Detection of Salient Mutations within Protein Sequences Through One-Class Support Vector Machines

Joshua Silva
Northeastern University, Boston, USA

Abstract

In recent years, the technology required for DNA sequencing has both improved and seen a reduction in cost to the point that, today, sequencing has enabled the recording of genomes for an ever-growing number of species. While there is a vast amount of information available it is difficult to sufficiently test in labs all the known proteins which have been sequenced. One area of interest with proteins, is the ability to understand which of their composing amino acids are important to their function. By building profiles of known protein sequences, harmful mutations to proteins can be predicted. This would benefit detection of novel genetic mutations and treatments for such genetic diseases. To enable this, support vector machines can be designed and trained with known genetic sequences. Through these support vector machines, changes to protein sequence can be predicted to be detrimental to the function of the protein.

Introduction

Proteins, which perform a variety of complex functions in all organisms, are composed of chains of amino acids. There are twenty commonly occurring amino acids across all species. These amino acids differ from each other by a side chain of their structure. Some have small side chains, others are long and flexible. Amino acid side chains can be attracted to or repelled by water. From their set of properties, the chain of amino acids forms a complex structure, which folds onto itself to yield a stable conformation. From this structure, proteins gain their function allowing for complex biological pathways to form.

Despite the insights which protein structure provides, only a minority of proteins have known structures, due to the work and cost required to obtain a protein structure. However, protein structure can be calculated or understood from the sequence of the protein, and while there is not a fully generalizable and complete method to predict structure, insights can be made using the knowledge of the protein sequence.

One area of interest is understanding which mutations to proteins will cause them to stop functioning. Proteins experience mutations to their sequence where one amino acid is exchanged with another. This occurs in nature; it is seen both within diseases caused by sequence mutations, such as sickle cell anemia, but more generally across different species. Through genomic sequencing of many organisms, libraries of protein sequences have been recorded. These libraries consist of protein families, or proteins which are similar across species both in sequence and function. By looking at the sequences of protein families, for each position within the protein sequence mutations can be categorized as detrimental to function or tolerable, and sometime beneficial. The impact of the mutation is attempted to be classified.

Two issues arise with using the protein sequence libraries. One, these protein sequences are all composed of mutations which do not render the protein nonfunctional, so one-class classification or outlier detection needs to be done. Secondly, the sequence space of known genetic variations to protein structure is significantly smaller than the set of possible sequences. While there may be 10,000s of known sequences, protein sequences are often made of hundreds amino acids, leading to a sequence space of 20^100. While the entire sequence space encloses sequences which would be considered in the same protein family as a protein of interest, the discrepancy in the two sizes leads to overfitting of models as the data closely resembles each other. To address these two issues, a one class support vector machine can be used for single class classification. This can be trained using the properties of the amino acids in the protein sequence rather than the amino acids as features, to allow for a less rigid definition of what an acceptable amino acid is at a given position of the sequence.

Background

Generally, support vector machines are a class of methods used to identify data into a set of possible classes. In the simple case there are two classes for the data to be classified as. For the model to be trained, the support vector machine takes in the input, independent variables or features and the classes of the training data. It then attempts to fit a line or boundary to separate the data into two groups based on the classes of the training data. The support vector machine tries to form a boundary which separates the sets as much as possible. The boundary is constructed to maximize the distance of the closest point of each class to the boundary. The data can be intuitively linearly separated, i.e. a line can separate the two groups of classes, but it is possible for other types of boundaries to be created. For example, if the data was two dimensional (*X* and *Y*), and one class was clustered in a group and the other encircled it, the two groups could be distinct, but a line would not be able to separate them. To solve this, the data can be transformed into another dimension, $Z = X^2+Y^2$.[1] Then a line could be used to separate the classes assuming Z were plotted against X. This is generalized using different kernel functions to create different boundaries in the data. These functions are constructed from the dot-product of vectors in the feature space. Kernel functions bypass the need to project or plot the data with the new dimension.[1]

For one-class support vector machines, there are not classes within the data to separate, where the boundary should start and end is not clear. To solve this, the data is mapped over the space consisting of all features which compose it. This results in the data being clustered around the origin in this feature space. The support vector machine maximizes the distance of the data in the feature space to the origin.[2] The boundary can then be constructed around the data. This boundary creates two groups, the data centered around the origin, and the space outside of the boundary, which classifies for outliers or unobserved classes from the training data. The boundary is minimized to the data using the following equation:

$$\min_{w,\varepsilon,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\text{vn}}\sum_{i=1}^{n} \varepsilon_i - \rho$$

Where *w* and ρ form the boundary constructed by the support vector machine, ε sets a margin to create noise in the boundary to limit overfitting to the data, and v which sets an upper bound of training data which can be considered outliers and a lower bound on the number of data required by the support vector machine.[2]

Related work

The goal of classifying mutations from protein sequences has existed for a while and various solutions to the problem have been proposed. An earlier tool, known as SIFT[3], determined potentially harmful mutations by calculating the probability of each amino acid at each position of the sequence. Based on the variability of amino acids present at that position within the sequence a threshold is set.[3] The probability of amino acids at that position are compared to the threshold value to determine if they are deleterious or not. This method did not cover the machine learning or search algorithms which this paper attempts to explore so it was not considered.

More recent models have gone beyond considering the amino acid positions in the sequence as independent elements. They have looked at pairwise differences in the amino acid sequence to determine the potential affect a mutation may have. This has been achieved through usage of a Potts or pairwise undirected graphical model.[4,5] Pairwise models, begin to capture more of the structure of the protein, as amino acids will interact across the sequence depending on their position in three-dimensional space. Models have recently gone beyond pairwise interaction, to consider the entire latent space of the sequence. This can capture higher order relationships in the protein sequences than an independent or pairwise model. This has been achieved through use of a variational autoencoder.[6]

While, these methods are capable of considering additional interactions of the amino acids within the protein sequence, they achieve this through greater computational complexity and hardware requirements. Computational resources were limited for the models created and explored in this paper, so a simpler method was chosen. For the amount of data available for each model, in the ten thousands of sequences, and the computational resources available one-class support vector machines were a good fit. Additionally, these methods used the used the amino acid at a given position of the sequences the sole feature in the models, the one class support vector machines described in this paper were trained using the chemical properties of the amino acids. There was not a method which used chemical properties as features to solve the mutation classification problem, so it was unclear if it would work and what benefit different models would provide.

Project Description

Sequence Weighting

Due to the nature of biological sequences, many amino acids are conserved across the different species. This results in a predominant amino acid present at each position in the protein sequence. This obscures the acceptable variations which should be captured by the boundary created by the support vector machine. To achieve this, sequences are weighed to promote the more unique sequences using the following equation:

$$\pi^s = \left(\sum_t Int(D_H(s,t) \; < \; \theta)\right)^{-1}$$

Where π^s is the sequence weight of sequence *s*, *t is* all sequences, D_H is the relative Hamming distance or the relative number of differences between sequences *s* and *t* with respect to the length of *s*, and θ is the threshold for uniqueness. θ was set to 0.2 as suggested in the EVmutation paper.[4]

Training/Test Set Construction

From an overall set of sequence data, sequences were mapped over amino acid features to create an initial data set. This was split randomly, with half of the entries used to train the model and the other half used to test the model. This is needed as the sequence of amino acids were ordered providing different distributions of proteins. The distributions motivating this randomization are present in figures 1 and 2.

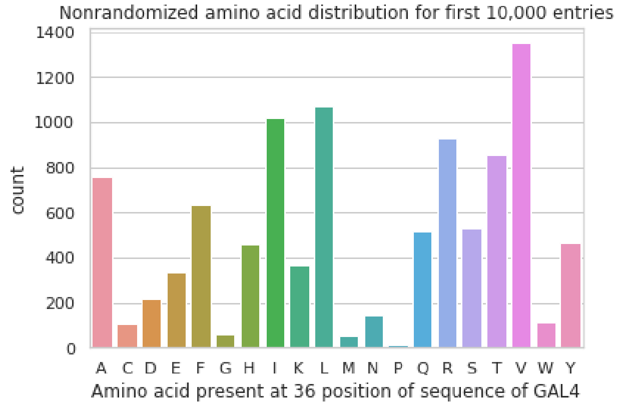

Figure 1: Distribution of Amino Acids present at the 36th amino acid position in the GAL4 sequence for the first 10,000 entries of the GAL4 dataset

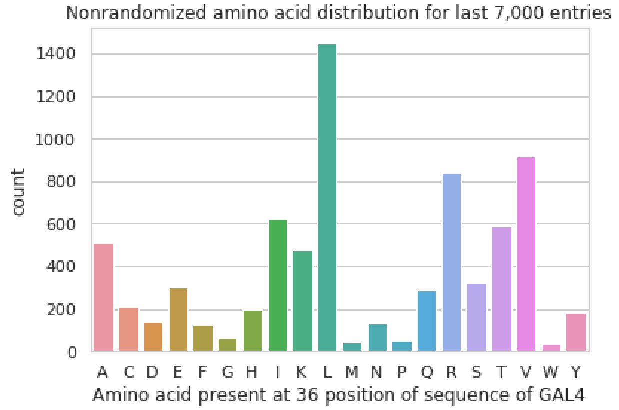

Figure 2: Distribution of Amino Acids present at the 36th amino acid position in the GAL4 sequence for the last 7,000 entries of the GAL4 dataset

Model Training/Testing

Models were trained using their respective training set along with their respective sequence weight. The model boundary was created using a radial basis function:

$$K(x, x') = \exp(\gamma \|x \; - \; x'\|^2)$$

Where x′ and x′ are samples within the training set. γ was set to the inverse of the product of the number of features and the standard deviation of the training data for the final models which were trained. The v term of the one-class support vector machine algorithm shown in the background section was set to 0.5. The support vector machine was trained using a shrinking heuristic which optimizes over a subset of relevant features as determined by the algorithm to improve performance. The excluded values in the subset have reached their bounds, so it is determined safe to remove them.

Experiments

Implementation

All support vector machines were created using the sklearn library for python using the modules for creation of one-class support vector machines.[7]

Data

Sequence data used for training and evaluation of models was obtained from the supplemental material of the EVmutation paper, accessed from https://marks.hms.harvard.edu/evmutation/.[4] Sequence data was present in the a2m format for multiple sequence alignment. Sequence alignment is the process where protein sequences are lined up, so the amino acids most closely match each other across different sequences. This is needed as in addition to mutations causing substitutions of amino acids, they can also be deleted from or inserted into the sequence. To avoid the computationally expensive alignment and searching of protein sequences within a database, this data was chosen both due to the sequences curated by protein family and also their proven usage in mutation detection.

As these sequences were present as purely the amino acid at each position in the sequences, the chemical properties for use as features needed to be obtained. The amino acid properties were obtained from the Chembl API for each amino acid of interest.[8] The sequences were then represented in terms of properties of interest obtained from Chembl. This data was modified to represent the side chains, the variable portion, of the amino acid. To comply with the a2m sequence format, feature sets were generated for gap representations and for notations representing a subset of multiple amino acids. The features selected for use in the models were: number of aromatic rings, hydrogen bond donors, hydrogen bond acceptors, molecular weight, number of heavy atoms, polar surface area, number of rotatable bonds, logD, and logP.

One-Class Support Vector Machines

Two types of support vector machines were trained. The first considers all the properties of the amino acids in a sequence as a single record, it trains on the sequence. This support vector machine classified the protein as function or nonfunctional based on the boundary created. For the GAL4 sequence, the model was trained with 9 properties * 75 amino acids or 675 features per sample. This was trained on roughly 9,000 sequences and tested on 9,000 as well. One class support vector machines are robust to a large number of features relative to the size of the training data. Initially without sequence waiting and using a γ value of the inverse of the number of features, a recall of only 0.0076. With the added training parameters, recall was improved to 0.12. While not an excellent recall, it is a significant increase to the starting recall observed.

The later type of models were trained only on amino acids for a specific position within the sequence. Sequence weighting is applied in both models using the sequence weight for the entire sequence. For these models, each sample in the training data had nine features, the properties of that specific amino acid, associated with it. These models were much faster to train and predict. Using the same 9,000/9,000 split they completed prediction in under 3.5 seconds after training began. The only model tested using the original parameters of the sequence-wide support vector machine was on only the 36th amino acid position. Without randomizing the training and test sets the recall was 0.25, with randomizing it decreased to 0.23 despite more similar distributions of amino acids at that position for the training and testing sets. Adding the sequence weights to the training step to promote the low frequency amino acids increased the recall to 0.27. The largest change to the recall came with the update of the γ value to include the standard deviation of the training set yielding a recall of 0.49. With these parameters, a more acceptable recall was reached. This model was then trained for all amino acid positions in the GAL4 sequence. This did not result in consistent recall across positions, as depicted in figure 3 below.

The training and test sets were consistent for each of these models trained, so it was not the case that one position contained an increased amount of unique sequences in the test set. The differences in recall across positions in the sequence are likely due to different degrees of amino acid variation at each of the positions, from more or less conserved regions of the sequence. Additionally, the treatment of the gap character, assigning 0 for all properties, may be handled improperly resulting in positions with more gap characters having worse performance.

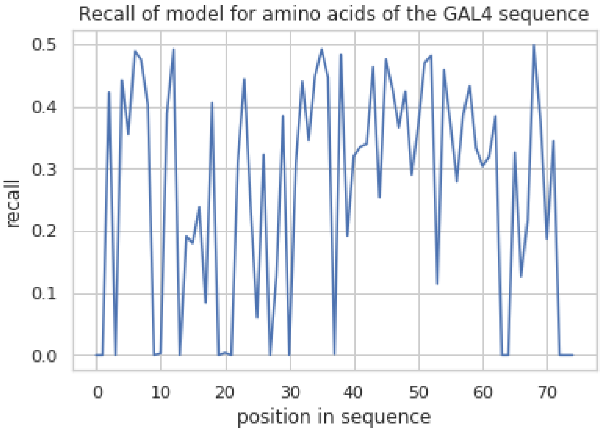

Figure 3: The recall for amino acid support vector machines trained on amino acid properties at each position of the GAL4 sequence.

Conclusion

The results above for one class support vector machines for detection of salient mutations within protein function are mixed, but in its current state mostly unpromising. For some of the positions in the protein sequence, recall from 0.3 to 0.5 was achieved. This indicates a possibility that the chemical properties of proteins can play a role in mutation classification. However, the recall could be much higher, as it is for more developed methods which distinguish only on the amino acid present. Also, the recall is below 0.1 for a large number of positions within the GAL4 protein sequence tested.

There are many directions to take this project in its current form. The chemical properties used as features may need to have additional properties considered, nonsignificant features removed, or normalized for consistent features. How the multiple amino acid representation and the gap character of the a2m sequence format are assigned features may also need to be changed to limit harmful bias they may cause. A one-class support vector machine may not be the correct method to solve this problem or the training parameters may need to have additional tuning as this created a significant increase for the models during their development. Two types of models were trained, one considering the whole protein sequence classifying protein functionality and the other on the likelihood of a mutation to reduce the functionality of a protein. It may be possible to use the later models together in some form of a neural network to produce a method which can predict overall protein functionality, with the added increases to recall and reduction to runtime which these models possess. Additionally, once a more optimized model is reached, it would be beneficial to test how generalizable the model is on other protein sequences. It would also be beneficial to test the optimized models using mutated sequences which were designed and tested to be nonfictional, to test if the model can accurately classify true negatives.

References

1. Stecanella, B. “An introduction to support vector machines (SVM).” MonkeyLearn. Web. June 22, 2017.
2. Vlasveld, R. “Introduction to one-class support vector machines.” Roemer’s Blog. Web. Jul 12, 2013.
3. Kumar, P. et al. “Predicting the effects of codding non-synonymous variants on protein function using the SIFT algorithm.” *Nature Protocols*, 4, 1073-1081, 2009.
4. Hopf, TA et al. "Mutation effects predicted from sequence co-variation." *Nature Biotechnology* 35, 7, Feb 2017, 128-135.
5. Mann, JK et al. "The Fitness Landscape of HIV-1 Gag: Advanced Modeling Approaches and Validation of Model Predictions by In Vitro Testing" *PLOS Comp. Biol.* 10, 8, Aug 2014.
6. Riesselman, AJ et al. "Deep generative models of genetic variation capture mutation effects." *Nature Methods* **15**, 816-822, 2018.
7. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
8. Gaulton A et al. “The ChEMBL database in 2017.” *Nucleic Acids Res.*, 45(D1) D945-D954, 2017.