K-means-based Feature Learning for Protein Sequence Classification

Paul Melman and Usman W. Roshan
Department of Computer Science, NJIT
Newark, NJ, 07102, USA
pm462@njit.edu, usman.w.roshan@njit.edu

Abstract

Protein sequence classification has been a major challenge in bioinformatics and related fields for some time and remains so today. Due to the complexity and volume of protein data, algorithmic techniques such as sequence alignment are often unsuitable due to time and memory constraints. Heuristic methods based on machine learning are the dominant technique for classifying large sets of protein data. In recent years, unsupervised deep learning techniques have garnered significant attention in various domains of classification tasks, but especially for image data. In this study, we adapt a k-means-based deep learning approach that was originally developed for image classification to classify protein sequence data. We use this unsupervised learning method to preprocess the data and create new feature vectors to be classified by a traditional supervised learning algorithm such as SVM. We find the performance of this technique to be superior to that of the spectrum kernel and empirical kernel map, and comparable to that of slower distance matrix-based approaches.

keywords: Protein classification, Unsupervised learning, K-means

1 Introduction

Identifying protein functionality is one of the principle challenges of modern biological sciences. While there do exist precise alignment techniques such as Smith-Waterman [15], due to their highly complex structures and behaviors, modeling large sets of proteins using deterministic methods is impractical, if not impossible, with currently available technology. Therefore, we must rely on heuristic techniques for analyzing proteins. One way to elucidate the behavior of a protein is to compare it to proteins with known properties, via protein classification [14].

Over the course of evolutionary history, genes and proteins with similar functionality diverge due to the accumulation of mutations. As a result, analogous proteins may become difficult to recognize. Through the use of machine learning techniques, it is possible to find relationships between protein sequences that would otherwise be obscured.

2 Related Work

K-means is a clustering algorithm that is used to partition points into k clusters based on the nearest cluster mean, or centroid [10]. The use of K-means clustering in image classification is based upon the principle of representation learning. An image is broken up into fragments by a sliding window, and these fragments, along with fragments of all the other images in the dataset, are then clustered by similarity. The centroids of the clusters represent features learned from the images, such as corners or diagonal lines. A new feature vector representation of the original image can then be created based on the presence or absence of the various features [3] [4]. This technique has been successfully employed in image classification tasks such as distinguishing between bacterial colonies of different species or identifying weeds [6] [17].

3 Methodology

3.1 Data & Materials

The data we use in this study originate from the SCOP, CATH, COG, and 3PGK protein datasets [2] [11] [18] [13]. The datasets and classification tasks were obtained from the Protein Classification Benchmark collection [16]. There are a total of 3242 classification tasks across all the datasets. For the SVM classifier we use the Scikit-learn Python library[12].

The 3PGK dataset, which consists of sequences of 3-phosphoglycerate kinase from various species, has 10 tasks that consist of classifying sequences into kingdoms based on phyla.

The SCOP dataset has three standard classification task categories: Classification of sequences into superfamilies based on families (246 tasks), classification

Table 1: Datasets

Dataset	# seqs	average seq len	# frags (len 14)	tasks	task types
3PGK	131	411	52137	10	1
CATH95	11373	150	1562581	1414	8
SCOP95	11944	173	1916986	1629	6
COG	17973	373	6467745	189	2

into folds based on superfamilies (191 tasks), and classification into structural classes based on folds (377 tasks). There are also three 5-fold cross validation task sets. The first consists of 98 superfamilies with five random splits each of training and test data where the positive examples come from one superfamily and the negative examples are taken from all other superfamilies for 490 total tasks. The same 5-fold split technique was used for 58 folds for 290 total tasks, and for 7 structural classes for 35 total tasks.

The CATH dataset has four standard classification task categories: Classification into homology groups based on similarity groups, with 165 tasks; classification into topology groups based on homology groups (199 tasks), classification into architecture groups based on topology groups (297 tasks), and classification into structural classes based on architecture groups (33 tasks). There are also four 5-fold cross validation task sets: By homology (375 tasks), by topology (235 tasks), by architecture (95 tasks), and by structural class (15 tasks).

The COG dataset has two types of classification task. In the first task category, the positive training sets consist of prokaryote protein sequences representing a particular biological function (COG), and the positive test sets consist of eukaryote protein sequences representing the same COG. The negative training set consists of sequences representing other COGs. There are a total of 117 tasks of this type. The second category involves separating proteins belonging to the kingdom Archaea from proteins belonging to any other kingdom. There are 72 of these tasks.

These datasets also came with published benchmarks that were computed by creating all against all BLAST and Smith-Waterman distance matrices and an SVM classifier.

3.2 Empirical and Spectrum Kernels

The first baseline condition we use for this study is the empirical kernel map. For this we use 3364 reference protein sequences from the seed pairwise alignments in PREFAB 4.0 [7]. The feature vector for each protein in the dataset is created by aligning it to each reference protein using BLAST [1]. Each dimension of the final vector is the BLAST score of the alignment to a different reference protein. Therefore, in the full reference condition, the feature vector for each protein has 3364 dimensions.

The second baseline we use is the spectrum kernel, which creates a feature vector by counting the number of occurrences of every possible amino acid triplet in each sequence [9]. Both the spectrum kernel and empirical kernel conditions used an SVM classifier [5].

3.3 String-based K-means Feature Learning

For this condition, every protein sequence in the dataset is split into fragments using a sliding window (Figure 1). The fragments are then clustered using string-based version of the K-means algorithm. To compute the centroid of each cluster, we find the mode of each character position across all the fragments in that cluster (Figure 2).

To compute the distance from a fragment to a centroid we examine two different measures. First, we use Hamming distance, where we compare the fragment and centroid at each character position and count the number of mismatches. A larger count represent more dissimilar strings and therefore a greater distance. The second distance measure we examine is based on the BLOSUM62 matrix, which is derived from empirical observations of amino acid substitution probabilities [8]. The distance is represented by the negative of the BLOSUM62 alignment score of the two strings. The negative is used so that this algorithm optimizes for the minimization of the distances between points (fragments) and their nearest centroids, just as the traditional K-means algorithm does.

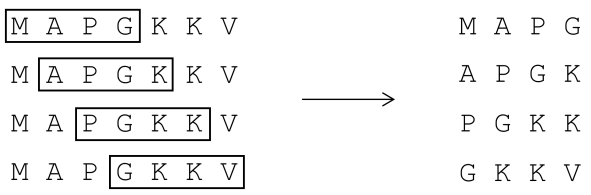

Figure 1: An example of the fragmentation process with fragment length 4 and a stride of 1.

To create a feature vector from the clusters, we use a method that is a cross between the triangle encoding and hard encoding schemes employed by Coates, Ng, and Lee [4], as described in Equation (1). For each fragment, we create a vector with k dimensions, where k is the number of clusters. For the feature f that corresponds to the index of the nearest centroid, the value is set to the mean of the distance to that centroid

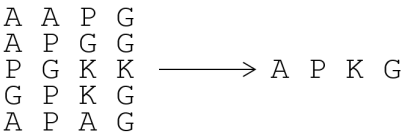

Figure 2: The centroid sequence is created by taking the mode character at each position.

Algorithm 1: String K-means Pseudocode

choose k random fragments as starting centroids
while $i = 0; i < max\_iter; i++$ **do**
  for *Each fragment* **do**
    Find distance to each centroid
    Assign to closest centroid
  for *Each cluster* **do**
    Calculate new centroid
  if *No change in centroids* **then**
    Break

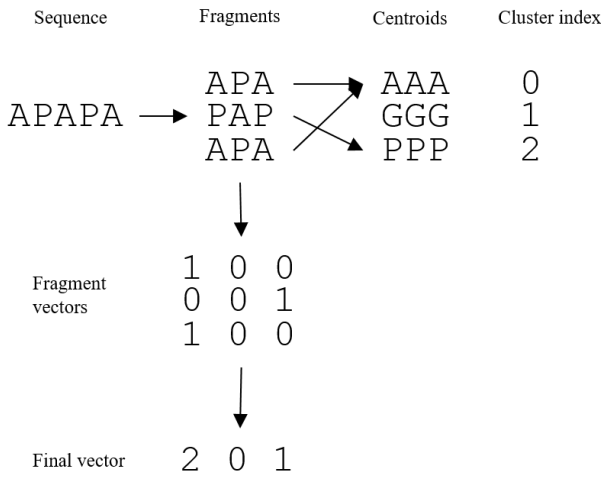

Figure 3: An illustration of how the feature vectors are created. Hamming distance and hard encoding are used in this example.

plus the mean of the distance to all centroids; for all other features, the value is set to zero to create a sparse vector with k dimensions. The vectors for all the fragments of a protein sequence are then sum-pooled to create the final feature vector for that sequence (Figure 3).

$$f_k(x) = \begin{cases} \mu(z) + z_k \text{ if } k = \arg\min_j \|c^{(j)} - x\|_2^2 \\ 0 \quad \text{otherwise} \end{cases} \quad (1)$$

where $z_k = \|x - c^{(k)}\|_2$.

We then train a linear support vector machine classifier on the feature vectors of the training dataset and evaluate it on the test dataset. This is done for each task in each task category based on the cast matrices obtained from the benchmark database.

4 Results

4.1 Comparison of Parameters

We ran the deep K-means algorithm with 2000, 4000, 8000, and 16,000 clusters on the CATH dataset (Figure 4) and with 2000 and 8000 clusters on COG (Figure 11). The results show a trend of improvement as the number of features increases. We also found that fragment length had little impact (Figure 5). These effects mirror those of Coates, Lee, and Ng [4]. However, unlike in Coates and Ng's later analysis of the K-means method for image classification, we did not experience problems with imbalanced or empty clusters [3]. We found that fragments became well distributed across clusters without the need for any additional processing (Figure 12). Additionally, we found that BLOSUM distance was superior to Hamming distance (Figure 7). Clustering made up most of our run time and, as expected, run time is longer for larger datasets (Table 2).

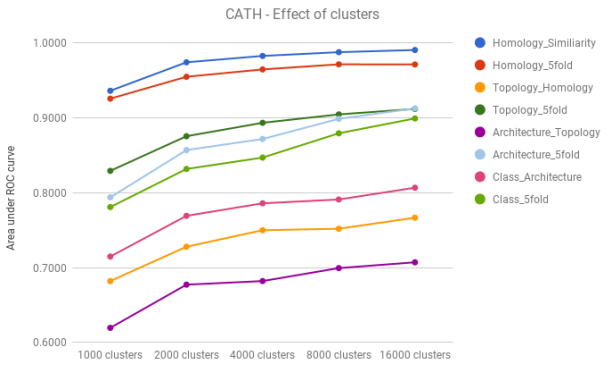

Figure 4: Effect of number of features (centroids) on CATH data.

4.2 Comparison to other Methods

We found that our K-means feature learning method outperformed the empirical and spectrum kernels on nearly every category of tasks. With 16,000 clusters, the K-means approach outperformed the empirical kernel map and the spectrum kernel on every task in CATH, and outperformed the all against all BLAST

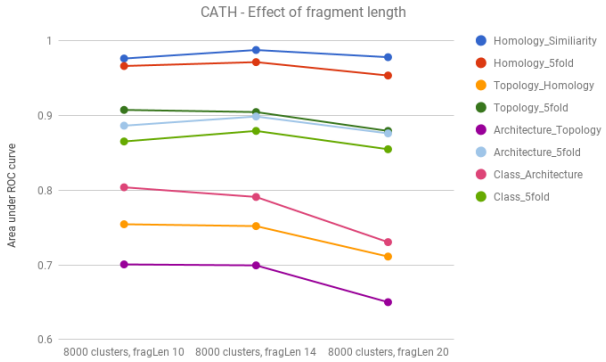

Figure 5: Effect of number of fragment length on CATH data.

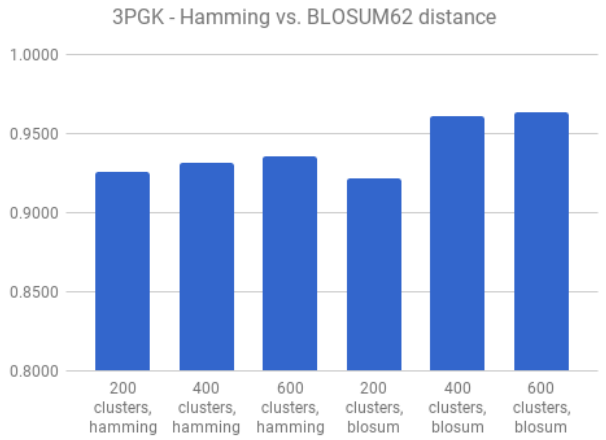

Figure 6: Effect of Hamming distance vs. BLOSUM distance on 3PGK.

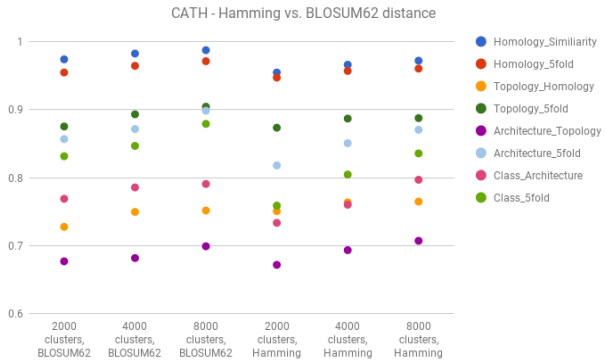

Figure 7: Effect of Hamming distance vs. BLOSUM distance on CATH.

Table 2: Runtimes for clustering on Intel Xeon E5-2630-v4 with 20 cores.

Dataset	time (minutes)
3PGK (600 clusters)	4
CATH (16k clusters)	799
SCOP (16k clusters)	1068
COG (8k clusters)	4493

Table 3: Average areas under the ROC curve for various conditions. BLAST and SW are all vs. all BLAST and Smith-Waterman distance matrices classified with SVM

Dataset	K-means	Empirical	Spectrum	BLAST	SW
3PGK	0.964	0.906	0.887	0.919	0.923
CATH	0.870	0.819	0.847	0.860	0.924
SCOP	0.842	0.810	-	0.841	0.896
COG	0.949	0.910	0.944	0.923	0.931

matrix on all the standard classification task sets, while under-performing slightly on the cross-validation sets (Figure 9). Our method also outperformed the empirical kernel on both COG sets and all but one SCOP sets (Figure 10), and beat the spectrum kernel on the Eukaryotes-Prokaryotes COG task set (Figure 11). Overall, our method had a higher average ROC AUC than all other methods on the 3PGK (Figure 8) and COG datasets, and beat all methods but all-vs-all Smith-Waterman alignment on CATH and SCOP (Table 3).

5 Discussion

Our K-means-based representation learning method performs on par with state of the art protein classification techniques. Overall, our method tends to outperform established methods on the standard protein classification tasks in the CATH and SCOP datasets, which generally seem to be more difficult by virtue of the lower performance by all methods, while slightly underperforming against BLAST similarity matrix methods on the 5-fold cross validation tasks (Figures 9 and 10). Our method under-performed against Smith-Waterman similarity matrices on CATH and SCOP, but outperformed them on COG and 3PGK. This may be

Table 4: Wilcoxon p-values for CATH. K-means performs significantly better than the spectrum kernel and empirical kernel map.

CATH	16k clusters	Spectrum	Empirical
16k clusters	-	0.008	<0.0001
Spectrum	0.008	-	0.003
Empirical	<0.0001	0.003	-

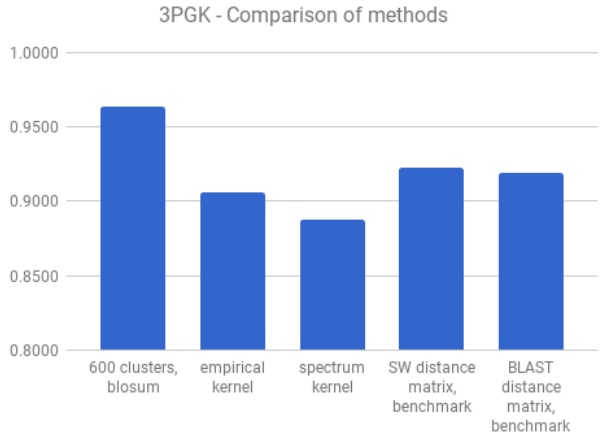

Figure 8: Comparison of K-means to empirical kernal map, spectrum kernel, and BLAST similarity matrix on 3PGK data.

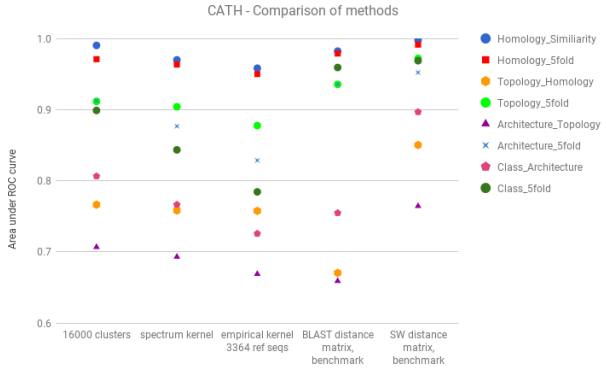

Figure 9: Comparison of K-means to empirical kernal map, spectrum kernel, and BLAST similarity matrix on CATH data.

due to the higher average sequence length of COG and 3PGK (as seen on Table 1).

Our method also shows promise in its ability to generalize. Despite low similarity between the CATH and COG sequence data, the features learned from the COG data were nearly as useful in learning to classify CATH proteins as features learned from CATH proteins (Figure 12). This suggests that the features being learned are generic protein features, though further testing is required to establish just how general they are. When applied to image data, K-means-based feature learning is able to learn generic visual features such as corners or lines or a particular orientation [4]. In principle, the same should be possible for proteins.

Furthermore, our method has potential utility for protein alignment as well. It may be possible to use

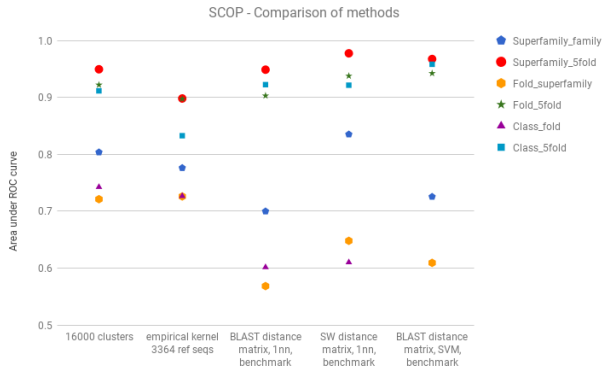

Figure 10: Comparison of K-means to empirical kernal map and BLAST and Smith-Waterman similarity matrices on SCOP data.

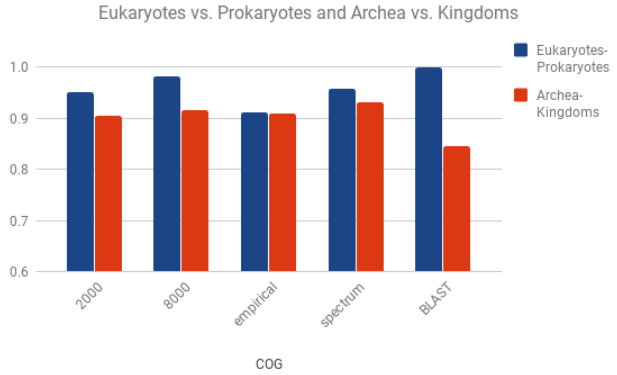

Figure 11: Results on COG data.

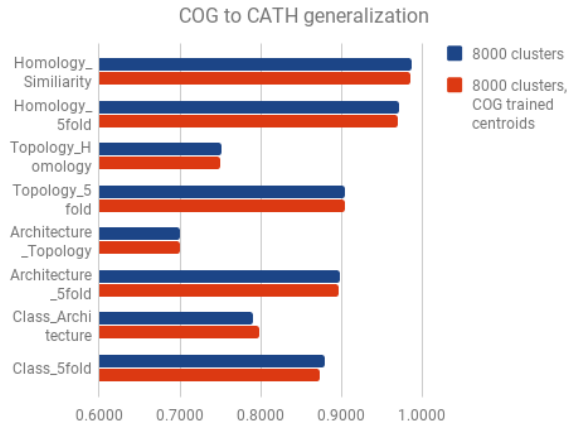

Figure 12: Comparison of classification performance on CATH dataset of centroids trained on CATH vs. centroids trained on COG.

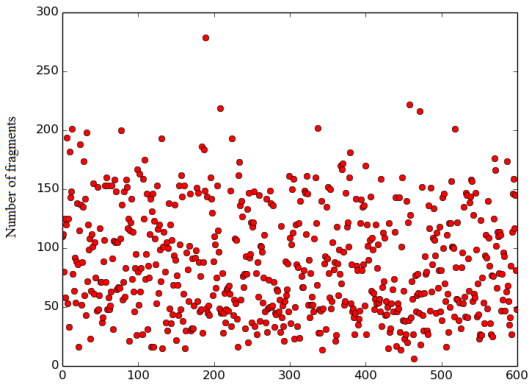

Figure 13: Distribution of 3PGK fragments across 600 clusters.

it to classify and rank alignments to determine the best ones.

Our code for fragmenting, clustering, and classifying protein sequences is available at `https://web.njit.edu/~usman/feature_learning_protein_classification`

6 Acknowledgment

We thank the NJIT Academic and Research Computing Systems Group (ARCS) for their support in running experiments for this study.

References

[1] Stephen F Altschul et al. "Basic local alignment search tool". In: *Journal of molecular biology* 215.3 (1990), pp. 403–410.

[2] Antonina Andreeva et al. "SCOP database in 2004: refinements integrate structure and sequence family data". In: *Nucleic acids research* 32.suppl_1 (2004), pp. D226–D229.

[3] Adam Coates and Andrew Y Ng. "Learning feature representations with k-means". In: (2012), pp. 561–580.

[4] Adam Coates, Andrew Ng, and Honglak Lee. "An analysis of single-layer networks in unsupervised feature learning". In: (2011), pp. 215–223.

[5] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[6] Murat Dundar et al. "Simplicity of kmeans versus deepness of deep learning: A case of unsupervised feature learning with limited data". In: (2015), pp. 883–888.

[7] Robert C Edgar. "MUSCLE: multiple sequence alignment with high accuracy and high throughput". In: *Nucleic acids research* 32.5 (2004), pp. 1792–1797.

[8] Steven Henikoff and Jorja G Henikoff. "Amino acid substitution matrices from protein blocks". In: *Proceedings of the National Academy of Sciences* 89.22 (1992), pp. 10915–10919.

[9] Christina Leslie, Eleazar Eskin, and William Stafford Noble. "The spectrum kernel: A string kernel for SVM protein classification". In: (2001), pp. 564–575.

[10] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[11] Frances Pearl et al. "The CATH Domain Structure Database and related resources Gene3D and DHS provide comprehensive domain family information for genome analysis". In: *Nucleic acids research* 33.suppl_1 (2005), pp. D247–D251.

[12] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[13] J Dennis Pollack, Qianqiu Li, and Dennis K Pearl. "Taxonomic utility of a phylogenetic analysis of phosphoglycerate kinase proteins of Archaea, Bacteria, and Eukaryota: insights by Bayesian analyses". In: *Molecular phylogenetics and evolution* 35.2 (2005), pp. 420–430.

[14] Rabie Saidi, Mondher Maddouri, and Engelbert Mephu Nguifo. "Protein sequences classification by means of feature extraction with substitution matrices". In: *BMC bioinformatics* 11.1 (2010), p. 175.

[15] Temple F Smith and Michael S Waterman. "Identification of common molecular subsequences". In: *Journal of molecular biology* 147.1 (1981), pp. 195–197.

[16] Paolo Sonego et al. "A protein classification benchmark collection for machine learning". In: *Nucleic Acids Research* 35.suppl_1 (2006), pp. D232–D236.

[17] JingLei Tang et al. "Weed identification based on K-means feature learning combined with convolutional neural network". In: *Computers and Electronics in Agriculture* 135 (2017), pp. 63–70.

[18] Roman L Tatusov et al. "The COG database: an updated version includes eukaryotes". In: *BMC bioinformatics* 4.1 (2003), p. 41.