DIHNOSIR: A graph-theoretic DBSCAN-based approach to clustering

Amir Shanehsazzadeh[1], Roland L. Dunbrack, Jr.[1*]

1 Institute for Cancer Research, Fox Chase Cancer Center, Philadelphia, PA, USA

* roland.dunbrack@fccc.edu

Abstract

Cluster analysis is a means of categorizing elements of a data set together based on properties of the data. Traditional density-dependent clustering algorithms such as DBSCAN are effective in clustering data with convex clusters and some noise (unstructured points). However, these algorithms fail to effectively cluster data with high (10 - 20)% noise and varying cluster density. In this work, we present two versions of DIHNOSIR: Density-Independent High Noise Optimized Sorting by Iterative Reduction, Strong DIHNOSIR and Weak DIHNOSIR. Strong DIHNOSIR runs DBSCAN over a 2-D grid of $(\epsilon, minPts)$ parameters and stores the clusters returned by DBSCAN that satisfy a user-defined constraint as subclusters $\{S_1, S_2, ..., S_m\}$. We create an undirected graph with vertices $\{S_1, S_2, ..., S_m\}$ and for any i, j we connect S_i and S_j using Simpson's similarity index (see Equation 4). If $SSI(i, j)$ is greater than a positive threshold value an edge is placed between S_i and S_j. The connected components of this graph are considered and the union of the subclusters that form said connected components are used to form the final clusters $\{C_1, C_2, ..., C_k\}$. Weak DIHNOSIR works similarly to Strong DIHNOSIR, but does not require a user-defined cluster constraint function. Instead, Weak DIHNOSIR removes merged clusters by considering the indegree of vertices in the transitive reduction of the directed graph formed with the subclusters, $\{S_1, S_2, ..., S_m\}$ as vertices and edges formed from S_i to S_j if $S_i \subseteq S_j$. The user is left the task of running additional iterations of DIHNOSIR and pruning the data by removing or combining the returned clusters based on the goals of their clustering. DIHNOSIR's power is demonstrated on Ramachandran plots representing flips and antibodies. The accurate clustering of these data allows for improved protein structure design, prediction, and validation. DIHNOSIR's source code and documentation can be found at https://github.com/amirshane/dihnosir.

Author summary

Ramachandran plots are an effective way of visualizing the energetically feasible values for the dihedral angles ψ and ϕ of amino acid residues in protein structures. These plots are one of many real-world examples of data with high (10 - 20)% noise content and varying cluster density. Generally there are three obviously dense regions in a Ramachandran plot: α, α_L, and β, which correspond to right handed alpha-helices, left handed alpha-helices, and beta sheets, respectively. However, the β region consists of three distinct sub-regions: β_s, β_p, and γ. Clustering Ramachandran plots with traditional algorithms such as DBSCAN is difficult due to the aforementioned properties of the data. The ability to cluster these plots would provide a great deal of information regarding the feasibility of potential backbone conformations of various proteins. Furthermore, a variety of other forms of data have similar properties that make them difficult to cluster with current density-dependent algorithms. Our approach was to create DIHNOSIR, a semiautonomous density-independent algorithm based on DBSCAN that works by running numerous instances of DBSCAN on the data and using a graph-theoretic approach on the subclusters returned by DBSCAN to create final clusters. DIHNOSIR's power is demonstrated by clustering two data sets of protein structural data that are derived from flip and antibody backbone dihedral angle measures.

Introduction

Cluster Analysis

Cluster analysis encompasses the numerous methods of unsupervised pattern discovery in data sets based on various similarities between data points. A cluster is a homogeneous grouping of objects. The definition of homogeneity varies but generally characteristics such as distance are used to determine how closely related two objects are. Robust clustering algorithms must not only be able to create homogeneous groupings of related data but they must also be able to determine non-homogeneous groupings of data that are non-informative, known as noise. Data with cluster "separation," substantial distance between clusters, are much easier to cluster. The addition of noise to a data set poses a significant issue. The position of the noise relative to the clusters is also important. Noise that is interspersed in between clusters is much more difficult to discern than noise that represents outliers in the data. Furthermore, different clustering methods must be used for different clustering objectives. [3] Therefore, we must not have a rigid definition of clustering. Rather, for best results, we must evaluate the data on a case-by-case basis to determine what exactly the objective of the clustering is.

With that said it is essential to have both a qualitative and quantitative understanding of the DBSCAN algorithm that DIHNOSIR is based on. Algorithms like DBSCAN [12] do not require the number of clusters but they are dependent on the parameters ϵ, which is the maximum radius to consider and $minPts$, which is the minimum number of points required to form a cluster. Although DBSCAN is robust when dealing with a variety of data. The question of what ϵ and $minPts$ combinations should be used is still open. DBSCAN relies heavily on these two parameters to determine the shape and size of the determined clusters. Hierarchical clustering [4] has been helpful in determining the best values of ϵ and $minPts$ for DBSCAN, but the fact that these algorithms are density-dependent results in them failing to cluster data with varying cluster density. Additionally, the varying cluster density results in these algorithms failing to be robust when faced with noise that is interspersed between clusters with significantly different densities. A more rigorous explanation of the DBSCAN algorithm is given in the Methods section.

It is not intuitively obvious how to determine whether a clustering is "good." Cluster validation is a field in itself. [8,13,15] While attempting to validate sub-clusterings of our data we cannot use a ground truth (beforehand knowledge of the answer). Furthermore, there are very few clustering metrics that are unsupervised and do not rely on a ground truth. Within DIHNOSIR we use one of these metrics: Silhouette Score. Silhouette Score is a measure of the quality of a model. This metric compares intra-cluster cohesion, similarity of elements within a cluster, to similarity with other clusters. [20] A mathematically rigorous explanation of Silhouette Score is given in the Methods section. We proceed by describing the type of data DIHNOSIR was originally designed for and then by giving a thorough explanation and justification of the algorithms and metrics we implement into DIHNOSIR. In order to showcase the individual steps used by DIHNOSIR we present pseudocode representations of DIHNOSIR's sub-algorithms. We also present DIHNOSIR's outputs for the flip and antibody data.

Ramachandran Plots

Ramachandran plots [18], developed by Gopalasamudram Ramachandran, are a popular method for visualizing the energetically allowed dihedral angles ψ and ϕ, planar angles around the C_α–C and C_α–N bonds, respectively, for amino acid residues in protein backbone structures. The Ramachandran plot consists of five discernible regions, each of which corresponds to a distinct backbone structure: α, α_L, β_s, and β_p, γ. [7] The α region corresponds to the right-handed helix structure. In this region the CO and NH dipoles are antiparallel and can form hydrogen bonds in opposite directions to create a helical structure. The orientation of the CO and NH dipoles determines whether a α-helix, π-helix, or 3_{10}-helix is formed. The α_L region is a near mirror image of the α region about the central diagonal and corresponds to the left-handed helix structure. Steric hindrance between C_β and O atoms prevents this region from being an exact mirror image. In the β_s region the CO and NH dipoles are parallel, which permits the formation of β-sheets. [23] In the β_p region the CO and NH dipoles are nearly perpendicular. This region is induced by optimization of the CO_{i-1}...CO interaction. [14] In the γ region the outer CO and NH dipoles form a highly distorted hydrogen bond, known as a γ-turn. This region is induced by the CO_{i-1}...NH_{i+1} interaction. [16]

Protein Backbone Conformations

The graphs presented in this paper are a simplification of the data. Generally, when dealing with real data there is an additional dimension that is scientifically qualitative rather than quantitative. This qualitative dimension gives a physical meaning to the data points and to the clusters that are returned. Our extra dimension represents the backbone conformations of the proteins being examined. This extra dimension can even be considered as a simplification of multiple dimensions, one for each residue in the protein. In this paper we look at the clusters as backbone conformations, for which each set of corresponding elements of the conformation belongs to its own Ramachandran plot. We use four letters to represent a conformation. The letter A represents the middle left region of the Ramachandran plot (right-handed α-helices), B represents the upper left region (β-sheets), L represents the middle right region (left-handed α-helices), and the letter E represents the lower right region (extended structures). The letter E is common for the amino acid Glycine (often refered to as G instead of E), which can exist in numerous allowed and disallowed regions of the Ramachandran plot due to its lack of a beta carbon C_β. [6]

The graphs in the Results section can be analyzed by choosing a specific residue in a conformation and locating the corresponding 2-D cell in the graph. The structure of the cluster should correspond to the region designated by the letter of the residue in the conformation. For example, if we look at the AA$\rightarrow$BL conformation for the flip data in Figure 1, residue 1 ($\mathrm{Res}_1 = \mathrm{A}$) has a cluster in the middle left region as expected. Similarly residue 2 ($\mathrm{Res}_2 = \mathrm{A}$), residue 3 ($\mathrm{Res}_3 = \mathrm{B}$), and residue 4 ($\mathrm{Res}_4 = \mathrm{L}$) have clusters in the middle left, upper left, and middle right regions, respectively, as expected.

DBSCAN

The DBSCAN algorithm [12] has been prominent since its introduction in 1996. Numerous variants of the original DBSCAN algorithm have also been created. [1, 21] DBSCAN does not require the number of clusters as an input. Instead it takes two inputs: ϵ, the largest radius to consider for a datum and $minPts$, the least number of points needed to form a cluster.

DBSCAN works by doing a distance-query for all points in the data set. Any datum with $minPts$ many points (including the datum itself) in an ϵ-neighborhood of the datum are labeled core points. A point p is directly-density reachable from a point q if q is a core point and $d(p, q) \leq \epsilon$. A point p is density-reachable from a point q if there exists a sequence of points $q = p_1, p_2, ..., p_n = p$ such that p_i is directly-density reachable from p_{i+1}. A point p is density-connected to a point q if there exists a point o such that both p and q are density-reachable from o. We abstractly define a cluster C in a data set X as a subset of X such that $\forall p, q \in X$, if $p \in C$ and q is density-reachable from p then $q \in C$ and $\forall p, q \in C$, p is density-connected to q. The DBSCAN algorithm first determines which datums are core points. Then it takes a core point p and forms a cluster with all points (core and non-core) that are density-reachable from p. The process is repeated with any core points within this cluster. All non-core points that are within a distance of ϵ from a cluster are deemed border points and the remaining non-core points are deemed noise.

Another way to think about DBSCAN is in a more abstract, graph-theoretic way. After determining all core points in a data set DBSCAN creates a graph of all points in the data with edges between ϵ-neighbors. The clusters are initially set to the connected components of the core points in the graph. Non-core points are added to a cluster if they are an ϵ-neighbor of said cluster. Otherwise they are added to noise.

Silhouette Score

The Silhouette Score [20] measures how well a clustering model fits an individual point. All individual points are considered by comparing intra-cluster cohesion to inter-cluster separation. For a given $x \in X$ we let

$$s(x) = \frac{b(x) - a(x)}{max(a(x), b(x))} \in [-1, 1] \tag{1}$$

where $a(x)$ is the average distance of x to all other points in its cluster and $b(x)$ is the minimum average distance from x to all points in the cluster nearest to x that x is not contained in. Summing over all datums gives the silhouette score of a clustering as

$$S(X) = \sum_{x \in X} \frac{s(x)}{|X|} \in [-1, 1]. \tag{2}$$

Silhouette Score works well with most distance metrics. The Minkowski distance and Euclidean or Manhattan metrics are most common. [19]

Graph Theory

DIHNOSIR's formation of clusters uses a graph theoretic approach. A graph G is an ordered pair (V, E) where V is a set of vertices also known as nodes and E is a set of edges which are 2-element subsets of V that indicate connectivity between the two elements. [24] In an undirected graph there is no distinction between a pair of vertices in an edge. In a directed graph an edge is directed from one vertex to another. Strong DIHNOSIR only uses undirected graphs, whereas Weak DIHNOSIR uses both directed and undirected graphs. A connected graph is an undirected graph for which there is a path between any two vertices of the graph. A subgraph G' of a graph G is an ordered pair (V', E') of vertices and edges for which $V' \subseteq V$ and $E' \subseteq E$. A connected subgraph of a graph G is thus a subgraph of G that is itself a connected graph. A connected component of a graph G is a connected subgraph of G that is not connected to the rest of the supergraph. The degree of a vertex in a graph is equal to the number of edges incident to this vertex. For a directed graph, the indegree of a vertex is the number of head ends adjacent to this vertex (edges directed at the vertex) and the outdegree of vertex is the number of tail ends adjacent to this vertex (edges directed from the vertex). The degree of a node in a directed graph is thus the sum of this node's indegree and outdegree. A directed acyclic graph (DAG) is a directed graph with no cycles (the graph contains no set of vertices $p_1, p_2, ..., p_n$ such that there exists a directed path from p_1 to p_2 to ... to p_n and from p_n to p_1). The transitive reduction of a directed acyclic graph is the subgraph of the DAG with the least number of edges such that the reachability relation is maintained (there exists a path from vertex p to vertex q in the transitive reduction if and only if there exists a path from p to q in the DAG).

Partially Ordered Sets

The more mathematically inclined readers may wish to interpret the transitive reduction as a method to represent directed acyclic graphs as partially ordered sets. A partially ordered set (poset) is a set P with an order relation $\leq$ that is reflexive ($x \leq x$), anti-symmetric ($x \leq y$ and $y \leq x \implies x = y$), and transitive ($x \leq y$ and $y \leq z \implies x \leq z$) for all $x, y, z \in P$. This order relation is defined for any $x, y \in P$ such that x and y are comparable (not all elements of P need to be comparable). Although we call our method a directed graph approach, it is not difficult to see that it is a poset approach as the transitive reduction converts the directed graph to a poset with the set being the subclusters, the order relation being the size of the two subclusters, and two subclusters defined as comparable based on whether or not the Simpson's similarity (Equation 4) between the two subclusters is greater than or equal to a positive value $ssiThreshold \in [0, 1]$. Thus, for given subclusters S_i, S_j we say $S_i \leq S_j$ if $SSI(i, j) \geq ssiThreshold$ and $|S_i| \leq |S_j|$.

Results

DIHNOSIR successfully clustered two data sets, which be label as flip data and antibody data. These data sets as well as the necessary scripts to apply DIHNOSIR to them can be found at https://github.com/amirshane/dihnosir. This repository contains the generalized source code for DIHNOSIR as well as two specific repositories for each of the two data sets.

Flip Data

DIHNOSIR successfully clustered a data set consisting of 714 points, which represent the structural conformations of flips between corresponding two-element amino acid sequences between two proteins, using both Strong and Weak DIHNOSIR. Two distance metrics, $d(x, y)$ and $d^{\infty}(x, y)$ were used. For specifics on the distance metrics as well as other parameters used see the Methods section.

Using the $d(x, y)$ distance metric, Strong DIHNOSIR returned the pruned clustering shown in Figure 1, which consists of a noise cluster of size 173 along with eight clusters of size 289, 79, 65, 46, 36, 13, 7, and 6 (from left to right). The clusters are labeled $AA \rightarrow BL$, $AL \rightarrow BA$, $AB \rightarrow BE$, $EA \rightarrow LL$, $AA \rightarrow BB$, $EE \rightarrow LB$, $EB \rightarrow LE$, and $EL \rightarrow LA$ to represent the structural conformation they correspond to. The arrow in the notation represents the flip (i.e. $AA \rightarrow BL$ indicates a flip from A to B in the first residue and A to L in the second residue). Using the $d^{\infty}(x, y)$ distance metric, Strong DIHNOSIR returned a pruned clustering that consists of a noise cluster of size 188 along with eight clusters of size 281, 70, 69, 46, 38, 13, 3, and 6 (from left to right). The clusters are labeled $AA \rightarrow BL$, $AL \rightarrow BA$, $AB \rightarrow BE$, $EA \rightarrow LL$, $AA \rightarrow BB$,

$EE \to LB$, $EB \to LE$, and $EL \to LA$ (see above). Using the $d(x, y)$ distance metric, Weak DIHNOSIR returned the pruned clustering shown in Figure 2, which consists of a noise cluster of size 181 along with seven clusters of size 338, 51, 49, 46, 30, 12, and 7 (from left to right). The clusters are labeled $AA \to BL$, $AL \to BA$, $AB \to BE$, $EA \to LL$, $AA \to BB$, $EE \to LB$, and $EB \to LE$ to represent the structural conformation they correspond to. Using the $d^{\infty}(x, y)$ metric, Weak DIHNOSIR returned a pruned clustering that consists of a noise cluster of size 197 along with seven clusters of size 339, 49, 34, 46, 30, 13, and 6 (from left to right). The clusters are labeled $AA \to BL$, $AL \to BA$, $AB \to BE$, $EA \to LL$, $AA \to BB$, and $EL \to LA$ (see above). For a comparison of the different results see Table 1. Regarding the graphs, each cell in the grid is a 2-D plot with domain and range equal to $[-180, 180]$. The corresponding conformations and residues are shown for each cell as well. Note: In the clustering it may seem that some clusters are disconnected and appear in different quadrants. It is important to realize that this data wraps around itself. This is difficult to visualize but an initial approach should be to imagine a piece of paper being folded horizontally and vertically at the same time.

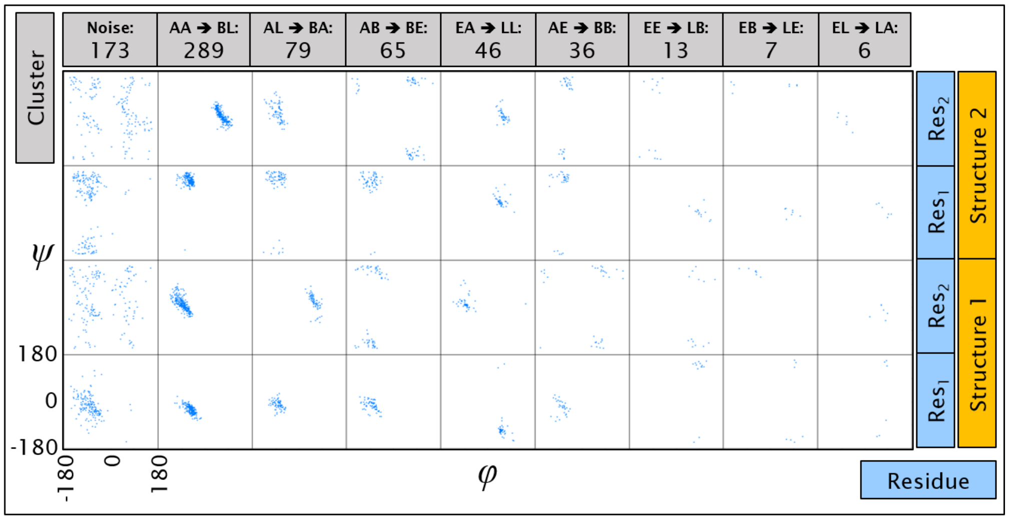

Fig 1. Clustering of Flip Data Using Strong DIHNOSIR with $d(x, y)$ Distance Metric

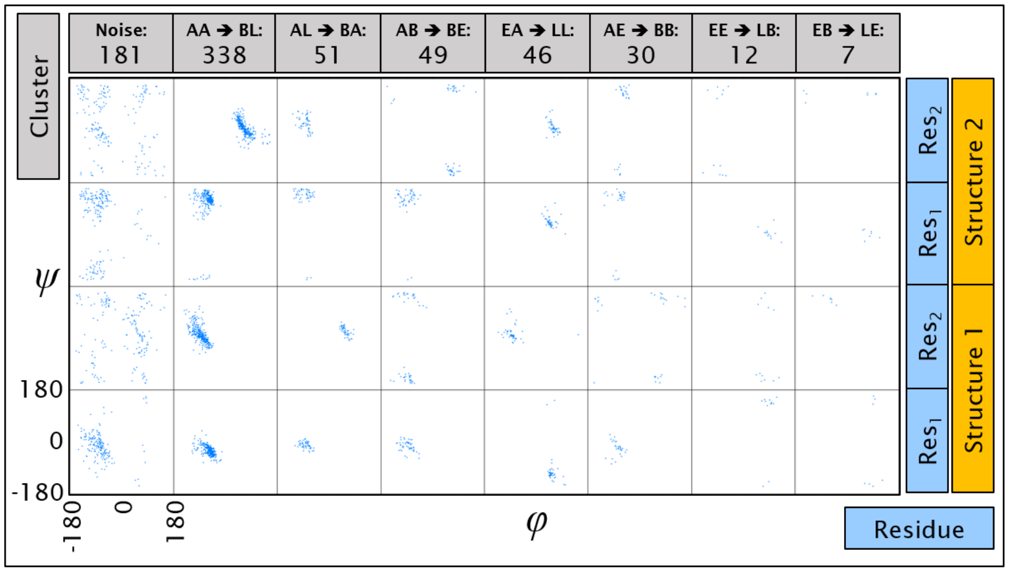

Fig 2. Clustering of Flip Data Using Weak DIHNOSIR with $d(x, y)$ Distance Metric

Size	Noise	*AABL*	*ALBA*	*ABBE*	*EALL*	*AABB*	*EELB*	*EBLE*	*ELLA*
Strong + $d(x, y)$	173	289	79	65	46	36	13	7	6
Strong + $d^{\infty}(x, y)$	188	281	70	69	46	38	13	3	6
Weak + $d(x, y)$	181	338	51	49	46	30	12	7	0
Weak + $d^{\infty}(x, y)$	197	339	49	34	46	30	13	0	6

Table 1. Sizes of Clusters Returned by Running DIHNOSIR on Flip Data

We analyzed the distribution of amino acid sequences that formed the clusters in the flip data by creating WebLogos. For the WebLogos created from the clustering using Strong DIHNOSIR and $d(x, y)$ see Figure 3. For the WebLogos created from the clustering using Weak DIHNOSIR and $d(x, y)$ see Figure 4. Note that the two central amino acids in the four amino acid sequence are flipped. The leftmost and rightmost amino acids are adjacent to the flipped amino acids and are not themselves flipped.

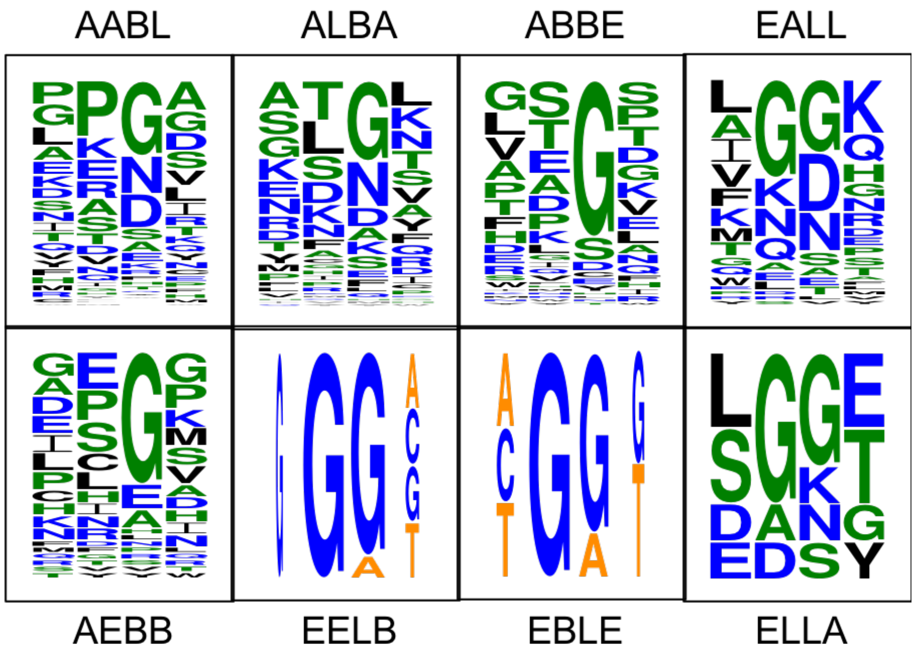

Fig 3. WebLogos of Flip Data Using Strong DIHNOSIR with $d(x, y)$ Distance Metric (y-axis is Probability)

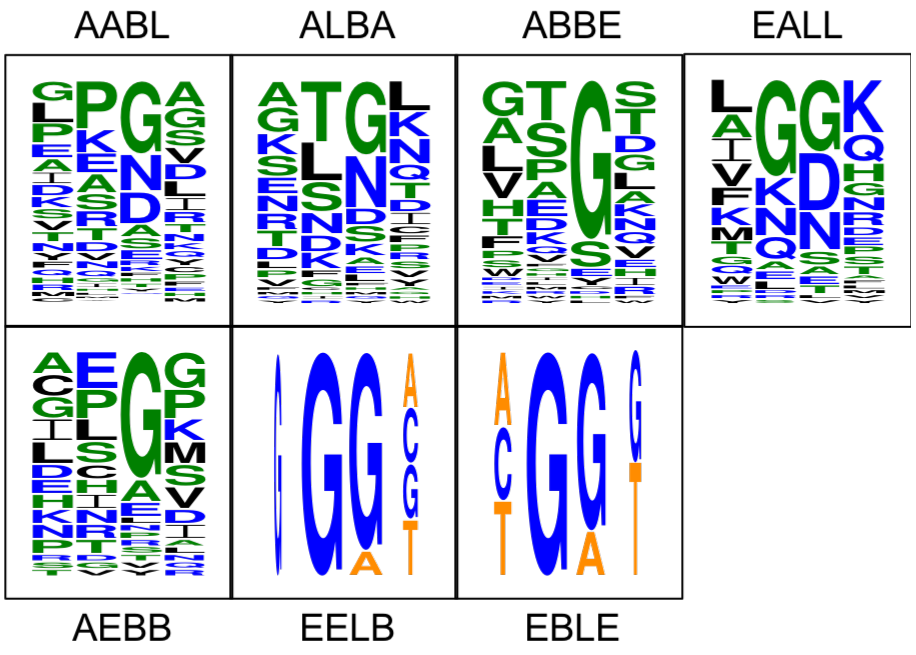

Fig 4. WebLogos of Flip Data Using Weak DIHNOSIR with $d(x, y)$ Distance Metric (y-axis is Probability)

Antibody Data

DIHNOSIR clustered a data set consisting of 1746 points, each of which represents an eleven residue CDR L1 (complementary-determining region) of an antibody, using both Strong and Weak DIHNOSIR. Two distance metrics, $d(x, y)$ and $d^{\infty}(x, y)$, were used. For specifics on the distance metrics as well as other parameters used see the Methods section.

The pruned clustering using Strong DIHNOSIR and the $d(x, y)$ metric consists of a noise cluster of size 165 along with seven clusters of size 982, 374, 100, 53, 28, 21, 23 (from left to right). The clusters are labeled $S_1, S_2, ..., S_7$, where S_1 = BBABBAEABBB and BBABBAEAABB (merged), S_2 = BBABBALLBBB, S_3 = BBBLAAABBBB, S_4 = BBAAAAABBBB, S_5 = BBABBBAABBB, S_6 = BBABBBBLBBB and BBABBBBLABB (merged), and S_7 = EBBBABBBBBB to represent the structural conformation they correspond to. The pruned clustering of the antibody data using Strong DIHNOSIR and the $d^{\infty}(x, y)$ metric is shown in Figure 5 and consists of a noise cluster of size 151 along with seven clusters of size 998. 375, 100, 53, 28, 23, 18 (from left to right). The clusters are labeled $S_1, S_2, ..., S_7$ (see above). Pruning was not needed for the results outputted by Weak DIHNOSIR. The clustering of the antibody data using Weak DIHNOSIR and $d(x, y)$ as a distance metric consists of a noise cluster as well as 5 clusters. The noise cluster consisted of 162 points and the five clusters consisted of 1045, 377, 100, 49, and 13 points. The clusters are labeled $S_1, S_2, ..., S_5$, where S_1 = BBABBBEABBB and BBABBAEAABB (merged), S_2 = BBABBALLBBB, S_3 = BBBLAAABBBB, S_4 = BBAAAAABBBB, and S_5 = EBBBABBBBBB, to represent the structural conformation they correspond to. The clustering of the antibody data using Weak DIHNOSIR and $d^{\infty}(x, y)$ as a distance metric, shown in Figure 6, consisted of a noise cluster as well as five clusters. The noise cluster consisted of 112 points and the 5 clusters consisted of 1063, 400, 100, 53, and 18 points. The clusters are labeled $S_1, S_2, ..., S_5$ (see above). For a comparison of the different results see Table 2 (S_7 in the table represents the EBBBABBBBBB conformation).

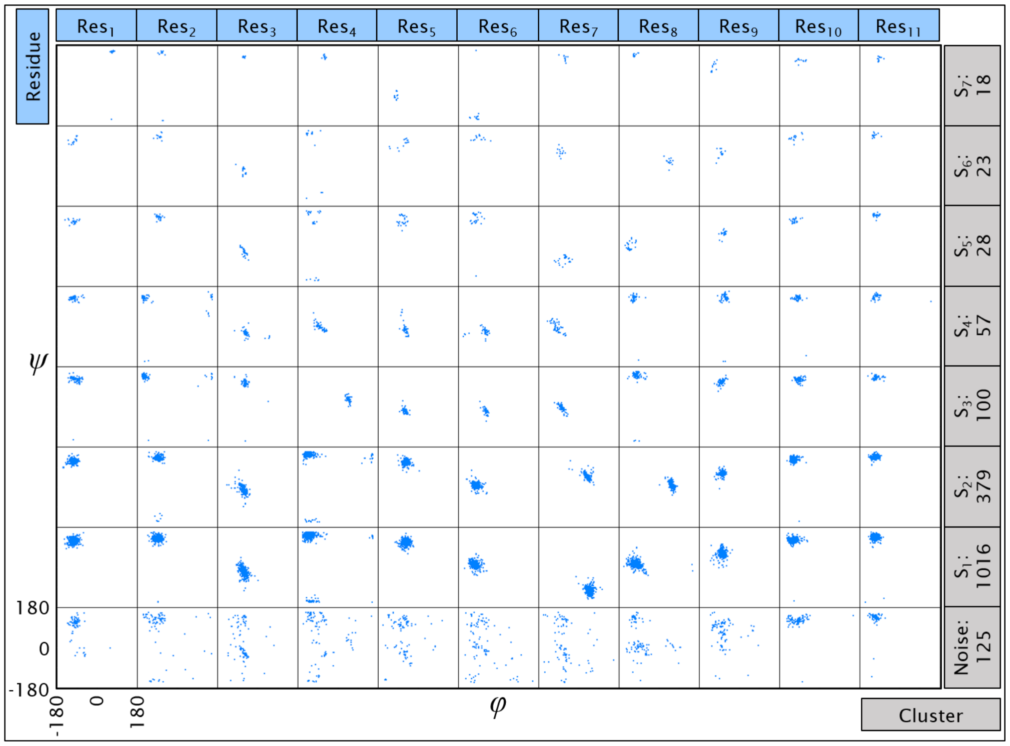

Fig 5. Clustering of Antibody Data Using Strong DIHNOSIR with $d^{\infty}(x, y)$ Distance Metric

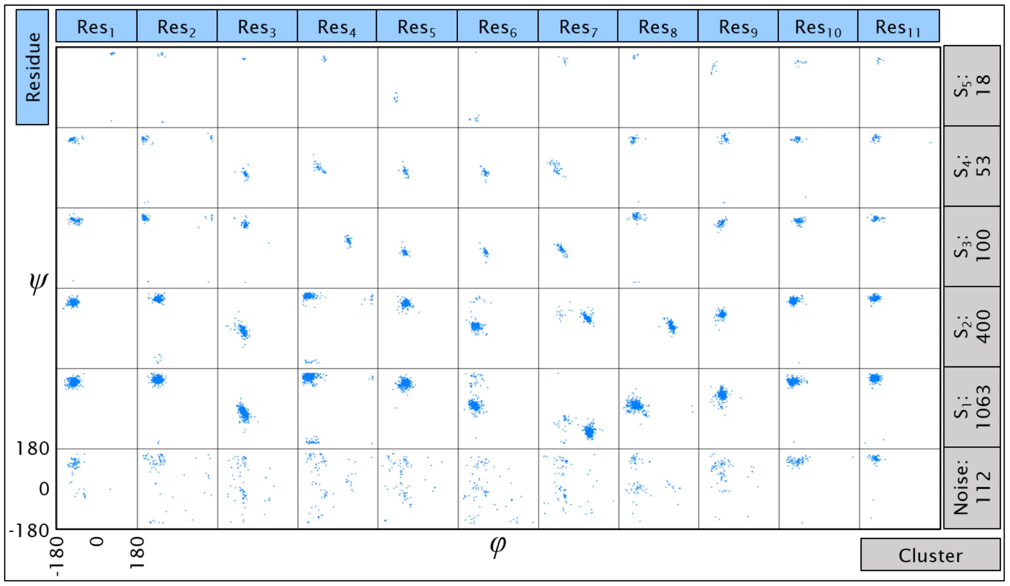

Fig 6. Clustering of Antibody Data Using Weak DIHNOSIR with $d^{\infty}(x, y)$ Distance Metric

Size	Noise	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Strong + $d(x,y)$	134	1010	375	100	53	28	20	26
Strong + $d^{\infty}(x,y)$	125	1016	379	100	57	28	23	18
Weak + $d(x,y)$	162	1045	377	100	49	0	0	13
Weak + $d^{\infty}(x,y)$	112	1063	400	100	53	0	0	18

Table 2. Sizes of Clusters Returned by Running DIHNOSIR on Antibody Data

Discussion

We have designed the novel clustering algorithm DIHNOSIR for the clustering of protein dihedral angle data as well as any other data with a distance metric. While other clustering algorithms such as DBSCAN may be able to cluster these data sets, the results can be highly sensitive to the input parameters. Additionally, multiple iterations of these algorithms may be needed to be set up manually and individually analyzed and pieced together to produce a valuable result. DIHNOSIR effectively automates this process with the DBSCAN algorithm and will thus be incredibly useful in analyzing data, particularly protein structural data. Strong DIHNOSIR's efficacy given a properly defined cluster constraint function that removes merged clusters is shown by the high quality clusterings of the flip and antibody data. We also show that Weak DIHNOSIR produces a comparable result to Strong DIHNOSIR despite the fact that it does not require a cluster constraint function. The source code has been made available, along with the flip and antibody data and various scripts needed to format the data for use with DIHNOSIR, effectively creating examples of its usage.

From a data set of 714 flips identified in ? proteins at a resolution of $1.6\mathring{A}$, DIHNOSIR identified eight flip types, the exact number that was expected. The eight flip types are $AA \to BL$, $AL \to BA$, $AB \to BE$, $EA \to LL$, $AA \to BB$, $EE \to LB$, $EB \to LE$, and $EL \to LA$. In the first position, A can go to B and E can go to L by flipping ψ by 180 degrees. In the second position, (A, B, L, E) can flip to (L, E, A, B) (i.e. $A \leftrightarrow L$ and $B \leftrightarrow E$ flips) by flipping ϕ by 180 degrees. We also define $AL \to BA$ to be equivalent to $BA \to AL$ (and similarly for the other flip types). We thus have $4 * 4/2 = 8$ flips.

Methods

Computational Methods

DIHNOSIR was coded using the Python programming language. The R programming language was also used to help visualize the data. The clustering algorithms and metrics used within DIHNOSIR are imported from scikit-learn. [17] The graph-theoretic methods are imported from NetworkX. [5]

Algorithms

Pseudocode Notation

In the following sections we explain the algorithms used in DIHNOSIR. These algorithms are presented in pseudocode. We use standard pseudocode conventions, however a few of the lines of pseudocode use syntax similar to the Python programming language. The *size* method takes a data structure such as a list or an array and returns the number of elements in it. With list and array data types we use the syntax for Python list/array slicing. For example, for a list *myList* the object $myList[n]$ would represent the $(n+1)$th element in the list. The *min* and *max* methods take a data structure and return the minimum and maximum elements of said data structure, respectively.

Subcluster Determination

The first step is to determine the subclusters. This is done by running multiple instances of DBSCAN on the data set over a 2-D grid of ϵ, $minPts$ parameters and creating a new data set composed of the returned subclusters that satisfy a user-defined constraint. The user must provide two general parameters, $minPts_{min}$ and n, as well as a more complicated user-defined constraint. The value for $minPts_{min}$ is the smallest value of $minPts$ to be considered. We recommend that $minPts_{min}$ be greater than 2. The default for $minPts_{min}$ is thus set at 3. This represents the minimum number of points the user wants in a cluster. We set $minPts_{max}$, the absolute maximum value of $minPts$ to be considered, equal to the number of samples in the data. We set ϵ_{min} and ϵ_{max} equal to the minimum and maximum distances in the data, respectively. This allows a fully comprehensive consideration of all distances in the data. The value n is used to determine the increment for ϵ in the grid:

$$\Delta\epsilon = \frac{\epsilon_{max} - \epsilon_{min}}{n}. \tag{3}$$

Within our grid we consider the set of ϵ values $\{\epsilon_{min} + \Delta\epsilon, \epsilon_{min} + 2\Delta\epsilon, ..., \epsilon_{max}\}$. We recommend that the value for n be approximately 100 or less for initial iterations. The default value for n is thus set at 100. Increasing n will generally increase the quality of the clustering but the improvements will eventually stagnate and the cost in computational complexity is linearly proportional to the value of n. DIHNOSIR runs DBSCAN over a subset of the $(minPts_{max} - minPts_{min} + 1) \times n$ grid:

$$\begin{bmatrix} (\epsilon_{min} + \Delta\epsilon, minPts_{min}) & \dots & (\epsilon_{max}, minPts_{min}) \\ (\epsilon_{min} + \Delta\epsilon, minPts_{min} + 1) & \dots & (\epsilon_{max}, minPts_{min} + 1) \\ \vdots & \ddots & \vdots \\ (\epsilon_{min} + \Delta\epsilon, minPts_{max}) & \dots & (\epsilon_{max}, minPts_{max}) \end{bmatrix}$$

Starting at the first row of this grid we run DBSCAN from left to right and add the returned subclusters to a new data set if they satisfy the user-defined constraint. We initialize a variable $breakCount$ to 0. If DBSCAN returns one cluster for a combination of ϵ and $minPts$ we break out of the row in the grid that we are in. If $\epsilon = \epsilon_{min} + \Delta\epsilon$ as well we add one to $breakCount$. Otherwise we set $breakCount$ to 0. If $breakCount$ reaches a value of 5 we multiply the $minPts$ increment $\Delta minPts$ (initially set to 1) by 2. This allows us to iterate more quickly through the grid and not add more computational expense by considering clusterings that are likely to only return a single cluster. At the end of this computation we use the method *unique* to remove any repeated subclusters. See Algorithm 1 for pseudocode. Note: The use of $breakCount$ is not highlighted in the pseudocode for sake of brevity.

Removing Merged Clusters

In order to use Strong DIHNOSIR, the user must provide a Boolean constraint function that prevents the merging of clusters in the data. This is why Strong DIHNOSIR is named as such, since it requires a strong understanding of the data in order to define

Algorithm 1: Subcluster Determination

```
createSubclusters($X$, $n$, $minPts_{min}$, $silhouetteThreshold$) Data: $X$, a square
        distance matrix; $n$, number of iterations; $minPts_{min}$, minimum cluster
        size; $silhouetteThreshold$, minimum Silhouette Score
Result: $S = \{S_1, S_2, ..., S_m\}$, a set of subclusters
$S \longleftarrow \emptyset$
$\epsilon_{min} \longleftarrow min(X)$
$\epsilon_{max} \longleftarrow max(X)$
$\Delta\epsilon \longleftarrow \frac{(\epsilon_{max} - \epsilon_{min})}{n}$
$minPts \longleftarrow minPts_{min}$
$\Delta minPts \longleftarrow 1$
while ($minPts \leq size(X)$) do
    $\epsilon \longleftarrow \epsilon_{min} + \Delta\epsilon$
    while ($\epsilon \leq \epsilon_{max}$) do
        $S^i \longleftarrow DBSCAN(X, \epsilon, minPts)$
        for $S^i_j$ in $S^i$ do
            if $silhouetteScore(S^i_j) \geq silhouetteThreshold$ then
                add $S^i_j$ to $S$
        $\epsilon \longleftarrow \epsilon + \Delta\epsilon$
    $minPts \longleftarrow minPts + \Delta minPts$
$S = unique(S)$
return $S$
```

such a function. This function is highly specific to the data being clustered and it is needed to ensure that a subcluster returned by DBSCAN is a subset of only a single true cluster in the data. If there is no data-specific function then the Silhouette Score can be used, however Silhouette Score works well primarily with the Euclidean, Manhattan, and Minkowski metrics. With Silhouette Score we suggest that the mean of the Silhouette Sample Scores in a subcluster returned by DBSCAN are relatively high and at least greater than 0.75. This constraint will be regarded as a method that takes a subcluster as an input and returns a Boolean value: $clusterConstraint(subcluster)$. See Algorithm 2 for pseudocode.

Algorithm 2: Strong Unmerge

```
strongUnmerge($S$)
Data: $S = \{S_1, S_2, ..., S_m\}$, a set of subclusters
Result: $K \subseteq S$
$K \longleftarrow \emptyset$
for $S_i$ in $S$ do
    if $clusterConstraint(S_i)$ then
        add $S_i$ to $K$
return $K$
```

Weak DIHNOSIR does not require a cluster constraint function. Instead, Weak DIHNOSIR uses a directed graph approach on the subclusters. Two variables denoted $silhouetteThreshold$ and $ssiThreshold$ are required. The default value for $silhouetteThreshold$ is set to 0.75. Subclusters with Silhouette Score lower than this threshold are removed and the algorithm proceeds by creating a directed graph with the remaining subclusters as vertices and edges from S_i to S_j if $SSI(i,j) \geq ssiThreshold$ where

$$SSI(i,j) = \frac{|S_i \cap S_j|}{min(|S_i|, |S_j|)} \in [0,1]. \tag{4}$$

The default value for $ssiThreshold$ is 1, which is equivalent to the relation $S_i \subseteq S_j$. Next, we take the transitive reduction (done by the $transitiveReduction$ method in the pseudocode) of this graph and define a fundamentally merged cluster as a vertex in the transitive reduction with an indegree of at least 2. Then a merged cluster is either a fundamentally merged cluster or a superset of a fundamentally merged cluster. See Algorithm 3 for pseudocode.

Cluster Formation

Once we have the appropriate subclusters $K = \{S_1, S_2, ..., S_k\}$ we create a graph using Simpson's similarity index. [6] We first create a k x k matrix denoted M_S to store the similarity between pairs subclusters. We iterate over all unique combinations (i, j) where $0 \leq i < j \leq n-1$ and we set $M_K[i][j]$ and $M_K[j][i]$ equal to $SSI(i,j)$. See Algorithm 4 for pseudocode.

We then create an undirected graph with edges formed between pairs of subclusters that have at least 0.1 similarity as measured by Simpson's similarity index.

$$G = (V, E);\ V = \{S_1, S_2, ..., S_k\};\ E = \{\{S_i, S_j\} : SSI(i,j) \geq ssiThreshold\}. \tag{5}$$

Note that we use the positive threshold value of $ssiThreshold$ to avoid trivial intersections. See Algorithm 5 for pseudocode.

Algorithm 3: Weak Unmerge

```
weakUnmerge($S$)
Data: $S = \{S_1, S_2, ..., S_m\}$,a set of subclusters; $ssiThreshold$, minimum SSI
      score
Result: $K \subseteq S$
$K$, $Q$, $R$, $E \longleftarrow \emptyset, \emptyset, \emptyset, \emptyset$
for $i : (0 \leq i < m)$ do
    for $j : (0 \leq j < m)$ do
        if $SSI(i, j) \geq ssiThreshold$ then
            if $|S_i| \leq |S_j|$ then
                add $\{i, j\}$ to $E$
            if $|S_i| \geq |S_j|$ then
                add $\{j, i\}$ to $E$
$G \longleftarrow transitiveReduction(directed(S, E))$
for $S_i$ in $S$ do
    if $indegree(S_i) < 2$ then
        add $S_i$ to $Q$
    else
        add $S_i$ to $R$
for $S_i$ in $Q$ do
    for $S_j$ in $R$ do
        if $|Q_i| \geq |R_j|$ then
            if $SSI(i, j) \geq ssiThreshold$ then
                break
    else
        continue
    add $S_i$ to $K$
return $K$
```

Algorithm 4: Similarity Matrix Creation

```
createSimMatrix($K$)
Data: $K = \{S_1, S_2, ..., S_k\}$, a set of subclusters
Result: $M_K \in \mathbb{R}^{k \times k}$, a square matrix of SSI values
$k \longleftarrow size(K)$
$M_K \longleftarrow \mathbf{0}^{k \times k}$
for $S_i$ in $S$ do
    for $S_j$ in $S$ do
        $M_K[i][j] \longleftarrow SSI(i, j)$
return $M_K$
```

Next we find the set of connected components of G: 354

$$G_C = \{G_1, G_2, ..., G_t\} = \{(V_1, E_1), (V_2, E_2), ..., (V_t, E_t)\}. \quad (6)$$

Algorithm 5: Graph Creation

```
createGraphs($M_K$, $ssiThreshold$) Data: $M_K \in \mathbb{R}^{k \times k}$, a square matrix of $SSI$
        values; $ssiThreshold$, minimum SSI score
Result: $G_C = \{G_C^1, ..., G_C^t\} = \{(V_1, E_1), (V_2, E_2), ..., (V_t, E_t)\}$, a set of
        connected components
$m \longleftarrow size(M_K)$
$V \longleftarrow \{1, 2, ..., k\}$
$E \longleftarrow \emptyset$
for $i : (0 \leq i < k)$ do
    for $j : (0 \leq j < k)$ do
        if $(M_K[i][j] \geq ssiThreshold)$ then
            add $\{min(i, j), max(i, j)\}$ to $E$
$G \longleftarrow undirected(V, E)$
$G_C \longleftarrow connectedComponents(G)$
return $G_C$
```

The set of clusters is thus: 355

$$C = \{C_1, C_2, ..., C_t\} \tag{7}$$

where 356

$$C_i = \bigcup_{S_j \in V_i} S_j. \tag{8}$$

See Algorithm 6 for pseudocode. 357

Algorithm 6: Cluster Formation

```
createClusters($G_C$) Data: $G_C = \{G_C^1, ..., G_C^t\} = \{(V_1, E_1), (V_2, E_2), ..., (V_t, E_t)\}$,
        a set of connected components
Result: $C$, a set of clusters
$C \longleftarrow \emptyset$
for $G_C^i$ in $G_C$ do
    if $(size(V_i) > 2)$ then
        $C_i \longleftarrow \bigcup_{S_j \in V_i} S_j$
        add $C_i$ to $C$
return $C$
```

DIHNOSIR

Strong DIHNOSIR combines the five aforementioned algorithms by first creating the data set of subclusters, removing merged clusters using a user-defined cluster constraint function, creating a similarity matrix based on this set of subclusters, creating an undirected graph using this matrix, and finally creating the clusters in the data. See Algorithm 7 for pseudocode.

Algorithm 7: Strong DIHNOSIR

```
strongDIHNOSIR($X$, $n = 100$, $minPts_{min} = 3$, $silhouetteThreshold = 0.75$,
        $ssiThreshold = 1$) Data: $X$, a square distance matrix; $n$, number of
        partitions; $minPts_{min}$, minimum cluster size; $silhouetteThreshold$,
        minimum Silhouette Score; $ssiThreshold$, minimum SSI score
Result: $C$, a set of clusters
$S \longleftarrow createSubclusters(X, n.minPts_{min}, silhouetteThreshold)$
$K \longleftarrow strongUnmerge(S)$
$M_K \longleftarrow createSimMatrix(K)$
$G_C \longleftarrow createGraphs(M_K, ssiThreshold)$
$C \longleftarrow createClusters(G_C)$
return $C$
```

Weak DIHNOSIR combines the five aforementioned algorithms by first creating the data set of subclusters, removing merged clusters using the directed graph approach, creating a similarity matrix based on this set of subclusters, creating an undirected graph using this matrix, and finally creating the clusters in the data. See Algorithm 8 for pseudocode.

Algorithm 8: Weak DIHNOSIR

```
weakDIHNOSIR($X$, $n = 100$, $minPts_{min} = 3$, $silhouetteThreshold = 0.75$,
$ssiThreshold = 1$)
Data: $X$, a square distance matrix; $n$, number of partitions; $minPts_{min}$,
        minimum cluster size; $silhouetteThreshold$, minimum Silhouette Score;
        $ssiThreshold$, minimum SSI score
Result: $C$, a set of clusters
$S \longleftarrow createSubclusters(X, n, minPts_{min}, silhouetteThreshold)$
$K \longleftarrow weakUnmerge(S, ssiThreshold)$
$M_K \longleftarrow createSimMatrix(K)$
$G_C \longleftarrow createGraphs(M_K, ssiThreshold)$
$C \longleftarrow createClusters(G_C)$
return $C$
```

Pruning

Pruning is a common practice in cluster analysis. [2, 11, 25] DIHNOSIR returns clusters of varying size, some of which are relatively small. The user has two options. The first is to simply throw these smaller clusters into the noise cluster. The second is to use the larger clusters as well as the noise cluster as a ground truth for the clustering and use a classification algorithm [10] to redistribute the points within the smaller clusters.

Flip Data

For all iterations we used $n = 250$. For the iterations of Weak DIHNOSIR we used $silhouetteThreshold = 0.6$. For the iteration of Weak DIHNOSIR using the $d(x, y)$ distance metric, we used $minPts_{min} = 4$. All other parameters were set to the default values.

Each datum is represented as a combination of four ordered pairs of ϕ and ψ measures

$$x = ((\phi_1, \psi_1), (\phi_2, \psi_2), (\phi_3, \psi_3), (\phi_4, \psi_4)). \tag{9}$$

Let (ϕ_{1_x}, ψ_{1_x}) represent (ϕ_1, ψ_1) for the datum x in a cluster. Define (ϕ_{i_x}, ψ_{ix}) similarly. We used two distance metrics. The first metric used is defined for a pair of points x, y as

$$d(x, y) = \frac{1}{2}\sum_{i=2}^{3}((1 - \cos(\phi_{i_x} - \phi_{i_y})) + (1 - \cos(\psi_{i_x} - \psi_{i_y}))). \tag{10}$$

The second metric is an L^∞ norm and is defined for a pair of points x, y as

$$d^\infty(x, y) = 2(max(\{max(1 - \cos(\phi_{i_x} - \phi_{i_y}),\ 1 - \cos(\psi_{i_x} - \psi_{i_y})) : \forall i \in [2, 3]\})). \tag{11}$$

The cluster constraint function utilizes a modified angular L^∞ norm to ensure that no two datums within a subcluster have corresponding dihedral measures that differ by more than 150 degrees. Given a subcluster S_i we initially define our norm as

$$K^\infty(S_i) = max(\{max(|\phi_{j_x} - \phi_{j_y}|, |\psi_{j_x} - \psi_{j_y}|) : \forall j \in \{2, 3\};\ \forall x, y \in S_i\}). \tag{12}$$

However, when dealing with Ramachandran plots we must realize that the difference between two angles can only lie between 0 and 180 degrees as the data wraps around itself. Any difference d that is greater than 180 degrees must be rotated by 360 degrees by using $360 - d$ as the difference. Thus our norm is equal to

$$D^{\infty}(S_i) = min(K^{\infty}(S_i),\ 360 - K^{\infty}(S_i)). \tag{13}$$

Our cluster constraint function is thus defined as

$$clusterConstraint(S_i) = \begin{cases} True & if\ D^{\infty}(S_i) \leq 150 \\ False & else \end{cases}. \tag{14}$$

Antibody Data

For the iterations of Weak DIHNOSIR we used $silhoutteThreshold = 0.7825$. All other parameters were set to the default values.

Each datum is an eleven residue element represented as a combination of eleven ordered pairs of ϕ and ψ measures

$$x = ((\phi_1, \psi_1), (\phi_2, \psi_2), ..., (\phi_{11}, \psi_{11})). \tag{15}$$

The antibody data was clustered twice using two distance metrics. The first metric is nearly identical to that seen in Equation 10 except it is modified for the 11 residue elements. It is defined for a pair of points x, y as

$$d(x, y) = \frac{1}{11} \sum_{i=1}^{11} ((1 - \cos(\phi_{i_x} - \phi_{i_y})) + (1 - \cos(\psi_{i_x} - \psi_{i_y}))). \tag{16}$$

The second metric is also similar to Equation 11 and is defined for a pair of points x, y as

$$d^{\infty}(x, y) = 2(max(\{max(1 - \cos(\phi_{i_x} - \phi_{i_y}),\ 1 - \cos(\psi_{i_x} - \psi_{i_y})) : \forall i \in [1, 11]\})). \tag{17}$$

The cluster constraint function used is nearly identical to that seen in Equation 14 except for a slight redefinition of the norm K^{∞} seen in Equation 12 for the 11 residue elements: 405

$$K^{\infty}(S_i) = max(\{max(|\phi_{j_x} - \phi_{j_y}|, |\psi_{j_x} - \psi_{j_y}|) : \forall j \in [1, 11]\ \forall x, y \in S_i\}). \tag{18}$$

The cluster constraint function is thus defined as 406

$$clusterConstraint(S_i) = \begin{cases} True & if\ D^{\infty}(S_i) \leq 150 \\ False & else \end{cases}. \tag{19}$$

References

1. Ali T, Asghar S, Sajid NA. Critical analysis of DBSCAN variations. 2010 International Conference on Information and Emerging Technologies. 2010. doi:10.1109/ICIET.2010.5625720

2. Aroche-Villarruel AA, Martínez-Trinidad JF, Carrasco-Ochoa JA, Pérez-Suárez A. A Different Approach for Pruning Micro-clusters in Data Stream Clustering. Pattern Recognition. MCPR 2015. Lecture Notes in Computer Science, vol 9116. Springer, Cham. 2015. doi: 10.1007/978-3-319-19264-2_4

3. Hennig C, Liao TF. How to find an appropriate clustering for mixed type variables with application to socioeconomic stratification (with discussion). Journal of the Royal Statistical Science, Series C (Applied Statistics) 62 (2013) 309–369. doi: 10.1111/j.1467-9876.2012.01066.x

4. Dockhorn A, Braune C, Kruse R. An Alternating Optimization Approach Based on Hierarchical Adaptations of DBSCAN. 2015 IEEE Symposium Series on Computational Intelligence. 2015. doi: 10.1109/SSCI.2015.113

5. Hagberg AA, Schult DA, Swart PJ. Exploring Network Structure, Dynamics, and Function using NetworkX. Proceedings of the 7th Python in Science conference (SciPy 2008).:11–5. Available from: http://conference.scipy.org/proceedings/scipy2008/paper_ 2/.

6. Ho BK, Brasseur R. The Ramachandran plots of glycine and pre-proline. BMC Structural Biology. BMC Structural Biology. 2005;5:14. doi:10.1186/1472-6807-5-14

7. Ho BK, Thomas A, Brasseur R. Revisiting the Ramachandran plot: Hard-sphere repulsion, electrostatics, and H-bonding in the α-helix. Protein Science. 2009Jan;12(11):2508–22. doi: 10.1110/ps.03235203, PMID: 14573863

8. Bezdek JC, Pal NR. Some new indexes of cluster validity. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 28 (3) (1998) 301–315. doi: 10.1109/3477.678624

9. Kormas KA, Meziti A, Mente E, Frentzos A. Dietary differences are reflected on the gut prokaryotic community structure of wild and commercially reared sea bream (Sparus aurata). MicrobiologyOpen. 2014;3(5):718–28. doi:10.1002/mbo3.202

10. Kotsiantis SB. Supervised machine learning: a review of classification techniques. Informatica 31 (3) (2007) 249-268. Available from: https://dl.acm.org/citation.cfm?id=1566770.1566773.

11. Kpotufe S, von Luxburg U. Pruning nearest neighbor cluster trees; 2011. arXiv:1105.0540 [stat.ML]. Available from: https://arxiv.org/abs/1105.0540.

12. Kriegel HP; Kröger P; Sander J; Zimek A. Density-based clustering. WIREs Data Mining and Knowledge Discovery 231-240., 2011. doi: 10.1002/widm.30

13. Liu Y, Li Z, Xiong H, Gao X, Wu J. Understanding of Internal Clustering Validation Measures. 2010 IEEE International Conference on Data Mining. 2010. doi: 10.1109/ICDM.2010.35

14. Maccallum PH, Poet R, Milner-White EJ. Coulombic interactions between partially charged main-chain atoms not hydrogen-bonded to each other influence the conformations of α-helices and antiparallel β-sheet. A new method for analysing the forces between hydrogen bonding groups in proteins includes all the Coulombic interactions. Journal of Molecular Biology. 1995;248(2):361–73. doi:

Coulombic interactions between partially charged main-chain atoms not hydrogen-bonded, PMID: 7739046

15. Halkidi M, Batistakis Y, Vazirgianni M. On clustering validation techniques. Journal of Intelligent Information Systems 17 (2-3) (2001) 107–145. doi: 10.1023/A:1012801612483

16. Milner-White EJ. Situations of gamma-turns in proteins. Journal of Molecular Biology. 1990;216(2):385–97. doi: 10.1016/S0022-2836(05)80329-8, PMID: 2254936

17. Pedregosa et al. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011. Available from: https://dl.acm.org/citation.cfm?id=2078195.

18. Ramachandran G, Sasisekharan V. Conformation of Polypeptides and Proteins. Advances in Protein Chemistry Volume 23. 1968;:283–437. doi: 10.1016/S0065-3233(08)60402-7, PMID: 4882249

19. de Amorim RC, Mirkin B. Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering. Pattern Recognition 45 1061-1075., 2012. doi: 10.1016/j.patcog.2011.08.012

20. Rousseeuw PJ. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics. 1987;20:53–65. doi: 10.1016/0377-0427(87)90125-7

21. Sander J, Ester M, Kriegel HP. et al. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. Data Mining and Knowledge Discovery (1998) 2: 169. doi: 10.1023/A:100974521

22. Strehl A, Joydeep G. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3: 583–617; 2002. doi: 10.1162/153244303321897735

23. Voet D, Voet JG. Biochemistry. Hoboken, NJ: Wiley; 2010.

24. Wilson RJ. Introduction to graph theory. Harlow: Prentice Hall; 2010.

25. Zhang J-S, Leung Y-W. Robust clustering by pruning outliers. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics). 2003;33(6):983–99. doi: 10.1109/TSMCB.2003.816993