SciLife Project

Sarah McComas

March 18, 2016

Abstract

Here will be my abstract it will be cool

1 Introduction

Understanding protein structure is crucial to deepening our knowledge of their function and how diseases affect them and us. One of the most abundant of these are known as transmembrane proteins, which exist across membranes and participate in a variety of 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 how the protein 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 protein structure given the properties of the protein's 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 (3). Since the development of machine learning algorithms and incorporation of multiple sequence alignments, the accuracy of these predictions now can reach 90% or higher (4).

Here I have drawn inspiration from such 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 (5) (as well as information from single sequences in order to compare predictor quality) with information about amino acid presence at a certain position in a structure. The MSA profile is crucial in understanding evolutionary information, and can analyze the certain amino acid through evolution and see how frequent it occurs in the structure. I could then use this information as a learning algorithm for a support vector machine, or SVM (6). 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 (4, 7, 8) 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 and leads to collaboration and

therefore an increase in database size and agreements on protein structure prediction. 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 (9) aim to solve. Many databases continue to separate that which has been computationally generated and what is experimentally proven by traditional methods, but with these improvement programs, 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 in supervised learning. This consisted of about 300 sequences and each entry consisted of the accession number of the protein, the 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 of about 50 sequences each from my database in order to perform cross validation. This is a very important step because otherwise, when the SVM is optimizing the data to try to separate the positive and negative classifications, it will overfit and become too specific to the training example it has been given. In creating a variety of sets in which the SVM can optimize its models, it becomes easier to achieve a good balance between overfitting and achieving a high accuracy of prediction. For this 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 will test on at the very end of the optimization to ensure that I haven't overfit my predictor.

Because alpha membrane proteins are important, it is not uncommon to see conservation in the protein or 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 false accuracy when testing the model because the homologous sequences are too similar and it would be as if I had trained the SVM on the same data I tested it. To avoid this, I input my entire database into CD-HIT (10) which reported back the sequences in groups of homologs. My dataset did not have many but did have some so from this, I made my cross validation sets to ensure that if there were homologs between sequences, that they would end up in the same cross validation set.

The SVM was trained in two separate manners. Firstly, the single sequence information was used. This was not expected to create a high quality predictor but it is important to compare to the MSA profiles to this. ******** (GIVE MORE INFO) ********** The MSA profile was created from running PSI-

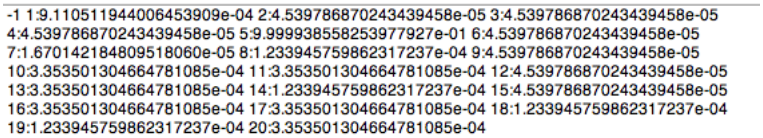

Figure 1: Each line began with a target value (a positive or negative number depending on structural position). Show maybe the PSIBLAST output. explain what the PSSM output looks like, the amino acid number, and what a single sequence output would look like instead.

BLAST (5) locally. This search took several days to complete, but once finished, a position specific scoring matrix, or PSSM, was generated for every amino acid in the sequence. This PSSM correlated to each amino acid's predicted presence in evolution at that certain position. This is the main difference between the MSA data for the SVM and the single sequence information; the single sequence information merely contains a 1 or 0 depending on 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) and must therefore be between 0 and 1. Before the BLAST output 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 all of my data would usually take about 8 to 12 hours. I began first by training the SVM on the single sequence data but did not further pursue any optimization. From SVM$^{\mathrm{light}}$ (6), 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, which can customize the kernel. This 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).

ACC and MCC here??

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. *******DONT FORGET TO MENTION ACC AND MCC LATER ON*****. 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 doing:

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

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 trick, along with default parameters. I have tested several other parameters with both this and the linear kernel for my predictor, but none of them reached even close to 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. According to the SVM$^{\text{light}}$ package information online, there are several means 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). 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.

One other means of testing the predictor quality is via the Receiver Operating Characteristics curve, or the ROC curve. 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. I made this ROC curve, as well as the aforementioned bar graph in Figure 2 with the help of toolkits from scikitlearn (13) and matplotlib(14). The most important feature of the ROC curve dictating predictor quality is the AUC, the area under the curve. 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 higher the AUC, the better.

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 Figure 4, I compared the averages to the data

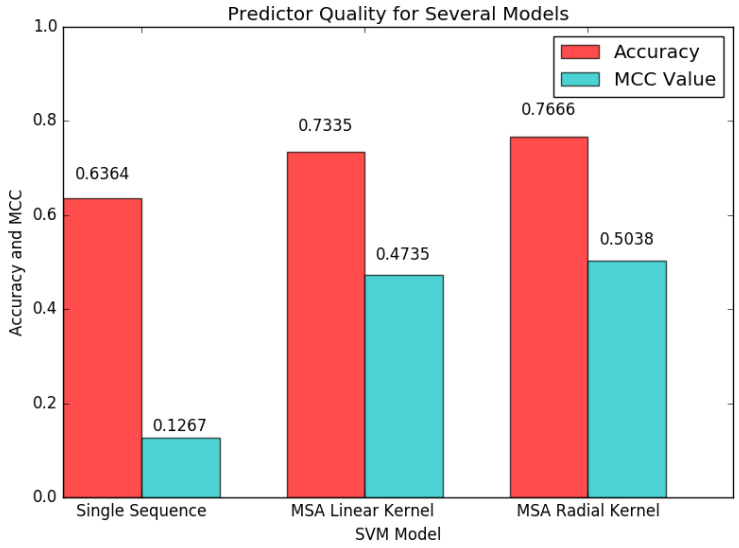

Figure 2: A simple caption

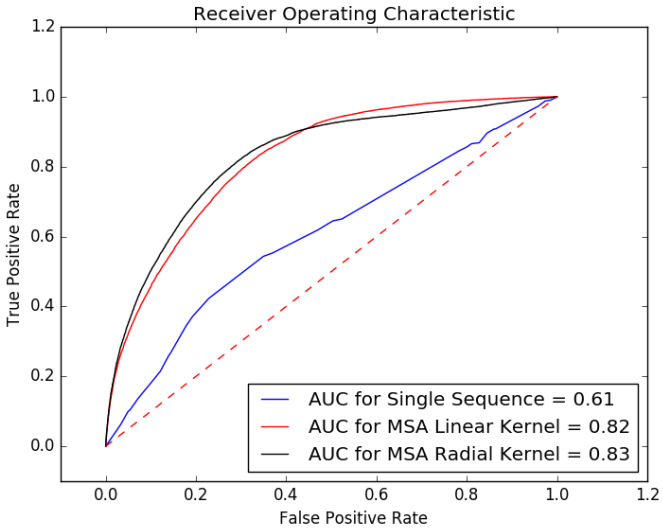

Figure 3: does this work

	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

Figure 4: should this be a table?

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. ****Should this be in discussion? **** This decrease in quality could be attributed to several factors. This could be happening because of overfitting my models, which happens if one 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. Therefore, it is likely that this decrease is due to poor chance, and a more random and hard to predict validation set.

4 Discussion

My predictor is good, significantly better than random, but only when the right data is applied. To test this, I **********. 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 from a variety of sources. A severe limitation for my predictor is time. When testing the **********, for one sequence, it took **********. TOPCONS and the like can perform this in under one minute. These programs are capable of doing this for a variety of reasons, one being that they use consensus prediction, which is a method of combining other predictor outputs into one predictor, thus increasing the accuracy greatly because other models agree with each other in the prediction. Predictors such as TOPCONS also have increased their speed by differentially searching databases rather than searching through all of the UniRef (15) database, which my predictor does when generating an MSA. This is the main reason that my predictor is slower, for TOPCONS also runs on a normal 4 core processor as my machine does. When my predictor runs on single sequence information only, the process is significantly faster. This is firstly because it is not running a PSI-BLAST search across the UniRef database, and also because the SVM has much less information to process. However, as the results have shown, this is not 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 so important because it adds a much more in depth analysis of each amino

acid at a certain position. This takes 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 oldest prediction methods in history. ****Chou fasman**** In making my predictor better and more like the state of the art predictors, I could also develop an algorithm to search between different databases depending on the sequence and the results found for it, which could reduce at least the amount of time that the BLAST is running. However, my main limitation would then by the SVM classifier, which takes several minutes (at least 5 for MSA profile classification) to finish. Optimizing my parameters for the SVM might increase the accuracy but it would do little to increase the efficiency of the predictor. If I am not comparing my predictor to those that are state of the art, my predictor can dictate alpha membrane topology fairly well. In comparing the output of a test file with a known topology, the patterns are very clear and nonrandom. For example, there are several concentrated spots in the sequence where the protein is in the membrane, a much more reasonable model of actual protein topology and is to be expected over 2 or 3 random segments of the sequence which are supposedly in the membrane.

5 References