Accuracy of machine learning vs manual methods in mapping protein secondary structure to sequence identity

Karen Gaffney & Isiac Orr

List of research questions

1. How accurately can secondary structure be predicted from sequence identity alone using machine learning?
2. How accurately can secondary structure be predicted from sequence identity alone using manual methods?
3. What are the optimal parameters for the manual classifying method?

Motivation

The protein function of a significant part of sequenced genomes are unknown. This is due to the high variability in secondary structures and functions that can result from the same subsequence. Consequently, a computational method for estimating secondary based on sequence provides a potent tool in resolving the structure and function of yet uncharacterized protein sequences.

Dataset

We are utilizing two data sets in this project. *Protein Secondary Sequence* is a dataset curated by Tamzid Hasan who transformed Protein Data Bank (PDB) sequence data with DSSP (hydrogen bond estimation algorithm) to calculate secondary structure from PDB entries. We also incorporated a manually made data set that links each amino acid to its propensity to adopt alpha helix to beta strand secondary structure.

Protein Secondary Sequence: https://www.kaggle.com/tamzidhasan/protein-secondary-sequence

Methods

Data

We utilized a data set from Kaggle that combined amino acid sequence data from the PDB with amino acid sequence data ran through DSSP (a hydrogen bond estimation algorithm) that predicted secondary structure. To load our data into colab, the Kaggle data set was downloaded to google drive. Goggle.colab import drive was utilized to mount the drive to the script and import the files. The relevant columns were extracted from Kaggle: sequence, secondary structure, and pdb_id. The sequence was simplified to only three secondary structure confirmations for simplicity A, B and V. And the last letter of the pdb_ids were scrapped to allow for PDB query later on if necessary.

In addition to the Kaggle data we also imported a csv file of amino acid proposenties to incorporate with our manual classification methods.

Manual Secondary Structure Estimator

To begin manually classifying the amino acid sequences, the frame lengths to be tested must be determined. In the code, they are stored in a user-entered list. The frame refers to how many subsequent amino acids are classified as a group. The frame length can then be passed to a manual classification method that will run structure estimation for each frame length.

The manual classification method runs an estimation for secondary structure for each frame length. To begin this process, the method creates an empty list for mean squared errors, as well as a dictionary to store the dataset formed with each frame length. Then the method calls several helper methods to complete the classification process.

The first of these helper methods determines a threshold used later to determine whether the frame is A, B, or V. This threshold is calculated by taking the average alpha helix propensity, and the average beta strand propensity and multiplying them by the frame length. Then one standard deviation for each is added. This is done so that unless the frame scores one standard deviation above average in the A or B category, it will be classified as A.

The second helper method computes the actual structure estimate. Iterating through each sequence, it pulls a group of subsequent amino acids according to the given frame length. It then calculated an A score and B score for that frame. If the A score is higher than the threshold, then “A” is added to the estimated structure for length of the frame. If this threshold is not met, then B is assigned if the b score is above the threshold. If neigher score meets the threshold, then the frame is set to V. The estimation is built sequentially as the loop parses frames. These estimations are stored in a separate data frame that is returned.

The next helper method checks the accuracy by comparing each letter code of the actual secondary structure, simplified to A, B, and V from the kaggle dataset. It keeps a running total of the number of incorrect structure letter-codes and returns these values normalized to the respective sequence length. It then appends this to the simplified kaggle data and returns it.

After the error and estimations have been generated and returned, the data frames are merged into one, and then sent to a mean squared error (mse) calculator that calculates a mean squared error for all of the sequences estimated using the current frame length. These are appended to a list of mse’s corresponding to different frame rates.

From this master manual secondary structure method, the mse and full data set are returned. Once returned they are fed to a visualizer method that uses three separate helper methods to visualize the error vs sequence length, the mse vs frame length, as well as the homogeneity of frames in the sequence for each frame length.

Machine Learning Classifier

To begin with our machine learning classifier, the data must first be prepped. The ml_data_prep function cuts the amino acid sequence of each protein into n frame lengths and then converts each amino acid code value into an integer representative, per the conversion dictionary. This allows us to look at the effect of frame size on accuracy and use a decision classifier tree by encoding the categorical data as integers. The resulting integer sequence is stored in n columns, each column representing the amino acid’s position in the frame sequence.

Next, in ml_structure_estimator_2, the classification model is trained on a smaller dataset, using a Decision Tree Classifier. The features are set as the positional columns with encoded amino acid data. The labels are the secondary sequence structure sourced from Kaggle. Then, the data is tested on a larger data set and produces predictions. The predictions are stored in an additional column.

To evaluate the accuracy of our data, the ml_accuracy function calculates the error of each protein sequence by finding the number of incorrect secondary structure predictions divided by the total length of the secondary structure. The mean squared error is calculated for each individual sequence as well.

This process is iterated for each frame length determined in the main() call. The data frame of mean squared error calculated for each protein sequence is concated into one master data frame. We can visualize the accuracy of our ML platform with the ml_visualizer. The ml_visualizer plots the mean squared error across the frame lengths and provides a confidence interval for each frame.

Results

Manual Secondary Structure Estimator

It is known that most secondary structures are around 4-16 amino acids in length. Knowing this, the manual estimator was tested using frame lengths of 2, 4, 6, 8, 10, 12, 14, and 16. As described above, the thresholds used to determine A or B were calculated by adding one standard deviation to the mean propensity value multiplied by the frame length for each propensity. The results from this are shown in figure 1.

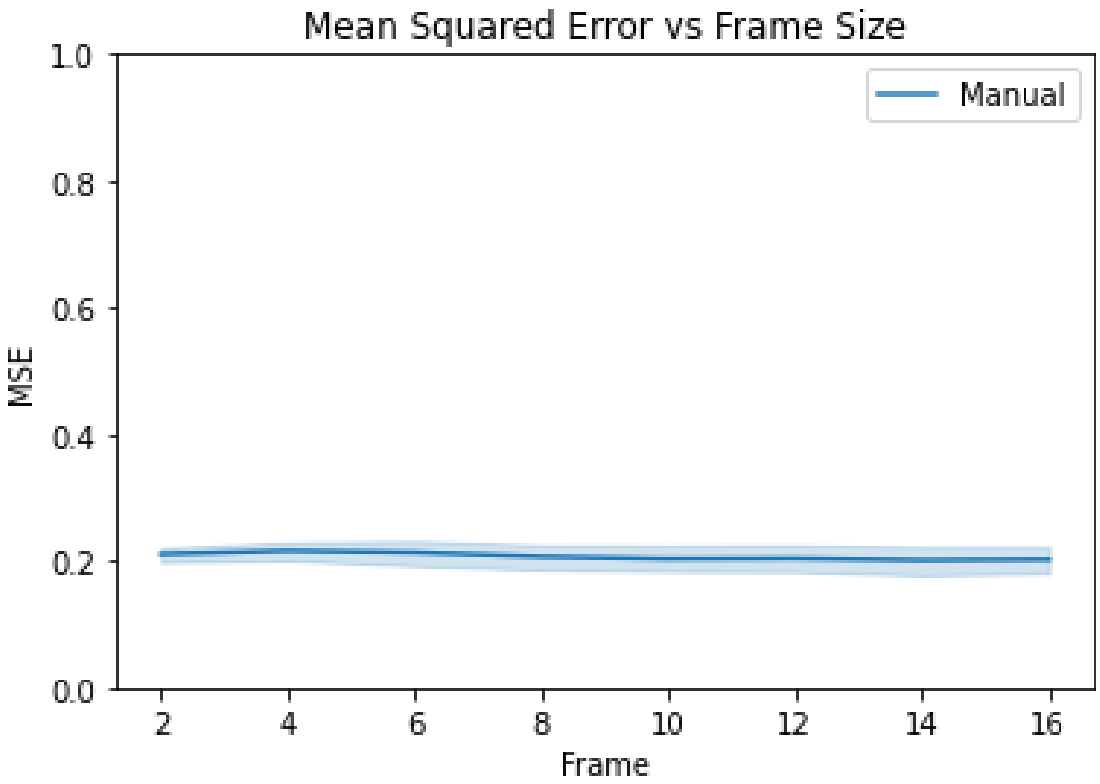

Figure 1: The mse vs frame length for the manual secondary structure estimator with 100 sequences estimated

In terms of mse, the manual method appears to have a maximum performance with ~20% error as can be seen in figure 1. This equates to an accuracy of 80%. With three categories, the random chance success is ~33% so the model performs better than random chance by 47%. Interestingly, the graphs suggests that frame length doesn't

Compared with the ML data shown later, this error margin is very low. It also shows little variation between frame sizes. This is far from what we expected. It is possible that it is something specific to the 100 sequences that are tested. For example, the sequences could be particularly difficult for the ML model to predict. Another possibility is that it is an artifact of the manual classifier calculations.

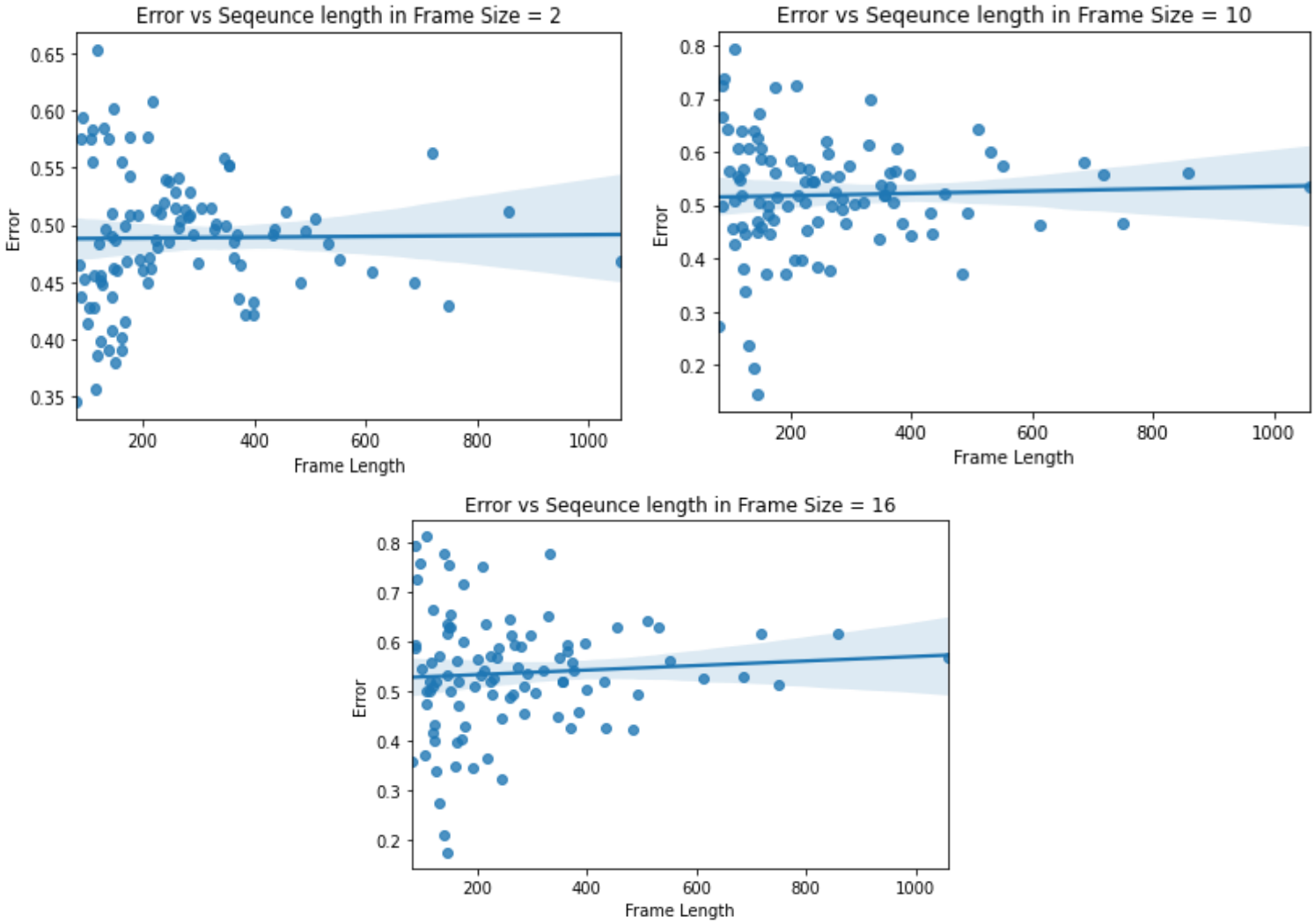

Figure 2: Error vs sequence length for three different frame lengths with 100 sequences estimated

There appears to be no significant relationship between sequence length and error as seen in figure 2. The line of best fit is completely horizontal, and the data points appear to be heavily biased toward lower sequence lengths. These results suggest that the estimator performs the same regardless of sequence.

This is not what we would have expected. We were expecting to have error get worse as sequence length increased because shorter proteins would be less variable. For example, it seems like it would be logically more likely that a smaller sequence is all an alpha helix, which thus makes it easier for the manual method to estimate. The results seen in this report seem to point to smaller sequences have comparable variability to longer ones, such that the manual estimate has consistent error margins for each.

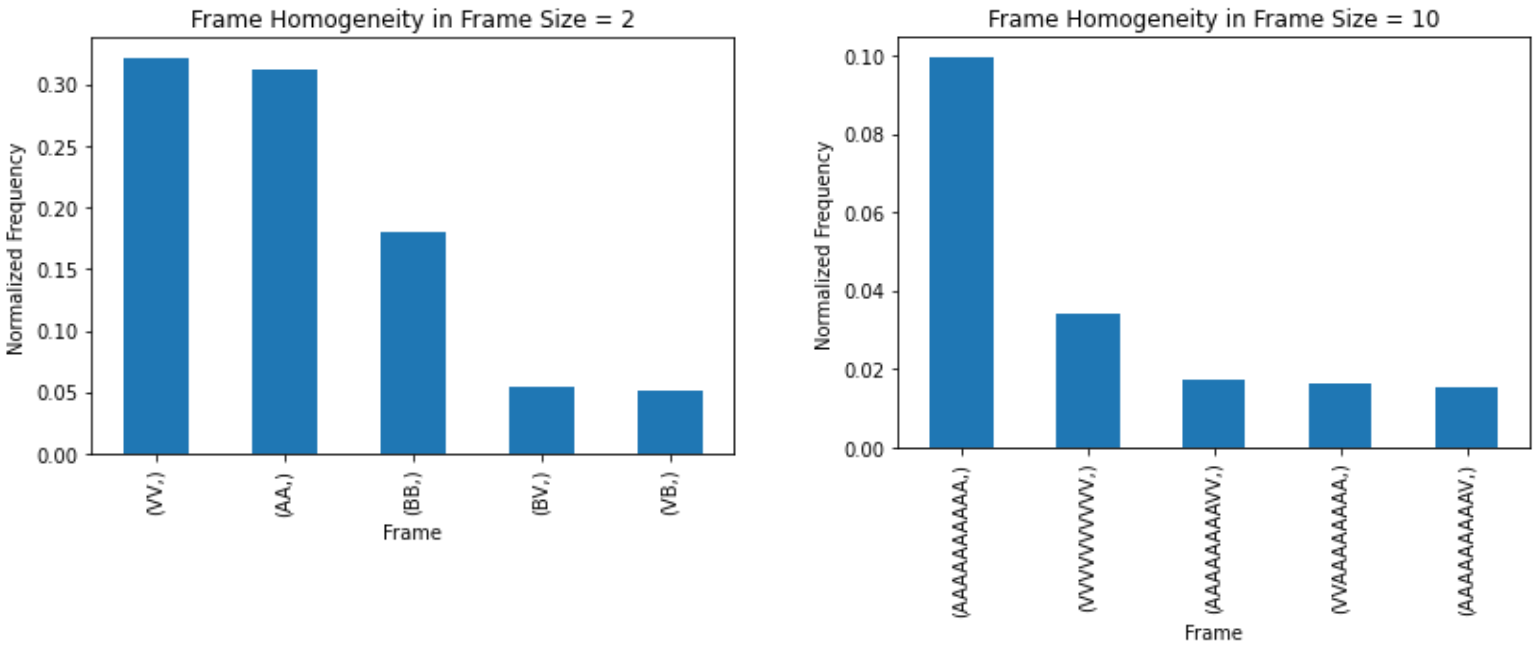

Figure 3: The homogeneity of frames for two of the tested frame lengths

Because the manual estimator estimates all of the amino acids in a frame as the dominant structure code, the accuracy will be inherently limited by frame homogeneity. These histograms show that the most common frames tend to be homogenous at the lower end of the frame length, but at higher frame lengths, the homogeneity drops drastically due to the increase in possible structure code combinations.

This is important because it shows that the manual estimator is functioning relatively well under the constraint of homogeneity. If the estimate only has the opportunity to be 100% correct on 10% of frames than anything above that ten percent is just partition frame matching. It does, however, illustrate the inherent limitation in using this frame approach. Cutting the sequence into frames and assigning the entire frame a secondary structure code makes it impossible to 100% accurately estimate the structure of any protein that has non-homogenous groups of secondary structures.

Overall, the manual secondary structure estimator performed very well. Error was low and the model was robust to changing frame sizes and sequence lengths. There is a major limit to practical use in that the method takes a very long time to compute a very small number of sequences compared to the ML model. Aside from this caveat, the manual method works better than the ML model we generated.

ML Secondary Structure Estimator

As most amino acid structures are 4 - 16 structures long this analysis ran frame sizes within those parameters. By encoding numerical signifiers for the amino acid code we were able to use a decision tree classifier. A classifier was used opposed to a regression model as our final answer would be a comparison of the categorical features of secondary structure, A, B, or V.

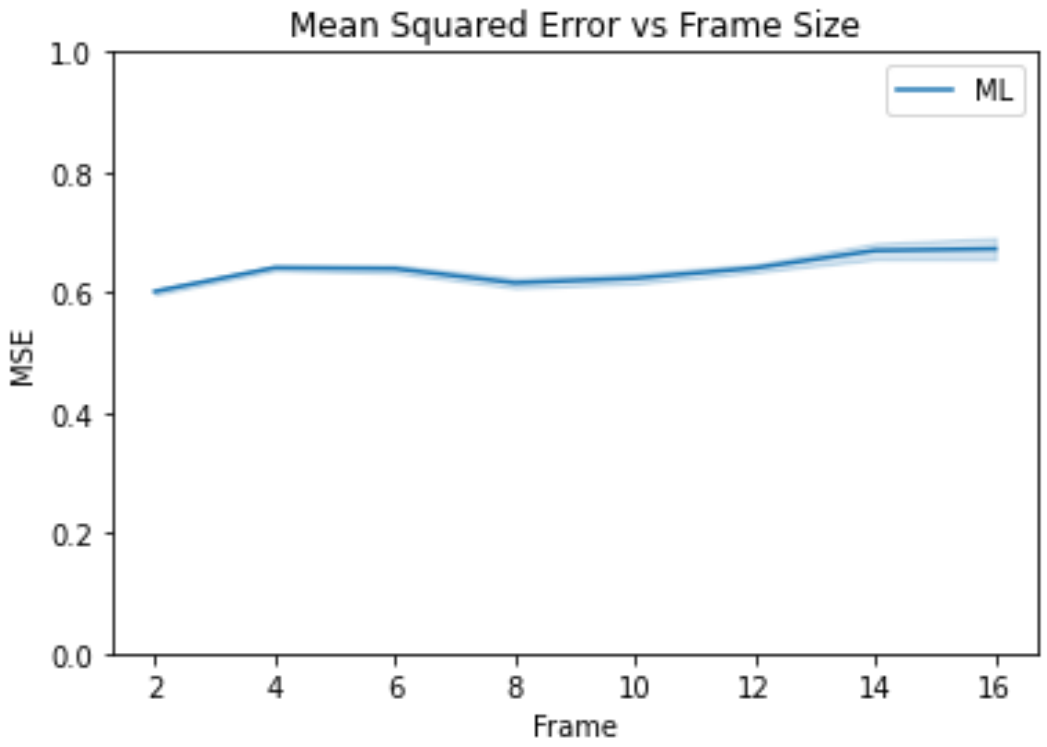

Figure 4: The effect of increasing frame size on mean squared error of the machine learning platform for secondary structure prediction with 95% confidence interval.

I predicted increasing the frame size would decrease the error of our method up to a point but would then increase error with increasing frame size. Increasing frame size provides more positional information for the sequence as there are more features. Which may lead to an initial decrease in error. However, a larger frame size encompasses more of the protein secondary structure, increasing the likelihood of complex A, V, B, hybrid domains that may confuse the machine learning algorithm. In addition, increasing the frame size means there are more possible combinations of A, B, and V. Therefore, there is less training data of possible secondary structure combinations at this size of frame. However, contrary to those hypotheses we found our analysis to stay constant at around 66%, otherwise known as random chance. There are three possible answers, A, B, or V. So the error of random chance would be 66%.

The overall accuracy of this machine learning platform is questionable. The root of this phenomenon may be that a decision classifier tree is inherently flawed with our current methods. Encoding a numerical operator for a categorical label can generate problematic relationships such as if Alanine = 1 and Asparagine = 3, the model will interpret Asparagine as being 3x Alanine. A conversion dictionary that coded amino acids not alphabetically, but by labels such as relative polarity

may have provided the platform corollary information while still allowing for the use of a decision tree classifier.

The benefit of this model however is that implementation of this method only takes 3 minutes for 17, 608 sequences, whereas the manual method takes 2 minutes for 100 sequences. At this rate we would expect the manual method to take just under 6 hours to analyze the larger data set.

Further finetuning of this machine learning platform by incorporating propensity information as a numeric classifier may increase the accuracy of this model while preserving the efficiency of machine learning. This tool may be helpful to spotlight areas of interest in secondary structure when doing a first pass, and then implementation of the manual classification method may generate more accurate results.

Challenge goals

The two primary challenge goals this project met are the use of multiple datasets, and machine learning. Combining multiple datasets provides the most robust picture of a particular sequence's characteristics, and gave us many features to use to train models. We used a training and testing dataset from Kaggle, and combined it with an amino acid propensity data set from a research paper. We also implemented a decision tree classifier model as a form of machine learning.

Our challenge goal was scaled back from including the pdb library as we could not devise a way to manually code a classifier for protein structure based on sequence. Because of this, there was no use for the PDB library.

Work Plan Evaluation

The work plan proved to be fairly accurate, with the exception of troubleshooting. There were several details involved in each step that required extensive troubleshooting to fit together. Treating the troubleshooting as a separate category, it took nearly 10 hours to fully complete.

Testing

The manual method was tested on a sequence dataset of 1, 5, 10, 100, 500, and 1000 to test scalability. No assert equals was used, but separate methods to assess the success of the manual and ML classifiers were used. These methods generated the data that is visualized in the results section of this report.

Collaboration

Karen and Isiac collaborated extensively in completing this project. Additionally, Hannah met with us and answered emailed questions that directed the final project. Due to time constraints, other outside resources were not extensively involved.