Classifying Protein Function from Structure via the k-Nearest Neighbors Algorithm

Anamika Kannan
Abstract:

Proteins can be classified in groups according to many factors, including structure, function, and sequence. To identify properties of a new protein, we can predict the class it belongs to and assign to the protein the properties that are characteristic of this class. Quickly determining these attributes would be extremely useful for scientists, as predicting the functional properties of proteins from structure could allow for high-throughput analysis of biological data. To determine whether we could extract useful functional information from the structures of proteins, we selected proteins from the Protein Data Bank and used the k-Nearest Neighbors algorithm to try to assign the proteins in the dataset to their corresponding class. After obtaining unexpectedly high accuracy for a 10-class classification problem, we used error analysis methods to discover that the dataset contained structurally homologous proteins, which artificially inflated the accuracy seen. We then used a sequence similarity culled dataset and tested the classification accuracy of kNN across 5 function classes, including an ‘unknown’ class, and got an accuracy of 36.6%. The performance for the unknown class was extremely poor compared to the performance of known function classes. This indicates that structural features do correlate with specific functions of proteins and validates using kNN as well as other machine learning approaches to classify protein function using distance matrices as structural representations.
Introduction:

Proteins are molecules necessary for structure, function, and regulation of the body’s cells, tissues, and organs. They are made up of a combination of the 20 different amino acids. Amino acids are made up of an amino group (H₃N⁺), an “R” group (side chain), and a carboxyl group (COO⁻). Proteins carry out many important tasks, such as providing cells with structure, repairing tissue, healing wounds, removing waste deposits, and transporting and storing essential nutrients. Critically, proteins also function as enzymes, catalyzing chemical reactions. Intermolecular forces between the amino acids—such as Van der Waal’s forces, hydrogen bonding, and solvation effects—drive the spontaneous folding of the protein. Because the amino acid residues drive the formation of the protein fold, perhaps the structure of the protein indirectly encodes information about the protein sequence itself. If this is the case, we can try to infer the function of the protein from a representation of its structure.

Experimentally, protein structure data is acquired through X-ray crystallography, NMR (nuclear magnetic resonance), and cryo-electron microscopy (cryo-EM). X-ray crystallography is a technique using x-ray diffraction to recover protein structure at the atomic level [1]. By measuring aspects of the diffraction such as angle and intensity, we can generate a multi-dimensional image of the electron density inside the crystal [1]. This data can then tell us the locations and positions of the atoms in the crystal, from which we can extrapolate important structural information such as chemical bonds [1]. NMR is a technique which uses radio frequency fields to probe the magnetic resonance of nuclei in a sample. NMR pulse sequences allow us to determine which atoms are near each other, which then allows us to recreate the protein structure. Cryo-EM is a form of transmission electron microscopy (TEM) [1]. In contrast to X-ray crystallography, however, it does not require crystallization of the specimen, but rather involves freezing the sample at cryogenic temperatures [1]. These methods have been used to develop a database of protein structures, known as the Protein Data Bank (PDB), and we can use this data as a basis for machine learning on protein structures.

These are the few staple methods used to recover protein structure; in contrast, for any given protein, determining its function requires various biological experiments—for example, binding assays, catalytic activity assays, imaging experiments, and more. Given a set of many
proteins, the time requirement to conduct these experiments on each one would be huge. Therefore, we wanted to experiment with the use of computational algorithms to classify protein function in a much faster way. Ultimately, the goal is to understand the correlation between structure and function of proteins.

We chose the k-Nearest Neighbors (kNN) algorithm to classify a curated dataset of protein structures. KNN is one of the simplest machine learning models because it does not require a training step, unlike linear models such as Support Vector Machines (SVMs) or nonlinear models such as neural networks. As a result, the algorithm is useful in establishing a baseline or lower bound for this classification task and is a good starting point before progressing to more powerful machine learning models. Because this is a somewhat simple algorithm that is being used for a complex problem, there is no scientific paper in the literature that describes using the kNN algorithm on protein distance matrices to determine function. However, there are researchers who have used the kNN algorithm to do secondary structure prediction of proteins [2,3].

Methodology:

Data:

We downloaded data from the Protein Data Bank (PDB) corresponding to 126,321 structures, as well as metadata—which includes unique IDs from the PDB, the class labels of the proteins, and the sequence length of the proteins [4]. One limitation of the kNN algorithm is that it needs all input data to be the same length. Therefore, we used the Structured Query Language (SQL) program to select the IDs of the proteins that all had a sequence length between 125 and 175 residues, as visualized below in Figure 1 in a length distribution histogram. This way, we could crop them all to be the same size without losing too much information (The algorithm we use in this paper requires inputs to be the same size). If we take proteins from range 400-700 amino acids in length, for example, we would have to crop all of them to be 400 amino acids in length, meaning that 300 amino acids would be cut from the proteins that are 700 amino acids long. This may lead to incorrect classification of the proteins. As shown in Figure 1, the peak of
the distribution is around 150 residues. We selected a range near the maximum of the distribution so that we could use a narrow window while still retaining a fairly large and diverse dataset.

![Approximate Protein Length Distribution](image)

**Figure 1: Protein length distribution for a subset of PDB.** The initial data mining process where the area in between the two bars indicate the protein data used from PDB.

Distance Matrix Representation:

While we can use the metadata to sort the proteins based on sequence length, the PDB data file has cartesian coordinates associated with every atom of the protein, from which we were able to extract the pairwise distances between alpha carbons (see image on right) on the protein backbone. From these distances, we generated a distance matrix representation, as shown below in Figure 2. The resolution of the structure, or the extent to which structural data collected by crystallography is accurate, varies up to at most three angstroms per coordinate—which allowed us to be confident that our pairwise distances were precise [5].
Figure 2: Distance matrix representation of a protein structure. This represents the pairwise distances between alpha carbons for a fluorescent protein (PAmCherry1), which has a beta barrel structure. The beta barrel is made up of parallel and antiparallel beta sheets, which give rise to the characteristic pattern of repeating boxes in the distance matrix. For example, the entry at (100,50) is the Euclidean distance between the 100th alpha carbon and 50th alpha carbon. This is equivalent to the entry at (50,100). Because the Euclidean metric is symmetric, the distance matrix representation is symmetric about the diagonal.

k-Nearest Neighbors Algorithm:

The kNN algorithm is a simple but powerful algorithm for machine learning. The goal of kNN is to classify new data points, given a labeled dataset. The input to the algorithm consists of vectors that each have a class label. The input dataset is split into a train dataset and a test dataset. The output of the algorithm is the class membership of the test dataset examples. For example, the input could be a set of images of objects or animals, and the task could be to classify what is represented in a given image. In our case, the input is the protein structure distance matrix, and the output is the known function of the protein.

To clarify the steps of the kNN algorithm, we can look at an example—applying kNN to natural image data, which is visualized below in Figure 3. I used starter code from the Stanford course, CS231n in order to load and visualize the natural image data [6], but I wrote the code for the algorithm myself. The classification is determined by assigning the test image to the class that is most common among its k nearest neighbors in the train dataset. For example, if k=1, the
test image is automatically assigned to the class of its single nearest neighbor in the train dataset. If k=5, however, the test image is assigned to the majority class most frequently occurring of its 5 nearest neighbors in the train dataset. In order to determine the distance between inputs, we require some kind of a distance metric, so we chose the Euclidean or L2 distance metric. This metric (below) is the distance between image 1 \((I_1)\) and image 2 \((I_2)\).

\[
d_2(I_1, I_2) = \sqrt{\sum_p (I_{1p} - I_{2p})^2}
\]

Specifically, this metric is the square root of the difference between all pixels of image 1 and their corresponding pixels on image 2. This value is squared so that even if the pixel on image 2 is greater than that on image 1, the metric will ultimately become a positive value. You can see that for identical images the distance will be zero.

The first step of the algorithm is to calculate the pairwise distance between images using the L2 distance formula. In Figure 4 (below), the y-axis represents the test data examples while the x-axis represents the train data examples. The indices of the minimum values along the rows of the matrix tell you the nearest neighbor train examples for each of the test examples. Finally, the algorithm returns the most common class label of these nearest neighbors.

![Figure 3: Cifar-10 natural image data.](image-url)
Cross validation is a method we use to find the best value of $k$, meaning the best number of neighbors that we will then take the majority of to accurately identify the class. In cross validation, we split the training dataset into five folds and run the kNN algorithm five times, each time removing one fold of the data and using it as test data. Figure 5 (below) shows the visualization of the cross-validation plot for each $k$ value and each of the five folds. Through this particular method of cross validation, we can ensure that each data point is in the test or validation set exactly once and is in the training set four times. Cross validation is extremely useful in machine learning as it allows for validation of the algorithm on many different subsets of the data available and characterization of the variance in performance across these subsets.

**Figure 5 (below): Cross-validation results.** Accuracy for each value of $k$ for each of the 5 data folds.
As an example, for the natural image data, we can look at the cross validation curve and find its peak, which corresponds to the best k value. Using this value, we can get the test data accuracy of the dataset. In this case, the peak was at k=10 and the accuracy was 28.2%.

Results:

As the input to the kNN algorithm, we used three-dimensional protein structures represented in a two-dimensional distance matrix. These matrices represent the distances between alpha carbons in Angstroms (1e-10 m). For our first experiment, we looked at classification across the ten classes given below in Table 1 (page 10).

We visualized a few examples of distance matrices per class, shown below in Figure 6a. Highlighted are distance matrices with a great deal of visible structural similarity, which indicated to us that there might be structural homologs in the dataset. This was confirmed by the results of the algorithm. The accuracy with k=1 was 76.0% across the ten class while the accuracy with k=5 was 68.8%. This was much higher than anticipated and we also saw from cross-validation that performance was the highest when k=1, indicating that there might be repeated structures or highly similar structures in the dataset. This is visualized in Figure 6b.

**Figure 6a: Distance matrix visualization.** 5 distance matrix samples for each of the 10 classes.
**Figure 6b: Cross-validation results for the first dataset.** Accuracy for each value of k for each of the 5 data folds. The accuracy is the highest for k=1.

<table>
<thead>
<tr>
<th>Protein class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oxidoreductase</td>
<td>An enzyme that balances the reduction of one compound with the oxidation of another through the transport of electrons (may utilize NADP/NAD+ as cofactors)</td>
</tr>
<tr>
<td>Transport protein</td>
<td>Refers to a class of proteins that transport important substances (for example, transmembrane proteins, which assist in the movement of substances into and out of the cell)</td>
</tr>
<tr>
<td>Signaling protein</td>
<td>A class including proteins (such as those involved in taste) which lead to signal transduction and send impulses to the brain based on the frequency of reception, through the opening and closing of ion channels</td>
</tr>
<tr>
<td>Immune system</td>
<td>Includes proteins such as antibodies and T cell receptors, which are known to be involved in the body’s immune response.</td>
</tr>
<tr>
<td>Viral protein</td>
<td>Proteins generated by a virus (as viruses takeover the host, the viral proteins serve as structural components which aid in the encoding of their own genes)</td>
</tr>
<tr>
<td>Sugar binding protein</td>
<td>Includes lectins, which identify and bind to specific carbohydrates located on cell surfaces and help with communication from cell to cell</td>
</tr>
<tr>
<td>Isomerase</td>
<td>Refers to an enzyme that catalyzes the hydrogen transfer between neighboring carbon atoms (one is oxidized while the other is reduced) —→ turns the specific compound into an isomer</td>
</tr>
<tr>
<td>Transcription factors</td>
<td>Proteins involved in the process of transcribing DNA into RNA (have the ability to bind to certain sequences and can control transcription of the related gene)</td>
</tr>
<tr>
<td>Hydrolase</td>
<td>An enzyme which catalyzes a bond hydrolysis via water</td>
</tr>
<tr>
<td>Transferase</td>
<td>An enzyme that transfers a chemical group of one molecule to an acceptor molecule</td>
</tr>
<tr>
<td>Structural</td>
<td>Proteins like collagen that are parts of structural components in cells</td>
</tr>
<tr>
<td>Unknown</td>
<td>Proteins with unknown function</td>
</tr>
</tbody>
</table>

Table 1: List of classes and their descriptions. The original dataset contains the first ten classes (blue and purple), the culled dataset contains the same ten classes, and the balanced dataset contains the last five classes (purple and red). 1, 2, and 3 refer to the original, culled, and balanced datasets respectively and note the number of proteins in each class.
We chose to narrow our focus to a culled dataset with a sequence percentage identity cutoff of 25% in order to eliminate repeated structures in the dataset. Unfortunately, the number of proteins per class in the culled dataset was not balanced, as shown in Table 1. The algorithm showed a lower, but more reasonable accuracy for k=1 (44.0%) and for k=5 (37.3%). Looking at the cross-validation curve in Figure 7, we can see that there is an improvement at k values greater than 1, with the best k across the validation set being k=5; this is what we expect to happen as more information is added with a larger k. Unfortunately, we see that the performance varies wildly across classes as shown in Figure 8 (below). Random guessing for a 10-class classification would give about a 10% accuracy assuming the classes are balanced. Because of the imbalance across classes in the data, as seen in Table 1, some of the underrepresented classes have a very low accuracy, sometimes below 10%. This imbalance is visualized in the “accuracy vs. k” plot in Figure 8, where we can see that there is a wide variance in accuracy between the 10 classes.

![Figure 7: Cross-validation results for the culled dataset.](image-url)
We then decided to select five classes which were more balanced and better represented in the culled dataset; these are indicated in Table 1 and some distance matrices are shown in Figure 9. Unlike previous experiments, we included an unknown class comprised of proteins from PDB whose structures are known but whose functions are not known. The various datasets and respective classes for each trial are shown in Table 1. For k=1, the accuracy was 32.3% and for k=5, the accuracy was 36.6%. As seen in the cross-validation results (Figure 10a on page 13), the best k value is not k=1. The curve is non-monotonically decreasing, similar to the second experiment across 10 classes.
As shown below in Figure 10b (page 13), we looked at the accuracies across the known classes and saw that the performance was more consistent after classes balancing. Most importantly, the performance for the unknown class is extremely low and is distinct from the known function classes. This indicates to us that structural features do correlate with specific functions of proteins and validates using kNN as well as other machine learning approaches to classify protein function using distance matrices as structural representations. The accuracy results from our experiments are summarized in Table 2 (next page).
Discussion / Next steps:

The accuracy of the kNN algorithm in all of our trials ranked higher than the random guess percentage, which was 10% for the 10-class datasets and 20% for the 5-class dataset. In our first dataset, the error in the algorithm was due to the structural homologs that existed in the dataset. Because of this, k=1 returned the highest accuracy—76.0%. This meant that for many examples, the algorithm only needed to look at one nearest neighbor to determine the class label, indicating that for these examples, very similar structures were in the training dataset.
Ultimately, removing the structural homologs and ensuring class balancing gave us more informative results.

The kNN algorithm has certain properties which are both strengths and limitations. For example the underlying assumption is simplistic, so it is easy to interpret the results. However, this also means it is not as powerful an algorithm, unlike neural networks, which can learn a nonlinear boundary between classes. In addition, the kNN algorithm is trivial to train because it has no explicit “training step”, whereas Support Vector Machines (SVMs) and neural networks have to learn a linear or nonlinear boundary respectively, separating training data points by their class. However, the kNN algorithm is very slow at test time, where for each test example, the distances to every train example have to be calculated. Finally, one of the major limitations of the kNN algorithm is that we need all of the input data to be the same size, which is why we came up with a scheme to crop the protein structural data. Perhaps, one way to avoid this is to come up with a metric to compare proteins of different lengths.

The distance matrix representation of the proteins has some advantages. First, it is rotationally invariant; even if the atomic Cartesian coordinates are rotated about the origin, all the pairwise distances stay the same and the distance matrix does not change. In addition, this representation encodes specific elements of secondary structure such as alpha helices, beta sheets, loop regions, and domain interactions. One major weakness of the distance matrix representation is that it does not encode the protein sequence. Likewise, there is no information about cofactors, which might greatly help classify the function of these proteins in some cases. For example, cofactors NADPH or FAD, are involved in redox chemistry and therefore, the proteins which bind these cofactors likely have the function of assisting with electron transport. Knowing the cofactors would immediately help classify these proteins.

**Conclusion**

While performing wet lab experiments to identify protein function is time consuming and resource-intensive, using computational algorithms such as the k-Nearest Neighbors algorithm to determine protein function can be very efficient. In our project, we used the kNN algorithm to discern protein function by analyzing structure. The proteins were represented by a
two-dimensional distance matrix, which encodes important structural information, including alpha helices and beta sheets. From our original dataset of ten classes, we observed that the abnormally high accuracy (76% for k=1) was the result of structural homologs within the dataset. Because of this, we ran the algorithm on a culled dataset with the same ten classes. We then saw wide variance in accuracies for each class, hinting at the fact that the ten classes were imbalanced—for larger classes, the accuracy tended to be much higher, reaching 77.8% for the largest class when k=5, while for the smallest classes, the accuracy was 0% when k=5. This discrepancy led us to our final dataset—five classes with a relatively better balanced number of proteins per class. For this dataset, we got an average accuracy of 36.6% across the five classes. We saw that the accuracies of the four known classes were much higher than that of the unknown class. This tells us that the kNN algorithm is able to learn a correlation between structure and function of proteins.

There are many possible next steps for this project. The first step would be to come up with a new metric that allows us to align and score proteins of different lengths. This would allow us to run this classification method across all known structures. Another future step is to use SVMs and neural networks for the same classification problem to see if we can improve the accuracy above the baseline established in this paper. Other classification tasks to try include predicting secondary structure at every residue as well as predicting the sequence of the protein from the distance matrix. Ultimately, we hope to be able to design new proteins with particular functions in mind, using these computational algorithms to validate and guide the design of these de novo engineered proteins.
Bibliography


