of multiple sequence alignments, the accuracy of these predictions now can reach 90% or higher [3].

Here, I have drawn inspiration from state-of-the-art predictors in creating my own predictor which determines if an alpha helix is inside the membrane or not. As these predictors before mine have, I used a multiple sequence alignment (MSA) profile generated from PSI-BLAST [2], which searches a database for sequences similar to a query sequence in efforts to gain evolutionary information about the query sequence. This generated MSA profile not only gives evolutionary information, but also can be used as an in depth analysis of a certain amino acid to see how frequently it occurs in a sequence at a given position. In order to analyze my predictor in depth, I also proceeded without PSIBLAST by only analyzing single sequence information. I could then use the information from the MSA profile or the single sequence as a learning algorithm for a support vector machine, or SVM [5]. SVM's use supervised or unsupervised learning to do binary classification, to determine what a positive and negative example for each datapoint look like. In this case, I used an SVM only with supervised learning techniques. This trained machine can then be implemented on new data where the structure is unknown in efforts to predict it. To use the MSA data on a machine learning algorithm is a successful and common practice in todays' top predictors for topology or secondary structure such as TOPCONS, TMHMM, PSIPRED, among many others [3, 7, 9] and is the main reason why these predictors are so successful in their accuracy and efficiency.

Another successful feature of modern computational predictors is the presence of open source information, web servers, and specific databases. Here, all groups can access information quickly and easily, which leads to collaboration and therefore an increase in database size. In this way, these groups can build off of each other to improve their own methods and the quality of the data presented. The two main limitations in these predictor qualities are time and accuracy. Now, accuracy is very high but there is still some disagreement between prediction models, which programs such as CASP [12] aim to solve. CASP essentially collects prediction data and compares the model's predictions in order to validate the quality of the prediction. Other programs use consensus prediction [3], which takes the predictions from other models as a factor for their own predictor, thereby increasing accuracy greatly. Even so, many databases continue to separate structures which have been computationally generated and those that are experimentally proven by traditional methods. With these improvements, one day we will no longer need traditional methods to validate structure.

2 Methods

I extracted features from a database of solely alpha membrane proteins with known topologies to train an SVM with supervised learning. This database consisted of about 300 sequences, and each entry had the accession number of a protein, its amino acid sequence, and the corresponding position with regards to the membrane (I for inside, O for outside, and M for membrane). In this case, all positions with an 'M' would become a positive example, while 'I' or 'O' would be a negative one. I also assigned each amino acid letter a corresponding number so that it was readable by the SVM. This lead to a standard SVM format, as seen and described in Figure 1.

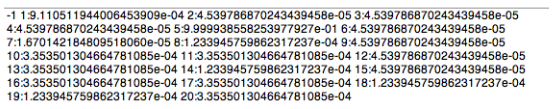
To increase accuracy in the SVM, I created 6 sets, each with about 50 proteins from my database in order to perform cross validation. This is a very important step because otherwise, when the SVM is trying to find the best hyperplane to separate the positive and negative classifications, it will become too specific to the training example it has been given, which is known as overfitting. In creating a variety of sets from which the SVM can optimize its models, it becomes easier to achieve a good balance between overfitting and a high accuracy of prediction. For the same reason, I trained and optimized my SVM on 5 of the 6 datasets to leave one dataset, known as the validation dataset, in which I tested at the very end of the optimization to ensure that I hadn't overfit my predictor.

Because alpha membrane proteins are important, it is not uncommon to see conservation in the protein sequence and therefore homologs are more than likely present. If they are present and happen to be separated into different cross validation sets, this can report falsely high accuracy when testing the model because the homologous sequences are too similar. It would be as if I had tested the SVM on the same dataset I trained it. To avoid this, I input my entire database into CD-HIT [16] which reported back the sequences in groups of homologs. My dataset did have these, so I made my cross validation sets to ensure that if there were homologous sequences, they would end up in the same cross validation set.

I trained the SVM in two separate manners. Firstly, I used the single sequence information ,which was not expected to create a high quality predictor but it is important to compare to the predictor created from MSA profiles. As the MSA profile was created from running PSI-BLAST locally on my entire database, this search took several days to complete. Once finished, a position specific scoring matrix, or PSSM, was generated for every amino acid in the sequence. This PSSM correlates to each amino acid's presence in evolution at that certain position. This PSSM is the main difference between using the MSA data and using single sequence information for the SVM. The single sequence information merely contains a 1 or 0 depending on if the amino acid exists at that position in the sequence or not, respectively. The PSSM is therefore much more thorough, but must be transformed first so that the SVM can read it as a feature value (illustrated in Figure 1), a value between 0 and 1. Before the MSA data could be read by the SVM, it was first transformed using the sigmoid function as shown below:

$$S(t) = \frac{1}{1 + e^{\mathrm{t}}}$$

There is no true limit on how much one can train an SVM, and here my only real limitation was time. Particularly for the PSI-BLAST output, training the SVM on an MSA profile would usually take about 8 to 12 hours. I first trained the SVM on single sequence data but did not further pursue any optimization, knowing that I did not have much time to optimize and I was unlikely to obtain a high quality model from this data. From SVM$^{\text{light}}$ [5], there are many options on which one can optimize beyond the default parameters, which is important for obtaining the best possible results in the predictor. After running default settings for the SVM on my MSA profile, I tried with 3 kernels, which are different algorithms for finding the right means of separating data in an n-dimensional space. I tried these kernels with several parameters as well, customizing the kernel. A parameter can be, for example, defining a trade-off between misclassification and the simplicity of the separation (the c parameter) or how constrained the model should be (the *gamma* parameter) [11]. According to the SVM$^{\text{light}}$ package information online, there are several guidelines in which to utilize the parameters (for example, c should be a float of any value, w should be a value below 1.0, and so on), but otherwise a guess and check method of parameter optimization was the most practical to me. I therefore tried parameters such as $c = 2.0$ and $w = 0.4$, or $c = 0.5$ and $g = 4.0$. I could find some information on successful parameters online but as each dataset is different, it was not useful to apply these parameters on my own data.



```
-1 1:9.11051194400645390 9e-04 2:4.5397868702434394 58e-05 3:4.5397868702434394 58e-05
4:4.5397868702434394 58e-05 5:9.9999385582539779 27e-01 6:4.5397868702434394 58e-05
7:1.6701421848095180 60e-05 8:1.2339457598623172 37e-04 9:4.5397868702434394 58e-05
10:3.3535013046647810 85e-04 11:3.3535013046647810 85e-04 12:4.5397868702434394 58e-05
13:3.3535013046647810 85e-04 14:1.2339457598623172 37e-04 15:4.5397868702434394 58e-05
16:3.3535013046647810 85e-04 17:3.3535013046647810 85e-04 18:1.2339457598623172 37e-04
19:1.2339457598623172 37e-04 20:3.3535013046647810 85e-04
```

Figure 1: A typical line of MSA input for the SVM. Each line begins with a target value: a positive or negative number depending on structural position. Following that are the feature numbers and their values, separated by a **:** . The feature number corresponds to the amino acid number, and the value is the PSSM after the sigmoid function. This would simply be a 1 or 0 if this was single sequence information.

3 Results

Once optimized, the predictor was capable of predicting protein topology with an average of about 75% accuracy, as shown in Figure 2. Of the 3 kernels tested when training the SVM with multiple sequence data, one (the polynomial) kernel only produced partial results because the running time was extremely long. I analyzed what results I could from this and found no improvement on my model so I chose not to report them here.

There are several means of testing the quality of the predictor. Firstly, the accuracy is reported immediately after the SVM has performed a classification. This dictates the percent of entries predicted correctly by the model in comparison to the chosen cross-validation set. This can be a fast and easy determinant of how the classifier is performing but is not as indicative of the model quality as a whole. For this, I can measure the Matthew's correlation coefficient, or MCC by calculating:

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

This MCC is therefore on a scale between 1 and -1, 1 meaning that the predictor is perfect, 0 is completely random prediction, and -1 is worse than random. This number is applicable to a model of any size and is one of the best means of visualizing predictor quality. Figure 2 depicts the average MCC for several of my models as well as the average accuracy for comparison. As is clear by the figure, the best predictor I have made comes from the radial kernel with default parameters. I have tested several other parameters with both the radial and the linear kernel for my predictor, but none of them reached the same accuracy or MCC value as the default parameters. The only limit I had in testing these parameters was time, so when I felt that I had no more time to optimize, I stopped trying to test parameters. Training the SVM's on the BLAST output was very time consuming, up to 12 hours per cross validation set for some kernels, such as the polynomial. This also played a factor in how much I could optimize my parameters.

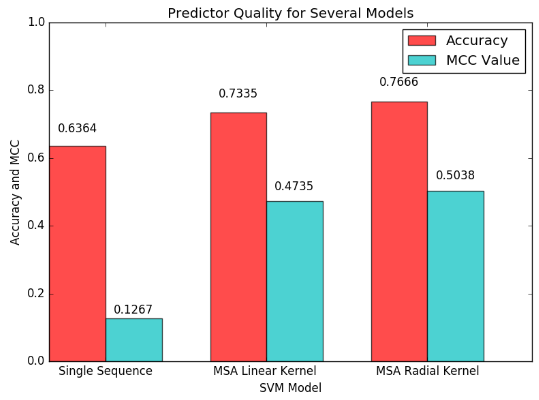


Figure 2: A comparison of the average values for several predictors using various kernels.

One other means of testing the predictor quality is via the Receiver Operating Characteristics curve, or the ROC curve [10]. This is a means of comparing sensitivity (which increases proportionally with the True Positive Rate) and specificity (which decreases as True Negative Rate increases). Figure 3 depicts a ROC curve I have created for my 3 main predictors: the single sequence, and the multiple sequence for two different kernels. The most important feature of this curve is the AUC, the area under the curve. This, like the MCC, dictates predictor quality. If the AUC is 0.5, this is an indication that the predictor is no better at distinguishing datasets than random chance is. Therefore, the closer the AUC to 1.0, the better.

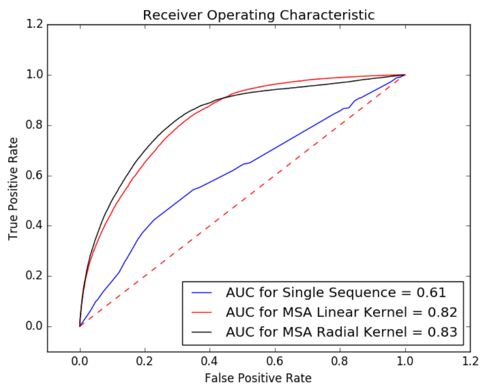


Figure 3: A ROC curve comparing the same predictors as Figure 2. I made this ROC curve and calculated the AUC utilizing tools from scikitlearn [13] and matplotlib [11].

It is clear in interpreting my results that the predictor using the radial kernel and multiple sequence information is the best; for accuracy, the MCC, and the AUC. However, these were calculated by taking the average of all of the cross validation set data, and the averages are not enough to entirely analyze the quality of the model. As seen in Table 1, I compared the averages to the data obtained by classifying the validation set against all other data (the validation set is the set that has not been used at all before this) and found a decrease in overall predictor quality.

Table 1: Comparison of the predictor's performance when introducing new data

	Linear Kernel		Radial Kernel	
	Average	Validation Set	Average	Validation Set
Accuracy	73.36%	71.19%	76.66%	75.49%
MCC	0.4735	0.4381	0.5038	0.4184

4 Discussion

My predictor can dictate alpha membrane topology fairly well. In comparing the predictor output for a sequence with a known topology, the patterns are very clear and nonrandom, and match the actual topology to a fairly high degree. Figure 4 depicts this, where I have included a screenshot of what a typical outcome for my predictor looks like. I can see from the prediction that there are several concentrated spots in the sequence where the protein is in the membrane, a reasonable model of actual protein topology.

The decrease in quality when comparing my predictor's averages versus the validation set as seen in Table 1 could be attributed to several factors. This could be happening because of overfitting my models, which happens if I were to trains the model to be too specific to the datasets given. However, I find this highly unlikely because I had the best results with and therefore used default parameters for both linear and radial kernels. I would expect overfitting to have become a problem if I had tested my validation set on a specific parameter. Therefore, it is likely that this decrease is due to poor chance, and a more random and hard to predict validation set.

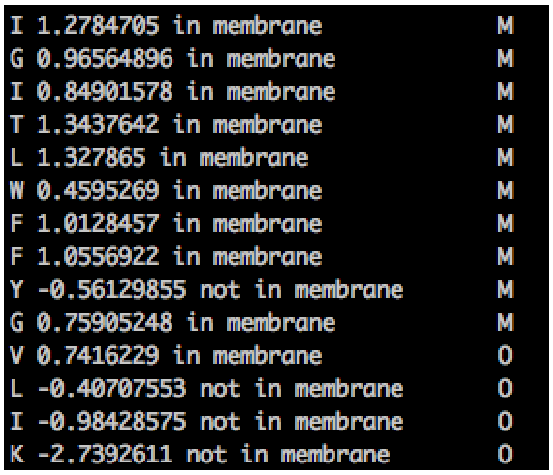

```
I 1.2784705 in membrane                M
G 0.96564896 in membrane               M
I 0.84901578 in membrane               M
T 1.3437642 in membrane                M
L 1.327865 in membrane                 M
W 0.4595269 in membrane                M
F 1.0128457 in membrane                M
F 1.0556922 in membrane                M
Y -0.56129855 not in membrane          M
G 0.75905248 in membrane               M
V 0.7416229 in membrane                O
L -0.40707553 not in membrane          O
I -0.98428575 not in membrane          O
K -2.7392611 not in membrane           O
```

Figure 4: On the left, the output from the predictor including the expected target values. On the right is the original file, M represents inside the membrane and I and O outside the membrane.

My predictor is good, significantly better than random, but only when the right data is applied. I tested this by running the predictor on a sequence from a database for transmembrane beta barrels and found that my predictor was as good as random when I compared the predicted results to the actual structure. In this, I confirmed that while capable of predicting membrane alpha protein position fairly well, my predictor is not nearly as ubiquitous as the state of the art programs such as TOPCONS or even TMHMM or PSIPRED, which can quickly and accurately predict structure and topology for a variety of proteins. These programs are capable of such high accuracy for several reasons, one being that they use consensus prediction, a method of combining other predictor outputs into one predictor. If the models agree with each other, the accuracy in the prediction can be increased significantly.

A severe limitation for my predictor is time. It takes about 45 minutes for one query sequence to be processed, including the time it takes to generate an MSA. TOPCONS and the like can perform this in under one minute. These predictors have increased their speed by differentially searching databases rather than searching through all of the UniRef [14] database for each query, which my predictor does. This database searching for the MSA profile generation lasts for 30 of the 45 minutes and is therefore the main reason that my predictor is slower. If I were to try to increase the speed of my predictor, I would first start by avoiding searching the UniRef database when possible, as TOPCONS has done. [15] When my predictor runs on single sequence information only, the process is significantly faster. This is in part because it does not have to search any database, but also because the SVM has much less information to process. However, as the results have shown, I would not consider this to be an advantageous tradeoff between time and predictor quality. If I were to make a webserver for this predictor, I would not consider using single sequence information given how low quality the predictor is. The evolutionary information is important because it adds a much more in depth analysis of each amino acid at a certain position, taking into account properties of the amino acid itself rather than just the mere presence in one sequence. Although the SVM clearly cannot actually detect or analyze properties of the amino acid, adding evolutionary information is a way of accounting for these properties, which is reminiscent of the original prediction methods in history, like Chou Fasman [4].

In conclusion, this predictor, although it does not hold up to state of the art predictors that precede it in respects to accuracy or efficiency, is capable of predicting alpha membrane proteins with a reliable quality. With methods such as those mentioned above, we can modernize how we determine protein structure. This can lead to a large increase in our capability to understand and manipulate more proteins, thereby increasing our understanding of disease mechanism, potential treatments, and much more.

References

[1] Almn, M. S., Nordstrm, K. J., Fredriksson, R., & Schith, H. B. (2009). *Mapping the human membrane proteome: a majority of the human membrane proteins can be classified according to function and evolutionary origin.* BMC biology, 7(1), 50.

[2] Altschul, S. F., Madden, T. L., Schffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.* Nucleic acids research, 25(17), 3389-3402.

[3] Bernsel, A., Viklund, H., Hennerdal, A., & Elofsson, A. (2009). *TOPCONS: consensus prediction of membrane protein topology. Nucleic acids research, 37(suppl 2), W465-W468.*

[4] Chou, P. Y., & Fasman, G. D. (1974). *Prediction of protein conformation.* Biochemistry, 13(2), 222-245.

[5] T. Joachims, 1999 *Making large-Scale SVM Learning Practical. Advances in Kernel Methods* - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press.

[6] Kabsch, W., & Sander, C. (1983). *How good are predictions of protein secondary structure?* . FEBS letters, 155(2), 179-182. Chicago

[7] Krogh, A., Larsson, B., Von Heijne, G., & Sonnhammer, E. L. (2001).*Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes.* Journal of molecular biology, 305(3), 567-580.

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

[9] McGuffin, L. J., Bryson, K., & Jones, D. T. (2000). *The PSIPRED protein structure prediction server.* Bioinformatics, 16(4), 404-405.

[10] Metz, C. E. (1978). *Basic principles of ROC analysis.* Seminars in Nuclear Medicine, 8:283?298.

[11] Michael Droettboom, John Hunter, Thomas A Caswell, Eric Firing, Jens Hedegaard Nielsen, Phil Elson, ? Matt Giuca. (2016). *matplotlib: matplotlib v1.5.1.* Zenodo. http://doi.org/10.5281/zenodo.44579

[12] Moult, J., Pedersen, J. T., Judson, R. and Fidelis, K. (1995), *A large-scale experiment to assess protein structure prediction methods. Proteins,* 23: ii?iv. doi:10.1002/prot.340230303

[13] Scikit-learn: *Machine Learning in Python,* Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[14] Suzek, B. E., Huang, H., McGarvey, P., Mazumder, R., & Wu, C. H. (2007). *UniRef: comprehensive and non-redundant UniProt reference clusters.* Bioinformatics, 23(10), 1282-1288. Chicago

[15] Tsirigos, K. D., Peters, C., Shu, N., Kll, L., & Elofsson, A. (2015). *The TOPCONS web server for consensus prediction of membrane protein topology and signal peptides.* Nucleic acids research, gkv485.

[16] Ying Huang, Beifang Niu, Ying Gao, Limin Fu and Weizhong Li. (2010). *CD-HIT Suite: a web server for clustering and comparing biological sequences.* Bioinformatics, 26:680