Predicting protein trans-membranal($\alpha - helical$) domain using HMM

Oren Sultan, Roee Lieberman, Maria Tseytlin, Roei Zucker, Lee Lankri

26-28/02/21

1 Background

Trans membranal proteins are among the most crucial proteins for the survival of the cell. Estimated to comprise 27% of the proteins found in humans, transmembrane proteins perform crucial roles such as transport of nutrients and cellular communication. As such, the ability to identify and predict the transmembrane domain of proteins is very attractive, as it is helpful in predicting a proteins function. An amino acid in a trans membranal protein have two distinct hidden phases that it can be in – either it is inside the membrane or outside of it. The acid also has multiple known phases it can be in (which type of amino acid) which can be considered dependent on the hidden phase. In this work we focused on predicting $\alpha - helical$ transmembrane proteins by using HMM.

2 Research Question & Objective

Is it possible to predict the $\alpha - helical$ trans membranal domain in proteins using HMM?

3 Data

3.1 Source

Our source database is in format of XML. The data is taken from PDBTM(6)

3.2 Properties

- **CHAINID:** the chain identifier
- **NUM_TM:** the number of transmembrane segments
- **TYPE:** the type of transmembrane segments or the type of the chain if it does not cross the membrane (non_tm) or if it is not a protein chain (lipid), we take only "alpha" segments.

Every chain contains the following properties:

- **SEQ:** the sequence of the protein
- **REGION:** locates the chain segment in the space relative to the membrane. We are looking only on TYPE="H" which is $\alpha - helical$ region. We take the "seq_beg" and "seq_end" to know the starting position and ending position of the region in the sequence. In the next figure we can see the structure of the CHAIN element with an example:

```
<CHAIN CHAINID="A" NUM_TM="7" TYPE="alpha">
    <SEQ>
        AVRENALLSS SLWVNVALAG IAILVFVYMG RTIRPGRPRL IWGATLMIPL
        VSISSYLGLL SGLTVGMIEM PAGHALAGEM VRSQWGRYLT WALSTPMILL
        ALGLLADVDL GSLFTVIAAD IGMCVTGLAA AMTTSALLFR WAFYAISCAF
        FVVVLSALVT DWAASASSAG TAEIFDTLRV LTVVLWLGYP IVWAVGVEGL
        ALVQSVGATS WAYSVLDVFA KYVFAFILLR WVANNERTVA VAGQTLGTMS
        SDD
    </SEQ>
    <REGION seq_beg="10" pdb_beg="31"  seq_end="31" pdb_end="52"  type="H"/>
    <REGION seq_beg="39" pdb_beg="60"  seq_end="59" pdb_end="80"  type="H"/>
    <REGION seq_beg="85" pdb_beg="106"  seq_end="104" pdb_end="125"  type="H"/>
    <REGION seq_beg="110" pdb_beg="131"  seq_end="131" pdb_end="152"  type="H"/>
    <REGION seq_beg="139" pdb_beg="160"  seq_end="160" pdb_end="181"  type="H"/>
    <REGION seq_beg="177" pdb_beg="198"  seq_end="196" pdb_end="217"  type="H"/>
    <REGION seq_beg="209" pdb_beg="230"  seq_end="230" pdb_end="251"  type="H"/>
</CHAIN>
```

Figure 1: Dataset's structure

3.3 Size

- **27300** chains
- **90365** $\alpha - helical$ regions in all sequences together(each sequence may include 0 or more $\alpha - helical$ regions
- **20352** protein sequences: we filter sequences which include only AMINO_CHARACTERS = ['R', 'H', 'K', 'D', 'E', 'S', 'T', 'N', 'Q', 'C', 'U', 'G', 'P', 'A', 'V', 'I', 'L', 'M', 'F', 'Y', 'W'], notice that we have one-to-many relationship between chains and sequences.

4 The Model

We used HMM architecture. We wanted our HMM to be able to sample both single transmembrane $\alpha - helical$ (bitopic) and polytopic transmembrane $\alpha - helical$ protein, as we can see in the following figure:

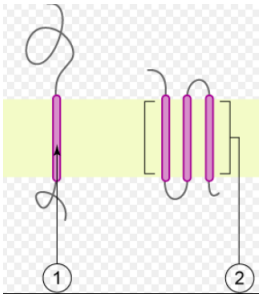

Figure 2: Schematic representation of transmembrane proteins: 1) a single transmembrane $\alpha - helical$ (bitopic membrane protein). 2) a polytopic transmembrane $\alpha - helical$ protein.

In addition, we wanted our model to be sensitive to the transmembrane regions length. That is why we choose the following architecture:

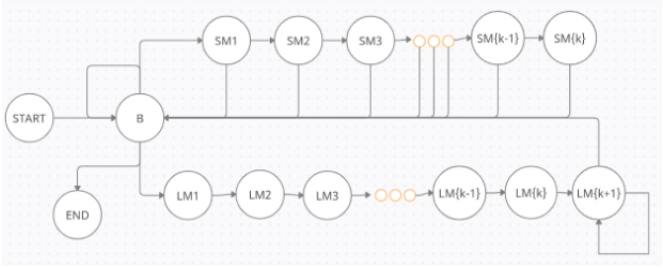

Figure 3: The HMM Model. $'B'$-Background, $'SM'$-short motif, $'LM'$-long motif. In our case $k = 30$.

This architecture inspired from a similar model for evaluating nuclasom regions in DNA(1) which is able to express the two structures and the fact that the length is variable. There are much more complicated models including HMMs, Deep learning and more$(2, 3)$, that solve the same problems as ours. But, the motivation for this project was to practice HMMs ,with context to biology, in a short period of time(hackathon). So, in order to be able to analyze the model's behavior more easily we stick with this simple architecture.

Our model is trying to differentiate between two major stages: an 'in' stage where the protein is inside the membrane and an 'out' state(background stage) where the protein can be either inside the cell or outside the cell. In both stages the sequence length is not constant, which is why we used two different emissions for these stages. In the actual model the $'B'$ stage model the $'out'$ state and all the other $'SM'$ and $'LM'$ stages model the $\alpha - helical$ that go through the cell membrane. The $'SM'$ stands for short motif and these stages are for the more likely motif length as seen in the data. In the next table we can see the probability distributions of different length of motifs:

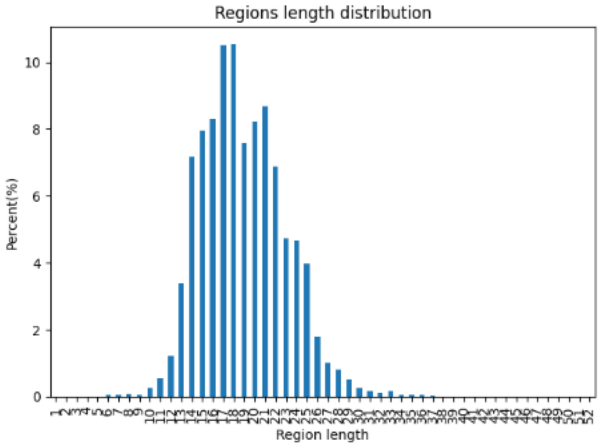

Figure 4: Regions length distribution- more likely motif lengths are < 30 so we chose $k = 30$.

We can see that the regions of $\alpha - helix$ in the sequences have length between 1-52 and the length in the range of 13-25 is the most common. We chose $k = 30$ to be the threshold between $'SM'$ to $'LM'$. Each $'SM\{i\}'$ node has the transition options of, going back to the $'B'$ stage (end of current motif) or keeping to $'SM\{i + 1\}'$, the aim of this architecture is to give the model better control in these motifs' length. The $'LM'$ are for the rest and longer possible motifs each $'LM\{i\}'$ can transit only to $'LM\{i + 1\}'$ except the last $'LM'$ which can go to $'B'$ or to keep going to itself, which makes the model able to sample any length of motif but with less control in these lengths.

5 The Algorithm

We examined two different algorithms, the Baum-Welch algorithm as we know from class and a supervised training algorithm which takes advantage of our labeled data that we generated. Even though we chose the Baum-Welch(section 5.2) we would like to explain how we run the supervised algorithm settings(next section), and later we will explain about the Baum-Welch algorithm(section 5.5)

5.1 The supervised algorithm

We used supervised training to train and initiate the HMM. We generated labels for the regions in the protein sequences according to the model architecture we explained, by using the data of the regions in the sequences gathered from the PDBTM. In the next figure we can see an example of observation and the label:

```
$,M,E,V,N,Q,L,G,F,I,A,T,A,L,F,V,L,V,P,S,V,F,L,I,I,L,Y,V,Q,T,E,S,Q,Q,K,S,S,^
start,B,B,B,B,B,B,B,B,B,B,B,SM1,SM2,SM3,SM4,SM5,SM6,SM7,SM8,SM9,SM10,SM11,SM12,SM13,B,B,B,B,B,B,B,B,B,B,B,end
```

Figure 5: Example for observation(the first row) and label(the second row) from the generated training data

Every observation starts with special character of '$' and ends with '^' and in between are the characters of the sequence of the protein. Every label starts with 'start' and ends with 'end' and in between we have $'B'$ for characters which are out of the $\alpha - helical$ region, and here since the motif's length is 13 which is < 30 we have $SM1, ..., SM13$. Notice that for longer than 30 regions we will see $LM1, LM2, ...$ and it can also happen that we have several regions in the sequence so we will see the counting starts again, for example for two regions with length < 30: $SM1, SM2, ...SM < lengthRegionOne >$,..., $SM1, SM2, ..., SM < lengthRegionTwo >$. Since we are interested only in predicting $'B'$(Outside the region) or $'I'$(Inside the region) after the model's prediction we wrapped it and replaced every non $'B'$ with $'I'$ so when evaluating the model we compare between two strings (actual vs. predicted) with the optional values: $'B'$ or $'I'$ for every index (more details in Results section) as can be seen in the next figure:

```
$,M,E,V,N,Q,L,G,F,I,A,T,A,L,F,V,L,V,P,S,V,F,L,I,I,L,Y,V,Q,T,E,S,Q,Q,K,S,S,^
start,B,B,B,B,B,B,B,B,B,B,B,I,I,I,I,I,I,I,I,I,I,I,I,B,B,B,B,B,B,B,B,B,B,B,end
```

Figure 6: Converting the label to include only $'B'$ for background and $'I'$ for the motif region(will be done also to the prediction)

So, in this settings we have state labels for each observation and we wish to derive the transition matrix and observations given these labels. The emissions simply become MLE estimates of the data partitioned by the labels and the transition matrix is calculated directly from the adjacency of labels. Therefore the transition and emission probabilities can be computed using the MLE.

5.2 The algorithm we chose - Baum-Welch

We examined both, the advantages of the supervised algorithm is that it runs faster than the Baum-Welch which is iterative algorithm. But, on the other hand we found that on larger test set the Baum-Welch stay stable with 85% accuracy where the supervised is going down. Finally, we prefered to practice the Baum-Welch instead of a supervised ML algorithm because it's more relevant to the course.

5.3 Notations

Let's denote the following:

$L = \{l_1, l_2, ..., l_{|L|}\}$ = the set of possible observations

$S = \{s_1, s_2, ..., s_{|S|}\}$ = the set of all of the hidden states
θ = the HMM parameters E and T
$E(l, s)$ = the probability to emit observation l in state s
$T(s, s')$ = the transition probability from state s to state s'
$O_i = \left(o_1^i, o_2^i, ..., o_{|O_i|}^i : o_j \in L\right)$ = the $i-th$ observation from the data set
$X_i = \left(x_1^i, ..., x_{|O_i|}^i : x_j \in S\right)$ = the label for the $i-th$ observation where x_j is the state that emits o_j
$D = \{D_i = (O_i, X_i)\}_{i \in [k]}$ = the data set
$e(l, s) = \left|\left\{\left(o_j^i = l, x_j^i = s\right) : i \in [k], j \in |O_i|\right\}\right|$ = the number of states $s$that emits the character l in the data
$t(s, s') = \left|\left\{\left(x_{j+1}^i = s', x_j^i = s\right) : i \in [k], j \in |O_i| - 1\right\}\right|$ = the number of s to s' transition in the data

5.4 The supervised training algorithm - labeled data settings

From log-likelihood definition:

$$LL(\theta; D) = log(Pr(D \mid \theta))$$

The observed sequences are independent, hence:

$$Pr(D \mid \theta) = Pr(\{D_1, ..., D_k\} \mid \theta) = \prod_{i=1}^{k} Pr(D_i \mid \theta)$$

From HMM definition:

$$P(D_i|\theta) = \left[\prod_{j=1}^{|X_i|-1} E\left(o_j^i, x_j^i\right) \cdot T\left(x_j^i, x_{j+1}^i\right)\right] \cdot E\left(o_{|X_i|}^i, x_{|X_i|}^i\right)$$

From all the above we get

$$LL\left(\theta|D\right) = \sum_{i=1}^{k}\left[\sum_{j=1}^{|X_i|-1} log\left(E\left(o_j^i, x_j^i\right)\right) + log\left(T\left(x_j^i, x_{j+1}^i\right)\right)\right] + log\left(E\left(o_{|X_i|}^i, x_{|X_i|}^i\right)\right) =$$

$$= \sum_{l \in L, s \in S} log\left(E\left(l, s\right)\right) \cdot e\left(l, s\right) + \sum_{s, s' \in T} log\left(T\left(s, s'\right)\right)$$

Now we can differentiate and find the θ that maxsimaize the log likelyhood which is

$$\hat{E}(l', s') = \frac{e(l', s')}{\sum_{s \in S} e(l', s)}, \hat{T}(s_1, s_2) = \frac{t(s_1, s_2)}{\sum_{s \in S} t(s_1, s)}$$

These estimators can be used with labeled data as supervised learning.

5.5 The Baum-Welch algorithm - unlabled data settings

The Baum-Welch algorithm

1. init θ (randomly) and ε small as you like.
2. for each iteration:
   (a) $old\_\theta = \theta$

(b) for each X_j sequence calculate the forward and backward tables

(c) E phase: estimate $e\ and\ t$ from the forward and backward tables and from old_θ

(d) M phase: estimate $E\ and\ T$, $\theta = \left[\hat{E}, \hat{T}\right]$

3. stop when $LL(\theta) - LL(old\_\theta) < \varepsilon$

The Baum-Welch algorithm works as a private case of an EM agorithm. The EM algorithm is to maximize $Q(\theta \mid \theta_{old}) = E_{Z|O,\theta_{old}}[LL(\theta, O, Z)]$ where Z is the current ditribution on the sequences according to θ_{old}. In our case the $Q(\theta \mid \theta_{old})$ which is the mean of the LL of the sample under the old_θ, is equivilent to how we estimate $\hat{E}, \hat{T}$. We'll prove this algorithm correctness with loop invarient L.I. - after each iteration $LL(\theta) \geq LL(old\_\theta)$ and $LL(\theta) \leq 0$. If the L.I. is true, therefore the LL as a series of θ_i(the series of the θ by the loop iterations) if a monotonic and bounded therefore it converge.

$$log\ P(O|\theta) = log\ P(O, Z|\theta) - log\ P(Z|O, \theta)$$

$$E_Z[log\ P(O|\theta)] = E_Z[log\ P(O, Z|\theta) - log\ P(Z|O, \theta)]$$

$$log\ P(O|\theta) = \sum_Z P(Z|X, \theta_{old})\, log\ P(O, Z|\theta) - \sum_Z P(Z|O, \theta_{old})\, log\ P(Z|O, \theta) =$$

$$= Q(\theta|\theta_{old}) + H(\theta|\theta_{old})$$

we substract $log\ P(O|\theta_{old}) = Q(\theta_{old}|\theta_{old}) + H(\theta_{old}|\theta_{old})$ from what we got to get the following:

$$log\ P(O|\theta) - log\ P(O|\theta_{old}) = Q(\theta|\theta_{old}) - Q(\theta_{old}|\theta_{old}) + H(\theta|\theta_{old}) - H(\theta_{old}|\theta_{old})$$

from gibbes inequalty $H(\theta|\theta_{old}) \geq H(\theta_{old}|\theta_{old})$ there

$$LL(\theta) - LL(old\_\theta) = log\ P(O|\theta) - log\ P(O|\theta_{old}) \geq Q(\theta|\theta_{old}) - Q(\theta_{old}|\theta_{old})$$

from here if we choose θ that improve Q than the LL improves also. Q can be differentiate and therefore can be improved easily.

6 Results

6.1 Model's evaluation by Confusion Matrix

For the testing of the model we used 2000 samples from the PDBTM (6) as a train group, and 400 samples as test group. After training we used the model to predict the Hidden states of the test group, and measured the results using different parameters. We first, evaluated the results by a confusion matrix as can be seen in the next figure.

	Predicted No	Predicted Yes	
Actual No	TN=65877 65.8%	FP=6592 6.58%	72469
Actual Yes	FN=7612 7.6%	TP=20039 20.01%	27651
	73489	26631	

Figure 7: The Confusion matrix

The Confusion matrix, representing the tagging of each amino acid in each sequence in the test sequences. Where Positive/Yes represents an amino acid being inside the membrane (motif), and Negative/No means outside of the membrane. The percentile represents the respective value divided by the overall amino acids. By using the confusion matrix we can derive:

- **Precision**:

$$\frac{TP}{TP+FP} = \frac{20039}{20039+6592} = 0.75$$

- **Recall**:

$$\frac{TP}{TP+FN} = \frac{20039}{20039+7612} = 0.72$$

6.2 Other testing methods

6.2.1 match rate:

We wanted to find the correct match rate for different assignments $\left(\frac{\text{correct labeling}}{\text{overall labeling}}\right)$. Since we saw that our overall success rate was **85%** (by summing the diagonal of the confusion matrix) we decided to test the match percentile of each sequence, and use them to determine if there are specific parameters that affect our sucess rate. We decided to test the match rate relative to the length of a protein sequence, and to the number of motifs, while most sequences were matched rather successfuly (mostly above 80%), there are inconsistencies by the different parameters, especially noticeable in the erratic changes when measuring by sequence length. We assume that the factor which causes it is either not among those we tested, or is too complex to predict using our selected model. We did notice however that shorter sequences (below 600 amino acids) can be more erratic than longer sequences, though it might be caused by the fact that shorter sequences are more common, and suprisingly the long sequences and those with a large amount of motifs had similar if not better success rate. In the next figure we can see the match rate by sequence length (A) and by number of motifs (B):

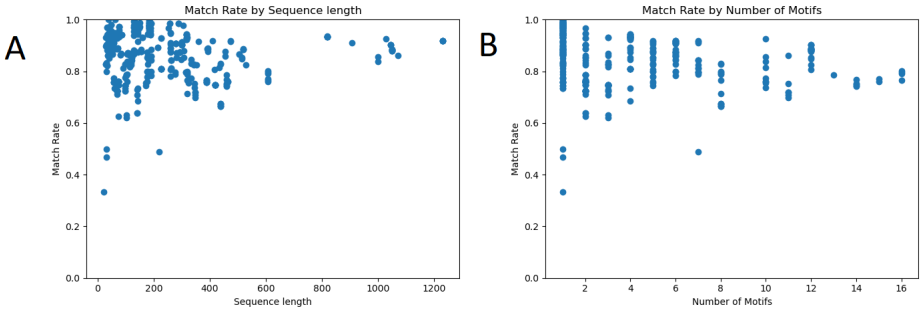

Figure 8: A: Measuring the match rate relative to the length of a sequence. B: Measuring the match rate relative to the number of transmembrane motifs in the sequnece

6.2.2 false-positive:

Next we decided to determine how likely we are to produce False Positive results relative to the same parameters. For each labeled sequence we counted the amount of times we labled a background/outside acid as a motif/inside acid. Because a longer sequence has more labels that could be wrong, we tested the relationship between the length of the sqeunece and the number of motifs, to the number of FP assignments, and the rate of FP assignments normalized by the length of the sequence. The results were quite similar to those of the match percentile, while longer sequences expectedly had more overall FP lables, when the number of FP was normalized by the length of the sequence, the rate of FP was less erratic for smaller sequnces, and similar for sequnces when averaged by the number of motifs, see figure below:

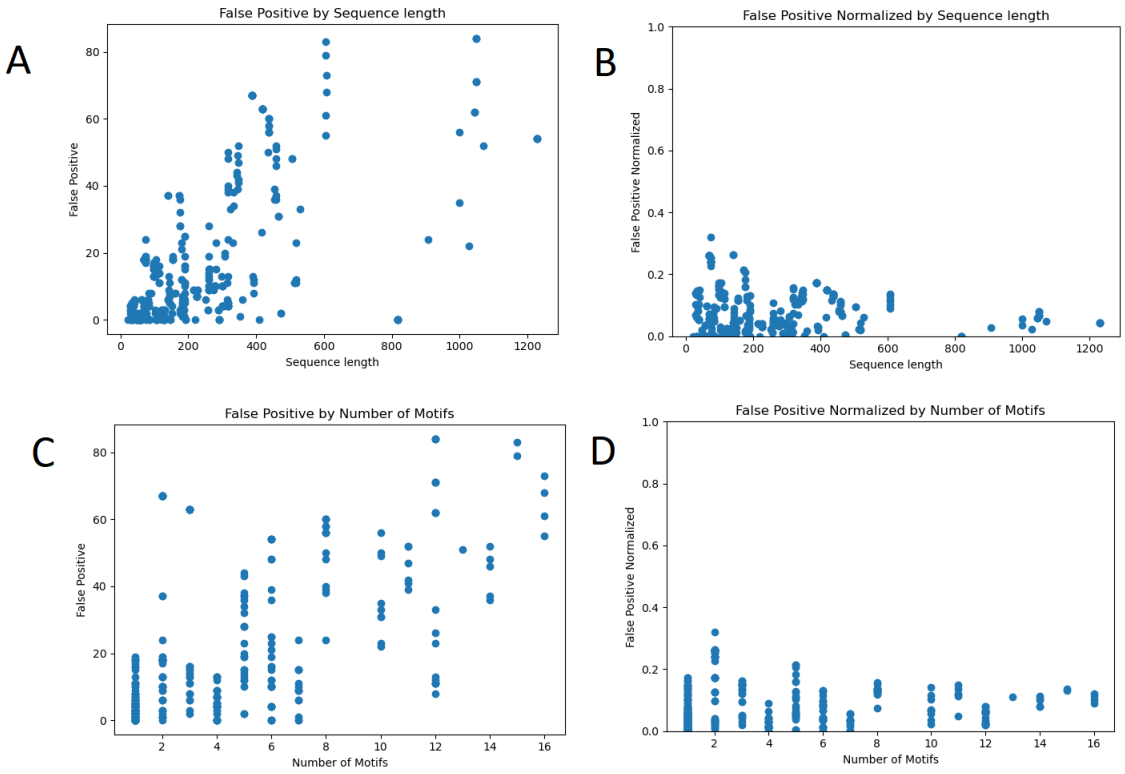

Figure 9: A: Measuring the overall FP relative to sequence length. B: Measuring the FP rate relative to sequence length. C: Measuring the overall FP relative to number of motifs. D: Measuring the FP rate relative to the number of motifs

6.2.3 number of motifs accuracy:

Since the number of times the protein crosses the membrane can have a massive effect on the protein structure, we decided to test how accurately we are able to predict the number of transmembrane motifs for a specific sequence (meaning, how many times a specific protein will cross the membrane). As before, the results showed greater variation for shorter sequneces when measured by length, and relatively lesser variation when measured by number of motifs.

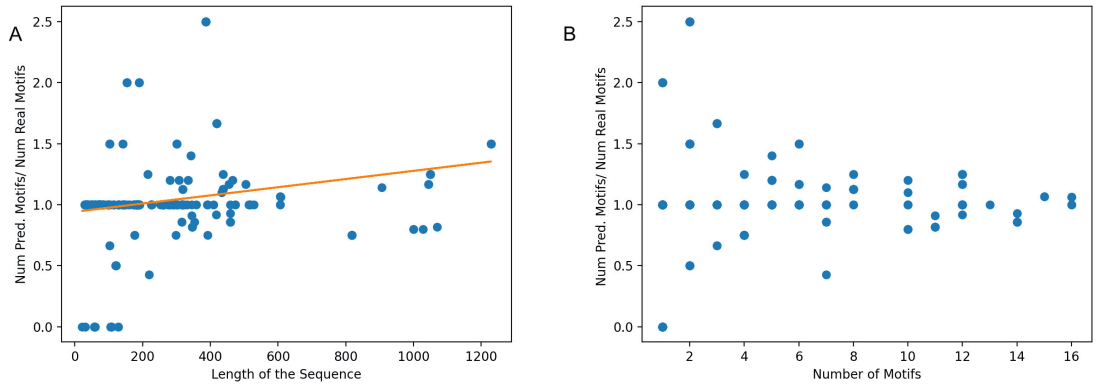

Figure 10: A: The ratio between predicted number of motifs and the real number of motifs. B: The ratio between predicted number of motifs and the real number of motifs.

6.2.4 sequence alignment:

To get another measure of how similar our predicted sequences were to the actual ones, we preformed a sequence alignment test on our predicted sequences. This test gives us a better indication of the predicted general structure of the proteins and not just the prediction of the state of each amino acid. We constructed a score matrix as such: 1 point for match, -1 for mismatch, and -2 for gap. Because a longer sequence has more instances that could be wrong, we examined the relationship between the average sequence alignment score (and the same score normalized by the length of the sequence) and the sequence length. The model was able to predict well more than half of the sequence for most lengths. For a decent number of lengths, the model did very well and predicted accurately most of the sequence, with a several nearly perfect scores. This indicates that the model was able to identify and assign quite well the appropriate states.

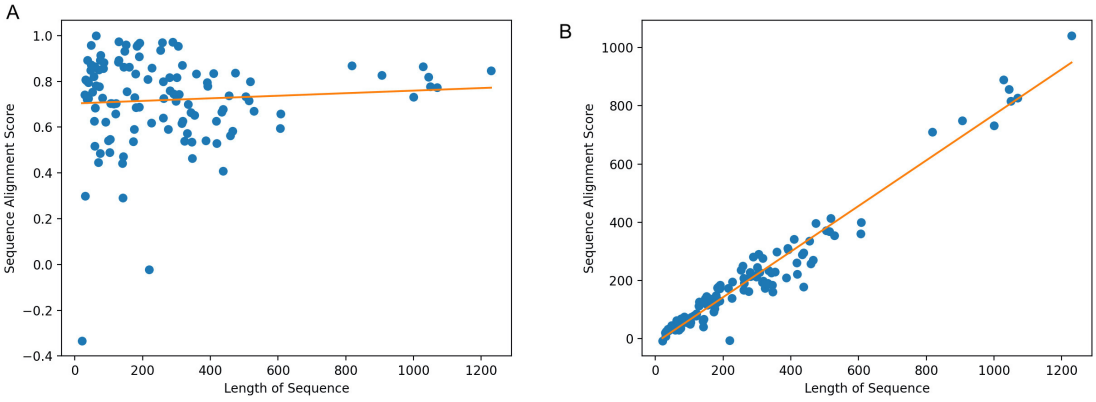

Figure 11: A : The average sequence alignment score normalized by the length of the sequence, for each sequence length. B: The average sequence alignment score for each sequence length.

7 Conclusions

- In this project we wanted to create a model that predicts $\alpha - helical$ transmembranal domains for specific protein sequence. While most of our predictions were reletively accurate, there are places where the model is a bit lacking. In all tested parameters we found that shorter sequences are more likely to show erratic behavior, while it could be caused by the fact that short protein sequnces are more common, it is also possible that shorter sequences show a less predictable behavior.
- It should also be noted that to make the model less complex, we elected to only predict $\alpha - helical$ transmembranal regions. While simpler, it might also reduce our ability for prediction, as the existance of multiple distinct models tagged as background might hinder our ability to determine what is a background.
- If given more time we would have expanded our model to include different motifs, and maybe create hidden states that are more specilized to specific motifs.

8 References

1. Yuan, G.C., Liu, Y.J., Dion, M.F., Slack, M.D., Wu, L.F., Altschuler, S.J. and Rando, O.J., 2005. Genome-scale identification of nucleosome positions in S. cerevisiae. Science, 309(5734), pp.626-630.
2. Reeb, J., Kloppmann, E., Bernhofer, M. and Rost, B., 2015. Evaluation of transmembrane helix predictions in 2014. Proteins: Structure, Function, and Bioinformatics, 83(3), pp.473-484.
3. Sonnhammer, E.L., Von Heijne, G. and Krogh, A., 1998, June. A hidden Markov model for predicting transmembrane helices in protein sequences. In Ismb (Vol. 6, pp. 175-182).
4. Schreiber, J., 2017. Pomegranate: fast and flexible probabilistic modeling in python. The Journal of Machine Learning Research, 18(1), pp.5992-5997.
5. Cock, P.J., Antao, T., Chang, J.T., Chapman, B.A., Cox, C.J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B. and De Hoon, M.J., 2009. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11), pp.1422-1423.
6. Bioinformatics 20, 2964-2972; Nucleic Acids Research 33 Database Issue, D275-D278; Nucleic Acids Research 41 Database Issue, D524-D529