Clustering Protein Sequences With Limited Distance Information

Konstantin Voevodski
Department of Computer Science
Boston University
kvodski@bu.edu

Maria-Florina Balcan
College of Computing
Georgia Institute of Technology
ninamf@cc.gatech.edu

Heiko Röglin
Department of Computer Science
University of Bonn
heiko@roeglin.org

Shang-Hua Teng
Computer Science Department
University of Southern California
shanghua@usc.edu

Yu Xia
Bioinformatics Program and Department of Chemistry
Boston University
yuxia@bu.edu

Clustering is very useful for exploring relationships between protein sequences. However, most clustering algorithms require distances between all pairs of points as input, which is infeasible to obtain for very large protein sequence data sets. Even with a *one versus all* distance query such as BLAST (Basic Local Alignment Search Tool) [AGM$^+$90], which efficiently compares a sequence to an entire database of sequences, it may not be possible to use it n times to construct the entire pairwise distance matrix, where n is the size of the data set. We present a clustering algorithm that gives an accurate clustering using only $O(k \log k)$ queries, where k is the number of clusters.

We analyze the correctness of our algorithm under a natural assumption about the data, namely the (c, ϵ) approximation stability property of [BBG09]. Balcan et al. assume that there is some relevant "target" clustering C_T, and optimizing a particular objective function for clustering (such as min-sum) gives clusterings that are structurally close to C_T. More precisely, they assume that any c-approximation of the objective is ϵ-close to C_T, where the distance between two clusterings is the fraction of misclassified points under the optimum matching between the two sets of clusters. Our contribution is designing an efficient algorithm that given this approximation stability assumption for the *min-sum* objective produces an accurate clustering using few *one versus all* distance queries. The *min-sum* objective function is to minimize $\Phi(C) = \sum_{i=1}^{k} \sum_{x,y \in C_i} d(x, y)$, where C is a k-clustering that partitions the points into k sets $C_1, \ldots, C_k$.

The algorithm presented here is related to the one presented in [VBR$^+$10]. The *Landmark-Clustering* algorithm presented there gives an accurate clustering if the instance satisfies the (c, ϵ)-property for the k-median objective. However, if the property is satisfied for the *min-sum* objective the structure of the clustering instance is quite different, and the algorithm given in [VBR$^+$10] fails to find an accurate clustering in such cases. The min-sum objective is also considerably harder to approximate. For k-median the best approximation guarantee is $(3 + \epsilon)$ given by [AGK$^+$04]. For the min-sum objective when the number of clusters is arbitrary there is an $O(\delta^{-1} \log^{1+\delta} n)$-approximation algorithm with running time $n^{O(1/\delta)}$ due to [BCR01].

Topic: computational biology
Preference: oral
Presenter: Konstantin Voevodski

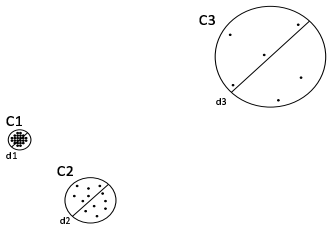

Figure 1: Cluster cores C_1, C_2 and C_3 are shown with diameters d_1, d_2 and d_3, respectively. The diameters of the cluster cores are inversely proportional to their sizes.

1 Algorithm

Our *Landmark-Clustering-Min-Sum* algorithm starts by uniformly at random choosing some points that we call *landmarks*. For each landmark we use a *one versus all* query to get the distances between this landmark and all other points. These are the only distances used by our procedure.

Our algorithm then expands a ball B_l around each landmark $l \in L$ one point at a time. We sort all landmark-point pairs (l, s) by the distance between l and s, and then consider the pairs (l, s) in order of increasing distance. In each iteration we add s to B_l, and check whether some ball B_{l^*} satisfies $|B_{l^*}| \cdot r > T$, where r is the next largest landmark-point distance, and T is an input parameter. If this is the case, we consider all balls that overlap B_{l^*} on any points, and compute a cluster that contains all the points in these balls. Points and landmarks in the cluster are then removed from further consideration. Our procedure terminates once we find k clusters. The algorithm takes k, the number of landmarks, and T as an argument. In our proof T is set based on c, ϵ, and the optimum objective value OPT. In practice we do not know these values, but we can try increasing estimates of T until the first k clusters contain enough of the points. This procedure gives a provably correct clustering if we do not know OPT but we know c and ϵ.

The most time-consuming part of our algorithm is sorting all landmark-points pairs, which takes $O(|L|n \log n)$, where n is the size of the data set and L is the set of landmarks. With a simple implementation that uses a hashed set to store the points in each ball, the total cost of computing the clusters and removing clustered points from other balls is at most $O(|L|n)$ each. All other operations take asymptotically less time, so the overall runtime of our procedure is $O(|L|n \log n)$. If the target clusters are balanced in size, it suffices to use $O(k \log k)$ landmarks to produce an accurate clustering.

We can prove that if the clustering instance satisfies the (c, ϵ)-property for the min-sum objective function, then with high probability *Landmark-Clustering-Min-Sum* finds a clustering that is ϵ-close to the target clustering C_T. Our proof requires that the distance function is a metric, we know the values of c, ϵ, and the optimal objective value OPT, and that the sizes of the target clusters are not too small. We next give some intuition for our argument.

Given our approximation stability assumption, the target clustering must have the structure shown in Figure 1. Each target cluster C_i has a "core" of well-separated points, where any two points in the cluster core are closer than a certain distance d_i to each other, and any point in a different core is farther than cd_i, for some constant c. Moreover, the diameters of the cluster cores are inversely proportional to the cluster sizes: there is some constant θ such that $|C_i| \cdot d_i = \theta$ for each cluster C_i. Given this structure, it is possible to classify the points in the cluster cores correctly if we extract the smaller diameter clusters first. For example, we can extract C_1, followed by C_2 and C_3 if we choose the threshold T correctly and we have selected a landmark from each cluster core. However, if we wait until some ball contains all of C_3, C_1 and C_2 may be merged.

2 Experiments

We present some preliminary results of testing our *Landmark-Clustering-Min-Sum* algorithm on protein sequence data. Instead of requiring all pairwise distances between the sequences as input, our algorithm is able to find accurate clusterings by using only a few BLAST calls. For each data

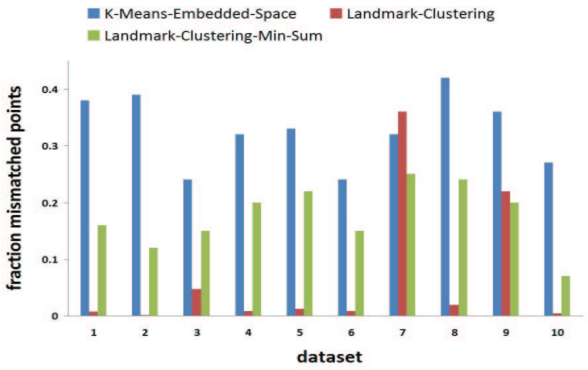

Figure 2: Comparing the performance of k-means in the embedded space (blue), *Landmark-Clustering* (red), and *Landmark-Clustering-Min-Sum* (green) on 10 data sets from Pfam. Datasets **1-10** are created by uniformly at random choosing 8 families from Pfam of size s, $1000 \leq s \leq 10000$.

set we first build a BLAST database containing all the sequences, and then compare only some of the sequences to the entire database. To compute the distance between two sequences, we invert the bit score corresponding to their alignment, and set the distance to infinity if no significant alignment is found. In practice we find that this distance is almost always a metric, which is consistent with our theoretical assumptions.

In our computational experiments we use data sets created from the Pfam [FMT$^+$10] (version 24.0, October 2009) database, which classifies proteins by their evolutionary relatedness. These are the same data sets that were used in the [VBR$^+$10] study, therefore we also show the results of the original *Landmark-Clustering* algorithm on these data, and use the same amount of distance information for both algorithms ($30k$ landmarks/queries for each data set, where k is the number of clusters). In order to compare a computationally derived clustering to the one given by the gold-standard classification, we use the distance measure from the theoretical part of our work, which is computed by solving a minimum weighted bipartite matching problem between the two sets of clusters.

Because our Pfam data sets are very large, we cannot compute the full distance matrix, so we can only compare with methods that use a limited amount of distance information. A natural choice is the following algorithm: uniformly at random choose a set of landmarks L, $|L| = d$; embed each point in a d-dimensional space using distances to L; use k-means clustering in this space (with distances given by the Euclidean norm). This procedure uses exactly d one versus all distance queries, so we can set d equal to the number of queries used by the other algorithms.

Figure 2 shows the results of our experiments. We can see that *Landmark-Clustering-Min-Sum* outperforms *k-means in the embedded space* on each data set. However, it does not perform better than the original *Landmark-Clustering* algorithm on most of these data sets. When we investigate the structure of the ground truth clusters in these data sets, we see that the diameters of the clusters are roughly the same. When this is the case the original algorithm will find accurate clusterings as well [VBR$^+$10]. Still, *Landmark-Clustering-Min-Sum* tends to give better results when the original algorithm does not work well (data sets 7 and 9).

We plan to conduct further studies to find data where clusters have different scale and there is an inverse relationship between cluster sizes and their diameters. This may be the case for data that have many outliers, and the correct clustering groups sets of outliers together rather than assigns them to arbitrary clusters. The algorithm presented here will consider these sets to be large diameter, small cardinality clusters. More generally, the algorithm presented here is more robust because it will give an answer no matter what the structure of the data is like, whereas the original *Landmark-Clustering* algorithm often fails to find a clustering if there are no well-defined clusters in the data.

References

[AGK+04] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3), 2004.

[AGM+90] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, 1990.

[BBG09] M. F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *Proc. of 20th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2009.

[BCR01] Y. Bartal, M. Charikar, and D. Raz. Approximating min-sum k-clustering in metric spaces. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, 2001.

[FMT+10] R.D. Finn, J. Mistry, J. Tate, P. Coggill, A. Heger, J.E. Pollington, O.L. Gavin, P. Gunesekaran, G. Ceric, K. Forslund, L. Holm, E.L. Sonnhammer, S.R. Eddy, and A. Bateman. The pfam protein families database. *Nucleic Acids Res.*, 38:D211–222, 2010.

[VBR+10] K. Voevodski, M. F. Balcan, H. Röglin, S. Teng, and Y. Xia. Efficient clustering with limited distance information. In *Proc. of the 26th Conference on Uncertainty in Artifcial Intelligence*, 2010.