CLASSIFICATION OF COHESIN FAMILY USING CLASS-SPECIFIC MOTIFS

(COHESIN AİLESİNİN SINIFA ÖZEL MOTİFLER İLE SINIFLANDIRILMASI)

by

Mithat Ercüment ESER, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Date of Submission	: Sep 19, 2013
Date of Defense Examination	: Oct 1, 2013

Acknowledgements

I would like to thank to Asst. Prof. Dr. Burak Arslan and Prof. Dr. Uğur Sezerman for giving me the opportunity to work such an interesting project and interesting approaches that help me a lot. I am also grateful for their understanding of my time limitations and encourage they have provided. I also would like to thank to my colleagues and Bülent Altay for understanding my dedication to the thesis.

Finally, I want to thank my dear parents and my sister for their love and support.

Mithat Ercüment ESER

İstanbul, Oct 31th, 2013

Table of Contents

Acknowledgements ii
Table of Contents iii
List of Figures v
List of Tables vi
Abstract vii
Résumé viii
Özet ix
1. Introduction 1
  1.1 Thesis Organization 3
2. Biological Background 5
  2.1 Amino acid 5
  2.2 Protein 9
  2.3 Protein Sequence and Motifs 10
  2.4 Reduced Amino acid Alphabets 12
  2.5 N-Grams 13
  2.6 Cellulosome, Cohesin and Dockerin 14
3. Computational Background 19
  3.1 Machine Learning 19
  3.2 Decision Trees 21
    3.2.1 Definition 21
    3.2.2 The Basics 22
    3.2.3 Divide and Conquer 22
    3.2.4 Building Decision Trees 23
    3.2.5 Information Gain 25
    3.2.6 Optimization of Decision Trees 28
      3.2.6.1 Subtree replacement 28

3.2.6.2 Subtree raising 29
3.2.7 DT scheme used in our experiments: J48 30
3.3 NAÏVE BAYES 31
3.4 Feature Selection 33
3.4.1 Filter Methods 36
3.4.2 Wrapper Methods 38
4. Experiments 44
4.1 Data Set 44
4.2 Amino Acid Grouping Schemes 45
4.3 Class Specific Motif Scoring Function 46
Definition 1. Motif Occurrence Rate in Family 47
Definition 2. Discriminatory Ratio (DR) 47
Definition 3. Inverse Family Frequency (IFF) 48
Definition 4. Motif Specificity Score (MSS) 48
4.4 Motif Discovery Results 49
4.5 Classification 52
4.6 Classification Results 54
5. Conclusion 56
References 57
Biographical Sketch 61

List of Figures

Figure 2.1 Amino acid structure 5
Figure 2.2 Lipid bilayer 8
Figure 2.3 Simple Cellulosome System taken from Weizmann.com 15
Figure 2.4 Cohesin and Dockerin Modules 17
Figure 2.5 Phylogenetic distribution of representative cohesins (a) and dockerins (b) in the three domains of life taken from (Peer et al, 2009) 18
Figure 3.1 Simple identification of protein sequences based on residue percentage 23
Figure 3.2 Weather data taken from Weka 24
Figure 3.3 A possible DT for the weather data (visualization in WEKA) 25
Figure 3.4 First split on weather data (taken from Hall et al. (2009)) 27
Figure 3.6 Subtree replacement. Taken from the book 'Data Mining: Practical Machine Learning Tools and Techniques' (modified) 29
Figure 3.7 Subtree raising. Taken from the book 'Data Mining: Practical Machine Learning Tools and Techniques' 29
Figure 3.8 The C4.5 algorithm applied to the weather data (visualization taken from WEKA) 31
Figure 3.9 Different components in the standart feature selection 35
Figure 3.10 Generalized Filter Algorithm taken from (Liu & Yu, 2005) 37
Figure 3.11 Generalized wrapper algorithm (Liu & Yu, 2005) 38
Figure 4.1 Specific motif number when MSS threshold is 1.5 51
Figure 4.2 Specific motif number when MSS threshold is 1.8 51
Figure 4.3 Specific motif number when MSS threshold is 2.0 52

List of Tables

Table 2.1 Standard amino acids and their properties 7

Table 2.2 Known anaerobic bacterias that produce cellulosome 15

Table 4.1 Distribution of Cohesin sub-families in our Dataset 44

Table 4.2 Reduced amino acids used in this work 45

Table 4.3 Some N-Gram Motifs found using Sezerman grouping scheme 49

Table 4.4 Classification Results sorted by Balanced Test Accuracy 54

Abstract

Bioinformatics is an area of science that helps developing and improving methods to store, retrieve, organize and analyze biological data. Thus, bioinformatics has gained important role for molecular biology. One of the methods to analyze this big data is to use classification of protein sequences to predict unseen proteins types. In addition to this, finding motifs, which are a part of protein sequence that contains biological function of the sequence, is important to understand protein structure and protein-protein relationships.

In this work, class-specific motifs with high specificity are found and supervised classification models are trained to classify new sequences to find types of cohesin protein using various machine learning algorithms like J48 Decision Tree, Support Vector Machines and Naïve Bayes and with different combinations of Reduced Amino acid Alphabets/Groupings. Results were compared by classification accuracies. Using 5-gram sized Sdm13 alphabet with 10 features and Naïve Bayes algorithm, highest accuracy of 99.09 % is achieved.

Résumé

La bioinformatique est un domaine de la science qui permet à développer et à améliorer les méthodes pour stocker, extraire, organiser et analyser des données biologiques. Ainsi, la bioinformatique a gagné un rôle important pour la biologie moléculaire. Une des méthodes pour analyser ces données massives consiste à utiliser la classification des séquences de protéine pour prédire les types de protéines inaperçus. En plus, trouver des motifs qui font partie de la séquence de protéine contenant la fonction biologique de la séquence, est important pour comprendre la structure des protéines et des relations protéine-protéine.

Dans ce travail, les motifs classe-spécifiques avec une spécificité élevée sont trouvées et les modèles de classification supervisée sont utilisés pour classifier les nouvelles séquences pour trouver les types de protéines cohésine, utilisant les algorithmes diverses d'apprentissage machine comme l'arbre de décision J48, les machines à vecteurs de support et Naïve Bayes et avec différentes combinaisons d'alphabets/groupements réduit d'acides. Les résultats sont comparés en utilisant la précision de la classification. La plus grande précision de 99,09% est atteinte en utilisant un alphabet Sdm13 de taille 5 grammes avec 10 caractéristiques et l'algorithme Naïve Bayes.

Özet

Biyoenformatik, biyolojik bilginin saklanması, elde edilmesi, organize edilmesi ve analiz edilmesini sağlayan ve iyileştiren bilim dalıdır. Bu sebeple biyoenformatik, moleküler biyoloji için önemli bir hale gelmiştir. Protein dizilimlerinin analizinde kullanılabilecek bir yöntem bunların sınıflandırılması ve yeni bulunan proteinlerin sınıfının belirlenmesini sağlamaktır. Bununla beraber proteinlerin görevlerini temsil eden küçük parçaları olan motiflerin bulunması, protein yapısını ve protein-protein ikişkilerini göstermesi açısından önemlidir.

Bu çalışmada, Cohesin protein ailesinin sınıfa özel yüksek özgüllük içeren motifleri çeşitli indirgenmiş aminoasit alfabeleri/gruplamaları ve farklı n-gram uzunlukları ile bulunup J48, Support Vector Machine ve Naïve Bayes ile sınıflandırılmıştır. Sonuçta 5-gram uzunluklu Sdm13 alfabesi ile seçilen 10 özellik ile Naïve Bayes algoritması kullanılarak % 99.09 başarı ile sınıflandırma sağlanmıştır.

1. Introduction

Bioinformatics is an area of science that helps developing and improving methods to store, retrieve, organize and analyze biological data. Bioinformatics has gained important role for molecular biology. Image and signal processing techniques of bioinformatics on large amount of raw data allow extraction of useful results. To achieve this, tools for efficient access to various information and algorithms together with statistics to assess relationships among members of large datasets is used. Creating software tools by using these elements has the most important role for generating useful biological knowledge.

The primary goal of bioinformatics is to increase the understanding of biological processes. Developing and applying computationally intensive techniques such as pattern recognition, data mining, machine learning algorithms, and visualization are used achieve this goal. Major research efforts in the field include sequence alignment, gene finding, genome assembly, drug design, drug discovery, protein structure alignment, protein structure prediction, prediction of gene expression and protein–protein interactions, genome-wide association studies, and the modeling of evolution.

Classification of proteins has significant use cases for bioinformatics like drug target identification, drug design, protein family characterization and protein annotation (Albayrak & Sezerman, 2012). Moreover, classification of proteins might aid us understanding protein-protein interactions and to which proteins one can bind. As a result of these, various methods including similarity search and machine learning have been used to classify the proteins.

The similarity search in which a sequence is searched against a database, is a technique commonly used to assign function to a protein thus predicting it's family. However this type of classification fails in the absence of significant similarity between query and target protein sequences. (Reinhardt & Hubbard, 1998)

According to previous works, another approach, machine learning algorithms that firstly require features extracted from raw sequence data were used for classification problems including protein interaction prediction, cluster analysis of gene expression data, annotation of protein sequences by integration of different sources of information, automated function prediction (Albayrak & Sezerman, 2012). Although large protein families have been examined and classification was established on them, novel proteins that have specialized duties indirectly on human lack the interest of bioinformaticians. Cohesin and Dockerin protein families can be given as example to those ones.

In the beginning of 1980s, The cellulosome was publicized as discrete multienzyme complex, which is used for binding and degradation of the cellulose most common and abundant organic polymer in nature . The cellulosome is an exciting and unique example of a molecular level interaction like Lego-like construction of biologically active.

Nature uses specific protein-protein interactions to control the most cellular and physiological processes. One of those protein-protein interactions is between cohesin-dockerin with a high-affinity (Kd 10-9-10-12 M), where dockerin module plugs into the cohesin module (Pagès et al., 1997). As both cohesin and dockerin protein domains found in scaffolding protein, this interaction is important for cellulosome construction. Both cohesin and dockerin have various types, and each type is specific to other, e.g one cohesin will not interact with all dockerins as well as being species specific. So classification of these protein domains will help us determine which ones will bind together

The outcome of cohesin-dockerin recognition can be used to design nanomaterials and nanodevices. Besides it's nanotechnological uses, this interaction can artificially be used for biofuels as renewable cellulose resource instead of petroleum. (Ed Bayer's Group, 2012)

A motif is a part of DNA or protein sequence that contains biological function of the sequence in which it is found. Identifying, characterizing and searching with sequence motifs helped computational methods to be created (Grant et al., 2011). Previously, Cobanoglu, et al. (2011) proposed a method using class-specific motifs and grouping schemes to classify GPCR class A proteins and achieved an accuracy of 98.1 %. Srinivasan et al. (2013) have used similarity scores with substitution matrix to extract n-gram motifs from sequences to discriminate protein families. However substitution matrix has additional time complexity that can be eliminated by using pre-substituted reduced amino acid alphabets.

In this work, class-specific motifs of cohesin family with high specificity are found and trained to classify new sequences to find types of cohesin using various machine learning algorithms like Decision Tree, Support Vector Machines and Naïve Bayes with different combinations of Reduced Amino acid. Results were compared according to balanced accuracies.

1.1 Thesis Organization

Chapter 2, "Biological Background" gives background information about basic biology introducing the reader to amino acid, protein and then topics for protein sequences including motifs and n-grams are introduced.

Chapter 3, “Computational Background” introduces Machine-learning, classification, decision trees and Naïve Bayes. Besides this, information about feature selection algorithms that are used in the work are given.

Chapter 4, “Experiments” includes the actual work in this thesis that includes how n-gram classification together with reduced amino acid alphabets is used for classification. Finally results of the work are given.

2. Biological Background

This section lets the reader to get understanding of basic biology and genetic domain that the thesis is worked on in the order of Amino acid, Protein sequence and motifs, Reduced amino acid alphabets and n-grams. Finally information about the protein families of Cohesin and Dockerin will be given.

2.1 Amino acid

Amino acids are compound of organic elements carbon, hydrogen, oxygen and nitrogen (Figure 2.1). Containing nitrogen is what it differs from other biological compounds like glucose and fatty acids.

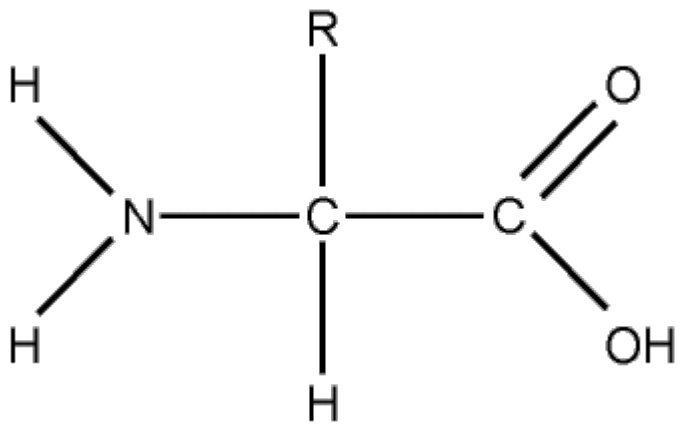


Figure 2.1 Amino acid structure

Most of the amino acids form polypeptides or peptides as polymer chains to form proteins. 21 of them are proteinogenic that means they are encoded by the genetic code.

Some of the amino acids cannot be produced by body, so they need to be taken from outside by foods. In humans lysine, leucine, isoleucine, methionine, phenylalanine, threonine, tryptophan, valine, histidine (children) and arginine (children).

A single carbon atom resides in between amino and carboxyl groups and it is called alpha carbon atom. Variable group (R) that the molecules of the 20 standard amino acids differ from one another is also attached to the carbon atom In the simplest of the acids, glycine, the R consists of a single hydrogen atom. Other amino acids have more complex R groups (side chains) that contain carbon as well as hydrogen and may include oxygen, nitrogen, or sulphur, as well.

Amino acids are linked together in proteins by a peptide bond, made by the reaction of the carboxyl group of one amino acid with the amino group of the next. After peptide bond is formed, a molecule of water (H2O) is released. This is a dehydration synthesis reaction, and usually occurs between amino acids.

The resulting C-N bond is called a peptide bond, and the resulting molecule is called an amide. Polypeptides and proteins are chains of amino acids held together by peptide bonds.

In order to interpret protein structures and protein-protein interactions, certain properties of amino acids should be known. Those properties include hydrophilicity or hydrophobicity, size, and functional groups. Hydrophobic means 'being afraid of water' and hydrophilic means 'fond of water'. As the names suggest, hydrophobic amino acids prefer to be in a non-aqueous environment like inside of lipid bilayer (Figure 2.2). Hydrophilic residues love water, and therefore like to sit at the outside of water-soluble proteins.

Table 2.1 Standard amino acids and their properties

Amino Acid	3-letter	1-letter	Side-chain polarity	Side-chain charge (pH 7.4)
Alanine	Ala	A	Nonpolar	Neutral
Arginine	Arg	R	Basic Polar	Positive
Asparaginc	Asn	N	Polar	Neutral
Aspartic	Asp	D	Acidic	Negative
Cysteine	Cys	C	Nonpolar	Neutral
Glutamic acid	Glu	E	Acidic polar	Negative
Glutamine	Gln	Q	Polar	neutral
Glycine	Gly	G	nonpolar	neutral
Histidine	His	H	Basic polar	P(10 %) N (90%)
Isoleucine	Ile	I	Nonpolar	neutral
Leucine	Leu	L	Nonpolar	Neutral
Lysine	Lys	K	Basic polar	positive
Methionine	Met	M	Nonpolar	neutral
Phenylalanine	Phe	F	nonpolar	neutral
Proline	Pro	P	Nonpolar	neutral

Serine	Ser	S	Polar	neutral
Threonine	Thr	T	Polar	Neutral
Trytophan	Trp	W	Nonpolar	Neutral
Tyrosine	Tyr	Y	Polar	neutral
Valine	Val	V	Nonpolar	neutral

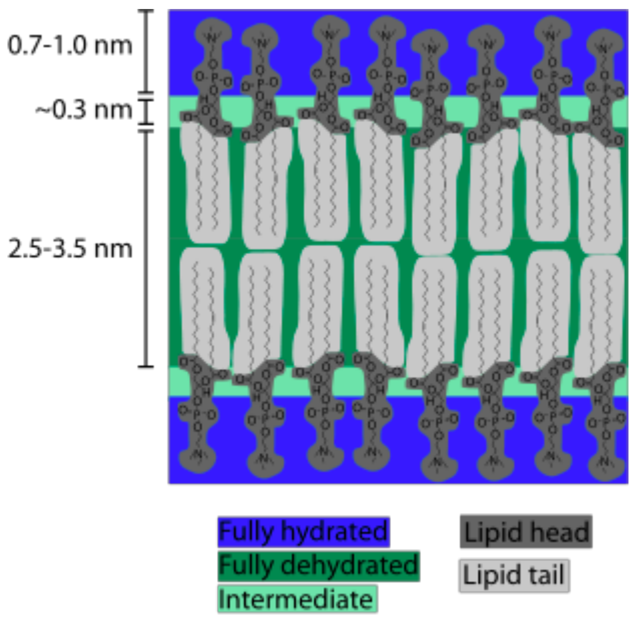


Figure 2.2 Lipid bilayer

Amino acids are shown in 1-letter or 3-letter alphabets codes. Table 2.1 shows the table of standart amino acids and their properties.

2.2 Protein

Proteins are biological molecules that are composed of one or more sequences of amino acids. Some of their functions in organisms are catalyzing metabolic reactions, replicating DNA, responding to stimuli, and transporting molecules from one location to another. Proteins are differentiated by their sequence of amino acids that are determined by the nucleotide sequence of their genes. That sequence is the key element that shows how the protein is folded into specific three-dimensional structure that determines its activity.

A polypeptide is a single linear polymer chain of amino acids bonded together by peptide bonds between the carboxyl and amino groups of adjacent amino acid residues. The sequence of amino acids in a protein is defined by the sequence of a gene, which is encoded in the genetic code.

Proteins are essential parts of organisms as other biological macromolecules such as polysaccharides and nucleic acids, and they participate in almost every process within cells. Most of the proteins are enzymes, which catalyze biochemical reactions and are vital to metabolism. Proteins also have structural or mechanical functions, such as actin and myosin in muscle and the proteins in the cytoskeleton, which form a system of scaffolding that maintains cell shape. Proteins are required by animals for their diets, since animals cannot synthesize all the amino acids they need and must obtain essential amino acids from food. Through the process of digestion, animals break down ingested protein into free amino acids that are then used in metabolism.

Proteins are combined of amino acids using information encoded in genes. Each protein has its own unique amino acid sequence that is specified by the nucleotide sequence of the gene encoding this protein. Codons that are three-nucleotide sets form the genetic code. A codon generates an amino acid, for example AUG (adenine-uracil-guanine) is the code for methionine. Total number of codons that is formed by combination of nucleoids are 64. That means there is redundancy in the genetic code, some of the amio acids are formed by more than one codon. Genes encoded in DNA are first transcribed into pre-messenger RNA (mRNA) by proteins such as RNA polymerase. Most organisms then process the pre-mRNA using various forms of Post-transcriptional modification to form the mature mRNA. After that mRNA is used as a template for protein synthesis by the ribosome.

The Structure of a protein, can provide important information about how the protein performs its function. The most used methods for structure determination are X-ray crystallography and NMR spectroscopy. They can give details about atomic resolution. Solved structures are usually inserted in the Protein Data Banks (PDB). However, protein structures have lower number than gene structures.

2.3 Protein Sequence and Motifs

Protein sequence is the order of amino acids connected by peptite bonds. The sequence starts from the free amino group of N-terminal end to carboxyl group of the C-terminal of the protein.

Protein sequences that has been found by protein sequencing, is stored in the sequence databases. Those databases includes from only one organism or from all organisms. Amino acids in the protein is showed by 1-letter or 3-letter abbreviations.

Both individual researchers and large companies help the database grow by sending their found sequences. This yields two problems, (i) low quality of sequences (ii) redundancy of them. So the databases might need to be controlled by humans and the sequences need to be verified by biological papers.

One of the high quality sequence databases is UniProt (Bailey, et al., 2009). It has both manually approved sequences by controllers and the ones that every researcher could add without approval, and includes many different database sources.

A motif is a part of DNA or protein sequence pattern that occurs repeatedly in a group of related protein or DNA sequences and contains biological function of the sequence in which it is found. Identifying, characterizing and searching with sequence motifs aided different computational methods to be created (Grant et al., 2011). Motif searching can be seen as basic string manipulation or more advanced like MEME , AlignAce, Amadeus, CisModule, FIRE, Gibbs Motif Sampler, PhyloGibbs, SeSiMCMC, ChIPMunk and Weeder. SCOPE, MotifVoter, and MProfiler are motif finder tools that use several algorithms together.

MEME uses statistics and position-dependent letter-probability matrices which describe the probability of each possible letter at each position in the pattern. However it has high time complexity.

An example to motif can be given as:

Asn, followed by anything but Pro, followed by either Ser or Thr, followed by anything but Pro

which can also be written as

```
N {P} [ST] {P}
```

2.4 Reduced Amino acid Alphabets

When N-grams are used for feature extraction from a sequence, number of features increase from 20 to 20^n. So feature selection via both general methods and bioinformatics specific RAAAs are used in this work.

Proteins have highly correlated sequence space and produces small number of folds, domains and structures [19]. To overcome this, the 20 standard amino acids can be grouped or classified using a wide variety of distinct criteria.

Reducing the number of amino acid alphabet, makes computation more effective. In addition to this it minimizes noise on sequence representation [20].

Basic principle for grouping is to use a number of physico-chemical properties such as hydrophobicity, charge, mass etc. Priority of these properties results in different grouping schemes.

Some of the examples of groupings used in this work:

- Random initialization: (Davies Random) SG DVIA RQN KP WHY LE MF
- Seeded initialization: (Davies Seeded 1) SGE DP RWN KQ HLVIMFY AT
- Seeded initialization with our modification: (Davies Seeded 2)

SGE DP RWN KQH LVIMFY AT

- Sezerman’s grouping: IVLM RKH DE QN ST YF
- Ab15: IV ML EQ FWY
- Lzmj11: IL FM KR NST HQPWY

• Lzbl10: FWY IMV HNST EKQR

2.5 N-Grams

An ***n*-gram** is a contiguous sequence of *n* items from a given sequence of text, speech or when applied in bioinformatics, amino acid sequence of proteins.

General usage of n-grams on prior works can be listed as follows;

- Design kernels to extract features for machine learning algorithms like Support Vector Machines
- Spell-checkers to inform user about misspelled word
- Improve compression algorithms
- Pattern recognition systems like Speech recognition and Optical Character Recognition applications
- Text classification using machine learning
- Genetic Sequence Analysis like BLAST
- Identify the language a text is in or the
- Identify species a small sequence of DNA was taken from

In protein sequences, n-grams are sequences of n amino acids in a sliding window over the length of the protein sequence. In a biological context, n-grams where n is equal to 1,2 and 3 correspond to amino acid, dipeptide and tri-peptide compositions, respectively. Given the sequence "ERCUMENT", there is one count of each of 3-grams ERC, RCU, CUM, MEN, ENT.

A short sequence that is an n-gram of amino-acids can be considered as word in bioinformatics, and with this knowledge statistical techniques can be applied on it to analyze many interesting properties of the sequences. Using n-grams, statistical analyses and information theoretic measures can be performed to understand the frequency distribution of n-grams in the protein sequences. Another uses of n-gram models can be considered as identifying sequence similarity, n-gram profiling, and in determining the conservation profile to identify protein homologs. (Albayrak & Sezerman, 2012)

The number of different n-gram features that can be discovered in a number of documents increases with n. Moreover, the number of occurrences of most n-grams will decrease with increasing n. Thus, although the number of features grows at least linearly with n, the number of features with a certain minimum frequency will grow much slower.

Another important aspect on n-grams is that, a subset of n-grams does not improve reliability, nor best frequency n-grams are better for classification. So pre-filtering of features according to just frequency of them may not yield better performance in terms of accuracy.

2.6 Cellulosome, Cohesin and Dockerin

The cellulosome is a discrete multienzyme complex, which is used for binding and degradation of the cellulose most common and abundant organic polymer in nature (Peer et al., 2009; Bayer et al., 1998). The cellulosome is an exciting and unique example of a molecular level interaction like Lego-like construction of biologically active. Figure 2.3 shows structure of a simple cellulosome system.

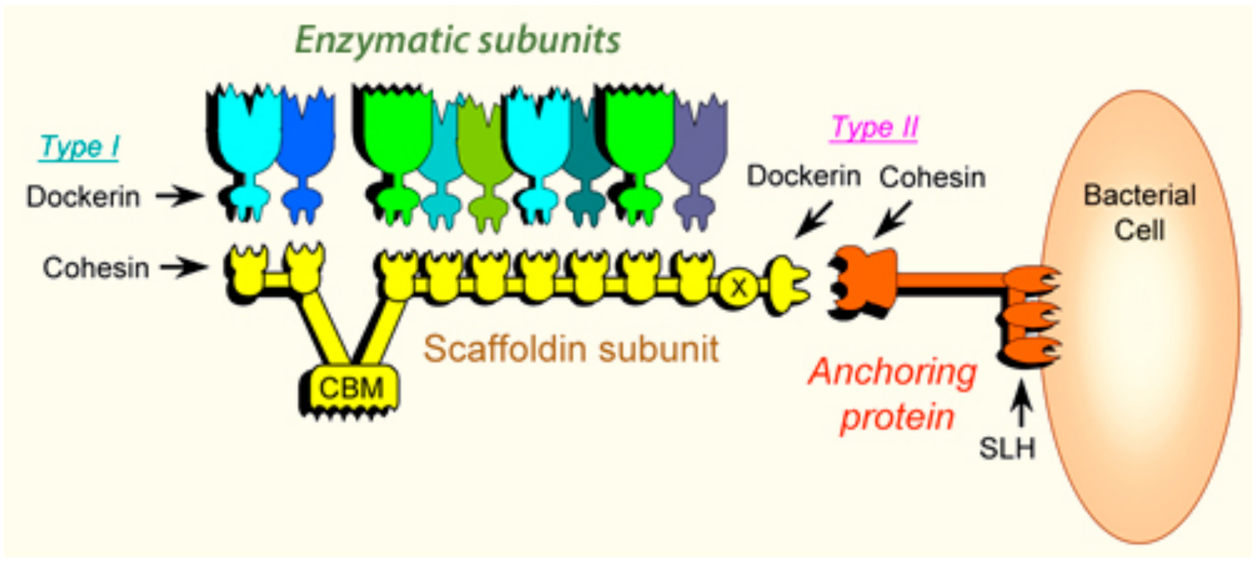


Figure 2.3 Simple Cellulosome System taken from Weizmann.com

Table 2.2 Known anaerobic bacterias that produce cellulosome

Acetivibrio cellulolyticus
Bacteroides cellulosolvens
Clostridium acetobutylicum
Clostridium cellulolyticum
Clostridium cellulovorans
Clostridium clariflavum
Clostridium josui
Clostridium papyrosolvens
Clostridium thermocellum
Ruminococcus albus (Only Dockerins)

Ruminococcus flavefaciens

Nature uses specific protein-protein interactions to control the most cellular and physiological processes. One of those protein-protein interactions is between cohesin-dockerin with a high-affinity (Kd 10-9-10-12 M), where dockerin module plugs into the cohesin module (Pagès, et al., 1997). As both cohesin and dockerin protein domains found in scaffolding protein, this interaction is important for cellulosome construction. Both cohesin and dockerin have various types, and each type is specific to other, e.g one cohesin will not interact with all dockerins as well as being species specific. So classification of these protein domains will help us determine which ones will bind together. All currently known bacterias that include cohesin and dockerin proteins are listed in **Table 2.2**. However the modules does not bind to only anaerobic bacterias, they have also be seen in other archaeal species, started by discovery of A.fulgidus (Peer et al., 2009). In **Figure 2.4** cohesin and dockerin modules are shown.

Previously Cohesin and Dockerins were clustered by Peer el all (Peer et al., 2009) and Figure 2.5 shows Phylogenetic distribution of representative cohesins and dockerins in all found species in that work, and the tree was obtained using the CLUSTALW program, and some of the modules was not classified.

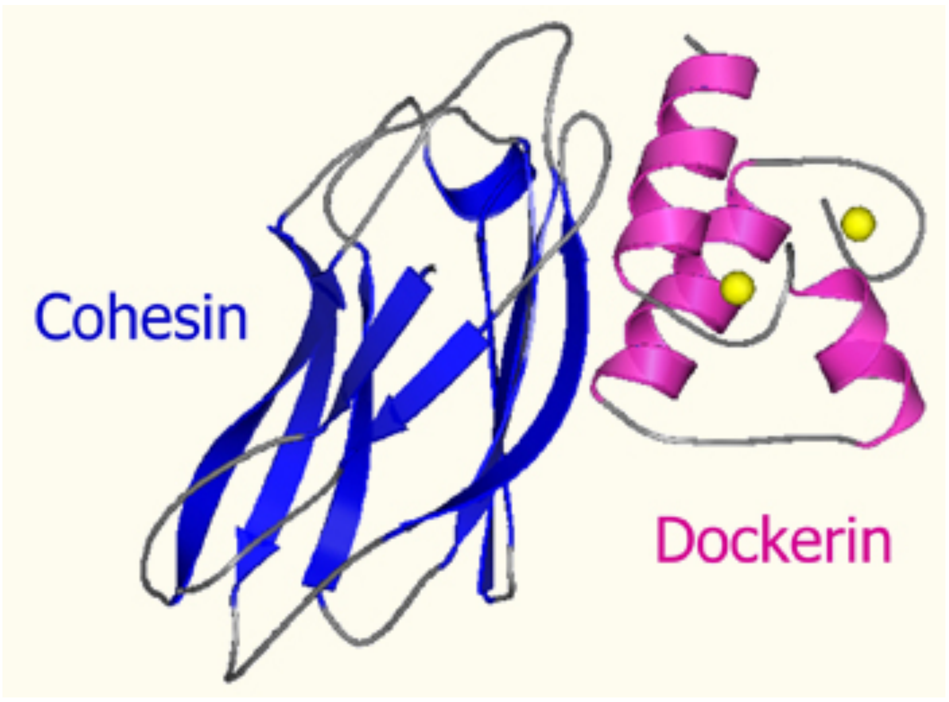


Figure 2.4 Cohesin and Dockerin Modules

The outcome of cohesin-dockerin recognition can be used to design nanomaterials, nanodevices as contructs called nanosomes. Besides it's nanotechnological uses, this interaction can artificially be used for biofuels as renewable cellulose resource instead of petroleum. (Ed Bayer's Group, 2012)

As not all cohesion and dockerins interact with each other, researchers classified them into three categories known as Type-1, Type-2 and Type-3 modules. Thus, a Type-1 dockerin and cohesin will interact.

Cohesin modules include 140 amino-acid residues and they don't have tryptophan, tyrosine, and cysteine residues (amino acids). Cohesins have variability in cellulosome-producing bacteria, and even within a single bacterium. (Peer et al., 2009)

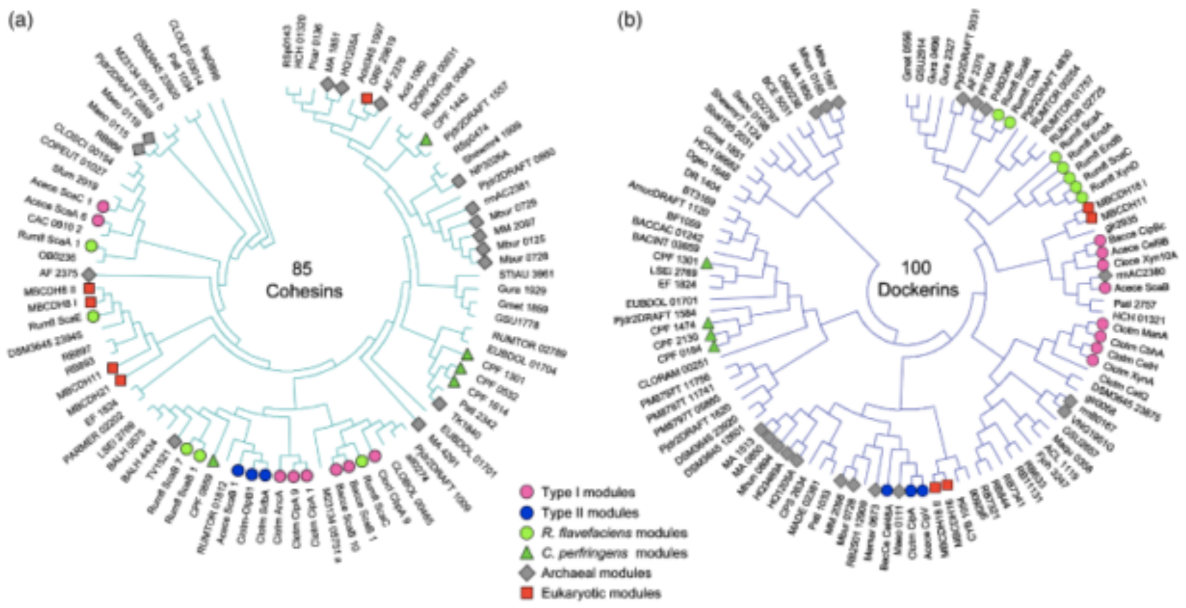


Figure 2.5 Phylogenetic distribution of representative cohesins (a) and dockerins (b) in the three domains of life taken from (Peer et al, 2009)

Dockerin modules consist of 60–70 amino-acid residues and are classified into types according to the cohesin with which they interact. (Peer et al., 2009)

3. Computational Background

This chapter describes key elements that should be known prior defining our work for supervised classification of proteins.

3.1 Machine Learning

The Department of Engineering at Cambridge University defines machine learning as follows

Machine learning is a multidisciplinary field of research focusing on the mathematical foundations and practical applications of systems that learn, reason and act. Machine learning underpins many modern technologies, such as speech recognition, robotics, Internet search, bioinformatics, and more generally the analysis and modeling of large complex data. Machine learning makes extensive use of computational and statistical methods, and takes inspiration from biological learning systems. [1]

It is important to add here that one of the tasks of machine learning is to find patterns in and make inferences based on unstructured data.

[1] http://cbl.eng.cam.ac.uk/Public/MLG/

One of the traditional areas of application for machine learning is classification, which is used in this work to find types of protein sequences. Based on our collection of sequences, the aim is to classify each cohesin and dockerin sequences into one of 3 possible types (1, 2 ,3) with regard to their tendency of interaction. Two of the methods used in Machine Learning for classification are: supervised methods and unsupervised methods. In supervised methods, the class label of each data sample is taken into account while building a classifier by looking at the features and their values of each class. In unsupervised methods, class labels are unknown and find out the real classes of data samples. Clustering is an example, that is, grouping together data samples which show similar patterns. Given that all the sequences we use in our work have already been clustered via CLUSTALW by he study of Peer el all (Peer et al., 2009) shown in Figure 2.5. Thus we will make use only of supervised methods to classify cohesin protein sequences.

Machine learning has different algorithms from several different families that solve problems in specific ways. The three families of classifiers that is used in this thesis are: Decision Trees, Bayesian classifier and Support Vector Machines. Detailed information about these will be given in next sections. Given the large number of features annotated in each sequence and the large number of sequences themselves, machine learning (using WEKA API programmed in Java) suits for the needs. Those classifiers have been chosen to be able to compare with previous classification tasks done in bioinformatics, as they are used widely.

Firstly Decision Tree schemes will be introduced and then how decision trees is built and optimized will be given.

3.2 Decision Trees

In this section, definition of decision trees and how they can be used in order to assign types to each one of the sequences in our data set based on the value of each feature will be mentioned. In addition to this, how decision trees are built and how they can be optimized will be detailed.

3.2.1 Definition

Decision Trees (DTs) are a specific machine-learning scheme, which is usually known as a “divide and conquer” approach. To solve a complicated problem, it is divided into various sub-problems/tasks and and a solution is found to each of these sub-problems. In the end, a solution to our original problem is found (thus “conquering”). In a classification problem, one is interested in assigning a class to a given input, based on the characteristics (attributes/features and their corresponding values) of that input.

Example classes that needs to be assigned to instances is shown below:

a) Yes or No (If we want to know whether it’s ok to play tennis on weather problem)

b) Type1, Type2, Type3 (When we are trying to know which type the protein is from)

c) Spam/Non-Spam (is the email spam or not).

In all the cases above there are a group of features and corresponding values that needs to be found in order to decide which class a given belongs to, in opposition to all the other classes it does not belong to. Decision Trees has different implementations with specific methods but all of them include “divide and conquer” approach.

3.2.2 The Basics

Decision Trees are basically a way to do classification by putting the data into different branches. Each node inherits all the attribute values of their ancestors. Except leaves, at each node in a decision tree, a rule is asked and according to the answer, data samples are assigned to one branch or another of the tree. This way, we start with our all collection of samples at the top node of the tree and from then, on at each node in the tree only a subset of the samples will be assigned to a specific branch. This flow continues until no more rules are remained and a final classification is made.

3.2.3 Divide and Conquer

A node in a DT is a rule in the tree at which a decision has to be made. The root node contains all the samples in our collection, which needs to be classified so this is the least informative node in the tree. From the root node, one attribute/variable is chosen to analyze in the samples in order to decide how to separate those samples from that point on. By creating branches using attribute/features as rules, whole tree is grown.

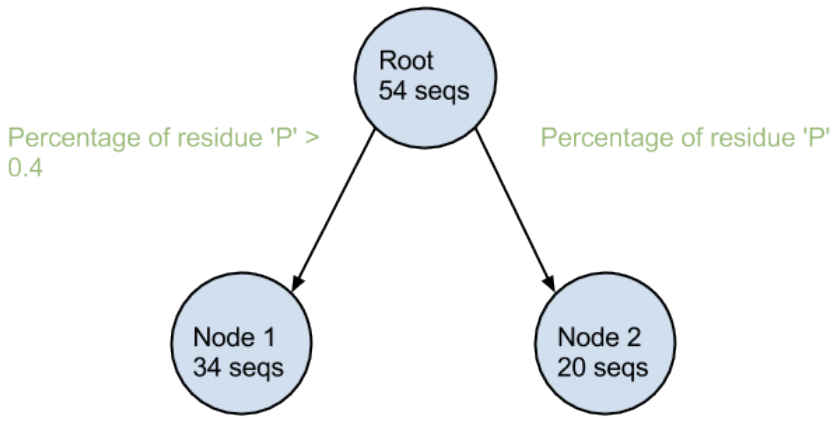


Figure 3.1 Simple identification of protein sequences based on residue percentage

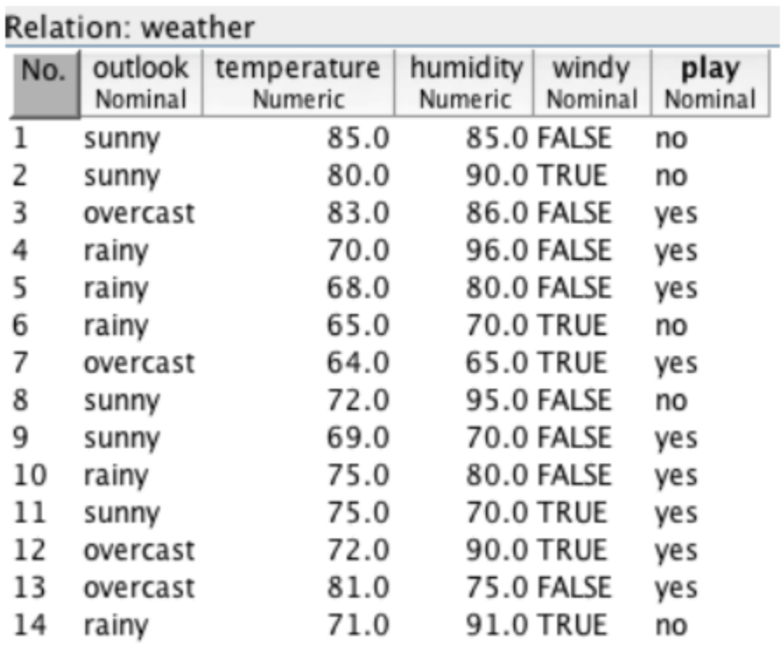
In the example above, after finding how often the residue “e” appears in each sequence, we can then make an initial decision how to find a rule. Leaf nodes of DTs do not branch unlike internal nodes.

3.2.4 Building Decision Trees

In building a decision tree, examining patterns in a collection of samples helps classifying unseen data. All these samples are put in “the root node”, since it is from this node that we will start growing our tree. This is done by analyzing all possible attributes in our training set for one of them that helps the most in reducing uncertainty (“entropy”) as to which class a training sample belongs. This helps separating samples, which are likely to belong together from those that are likely to be different.

A very common problem of Weather is chosen here to demonstrate decision trees. Because it has its small number of attributes and it’s easy to understand how DTs work.

In this section and sections to follow, all tables and figures related to the weather problem have been taken either from the book Data Mining: Practical Machine Learning Tools and Techniques (Ian & Eibe, 2011) and realized on WEKA. The table below contains the data with respect to the weather problem (Figure 3.2):



Relation: weather

No.	outlook Nominal	temperature Numeric	humidity Numeric	windy Nominal	play Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no

Figure 3.2 Weather data taken from Weka

Here there are five variables and 14 instances (training samples) from which we have to build our DT. There are 4 variables/attributes (outlook, temperature, humidity and windy), which are used to help predict another variable, called the class variable, which tells if game will be played or not. Temperature and humidity are numeric (contiguous) attributes, whereas outlook, windy and play are nominal attributes. Numeric attributes have as values either integers or real numbers, whereas nominal (categorical) attributes have a small set of possible values. For each node, we have to decide if it's a leaf node or needs to be splited and if it should be splited, which node should be used. Below is the (Figure 3.3) a fully-grown tree for the weather problem:

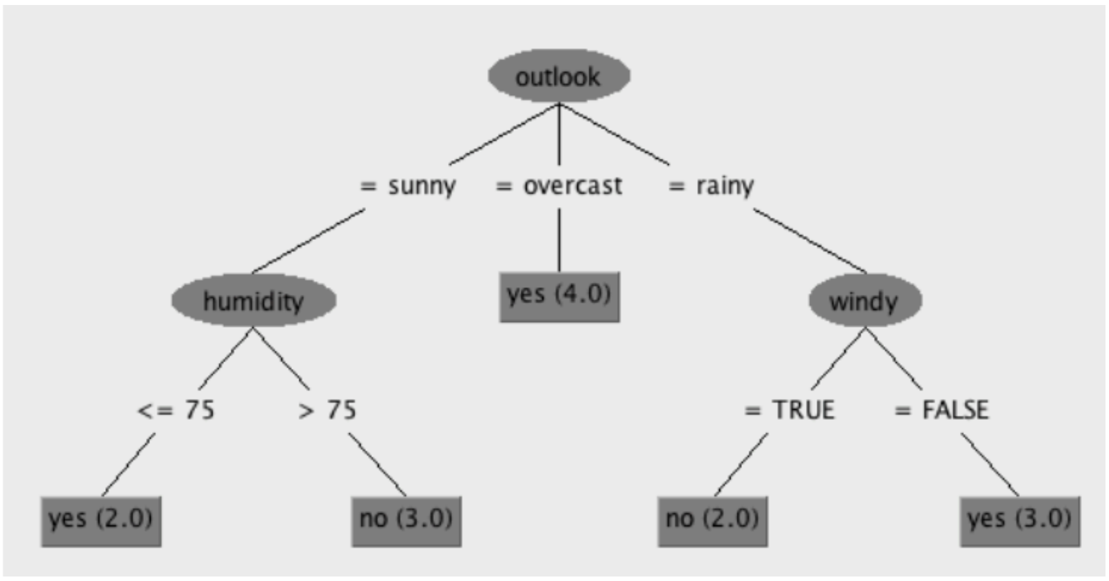


Figure 3.3 A possible DT for the weather data (visualization in WEKA)

3.2.5 Information Gain

Information Gain (IG) is dependent on the information (or entropy). The information in a system can be said to be higher when uncertainty is high in the system that means it is more difficult it is to predict an outcome generated by the system. In a simple case, if there are 3 colored balls and each one is of a different color, chance of guessing the color of a randomly selected ball is about 33%. However, if there are 10 differently colored balls, chance will be 10%. Second scenario contains more information than the first. Information is usually calculated through a mathematical measure called entropy. The higher the entropy, the higher the information and therefore the higher the uncertainity. Entropy is represented by a capital (H). The formula for calculating entropy given in bits because of the log being 2 is the following:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{3.1}$$

Here P is a probability distribution, in which the probabilities of each possible and discrete value Pi can take must add up to 1. Calculating the entropy at the root node of our weather problem:

Entropy at root = - 5/14 * log2 5/14 – 9/14 *log29/14 = **0.940 bits**

Information Gain for each attribute considered splitting can be calculated. The main procedure is calculating how much reduction in entropy each attribute is able to provide for the data and pick the one that provides the most reduction. IG for each possible attribute with relation to a specific node in the following manner is calculated, with the index i iterating over the child nodes of the current node:

$$IG(attribute_x) = entropy(curretnode) - \sum_{i=1}^{n} P(childnode)_i * entropy\ (chilnode)_i \tag{3.2}$$

Splitting on the attribute “outlook”, for example, at our root node, gives us the outcome shown in Figure 3.4:

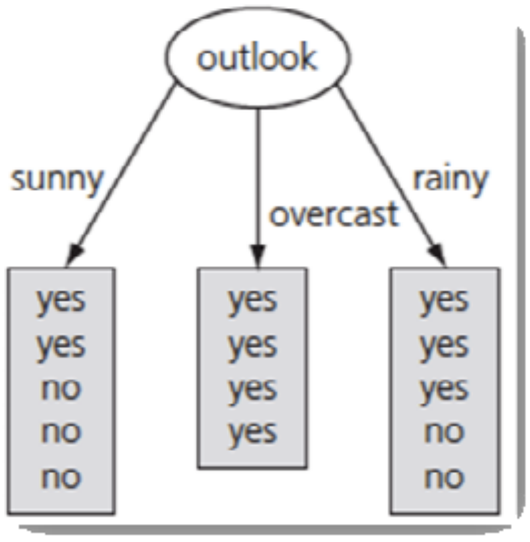


Figure 3.4 First split on weather data (taken from Hall et al. (2009))

The IG for attribute "outlook" in the weather problem is therefore:

IG (outlook) = info [5,9] – info [2,3], [4,0], [3,2] = IG (outlook) = 0.940 – [5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971] = 0.940 – 0.693 **= 0.247 bits**

For the IG of other 3 attributes as well, we get:

IG (temperature) = 0.029 bits IG (windy) = 0.048 bits IG (humidity) = 0.152 bits

Since maximum increase in Information Gain is important, attribute outlook at the root node is selected. This is done recursively for nodes created subsequently, and no descendent nodes of a node should be split on a nominal attribute already used before. DTs usually stop growing either when there are no attributes remained to split on. In section 3.2.6 two ways of pruning decision trees, tree raising and tree replacement that is used to make DTs smaller thus decreasing overfit for training data will be explored.

3.2.6 Optimization of Decision Trees

Generally Decision Trees is to first fully grow the tree so that each leaf only contains samples belonging to one class and this might perform good classification performance on training set, however it might show poor performance on test set. This known to be a overfitting problem, thus classifier being too specific to training set. Making decision trees impure to some degree, they do better when applied to new data. Modifying the fully grown tree so that it becomes more suitable for classifying new data is called post-pruning and usually consists of one (or both) of the following operations: subtree replacement and subtree raising.

3.2.6.1 Subtree replacement

Subtree replacement involves eliminating internal nodes of part of a tree (subtree) and replacing them by a leaf node found at the bottom of the subtree being eliminated. Figure 3.5 below, which represents labor negotiations in Canada, clarifies the idea. The label “good” indicates that both labor and management agreed on a specific contract. The label ”bad” indicates that no agreement was reached. The whole subtree starting at the node working hours per week in Figure 3.5a has been replaced by the its leaf node bad in Figure 3.5b.

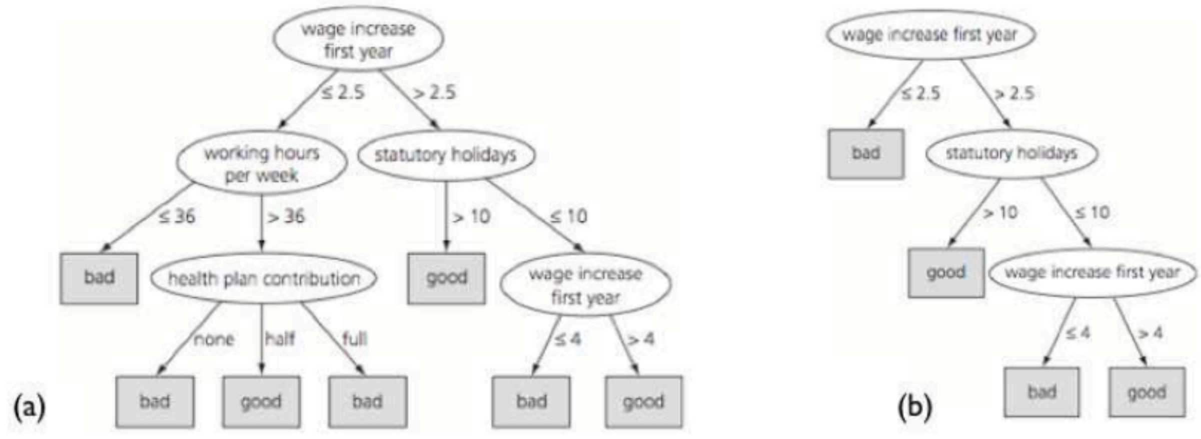


Figure 3.5 Subtree replacement. Taken from the book 'Data Mining: Practical Machine Learning Tools and Techniques' (modified)

3.2.6.2 Subtree raising

In subtree raising, a subtree that was lower before in a tree is moved up to occupy a higher position, substituted for what was previously found in that position (Figure 3.6).

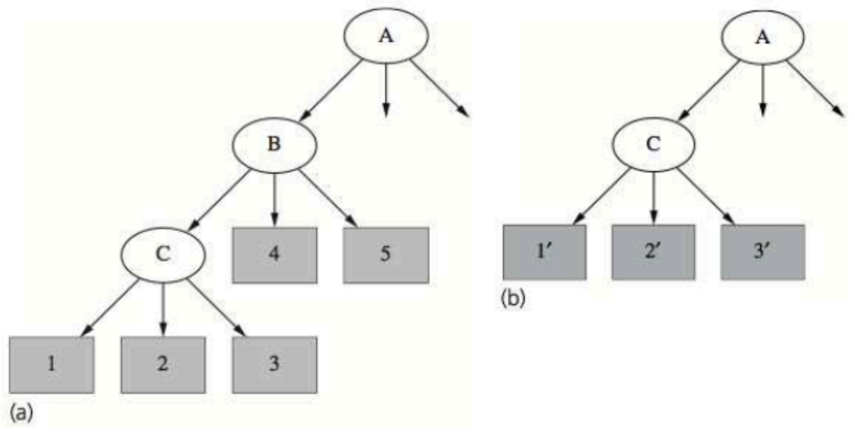


Figure 3.6 Subtree raising. Taken from the book 'Data Mining: Practical Machine Learning Tools and Techniques'

In Figure 3.6 , node C has been raised and substituted for node B.

In this chapter there were various ways to build and optimize decision trees. The choice of method is usually driven by the accuracy of classification and a balance must be reached between having a decision tree built based on and optimized for the training data (which therefore classifies those training samples very well) and a tree that is able to perform well on unseen (new) test data. In the next section (section 3.3.7, DT classifier in our experiments called J48 is explained briefly, that has built-in ways of deciding on the optimal final decision tree.

3.2.7 DT scheme used in our experiments: J48

The C4.5 algorithm was developed by Ross Quinlan (Quinlan, C4.5: Programs for Machine Learning, 1993) and builds upon Quinlan's previous ID3 algorithm (Quinlan, 1986) C4.5 is probably the most widely used DT algorithm in machine learning and a benchmark algorithm against whose performance any other algorithm should desirably be compared. It is a top-down, depth-first algorithm and uses a divide-and-conquer strategy. For numerical attributes, C4.5 makes use of binary splits (see Figure 3.7 below) and for nominal attributes it might use other n-ary splits (binary, tertiary, etc.). The default is to perform post-pruning and in the pre-pruning training process, nodes are split until they are pure, meaning they contain only samples belonging to a single class. Information Gain (IG) is used to decide which attribute is used for splitting a certain node and in the post-pruning process estimation of error is calculated by supposing that every sample that reaches a leaf will be classified as belonging to the majority class in that leaf. We can see below in Figure 3.7 The C4.5 algorithm applied to the weather data (visualization taken from WEKA) what a typical C4.5 Decision Tree looks like, applied to the weather data set that comes with WEKA.

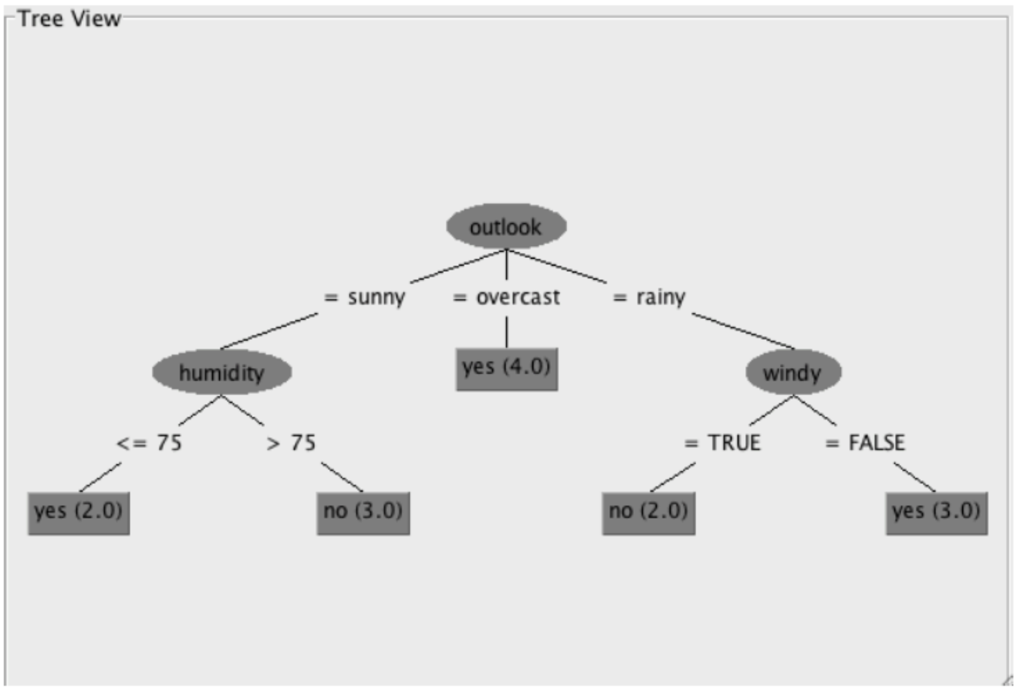


Figure 3.7 The C4.5 algorithm applied to the weather data (visualization taken from WEKA)

3.3 NAÏVE BAYES

Naïve Bayesian classifiers that are based on Bayes' Theorem are probabilistic algorithms for classification and which make the strong (naïve) assumption that the variables in the data are independent from one another. It means that, it assumes that all features in the datasets are independent of another one, however the class variable C is dependent on each of the features. "Naïve Bayes is widely used in machine learning due to its efficiency and its ability to combine evidence from a large number of features" (Schütze, 1999). However, as in the protein classification experiments, many of the variables are not independent from one another and treating them as if they were might lead to a decrease in the classification accuracy of classifiers such as Naïve Bayes.

A Naïve Bayesian model must first approximate the parameters that will be used by the model in order for it to arrive at a classification. These parameters are class probability and the feature probability distributions, both of which are calculated based on the training set. A class's prior can be calculated by dividing the number of samples in the training set that belong to that class by the total number of samples in the training data. The feature probability distributions can be calculated by first separating the data set into the different classes and then calculating, for each attribute in each class, the mean and variance of that attribute in that class. If µ2 is the mean of the values of X regarding class c, and σ2c to be the variance of the values of X regarding class c, then the probability of a certain value of X given a class, P (x=v | c) can be found by inserting it in the equation of a normal distribution containing as parameters the mean and covariance of the values of X for a specific class:

$$P(x = v|c) = \frac{1}{\sqrt{2\,\pi\sigma_c{}^2}}\, e^{-\frac{\pi - \mu_c{}^2}{2\,\sigma_c{}^2}} \tag{3.4}$$

In order to make a decision as to which class a certain data sample belongs to, the model calculates the conditional probability of each possible class given the observed values of each of the features present in the data. The Naïve Bayesian probabilistic model is described below:

Probability (C | F1, F2, F3, …, Fn) = P (C) * P (F1|C) * P (F2 |C) * … * P (Fn |C) / P (F1… Fn) (3.5)

The denominator of the formula does not depend on the class and since the feature values are given, only nominator is enough to be calculated. Therefore, the probability of a sample belonging to a certain class is given by this updated formula:

$$P(C)\prod_{i=1}^{n} p(F_i \mid C) \tag{3.6}$$

Each possible values of the target class (C) in the data is calculated and the class whose probability is the highest is chosen:

$$classify\ (f_1, \ldots, f_n) = argmax_c\, p(C = c) \prod_{i=1}^{n} p\ (F_i = f_i\ (C = c) \tag{3.7}$$

Decision Trees and Naïve Bayesian Classifiers was mentioned in this chapter. In addition, each Decision Scheme scheme has its own specificities. Depending on the data, one classifier might perform better over another and show much better accuracies, to find out which is better they should be run. They both try to decide on an optimal classifier configuration based on the features and their values, to increase the accuracy of classification. In addition to these, Support Vector Machines, which shows worse than DT and NBC is used but it is not detailed in this work.

3.4 Feature Selection

Supervised classification in bioinformatics is challenging partly because of the high dimensionality of the input variables. Many learning algorithms are known to lose accuracy when unnecessary features are given as input. Moreover, accuracy of some classification algorithms' like Naïve-Bayes may decrease quickly if correlated features are added (Kohavi & John, 1997). Those redundant features duplicate much or all of information contained in one or more other attributes. To tackle with the problems feature selection is used to remove irrelevant, redundant or noisy data by selecting a subset of the original feature set (Liu & Yu, 2005).

Feature selection is required if the dimension of input variables are higher than number of samples. Dimension reduction or feature selection has various advantages such as (i) providing more human understandable information and allow the data to be more easily visualized by focusing on a smaller number of features; (ii) generating more accurate estimates by excluding noises and reducing overfitting and (iii) provide faster and more efficient models that use less time and memory (Ma & Huang, 2008; Guyon & Elisseeff, 2003; Saeys et al., 2007)

Feature selection methods can be classified into three categories: filter, wrapper and embedded. First two are the most known ones, and embedded is used frequently user has no knowledge it is being used as it's built in some classifiers. Filter approach separates feature selection from classifier construction. Although their simplicity and efficiency, they are not effective excluding redundant and correlated features (Koller & Sahami, 1996). Wrapper approach evaluates classification performance of selected features and keeps searching until certain accuracy criterion is satisfied. Wrappers can be computationally expensive and have a risk of overfitting to the model. Embedded approach embeds feature selection within classifier construction, tree classifications with pruning is an example of this type.

Feature selection algorithms with filter and embedded models may return either a subset of selected features or the weights (measuring feature relevance) of all features. According to the type of the output, they can be divided into feature weighting and subset selection algorithms. Algorithms with wrapper model usually return feature subset. Figure 3.8 shows how the process of feature selecting works. Firstly with subset generation, a selected search algorithm produces candidate feature subsets. Each subset is then evaluated and compared to others according to a given evaluation criterion.

After comparison of all subsets repeatedly until stopping criterion, best one is selected. Finally selected subset is validated using the pre-selected classifier.

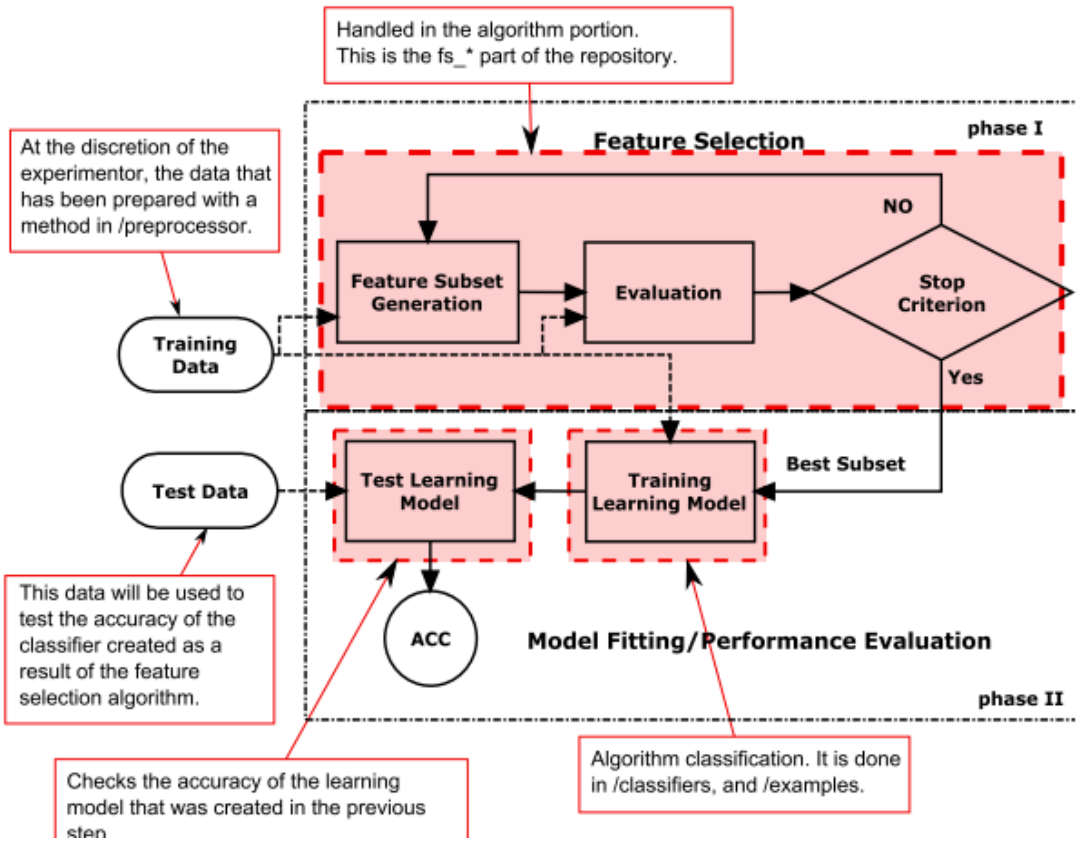

Handled in the algorithm portion.
This is the fs_* part of the repository.
At the discretion of the experimentor, the data that has been prepared with a method in /preprocessor.
Feature Selection
phase I
NO
Feature Subset Generation
Evaluation
Stop Criterion
Yes
Training Data
Test Data
Test Learning Model
Training Learning Model
Best Subset
ACC
Model Fitting/Performance Evaluation
phase II
This data will be used to test the accuracy of the classifier created as a result of the feature selection algorithm.
Checks the accuracy of the learning model that was created in the previous step
Algorithm classification. It is done in /classifiers, and /examples.

Figure 3.8 Different components in the standart feature selection

Filter methods rank each feature according to some metrics such as Information Gain, Chi-Squared, and Pearson Correlation Coefficient to rank highest features and remaining low ranking features are removed.

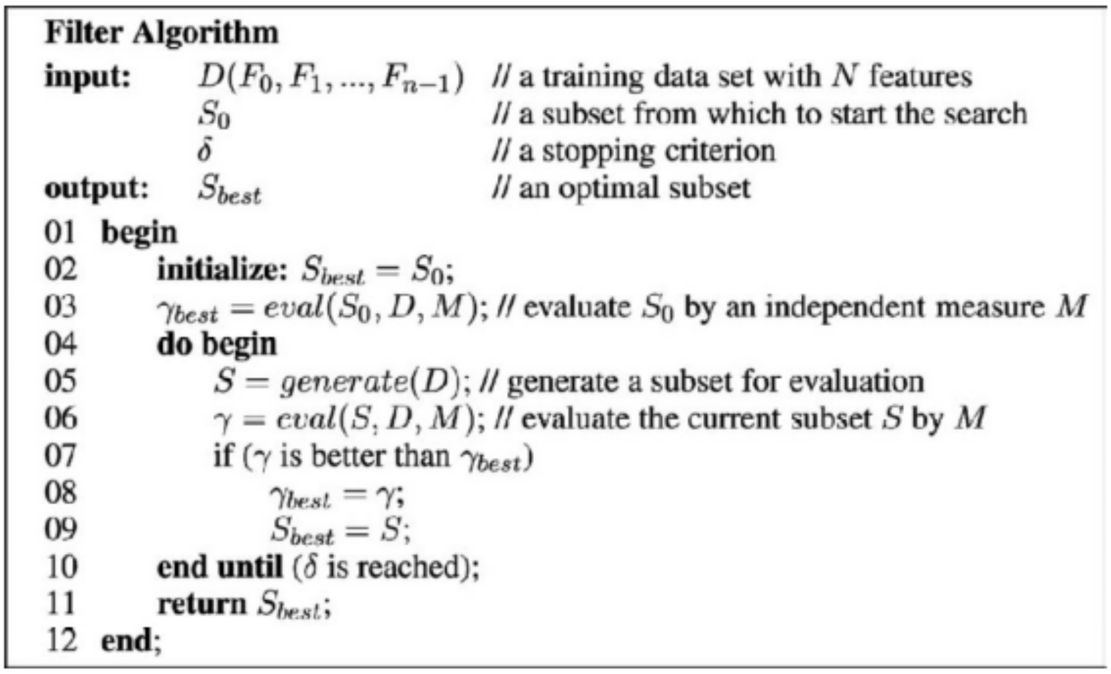
As filter methods explains how useful of a feature is by using a metric, wrapper methods gives that information via integrating with a learning method. So the features are optimal for that learning method.

Feature selection should not be confused with other methods such as feature extraction, feature construction, feature weighting and feature creation. Feature extraction is a method that higher dimensional space is projected into lower dimensional one by using linear algebra, transformations or combinations. An example to feature extraction can be given as PCA (Principal Component Analysis) that creates new features with linear combinations of original features to maximize variation in the data. (Blum & Langley, 1997)

3.4.1 Filter Methods

Filter name comes from how they work, that is filtering out irrelevant features before induction occurs. From the characteristics of training set, some features are selected and others are excluded. As learning algorithm is not used to evaluate candidate sets, they can be combined with any learning algorithm after the filtering is complete. As a benefit, filter methods are a computationally efficient form of data pre- processing, unlike wrapper methods (Blum & Langley, 1997).

Figure 3.9 shows generalized form of a filter algorithm. Given a dataset D, begin with a given subset S0 (an empty set, a full set, or any randomly selected subset) and search through the feature space using a defined search strategy (may be forward or backward selection). Each generated subset S is evaluated by an independent measure M and compared with the previous best. If a better one is found, it's flagged as best. Until a stopping criterion defined the search continues. The last current best subset S(best) is selected as best subset for final feature subset (Liu & Yu, 2005).



```
Filter Algorithm
input:   $D(F_0, F_1, ..., F_{n-1})$   // a training data set with $N$ features
         $S_0$                         // a subset from which to start the search
         $\delta$                      // a stopping criterion
output:  $S_{best}$                    // an optimal subset
begin
    initialize: $S_{best} = S_0$;
    $\gamma_{best} = eval(S_0, D, M)$; // evaluate $S_0$ by an independent measure $M$
    do begin
        $S = generate(D)$; // generate a subset for evaluation
        $\gamma = eval(S, D, M)$; // evaluate the current subset $S$ by $M$
        if ($\gamma$ is better than $\gamma_{best}$)
            $\gamma_{best} = \gamma$;
            $S_{best} = S$;
    end until ($\delta$ is reached);
    return $S_{best}$;
end;
```

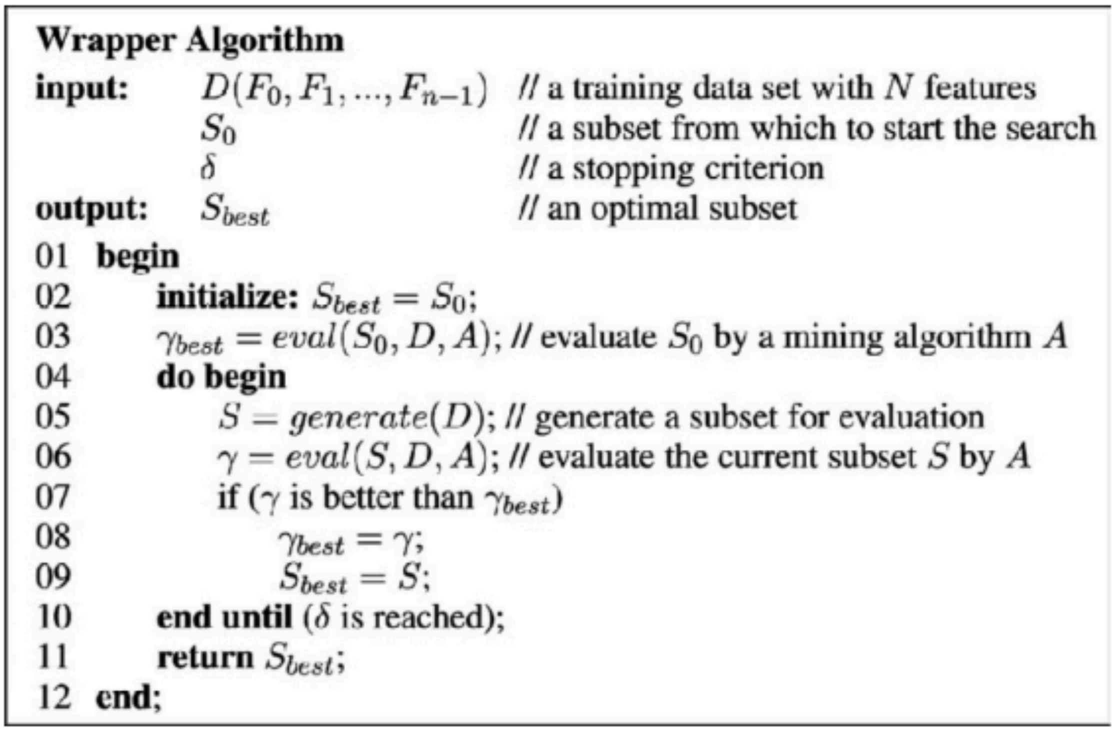
Figure 3.9 Generalized Filter Algorithm taken from (Liu & Yu, 2005)

There are various filtering metrics, one is correlation with the target function and then select the k features with the highest values. Other commonly used metrics are Information Gain, Odds Ratio, Log Probability Ratio, FOCUS, RELIEF, Potential Difference, Pearson Correlation Coefficient.

Some problems with filtering methods may arise, for example if the number of selected features are too high, then irrevelant features may be selected. If the number is too low, then relevant features might not be selected. Both of the situations will effect accuracy, especially in Naïve Bayes. Additionaly, as filtering methods don't work in conjunction with learning algorithms, they may miss some useful features.

3.4.2 Wrapper Methods

As they work closely with the learning algorithm, not as pre-processor or post-processor, they are called with name Wrapper. Their advantage is having better accuracy compared to filter and embedded methods. Classifier, which will use the feature subset should provide a better estimate of accuracy than a separate.



```
Wrapper Algorithm
input:   $D(F_0, F_1, ..., F_{n-1})$   // a training data set with $N$ features
         $S_0$                         // a subset from which to start the search
         $\delta$                      // a stopping criterion
output:  $S_{best}$                    // an optimal subset
begin
    initialize: $S_{best} = S_0$;
    $\gamma_{best} = eval(S_0, D, A)$; // evaluate $S_0$ by a mining algorithm $A$
    do begin
        $S = generate(D)$; // generate a subset for evaluation
        $\gamma = eval(S, D, A)$; // evaluate the current subset $S$ by $A$
        if ($\gamma$ is better than $\gamma_{best}$)
            $\gamma_{best} = \gamma$;
            $S_{best} = S$;
    end until ($\delta$ is reached);
    return $S_{best}$;
end;
```

Figure 3.10 Generalized wrapper algorithm (Liu & Yu, 2005)

Figure 3.10 shows pseudo-code for the wrapper method. For each generated subset S, the algorithm is evaluated by applying the learning algorithm to the data with subset S and evaluating the accuracy. Therefore, different learning algorithms may produce different feature selection results. Varying the search strategies via the function generate(D) and learning algorithms (A) can result in different wrapper algorithms.

Learning algorithm is used for feature selection, so the feature set depends on that algorithm, and is supposed to give better performance for accuracy. However compared to filter methods, they are more computationally expensive. (Liu & Yu, 2005)

4. Experiments

4.1 Data Set

As previously introduced, not all cohesin and dockerins interact with each other so researchers classified them into three categories known as Type-I, Type-II and Type-III modules. Thus, only a Type-I dockerin and cohesin will interact. Cohesin modules include 140 amino-acid residues and they do not have tryptophan, tyrosine, and cysteine residues. The dataset is provided by Weizmann Institute of Science and it is separated into Training and Test sets as seen Table 4.1 and redundant sequences are removed.

The dataset contains 128 instances, and the distribution of them are seen in Table 4.1.

Table 4.1 Distribution of Cohesin sub-families in our Dataset

		# of sequences
Training Set	Type I	42
	Type II	26
	Type III	5
Test Set	Type I	34
	Type II	17
	Type III	4

Dataset contains very unbalanced classes for instances, because type 3 sub-family of cohesin is too low. As a result we'll also include balanced accuracy to evaluate the results.

4.2 Amino Acid Grouping Schemes

When n-grams are used for feature extraction from a sequence, number of features increase from 20 to 20^n with a sequence of length n. To overcome this, the standard amino acid alphabet consisting of 20 amino acids can be grouped using substitution matrices, bio-chemical properties or inter-group energetic interactions. Reducing the number of amino acid alphabet makes computation more effective. and minimizes noise on sequence representation (Davies et al., 2008). In this study, grouping schemes are the key element of increasing probability to find protein motifs by detecting similar n-grams in addition to identical ones.

The grouping schemes used for motif extraction and classification phase in this work and where they are taken from are shown in Table 4.2 alphabetically.

Table 4.2 Reduced amino acids used in this work

Alphabet	Size	Reference
Ab	10-19	(Andersen & Brunak, 2004)
Davies	7, 9	(Cobanoglu et al., 2011)
Dssp	10-14	(Solis & Rackovsky, 2000)
Gbmr	10-14	(Solis & Rackovsky, 2000; Hall, Frank, & Holmes, 2009)
Hsdm	10,12, 14-17	(Prlic et al., 2000) (Li et al., 2003)
Lwi	10-19	(Prlic et al., 2000)

Lzbl	10-16	(Liu et al., 2002)
Lzmj	10-16	(Liu et al., 2002)
Ml	10-15	(Murphy et al., 2000)
Sdm	10-14	(Prlic et al., 2000)
Sezerman	11	(Cobanoglu et al., 2011)

4.3 Class Specific Motif Scoring Function

A motif is expected to be class-specific if it is seen in one family but never seen or seen in few numbers on other families. A scoring based function should be used to find those motifs and eliminate useless ones from the list of all motifs. This would give a motif, which is found in many sequences in a family a high score. Motifs that occur in few sequences in one family or occur multiple families would be given low score that helps them be removed from high correlated list of motifs.

As discussed above, scoring function uses n-grams to find discriminative motifs from given protein sequences. To give lower score to the n-grams that are found in multiple families, we were inspired from a wide known method in text mining The Term Frequency Inverse Document Frequency (TFIDF) (Salton, 1991). This metric is combined with discriminatory ratio (DR) of n-gram in a family against other families.

Occurrence frequency of a motif in only one sequence of a target family is not same as occurrence frequency in a high ratio of sequences. Thus in order to compare discriminatory ratio (DR), we need to find Motif Occurrence Rates of motifs.

Definition 1. Motif Occurrence Rate in Family

$$MORF(i,f) = \frac{\left|\{s \in f : i \in s\}\right|}{|f|} \tag{4.1}$$

where,

- $\left|\{s \in f : i \in s\}\right|$: number of sequences of family f where the motif i appears.
- $|f|$ is the total number of sequences in family f

Definition 2. Discriminatory Ratio (DR)

$$DR(i,f) = \log\left(\frac{MORF(i,f)}{\{MORF(i,m) : m \in F, m \neq f\}}\right) \tag{4.2}$$

where,

- MORF (i, f) is Motif Occurrence Rate of i in family f,
- m is a family in set F that is not f,
- $\overline{MORF(i,m)}$ is arithmetic mean of motif i's occurrence rate in all m (family except f). Minimum rate of 0.1 added to the mean to prevent division by zero.

Discriminatory Ratio gives a high score if an n-gram is highly occurred in family f, and found less in others. However, regardless of occurrence rate, if more than one family includes the motif, then that motif will not be specific to a family. So inspired from Inverse Document Frequency, Inverse Family Frequency is calculated.

Definition 3. Inverse Family Frequency (IFF)

$$IFF(i) = \log\left(\frac{|F|}{|\{f \in F : MORF(i, f) > 0\}|}\right) \tag{4.3}$$

where

- F is the set of all families,
- MORF is the Motif Occurrence Rate in Family function defined above.

The denominator gives number of families that contains motif i in at least one sequence. This is clearly similar to IDF, however IDF is for documents (or sequences in our case), and by using Motif Occurrence Rate it can now be used for determining inverse family frequency.

By using Inverse Family Frequency (4.3) and Discriminatory Ratio (4.2), Motif Specificity Score (4.4) of a motif for a family can be calculated.

Definition 4. Motif Specificity Score (MSS)

$$MSS(i, f) = IFF(i) \times DR(i, f) \tag{4.4}$$

where,

- IFF (i, f) denotes the Inverse Family Frequency (4.3) of motif i in family f,
- DR (i, f) denotes Discriminatory Ratio (4.2) of motif i in family f.

Table 4.3 Some N-Gram Motifs found using Sezerman grouping scheme

Motif	MSS (4)	Type I Rate (1)	Type II Rate (1)	Type III Rate (1)
DIS	0.54	65 %	20 %	0 %
YID	2.36	71 %	0 %	0 %
gYQ	0.88	5%	86 %	0 %
RYaY	2.77	0 %	0 %	100 %

Motif Specificity Score of a motif for a particular family is positively correlated with the occurrence rate of a motif in that family but inversely correlated with the occurrence rate in families, which that motif is present in other families.

MSS is calculated for every motif and family, however by comparing the higher occurrence rate beforehand, only higher values are used for numerator in DR. As result, instead of running time O(n), O(logn) is achieved.

4.4 Motif Discovery Results

To verify our Motif Specificity Function, we tested the algorithm with training dataset that consist of sub-families of Cohesin: Type I, Type II and Type III.

We calculated both the occurrence rates and Motif Specificity scores of every n-gram motif with different size of n in cohesin families using training dataset.

Some results obtained using Sezerman amino acid reduction scheme is seen on Table 4.3. Analysis has shown that as more families include a motif, MSS drastically decreases by the effect of IFF. This can be clearly seen from “YID” and “DIS” motifs.

In addition, occurrence numbers of common motif is also important. As the gap of motif occurrence rates between families decreases, MSS also decrease by the effect of Discriminatory Ratio as seen between “gYQ” and “DIS” motifs.

Our method is similar to what Srinivasan et al. (2013) have proposed. However there are key differences between that work and ours. Firstly they used total number of motifs found in a family to use in discriminate ratio instead of our occurrence rate. For unbalanced datasets like our cohesin family, this gave high score to the families, which include high number of sequences. Besides this, they used substitution matrix to extract n-gram motifs of sequences, and that has increased of running time complexity. By using pre-substituted amino acid grouping schemes, we have omitted that additional time.

In the figures below, the effect of MSS and grouping schemes to class specific motifs are shown. If the MSS threshold is increased, number of class specific motifs decrease. Up until 4-grams, grouping scheme (Sezerman as chosen) has negative effect on number of class specific motif, that is because combinations of distinct n-grams are very few when n is small. When grouping alphabets are used, this number also decreases by combining similar ones.

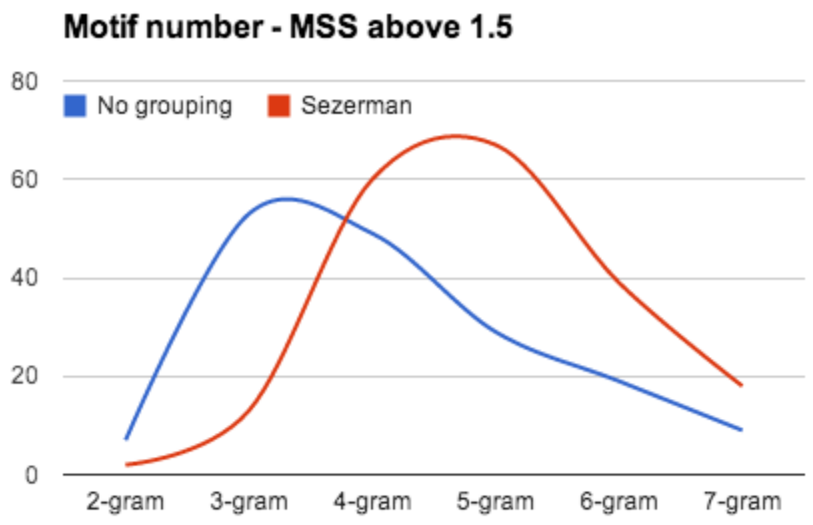


Figure 4.1 Specific motif number when MSS threshold is 1.5

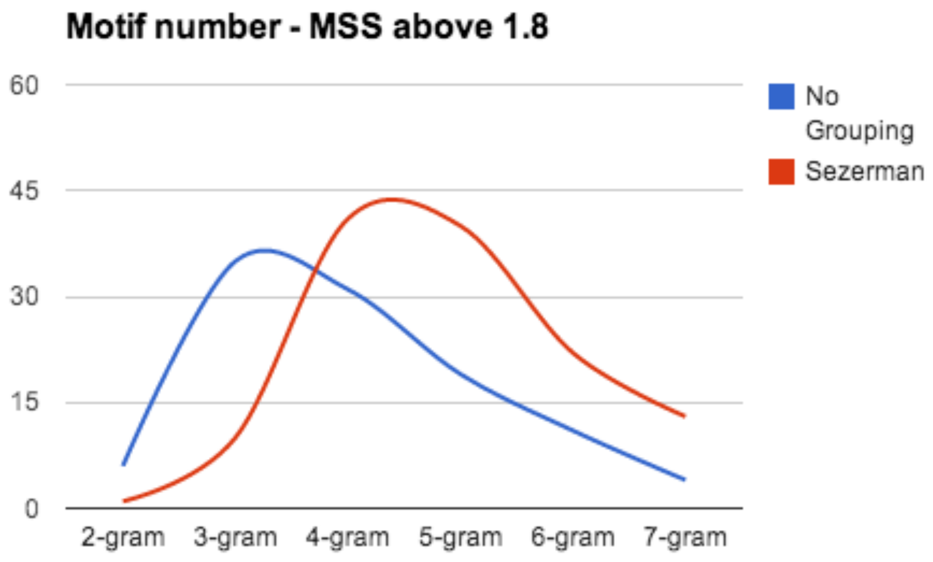


Figure 4.2 Specific motif number when MSS threshold is 1.8

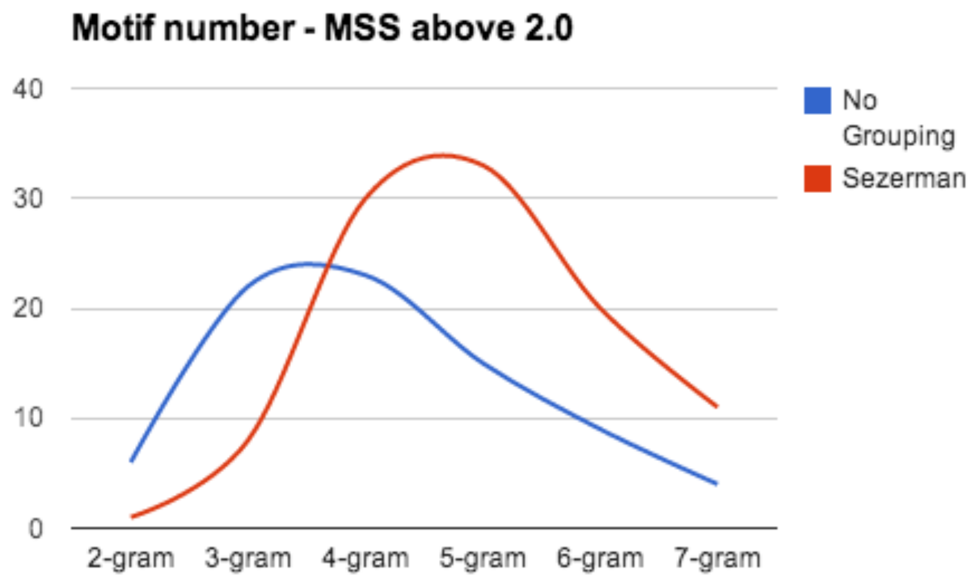


Figure 4.3 Specific motif number when MSS threshold is 2.0

4.5 Classification

By sorting the results using Motif Specificity Score for each family, class-specific motifs are obtained and in classification phase, this is used to help classification methods positively alike feature selection. Top highly correlated features with classes are filtered, thus motifs that contain useless information are discarded. However removal of redundant features is not implemented as this work is only about classification using class-specific motifs.

Classification was carried out using widely distributed Weka (v3.7.9) (Hall et al., 2009) bindings for Java language. Support Vector Machines (LibSVM (Chang & Lin, 2001)) and J48 Decision Tree algorithms in standard package of Weka are used. For SVM, grid search to find optimal parameters on training data set is also implemented. The classifier was trained using five-fold cross generate a model. In five-fold cross validation, the training set is randomly partitioned into five roughly equal-sized parts.

Of the 5 parts, 4 parts are used as training data and the remaining single part is retained as the validation data for testing the model. The cross-validation process is then repeated 5 times, with each of the 5 parts used exactly once as the validation data. Although the performance of the classifier is evaluated using cross-validation, Weka outputs a model built from the full training set and that model is used to test on the normalized test set.

Feature values of both training and test set are normalized motif counts in each sequence (i.e motif count in sequence divided by total number of motifs in that sequence) that are calculated using 55 amino acid grouping schemes described above. Top n 2- to 4-gram motifs for each family that have MSS higher than zero, as n starts from 5 and increased by 5 until reached to 100 are selected.

Because using every combination of the above elements required high computational resources, the tests were run on Amazon Web Services, Elastic Clouds (http://aws.amazon.com/). The framework was written in JAVA, with Weka libraries imported. Another language used is Javascript which extracts all features as motifs from raw data of FASTA format to be classified.

Because of very imbalanced class distribution, balanced accuracy was used that is defined below.

Balanced accuracy :

$$\frac{\sum_{i=0}^{n} \frac{Number\ of\ correctly\ classified\ of\ class_{\ i}}{total\ number\ of\ instances\ of\ class_i}}{number\ of\ classes} \tag{4.5}$$

4.6 Classification Results

We have computed the reduced amino acid composition with 6 different n-gram sizes for cohesin proteins and used a feature selection procedure introduced in previous section to reduce the number of features that can be used to represent a protein sequence in feature space prior to generating a model using SVM, J48 and Naïve Bayes classifiers to predict the sub-family of a protein.

Table 4.4 shows classification results sorted by balanced accuracy of Test Dataset. In order to analyze, during constructing tables, if more than one result with the same accuracies obtained, priority was given to the ones that has least features (effecting running time), lower n-gram size and lengthier grouping size (more information in motif).

Table 4.4 Classification Results sorted by Balanced Test Accuracy

N-gram	**Grouping**	**Features**	**Method**	**B.Acc. Test %**
2	Dssp14	10	J48	95.55
	Dssp12	5	Naïve Bayes	95.07
	Gbmr12	15	J48	94.65
3	Dssp13	10	Naïve Bayes	97.29
	Lzmj16	50	Naïve Bayes	96.39
	Lzmj16	15	J48	95.97
4	Lwi11	60	Naïve Bayes	**99.09**
	Sezerman	25	Naïve Bayes	98.19
	Sdm12	25	Naïve Bayes	98.19
5	Sdm13	10	Naïve Bayes	**99.09**
	Sdm14	15	Naïve Bayes	**99.09**
	Dssp10	25	Naïve Bayes	**99.09**

6	Hsdm12	15	Naïve Bayes	98.19
	Hsdm12	20	Naïve Bayes	95.97
	Hsdm12	10	Naïve Bayes	95.49
7	Hsdm12	20	Naïve Bayes	97.29
	Hsdm12	15	Naïve Bayes	96.39
	Hsdm12	10	Naïve Bayes	95.49

As seen in the results, N-gram size of 5, Sdm13 alphabet with 10 features for each class has given the best accuracy of 99.09%. If more features are used, this accuracy can also be accomplished by Sdm14, Dssp10, Lwi11.

When classification all features but feature number are same, increasing or decreasing the feature numbers from optimum, effects the accuracy negativelly. For example in 6-gram sized motifs in Table 4.4, increasing feature number to 20 from 15 decreases balanced test accuracy from 98.19 % to 95.97 %. Additionaly, decreasing feature number from 15 to 15, decreases balanced test accuracy from 98.19 % to 95.46 %.

Generally, Naïve Bayes performed as the best supervised classification method regardless of other criterias, whereas Support Vector Machines could not get into the list in Table 4.4 but experiments show that SVM has best accuracy of 93.75% being the least optimal classifier for our work.

5. Conclusion

The main task of this work was classification of cohesin protein sequences and finding class discriminative motifs of each type. Because all the proteins are similar, to find best classification model, different combinations of n-grams sizes, amino acid groupings/alphabets, feature numbers and classification algorithms were used.

N-grams help improve local ordering of amino acids by combining them. Similarly, amino acid groupings help reduce complexity of the sequences by grouping them and associating to one amino acid according to their properties. Both of them are used to find motifs that have high specificity for the protein sub-families.

As the result of classification tasks using class specific motifs selected by different n-gram sizes, amino acid groupings success rate of 99.09 % is achieved. It is possible to find class-specific motifs with the method described in this work using Reduced Amino Acid Alphabets with n-grams that discriminate Cohesin family proteins and those motifs can be used for supervised classification to accurately classify Cohesin proteins.

References

Albayrak, A., & Sezerman, U. (2012). Discrimination of thermophilic and mesophilic proteins using reduced amino acid alphabets with n-grams. Current Bioinformatics , 7 (2), p.152-158.

Andersen, C., & Brunak, S. (2004). Representation of protein-sequence information by amino acid subalphabets. Ai Magazine , 25 (1), p.97-104.

Bailey, T., Bodén, M., Buske, F., Frith, M., E. Grant, C., Clementi, L., et al. (2009). MEME SUITE: tools for motif discovery and searching. Nucleic Acids Research , 37, p.202-208.

Bayer EA, E., Chanzy, H., Lamed, R., & Shoham, Y. (1998). Celllose, Cellulases and cellulosomes. 8, p.548-557.

Blum, A. L., & Langley, P. (1997). Selection of Relevant Features and Examples in Machine Learning. Artificial Intelligence , p.245-271.

Chang, C., & Lin, C. (2001). {LIBSVM}: a library for support vector machines. [Accessed May, 2013]

Cobanoglu, M., Saygin, Y., & Sezerman, U. (2011). Discovering Key Ligand Interaction Sites for Classification of GPCR Sequences. IEEE/ACM Trans Comput Biol Bioinform , p.1495-508.

Davies, M., Secker, A., Freitas, A., Clark, E., Timmis, J., & Flower, D. R. (2008). Optimizing amino acid groupings for GPCR classification. Bioinformatics , 24 (18), p.1980–6.

Ed Bayer's Group, B. (2012). Biofuels., from Weizmann Institute: http://www.weizmann.ac.il/Biological_Chemistry/scientist/Bayer/biofuels [Accessed May, 2013]

Grant, C. E., Bailey, T. L., & Noble, W. S. (2011). FIMO: scanning for occurrences of a given motif. Bioinformatics , 27 (7), p.1017–8.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. JMach LearnRes , p.1157–82.

Hall, M., Frank, E., & Holmes, G. (2009). The WEKA data mining software: an update. SIGKDD Explor Newsl , 11 (1), p.10-8.

Ian, H., & Eibe, F. (2011). Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. Artificial Intelligence: Special Issue on relevance , 97 (1-2), p.273-324.

Koller, D., & Sahami, M. (1996). Toward Optimal Feature Selection. Proceedings of the Thirteenth International Conference on Machine Learning, (pp. p.284-292).

Li, T., Fan, K., Wang, J., & Wang, W. (2003). Reduction of protein sequence complexity by residue grouping. Protein Eng , 16, p.323-30.

Liu, H., & Yu, L. (2005). Toward Integrating Feature Selection Algorithms for Classification and Clustering. IEEE Transactions on Knowledge and Data Engineering , 17 (4), p.491-502.

Liu, X., Liu, D., Qi, J., & Zheng, W. (2002). Simplified amino acid alphabets based on deviation of conditional probability from random background. Phys Rev E Stat Nonlin Soft Matter Phys , 66, p.1.

Ma, S., & Huang, J. (2008). Penalized feature selection and classification in bioinformatics. Briefings in bioinformatics , 9 (5), p.392–403.

Murphy, L. R., Wallqvist, A., & Levy, R. M. (2000). Simplified amino acid alphabets for protein fold recognition and implications for folding. Protein Eng , 13, p.149-52.

Pagès, S., Belaich, A., Belaich, J.-P., Morag, E., Lamed, R., Shoham, Y., et al. (1997). Species-specificity of the cohesin–dockerin interaction between Clostridium thermocellum and Clostridium cellulolyticum: prediction of specificity determinants of the dockerin domain. Proteins , 29, p.517–527.

Peer, A., Smith, S. P., Bayer, E. A., Lamed, R., & Borovok, I. (2009). Noncellulosomal cohesin- and dockerin-like modules in the three domains of life. FEMS microbiology letters , 291 (1), p.1-16.

Prlic, A., Domingues, F. S., & Sippl, M. J. (2000). Structure-derived substitution matrices for alignment of distantly related sequences. Protein Eng , 13, p.545-50.
Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann.

Quinlan, J. R. (1986). Induction of Decision Trees. Mach. Learn. , 1 (1), p.81-106.
Reinhardt, A., & Hubbard, T. (1998). Using neural networks for prediction of the subcellular location of proteins. Nucleic Acids Res. , 26, p.2230–2236.

Saeys, Y., Inza, I., & Larranaga, P. (2007). A review of feature selection techniques in bioinformatics. Bioinformatics , p.2507–17.

Salton, G. (1991). Developments in automatic text retrieval. Science , 253, p.974-980.

Schütze, H. (1999). Foundations of Statistical Natural Language Processing. MIT Press.
Solis, A. D., & Rackovsky, S. (2000). Optimized representations and maximal information in proteins. Proteins , 38 (2), p.149-64.

Srinivasan, S. M., Vural, S., King, B. R., & Guda, C. (2013). Mining for class-specific motifs in protein sequence classification. BMC Bioinformatics , 14 (1), p.96.

Weizmann Institute of Science. (2012). Weizmann Institute of Science. Retrieved 5 5, 2013, from http://www.weizmann.ac.il/

Zhao, Z., Morstatter, F., Sharma, S., Alelyani, S., Anand, A., & Liu, H. (2010). Advancing Feature Selection Research. ASU feature selection repository, Arizona State University. Arizona State University.

Biographical Sketch

The author of this thesis was born in 1985 in Manisa, Turkey. He has studied in Dundar Ciloglu Anatolian High School between 1997 and 2003, and started his undergraduate education in the Computer Engineering Department of the Faculty of Electric and Electronics of Istanbul Technical University in 2003-2009 terms. Consequent to the graduation from the undergraduate degree, in 2010 he has enrolled to the Computer Engineering Master’s Degree in Galatasaray University Institute of Sciences. Since 2008 he has been working at Cardtek as software engineer in Mobile Payment.

Paper titled "Classification of Cohesin Family Using Class Specific Motifs" of this thesis has been accepted as a full paper in HIBIT 2013.