

https://dspace.org/

Dynamic protein classification: Adaptive models based on incremental learning strategies

Mohamed, Shakir

2008-03-18T09:27:41Z
http://hdl.handle.net/10539/4678

Downloaded from a DSpace repository. https://dspace.org/

Dynamic Protein Classification: Adaptive Models Based on Incremental Learning Strategies

Shakir Mohamed

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 2007

Declaration

I declare that this dissertation is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Signed this ____ day of _____ 20__

_____

Shakir Mohamed

Abstract

One of the major problems in computational biology is the inability of existing classification models to incorporate expanding and new domain knowledge. This problem of static classification models is addressed in this thesis by the introduction of incremental learning for problems in bioinformatics. The tools which have been developed are applied to the problem of classifying proteins into a number of primary and putative families. The importance of this type of classification is of particular relevance due to its role in drug discovery programs and the benefit it lends to this process in terms of cost and time saving. As a secondary problem, multi–class classification is also addressed. The standard approach to protein family classification is based on the creation of committees of binary classifiers. This one-vs-all approach is not ideal, and the classification systems presented here consists of classifiers that are able to do all-vs-all classification.

Two incremental learning techniques are presented. The first is a novel algorithm based on the fuzzy ARTMAP classifier and an evolutionary strategy. The second technique applies the incremental learning algorithm Learn++. The two systems are tested using three datasets: data from the Structural Classification of Proteins (SCOP) database, G-Protein Coupled Receptors (GPCR) database and Enzymes from the Protein Data Bank. The results show that both techniques are comparable with each other, giving classification abilities which are comparable to that of the single batch trained classifiers, with the added ability of incremental learning. Both the techniques are shown to be useful to the problem of protein family classification, but these techniques are applicable to problems outside this area, with applications in proteomics including the predictions of functions, secondary and tertiary structures, and applications in genomics such as promoter and splice site predictions and classification of gene microarrays.

Acknowledgements

"Fools can learn from their own experience; the wise learn from the experience of others" [Democritus]

This statement is especially true for any researcher in computational intelligence. There are many people who have helped me in learning many lessons, and I would like to thank them here. Foremost among these are my parents as well as my brothers for their support during this time. I especially thank my brother, Shaheed Mohamed, who has endured many agonising hours listening to my never ending barrage of words.

I would also like to thank my two supervisors: Prof. Tshilidzi Marwala and Prof. David Rubin for their encouragement and guidance during this time. I especially thank Prof. Marwala for making everyone, including myself strive for excellence and the experiences outside of research that he has given me. Most importantly I would like to thank my fellow colleagues of the research unit: Thando Tettey, Fulufhelo Nelwamondo and Busiswe Vilakazi. It is because of their constant 'pressure' and insatiable ambition that has made this process go as smoothly and as successfully as it has. I would also like to thank my students for their interest in this research and their encouragement.

Finally, the financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and is not necessarily to be attributed to the NRF.

Contents

Declaration i

Abstract ii

Acknowledgements iii

Contents iv

List of Figures ix

List of Tables xi

List of Publications xiii

Nomenclature xiv

1 Modelling and Classification in Proteomics 1

1.1 Introduction 1

1.2 Protein Classification and Current Methods 2

1.3 Problem Statement 3

1.4 An Approach using Computational Intelligence 4

1.5 Outline of the Dissertation 5

2 Protein Sequencing and Approaches to Classification 8

2.1 Introduction 8

2.2 Molecular Biology Primer 9

2.3 Sequencing and Protein Databases 12

2.3.1 Protein Sequencing 12

2.3.2 Representation and Storage of Protein Data 14

2.3.3 Overview of SCOP 15

2.3.4 Overview of GPCRDB 19

2.3.5 Overview of PDB 21

2.3.6 Sequence Extraction and Preprocessing 21

2.4 Sequence Alignment Techniques for Classification 23

2.5 Problems with Alignment Based Techniques 24

3 Biological and Machine Learning Systems 26

3.1 Introduction 26

3.2 An Overview of Biological Evolution and Learning 27

3.3 Machine Learning 30

3.4 Common Machine Learning Tools 31

3.4.1 Artificial Neural Networks 31

3.4.2 Support Vector Machines 33

3.4.3 k-Nearest Neighbour Classification 34

3.5 Multiple Classifier Systems 35

3.6 Machine Incremental Learning 37

3.6.1 Aspects of Incremental Learning 37

3.6.2 Algorithms for Incremental Learning 39

3.7 Evolutionary Optimisation via Genetic Algorithm 42

4 Incremental Learning of Protein Data **44**

4.1 Introduction 44

4.2 Vectorisation of Protein Sequences 45

4.2.1 Global Feature Generation 45

4.2.2 Local Feature Generation 46

4.2.3 Data Normalisation 48

4.2.4 Exploratory Data Analysis 49

4.3 Fuzzy ARTMAP and Incremental Learning 52

4.3.1 System Overview 52

4.3.2 Ensemble Diversity and System Training 53

4.3.3 Incremental Learning of Protein Data 58

4.4 Ensemble System Testing and Experimental Results 59

4.4.1 Testing of SCOP Data 59

4.4.2 Testing of GPCR Data 64

4.4.3 Testing on Enzyme Data 66

4.5 Learn++ for Protein Classification 68

4.5.1 Overview of Learn++ 68

4.5.2 System Overview 71

4.5.3 Application of Learn++ 71

4.6 Learn++ Testing and Experimental Results 72

4.6.1 Testing of SCOP Data 72

4.6.2 Testing of GPCR Data 73

4.6.3 Testing of Enzyme Data 74

4.6.4 Design Strategies for use of Learn++ 75

5 Conclusion **77**

5.1 Discussion and Comparison of Techniques 77

5.2 Suggestions for Future Research 80

5.3 Concluding Remarks 81

A Additional Properties of Amino Acids **82**

B Theory and Principles of Fuzzy ARTMAP **84**

B.1 Introduction 84

B.2 Fuzzy ARTMAP 84

C Learn++ Incremental Algorithm **88**

C.1 Introduction 88

C.2 Details of Learn++ 88

Glossary **91**

References **93**

Bibliography **103**

Publication Reprints

List of Figures

1.1 General pattern recognition approach 5

2.1 Central dogma of molecular biology 9

2.2 Four Levels of Protein Description 10

2.3 Chemical structure of an amino acid 11

2.4 Amino acid sequence of α-chain Haemoglobin 11

2.5 Example of a gel electrophoresis run 13

2.6 SCOP hierarchy 16

2.7 Example of text file showing the SCOP classification. 17

2.8 Sample FASTA file format extracted from the ASTRAL database. . 17

2.9 Growth of sequence databases 18

2.10 Average Daily growth of sequence databases 19

2.11 Sample EMBL file format extracted from the GPCR database. . . . 20

2.12 Distribution of the number of sequences per family. 23

3.1 Common neural network architectures 32

3.2 Representation of the Fuzzy ARTMAP Architecture 40

4.1 Sequence length distribution for SCOP dataset 51

4.2 Sequence length distribution for GPCR data 52

4.3 Overview of system architecture 53

4.4 Incremental performance of system 63

4.5 Combining the hypotheses of individual classifiers to allow for incremental learning. 70

4.6 Overview of Learn++ based incremental learning system 71

B.1 Detailed structure of the fuzzy ARTMAP 85

List of Tables

2.1 Various publicly available protein databases on the Internet 14

4.1 Summary of features used for classifier designed 48

4.2 Comparison of J_3 measures for some well known data sets 50

4.3 Data format for error matrix between classifiers 56

4.4 SCOP 1.69 protein families considered in this work 60

4.5 Comparative performance of FAM versus other classifiers on the SCOP dataset. 61

4.6 Error and agreement values for 15 classifiers of the population 62

4.7 Training and generalisation performance of system on SCOP data . 62

4.8 Separation of data into individual databases for testing using GPCR data. 64

4.9 Comparative performance of FAM versus other classifiers on the GPCR dataset. 65

4.10 Error and agreement values for 15 classifiers of the population 65

4.11 Training and generalisation performance of system on GPCR data . 66

4.12 Separation into individual databases for training and testing using enzyme data. 66

4.13 Comparative performance of FAM versus other classifiers on the enzyme dataset. 67

4.14 Error and agreement values for 15 classifiers of the population 68

4.15 Training and generalisation error of system on Enzyme data 68

4.16 Comparison of errors for Learn++ as the number of training databases in increased. 73

4.17 Training and generalisation performance of system on GPCR data . 74

4.18 Training and generalisation performance of system on Enzyme data 74

A.1 International notations for amino acid R-groups 82

List of Publications

The following publications were produced during the course of this research:

- S. Mohamed, D. Rubin, and T. Marwala,"Multi-class protein classification using Fuzzy ARTMAP" in Proceedings of the *IEEE International Conference on Systems, Man and Cybernetics*, (Taipei, Taiwan), IEEE SMC, pp. 1676–1681, 8 - 11 October 2006.

- S. Mohamed, D. Rubin, and T. Marwala,"Adaptive GPCR Classification Based on Incremental Learning" in Proceedings of the *17th Annual Symposium of the Pattern Recognition Association of South Africa*, (Parys, South Africa), PRASA, pp. 121 – 126, 29 November – 1 December 2006.

The full papers appear at the end of this dissertation.

Nomenclature

CNS Central Nervous System

DNA Deoxyribose Nucleic Acid

FAM Fuzzy ARTMAP

GA Genetic Algorithm

GLM Generalised Linear Model

GPCR G-Protein Coupled Receptor

MLP Multilayer Perceptron

PDB Protein Data Bank

RBF Radial Basis Function

RNA Ribonucleic Acid

SCOP Structural Classification of Proteins

SVM Support Vector Machine

XML Extensible Markup Language

Chapter 1

Modelling and Classification in Proteomics

1.1 Introduction

Proteomics, the science of studying proteins and protein interactions [1] plays a vital role in modern biology. An area of great importance in proteomics is the classification of proteins into families or other biologically significant groupings. This science allows us to classify known proteins, to predict the families of new proteins and allows the structural and functional properties of proteins to be inferred, giving us a deeper understanding of how proteins function in making up the living cell. More practically, this science is a driving agent in the discovery of new drugs and drug therapies for the treatment of various diseases [2]. Bioinformatics is the area of research that has developed many of the tools which allow for this classification of proteins into families as well as the development of a wide range of other computational tools for modelling of DNA, RNA, interaction pathways between various molecules, and the effect these compounds and interactions may have in the body.

The benefits of a computational analysis of biological systems is most clear when analysing the process of drug design. The development of new drugs often takes up to 15 years and costs up to $700 million per drug under investigation [1]. This drug design consists of two phases: a discovery phase and testing phase. The drug discovery phase is further broken up into the target identification, lead discovery and

optimisation, toxicology and pharmacokinetic phases [3]. It is in the drug discovery phase that computational tools have had the most impact. In pharmaceutical drug discovery programs it is often useful to classify the sequences of proteins into a number of known families. In mathematical notation, if it is known that a protein sequence $\mathcal{S}$ is obtained for some disease $\mathcal{X}$, and that $\mathcal{S}$ belongs to family $\mathcal{F}$, treatment for the disease is initially determined using a combination of drugs that are known to apply to $\mathcal{F}$ [4].

Consider the example of the HIV protease, a protein produced by the human immunodeficiency virus. The target identification stage involves the discovery of this HIV protease and the identification of this protein as a disease causing agent. The objective of drug design is to design a molecule that will bind to and inhibit the drug target. A great deal of time and money can be saved if the effect of molecules can be determined before these molecules are actually synthesised in a laboratory. Bioinformatics tools are used to predict the family to which a newly designed drug belongs, and hence allows the structure and function of the drug to be inferred, and to determine if it will have any effect on the drug target.

This dissertation focuses on the problem of protein sequence classification. The usefulness of this classification, if not applied for drug discovery processes, at least allows proteomics data to become more organised, being more valuable in this form for use by the greater research community. Incremental learning will be introduced as a key tool in the classification of proteins sequences into families.

1.2 Protein Classification and Current Methods

The problem of protein family classification remains a core problem in computational biology, due mainly to the complex nature of interactions between the amino acids residues which form the protein primary structure. The completion of the sequencing of the human genome and other sequencing projects, has resulted in the accumulation of a large volume of sequence data. This data cannot be fully utilised until an understanding of the function and structure of the sequences is obtained and their role in biological processes is understood. An important step in this regard

has been the introduction of many protein databases that organise protein data into a number of families and super-families based on structure such as SCOP [5] and CATH [6], among others.

The classification of protein sequences into sets of known families is thus important as a means of probable function or structure assignment to hypothetical or uncharacterised proteins [7], and as mentioned, this family classification is also used widely in pharmaceutical drug discovery programs. A number of advantages of family classification have been identified [8] as a basic approach to large scale genomic annotation:

1. It improves the identification of proteins that are difficult to characterise based on pairwise alignments;
2. It provides an effective means of attaining relevant biological information from vast amounts of data;
3. It reflects the underlying gene families, which is essential for comparative genomics and phylogenetics.

Two broad approaches to the classification of proteins into families can be considered. The first approach is based on analysing protein sequences using "stringology" [9], the science of analysing strings. This approach is involved with the comparison and alignment of strings, the analysis of protein sequences using regular expressions, the creation of grammars and the development of phylogenetic trees, and has been the subject of considerable research. Many of the now standard tools used for protein or nucleotide sequence analysis are based on these systems, the Basic Local Alignment and Search Tool (BLAST), being the most common among them [10].

1.3 Problem Statement

Evolutionary science has now given us a deep understanding of the evolution and development of proteins and has highlighted some deficiencies of the string based approache to structure and function annotation. The major concern being the assumption of a particular order in which characters of the string should appear,

which goes against common evolutionary knowledge. In light of this problem, this dissertation aims to explore a number of problems, which are:

1. To consider the applicability of various machine learning tools to the solve the problem of protein family classification, which do not make assumptions regarding the order of amino acids in a protein sequence;
2. To consider parallels between biological and machine learning strategies, apply considerations based on this understanding in the design of learning architectures and to provide a mathematical and empirical context to these learning paradigms;
3. To introduce the fuzzy ARTMAP as an alternate machine learning tool for the classification of protein primary structures into families; and
4. To introduce the area of incremental learning to the protein bio-sequence analysis community, an area which up to this point has not been considered.

The approach followed comes as a result of the increased realisation and penetration of the science of machine intelligence in computational biology. This approach is based on the methodologies and tools from the science of Computational Intelligence.

1.4 An Approach using Computational Intelligence

Computational intelligence is a term which has become synonymous with many terms including artificial intelligence, machine learning and pattern recognition. This term includes these areas of research as well as other fields such as data mining and evolutionary optimisation. Computational intelligence is simply the development and use of tools which simulate in some way, learning and development similar to that exhibited by the human brain. This research uses a computational intelligence approach to the classification of protein sequences into families. In general, a system of this nature follows the steps that are outlined in figure 1.1.

The data acquisition, feature generation, pattern classification and performance

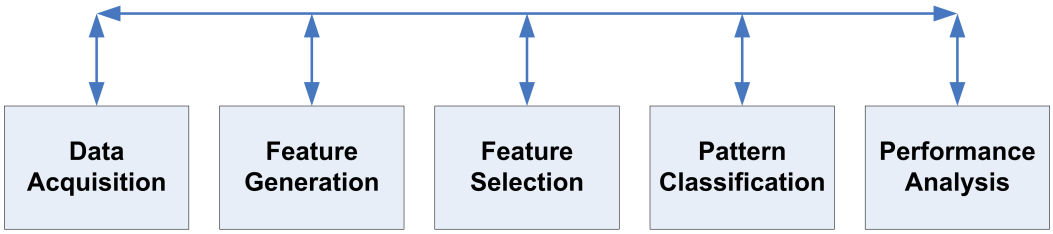


Figure 1.1: General approach to the design and implementation of a pattern recognition system.

analysis stages shown in this figure are followed in this research. This process describes the process of obtaining data from a number of sequence repositories, the processing of these sequences in some appropriate manner, the design of a system that will classify the extracted and processed sequences, and an analysis of how this system performs in the classification task. Each of these stages will be discussed in detail with specific focus on the design of a classification system, which is able to classify proteins into a large number of families. The classification system will also have the ability to include new data into the classification model. This ability of a classification system to learn new data is generally known as incremental learning. Incremental learning is a difficult computational problem and is the major focus of this dissertation.

1.5 Outline of the Dissertation

Having sketched a brief introduction to the problem being analysed, the remainder of this dissertation is structured as follows:

Chapter 2: Provides the necessary biological understanding relating to protein science. This section describes proteins from the fundamental building blocks and describes the approaches to sequencing proteins and the current techniques for classifying proteins.

Chapter 3: Discusses the fundamentals of machine learning and various popular machine learning tools such as Neural Networks and Support Vector Machines. This chapter also provides a formal definition of incremental learning and discusses the properties of incremental learning.

Chapter 4: Discusses two approaches to incremental learning of protein data. The first algorithm is based on an evolutionary strategy and combines aspects of evolution and diversity to form an incremental system based on the fuzzy ARTMAP. The second approach uses a recently introduced technique, Learn++, and looks at the suitability of this technique for the analysis of protein data.

Chapter 5: This concluding chapter compares the two different approaches discussed in chapter 4, and the merits and drawbacks of each. Concluding remarks concerning incremental learning of protein data are made and suggestions for further research are presented.

Appendix A: This appendix provides some additional properties of proteins that were mentioned in the main body, but did not warrant detailed discussion there. These aspects are included here for completeness and to give additional understanding of those concepts that were touched on.

Appendix B: This appendix presents in detail the principles and operation on which the fuzzy ARTMAP is built upon. It discusses the mathematical formulation of the Fuzzy ARTMAP and how this system is able to perform incremental learning and classification.

Appendix C: This section discusses in detail the underlying algorithm of Learn++.

Appendix : This appendix provides references to the two papers that were published in relation to this thesis.

Glossary: This section gives definitions for many of the terms which are used in this thesis.

The main contributions of this thesis are the *introduction* of incremental algorithms to protein analysis, presented in chapter 4. A review of the literature currently available has shown that this work is the *first* use of the fuzzy ARTMAP for protein analysis and the consideration of how it can be extended and applied to the problem of incremental learning of protein families. The Learn++ algorithm is also applied for the first time in this type of research. This work is also important as it considers the case of multi–class classification using all-vs-all methods as opposed to the more

common one-vs-all multi–class problem, which is usually considered in protein analysis using machine learning. Chapter 2 provides the reader not familiar with protein science, with a brief introduction to the key concepts of molecular biology, which must be kept in mind when designing and analysing protein classification systems. Chapter 3 has its value as a concise overview of biological and machine learning algorithms. The biological interpretations have been aggregated from multiple sources and presents a thorough overview of learning in biological systems and the equivalent representation in machine systems. It is the intention of the author to present the research such that it is understandable and useful by researchers from many diverse fields such as information engineering, molecular biology and computer science.

Chapter 2

Protein Sequencing and Approaches to Classification

2.1 Introduction

This chapter aims to introduce the reader not familiar with aspects of protein science to the fundamentals of molecular biology. The chapter discusses the basic chemical and biological aspects of proteins, their role in biological systems and how these proteins are sequenced and represented. The rapid development in protein discovery is also presented and helps us understand the need for incremental learning in bioinformatics problems. The current techniques for the problem of protein classification are also discussed to allow contextualisation of our approach to the existing tools and techniques.

The "central dogma of molecular biology" shown in figure 2.1 was introduced by Crick in 1958, and is a philosophy that describes the transformation of DNA into RNA and then into proteins [11]. The process of transforming DNA into RNA is known as transcription, and that of transforming RNA into a protein is known as translation. This dogma describes a flow of information in biological systems. Bioinformatics gives an understanding of the principles behind this dogma, and is an area of science that focuses on using computational tools to understand the formation and interaction of DNA, RNA and proteins with each other and with other

biological systems. The fundamentals of DNA and RNA are similar to proteins and the books by Clote and Backofen [11] and Baxevanis and Oulette [7], are excellent references for background in these areas. This research focuses on proteins, which will be described in more detail.

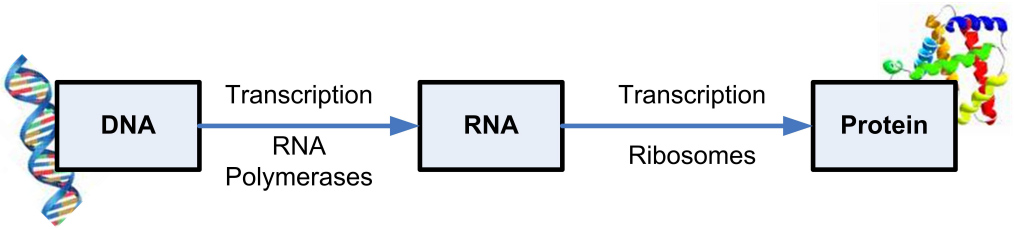


Figure 2.1: The central dogma of molecular biology. Information passes from DNA to RNA to Proteins. RNA polymerases and ribosomes aid in the transformation of information from one form to another.

2.2 Molecular Biology Primer

Proteins are the compounds in the biological system which do the work of changing cell chemistry and acting as catalysts. The *structural proteins* provide structural support and infrastructure for cells, as well bones and connective tissue, such as collagen. *Enzymes* act as biological catalysts making the chemical reactions necessary for life possible. In general, proteins are the sensors that allows us to see (like Rhodopsin) and smell, the detectors that allow us to form an immune response to invading cells, and the signals for intercellular communication (like Insulin) [12]. A proteins structure is described on four different levels:

Primary Structure The protein is described by its fundamental building blocks called Amino Acids.

Secondary Structure Describes the regions in the primary structure where secondary structure elements such as helices and flat planes or sheets (α helices and β sheets) occur.

Tertiary Structure The 3-dimensional structure of a protein domain, i.e. if a protein consists of several protein subunits, then the tertiary structure describes the structure of a subunit.

Quaternary Structure The 3-dimensional, native structure of the fully functional protein.

The four descriptions of the protein structure are shown diagramatically in the figure 2.2.

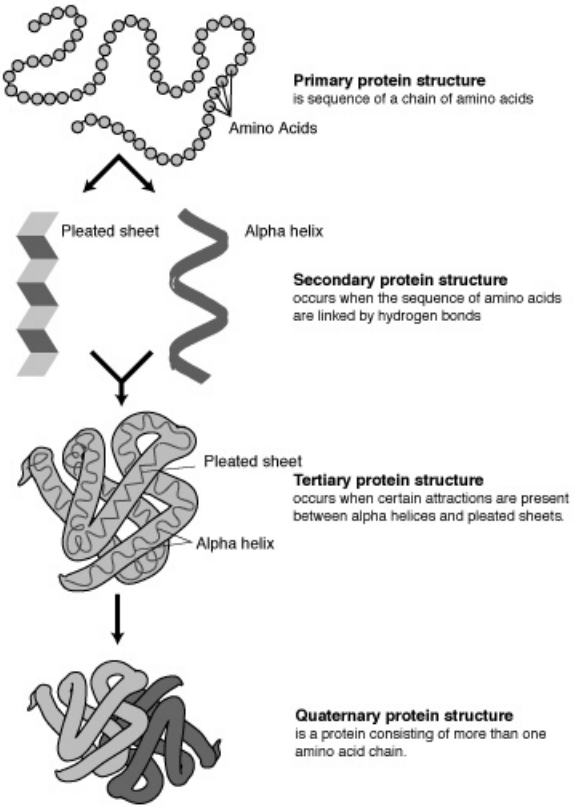


Figure 2.2: Description of a protein in primary, secondary, tertiary and quatenary structures [13].

All proteins are comprised of a set of about 20 naturally occurring building blocks known as amino acids [1]. All amino acids have the same base chemical structure which consists of a central carbon atom with a R-group, which is different for each of the 20 amino acids. The chemical structure of the general amino acid is shown in figure 2.3 with the `COOH` part of the amino acid known as the carboxy terminus and the NH_2 part known as the amino terminus [3].

The International Union of Pure and Applied Chemistry (IUPAC) three-letter and single-letter abbreviations for each of the 20 possible R-groups are given in table A.1

[1] There are actually now 22 known amino acids. Refer to appendix A for more details.

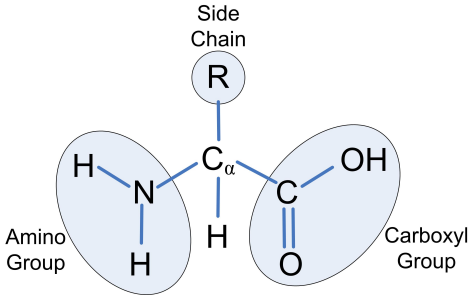


Figure 2.3: Chemical structure of the generic amino acid. The Amino Group, the Carboxy Group and the alpha carbon are identical for all 20 amino acids while each has its own distinctive R group. Redrawn from [3]

in Appendix A [14]. The letters B and Z in the table are ambiguous characters and are sometimes included by computers programs to indicate the end of the protein sequence [11].

Given this notation for amino acids, a protein is represented by a string consisting of letters of the amino acid alphabet. A typical example of a protein primary structure is shown in figure 2.4 [5]. This is the amino acid sequence representation of a part of haemoglobin, specifically the alpha chain.

```
SLTKTERTIIVSMWAKISTQADTIGTETLERLFLSHPQTKTYFPHFDLHPGSAQLRAHGSKVVAAVGDAVK
SIDDIGGALSKLSELHAYILRVDPVNFKLLSHCLLVTLAARFPADFTAEAHAAWDKFLSVVSSVLTEKYR
```

Figure 2.4: Amino acid sequence of α-chain Haemoglobin

After translation, the protein does not remain in the form of a simple linear chain. The protein undergoes post-translational modifications and folding and forms a complex structure. The final 3-dimensional structure is determined largely by the order in which the amino acids are assembled in a protein [3]. Many proteins may also have the same final quatenary structure with a different primary structure.

It is common to find a large number of proteins with slightly different primary structures, but which perform the same function. These changes in the primary structure are the result of molecular evolutionary processes such as selection and inheritance. The similarities and differences between protein primary structures gives an indication as to the structure and function of these proteins. Proteins which are related to each other are said to be homologous, but this is not always true, as in

the case of divergent evolution of proteins [12]. Many proteins that have similar structures evolve from a common ancestor and then undergo changes which result in different primary structures, but with similarity to the original structure. This is known as divergent evolution. Point mutation is the best known mechanism of molecular evolution and implies either the addition, deletion or substitution of an amino acid.

The scientific process used in molecular biology begins with the identification of tissue which is to be analysed. This could be human tissue from the kidney or the liver for example. The proteins which are active in these areas are then isolated and identified using a number of sequencing techniques, to obtain the protein primary or secondary structure. The protein primary structure could then be compared with a number of other well known proteins to detect any sequence similarity, which if found could allow the biologist to infer the final structure or function of the protein. This structure or function is used to aid in the analysis of bio-molecular and signalling pathways in the body, in understanding the operation of organs and proteins in the body, and can also be used in drug discovery processes to discover active and binding sites of the proteins to inhibit the action of diseases. This scientific processes will be discussed in more detail in the sections that follow, giving a brief insight into molecular biology and the applications in bioinformatics.

2.3 Sequencing and Protein Databases

2.3.1 Protein Sequencing

The raw data that is used in the computational analysis of proteins are generated using a small set of techniques. In **gel electrophoresis**, fragments of protein (or DNA or RNA) are placed in a gel-like matrix, and thereafter an electric field is applied across the ends of the gel. The field causes the molecules to accelerate resulting in positively charged molecules moving toward the negative electrode and vice versa for negatively charged molecules [12]. The time it takes the molecules to move across the gel depends on the size and the charge of the molecule. The movement of the

molecules in the gel matrix creates bands which, when appropriately stained allows a protein to be compared with other proteins or to sort proteins according to chemical properties such as their isoelectric point (pI) [15]. Figure 2.5 shows a typical image of the result of a gel electrophoresis. A process known as **blotting** is then usually applied to further process individual bands in the gel matrix. **Mass spectrometry** is also applied to the protein electrophoresis and allows unique identification of proteins by accelerating individual spots in the electrophoresis through a charged tube and then analysing the resulting peptide mass fingerprint [3].

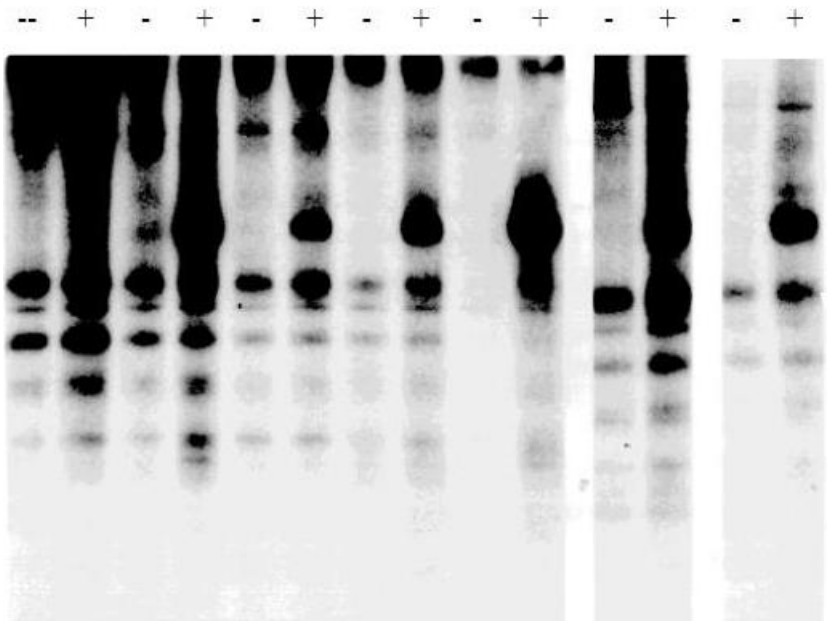


Figure 2.5: Example of a gel electrophoresis run. The mixtures in each column are separated vertically according to their charge. The dark bands indicate the presence of a particular sized or charged molecule in the mixture [12].

The structural characteristics of proteins are primarily gathered by **x-ray crystallography**. This process essentially grows a pure crystal of the protein, and uses the diffraction pattern of x-rays applied to the crystal to indirectly determine the position of the molecules in the protein. This process, which may take years to complete, may not be possible for certain proteins. This is one of the major driving forces behind computational 3-dimensional modelling techniques. An alternate approach to x-ray crystallography is multidimensional **nuclear magnetic resonance** (NMR), which does not require crystallisation, but which also has difficulties associated with imaging large molecules such as proteins. Details concerning these techniques as well as others can be found in [3, 12, 15, 16].

2.3.2 Representation and Storage of Protein Data

The data obtained for proteins after sequencing is stored in a number of different formats for use with a computer. In general the files are stored as ASCII format text files. These files are interoperable across all operating systems and platforms and there are many inexpensive and free editors to create these files. The ASCII files though may have a unique formatting, with the structure of the text files such as spacings between values and individual lines having different meanings depending on the formatting convention used. Some of the more common formatting conventions include the European Molecular Biology Laboratory Library format (EMBL), SwissProt format, FASTA format or Extensible Markup Language (XML) format [16]. Each format is associated with a particular repository of protein sequences and data. The XML format is the latest format which allows data to be stored in ASCII format but with self describing fields in the text file, so that one need not have prior knowledge of a particular file structure before using it. This format will become popular as molecular biologist who actually create these files become more aware of the benefits of this technology, which is already prevalent in other fields of software and network engineering.

Many publicly available databases exist where molecular biologists and information engineers may gain access to up-to-date protein data. Some of the databases associated with protein data in particular are listed in table 2.1.

Table 2.1: Various publicly available protein databases on the Internet. Database names in bold are used in this research.

Database Name	URL
RefSeq	`http://www.ncbi.nlm.nih.gov/refSeq`
UniProt	`http://www.uniprot.org`
Protein Information Resource (PIR)	`http://pir.georgetown.edu`
Protein Data Bank (PDB)	`http://rcsb.org/pdb`
SwissProt (ExPASy)	`http://www.expasy.org/sprot`
SwissProt (EBI)	`http://www.ebi.ac.uk/swissprot`
Structural Classification (SCOP)	`http://scop.mrc-lmb.cam.ac.uk/scop`
Class Architecture (CATH)	`http://www.biochem.ucl.ac.uk/bsm/cath`
Dali Domain Dictionary (DALI)	`http://www.ebi.ac.uk/dali`
G-Protein CR Database (GPCR)	`http://www.gpcr.org/7tm/`

Each of the databases listed are arranged differently. Some are universal protein databases and cover proteins from all species, whereas others concentrate on a particular protein family, organism or group of proteins [7]. For both types of database, the data can be either fully automated when stored, or stored only after manual intervention and curation to enhance the information that is stored about a protein.

Three protein databases are used in this research, with each using a different format for the text file that is used. The three file formats are the FASTA, EMBL and PDB formats, used for data obtained from the SCOP, GPCR and PDB databases respectively. The FASTA format is a minimalist format giving the amino acid sequence along with some basic information such as version numbers and IDs for the same protein in other databases. The EMBL format is a more detailed format which, apart from supplying the amino acid sequence gives information such as the journal where the protein discovery was first reported, PUBMED IDs, date submitted, protein family and other comments. The PDB file structure is also very simple, and contains the amino acids and the relative position in three-dimensional space of each molecule in the protein. Examples of these file formats will be given in the discussion that follows.

2.3.3 Overview of SCOP

One database that is used in this research is the Structural Classification of Proteins (SCOP) Database [5]. This database hierarchically organises proteins according to their structure and evolutionary origin. It is a database which undergoes manual curation and allows us to understand the structure of many proteins and the relationship of these proteins to other proteins. This ultimately allows us to gain insight into the function of a protein in general and specific terms relating to it, in order to understand its evolutionary history [17].

SCOP is a hierarchically structured database with classifications for Class, Fold, Superfamily, Family and Domain. The unit of categorisation is the protein domain with protein domains typically representing the units of protein evolution, structure, and function.

SCOP *families* consist of clear common evolutionary origin as evidenced by extremely similar structure and function. *Superfamilies* consist of families whose proteins share very common structure and function and give reason to believe that they are evolutionarily related. *Folds* consist of superfamilies that have similar core structural topologies. Folds are grouped into one of four *classes* depending on the type of secondary structure elements which are prevalent [17]. An example of the structural arrangement from the SCOP database is shown below indicating the hierarchy of classifications.

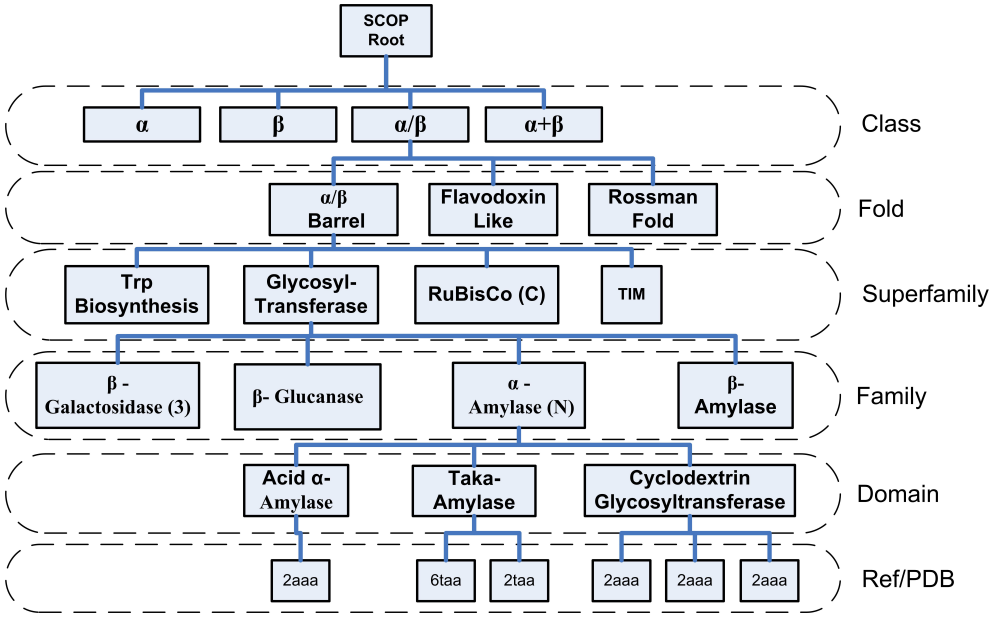


Figure 2.6: SCOP hierarchy showing the different levels of classification in the database with some representative groupings of each level in the database. Redrawn from [17].

The SCOP database is accessible as a set of text files (ASCII format). These files only consist of the protein classifications, but do not contain the actual protein sequences themselves. A sample of the SCOP database file name `dir.des.scop.txt` is given in figure 2.7, and shows the hierarchical labelling of proteins for each of the SCOP groupings, identified by the SCOP ID.

Each of the IDs given in the file can be matched against the same ID which appears in an associated database of protein sequences, the ASTRAL database [18]. This database is partially derived from the SCOP database. A short listing of the sequences in this database is shown in figure 2.8.

```
46456 cl a -All alpha proteins
46457 cf a.1 -Globin-like
46458 sf a.1.1 -Globin-like
46459 fa a.1.1.1 -Truncated hemoglobin
46460 dm a.1.1.1 -Protozoan/bacterial hemoglobin
46461 sp a.1.1.1 -Ciliate (Paramecium caudatum)
14982 px a.1.1.1 d1dlwa_ 1dlw A:
100068 px a.1.1.1 d1uvya_ 1uvy A:
46462 sp a.1.1.1 -Green alga (Chlamydomonas eugametos)
14983 px a.1.1.1 d1dlya_ 1dly A:
100067 px a.1.1.1 d1uvxa_ 1uvx A:
81667 sp a.1.1.1 -Cyanobacteria (Synechocystis sp.), pcc 6803
...
```

Figure 2.7: Example of text file showing the SCOP classification.

```
>d1dlwa_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin ...
slfeqlggqaavqavtaqfyaniqadatvatffngidmpnqtnktaaflcaalggpnawt
grnlkevhanmgvsnaqfttvighlrsaltgagvaaalveqtvavaetvrgdvvtv
>d1dlya_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin ...
slfaklggreaveaavdkfynkivadptvstyfsntdmkvqrskqfaflayalggasewk
gkdmrtahkdlvphlsdvhfqavarhlsdtltelgvppeditdamavvastrtevlnmpq
q
>d1s69a_ a.1.1.1 (A:) Protozoan/bacterial hemoglobin ...
stlyeklggttavdlavdkfyervlqddrikhffadvdmakqrahqkafltyafggtdky
dgrymreahkelvenhglngehfdavaedllatlkemgvpedliaevaavagapahkrdv
lnq
```

Figure 2.8: Sample FASTA file format extracted from the ASTRAL database.

This data is in the FASTA format, with the first line as a header line and the remaining lines being the actual protein sequence. For example, given the protein ID `a.1.1.1`, it is a simple procedure to map the first sequence in the ASTRAL database to the SCOP classification as belonging to the `Truncated Haemoglobin` family shown in figure 2.7.

Sequence databases are continually in a state of change. With rapid technological development, it is becoming easier to sequence newly discovered and synthesized proteins. This rapid discovery is represented in the continual change in the size of the many protein databases. The graph of figure 2.9 [19] shows the growth of a number of well established protein and genome sequence databases over the past 26

years. The graphs show that there is a steady increase in the number of sequences year–on–year. Looking at the Protein Data Bank (PDB), the number of sequences in the database per year has changed from in the order of 10^2 in 1980 to a number of entries in the order of 10^4 in 2006 – a 100% increase in the size of the database in 26 years.

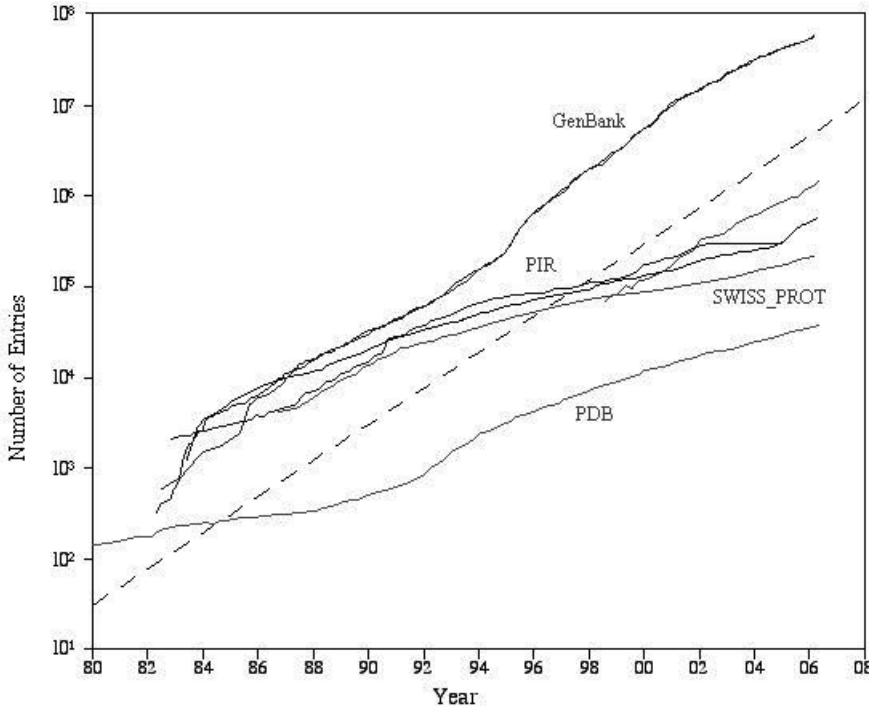


Figure 2.9: Growth in the number of entries in various protein sequence databases [19].

Figure 2.10 shows the daily growth of a number of sequence databases. The growth is the number of sequences submitted daily and is averaged over a 6 month period, up to August 2006. The graph shows that protein databases such as SwissProt and PIR have a slow increase in the number of sequences which are submitted, and which contributes to the yearly growth in database size that appears in figure 2.9. Both these graphs highlight the dynamic nature of protein discoveries and the importance of any system which deals with this data to react to this increase in the amount of data. This dissertation presents tools to deal with this inherent nature of protein science.

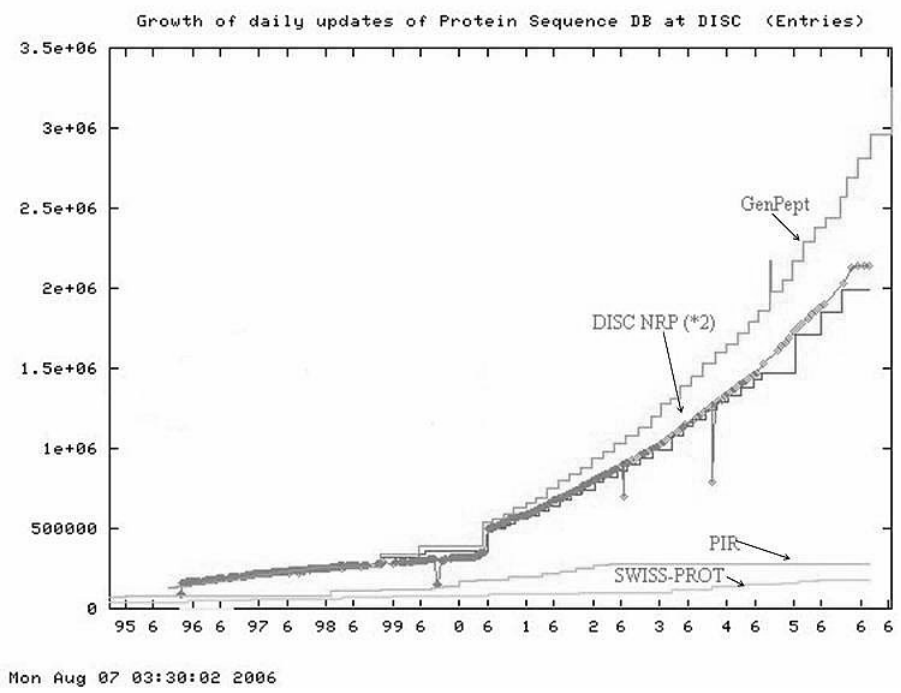


Figure 2.10: Average daily growths experienced by sequence databases, sampled over 6 month intervals [20].

2.3.4 Overview of GPCRDB

The G-Protein Coupled Receptors (GPCRs) are a superfamily of proteins and forms the largest superfamily found in the human body. The GPCRDB is a database dedicated to the storage and annotation of G-Coupled proteins and at present consists of 16764 entries [21]. GPCRs play important roles in cellular signalling networks in processes such as neurotransmission, cellular metabolism, secretion, cellular differentiation and growth and inflammatory and immune responses [22]. Because of these properties, the GPCRs are the targets of approximately 60% – 70% of drugs in development today [23], 50% of current drugs on the market and approximately 20% of the top 50 best selling drugs target GPCRs. This results in greater than US$23.5 billion in pharmaceutical sales revenue from drugs which target this superfamily [23]. GPCRs are associated with almost every major therapeutic category or disease class, including pain, asthma, inflammation, obesity, cancer, as well as cardiovascular, metabolic, gastrointestinal and CNS diseases [24]. This obvious importance of the GPCRs is the reason they are used in this research.

The key features of the GPCRs are that they share no overall sequence homol-

ogy and have only one structural feature in common [22]. The GPCR superfamily consists of five major families and several putative families, of which each family is further divided into level I and then into level II subfamilies. The extreme divergence among GPCR sequences is the primary reason for the difficulty of classifying these sequences [25], and another important reason as to why they are used in this research.

In this research eight GPCR families are considered from the number of families available in the GPCRDB. The GPCR sequences are stored in the EMBL format, which consists of a number of labelled fields considering aspects of a sequence such as identifiers in a number of databases, the date of discovery and relevant publications dealing with the protein sequence. The database itself is updated every three to four months. A sample of a file from this database is shown in figure 2.11 showing the fields such as journal and author that are included in the file.

```
D   5HT1A_FUGRU    STANDARD;      PRT;   423 AA.
DT   01-MAY-2005 (Rel. 47, Last annotation update)
DE   5-hydroxytryptamine 1A-alpha receptor (5-HT-1A-alpha) (Serotonin
DE   receptor 1A-alpha) (5-HT1A-alpha) (F1A).
OC   Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
RX   MEDLINE=97361762; PubMed=9218723; DOI=10.1016/S0378-1119(97)00064-4;
RA   Yamaguchi F., Brenner S.;
RT   "Molecular cloning of 5-hydroxytryptamine (5-HT) type 1 receptor genes
RT   from the Japanese puffer fish, Fugu rubripes.";
RL   Gene 191:219-223(1997).
CC   -!- SIMILARITY: Belongs to the G-protein coupled receptor 1 family.
DR   EMBL; X95936; CAA65175.1; -.
...
SQ   SEQUENCE   423 AA;  47001 MW;  7B1308626B40190F CRC64;
     MDLRATSSND SNATSGYSDT AAVDWDEGEN ATGSGSLPDP ELSYQIITSL FLGALILCSI
     FGNSCVVAAI ALERSLQNVA NYLIGSLAVT DLMVSVLVLP MAALYQVLNK WTLGQDICDL
     FIALDVLCCT SSILHLCAIA LDRYWAITDP IDYVNKRTPR RAAVLISVTW LIGFSISIPP
     MLGWRSAEDR ANPDACIISQ DPGYTIYSTF GAFYIPLILM LVLYGRIFKA ARFRIRKTVK
     KTEKAKASDM CLTLSPAVFH KRANGDAVSA EWKRGYKFKP SSPCANGAVR HGEEMESLEI
     IEVNSNSKTH LPLPNTPQSS SHENINEKTT GTRRKIALAR ERKTVKTLGI IMGTFIFCWL
     PFFIVALVLP FCAENCYMPE WLGAVINWLG YSNSLLNPII YAYFNKDFQS AFKKILRCKF
     HRH
//
```

Figure 2.11: Sample EMBL file format extracted from the GPCR database.

2.3.5 Overview of PDB

The Protein Data Bank (PDB) is the international repository of information about the three dimensional structures of biological macromolecules [26] and the contents of the database are obtained via X-ray crystallography and Nuclear Magnetic Resonance (NMR) imaging. This database was used to obtain amino acid sequences of enzymes. Enzymes are also biologically important macromolecules, which are responsible for catalysis in biochemical reactions. These biochemical reactions include several vital functions such as *metabolic processes* that convert nutrients into energy, *biosynthesis* for the creation of new molecules, removal of toxic foreign chemicals by *detoxification*, and *information storage* by the processing of DNA [27].

Enzymes are grouped into six classes based on the types of reaction that they catalyse. These six classes are the *Oxidoreductases, Transferases, Hydrolases, Lysases, Isomerases*, and *Ligases* [3] and these enzymes are named and grouped according to the Enzyme Classification (EC) system, `EC1.*.*.*`, ..., `EC6.*.*.*` [7]. This data is extracted in the PDB format, and as for the previous two databases discussed, the primary structures were extracted.

2.3.6 Sequence Extraction and Preprocessing

The current version of the SCOP database 1.69 [5] and the corresponding ASTRAL database [18] with less than 95% sequence identity is used in this work. This choice of sequence identity is used commonly by researchers in this field, and for this reason has been used. Here, as a result of removing the highly redundant sequences from the ASTRAL database, the corresponding SCOP entries had to be extracted and matched against the sequence ID in the ASTRAL database. The SCOP database has just over 70,000 entries, while the ASTRAL file has about 12,000 entries, resulting in the matching process between the two databases being a lengthy procedure.

Once this matching is completed, "outlier" sequences must be removed. Two types of outliers were observed from the available databases. The first is that there exists certain ASTRAL sequences for which there is no corresponding SCOP entry, which

is due to the fact that the ASTRAL database is only partially derived from the SCOP database. There were a total of 345 such sequences and these were removed from the sequence list. The second type of outlier consists of sequences which have characters which are not part of the standard 20-letter amino acid alphabet — the letters are B, Z and X. The letter B is used as either Aspartate or Asparagine, if the specific amino acid not determined during the sequencing process. Similarly the letter Z is either Glutamate or Glutamine if not determined [14]. The letter X is is used to represent an unknown amino acid. As mentioned, these characters are also used by some computer programs to indicate the start and end of a protein sequence [11]. From the data, 120 such sequences were observed and since they are a small number of sequences, they are removed from the sequence list.

The remaining sequences form the final database of sequences that is available for use in the classification system. The sequences consist of 2784 families. These families each consist of varying numbers of sequences per family. The histogram of figure 2.12 shows the distribution of the number of sequences per family in the database. The histogram shows that most of the families have a very small number of sequences per families. The lower graph shows the distribution of the first 2700 families of the database, which have been sorted in ascending order. This graph shows that most families have less than 20 sequences per family. This small amount of data limits the use of these families in later stages of the work that will be described, and will influence which families are selected for use in the classification system, which generally requires large amounts of training data.

Unlike the data obtained from the ASTRAL database which has many sequences in one text file, the GPCR EMBL formatted data consists of only one protein per EMBL file. The sequences from these many files are extracted and stored together, grouped by their GPCR families. Similar preprocessing involving the removal of sequences with non-standard characters was performed. The enzyme data obtained from the Protein Data Bank is stored in the PDB format. The data in these files also consists of a single protein per PDB file. Again, the sequences were extracted from these files and stored together according to enzyme family.

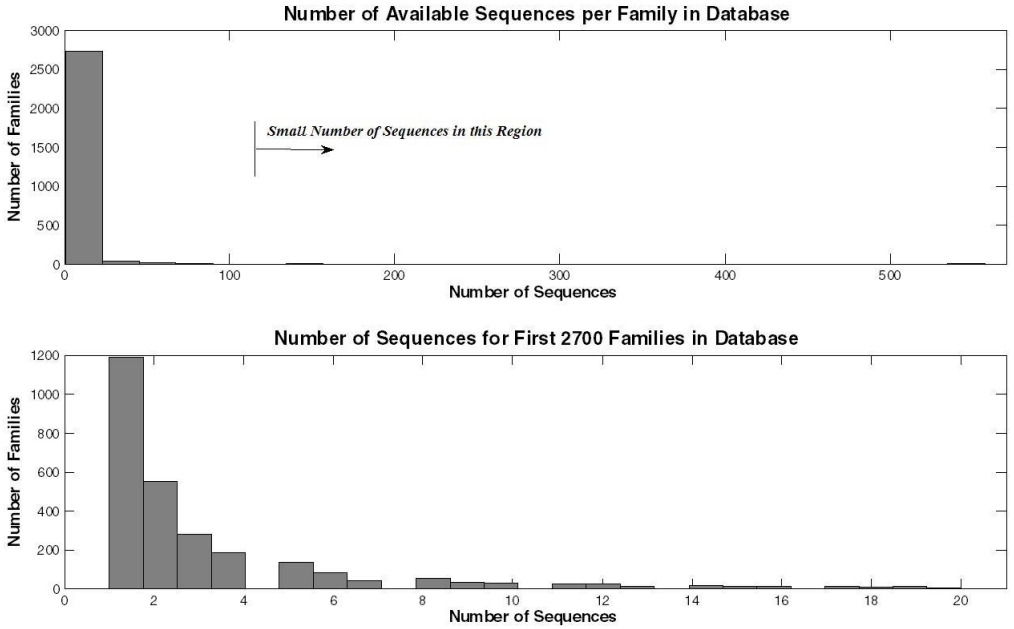


Figure 2.12: Distribution of the number of sequences per family. The distribution is skewed to a low number of sequences per family, with most families having less than 20 sequences per family.

2.4 Sequence Alignment Techniques for Classification

Sequence alignment is the procedure of comparing two (pair-wise alignment) or more (multiple alignment) DNA or protein sequences by searching for a match between characters or groups of characters in each sequence [16]. The degree of similarity is described by a fractional value and there exits three categories of computational methods to perform these alignments.

The simple or **pairwise alignments** determine similarity by aligning a query sequence with every other sequence in a sequence database using an amino acid similarity matrix. Smith–Waterman [28] and Needleman–Wunsch algorithms [29] are dynamic programming techniques that find optimal local and global alignments respectively. Once an optimal alignment is determined, a scoring matrix is used which allows us to determine the degree of similarity between the aligned sequences. While the algorithms are efficient in determining the optimal alignment between two sequences, it becomes computationally infeasible for use in a database-wide search. This problem though has been overcome by a number of heuristic database search techniques such as BLAST [10] and FASTA [30], which have become more prevalent

and efficient for database-wide searches.

The **multiple alignments** search against a database of known sequences by first aligning a set of sequences from the same protein superfamily, family or subfamily and creating a consensus sequence to represent the particular group. The query sequence is then compared against each of the consensus sequences using a pairwise alignment. The query sequence is classified as belonging to the group with which it has the highest similarity score [25]. Some popular techniques for performing multiple sequence alignments are Position Specific Scoring Matrices (PSSM) [31] and ClustalW [32]. The third category uses **profile Hidden Markov Models** (HMMs) as an alternative to the consensus sequences, but is otherwise identical to the multiple alignment technique. The focus of this research is not on alignment based techniques and thus they are not described in detail here. The alignment based techniques are described in detail in [1, 3, 7, 16].

2.5 Problems with Alignment Based Techniques

Many shortcomings have been identified with respect to the effectiveness of sequence alignments, which is the reason why these techniques are not considered here. The principle argument against sequence alignment is the assumption that the order of homologous segments is conserved [33]. This assumption contradicts accepted understanding that evolution causes genetic recombination and reshuffling of nucleotides and amino acids [34]. The other argument lies in the lack of computational efficiency of the approaches.

This has led to the development of so called "alignment–free" techniques. These techniques rely mainly on machine learning approaches [35] and the application of Information theory, Kolmogorov complexity and Chaos theory [33]. Popular machine learning tools that have been applied to problems in protein classification include the Multi–layer Perceptron neural networks [36, 37], Support Vector Machines [38, 39], k-Nearest Neighbour Classifiers [40] and Naïve Bayes Classifiers [25], among others.

A pattern recognition approach is adopted in this research to classify protein primary structures into a number of primary and putative families. The pattern recognition approach allows the time complexity to be limited to the initial training procedure and does not make any assumptions as to the order of homologous segments of a protein. The theoretical foundations of the pattern recognition approach and the machine learning tools that form the basis of the work presented will be described in detail in the next chapter.

Chapter 3

Biological and Machine Learning Systems

3.1 Introduction

The use of machine learning tools have become central to the application of the alignment free classification techniques that were described in the previous chapter. This chapter aims to provide a brief description of some of the key machine learning tools that are used in this research and the principles of operation behind these computational techniques.

Machine learning is initially contextualised by reviewing concepts from neurobiology and the principles of human learning. Specific focus is given to the principles and types of *incremental learning*, which is the continuous learning that humans use everyday. The major machine learning tools, viz. Artificial Neural Networks, Support Vector Machines and the k–Nearest Neighbour Classifiers will be reviewed. The incorporation of incremental learning into these learning systems is then reviewed with a focus on the tools and techniques that are used in this research.

3.2 An Overview of Biological Evolution and Learning

The human Central Nervous System (CNS) is the key component of learning in man. The analysis of the development of the human CNS gives insight into the development of artificial learning systems and how biological learning can be mimicked. This understanding is obtained by analysing three phases of development [41]:

Evolution The development of the human cognitive system which has taken place over thousands of years due to to evolutionary processes such as inheritance and natural selection.

Neuro-development The establishment of individual brain structures during prenatal and the immediate post-natal phases.

Learning Adaptation of established neural systems and the adjustment of parameters such as the number of neurons and synaptic connections in response to environmental changes.

Each of these three phases gives designers valuable ideas for the development of artificial learning systems. The *evolutionary stage* occurs over a much larger time scale than the other two phases of development, and has been successfully implemented as optimisation strategies and applied to many engineering problems. Research in neurobiology has shown an evolutionary increase in the size of the isocortex which has resulted in the modularisation of functions associated with the isocortex. This modular structure which has been observed in the brain can easily be imitated in an artificial system, where we have *ensembles* of networks. Each network in this ensemble or committee is trained with the same data and the combination of their separate decisions is used to come to some final output decision. Alternatively, it is possible to have a number of artificial networks with each network focusing on a specialised subset of the available knowledge.

The *neuro–developmental* phase consists of the prenatal stage and the immediate postnatal stages, which consists of the the first few months after birth. During this phase, the human brain undergoes rapid development, with the creation of a large number of neurons and synaptic connections, which allows the newborn to survive

the initial phases of life. This stage is followed by a stage of rapid neuron death and a decrease in the number of synaptic connections, a process known as apoptosis. In artificial systems, this is implemented by creating learning models which are over–specified or over–designed, a technique commonly used in engineering design problems. The design of the learning system is then adapted using pruning techniques where extra artificial neurons are removed and the number of synaptic connections is decreased [42].

In terms of neurobiology, the *learning phase* sees the refinement of the existing neural structure, which was created during the neuro–developmental phase. This refinement and learning allows us to consider three types of learning which could be implemented in artificial systems. They are either [43]:

- *Supervised Learning* — where the learning system is taught or guided to the correct relationships between input and output data.
- *Reinforcement Learning* — where a reinforcement signal or reward is provided to the learning system when it correctly learns relationships between data, but it is not shown these relationships directly.
- *Unsupervised Learning* — where there is no external signal guiding the learning system and learning is achieved using clustering techniques. These can range from k-means clustering to more advance techniques such as hierarchical clustering.

In all three cases though, this learning can be represented mathematically, where for some input signal $x(t) \in \mathbb{R}^n$ and an output $y(t) \in \mathbb{R}^m$ for some time step $t \in \mathbb{N}$, the system learns a mapping $F : \mathbb{R}^n \supset X \to Y \subset \mathbb{R}^m$ or simply $x \longmapsto y$ is learned. The mapping F is usually some matrix of weight values in artificial learning systems.

Humans, through each of the three phases of development, adapt to the changing environment or the changing nature of the inputs which it receives. This type of learning is called **incremental learning**, which leads to the following definition:

Definition 1 (Incremental Learning) *An incremental learning systems consists of a standard learning scheme combined with a mechanism to allow for adjustments either in the structure of the learning system or its parameters or to changes in the presentation or constitution of its input signals [41].*

A standard learning scheme in the context of the human brain consists of the existence of neurons, synaptic connections and distinctive cortical regions in the brain which allow for the storage of information. The definition is very broad and allows a number of types of incremental learning to be explored, again brought about by the phases of human central nervous system development that have been discussed. Three types of incremental learning have been identified [41] :

Structural Incremental: Has been developed by considering both evolution and neuro–development, and implies a change in the structural or functional capacity of a learning system during learning.

Learning Parameter Incremental: A set of learning parameters are adapted during the learning phase.

Data Incremental: Developed from the learning phase, where data sets or its complexity is increased in steps during the learning.

A mixture of these cases usually exists, and will be demonstrated in later chapters of this dissertation. In general, this incremental learning system can be represented mathematically as the adaptation of a system characterised by a state λ_1, consisting of a configuration of neurons, synaptic connections and a set of information which it recognises, to a new state λ_2. The mapping which the system learns in each case is represented by a conditional probability distribution, where ω_i represent the classes or groupings of data that the learning system is initially trained to recognise and $\mathbf{x}$ represents the input data to the learning system, and ω_j represents the increased number of classes that the classifier is able to recognise after incremental learning.

$$P\left(\omega_i|\mathbf{x}\right)|_{\lambda_1} \quad \xrightarrow{incremental} \quad P\left(\omega_j|\mathbf{x}\right)|_{\lambda_2} \tag{3.1}$$
$$i = 1, \ldots, m; \quad j = 1, \ldots, n; \quad n > m.$$

Further insight into the comparisons between biological and machine learning can be found in the work by Chalup [41] and Carrier [44].

3.3 Machine Learning

Just as biological systems gain knowledge from their environment, machine learning systems gain knowledge in the form of a feature vector or pattern. A pattern is a vector $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, with each x_i representing particular knowledge of the system or domain being analysed. In protein systems, this domain knowledge usually represents some properties of the amino acid sequence under analysis. This feature vector is characterised as belonging to one of M classes, $\omega = \{\omega_1, \omega_2, \ldots, \omega_M\}$, in a classification problem. If we consider the problem of protein primary structure analysis, then the available families which exist are the classes of the system, and the protein primary structure or some transformed representation of this structure will be the features. The generation of the features is an important step, and many techniques exist for feature generation, each dependant on the problem area which is being analysed. The key aspect is that the features which are generated have discriminative ability: feature vectors from each class must be sufficiently different in the n-dimensional feature space to allow, for example, differentiation of patterns belonging to class ω_1 from class ω_2.

In general the problem of classifying patterns can be viewed as a probabilistic one, where patterns will be classified into particular classes depending on the probability of the pattern belonging to the class. We form M conditional probabilities $P(\omega_i|\mathbf{x})$, $i = 1, 2, \ldots, M$, which are known as *a posteriori* probabilities. The classification of a pattern into a particular class then follows the application of the following rule: consider an unknown input pattern $\mathbf{Z}$, which could belong to one of the M classes [45].

$$\begin{aligned} \text{Assign } \mathbf{Z} &\longrightarrow \omega_j \text{ if} \\ P(\omega_j|x) &= \max_k P(\omega_k|x) \end{aligned} \tag{3.2}$$

That is, a pattern belongs to a particular class j if class j has the maximum *a posteriori* probability of all k classes in the system.

The neural network is a machine learning system that simulates the brain's structures and operations through the use of fundamental processing units or perceptrons analogous to neurons, and weighted connections analogous to synapses. Many other machine learning systems also exist that do not simulate the brain structure, but achieve the same goals. What ties the different techniques together is the objective of the learning strategy. Each in some form attempts to infer *a posteriori* probabilities from a set of given training data, and bases the classification result on these *a posteriori* probabilities. A large number of machine learning tools have been used in this research, and each will be briefly discussed in the next section, providing the necessary background required for the analysis of the algorithms to be presented in later chapters.

3.4 Common Machine Learning Tools

3.4.1 Artificial Neural Networks

Neural networks are used in applications ranging from pattern recognition to regression analysis, due to their ability to learn the underlying structure of data. The simplest of these networks are the Generalised Linear Models (GLM) [46], which are single layer networks which consist of an input and an output layer. These networks implement well known statistical techniques such as linear regression and find use in the initial analysis of data when implementing a neural network system. A system of this type is shown in figure 3.1a.

A more common neural network architecture is the multilayer perceptron (MLP) [47]. The multi-layer perceptron network is a two–layer feed-forward network of the type shown in figure 3.1b and in this application is trained using a supervised learning algorithm. The supervised learning algorithm supplies both the inputs $\mathbf{x}$, and the target outputs $\mathbf{y}$, to the network and by using the back–propagation algorithm,

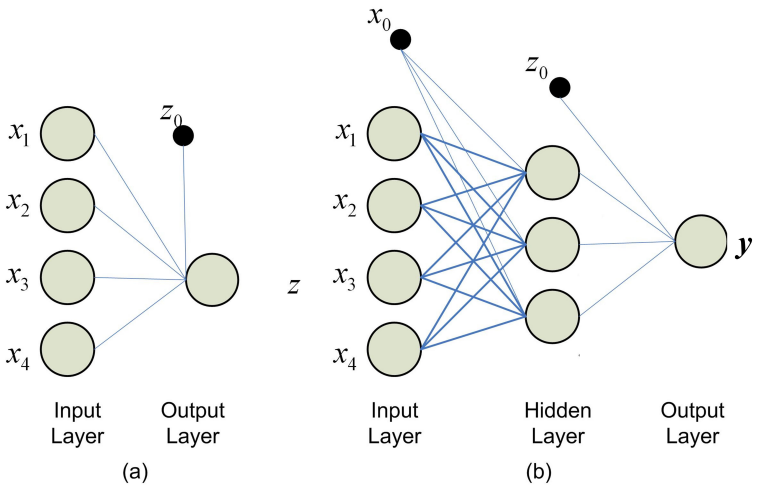


Figure 3.1: Common neural network architectures. a) Generalised Linear Model neural network; b) Multi-Layer Perceptron neural network. Black circles represent the biases and the greyed circles represent the individual neurons. The connecting lines are weights between the neurons.

the weights between each of the neurons in the network are adjusted [47].

The mapping between the inputs and the outputs is given by equation 3.3:

$$y_k = f_{outer}\left(\sum_{j=1}^{M} w_{kj}^{(2)}\left(\sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{3.3}$$

In equation 3.3, $w_{ji}^{(1)}$ and $w_{jk}^{(2)}$ indicate the weights in the first and second layers, respectively, going from input i to hidden unit j, M is the number of hidden units, d is the number of output units while $w_{j0}^{(1)}$ indicates the bias for the hidden unit j and $w_{k0}^{(2)}$ indicates the bias for the output unit k.

The weights of the neural network are optimised via backpropagation training using, most commonly, scaled conjugate gradient algorithm [48]. For classification problems, the output activation function is the sigmoid function, and for a training set $D = \{\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^{N}$ with N being the total number of training patterns, the cost function E, which is minimised, may be written using the cross-entropy error as follows [46]:

$$E = -\beta \sum_{n}^{N} \sum_{k}^{K} \zeta \ln(y_{nk}) + (1 - t_{nk}) \ln(1 - y_{nk}) + \sum_{j}^{w} \frac{\alpha_j}{2} w_j^2 \tag{3.4}$$

In this equation, n is the index for the training pattern, hyperparameter β is the data contribution to the error, k is the index for the output units, t_{nk} is the target output corresponding to the nth training pattern and kth output unit and y_{nk} is the corresponding predicted output. The second term in the expression is the regularisation parameter which penalises weights of large magnitudes. The regularisation parameter coefficient, α, determines the relative contribution of the regularisation term on the training error. The presence of the regularisation parameter gives significant improvements in the generalisation ability of the network [47].

The key property of the MLP architecture, according to the universal approximation theorem [49], is that a model of this form is able to approximate any continuous function of arbitrary complexity, given that the number of hidden layer neurons is sufficiently large. Another common type of Neural Network known as the Radial Basis Function (RBF) is popular in pattern recognition. These networks are based on a distance measure between an input vector and a prototype vector and exhibit faster training times than the MLP. Details will not be given, but the RBF networks are discussed in detail by Bishop [47].

3.4.2 Support Vector Machines

Support Vector Machine (SVM) methods for classification were originally developed by Vapnik et al, and based on statistical learning theory [50]. The SVM was initially conceived for a two class classification problem, but algorithms have now been developed for regression and multi–class problems [51]. SVM has become popular in protein science because of its good generalisation ability and has been used by authors such as Huang et al [35], Ding and Dubchak [36] and Zhao et al [52]. For simplicity, the two class problem will be briefly described.

Consider a set of labelled training patterns $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$, where $x_i \in R^n$ and $y_i \in \{-1, +1\}$. SVM is a margin classifier which draws an optimal hyperplane in a high–dimensional feature space, R^{nh}. The mapping of the data into the higher dimensional space is achieved by using a mapping function $\Phi(\cdot)$. This hyperplane is a decision boundary that maximises the distance between the hyperplane and near-

est data points of each class in the space R^{nh}. The decision function is implemented as

$$f(x) = sgn\left(\mathbf{w} \cdot \mathbf{x} + b\right)$$

The vector $\mathbf{w}$ is a function of a *kernel function* K. The kernel function takes into account the mapping of $\mathbf{x}$ into the higher dimensional space, and the determination of $\mathbf{w}$ is an optimisation problem to determine the best hyperplane separating the classes. The major consideration when using SVM is the choice of the Kernel function. Any function can be used as long as it satisfies Mercers condition [53], but standard kernel functions have become widely used. The kernel functions which are common include the Gaussian, RBF, polynomial, and exponential RBF kernels.

In the problem considered in this work, two options exist. The first is to view the problem as a set of M two–class problems as has been considered by Ding and Dubchak [36], where M is the number of classes to consider. This method is not computationally feasible for problems with a large number of classes, since the number of classifiers that must be created is at least quadratic with respect to the number of classes [54]. The second approach is to extend the mathematical formulation of SVM to the multi–class case, and this is the case that is considered here. Details of this extension can be found in Vapnik [50], and comparisons of these two approaches can be found in Hsu and Lin [55] and Rifkiy and Klatau [56].

3.4.3 *k*-Nearest Neighbour Classification

The k-Nearest Neighbour (kNN) algorithm or Rule, is one of the simplest algorithms for classification. Given an unknown feature vector $\mathbf{x}$ and a distance measure [53]:

- Out of N training vectors, identify the k nearest neighbours, irrespective of class label. k is chosen to be odd for a two class problem, and in general not to be a multiple of the number of classes M.
- Out of these k samples, identify the number of vectors, k_i, that belong to class $\omega_i, i = 1, 2, \ldots, M$. Obviously, $\sum_i k_i = k$.
- Assign $\mathbf{x}$ to the class ω_i with the maximum number of k_i samples.

Various distance measures can be used, including the Euclidean and Mahalanobis distance. The only serious drawback of the kNN algorithm is the complexity of the search for the nearest neighbour(s) among the N available training samples [43]. That said, the kNN is still a good algorithm to be used for comparative analysis of performance among various classification algorithms.

3.5 Multiple Classifier Systems

Multiple classifier systems are seen as a suitable design technique to improve the classification performance of a system by combining the results of various classification systems in order to obtain a final result that is better than using each of them separately [53]. These systems are motivated by the need for modular structures as exhibited by the brain. These are generally known as ensemble methods and have been proven to have the following advantages [57]:

- They efficiently use all the networks of a population,
- They efficiently use all the available data for training without over–fitting,
- They inherently perform regularisation by "smoothing in the functional space" which helps to avoid over–fitting.
- They utilise local minima to construct improved estimates.
- They are ideally suited for parallel computation.

In this case, we assume that we are given a set of L classifiers, which have already been trained to provide outputs of the class *a posteriori* probabilities. For a classification task of M classes, we require some combining function $C(\cdot)$ that fuses the outputs of each of the individual classifiers. Here, each classifier produces an estimate of the a posterior probability $P_j(\omega_i|\mathbf{x}), i = 1, 2, \ldots, M$, where $j = 1, 2, \ldots, L$ for an unknown input vector $\mathbf{x}$. It is assumed that $P_j(\omega_i|\mathbf{x})$ are mutually exclusive. A number of different combining functions have been proposed over the years and these include the product rule, the sum rule, the min rule, max rule, median rule and the majority voting rule. These techniques are each reviewed in detail by Kittler

et al [45].

In this work we have made use of the majority voting rule. The majority voting rule considers each of the output decisions of each of the classifiers as a vote for a particular class. The output decision is made according to the class that receives the maximum number of votes. In order to allow counting of votes, the posterior probabilities must be hardened to obtain discrete outcomes. The hardened vector is given by equation 3.5, where k is an index of the number of classes and i is an index of the number of classifiers.

$$\Delta_{ki} = \begin{cases} 1 & \text{if } P(\omega_k|\mathbf{x}_i) = \max_{j=1}^{M} P(\omega_j|\mathbf{x}_i) \\ 0 & otherwise. \end{cases} \tag{3.5}$$

Given the hardened vector, the majority voting rule [53] is given by equation 3.6:

$$\begin{aligned} \text{Assign } Z &\longrightarrow \omega_j \text{ if} \\ \sum_{l=1}^{L} \Delta_{jl} &= \max_{k=1}^{M} \sum_{i=1}^{L} \Delta_{ki} \end{aligned} \tag{3.6}$$

Note that for each class ω_k, the sum on the right hand side of equation 3.6 simply counts the votes received for this hypothesis from the individual classifiers. The class which receives the largest number of votes is then selected as the consensus (majority) decision [45].

The key consideration when using this type of combining scheme is to ensure *diversity* in the ensemble of classifiers. A number of approaches to introducing diversity exist. One method is to train the networks of the ensemble using different subsets of the available training data using either the same sets of features or using different feature sets [53]. This aspect of diversity will be looked at in more detail in a subsequent chapter.

3.6 Machine Incremental Learning

3.6.1 Aspects of Incremental Learning

A number of aspects must be considered when implementing a learning system that operates incrementally. Some of the main issues in the design of incremental learners are [44]:

- *Ordering Effects*: The order in which knowledge is acquired is an inherently important aspect of incremental learning. This order may be of importance in certain cases where the chronology represents some properties of the system under analysis, such as non-stationary time series data. This is not the case in the analysis of protein data
- *Learning Curve*: The system may in some cases begin from zero domain knowledge, and thus experience a learning curve as it acquires knowledge of the problem domain. The quality of predictions will improve slowly over time as the classifier acquires more domain knowledge. In general it is difficult to say when the system has learnt "enough" to make accurate predictions.
- *Open World View*: The standard batch trained methodologies assume a closed view of the world and the knowledge domain which is being analysed. In other words, the classifier is trained on all possible classes or groups of outcomes that could occur. If all the data relevant to the problem domain is available *a priori*, then this closed world view can be accepted. In the case of protein science where we are aware of the changing nature of classes and available proteins, the closed view cannot be accepted. In this case classifiers must have an open world view, where new data is able to be added to the existing knowledge allowing the classifier to accommodate a wider set of classes and data examples.

The so called "catastrophic interference" or forgetting phenomenon is also an important aspect which must be considered when dealing with incremental learning systems. Catastrophic forgetting occurs when new information added to a learning system causes previously learned knowledge to be forgotten, and is closely related

to the stability–plasticity dilemma introduced by Carpenter el al [58]. A completely stable classifier will preserve all exiting knowledge but will not accommodate any new knowledge. In the reverse case, a completely plastic classifier will accommodate all new data and forget all data that was previously learnt.

The approach to training a MLP or RBF neural network, after allowing sufficient time for data to be accumulated is a typical example of a stable classifier approach to learning new data. This approach is not suitable in many instances since all the data must be stored and the training times increases dramatically as more data is added to the training database. One problem with this approach is that it becomes problematic if any original data is lost. Similarly, may plastic classifiers exist [59, 60] that can learn data as it is received. These systems also prove problematic since they may require access to the old data, may forget what has previously been learnt or will not allow data of new classes to be added to the classification system. These aspects of stability and plasticity must be taken into account when designing an incremental learning system and deciding on the trade-off between the two.

A second type of interference known as "retroactive interference" occurs when a system is trained with data which is similar to data which it has been previously trained with and causes interference in the recall of this old data. In this case, memory items are shared in the model, so the interference of old data is desirable since it gives a greater degree of memory management. Therefore retroactive interference is a desirable property of learning systems. Wang and Yuwono provide a good discussion of both catastrophic and retroactive interference [61].

Ideally, an incremental learning system is one that has both a high degree of plasticity and stability, hence the stability–plasticity dilemma. A trade-off between the two must be made in any learning system and this concept must be carefully considered when analysing the performance of any classifier. From the above discussion, a desirable incremental learning system is one that exhibits retroactive interference without catastrophic forgetting, has no bias to the order in which training patterns are received and which has an open world view of the system that is being modelled.

Incremental learning algorithms can be tested in two ways: either new data can be introduced without any new classes, or new classes can be introduced to the learning system. Both of these approaches to testing are demonstrated in this research. Specifically, the GPCR and Enzyme data are used to test the addition of new data without new classes and the SCOP data is used to test the ability of the learning system when new classes are introduced.

3.6.2 Algorithms for Incremental Learning

A number of incremental algorithms have been reported in the literature, which meet the criteria for an incremental learning system. The fuzzy ARTMAP is one of the few classifiers which have shown to have incremental learning ability. The fuzzy ARTMAP will be described in detail, and is considered above others due to the detailed and *established theory* of the classifier, its ability for *multi–class classification* and its widespread use in other classification problems ranging from electric load forecasting, classification of prehensile EMG patterns, classification of handwriting, and the learning of mathematical functions, among others [58, 62, 63, 64, 65]. Other incremental classifiers have also been developed such as the incremental SVM and the incremental fuzzy decision tree.

Fuzzy ARTMAP Classifier

The application of the fuzzy ARTMAP to protein classification is one of the major contributions of this work, and will be described in detail. The fuzzy ARTMAP (FAM) was introduced by Carpenter et al [58], and is a supervised neural network, similar to the multi–layer perceptron. The key features of this type of network architecture is that it is capable of fast, online, supervised, incremental learning, classification and prediction [58].

Figure 3.2 shows a conceptual model of the fuzzy ARTMAP. This system takes n-dimensional input patterns and maps them into the n-dimensional feature space. The system divides this input space into a number of hyperboxes of varying size,

and maps these hyperboxes to a category in the output space, i.e to the class label. The network learns and adjusts its parameters on a per–pattern basis, rather than after entire cycles as in the standard neural network model. This is known as instance-based learning and thus as each individual input pattern is mapped into the feature space, existing hyperboxes are increased to accommodate the new pattern or a new hyperbox is created. If a new hyperbox is created, this hyperbox is also related to the output class. This entire process is controlled through a set of internal weights and a process known as match tracking. It is this instance–based learning that gives the fuzzy ARTMAP its incremental learning ability. This instance–based learning also makes the order in which training patterns are received an important factor, one which is not often considered in the use of fuzzy ARTMAP networks [62].

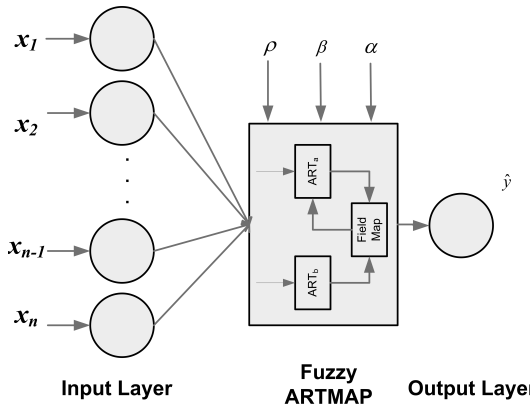


Figure 3.2: Representation of the Fuzzy ARTMAP Architecture

The fuzzy ARTMAP is controlled by three parameters: the vigilance ρ, the learning rate β and the choice parameter α. The choice parameter is a constant and is kept small, generally 0.001 [66], as used in this application. The learning rate adjusts the factor by which the hyperboxes are increased each time a new training pattern is received, and can be any value between zero and one. For $\beta < 1$, the network is said to be in *fast–commit slow–recode* mode, resulting in the hyperboxes increasing in a size proportional to the value of β. If $\beta = 1$, the system is in *fast learning* mode and the hyperboxes will be enlarged just enough to include the point represented by the input vector. The vigilance controls how large any hyperbox can become, and will result in new hyperboxes being formed, if the measured *degree* to which an input pattern *belongs* to a hyperbox is less than the vigilance. From this it is observed that the larger the vigilance (higher expected degree of belonging) the smaller the

hyperboxes created in the input space. This is a key factor to consider in the application of fuzzy ARTMAP systems, since large values of ρ will result in what is known as category proliferation, which will be observed as over–training in the system [62]. Details of the principles and operation of the FAM is given in Appendix B.

The fuzzy ARTMAP has been shown to be comparable to other classifiers. In the original work on the FAM presented by Carpenter et al, they have shown a comparison between the FAM and the neural network trained using various optimisation strategies. Their results in classifying two spirals from each other, showed that the FAM requires fewer training epochs and thus has faster training times than the neural network on this task [58]. The FAM has also demonstrated these properties and its popularity as a classifier in numerous other tasks, ranging from applications in character recognition [66], medical imaging [67], electronic nose [68], multi-spectral remotely-sensed images [69] and classification of multivariate chemical data [70].

Other Incremental Algorithms

As mentioned, a number of other incremental algorithms have been developed. Since these are not the focus of this dissertation they will only be mentioned briefly:

Incremental Fuzzy Decision Tree This system, described by Guetova et al [71] combines non-incremental algorithms for top down induction in decision trees with fuzzy logic and a measure of information gain, to realise an incremental learning system. The system is reported to be a fast classifier producing results which are equivalent to the non-incremental learning case.

Incremental SVM Many versions of an Incremental Support Vector Machine have been developed. This includes the Incremental and Decremental SVM of Cauwenbergs and Poggio [72], SVM for incremental learning of non stationary data described by Syed et al [73], incremental training using hot–start methods described by Shilton et al [74], and incremental training described by Ruping [75]. In these cases the authors have identified that the generation of the optimal hyperplane or the support vectors are an efficient way of representing large amounts of data. These support vectors are also all that is needed to recreate

a SVM for classification of data. Authors then examine and describe various ways of changing these support vectors to accommodate a wider range of data. None of the methods described though are applicable to the multi-class case, as all the algorithms consider the binary classification problem.

Incremental Common Sense Models This system has been developed by Giraud-Carrier and Martinez [76] describes a self organising learning model that combines inductive learning with rule–based and similarity based reasoning to give incremental learning.

There are many other systems based on Information theoretic measures and fuzzy propositions which are available in the literature such as incremental construction of support vector machine classifiers with provable performance guarantees [77], incremental learning based on reproducible kernel Hilbert spaces [78], or incrementally adding new IF-THEN (antecedent–consequent) rules to an existing fuzzy inference system [79], but these will not be discussed here. The reason why these are not considered is because unlike the fuzzy ARTMAP, these classifiers do not support multi-class classification and since they are binary classifiers, do not meet the requirement for an open world view.

3.7 Evolutionary Optimisation via Genetic Algorithm

Genetic algorithms (GA) find approximate solutions to problems by applying the principles of evolutionary biology, such as crossover, mutation, reproduction and natural selection [80]. The GA search process consists of the following steps:

1. Generation of a population (pool) of candidate solutions (chromosomes) $s_i, i = 1, \ldots, n$, where n is the number of chromosomes in the population.

2. Evaluation of the fitness $f(s_i)$ for each chromosome in the gene pool. Chromosomes with the lowest fitness are discarded and make way for a new set of chromosomes. Replacement sets of chromosomes are created by the genetic operations of crossover and mutation on the fittest individuals.

3. Steps 1 and 2 are repeated for a given number of generations until a specified fitness level is attained or the maximum number of generations is exceeded [81].

The genetic algorithms represent input data from the problem by an encoding such as binary or floating point and use the genetic operations to iteratively evaluate solutions from the population of potential solutions to determine the global optimum [81]. The GA evaluates candidate solutions through a fitness function. By maximising this fitness function, the fittest individuals are propagated throughout the generations allowing the global maximum to be determined after a specified number of generations. The fitness function contains information from the problem space and is the mechanism by which properties of the problem space is transferred to the GA, which is independent of the problem. The genetic operations are important since they add an element of randomness to the search process, allowing a wider range of the solution space to be explored.

In the application of the GA to this work, the floating point representation is always used. The floating point representation has proven to be more stable from run to run and has a lower standard deviation on the best answers than the binary representation [80]. Detailed aspects of the operation of the Genetic Algorithm can be found in the books by Michalewicz [80], Goldberg [81] and Haupt and Haupt [82].

Chapter 4

Incremental Learning of Protein Data

4.1 Introduction

This chapter presents the major work which has been conducted in this research. The general pattern recognition approach is followed and thus the chapter begins with a description of the transformation of the alphabetic sequences described in Chapter 2 into numerical features. Two approaches to classifying proteins in an incremental setting are then presented. The first presents a detailed algorithm for incremental learning based on the fuzzy ARTMAP and aspects of ensemble networks. The second technique is based on an ensemble technique known as Learn++. We implement Learn++ and present design strategies for the use of Learn++ in incremental settings.

A current review of the available literature has shown that incremental learning has not been considered before in the area of protein primary structure classification. Vijaya et al [83] have considered the problem of incremental clustering of protein sequences, to help discover superfamily–family–subfamily relationships in protein data, but this is a different problem. The problem considered by Vijaya et al, is to find natural groupings by clustering data from similarity scores, and the families which will be discovered may not have any biological significance. In this study, established, structurally significant families of proteins are used and protein sequences are classified into these families. These sequences may be generated in

a laboratory as part of a drug discovery process, and the classification will allow biologists to infer structures and functions of the protein under design, to determine if it will have any effect on the drug target.

4.2 Vectorisation of Protein Sequences

Three protein databases have been used for the testing of algorithms that are presented in this chapter. The datasets include data from the SCOP database, the GPCRDB as well as the PDB, that was discussed in Chapter 2. The raw data obtained from these databases consists of the protein sequences from various protein families. These protein sequences are first converted into numerical feature values before any of the tools which have been developed can be applied.

The transformation of these sequences into numerical values is commonly known as sequence vectorisation. Two types of features have been identified in the literature, these being global and local features, both of which are used in this work. The definitions used by Huang et al [84] are used, and are defined as:

Global Features Represent the nature of the entire protein sequence and must capture global similarity between related features allowing for comparison. The amino acid composition is an example of a global descriptor.

Local Features These features capture local interactions between amino acids and groups of amino acids in a protein sequence. The n-gram methods is a common example of a local descriptor.

4.2.1 Global Feature Generation

First consider the amino-acid composition of the protein sequences. The composition is simply the presence frequency of each of the 20-possible amino acids in the given sequence. Thus the composition is calculated by [52]:

$$\nu_i = \frac{s_i}{\sum_{j=1}^{20} s_j}, \text{ for } i = 1, 2, \ldots, 20. \tag{4.1}$$

where ν_i is the value for the ith feature and s_i is the number of times the ith amino acid appears in the sequence. This results in 20 features: a frequency of appearance for each of the twenty possible amino acids. If a particular amino acid does not appear at all in the sequence, the corresponding feature value is zero.

A further set of features based on the hydropathy of amino acids in a given protein sequence is also calculated. The hydropathy of an amino acid in a protein sequence is the attractive property of an amino acid to a water molecule. Amino acids are thus hydrophobic, hydrophilic (polar) or neutral. We use the Chothia and Finkelstein hydropathy classification as used by Dubchak et al [85], and which is given in Appendix A. We calculate three descriptors, the hydropathy composition (π_C), the hydropathy distribution (π_D) and the Hydropathy transmission (π_T) for the sequences [85]. The composition π_C, is calculated similarly to the amino acid composition described previously in equation 4.1. In this case we calculate the presence frequency of hydrophobic, hydrophilic and neutral amino acids in the sequence. This results in three features being generated.

The transmission π_T, is defined by three values. The first is the number of times a polar residue is followed by a neutral residue or vice versa. Similarly the other two are the number of times a neutral residue is followed by a hydrophobic residue or vice versa and the number of times the polar residue is followed by a hydrophobic residue or vice versa.

By definition, the distribution π_D, looks at intervals of 25%, 50%, 75% and 100% along the sequence length. For each interval the presence frequency of hydrophobic, hydrophilic and neutral molecules for each percentage interval is calculated. Thus this results in 12 features, 4 features for each of the three hydropathy groups. In total 38 features (20+3+3+12) are generated based on global sequence descriptors.

4.2.2 Local Feature Generation

The n-gram method is established as a good descriptor when performing text document classification and document authorship attribution [86], and has also been

demonstrated to be a good descriptor of local similarities in a sequence by many authors such as Cheng et al [25] and Tomovic et al [39]. Essentially the n-gram method considers the presence frequency of consecutive n-letter combinations in the protein sequence, for integer n. More formally, n-grams can be defined as:

Definition 2 (n-gram) *Given a sequence of characters $\mathcal{S} = (s_1, s_2, \ldots, s_N)$, over a defined alphabet $\mathcal{A}$ with n and the sequence length N being positive integers, an n-gram of the sequence $\mathcal{S}$ is any n-long subsequence of consecutive characters. The ith n-gram of $\mathcal{S}$ is the sequence $(s_i, s_{i+1}, \ldots, s_{i+n-1})$ [39].*

For example, consider the short sequence `SLTKTERTIIVSM`, the 2-grams of this sequence, following from the definition are: `SL`, `LT`, `TK`, `KT`, etc. Similarly 3-grams and higher n-grams can be determined from a given sequence. The use of n-grams have been used in many diverse areas, and the key properties of n-grams are [39, 87]:

- *Robustness*: Relatively insensitive to sequence variations and errors. They are also length invariant;
- *Completeness*: The sequence alphabet is known in advance;
- *Domain Independence*: The n-grams can be determined for any defined alphabet and any area of analysis.
- *Efficiency*: Only one pass processing is required;
- *Simplicity*: No detailed knowledge of the rules of the arrangement of the characters in the sequence is needed.

The only major drawback of the use of n-grams is the effect known as *exponential explosion* [39]. This simply implies that the larger n-grams such as 4- and 5-grams result in an extremely high dimensionality and computational expense in terms of memory. As an example, consider the 20 amino acid protein alphabet. The total number of n-grams produced increases from $20^2 = 400$ for 2-grams, to $20^3 = 8000$ for 3-grams to $20^4 = 160000$ for 4-grams.

Two letter combinations (2-grams) are known as digrams or bigrams. While higher

n-grams such as 3-grams and 4-grams have been considered in the available literature, only digrams are considered in this work since it has been shown [25, 35] to work well in protein classification systems. Given a sequence, features are generated by calculating the presence frequency of all possible 2-grams for the amino acid alphabet, which can be calculated using equation 4.1. This results in 400 features representing the local properties of the amino acids.

4.2.3 Data Normalisation

The combined set of features results in a feature vector with a dimensionality of 438. This feature vector is now used in the design of the protein family classifiers. Table 4.1 is a summary of the features that have been used in the system.

Table 4.1: Summary of features used for classifier designed

Symbol	Parameter	Dimension
ν	Composition	20
π_C	Hydropathy Composition	3
π_D	Hydropathy Distribution	3
π_T	Hydropathy Transition	12
η	n-gram Composition	400
Total		438

The generated features undergo min-max normalisation. The normalisation is a requirement for using the FAM, since the FAMs complement coding scheme assumes normalised data [58]. This normalisation is also important when training classifiers such as the multi–layer perceptron, since it prevents the weights for any single feature from becoming too large [47]. The equation for the min–max normalisation for a single feature x_i in a data set is [43]:

$$x_{i,norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.2}$$

where x_{min} and x_{max} are the minimum and maximum values respectively for that feature observed from the training data.

The performance of the trained classifiers are evaluated using the standard classification error rate ϵ on a validation set $\mathbf{V}$ for a classifier trained on data set $\mathbf{T}$, and is given by:

$$\epsilon\left(\mathbf{V},\mathbf{T}\right)=\frac{n_c}{N} \tag{4.3}$$

where n_c is the number of incorrect predictions and N is the total number of patterns in the validation data set.

4.2.4 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the process of "delving" into the data, allowing interesting properties of the data to be examined and an initial understanding of the data to be obtained [43]. Protein data sets are sometimes referred to as "extreme datasets" [88] since they generally exhibit high dimensionality as is the case with the data under investigation; or skewed data distributions or both. A class separability measure and an analysis of data distributions will be used to explore the available data giving an indication of the problems that may be experienced in the system analysis at later stages.

In order to gain an initial understanding of the data, a cluster–scatter analysis [53] is performed. This cluster–scatter analysis is a mathematical formulation described below allowing the degree of overlap or scatter between each of the various classes in the data set to be measured. The *within–class scatter* matrix is defined by equation 4.4 as:

$$S_w=\sum_{i=1}^{M}P_iS_i \tag{4.4}$$

where S_i is the covariance matrix for class ω_i given by equation 4.5 as:

$$S_i=E[(\mathbf{x}-\mu_i)(\mathbf{x}-\mu_i)^T] \tag{4.5}$$

and P_i is the *a priori* probability of class ω_i, i.e. $P_i\cong n_i/N$, where n_i is the number of samples in class ω_i out of the total of N samples. $E\left[\cdot\right]$ is the mathematical expectation of the argument and $\mu_i=E[x_i]$.

The *between–class scatter* matrix is defined by equation 4.7 as:

$$S_b = \sum_{i=1}^{M} P_i(\mu_i - \mu_0)(\mu_i - \mu_0)^T \tag{4.6}$$

$$\mu_0 = \sum_{i}^{M} P_i \mu_i \tag{4.7}$$

The *mixture–scatter* matrix is then simply defined by equation 4.8

$$S_m = S_w + S_b \tag{4.8}$$

Using the above formulations, the J_3 measure is used to determine the degree of inter-class–extra-class spread of data and is defined by equation (4.9) as:

$$J_3 = trace\{S_w^{-1} S_m\} \tag{4.9}$$

In equation 4.9, *trace* implies the standard trace of a matrix from algebra, i.e the sum of the diagonal elements. The value of J_3 takes on high values for data sets that are well clustered around their mean and separated from other classes. This value can be used beforehand to give an initial indication of the class separability and the difficulty that might be experienced in training a classifier using a given data set. A detailed discussion of the J_3 measure can be found in the book by Theodoridis and Koutroumbas [53]. Table 4.2 is a comparison of the J_3 measures that have been calculated for the protein data sets used in this research as well as a number of other easily accessible data obtained from the UCI Machine Learning Repository [89].

Table 4.2: Comparison of J_3 measures for some well known data sets

Data Set	# Classes	J_3
SCOP Data	12	1.48
GPCR Data	8	16.22
Enzyme Data	6	28.11
Iris Data	3	42.39
5-Class Linearly Separable Data	5	82.52
Glass Data	6	20.8951
Vowel Data	10	7.2309

This table suggests that the classification using the SCOP data might pose a problem due to a high degree of class overlap and the selection of the type of classification

systems must be chosen carefully. The GPCR data has a higher J_3 measure than that for the SCOP data, indicating that it has a suitable degree of class separability, and similarly for the enzyme data.

The distribution of the sequence lengths in the data that is used is an important factor to consider. Figures 4.1 and 4.2 show histograms of the sequence length distribution for the data that is used. The figures show that the the data is a unimodal distribution, with most sequences having a length of about 100 amino acids for the SCOP data and a length of about 350 amino acids for the GPCR data. The distributions also show that the data does include data of lengths both longer and shorter than those indicated at the modes. We can use this as an indication that the data used is sufficiently representative of the protein data in general and that results from experiments that are conducted can be used to show that the algorithms are not highly dependant on sequence lengths for classification.

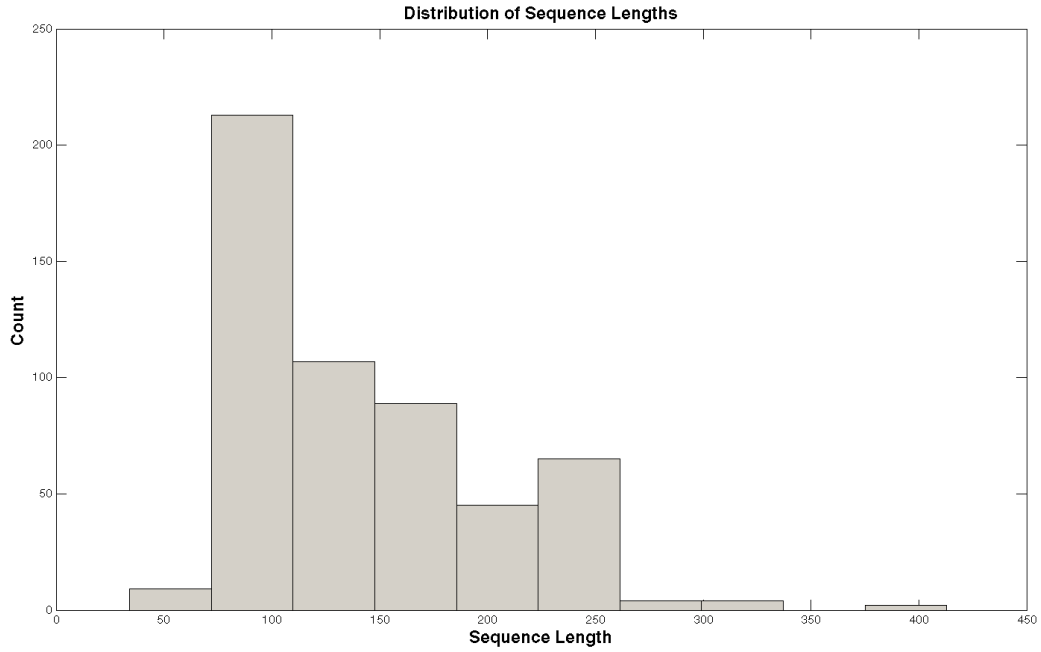


Figure 4.1: Sequence length distribution for SCOP dataset

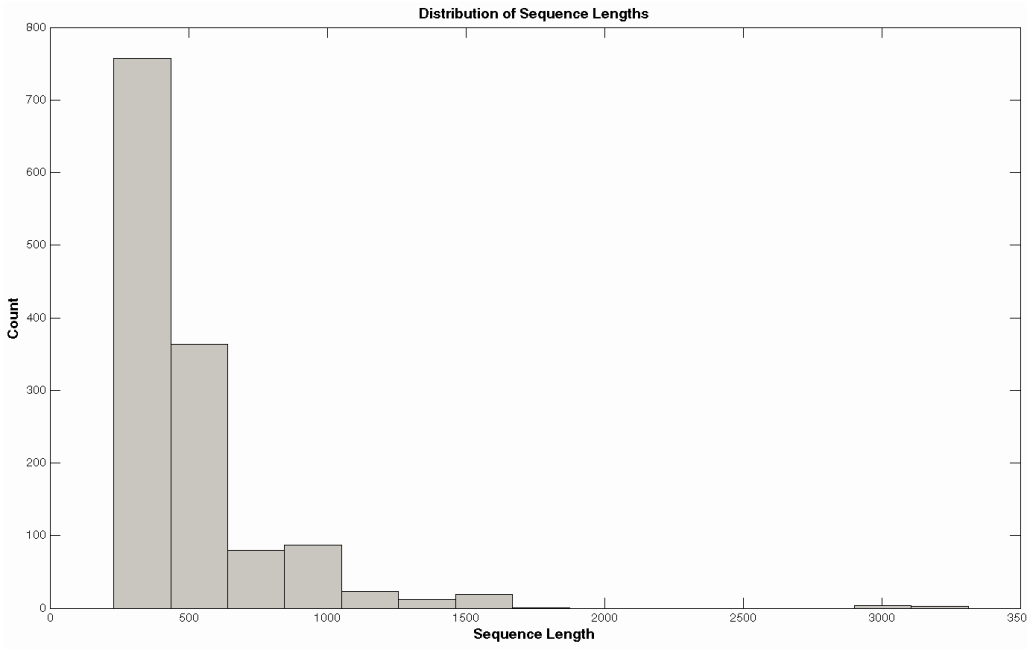


Figure 4.2: Sequence length distribution for GPCR data

4.3 Fuzzy ARTMAP and Incremental Learning

4.3.1 System Overview

Based on the biological evolutionary concepts and neuro-developmental stages that were discussed previously in Chapter 2, an incremental learning system based upon an evolutionary strategy is designed and implemented. A schematic representation of the system is shown in figure 4.3.

Input sequences are extracted from a protein database and then converted into a numerical feature vector. We then create a population of classifiers to introduce classification diversity, with a selection function based on kappa analysis, to choose the best classifiers from this population. An ensemble of classifiers is used as a means of introducing modularity in the learning system. This system is implemented using the fuzzy ARTMAP (FAM) and a series of experiments are conducted to evaluate the performance of this system. Pseudocode for the creation and operation of the system is shown in algorithm listing 4.3.1. The ability of the FAM as an alternative classifier to many of the other more popular classifiers is demonstrated by comparing the classification ability of these systems on the SCOP, GPCR and enzyme data

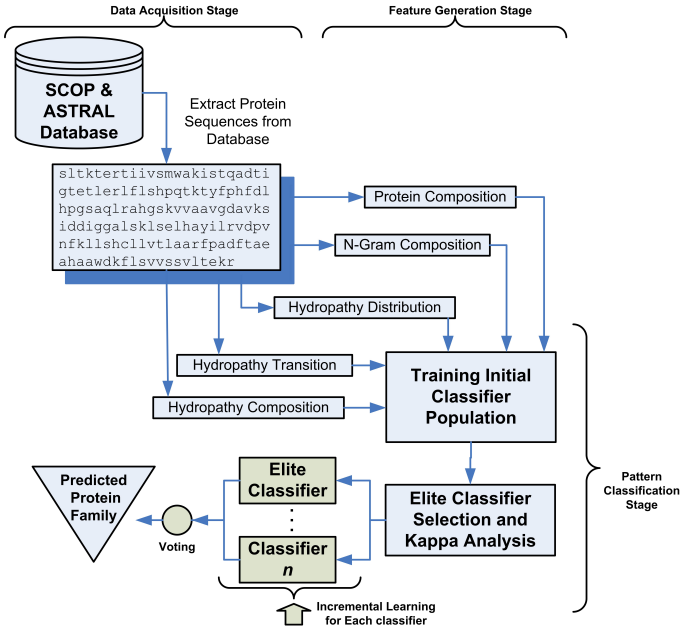


Figure 4.3: Overview of system architecture

sets. The incremental learning system described by algorithm listing 4.3.1 is then demonstrated on the three datasets showing that the system is able to learn new data without forgetting previously learned data.

4.3.2 Ensemble Diversity and System Training

Apart from the biological motivation for the use of a committee to simulate levels of modularity in the learning systems, the committee approach has been shown to provide an error which is less than that of the single best classifier in the system, with authors such as Perrone and Cooper [57] and Merz [90] having done extensive research, and having proven mathematically that this is true. This fact can be used to improve the classification ability of the system.

The creation of a committee of classifiers is based on a novel approach, implementing an evolutionary strategy which was summarised in algorithm listing 4.3.1. An initial population of j classifiers is trained, each classifier having been trained with a different permutation of the input training data, $\{\mathbf{X}_1, \mathbf{Y}_1\}$, where $\mathbf{X}$ is the feature vector and $\mathbf{Y}$ the corresponding output class. This permutation is needed in order to add diversity to the classifiers being created. As mentioned, the fact that

Algorithm 4.3.1: FUZZY ENSEMBLE(D)

Training Phase

comment: Create population j of classifiers each trained with a different permutation of the input data $\mathbf{X}_1$

Each classifier is a hypothesis $h_t : \mathbf{X}_1 \mapsto \mathbf{Y}_1$

$\epsilon = \frac{1}{N} \sum n|_{h_t(x_i) \neq y_i}$

comment: Sort the classifiers based on increasing error on a validation data set.

SORT(ϵ)

comment: Select the lowest error classifier as the elite classifier h_{elite}

comment: Calculate the agreement κ, of the 15 best classifiers (based on error) with respect to the elite classifier

$\kappa = \frac{N \sum_{i=1}^{N} x_{ii} - \sum_{i=1}^{N} x_{i+} x_{+i}}{N^2 - \sum_{i=1}^{N} x_{i+} x_{+i}}$

comment: Genetic Algorithm selection of p classifiers based on a trade–off between error ϵ and agreement κ

$GA_{fitness}(\kappa, \epsilon) = \sum_{i=1}^{p} \kappa_i + \sum_{i=1}^{p} \epsilon_i$

Create ensemble classification system using the elite classifier h_{elite} and the p selected classifiers $h_t, t = 1, \ldots, p$

comment: Fusion of individual classifier predictions using majority voting strategy

Operation Phase

If predicting sequence family, convert to feature representation and classify using the fuzzy ARTMAP based system created during this previous training phase

comment: If incrementing system knowledge, increment each of the classifiers in the Fuzzy ARTMAP based system independently, using the training data for new sequences

$h_t^{incr} = \mathcal{T}\left(h_t, \mathbf{X}_k \mapsto \mathbf{Y}_k\right),$

where the transformation $\mathcal{T}$ is the incremental training process and k is the dataset to be added to the system

the fuzzy ARTMAP learns in an instance–based fashion, makes the order in which the training patterns are received an important factor [62]. In the experiments performed, the initial population consists of 30 classifiers.

The classification error ϵ of each of these classifiers is then evaluated against a validation data set. The classifiers are then ranked in terms of increasing error. The lowest error classifier from this population is the *elite classifier* and is the classifier that automatically becomes a member of the ensemble system. The inclusion of this elite classifier ensures that at least one high accuracy classifier is selected for the committee.

The next step is to select the remaining p classifiers. In this application we select a further 4 classifiers, as this is thought to add sufficient diversity to the final solution. This value can be increased as necessary. The selection of the other members of the committee is important and requires a number of factors to be considered:

- It is not desirable to select classifiers that perform exactly as the elite classifier, since this gives no diversity to the predictions that are generated, and thus there is no room for improvement.
- Low accuracy classifiers should not be selected, since these will confuse the predictions obtained and thus result in final outcomes that are more erroneous than a single classifier.

It would appear that these two conditions are contradictory, since high accuracy classifiers would tend to agree on the same predictions, against what we require for point 1. A trade-off between the classifier accuracy and the level of agreement between classifiers is then ideally what is required. This introduces the need for a formal definition of agreement between classifiers.

We use the definition of agreement considered by Petrakos et al [91]. We define the agreement between any two classifiers κ based on the error matrix of the two classifiers [92]. The error matrix shows the number, and for which classes the two classifiers agree on a prediction. Table 4.3 shows the format for an error matrix

between two classifiers.

Table 4.3: Data format for error matrix between classifiers

	Classifier 2				
Classifier 1	C_1	C_2	...	C_N	Totals
C_1	x_{11}	x_{12}	...	x_{1N}	x_{1+}
C_2	x_{21}	x_{22}	...	x_{2N}	x_{2+}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_N	x_{N1}	x_{N2}	...	x_{NN}	x_{N+}
Totals	x_{+1}	x_{+2}	...	x_{+N}	

x_{11} in the table is the number of test patterns that both classifier 1 and 2 agreed belonged to class C_1. x_{21} is the number of test patterns that classifier 1 predicted belonging to class C_2, but that classifier 2 predicted belonged to class C_1. Similarly, the entire error matrix can be generated using the prediction made by any two classifiers. We determine the error matrices for 15 of the best classifiers in terms of predictions with respect to the elite classifier. The agreement is calculated using the following set of equations, where N is the number of training patterns used in generating the error matrix [92]. This formulation is known as kappa analysis.

$$\theta_1 = \sum_{i=1}^{N} x_{ii} \tag{4.10}$$

$$\theta_2 = \sum_{i=1}^{N} x_{i+} \cdot x_{+i} \tag{4.11}$$

$$\kappa = \frac{N\theta_1 - \theta_2}{N^2 - \theta_2} \tag{4.12}$$

The selection of classifiers from this population, which must essentially minimise both the error of the individual classifiers and the agreement of the classifiers with the elite classifier is an optimisation problem. The Genetic Algorithm has been chosen as the optimisation tool for this system. The Genetic Algorithm (GA) is a stochastic optimisation tool that borrows concepts from evolutionary biology such as selection, crossover and mutation [82]. The GA minimises a cost function that is defined for a particular problem by stochastically exploring the space of available solutions. The GA implemented for the selection of classifiers, is designed to select four classifiers and minimises both the agreement and the error of the selected combination of classifiers.

The GA will select 4 classifiers apart from the elite classifier, resulting in two vectors,

$$\begin{aligned} \epsilon_{\mathbf{GA}} &= \{\epsilon_1; \epsilon_2; \epsilon_3; \epsilon_4\} \\ \kappa_{\mathbf{GA}} &= \{\kappa_1; \kappa_2; \kappa_3; \kappa_4\} \end{aligned}$$

A linear combination of these two matrices is used to define the cost value of a particular selection of classifiers. It is this cost that the GA will attempt to minimise. The cost function is defined by equation 4.13. λ is introduced as a scalar constant to allow the relative importance of the agreement in the system to be adjusted. In this application $\lambda_1 = \lambda_2$, which gives equal importance to both the error and the agreement.

$$f(\epsilon, \kappa) = \lambda_1 \sum_{i=1}^{4} \kappa_i + \lambda_2 \sum_{i=1}^{4} \epsilon_i \tag{4.13}$$

The GA selects the four best classifiers that minimises the cost function of equation 4.13. The Genetic Algorithm was designed to produce 50 generations of solutions with each generation being a population of 30 possible solutions. The crossover rate was set to a high value of 0.8 and a mutation rate of 0.4, and were empirically determined to be the best values for the experiment. The high crossover rate was selected to ensure that there is a larger degree of exploitation of the space of available solutions, especially considering that in many cases the values of the error and the agreement calculated are the same for many classifiers. The crossover functions are modified from the standard crossover functions in this case, to ensure that unique classifiers are selected during each generation, that is, preventing the same classifier from being selected twice in a particular generation.

In all cases the solutions are represented in floating point form. For the Genetic Algorithm, the floating point operation has been shown to be better than the binary representation since it has shown to be faster in searching the space of possible solutions, providing higher precision and being more consistent from run-to-run [80]. Using this representation, the arithmetic crossover and non-uniform mutation operators have been used. The arithmetic crossover provides better stability in generating solutions, with lower standard deviation of the best solutions as compared to other

types of crossover such as simple crossover. The non-uniform mutation also aids in the search of an optimal solution by allowing faster convergence and greater accuracy [80].

These selected classifiers are then used in parallel, with each of the five classifiers in the system producing an independent set of predictions. These predictions must then be fused together to form the final decision. A number of decision fusion techniques exist. Some of these include the majority and weighted majority voting, trained combiner fusion, median, min and max combining rules that were discussed in Chapter 3. We adopt the majority voting decision fusion scheme, which simply considers each of the predictions produced by the five classifiers as a vote, with the final prediction for any pattern given by the prediction that receives the largest number of votes.

4.3.3 Incremental Learning of Protein Data

The ensemble system is not a useful system if it is not able to accommodate newly discovered sequences that are produced daily, as was shown in figure 2.9 and figure 2.10. The fuzzy ARTMAP through its instance–based learning is able to incrementally learn new data. This incremental learning can consider two types of data:

1. It is possible to add new sequence information for families which the classifier has already been trained with.

2. Data of completely new classes can be added to the system, increasing the knowledge that the system has of the general protein domain. Since this is a supervised system, new classes will be identified by class labels which are different from those seen previously.

The base system will in general be trained with data of a number of classes. Once new data becomes available, incremental learning of the system is based on incrementally training each of the 5 FAM classifiers in the system with the new data. The system can now be tested with data from all classes it has been trained with,

including classes which have been incrementally added to the system. The incremental learning demonstrated by the fuzzy ARTMAP system is a mixture of the *learning parameter* and *data* incremental learning types that were discussed in section 3.2 (page 29). During incremental learning, the fuzzy ARTMAP will adjust its internal parameters such as the vigilance and the map field, which is *learning parameter* incremental learning discussed in section 3.2, and since new classes may be added to the system, this is *data incremental* learning, also discussed in section 3.2.

Each of the classifiers in the final ensemble is independent during its learning process and the results are combined by majority voting to reach the final decision. Majority voting is used as opposed to the weighted majority voting scheme, due to the complexity that will arise in determining voting weights, which must change for every incremental round for the weighted version. At this stage in the research, the simple majority voting is used to prove the applicability of incremental learning to the analysis of protein primary structures. This weighted voting may be implemented at a later stage after developing an adaptive voting–weight estimation algorithm.

4.4 Ensemble System Testing and Experimental Results

The ensemble system described, based on the FAM has been tested using the data described previously. The organisation of this data is described for each of the databases being used. The same data and data partitioning will be used in testing the algorithm to be described in the next section.

4.4.1 Testing of SCOP Data

Data Organisation

The training of this system requires the creation of a training, validation and testing dataset. The available data will be separated into three databases. Database $\mathcal{D}_1$ will consist of data of classes 1 to 8 and database $\mathcal{D}_2$ will consist of data of classes 1

to 10, adding two new classes to the data. Finally, database $\mathcal{D}_3$ will consist of data of classes 1 to 12, adding a further two classes to the system. The validation data set $\mathcal{D}_{val}$ described, is required only for the database that is used to create the base classifier system, i.e. Database 1. A Further testing dataset consisting of all classes is kept to test the performance of the system on unseen data. The separation of data is listed in table 4.4.

Table 4.4: SCOP 1.69 protein families considered in this work, and the separation of the data into training, validation and testing data sets

#	SCOP Family	PDB ID	$\mathcal{D}_1$	$\mathcal{D}_{val}$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_{test}$
1	Phycocyanin-like Phycobilisome Proteins	46532	9	2	4	4	4
2	Monodomain Cytochrome C	46627	19	5	7	7	10
3	Glutathione S-transferase (GST), N-terminal domain	52862	17	4	8	7	9
4	Calmodulin-like	47502	20	5	8	9	11
5	Crystallins/Ca-binding development proteins	49696	9	2	3	3	4
6	MHC antigen recognition domain	54453	24	6	11	11	13
7	Eukaryotic Proteases	50514	23	6	9	9	12
8	Alcohol Dehydrogenase–like, N-terminal domain	50136	11	3	5	5	5
9	Nucleosome Core Histones	47114	–	–	15	3	5
10	Thioltransferase	52834	–	–	18	5	6
11	Cytochrome B5	55857	–	–	–	19	5
12	Proteasome Subunits	56251	–	–	–	40	10

Comparative Performance

The fuzzy ARTMAP is shown in this section to be an alternate classifier to other common classifiers discussed in the literature. Specifically, the fuzzy ARTMAP [93] is compared to the multi-layer perceptron (MLP) neural network , the k- Nearest neighbours classifier , the Generalised Linear Model Classifiers (GLM) [46] as well as the Support Vector Machines (SVM) [94].

The training dataset for this comparative test consists of databases $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_{val}$ combined into a single set, with the testing set being left for independent assessment of the classifier. Each of these classifiers is trained on the training dataset, using all the generated features. The fuzzy ARTMAP is trained with a vigilance of $\rho = 0.75$, in fast learning mode. The MLP is trained for 100 cycles with 15 nodes in the hidden layer. These values are empirically determined. The k-nearest neighbours classifier is trained and tested for different values of k. Similarly, the SVM is tested with various kernel functions, specifically the polynomial and radial basis function kernels. The error reported is the error for the independent validation data set as well as the test dataset and is shown in Table 4.5.

Table 4.5: Comparative performance of FAM versus other classifiers on the SCOP dataset.

Classifier	Test Error (%)
Generalised Linear Model	35.11
5-Nearest Neighbours	31.91
3-Nearest Neighbours	27.66
1-Nearest Neighbours	20.21
Multi–layer Perceptron, $n_{hid} = 25$, $cyc = 200$	13.83
Fuzzy ARTMAP $\rho = 0.75$	15.80
SVM - RBF $\gamma = 5$	12.77
SVM-Polynomial 2.27 degree	11.70

Base Classifier Training and Incremental Performance

Training the base classifier involves using the dataset $\mathcal{D}_1$ as the training database, and evaluating the performance of this initial set of classifiers using the validation data set $\mathcal{D}_{val}$. Table 4.6 shows the agreements and the errors calculated for the top 15 classifiers in the population. The table shows that the minimum error classifiers do not necessarily have the highest agreement with the elite classifier.

The Genetic Algorithm [95] selected classifiers `2, 4, 6` and `7` to form the ensemble, in addition to the elite classifier. This final system was then incrementally trained on databases $\mathcal{D}_2$ and $\mathcal{D}_3$ and tested on the testing data $\mathcal{D}_{test}$. Table 4.7, shows the performance of the system after each incremental training session – Train 1, 2 and

Table 4.6: Error and agreement values for 15 classifiers of the population

Classifier	Val Error ϵ (%)	Agreement κ
1	6.0606	*Elite*
2	6.0606	1.000
3	9.0909	0.8952
4	9.0909	0.8943
5	9.0909	0.9292
6	9.0909	0.8952
7	9.0909	0.9292
8	12.1212	0.9290
9	12.1212	0.9292
10	12.1212	0.8952
11	12.1212	0.8619
12	12.1212	0.8272
13	12.1212	0.9290
14	12.1212	0.8605
15	15.1515	0.9290

3. The format of this table is the same as that used by Polikar et al [96]. The performance on the training data sets is also shown, to evaluate the performance of the algorithm on previously seen instances, to make sure that previously acquired knowledge was not lost.

Table 4.7: Training and generalisation performance of system on SCOP data

Dataset	Train 1	Train 2	Train 3
$\mathcal{D}_1$	0	0	0
$\mathcal{D}_2$	—	0	0
$\mathcal{D}_3$	—	—	0
$\mathcal{D}_{val}$	6.0606	6.0606	12.1212
$\mathcal{D}_{test}$	34.0426	27.6596	15.9574

From this table it can be seen that the performance of the overall system on the testing data improves as more data is added. Also, the errors of the training databases do not change and remains at 0%. The performance of the validation dataset shows a decrease in the classification accuracy. This increase in error, can be tolerated and is expected since there is a tradeoff that must be made between the stability and plasticity of the overall classification system. This validation data is not required for further development of the system once the initial base classifiers have been trained and should be incrementally added to the knownledge of the system.

The incremental learning of the system on the 12th class was analysed instance-by-instance to observe the performance of the individual classifiers and the overall ensemble as more data of this class is added to the system. The result is shown in figure 4.4. The graph shows that as more training patterns are added, the ability of the system to *correctly* predict the family of a given protein sequence increases, as shown by the decreasing error. This demonstrates the initial learning curve (discussed in Chapter 3) that the system must overcome, and that the system is responsive to new data, quickly overcoming this learning curve.

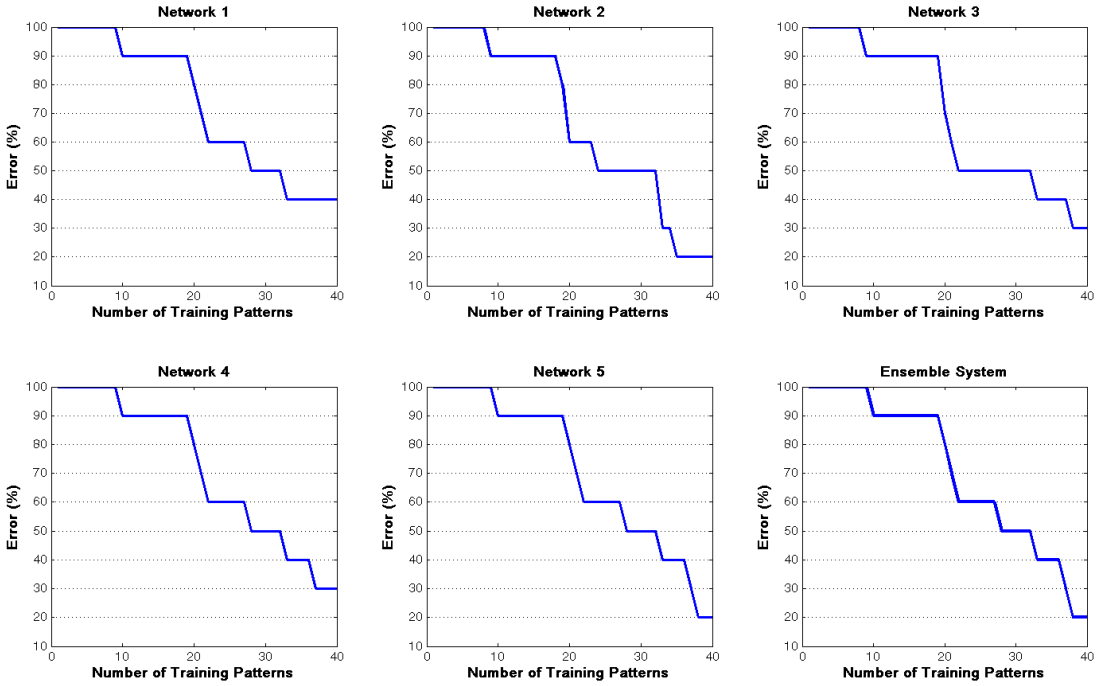


Figure 4.4: Incremental performance of each classifier in the system as training data of class 12 is increased. The combined incremental ability of all classifiers is shown in the final graph.

Analysis of the graph of figure 4.4, shows that 8 sequences must be added to the system, before the classifier is able to recognise any of the testing data. This suggests that there is some minimum number of sequences that is needed to be added, or that a threshold exists that must be passed before any improvement in prediction performance is obtained during incremental learning. This minimum is clearly dependant on the family being added, since some families have a high degree of sequence similarity, thus requiring only a few training instances, while other families having high degrees of dissimilarity require more training instances before any improvement is

observed. The implications of the results presented here will be discussed in the next chapter.

4.4.2 Testing of GPCR Data

Data Organisation

The GPCR data is also divided into 6 separate databases $\mathcal{D}_1, \ldots, \mathcal{D}_6$, with a validation set for database $\mathcal{D}_1$. In this case, the datasets have data of all 8 classes which are available. The separation of data into these databases is shown in table 4.8.

Table 4.8: Separation of data into individual databases for testing using GPCR data.

GPCR Family	$\mathcal{D}_1$	$\mathcal{D}_{val}$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$	$\mathcal{D}_{test}$
GPCR type 1	32	10	43	43	43	43	43	43
GPCR type 2	23	8	30	30	30	30	30	30
GPCR type 3	16	6	22	22	22	22	22	22
GPCR type 4	6	2	9	9	8	8	8	8
GPCR Fz/Smo	12	4	16	15	16	16	16	16
MLO family	3	1	4	5	5	5	5	4
Class H	32	11	43	43	43	43	43	43
Pheromone Receptor 2	20	6	26	26	26	26	27	27

Comparative Performance

As before, we compare the fuzzy ARTMAP with other more common machine learning tools such as the Support Vector Machines and Multi-layer perceptron. Table 4.9 shows the performance of the classifiers that were considered in the experiment with the SCOP data. The parameters that are used for each of the classifiers is included in the table. The classifiers are trained with all the training data combined into a single training set and tested on the test set $\mathcal{D}_{test}$. The table shows that the FAM, is consistent in its classification performance, again proving to have comparable accuracy to many other classification systems.

Table 4.9: Comparative performance of FAM versus other classifiers on the GPCR dataset.

Classifier	Test Error (%)
Generalised Linear Model	25.91
Multi–layer Perceptron, $n_{hid} = 15$, $cyc = 200$	15.03
Fuzzy ARTMAP $\rho = 0.75$	11.90
SVM - RBF $\gamma = 2.3$	17.10
SVM-Polynomial 2.23 degree	10.36

Base Classifier Training and Incremental Performance

As before, the base classification system was trained using database $\mathcal{D}_1$. Table 4.10 shows the error of the first 15 classifiers of the population and agreement with the elite classifier. The performance measure is the error of the system on the validation data set.

Table 4.10: Error and agreement values for 15 classifiers of the population

Classifier	Val Error ϵ (%)	Agreement κ
1	27.0833	*Elite*
2	29.1667	0.8940
3	29.1667	0.9730
4	29.1667	0.8438
5	31.2500	0.8929
6	31.2500	0.8929
7	31.2500	0.8929
8	31.2500	0.8455
9	31.2500	0.8683
10	31.2500	0.8929
11	31.2500	0.8929
12	31.2500	0.8929
13	31.2500	0.8929
14	31.2500	0.8430
15	33.3333	0.8430

The GA, for this data set selected classifiers `2, 3, 4` and `12` to form the final ensemble system. Again, the system consisting of the elite classifier and the four classifiers selected by the GA are incrementally trained using databases $\mathcal{D}_2, \ldots, \mathcal{D}_6$, with the ensemble being tested after each increment with the testing database $\mathcal{D}_{test}$. The performance of the system is shown in table 4.11, with the format of the table

the same as the format described for the testing with the SCOP data. This data

Table 4.11: Training and generalisation performance of system on GPCR data

Dataset	Train 1	Train 2	Train 3	Train 4	Train 5	Train 6
$\mathcal{D}_1$	0	0	0	0	0	0
$\mathcal{D}_2$	—	0	0	0	0	0
$\mathcal{D}_3$	—	—	0	0	0	0
$\mathcal{D}_4$	—	—	—	0	0	0
$\mathcal{D}_5$	—	—	—	—	0	0
$\mathcal{D}_6$	—	—	—	—	—	0
$\mathcal{D}_{val}$	25.0000	22.9167	22.9167	27.0833	25.0000	27.0833
$\mathcal{D}_{test}$	22.7979	18.6528	19.1710	19.6891	18.6528	16.5803

shows that the system is extremely capable of remembering data that it has been previously trained with, as shown by the many occurences of 0% which appear in the table for the training databases. The system also shows that the perfromance does increase as more data of each of the classes is added to the system.

4.4.3 Testing on Enzyme Data

Data Organisation

The enzyme data is divided into 5 separate databases $\mathcal{D}_1, \ldots, \mathcal{D}_5$, with a validation set for database $\mathcal{D}_1$. Each database consists of data of all 6 classes which are available. The separation of data into these databases is shown in table 4.12.

Table 4.12: Separation into individual databases for training and testing using enzyme data.

Enzyme Class	$\mathcal{D}_1$	$\mathcal{D}_{val}$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_{test}$
Oxidorectases	210	29	230	231	231	231	230
Transferases	243	35	278	278	278	278	278
Hydrolases	544	77	621	620	620	620	621
Lyases	83	12	95	96	96	96	96
Isomerases	41	6	47	47	47	47	47
Ligases	16	2	18	18	18	18	18

Table 4.13: Comparative performance of FAM versus other classifiers on the enzyme dataset.

Classifier	Test Error (%)
Generalised Linear Model	42.64
Multi–layer Perceptron, $n_{hid} = 15$, $cyc = 200$	10.00
Fuzzy ARTMAP $\rho = 0.85$	11.40
SVM - RBF $\gamma = 2.3$	9.76
SVM-Polynomial 2.23 degree	10.52

Comparative Performance

As before, we compare the fuzzy ARTMAP with other more common machine learning tools such as the Support Vector Machines and Multi-layer perceptron. Table 4.13 shows the performance of the classifiers that were considered in the experiment with the enzyme data. The parameters that are used for each of the classifiers is included in the table. The classifiers are trained with all the training data combined into a single training set and tested on the test set $\mathcal{D}_{test}$. The table shows that the FAM has comparable accuracy to many other classification systems.

Base Classifier Training and Incremental Performance

As for the previous two datasets, the base classification system was trained using database $\mathcal{D}_1$. Table 4.14 shows the error of the first 15 classifiers of the population and agreement with the elite classifier. The error that of the system on the validation data set.

The GA, for this data set selected classifiers `2, 3, 5` and `7` to form the final ensemble system. Again, the system consisting of the elite classifier and the four classifiers selected by the GA are incrementally trained using databases $\mathcal{D}_2, \ldots, \mathcal{D}_5$, with the ensemble being tested after each increment with the testing database $\mathcal{D}_{test}$. The performance of the system is shown in table 4.15, with the format of the table the same as the format described for testing with the previous two datasets. These results confirm what has been previously observed. The data shows that the system is extremely capable of remembering data that is has been trained upon. The system also shows that the performance does increase as more data of each of the classes is

Table 4.14: Error and agreement values for 15 classifiers of the population

Classifier	Val Error ϵ (%)	Agreement κ
1	22.9814	*Elite*
2	22.9814	0.7261
3	26.0870	0.6486
4	26.7081	0.5901
5	26.7081	0.5754
6	26.7081	0.5925
7	26.7081	0.5638
8	26.7081	0.6559
9	27.3292	0.7138
10	27.9503	0.6407
11	29.1925	0.6260
12	29.1925	0.5823
13	29.8137	0.5720
14	30.4348	0.5720
15	30.4348	0.6746

Table 4.15: Training and generalisation error of system on Enzyme data

Dataset	Train 1	Train 2	Train 3	Train 4	Train 5
$\mathcal{D}_1$	0	1.9395	2.2498	1.8619	1.7067
$\mathcal{D}_2$	—	0	0	0	0
$\mathcal{D}_3$	—	—	0	0.0775	0.0775
$\mathcal{D}_4$	—	—	—	0	0.1550
$\mathcal{D}_5$	—	—	—	—	0
$\mathcal{D}_{val}$	19.8758	15.5280	18.0124	14.9068	13.0435
$\mathcal{D}_{test}$	21.0853	15.0388	15.1938	13.9535	11.1628

added to the system.

4.5 Learn++ for Protein Classification

4.5.1 Overview of Learn++

Learn++ is the most recent incremental learning system and was introduced by Polikar et al [96]. Learn++ is based on the well known AdaBoost, and exploits the power of multiple classifier systems to give incremental learning ability. The algorithm pseudocode for Learn++ is given in algorithm listing 4.5.1. Consider that

there exist a number of different databases $\mathcal{D}_j$ for $j = 1, \ldots, J$ and a classification system, like MLP that will be called `weakLearn`.

Learn++ generates multiple 'weak' base classifiers each trained with different sub-

Algorithm 4.5.1: LEARN++(D)

for $j \leftarrow 1$ **to** J

do

- Initialise $w_1(i) = D(i) = 1/m \ \forall i$
- **for** $t \leftarrow 1$ **to** T_k
  - **comment:** Create sampling distribution
  - $D_t = \mathbf{w}_t / \sum_{i=1}^{m} w_t(i)$
  - Randomly choose training set TR_t and testing set TE_t from distribution D_t
  - **comment:** Use training set obtaining a hypothesis
  - $h_t \leftarrow \text{WEAKLEARN}(TR_t, TE_t)$ (1)
  - **comment:** Determine hypothesis error
  - $\epsilon_t = \sum_{h_t(x_i) \neq y_i} D_t(i)$
  - **if** $\epsilon_t > 0.5$
    - **then** Discard hypothesis and go to step 1
    - **else** Calculate $\beta = \epsilon/(1-\epsilon)$
  - Get composite hypothesis H_t
  - $H_t \leftarrow \text{WEIGHTEDMAJORITYVOTING}(h_t)$ and determine composite error
  - $E_t = \sum_{H_t(x_i) \neq y_i} D_t(i)$
  - **if** $E_t > 0.5$
    - **then**
      - Go to step (1)
      - Otherwise calculate $B_t = E_t/(1-E_t)$
  - **comment:** Update Weights
  - $w_{t+1}(i) = \begin{cases} B_t & \text{if } H_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
  - $w_{t+1}(i) = w_t(i) \times B_t^{1-[|H_t(x_i) \neq y_i|]}$

Call weighted Majority Voting on combined hypotheses H_t

sets of the training data. A weak learner is simply a classifier which is not optimised for maximum performance. In the case of Learn++, a weak learner is specifically a classifier with the ability to classify data with an accuracy of 50%. Each database $\mathcal{D}_j$, which is an independent set of data that is to be added to the system consists of training instances $\mathbf{x}_i$, and corresponding labels $\mathbf{y}_i$. The algorithm specifies a base classifier and an integer T_k, which is the number of classifiers to add to the system for a given database $\mathcal{D}_j$. The system trains each of the T_k classifiers with a different

subset of the training data, effectively creating multiple hypotheses for the training data. These multiple hypotheses are combined using a weighted majority voting scheme, where the voting weights for each classifier in the system is determined using the performance of that particular classifier on the entire set of training data used for the current increment. Figure 4.5 is a representation of how the many weak learners combine their individual hypotheses to form the combined hypothesis that approximates the true hypothesis. Newer hypotheses (classifiers) which are added have knowledge of new areas of the broader knowledge domain and thus allow for incremental learning.

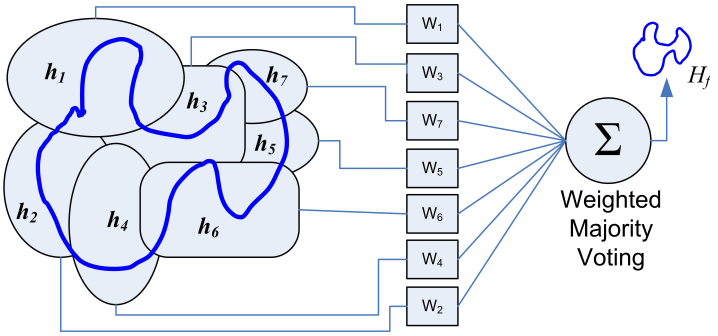


Figure 4.5: Combining the hypotheses of individual classifiers to allow for incremental learning. Redrawn from [96]

Each new database that is appended to the system adds an additional T_k classifiers to the system. If a new database has data of a new class, then the new classifiers in the system will have the knowledge of the new data while the overall system still maintains the knowledge that was acquired in previous training increments. This avoids the problem of catastrophic forgetting. This system, due to the ensemble of classifiers approach that is used, inherits the performance boosting ability of Adaboost. This incremental learning algorithm is also not sensitive to the order in which training patterns are received. This property has been demonstrated by Polikar et al [96] on the benchmark databases that were used in testing Learn++. This architecture thus meets all the requirements for incremental learning that were discussed previously: avoiding catastrophic forgetting, retroactive interference, an open–world view by allowing new classes to be added to the system, and unbiased performance with respect to order of the training data. A more detailed description of the Learn++ system is given in Appendix C.

4.5.2 System Overview

The Learn++ system which is implemented here uses the same datasets as described previously, i.e. SCOP, GPCR and enzyme datasets. The base classifier used for Learn++ is the multi–layer perceptron. As with the data for the incremental system using fuzzy ARTMAP, the data undergoes a min–max normalisation using equation 4.2. The min–max normalisation is of particular importance since a MLP is used as the base classifier and we wish to prevent any single feature of the data set from causing unbiased weights in the neural network model [47]. The performance of the system is also measured using the classification rate given by equation 4.3. An overview of the system and its operation is shown in Figure 4.6.

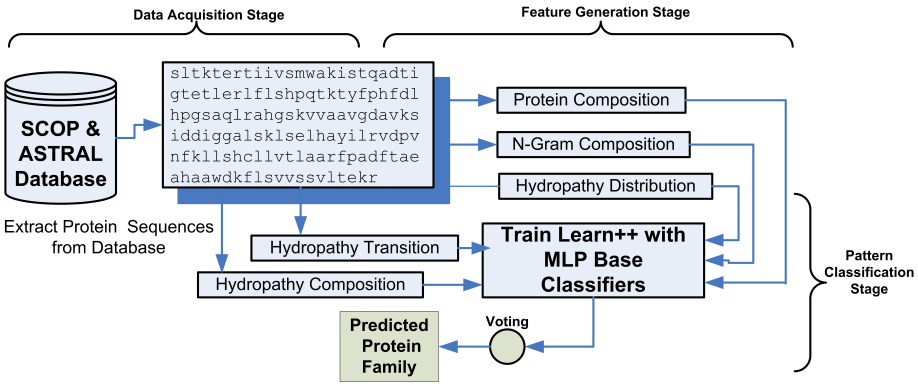


Figure 4.6: Overview of Learn++ based incremental learning system

4.5.3 Application of Learn++

Learn++ is applied to this problem using the MLP as the weak learning algorithm. The MLP allows one to set the number of training cycles and the number of hidden units in the system, and an under-design of the MLP will allow the MLP to work as a weak classifier allowing the error of 50% to be achieved. The MLP was trained in all cases described here, with 8 nodes in the hidden layer for 150 training epochs. The number of classifers that is added at each stage has been decided empirically. The single factor that is considered in this selection is that when incremeting the system with database 2 say, the number of classifiers that is added should be greater than the number of classifiers that was added for the previous database training.

The incremental learning demonstrated by Learn++, is a mixture of the *structural* and *data* incremental learning types that were discussed in section 3.2 on page 29. During incremental learning, Learn++ will add new classifiers to its ensemble, changing its internal structure, which is structural incremental learning, and since new classes may be added to the system, this is data incremental learning.

4.6 Learn++ Testing and Experimental Results

4.6.1 Testing of SCOP Data

Data Organisation

The data used in this section is the same as the data that was used in the SCOP database testing for the ensemble system. A validation set is not required in this case and is thus combined with the training database 1. The comparative performance of the MLP which is used as the base classifier in this system was listed in table 4.5.

Incremental Performance

As before each database is incrementally added to the system and the performance of the system for the training data and the testing data is shown in the table, so that the performance of the system in remembering data that it has been previously trained with can be evaluated. The system was initially trained with 10 classifiers in the ensemble. Thereafter, 15 classifiers were added to the system for database $\mathcal{D}_2$, and finally 19 classifiers were added for incremental training of database $\mathcal{D}_3$. The performance of this system is shown in table 4.16.

This table demonstrates the applicability of Learn++ to the problem of incremental learning of protein data. Issues of stability and plasticity are also evident from the data, but as for the testing of the previous system, this is not unexpected.

Table 4.16: Comparison of errors for Learn++ as the number of training databases in increased.

Dataset	Train 1	Train 2	Train 3
$\mathcal{D}_1$	0	5.45	9.09
$\mathcal{D}_2$	—	0	10.23
$\mathcal{D}_3$	—	—	12.30
$\mathcal{D}_{test}$	31.91	22.34	15.96

4.6.2 Testing of GPCR Data

Data Organisation

The same organisation of the the GPCR data that was used for testing in the previous experiments is used here. Again, the validation set is combined with database 1 since it is not required for this incremental system. As before, the comparative performance of the MLP when using this data is given in table 4.9.

Incremental Performance

The system is systematically incremented using data from databases $\mathcal{D}_1, \ldots, \mathcal{D}_6$, and the performance of the system is reported in table 4.17. For the training of this system, the number of classifiers that was added to the system is $\mathbf{T}_k = \{5, 10, 18, 23, 28, 34\}$ for each database that is incrementally added to the classification system. The table shows the desirable property of incremental learning where the performance of Learn++ on $\mathcal{D}_1$ increases as more training data is added. This row is not zero as in the case of the fuzzy ARTMAP system, since the training of the MLP stops after a specified number of training epochs, not when the error reaches zero as in the case of the FAM.

Table 4.17: Training and generalisation performance of system on GPCR data

Dataset	Train 1	Train 2	Train 3	Train 4	Train 5	Train 6
$\mathcal{D}_1$	27.60	21.35	17.71	17.71	17.71	16.15
$\mathcal{D}_2$	—	18.65	14.51	13.99	11.92	11.40
$\mathcal{D}_3$	—	—	18.13	17.10	16.06	16.58
$\mathcal{D}_4$	—	—	—	17.10	17.10	17.62
$\mathcal{D}_5$	—	—	—	—	16.06	16.06
$\mathcal{D}_6$	—	—	—	—	—	20.73
$\mathcal{D}_{test}$	29.02	23.32	20.21	20.73	20.21	18.65

4.6.3 Testing of Enzyme Data

Data Organisation

The same organisation of data that was described by table 4.12 is used here and as before, the validation set was combined with the dataset for $\mathcal{D}_1$ since it is not needed for this system. The comparative performance of the MLP was also given in Table 4.13.

Incremental Performance

As before, the system is systematically incremented using data from databases $\mathcal{D}_1, \ldots, \mathcal{D}_5$, and the performance of the system is reported in table 4.18. For the training of this system, the number of classifiers that was added to the system for each training database is $\mathbf{T}_k = \{5, 7, 10, 13, 15\}$. This dataset serves to confirm the

Table 4.18: Training and generalisation performance of system on Enzyme data

Dataset	Train 1	Train 2	Train 3	Train 4	Train 5
$\mathcal{D}_1$	13.96	18.54	17.46	16.40	14.90
$\mathcal{D}_2$	—	5.04	9.30	11.55	12.09
$\mathcal{D}_3$	—	—	11.09	12.48	0.12.79
$\mathcal{D}_4$	—	—	—	9.38	11.09
$\mathcal{D}_5$	—	—	—	—	7.44
$\mathcal{D}_{test}$	25.58	20.47	17.60	16.05	13.95

previous conclusions that the system learns new data that is presented to it, while

not suffering from catastrophic forgetting.

4.6.4 Design Strategies for use of Learn++

From the experimental results, the following strategies were decided on. In designing a system based upon Learn++, a base classifier such as MLP or SVM should be used, since these are well established and are known to give good results, especially in the context of protein classification. Fast optimisation algorithms should be used such as the scaled conjugate gradient algorithm for the MLP [47], to reduce the computational load of incrementally adding new data to the system. When incrementing the system with new classes, it is best to ensure that the number of classifiers that are added to the system is greater than the number of classifiers added during a previous system increment. It is also better to include some data from classes that have previously been seen. This will ensure that if any pattern is classified into one of the new classes, the votes from the previous classifiers do not 'outvote' votes received from new classifiers.

Learn++ has the advantage that it allows any supervised classifier to be used in an incremental manner. This is a great benefit, especially if there exists a large infrastructure, tool-set and expertise based upon a particular supervised classifier, thus allowing existing knowledge and tools to be used.

While the standard batch trained classifiers showed roughly the same accuracy as the final accuracy of the incremental system for both datasets, the incremental system would save a great deal of time and effort in training classification systems. In the batch learning approach to classifier creation, if a new family is identified and is to be added to the system, an entirely new classifier must be created which includes this new family. This training includes optimising the classifier parameters and architecture again, which involves a considerable amount of effort. In the incremental strategy, the system can learn on-line, gaining new knowledge and may demonstrate immediate results if used. It also has the advantage of not requiring previously used data, which is essential if this data is no longer available.

Comparisons of the two algorithms presented in this section will be made along with concluding remarks in the next chapter.

Chapter 5

Conclusion

5.1 Discussion and Comparison of Techniques

The previous four chapters have sketched in detail the tools and techniques that are currently used in the classification of protein primary structures into families and the introduction of two algorithms for incremental learning of this protein data. There has been a great deal of work in the classification of these proteins using a wide range of computational intelligence techniques ranging from the k-Nearest Neighbours classifiers and Naive Bayes classifiers to more complex tools such as the Multi–layer perceptron and the Support Vector Machines. While these systems have allowed a wider set of evolutionary mechanisms involving proteins to be included in the design of classification systems, such as invariance to the order of amino acid motifs in a sequence, they remain static structures which cannot incorporate newly discovered proteins into their models.

With this in mind, *Incremental Learning* was proposed as a machine learning approach to the classification of proteins. Two incremental systems were presented, the first based on an evolutionary strategy and the fuzzy ARTMAP classifier, while the second was based on the boosting algorithm Learn++. These are similar, but at the same time different approaches to incremental learning, in the sense that both systems use a committee based approach, but the fuzzy ARTMAP system uses the ensemble for performance enhancement whereas Learn++ uses the ensemble to provide the incremental learning ability.

The results presented indicate that the fuzzy ARTMAP is a suitable machine learning tool for the classification of protein sequences into structural families, which is comparable to many of the more established tools. An analysis of the sequences also shows that the system is able to classify proteins of varying lengths, and thus the length of the protein sequences used are not important. The J_3 measures further suggest that the accuracy of the classification could be improved if some form of dimensionality reduction or feature selection is applied. These techniques have been applied by many authors using numerous approaches. Principal Component Analysis is used as a technique for dimensionality reduction by Zhao et al [52] and Cheng et al [25] uses the chi-squared test as a means of feature selection. Feature selection can also be applied using various sub-optimal feature selection techniques such as the floating forward selection search using the J_3 measure as the distance function [53] or the Genetic Algorithm can be used as demonstrated by Mohamed et al [97].

For the fuzzy ARTMAP based system, the agreement κ was used to measure diversity of the system. The use of the correlation coefficient, or the use of a *disagreement* [92] should also be explored, to determine if these alternate measure gives some degree of refinement in the selection of the classifiers. The genetic algorithm is also important in the committee. Due to the stochastic nature of the GA, it is possible that different GA optimisations produce a different selection of classifier members. This though is not as likely in the case of the data presented here, since many of the classifiers had the same agreement or error, resulting in the GA converging to the same selection choice. That said, the optimisation of the GA is efficient and runs very fast due to the fact that it uses pre-calculated results such as the error matrix and agreement values. It might seem that the contribution of the GA is not significant if the case of the testing using the GPCR data is considered. This might be the case for this data, but the algorithm is designed to be generally applicable, and thus this might not be the case for another set of data, which also need not necessarily be protein data.

The incremental learning using Learn++ was also demonstrated as a suitable classification system. This approach had the advantage that it allows any supervised

classifier to be used in an incremental manner. This is desirable especially if the software infrastructure for the use of MLP say, is already in place. This property allows this infrastructure to be used, saving a great deal of development time and cost. Learn++ is also not sensitive to the order in which training examples are received and has been proven by Polikar et al on numerous benchmark databases that were used [96].

The major disadvantage is the computational cost associated with Learn++. To allow incremental learning of the data, 35 and 118 classifiers had to be generated for the system using the SCOP and GPCR data respectively, while the same performance was obtained from the single classifier trained in batch mode. While all incremental problems can be made batch problems, given that enough time is allowed to accumulate the required data, Learn++ presents a trade-off between time-to-use of the available data and the computational cost. This single property emphasises the need for further research into limiting the number of classifiers needed to achieve incremental learning, which would result in minimising the computational cost.

Learn++ also suffers from a problem of "out-voting" [98]. Consider the situation where database $\mathcal{D}_1$ has data of classes 1-10, and database $\mathcal{D}_2$ has data of classes 11-15 only. If a Learn++ system were trained with this data arrangement, the classifiers would have large voting weights with respect to their own data. If data from say class 15 is classified, the situation can easily occur where the votes from classifiers trained with $\mathcal{D}_1$ outvote the classifiers who vote for class 15. It is for this reason that the data used in the testing was ordered in such a way that it contained data of classes which had previously been added to the system. The use of a Learn++ variant, Learn++.MT, should be considered, since it has been reported to address this problem [98].

Both systems that have been described meet the criteria for an incremental classification system. They both do not suffer from catastrophic interference. While some forgetting has been observed, this has been limited and is expected due to the tradeoff between stability and plasticity of classifiers in the system. The systems also exhibit retroactive interference, where it was observed that the classification

performance on training data did not change and even improved in some cases as more data of the same class was added, as shown in tables 4.7, 4.11 and 4.17. Both systems do not have a bias to the order in which training examples are received and since they allow new classes to be added to the system, they demonstrate an open–world view.

5.2 Suggestions for Future Research

While incremental learning for biological data analysis has been shown to be suitable for the problem, a great deal of further work must be done:

- Alternate features from protein sequences should be explored that maximise the J_3 measure, allowing for easier classification.
- Testing of these systems on further sets of data such as nucleotide data should be conducted to show the applicability of these tools to problems outside proteomics.
- Feature selection is required. This will reduce the computational load of these system and possibly enhance the classification performance.
- The possible parallelisation of the system should also be investigated to allow for improved speed, which is especially needed as more data is added to the system.
- Alternate incremental classifiers should also be researched, such as incremental decision trees or completely new classifiers specifically designed and optimised for the problem of incremental learning.
- The fuzzy ARTMAP during incremental learning is sensitive to the vigilance parameter used [96], and thus it may be necessary to examine the effect of an adaptive vigilance in the incremental learning of the system [99].
- Also, in keeping with biological tendencies towards increasing modularity, it may be necessary to create modular classification architectures that focus on

small subsets of families, since classifiers trained on fewer classes can be optimised more efficiently due to the smaller degree of class overlap with fewer classes.

- In the comparative testing phases, it was found that the SVM performed better at the classification task. Future work must then consider the problem of incremental SVM for the multi-class case and the development of the mathematics for this system.

5.3 Concluding Remarks

Initial researchers into incremental learning such as Elman [100] claimed that incremental learning is always superior to batch learning. A softer approach is adopted in this research, rather emphasising that this approach is more appropriate for the case where a large number of classes exist from differing families and with different numbers of sequences per family. Many researchers have specifically looked at the classification of a single structural *superfamily*, such as the Nuclear Receptor Superfamily, and in this case both the batch and the incremental approaches to learning seem appropriate. A great deal of research has been conducted into the proteins which are members of this superfamily, and the addition of new sequences could add little value to the classification ability of the system. Where this is not the case, the incremental approach would be better.

The techniques presented here are also not limited to the problem of structural family classification, and can be easily extended to secondary and tertiary structure prediction, functional annotations and the prediction of protein–protein interaction sites. Apart from systems in proteomics, genomic applications also exist, such as the classification of promoter, and splice sites. Each classification task benefits from the improvements which can be gained from using an ensemble system and incremental learning. These results show great promise for the future of computational biology, where newly discovered data needs to be accurately incorporated into existing models, allowing for highly agile discovery processes.

Appendix A

Additional Properties of Amino Acids

The following table lists internationally approved three letter and single letter abbreviations for the standard amino acids. In the text of chapter 2, it was mentioned

Table A.1: International notations for amino acid R-groups

Amino Acid	Code (1 ch)	Code (3 ch)
Alanine	A	Ala
Arginine	R	Arg
Asparagine	N	Asn
Aspartic Acid	D	Asp
Aspartate or Asparagine†	B	Asx
Cysteine	C	Cys
Glutamine	Q	Gln
Glutamic Acid	E	Glu
Glutamate or Glutamine†	Z	Glx
Glycine	G	Gly
Histidine	H	His
Isoleucine	I	Ile
Leucine	L	Leu
Lysine	K	Lys
Methionine	M	Met
Phenylalanine	F	Phe
Proline	P	Pro
Serine	S	Ser
Threonine	T	Thr
Trytophan	W	Trp
Tyrosine	Y	Tyr
Valine	V	Val

† Ambiguous symbols which are ignored in this work

that there are 20 known amino acids. This statement is in fact not correct, as there

are now 22 known amino acids. The two new amino acids are rare, and it is for this reason that the reference to 20 amino acids is made.

The 21st amino acid was discovered in 1985 and is known as ***Selenocysteine***. It is structurally similar to *Cysteine*, but contains a selenium atom in place of the usual Sulfur atom [101]. The 22nd amino acid was discovered more recently in 2001, and is known as ***Pyrrolysine*** and is a derivative of *Lysine* [102]. Since the discovery of the amino acids are relatively new, there appearance in any current sequence databases is not expected, but it would be a simple task to expand the methods that have been presented to accommodate this expanded amino acid alphabet.

The hydropathy was used as a key feature in representing the amino acid sequences in numerical form. The Chothia and Finkelstein hydropathy classification was used and is given by the following sets, with the members of the sets represented using the single character amino acid notation.

$$Polar = \{K, R, E, D, Q, N\} \tag{A.1}$$

$$Hydrphobic = \{C, V, L, I, M, F, W\} \tag{A.2}$$

$$Neutral = \{G, A, S, T, P, H, Y\} \tag{A.3}$$

Appendix B

Theory and Principles of Fuzzy ARTMAP

B.1 Introduction

The description of the fuzzy ARTMAP presented in this appendix is a summary of the Fuzzy ARTMAP architecture described in detail by Carpenter et al [58], which was introduced in 1992. The FAM has been established as one of the few learning architectures that has fast training times, has good accuracy and incremental ability. These properties make the Fuzzy ARTMAP a unique and useful learning architecture, which has been applied to various problems in engineering such as electric load forecasting [63], prehensile EMG pattern analysis [64] and fault diagnosis of navigation systems [65], among others.

B.2 Fuzzy ARTMAP

The fuzzy ARTMAP is based on Adaptive Resonance Theory (ART). The fuzzy ARTMAP (FAM) neural network consists of two fuzzy ART modules, designated ART_a and ART_b, as well an an inter-ART module, shown in figure B.1. Inputs are presented at the ART_a module, while their corresponding outputs are presented at the ART_b module. The inter-ART module includes a MAP field whose purpose is to

determine whether the correct mapping has been established from inputs to outputs.

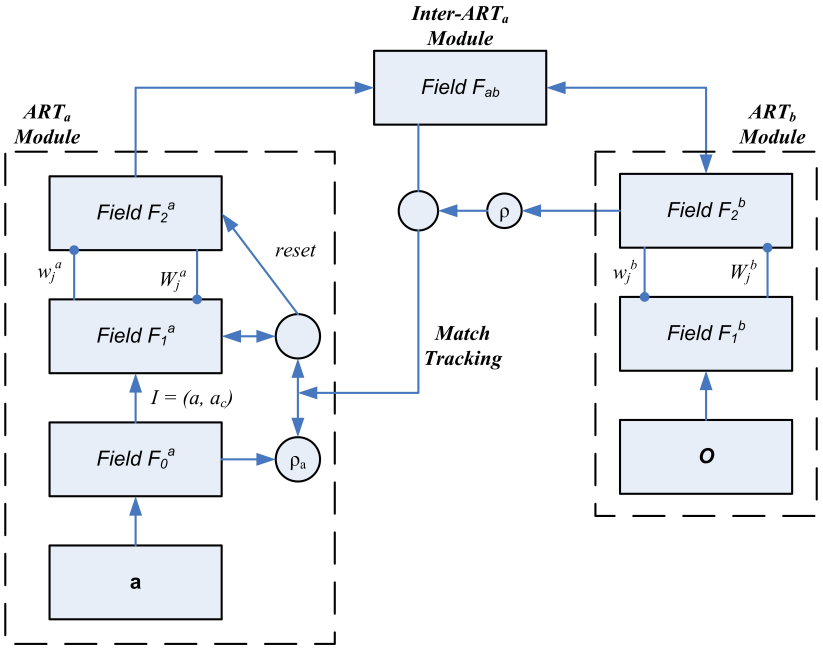


Figure B.1: Detailed structure of the fuzzy ARTMAP. Redrawn from [58].

Some pre-processing of input patterns of the pattern classification task takes place before they are presented to the ART_a module of the FAM. The first pre-processing stage takes as an input M_a–dimensional input pattern from the pattern classification task and transforms it onto an output vector $\mathbf{a} = (a_1, \ldots, a_{M_a})$, whose every component lies in the interval [0, 1], i.e. a min–max normalisation is applied. The second preprocessing stage accepts as an input the vector $\mathbf{a}$ of the first preprocessing stage and produces a vector $\mathbf{I}$ such that

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) = (a_1, \ldots, a_{M_a}, a_1^c, \ldots, a_{M_a}^c)$$

where

$$a_i^c = 1 - a_i; 1 \leq i \leq M_a$$

The above transformation is known as complement coding. Complement coding is performed in the ART_a at a preprocessor field designated by F_0^a (see figure B.1). From this point forward, the vector $\mathbf{I}$ will be referred to as the input vector. The output pattern $\mathbf{O}$ is also produced by complement coding the class labels that are applied to the ART_b module.

FAM frequently operates in two distinct phases: the training phase and the operation phase. The training phase of FAM works as follows: given a list of training input/output pairs, such as $\{\mathbf{I}^1, \mathbf{O}^1\}, \ldots, \{\mathbf{I}^r, \mathbf{O}^r\}, \ldots, \{\mathbf{I}^{NT}, \mathbf{O}^{NT}\}$, we want to train FAM to map every input pattern of the training list to its own corresponding output pattern. In order to achieve the aforementioned goal, we present the training list repeatedly to the FAM architecture. That is, present $\mathbf{I}^1$ to ART_a and $\mathbf{O}^1$ to ART_b, then $\mathbf{I}^2$ to ART_a and $\mathbf{O}^2$ to ART_b, and finally $\mathbf{I}^{NT}$ to ART_a and $\mathbf{O}^{NT}$ to ART_b; this corresponds to one list presentation. We present the training list as many times as necessary for FAM to correctly classify all the input patterns. The task is considered accomplished (i.e. learning is complete) when the weights do not change during a list presentation. This scenario is called off–line learning.

The operation phase of the FAM works as follows: given a list of test input patterns, such as $\tilde{\mathbf{I}}^1, \ldots, \tilde{\mathbf{I}}^r, \ldots, \tilde{\mathbf{I}}^{NS}$, we want to find the FAM output produced when each one of the test patterns is presented at its F_1^a field. In order to achieve this goal, we present the test list once to the trained FAM architecture.

For pattern classification problems, FAM creates clusters of input data in the input pattern space. These clusters are hyperboxes that enclose within their boundaries all the input patterns that choose them as their representative clusters. At the end of the training, the clusters (hyperboxes) created define appropriate decision regions that split the input space into subspaces that are mapped to a single output category (class). It is possible that more than one subspace of the input pattern space is mapped to the same output class.

There are a number of parameters that affect the performance of the FAM in classification problems. These are the choice parameter β_a, and the baseline vigilance parameter ρ_a. The choice parameter assumes a value in the interval $[0, 1]$, and the baseline vigilance parameter also assumes values in the interval [0, 1]. The FAM equations in which the choice parameter ρ_a appear are:

$$T_j^1(\mathbf{I}) = \frac{\left|\mathbf{I} \wedge \mathbf{w}_j^a\right|}{\beta_a + \left|\mathbf{w}_j^a\right|}$$

$$\frac{\left|\mathbf{I} \wedge \mathbf{w}_j^a\right|}{\mathbf{I}} \geq \rho_a$$

The first equation above calculates the bottom–up input applied at node j of F_2^a (choice function) due to the presentation of pattern $\mathbf{I}$ at the field F_1^a of the FAM architecture. The node in F_2^a that receives the maximum bottom–up input is chosen to represent the input pattern $\mathbf{I}$. The node thus chosen in F_2^a is considered appropriate to represent the input pattern $\mathbf{I}$ if and only if it satisfies the inequality, which is referred to as the vigilance criterion. The right hand side of the inequality is set equal to the baseline vigilance. During FAM training, the vigilance parameter value is allowed to increase above the baseline vigilance value, with the range of the vigilance parameter in the interval $[\rho_a, 1]$. Small values of the baseline vigilance parameter result in coarser clustering of the input patterns, while large values of the baseline vigilance result in finer clustering of the input pattern of the pattern classification task. The choice parameter β_a has an effect in the order according to which nodes in F_2^a will be accessed due to the presentation of an input pattern applied at F_1^a field of the FAM [62].

Another FAM parameter that is not often referred to in the literature as a parameter is the order of training pattern presentation. It has already been established that the FAM performance depends on the order in which data is presented to the FAM during the training process. As a result, FAMs performance is frequently evaluated by averaging the performance of FAM for different orders of training data presentation [62] and this is achieved in this work by training classifiers with different orderings of the input data, as described in chapter 4.

Appendix C

Learn++ Incremental Algorithm

C.1 Introduction

Learn++ was described in chapter 4, along with the pseudocode for the implementation of the Learn++ algorithm. As mentioned, Learn++ uses the ensemble system to give the incremental learning property. This appendix provides a more detailed description of the algorithm for completeness.

C.2 Details of Learn++

Learn++ is an incremental learning algorithm that uses an ensemble of classifiers that are combined using weighted majority voting. Learn++ was developed by Polikar et al [96], and was inspired by the adaptive boosting algorithm, AdaBoost. Each classifier is trained using a training subset that is drawn according to a distribution. The classifiers are trained using a weak learning algorithm. The requirement for the weak learning algorithm is that it must be able to give a classification rate of less than 50% initially. For each database $\mathcal{D}_k$ that contains training sequences, S, where S contains the pattern vector and the corresponding output class. Learn++ starts by intialising weights w according to the distribution D_T, where T is the number of hypotheses. Initially the weights are initialised according to a uniform distribution, which gives equal probability for all instances to be selected to the first

training subset. This initial distribution is given by:

$$D = \frac{1}{m}$$

where m is the number of training samples in $\mathcal{D}_k$. The training data is then divided into training subset T_R and testing subset T_E and are chosen according to the distribution. After the training and testing subset have been chosen, the weakLearn algorithm is called and trained using the testing subset T_R. A Hypothesis h_t is obtained from weakLearn, and is tested using both the training and testing subsets to obtain the error ϵ, given by

$$\epsilon_t = \sum_{t:h_t(x)\neq y} D_t(i)$$

This error must be less than $\frac{1}{2}$. A normalised error β is computed using

$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$$

If the error is greater than $\frac{1}{2}$, the hypothesis is discarded and a new training and testing subset is chosen according to D_T and a new hypothesis is created using weakLearn. All classifiers generated thus far are combined using the weighted majority voting to obtain a composite hypothesis H_t:

$$H_t = \arg\max_{y\in Y} \sum_{t:H_t(x_i)\neq y_i} D_t(i)$$

If the error is greater than $\frac{1}{2}$, then the current hypothesis is discarded and new training and testing subsets are chosen according to the distribution. If the error is less than $\frac{1}{2}$, the normalised error of the composite hypothesis is computed

$$B_t = \frac{E_t}{1-E_t}$$

The error is used in the distribution update rule where the weights of the correctly classified instances are reduced, consequently increasing the weights of the instances that were misclassified by the current hypothesis. This ensures that instances that

were misclassified by the current hypothesis have a higher probability of being selected for the subsequent training set. The distribution update rule is given by:

$$w_{t+1}(i) = w_t(i) \times B_t^{1-[|H_t(x_i) \neq y_i|]}$$

Once the T_k hypotheses are created for each database, the final hypothesis is computed by combining the composite hypotheses using the weighted majority voting given by

$$H_t = \arg\max_{y \in Y} \sum_{k-1}^{K} \sum_{t:H_t(x)=y} \log\left(\frac{1}{B_t}\right)$$

Glossary

Alignment A pairing of two homologous nucleotide or protein sequences for the purpose of identifying the location of accumulated changes since they last shared a common ancestor.

Blotting and Hybridisation The transfer of molecules from a gel onto a membrane followed by washing with a labelled probe that binds specifically to a molecule of interest.

Central Dogma of Molecular Biology Process by which information is extracted from the nucleotide sequence of a gene and then used to make a protein (DNA → RNA → Protein).

Curation The process of checking and manually updating the information concerning proteins stored within a protein database.

Enzyme A biological catalyst (usually a protein) that causes a specific chemical reaction to proceed more quickly by lowering its activation energy.

Family Consists of proteins that are more than 50% identical in amino acid sequence across their entire length.

Fuzzy ARTMAP (FAM) Classifier based on a fuzzy measure that allows multi–class classification, incremental learning and quick classification times.

Gel Electrophoresis Process in which an electric field is used to pull charged molecules (proteins) through a polyacrylamide, starch or agarose gel to separate and compare them by size and or charge.

Genetic Algorithm (GA) Evolutionary optimisation tool which allows stochastic search for global optimal solution.

Homologous Proteins These are proteins which come from a single evolutionary ancestor. These proteins tend to have similar structures and functions.

Hydrophobic Having limited interaction with water molecules; literally, "afraid of water".

Isocortex Also known as the neocortex and and is involved in higher functions such as sensory perception, generation of motor commands, spatial reasoning, conscious thought, and in humans, language.

Labelled Feature Vector A representation for the examples used to train a pattern recognition algorithm. a labelled feature vector consists of a list of feature values for the example, along with a label indicating the correct classification of the example.

Multilayer Perceptron (MLP) The most common neural network architecture that consists of an input layer, a number of hidden layers and an output layer. The classifier is trained using the popular backpropagation algorithm or its many variants.

Nearest Neighbour Classifier A statistical method that classifies objects or concepts according to similarity of their features.

Pathogen A disease causing agent.

Primary Structure Sequence in which the various amino acids are assembled into a protein.

Proteome The sum total of an organism's proteins.

Sequence (1) The linear order of nucleotides in a DNA or RNA molecule or the order of amino acids in a protein. (2) The act of determining the linear order of nucleotides or amino acids in a molecule.

Structural Protein A term used to describe proteins generally involved in maintaining a cell or tissue's shape such a those that provide rigidity and support in bones and connective tissues.

Support Vector Machine (SVM) A classifier based on statistical learning theory and the principle of finding the optimal separating hyperplane in a higher dimensional space, which is obtained by using a kernel function.

References

[1] C. W. Sensen, ed., *Handbook of Genome Research Volume 2*. New York: Wiley, 2005.

[2] P. E. Bourne and H. Weissig, eds., *Structural Bioinformatics*. New Jersey: Wiley-Liss Inc., 2003.

[3] D. E. Krane and M. L. Raymer, *Fundamental Concepts of Bioinformatics*. San Francisco: Pearson Education, 2003.

[4] J. T. L. Wang, Q. Ma, S. Shasha, and C. H. Wu, "New techniques for extracting features from protein sequences," *IBM Systems Journal*, vol. 40, no. 2, pp. 426–441, 2001.

[5] A. G. Murzin, S. E. Brenner, T. J. P. Hubbard, and C. Chothia, "SCOP: A structural classification of proteins database for the investigation of sequences and structures.," *Journal of Molecular Biology*, vol. 247, pp. 536–540, 1995.

[6] F. M. G. Pearl, D. Lee, J. E. Bray, I. Sillitoe, A. E. Todd, A. P. Harrison, J. M. Thornton, and C. A. Orengo, "Assigning genomic sequences to CATH," *Nucleic Acids Research*, vol. 28, pp. 277–282, 2000.

[7] A. D. Baxevanis and B. F. F. Oullette, eds., *Bioinformatics. A Practical Guide to the Analysis of Genes and Proteins*. New York: Wiley Interscience, third ed., 2005.

[8] C. H. Wu, H. Huang, L. L. Yeh, and W. C. Barker, "Protein family classification and functional annotation," *Computational Biology and Chemistry*, vol. 27, pp. 37–47, 2003.

[9] M. Crochemore and W. Rytter, *Jewels of Stringology: Text Algorithms*. Singapore: World Scientific, 2003.

[10] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment and search tool," *Journal of Molecular Biology*, vol. 215, pp. 403–410, 1991.

[11] P. Clote and R. Backofen, *Computational Molecular Biology: An Introduction*. New York: Wiley, 2002.

[12] L. Hunter, *Artificial Intelligence and Molecular Biology*, ch. Molecular Biology for Computer Scientists. Cambridge Massacheutets: AAAI/MIT Press, 1993.

[13] N. G. R. Institute, "Protein structures." `http://www.genome.gov/Pages/Hyperion/DIR/VIP/Glossary/ Illustration/protein.cfm`. Last Accessed: December 2006.

[14] G. P. Moss, "IUPAC nomenclature and symbolism for amino acids and peptides," 1983. `http://www.chem.qmul.ac.uk/iupac/AminoAcid/`. Last Accessed: December 2006.

[15] C. W. Sensen, ed., *Handbook of Genome Research Volume 1*. New York: Wiley, 2005.

[16] D. W. Mount, ed., *Bioinformatics: Sequence and Genome Analysis*. New York: Cold Spring Harbour, second ed., 2004.

[17] S. E. Brenner, C. Chothia, T. J. P. Hubbard, and A. G. Murzin, "Understanding protein structure: Using SCOP for fold interpretation," *Methods in Enzymology*, vol. 266, no. 1, pp. 635–643, 1996.

[18] J. M. Chandonia, G. Hon, N. S. Walkr, L. L. Conte, P. Koehl, and M. Levitt, "The ASTRAL compendium in 2004," *Nucleic Acids Research*, vol. 32, pp. 189–192, 2004.

[19] "Growth of the sequence and 3D structure databases." `http://www.genome.jp/`. Last Accessed: September 2006.

[20] "Database growth." `http://www.dna.affrc.go.jp/growth/`. Last Accessed: September 2006.

[21] F. Horn, J. Weare, M. W. Beukers, S. Hørsch, A. Bairoch, W. Chen, Ø. Edvardsen, F. Campagne, and G. Vriend, "GPCRDB: an information system for

G-protein coupled receptors," *Nucleic Acids Research*, vol. 26, no. 1, pp. 277–281, 1998.

[22] B. Qian, O. S. Soyer, R. Neubig, and R. A. Goldstein, "Depicting a proteins two faces: GPCR classification by phylogenetic tree-based HMMs," *Federation of European Biochemical Societies (FEBS) Letters*, vol. 554, pp. 95–99, 2003.

[23] K. Lundstrom, "Structural genomics of GPCRs," *Trends in Biotechnology*, vol. 23, pp. 103–108, February 2005.

[24] Y. Fang, J. Lahiri, and L. Picard, "G-protein coupled receptor microarrays for drug discovery," *Drug Discovery Today*, vol. 7, no. 16, pp. 755–761, 2003.

[25] B. Y. M. Cheng, J. G. Carbonell, and J. Klein-Seetharaman, "Protein classification based on text document classification techniques," *Proteins: Structure, Function, and Bioinformatics*, vol. 58, pp. 955–970, 2005.

[26] J. D. Westbrook, Z. Feng, L. Chen, H. Yang, and H. M. Berman, "The Protein Data Bank and structural genomics," *Nucleic Acids Research*, vol. 31, pp. 489–491, 2003.

[27] T. D. H. Bugg, *Introduction to Enzyme and Coenzyme Chemistry.* Oxford, UK: Blackwell Publishing, second ed., 2004.

[28] T. Smith and M. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, pp. 195–197, 1981.

[29] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 3, no. 48, pp. 443–453, 1970.

[30] W. Pearson and D. Lipman, "Improved tools for biological sequence comparison," *Proceedings of the National Academy of Sciences of the USA*, vol. 85, pp. 2444–2448, 1988.

[31] S. Henikoff and J. G. Henikoff, "Position–based sequence weights," *Journal of Molecular Biology*, no. 243, pp. 574–578, 1994.

[32] J. D. Thomson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.," *Nucleic Acids Research*, vol. 22, no. 22, p. 5, 1994.

[33] S. Vigna and J. Almeida, "Alignment–free sequence comparison – a review," *Bioinformatics*, vol. 19, no. 4, pp. 513–523, 2003.

[34] C. A. Orengo, D. Jones, and J. M. Thornton, *Bioinformatics: Genes, Proteins and Computers*. New York: BIOS Scientific Publishers, 2004.

[35] D. Huang, X. Zhao, and G. Huang, "Classifying protein sequences using hydropathy blocks," *Pattern Recognition*, vol. 30, pp. 2293–2300, December 2006.

[36] C. H. Q. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," *Bioinformatics*, vol. 17, no. 4, pp. 349–358, 2001.

[37] W. R. Weinert and H. S. Lopes, "Neural networks for protein classification," *Applied Bioinformatics*, vol. 3, no. 1, pp. 41–48, 2004.

[38] Y. Tang and Y. Zhang, "Granular support vector machines with association rules mining for protein homology prediction," *Artificial Intelligence in Medicine*, no. 35, pp. 121–134, 2005.

[39] A. Tomovic, P. Janicic, and V. Keselj, "N-gram-based classification and unsupervised hierachical clustering of genome sequences," *Computer Methods and Programs in Biomedicine*, vol. 81, pp. 137–153, 2006.

[40] S. Diplaris, G. Tsoumakas, P. A. Mitkas, and I. Vlahavas, "Protein classification with multiple algorithms," in *10th Panhellenic Conference on Informatics (PCI 2005)*, vol. LNCS 3746, (Greece), pp. 448–456, 11-13 November, 2005.

[41] S. K. Chalup, "Incremental learning in biological and machine learning systems," *International Journal of Neural Systems*, vol. 12, no. 6, pp. 90–127, 2002.

[42] R. Reed, "Pruning algorithms — a survey," *IEEE Transactions on Neural Networks*, vol. 4, pp. 740–747, September 1993.

[43] D. T. Larose, ed., *Discovering Knowledge in Data: An Introduction to Data Mining*. New York: Wiley Interscience, 2005.

[44] C. Giraud–Carrier, "A note on the utility of incremental learning," *AI Communications*, vol. 13, no. 4, pp. 215–223, 2000.

[45] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[46] I. T. Nabney, *Netlab: Algorithms for Pattern Recognition*. London: Springer, 2003.

[47] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford University Press, 2003.

[48] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, no. 6, pp. 525–533, 1993.

[49] D. Tikk, L. T. Kóczy, and T. D. Gedeon, "A survey of universal approximation and its applications in soft computing," *Research Working Paper RWP-IT-2001. School of Information Technology, Murdoch University, Perth*, 2001.

[50] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Berlin: Springer, second ed., 1999.

[51] N. Cristianini and J. Shawe–Taylor, *An Introduction to Support Vector Machines and other Kernel-based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000.

[52] X. Zhao, Y. Cheung, and D. Huang, "A novel approach to extracting features from motif content and protein composition for protein sequence classification," *Neural Networks*, vol. 18, pp. 1019–1028, October 2005.

[53] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. New York: Academic Press, first ed., 1999.

[54] Y. Even-Zohar and D. Roth, "A sequential model for multi-class classification," 2001. `http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0106044`. Last Accessed: December 2006.

[55] C. W. Hsu and C. J. Lin, "A comparison of methods for multi–class SVM.," *IEEE Transactions on Neural Networks*, vol. 13, pp. 415–425, 2002.

[56] R. Rifkiy and A. Klatau, "In defence of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, pp. 101–141, 2004.

[57] M. P. Perrone and L. N. Cooper, *Neural Networks for Speech and Image Processing*, ch. When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. Chapman Hall, 1993.

[58] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps.," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698–713, 1992.

[59] J. Freeman and D. Saad, "On-line learning in radial basis function networks," *Neural Computing*, no. 9, p. 16011622, 1997.

[60] J. Gomm and D. Willians, "An adaptive neural network for on-line learning and diagnosis of process faults," in *An adaptive neural network for on-line learning and diagnosis of process faults, IEE Colloquium on Qualitative and Quantitative Modelling Methods for Fault Diagnosis*, pp. 9/1 – 9/5, 1995.

[61] D. Wang and B. Yuwono, "Incremental learning of complex temporal patterns," *IEEE Transactions on Neural Networks*, vol. 7, no. 31, pp. 1465–1472, 1996.

[62] A. Koufakou, M. Georgiopoulos, A. Anagnostopoulos, and T. Kasparis, "Cross–validation in fuzzy ARTMAP for large databases," *Neural Networks*, vol. 14, pp. 1297–1291, 2001.

[63] M. L. M. Lopes, C. R. Minnussi, and A. D. P. Lotufo, "Electric load forecasting using a fuzzy ART&ARTMAP neural network," *Applied Soft Computing*, vol. 5, pp. 235 – 244, 2005.

[64] M. Vuskovic and S. Du, "Classification of prehensile EMG patterns with simplified fuzzy ARTMAP networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 2539–2544, May 2002.

[65] H. Y. Zhang, C. W. Chan, K. C. Cheung, and Y. J. Ye, "Fuzzy ARTMAP neural network and its applications to fault diagnosis of navigation systems," *Automatica*, pp. 1065 – 1070, 2001.

[66] M. Busque and M. Parizeau, "A comparison of fuzzy ARTMAP and multilayer perceptron for handwritten digit recognition," tech. rep., Universite Laval, October 1997. `citeseer.ist.psu.edu/busque97comparison.html`. Last Accessed: September 2006.

[67] N. Murshed, F. Bortolozzi, and R. Sabourin, "Fuzzy ARTMAP based classification system for detecting cancerous cells," in *Proceedings of the International Conference on Pattern Recignition (ICAPR98)*, pp. 334 – 343, Springer, 1998.

[68] E. Llobeta, E. Hinesb, J. Gardnerb, P. Bartlettc, and T. Mottramd, "Fuzzy ARTMAP based electronic nose data analysis," *Sensors and Actuators B: Chemical*, vol. 61, no. 1-3, pp. 183–190, 1999.

[69] B. Mannan, J. Roy, and A. Ray, "Fuzzy ARTMAP supervised classification of multi-spectral remotely sensed images.," *International Journal of Remote Sensing*, vol. 19, no. 4, pp. 334 – 343, 1998.

[70] X. Song, P. Hopke, M. Bruns, D. Bossio, and K. Scow, "A fuzzy adaptive resonance theory-supervised predictive mapping neural network applied to the classification of multivariate chemical data," *Chemometrics and Intelligent Laboratory Systems*, vol. 41, no. 2, pp. 161–170, 1998.

[71] M. Guetova, S. Holldobler, and H. Storr, "Incremental fuzzy decision trees," in *Lecture Notes in Computer Science: Advances in Artificial Intelligence: 25th Annual German Conference on AI*, vol. 2479/2002, (Aachen, Germany), p. 67, September 2002.

[72] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," *Advances in Neural Information Processing Systems (NIPS 2000)*, vol. 13, pp. 409–415, 2001.

[73] N. A. Syed, H. Liu, and K. K. Sung, "Handling concept drifts in incremental learning with support vector machines," in *Proceedings of the ACM Conference on Knowledge Discovery and Data mining*, (San Diego, USA), pp. 317 – 321, 1999.

[74] A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi, "Incremental training of support vector machines," *IEEE Transactions of Neural Networks*, vol. 16, no. 1, pp. 114–131, 2005.

[75] S. Rüping, "Incremental learning with support vector machines," in *First IEEE International Conference on Data Mining (ICDM'01)*, p. 641, 2001.

[76] C. Giraud-Carrier and T. Martinez, "An incremental learning model for commonsense reasoning," Tech. Rep. CS-EXT-1994-035, ACM, University of Bristol Bristol, UK, 1994.

[77] D. Caragea, A. Silvescu, and V. Honavar, *Architectures for Intelligence*, ch. Learning in open-ended environments: Distributed learning and incremental learning. Springer-Verlag, 2001.

[78] S. Vijayakumar and H. Ogawa, "RKHS-based functional analysis for exact incremental learning," *Neurocomputing*, vol. 29, pp. 85 – 113, 1999.

[79] G. G. Yen and P. Meesad, "An effective neurofuzzy paradigm for machine condition health monitoring," *IEEE Transactions on Systems, Man and Cybernetics: Part B*, vol. 31, pp. 523–536, 2001.

[80] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer, third, revised and extended ed., 1999.

[81] D. E. Goldberg, *Genetic Algorithms in Search, Optimisation & Machine Learning*. New York: Addison–Wesley Publishing Company, Inc., 1989.

[82] R. L. Haupt and S. E. Haupt, eds., *Practical Genetic Algorithms*. New York: Wiley Interscience, second ed., 2004.

[83] P. A. Vijaya, M. N. Murty, and D. K. Subramanian, "An efficient incremental protein sequence clustering algorithm," in *Proceedings of the IEEE Conference on Convergent Technologies for Asia-Pacific Region*, pp. 409–413, October, 2003.

[84] C. Huang, C. T. Lin, and N. R. Pal, "Hierarchical learning architecture with automatic feature selection for multi-class protein fold classification," *IEEE Transactions on Nanobioscience*, vol. 4, pp. 221–232, December 2003.

[85] I. Dubchak, I. Muchnik, S. R. Holbrook, and S. Kim, "Prediction of protein folding class using global description of amino acid sequence," in *Biophysics*, vol. 92, (USA), pp. 8700–8704, National Academy of Science, September, 1995.

[86] V. Keselj, F. Peng, N. Cerone, and C. Thomas, "N-gram based author profiles for authorship attribution," in *Pacific Association for Computational Linguistics*, (Halifax, Canada), pp. 255 – 264, PACLING'03, 2003.

[87] L. French, A. Ngom, and L. Rueda, "Fast protein superfamily classification using principal component null space analysis," in *18th Canadian Conference of AI*, (Canada), pp. 163 – 172, AI 2005, 2005.

[88] L. O. Hall, "Reusing information by learning models from extreme data sets," in *Proceedings of the IEEE International Conference on Information Reuse and Integration*, pp. viii – viii, August 2004.

[89] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI repository of machine learning databases," 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Last Accessed: September 2006.

[90] C. J. Merz, "Using correspondence analysis to combine classifiers," *Machine Learning*, vol. 36, no. 1-2, pp. 33–58, 1999. `citeseer.ist.psu.edu/merz98using.html`. Last Accessed September 2006.

[91] M. Petrakos, J. A. Benediktsson, and I. Kannellopoulos, "The effect of classifier agreement on the accuracy of the combined classifier in decision level fusion," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, pp. 2539 – 2546, November 2001.

[92] Y. M. Bishop, S. E. Fienberg, and P. W. Holland, *Discrete Multivariate Analysis*. Massachusetts: MIT Press, 1977.

[93] A. Garrett, "Fuzzy ART and fuzzy ARTMAP neural networks." http://www.mathworks.com/matlabcentral. Last Accessed: December 2006.

[94] "OSU SVM toolbox for MATLAB." http://sourceforge.net/projects/svm/. Last Accessed: December 2006.

[95] C. R. Houck and J. A. J. amd M. G. Kay, "Genetic algorithm optimisation toolbox (GAOT)." http://www.ie.ncsu.edu/mirage/GAToolBox/gaot. Last Accessed: December 2006.

[96] R. Polikar, L. Udpa, A. S. Udpa, and V. Hanovar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 31, no. 4, pp. 497–508, 2001.

[97] S. Mohamed, D. Rubin, and T. Marwala, "Multi-class protein classification using Fuzzy ARTMAP," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, (Taipei, Taiwan), pp. 1676–1681, 8 - 11 October 2006.

[98] M. Muhlbaier, A. Topalis, , and R. Polikar, "Learn++.MT: A new approach to incremental learning," in *Lecture Notes in Computer Science: Multiple Classifier Systems*, vol. 3077/2004, pp. 52–61, 2004.

[99] A. Canuto, M. Fairhurst, and G. Howells, "Improving ARTMAP learning through variable vigilance," *International Journal of Neural Systems*, vol. 11, no. 6, pp. 509–522, 2001.

[100] J. L. Elman, "Learning and development in neural networks: The importance of starting small," *Cognition*, vol. 48, pp. 71–99, 1993.

[101] A. Bock, K. Forchhammer, J. Heider, W. Leinfelder, G. Sawers, B. Veprek, and F. Zinoni, "Selenocysteine: the 21st amino acid," *Molecular Microbiology*, vol. 5, no. 3, pp. 515–520, 1991.

[102] "22nd amino acid made directly." `http://pubs.acs.org/cen/news/8233/8233amino.html`. Last Accessed: September 2006.

Bibliography

- T. Pavlidis, *Structural Pattern Recognition*. Berlin: Springer–Verlag, 1977.
- J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro–Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. New Jersey: Prentice Hall, 1997.
- R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis. Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, UK: Cambridge University Press, 1998.
- A. Ceroni, P. Frasconi, and G. Pollastri, "Learning protein secondary structure from sequential and relational data," *Neural Networks*, vol. 18, pp. 1029–1039, October 2005.
- J. T. L. Wang, Q. Ma, S. Shasha, and C. H. Wu, "Application of neural networks to biological data mining: A case study in protein sequence classification," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, (Boston), pp. 305–309, ACM, 2000.
- A. Ben-Hur and D. Brutlag, "Remote homology detection: a motif based approach," *Bioinformatics*, vol. 19, no. 1, pp. i26–i33, 2003.
- E. Zinzaras, N. P. Brown, and A. Kowald, "Non–parametric classification of protein secondary structures," *Computers in Biology and Medicine*, vol. 36, no. 2, pp. 145–156, 2006.
- K. A. Olszewski, L. Yan, D. Edwards, and T. Yeh, "From fold recognition to homology modelling: an analysis of protein modelling challenges at different levels of prediction complexity," *Computers and Chemistry*, vol. 24, pp. 499–510, 2000.

- S. Thurner, M. C. Feurstei, S. B. Lowen, and M. C. Teich, "Receiver-operating characteristic analysis reveals superiority of scale-dependant wavelet and spectral measures for assessing cardiac dysfunction," *Physical Review*, vol. 81, no. 25, pp. 5688 – 5691, 1998.

- N. Nagarajan and G. Yona, "A multi-expert system for the automatic detection of protein domain from sequence information," in *RECOMB '03*, (Berlin), pp. 224 –234, ACM, April 2003.

- S. F. Smith, "Protein family classification using structural and sequence information," in *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 168 – 174, IEEE CIBCB, October 2004.

- S. F. Smith, "A framework for protein classification," in *Proceedings of the German Conference on Bioinformatics*, vol. II, pp. 55–57, 2003.

- T. Jaakola, M. Diekhans, and D. Haussler, "A discriminative framework for detecting remote protein homologies," *Journal of Computational Biology*, vol. 7, no. 1/2, pp. 95–114, 2000.

- C. C. Aggarwal, "On effective classification of strings with wavelets," in *SIGKDD '02 - Special Interest Group on Knowledge Discovery and Data Mining*, (Canada), pp. 163 – 172, ACM, 2002.

- A. Bariroch, R. Appweiler, C. H. Wu, W. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donnovan, N. Redaschi, and L. Yeh, "The universal protein resource (uniprot)," *Nucleic Acids Research*, vol. 33, pp. D154 – D159, 2005.

- J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," Tech. Rep. 97-02A, Artificial Intelligence Research Group, Department of Computer Science, Iowa State University, May, 3 1997.

- G. Lubec, L. Afjehi-Sadat, J. Yang, and J. P. P. John, "Searching for hypothetical proteins: Theory and practice based on original data and literature," *Progress in Neurobiology*, vol. 77, pp. 90–127, 2005.

- G. Fung and O. L. Mangasarian, "Incremental support vector machine classification," in *Second SIAM International Conference of Data Mining*, p. 15, April 2002.

- K. Kelly, "Newly discovered protein kills anthrax bacteria by exploding their cell walls," April 2006. Medical News Today, `http://www.medicalnewstoday.com`, Last Accessed 21 June 2006.

- C. Binette, "Newly discovered protein could hold key to preventing heart disease," May 2006. Medical News Today, `http://www.medicalnewstoday.com`, Last Accessed 21 June 2006.

2006 IEEE Conference on
Systems, Man, and Cybernetics
October 8-11, 2006, Taipei, Taiwan

Multi-class Protein Sequence Classification Using Fuzzy ARTMAP

Shakir Mohamed, David Rubin and Tshilidzi Marwala

***Abstract*— The classification of protein sequences into families is an important tool in the annotation of structural and functional properties to newly discovered proteins. We present a classification system using pattern recognition techniques to create a numerical vector representation of a protein sequence and then classify the sequence into a number of given families. We introduce the use of fuzzy ARTMAP classifiers and show that coupled with the genetic algorithm based feature subset selection, the system is able to classify protein sequences with an accuracy of 93 %. This accuracy is compared with numerous other classification tools and demonstrates that the fuzzy ARTMAP is suitable due to its high accuracy, quick training times and ability for incremental learning.**

I. Introduction

Biosequence analysis has received increased attention in recent years since the completion of the human genome project. As a subfield, protein sequence analysis has also become important due to its application in drug discovery programs [1] and in the analysis of prion diseases. Proteins are linear polymers or chains of amino acids and the representation of this amino acid sequence is known as the protein primary structure. All proteins are constructed from a 20-letter amino acid alphabet $\mathcal{AA} = \{$A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y$\}$. Each amino acid has specific properties such as charge and polarizability, isoelectric properties and hydropathy properties. These properties allow a protein to be represented by a measure of these properties.

In light of recent advances in computational intelligence tools and an understanding that contiguity between amino acid sequences between proteins cannot be assumed, this paper aims to add to the set of available tools for the classification of protein sequences into families. The paper demonstrates the use of fuzzy ARTMAP's as an alternative machine learning tool that provides improved classification ability, but also improved training and classification times.

Section II provides a background as to the large number of available protein databases and the approaches to protein sequence classification. Section III, then provides an overview of the classification system with section IV, VI and VII discussing the feature generation and the testing of the system. Finally, section VIII provides a discussion of the results and conclusions are made in section IX.

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the authors and not to be attributed to the NRF.

Authors are with the School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, South Africa. {d.rubin,t.marwala}@ee.wits.ac.za

II. Background

A. Protein Databases

The protein community has over the years established many publicly available protein–information related databases. Some of these include the well known Protein Data Bank (PDB) and UniProt among others, with each of these databases serving a particular segment of the protein analysis community. The protein database used in this work is the Structural Classification of Proteins (SCOP) [2] version 1.69. This database organises protein sequences according to a hierarchy of protein classes, folds, superfamilies and families; with each level of the hierarchy showing a different type of structural relationship between individual protein sequences.

The SCOP database only consists of classifications of proteins according to a protein sequence ID, but does not supply any of the sequences. The sequences are obtained from the ASTRAL Database 1.69. The database with no more than 95% sequence identity was selected for use in this work.

B. Alignment–Based Sequence Comparison

The problem of classifying protein sequences is not new, and major work in this area applies the use of sequence alignment techniques. These techniques operate on the principle of detecting sequence homologies or conserved regions of protein sequences in a set of available protein sequences. Essentially what these techniques do is align a sequence under test, either pairwise or by multiple alignment with a set of known protein sequences. A distance measure between the sequence under test and each of the alignments is calculated and the sequence to which the given sequence has the smallest distance is the closest matching protein sequence. Thus the sequence belongs to the family of the closest matching sequence. Many alignment–based techniques have been developed, most notably are the Basic Local Alignment and Search Tool – BLAST, FASTA and position specific weight matrices. A comprehensive review of these techniques is given by Durbin [3].

While these techniques have proven a degree of success, they are limited by the high computational load when the number of sequences in the database against which the search is applied is large, and is also dependent on the length of individual sequences which may vary considerably between proteins [4]. Another drawback of the alignment based approach is the assumption that the order of amino acids in a sequence is conserved between homologous segments [4], which is not in line with the genetic recombination and

1-4244-0100-3/06/$20.00 ©2006 IEEE

divergent evolution which may occur during the evolution of a protein. These problems have given rise to the alignment–free techniques.

C. Alignment Free Sequence Comparison

The alignment–free techniques consist of representing individual protein sequences by a vector of numerical measures that capture the essence of the sequence. These measures can be used in conjunction with a wide range of pattern recognition and data mining tools in order to generate models of sequence relationships and to classify the protein sequences into a wide range of groupings. A review of these techniques is provided by Vigna and Almeida [4].

III. System Overview

The question being addressed here is: given an unknown protein sequence $\mathcal{S}$ and a set of known protein families $\mathcal{F}$, we must classify this sequence into one of the families in $\mathcal{F}$. In general, sequences from the same family have similar structural and functional properties. Thus if an unlabeled sequence $\mathcal{S}$ is found to belong to some family $\mathcal{F}_i$, we can infer the structure and function of $\mathcal{S}$. An overview of the pattern recognition process is shown in figure 1.

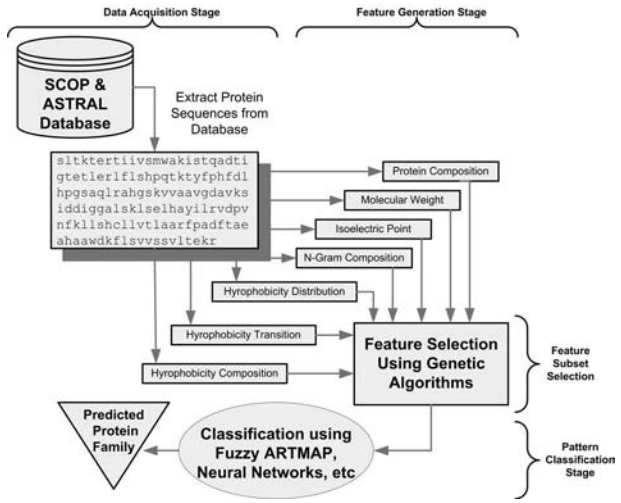


Fig. 1. Overview of Sequence Classification System

This figure shows the well known pattern recognition process, which begins with the data acquisition stage. This process involves extracting all the sequence classification information in the SCOP database for which we have the sequences in the ASTRAL database. This set of sequences is then converted to a vector based representation. The resulting vector has a large dimensionality which is reduced by using the genetic algorithm to stochastically search for the best subset of features. The resulting feature subset is then classified using a wide range of techniques. In this paper we introduce the Fuzzy-ARTMAP classifier and demonstrate the benefits of using the classifier over other established classifiers in the current literature. To the best of our knowledge, the fuzzy ARTMAP has not been considered before in the context of protein sequence family classification and could prove useful in other areas of computational biology.

IV. Protein Data Analysis

A. Sequence Extraction and Preprocessing

As mentioned, the SCOP database 1.69 and the corresponding ASTRAL database with less than 95% sequence identity is used. As a result of removing the highly redundant sequences from the Astral database, the corresponding SCOP entries had to be extracted and matched against the sequence ID in the Astral database. The SCOP database has just over 70,000 entries, while the Astral file has about 12,000 entries, resulting in the matching process between the two databases being a lengthy procedure.

Once this matching is completed, "outlier" sequences must be removed. Two types of outliers were observed from the available databases. The first is that there exists certain Astral sequences for which there is no corresponding SCOP entry. There were a total of 345 such sequences and these were removed from the sequence list. The second type of outlier consists of sequences which have characters which are not part of the standard 20-letter amino acid alphabet — the letters are B and Z and have ambiguous meanings. From the data, 120 such sequences were observed and were also removed from the sequence list.

For this work, only a subset of the available families were considered for the experimental work. Table I shows the table of protein families considered and the number of sequences in each family that were extracted from the Astral Database. A total of 8 protein families or classes

TABLE I
SCOP 1.69 protein families considered in this work

#	SCOP Family	PDB ID	# Seq.
1	Phycocyanin-like phycobilisome proteins	46532	23
2	monodomain cytochrome c	46627	48
3	Glutathione S-transferase (GST), N-terminal domain	52862	45
4	Calmodulin-like	47502	53
5	Nucleosome core histones	47114	23
6	Tyrosine-dependent oxidoreductases	51751	68
7	Crystallins/Ca-binding development proteins	49696	21
8	Alcohol dehydrogenase-like, N-terminal domain	50136	29

will be used in this work. These protein sequences must be transformed into numerical features. Two types of features have been identified in the literature, these being global and local features. Huang *et al* [5] provide a good description of the difference between global and local features and this distinction is used in this work.

B. Global Feature Generation

Global features represent the nature of the entire protein sequence. These features must capture the global similarity between related sequences allowing for comparison. We consider two types of global features in this work. The first type are physico-chemical properties of the sequences which are known from biology. We specifically consider the

molecular weight (W) and the isoelectric point (pI) of the proteins sequences.

Apart from these two features, we also consider the amino-acid composition of the sequence. The composition is simply the presence frequency of each of the 20-possible amino acids in the given sequence. Thus the composition is calculated by [6]:

$$\nu_i = \frac{s_i}{\sum_{j=1}^{20} s_j}, \text{ for } i = 1, 2, \ldots, 20. \quad (1)$$

where ν_i is the value for the i*th* feature and s_i is the number of times the i*th* amino acid appears in the sequence. This results in 20 features: a frequency of appearance for each of the possible amino acids. If a particular amino acid does not appear at all in the sequence, the corresponding feature value is zero.

A third set of features based on the hydropathy of amino acids in a given protein sequence is also calculated. The hydropathy of a protein sequence is the attractive property of an amino acid to a water molecule. Amino acids are thus Hydrophobic, Hydrophilic (polar) or neutral. We use the Chothia and Finkelstein [7] hydropathy classification. We calculate three descriptors, the hydropathy composition ($\mathcal{C}$), the hydropathy distribution ($\mathcal{D}$) and the Hydropathy transmission ($\mathcal{T}$) for the sequences as described by Dubchak [7].

The composition $\mathcal{C}$ is calculated similarly to the amino acid composition described previously. In this case we calculate the presence frequency of hydrophobic, hydrophilic and neutral amino acids in the sequence. This results in three features being generated. The transmission is defined by three values. The first is the number of times a polar molecule is followed by a neutral molecule or vice versa. Similarly the other two are the number of times a neutral molecule is followed by a hydrophobic molecule or vice versa and the number of times the polar molecule is followed by a hydrophobic molecule or vise versa.

The distribution looks at intervals of 25%, 50%, 75% and 100% along the sequence length. For each interval the presence frequency of hydrophobic, hydrophilic and neutral molecules for each percentage interval is calculated. Thus this results in 12 features, 4 features for each of the three hydropathy groups. A more detailed description of these features can be found in Dubchak [7]. In total 40 features (2+20+3+3+12) are generated based on global sequence descriptors.

C. Local Feature Generation

The local features capture local interactions between amino acids and groups of amino acids in a protein sequence. The n-gram method is well established as a good descriptor of local similarities in a sequence and has been used by many authors such as Tomovic *et al* [8] and Cheng *et al* [1]. Essentially the n-gram method considers the presence frequency of consecutive n-letter combinations in the protein sequence, for integer n. For example, consider the short sequence `SLTKTERTIIVSM`, the 2-grams of this sequence are: `SL`, `LT`, `TK`, `KT`, etc. Given a sequence, features are generated by calculating the presence frequency of all possible n-grams for the amino acid alphabet. Two letter combinations are known as digrams or bigrams. While higher n-grams such as 3-grams and 4-grams have been considered in the available literature, only digrams are considered in this work since it has been proven by numerous authors [1], [9] to work well in protein classification systems. This results in 400 features representing the local properties of the amino acids. The combined set of features results in a feature vector with a dimensionality of 440. This feature vector is now used in the design of the protein family classifiers.

V. Protein Family Classification

A. Current Classification Tools in Use

The feature based approach to protein sequence classification makes possible the use of a wide range of classification tools. Most protein databases supply Hidden Markov Models (HMM) for each of the families in the database, and the HMM's can be used to determine which family an unknown sequence belongs to. More recently, the use of Multi–Layer Perceptron (MLP) Neural Networks has been introduced to the problem of classification. Neural networks have been applied by authors such as Dubchak [7], Nagarajan *et al* [10] and Weinert and Lopes [11]. Each has shown success in the areas of domain detection or protein folding prediction. Other types of classifiers have also been used. Zhao *et al* [6] have made use of the Support Vector Machines while Radial Basis Function (RBF) Neural Networks and k–Nearest Neighbour (k-NN) classifiers have also been used [12].

B. Fuzzy ARTMAP's for Classification

This paper introduces the Fuzzy ARTMAP as a classifier for the protein classification task. The fuzzy ARTMAP is based on adaptive resonance theory and was introduced by Carpenter *et al* [13]. This learning system is built upon two fuzzy ART modules and employs calculus based fuzzy operations in the learning procedure. A diagram showing the structure of a fuzzy ARTMAP system is shown in figure 2.

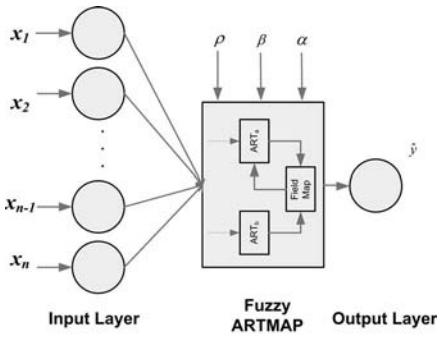


Fig. 2. Representation of the Fuzzy ARTMAP Architecture

The fuzzy ARTMAP, divides the input feature space into a number of hyperboxes in the n-dimensional space. It contains a map field which maps the individual hyperboxes to the output classes of the classification system. As a result, the

fuzzy ARTMAP is able to model complex input spaces well. It requires two variables, where the vigilance parameter ρ, represents the tradeoff between classification accuracy and incremental learning ability. The learning rate β, is a factor by which the hyperboxes are adjusted with each training pattern during the training phase. In this system, $\beta = 1$, which is known as fast learning. Further details on the Fuzzy ARTMAP and its training can be found in [13].

C. Experimental Procedure

In testing the system, two experiments are conducted. The first experiment considers comparing the classification ability of the fuzzy ARTMAP against a wide range of other classifiers. In this experiment all the features which have been generated are used as inputs to the system.

In the second experiment, wrapper based feature selection is employed. It is well known that large dimensional feature vectors may contain many redundant features which may 'confuse' a classifier and result in reduced classifier accuracy, increased training time and increased number of required training patterns [14]. To observe the performance benefits of feature subset selection, the approach is applied and compared using fuzzy ARTMAP's, MLP neural networks and the k-Nearest Neigbour classifiers.

VI. Family Classification Without Feature Selection

A. Overview of Experiment

This experiment involves using the *entire* set of generated features and comparing the performance of a wide range of classifiers to determine the effectiveness of the fuzzy ARTMAP in relation to these well known classifiers. This initial performance is a good benchmark of the viability of fuzzy ARTMAP.

B. Results

In this experiment, the fuzzy ARTMAP is compared with five other classifiers. The classifiers are the Generalised Linear Model, the k–Nearest Neighbour Classifier, the Multi–Layer Perceptron Neural Network and the Radial Basis Function Neural Network.

The Generalised Linear Models (GLM) are single layer neural network models which implement well known statistical functions such as regression. They are trained using the Iterated Reweighted Least Squares Algorithm (IRLS), and is used here to demonstrate the lack of linear relationships between the input features and the poor performance if such an assumption is made. The k–Nearest Neighbour classifier is used with the number of neighbours, $k = 1$. The MLP networks are trained with 25 nodes in the hidden layer. The network is trained using the scaled conjugate gradient algorithm with weight decay. The weight decay hyperparameter α was set to 0.01. The Radial Basis Function neural networks were tested using three different kernel functions, viz. Gaussian, Thin Plate Spline and the logarithmic kernel functions. The fuzzy ARTMAP's were trained with a vigilance parameter of $\rho = 0.5$. Details of the alternative machine learning tools discussed here can be found in [15].

Table II shows the classification performance of each of the classifiers described above. Each was tested using 10-fold cross validation. The performance measure is the error rate ϵ, which is defined as:

$$\epsilon = \frac{\text{Number of Misclassifications}}{\text{Total Number of Patterns}} \tag{2}$$

TABLE II
Comparison of performance of different classifiers using all generated features

Classifier	Error Rate
GLM	0.9000
RBF – Gaussian Kernel	0.1692
RBF – Thin Plate Spline	0.1762
RBF – $r^4 logr$	0.1577
MLP	0.1458
k-Nearest Neighbour	0.1516
fuzzy ARTMAP	0.1290

VII. Genetic Algorithm Based Feature Selection

A. Selection Process Overview

As mentioned, high dimensional feature vectors increase the costs in terms of computation times, number of patterns required and in the accuracy of classification systems. In order to avoid these problems, adequate feature selection techniques are required that will reduce the dimensionality of the feature vectors, but still retain the class discriminative ability of the features.

Feature selection techniques can be grouped into two classes – filter or wrapper approaches [14]. The wrapper based approach is used in this work The wrapper based approaches incorporate a learning algorithm in the feature selection process and generally exhibit high classification accuracies due to the selection of features which maximise the performance of a chosen classifier. The genetic algorithm is used here as the search tool.

The genetic algorithm (GA) is a tool that mimics concepts from evolutionary biology such as natural selection, crossover and mutation. The stochastic nature of the GA makes it a good tool for feature selection, since it can quickly explore the possible search space and determine the global maximum of a fitness function. We compare this feature selection process between the fuzzy ARTMAP, the k-Nearest Neighbour classifier and a Multi-Layer Perceptron Neural network to explore the performance increase in using a subset of features. The desired number of features to be selected from the original feature vector is used as the number of genes in a chromosome. Each gene in this chromosome is an integer between 1 and 440, and represents the feature number which is selected for a particular generation. Points along the chromosomes are selected for crossover and mutation. The genes selected at each of the points are altered using floating point operations, effectively causing a different set of features to be chosen. This process is repeated for a fixed number of

generations resulting in the determination of the best feature subset for the required number of features.

B. Experimental Results

The feature selection process was performed using the multi–layer perceptron (MLP) neural network, the k-Nearest Neighbour Classifier and the fuzzy ARTMAP as the learning algorithm. The required number of features was varied from 10 features to 100 features. The error rate was used as the GA fitness function to be minimised and was calculated using 10-fold cross-validation. Each of the selection procedures was repeated 5 times, and the resulting mean accuracy of the selected features and the standard deviation are shown in figure 3.

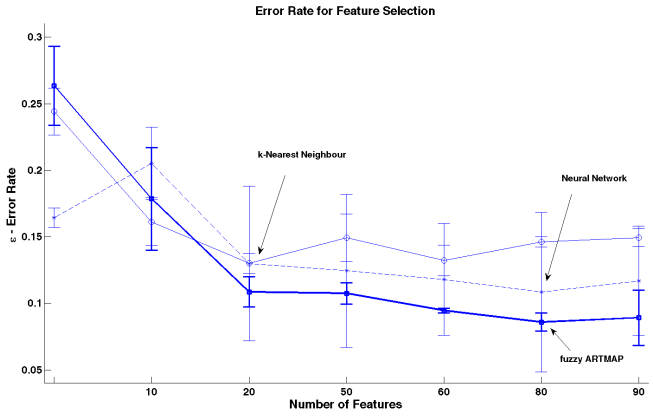


Fig. 3. Feature selection Results Fuzzy ARTMAP and MLP

From the graph it is clear that 90 features is the optimal feature set that is chosen by both the fuzzy ARTMAP system and the MLP system, while the k-NN has selected 50 features as the optimal. A separate MLP, k-NN and fuzzy ARTMAP network was trained using this selected feature subset using 10–fold cross-validation. It was found that the MLP has an error rate of 0.1084, the k-NN an error rate of 0.1097 and the fuzzy ARTMAP an error rate of 0.0654 over the 10–folds.

VIII. Discussion and Future Work

The results presented show in both that the fuzzy ARTMAP provides improvement over many other classification systems. This clear benefit is due to the way in which the fuzzy ARTMAP creates a mapping of the input feature space, by the division of the input space into hyperboxes.

Furthermore, the fuzzy ARTMAP is also preferred since it has very quick training times, and this is due to the fact that a number of hyperboxes are created in the space and related to each other by the mapping field. In the experiments conducted here, the average training and testing time for the 10-fold cross validation was $5.77s$ for the fuzzy ARTMAP and the k-NN has a time of $0.984s$, as opposed to the training and learning time of $95.953s$ for the MLP networks. This fast training time of the fuzzy ARTMAP is desired especially for a system where a much larger set of data is used than that used here.

Another advantage of the fuzzy ARTMAP over other learning systems is its ability for incremental learning. Incremental learning is the addition of knowledge to a previously trained system. This property is desired since new protein sequences are being discovered every day, and whose knowledge can be added to the system as it becomes available. This would allow more accurate protein classifications and is work which has been identified for future exploration. Further work also includes a further analysis and comparison of the system to a wider range of classifiers and the use of alternate features such as sequence entropy for classification.

IX. Conclusion

The successful classification of a protein sequence into a number of known protein families has been demonstrated using the fuzzy ARTMAP. This system has been used with a Genetic Algorithm Based Feature selection to obtain a subset of features that improves the classification ability of the classifier. This system has been compared to MLP and a number of other classification systems, and has proven that the fuzzy ARTMAP provides classification improvement over these systems. The fuzzy ARTMAP is desirable as a classifier because of its fast training times and its incremental learning ability, an area which will be explored in future.

References

[1] B. Y. M. Cheng, J. G. Carbonell, and J. Klein-Seetharaman, "Protein classification based on text document classification techniques," *Proteins: Structure, Function, and Bioinformatics*, vol. 58, pp. 955–970, 2005.

[2] A. G. Murzin, S. E. Brenner, T. J. P. Hubbard, and C. Chothia, "SCOP: a structural classification of proteins database for the investigation of sequences and structures.," *Journal of Molecular Biology*, vol. 247, pp. 536–540, 1995.

[3] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis*. Cambridge University Press, 1998.

[4] S. Vigna and J. Almeida, "Alignment–free sequence comparison – a review," *Bioinformatics*, vol. 19, no. 4, pp. 513–523, 2003.

[5] C. Huang, C. T. Lin, and N. R. Pal, "Hierarchical learning architecture with automatic feature selection for multi-class protein fold classification," *IEEE Transaction of Nanobioscience*, vol. 4, pp. 221–232, December 2003.

[6] X. Zhao, Y. Cheung, and D. Huang, "A novel approach to extracting features from motif content and protein composition for protein sequence classification," *Neural Networks*, vol. 18, pp. 1019–1028, October 2005.

[7] I. Dubchak, I. Muchnik, S. R. Holbrook, and S. Kim, "Prediction of protein folding class using global description of amino acid sequence," in *Biophysics*, vol. 92, (USA), pp. 8700–8704, National Academy of Science, USA, September, 1995.

[8] A. Tomovic, P. Janicic, and V. Keselj, "n-gram-based classification and unsupervised hierachical clustering of genome sequences," *Computer Methods and Programs in Biomedicine*, vol. 81, pp. 137–153, 2006.

[9] D. Huang, X. Zhao, and G. Huang, "Classifying protein sequences using hydropathy blocks," *Pattern Recognition*, vol. In Press, Corrected Proof, Online: 19 January 2005. URL: `http://www.sciencedirect.com`, Last Accessed: 8 February 2006.

[10] N. Nagarajan and G. Yona, "A multi-expert system for the automatic detection of protein domain from sequence information," in *RECOMB '03*, (Berlin), pp. 224 –234, ACM, April 2003.

[11] W. R. Weinert and H. S. Lopes, "Neural networks for protein classification," *Applied Bioinformatics*, vol. 3, no. 1, pp. 41–48, 2004.

[12] S. Diplaris, G. Tsoumakas, P. A. Mitkas, and I. Vlahavas, "Protein classification with multiple algorithms," in *10th Panhellenic Conference on Informatics (PCI 2005)*, vol. LNCS 3746, (Greece), pp. 448–456, 11-13 November, 2005.

[13] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps.," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698–713, 1992.
[14] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," Tech. Rep. 97-02A, Artificial Intelligence Research Group, Department of Computer Science, Iowa State University, May, 3 1997.
[15] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. New York: Academic Press, first ed., 1999.

17th Annual Symposium of the
Pattern Recognition Association of South Africa
29 Nov - 1 Dec 2006, Parys

Adaptive GPCR Classification Based on Incremental Learning

Shakir Mohamed, David Rubin, Tshilidzi Marwala

School of Electrical and Information Engineering
University of the Witwatersrand, Johannesburg
{s.mohamed, d.rubin, t.marwala}@ee.wits.ac.za

Abstract

The problem of protein structural family classification remains a core problem in computational biology, with application of this technology applicable to problems in drug discovery programs and hypothetical protein annotation. Many machine learning tools have been applied to this problem using static machine learning structures such as neural networks or support vector machines that are unable to accommodate new information into their existing models. We utilize the fuzzy ARTMAP as an alternate machine learning system that has the ability of incrementally learning new data as it becomes available. The fuzzy ARTMAP is found to be comparable to many of the widespread machine learning systems. The use of an evolutionary strategy in the selection and combination of individual classifiers into an ensemble system, coupled with the incremental learning ability of the fuzzy ARTMAP is proven to be suitable as a pattern classifier. The algorithm presented is tested using data from the G-Coupled Protein Receptors Database and shows good accuracy of 83%. The system presented is also generally applicable, and can be used in problems in genomics and proteomics.

Keywords: Bioinformatics, GPCR, Incremental Learning, Fuzzy ARTMAP

1. Introduction

Biosequence analysis has received increased attention in recent years since the completion of the human genome project. As a sub-field, protein sequence analysis has also become important due to its application in drug discovery programs [1] and in the analysis of prion diseases. The benefits of a computational analysis of biological systems is most clear when analysing the process of drug design. The development of new drugs often takes up to 15 years and costing up to $700 million per drug under investigation [1]. This drug design consists of two phases: a discovery phase and testing phase [2]. It is in this drug discovery phase that computational tools have had the most impact. In pharmaceutical drug discovery programs it is often useful to classify the sequences of proteins into a number of known families. In a mathematical notation, if it is known that a sequence $\mathcal{S}$ is obtained for some disease $\mathcal{X}$, and that $\mathcal{S}$ belongs to family $\mathcal{F}$, treatment for the disease is initially determined using a combination of drugs that are known to apply to $\mathcal{F}$ [3].

Consider the example of the HIV protease, a protein produced by the human immunodeficiency virus. The target identification stage involves the discovery of this HIV protease and the identification of this protein as a disease causing agent. The objective of drug design is to design a molecule that will bind to and inhibit the drug target . A great deal of time and money can be saved if the effect of molecules can be determined before these molecules are actually synthesised in a laboratory. Bioinformatics tools are used to predict the structures and hence the functions of the molecules under design and to determine if they will have any effect on the drug target.

The G-Protein Coupled Receptors (GPCRs) are the most important superfamily of proteins found in the human body. Many classification systems have been developed over the years based on machine learning to classify sequences as belonging to one of the GPCR families, and have shown great success in this task. These classification systems produce static classifiers which cannot accommodate any new sequences that may be discovered, and do not aid in solving any of these grand problems.

This paper introduces the use of a classification system based upon an evolutionary strategy, incremental learning and the Fuzzy ARTMAP to realise a protein classification system for the GPCR protein superfamily that allows all-vs-all comparison of these proteins. Being an incremental system, the classifier is dynamic and has the ability to incorporate new information into the classification model.

2. Importance of the GPCRs

The G-Protein Coupled Receptors (GPCRs) are a superfamily of proteins and forms the largest superfamily of proteins found in the human body. The GPCRDB is a database dedicated to the storage and annotation of G-Coupled proteins and at present consists of 16764 entries [4]. GPCRs play important roles in cellular signalling networks in processes such as neurotransmission, cellular metabolism, secretion, cellular differentiation and growth and inflammatory and immune responses.Because of these properties, the GPCRs are the targets of approximately 60% – 70% of drugs in development today [5] and results in more than US$23.5 billion in pharmaceutical sales revenue from drugs which target this superfamily.

The GPCR superfamily consists of five major families and several putative families, of which each family is further divided into level I and then into level II subfamilies. The extreme divergence among GPCR sequences is the primary reason for the difficulty of classifying these sequences [1].

In this research eight GPCR families are considered from the number of families available at the GPCRDB. The GPCR sequences are stored in the EMBL format, which consists of a number of labelled fields considering aspects of a sequence such as identifiers in a number of databases, the date of discovery and relevant publications dealing with the protein sequence. The database itself is updated every three to four months.

3. Review of Important Tools

A number of tools are used in the system to be presented, these include the fuzzy ARTMAP (FAM) classifier and the Genetic Algorithm, both of which are reviewed here.

3.1. Overview of Fuzzy ARTMAP

The fuzzy ARTMAP was introduced by Carpenter *et al* [6] and is based on Adaptive Resonance Theory (ART) and a fuzzy learning algorithm. A diagram showing the structure of a fuzzy ARTMAP system is shown in figure 1. Fuzzy ARTMAP has

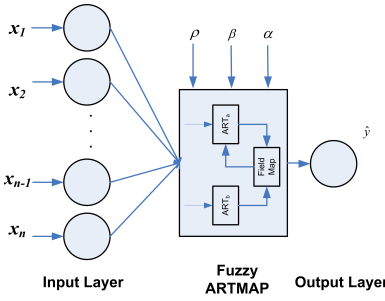


Figure 1: Representation of the Fuzzy ARTMAP Architecture

two free variables. The vigilance parameter ρ represents the tradeoff between classification accuracy and incremental learning ability. The learning rate β is a factor by which the hyperboxes are adjusted with each training pattern during the training phase. In this work, $\beta = 1$, which is known as fast learning. For further details on the Fuzzy ARTMAP and its training, the reader is referred to [6].

3.2. Overview of the Genetic Algorithm

Genetic algorithms (GA) find approximate solutions to problems by applying the principles of evolutionary biology, such as crossover, mutation, reproduction and natural selection [7]. The GA search process consists of the following steps: 1) Generating a pool of candidate solutions (chromosomes) and encoding all values in a binary or floating point representation. 2) Evaluation of the fitness for each chromosome in the gene pool. The fitness is determined via a fitness function defined for the problem being solved, and chromosomes with the lowest fitness are discarded and make way for a new set of chromosomes. Replacement sets of chromosomes are created by the genetic operations of crossover and mutation on the most fit individuals. These genetic operations add an element of randomness to the search process allowing a wider range of the solution space to be explored. 3) Steps 1 and 2 are repeated until a specified fitness level is attained or the maximum number of generations is exceeded [7].

4. Prior Work

The problem of incremental learning has not been considered before as it is presented here. Vijaya et al [8] consider the incremental clustering of protein sequences, but that is a different problem from that considered here. The fuzzy ARTMAP has been chosen as the incremental classifier and as mentioned, has been shown to be an effective incremental classifier [6]. The Support Vector Machine (SVM) is widely used in protein classification and it would appear that the use of an incremental SVM would be more suitable. While some algorithms for incremental SVM [9] exist, the problem with many of these systems is that they cater to the binary-classification problem only and are not applicable to multi–class classification problems, which is the case for the classification of proteins into families. Other incremental classification systems also exist, such as incremental common-sense models and incremental fuzzy decision trees. Of these incremental classification systems, the fuzzy ARTMAP is the most established and well known and is thus used.

5. System Overview

A schematic representation of the system is shown in figure 2. Input sequences are extracted from a protein database and then

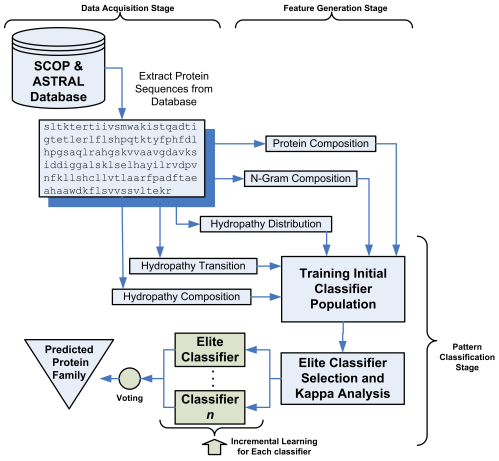


Figure 2: Overview of System Architecture

converted into a numerical feature vector. We then create a population of classifiers to introduce classification diversity, with the selection of suitably diverse classifiers from this population using the Genetic Algorithm coupled with kappa analysis. An ensemble of classifiers is used as a means of introducing modularity in the learning system. This system is implemented using the fuzzy ARTMAP (FAM) and a series of experiments are conducted to evaluate the performance of this system. Pseudocode for the creation and operation of the system is shown in algorithm listing 5. The ability of the FAM as an alternative classifier to many of the other more popular classifiers is demonstrated by comparing the classification ability of these systems using the GPCR data set. The incremental learning system described by algorithm listing 5 is then tested using the GPCR data and shown to be able to learn new data as well as maintain existing data.

6. Protein Vectorisation

The data obtained from the GPCRDB is in the form of amino acid sequences. In order for these sequences to be used in classification systems, they must be converted into a numerical form. Before this conversion though, preprocessing in the form of outlier removal must be completed. Outlier removal consists of removing sequences which have characters which are not part of the standard 20-letter amino acid alphabet — the letters are B and Z and have ambiguous meanings. Once this process is complete, these protein sequences must be transformed into numerical features. Two types of features have been identified in the literature, these being global and local features. Huang *et al* [10] provide a good description of the difference between global and local features and this distinction is used in this work.

Algorithm 5.1: FUZZY ENSEMBLE(D)

Training Phase

comment: Create population *j* of FAM classifiers each trained with a different permutation of the input data $\mathbf{X}_1$

Each classifier is a hypothesis $h_t : \mathbf{X}_1 \mapsto \mathbf{Y}_1$

$\epsilon = \frac{1}{N} \sum n|_{h_t(x_i) \neq y_i}$

comment: Sort classifiers based on incr. error on validation set.

SORT(ϵ)

comment: Select lowest error classifier as elite classifier h_{elite}

comment: Calculate the agreement κ, of the 15 best classifiers (based on error) with respect to the elite classifier

$\kappa = \frac{N \sum_{i=1}^{N} x_{ii} - \sum_{i=1}^{N} x_{i+} \cdot x_{+i}}{N^2 - \sum_{i=1}^{N} x_{i+} \cdot x_{+i}}$

comment: Genetic Algorithm selection of p classifiers based on a trade–off between error ϵ and agreement κ

$GA_{fitness}(\kappa, \epsilon) = \lambda \sum_{i=1}^{p} \kappa_i + \sum_{i=1}^{p} \epsilon_i$

Create ensemble classifier using the elite classifier h_{elite} and the p selected classifiers $h_t, t = 1, \ldots, p$

comment: Fusion of individual predictions using majority voting.

Operation Phase

If predicting sequence family, convert to feature representation and classify using the Fuzzy ARTMAP based system created during this previous training phase

comment: If incrementing system knowledge, increment each classifiers in the Fuzzy ARTMAP base system independently, using the training data for new sequences

$h_t^{incr} = \mathcal{T}\left(h_t, \mathbf{X}_k \mapsto \mathbf{Y}_k\right),$

where the transformation $\mathcal{T}$ is the incremental training process and k is the dataset to be added to the system

6.1. Global Feature Generation

Global features represent the nature of the entire protein sequence. These features must capture the global similarity between related sequences allowing for comparison. Consider the amino-acid composition of the sequence. The composition is simply the presence frequency of each of the 20-possible amino acids in the given sequence. Thus the composition is calculated by [11]:

$$\nu_i = \frac{s_i}{\sum_{j=1}^{20} s_j}, \text{ for } i = 1, 2, \ldots, 20. \quad (1)$$

where ν_i is the value for the i*th* feature and s_i is the number of times the i*th* amino acid appears in the sequence. This results in 20 features: a frequency of appearance for each of the possible amino acids. If a particular amino acid does not appear at all in the sequence, the corresponding feature value is zero.

A second set of features based on the hydropathy of amino acids in a given protein sequence is also calculated. Amino acids are either hydrophobic, hydrophilic (polar) or neutral. We use the Chothia and Finkelstein [12] hydropathy classification. We calculate three descriptors, the hydropathy composition ($\mathcal{C}$), the hydropathy distribution ($\mathcal{D}$) and the Hydropathy transmission ($\mathcal{T}$) for the sequences as described by Dubchak [12].

The composition $\mathcal{C}$ is calculated similarly to the amino acid composition described previously. In this case we calculate the presence frequency of hydrophobic, hydrophilic and neutral amino acids in the sequence. This results in three features being generated. The transmission is defined by three values. The first is the number of times a polar molecule is followed by a neutral molecule or vice versa. Similarly the other two are the number of times a neutral molecule is followed by a hydrophobic molecule or vice versa and the number of times the polar molecule is followed by a hydrophobic molecule or vice versa.

The distribution looks at intervals of 25%, 50%, 75% and 100% along the sequence length. For each interval the presence frequency of hydrophobic, hydrophilic and neutral molecules for each percentage interval is calculated. This results in 12 features, 4 features for each of the three hydropathy groups. A more detailed description of these features can be found in Dubchak [12]. In total 38 features (20+3+3+12) are generated based on global sequence descriptors.

6.2. Local Feature Generation

The local features capture local interactions between amino acids and groups of amino acids in a protein sequence. The n-gram method is well established as a good descriptor of local similarities in a sequence and has been used by many authors such as Cheng *et al* [1], Tomovic *et al* [13] and Zhao *et al* [14]. Essentially the n-gram method considers the presence frequency of consecutive n-letter combinations in the protein sequence, for integer n. For example, consider the short sequence `SLTKTERTIIVSM`, the 2-grams of this sequence are: `SL`, `LT`, `TK`, `KT`, etc. Given a sequence, features are generated by calculating the presence frequency of all possible n-grams for the amino acid alphabet.

A total of 438 features have been generated and as a final post-processing step undergo min-max normalisation. The normalisation is a requirement for using the FAM, since the FAMs complement coding scheme assumes normalised data.

7. Incremental Algorithm and Diversity

The creation of the committee–based sytem is based on a novel approach, implementing an evolutionary strategy which was summarised in algorithm listing 5. We first train an initial population of j classifiers, each classifier having been trained with a different permutation of the input training data. This permutation is needed in order to add diversity to the classifiers being created. As mentioned, the fact that the fuzzy ARTMAP learns in an instance–based fashion, makes the order in which the training patterns are received an important factor [15]. In the experiments performed, the initial population consists of 30 classifiers.

The classification error ϵ of each of these classifiers is then evaluated against a validation data set. The classifiers are then ranked in terms of increasing error. The lowest error classifier from this population is the *elite classifier* and is the classifier that automatically becomes a member of the ensemble system. The inclusion of this elite classifier ensures that at least one high accuracy classifier is selected for the committee.

The next step is to select the remaining n classifiers. In this application we select a further 4 classifiers. The selection of the other members of the committee is important and requires a number of factors to be considered:

- We do not wish to select classifiers that perform exactly as the elite classifier, since this gives no diversity to the predictions that are generated, and thus there is no room for improvement.

- We do not wish to select low accuracy classifiers that will confuse the prediction obtained and thus result in predictions that are more erroneous than a single classifier.

It would appear that these two conditions oppose each other, since high accuracy classifiers would tend to agree on the same predictions, against what we require for point 1. A trade-off between the classifier accuracy and the level of agreement between classifiers is then ideally what is required. This introduces the need for a formal definition of agreement between classifiers.

We use the definition of agreement considered by Petrakos et al [16], and the mathematical description that follows is generally known as kappa analysis. We define the agreement between any two classifiers κ based on the error matrix of the two classifiers [17]. The error matrix shows the number, and for which classes the two classifiers agree on a prediction. Table 1 shows the format for an error matrix between two classifiers.

Table 1: Data Format for Error Matrix Between Classifiers

	Classifier 2				
Classifier 1	C_1	C_2	$\dots$	C_Q	Totals
C_1	x_{11}	x_{12}	$\dots$	x_{1Q}	x_{1+}
C_2	x_{21}	x_{22}	$\dots$	x_{2Q}	x_{2+}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_Q	x_{Q1}	x_{Q2}	$\dots$	x_{QQ}	x_{Q+}
Totals	x_{+1}	x_{+2}	$\dots$	x_{+Q}	

In the ablove table, Q is the number of classes in the data. x_{11} in the table is the number of test patterns that both classifier 1 and 2 agreed belonged to class C_1. x_{21} is the number of test patterns that classifier 1 predicted belonging to class C_2, but that classifier 2 predicted belonged to class C_1. Similarly, the entire error matrix can be generated using the prediction made by any two classifiers. We determine the error matrices for 15 of the best classifiers in terms of predictions with respect to the elite classifier. The agreement is calculated using the following set of equations, where N is the number of training patterns used in generating the error matrix [17].

$$\theta_1 = \sum_{i=1}^{N} x_{ii} \tag{2}$$

$$\theta_2 = \sum_{i=1}^{N} x_{i+} \cdot x_{+i} \tag{3}$$

$$\kappa = \frac{N\theta_1 - \theta_2}{N^2 - \theta_2} \tag{4}$$

The selection of classifiers from this population, which must essentially minimise both the error of the individual classifiers and the agreement of the classifiers with the elite classifier, is an optimisation problem. We have chosen to implement a Genetic Algorithm as the optimisation tool for this system. The Genetic Algorithm (GA) is a stochastic optimisation tool that borrows concepts from evolutionary biology such as selection, crossover and mutation [18]. The GA minimises a cost function that is defined for a particular problem by stochastically exploring the space of available solutions. The GA implemented for the selection of classifiers is designed to select 4 classifiers and minimises both the agreement and the error of the selected combination of classifiers.

The GA will select 4 classifiers, resulting in two vectors

$$\epsilon_{\mathbf{GA}} = \{\epsilon_1; \epsilon_2; \epsilon_3; \epsilon_4\}$$
$$\kappa_{\mathbf{GA}} = \{\kappa_1; \kappa_2; \kappa_3; \kappa_4\}$$

We use a linear combination of these two matrices to define the cost value of a particular selection of classifiers. It is this cost that the GA will attempt to minimise. The cost function is defined by equation 5. λ is introduced as a scalar constant to allow the relative importance of the agreement in the system to be adjusted. In this study $\lambda = 1$, which gives equal importance to both the error and the agreement.

$$f(\epsilon, \kappa) = \lambda \sum_{i=1}^{4} \kappa_i + \sum_{i=1}^{4} \epsilon_i \tag{5}$$

The GA selects the 4 best classifiers that minimises the cost function of equation 5. The Genetic Algorithm was designed to produce 50 generations of solutions with each generation being a population 30 possible solutions. The crossover rate was set to a high value of 0.8 and a mutation rate of 0.4, and were empirically determined to be the best values for the experiment. The crossover functions are modified from the standard crossover functions in this case, to ensure that unique classifiers are selected during each generation, that is, preventing the same classifier from being selected twice in a particular generation.

These selected classifiers are then used in parallel, with each of the five classifiers in the system producing an independent set of predictions. These predictions must then be fused together to form the final decision. A number of decision fusion techniques exist. Some of these include the majority and weighted majority voting, trained combiner fusion, median, min and max combiner rules [19]. We adopt the majority voting decision fusion scheme, which simply considers each of the predictions produced by the five classifiers as a vote, with the final prediction for any given pattern given by the prediction that receives the largest number of votes.

7.1. Incremental Learning of Protein Data

The ensemble system is not a useful system if it is not able to accommodate newly discovered sequences that are produced daily. The ability of a classifier to allow this type of knowledge update was also defined as incremental learning. The fuzzy ARTMAP through its instance–based learning is able to incrementally learn new data. This incremental learning can consider two types of data:

1. It is possible to add new sequence information for families which the classifier has already been trained with.
2. Data of completely new classes can be added to the system, increasing the knowledge that the system has of the general protein domain.

The base system will in general be trained with data of a number of classes. Once new data becomes available, incremental learning of the system is based on incrementally training each of the 5 FAM classifiers in the system with the new data. The system can now be tested with data from all classes it has been trained with, including classes which have been incrementally added to the system.

8. System Testing and Experimental Results

8.1. Testing Using GPCR Data

The GPCR data is also divided into 6 separate databases $\mathcal{D}_1, \dots, \mathcal{D}_6$, with a validation set for database $\mathcal{D}_1$. In this

case, the datasets have data of all 8 classes which are available. This specific partioning is used to demonstrate data incremental learning, where new data of classes which the system has already been trained with is added to the system. This case is more appropriate for use with GPCR data where the families are established. The separation of data into these databases is shown in table 2.

Table 2: Separation of data into individual databases for testing using GPCR data. $\mathcal{D}_v$ and $\mathcal{D}_t$ are the validation and testing datasets respectively.

Family	$\mathcal{D}_1$	$\mathcal{D}_v$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$	$\mathcal{D}_t$
Type 1	32	10	43	43	43	43	43	43
Type 2	23	8	30	30	30	30	30	30
Type 3	16	6	22	22	22	22	22	22
Type 4	6	2	9	9	8	8	8	8
Fz/Smo	12	4	16	15	16	16	16	16
MLO	3	1	4	5	5	5	5	4
Class H	32	11	43	43	43	43	43	43
Pheromone 2	20	6	26	26	26	26	27	27

8.2. Comparative Performance

We compare the Fuzzy ARTMAP with other more common machine learning tools such as the Support Vector (SVM) Machines and Multi-layer perceptron (MLP). These have been chosen since they have found widespread use in the literature [1, 3, 14]. Table 3 shows the performance of the classifiers that were considered in the experiment. The parameters that are used for each of the classifiers is included in the table. The classifiers are trained with all the training data combined into a single training set and tested on the test set $\mathcal{D}_t$, using the features that were described in section 5. The table shows that the

Table 3: Comparative performance of FAM versus other classifiers on the GPCR dataset.

Classifier	Error (%)
Generalised Linear Model	25.91
Multi–layer Perceptron, $n_{hid} = 15$, $cyc = 200$	15.03
Fuzzy ARTMAP $\rho = 0.75$	11.90
SVM - RBF $\gamma = 2.3$	17.10
SVM-Polynomial 2.23 degree	10.36

FAM has comparable accuracy when compared to many other classification systems.

8.3. Base Classifier Training and Incremental Performance

The base classification system was trained using database $\mathcal{D}_1$. Table 4 shows the error of the first 15 classifiers of the population and agreement with the elite classifier. The error is the error of the system on the validation data set. The GA for this data set selected classifiers `2, 3, 4, 12` to form the final ensemble system. Again, the system consisting of the elite classifier and the four classifiers selected by the GA are incrementally trained using databases $\mathcal{D}_2, \ldots, \mathcal{D}_6$, with the ensemble being tested after each increment with the testing database $\mathcal{D}_t$. The performance of the system is shown in table 5. This data shows

Table 4: Error and Agreement values for 15 classifiers of the population

Classifier	Val Error ϵ (%)	Agreement κ
1	27.0833	*Elite*
2	29.1667	0.8940
3	29.1667	0.9730
4	29.1667	0.8438
5	31.2500	0.8929
6	31.2500	0.8929
7	31.2500	0.8929
8	31.2500	0.8455
9	31.2500	0.8683
10	31.2500	0.8929
11	31.2500	0.8929
12	31.2500	0.8929
13	31.2500	0.8929
14	31.2500	0.8430
15	33.3333	0.8430

Table 5: Training and generalisation performance of system on GPCR data

Set	Train 1	Train 2	Train 3	Train 4	Train 5	Train 6
$\mathcal{D}_1$	0	0	0	0	0	0
$\mathcal{D}_2$	—	0	0	0	0	0
$\mathcal{D}_3$	—	—	0	0	0	0
$\mathcal{D}_4$	—	—	—	0	0	0
$\mathcal{D}_5$	—	—	—	—	0	0
$\mathcal{D}_6$	—	—	—	—	—	0
$\mathcal{D}_v$	25.00	22.92	22.92	27.08	25.00	27.08
$\mathcal{D}_t$	22.79	18.65	19.17	19.69	18.65	16.58

that the system is extremely capable of remembering data that has been trained upon, as shown by the many 0% which appear in the table for the training databases. The many zeros are not an indication of overtraining. The FAM is trained so that it learns all its training data with a 0% error. What the results show is that after it has learnt its initial training data, the memory is not degraded by the addition of additional data. The system also shows that the perfromance does increase as more data of each of the classes is added to the system.

9. Analysis of Results

The results presented indicate that the Fuzzy ARTMAP is a suitable machine learning tool for the classification of protein sequences into structural families, which is comparable to many of the more established tools. An analysis of the sequences also shows that the system is able to classify proteins of varying lengths from 32 to 350 amino acids in length, and thus the length of the protein sequences used are not important. The accuracy of the classification could be improved if some form of dimensionality reduction or feature selection is applied. These techniques have been applied by many authors using numerous techniques. Principal Component Analysis has been used as a technique of dimensionality reduction and Cheng et al [1] uses the chi-squared test as a means of feature selection. Feature selection can also be applied using various sub-optimal feature selection techniques such as the floating forward selection search

or the Genetic Algorithm can be used as demonstrated by Mohamed *et al* [20].

The agreement κ was used to measure diversity of the system. This might not be the best measure of the relationship between predictions of classifiers to the elite classifier. The use of the correlation coefficient could be explored or the use of a disagreement [17] should also be explored, to determine if this measure gives some degree of refinement in the selection of the classifiers. The genetic algorithm is also important in the committee. Due to the stochastic nature of the GA, it is possible that different GA optimisations produce a different selection of classifier members. This though is not as likely in the case of the data presented here, since many of the classifiers had the same agreement or error, resulting in the GA converging to the same selection choice. That said, the optimisation of the GA is efficient and runs very fast due to the fact that it uses pre-calculated results such as the error matrix and agreement values.

10. Conclusion

The algorithm presented is applicable in general to all classification problems. Where the case exists that any new information that may be obtained will not significantly improve the classification ability of the system, then the batch training approach may be more suitable. Where this is not the case such as families whose sequences have low sequence similarity, then the incremental approach may be better and will be more desirable, especially if prior training data is no longer available.

The techniques presented here are also not limited to the problem of structural family classification, and can be easily extended to secondary and tertiary structure prediction, functional annotations and the prediction of protein–protein interaction sites. Apart from systems in proteomics, genomic applications also exist, such as the classification of promoter, and splice sites. Each classification task benefits from the improvements which can be gained from using an ensemble system and incremental learning. These results show great promise for the future of computational biology, where newly discovered data needs to be accurately incorporated into existing models, allowing for highly agile discovery processes.

11. References

[1] B. Y. M. Cheng, J. G. Carbonell, and J. Klein-Seetharaman, "Protein classification based on text document classification techniques," *Proteins: Structure, Function, and Bioinformatics*, vol. 58, pp. 955–970, 2005.

[2] D. E. Krane and M. L. Raymer, *Fundamental Concepts of Bioinformatics*. San Francisco: Pearson Education, 2003.

[3] J. T. L. Wang, Q. Ma, S. Shasha, and C. H. Wu, "New techniques for extracting features from protein sequences," *IBM Systems Journal*, vol. 40, no. 2, pp. 426–441, 2001.

[4] F. Horn, J. Weare, M. W. Beukers, S. Hørsch, A. Bairoch, W. Chen, Ø. Edvardsen, F. Campagne, and G. Vriend, "GPCRDB: an information system for G-protein coupled receptors," *Nucleic Acids Research*, vol. 26, no. 1, pp. 277–281, 1998.

[5] K. Lundstrom, "Structural genomics of GPCRs," *Trends in Biotechnology*, vol. 23, pp. 103–108, February 2005.

[6] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps.," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698–713, 1992.

[7] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer, third, revised and extended ed., 1999.

[8] P. A. Vijaya, M. N. Murty, and D. K. Subramanian, "An efficient incremental protein sequence clustering algorithm," in *Proceedings of the IEEE Conference on Convergent Technologies for Asia-Pacific Region*, pp. 409–413, October, 2003.

[9] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," *Advances in Neural Information Processing Systems (NIPS 2000)*, vol. 13, pp. 409–415, 2001.

[10] C. Huang, C. T. Lin, and N. R. Pal, "Hierarchical learning architecture with automatic feature selection for multiclass protein fold classification," *IEEE Transactions on Nanobioscience*, vol. 4, pp. 221–232, December 2003.

[11] X. Zhao, Y. Cheung, and D. Huang, "A novel approach to extracting features from motif content and protein composition for protein sequence classification," *Neural Networks*, vol. 18, pp. 1019–1028, October 2005.

[12] I. Dubchak, I. Muchnik, S. R. Holbrook, and S. Kim, "Prediction of protein folding class using global description of amino acid sequence," in *Biophysics*, vol. 92, (USA), pp. 8700–8704, National Academy of Science, September, 1995.

[13] A. Tomovic, P. Janicic, and V. Keselj, "N-gram-based classification and unsupervised hierachical clustering of genome sequences," *Computer Methods and Programs in Biomedicine*, vol. 81, pp. 137–153, 2006.

[14] D. Huang, X. Zhao, and G. Huang, "Classifying protein sequences using hydropathy blocks," *Pattern Recognition*, vol. 30, pp. 2293–2300, December 2006.

[15] A. Koufakou, M. Georgiopoulos, A. Anagnostopoulos, and T. Kasparis, "Cross–validation in fuzzy ARTMAP for large databases," *Neural Networks*, vol. 14, pp. 1297–1291, 2001.

[16] M. Petrakos, J. A. Benediktsson, and I. Kannellopoulos, "The effect of classifier agreement on the accuracy of the combined classifier in decision level fusion," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, pp. 2539 – 2546, November 2001.

[17] Y. M. Bishop, S. E. Fienberg, and P. W. Holland, *Discrete Multivariate Analysis*. Massachusetts: MIT Press, 1977.

[18] R. L. Haupt and S. E. Haupt, eds., *Practical Genetic Algorithms*. New York: Wiley Interscience, second ed., 2004.

[19] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[20] S. Mohamed, D. Rubin, and T. Marwala, "Multi-class protein classification using Fuzzy ARTMAP," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, (Taipei, Taiwan), pp. 1676–1681, 8 - 11 October 2006.