Exploring Similarities of Conserved Domains/Motifs

Sotiria Palioura *

Abstract

Traditionally, proteins are represented as amino acid sequences. There are, though, other (potentially more exciting) representations; e.g. each protein can be viewed as (maybe partially) containing some conserved domains/motifs. Using this "functional representation" of proteins we are suggesting a method for exploring similarities of conserved domains/motifs.

1 Introduction

Recent advances in genome sequencing projects and protein crystallization techniques have presented bioinformatics with the challenge of storing and organizing a vast amount of molecular biology information. Pilot databases have been developed for easy storage and retrieval of the data generated by functional and structural genomics efforts [1]. Most such databases have been built by focusing on protein-protein similarities in terms of several evolutionary or structural protein characteristics. Currently, there exist protein classifications with respect to evolutional homology (COGs), conserved domains (CDD), structural fold similarities (SCOP) and gene expression patterns (GEO) [2]. Such classifications allows us to survey and process the stored information by providing a more concise representation of genomic and proteomic data.

Several supervised and unsupervised machine learning techniques have been explored for data mining in the above databases (k-means, hierarchical clustering algorithms, self-organizing maps, etc.) [3]. The goal of the above algorithmic approaches is to facilitate the analysis of existing datasets by revealing biologically meaningful information in a systematic way. Such observations can subsequently be incorporated into gene and protein annotation search programs, such as BLAST and FASTA, thus leading to more accurate functional and structural prediction of uncharacterized sequenced proteins.

Towards resolving the challenging genome and protein annotation problem, we are proposing a data mining technique that could provide invaluable insight into protein function. The method explores the frequency of co-occurrence of pairs of conserved domains/motifs in the proteome. It takes advantage of a standard data mining technique for large datasets based on matrix multiplication. A "functional representation" of a large number of proteins in the conserved domain/motif space serves as input to the algorithm. The approach is systematic, automated and can be easily adjusted to accommodate new proteomic information.

*Department of Molecular Biophysics & Biochemistry, Yale University, New Haven, CT 06520, sotiria.palioura@yale.edu.

2 Methods

The "heart" of our method is the construction of a protein vs. conserved domain/motif[1] matrix (from now on we will call this matrix A). The rows of A denote proteins and the columns of A denote conserved domains/motifs. A is an $m \times n$ matrix, where m is the number of proteins that we examine and n the number of conserved domains/motifs in our database. The (i, j)-th element of A (denoted by A_{ij}) represents the "presence" or "absence" of conserved domain/motif j in protein i. We allow A_{ij} to be any number from 0 to 1, where a fractional value represents "partial" presence of the conserved domain/motif in this protein; we elaborate on this later.

In order to create A, we will use the SMART database and NCBI's Conserved Domain Database (CDD). We unfortunately need both databases; a Conserved Domain (CD) search in CDD only recognizes conserved domains and not small functional motifs [6]. For the latter task we will use the SMART database [7]. SMART will also provide explicit amino acid sequences for more than 400 domain families to our algorithm[2].

We will now describe in detail the construction of A (m will be in the range of 50000-60000 and $n > 400$). A is initialized to be an all-zeros matrix.

Step 1 For all the domain families in SMART find their corresponding amino acid sequence.

This step is trivial; SMART provides a table of domain families with links to their amino acid sequence.

Step 2 We run queries in CDD using an amino acid sequence that denotes a domain family.

Here we note that we will use the *Expect* advanced option in our CDD search. More specifically, we will run queries for a few possible e-values of *Expect*, ranging from 10^{-6} up to 1 (we do not use higher e-values in order to avoid false positives, see [6]).

The output of the query (for some particular e-value) consists of proteins containing the domain family. Depending on the e-value, the protein might contain a slight variation of the domain family; e.g. some amino acids might be different or misaligned. If the output of a query is empty we run the same amino acid sequence in the SMART database, since the sequence might be a small motif and thus not recognizable by CDD. Again, SMART outputs a list of proteins containing this motif.

Step 3 For every protein i that contains the domain family j we set the "score" A_{ij}.

Setting the "score" A_{ij} is not obvious. One possibility would be to set $A_{ij} = 1$ if the query on domain family j returned protein i with e-value= 10^{-6} and 0 otherwise, meaning that we only account for the presence[3] or absence of the domain family in the protein. Obviously, we could determine presence or absence by using any of the aforementioned e-values. A more meaningful representation, capturing the *partial* presence of a domain family in the protein, emerges by relaxing the initial scoring scheme, e.g. by setting $A_{ij} = 0.95$ instead of 0 if the query on domain family j returned protein i with e-value= 10^{-4} etc. Deciding the exact scoring scheme will happen at a later stage by training a generic scoring scheme on our data.

Repeat Steps 2 and 3 for all domain families

[1] There is no clear distinction between the terms motif and conserved domain. Generally, a motif is considered to be a small conserved amino acid sequence (even of length 3) while a conserved domain is a much longer sequence. For more details, see [6].

[2] We use the term domain families to denote conserved domains *and* motifs.

[3] Indeed, we only allow a *very strong* presence of the domain family, otherwise we consider it absent.

We should also note that the construction of the above matrix can be automated by writing small scripts (using Perl) to mine the CDD and SMART databases.

As mentioned in the introduction, our goal is to mine interesting correlations between domain families; more specifically, to identify for *each* domain family the domain families that are its *nearest neighbors*. We will measure the proximity of two domain families by examining their pattern of co-occurrences in the proteins of the dataset (as denoted by matrix A).

There are a few obvious candidates for measuring the proximity of two domain families based on their respective patterns. We denote by p the m-dimensional vector of the first domain family and by q the m-dimensional vector of the second domain family. Then, $|p-q|$ is defined as the Euclidean distance between the two vectors[4]. Obviously, if $|p-q|$ is small, the two domain families are similar in the sense described above.

Instead of this -more intuitive- distance measure, we will borrow a different measure from Data Mining Applications in Computer Science; namely the outer product of p and q. Indeed, experimental evidence in a variety of datasets has shown that this measure is superior to the ones mentioned above. We remind that the outer product of p, q is defined as:

$$p \cdot q = \sum_{i=1}^{m} p_i q_i = |p||q| \cos(\widehat{p,q})$$

where p_i and q_i are the elements of p and q, $|p|, |q|$ are the lengths of p and q[5] and $\cos(\widehat{p,q})$ is the *angle* between p and q. Intuitively, if this angle is small (in which case the cosine of the angle is close to 1), then we know that the two vectors (and thus the corresponding domain families) are "close". Thus, if we compute $p \cdot q, |p|$ and $|q|$ we can compute

$$\cos(\widehat{p,q}) = \frac{p \cdot q}{|p||q|}$$

Surprisingly, computing the dot products for all possible pairs of domain families is very easy: it amounts to the multiplication of two matrices, A^T and A. We remind that A^T denotes the transpose of matrix A, that is a matrix whose rows are columns of A and vice versa. It is easy to see that if A is an $m \times n$ matrix, A^T is an $n \times m$ matrix.

We now give an example using a generic matrix A. Say that A is a matrix describing 8 proteins with respect to 4 domain families, e.g.

$$A = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \\ p_{51} & p_{52} & p_{53} & p_{54} \\ p_{61} & p_{62} & p_{63} & p_{64} \\ p_{71} & p_{72} & p_{73} & p_{74} \\ p_{81} & p_{82} & p_{83} & p_{84} \end{bmatrix}$$

Then,

$$A^T = \begin{bmatrix} p_{11} & p_{21} & p_{31} & p_{41} & p_{51} & p_{61} & p_{71} & p_{81} \\ p_{12} & p_{22} & p_{32} & p_{42} & p_{52} & p_{62} & p_{72} & p_{82} \\ p_{13} & p_{23} & p_{33} & p_{43} & p_{53} & p_{63} & p_{73} & p_{83} \\ p_{14} & p_{24} & p_{34} & p_{44} & p_{54} & p_{64} & p_{74} & p_{84} \end{bmatrix}$$

[4]We could also use the 1-norm or any other p-norm, but we do not elaborate on this issue.

[5]We remind that $|p|^2 = \sum_{i=1}^{m} p_i^2$

Finally, the product A^TA is equal to

$$A^TA = \begin{bmatrix} \sum_{k=1}^{8} p_{k1}p_{k1} & \sum_{k=1}^{8} p_{k1}p_{k2} & \sum_{k=1}^{8} p_{k1}p_{k3} & \sum_{k=1}^{8} p_{k1}p_{k4} \\ \sum_{k=1}^{8} p_{k2}p_{k1} & \sum_{k=1}^{8} p_{k2}p_{k2} & \sum_{k=1}^{8} p_{k2}p_{k3} & \sum_{k=1}^{8} p_{k2}p_{k4} \\ \sum_{k=1}^{8} p_{k3}p_{k1} & \sum_{k=1}^{8} p_{k3}p_{k2} & \sum_{k=1}^{8} p_{k3}p_{k3} & \sum_{k=1}^{8} p_{k3}p_{k4} \\ \sum_{k=1}^{8} p_{k4}p_{k1} & \sum_{k=1}^{8} p_{k4}p_{k2} & \sum_{k=1}^{8} p_{k4}p_{k3} & \sum_{k=1}^{8} p_{k4}p_{k4} \end{bmatrix}$$

Now, we divide each element of A^TA by the product of the lengths of the vectors that form it. We will abuse notation and call this new matrix $\cos(A^TA)$, since it essentially denotes the cosine of the angle between every pair of domain families that we examine. We will not explicitly state every element of $\cos(A^TA)$, but instead we will give the formula for $\Big(\cos(A^TA)\Big)_{ij}, i = 1 \ldots 4, j = 1 \ldots 4$ (the (i,j)-th element of $\cos(A^TA)$)

$$\Big(\cos(A^TA)\Big)_{ij} = \frac{\sum_{k=1}^{8} p_{ki}p_{kj}}{\sqrt{\sum_{k=1}^{8} p_{ki}^2}\sqrt{\sum_{k=1}^{8} p_{kj}^2}}$$

Observe that the (i,j)-th element of $\cos(A^TA)$ denotes the proximity of domain families i and j. Thus, by observing $\cos(A^TA)$ (a 4×4 matrix) we can deduce the proximity of any pair of domain families. Finally, we note that the (i,j)-th element of $\cos(A^TA)$ is equal to the (j,i)-th element of $\cos(A^TA)$; obviously, the proximity of the domain families i,j and j,i is the same[6].

It should now be clear how to generalize the above scheme in order to handle a large number of proteins and domain families. A would be an $m \times n$ matrix, A^T an $n \times m$ matrix and their product A^TA an $n \times n$ matrix[7]. We remind that n is the number of domain families that we are examining; thus, $\cos(A^TA)$ is a matrix of domain families vs. domain families and its (i,j)-th element denotes the proximity of the i,j domain families.

To help the reader understand the process, we note that the diagonal elements of $\cos(A^TA)$ are all equal to 1 (as expected, since they denote the distance between the same domain family!). We are intending to process the above matrix as follows:

1. For every row of $\cos(A^TA)$ sort its elements in decreasing order.
2. For every row of $\cos(A^TA)$ plot the sorted sequence (angles calculated vs. domain families).

A number of useful observations can emerge from the above processing. For every domain family, we can identify the domain families that are its nearest neighbors (meaning a domain family that has a similar occurrence pattern across all proteins), as well as the domain families that are its farthest neighbors. Intuitively, this should reveal information about the functions associated with the domain families and possibly (hopefully!) new and interesting relationships between them.

3 Discussion

We will now briefly discuss a few issues related to the previous technique. The number of proteins used in the matrix will depend on the amount of the proteome being characterized in

[6]This happens because $\cos(A^TA)$ is always a *symmetric* matrix independently of the structure of A.

[7]Creating A^TA as well as computing $\cos(A^TA)$ is a trivial task, e.g. using a software like Matlab.

terms of conserved domains/motifs in the respective databases. The method can be used for a very large number of proteins, e.g. we can handle 200,000 proteins. In fact, using more proteins should return more accurate results.

Since the method is highly systematic and automated, newly characterized proteins and conserved domains/motifs can easily be incorporated in our matrix. However, the unfiltered use of a vast number of proteins may bias the results, because of a potential over-representation of some types of homologous proteins in the sample. Therefore, scaling the proteins used in the matrix must be done with respect to their origin and homology.

This method opens new perspectives in scoring protein similarity, not only in terms of amino acid sequence, but also in terms of co-occurring conserved domains/motifs. In particular, we could also apply this technique using secondary structural elements and/or folds instead of conserved domains/motifs to characterize the proteins. Thus, a hybrid scoring scheme assessing protein similarities could be developed, combining information from amino acid similarity searches, domain families co-occurrences and protein folds. The weight of each of these aspects in the final scheme could be determined by training the scoring scheme on the existing database. The potential of such a scheme is clear; it could aid researchers to focus on experiments that are more meaningful from a biological viewpoint, thus rendering experimental approaches highly efficient with respect to both time and resources.

Finally, we conclude by mentioning another exciting possibility: we could use clustering algorithms (spectral, k-means or other more heuristic approaches) in our matrix A in order to cluster the proteins using their representation with respect to conserved domains/motifs. This clustering would essentially be a functional classification of the proteins.

Acknowledgements: I would like to thank Petros Drineas for many helpful discussions and for explaining the Computer Science approach to Information Retrieval.

References

[1] Luscombe N.M., Greenbaum D. and Gerstein M.(2001) *What is Bioinformatics? A Proposed Definition and Overview of the Field*, Methods Inform Med, 40, 346-358.

[2] Gerstein M. (2000), *Integrative database analysis in structural genomics*, Nature Structural Genomics, 7, 960-963.

[3] Gerstein M. and Jansen R. (2000) *The current excitement in bioinformatics-analysis of whole-genome expression data: how does it relate to protein structure and function?*, Current Opinion in Structural Biology, 10, 574-584.

[4] G.H.Golub and C.F.Van Loan, *Matrix Computations*, Johns Hopkins University Press, London, 1989

[5] Schultz J., Copley P.R., Doerks T, Ponting C.P. and Bork P. (2000) *SMART: a web-based tool for the study of genetically mobile domains*, Nucleic Acids Research, 28, 231-234.

[6] *http://www.ncbi.nlm.gov/Structure/cdd/cdd.shtml*

[7] *http://www.smart.embl-heidelberg.de*