G-Class: A Divide and Conquer Application for Grid Protein Classification

Helen E. Polychroniadou, Fotis E. Psomopoulos, and Pericles A. Mitkas

Electrical and Computer Engineering Dept., Aristotle University of Thessaloniki,
54124, Thessaloniki, Greece
Tel.: +30-2310-996349, Fax: +30-2310-996398
elpol@egnatia.ee.auth.gr, fpsom@auth.gr, mitkas@eng.auth.gr

Abstract. Protein classification has always been one of the major challenges in modern functional proteomics. The presence of motifs in protein chains can make the prediction of the functional behavior of proteins possible. The correlation between protein properties and their motifs is not always obvious, since more than one motif may exist within a protein chain. Due to the complexity of this correlation most data mining algorithms are either non efficient or time consuming. In this paper a data mining methodology that utilizes grid technologies is presented. First, data are split into multiple sets while preserving the original data distribution in each set. Then, multiple models are created by using the data sets as independent training sets. Finally, the models are combined to produce the final classification rules, containing all the previously extracted information. The methodology is tested using various protein and protein class subsets. Results indicate the improved time efficiency of our technique compared to other known data mining algorithms.

1 Introduction

The development of elaborate and specialized bioinformatics computational tools has led to revolutionary changes in the analysis of biological sequences. Visualizing, manipulating and predicting molecular structure and function, separating DNA sequences according to protein coding regions, classifying protein, DNA and RNA molecules or detecting weak similarities have come to rely vitally on computational methods [2, 15]. The continuous increase in size of biological databases requires new, computation-sensitive data mining approaches, but also presents unique opportunities for new fields of inquiry; among these lies one of bioinformatics more ambitious goals, the prediction of the functional behavior of proteins.

Proteins are large molecules composed of one or more chains of amino acids in a specific order, which is determined by the base sequence of the nucleotides in the gene coding the protein. A typical protein representation is shown in **Fig. 1**, where the 534 amino acids, constituting the protein chain P00747 (Plasminogen precursor), are represented by their formal one-character abbreviation.

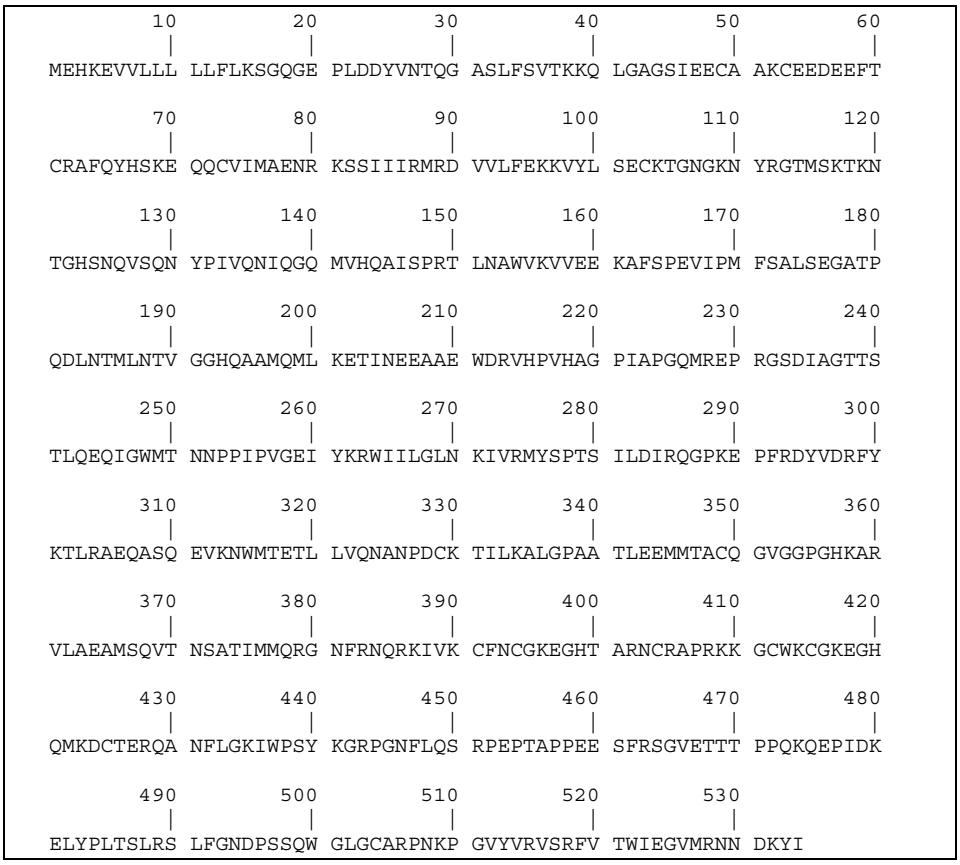

Fig. 1. P00747 (Plasminogen precursor – PLMN_HUMAN) protein chain

Common behavioral characteristics and strong structural similarities enable classification of proteins into protein families. The lack of cost- and time-efficient experimental methods has given rise to computational approaches for protein properties identification. Moreover, the overlapping of protein families almost at all times - the classification of a protein into multiple families with different similarity levels - makes the procedure even harder, due to its ascending complexity.

Despite the preceding difficulties, protein functionality prediction could hardly be achieved were it not for *motifs*, short amino acid sequences of specific order, which appear in protein chains and play a decisive role in protein behavior. Although a straightforward mapping between motifs and protein properties is hard to achieve due to the presence of multiple motifs in each protein chain, they can facilitate prediction of protein functionality, if the latter is considered to be derived by the combining effect of many, either conflicting or consistent, motifs.

Two different types of motifs can be distinguished; *patterns* and *profiles*. The first is the simplest form of a motif and can be represented by an expression like the following example:

```
[AC] - G - x - V - x(4) - [ED]
```

This syntax provides specific information about the particular sequence of amino acids. It states that the first position can be occupied by an Alanine (A) or a Cystine (C), while in the second position there must be a Glycine (G). The symbol x that follows stands for any of the 20 amino acids, and x(4) means four such places.

More complex than a pattern, the profile constitutes a multiple sequence alignment, symbolized by alignment matrices. The question of occurrence of a particular motif in a protein chain is where the main distinction between the two types lies. Comparison with a profile provides information of similarity, in contrast to a definite, but simpler, answer of YES or NO obtained when a pattern is used.

The overall procedure of motif identification and detection in a protein sequence can be carried out in two different ways; *unsupervised* or *supervised*. The former can be accomplished by a large variety of machine learning algorithms, whereas the latter requires prior information, such as expert opinion or experiment conclusions. A comprehensive and well-documented database created by experts in the field of molecular biology, such as the PROSITE [7, 25] the PFAM [3, 26] or the PRINTS [1, 27] databases can facilitate motif search in a way that satisfies these requirements.

Related literature features many data mining algorithms that utilize the presence of motifs in protein sequences to perform protein classification, originating from the field of pattern recognition [6], as well as that of artificial intelligence [5, 14]. They include many different techniques, such as decision trees [19], statistical models, neural networks [4] or finite state automata [16, 17].

This paper presents G-Class (*Grid Classification*), a novel methodology that aims to benefit from the use of grid technology in data mining applications. The combination of the two leading technologies can help overcome the computational difficulties often encountered in protein classification problems.

The G-Class methodology follows a “divide and conquer” approach comprising 3 steps: First, protein data from an expert-based database are divided into multiple disjoint sets, each one preserving the original data distribution. The new sets are used as training sets, and multiple models are derived by means of standard data mining algorithms [19, 23]. Finally, the models are combined to produce the final classification rules, which can be used to classify a given instance and evaluate the methodology.

The paper is organized as follows: Section 2 presents an outline of the G-Class methodology and Section 3 introduces the procedure of Data Splitting. Section 4 introduces the Classifier Training process, together with some background information on Grid Computing while Section 5 describes the process of the Classifier Combination. Section 6 presents some comparative experimental results and, finally, Section 7 summarizes and concludes the paper.

2 Methodology Outline

The main goal of the G-Class methodology is to utilize existing data mining algorithms in a parallel-enabled environment, such as the Grid, in order to create a protein classification model. The methodology is illustrated in a block diagram in **Fig. 2**. The different steps of the process are described below.

Initially, data from a supervised motif database (in our case PROSITE) are formatted in a single `.arff` file. As the data are going to be used for the training of the model, the class in which the protein belongs to, as well as the motifs it contains, must be known. The `.arff` file generally has the form presented in **Fig. 3**.

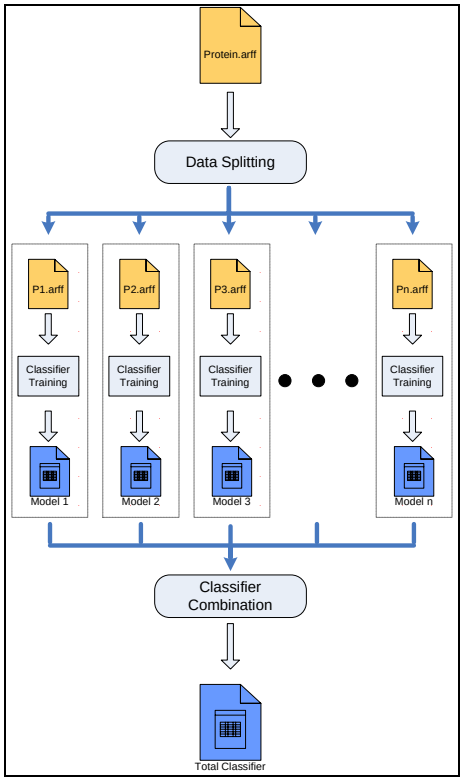

Fig. 2. G-Class Methodology

According to the specifications, an `.arff` file is divided into two parts: the first part contains all the information describing the data and the second part contains the actual data.

In the first part, the attribute `protein` lists all the protein codes that should be included in the training set. The next attributes are binary, specifying the occurrence or not of a specific motif in a protein chain. The last attribute (`CLASS`) contains a listing of all protein classes that should be taken under consideration during the classification process. It must be noted that for each protein listed in the `protein` part,

all the corresponding classes must be listed in the `CLASS` part. Finally, the second part lists all protein data in the form of a string vector, such as the following:

```
<protein code>, NO, NO, NO, YES, NO, ..., NO, YES, NO, <class code>
```

In case a protein belongs to more than one class, then for each one of these classes a new instance is listed in the `@data` part of the file.

```
@relation protein

@attribute protein {<List of Protein Codes>}
@attribute PS00010 {YES,NO}
@attribute PS00011 {YES,NO}
@attribute PS00012 {YES,NO}
...
@attribute PS60000 {YES,NO}
@attribute CLASS {<List of Class Codes>}
@data
<List of data Instances>
...
```

Fig. 3. Form of an `.arff` file

The single input file is then split into multiple files of the same format. The process makes sure that each new file contains a unique subset of the original data, maintaining also the same class distribution found in the original file. This allows for the independent use of them and therefore enables parallel processing.

In order to extract a useful knowledge model from each one of the datasets any classification algorithm can be used [11, 13].

Finally, the multiple knowledge models are combined into a single unified model, by means of a voting procedure, and the new model is tested both on the original dataset and on test datasets.

3 Data Splitting

The first step of the G-Class methodology, after the creation of the original single dataset, is to divide it into multiple `.arff` files. The process is performed by the algorithm `ArffSplitter` presented in **Fig. 4**.

As input the algorithm requires an `.arff` file with the data to be divided. It then reorders the data so as to cluster entries with the same protein class. Finally, instead of simply chopping the file into equal parts, `ArffSplitter` employs a method similar to Round-Robin to equally distribute the members of each class to all the files, thus preserving the initial class distribution.

Experimentally, it is found that the method is robust in the class allocation, both for different number of splits and for varying number of classes involved, as shown in **Fig. 5**.

```
algorithm ArffSplitter
input:
      a single .arff file
      n: the number of splits
output:
      n .arff files
begin
  Extract the data from the file and rearrange it so that entries
                    with the same protein class are grouped together.

  Create n new .arff files and set the data description part in each
                     one exactly the same as the original one,
                     leaving however the protein attribute empty.

  do (for each entry i in the original reordered file)
    copy entry[i] to file [i MOD n]
    add the protein code of entry[i] to the protein attribute list of
                                                        file [i MOD n].
  end do
end algorithm ArffSplitter
```

Fig. 4. *ArffSplitter* algorithm

4 Classifier Training Process

The splitting of the original data being complete, the next step in the G-Class methodology is creating of the individual knowledge models. Each data set is used to train and test a separate classifier. Since each dataset contains a disjoint subset of the original data, they can be processed in parallel for time efficiency. Any classification algorithm that can be applied on nominal values can be used in this phase. For comparative reasons, the experiments were performed using the C4.5 classification algorithm from the WEKA [21] software package. Moreover, to ensure the quality of the models, the C4.5 algorithm [18] was applied using stratified 10-fold cross-validation.

In order to facilitate an efficient way to train multiple models simultaneously, the training phase makes use of the EGEE Grid resources. At this point a short introduction to the concepts of the Grid and Grid Computing is necessary.

The concept of “the Grid” started to materialize in the mid 1990’s in order to provide a distributed computing infrastructure for advanced science and engineering [8, 10]. It aims to facilitate the flexible, secure, coordinated and controlled resource sharing among computers, to distribute computing power, data storage and specialized equipment use among computing nodes scattered all over the world, and assist “virtual organizations”, dynamic collections of individuals, institutions, and resources, where cooperation is not subject to geographical distance, countries or legal entities.

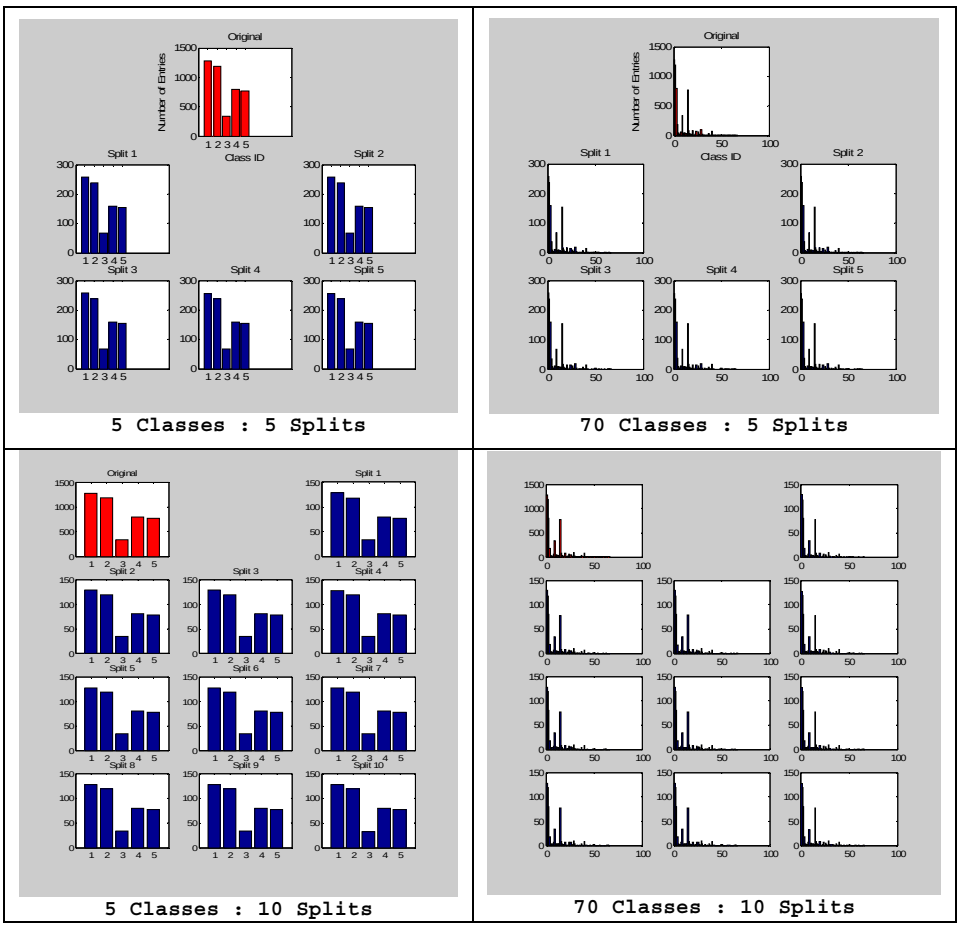

Fig. 5. Class Distributions for varying number of classes and splits (the red-colored graphs are the distributions of the original files)

A commonly accepted definition of the Grid is the following:

"*A Grid is a system that:*
1. coordinates resources that are not subject to centralized control …
2. … using standard, open, general-purpose protocols and interfaces …
3. … to deliver nontrivial qualities of service." [9]

The EGEE (Enabling Grids for E-sciencE) project [24], funded by the European Commission, provides a grid infrastructure ideal for scientific research, especially when the time and resources needed for running the applications are considered impractical with traditional IT infrastructure. In our case, the Grid offers the resources

needed to run multiple training processes in parallel, thus reducing the total time cost of the classification procedure.

In order to run a training process, or to "*submit a job*" in Grid terms, it must be described in a `.jdl` file (Job Description Language), which is subsequently submitted to the Grid together with the actual executable and any necessary input files. The Grid infrastructure is then responsible for assigning the suitable resources, according to the description in the `.jdl` file, and queue the job in the appropriate Grid node. After the successful execution of the training process, the resulting output files are returned [28]. This procedure is repeated for the multiple training processes, each one of them assigned to a Grid node for execution.

At this point it must be noted that, contrary to other parallel classification techniques [22], the G-Class methodology is independent of the actual data mining algorithm. Any classification algorithm available in literature can be utilized in the classifier training process, thus providing a degree of freedom to the methodology. Also, due to the fact that the training datasets were equivalent regarding the actual data representation, the final knowledge models are also of equivalent accuracy.

5 Classifier Combining

The final step in the G-Class methodology is the efficient combination of the multiple knowledge models, which were extracted in the previous phase.

```
algorithm ClassifierCombiner
input:
    the multiple classifier models
    the original data set
output:
    a classifier of the whole data set
begin
  do (for each instance i of the original data set file)
    do (for each separate classifier model)
      create distribution vector of the classifier for the
                                                    single instance
    end do
    calculate average vector of all classifiers for the
                                                    single instance
    use final distribution vector to make decision on the
                                classification of the  given instance
  end do
  use all classification decision and probability results to test
                      the final classification on the original data set
end algorithm ClassifierCombiner
```

Fig. 6. *ClassifierCombiner* algorithm

The models are combined by a voting process, where each one is represented by a distribution vector. The performance of the total classifier for a single instance is derived by the average of all the distribution vectors. The overall efficiency of the model is calculated by testing it on the original dataset. The algorithm used is presented in more detail in **Fig. 6**.

Results found through extensive experimentation have shown that the accuracy of the combined model is equal to the accuracy of models that have been extracted directly form the original dataset.

6 Experiments

In order to validate the correctness of the methodology, a number of experiments were performed in comparison with other recorded methods.

In the first case, the input dataset was derived from the 10, 20 and 30 class sets presented in [12]. The results are shown in **Table 1**. The number of splits was selected based on the size of the new `.arff` files that would be produced each time, in order to maintain a similar processing time. It is obvious, that when the number of splits is 1, the original `.arff` file was processed.

Table 1. Results from 10, 20 and 30 classes

							GenMiner [12]	
	# of Splits: 1		**# of Splits: 2**		**# of Splits: 5**			
	Acc. (%)	**Time**	**Acc. (%)**	**Time**	**Acc. (%)**	**Time**	**Acc. (%)**	**Time**
10	76.8	1′ 30′′	79.7	1′ 08′′	-	-	80.1	5′ 53′′
20	68.9	31′ 00′′	71.1	6′ 55′′	70.9	3′ 43′′	84.3	10′ 56.8 ′′
30	65.9	6 h	68.1	2 h 18′ 54′′	67.9	7′ 24′′	74.5	-

An improvement in the processing times can be seen from the table, while the accuracy is fairly constant.

At this point, it must be noted that although the number of the selected classes is 10 (20 and 30 in the other cases), the actual number of classes involved in each of the classification process is much larger, due to the overlapping of the classes. In the 10-class dataset, there is a total of 662 proteins over 27 different classes, in the 20-class dataset there are 2214 proteins over 64 classes and in the 30-class dataset, 2788 proteins over 105 classes.

In fact, the last dataset is the largest one (both in classes and in entries) that can be processed as a single file by WEKA. For this reason, two more datasets were created. The first contains the 5 most populated protein classes, with 4380 proteins. Due to overlapping this dataset actually contains 66 different protein classes, including the initial 5 ones. The second contains the 10 most populated classes (7027 proteins over 96 classes). These two datasets could not be directly processed by WEKA. Using the G-Class methodology the results are presented in **Table 2**.

The numbers of splits for each dataset are different to keep similar file sizes, in order to have comparable results.

Table 2. Results from 5 and 10 most populated classes

	# of Splits: 2		# of Splits:3		# of Splits: 6	
	Acc. (%)	**Time**	**Acc. (%)**	**Time**	**Acc. (%)**	**Time**
5	78.2	1 h 14′ 34′′	78.2	56′ 00′′	78.2	6′ 10′′

	# of Splits: 3		# of Splits:4		# of Splits: 10	
10	81.9	6h 46′ 46′′	81.9	3h 56′ 08′′	81.6	10′ 17′′

Results show a substantial improvement in the processing time while keeping almost constant model accuracy. All the previous results are presented in **Fig. 7**.

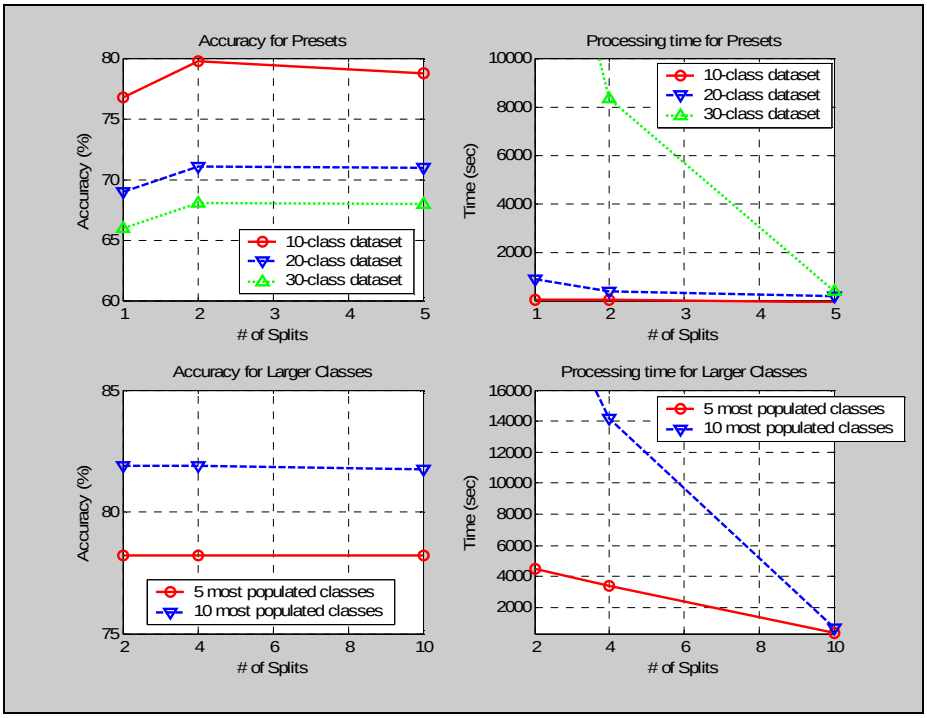

Fig. 7. Experimental Results

The processing time in all cases follows the $e^{-\alpha x}$ model, where α depends on the size of the original dataset and x is the number of splits. The accuracy of the methodology is fairly constant over the number of splits, with minor fluctuations owing to the distribution of the instances of the overlapping protein classes over the different dataset splits.

7 Conclusions

We have presented G-Class, a protein classification methodology for a Grid environment. A protein dataset is divided into multiple disjoint sets, where each one preserves the original class distribution. The new sets are then mined in parallel for knowledge, using any classification algorithm, and the extracted knowledge models are combined by means of a voting procedure.

Results indicate that the methodology is time efficient and shows an overall accuracy comparable with other known methods. It must be noted that the parallelization of the procedure allows for the processing of much larger datasets, as compared to other techniques.

References

1. T.K. Attwood, M.D.R. Croning, D.R. Flower, A.P. Lewis, J.E. Mabey, P. Scordis, J. Selley, W. Wright, "*PRINT-S: the database formerly known as PRINTS*", Nucleic Acids Res. 28, pp. 225–227, 2000.
2. P.F. Baldi, S. Brunak, **Bioinformatics: A Machine Learning Approach**, The MIT Press, Cambridge, MA, 2001.
3. A. Bateman, E. Birney, R. Durbin, S.R. Eddy, K.L. Howem, E.L.L. Sonnhammer, "*The Pfam protein families database*", Nucleic Acids Res. 28, pp. 263–266, 2000.
4. C. Bishop, **Neural Networks for Pattern Recognition**, Oxford University Press, New York, 1995.
5. S. Diplaris, G. Tsoumakas, P.A. Mitkas, I. Vlahavas, "*Protein classification with multiple algorithms*", In Proc. Of 10th Panhellenic Conference in Informatics, Volos, Greece, 21-23 November, Springer-Verlag, LNCS 3746, pp. 446-456, 2005.
6. R. Duad, P. Hart, **Pattern Classification and Scene Analysis**, Wiley, New York, 1973.
7. L. Falquet, M. Pagni, P. Bucher, N. Hulo, C.J. Sigrist, K. Hofmann, A. Bairoch, "*The PROSITE database, its status in 2002*", Nucleic Acids Res. 30, pp. 235–238, 2002.
8. I. Foster, C. Kesselman, **The Grid 2: Blueprint for a New Computing Infrastructure**, 2nd Edition, Morgan Kaufmann, 2003.
9. I. Foster, "*What is the Grid? A three point checklist*", Grid Today, Vol. 1 No. 6, 22 July, 2002.
10. I. Foster, C. Kesselman, S. Tuecke, "*The Anatomy of the Grid: Enabling Scalable Virtual Organizations*", International Journal of Supercomputer Applications, Vol. 15 No. 3, 2001.
11. J. Han, M. Kamber, **Data Mining: Concepts and Techniques**, Morgan Kaufmann, 2001.
12. G. Hatzidamianos, S. Diplaris, I. Athanasiadis, P.A. Mitkas, "*GenMiner: a Data Mining Tool for Protein Analysis*", 9th Panhellenic Conference on Informatics, pp. 346 – 360, November 21 – 23, 2003, Thessaloniki, Greece.
13. B. Kero, L. Russell, S. Tsur, W.M. Shen, "*An Overview of Data Mining Technologies*", In: The KDD Workshop in the 4th International Conference on Deductive and Object-Oriented Databases, Singapore, 1995.
14. C. Makris, Y. Panagis, K. Perdikuri, E. Theodoridis, A. Tsakalidis, "*Algorithms for Protein Clustering*", In Proceeding of the 2nd International Greek Biotechnology Forum, pp 38-44, July 1-3, 2005.
15. G. Piatetsky-Shapiro, W.J. Frawley, **Knowledge Discovery in Databases**, MIT Press,

1992.
16. F. Psomopoulos, S. Diplaris, P.A. Mitkas, "*A Finite State Automata Based Technique for Protein Classification Rules Induction*", In Proceedings of the Second European Workshop on Data Mining and Text Mining for Bioinformatics, pp. 54-60, ECML/PKDD 2004, Pisa, Italy, September 20-24, 2004.
17. F.E. Psomopoulos, P.A Mitkas, "*A protein classification engine based on stochastic finite state automata*", Lecture Series on Computer and Computational Sciences VSP/Brill, 4B (2005), pp. 1371-1374, Proc. of the symp. 35: Computational Methods in Molecular Biology at the ICCMSE 2005, Loutraki, Greece, 21-26 October, 2005.
18. J.R. Quinlan, **C4.5: Programs for Machine Learning**, Morgan Kaufmann, San Mateo, CA, 1993.
19. P. Tan, M. Steinbach, V. Kumar, **Introduction to Data Mining**, Addison-Wesley, 2006.
20. D. Wang, X. Wang, V. Honavar, D. Dobbs, "*Data-driven generation of decision trees for motif-based assignment of protein sequences to functional families*", In: Proceedings of the Atlantic Symposium on Computational Biology, Genome Information Systems & Technology, 2001.
21. I.H. Witten, E. Frank, **Data Mining: Practical Machine Learning Tools and Techniques**, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
22. M. Zaki, C. Ho, R. Agrawal, "*Parallel Classification for Data Mining on Shared-Memory Multiprocessors*", ICDE, 1998.
23. C. Zhang, S. Zhang, **Association Rule Mining**, Springer, 2002.
24. http://www.eu-egee.org/, *EGEE project website*
25. http://au.expasy.org/prosite/, *PROSITE website*
26. http://www.sanger.ac.uk/Software/Pfam/, *PFAM website*
27. http://www.ebi.ac.uk/interpro/, *PRINTS website*
28. http://gridcafe.web.cern.ch/gridcafe/, *Grid Café website*