Computational Biology and Chemistry 59 (2015) 1–7

Contents lists available at ScienceDirect

Computational Biology and Chemistry

journal homepage: www.elsevier.com/locate/compbiolchem

Research Article

Machine Learnable Fold Space Representation based on Residue Cluster Classes

Ricardo Corral-Corral[a], Edgar Chavez[b], Gabriel Del Rio[a,*]

[a] Department of Biochemistry and Structural Biology, Instituto de Fisiologa Celular, Universidad Nacional Autónoma de México, México D. F., México
[b] Centro de Investigación Científica y de Educación Superior de Ensenada, México

ARTICLE INFO

Article history:
Received 12 November 2014
Received in revised form 17 July 2015
Accepted 25 July 2015
Available online 30 July 2015

ABSTRACT

Motivation: Protein fold space is a conceptual framework where all possible protein folds exist and ideas about protein structure, function and evolution may be analyzed. Classification of protein folds in this space is commonly achieved by using similarity indexes and/or machine learning approaches, each with different limitations.
Results: We propose a method for constructing a compact vector space model of protein fold space by representing each protein structure by its residues local contacts. We developed an efficient method to statistically test for the separability of points in a space and showed that our protein fold space representation is learnable by any machine-learning algorithm.
Availability: An API is freely available at https://code.google.com/p/pyrcc/.

1. Introduction

All possible protein folds are assumed to occupy an abstract space referred to as fold space. This fold space has become a conceptual framework to unify ideas about protein structures with protein function and protein evolution (Cheng and Brooks, 2013). For instance, it is debated whether this space is discrete or continuous (Kolodny et al., 2006; Skolnick et al., 2009; Sadreyev et al., 2009). Relevant to our study is the common use of protein similarity measures (*e.g.*, root mean square deviation or RMSD) aimed to infer their proximities in this space (Minary and Levitt, 2008). In this case, the inference derived from such measurements assumes that the proximity of protein folds is the only relevant property to explain protein fold evolution and function.

Instead of focusing only on the proximity of protein folds, vector space models have been used to expand the protein fold space representation. In this space, each protein structure is represented in a fixed dimension space (*e.g.*, euclidean space) by a point (position vector); adjusting the positions of these vectors by approximating their relative distances to protein similarity measures may derive these position vectors. For example, using sequential structure alignment program scores between each pair of structures as a protein similarity measure (Orengo and Taylor, 1996), followed by multi dimensional scaling allowed the assignment of positions vectors representing each protein structure (Michie et al., 1996).

Using DALI as similarity measure, Holm (Holm and Sander, 1998) showed two-dimensional projections to explore protein neighbors in fold space. DALI has also been used as similarity measure to visualize class distribution and fold usages between two bacterial species (Hou et al., 2003) and to explore protein function assignment based on position on this fold space representation (Hou et al., 2005).

Fold space constructions based on protein structure similarities have two important limitations. First, the time needed to run a structural comparison for each pair of structures is restrictive: 25000 central processing unit hours were needed for calculating similarities between 1898 protein structures (Hou et al., 2005). Second, the position of each point depends heavily on the set of structures being analyzed, and in such case the inclusion of a single new structure can displace all previously assigned positions; thus, there are as many fold space representations as different protein structures data sets, even adopting a unique similarity score.

An alternative fold space representation may be built by assigning the position of the vector considering only the structural features of the protein it represents. In this way, a new structure can find its location in this fold space without altering the existing ones. One implementation of a vector space model is FragBag (Budowski-Tal et al., 2010), which represents protein structures in 400 dimensions; here, each component in the vector representing a protein fold corresponds to the number of occurrences of a

* Corresponding author. Tel.: +44 000 0000000; fax: +44 000 0000000
E-mail address: gdelrio@ifc.unam.mx (G. Del Rio).

http://dx.doi.org/10.1016/j.compbiolchem.2015.07.010
1476-9271/

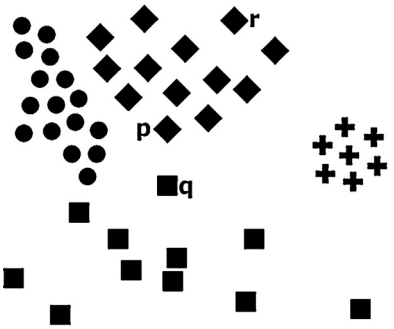

Fig. 1. Proximity in space does not always imply membership to a class. The figure illustrates cases where proximity in a space does not imply membership to a class. Cross points define a class where distances between every pair of its members should be less than the distance to any point belonging to other class, an assumption made when using similarity scores. An exception to this assumption is represented by the rhombus class members p and r, which distance is larger than for member class p to a square class member q. This is true for any distribution within the classes. To show this, circle class has a dense and regular distribution, rhombus class is regular and sparse, while square is very sparse and irregular. In any case, each class is clearly segregated. Please note that the different fold classes may be learned using machine-learning classifiers, but here we illustrate that using similarity measures as the only criterion to distinguish class membership may induce errors.

particular contiguous protein sequence fragment. Another approach uses knot invariants as values in each component for vector points in 30 dimensions (Rogen and Fain, 2003). In these cases, it is assumed that proximal protein structures should belong to the same structural class, assumption that is not necessarily correct as we will argue below.

Once a protein fold space construction is chosen, a metric distance induced by the space can be used as a measure of similarity and it is expected to be in agreement with direct structural measures (such as RMSD, GDT, TMscore, etc), but overcoming the problems noted above about these scores not being a metric (Sippl, 2008).

In such space, a given set of protein structures that are considered to have the same fold may be close in this space representation. Yet, it may occur that some proteins with different folds may be closer than proteins with the same fold (see Figure 1). Thus, confusion may be induced at distinguishing class membership in this fold space when only similarity measures are considered.

To address this problem, the boundaries between proteins with different folds may be obtained using empirical data and machine learning algorithms, which naturally segregate protein structures sharing common features. In these models it is also possible to evaluate the separability of this space independently of any machine-learning algorithm using a statistical test (Zighed et al., 2002). Therefore, it is possible to generate a protein fold space representation independent on the protein similarity measure used and to test for the separability of this space independently of any particular classification algorithm. This protein fold space may then be used to analyze protein structure-function relation and protein evolution without the limitations previously noted of current protein fold space construction approaches.

In this work, we propose a compact (low dimensional) fold space representation based on Residue Cluster Classes (RCCs), a Sperner family that includes all sets of residues in simultaneous contact. We also present an efficient computational method useful to test for the separability of this fold space representation. As a proof of principle, we analyzed the CATH classification and automatically detected conflicts in CATH. Furthermore, we show that our method improves state of the art protein structure neighbor retrieval methods. To facilitate the construction of protein folds represented by RCCs, we present an API available at https://code.google.com/p/pyrcc/.

2. Materials and Methods

2.1. Datasets

2.1.1. CATH datasets

CATHALL1 set includes all domains in CATH release v3.5 that were parsable with our API and consists of 168964 domains. CATHALL2 set includes all domains in CATH release v4.0 and contains 235858 domains. CATCHOP was obtained from a random sample of CATHALL1 considering only six domains per topology; topologies with less than six members were excluded rendering a total of 5220 domains (see supplemental Table S1 for a complete list).

2.1.2. SCOP datasets

The SCOP30 dataset was provided by the authors of ContactLib (Xuefeng Cui et al., 2014) and contains 3295 SCOP domains. SCOP30 contains 2639, 3232 and 3290 neighbours at SAS levels 20,35 and 50. Yet, only 2049, 2620 and 2722 domains have at least one neighbor in SAS 20, 35 and 50, respectively. The SCOPtrain1 is a random sample of 136300 SCOP 1.75B. SCOPtrain2 contains all 203025 domains from SCOP release v2.5. The SCOPtrainAUC includes 109310 SCOPtrain domains absent in SAS50 group, and only belonging to a class in SCOP30. If a class (at any level) contains more than 2000 domains, 2000 domains were chosen randomly to represent that class.

2.2. Construction of Residue Cluster Classes

2.2.1. Definitions

Residue Neighbourhood ($N_\epsilon(r)$). Let P be a protein with residues $R = r_1, r_2, \ldots, r_n$. The system τ_{prim} is defined as:

$$\tau_{prim} = \{\{r_i, r_j\} : \text{there exists a bond between } r_i \text{ and } r_j\}$$

Given a metric $d : R \times R \to [0, \infty)$ and a cut-off distance ϵ, the neighbourhood $N_\epsilon(r)$ of a residue r is given by:

$$N_\epsilon(r) = \{x \in R : \exists a \in A(x), b \in A(r); d(a, b) \leq \epsilon\}$$

Where $A(r)$ is the set of non-hydrogen atoms of residue r. Thus, $N_\epsilon(r)$ is the set of all residues near r, *i.e.*, they are at no more than a distance ϵ from r.

Residue Cluster (RC). A residue cluster on P is a subset $A \subseteq 2^R$ such that $A \subseteq N_\epsilon(a)$ for all $a \in A$. If $|A| = k$, then A is a k-Residue Cluster, ${}_kRC$

Residue Cluster Class (RCC). A class over a RC is defined by the primary structure on τ_{prim}. A pair of residues r_i, r_j are contiguous if $\{r_i, r_j\} \in \tau_{prim}$ and a set $\bar{s_L}$ of L residues form a segment in τ_{prim} if it contains $(L-1)$ contiguous residues. By convention, $\bar{s_1} = \{r\}$ for any $r \in R$.

Let be γ the family of all segments in τ_{prim}. C is a set cover of a k-Residue Cluster ${}_kRC$ if

$$C = \{\bar{s_{L_\alpha}} : \alpha \in A, \bar{s_{L_\alpha}} \in \gamma\}$$

such that

$${}_kRC = \bigcup\nolimits_{\alpha \in A} \bar{s_{L_\alpha}}$$

and

$$c_i \cap c_j = \emptyset, \forall c_i, c_j \in C$$

Therefore, $C = \bar{s_{L_1}}, \bar{s_{L_2}}, \ldots, \bar{s_{L_A}}$ is a covering if

$$\sum_{i=1}^{A} L_i = k$$

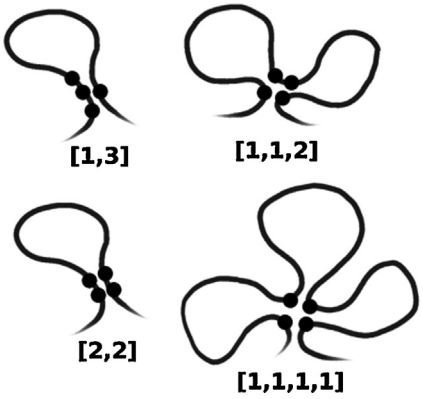

Fig. 2. Class names for residue clusters of same size. The curve represents a protein backbone, and black circles represent residues in a cluster being considered. Four distinct classes ([1,3], [1,1,2], [2,2] and [1,1,1,1]) arising from $_4$RC's are shown. The class name is defined by the number of residues found in each segment that defines the RCC (*e.g.*, $_4$RCC). For convention, numbers are written in non-decreasing order.

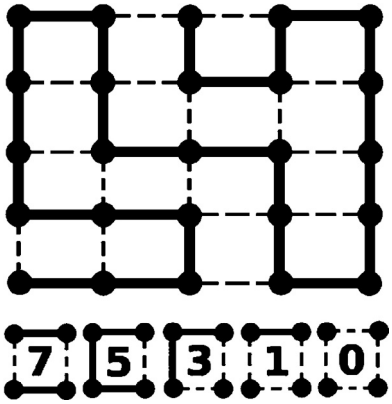

Fig. 3. Schematic construction of an RCC vector. The figure represents a protein three-dimensional structure in a lattice, where each residue is represented by a circle and the dotted lines represent residue contacts (tertiary contacts). A self-avoiding walk (SAW) showed as a straight line visiting all points of a grid represents the protein backbone. Considering four points in any square as a cluster (an RCC of size 4), there can be five classes of clusters showed at the bottom. Numbers in each cluster class are the number of occurrences of that cluster formed by the SAW. This same idea can be extended to real proteins to quantify clusters of size 3, 4, 5 and 6 to generate the final vector representation of a protein.

and can be uniquely identified by the vector class $\vec{C} = (L_1, L_2, \ldots, L_A)$ with $L_i \leq L_j$ for any i, j such that $i \leq j$.

Lets define the k-Residue Cluster Class $_kRC_{\vec{C}}$ as the set of all $_kRC$ of class $\vec{C}$. The residue cluster class system $\mathfrak{R}$ is a Sperner family (Lubell, 1996). It is defined as the set of all residue clusters for all k, $\vec{C}$ such that none is subset of some other, that is:

$$\mathfrak{R} = \{W \in R : \exists X \in_k RC_{\vec{c}}, X \subseteq W\}$$

Let $M(\vec{c}_i)$ be the number of occurrences of a residue cluster class $\vec{c}_i$ in a system $\mathfrak{R}$.

The vector point $\vec{\mathfrak{R}}$ of $\mathfrak{R}$ is defined by

$$\vec{\mathfrak{R}} = \sum_{i \in I} M(\vec{c}_i) v_i$$

with $v_i \in B$, where $B = \{v_1, v_2, \ldots, v_{|I|}\}$ is the standard orthonormal basis for $\mathbb{R}^{|I|}$ and I is an index set over the set of cluster classes considered.

Thus, an RCC construction is illustrated in Figure 2 where different classes are assigned to a $_kRC_{\vec{C}}$ according to sequence contiguity.

2.3. Construction of vector points for domain structures

Clusters were computed using graph theoretical algorithms. First, a graph G is constructed with a $V(G)$ vertex set that represents the residues set for a given protein, and $E(G)$ edges set is the set of all residue clusters of size two. This graph is equivalent to a residue contact map and is constructed with the same program used in (Cusack et al., 2007): any two residues are paired if they share at least one atom pair at 5Å or less in the three dimensional representation of a folded protein. As any clique in G (*i.e.* a complete subgraph) is consistent with our cluster definition, all maximal cliques (cliques that are not a subgraph of any other clique) are considered as clusters, and classes are assigned directly from sequence contiguity. Maximal cliques were obtained using the algorithm previously described (Tomita et al., 2006) as implemented in NetworkX (Hagberg et al., 2008).

Thus, for any given protein there are 3 $_3$RCC ([1,1,1], [1,2], [3]; see Figure 2 for the nomenclature used to refer to these RCCs), 5 $_4$RCC ([1,1,1,1], [1,1,2], [2,2], [1,3], [4]), 7 $_5$RCC ([1,1,1,1,1], [1,1,1,2], [1,2,2], [1,1,3], [2,3], [1,4], [5]) and 11 $_6$RCC ([1,1,1,1,1,1], [1,1,1,1,2], [1,1,2,2], [2,2,2], [1,1,1,3], [1,2,3], [3,3], [1,1,4], [2,4], [1,5], [6]); thus, there are 26 RCCs for any given protein. To represent these in a vector, we used the frequency of occurrence of each $_k$RCC as the value for each dimension of this vector. Figure 3 illustrates an example of the construction of a vector using a self-avoiding walk over a lattice grid. An API was made for cluster calculations from protein structures and can be accessed via https://code.google.com/p/pyrcc/.

2.4. Calculation of the Cut Edge Weight statistic

The Cut Edge Weight statistic (Zighed et al., 2002) was used to test the hypothesis that a given class distribution of vector points (*i.e.*, protein domains in CATH) is random. The construction of the statistic is limited by the construction of the relative neighbourhood graph (RNG) of the points. A naive algorithm takes $O(n^3)$ and lower bounds have been reported only for particular cases in low dimensions. Here we include a demonstration that a RNG is always a subgraph of the Half-Space Proximal (HSP) graph (see below).

To obtain the RNG, the HSP graph H of the set of 26-dimension vectors was obtained (see below for HSP construction procedure). Then, for each edge (a, b) in H we remove the edge if the following condition was satisfied:

$$\max\{d(a, c), d(b, c) < d(a, b)\}$$

where $d(i, j)$ is the euclidean distance between points i and j.

This condition was applied for any c in H. After this procedure, the resulting graph was the RNG used for statistics computation as described in Zighed et al. (2002).

A detailed description of HSP and a proof for containing RNG is described below. The source code to compute this statistical test can be obtained from the authors upon request.

2.5. Relative Neighborhood Graph as a subgraph of the Half-Space Proximal graph

The Half Space Proximal (HSP) is a local test for building a geometric spanner of bounded dilation over a set of points in the space, it was introduced in Chavez et al. (2006). Assume there is a set V of points loaded with a metric. To fix ideas think in the plane and the euclidean distance between points, in our case the set of points are 26 dimensional vectors representing protein structures, although these could be any set of objects in an abstract metric space. All the construction will be based on the distance metric. For each point u in V we compute its HSP neighbours as follows.

For a point u in V take its nearest element v and add an edge from u to v; remove all the elements that are closer to v than to u. The

region of points closer to v than to u is called the forbidden region from the point u with respect to v. From the remaining points, those points not in the forbidden region, take the nearest point to u and repeat until all points in V belong to some forbidden region. In the end, we will have a directed graph with vertex set V and the edges found with the above procedure. In this paper we are interested in the HSP as a super graph of the Relative Neighbourhood Graph (RNG), faster to build, as described below.

For the same set V two points u, v will share an edge in the RNG if there is not a point $z \in V$ such that z is in the intersection of the circles centered in u,v, respectively, with radius $|uv|$. Then, RNG and the HSP are related and we will show that RNG $\subseteq$ HSP.

Lemma 1. *If there is not an edge from the point u to v, then there exist a point z such that it connects to u and z is in the intersection of the two circles centered at u and v with radius* $|uv|$.

Proof. If there is not an edge from the point u to v, then, v is in a forbidden region of u and some point z. The point z connects to u and is closer to u than from v, so, z is in the circle centered at u with radius $|uv|$. As v is in the forbidden region of u with respect to z, then v is closer to z than to u, so, z is in the circle centered at v with radius $|uv|$.

One way to characterize the RNG is by observing that two points p and q will share an edge whenever there is not a third point r that is closer to both p and q than they are to each other.

Lemma 2. *If an edge is in the RNG then, it is in the HSP.*

Proof. Lets suppose that two points p and q are not connected by the HSP, then, there exist a point z that is in the intersection of the two circles centered at p and q with radius $|pq|$, then the two point p, q are not connected in the RNG.

A brute force approach to build the RNG built over a set of n points require checking $O(n^2)$ pairs to see if they share and edge; furthermore, each pair requires $O(n)$ points to be tested for inclusion in the intersection. This yields a total complexity of $O(n^3)$ operations. On the other hand, all the HSP neighbours of a point u can be computed in an amortized $O(n)$ operations. The total complexity of building the HSP is then $O(n^2)$. The total number of HSP neighbours of any point u is bounded, and depends only on the dimension of the space, and it is independent of the size of the point set. This implies that the number of edges in the HSP is only linear on n.

Using the above observations to test if an edge is in the RNG graph, we only need to test the linear number of edges of the HSP. Each test cost $O(n)$ operations, and consist in checking if the intersection is empty. Even with brute force, the total time for building the RNG would be $O(n^2)$ instead of $O(n^3)$ using other construction algorithms.

Figure 4 illustrates the inclusion of the RNG in the HSP. Dotted edges in the figure are those to be removed from the HSP. To exemplify the improvement on computing time by our approach, Figure 5 shows execution times to calculate RNG, HSP and RNG from HSP for two randomly generated datasets.

2.6. Learner details

A learner including an ensemble of 250 extremely randomized trees (ERT) was trained for cross-validation tests, predicting the Class, Architecture and Topology in CATH classification, and Class, Topology and Superfamily in SCOP. Bootstrap samples were not used for building the trees, and a minimum of 3 nodes where required to remain after each split. In each tree, Gini index was used for deciding attributes for splitting.

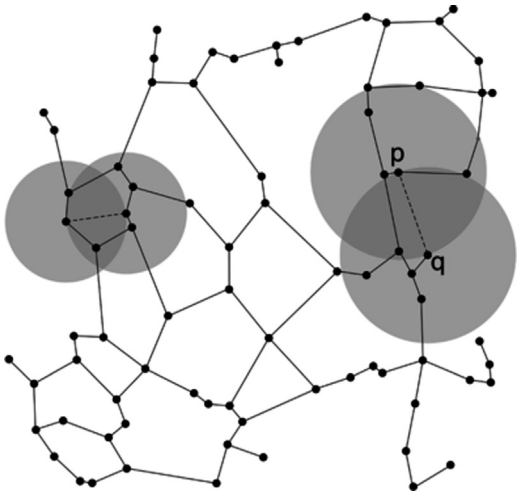

Fig. 4. Schematic representation about the differences of RNG and HSP. Light gray circles are traced with radius equals to the distance $|pq|$ between any pair of points in space (p and q in the figure); the dark gray regions (also known as lunes) test for the presence of any point in space within those light gray circles. If there exist a point z that is in the intersection of the two circles centered at p and q with radius $|pq|$, then the two points p, q are not connected in the RNG.

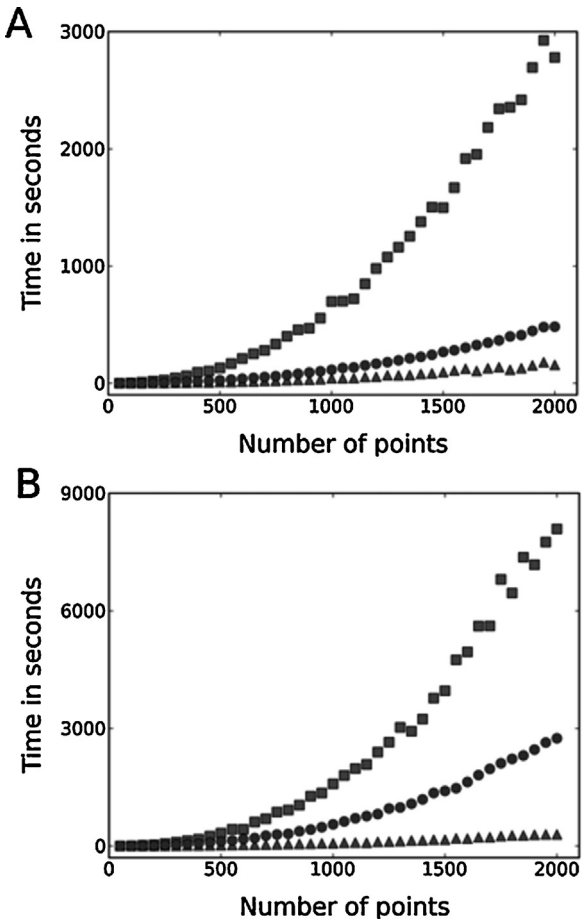

Fig. 5. Execution times to obtain RNG from random datasets. The points in the plot represent the computing time to generate a RNG (squares), HSP (triangles) and RNG from HSP (circles) over random point sets of different sizes in 2 dimensions (A) and 50 dimensions (B) (see Methods). RNG from HSP is clearly more efficient than brute force RNG, although this relative efficiency gains decreases as the number of dimensions increases, since all edges in HSP are checked for inclusion of RNG, and the number of edges in HSP tends to be larger on higher dimensions. These data sets were obtained using the random number generator model (without a seed number) implemented in the random standard package in Python 2.7.

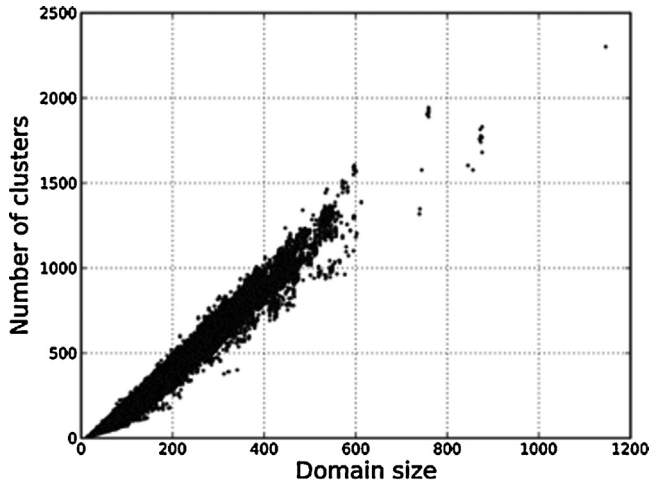

Fig. 6. Relationship between domain size and number of clusters in each domain. The number of residues found in every protein domain of the CATCHOP dataset is plotted against the total number of RCCs found in the corresponding domain. Note the scarcity of data in domains larger that 600 residues in CATH.

Same setup but with 300 trees was used for AUROC analysis. These classifiers were implemented using sklearn (Pedregosa et al., 2011).

To learn the structural classification from CATH, we used the CATHALL1 and CATHAll2 sets; for SCOP, we used the SCOPtrain1 and SCOPTtrain2 sets.

3. Results

3.1. Defining vector dimensionality

The CATHALL1 set (see methods) was used to identify RCCs. Figure 6 shows a linear relationship between protein size and number of RCCs; RCCs larger than 10 residues were not observed. To define the largest RCC to be used for protein structure representation, we identified the frequency of occurrence of the largest RCCs in the CATHALL1 set (see supplemental Figure S1). The fraction of domains having RCCs of size six and seven account for the 91.61% and 7.92% respectively; furthermore, they are represented in 94.51% and 48.13% of topologies. Thus, RCCs larger than 6 residue were rare and found in only very few topologies. For the rest of our study we used RCCs of size 3 to 6 that include a total of 26 different RCCs, which were used to build our vector space model.

3.2. Separability of fold space

Next, we tested for the separability of our vector space; while learnability is addressed in Vapnik-Chervonenkis learning theory, this is learner dependent and do not evaluate the practical separability of a particular dataset (Holden and Niranjan, 1995). To quantify the separability (and thus, learnability) of our fold space representation, we used the Cut Edge Weight statistic (Zighed et al., 2002). In this procedure, the null hypothesis is that data is randomly distributed and cannot be classified. The implementation of this test is however computationally expensive, and thus we used the CATCHOP set (see Methods). Using this test in CATCHOP set, we observed that the null hypothesis is false ($p = 0$) and consequently domains belonging to different topologies are separable.

3.3. Learning of fold space for structural classification

To use the separability of our fold space representation, we trained an extremely random forest (ERF) classifier to automatically assign classifications by class, architecture and topology according to the CATH definition. We performed 10-fold stratified cross-validation test to assess the ability of our ERF classifier to learn general rules from the CATHALL1 set (see Methods section); in this case, 10% (1/10) of the CATHALL1 domains were randomly chosen as part of the training samples (the rest was part of the test set) and this random procedure was performed 10 times; please note that in this way we do not generate specific training sets avoiding bias in the test. We also trained our ERF classifier over the entire data set. Performance predictions for these two approaches are reported in Table 1. We repeated this procedure for a new release of CATH included in CATHALL2 set and the results are reported in supplemental Table S2.

Table 1
CATH Cross-Validation scores

Classification level	Cross-Validation accuracy	standard-error	full train accuracy
C	0.9693	0.0014	0.9987
CA	0.8830	0.0019	0.9967
CAT	0.8494	0.0022	0.9984

Performance on 10-fold cross-validation over CATH domains. Mean of accuracies for each of the 10 iterations and corresponding standard errors are shown. Full train accuracy was obtained by training and testing on the same full CATHALL1 dataset.

Table 2
Confusion matrix of Classes over CATH (v3.5) domains.

True/Predicted	mainly-α	mainly-β	$\alpha\beta$	few secondary structure
mainly-α	36936	0	23	0
mainly-β	0	39944	38	1
$\alpha\beta$	42	74	89310	0
fss	19	7	17	2553

In each row (i) and column (j), the number of domains belonging to class i, predicted as j are shown. A perfect prediction has only numbers in the main diagonal. Note that no discrepancies exist between mainly-α and mainly-β.

The discrepancies between the ERF classifier using the full data in CATHALL1 as the training set and the CATH classification are summarized in the Confusion matrix presented in Table 2. Some examples of these discrepancies are presented in Figure 7 (see supplemental Tables S3 for a complete list).

Classification performance was evaluated in the same way over SCOP domains (See Table 3 for results obtained with SCOPtrain1 set

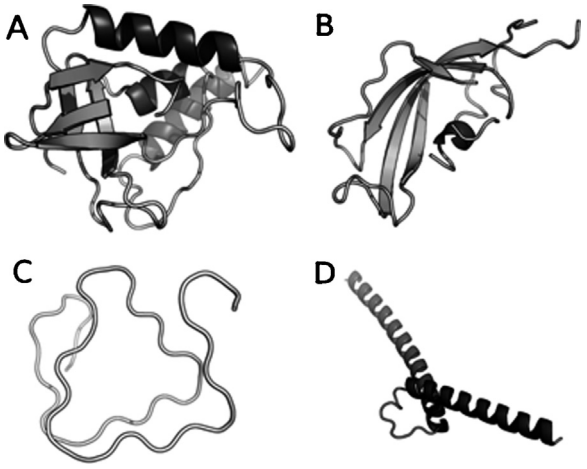

Fig. 7. Examples of domain classification discrepancies between ERT and CATH. A) 1a3uA00: predicted $\alpha\beta$, in CATH mainly-β; B) 1w26A03: predicted mainly-β, in CATH $\alpha\beta$, C) 1h6wA02: predicted few secondary structure, in CATH mainly-β, D) 1nkpD00: predicted mainly-α, in CATH few secondary structure.

Table 3
SCOP Cross Validation scores

Classification level	accuracy
Class	0.9243
Topology	0.8632
Superfamily	0.8781
Family	0.9097

Performance on 10-fold cross-validation over SCOPtrain1 domains. Final accuracies are the mean of accuracies for each of the 10 iterations.

and Supplemental Table S4 for results obtained with SCOPtrain2 set).

3.4. Fold space visualization

To visually inspect fold space structure, we used an unsupervised dimensionality reduction approach; this dimensionality reduction was performed by Laplacian-Eigenmaps (Belkin and Niyogi, 2003) over the CATCHOP set (see Methods). This method has been shown to give a better approximation of the vecinity of each point by approximating the Laplace Beltrami operator of the original space on the reduced space (Belkin and Niyogi, 2008). Projection into two dimensions is shown in Figure 8. In this representation, mainly-α and mainly-β domains are observed on opposite directions, while $\alpha\beta$ domains are found in between these groups.

3.5. Structural neighbour retrieval

ContactLib has been reported to achieve the best performance in finding protein structural neighbors when compared against the best alignment-free method FragBag (Xuefeng Cui et al., 2014); FragBag on the other hand was reported to improve on 6 state of the art structural aligners, including SGM, PRIDE, BLAST, STRUCTAL, CE and SSM. FragBag was tested in a subset of SCOP 1.75B domains, referred to as SCOP30; three SAS thresholds (2.0, 3.5 and 5 Å) identify three types of neighborhoods; SAS20 is the most sensitive and SAS50 the most relaxed. Agreement between SAS criterion and SCOP superfamily assignment is low (see supplementntary Table S5), thus only the domains within each SAS neighborhood belonging to the same superfamily as defined in SCOP are considered true neighbours.

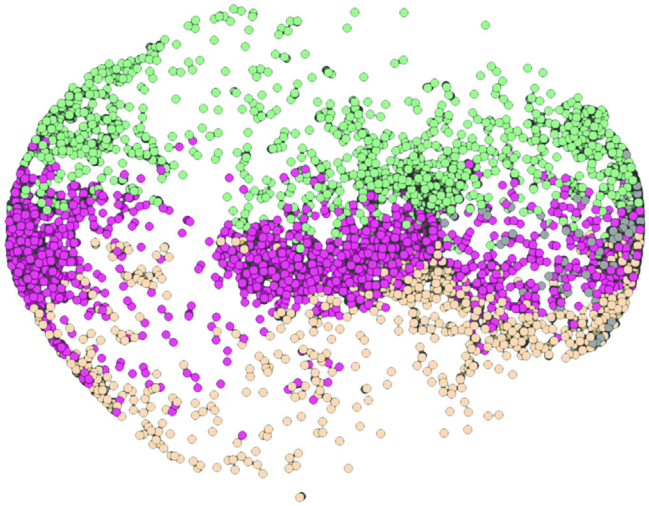

Fig. 8. Fold space visualization by dimensionality reduction. Domains in CATCHOP represented by Spectral-Embedding dimensionality reduction of the original 26-dimensional space. Different colours of the dots correspond to CATH class assignments: green dots for mainly-α, orange dots for mainly-β, purple dots for $\alpha\beta$ and gray dots for few secondary structure class.

Table 4
Mean AUROC

	Mean AUROC		Fraction above
SAS level	RCC	ContactLib	75%
20	0.950	0.956	0.958
35	0.940	0.918	0.939
50	0.935	0.906	0.929

Summary of mean AUROC values for the task of neighbour retrieval. Direct comparison between mean AUROC obtained by our method (RCC) and ContactLib. Last column also shows the percentage of query structures for which AUROC was higher than 75%.

We calculated the Area Under the Receiver Operator Characteristic curve (AUROC) to test for the capacity of our fold space representation to identify true neighbors according to SAS and SCOP criteria; this approach was chosen to summarize the results obtained for all the SCOP domains analyzed. An AUROC score of 1.0 is a perfect score, and 0.5 the score expected from a random method. We calculated the AUROC for each structure in SCOP30 (3295 domains; see Methods) using both unsupervised and supervised approaches. In the unsupervised approach, SCOP30 structures were ordered from a given query structure using the euclidean distance; this ordered list was compared with the true neighbors provided by SAS and SCOP criterion to quantify the corresponding AUROC. Mean AUROCs for this experiment were 0.87, 0.85, 0.84 for SAS 20, 35 and 50 respectively. For comparison, note that the best score obtained in this experiment by FragBag is 0.747 for SAS20.

Table 4.

In the supervised analysis, AUROCs where obtained by training a random forest classifier with a subset of SCOP excluding structures in SCOP30 (see Methods). Then, for a query structure, its class is predicted and the ordering of the rest of domains is determined by the log-probabilities of belonging to this predicted class.

Our mean AUROCs for SAS 20, 35 and 50 are 0.950, 0.940 and 0.935, while ContactLib achieves 0.956, 0.918 and 0.906. ContactLib reported 75% of AUROC scores > 0.936, while our minimum AUROC score for 75% of queries is 0.958. For SAS35 and SAS50, 75% of AUROC scores are above 0.939 and 0.929 respectively. Our median AUROCs for any SAS level is > 0.99.

4. Discussion

Here we introduce a protein structure representation using a Sperner family defined by clusters of protein's residues that are in close proximity in the three dimensional space. In Figure 6 it is shown the linear relationship between protein size and total number of such clusters that may be associated to the near constant density of proteins. The small deviation in the number of clusters for small proteins is also consistent with the observation that small proteins tend to have higher average densities (Fischer et al., 2004).

We found that proteins usually harbour residue clusters with no more than six residues, independently of secondary structure composition. Thus, residue clusters of size 3 and up to 6 were identified according to the contiguity of residues in sequence (see Figure 2). This Residue Cluster Class System is used to construct a vector space model representation of protein fold space in 26 dimensions (see Figure 3). This representation of protein structure may be used to test for the separability of protein folds independently of protein similarity measures (*e.g.*, root mean square deviation) or the use of any particular classifier.

It has been shown that the cut edge weight statistic is a proper test for data separability and consequently learnability of any given data set (Zighed et al., 2002). However, this requires computing an RNG that is time consuming ($O(n^3)$ operations, where n is the number of protein structures to be analyzed; see Methods). Here we show that a HSP can be computed faster than a RNG and

demonstrate that a HSP always contains a RNG (see Figure 4), thus HSP can be used as an alternative to generate the RNG in amortized $O(n^2)$ (see Figure 5).

We tested for the separability of this fold space with domains and classes reported in CATH at the topology level and found that it is feasible to automatically learn this fold space structure. For an intuitively view of this fold space, Figure 8 presents a two-dimensional embedding of domains coloured according to its Class. Even though all features in 26 dimensions cannot be preserved in such embedding, it can be appreciated that $\alpha\beta$ domains laid in between mainly-α and mainly-β domains. Next we run different tests to determine if this separation reproduces the one proposed by CATH.

The accuracies of an Extremely Random Forest classifier in Table 1 shows a good performance in a cross validation test. Our score of 0.969 predicting CATH Class can be directly compared with that of 0.857 reported in Shivashankar et al. (2011).

A near perfect score is obtained when evaluated using the same training and test set. The discrepancies on this test are presented in Table 2. Although no confusion exists between our ERF and CATH in classifying mainly-α and mainly-β, there are several differences between $\alpha\beta$ and mainly-α or mainly-β. Some examples of these discrepancies are presented in Figure 7 where predicted classes by ERF visually appear to be more convenient than the CATH assignment. In the context of fold space, these structures may be in the region populated with domains belonging to the predicted class, but may represent an issue with the original assignment given by CATH, as has been already mentioned early by CATH developers (Michie et al., 1996).

For the task of structural neighbour retrieval, we showed that our vector space model representation of protein fold space outperforms that of FragBag using the euclidean distance. However, this distance criterion applied on our vector space model matched the hit-rate score of ContactLib. This trend in our approach may be explained by considering that protein structures under the same structural classification might have boundary limits that a distance may not easily detect, as noted in Figure 1. When a supervised approach is used to learn this boundaries, our results outperform ContactLib in terms of mean AUROC and overall AUROC scores distribution, but it is almost identical in the case of mean AUROC for SAS20. The result that using two different approaches find a plateau at 0.96 of mean AUROC, suggest an upper boundary possible under this SCOP Superfamily assignment. In any case, the cross-validated scores presented here may serve as a baseline for future predictors over this fold space representation to learn CATH or SCOP assignments.

5. Conclusion

In summary, we present a vector representation of protein folds that reproduce basic aspects of protein packing. We describe an efficient algorithm to test for the separability of this vector space and show that our vector representation of protein fold space is separable and consequently learnable by any classifier. This property was used for direct prediction of structural classifications and for the task of retrieving protein structural neighbours. To facilitate other research groups to use the algorithms described here to represent protein structures as vectors, we have implemented an API in Python available at https://code.google.com/p/pyrcc/.

Funding This work was supported in part by PAPIIT/UNAM (IN205911) to GDR. CONACyT and UNAM supported RCC studies.

Conflict of interest statement. None declared.

Acknowledgements

We thank the technical support from Dra. Maria Teresa Lara, the IT core facility at the Institute of Cellular Physiology/UNAM and the supercomputer center at the DGTIC/UNAM.

Appendix A. Supplementary Data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.compbiolchem.2015.07.010.

References

Cheng, S., Brooks 3rd, C.L., 2013. Viral capsid proteins are segregated in structural fold space. PLoS computational biology 9 (2), e1002905.
Kolodny, et al., 2006. Protein structure comparison: implications for the nature of 'fold space', and structure and function prediction. Current opinion in structural biology 16 (3), 393–398.
Skolnick, et al., 2009. The continuity of protein structure space is an intrinsic property of proteins. Proceedings of the National Academy of Sciences of the United States of America 106 (37), 15690–15695.
Sadreyev, et al., 2009. Discrete-continuous duality of protein structure space. Current opinion in structural biology 19 (3), 321–328.
Minary, P., Levitt, M., 2008. Probing protein fold space with a simplified model. Journal of molecular biology 375 (4), 920–933.
Orengo, C.A., Taylor, W.R., 1996. SSAP: sequential structure alignment program for protein structure comparison. Methods in enzymology 266, 617–635.
Michie, A.D., et al., 1996. Analysis of domain structural class using an automated class assignment protocol. Journal of molecular biology 262 (2), 168–185.
Holm, L., Sander, C., 1998. Touring protein fold space with Dali/FSSP. Nucleic acids research 26 (1), 316–319.
Hou, J., et al., 2003. A global representation of the protein fold space. Proceedings of the National Academy of Sciences of the United States of America 100 (5), 2386–2390.
Hou, J., et al., 2005. Global mapping of the protein structure space and application in structure-based inference of protein function. Proceedings of the National Academy of Sciences of the United States of America 102 (10), 3651–3656.
Budowski-Tal, et al., 2010. FragBag, an accurate representation of protein structure, retrieves structural neighbors from the entire PDB quickly and accurately. Proceedings of the National Academy of Sciences of the United States of America 107 (8), 3481–3486.
Rogen, P., Fain, B., 2003. Automatic classification of protein structure by using Gauss integrals. Proceedings of the National Academy of Sciences of the United States of America 100 (1), 119–124.
Sippl, M.J., 2008. On distance and similarity in fold space. Bioinformatics (Oxford, England) 24 (6), 872–873.
Zighed, Djamel A., et al., 2002. Separability Index in Supervised Learning. Principles of Data Mining and Knowledge Discovery. Volume 2431, Lecture Notes in Computer Science. Springer, Berlin Heidelberg, pp. 475–487.
Lubell, D., 1996. A short proof of Sperner's lemma. Journal of Combinatorial Theory 1 (2), 299.
Cusack, M.P., et al., 2007. Efficient identification of critical residues based only on protein structure by network analysis. PloS one 2 (5), e421.
Tomita, E., et al., 2006. The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical Computer Science 363 (1), 28–42.
Hagberg, A., et al., 2008. Exploring network structure, dynamics, and function using networkx. In: Conference: SCIPY.
Pedregosa, F., et al., 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, 2825–2830.
Holden, S.B., Niranjan, M., 1995. On the practical applicability of VC dimension bounds. Neural computation 7 (6), 1265–1288.
Belkin, M., Niyogi, P., 2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. Neural computation 15 (6), 1373–1396.
Fischer, H., et al., 2004. Average protein density is a molecular-weight-dependent function. Protein Sci 13 (10), 2825–2828.
Xuefeng Cui, et al., 2014. Fingerprinting protein structures effectively and efficiently. Bioinformatics 30 (7), 949–955, doi: 10.1093/bioinformatics/btt659.
Belkin, Mikhail, Niyogi, Partha, 2008. Convergence of Laplacian Eigenmaps.
Shivashankar, S., et al., 2011. Multi-view methods for protein structure comparison using latent dirichlet allocation. Bioinformatics 27 (13), i61–i68, doi: 10.1093/bioinformatics/btr249.
Chavez, E., et al., 2006. Half-space proximal: A new local test for extracting a bounded dilation spanner of a unit disk graph. Principles of Distributed Systems. Springer LNCS, pp. 235–245.