Protein Classification in a Machine Learning Framework

Ph.D. thesis

Attila Kertész-Farkas

Research Group on Artificial Intelligence

Supervisors:
Dr. András Kocsor
Dr. János Csirik

August 2008
Szeged

A thesis submitted for the degree of Ph.D. thesis
of the University of Szeged

University of Szeged
Doctoral School in Mathematics and Computer Science
Ph.D. Programme in Informatics

Preface

At present in most sophisticated pattern recognition tasks humans still outperform computers; however, in certain specific tasks computer models can do better than humans. My scientific interests include building mathematical and algorithmic models in the fields of Bioinformatics demonstrably work well in various real-life applications. One of my aims is to develop methods that can find, recognize and learn regularities and relationships among data. I believe that sophisticated data representation techniques can help us to understand the given data better and allows us to include more knowledge into the mathematical models.

This dissertation was written in this spirit. In general the tasks and models that the author studied were examined with great care. I performed several analyses from different points of view to understand how they behave, and where possible, additional information was included in our models to improve their accuracy. Surely understanding how they work - or do not work - and how they fulfill our intuitive expectations is a vital step in the construction of novel - and hopefully better - technologies in bioinformatics.

Acknowledgement. First of all, I would like to thank my supervisors, Dr. András Kocsor and Prof. János Csirik for supporting my work with useful comments and letting me work at an inspiring department, the Research Group on Artificial Intelligence. I would like to thank all my colleagues and collaborators, namely Sándor Pongor and János Zsigmond Kelemen, for the fruitful discussions and for their constant encouragement pursuing my studies. I am also grateful to all of my co-authors and to the Doctoral School for supporting my work. I would also like to express my gratitude to David P. Curley for scrutinizing and correcting this thesis from a linguistic point of view and to Zsolt Szikora for his encouragement and support.

Last, but not least, my thanks goes to my parents, Zsuzsanna and Jenő, and to my sister, Krisztina, for providing a secure family background during the time spent writing this work. I would like to dedicate this thesis to them.

Attila Kertész-Farkas, August, 2008.

Notations and Abbreviations

$\mathbb{N}$,$\mathbb{R}$,$\mathbb{R}_+$	natural, real and positive real numbers (respectively)
$\|.\|_p$	vector norm, defined by $\|w\|_p = \sqrt[p]{\sum_{i=1}^{n} w_i^p}$ for a $w \in \mathbb{R}^n$, where w_i denotes the ith component of w
$\langle v, w \rangle$	scalar product, defined by $\langle v, w \rangle = \sum_{i=1}^{n} v_i w_i$, where $v, w \in \mathbb{R}^n$
w^T	transpose of a vector $w \in \mathbb{R}^n$
Σ, Γ	alphabets
$\Sigma^\star, \Gamma^\star$	languages generated by alphabets Σ, Γ, respectively
$\mathcal{S}$	set of strings
K	Kolmogorov Complexity
C	text-compressor
κ	kernel function
3PGK	3-Phosphoglycerate Kinase Protein
ANN	Artificial Neural Network
AUC	Area Under Curve
BLAST	Basic Local Alignment Search Tool
CATH	protein domain 3Dimensional database
CBD	Compression-based Distance Measure
COG	Clusters of Orthologous Groups of proteins
DALI	Distance-matrix ALIgnment
KF	Kalman Filter
kNN	k-Nearest Heighbour
LAK	Local Alignment Kernel
LLE	Locally Linear Embedding
LogReg	Logistic Regression
LRA	Likelihood Ratio Approximation
NW	Needleman-Wunsh
PRIDE	Probability of IDEntity
RF	Random Forest
RFE	Recursive Feature Elimination
ROC	Receiver Characteristic Curve
SCOP	Structural Classification of Protein
SVM	Support Vector Machine
SVK	Support Vector Kernel
SW	Smith-Waterman

Contents

This work is dedicated to my parents and my sister.

Chapter 1

Introduction

"In theory, there is no difference
between theory and practice.
But, in practice, there is."
Jan L.A. van de Snepscheut

1.1 Bioinformatics

Today the definition of Bioinformatics is not a clear term and it is difficult to define its border exactly. Loosely speaking, bioinformatics is a marriage of biology, informatics and mathematics and it employs computational tools and methods for managing, analysing and manipulating sets of biological data. This integrated multidisciplinary field includes for example biochemistry, genetics, structural biology, artificial intelligence, machine learning, data mining, information theory, software engineering, statistics, database theory, information visualisation and algorithm complexity and design. Major research efforts in this field include sequence alignment, gene finding, genome assembly, protein structure alignment, protein structure prediction, the prediction of gene expression, protein-protein interactions and the modeling of evolution processes.

One of the main tasks of bioinformatics are the gathering, organization and computational analysis of sequence databases. The classification of sequence data is at the heart of this work, since when sequencing a new genome, perhaps its function and structure are among the most important questions. To determine them, a newly sequenced protein is compared to well-known databases via a similarity function. Then their function and structure can either be inferred from the most similar, well-known protein sequences, or they can be classified into a known protein group by machine learning approaches like Artificial Neural Networks or Support Vector Machines.

1.2 Summary by Chapters

Chapter 2 does not contain any scientific contributions from the Author. This chapter seeks to provide an introduction, contains some basic terms and notations and it also presents problems and challenges in biological sequence classification as well as providing the basis for understanding the main results of this thesis.

In Chapter 3 we give a short description of our protein benchmark databases which were intended to provide standard datasets on which the performance of machine learning and sequence similarity methods could be compared. These are freely available. During the design of these databases we were interested in covering the practical problems of protein classification. Here, we also describe several strategies that we used to construct positive, negative, train and test sets as well as present an experimental comparison along with the classification results obtained by the state-of-the-art machine learning methods and protein similarity measures, whose results can be used as an base line for further comparison studies.

The function and the structure of a newly sequenced protein is usually inferred from the most similar sequences' properties using similarity functions. A good similarity function should rank a positive item higher and should rank a negative item lower for certain protein class. Then the performance of a given similarity method can be evaluated by seeing how it ranks an unseen set of sequences on a particular protein class. In Chapter 4 the Author examined how this ranking ability could be improved by using the likelihood ratio approximation.

Information Distance is a recently developed universal metric for strings. Due to the fact that it is non-computable in the Turing sense, it has to be approximated by text file compressors. Chapter 5 gives an insight into the behaviour of Compression-based Distances (CBDs) in genome sequences. First we will investigate the CBDs from the sequence representation point of view; namely, how the reduced and enlarged alphabets help to distinguish protein classes. We will also examine whether a hybrid combination of CBDs with other fast but problem specific comparison methods really influences the ability to distinguish or classify protein sequences.

Sequence groups are vastly different in terms of most their parameters, and a method that performs well on one group may perform worse on another and vice versa, and very often there are no clear trends in the results. The learning of a similarity in a supervised manner may provide a general framework for adapting a similarity function to a specific sequence class. In Chapter 6 we describe a novel method which learns a similarity function over protein sequences by using a binary classifier and pairs of equivalent sequences (belonging to the same class) as positive samples and non-equivalent sequences (belonging to different classes) as negative training samples.

The content of Chapter 7 differs from the previous chapters. It describes DNA chips (a.k.a. a microarray) that contain gene expression data obtained from healthy and/or diseased tissues. These data items are arranged in a matrix form whose columns represent a tissues and its rows represent genes. Here the task is to identify the smallest set of genes (rows) that best separates the class of tissues (columns); that is, we need to identify those genes that determine the absence or presence of a particular disease.

Knowing these genes more accurate treatment and diagnoses can be applied for a patient. Chapter 7 describes the Kalman Filter (KF) method as a noise-reduction step for DNS chip data. The performance of this method essentially depends on its parameters. Here, we present a new automatic parameter tuning technique which significantly improves the performance of the KF approach. The results we get a more robust disease-state estimator on publicly available binary and multiclass microarray datasets in combination with the most widely used classification methods available.

1.3 Summary by Results

In the following we summarize the results of the Author by arranging them into four distinct thesis points. Table A.1 shows the relation between the thesis points and the publication, where they were presented by the Author.

I Protein benchmark

a The Author participated in building the Protein Classification Benchmark database in order to provide standard datasets on which the performance of the machine learning algorithms and similarity/distance measures could be compared. The number of total classification tasks exceeds 9500. Here the contribution of the Author was the evaluation of the state-of-the-art machine learning techniques on the classification tasks and he provided a parameter set which probably gives the best results as a baseline for newly developed methods [1].

b The Author developed a general mathematical framework for constructing a positive train and test set, which was termed by supervised cross-validation. This technique gives a reliable estimation on how an algorithm will generalize a new distantly related subtype within a known protein class that can also be viewed as a generalization ability of the learned model. He also designed and evaluated the comparative experiments and the resulting datasets provided lower, and in our opinion, more realistic estimates of the classifier performance than those of cross-validation schemes (10-fold or leave- one-out) [2].

The Author examined how depend the classification results on the filtering of the categories from the negative set in order to speed the execution time of the preprocessing and learning method up and to avoid the class-imbalanced problem. The Author designed and evaluated the experiments that led him recommend to misuse it since the resulted negative class may be to specific ant less representative with respect to the entire database. Although this result may be considered as a negative results, in our opinion we should mention it because it makes the characterization of the hierarchically organized protein datasets more complete from classification point of view [2]. Hence when constructing the positive train set, we suggest using

the supervised cross-validation but for the negative set we suggest using the random filtering method [2].

II Likelihood ratio scoring

a The Author suggested the application of a simple likelihood ratio approximation for improving the ranking ability of a protein similarity measure. He designed and evaluated the comparative experiments which justified his view that this likelihood scoring significantly improves the performance of similarity functions [3].

III Compression-based Distances (CBDs)

a The Author examined the behaviour of CBDs on protein classification from several aspects. An analysis of the results showed that the CBDs perform less well than substructure-based comparisons like the outstanding Smith-Waterman algorithm in protein similarity. This is in fact expected, since Smith-Waterman calculations include a substantial amount of biological knowledge encoded in the amino acid substitution matrix while CBDs do not use any apriori information. [4; 5].

The Author examined the efficiency of the CBDs as a function of the size of the alphabet. An alphabet reduction was carried out by grouping the similar types of amino acids an on the alphabet extension was obtained by representing each bi-gram and tri-gram with a new character. The Author designed and evaluated the experiments that did not display, for amino acids or nucleotide sequences, any noticeable relationship between the performance and the size of the alphabet [5]. These results may be regarded as a negative results, but considering them as an observation they could help bioinformatics applications.

b The Author investigated the combination of CBMs with an additional cheap, but problem-specific similarity measure. He designed and evaluated the comparative test which showed that this mixed measure can slightly exceed the performance of the computationally expensive Smith-Waterman and two Hidden Markov Model-based algorithms as well. [4].

IV Equivalence learning

a The Author introduced the notion of equivalence learning as a new way of carrying out similarity learning, and he developed it for protein classification. He designed and evaluated exhaustive experiments and the results show that this novel protein classification technique performed better than the others [6].

b The Author developed a new class of kernel functions, namely the Support Vector Kernel (SVK), He theoretically proved that it is a valid kernel function, and He defined two new ways to learn SVK along with a new parameter setting technique. He designed and evaluated the experiments as well. [7].

V Noise reduction for the microarray

a The contribution of the Author to this task was the design of the experiments and evaluations of the classification and the feature selection methods on microarray datasets. The Author designed an automatic parameter-tuning algorithm for the Kalman Filter as well, which is a common and indivisible result with the first author of [8].

The results presented in the dissertation resulted in several publications. Table 1.1 summarizes which publication covers which item of the thesis points.

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
I	a	b						
II			a					
III				a,b	a			
IV						a	a,b	
V								a

Table 1.1: The relation between the theses and publications.

Chapter 2

Background

DNA molecules encode inheritable information fundamental to the life of the cell. Usually DNA is represented as a long sequence of *nucleotides* that is a long but finite string over the 4 letter "nucleotide" alphabet $\{A, C, T, G\}$. A major discovery of molecular biology was that the DNA that encoded biological information is copied by the RNA and that the RNA-mediated information is used to assemble the proteins. Proteins thus decode biological information into biological function. This flow of information (Fig. 2.1), from DNA to RNA and from RNA to protein, is stated as the *Central Dogma* of molecular biology, and it was first proposed by Francis Crick in 1957 [9].

Protein is a linear sequence of amino acids that is coded by a triplet of nucleotides. We should mention here that the number of triplets of nucleotides is $4^3 = 64$, but there are only 20 amino acids and usually more than one triplet stands for one amino acid. Each protein is uniquely determined by the order of its own amino acids. The set of amino acid symbols is $\Sigma = \{a, r, n, d, c, e, q, g, h, l, i, k, m, f, p, s, t, w, y, v\}$ and thus a protein can be represented by a finite string over the alphabet Σ, which is often called the *primary structure* of the protein. We will use the term *residue* for amino acid or nucleic acid. The method that determines the residue order in a protein or genome is called *sequencing* [10].

The 3-dimensional (3D) structure of a protein, called the *secondary structure*, is evolved by folding the chain of its own amino acids, called *protein folding*, in such a

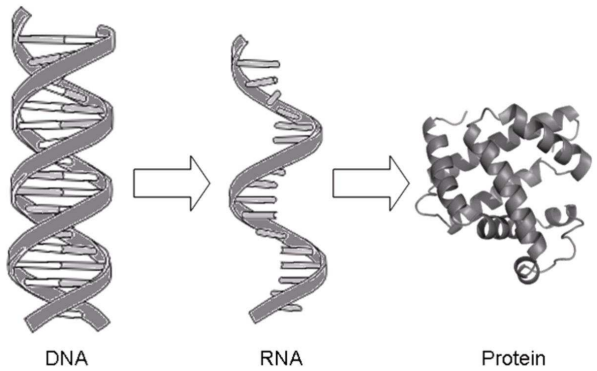


Figure 2.1: The Central Dogma of molecular biology. First the information in DNA is copied to produce an RNA and then this RNA chain is used to synthesise a protein.

way that the minimal energy level is achieved. The *function of a protein* is uniquely determined by its structure. Around 90% of the protein structures available in the Protein Data Bank have been determined by X-ray crystallography. This method allows one to measure the 3D density distribution of electrons in the protein (in the crystallized state) and thereby infer the 3D coordinates of all the atoms to be determined up to a certain resolution. Roughly 9% of the known protein structures have been obtained by Nuclear Magnetic Resonance techniques, which can also be used to determine the secondary structure. Note that aspects of the secondary structure as whole can be determined via other biochemical techniques such as circular dichroism. The secondary structure can also be predicted to a high degree of accuracy with techniques like Cryo-electron microscopy. [10]

Nucleotide sequence:
```
cgt att aat act gtt cgt ggt cct att act att tct gaa ...
```

Amino acid sequence:
```
rintvrgpit iseagftlth ehicgssagf lrawpeffgs ...
```

3D structural representation of a protein:
```
ATOM 1 N  ARG A 36 42.272 77.836 -5.288 1.00 87.57 N
ATOM 2 CA ARG A 36 41.797 76.649 -5.995 1.00 75.63 C
ATOM 3 C  ARG A 36 41.716 75.388 -5.141 1.00 41.28 C
ATOM 4 O  ARG A 36 42.659 74.996 -4.455 1.00 40.52 O
⋮
```

Figure 2.2: Three different representations of the first part of a protein named `1pta` that was sequenced from Bacteria Brevundimonas Divinuta.

Unfortunately the direct determination of the 3D structures is costly, time consuming and difficult as it involves performing unstable chemical and biological experiments. The generation of a protein sequence is much simpler than that of a protein structure. However, the structure of a protein provides much more insight into the function of the protein than its sequence. Between 35-50% of the proteins in sequenced genomes have no assigned functionality and their role in the cell is unknown. Thus a number of heuristic methods for the computational prediction of a protein structure from its sequence have been proposed. The fact that a sequence of amino acids used in biological processes actually takes on a reproducible three-dimensional structure in a relatively short time of seconds or less is one of the marvels of nature. Consider the paradox noted by Levinthal, which asks the reader to consider that if, say, we allow three possible conformations for each amino acid residue on the peptide chain, a length of 100 amino acids protein would have $3^{100} = 10^{47}$ configurations. The computational effort required to study protein folding is enormous, therefore. Using crude workload estimates for a petaflop/second capacity machine, we find that it would require three years to simulate 100 microseconds. [11]

In December 1999, IBM announced the start of a five-year Blue Gene project at IBM Research (after the successful Deep Blue chess project) whose aim was to build

a massively parallel supercomputer, to be applied to the study of biomolecular phenomena such as protein folding. The project had two main goals: (i) to advance our understanding of the mechanisms behind protein folding via large-scale simulation, and (ii) to explore novel ideas in massively parallel machine architecture and software. [12]

The 3D structure prediction *ab initio* methods use just the raw sequence of the protein. The method called *Threading* threads the target sequence through the backbone structures of a collection of template proteins (known as the fold library) and a "goodness of fit" score is calculated for each sequence-structure alignment. *Homology Modeling* seeks to build a reliable 3D model for a protein of unknown structure from one or more related proteins of known structure. *Rosetta@home* is a distributed computing project which tries to predict the structures of proteins with massive sampling on thousands of home computers. *Foldit* is a video game designed to use human pattern recognition and puzzle solving abilities to improve existing software. *PROF* is a tool for the secondary structure prediction program that uses an Artificial Neural Networks to learn secondary structure from known structures and the three-quarters of PROF's prediction are correct [11].

Several methods have been developed for the classification of proteins via their 3D structure. These seek to classify the data in the Protein Data Bank in a structured way. Several databases now exist which classify proteins using different methods. SCOP (see Fig 2.3), CATH and FSSP are the largest ones. The methods used are purely manual, manual and automated, or purely automated. Work is now being done to better integrate the current data. The resulting classification is consistent between SCOP, CATH and FSSP for the majority of proteins which have been classified, but there are still some differences and inconsistencies. [13]

A *protein domain* is an 'independent' functional unit of protein sequence and structure, and it exists independently of the rest of the protein sequence. Each domain has a compact three-dimensional structure, and it often can be independently stable and folded. Many proteins consist of several structural domains and one domain may appear in a variety of evolutionarily related proteins. Domains vary in length from between about 25 amino acids up to 500 amino acids, but it is difficult to exactly define the border of a domain. [14]

The word protein comes from the Greek word $\pi\rho\omega\tau\alpha$ ("prota"), meaning "of primary importance". Proteins were first described and named by the Swedish chemist Jöns Jakob Berzelius in 1838. However, the central role of proteins in living organisms was not fully appreciated until 1926, when James B. Sumner showed that the enzyme urease was a protein. The first protein to be sequenced was insulin, and it led to Frederick Sanger receiving the Nobel Prize for this in 1958. The first protein structures to be solved included hemoglobin and myoglobin, by Max Perutz and Sir John Cowdery Kendrew, respectively, in 1958. The three-dimensional structures of both proteins were first determined by X-ray diffraction analysis; Perutz and Kendrew shared the 1962 Nobel Prize in Chemistry for their works. [10]

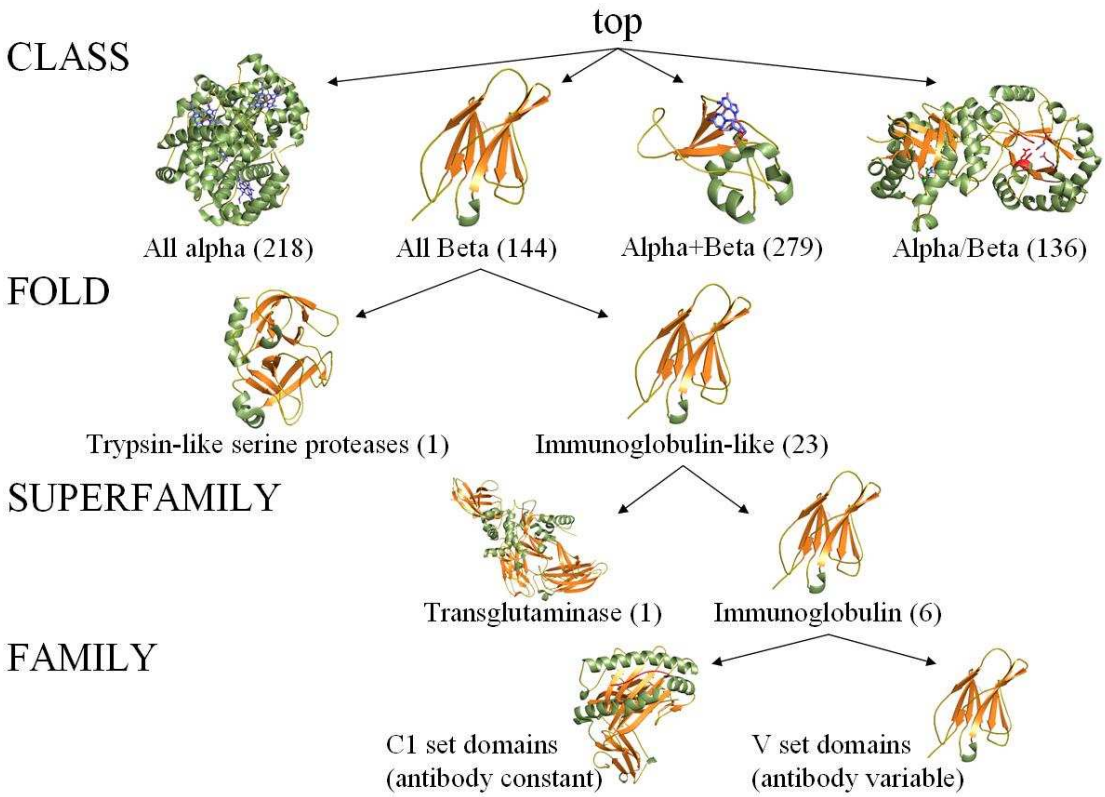


Figure 2.3: Hierarchical classification of proteins by their structure in the SCOP database.

2.1 Protein Similarity

Similarity is a highly intuitive concept and its use in various fields is quite different. In the 18th century Linnaeus, a Swedish naturalist, classified living things according to a hierarchy: Kingdom, Phylum, Class, Order, Family, Genus, Species, and his classification was based on an observed similarity and widely reflects biological ancestry. The characteristics derived from a common ancestor are called *homologous*. For example the eagle's wing and the human arm are homologous. Other apparently similar characteristics may have arisen independently by convergent evolution. For instance, the common ancestor of the eagle and bee did not have wings, so they are not homologous. [15].

Proteins that are derived from a common ancestor are called *homologous*. Homologous sequences are called *orthologous* if they were separated by a speciation event. That is, when a species diverges into two separate species, the divergent copies of a single gene in the resulting species are said to be *orthologous*. Orthologs, or orthologous proteins, are proteins in different species that are similar to each other because they originated from a common ancestor. Homologous sequences are called *paralogous* if they were separated by a gene duplication event; that is, when a gene in an organism is duplicated to occupy two different positions in the same genome, then the two copies are paralogous. A set of sequences that are paralogous are called paralogs of each other. Paralogs typically have the same or similar function, but sometimes they do not: due to the lack of original selective pressure upon one copy of the duplicated gene, this

copy is free to mutate and can acquire new functions. [15]

Sequence analysis provides unambiguous evidence for the relationship of species. For higher organisms, sequence analysis and the classical tools of comparative anatomy, palaeontology and embryology are often consistent [16].

A sequence similarity method returns a higher value on 'similar' sequences, and a lower value on 'different' ones. We can define in an analogous way a (sequence) distance (dissimilarity) function which gives a zero or minimal value on equal or 'similar' sequences, and a high value on 'different' sequences. Since any dissimilarity function can be transformed into a similarity function by a monotone decreasing function, and we will also use the term *proximity* measure (or function) for a similarity or distance measure.

Sequence similarity analysis is the measurement procedure used to infer homology. In general if an unknown sequence is found, its function and structure can be deduced indirectly by finding similar sequences whose features are known. Given two or more sequences, we wish to (i) measure their similarity, (ii) determine the residue-residue correspondences, (iii) observe patterns of conservation and variability and (iv) infer evolutionary relationships. [16]

Usually sequence similarity is carried out by sequence alignment, which is a way of arranging the primary sequences of nucleotides or amino acids so as to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Aligned sequences are typically represented as rows within a matrix. Gaps are inserted between the residues so that residues with identical or similar characters are aligned in successive columns. The exact mathematical definition of sequence alignment can be formulated as

Definition 2.1 *A* global alignment *of two strings s, t over the same alphabet Σ is a string $[st]$ over the alphabet*

$$\Gamma = \left\{ \{\Sigma \cup -\}^2 \setminus (-,-) \right\} \tag{2.1}$$

such that

$$l([st]) = s \quad \textit{and} \quad r([st]) = t, \tag{2.2}$$

where the symbol '$-$' is called blank (or space) and $- \notin \Sigma$. The function $l : \Gamma^\star \to \Sigma^\star$ is a projection function that is defined for a string $a = a_1 \dots a_n$ ($a_i \in \Gamma$) by $l(a) = l'(a_1) \dots l'(a_n)$ (the concatenation of $l'(a_i)$'s) , where $l' : \Gamma \to \Sigma$ stands for

$$l'(u,v) = \begin{cases} \lambda & \textit{if } x = - \\ u & \textit{otherwise}, \end{cases}$$

where λ denotes the empty string. The function r is defined in a similar way, but it returns the second letter v.

Here Eq. 2.1 ensures that only one symbol from either string is aligned to only one symbol from the other string or to a gap, but two blanks cannot be aligned. Eq. 2.2

ensures that all symbols in both sequences have to be aligned in the same order as they appear in s and t.

Let $[st]$ be an alignment for strings s and t. The i^{th} letter $[st]_i \in \Gamma$ in the alignment $[st]$ is called a *match* if $[st]_i = (a, a)$ and called a *substitution* if $[st]_i = (a, b)$, $(a \neq b, a, b \in \Sigma)$. Now let Δ be the set of the indices of matches and substitutions. The letter *types of* $(a, -) \in \Gamma$ *and* $(-, a) \in \Gamma$ (respectively) are called insdel (that is, insertion or deletion) and a sequence of insdels $g_{ij} = a_i \dots a_j$ is called the *gap* of length $|g_{ij}| = j - i + 1$ when a_k $(i \leq k \leq j)$ are insdels of the same type, but a_{i-1} and a_{j+1} are either insdel of the other type or not insdels. Let Λ be the set of gaps.

For example, let `x=tcctgcctctgccatca` and `y=tcgtgcatctgcaatcatg` be two nucleotide sequences. One possible alignment of the two sequences x and y is:

```
tcctgcctctgc---catca
||:|||:|||||   |||:
tcgtgcatctgcaatcatg-
```

And another possible alignment of x and y is:

```
tcctgcctctgccatca--
||:|||:|||||:||||
tcgtgcatctgcaatcatg
```

Here the vertical lines between the letters denote the match while colons represent substitutions. The absence of the former two marks denotes an insdel.

But which is the better alignment? The answer is depend on how the matches, substitutions and insdels are measured. The cost of an alignment can be calculated by the following general formula:

$$cost([st]) = \sum_{i \in \Delta : [st]_i = (a,b)} c(a, b) - \sum_{g \in \Lambda} p(|g|),$$

where c is the cost function for the match and for the substitution, which is usually given by a matrix with size $|\Sigma| \times |\Sigma|$. In bioinformatics the most popular substitution matrices are the BLOSUM [17] and PAM [18]. The gap penalty function $p : \mathbb{N} \to \mathbb{R}$ is an affine penalty function that is given by $p(n) = g_o \cdot n + g_e$, where g_o is called the cost of *gap open*, g_e is called the cost of *gap extension*, and $p(0) = 0$.

If two sequences in an alignment are derived from a common ancestor, mismatches can be interpreted as point mutations and gaps as insdels (that is, insertion or deletion mutations) introduced in one or both lineages in the time since they diverged from one another. In protein sequence alignment, the degree of similarity between amino acids occupying a certain position in the sequence can be interpreted as a rough measure of how conserved a particular region or sequence motif is among lineages. The absence of substitutions, or the presence of only very conservative substitutions (that is, the substitution of amino acids whose side chains have similar biochemical properties) in a particular region of the sequence, suggest that this region has structural or functional importance. Although DNA nucleotide bases are more similar to each other than

to amino acids, the conservation of base pairing can indicate a similar functional or structural role.

Very short or very similar sequences can be aligned by hand; however, most interesting problems require the alignment of lengthy, highly variable or extremely numerous sequences that cannot be aligned solely by human effort. Instead, human knowledge is primarily applied in constructing algorithms to produce high-quality sequence alignments, and occasionally in adjusting the final results to reflect patterns that are difficult to represent algorithmically (especially in the case of nucleotide sequences). Computational approaches to sequence alignment generally fall into two categories: global alignments and local alignments. Calculating a global alignment is a form of global optimization that "forces" the alignment to span the entire length of all query sequences. In contrast, local alignments identify regions of similarity within long sequences that are often widely divergent overall. Local alignments are often preferable, but can be more difficult to calculate because of the additional challenge of identifying the regions of similarity. A variety of computational algorithms have been applied to the sequence alignment problem, including slow but formally optimizing methods like dynamic programming and efficient heuristic or probabilistic methods designed for large-scale database searches. [16]

The Needleman-Wunsch (NW) approach [19] was the first application of dynamic programming to biological sequence comparison, and it looks for the best global alignment of two strings s and t with length m and n (respectively). Next we will give the pseudo code of an extension of the NW method to a gap penalty function p [20]:

$$
\begin{aligned}
L[i,0] &= p(i) \qquad (0 \le i \le n) \\
L[0,j] &= p(j) \qquad (0 \le j \le m) \\
L[i,j] &= \max \begin{cases} \max_{1 \le k \le i} \left(L[i-k,j] - p(k) \right) \\ \max_{1 \le k \le j} \left(L[i,j-k] - p(k) \right) \\ L[i-1,j-1] + c(a,b) \end{cases} \qquad \begin{matrix} (1 \le i \le n) \\ (1 \le j \le m) \end{matrix}
\end{aligned}
$$

Here the function c is the cost function. Then the total cost of the best global alignment is $L[n,m]$ and the corresponding alignments are paths from $L[0,0]$ to $L[n,m]$.

Definition 2.2 *A* local alignment *of two strings s, t over the same alphabet Σ is the same as the global alignment case, but Eq. 2.2 is replaced by following constraint:*

$$l([st]) \sqsubseteq s \text{ and } r([st]) \sqsubseteq t, \tag{2.3}$$

where $u \sqsubseteq v$ means u is substring of v (that is, $u \sqsubseteq v \iff \exists x, y \in \Sigma^\star : v = xuy$).

The best local alignment for two strings s and t of length m and n, respectively, can be calculated by dynamic programming using a matrix L of size $(m+1) \times (n+1)$ by the following pseudo code known as Smith-Waterman [21].

$$
\begin{aligned}
L[i,0] &= 0 \qquad (0 \le i \le n) \\
L[0,j] &= 0 \qquad (0 \le j \le m) \\
L[i,j] &= \max \begin{cases} \max_{1\le k\le i} \left(L[i-k,j] - p(k)\right) & \\ \max_{1\le k\le j} \left(L[i,j-k] - p(k)\right) & (1 \le i \le n) \\ L[i-1,j-1] + c(a,b) & (1 \le j \le m) \\ 0 & \end{cases}
\end{aligned}
$$

Here p stands for the gap penalty function and c represents the cost function. The cost of the best alignment is the maximum value in the matrix L and the corresponding alignments are paths to this maximal value in L from a cell with zero value.

A kernel function can be regarded as a similarity function which has the additional property of always being positive semi-definite (see [22]). This is a simple way of extending the well-tried linear, vector and scalar product-based applications to a non-linear model while preserving their computational advantages [23] and it can be directly applied to non-vectorial data like strings, trees and graphs. Over the past decade, many kernels have been developed for sequences such as the String Kernel [24], Mismatch Kernel [25], Spectrum Kernel [26], Local Alignment Kernel [27] and the Fisher Kernel [28]. For an extensive review of their applications, see [23].

Now we will give a short introduction to kernel functions taken from [22]. Here let $\mathcal{X}$ be a nonempty set.

Definition 2.3 *A symmetric function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a* positive definite kernel *on $\mathcal{X}$ if*

$$\sum_{i,j=1}^{n} c_i c_j \kappa(x_i, x_j) \ge 0 \tag{2.4}$$

holds for any $n \in \mathbb{N}$, $x_1, \dots x_n \in \mathcal{X}$ and $c_1, \dots c_n \in \mathbb{R}$. Let $\mathcal{K}(\mathcal{X})$ be the class of kernel functions over the space $\mathcal{X}$.

Proposition 2.1 *$\mathcal{K}(\mathcal{X})$ is closed under addition, multiplication, positive scalar addition and positive scalar multiplication, i.e. for $\kappa_1, \kappa_2 \in \mathcal{K}(\mathcal{X})$ and $\lambda \in \mathbb{R}_+$ the following functions are in $\mathcal{K}(\mathcal{X})$:*

i) $\kappa(x,z) = \kappa_1(x,z) + \kappa_2(x,z)$,
ii) $\kappa(x,z) = \kappa_1(x,z)\kappa_2(x,z)$,
iii) $\kappa(x,z) = \kappa_1(x,z) + \lambda$,
iv) $\kappa(x,z) = \lambda\kappa_1(x,z)$.

Proposition 2.2 *Kernels are also closed under tensor product and direct sum, i.e. for $\kappa_1 \in \mathcal{K}(\mathcal{X})$, $\kappa_2 \in \mathcal{K}(\mathcal{Y})$ the following functions are in $\mathcal{K}(\mathcal{X} \times \mathcal{Y})$:*

i $\kappa_1(x,y) \otimes \kappa_2(u,v) = \kappa_1(x,y)\kappa_2(u,v)$,
ii $\kappa_1(x,y) \oplus \kappa_2(u,v) = \kappa_1(x,y) + \kappa_2(u,v)$.

Proposition 2.3 *If $f : \mathcal{X} \to \mathbb{R}$ is an arbitrary function, then $\kappa(x, z) = f(x)f(z)$ is a positive definite kernel function.*

Definition 2.4 *A symmetric function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a* conditionally negative definite kernel *on $\mathcal{X}$ if*

$$\sum_{i,j=1}^{n} c_i c_j \kappa(x_i, x_j) \leq 0 \tag{2.5}$$

holds for any $n \in \mathbb{N}$, $x_1, \ldots x_n \in \mathcal{X}$ and $c_1, \ldots c_n \in \mathbb{R}$, $\sum_{i=1}^{n} c_i = 0$.

An important connection between the positive definite and conditionally negative definite kernels is stated in the following

Proposition 2.4 *A function τ is conditionally negative definite iff $\exp(-\tau)$ is positive definite.*

On the vector space $\mathbb{R}^n$ the simple scalar product is a positive kernel function. For a positive-definite matrix Θ, the weighted inner product $\langle x, y \rangle_\Theta = x'\Theta y$ is also a positive definite kernel function. Possibly one of the most well known and widely used non-linear kernel functions is the Gaussian Radial Basis Function (RBF), which is defined by

$$\kappa(x, y) = \exp(-\sigma \|x - y\|_2^2), \tag{2.6}$$

where $\sigma \in \mathbb{R}_+$ is the so-called width parameter. We should add that $\|x - y\|_2^2$ is conditionally negative definite.

Now we will present the Local Alignment Kernel (LAK) which can be considered as a kernel version of SW [27].

Definition 2.5 *Suppose $\mathcal{S}$ is a set of strings and $\kappa_d \in \mathcal{K}(\mathcal{S})$ $(1 \leq d \leq D)$. The* convolution kernel of $D \in \mathbb{N}_+$ kernels $\kappa_1, \ldots, \kappa_D$ *is*

$$\kappa_1 \star \cdots \star \kappa_D(x, y) = \sum_{x = x_1 \ldots x_D, y = y_1 \ldots y_D} \kappa_1(x_1, y_1) \cdots \kappa_D(x_D, y_D) \in \mathcal{K}(\mathcal{S}). \tag{2.7}$$

The kernel function $\kappa_1 \star \cdots \star \kappa_D$ is a positive definite kernel function because it is the tensor product of the kernels $\kappa_1, \ldots, \kappa_D$.

Convolution kernels were introduced in [29] for general discrete structures such as trees. Let $\kappa_0(s, t) = 1$ be a trivial kernel that is always equal to 1 and let us define the following three kernels as well:

$$\begin{aligned}
\kappa_a^\beta(s, t) &= \begin{cases} \exp(\beta c(s, t)) & \text{if } |s| = 1 \text{ and } |t| = 1, \\ 0 & \text{otherwise,} \end{cases} \\
\kappa_p^\beta(s, t) &= \exp(\beta(p(|s|) + p(|t|))) \\
\kappa_n^\beta &= \kappa_0 \star \left(\kappa_a^\beta \star \kappa_p^\beta\right)^{(n-1)} \star \kappa_a^\beta \star \kappa_0,
\end{aligned}$$

where c is the cost of the substitution and p is an affine penalty function for gaps. Then LAK with parameter β is defined by

$$\kappa_{LAK}^{\beta} = \sum_{i=0}^{\infty} \kappa_i^{\beta}. \tag{2.8}$$

The LAK procedure sums up the contributions of all the possible local alignments instead of keeping just the best ones, as SW does. The precise connection between the SW score and the LAK score is stated in

Proposition 2.5 *The SW score is related to the LA kernel by the equality*

$$\lim_{\beta \to +\infty} \frac{1}{\beta} \ln \kappa_{LA}^{\beta}(s,t) = SW(s,t).$$

2.2 Machine Learning and Pattern Classification

The major focus of machine learning research is to extract information from data automatically, by computational and statistical methods. Machine learning is closely related not only to data mining and statistics, but also theoretical computer science [30]. Machine learning has a wide spectrum of applications including natural language processing, syntactic pattern recognition, search engines, medical diagnosis, brain-machine interfaces and cheminformatics, detecting credit card fraud, stock market analysis, speech and handwriting recognition, object recognition in computer vision, game playing and robot locomotion [30].

Machine learning algorithms are organized into a taxonomy, based on the algorithm types [31]:

- Supervised learning technique attempt to learn a function from training data. The training data consist of pairs of input objects (typically vectors), and desired outputs. The output of the function can be a continuous value (called regression), or can predict a class label of the input object (called classification). The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples (i.e. pairs of input and target output). To achieve this, the learner has to generalize from the presented data to unseen samples.

- In unsupervised learning approach the manual labels of inputs are not used. One form of unsupervised learning is clustering.

- Reinforcement learning is a sub-area of machine learning concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states.

Pattern recognition is a sub-topic of machine learning and aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space. Most research in pattern recognition is about methods for supervised learning and unsupervised learning [32].

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield absolute guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common [31].

In simple validation the labeled samples are split into two parts: one is used to adjust the parameter of the learner algorithm, called train set, and the other set – the validation set or test set– is used to estimate the generalization error. A simple generalization of the above method is m-*fold cross validation*. Here the training set is randomly divided int m disjoint sets of equal size m/n, where n is the total number of patterns in the dataset. The classifier is trained m times, each time with a different set held out as a test set. The estimated performance is the mean of these m errors. In the limit where $m = n$, the method is called the *leave-one-out* approach. For more details about validation techniques the reader should read [31].

2.3 Protein Classification

The classification of proteins is a fundamental task in genome research. When a new genome is sequenced perhaps the first key question is its structure and function. In order to predict them the newly sequenced protein is compared to well-known databases via a similarity function and then their function and structure can be inferred from the most similar, well-known protein sequence groups [33].

When the 3D structure of a protein is known, structure similarity is a very good indicator of protein homology. Currently one of the most popular 3D structural similarity methods is the DALI (Distance-matrix ALIgnment) method [34], which computes the best alignment of 3D coordinates of the atoms in two proteins [33; 34].

Since 3D structure determination experiments are costly and difficult, usually just the amino acid sequence is determined and the structure and function are inferred via sequence similarity (or alignment) methods. Unfortunately, the sequence similarity methods do not take account of factors in 3D i.e. they do not use information about structure, hydrophobicity and distance constraints affecting insdels. The relationship between the similarity score for a sequence and for a 3D structure is shown in Fig. 2.4. For example, if the sequence identity is higher than 50%-70% then a high structure similarity is related that infers an almost certain homology. If the sequence similarity falls in the range 5%-30% then the corresponding structure similarity can be either high or low [35].

The early methods of protein classification relied on the pairwise comparison of se-

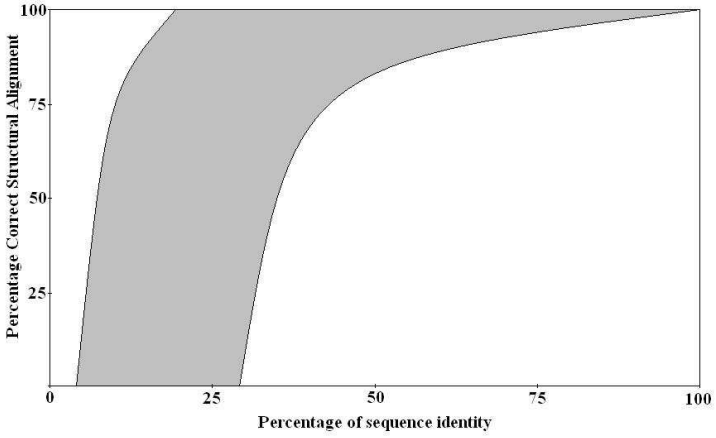


Figure 2.4: The connection between the sequence identity and 3D structural similarity.

quences, based on the alignment of sequences, using exhaustive dynamic programming methods [19; 21] or faster, heuristic algorithms [36; 37]. Pairwise comparison yielded a similarity measure that could be used to classify proteins on an empirical basis. The next generation of methods then used generative models for the protein classes and the similarity of a sequence to a class was assessed by a score computed between the model and the sequence. Hidden Markov Models (HMMs) are now routinely used in protein classification [38], but there are many other, simpler types of description in use (for a review, see [39]). Discriminative models (such as Artificial Neural Networks and Support Vector Machines) are used in a third generation of protein classification methods where the goal is to learn the distinction between class members and non-members. Roughly speaking, 80-90% of new protein sequence data can be classified by simple pairwise comparison. The other, more sophisticated techniques are used mostly to verify whether a new sequence is a novel example of an existing class or whether it represents a truly new class in itself. As the latter decisions refer to the biological novelty of the data, there is a growing interest in new, improved classification methods [40].

2.4 Performance Evaluation and Receiver Operating Characteristics (ROC) Analysis

Receiver Operator Characteristics (ROC) analysis is a visual as well as numerical method used for assessing the performance of classification algorithms, such as those used for predicting structures and functions from sequence data. This section summarises the fundamental concepts of ROC analysis and the interpretation of results using examples of sequence and structure comparison, and it is mostly based on [41].

Originally developed for radar applications in the 1940s, ROC analysis became widely used in medical diagnostics, where complex and weak signals needed to be distinguished from a noisy background [42]. Subsequently, it gained popularity in machine learning and data mining [43; 44]. Fawcett provides a good general introduction to the subject

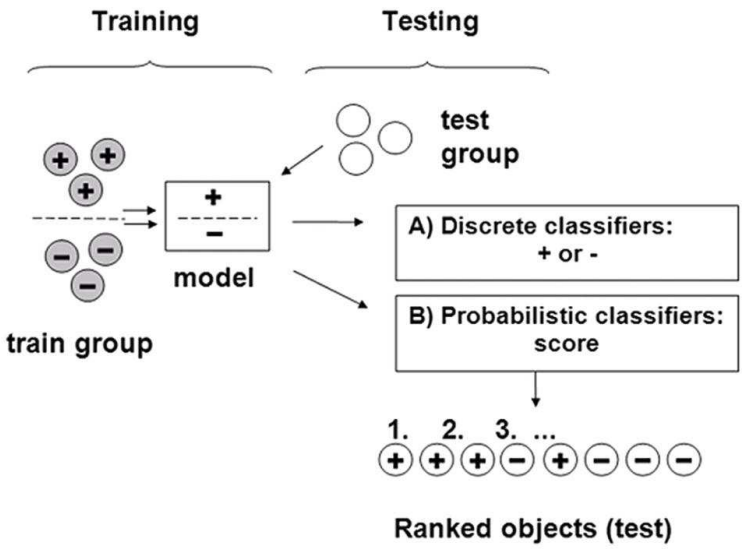
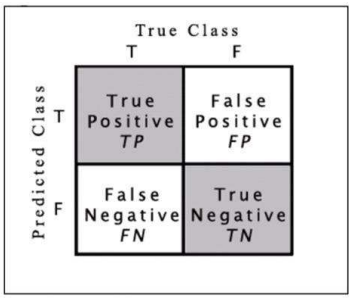


Figure 2.5: Binary classification. Binary classifiers algorithms (models, classifiers) capable of distinguishing two classes are denoted by + and -. The parameters of the model are determined from known + and - examples, this is the training phase. In the testing phase, test examples are shown to the predictor. Discrete classifiers can assign only labels (+ or -) to the test examples. Probabilistic classifiers assign a continuous score to the text examples, which can then be used for ranking [41].

[45]. Applications to bioinformatics were fostered by the seminal paper of Gribskov and Robinson [46]. The current popularity of ROC analysis in bioinformatics may be due to the visibly increasing use of machine learning techniques in computational genomics. Due to this sequence of events, current bioinformatics applications of ROC analysis use concepts and approaches taken from a variety of fields. This chapter seeks to provide an overview of ROC analysis, applied to molecular biology data.

The fundamental use of ROC analysis is its application to binary (or two-class) classification problems. A binary classifier algorithm maps an object (such as an unannotated sequence of 3D structure) into one of two classes, which we usually denote by '+' and '-'. Generally, the parameters of such a classifier algorithm are derived from training on known '+' and '-' examples, then the classifier is tested on '+' and '-' examples that were not part of the training sets (see Fig. 2.5) [31].

A discrete classifier predicts only the classes to which a test object belongs. There are four possible outcomes: true positive, true negative, false positive, false negative [31], which are schematically shown in Fig. 2.6. If an object is positive and it is classified as positive, it is counted as a true positive (TP); if it is classified as negative, it is counted as a false negative (FN). If the object is negative and it is classified as negative, it is counted as a true negative (TN); if it is classified as positive, it is counted as a false positive (FP). If we evaluate a set of objects, we can count the outcomes and prepare a confusion matrix (also known as a contingency table), a two-by-two table that shows the classifier's correct decisions on the main diagonal and the errors off this diagonal (see Fig 2.6, on the left). Alternatively, we can construct various numerical measures that characterise the accuracy, sensitivity and specificity of the test (Fig. 2.6 on the right). These quantities have values that lie between 0 and 1 and can be interpreted as probabilities. For instance, the false positive rate is the probability that



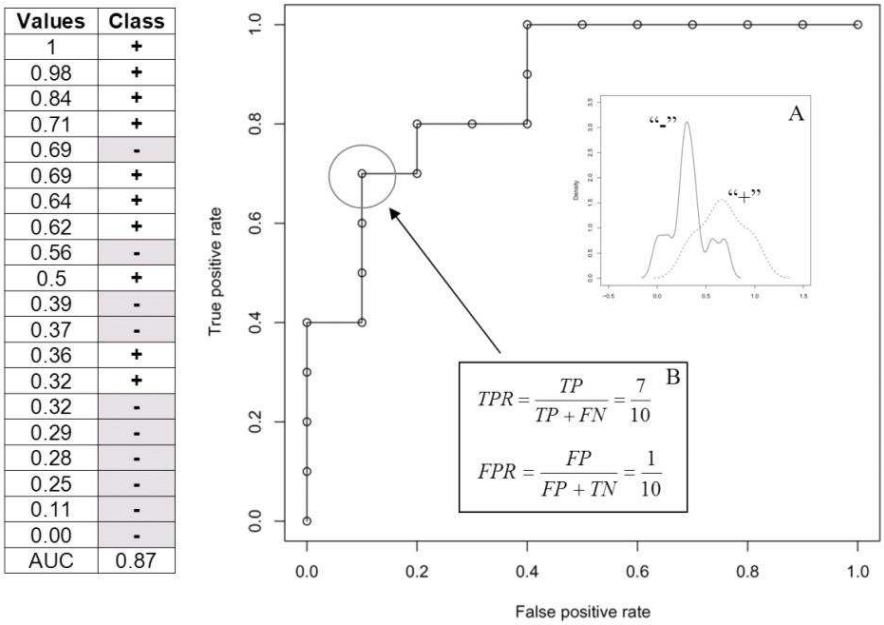
$$ACC = accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$SP = specificity = \frac{TN}{FP + TN}$$

$$TPR = sensitivity = \frac{TP}{TP + FN}$$

$$FPR = (1 - specificity) = \frac{FP}{FP + TN}$$

Figure 2.6: The confusion matrix and a few performance measures. TP, TN, FP, FN are the number of true positives, true negatives, false positives and false negatives in a test set, respectively. TPR is the true positive rate or sensitivity, FPR is the false positive rate. A ROC curve is a TPR vs. FPR plot [41].



Values	Class
1	+
0.98	+
0.84	+
0.71	+
0.69	-
0.69	+
0.64	+
0.62	+
0.56	-
0.5	+
0.39	-
0.37	-
0.36	+
0.32	+
0.32	-
0.29	-
0.28	-
0.25	-
0.11	-
0.00	-
AUC	0.87

Figure 2.7: Constructing a ROC curve from ranked data. The TP,TN, FP, FN values are determined by comparing them to a moving threshold, an example being shown by an arrow in the ranked list (left). Above the threshold + data items are TP, - data items are FP. Thus a threshold of 0.6 produces the point FPR=0.1, TPR=0.7 as shown in inset B. The plot is produced by moving the threshold through the entire range. The data items were randomly generated based on the distributions shown in inset A [41].

a negative instance is incorrectly classified as being positive. Many similar indices are reviewed in [47; 48].

Probabilistic classifiers, on the other hand, return a score that is not necessarily a sensu stricto probability but represents the degree to which an object is a member of one particular class rather than another [49]. We can use this score to rank a test set of objects, and a classifier works correctly if the positive examples are at the top of the list. In addition, one can apply a decision threshold value to the score, like a value above where the prediction is considered positive. Doing this, we can change the probabilistic classifier into a discrete classifier. Naturally, we can select different threshold values, and, in this way, we can generate a (infinitely long) series of discrete classifiers for one probabilistic classifier.

Ranks	a	b	c	d	e	f	g	h
1	+	+	-	-	-	-	-	-
2	+	+	+	-	-	-	-	-
3	+	+	+	+	-	-	-	-
4	+	+	+	+	-	-	-	-
5	-	-	+	+	-	-	-	-
6	+	-	-	+	+	-	-	-
7	+	-	-	-	+	-	-	-
8	+	-	-	-	+	-	-	-
9	-	-	-	-	+	+	-	-
10	+	-	-	-	-	+	-	-
11	-	-	-	-	-	+	-	-
12	-	-	-	-	-	+	+	-
13	+	-	-	-	-	-	+	-
14	+	-	-	-	-	-	+	-
15	-	-	-	-	-	-	+	-
16	-	-	-	-	-	-	-	-
17	-	-	-	-	-	-	-	+
18	-	-	-	-	-	-	-	+
19	-	-	-	-	-	-	-	+
20	-	-	-	-	-	-	-	+
AUC:	0.87	1.00	0.93	0.88	0.70	0.50	0.30	0.00

Figure 2.8: Examples of Area Under the Curve (AUC) values for synthetic data. The numerical data are the same as those used in Fig. 2.5 (and shown here in column 'a'). The data in columns 'b'-'h' were generated by assigning the +/- values in the indicated manner. [41]

A ROC curve (Fig. 2.7) is obtained by selecting a series of thresholds and plotting sensitivity on the y-axis versus 1-specificity on the x-axis. Using the abbreviations of Fig. 2.6, this is a TPR (true positive rate) vs. FPR (false positive rate) plot. The output of our imaginary classifier is the ranked list shown on the left hand side of Fig. 2.7. We can produce the ROC curve shown at the bottom left of the figure by varying the decision threshold between the minimum and maximum of the output values and plotting the FPR (1 - specificity) on the x-axis and the TPR (sensitivity) on the y-axis. (In practice, we can change the threshold so as to generate the next output value; doing this, we can create one point for each output value). The empirical ROC curve generated for this small test set is a step function, and it will approach a continuous curve for large test sets.

Each point on this curve corresponds to a discrete classifier that can be found using a given decision threshold. For example, when the threshold is set to 0.6, the TPR is 0.7, and the FPR is 0.1. A ROC curve is thus a two-dimensional graph that visually depicts the relative trade-offs between the errors (false positives) and benefits (true positives) [49]. We can also say that a ROC curve characterizes a probabilistic classifier, and that each point of this curve corresponds to a discrete classifier.

A perfect probabilistic classifier has an rectangular shape and its integral - the "area under the ROC curve" (AUC) - is equal to 1. A random classifier that returns random answers irrespective of the input is approximately a diagonal line and the integral of this curve is 0.5. A correct classifier has a ROC curve above the diagonal and an AUC $>\sim$ 0.5. On the other hand, classifiers that consistently give the opposite predictions, ("anticorrelated" classifiers) give ROC curves below the diagonal and AUC values between zero and 0.5.

From a mathematical point of view, AUC can be viewed as the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance, i.e. it is equivalent to the two sample Wilcoxon rank-sum statistic [50]. Alternatively, AUC can also be interpreted either as the average sensitivity over all false

positive rates or as the average specificity over all sensitivities.

In practice, AUC is often used as a single numerical measure of ranking performance. We note that ranking is dependent on the call distribution of the ranked set, so one cannot set an absolute threshold above which the ranking is considered good. In general, a high AUC value does not guarantee that the top ranking items will be true positives, as is apparent from the synthetic data in Fig 2.8.

When training and evaluating binary classifiers, having the same number of positive and negative examples is recommended [31; 47]. This condition is practically never met in bioinformatics databases, however, because we have far fewer positives than negatives and the ratio of the negative and positive samples may significantly differ from 1.0. Fortunately, ROC analysis is insensitive to the imbalanced class problem. When the objects in either class are increased (which have the same distribution) or resampled e.g. by bootstrapping the AUC value will be unchanged [49], hence AUC values got from databases with different ratios can be compared.

2.5 Visualization Methods

Visualization is an important topic in the analysis of high-dimensional measurements, principally because it facilitates a better understanding of the data. Here we will summarize three graphical representation methods that we will extensively use during the method analyses.

2.5.1 Heat Map

The *Heat Map* technique can be employed as a graphical representation of the 2D data matrices where the values taken by a feature are represented as a colour intensity on a 2D map. The visualizations are generally performed in Matlab.

2.5.2 Radial Visualization (RadViz)

The *Radial Visualization (RadViz)* approach [51] is another proposed visualization scheme where the features are represented as anchors that are equally spaced around the unit circle. The samples are then represented as points inside this unit circle. Their positions depend on the feature values: the higher the value for a feature, the more the anchor attracts the corresponding point. This method was especially developed for the visualization of gene expression data (see Chapter 7) with relatively few (3-20) features (i.e. genes), thus requiring a priori feature selection. Here the program that we used in our experiments was our own Matlab implementation.

2.5.3 Locally Linear Embedding (LLE)

The *Locally Linear Embedding (LLE)* technique [52] is a non-linear mapping from a high-dimensional original space to a lower dimensional Euclidean space.

The general problem of embedding a high dimensional vector data $x_1, \dots x_n \in \mathbb{R}^m$ is to produce a matrix $Y = [y^1 \dots y^k]$ of size of $n \times k$ $(k < n)$, where the ith row y_i provides the embedding coordinates of the object x_i. A reasonable goal is to minimize the following objective function

$$\sum_{i,j} \|y_i - y_j\|_2^2 W_{ij} = \mathsf{Tr}(Y^T(D - W)Y), \tag{2.9}$$

where W is a symmetric weight matrix whose element W_{ij} tells us how x_i and x_j are close to each other, and the matrix D is a diagonal matrix such that $D_{ii} = \sum_j W_{ij}$. The matrix $\mathcal{L} = D - W$ is the Laplacian matrix which is a symmetric positive definite matrix that can be thought of as an operator on functions defined on the objects x_is [53]. The minimization problem reduces to that of finding

$$\underset{Y:Y^TDY=I}{\mathsf{argmin}} \ \mathsf{Tr}(Y^T\mathcal{L}Y). \tag{2.10}$$

Standard methods show that the solution is provided by solving the following generalized matrix eigenvalue problem $\mathcal{L}y = \lambda Dy$. Now let $f_0, \dots f_k$ be the solution of equation 2.10, ordered according to their eigenvalues,

$$\begin{aligned} Lf_0 &= \lambda_0 D f_0 \\ Lf_1 &= \lambda_1 D f_1 \\ &\vdots \\ Lf_k &= \lambda_k D f_k \\ 0 &= \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_k \end{aligned}$$

Ignoring the eigenvector f_0 corresponding to the 0 eigenvalue and using the next eigenvectors for embedding in an k-dimensional Euclidean space for the object x_i, we get the following optimal embedding

$$x_i \rightarrow (f_1(i), \dots, f_k(i)). \tag{2.11}$$

The main aim of LLE is to use an approximation matrix A such that A_{ij} best reconstructs each data x_i from its l neighbours (the only one parameter of the method); that is, to minimize the following cost

$$\min \sum_{i=1}^{n} \|x_i - \sum_j A_{ij} x_j\|_2^2$$

with the restriction that $\sum_j A_{ij} = 1$ for each i [54]. The matrix A is not symmetric and the embedding is computed by taking eigenvectors corresponding to the k lowest eigenvalues of the matrix $W = (I - A)^T(I - A)$.

Using this method, the dataset can be mapped into the 2D space, and then conve-

niently plotted on a graph. The resulting two dimensions are abstract, however, and do not correspond to any real variable. The method we used was implemented in Matlab and downloaded from [52], and the number of neighbours parameter l required by the procedure was set to the number of samples.

Part I

Protein Benchmark Collection

Chapter 3

Protein Databases

Protein classification by machine learning algorithms is now widely used in the structural and functional annotation of proteins. Our Protein Classification Benchmark collection was created in order to provide standard datasets which cover most protein classification problems as well as on which the performance of machine learning methods can be compared. It is primarily meant for method developers and users interested in comparing methods under standardized conditions. The Protein Benchmark Collection is available at `http://hydra.icgeb.trieste.it/benchmark`
The Author's contributions to this project were the design of the classification tests (i.e. construction of the positive and negative train and test sets) and evaluation of the most popular machine learning and protein similarity techniques applied on the database. The Author's colleagues (without the Author) were responsible for the protein sequence selection, the design and construction of the web page and database management.

3.1 Introduction

The Benchmark database is a collection of several classification tasks (that is, the subdivision of a dataset into +train, +test, –train and –test groups) defined on a given database so as to represent various degrees of difficulty. For instance, the sequences in one database are closely related to each other within the group, while there are relatively weak similarities between the groups. On the other hand, other databases are less closely related to each other in terms of sequence similarity and the similarities between groups are also weak. Finally, sequences of the same protein in different organisms that can be divided into taxonomic groups that represent a case where both the within-group and between-group similarities are high. At present the collection contains 35 benchmark tests consisting of 10–490 classification tasks and the total number of the classification tasks is 9447.

The database and a collection of documents with help files can be accessed at `http://hydra.icgeb.trieste.it/benchmark/`. The records can be accessed directly from the homepage (see Fig. 3.1). Each record contains statistical data, a detailed description of the methodology used to produce the data and the analysis re-

sults. The results are represented as tables of AUC values obtained by ROC analysis and detailed tabulated lists can be generated on–line in various formats.

The Author's colleagues carried out the collection and maintaining of sequences. We will give a brief description of the sequence dataset in Section 3.2. In addition, their roles were to design the architecture of the database as well as to design, implement and maintain the web page for benchmark databases. The Author's contributions were the evaluation of the performance of the most popular classification methods and evaluation of the sequence similarity methods. These are presented and summarized along with some suggestions on how best to use them in sections 3.4 and 3.3. The Author also helped to design the classification tasks, summarized in sections 3.5 and 3.6. After, Section 3.7 summarizes our findings.

3.2 Protein Sequences

The sequences we selected are deposited in concatenated FASTA format (`http://www.ncbi.nlm.nih.gov/blast/fasta.shtml`), the structures are in PDB format (`http://www.rcsb.org/static.do?p=file_formats/pdb/index.html` or `http://www.pdb.org/`).

3PGK. The dataset was constructed from evolutionarily related sequences of a ubiquitous glycolytic enzyme, 3-phosphoglycerate kinase (3PGK, 358 to 505 residues in length). 131 3PGK sequences were selected which represent various species of the archaean, bacterial and eukaryotic kingdom [55]. The Archea consist of Euryarchaeota(11 of species) and Crenarchaeota(4) phylums, the Bacteria consist of 4 phylums, namely Proteobacteria(30), Firmicutes (35), Chlamydia(3), Actinobacteridae(5) and finally the Eucaryota sequences were obtained from 7 phylums, namely Metazoa(12), Euglenozoa(5), Fungi(10), Alveolata(4), Mycetozoa(1), Viridaeplantae(8) and Stramenopiles(3). Here the sequences are uniform in length and are closely related to each other, i.e. there is a strong similarity both between the members of a given group (phylum) and between the various groups. This set seems to be small, but it is quite difficult to handle because the groups greatly differ in the number of members, and the average similarity within and between groups with any particular sequence similarity method.

SCOP95. The sequences were taken from the SCOP database 1.69 [56].The entries of the SCOP95 (<95% identity) were downloaded from the ASTRAL site `http://astral.berkeley.edu`. 121 non-contiguous domains were discarded and 11944 entries were retained. The domain sequences included in this dataset are variable in terms of length and often there is relatively little sequence similarity between the protein families.

SCOP40. The sequences were taken from the SCOP database 1.69 [56]. The entries of the SCOP40 (<40% identity) were downloaded from the ASTRAL site `http://astral.berkeley.edu/`. 53 non-contiguous domains were discarded and 7237 entries were retained. Protein families, with at least 5 members and at least 10 members outside the family but within the same superfamily in SCOP95 were selected. The SCOP40 dataset is even more difficult since here sequences more similar to each other

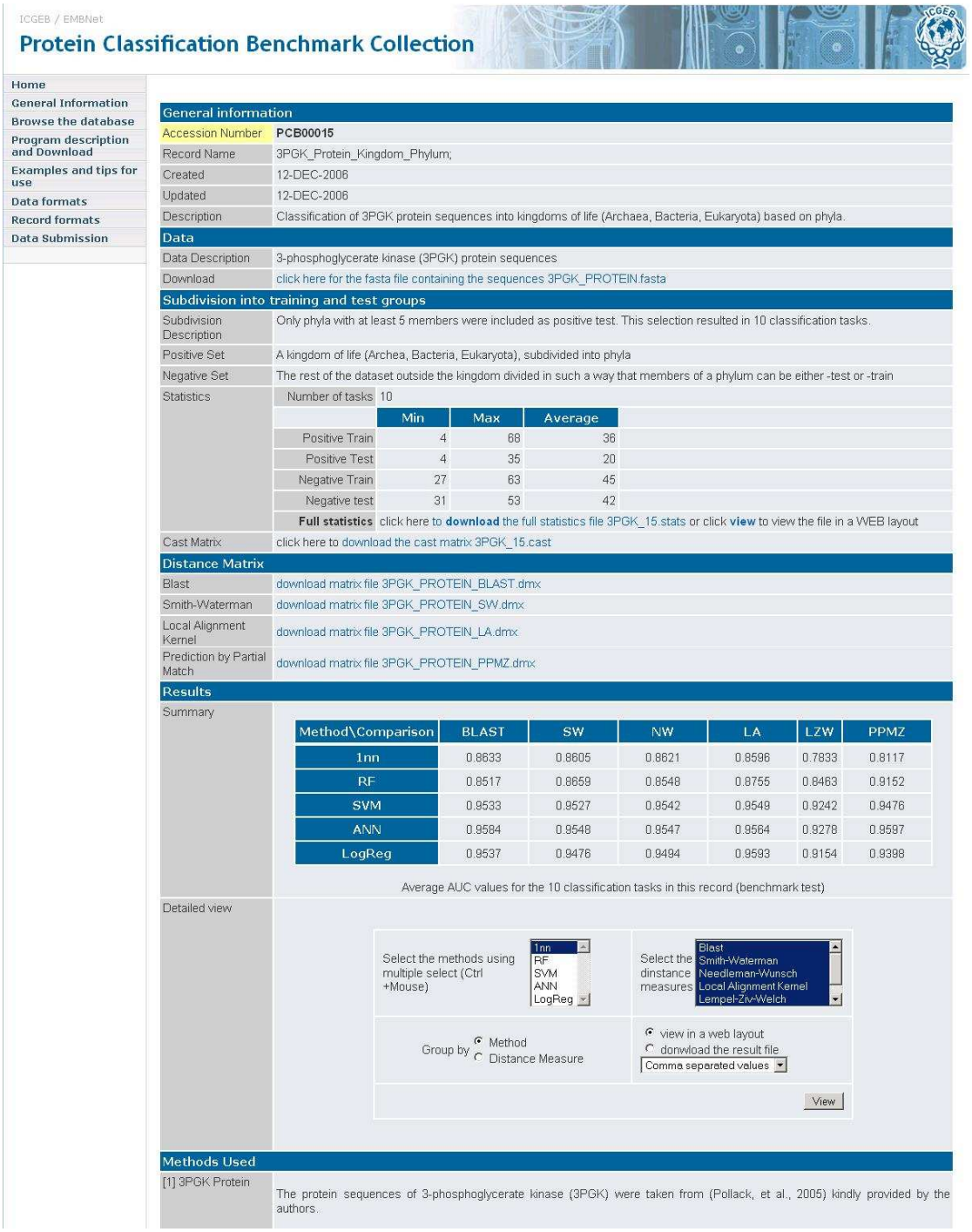


Figure 3.1: A screenshot of the benchmark database.

Table 3.1: Classification of SCOP95 sequences.

SCOP95 classes	#Sequences	#Families	#Superfamilies	#Folds
α	2141	607	375	218
β	3077	559	289	143
α/β	2801	629	222	136
$\alpha+\beta$	2612	711	407	278
Mulitdomain	204	60	45	45
Membrane and cell surface	222	98	87	47
Small	887	170	107	74
Total	11944	2834	1532	941

Table 3.2: Classification of SCOP40 sequences.

SCOP95 classes	#Sequences	#Families	#Superfamilies	#Folds
α	258	102	5	4
β	377	65	6	5
α/β	679	113	11	11
$\alpha+\beta$	23	5	1	1
Mulitdomain	20	6	1	1
Membrane and cell surface	0	0	0	0
Small	0	0	0	0
Total	1375	291	24	22

than 40% are represented by a single prototype sequence.

CATH95. The sequences were taken from the CATH database v.3.0.0 [57]. The entries of the CATH95 (>95% identity) selection were downloaded from the `ftp://ftp.biochem.ucl.ac.uk/pub/cathdata/v3.0.0/` site. The 1648 non-contiguous domains were discarded and 11373 were retained.

COG. This dataset is a subset of the COG database of functionally annotated orthologous sequence clusters [58]. In the COG database, each COG cluster contains functionally related orthologous sequences belonging to unicellular organisms, including archaea, bacteria and unicellular eukaryotes. For a given COG group, the positive test set included the yeast sequences, while the positive training set was the rest of the sequences. Of the over 5665 COGs we selected 117 that contained at least 8 eukaryotic sequences and 16 additional prokaryotic sequences. This dataset contains 17973 sequences. In the COG database, the recognition tasks were designed to answer the following question: can we annotate genomes of unicellular eukaryotes based on prokaryotic genomes? In this dataset the members of a given group are very similar to each other, but the members of different groups have relatively little similarity between the various groups.

3.3 Protein Sequence Comparison

Dataset versus dataset (All-vs-all) comparisons were calculated by using several protein sequence similarity methods. Here we summarize what we applied.

Smith-Waterman (SW). The SW [21] method is described in Section 2.1. The code we applied was the part of the Bioinformatics toolbox 2.0 of Matlab. For the substitution matrix we used the BLOSUM62 matrix [17], while the gap open and extension

Table 3.3: Classification of CATH95 sequences.

CATH95 classes	#Sequences	#H groups	#T groups	# A groups
α	2672	628	279	5
β	3334	393	176	19
$\alpha - \beta$	5107	839	445	14
FewSS	260	100	89	1
Total	11373	1960	989	39

parameter values were set to 11 and 1, respectively.

Needleman-Wunsch (NW). The NW [19] global alignment procedure is also described in Section 2.1. It was downloaded from the European Molecular Biology Open Software Suite (EMBOSS) [59] and we applied the same parameter settings as in SW.

Basic Local Alignment Search Tool (BLAST). Blast [37] is a fast heuristic method for the SW algorithm. The program was downloaded from the NCBI, version 2.2.13. The cut-off parameter value was 25, while the other parameter values we used were the same as in SW.

Local Alignment Kernel (LAK). It was also described in the background section. The program was downloaded from the Author's homepage (`http://cg.ensmp.fr/~vert`). The substitution matrix, gap opening and extensions were the same as in SW, but the scaling factor β was set to 0.5, as suggested in [27].

PRobability of IDEntity (PRIDE). This is a 3D structure comparison method[60] based on representing protein structures in terms of alpha-carbon distance distributions, and comparing two sets of distributions (representing two protein structures, respectively) via contingency table analysis. The program was provided by Zoltán Gaspari.

Distance-matrix ALIgnement (DALI). In order to evaluate the DALI method the program we used was DALI-lite version 2.4.2 [34] and was downloaded from `http://ekhidna.biocenter.helsinki.fi/dali_lite/downloads`.

Compression-based Distances (CBDs). Comparisons with CBDs [61] were calculated by the following formula

$$CBD(s,t) = \frac{C(st) - \min\{C(s), C(t)\}}{\max\{C(s), C(t)\}}, \tag{3.1}$$

where s and t are sequences to be compared and $C(.)$ denotes the length of a compressed string, compressed by a particular compressor C, like the LZW or PPMZ algorithm [4]. For details, see Chapter 5.

The LZW algorithm was our implementation in C, while the PPMZ algorithm was downloaded from Charles Bloom's homepage (`http://www.cbloom.com/src/ppmz.html`)

3.4 Classification Techniques

In this section we describe the classifier algorithms that we generally used in our experiments. At the end of each description we explain why we decided to use it, we explain our parameter settings and we summarize our observations.

3.4.1 k-Nearest Neighbour (kNN)

The k-Nearest-Neighbour (kNN) algorithm [31] is a simple class prediction technique which achieves a high performance with a priori assumptions from only a proximity procedure. The training just involves the storing of the train data along with the class labels. In the actual classification of a new point, the distances from each stored training sample are calculated and the class label is predicted using the most frequently class labels of the closest k training samples.

The kNN classifier is a fast algorithm, is based on simple distance calculations between objects and it does not involve any difficult numerical calculations, have not any convergence problems or numerical instability [62]. This method always gives reliable results in all circumstances. In our experiments we mostly used the 1NN (i.e. $k = 1$) case because the k parameter does not influence the results significantly.

For the evaluation of the model we used the ROC analysis procedure, where the ranking variable was the similarity score between the closest training data and the test sample.

3.4.2 Support Vector Machines (SVMs)

The Support Vector Machine (SVM) classifier [63] is the state-of-the-art supervised classifier technique. Let $L = \{(x_i, y_i) \mid x_i \in \mathbb{R}^n,\ y_i \in \{-1, 1\}\}$ be a set of vectors with class labels called the training data. The SVM method calculates the separating linear hyperplane $f(z) = \langle w, z\rangle + b$ with maximum margin by solving the following constrained convex programming problem

$$\min_{w,\xi} \quad \|w\|_2^2 + \sum \xi_i \tag{3.2}$$

$$s.t. \quad y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i, \tag{3.3}$$

where w is the norm vector of the separating hyperplane and ξ_i is the slack variable introduced for non-linearly separable classes. The norm vector w of the function f can be expressed as a weighted linear combination of the training vectors; that is, $w = \sum y_i \alpha_i x_i$, where $\alpha_i \geq 0$ is the so-called Lagrangian multiplier corresponding to x_i. The vector x_i for which $\alpha_i > 0$ is called the *support vector*. Then the decision boundary can be written by

$$f(z) = \sum_i y_i \alpha_i \langle z, x_i\rangle + b \tag{3.4}$$

and the prediction of the class label of an unseen data t can be done according to the sign of $f(t)$.

The SVM approach has been successfully applied in computational biology including protein remote homology detection [28], microarray gene expression analysis [64], the recognition of translation start sites [65], the functional classification of promoter regions and the prediction of protein-protein interactions [66].

The SVM library that we used in our experiments was the LibSVM [67] version of 2.83. We decided to use this because it is a fast implementation (developed in C++), it also has a lots of interfaces to several programming language like Matlab, R, Weka, Python, C# and Lisp, and it is widely used and trusted software. In our experiments the Radial Basis Function kernel was utilized instead of the simple scalar product and its width parameter σ was the median Euclidean distance from any positive training example to the nearest negative example [68]. This parameter setting seemed a good choice on average in our datasets. For the evaluation of the model, in a ROC analysis the score for the ranking variable was calculated via Eq. 3.4.

3.4.3 Artificial Neural Networks (ANNs)

The Artificial Neural Networks (ANNs) approach was originally developed with the aim of modelling information processing and learning in the brain [69]. ANNs are good at fitting functions and recognizing patterns. In fact there is a proof that a fairly simple neural network can fit any practical function. Within the bioinformatics area this learner has been employed, for instance, in biological sequence analysis, the recognition of signal peptide cleavage sites, gene recognition [70], the prediction of protein functional domains [71] and the classification of cancer subtypes [72].

An ANN is composed of interconnected simple neurons, also called perceptrons, organized in levels (see Fig. 3.2), and trained in a supervised fashion [69].

In our experiments the network structure consisted of one hidden layer with 40 neurons and the output layer consisted of one neuron. The value produced by the output neuron was applied as a ranking variable in the ROC analysis. In each neuron the log-sigmoid function was used as the transfer function, which is one of the most popular transfer functions. In the training phase we evaluated several methods. The

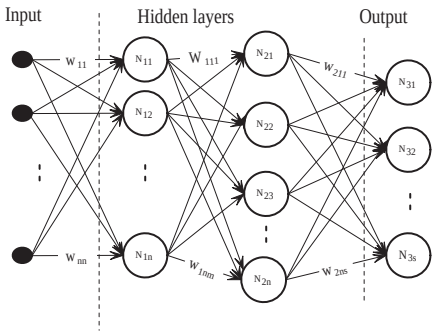


Figure 3.2: A typical ANN structure.

Levenberg-Marquardt (LM) method performance was relatively poor here, but it is recommended for function approximation rather than pattern recognition. The Resilient Backpropagation (RB) [73] method was the fastest algorithm and the memory requirements are relatively small in comparison to other training methods. The performance of BFGS Quasi-Newton [74] is similar to the LM method but it requires less memory but it has a longer execution time. The Scaled Conjugate Gradient (SCG) [75] algorithm performed best on average over a wide variety of classification tasks. It is almost as fast as RB or faster, but it provides a better performance and has relatively modest memory requirements. So, in our experiments we decided to use the SCG method for training an ANN. The package we applied was the Neural Network Toolbox 5.0 version part of Matlab.

3.4.4 Random Forests (RF)

The Random Forests (RF) [76] technique is a recently proposed meta-classifier method, which is becoming evermore popular in machine learning and in computational biology like drug discovery [77] and tumour classification [78]. The RF approach combines decision tree predictors in such a way that each tree is grown on a bootstrap sample of the training set. Let the number of training cases be n, and the number of features be M. Then each tree is constructed using the following algorithm:

I Select the number m of features $(m << M)$ to be used to determine the decision at a node of the tree.

II Choose a training set for this tree by bootstrapping. Use the rest of the cases to estimate the error of the tree by predicting their classes.

III For each node of the tree, randomly choose m features (from an independent, identical distribution, out of the feature set) on which to base the decision at that node. Calculate the best split based on these m variables in the training set.

IV Each tree is fully grown and not pruned (as may be done when constructing a normal tree classifier).

The output of the RF procedure for an unseen sample is the class label that occurs most frequently in the classes output by the individual trees. It also returns the probability that the test sample belongs to the positive class, which was used as an ranking variable in order to evaluate the performance of this classifier with ROC analysis.

The generalization error for forests converges to a fixed limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Using a random selection of features to split each node yields error rates that compare favourably with Adaboost [79], but are more robust with respect to noise[76].

We used this classifier because it is even more popular in computational biology and it has been implemented in Java as a part of the WEKA package [80] and in R as well. In our experiments the parameter m was set to the default value $(log(n+1))$

and the number of trees was set to 20, which seemed a good trade-off between speed and memory usage.

3.4.5 Logistic Regression (LogReg)

The Logistic Regression (LogReg) approach is one of the generalized linear models which is used when the response variable is a dichotomous variable (i.e. it can take one of two possible values) and the input variables are continuous [81]. Unlike linear regression, logistic regression does not assume any linear relationship between the input and the output, but it does exploit the advantages of the linear methods. It does this by utilizing the $\ln\left(\frac{p}{1-p}\right) = \langle w, x\rangle + b$ function, called a link function, and thus it leads to a non-linear relationship, where the $p = P(y_i = 1)$ is the probability that x_i belongs to the positive class.

This is a simple method that is similar to linear regression and it has no actual tuning parameter. The LogReg procedure we applied in our study was part of Weka version 3-4. [80].

3.4.6 The Empirical Feature Map

In our studies the sequences were represented by the so-called Empirical Feature Map method; that is, a sequence s is represented by a feature vector

$$F_s = [f_{x_1}, f_{x_2}, \ldots, f_{x_n}]. \tag{3.5}$$

Here n is the total number of proteins in the training set and f_{x_i} is a similarity/distance score between sequence $\mathcal{S}$ and the ith sequence in the training set. For the underlying proximity measure we used various proximity methods given in the Section 3.3.

The disadvantage of this sequence representation is that the number of features is equal to the number of the training elements. Therefore, methods that are based on spectral decomposition or matrix inverse are difficult to use because numerical instability may occur. However, we decided to use this sequence vectorization method because it provides a significantly better representation method than the n-gram word count method (i.e. double, triplet character composition).

3.5 Supervised Cross-Validation

Datasets based on traditional cross-validation (k- fold, leave-one-out, etc.) may not give reliable estimates on how an algorithm will generalize to novel, distantly related subtypes of the known protein classes. Supervised cross-validation, i.e. the selection of test and train sets according to the known subtypes within a database, was successfully used earlier in conjunction with the SCOP database. Our goal was to extend this principle to other databases and to design standardized benchmark datasets for protein classification. The hierarchical classification trees of protein categories provide a simple

and general framework for designing supervised cross-validation strategies for protein classification. Benchmark datasets can be designed at various levels of the concept hierarchy using a simple graph-theoretic distance. The resulting datasets provide lower - and in our opinion more realistic - estimates of the classifier performance than do random cross-validation schemes.

Let us assume that a database consists of objects that are defined according to terms arranged in hierarchical classification tree. Then the dataset can be regarded as a rooted tree in which the leaves are the protein entries of the database and the root of the tree is the database itself. Each of the other nodes defines a subgroup of protein entries that are the leaves connected to the given node. Fig. (3.3A) shows a typical example of a balanced tree. In this way the nodes in $L(i)$ represent a partition of the database into disjoint groups labeled by the categories at level i, where $L(i)$ denotes the set of nodes at depth i - or level i, - consisting of nodes having the same depth value. We note that here the level i is increasing from the leaves i.e. the database entries belong to $L(0)$ while the root is at $L(H)$, where H stands for the highest level.

In order to construct supervised classification tasks for a given database, we need to know the subdivision of at least two adjacent hierarchical levels (e.g. superfamily/family). The '$+$' and '$-$' groups are defined at the lower lever $L(i)$, while the

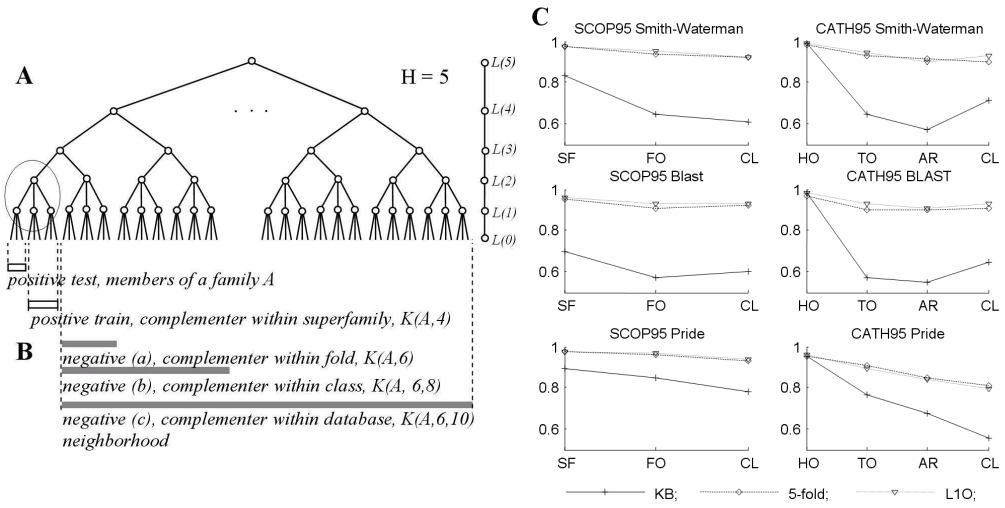


Figure 3.3: The application of a supervised cross-validation scheme to an arbitrary classification hierarchy. (A) Definition possibilities for positive and negative sets within a classification hierarchy. The hierarchy is a schematic and partial representation of that of the SCOP database. The positive set is defined at the $L(2)$ level, while the +test and +train sets are defined at the $L(1)$ underlying level. (B) The boundaries of the negative set can be fixed in terms of the number of steps within the tree hierarchy, calculated with respect to the positive set A. For instance, $K(A, 4)$ defines a neighbourhood (a) whose members are 4 steps apart from the members of group A. (C) A comparison of supervised and random cross-validation schemes on the SCOP and CATH databases benchmark tests at various levels of the classification hierarchy, using Smith-Waterman (top), BLAST (middle) for sequence comparison and PRIDE (bottom) for structure comparison. Categories on the X axis are the levels of the classification (In SCOP: SF: superfamilies; FO: folds divided; CL; classes; In CATH: HO: homology groups; TO: topology groups; AR; architecture groups; CL: classes), the Y axis shows the average ROC scores in a benchmark test. KB=supervised (knowledge-based) (+); L1O=($\Diamond$) leave-one-out, 5-fold cross-validation ($\triangledown$). Note that the random subdivisions give higher values than the supervised (knowledge-based) ones.

Table 3.4: The distribution of proteins in benchmark tests defined on SCOP95.

SCOP95	(1) Superfamilies divided by families		(2) Folds divided by superfamilies		(3) Classes divided by folds	
	#+Test sequences	#+Test families	#+Test sequences	#+Test families	#+Test sequences	#+Test families
α	614	43	522	145	1899	453
β	1507	55	1727	178	2921	466
α/β	1392	86	734	150	2675	557
$\alpha+\beta$	503	41	583	134	2300	505
Mulitdomain	33	4	0	0	148	24
Membrane and cell surface	0	0	27	5	167	62
Small	274	17	308	38	817	120
Total	4323	246	3901	650	10927	2187

Table 3.5: The distribution of proteins in benchmark tests defined on CATH95.

CATH95	(1) Homology groups divided into sequence similarity groups		(2) Topology groups divided by homology groups		(3) Architecture divided by topology groups		(4) Classes divided by architecture groups	
	#+Test sequences	#+Test H-groups	#+Test sequences	#+Test H-groups	#+Test sequences	#+Test H-groups	#+Test sequences	#+Test H-groups
α	503	198	1277	403	2329	594	2590	617
β	498	134	1508	262	2896	360	3253	390
$\alpha-\beta$	773	282	2370	578	4478	800	5009	833
FewSS	58	35	133	67	235	92	251	99
Total	1832	649	5288	1310	9938	1846	11103	1939

training/test subdivision is defined at the higher level $L(i+1)$. Let a be an inner node at level $L(i)$ in the hierarchy and let A be the set of leaves (database entries) that belong to a. Next let $K(A,n)$ be the set of proteins that are exactly n steps away from the members of the set A on the shortest path. We will use the notation $K(A,n,m)=\bigcup_{k=n}^{m}K(A,k)$ for short. Now the supervised cross-validation is defined in the following way. Let us chose a protein category a at $L(i)$, then let $K(A,i\cdot 2)$ and $K(A,(i+1)\cdot 2)$ be the positive test and training sets, respectively, and the negative set can be chosen from $K(A,k\cdot 2,l\cdot 2)$, where $i+1<k\leq H$ are natural numbers.

The principle is elucidated in Fig. (3.3A), where the positive sets are members of the same family, which are 2 steps away from each other ($K(A,2)$). The members of the negative set on the other hand are 4 steps away from any member of the positive set ($K(A,4)$). Because of its generality, this principle can be applied both to other levels of this hierarchy and to other tree hierarchies.

Figure (3.3C) compares the supervised and the random cross-validation methods at various levels of the SCOP and CATH hierarchies. The results show that the ROC scores of random cross-validation tests are substantially higher than those in the supervised case. At all levels of the hierarchies there is a clear tendency in the ranking: leave-one-out test $\sim$ 5-fold cross-validation $>$ supervised cross-validation. This tendency indirectly explains why an outstanding performance obtained with a random cross-validation technique does not necessarily guarantee a good performance on sequences from new genomes; in other words, random cross-validation techniques may grossly overestimate the predictive power of a method on new genomes.

The tendencies shown in Figure 3.3C confirm the well-known fact that a prediction

at a lower levels of the hierarchy is more efficient than at higher level. It is also apparent that the difference between the random and the supervised subdivision is larger at the higher levels of the hierarchy, in spite of the fact that the domain definitions and the hierarchies of SCOP and CATH are different. On the other hand, the difference between of the two databases is reflected by the different shapes of the corresponding curves.

3.6 Selecting Negative Datasets of a Manageable Size

In the previous section, we varied just the subdivision of the positive set and kept the subdivision of the negative set constant. Naturally one can also vary the way how the negative test is subdivided. In a typical binary classification task described in the previous section, the positive sets consisted of several dozen to several hundred sequence objects but the negative set was simply defined as "the rest of the database". In other words, the dataset was large and imbalanced because of the size of the negative sets. Filtering the negative set is a plausible idea and we compare three strategies for doing this: (i) Random subsampling of the negative set (10%, 20% or 30%); (ii) The election of the nearest neighbours of the positive group based on a Smith-Waterman score and (iii) The selection of the category neighbours. The latter is a supervised selection, and the principle is outlined in Fig. 3.3B. If the positive set is a given superfamily, the negative set can be chosen just like the other superfamilies within the same fold (a), or other superfamilies within the same class (b).

A higher performance is characterized by a higher average and a lower standard deviation. The results show that a random filtering of the negative group does not substantially influence the classification performance of the 1NN classifier or of the SVM classifier. However when we choose the nearest neighbours based on the Smith-Waterman algorithm, there is a decrease in the performance of the 1NN classifier but the performance of the SVM classifiers remains roughly the same. On the other hand when we select the negative sets on the basis of the classification hierarchy, both classifiers show a performance decrease. The number of structural categories present in the negative groups provides some insight into these tendencies.

Selecting the nearest neighbours in terms of a Smith Waterman distance or in terms of a structural hierarchy sharply reduces the number of structural categories present in the negative set. The second method also produces very small datasets. As a result of these two factors, the negative sets will be too specific i.e. they may be less representative of the entire database. Based on the above results, the random filtering of the negative sets seems to be the most sensible compromise, since the classification performances remain close to those of the original dataset, and the number of structural categories is higher than in the case of the other two methods.

Finally we should mention that the difference between the behaviour of 1NN and SVM in these calculations may be due to the fact that a ROC analysis of 1NN is based on a one-class scenario (outlier detection), whereas the performance of support vector

Table 3.6: Dependence of classifier performance and the subgroup composition on the selection of the negative set (SCOP95 dataset, SW score).

Size of	1NN	SVM	Number[c] of			
negative set (%)	ROC score ± s.d[b]		Families	Superfamilies	Folds	Classes
Full negative set (11162-11929 sequences)[d]						
100	0.8352 ± 0.17	0.8641± 0.15	2824 (2779-2832)	1531 (1531 - 1531)	941 (940 - 941)	7
Random selection[e]						
10	0.8352 ± 0.13	0.8865 ± 0.17	734 (700 - 764)	520 (480 - 552)	369 (341 - 395)	7
20	0.8351 ± 0.14	0.8776 ± 0.17	1182 (1124 - 1223)	770 (728 - 810)	514 (480 - 553)	7
30	0.8353 ± 0.17	0.8754 ± 0.14	1522 (1458 - 1572)	943 (902 - 983)	609 (577 - 637)	7
Nearest neighbor selection[f]						
10	0.5836 ± 0.32	0.8738 ± 0.19	452 (383 - 559)	270 (226 - 347)	205 (164 - 265)	6 or 7
20	0.6410 ± 0.23	0.8954 ± 0.16	810 (727 - 940)	438 (384 - 536)	328 (280 - 403)	6 or 7
30	0.7389 ± 0.28	0.8880 ± 0.17	1136 (1038 - 1287)	595 (529 - 717)	434 (390 - 510)	6 or 7
Supervised selection[g] (Knowledge-based)						
10.9 ± 1.3 (a)	0.7083±0.21	0.7651 ± 0.19	26 (2 - 79)	11 (1 - 30)	1	1
20.8 ± 4.8 (b)	0.7571 ±0.18	0.8664 ± 0.14	558 (165 - 705)	287 (106 - 406)	165 (74 - 287)	1

[a] AUC values were calculated for the 246 classification tasks defined on the SCOP95 dataset.

[b] The results were calculated as the average and standard deviation of the AUC values. The random selection was repeated 10 times, and the s.d. value is the average of the 10 calculations.

[c] The numbers indicate an average (minimum-maximum) within the classifier tasks.

[d] The negative sets (11162-11929 sequences) were subdivided into roughly equal -train and -test groups in such a way that members of a given family were either train or test.

[e] The given percentage of the -train and -test set was selected randomly.

[f] The members of the -test and -train sets were ranked according to their highest Smith Waterman score for the +train group, and the closest sequences corresponding to the given percentage were selected.

[g] The selection was based on the ranges indicated by a and b in Figure 3.3B.

machines is evaluated with respect to a decision surface separating the +train and –train classes.

3.7 Discussion

Tables 3.7-3.9 summarize the classification performance obtained by using various methods (columns) and various proximity algorithms (rows). We should mention that because of time constraints some of the comparison algorithms (PPMZ, DALI) were not calculated for the biggest datasets. The results show a sufficiently clear tendency: the ranking of the machine learning methods is roughly the same on all the datasets, i.e. usually SVM gave the best performance closely followed by ANN and LogReg. Note

Table 3.7: Comparison of distance measures and machine learning algorithms on SCOP40.

	1NN	RF	SVM	ANN	LogReg
BLAST	0.7577	0.6965	0.9047	0.7988	0.8715
SW	0.8154	0.8230	0.9419	0.8875	0.9063
NW	0.8252	0.8030	0.9376	0.8834	0.9175
LA	0.7343	0.8344	0.9396	0.9022	0.8766
LZW	0.7174	0.7396	0.8288	0.8346	0.7487
PPMZ	0.5644	0.7253	0.8551	0.7254	0.8308
PRIDE	0.8644	0.8105	0.9361	0.9073	0.9029
DALI	0.9892	0.9941	0.9946	0.9897	0.9636

Table 3.8: Comparison of distance measures and machine learning algorithms on SCOP95–10.

	1NN	RF	SVM	ANN	LogReg
BLAST	0.6985	0.6729	0.7293	0.5703	0.7218
SW	0.8354	0.8645	0.8884	0.8607	0.8574
NW	0.8390	0.8576	0.8910	0.8396	0.8607
LA	0.7884	0.8823	0.8717	0.9002	0.8718
LZW	0.7830	0.8208	0.8402	0.7851	0.7574

Table 3.9: Comparison of distance measures and machine learning algorithms on 3PGK.

	1NN	RF	SVM	ANN	LogReg
(A) protein sequences					
BLAST	0.8633	0.8517	0.9533	0.9584	0.9537
SW	0.8605	0.8659	0.9527	0.9548	0.9476
NW	0.8621	0.8548	0.9542	0.9547	0.9494
LA	0.8596	0.8755	0.9549	0.9564	0.9593
LZW	0.7833	0.8463	0.9242	0.9278	0.9154
PPMZ	0.8117	0.9152	0.9476	0.9597	0.9398
(B) DNA sequences					
BLAST	0.7358	0.8244	0.8102	0.8077	0.7691
SW	0.8864	0.7674	0.9630	0.9698	0.9772
NW	0.8437	0.8959	0.9455	0.9433	0.9612
LA[a]	n.a	n.a	n.a	n.a	n.a
LZW	0.7143	0.7107	0.8343	0.8297	0.7844
PPMZ	0.7336	0.7662	0.7881	0.7918	0.8612

LA is not defined on DNA sequences

that this ranking does not refer to the potential or the optimum performance of the machine learning algorithms, but rather reflects their actual performance obtained with the default settings employed here.

On the other hand, the ranking of the proximity measures displays some variation between the datasets. Overall we see a clear general trend i.e. the 3D comparison algorithms are better than sequence-based methods, and exhaustive methods (SW, NW) perform better than heuristic algorithms (BLAST). Nevertheless, BLAST performed slightly better than SW on the 3PGK DNA sequences. These variations are in a way expected, after recalling that the within-group and between-group similarities are different in the various datasets.

We hope that these datasets will be useful for method developers and for users interested in assessing the performance of various methods.

Part II

Protein Similarity

Chapter 4

Likelihood Ratio Approximation to Protein Sequence Classification

The aim of this study was to test the utility of the likelihood ratio scoring technique on similarity searches using an a priori classified sequence database. Here the Author observed that the simple likelihood ratio (LR) can improve the ranking ability of a protein similarity measure. He designed and evaluated the comparative experiments which justified his view that this LR scoring significantly improves the performance of proximity functions. In this project the Author's colleagues selected the databases for the experiments as well as they implemented several routines to manage and characterise the databases.

4.1 Introduction

The closeness of a raw sequences x to a class of proteins '$+$' can be calculated by a similarity function s and by the maximum similarity value between x and the members of the positive class. Here it is denoted by $s(x,+) = \max_{z \in +}\{s(x,z)\}$. Then a set of sequences can be ranked by a similarity function for the positive class, and its ranking ability can be evaluated by a ROC analysis. Formally, we can denote this ranking technique by

$$POS(x) \sim s(x,+).$$

In addition, the prediction of the positive class can be carried out by choosing a priori decision threshold that is similar to the concept of the outlier detection.

LR is a familiar concept in statistics for hypotheses testing. A statistical model is often a parametrized family of probability density functions or probability mass functions $f(x;\theta)$. A simple-vs-simple hypotheses test hypothesises single values of θ under both the null ($H_0 = \theta_0$) and alternative ($H_A = \theta_A$) hypotheses. The likelihood ratio test statistic is defined by [31]:

$$LR(x) = \frac{f(x;\theta_0)}{f(x;\theta_A)},$$

(some references may use the reciprocal as the definition). The LR acceptes the null-hypothesis when the ratio exceeds a certain pre-defined threshold c, rejects otherwise and its optimality is known as the Neyman-Pearson lemma [31].

It is well known that the Bayes decision rule for a binary classification can be reformulated as a likelihood ratio test in the following way

$$LR(x) = \frac{p(x \mid +)}{p(x \mid -)},$$

where $p(x \mid +)$ (resp. $p(x \mid -)$) is the class-conditional density function for the class 'x' (resp. '$-$') and x is classified as $+$ when the ratio above a certain threshold [31]. In the framework of protein classification, LR can be approximated by the formula:

$$LRA(x) = \frac{p(x \mid +)}{p(x \mid -)}$$

where x is the query protein, the '$+$' symbol stands for a particular protein class, '$-$' denotes the complementer class, and $p(x \mid +)$ and $p(x \mid -)$ represent the class-conditional probability density functions.

In the next section we will derive the LR approximation (LRA) from arbitrary similarity or dissimilarity measures, and then we will present our results along with some conclusions and a few thoughts.

4.2 Approximating Class Conditional Probabilities using Sequence Proximity Functions

Let $\mathcal{S}$ denote a finite set of proteins in the database which consists of two classes called '$+$' and '$-$'. We shall partition the set $\mathcal{S}$ into two subsets called a train set and a test set and call the positive and negative elements in the train set train+ and train-, respectively. Furthermore, let $h(x, y)$ be a (sequence) similarity function over a set $\mathcal{S}$.

For any protein $x \in \mathcal{S}$ and sequence similarity function h we can define the similarity between x and a class '+' by $s(x, +) = \max_{y \in train+} h(x, y)$, and similarly between x and a class '$-$' by $s(x, -) = \max_{y \in train-} h(x, y)$, respectively. The class-conditional probability functions can be approximated by

$$p(x \mid +) \approx \frac{s(x, +)}{M_+} \tag{4.1}$$

$$p(x \mid -) \approx \frac{s(x, -)}{M_-}, \tag{4.2}$$

where $x \in \mathcal{S}$, $M_+ = \sum_{x \in \mathcal{S}} s(x, +)$ and $M_- = \sum_{x \in \mathcal{S}} s(x, -)$. These are valid probability functions because their range is [0,1] and their sum over all proteins is equal to 1.

Now, based on Eqs. 4.1 and 4.2, we have the following estimation for the log

likelihood ratio:

$$\begin{aligned}\log LRA(x) = \log\left(\frac{p(x \mid +)}{p(x \mid -)}\right) \approx \log\left(\frac{s(x,+)/M_+}{s(x,-)/M_-}\right) = \\ = \log\left(\frac{s(x,+)}{s(x,-)}\right) + \log\left(\frac{M_-}{M_+}\right).\end{aligned} \tag{4.3}$$

The bias term in the last expression is independent of the element x, so it does not count in the ranking of the elements of $\mathcal{S}$, i.e it has no effect on a ROC analysis or the AUC value. The function $\log$ can also be omitted because it is a strict monotone increasing function that does not affect the ranking. This is why we can define the LRA from similarity function by

$$LRA(x) = \frac{s(x,+)}{s(x,-)}. \tag{4.4}$$

In essence, ROC curves and AUC calculations are used to investigate the performance of learning algorithms under varying conditions e.g. with misclassification costs or class distributions. In practice one can build classifiers using threshold values, which uses a bias term that appears in Eq. 4.3.

Note that Eq. 4.4 implies that the probability of an object belonging to a class is proportional to its maximum similarity of the members of that class.

Now let us define the function d (similarly to s) by $d(x,+) = \min_{y \in train+} l(x,y)$, and $d(x,-) = \min_{y \in train-} l(x,y)$, where l stands for a dissimilarity function. Moreover let us suppose that d and s are related to each other by a monotone decreasing function f such as:

$$f_1^\beta(x) = \exp(-\beta x) \tag{4.5}$$

$$f_2^\beta(x) = \frac{1}{\log 2}\left(-\beta x + \log(1+\exp(\beta x))\right), \tag{4.6}$$

where x is either a similarity measure (obtained by s) or a dissimilarity measure (obtained by d), and β is a positive tunable parameter. Eq. 4.5 or Eq. 4.6 map to [0,1] and the parameter β regulates the slope. The results of a conventional sequence similarity search are ranked according to an s or d value got between a query sequence and the individual database sequences. On the other hand, if the database is a priori classified into classes, we can associate the class labels with the ranked list. Let '$+$' and '$-$' denote the first and second top ranking classes. In order to construct a log likelihood approximant, we will assume that the probability of the query sequence being a member of a class is proportional to the highest similarity or the lowest dissimilarity value obtained between the sequence and the members of the given class. Denoting these values by $s(x,+)$ and $s(x,-)$ (or $d(x,+)$ and $d(x,-)$), respectively, our fundamental log likelihood ratio approximant can be written as:

$$LRA \sim \log\left(\frac{s(x,+)}{s(x,-)}\right) \sim \log\left(\frac{f^\beta(d(x,+))}{f^\beta(d((x,-))}\right) \tag{4.7}$$

where f represents one of Eq. 4.5 or Eq. 4.6. Here the superscript indicates that the approximant may have a tunable parameter.

Finally we mention that this approach could be extended to not just one but to the sum or the mean of the similarity scores between the x and the k closest member in the set $+$, similarly to the concept of the k-nearest neighbour approach. These will be denoted by

$$POS_k(x) \sim s_k(x, +)$$

and

$$LRA_k(x) = \frac{s_k(x, +)}{s_k(x, -)},$$

respectively. In our expreiments we used the avarage of the top $k = 3$ scores.

4.3 Results and Discussion

The ranking performance of the various methods is summarized in Table 4.1 and in Table 4.2. Here it is apparent that the LRA scoring substantially improves the ranking efficiency, as indicated by the high cumulative AUC values. It is also apparent that the performances of the various LRA schemes do not substantially differ among each other, i.e. the choice of monotone decreasing function does not seem to be critical. This is demonstrated by the fact that the data obtained 'without a transformation' (i.e. using Eq. 4.4 for a similarity measure) are nearly identical with those obtained after transformations employing Eq. 4.5 or Eq. 4.6.

As the datasets and the algorithms are quite different in nature, we are led to think that the consistent performance improvement is due to LRA scoring. On the other hand, the results are dataset-dependent: they vary from group to group within each of the three datasets. Nevertheless, the improvement caused by LRA scoring is apparent on the less-perfect COG groups, of which four are included in Table 4.2C.

This group-specific behaviour is also apparent in how the results depend on the value of the β parameter. Here we see that there is a large range where the classification performance is independent of β, but as the value approaches one, there is a substantial variation between the groups. Typically, there may be a substantial decrease in the AUC value, but in some cases there is a slight increase.

Summarizing we may conclude that LRA scoring helped to provide a consistent

Table 4.1: Comparison of the scoring performance of the BLAST and Smith-Waterman (SW) scoring methods by a ROC analysis.

		SCOP	3PGK	COG
SW	POS	0.850	0.791	n.a
	LRA	0.932	0.944	n.a.
BLAST	POS	0.825	0.792	0.987
	LRA	0.892	0.941	0.999

Table 4.2: Comparison of LRA scoring with simple scoring and using a nearest neighbour classification

	POS	LRA by			POS_3	LRA_3 by		
		Eq. 4.4	Eq. 4.5[1]	Eq. 4.6[1]		Eq. 4.4	Eq. 4.5[1]	Eq. 4.6[1]
A. Fold classification (Dataset SCOP)								
SW	0.8496	0.9318	0.9309	0.9304	0.8636	0.9447	0.9440	0.9439
LAK	0.8223	0.9509	0.9393	0.9393	0.8315	0.9601	0.9428	0.9428
BLAST	0.8253	0.8918	0.9197	0.9193	0.8296	0.9019	0.9314	0.9312
LZW	0.8060	0.8468	0.8464	0.8464	0.8264	0.8627	0.8623	0.8623
PPMZ	0.6625	0.7641	0.7778	0.7779	0.6793	0.7571	0.7707	0.7708
B. Taxonomic classification (Dataset 3PGK)								
SW	0.7906	0.9442	0.9169	0.9164	0.7778	0.9540	0.9142	0.9139
BLAST	0.7922	0.9411	0.9157	0.9144	0.7780	0.9514	0.9129	0.9124
LAK	0.7862	0.9393	0.9204	0.9200	0.7765	0.9423	0.9139	0.9137
LZW	0.7370	0.8658	0.8642	0.8642	0.7625	0.8926	0.8915	0.8915
PPMZ	0.7523	0.9004	0.9030	0.9030	0.7588	0.9024	0.9051	0.9051
C. Functional classification based on BLAST scores (3 clusters from the Dataset COG)								
COG0631	0.8659	0.9961	0.9961	0.9961	0.8659	0.8663	0.9961	0.9961
COG0697	0.8566	0.8568	0.9901	0.9901	0.8565	0.8568	0.9891	0.9891
COG0814	0.8208	0.8212	0.9909	0.9909	0.8207	0.8212	0.9906	0.9906

[1]Equation was used in the Eq.4.7.

performance improvement in the protein sequence classification tasks analyzed in this study. But the extent of the improvement seems to depend on the protein group as well as on the database. We should add that the AUC value characterizes the ranking performance of a variable (in our case the LR score), but it does not describe the actual performance of any particular classifier that can be built using that variable. In practice one can build classifiers using threshold values, such as the empirical score or E-value thresholds used in conjunction with BLAST [37], or by using any of the database-dependent optimization techniques [31]. Finally we should mention that the principle described here can be easily implemented, and the fact that it can be applied to a wide range of scoring methods and classification scenarios makes us hope that it will be applicable to other areas of protein classification as well.

Chapter 5

Compression-based Distance Measures for Protein Sequence Classification

Text compressor algorithms can be used to construct distance metric measures (CBDs) suitable for character sequences. The aim of our study was to give an insight to the behaviour of various types of compressor algorithms for the comparison and classification of protein and DNA sequences.
The Author's contributions to this project were (i) the investigation of the CBDs for the sequence representation (reduced and enlarged alphabets), (ii) the hybrid combination of CBDs with a fast but problem-specific comparison method. The Author designed and evaluated all the experiments and the results are summarized in Section 5.4.
The contributions of the Author's colleagues to this project were the sensitivity analysis of CBDs for biological mutations, for the complexity and for the protein sequence length.

5.1 Introduction

The notion of an information distance between two discrete objects is the quantity of information in an object in terms of the number of bits needed to reconstruct the other. This notion arises from the theory of the thermodynamics of computation, which was first mentioned in [82; 83] and in [84] in the context of image similarity. Later, an introduction with related definitions and theory was published in [85] in 1998. More formal definitions, theory and related details can be found in [86].

The interest in compression-based methods was fostered by Ming Li et al.'s paper [87] in which they applied the GenCompress algorithm to estimate the information distance. Then various practical applications and study appeared: language classification [88; 89], hierarchical clustering [61; 90], music classification [91], protein sequence and structure classification [4; 92; 93], SVM kernel for string classification [94] and clustering fetal heart rate tracings [95].

Here we will approximate the information distance by using various types of text compressors and we will examine their behaviour in protein classification and protein sequence comparison from three different point of view. i) First, in Section 5.4.1 we will examine the efficiency of the CBDs as a function of the size of the alphabet. ii) And we will investigate the combination of CBDs with an additional cheap, but problem-specific similarity measure from a protein classification point of view in the section 5.4.3.

But before we start we will give a brief theoretical introduction to the Information Distance in Section 5.2 and then in Section 5.3 we will summarize the types of compressors in section that we used in our experiments.

5.2 Information Distance

A distance function D is a positive real-valued function defined on the Cartesian product of an arbitrary set X. It is also called a metric on X if it satisfies the following so-called metric properties:

- Non-negativity and identity: $D(x, y) \geq 0$ and $D(x, y) = 0$ iff $x = y$.
- Symmetry: $D(x, y) = D(y, x)$
- Triangle inequality: $D(x, z) \leq D(x, y) + D(y, y)$

for every $x, y, z \in X$. These metric properties not only provide a useful characterisation of distance functions that satisfy them, but a metric function is also good as a reliable distance function.

Any finite object can be represented by binary strings without loss of generality (w.l.o.g). For example any genome sequence, arbitrary number, program represented by its Gödel number, image, structure, and term can be encoded by binary strings. Here, the string x is a finite binary string and its length is defined in the usual way and will be denoted by $l(x)$. An empty string will be denoted by ϵ and its length $l(\epsilon) = 0$. The concatenation of x and y will be simply denoted by xy. A set of strings $\mathcal{S} \subseteq \{0, 1\}^*$ is called a prefix-free set if any string member is not a prefix of another member. A prefix-free set has a useful characterisation, namely it satisfies the Kraft inequality. Formally, for a prefix-free set $\mathcal{S}$, we have

$$\sum_{x \in \mathcal{S}} 2^{-l(x)} \leq 1. \tag{5.1}$$

An important consequence of this property is that in a sequence $s_1 s_2 \ldots s_n$ ($s_i \in \mathcal{S}$) the end of a substring s_i is immediately recognizable; that is, the concatenation of strings can be separated by commas into codewords without looking at subsequent symbols. This sort of code is also called self-punctuating, self-delimiting or instantaneous code.

A partial recursive function $F(p, x)$ is called a prefix computer if for each x, the set $\{p \mid F(p, x) < \infty\}$ is a prefix-free set. The argument p is called a prefix program because no punctuation is required to tell F where p ends and the input to the machine

can be simply the concatenation px. The conditional Kolmogorov Complexity of the string x with respect to (w.r.t.) y, with prefix computer F, is defined by

$$K_F(x \mid y) = \min\{l(p) \mid F(p, x) = y\}. \tag{5.2}$$

When such a p does not exist, its value is infinite. The invariance theorem states that there is a universal or optimal prefix computer U for every other prefix computer F and for any x, y such that

$$K_U(x \mid y) \leq K_F(x \mid y) + c_F, \tag{5.3}$$

where c_F depends on F but not on x, y. This universal computer U is fixed as the standard reference, and we call $K(x \mid y) = K_U(x \mid y)$ the conditional Kolmogorov Complexity of x w.r.t. y; moreover, the unconditional Kolmogorov Complexity of a string x is denoted and defined by $K(x) = K(x \mid \epsilon)$. The Kolmogorov Complexity of x w.r.t. y is the shortest binary prefix program that computes x with additional information obtained from y. However, it is non-computable in the Turing sense; that is, no program exists that can compute it in practice since it can be reduced to the halting problem [86].

With the information content, the information distance between strings x and y is defined as the length of the shortest binary program on the reference universal prefix computer with input y, which computes x and vice versa, formally:

$$ID(x, y) = \min\{l(p) \mid U(p, x) = y,\ U(p, y) = x\}. \tag{5.4}$$

This is clearly symmetric, and it has been proven that it satisfies the triangle inequality up to an additive constant. ID can be computed by reversible computations up to an additive constant, but there is an easier but weaker approximation of ID. It has been shown that ID, up to an additive logarithmic term, can be computed by the so-called max distance E:

$$E(x, y) = \max\{K(x \mid y), K(y \mid x)\}. \tag{5.5}$$

In general, the "up to an additive logarithmic term" means that the information required to reconstruct x from y is always maximally correlated with the information required to reconstruct y from x that is dependent on the former amount of information. Thus E is also a suitable approximation for the information distance.

In a discrete space with a distance function D, the rate of growth of the number of elements in balls of size d, centred at x, denoted by $\#B_D(x, d)$, can be considered as a certain characterisation of the space. For example, the distance function $D(x, y) = 1$ for all $x \neq y$ is not a realistic distance function, but it satisfies the triangle inequality. As for the Hamming distance, H, $\#B_H(x, d) = 2^d$, hence it is finite. For a function D to be a realistic distance function it needs to satisfy the so-called normalization property [86]:

$$\sum_{y:y \neq x} 2^{-D(x,y)} \leq 1. \tag{5.6}$$

This holds for the information distance $E(x, y)$ and also for ID because they both satisfy the Kraft inequality. Moreover,

$$\log(\#B_E(x, d)) = d - K(d \mid x) \tag{5.7}$$

up to an additive constant. This means that the number of elements in the ball $B_E(x, d)$ grows exponentially w.r.t. d up to an additive constant. In addition, a more complex string has fewer neighbours and this has been shown by the theorem "tough guys have few neighbours thermodynamically" [82]. E is universal (up to an additive error term) in the sense that it is smaller than every other upper-semi-computable function f which satisfies the normalization property. That is, we have

$$E(x, y) \leq f(x, y) + O(log(k)/k), \tag{5.8}$$

where $k = \max\{K(x), K(y)\}$. This seems quite reasonable, as we have greater time to process x and y, and we may discover additional similarities between them; then we can revise our upper bound on their distance. As regards semi-computability, $E(x, y)$ is the limit of a computable sequence of upper bounds.

The non-negativity and symmetry properties of the information distance $E(x, y)$ are a consequence of the definition, but $E(x, y)$ satisfies the triangle inequality only up to an additive error term [85].

The non-normalized information distance is not a proper evolutionary distance measure because of the length factor of strings. For a given pair of strings x and y the normalized information distance is defined by

$$D(x, y) = \frac{\max\{K(x \mid y), K(y \mid x)\}}{\max\{K(x), K(y)\}}. \tag{5.9}$$

In [89] it has been shown that this satisfies the triangle inequality and vanishes when $x = y$ with a negligible error term. The proof of its universality was given in [96], and the proof that it obeys the normalization property is more technical (for details, see [61; 89]).

The numerator can be rewritten in the form $\max\{K(xy) - K(x), K(yx) - K(y)\}$ within logarithmic additive precision due to the additive property of prefix Kolmogorov complexity [61]. Thus we get

$$D(x, y) = \frac{K(xy) - \min\{K(x), K(y)\}}{\max\{K(x), K(y)\}}. \tag{5.10}$$

Since the Kolmogorov complexity cannot be computed, it has to be approximated, and for this purpose, real file compressors are employed. Let C(x) be the length of the compressed string compressed by a particular compressor like gzip or arj. Then the approximation for the information distance E can be obtained by using the following

formula [97]:

$$CBD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \tag{5.11}$$

A CBD is a metric up to an additive constant and satisfies the normalization property if C satisfies the following properties up to an additive term:

- Idempotency: $C(xx) = C(x)$ and $C(\lambda) = 0$, where λ is the empty string.
- Symmetry: $C(xy) = C(yx)$
- Monotonicity: $C(xy) \geq C(x)$
- Distributivity: $C(xy) + C(z) \leq C(xz) + C(yz)$.

The proof can be found in Theorem 6.2 of [61]. A compressor satisfying these properties is called a normal compressor.

There is no bound for the difference between the information distance and its approximation; that is, $|E - CBD|$ is unbounded. For example, the Kolmogorov complexity of the first few million digits of the number π , denoted by pi, is a constant because its digits can be generated by a simple program but $C(pi)$ is proportional to $l(pi)$ for every known text compressor C.

Here we should mention that the derivation of Eq. 5.9 from Eq. 5.10 requires th use of Theorem 3.5 stated in [86]. But while [98] showed that the proof of this theorem is actually wrong and no correct proof could be found in the literature. In our opinion this factor is negligible in the practice since compressor algorithms provide an unbounded estimation.

5.3 Data Compression Algorithms

In computer science, the purpose of data compression is to store the data more economically without redundancy, and it can be compressed whenever some events are more likely than others. In general, this can be done by assigning a short description code to the more frequent patterns and a longer description code to the less frequent patterns. If the original data can be fully reconstructed, it is called lossless compression. If the original data cannot be exactly reconstructed from those descriptions, it is known as lossy compression. This form is widely used in the area of image and audio compression because the fine details can be removed without being noticed by the listeners. If a set of codes $\mathcal{S}$ satisfies the Kraft inequality, it is a lossless compression and it is a prefix-free set; conversely, if it does not satisfy the Kraft inequality, it is a lossy compression and it is not a prefix-free set. None of data or text string can be losslessly compressed to an arbitrary small size by any compression method. The limit of the compression process that is needed to fully describe the original data is related to Shannon entropy or to the information content [99]. A good collection of compressors and their related descriptions are available at `http://datacompression.info/`. Now, we will briefly describe the compressors used in our experiments.

Adaptive Huffman(Dynamic Huffman Coding, AH). Here, the assumption is that the sequence is generated by a source over a d-ary alphabet D over a probability distribution P; that is, for each $x \in D$, the $p(x)$ is the probability of the letter x and $\sum_{x \in D} p(x) = 1$. Let $C(.)$ be a source code and $C(x)$ be the codeword associated with x, and for ease of notation, let $l(x)$ stand for $l(C(x))$, the length of the codeword. It should be demanded that C is non-singular; that is, $x \neq y \rightarrow C(x) \neq C(y)$ and it gives a prefix-free set. The expected length $L(C)$ of a source code is defined by

$$L(C) = \sum_{x \in D} p(x) l(x). \tag{5.12}$$

The Huffman coding is the optimal source coding; that is

$$L^* = \min_C \{L(C)\}, \tag{5.13}$$

subject to C satisfying the Kraft inequality. Solving this constrained optimization problem using Lagrange multipliers yields the optimal code lengths $l^* = -\log_d p(x)$. The non-integer choice of codeword length gives $L^* = H(P)$; that is, the length of the optimal compression of data is equal to the Shannon entropy of the distribution P of the codewords. Moreover, for integer codeword lengths, we have $H(P) \leq L^* \leq H(P) + 1$ and the codes can be built by a method like Shannon-Fano coding [99].

In practice, the key part of the Huffman coding method is to assess the probability of letter occurrence. The adaptive Huffman coding constructs a code when the symbols are being transmitted, having no initial knowledge of the source distribution, which allows one-pass encoding and adaptation to changing conditions in the data. In our experiments the program that we used was downloaded from `http://www.xcf.berkeley.edu/~ali/K0D/Algorithms/huff/`

Lempel-Ziv-Welch (LZW). The LZW [100] is a one-pass stream parsing algorithm that is widely used because of its simplicity and fast execution speed. It divides a sequence into distinct phrases such that each block is the shortest string which is not a previously parsed phrase. For example, let $x = 01111000110$ be a string of length 11, then the LZW compressor, denoted by $C(x)$, produces 6 codes: 0,1,11,10,00,110, thus $l(C(x)) = 6$. Here, the assumption is that sequences are generated by a higher but finite order stationary Markov process P over a finite alphabet. The entropy of P can be estimated by the formula $n^{-1} C(x) \log_2 C(x)$ and the convergence is almost guaranteed as the length of the sequence x tends to infinity. In practice, a better compression ratio can sometimes be achieved by LZW than by Huffman coding because of this more realistic assumption. Here we used our own implementation of the original LZW algorithm.

GenCompress (GC). This was developed for the compression of large genomes by exploiting hidden regularities and properties in them, such as repetitions and mutations [101]. The main idea will now be described. First, let us consider a genome sequence $x = uv$ and suppose that its first portion u has already been compressed. Find the largest prefix v' of v with an edit distance method such that it is maximally partial

matched by a substring u' in u that has minimal edit operations needed to obtain v' from u'. After, only the position of u' and the edit operations list is encoded by Fibonacci coding. GenCompress is a one-pass algorithm and it is the state-of-the-art compression method for genomic sequences. However, it is not fast in practice because its performance is considered more important, so it is recommended for offline computations [101]. We downloaded this program from `http://www.cs.cityu.edu.hk/cssamk/gencomp/GenCompress1.htm`

Prediction by Partial Matching (PPM). PPM [102] is an adaptive statistical data compression technique that uses a context tree to record the frequency of characters within their contexts. Then it predicts the next symbol in the stream using Arithmetic Coding (AC) to encode an event with an interval that is assigned to the event w.r.t. their distribution, and the codeword is the shortest binary number taken from this interval. A more likely event has a correspondingly longer interval, and thus a shorter codeword can be selected [103]. In our experiments we used the implementation from `http://compression.ru/ds/`.

Burrows-Wheeler Transform (BWT). While BWT [104] is not a compressor method, it is closely related to data compression and is used as a pre-processing method to achieve a better compression ratio. It is a reversible block-sorting method that produces all n-cyclic shifts of the original sequence, then orders them lexicographically, and outputs the last column of the sorted table as well as the position of the original sequence in the sorted table [104]. This procedure brings groups of symbols with a similar context closer together and these segments can be used to better estimate the entropy of a Markov process [105]. The implementation that we used was downloaded from `http://www.geocities.com/imran_akthar/files/bwt_matlab_code.zip`.

Advanced Block-sorting Compressor (ABC). This compressor contains several advanced compression techniques like BWT, run-length encoding, AC, weighted frequency count and sorted inversion frequencies. For details, see [106]. The compression speed is approximately half that of GZIP and BZIP2. This program we used in our experiments was downloaded from `http://www.data-compression.info/ABC/`

5.4 Experiments and Discussion

In this section we will describe several experiments that were carried out to determine how CBDs can exploit any similarity among sequences, from different points of view.

5.4.1 The Effect of Alphabet Size

Here we examined the sensitivity of CBDs to sequence manipulation, such as alphabet reduction and expansion. To do this, we chose 3PGK dataset. The sequences are available both as amino acid residues (358 to 505 residues in length) and nucleotide residues (1074-1515 in length).

An alphabet reduction was evaluated just on amino acids in such a way that they were grouped into different groups and each amino acid residue was replaced by its

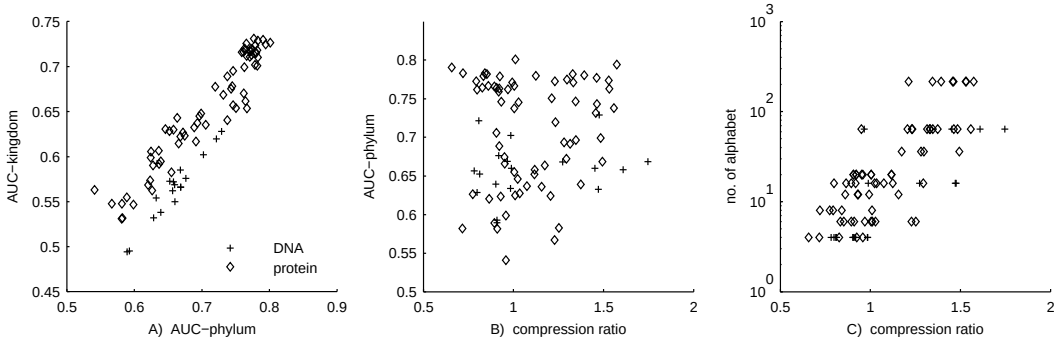


Figure 5.1: The correlation between methods on nucleotide (+) and amino acid ($\diamond$) sequences taken from 3PGK. (A) the relationship between the AUC-phylum and AUC-kingdom; (B) the correlation between compression ratio and AUC; (C) the connection between the compression ratio and the alphabet size.

Table 5.1: AUC results for amino acid sequences with an alphabet reduction and BWT acting on 3PGK.

	Original	SB16	SB12	SB8	SB6	SB4	Dayhoff6	BWT
ABC	.738	.766	.771	.783	<u>.801</u>	.764	.745	.655
AH	.658	.628	.636	.625	.567	.541	.583	.652
GC	.763	.759	.746	.773	.762	.779	.767	.689
LZW	.666	.620	.623	.626	.589	.582	.581	.599
PPM	.764	.762	.766	.783	.779	.790	.782	.706

The pairwise similarity/distance matrices were also calculated on the original amino acids sequences with naive distance and SW, and the AUC values we got were .685 and .797, respectively. The largest score is underlined.

group identifier. The Dayhoff classes ("AGPST, DENQ, HKR, ILMV, FWY, C") were obtained from [107] and here are referred to as Dayhoff6. Other classes were taken from Table 1 of [108] and are denoted by SB4, SB6, SB8, SB12, and SB16, respectively. The number in each class name denotes the number of amino acid clusters. The alphabet expansion was the residue composition; that is, each bi-gram and tri-gram was considered as a single letter yielding an alphabet with number of elements n^2 and n^3, respectively, where n is the number of a particular alphabet. In DNA sequences it provided an alphabet of 16 or 64 letters, respectively. In amino acid sequences, with a reduced alphabet, we applied this on alphabets SB4, SB6, SB8 and Dayhoff6 classes. The BWT transformation was also evaluated on the original amino acid and nucleotide sequences to see how well it supports compression performance. However, the GenCompress (GC) algorithm was not evaluated on sequences obtained by alphabet expansion because it was originally developed for genome sequences and not for strings of arbitrary characters and alphabets.

To evaluate how well a distance matrix reflects the groups, a ROC analysis [46] was used in the following way. The similarity of two proteins was used as the score of a binary classifier, putting them into the same group such as kingdom and phylum, denoted by AUC-kingdom and AUC-phylum, respectively. Here, a ROC analysis was performed on each distance matrix calculated by a CBD on each dataset constructed from the 3PGK by alphabet reduction, expansion and BWT, respectively. Fig. 5.1A

Table 5.2: AUC results for an alphabet expansion on 3PGK of a reduced alphabet.

	SB8		SB6			SB4			Dayhoff		
	A[1]	AA[2]	A[1]	AA[2]	AAA[3]	A[1]	AA[2]	AAA[3]	A[1]	AA[2]	AAA[3]
ABC	.783	.746	.801	.668	.794	.764	.672	.699	.745	.738	.774
AH	.625	.639	.567	.664	.746	.541	.646	.675	.583	.624	.750
LZW	.626	.696	.589	.693	.763	.582	.637	.720	.581	.691	.732
PPM	.783	.771	.779	.775	.777	.790	.780	.773	.782	.782	.780

[1]The values were taken from Table 5.1. [2]Bi-gram and [3]tri-gram residue compositions were used. Here, the GC compressor was not used. The largest score is underlined.

shows a high correlation between the AUC-kingdom and AUC-phylum, hence in the following just the AUC-phylum is shown. In Fig. 5.1B the correlation between the compression ratio and the AUC-phylum is plotted, where the compression ratio cr is defined by

$$cr = \frac{avg.uncompr.size}{avg.compr.size} \cdot \frac{\lceil \log_2(alphabetsize) \rceil}{\log_2(256)}. \tag{5.14}$$

Here $\lceil . \rceil$ stands for the larger integer. The second term is a scaling factor intended to provide a fair comparison for compressors with a different size of alphabet of 3PGK. After computing it, we found no apparent connection between the compression performance and the quality of CBDs (measured by AUC), but a logarithmic relationship was found between the compression ratio and the size of the alphabet (see Fig.5.1C). This suggests that there is no connection between the alphabet size and AUC, but in some cases an improvement can be attained. For example, the alphabet when reduced to 4-8 can improve the AUC values with some compressors, as Tables 5.1 illustrates. Tables 5.2 and 5.3 list results of expanded alphabets on reduced amino acids and of the original nucleotide sequences, with varying results. At this point it should be mentioned that the compressor of a reduced alphabet can be considered as a lossy compressor. The BWT method was also evaluated both on amino acids and nucleotide sequences, but it did not improve the AUC values possibly because the sequence lengths were too short.

Table 5.3: AUC results for an nucleotide sequences with an alphabet expansion on 3PGK.

	Original	AA[1]	AAA[2]	BWT
ABC	.676	.633	.668	.639
AH	.589	.660	.669	.593
GC	.702	n.a.	n.a.	.634
LZW	.652	.668	.660	.628
PPM	.722	.729	.658	.657

[1]Bi-gram and [2]tri-gram residue compositions were used. The pairwise distance/similarity matrices were also calculated on the original amino acids sequences via naive distance and SW, and the AUC values we got were .662 and .807, respectively. The largest score is underlined.

5.4.2 Protein Classification Results

To evaluate these CBDs for protein classification, we used the SCOP40 database. For a comparison we also employed the so-called naive distance that was simply the Euclidean distance of a vector bi-gram count of sequences and the results of the two alignment-based SW and BLAST methods were taken from Table 3.7. The results summarized in Table 5.4 reveal that CBDs perform better than the naive methods but perform less well than the alignment-based SW and BLAST with most classifier.

5.4.3 Combination of CBDs with BLAST in Protein Classification

We were curious to see whether CBDs could be used in a mixed-feature setting. In particular we were interested in finding out if a combination of the BLAST score and the compression-based measures could approach the performance of the SW algorithm. Although numerous combination schemes are available in the literature [109], here the process of combining of distance measures was carried out using the multiplication rule:

$$F(x,y) = \left(1 - \frac{BLAST(x,y)}{BLAST(x,x)}\right) \cdot CBD(x,y), \tag{5.15}$$

where $BLAST(x,y)$ is a BLAST score computed between a query x and a subject y and $BLAST(x,x)$ is the BLAST score of the query compared with itself. The term in parentheses is used to transform the BLAST score into a normalized distance measure whose value lies between zero and one. Eq.(5.15) is a straightforward method for combining CBDs with more specialized methods. The rationale behind it is that an application specific bias may improve the performance of the applied independent compression method. The performance of this combined measure (LZW-BLAST, PPMZ-BLAST) was in fact close to and, in some cases, even slightly superior to that of the SW algorithm. We consider this result encouraging since Eq. (5.15) does not contain any tuned parameter.

Table 5.5 and Fig 5.2 provide a comparison of the classification performance of this

Table 5.4: The overall AUC results on protein classification with several classifiers and feature methods applied on SCOP40.

Method name[1] (AUC[2])	1NN	SVM	RF	LogReg	ANN	Avg
ABC (.699)	.726	.843	.779	.806	.834	.798
AH (.690)	.711	.877	.824	.751	.800	.793
GC (.674)	.644	.775	.691	.753	.769	.726
LZW (.769)	.751	.856	.821	.718	.794	.788
PPM (.700)	.798	.851	.800	.813	.583	.823
naive (.597)	.653	.882	.848	.786	.837	.801
BLAST[3] (.684)	.775	.905	.697	.872	.799	.806
SW[3] (.823)	.815	.942	.823	.906	.888	.875

[1]Method used in the vectorization step. [2]The AUC values in parentheses were obtained via distance matrices in the way described in Section 5.4.1.[3] The values were taken from Table 3.7.

Table 5.5: The classification and class separation results of various proximity measures on protein domain sequences

Proximity measure	SVM[a]	1NN[a]	FSR[b]
SW p-value	0.901	0.930	1.629
BLAST	0.884	0.876	0.743
LZW	0.869	0.847	0.670
PPMZ	0.787	0.764	0.994
LZW + BLAST	0.907	0.688	0.751
PPMZ + BLAST	0.884	0.652	0.803
SVM-Fisher[c]	0.682	n.a.	n.a.
SAM[d]	0.657	n.a.	n.a.
PSI-BLAST[e]	0.589	n.a.	n.a.

[a]The performance of each classifier was measured by a ROC analysis. [b]Fisher Separation Ratio [31]. [c]SVM-Fisher denotes the method described in [28], [d]SAM denotes a profile-HMM classifier [110] while [e]PSI-BLAST represents the algorithm in [111]. The dataset was taken from [68].

mixed comparison method with SW, BLAST, LZW and PPMZ. Here we also plotted the performance of two methods based on Hidden Markov Models (HMMs). HMMs are currently the most popular tools for detecting remote protein homologies [112]. The SAM algorithm [110] is a profile-based HMM, while the SVM-Fisher method [28] couples an iterative HMM training scheme with an SVM-based classification and is reportedly one of the most accurate methods for detecting remote protein homologies. Here the performance of both HMM-based methods seems to fall short of the best-performing CBDs.

The Fig 5.3 illustrates the class separation of the 3PGK sequences. Here the visualization was performed by the LLE method [52] with pairwise proximity measures. In Table 5.5 these class separation were also given in quantitative terms calculated by the Fisher Separation Ratio (FSR) [31].

Even though the results may depend on the database used and should be confirmed by a more extensive statistical analysis, we find it encouraging that the combination of two, low time-complexity measures - CBD and a BLAST score - can compete with HMMs in terms of classification accuracy.

5.5 Conclusions

Our original reason for carrying out this study was to gain an insight into the behaviour of CBDs in protein classification tasks. In general, CBDs perform less well than substructure-based comparisons such as the SW algorithm in protein similarity. This is in fact expected, since SW calculations include a substantial amount of biological knowledge encoded in the amino acid substitution matrix. We also found that a combination of two, low time-complexity measures (BLAST score and CBDs) can approach or even exceed the classification performance of computationally intensive methods like the SW algorithm and HMM methods.

In our experiments, we were unable to find any statistical connection between the compression ratio of the sequences and the modularity expression (which was measured by AUC). Perhaps this was because the sequence lengths we examined were very short.

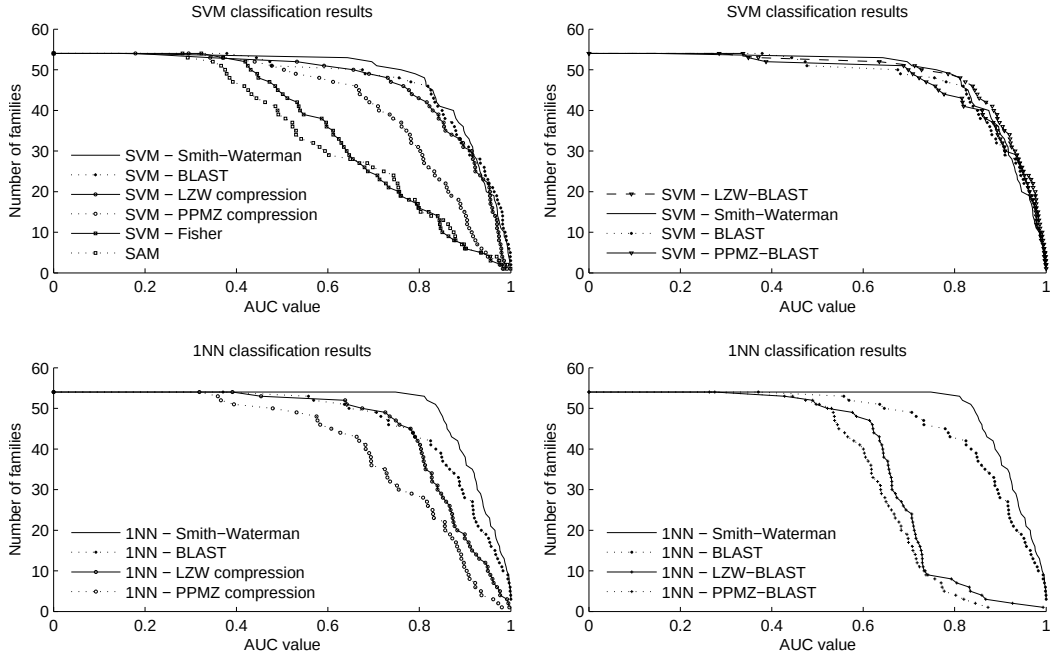


Figure 5.2: The relative classification performance of the various similarity and distance measures in SVM and 1NN classifiers, as determined on dataset taken from [68]. Each graph plots the total number of families where the AUC exceeds a score threshold indicated on the x-axis. A higher curve corresponds to more accurate classification performance. The data for SVM-Fisher [28] and SAM profile-HMM [110] were taken from [68]. Here the items in each legend were ordered by the area under the corresponding curve.

Moreover, the well-tested BWT method of data compression could not be used here. Similar findings were also described in [113].

In general, the results suggest that there is no monotone tendency as a function of the reduced size of the alphabet. The performance of the statistics-based AH compressor decreased when the alphabet was reduced, while with the partial matching-based algorithms like PPM and GC the performance improved. This could mean that mutations of amino acids found in the same cluster do not really change their structure and function(s). We should add that a compressor applied to reduced alphabet sequences can be viewed as a lossy compressor, and the selection the cluster of amino acids can represent a few biological knowledge in the technique. An alphabet expansion with bi-gram and tri-gram composition usually increases the performance of the statistical AH compressor, as in the expanded letter distribution the neighbours of the original letters are taken into account.

Overall, partial matching-based compressors (PPM, GC) seem to outperform the various types of compressors available, both in ROC analysis and protein classification.

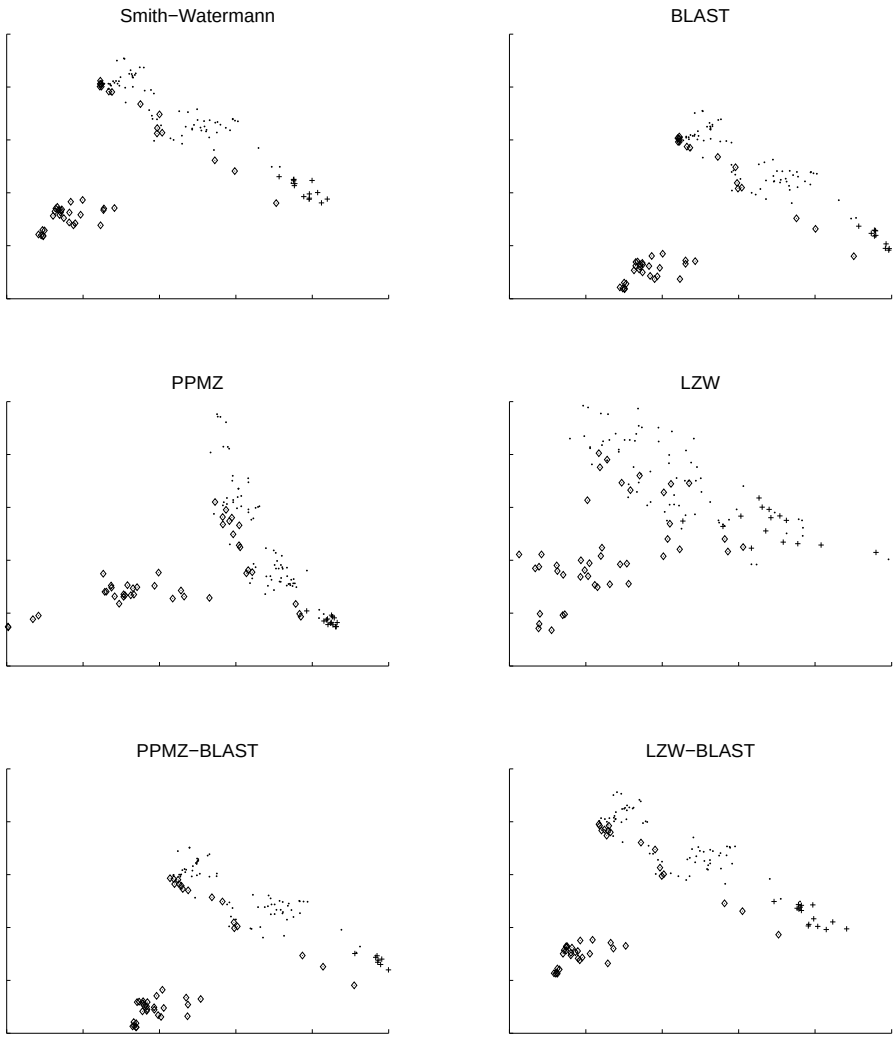


Figure 5.3: Separation of Eukaryotic (diamond), bacterial (dot) and archean (cross) 3PGK sequences. These figures were obtained by the LLE method [52].

Chapter 6

Equivalence Learning for Protein Classification

The learning of a similarity in a supervised manner could provide a general framework to set a similarity function to a specific sequences classes. Here we describe a novel method which learns a similarity function between proteins by using a binary classifier and pairs of equivalent sequences (belonging to the same class) as positive samples and non-equivalent sequences (belonging to different classes) as negative training samples. In this project the Author's contributions were the introduction of the notion of equivalence learning as a new way of carrying out similarity learning, which he later applied to to protein classification. He also developed a new class of kernel functions, namely the Support Vector Kernel (SVK), He theoretically proved that it is a valid kernel function, and He defined two new ways to learn SVK along with a new parameter setting technique. He designed and evaluated the experiments as well. In this project the Author's colleagues developed the similarity representation (see Section 6.2) and their critical reading of manuscripts and useful suggestion significantly improved the quality of papers.

6.1 Introduction

In protein classification the aim is to classify a protein s into its unique class; that is, $F(s) = y$, where y stands for the class label. This classification of proteins naturally determines the following equivalence relation

$$\delta(s,t) = \begin{cases} 1 & F(s) = F(t) \text{ i.e. } s \text{ and } t \text{ belong to the same class,} \\ 0 & \text{otherwise,} \end{cases}$$

which is reflexive, symmetric and transitive. A pair of sequences is called equivalent when both of them belong to the same sequence group and it is called non-equivalent when they do not; moreover the set of equivalent (resp. non-equivalent) pairs is called the positive (resp. negative) class. The learning of this function δ essentially becomes a two-class classification schemes, and hence two-class classification techniques can be

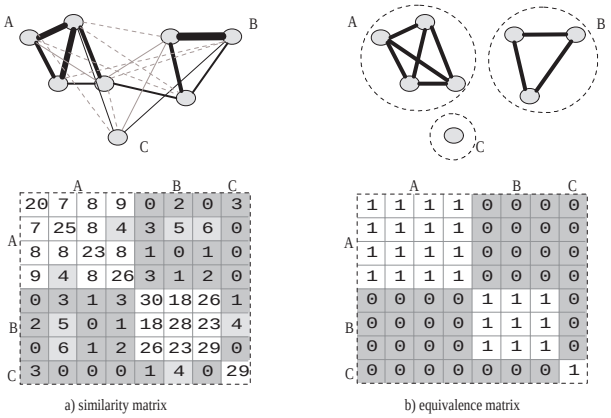


Figure 6.1: The Principle of Equivalence Learning I. A similarity matrix (left) can be determined by an all-vs.-all comparison of a database using algorithms such as BLAST and Smith-Waterman. The equivalence matrix (right) is a representation of groups (A,B) and an outlier (C) is identified by human experts.

applied. This approach is called *equivalence learning (EL)*.

For example, let us consider a database of objects and a similarity measure computed between each pair of objects. This arrangement can be visualized as a weighted graph (network) of similarities where the nodes are the proteins and the weighted edges represent the similarities between them. The network can also be represented by a symmetrical matrix in which the cells store pairwise comparison values between the objects. Fig. 6.1a shows a hypothetical database of 8 objects. We can vaguely recognize two groups where the members are more similar to each other than to the objects outside the groups. Let us now suppose that an expert looks at the similarity data and decides that the two groups represent two classes, A and B, and there is another object (C) that is not a member of either of these. Fig. 6.1b illustrates this new situation. The members of the groups are now connected by an equivalence relation that exists only between the members of a given group. As a result, the similarity matrix becomes a simpler equivalence matrix, where only the elements between the members of the same class are non-zero.

A key issue here is to decide how we should represent the similarity between proteins so that we can efficiently predict an equivalence. A simple numerical value is not sufficient so, instead, we will use a vectorial representation of the edges. Hence we will need a projection method $P : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}^n$ which captures the similarity between a sequence pair from a different point of view and stores them in vector form. In Section 6.2 we will describe vectorization methods which realize this projection mapping P using an advanced technique taken from fuzzy theory. Then the equivalence learning task can be reduced to the learning of the two-class classification problem $F(P(s,t)) \rightarrow \{0,1\}$, which can be carried out via a typical machine learning algorithm like ANNs. Instead of the predicted class label, just the score obtained from the learned model can be viewed as a learned similarity function. In Section 6.3 we will present the kind of properties required to have a valid learned metric or kernel function and we will present ways to

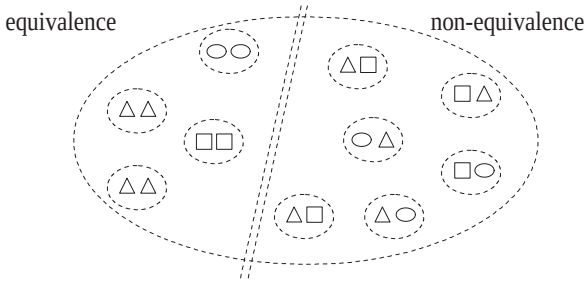


Figure 6.2: Principle of equivalence learning II. Equivalence learning is a two-class problem defined on object pairs.

learn them. In the last two chapters we will present and discuss the experimental results than draw some conclusions.

But before we start we will summarize the methods that have already been developed for distance and similarity learning.

6.1.1 Related Studies

The goal of the *distance metric learning (DML)* approach is to learn a Mahalanobis metric

$$\|x - y\|_M^2 = (x - y)^T M (x - y), \tag{6.1}$$

for n-dimensional real valued vectors x, y; that is, learn a positive semidefinite matrix M in such a way that it reduces the intra-class variability and increases the inter-class

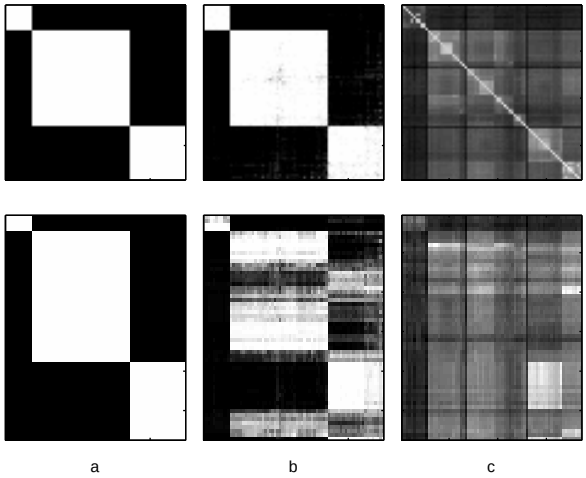


Figure 6.3: A heat map representation of the equivalence matrix (a) and equivalence matrix learned by RF (b) with the C_D composition (see Section 6.2) method on one of the classification tasks of 3PGK. The train matrices (above) were calculated in a pairwise manner between the train sequences, and the test matrices (below) were calculated between the train and test sequences in a pairwise manner as well. The Smith-Waterman similarity matrices (c) are shown here for comparison.

variability. Let us denote a set of labelled points by $L = \{(x_i, y_i) \mid x_i \in \mathbb{R}^n, y_i \in N\}$, where y_i stands for the class label and let S be a set of pairs of vectors whose class labels match and let D be a set of pairs whose class labels do not match. In [114] they propose a method to learn this Mahanalobis distance by solving the following optimization problem

$$\min_{M} \quad \sum_{(x,y)\in S} \|x-y\|_M^2 \tag{6.2}$$

$$s.t. \quad \sum_{(x,y)\in D} \|x-y\|_M^2 \geq 1 \tag{6.3}$$

$$M \text{ positive semidefinite.} \tag{6.4}$$

But this is a convex problem as the optimization involves diagonalization and eigen decomposition methods, hence the authors gave an iterative procedure using gradient descent and projection techniques.

Let us define the following positive definite matrices $K_{ij}^M = x_i' M x_j$ and

$$Y_{ij} = \begin{cases} 1 & y_i = y_j \text{ i.e. } x_i \text{ and } x_j \text{ belong to the same class,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

The matrix Y is also called the ideal kernel matrix and it was first defined in [115]. It can be viewed as an "oraculum" that tells us which object pairs belong to the same class. This ideal kernel matrix would be perfect for classification but it cannot be computed for the unlabelled part of the data set. In [116; 117] they suggested optimizing M such that the kernel matrix K^M is aligned to the ideal matrix Y on the training sample and relying on it will produce a better alignment on the unlabelled data.

Then, the goal is to find a matrix M with an adjustable parameter γ such that

$$\|x_i - x_j\|_M^2 = \begin{cases} \leq \|x_i - x_j\|^2 & (x_i, x_j) \in S \\ \geq \|x_i - x_j\|^2 + \gamma & (x_i, x_j) \in D. \end{cases} \tag{6.6}$$

The constrained problem that we get is

$$\min_{M,\gamma,\xi_{ij}} \quad \frac{1}{2}\|M\|_2^2 + \frac{C_S}{D_S} \sum_{(x_i,x_j)\in S} \xi_{ij} + \frac{C_D}{N_D} \sum_{(x_i,x_j)\in D} \xi_{ij} - C_D \nu \gamma \tag{6.7}$$

subject to the constraints

$$\|x_i - x_j\|^2 \geq \|x_i - x_j\|_M^2 - \xi_{ij} \qquad (x_i, x_j) \in S \tag{6.8}$$

$$\|x_i - x_j\|_M^2 \geq \|x_i - x_j\|^2 - \xi_{ij} + \gamma \qquad (x_i, x_j) \in D \tag{6.9}$$

$$\xi_{ij} \geq 0, \quad \gamma \geq 0. \tag{6.10}$$

Here N_S and N_D are the numbers of the pairs in S and D, respectively, and C_S, C_D

and ν are non-negative adjustable parameters. The solution can be written in the form

$$M = \sum_{(x_i,x_j)\in D} \alpha_{ij}(x_i - x_j)(x_i - x_j)^T - \sum_{(x_i,x_j)\in S} \alpha_{ij}(x_i - x_j)(x_i - x_j)^T, \quad (6.11)$$

where the α_{ij} elements are the Lagrangian multipliers. This method can be readily extended to kernel induced feature spaces. Let us suppose that $\kappa(.,.)$ represents a kernel function and let the corresponding map be denoted by $\phi(.)$, and let $\kappa_{ij} = \kappa(x_i, x_j) = \phi(x_i)^T\phi(x_j)$. Then we have

$$\begin{aligned} K(x_a, x_b) &= \phi(x_a)^T M \phi(x_b) & (6.12)\\ &= -\sum_{(x_i,x_j)\in S} \alpha_{ij}(\kappa_{ai} - \kappa_{aj})(\kappa_{bi} - \kappa_{jb}) & (6.13)\\ &+ \sum_{(x_i,x_j)\in D} \alpha_{ij}(\kappa_{ai} - \kappa_{aj})(\kappa_{bi} - \kappa_{jb}) \end{aligned}$$

In [118] they present an information-theoretic approach to learn a Mahalanobis distance function. It uses the fact there exists a simple bijection (up to a scaling function) between the set of Mahalanobis distances and the set of equal mean multivariate Gaussian distributions, then a given Mahalanobis distance can be expressed by its corresponding multivariate Gaussion as $p(x, M) = 1/Z \exp(-1/2\|x - \mu\|_M)$, where Z is a normalizing constant and μ is the mean of the Gaussian[118].

Next an alignment to a given Mahalanobis distance A_0 can be determined by the Kullback-Leibler divergence, as in the following:

$$KL(p(x, A_0)\|p(x, A)) = \int p(x, A_0) \log \frac{p(x, A_0)}{p(x, A)} dx. \quad (6.14)$$

Then the metric learning problem can be formulated as

$$\begin{aligned} \min_A \quad & KL(p(x, A_0)\|p(x, A)) & (6.15)\\ s.t. \quad & \|x_i, x_j\|_A^2 \le u \quad (x_i, x_j) \in S & (6.16)\\ & \|x_i, x_j\|_A^2 \ge l \quad (x_i, x_j) \in D. & (6.17) \end{aligned}$$

The objective can be expressed as a particular type of Bergman divergence, then one can apply Bergman's method to solve the above metric learning problem.

In [119] the authors propose a distance metric learning for kNN classification by optimizing k-nearest neighbours belonging to the same class, while examples from different classes are separated by a large margin.

However these learned Mahalanobis distance are still a linear feature weighting method for vectors.

In [120] they propose another way to learn a kernel matrix aligned to the ideal kernel

matrix. Let the kernel matrix be like so:

$$K = \begin{pmatrix} K_{tr} & K_{tr,te} \\ K_{tr,te}^T & K_{te} \end{pmatrix}, \tag{6.18}$$

where K_{tr} is the kernel matrix of the train data, K_{te} is the kernel matrix of the unlabelled data (i.e. test) and the $K_{tr,te}$ is the kernel matrix of the train and the test data. Let $\mathcal{K} = \{K \succeq 0\}$ denote the cone of positive definite kernel matrices. The kernel matrix which is maximally aligned with the ideal matrix can be found by solving the following optimization problem:

$$\begin{aligned} \max_K \quad & \langle K_{tr}, Y \rangle_F & (6.19)\\ s.t. \quad & \mathsf{Tr}(K) = 1, & (6.20)\\ & K \in \mathcal{K}, \quad \|K\|_2^2 = 1, & (6.21) \end{aligned}$$

where $\langle ., . \rangle_F$ denotes the Frobenius norm of matrices defined by $\langle A, B \rangle_F = \sum_{i,j} A_{ij} B_{ij}$, $\|.\|$ is the matrix norm defined by $\|A\| = \sqrt{\langle A, A \rangle}$ and $\mathsf{Tr}(A) = \sum_i A_{ii}$ is the trace of the matrix A.

The *Multiple Kernel Learning (MKL)* approach is dedicated to learning the optimal linear combination of some initial kernels. Let the set $\{K_1, K_2, \ldots, K_l\}$ be the initial kernel matrix. Here the aim is to determine a set of coefficients such that $\sum_{i=1}^{l} \mu_i K_i$ provide a better classification performance under certain conditions. Adding this to the former problem, we get

$$\begin{aligned} \max_K \quad & \langle K_{tr}, Y \rangle_F & (6.22)\\ s.t. \quad & K = \sum \mu_i K_i, & (6.23)\\ & K \in \mathcal{K}, \quad \|K\|_2^2 = 1. & (6.24) \end{aligned}$$

In [120] they present a way of learning kernel matrices such that it maximizes the margin of the separation hyperplane on the training data both with a hard margin and a soft margin classification. The generalized performance measure of the kernel matrix on the training data is defined by

$$\omega_{C,\tau(K)} = \max_{\alpha} 2\|\alpha\|_1 - \alpha'(G(K) + \tau I)\alpha, \tag{6.25}$$

where $0 \leq \alpha \leq C$ and $\alpha' y = 0$. In some convex subset of $\mathcal{K}$ of positive seimidefinite matrices with trace equal to c,

$$\begin{aligned} \min_{K \in \mathcal{K}} \quad & \omega_{C,\tau}(K_{tr}) & (6.26)\\ s.t \quad & \mathsf{Tr}(K) = c, & (6.27) \end{aligned}$$

can be realized in a semidefinite programming technique which is a subfield of convex

optimization concerned with the optimization of a linear objective function over the intersection of the cone of positive semidefinite matrices with an affine space, and it can be effectively solved by interior point methods [120]. Minimizing $\omega_{C,\tau}(K)$ constrained to the linear subspace $K = \sum \mu_i K_i$ with $\mu \geq 0$ leads to a quadratically constrained quadratic programming (QCQP) problem.

In [121] the authors propose a method (called mSVM) to optimize a linear combination of kernels in a direct way for SVM classification. This is indeed a direct sum of feature spaces and can be formulated in the following way:

$$f_{w,b,\beta}(x) = \sum\nolimits_{k=1}^{l} \beta_l \langle w_k, \phi_k(x) \rangle + b, \tag{6.28}$$

where β is a standard simplex $\beta \in \Delta = \{\beta \mid \|\beta\|_1 = 1, \quad \beta \geq 0\} \subset \mathbb{R}^l$. Thus the resulting optimization problem can be written as

$$\min_{\beta,w,b,\xi} \quad \sum_{k=1}^{l} \beta_k \|w_k\|_2^2 + C \sum_{i=1}^{n} \xi_i \tag{6.29}$$

$$s.t. \quad \xi_i = l(y_i f_{w,b,\beta}(x_i)), \tag{6.30}$$

where $l(z) = max(0, 1 - z)$ is the hinge loss function. Unfortunately, this is not a convex optimization problem, but using a variable transformation with $v_i = \beta_k w_k$ and using a quadratic loss function we get the following convex problem:

$$\inf_{\beta,w,b,\xi} \quad \sum_{k=1}^{l} \|v_k\|_2^2 / \beta_k + C \sum_{i=1}^{n} \xi_i \tag{6.31}$$

$$s.t. \quad \xi_i = l\left(y_i \left(\sum\nolimits_k v_k' \phi_k(x_i) + b\right)\right). \tag{6.32}$$

The Author of [122] gave a new form of regularization that allows one to exploit the geometry of the marginal distribution. They focused on a semi-supervised framework that incorporates labelled and unlabelled data in a general-purpose learner. In [123] they proposed a method for modifying a kernel function to improve the performance of SVMs via a Riemannian geometrical structure induced by the kernel function. The idea is to enlarge the spatial resolution around the separation boundary surface by a conformal mapping such that the separation between classes is maximised.

In bioinformatics the DML approach is employed for the reconstruction of biological networks e.g. predicting interactions between genes. In [66] they proposed a new kernel function for biological network inference by tuning the results of [116; 124]. In [125] the author proposed a method for predicting the absence or presence of an edge in a biological network using the Laplacian matrix induced by the graph. [126] employed a variant of kernel canonical correlation analysis to identity correlations between heterogenous datasets, especially that between a protein network and other genomic attributes. And in [127] they proposed learning a string similarity function for gene names using Logistic Regression via a linear weighted combination of a set of standard string similarity methods like MKL.

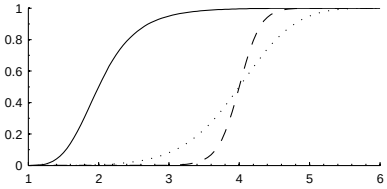


Figure 6.4: κ functions with different parameter values. Here $n = 6$, ($\mu_1 = 2$, $\lambda_2 = -3$, solid line), ($\mu_2 = 4, \lambda_2 = -3$, dashed), ($\mu_3 = 4, \lambda_3 = -8$, dotted).

The alignment-based similarity measures for global alignment (like NW) and local alignment (like SW) strongly rely on the substitution matrix. In [128] they developed an approach, called OPTIMA, which seeks to create an appropriate substitution matrix using homologous and non-homologous sequences and an optimization procedure based on the steepest-descent approach. In [129] the authors presented an approach to learn a substitution matrix but it based on LAK, (see Chapter 2.1), using the fact LAK is differentiable with respect to the amino acid substitution and its derivative can be computed efficiently by dynamic programming. The optimized substitution matrix is determined by classical gradient descent and setting an objective function that measures how well the local alignment kernel discriminates homologs from non-homologs in the COG database.

But to the best of our knowledge there has been no study of similarity learning applied to protein classification tasks.

6.2 Vectorization Step

Now we will show how to define projection functions which map any sequence pair into a fixed-length vector of real numbers; that is, $P : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^n$. First, we will define a method for representing a sequence in vector form.

Let us consider a set of sequences $\{f_1, f_2, \ldots, f_n\} \subseteq \mathcal{S}$ as a feature set in a fixed ordering and let sf be an arbitrary similarity function. For a sequence $s \in \mathcal{S}$ let the corresponding vector w be a ranking vector which tells us how s orders the members of the feature set via the given similarity function sf. Thus w_i (the ith feature of w) means the order index of the ith feature sequence in the ranking with respect to s and sf.

Then for each component of the vector w the function κ is used for normalization, which is defined by the following way:

$$\kappa_{\lambda,\mu}(x) = \frac{1}{1 + (\frac{\mu}{n-\mu} \cdot \frac{n-x}{x})^{-\lambda}}.$$

κ was first introduced in [130]. This function is very useful in fuzzy theory for modelling the membership function, but it can also be used as a probability distribution function. It is useful for normalizing functions whose domain is bounded.

This is an increasingly monotone function and maps order numbers from $[1, n] \subseteq \mathbb{R}$

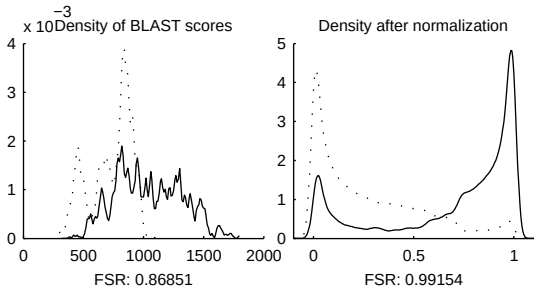


Figure 6.5: The density of scores for equivalent (solid line) and nonequivalent (dotted line) pairs, which are obtained by BLAST (left) and after normalization by κ (right). Below each figure the Fisher Separation Ratio (FSR) is shown.

onto the interval $[0, 1]$. The shape of this function κ is similar to a sigmoid function, but the parameter $\mu \in (1, n)$ regulates the location of the inflection point while λ regulates its gradient at the point μ, and $\kappa_{\lambda,\mu}(\mu) = 0.5$. The protein sequence vectorization method we apply will be denoted by $\phi^{\kappa}_{sf} : S \to R^n$, where sf is a similarity function used for ranking and each component of the vector we get is normalized by using κ. It should be mentioned here that each vector component lies between 0 and 1 and that each vector has the same (but not unit) length.

The next step is to form a vector from two vectors and we will do this componentwise. The widely-used Dombi operator class [131] can be used to describe a multivalued logic function like

$$o^{\alpha}(x, y) = \frac{1}{1 + \left((\frac{1-x}{x})^{\alpha} + (\frac{1-y}{y})^{\alpha} \right)^{1/\alpha}}.$$

If $\alpha > 0$, then o^{α} is a conjunctive operator, and if $\alpha < 0$, then o^{α} is a disjunctive operator. The most common case is $\alpha = \pm 1$. The equivalence relation in logic is defined by

$$x \equiv y = (\bar{x} \vee y) \wedge (x \vee \bar{y}),$$

where $\bar{x}$ is the negation and $\bar{x} = 1 - x$. Using the Dombi operator class we get

$$e(x, y) = xy + (1 - x)(1 - y).$$

Two important requirements are valid in classical logic. These are that

$$x \equiv x = 1 \quad x \equiv \bar{x} = 0.$$

Most papers concentrate on the first case and wish to find an equivalence in $e(x, x) = 1$ and ignore $e(x, n(x)) = 0$. In many-valued logic both conditions cannot be simultaneously valid. If we interpret the x values as a certainty then if $x = \frac{1}{2}$ and $y = \frac{1}{2}$, the

Table 6.1: Summary of the vector composition methods we applied.

Name	Formula			Description
Sum	$C_+(u,v)$	$=$	$u+v$	sum of the vector comp.
Product	$C_\bullet(u,v)$	$=$	$u \cdot v$	product of the vector comp.
Quadratic	$C_Q(u,v)$	$=$	$(u-v)^2$	quadratic difference
Hellinger	$C_H(u,v)$	$=$	$(\sqrt{u}-\sqrt{v})^2$	a normalized quadratic dist.
Dombi	$C_D(u,v)$	$=$	$u \cdot v + (1-u)\cdot(1-v)$	Dombi operator
Conjuct.	$C_C(u,v)$	$=$	$1/(1+(e^{(u-v)}+e^{(v-u)})/2)$	mean conjuctive op.

result of equivalence is also $\frac{1}{2}$, and the resulting $e(x,y)$ has the following properties:

$$e(0,0) = e(1,1) = 1 \quad e(0,1) = e(1,0) = 0 \quad e(\frac{1}{2},\frac{1}{2}) = \frac{1}{2}$$

The *mean conjunctive operator* of the Dombi operator class is

$$c(x,y) = \frac{1}{1+\frac{1}{2}\left(\frac{1-x}{x}+\frac{1-y}{y}\right)} \tag{6.33}$$

This operator class is used with the sigmoid function, that is

$$x = \frac{1}{1+e^{-\lambda(u-v)}} \qquad y = \frac{1}{1+e^{-\lambda(v-u)}}$$

x (and y) can be interpretated as the degree of the preference of u over v. The exponential composition method can be given that the (degree of u greater than v) and (degree of v greater than u), where for and we use 6.33. For $\lambda = 1$ we get

$$c_E(u,v) = \frac{1}{1+\frac{1}{2}\left(e^{(u-v)}+e^{(v-u)}\right)}.$$

Table 6.1 summarizes the methods which were used in our experiments. Here, for ease of notation, the operators were defined on vectors in a coordinate-wise manner, i.e. for any vector $u, v \in \mathbb{R}^n$, $(u \cdot v)_i = u_i v_i$, $(\sqrt{v})_i = \sqrt{v_i}$ and $(v^n)_i = (v_i)^n$.

Now we will use the notation $P_C^V : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_+^n$ to stand for a projection function which maps any sequence pair into an n-dimensional vector space. The superscript V denotes the vectorization method for both sequences and the subscript C defines the vector composition method. For example, if the Smith-Waterman (SW) similarity method is used as a ranking function to vectorize a sequence with normalization κ, and composition method $C_\bullet$ is used, then the projection function we get will be denoted by $P_\bullet^{SW}(x,y) = C_\bullet(\phi_{SW}^\kappa(x), \phi_{SW}^\kappa(y))$.

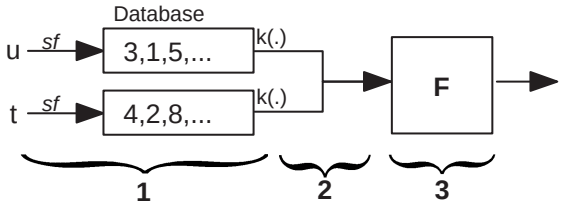


Figure 6.6: A diagram of the equivalence learning task. The first step is the vectorization of sequences u and t carried out by ranking of the database via a sequence similarity measure sf. Then the non-linear normalization function κ is employed. The second step is the combination of the two vectors with a particular method from Table 6.1. The third and last step is the application of a learned model that returns a score for an input sequence pair which can be interpreted as a degree of the similarity.

6.3 Learned Kernel Functions

Here we define a class of kernel functions and we present two ways to learn it.

Lemma 6.3.1 *Let θ be a positive-valued n-dimensional vector and let $P^{\phi}_{C} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^{n}_{+}$ be a symmetric projection method where ϕ is an arbitrary positive-valued vectorization method. The following functions*

$$\langle P^{\phi}_{\bullet}(s,t), \theta \rangle \tag{6.34}$$
$$\langle P^{\phi}_{D}(s,t), \theta \rangle \tag{6.35}$$
$$\exp(\sigma \langle P^{\phi}_{+}(s,t), \theta \rangle) \tag{6.36}$$

are positive definite kernel functions, where $\sigma \in \mathbb{R}_{+}$, and

$$\langle P^{\phi}_{Q}(s,t), \theta \rangle \tag{6.37}$$
$$\langle P^{\phi}_{H}(s,t), \theta \rangle \quad \textit{and} \tag{6.38}$$

are conditionally negative definite kernel functions.

The proof is straightforward but not obvious because in the inner product the two variables s, t vary in the first argument, while the second argument is constant. For more details about kernel functions and their properties, the reader should read [22].

Proof
Let $\theta \in \mathbb{R}^{n}_{+}$ be a vector. Then $\langle P^{\phi}_{\bullet}(s,t), \theta \rangle$ is a weighted scalar product, weighted in the following way

$$\langle P^{\phi}_{\bullet}(s,t), \theta \rangle \;\; = \;\; \langle \phi(s) \cdot \phi(t), \theta \rangle = \phi(s)^{T} \Theta \phi(t),$$

where Θ is a diagonal positive-valued matrix (which is always a positive definite matrix) whose diagonal elements are taken from θ. Thus it is a positive definite kernel function.

The function (6.35) can be expressed in the following way

$$\langle P_D^{\phi}(s,t),\theta\rangle = \underbrace{\langle \phi(s)\cdot\phi(t),\theta\rangle}_{\kappa_1} + \underbrace{\langle (1-\phi(s))\cdot(1-\phi(t)),\theta\rangle}_{\kappa_2}. \tag{6.39}$$

Here κ_1 is a positive definite kernel function since it is the same as the kernel in (6.34). Now we will prove that κ_2 is a kernel function. Exploiting the definition of a positive definite kernel we have to show that

$$\sum_{i,j=1}^{n} c_i c_j \kappa_2(s_i,s_j) \geq 0, \tag{6.40}$$

for any $n \in \mathbb{N}$, $c_1,\ldots,c_n \in \mathbb{R}$ and sequences $s_1,\ldots,s_n \in \mathcal{S}$. Using linear algebra techniques we get

$$\begin{aligned}
\sum_{i,j=1}^{n} c_i c_j \kappa_2(s_i,s_j) &= \sum_{i,j=1}^{n} c_i c_j \langle (1-\phi(s_i))\cdot(1-\phi(s_j)),\theta\rangle \\
&= \sum_{i,j=1}^{n} c_i c_j (1-\phi(s_i))^T \Theta (1-\phi(s_j)) \\
&= \left(\sum_{i=1}^{n} c_i \sqrt{\theta}(1-\phi(s_i))\right)^2 \geq 0.
\end{aligned}$$

Thus, κ_2 is a positive definite kernel function. Noting the fact the class of positive definite kernels is closed under addition we conclude the function (6.35) is a positive definite kernel.
The function (6.36) is also a positive definite kernel which follows immediately from

$$\begin{aligned}
\exp(\langle P_+^{\phi}(s,t),\theta\rangle) &= \exp(\langle \phi(s)+\phi(t),\theta\rangle) = \exp(\langle \phi(s),\theta\rangle + \langle \phi(t),\theta\rangle) = \\
&= \exp(\langle \phi(s),\theta\rangle)\exp(\langle \phi(t),\theta\rangle)
\end{aligned}$$

and from the Proposition 2.3.
The function $\langle P_Q^{\phi}(s,t),\theta\rangle$ is a quadratic Euclidean metric weighted by θ, that is

$$\begin{aligned}
\langle P_Q^{\phi}(s,t),\theta\rangle &= \langle \phi^2(s)+\phi^2(t)-2\phi(s)\cdot\phi(t),\theta\rangle \\
&= \langle \phi^2(s),\theta\rangle + \langle \phi^2(t),\theta\rangle - 2\langle \phi^2(s)\cdot\phi^2(t),\theta\rangle \\
&= \phi(s)^T\Theta\phi(s) + \phi(t)^T\Theta\phi(t) - 2\phi(s)^T\Theta\phi(t).
\end{aligned}$$

This is a conditionally negative definite function because for any $n \in \mathbb{N}$, $c_1,\ldots,c_n \in \mathbb{R}$, $\sum_{i=1}^{n} c_i = 0$ and sequences $s_1,\ldots,s_n \in \mathcal{S}$ we have

$$\sum_{i,j=1}^{n} c_i c_j \langle P_Q^{\phi}(s_i,s_j),\theta\rangle = \sum_{ij=1}^{n} c_i c_j \left(\phi(s_i)^T\Theta\phi(s_i) + \phi(s_j)^T\Theta\phi(s_j) - 2\phi(s_i)^T\Theta\phi(s_j)\right)$$

$$
\begin{aligned}
= & \underbrace{\left(\sum_{j=1}^{n} c_j\right)}_{=0} \left(\sum_{i=1}^{n} c_i \left(\phi(s_i)^T \Theta \phi(s_i)\right)\right) \\
& + \underbrace{\left(\sum_{i=1}^{n} c_i\right)}_{=0} \left(\sum_{j=1}^{n} c_j \left(\phi(s_j)^T \Theta \phi(s_j)\right)\right) \\
& \underbrace{-2 \sum_{ij=1}^{n} c_i c_j \left(\phi(s_i)^T \Theta \phi(s_j)\right)}_{\geq 0} \leq 0,
\end{aligned}
$$

which proves the original assertion. Likewise the function $\langle P_H^{\phi}(s,t), \theta \rangle$ is a conditionally negative definite kernel, and this can be proved in a similar way. □

Remark 6.3.1
We note that $\langle P_Q^{\phi}(s,t), \theta \rangle$ and $\langle P_H^{\phi}(s,t), \theta \rangle$ are both conditionally negative definite and vanish when $s = t$; thus they are valid metric functions and obey the triangle inequality.

Remark 6.3.2
We note that $\langle P_Q^{\phi}(s,t), \theta \rangle$ and $\langle P_H^{\phi}(s,t), \theta \rangle$ are conditionally negative definite kernels, hence $\exp(-\sigma \langle P_Q^{\phi}(s,t), \theta \rangle)$ and $\exp(-\sigma \langle P_H^{\phi}(s,t), \theta \rangle)$ are positive definite kernels for any $\gamma \in \mathbb{R}_+$ by Proposition 2.4.

The SVM approach provides a decision boundary

$$f(z) = \langle w, z \rangle + b$$

between two classes such that the margin associated with w is a maximum [63]. It is well known that the norm vector w can be expressed as a weighted linear combination of support vectors x_i; that is, $w = \sum_i \alpha_i x_i$, where α_i is the so-called Lagrangian multiplier corresponding to the support vector x_i and its sign corresponds to the class of support vectors i.e. it is negative (resp. positive) if x_i corresponds to the negative (resp. positive) class. Thus the decision boundary we get can be expressed by

$$f(z) = \sum_i \alpha_i \langle z, x_i \rangle + b.$$

The Gaussian RBF kernel $\kappa(z, x_i) = exp(-\sigma \|z - x_i\|)$ can be interpreted as a "hyper ball" around a support vector x_i, where σ regulates the radius, and it measures how close a sample z is to the support vector x_i. Replacing the inner product by the Gaussian RBF kernel in $f(z)$, the decision boundary can be viewed as a weighted sum of how well the sample z corresponds to each support vector. During the SVM training phase the parameters of this function are learned in such a way that it separates the two classes with the largest, possibly non-linear margin.

Using the former inner products in the SVM's decision boundary, we get the following functions:

$$SVK_P(s,t) = \sum\nolimits_i \alpha_i \exp(\sigma\langle P(s,t), x_i\rangle) \tag{6.41}$$

over $\mathcal{S} \times \mathcal{S}$, where $\sigma > 0$ and P stands for P_D^ϕ, $P_\bullet^\phi$, P_+^ϕ, $-P_Q^\phi$ and $-P_H^\phi$, respectively, where $-P$ is the multiplication of P by the scalar -1. The x_is are called support vectors and they are either a vectorized equivalent pair or non-equivalent pair. The condition for Eq. (6.41) to be a valid kernel function is that the α_i coefficients have to be positive-valued because the class of kernel function is closed under the direct sum and positive scalar multiplication. The class of such kernel functions is called the *Support Vector Kernel (SVK)*.

To learn this sort of function the One-Class SVM can be used, but only one of the classes can be used otherwise a negative Lagrangian multiplier would be occurred and then the function we get would not be a kernel anymore. Now we will present another way where both the equivalence and the non-equivalence sequences can be applied. First let

$$L = \{(s_i, t_i) \mid i = 1 \ldots l\}$$

be the training set containing both equivalent and non-equivalent pairs, let a randomly chosen small part of L be the set of support vectors denoted by

$$SV = \{x_j = P(s_{i_j}, t_{i_j}) \mid j = 1 \ldots k \leq l,\ (s_{i_j}, t_{i_j}) \in L\},$$

and let

$$A_{ij} = exp(\sigma\langle P(s_i, t_i), x_j\rangle),$$

where P and σ represent one of the SVK types and $x_i \in SV$. Then the α_i parameters in the SVK function can be learned by solving the following linear equation system with non-negative Least-Square optimization:

$$\begin{aligned} A_{11}\alpha_1 + A_{12}\alpha_2 + \ldots + A_{1k}\alpha_k &= \delta(s_1, t_1) \\ A_{21}\alpha_1 + A_{22}\alpha_2 + \ldots + A_{2k}\alpha_k &= \delta(s_2, t_2) \\ &\vdots \\ A_{l1}\alpha_1 + A_{l2}\alpha_2 + \ldots + A_{lk}\alpha_k &= \delta(s_l, t_l) \end{aligned}$$

We should mention that if the number of equations is equal to the number of variables α_i, this system may become numerically unstable and hard to solve. In our experiments we used half of the training set as support vector data.

About the σ parameter. In (6.41) the parameter σ that we used was a feature weighting technique; that is,

$$\langle P(s,t), \theta\rangle_\Lambda = P(s,t)\Lambda\theta,$$

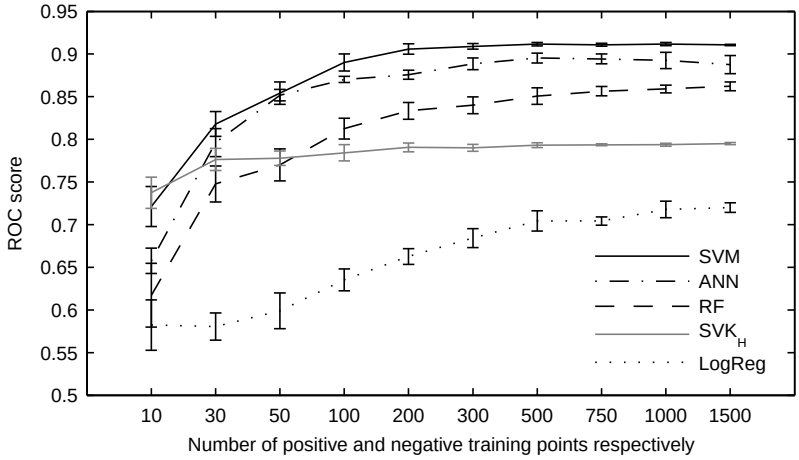


Figure 6.7: Dependence of the equivalence learning results on the train set size. For the projection method P_{+}^{SW} was used, but in the SVK_H case just P_H^{SW} was applied.

where Λ is a positive-valued diagonal matrix whose elements were found via Fisher Linear Discriminant (FLD) analysis [31]. The vector $w = S^{-1}(m_1 - m_2)$, where S is the within scatter matrix and m_i is the mean vector of the class i, obtained by FLD, normalized to the unit length, and the absolute value of each component was used to form the diagonal matrix Λ. In Λ, each ith component denotes the magnitude of the ith dimension in the class separation.

6.4 Results and Discussion

In each classification task the train set L of the equivalence and non-equivalence pairs was a small, randomly selected part of the full train sequences. This step is necessary in order to avoid overlearning, to speed up the training and to reduce the training set to a computationally manageable size because the training pairs grow quadratically w.r.t the number of train sequences. In order to select the best number of training pairs we calculated the learned similarity function by varying the size of datasets and repeated the procedure 10 times (see Fig. 6.7). We may conclude here that the standard deviation is generally small and increasing the training points only makes it smaller. We should mention here that a reasonable choice of number of training points depends on the variability of the training set; protein groups in real-life databases are known to be vastly different in the number of members, in average protein size, similarity within group and so on. In our experiments 500 positive and negative training pairs were used for learning, respectively.

The vectorization step for the ranking of the feature set was carried out by the BLAST method. For the parameter setting of the normalization function $\kappa_{\lambda,\mu}$ in our experiments we found that the best results were obtained when μ was set to the number of the non-equivalence pairs and λ was set to the tangent at the point μ, that is $\lambda = 1/(\mu/n - 1)$, where n is the number of features (data not shown). This means that the point where the function κ takes the value of 0.5 is the ratio of number

Table 6.2: The Fisher Separation Ratio (FSR) with different composition methods applied on the original data and on the normalized training data.

FSR	C_+	$C_\bullet$	C_Q	C_H	C_D	C_C
original	0.055	0.073	0.087	0.136	n.a.[1]	0.034
normalized	0.067	0.136	0.391	0.35	<u>0.392</u>	0.146

[1]The Dombi operator (C_D) is only defined in the [0,1] interval.

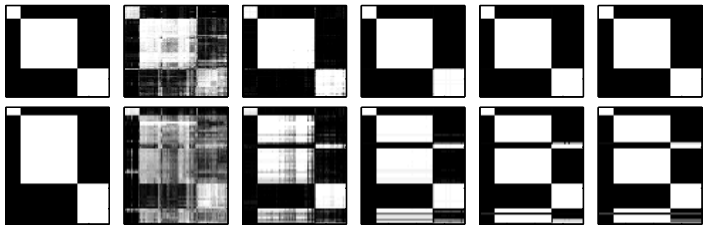


Figure 6.8: An iteration of ideal similarity learning by RF with P_+^{SW}. The leftmost heat map pair is the target equivalence matrix which was calculated by using class labels.

of the non-equivalence pairs within every sequence pair. After, a vector composition method was applied. In order to see how well the vectors of the equivalence and the non-equivalence sequence pairs were separated from each other the FSR method was applied. Moreover for a comparison of how the normalization aids the separation, the FSR method was also evaluated on the original data; that is, in the vectorization step just the similarity score was used instead of the normalized order number. The results are shown in Table 6.2.

Overall we may conclude that the normalization by κ makes the classes more separated, it makes equivalence learning more robust and less sensitive to different composition and learning methods, and it helps give a good performance scores (data not shown).

The EL obtained by the machine learning algorithms can be used as an underlying similarity function in the pairwise vectorization approach. This step can be repeated in an iterative fashion. Here, as shown in Figure 6.8, the results become stable after 3-4 steps. Our empirical test told us that the trained similarity matrices (Figure 6.8, top) really converge to the ideal equivalence matrix but the test similarity matrix (bottom)

Table 6.3: The AUC results of equivalence learning using different composition and learning methods.

	C_+	$C_\bullet$	C_Q	C_H	C_D	C_E	*avg*
SVM	0.832	0.781	0.873	0.848	0.857	0.861	*0.842*
ANN	0.797	0.804	0.873	0.860	0.873	0.858	*0.844*
LogReg	0.721	0.781	0.860	0.828	0.875	0.845	*0.818*
RF	0.847	0.841	0.880	0.874	0.878	<u>0.905</u>	*0.871*
avg	*0.799*	*0.802*	*0.872*	*0.853*	*0.871*	*0.867*	*0.844*

The AUC value of the BLAST similarity method is 0.737, which measures how well it can express an equivalence. The largest value is underlined here.

kept some of its original mistakes, and during the iteration process these errors became more pronounced.

On the 3PGK dataset, the AUC values for the results of equivalence learning with different learning and composition methods are shown in Table 6.3. The AUC value based on the original BLAST similarity matrix is 0.737 and it rose to 0.905 with the RF learner and the C_C composition method. On average it rose to 0.815. In general the composition methods $C_\bullet$ and C_+ do not perform as well as the others, and for equivalence learning the LogReg method performed worse than the others. On average, we obtained the best AUC results with the RF learner and C_D composition method. On the COG database we got similar results and trends (data not shown).

6.4.1 Classification Results with EL

Here we evaluated equivalence learning in the context of protein classification. In the case of 1NN the classification was performed by simply the score obtained by EL. With the other vector-based learning methods (SVM, RF, ANN and LogReg), the ith vector component for a sequence s was the learned equivalence scores between s and the ith training sequence. To evaluate the classification performance we used ROC analysis and the ranking variable was the score obtained by the given learned model.

The results of the ROC analysis are shown in Table 6.4. These results suggest that equivalence learning makes the classification easier; that is, better classification results can be achieved with a simpler method. For example, the AUC of 1NN improved from 0.863 to 0.964 with the equivalence matrix learned by RF and the Dombi operator on the dataset 3PGK.

The accuracy scores were also calculated and the accuracies obtained are given in Table 6.5. In both the 1NN and LRA case for classification the decision boundary we used was the threshold which gives the highest accuracy score on the train data. Here the best classification result is 0.876 achieved using RF with the C_H composition method when the SVM was used for classification. We think that ROC analysis provides

Table 6.4: The AUC results for protein classification on 3PGK.

CM[1]	EL[2]	C_+	$C_\bullet$	C_Q	C_H	C_D	C_C
1NN	SVM	0.873	0.910	0.946	0.940	<u>0.956</u>	0.899
(0.863)	RF	0.955	0.949	0.960	0.953	<u>0.964</u>	0.937
LRA	SVM	0.859	0.891	0.954	0.952	0.950	<u>0.971</u>
(0.941)	RF	0.945	0.943	<u>0.956</u>	0.950	0.956	0.943
RF	SVM	0.918	0.910	0.929	<u>0.944</u>	0.923	0.934
(0.852)	RF	0.944	0.939	0.948	0.941	0.948	<u>0.952</u>
SVM	SVM	0.946	0.957	0.956	<u>0.959</u>	0.944	0.803
(0.953)	RF	0.957	0.953	0.964	0.962	<u>0.966</u>	0.962

[1]The classification method applied. In parentheses we presented the base-line classification results for a comparison, where the BLAST method was used in the EFM. [2]The learning method was used to learn the equivalence. In each row the largest value is underlined. The LRA method is described in Section 4.

Table 6.5: The accuracy scores in protein classification.

CM[1]	EL[2]	C_+	$C_\bullet$	C_Q	C_H	C_D	C_C
1NN	SVM	0.822	0.829	0.840	<u>0.842</u>	0.832	0.697
(0.778)	RF	0.851	0.799	0.837	0.846	0.846	<u>0.860</u>
LRA	SVM	0.744	0.778	<u>0.836</u>	0.813	0.839	0.831
(0.872)	RF	0.833	0.808	0.828	0.820	<u>0.844</u>	0.840
RF	SVM	0.807	0.811	0.846	<u>0.876</u>	0.872	0.819
(0.860)	RF	0.813	0.775	0.838	0.844	0.831	<u>0.858</u>
SVM	SVM	0.801	0.813	0.835	0.834	0.826	<u>0.872</u>
(0.834)	RF	0.830	0.792	0.850	0.852	0.851	<u>0.860</u>

[1]The classification method used. In parentheses we presented the base-line classification results for a comparison, where the BLAST method was used in the EFM. [2]The learning method used to learn the equivalence. In each row the largest value is underlined.

a more robust analysis of a learning method than the accuracy measure because the ROC analysis also measures the magnitude of the misclassified test samples; that is, how much was it misclassified. Furthermore on a ranking with high AUC value, a better decision threshold can be set with a more sophisticated technique.

Table 6.6 gives an overall comparison of several similarity methods with different classification methods. The best results were obtained with EL in almost every cases.

The classification with 1NN on SVKs gave the best AUC results, and the results were especially good with the 3PGK dataset. SVKs with SVM perform less well than LAK, but we should remark here that the time complexity of LAK is quadratic and for a sequence pair it computes more operations than the SW method.

The Fig. 6.9 illustrates a pairwise correlation between the 1NN classification results

Table 6.6: The overall protein classification results with various similarity and classification methods on 3PGK and COG.

Methods	1NN	SVM	RF	ANN	LogReg	*avg*
3PGK						
BLAST	0.863	0.953	0.852	0.958	0.953	*0.921*
SW	0.861	0.953	0.866	0.955	0.948	*0.916*
LA	0.860	0.955	0.876	0.956	<u>0.959</u>	*0.922*
LZW	0.783	0.924	0.846	0.928	0.915	*0.879*
PPMZ	0.812	0.948	0.915	0.959	0.940	*0.914*
EL[1]	<u>0.964</u>	<u>0.966</u>	<u>0.948</u>	<u>0.963</u>	0.927	*<u>0.953</u>*
COG						
BLAST	0.892	0.968	0.937	0.968	0.960	*0.945*
SW	0.762	0.915	0.892	0.961	0.874	*0.881*
LAK	0.889	0.972	0.931	0.973	<u>0.966</u>	*0.946*
LZW	0.843	0.941	0.883	0.924	0.920	*0.902*
PPMZ	0.873	0.955	0.910	0.937	0.942	*0.923*
EL[1]	<u>0.956</u>	<u>0.978</u>	<u>0.956</u>	<u>0.974</u>	0.956	*<u>0.964</u>*

[1]The similarity scores we got by equivalence learning with the RF learner and the C_D composition method. In each column the largest value is underlined.

Table 6.7: The AUC results for protein classification with LAK and SVK learned by nnLS and 1SVM.

	3PGK			COG		
	nnLS	1SVM	LAK	nnLS	1SVM	LAK
1NN	0.975	0.972	0.860	0.891	0.901	0.888
SVM	0.963	0.930	0.966	0.957	0.953	0.972

we obtained. Each point represents a group in the COG database and its x coordinate stands for the 1NN classification result (AUC value) obtained by BLAST, while its y coordinate represents the 1NN classification result (AUC value) obtained by EL. In this figure most points are located above the diagonal line which means that the EL technique yields better AUC values than those obtained by using the BLAST methods.

6.5 Conclusions

Equivalence learning provides a two-class classification approach for object-pairs defined within a multi-class scenario, where the underlying idea is not to classify objects into their respective classes, but rather classify them as equivalent (belonging to the same class) or non-equivalent (belonging to different classes). The method is based on a spectrum of the similarity between the objects represented in vector form. Here we used techniques taken from fuzzy theory like the normalization function κ and Dombi operator class which make the equivalence learning more robust. We hope this technique will also prove more popular in sequence vectorization in other fields of bioinformatics. The similarity method used during ranking represents biological knowledge and any special method can be used for a specific sequence group or task.

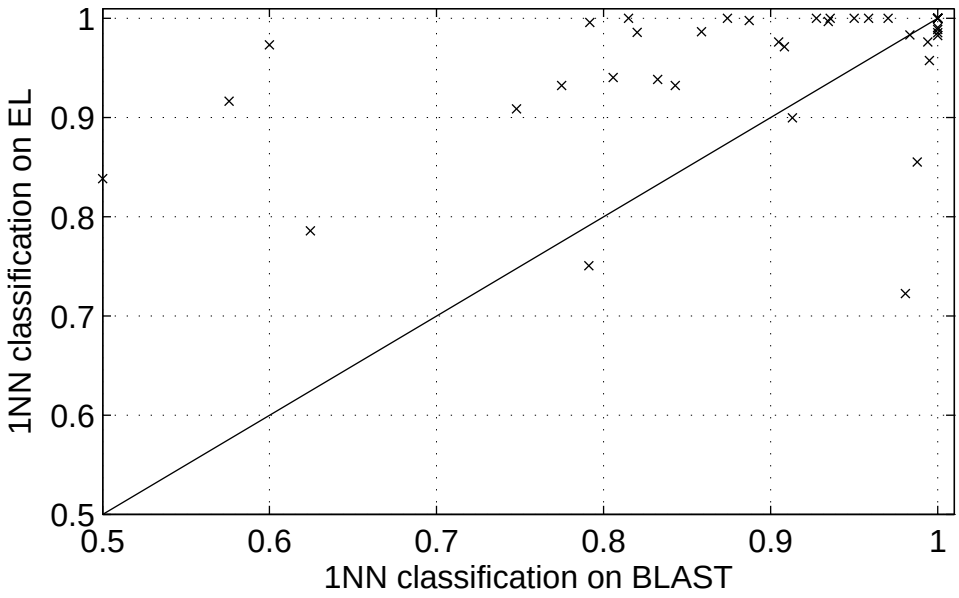


Figure 6.9: Correlation of the 1NN classification results on the COG groups.

What is more, SVK provides a way of constructing a valid kernel function from any similarity function.

Equivalence learning could be applied as a postprocessing method after the similarity scores between the query and the database have been calculated in a pairwise manner. The time and space requirements are dwarfed by the cost of the similarity method, but significantly better results can be obtained. We think that this is because not just the similarity between the query and database is used, but because the relationship between the equivalent and the non-equivalent sequences is also used, hence additional information can be included in the classification.

Part III

Microarray Data

Chapter 7

Kalman-Filtering for Microarray Data

Here we present the Kalman filter (KF) as a pre-processing step for microarray-based molecular diagnosis. Here the Author designed and evaluated the experiments for the classification and a feature selection method on microarray datasets. The Author also designed an automatic parameter-tuning algorithm for the Kalman Filter as well, which is a common and indivisible result with the first author of [8].

7.1 Introduction

A DNA microarray is a high-throughput technology used in molecular biology and medicine to help gain a deeper insight into sophisticated cellular processes. It consists of an arrayed series of thousands of microscopic spots of short fragments of DNA (or RNA) and it is used to obtain a molecular fingerprint of gene expression in cells that may or may not be translated into proteins. The method has enabled large numbers of genes – from healthy or diseased samples (like various types of cancerous cell populations) – to be studied in a single experiment [132].

These data sets are arranged in a matrix form whose rows represent genes and columns represent a cell (or experiment), like that shown in Fig 7.1 and Fig 7.4. Here the task is to identify the smallest set of genes (rows) that best separate the different types of cells (columns); that is, to identify those genes that determine the absence or presence of a particular disease. A knowledge of this means that doctors can apply more accurate treatments and diagnoses [133].

Nowadays a large number of machine learning algorithms have been proposed for processing of microarray data. The SVM classifiers [63] have been used effectively in microarray gene expression analysis [64] for disease state estimation. ANNs have been employed for instance in the classification of cancer subtypes [72]. The kNN algorithm [31] was used for cancer diagnosis in (Ramaswamy et al., 2001). The RF technique was used for drug discovery [77] and in tumour classification [78].

Gene expression measurements capture a large amount of expression variance. A

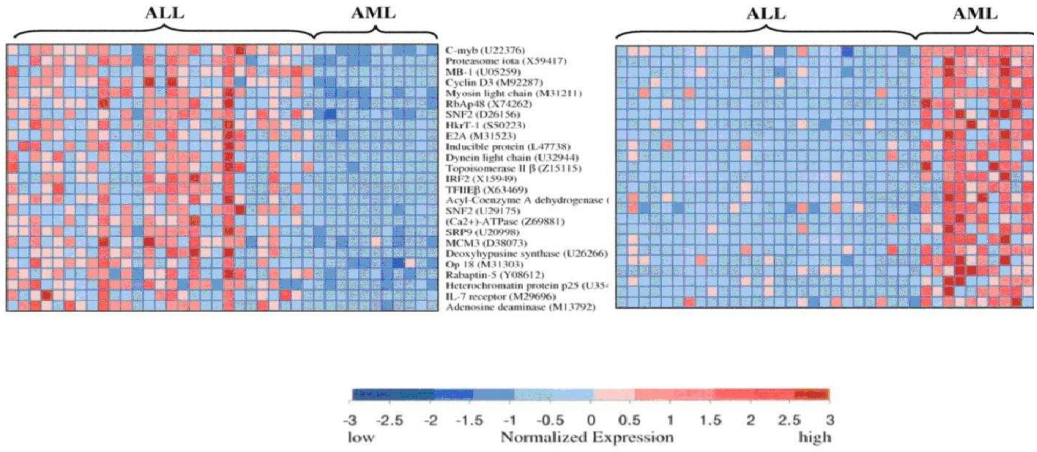


Figure 7.1: An example of gene-expression based molecular classification of leukemia subtypes. Each column is related to a state (disease) and each row is related to a gene whose name is also indicated in the figure. The expression level of genes is indicated by the colour intensity in each cell. Here Acute Lymphoblastic Leukemia (ALL) (right) and Acute Myeloid Leukemia (AML) (left) were diagnosed.

large number of error sources also corrupt the gene expression data, even though normalization procedures are meant to reduce such influences. The two previously mentioned types of variation alter the true gene expression states associated with the particular diseases in question. Under such circumstances, the Kalman filter provides a reasonable framework for pre-processing the expression data by removing the noise and estimating the multi-variable noise-free tumour specific states.

The Kalman filter [134–136] is a powerful mathematical tool that has been widely used in many fields of engineering from systems and control theory to signal processing, due to its robustness even under the violation of the normality assumption. It has also been used in supervised learning as well as in myriads of real world applications. Its application in the bioinformatics field however, has been quite limited [137], not taking advantage of its full potential as a multivariate signal processor. Our aim here was to adapt the linear Kalman filter to handle the microarray data in order to reduce its noise level. Using various supervised learning tools, we tested the performance of the filtered datasets in classification setups. We also investigated whether the visualization of such filtered datasets could yield a more comprehensible representation of separate classes.

In the next section we will give a brief introduction to the Kalman Filter along with a new parameter adaptation technique. In Section 7.3 we will summarize the datasets and methods we used in our experiments, and then in the other two sections we will provide an evaluation of the methods studied and then draw some conclusions based on our findings.

7.2 The Kalman Filter (KF)

The Kalman filter (KF) is based on the assumption of a continuous system that can be modelled as a normally distributed random process X, with mean $\overline{x}$ (the state) and

variance P (the error covariance):

$$X \sim N(\overline{x}, P) \tag{7.1}$$

This method furthermore assumes that the output of the system can be modelled as a random process Z that is a linear function of the state

$$\widehat{x} \tag{7.2}$$

plus an independent, normally distributed, zero-mean white noise process V,

$$\overline{z} = H\overline{x} + \overline{v},$$

where $V \sim N(0, R)$ and $E(XV) = 0$. For our study we modelled the microarray data flow using the following simplified discrete time state-space representation of equations (7.1) and (7.2):

$$\begin{aligned} x_k &= x_{k-1} + w_k \\ z_k &= x_k + v_k. \end{aligned} \tag{7.3}$$

The first equation is a linear form of (7.1) containing the addition of an innovation process $W \sim N(0, Q)$. Vectors w_k and v_k may be interpreted as the modelling error (i.e. the deviation from a mean, stem state towards the particular biological states in question) and measurement noise respectively, the latter comprising the previously mentioned functional and experimental variances. Note that since the state transition matrix equals the unit matrix I, as does the output matrix H, they have been omitted for simplicity. Given the models of the white noise processes W and V (Q and R respectively) and the array measurements z_k, the aim of the KF approach here is to estimate the state vectors $\widehat{x}^-$ containing noise-free gene expression data.

If we assume the microarray profiling process to be stationary (i.e. its statistical properties remain constant over time), the Kalman iterative estimation will converge to the steady state KF, in which case the error covariance can be computed by solving the discrete algebraic Riccati equation:

$$P = P - P(P + R)^{-1}P + Q \tag{7.4}$$

Hence, the Kalman gain is given by:

$$K = (P + R)^{-1} \tag{7.5}$$

Note that since the state transition matrix equals the unit matrix I, as does the output matrix H, they have been omitted. Finally, the estimated expression state vector is

$$\widehat{x}_k = \widehat{x}^- + K(Z_k - \widehat{x}^-), \tag{7.6}$$

where $\widehat{x}_k$ is an estimate of $\overline{x}$ based on the previous samples.

7.2.1 Parameter Setting

Given the training vector set, $\widehat{x}^-$ can be fixed as the average of the class means, where for each class the means are computed from the member samples. We then use the training set to initialize and tune the two KF parameters, namely Q and R. To reduce the dimensionality of the problem, we performed singular value decomposition [138]:

$$Z = UDY \tag{7.7}$$

The rows of Y are eigengenes and capture most of the variance of the original training dataset, while the columns correspond to the samples. The covariance matrix Q of the innovations can thus be interpreted as the between class covariance (i.e. the covariance of the class means with $\widehat{x}^-$ subtracted) evaluated on the reduced dimensionality training set Y. The measurement noise model R is estimated as a weighted form of the within class covariance of Y (i.e. the covariance of Y with the class means subtracted). To avoid over-fitting we tune these parameters by introducing some uncertainty variance such that $Q = Q + qI$ and $R = R + rI$. Our tests led us to conclude that in the case of single channel raw intensity array data (i.e. Affymetrix) $q = Q_{11}$ and $r = R_{11}$ are good choices for a reasonably good performance. Here the 11 index refers to the first eigengene usually considered as the offset of the microarray dataset, in which case it has a quite small variance. In the case of expression log-ratio data (usually coming from dual channel cDNA chips) or very sparse expression matrices, these parameters yield acceptable results when we choose

$$\begin{aligned} q &= \sum_i Q_{ii} \\ r &= \sum_i R_{ii}, \end{aligned} \tag{7.8}$$

where n is the number of training samples. With the tuned parameters we compute the low dimensional Kalman gain K_Y using equations (7.4) and (7.5). Next, from (7.6) and (7.7), the filtered gene-expression state vector is given by:

$$x_k = X + UDK_Y D^{-1} U^T (z_k - x^-), \tag{7.9}$$

where z_k now spans the entire dataset, including both the train and test measurements.

7.3 Datasets

We tested the Kalman filtering-classification scheme on a number of publicly available datasets, which are summarized in Table 7.1. In all dataset the train and test sets were

Table 7.1: Principal features of the datasets used

Name	#Classes	#Genes	#Train	#Test	Source
ALL-AML	3	7129	38	34	Golub (1999)
BC[b]	2	24481	78	19	van't Veer (2002)
Leukemia[a]	7	12558	215	112	Yeoh (2002)
LC	2	12533	32	149	Gordon (2002)
MLL	3	12582	57	15	Armstrong (2002)
SRBCT[a]	4	2308	63	25	Khan (2001)
Tumours	14	16063	144	54	Ramaswamy (2001)

[a] dataset containing log-ratio expression data.

[b] sparse dataset.

given.

ALL-AML. The leukemia (ALL-AML) dataset of Golub [139] is a popular dataset and is often used to test binary classification algorithms. Using the original sample annotation we partitioned this dataset into three leukemia classes. Hence the dataset consisted of T lineage acute lymphoblastic leukaemia (T-ALL), B lineage acute lymphoblastic leukemia (B-ALL) and acute myeloid leukemia (AML) samples.

MLL. The mixed lineage leukemia (MLL) dataset [140] consists of acute lymphoblastic leukemia (ALL) and AML samples along with ALLs carrying a chromosomal translocation involving the MLL gene.

Leukemia. The paediatric acute lymphoblastic leukemia dataset [141]. This is composed of B-ALL subtypes expressing BCR-ABL, E2A-PBX1 and TEL-AML1, respectively, a hyper-diploid karyotype, as well as MLL, T-ALL and a novel leukemia subtype.

Tumours. The 'various tumour types' dataset [142] is considered a difficult dataset and consists of 14 classes of tumours: breast, prostate, lung, colorectal, lymphoma, bladder, melanoma, uterus, leukemia, renal, pancreas, ovary, mesothelioma and central nervous system tumours.

Lung Cancer (LC). The dataset LC of [143] contains microarray data that accounts for two distinct pathological alterations of the lung: malignant pleural mesothelioma and adenocarcinoma.

Small, Round Blue Cell Tumours (SRBCT). The SRBCT of a childhood dataset [72] includes a training set of neuroblastoma, rhabdomyosarcoma, Burkitt lymphoma and the Ewing family of tumour samples and an independent test set that, besides the samples belonging to the training classes, also contains samples that should not be classified into any of these tumour types.

Breast Cancer (BC). Van't Veer et al. [144] provides a dataset BC consisting of samples coming from breast cancer patients that were clustered by the original authors into two classes according to the patient's response to adjuvant therapy: relapse and non-relapse.

7.4 Feature Selection, Recursive Feature Elimination (RFE)

A common goal in microarray data classification for diagnostic purposes is to select a minimal number of genes that could work as signatures for specific tumours. Since SVM is generally thought to perform best on classification problems, we will briefly introduce the Recursive Feature Elimination (RFE) algorithm, a recently proposed feature selection method described in [145] which was designed close to SVM. The method seeks to recursively eliminate features, keeping the "best" m that lead to the largest margin of class separation using an SVM classifier. Inspecting the subset of n surviving features at a certain point in the procedure, the algorithm basically does the following:

I Train the SVM with the n dimensional data, and thus obtain the norm vector w of the separating hyperplane.

II Compute the feature ranking criteria $c_i = (w_i)^2$, $i = 1, \ldots, n$.

III Find and eliminate the feature with the smallest ranking criterion $f = \text{argmin}_i\{c_i\}$.

The above procedure is repeated until the number of remaining features reaches m. Here, the RFE algorithm we used was part of the Spider package. RFE was employed with a linear kernel SVM, included in the same software package.

7.5 Results and Discussion

We applied the KF method on the previously described datasets and for a comparative study the SVM, ANN, 1NN and RF supervised learning methods were evaluated in a full gene set manner. Table 7.3 summarizes the Accuracy and ROC scores we obtained. Evidently, the KF method definitely improves the classification results of the ANN, 1NN and RF. The SVM results were boosted in 64% of the overall scores.

To assess the significance of filtering on microarray data classification, we performed paired two sample t-tests to compare the accuracy and ROC scores of the classification procedures on the original datasets with their counterparts in the KF case. The t-statistic was applied in one-tail fashion testing against the alternative hypothesis that the mean of accuracies/ROC scores produced by a certain method on the raw datasets is less than the mean of the matched performance measures on the pre-processed datasets. Table 7.2 shows that with 95% confidence the KF approach significantly improves the accuracy of the ROC score. In our study we also compared the KF scheme with a different approach to multivariate filtering. Principal Component Analysis (PCA) [146] based filtering consists of removing the non-significant variance components computed using the eigen-decomposition of the covariance matrix of the training set. The PCA results with SVM are shown in Table 7.3. Unlike PCA, the KF method keeps the dataset in the original gene space, and it is also a supervised procedure. This point is made clear by the P-values in Table 7.2. In the SVM framework, the PCA filtered

Table 7.2: Significance test results

t-test (α=0.05)	Accuracies	ROC scores
$P_{SVM>KF+SVM}$	0.034	0.82
$P_{PCA+SVM>KF+SVM}$	0.028	0.80
$P_{ANN>KF+ANN}$	0.028	0.025
$P_{1NN>KF+1NN}$	0.036	0.0002
$P_{RF>KF+RF}$	0.052	0.0075
$P_{SVM>PCA+SVM}$	0.82	0.915

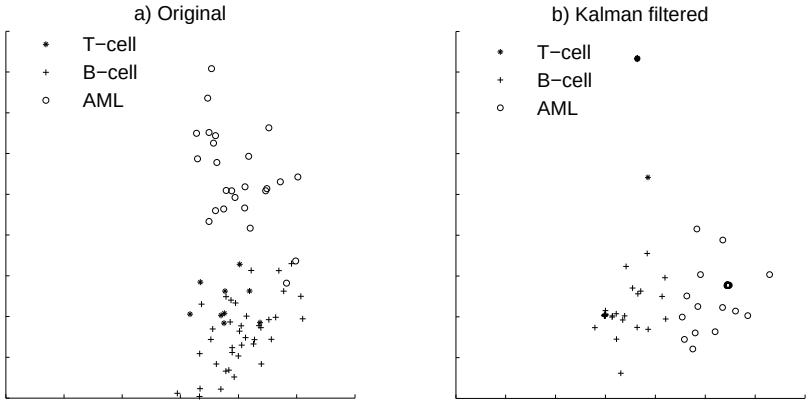


Figure 7.2: The original (a) and the Kalman filtered (b) AML-ALL dataset visualised by LLE.

datasets did not yield any improvement (at a significance level of 0.05) in the Accuracy and ROC score compared to the original data. Using the same learning algorithm, the KF resulted a significant improvement in accuracy compared with those using the PCA technique. The advantage of a pre-processing approach like this one here is not just a better classification performance, but also an improved visualization capability of the data. Fig. 7.2a depicts the original AML-ALL dataset, while Fig. 7.2b depicts the Kalman filtered dataset. The LLE representation clearly shows that the classes are more delineated with filtering than without. The heat map with a hierarchical clustering presented in Fig. 7.4 demonstrates how effectively the noise of the features was removed by the KF technique. The standard deviation of the gene expression values was reduced in each class and the tumour groups were separated into distinct clusters.

7.5.1 Gene Selection

A common goal in microarray data classification for diagnostic purposes is to select a minimal number of genes that could work as signatures for specific tumours. The RFE feature selection method was evaluated on the original and the Kalman filtered datasets to test whether filtering could help find more reliable subsets of marker genes. The results we obtained, summarized in Table 7.5, show that the number of Kalman filtered features necessary for a good discrimination of tumour types is smaller than the size of the raw feature set required for a similar performance. The same result is noticeable in Fig. 7.3 where, in a three-bestfeature setup, the MLL classes are well

Table 7.3: A comparison of the classification performance on the original dataset and on the Kalman filtered dataset.

	SVM			ANN		1NN		RF	
	Original	PCA[a]	KF	Original	KF	Original	KF	Original	KF
ALL-AML									
ROC score	0.99	0.99	0.99	0.97	0.99	0.73	1.00	0.92	0.95
ACC	0.91	0.82	0.97	0.91	1.00	0.82	1.00	0.74	0.94
BC									
ROC score	0.88	0.81	0.70	0.67	0.74	0.23	0.68	0.64	0.68
ACC	0.58	0.63	0.68	0.37	0.74	0.63	0.63	0.63	0.63
Leukeamia									
ROC score	0.97	0.96	0.98	0.90	0.98	0.60	0.88	0.94	0.96
ACC	0.50	0.29	0.70	0.37	0.58	0.89	0.87	0.86	0.76
LC									
ROC score	1.00	0.99	0.99	1.00	0.99	0.59	0.99	0.99	0.99
ACC	0.99	0.98	0.98	0.99	0.98	0.94	0.98	0.93	0.98
MLL									
ROC score	1.00	1.00	1.00	1.00	1.00	0.87	1.00	0.92	0.98
ACC	1.00	1.00	1.00	1.00	1.00	0.93	1.00	0.80	1.00
SRBCT									
ROC score	0.99	0.99	1.00	0.99	1.00	0.66	1.00	0.99	1.00
ACC[b]	0.97	0.97	0.99	0.94	0.95	0.91	0.95	0.93	0.98
Tumours									
ROC score	0.95	0.91	0.94	0.90	0.94	0.72	0.92	0.84	0.87
ACC	0.74	0.63	0.80	0.50	0.80	0.46	0.67	0.48	0.67

[a]See the text. [b]Denotes the mean of the class accuracy.

Table 7.4: FSR on 10 features selected via RFE

	ALL-AML	BC	Leukaemia	LC	MLL	SRBCT	Tumours
Original	14.088	1.480	4.079	5.757	8.481	3.621	3.406
KF	19.737	2.677	66.299	4.164	67.659	105.181	29.668

separated in the KF data but they are overlapping in the original vector set. Fig. 7.4 shows a heat map visualization of the MLL dataset with 50 selected features. While on the train set KF obviously removes the measurement noise, which results in clearly separated tumour groups, the variance of the test set was also noticeably diminished by the filter. Note that the genes selected from the original and the filtered datasets are quite distinct.

To compare the quality of features selected from the original datasets with the filtered ones, the Fisher Separation Ratio (FSR) [31] was used. The FSR is a scalar which is large when the between-class covariance is large and when the within-class covariance is small. Here the between-class scatter matrix is the scatter of the class mean vectors around the overall mean vector, while the within-class scatter matrix denotes the weighted average scatter of the covariance matrices of the sample vectors belonging to each class. Table 7.4 lists the FSR scores for 10 features independently selected from each dataset. The significantly larger scores ($P = 0.0245$ obtained from a t-test, as described previously) produced by the KF features collectively demonstrate the greater predictive power of the estimated expression data that best define the causal biological states.

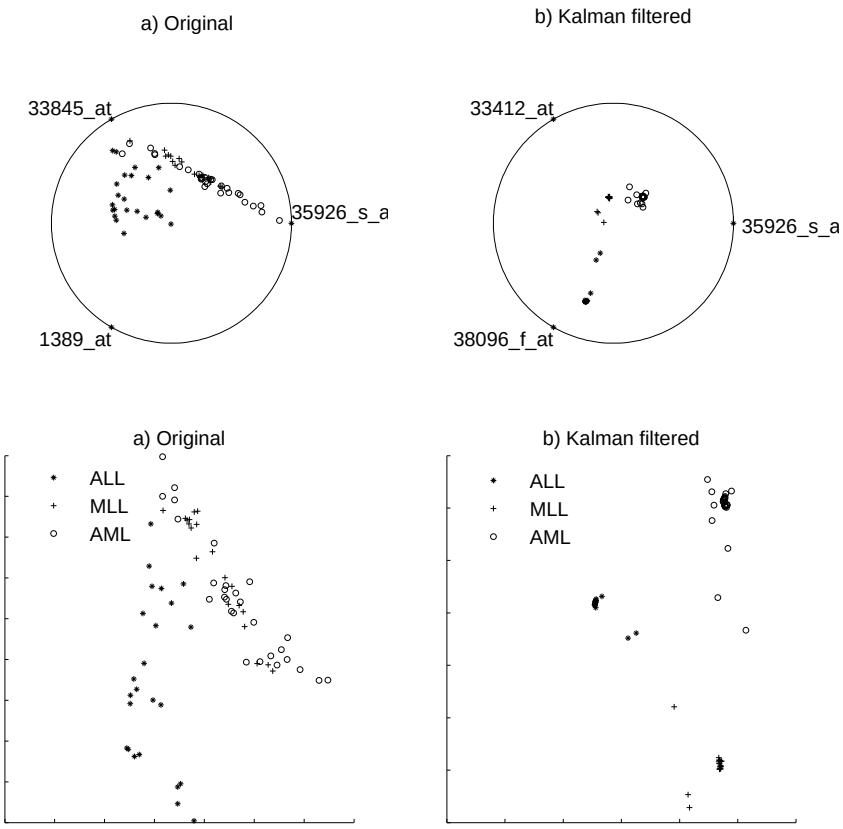


Figure 7.3: Visualization of the original (a) and the Kalman filtered (b) MLL dataset. Here the RadViv (top) method was used on three genes selected by RFE and plotted on the unit circle. The same genes were used with LLE(bottom).

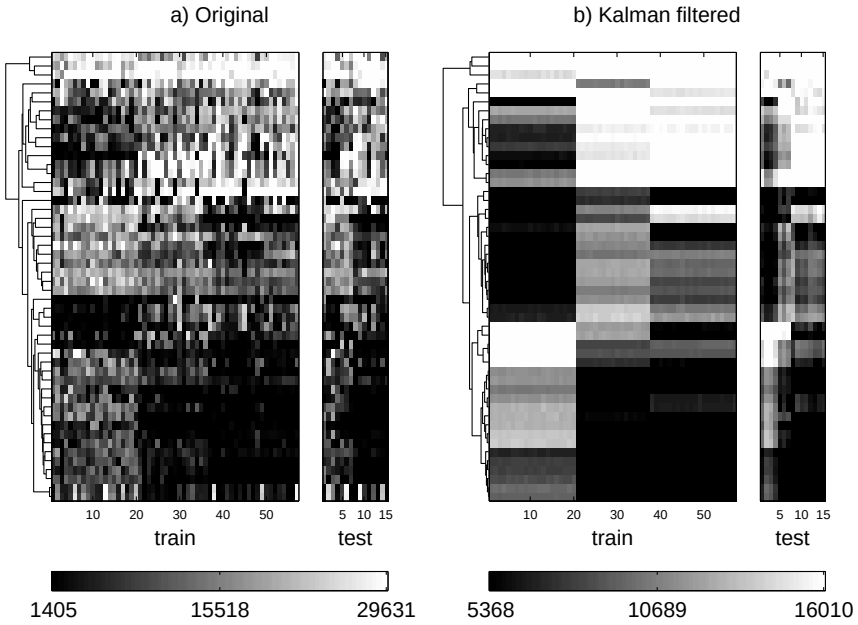


Figure 7.4: A Heat map representation of the best 50 genes selected by RFE from the MLL dataset. On the Kalman filtered datast (right) the features are less noisy and the tree classes are further apart than in the original dataset (left).

Table 7.5: The accuracy and ROC scores obtained by SVM as a function of the number of selected features.

Dataset name		Number of selected features with RFE								
		2	3	5	7	10	15	20	30	50
Accuracies										
ALL-AML	Original	0.53	0.56	0.68	0.68	0.68	0.65	0.74	0.76	0.85
	KF	0.74	0.94	0.82	0.85	0.97	1.00	0.94	0.97	0.97
BC	Original	0.79	0.63	0.63	0.63	0.63	0.63	0.58	0.58	0.58
	KF	0.63	0.63	0.63	0.63	0.63	0.58	0.58	0.58	0.63
Leukemia	Original	0.26	0.46	0.58	0.60	0.66	0.82	0.75	0.78	0.77
	KF	0.19	0.32	0.59	0.68	0.79	0.79	0.77	0.81	0.54
LC	Original	0.95	0.98	0.99	0.99	0.97	0.98	0.97	0.97	0.98
	KF	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
MLL	Original	0.67	0.67	0.73	0.87	0.87	0.87	0.93	1.00	0.93
	KF	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SRBCT[a]	Original	0.81	0.74	0.84	0.81	0.81	0.85	0.89	0.92	0.97
	KF	0.88	0.88	0.95	0.99	0.99	0.99	0.99	0.99	0.99
Tumours	Original	0.13	0.11	0.19	0.24	0.26	0.43	0.50	0.46	0.54
	KF	0.17	0.17	0.35	0.48	0.52	0.65	0.65	0.69	0.74
ROC scores										
ALL-AML	Original	0.68	0.65	0.83	0.83	0.89	0.87	0.90	0.92	0.95
	KF	0.87	0.93	0.92	0.94	0.99	0.99	0.99	0.99	0.99
BC	Original	0.89	0.81	0.76	0.79	0.75	0.73	0.75	0.62	0.78
	KF	0.69	0.69	0.68	0.68	0.68	0.68	0.68	0.68	0.68
Leukemia	Original	0.74	0.82	0.88	0.89	0.90	0.95	0.93	0.92	0.95
	KF	0.74	0.84	0.95	0.96	0.98	0.98	0.99	0.98	0.98
LC	Original	0.97	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	KF	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
MLL	Original	0.86	0.88	0.93	0.98	0.99	0.96	0.95	1.00	1.00
	KF	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SRBCT	Original	0.84	0.79	0.90	0.90	0.89	0.93	0.98	0.97	0.99
	KF	0.92	0.93	0.97	1.00	1.00	1.00	1.00	1.00	1.00
Tumours	Original	0.61	0.65	0.75	0.79	0.80	0.81	0.85	0.84	0.88
	KF	0.68	0.77	0.86	0.89	0.87	0.90	0.91	0.91	0.93

[a] Denotes the mean of the class accuracy.

7.6 Conclusions

The KF method, then, provides a systematic approach to filtering, each gene expression being estimated using the variances of all the individual features, naturally assuming that many genes reflect the biological state of the sample due to the transcriptional network. Hence, it remains a matter for further study (i.e. PCR analysis) to determine whether the selected features can also independently predict and diagnose a tumour outcome. The performance of the KF technique here essentially depends on the tuning of the covariance matrices Q and R. Our choice of parameters proved to be reasonable for classification, although an improvement based on larger training data or better tuning formulae is possible. The filtering of one dataset took just a few seconds of CPU time, thus the technique presented here is a fast and scalable method for pre-processing data on the microarray.

Chapter 8

Conclusions

As the reader has seen, the topic of this thesis is the application of machine learning techniques to protein and disease state classification. As a brief summary of our conclusions we could say: if more useful biological knowledge can be included in the model, better results can be obtained. This can be interpreted as an illustration of the No Free Lunch hypothesis, i.e. we cannot get anything for free; there is usually a price to be paid for something.

The first part of the thesis describes our results for protein classification using machine learning techniques. In general, it seems that the classification results do not critically depend on the classifier method but strongly depend on how a sequence is represented. Here biological knowledge is included in the proximity measure that is applied in the so-called Empirical Feature Mapping technique for sequence representation. Most of the biological knowledge here is exploited by a 3D comparison (like DALI and PRIDE) and by the alignment-based methods (like BLAST, SW, NW). The CBDs do not incorporate any detailed biological knowledge; they are simply based on substring repetitions and distributions, but these substrings are not weighted by their importance. The early n-gram technique (double, triplet character composition) also offers a weak representation since the amino acid composition is not useful for determining the function or the structure of a protein. Then, as expected, the general performance of CBDs with the n-gram technique is not as good as the exhaustive 3D comparison or alignment-based methods in protein sequence classification and separation.

The situation is similar with the LRA technique. Here the protein ranking ability of a proximity measure is not only based on the similarity value between the test sample and the closest member in the positive class, but it also takes into account the similarity score of the nearest member in the negative class. Hence the LRA approach exploits additional information which improves the sequence ranking.

Equivalence Learning provides a two-class classification approach which is realised by a binary classifier and takes advantage of relationships among equivalent and non-equivalent object-pairs. This gives better protein classification results than when a sequence is represented "alone".

The KF procedure also exploits additional information during noise reduction as their parameters can be adjusted in a supervised way.

Unfortunately, the supervised methods are sensitive to the distribution of the train and test set and they can easily overlearn, which reduces the generalization ability of a classifier. We recommend using the supervised cross-validation approach, which can provide a more realistic estimation on how the algorithm will detect a novel subtype of a known protein class.

The results presented in this dissertation may suggest some further questions on this topic. It may be interesting to know whether the KF method can be used to remove the noise from sequence proximity methods in protein classification. Can we include some biological knowledge in the CBDs? Can we do this in such a way that they still preserve the metric property? We showed that a simple combination of two low-time complexity proximity measures can distinguish protein classes just as well as the exhaustive computationally intensive SW. Can we achieve a performance of the 3D structural comparison method like DALI either by the learning of a hybrid combination of comparison methods in a supervised fashion or by setting the mixed similarity to the ideal kernel matrix?

As we have seen, additional knowledge included to the models can indeed provide better results, but data conversion is just as important. Formalizing, modelling the biological processes and information with mathematical techniques may provide new different viewpoints and fine insights with their help into the nature of protein sequences.

Appendix A

Summary in English

Today the definition of Bioinformatics is not a clear term and it is difficult to define its border exactly. Loosely speaking, bioinformatics is a marriage of biology, informatics and mathematics and it employs computational tools and methods for managing, analysing and manipulating sets of biological data. This integrated multidisciplinary field includes biochemistry, genetics, structural biology, artificial intelligence, machine learning, data mining, information theory, software engineering, statistics, database theory, information visualisation and algorithm complexity and design. Major research efforts in this field include sequence alignment, gene finding, genome assembly, protein structure alignment, protein structure prediction, the prediction of gene expression, protein-protein interactions and the modeling of evolution processes.

The main tasks of bioinformatics are the gathering, organization and computational analysis of sequence databases. The classification of sequence data is at the heart of this work, since when sequencing a new genome, perhaps its function and structure are among the most important questions. To determine them, a newly sequenced protein is compared to well-known databases via a similarity function. Then their function and structure can either be inferred from the most similar, well-known protein sequences, or they can be classified into a known protein group by machine learning approaches like Artificial Neural Networks or Support Vector Machines.

A.1 Summary by Chapters

Chapter 2 does not contain any scientific contributions from the Author. This chapter seeks to provide an introduction, contains some basic terms and notations and it also presents problems and challenges in biological sequence classification as well as providing the basis for understanding the main results of this thesis.

In Chapter 3 we give a short description of our protein benchmark databases which were intended to provide standard datasets on which the performance of machine learning and sequence similarity methods could be compared. These are freely available. During the design of these databases we were interested in covering the practical problems of protein classification. Here, we also describe several strategies that we used to

construct positive, negative, train and test sets as well as present an experimental comparison along with the classification results obtained by the state-of-the-art machine learning methods and protein similarity measures, whose results can be used as an base line for further comparison studies.

The function and the structure of a newly sequenced protein is usually inferred from the most similar sequences' properties using similarity functions. A good similarity function should rank a positive item higher and should rank a negative item lower for certain protein class. Then the performance of a given similarity method can be evaluated by seeing how it ranks an unseen set of sequences on a particular protein class. In Chapter 4 the Author examined how this ranking ability could be improved by using the likelihood ratio approximation.

Information Distance is a recently developed universal metric for strings. Due to the fact that it is non-computable in the Turing sense, it has to be approximated by text file compressors. Chapter 5 gives an insight into the behaviour of Compression-based Distances (CBDs) in genome sequences. First we will investigate the CBDs from the sequence representation point of view; namely, how the reduced and enlarged alphabets help to distinguish protein classes. We will also examine whether a hybrid combination of CBDs with other fast but problem specific comparison methods really influences the ability to distinguish or classify protein sequences.

Sequence groups are vastly different in terms of most their parameters, and a method that performs well on one group may perform worse on another and vice versa, and very often there are no clear trends in the results. The learning of a similarity in a supervised manner may provide a general framework for adapting a similarity function to a specific sequence class. In Chapter 6 we describe a novel method which learns a similarity function over protein sequences by using a binary classifier and pairs of equivalent sequences (belonging to the same class) as positive samples and non-equivalent sequences (belonging to different classes) as negative training samples.

The content of Chapter 7 differs from the previous chapters. It describes DNA chips (a.k.a. a microarray) that contain gene expression data obtained from healthy and/or diseased tissues. These data items are arranged in a matrix form whose columns represent a tissues and its rows represent genes. Here the task is to identify the smallest set of genes (rows) that best separates the class of tissues (columns); that is, we need to identify those genes that determine the absence or presence of a particular disease. Knowing these genes more accurate treatment and diagnoses can be applied for a patient. Chapter 7 describes the Kalman Filter (KF) method as a noise-reduction step for DNS chip data. The performance of this method essentially depends on its parameters. Here, we present a new automatic parameter tuning technique which significantly improves the performance of the KF approach. The results we get a more robust disease-state estimator on publicly available binary and multiclass microarray datasets in combination with the most widely used classification methods available.

A.2 Summary by Results

In the following we summarize the results of the Author by arranging them into four distinct thesis points. Table A.1 shows the relation between the thesis points and the publication, where they were presented by the Author.

I Protein benchmark

a The Author participated in building the Protein Classification Benchmark database in order to provide standard datasets on which the performance of the machine learning algorithms and similarity/distance measures could be compared. The number of total classification tasks exceeds 9500. Here the contribution of the Author was the evaluation of the state-of-the-art machine learning techniques on the classification tasks and he provided a parameter set which probably gives the best results as a baseline for newly developed methods [1].

b The Author developed a general mathematical framework for constructing a positive train and test set, which was termed by supervised cross-validation. This technique gives a reliable estimation on how an algorithm will generalize a new distantly related subtype within a known protein class that can also be viewed as a generalization ability of the learned model. He also designed and evaluated the comparative experiments and the resulting datasets provided lower, and in our opinion, more realistic estimates of the classifier performance than those of cross-validation schemes (10-fold or leave- one-out) [2].

The Author examined how depend the classification results on the filtering of the categories from the negative set in order to speed the execution time of the preprocessing and learning method up and to avoid the class-imbalanced problem. The Author designed and evaluated the experiments that led him recommend to misuse it since the resulted negative class may be to specific ant less representative with respect to the entire database. Although this result may be considered as a negative results, in our opinion we should mention it because it makes the characterization of the hierarchically organized protein datasets more complete from classification point of view [2]. Hence when constructing the positive train set, we suggest using the supervised cross-validation but for the negative set we suggest using the random filtering method [2].

II Likelihood ratio scoring

a The Author suggested the application of a simple likelihood ratio approximation for improving the ranking ability of a protein similarity measure. He designed and evaluated the comparative experiments which justified his view that this likelihood scoring significantly improves the performance of similarity functions [3].

III Compression-based Distances (CBDs)

a *The Author examined the behaviour of CBDs on protein classification from several aspects. An analysis of the results showed that the CBDs perform less well than substructure-based comparisons like the outstanding Smith-Waterman algorithm in protein similarity. This is in fact expected, since Smith-Waterman calculations include a substantial amount of biological knowledge encoded in the amino acid substitution matrix while CBDs do not use any apriori information. [4; 5].*

The Author examined the efficiency of the CBDs as a function of the size of the alphabet. An alphabet reduction was carried out by grouping the similar types of amino acids an on the alphabet extension was obtained by representing each bi-gram and tri-gram with a new character. The Author designed and evaluated the experiments that did not display, for amino acids or nucleotide sequences, any noticeable relationship between the performance and the size of the alphabet [5]. These results may be regarded as a negative results, but considering them as an observation they could help bioinformatics applications.

b *The Author investigated the combination of CBMs with an additional cheap, but problem-specific similarity measure. He designed and evaluated the comparative test which showed that this mixed measure can slightly exceed the performance of the computationally expensive Smith-Waterman and two Hidden Markov Model-based algorithms as well. [4].*

IV Equivalence learning

a *The Author introduced the notion of equivalence learning as a new way of carrying out similarity learning, and he developed it for protein classification. He designed and evaluated exhaustive experiments and the results show that this novel protein classification technique performed better than the others [6].*

b *The Author developed a new class of kernel functions, namely the Support Vector Kernel (SVK), He theoretically proved that it is a valid kernel function, and He defined two new ways to learn SVK along with a new parameter setting technique. He designed and evaluated the experiments as well. [7].*

V Noise reduction for the microarray

a *The contribution of the Author to this task was the design of the experiments and evaluations of the classification and the feature selection methods on microarray datasets. The Author designed an automatic parameter-tuning algorithm for the Kalman Filter as well, which is a common and indivisible result with the first author of [8].*

The results presented in the dissertation resulted in several publications. Table A.1 summarizes which publication covers which item of the thesis points.

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
I	a	b,c						
II			a					
III				a,d	a,b,c			
IV						a	a,b	
V								a

Table A.1: The relation between the theses and publications.

Appendix B

Summary in Hungarian

Bár a bioinformatika területét nehéz lenne pontosan körülhatárolni, de elmondhatjuk, hogy a biológia, a matematika és az informatika egy közös része. A hetvenes-nyolcvanas-as években a molekuláris biológia és kémia területén jelentek meg a sok adatot termelő technikai újítások, másrészt elérhetővé váltak az olcsó de hatékony számítógépek is. Ezek együttesen vezettek egy új tudományág, a bioinformatika kialakulásához.

A bioinformatika egyik, és talán legszélesebb ága a kutató laborokban termelt adatok feldolgozásával, archiválásával, rendezésével, rendszerezésével valamint az adatok kiértékelésével és a bennük rejlő összefüggések feltárásával foglalkozik. Jelen disszertáció témaja is erre a területre sorolható.

B.1. Fejezetek áttekintése

A 2. fejezet nem tartalmaz tudományos hozzájárulást a Szerzőtől. E fejezet csupán bevezető jellegű, amely megadja a disszertációban előforduló definiciókat és fogalmakat, leírja e tudományterület főbb kihivásait és problémáit, alapot ad a disszertáció főbb eredményeinek megértéséhez.

A 3. fejezetben ismertetjük az általunk elkészített fehérje-osztályozási adatbázist. Ezt az adatbazist azért hoztuk létre, hogy az újonnan kifejlesztett gépi tanulási algoritmusok és szekvencia-hasonlósági módszerek egy egységes adatbázison legyenek összehasonlíthatók. Az adatbázis tervezésénél arra törekedtünk, hogy az osztályozási feladatok minél szélesebb körben lefedjék a gyakorlatban is felmerülő fehérje-osztályozási problémákat. Itt tárgyaljuk az osztályozási feladatokat (pozitív/negatív tanuló/teszt halmazokat) kijelölő startégiákat és hasonlítjuk őket össze részletesen. Ezenkívül itt közöljük az adatbázison elért osztályozási eredményeket a napjainkban leginkább elterjedt gépi tanuló és szekvencia-hasonlító módszerekkel, amely eredmények mintegy baseline-ként használhatók fel későbbi összehasonlításokban.

Egy újonnan szekvenált fehérje esetében talán az első legfontosabb kerdés annak térszerkezete, illetve funkciója. Ennek meghatározása történhet hasonlósági függvény segítségével úgy, hogy a hozzá - már alaposan tanulmányozott - leghasonlóbb szekvencia tulajdonságaiból következtetünk. Ilyen esetben egy hasonlósági függvénytől elvárjuk,

hogy egy, a csoportba tartozó elemet a rangsorban előre, míg egy nem a csoportba tartozó elemet a rangsorban hátra tegyen, a csoporthoz viszonyítva. Így, ilyen hasonlósági függvények rankingelési képességét kiértékelhetjük az ún. ROC-analízissel. A 4. fejezetben vizsgáljuk meg, hogy hogyan változik a rankingelési képessége egy tetszőleges hasonlósági függvénynek a „log-likelihood ratio" módszerrel.

Az „információs távolság" egy napjainkban kifejlesztett univerzális metrika sztringek távolságának mérésére. Mivel e mérték nem kiszámítható Turing értelemben, ezért ennek becslésére hagyományos tömörítő algoritmusokat alkalmaznak. A 5. fejezetben egy betekintést adunk arra, hogy e metrika hogyan viselkedik fehérje-szekvenciák esetében, mely jellemzőikre invariáns és melyekre érzékeny.

Fehérje csoportok nagyban különböznek paramétereikben, mint például: csoportméret, csoporton belüli és csoportok közötti hasonlóság, stb. Egy tetszőleges módszer, amelyik jobban működik egy csoporton, kevésbé működhet jól egy másik fehérjecsoporton, és viszont. A fehérje-hasonlóság tanulása felügyelt módon egy általános módszertant nyújthat hasonlósági függvények specifikus fehérjecsoporthoz igazításához. A 6. fejezetben megadunk egy módszert fehérjék hasonlóságának tanulására, amely kétosztályos tanuló-algoritmuson és ekvivalens fehérjepárok (amelyek egy osztályba tartoznak) és nem-ekvivalens párok (amelyek különböző osztályba tartoznak) halmazán alapszik. Továbbá megadunk egy újszerű módszert metrikák és kernel függvények tanulására is. A fejezetben természetesen megtalálható a módszerhez tartózó tapasztalati kiértékelés is.

A 7. fejezet tartalmában kicsit eltér az előzőektől. E fejezet DNS-chipekkel foglalkozik, amelyek többféle típusú (általában beteg vagy egészséges, vagy többféle betegségű) szövetekből (sejtekből) nyert génexpressziós adatokat tartalmaznak. Ezek az adatok mátrix formában rendezettek, ahol egy oszlop a mátrixban egy mintához, míg egy sor egy génhez tartozik. A feladat itt az, hogy minél pontosabban azonosítsuk azokat a géneket (mátrix sorait), amelyekkel a különböző tipusú minták a legjobban osztályozhatók (mátrix oszlopai). Átfogalmazva: válasszuk ki azokat a géneket, amelyek legjobban meghatározzák a betegség jelenlétét vagy hiányát. E gének ismeretében pontosabb diagnózis és hatékonyabb kezelés állítható fel a beteg számára.

A 7. fejezetben bemutatjuk a Kálmán-szűrő használatát zajszűrőként DNS-chipekre alkalmazva, valamint megadunk egy automatikus módszert a Szűrő paramétereinek beállítására, amellyel a DNS chipek széles típusán jobb osztályozási eredményeket érhetünk el már kisebb génhalmaz használatával. A módszert teszteltük publikus többosztályos DNS-chip adatbázisokon a napjainkban legelterjedtebb osztályozási módszerekkel.

B.2. Eredmények tézisszerű összefoglalása

A következőkben összegezzük a Szerző eredményeit négy fő tézispontba rendezve. A B.1. táblázat tartalmazza a kutatásokból származó publikációkat, valamint azok tartalmának viszonyát az egyes tézispontokhoz.

I A fehérje-osztályozási adatbázis

a A Szerző részt vett egy ingyenesen hozzáférhető fehérje osztályozási adatbázis elkészítésében, amelyen az újonnan kifejlesztett gépi tanulási algoritmusok és szekvencia-hasonlósági módszerek kiértékelhetők és összehasonlíthatók. Az osztályozási feladatok száma eléri a 9500-at. A Szerző feladata volt a napjainkban leginkább használatos state-of-the-art gépi tanulási és hasonlósági algoritmusok beállítása és kiértékelése az összes osztályozási feladaton. Az algoritmusok paraméterei és az eredmények az adatbázis weblapján elérhetők, így ezek az adatok felhasználatók összehasonlító tesztek elvégzésekor új algoritmusok kifejlesztésénél. [1].

b A Szerző kifejlesztett egy általános matematikai keretet a hierarchikusan rendszerezett fehérje adatbázisokra az osztályozási feladatok létrehozásakor a pozitív tanuló- és teszt elemek kiválasztására. Ezt a módszert felügyelt kereszt-validációnak nevezte el. Eredményként egy megbízhatóbb becslést kapunk arra, hogy egy betanított osztályozó algoritmus mennyire képes felismerni egy csoportban egy új, de még nem látott részcsoportot. Ez tekinthető az osztályozó generalizációs képességének is. A szerző megtervezte és elvégezte a szükséges teszteket az összehasonlításhoz és azt tapasztaltuk, hogy az így kapott osztályozási feladatok nehezebbek, de véleményünk szerint sokkal valósabb eredményt adnak a hagyományos, kereszt-validációs módszerekkel szemben (pl. 10-fold, leave one out). [2].

A Szerző megvizsgálta, hogy hogyan módosulnak az osztályozási eredmények, ha a hierarchikus rendezést kihasználva kategóriákat kihagyunk a negatív osztályból az előfeldolgozó és a tanuló algoritmusok futási idejének rövidítése és az ún. „class-imbalanced" probléma elkerülése céljából. A Szerző megtervezte és kiértékelte az összehasonlító teszteket, amelyek alapján nem javasolja a módszer alkalmazását, mert a kapott negatív halmaz nem reprezentálja a valós negatív fehérje-univerzumot [2].

II Likelihood ratio scoring

a A szerző megvizsgálta, hogy a likelihood ratio módszer hogyan változtatja meg a hasonlósági módszerek rankingelési képességét. A Szerző megtervezte és kiértékelte az összehasonlító teszteket, és arra a megállapításra jutott, hogy ezzel a módszerrel jelentős javulás érhető el fehérje rankingelés terén [3].

III Tömörítő alapú távolság (CBDs)

a A Szerző megvizsgálta a tömörítő alapú távolságok (CBD) viselkedését fehérje-szekvenciákon. A CBD metrikák hatékonyságán azt értjük, hogy ugyanabba az osztályba tartozó fehérje-szekvenciákhoz kicsi, míg különbözőkbe tartozókhoz nagy távolságértéket rendel. A kísérleti eredmények azt

mutatják, hogy a CBD metrikák kevésbé teljesítenek olyan jól, mint a részszekvencia alapú összehasonlító algoritmusok, mint például a legjobbnak tartott Smith-Waterman. Ez magyarázható azzal, hogy a Smith-Waterman tartalmaz biológiai tudást, ami lényegében az aminosav-helyettesítési mátrixba van kódolva, míg a CBD alapú távolságmódszerek nem alkalmaznak semmilyen priori tudást [4; 5]. A Szerző megvizsgálta, hogy a CBD metrikák hatékonysága hogyan változik a fehérje-szekvenciákat reprezentáló ábécé méretének függvényében. A fehérje-ábécé csökkentéskor az azonos típusú aminosavakat, míg az ábécé növelésekor a betű-ketteseket és betűhármasokat (bi-gram, tri-gram) ábrázoltuk egy új karakterrel. A Szerző megtervezte és kiértékelte a kísérleteket, ami alapján számottevő összefüggés nem volt megfigyelhető sem aminosav-szekvenciákon sem nukleotid-szekvenciákon [5]. Ezek az eredmények nem pozitív eredmények, de mint észrevételek segíthetik a bioinformatikai alkalmazásokat, tehát nem érdemes figyelmen kívül hagyni.

b *A Szerző megvizsgálta, hogy a CBD metrikát egy gyors, de alkalmazás specifikus heurisztikával (BLAST) kombinálva a kapott összetett mérték milyen hatékonyságú fehérje-klasszifikációban. A Szerző megtervezte és kiértékelte az összehasonlító teszteket. Eredményül megállapíthatjuk, hogy kombinált CBM és BLAST mértékkel közel azonos osztályozási eredmény érhető el, mint a legjobbnak tartott, de költséges Smith-Waterman módszerrel, és jobb eredmények érhetők el, mint két rejtett-markov model alapú (Fisher kernel, SAM) mértékekkel. [4]*

IV Ekvivalencia tanulása

a *A Szerző bevezette az ekvivalencia-tanulás fogalmát, mint egy újtípusú hasonlóság-tanulást. A Szerző megtervezte és kiértékelte az összehasonlító teszteket, amelyek azt mutatják, hogy ekvivalencia-tanulással jobb fehérje-osztályozást sikerült elérni. [6].*

b *A Szerző új típusú kernel függvények osztályát – Szupport Vektor Kernel (SVK) – is definiálta, és elméleti úton megmutatta, hogy az SVK kielégíti a kernel-függvényekre vonatkozó feltételeket. A szerző megadott két módszert is az SVK tanulására, megtervezte és elvégezte az összehasonlító teszteket. [6; 7].*

V Zajszűrés DNS-chipekre.

a *A szerző hozzájárulása ehhez a tanulmányhoz az összehasonlító tesztek megtervezése és kiértékelése a gén-kiválasztási és minta- osztályozási feladatokra DNS-chip adatbázisokon. A szerző tervezett egy automatikus paraméter-behangolási módszert is a Kálmán-szűrőhöz, amely közös és oszthatalan eredménye az első szerzővel. [8]*

A disszertációban szereplő eredmények több cikkben kerültek publikálásra. Az B.1 táblázat összegzi, hogy mely tézispontot mely publikáció közli.

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
I	a	b						
II			a					
III				a,b	a			
IV						a	a,b	
V								a

B.1. táblázat. Tézispontok és a Szerző publikációinak viszonya.

Bibliography

[1] Paolo Sonego, Mircea Pacurar, Somdutta Dhir, Attila Kertész-Farkas, András Kocsor, Zoltán Gáspári, Jack A. M. Leunissen, and Sándor Pongor. A protein classification benchmark collection for machine learning. *Nucleic Acids Research*, 35(Database-Issue):232–236, 2007.

[2] Attila Kertész-Farkas, Somdutta Dhir, Paolo Sonego, Mircea Pacurar, Sergiu Netoteia, Harm Nijveen, Arnold Kuzinar, Jack Leunissen, András Kocsor, and Sándor Pongor. Benchmarking protein classification algorithms via supervised cross-validation. *J Biochem Biophys Methods*, 35:1215–1223, 2007.

[3] László Kaján, Attila Kertész-Farkas, Dino Franklin, Neli Ivanova, András Kocsor, and Sándor Pongor. Application of a simple likelihood ratio approximant to protein sequence classification. *Bioinformatics*, 22(23):2865–2869, 2006.

[4] András Kocsor, Attila Kertész-Farkas, László Kaján, and Sándor Pongor. Application of compression-based distance measures to protein sequence classification: a methodological study. *Bioinformatics*, 22(4):407–412, 2006.

[5] Attila Kertész-Farkas, András Kocsor, and Sándor Pongor. The application of the data compression-based distances to biological sequences. In Frank Emmert-Streib and Matthias Dehmer, editors, *Information Theory and Statistical Learning*, Lecture Notes in Computer Science. Springer, 2008.

[6] Attila Kertész-Farkas, András Kocsor, and Sándor Pongor. Equivalence learning in protein classification. In Petra Perner, editor, *MLDM*, volume 4571 of *Lecture Notes in Computer Science*, pages 824–837. Springer, 2007.

[7] József Dombi and Attila Kertész-Farkas. Using fuzzy technologies for equivalence learning in protein classification. *Accepted for publication in Journal of Computational Biology*, 2008.

[8] János Z. Kelemen, Attila Kertész-Farkas, András Kocsor, and László G. Puskás. Kalman filtering for disease-state estimation from microarray data. *Bioinformatics*, 22(24):3047–3053, 2006.

[9] Leland Hartwell, Leroy Hood, Michael L. Goldberg, Ann Reynolds, Lee M. Silver, and Ruth C. Veres. *Genetics: From Genes to Genomes, 2nd eds*. McGraw-Hill, New York, 2003.

[10] Wikipedia. http://en.wikipedia.org/wiki/protein.

[11] Wikipedia. http://en.wikipedia.org/wiki/protein_folding.

[12] F. Allen, G. Almasi, W. Andreoni, D. Beece, B. J. Berne, A. Bright, J. Brunheroto, C. Cascaval, J. Castanos, P. Coteus, P. Crumley, A. Curioni, M. Denneau, W. Donath, M. Eleftheriou, B. Fitch, B. Fleischer, C. J. Georgiou, R. Germain, M. Giampapa, D. Gresh, M. Gupta, R. Haring, H. Ho, P. Hochschild, S. Hummel, T. Jonas, D. Lieber, G. Martyna, K. Maturu, J. Moreira, D. Newns, M. Newton, R. Philhower, T. Picunko, J. Pitera, M. Pitman, R. Rand, A. Royyuru, V. Salapura, A. Sanomiya, R. Shah, Y. Sham, S. Singh, M. Snir, F. Suits, R. Swetz, W. C. Swope, N. Vishnumurthy, T. J. C. Ward, H. Warren, and R. Zhou. Blue gene: a vision for protein science using a petaflop supercomputer. *IBM Syst. J.*, 40(2):310–327, 2001.

[13] Wikipedia. http://en.wikipedia.org/wiki/protein_structure.

[14] Wikipedia. http://en.wikipedia.org/wiki/structural_domain.

[15] Wikipedia. http://en.wikipedia.org/wiki/homology_(biology).

[16] Wikipedia. http://en.wikipedia.org/wiki/sequence_alignment.

[17] Steven Henikoff, Jorja G. Henikoff, and Shmuel Pietrokovski. Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, 15(6):471–479, 1999.

[18] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *In Atlas of Protein Sequences and Structure*, 5:345–352, 1978.

[19] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–53, 1970.

[20] P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, 26:787–793, 1974.

[21] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[22] C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer, Berlin, 1984.

[23] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[24] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, 2002.

[25] Christina S. Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

[26] Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002.

[27] Jean-Philippe Vert, Hiroto Saigo, and Tatsuya Akutsu. Local alignment kernels for biological sequences. In Bernhard Schoelkopf, Koji Tsuda, and Jean-Philippe Vert, editors, *Kernel Methods in Computational Biology*, Cambridge, MA, 2004. MIT Press.

[28] T. Jaakkola, M. Diekhaus, and D. Haussler. Using the fisher kernel method to detect remote protein homologies. *7th Intell. Sys. Mol. Biol.*, pages 149–158, 1999.

[29] D. Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999.

[30] Wikipedia. http://en.wikipedia.org/wiki/machine_learning.

[31] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley Interscience, 2 edition, 2000.

[32] Wikipedia. http://en.wikipedia.org/wiki/pattern_recognition.

[33] K. Vlahovicek, L. Kajan, V. Agoston, and S. Pongor. The SBASE domain sequence resource, release 12: prediction of protein domain-architecture using support vector machines. *Nucleic Acids Res*, 33(Database issue):D223–5, 2005.

[34] L. Holm and J. Park. Dalilite workbench for protein structure comparison. *Bioinformatics*, 16(6):566–567, June 2000.

[35] Danielle Talbot. Improving sequence alignment for protein modelling, http://www.jenner.ac.uk/ybf/danielletalbot.ppt.

[36] W. R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods Enzymol*, 183:63–98, 1990.

[37] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, October 1990.

[38] S. Eddy. Hmmer user's guide: biological sequence analysis using prole hidden markov models, 1998.

[39] David W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, September 2004.

[40] Sandor Pongor. personal communication.

[41] Paolo Sonego, András Kocsor, and Sándor Pongor. Roc analysis: applications to the classification of biological sequences and 3d structures. *Briefings in Bioinformatics*, 9(3):198–209, 2008.

[42] LB Lusted. Signal detectability and medical decision-making. *Science*, 171(997):1217–1219, 1971.

[43] Foster J. Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.

[44] P.A. Flach, H. Blockeel, C. Ferri, J. Hernandez-Orallo, and J. Struyf. *Decision support for data mining: introduction to ROC analysis and its application*, pages 81–90. Kluwer Academic Publishers, January 2003.

[45] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[46] M. Gribskov and N. Robinson. Use of receiver operating characteristic (roc) analysis to evaluate sequence matching, comput. chem. (20), pp. 25–33, 1996.

[47] Pierre Baldi and Søren Brunak. Review assessing the accuracy of prediction algorithms for classification: An overview.

[48] V. Bajic. Comparing the success of different prediction software in sequence analysis: A review, 2000.

[49] T. Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004.

[50] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, April 1982.

[51] Chris Brunsdon. An investigation of methods for visualizing highly multivariate datasets. In *Case Studies of Visualization in the Social Sciences*, pages 55–80, 1998.

[52] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[53] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[54] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comp.*, 15(6):1373–1396, June 2003.

[55] J.D. Pollack, Q. Li, and D.K. Pearl. Taxonomic utility of a phylogenetic analysis of phosphoglycerate kinase proteins of archaea, bacteria, and eukaryota: insights by bayesian analyses. *Mol. Phylogenet. Evol.*, 35:420–430, 2005.

[56] A. Andreeva, D. Howorth, and C. Brenner. Scop database in 2004: refinements integrate structure and sequence family data, 2004.

[57] L. H. Greene, T. E. Lewis, S. Addou, A. Cuff, T. Dallman, M. Dibley, O. Redfern, F. Pearl, R. Nambudiry, A. Reid, I. Sillitoe, C. Yeats, J. M. Thornton, and C. A. Orengo. The cath domain structure database: new protocols and classification levels give a more comprehensive resource for exploring evolution. *Nucleic Acids Res*, 35(Database issue), January 2007.

[58] R. L. Tatusov, N. D. Fedorova, J. D. Jackson, A. R. Jacobs, B. Kiryutin, E. V. Koonin, D. M. Krylov, R. Mazumder, S. L. Mekhedov, A. N. Nikolskaya, B. S. Rao, S. Smirnov, A. V. Sverdlov, S. Vasudevan, Y. I. Wolf, J. J. Yin, and D. A. Natale. The cog database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4, September 2003.

[59] P. Rice, I. Longden, and A. Bleasby. Emboss: the european molecular biology open software suite. *Trends Genet*, 16(6):276–7, 2000.

[60] Zoltán Gáspári, Kristian Vlahovicek, and Sándor Pongor. Efficient recognition of folds in protein 3d structures by the improved pride algorithm. *Bioinformatics*, 21(15):3322–3323, 2005.

[61] R. Cilibrasi and P. M. B. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[62] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, August 2001.

[63] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2 edition, 1999.

[64] Michael P. Brown, William N. Grundy, David Lin, Nello Cristianini, Charles W. Sugnet, Terrence S. Furey, Manuel Ares, and David Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *PNAS*, 97(1):262–267, January 2000.

[65] A. Zien, G. Ratsch, S. Mika, B. Schölkopf, T. Lengauer, and K. R. Muller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.

[66] Jean-Philippe Vert, Jian Qiu, and William S Noble. A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, 8(Suppl 10):S8, 2007.

[67] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.

[68] Li Liao and William Stafford Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *RECOMB '02: Proceedings of the sixth annual international conference on Computational biology*, pages 225–232, New York, NY, USA, 2002. ACM.

[69] Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK, 1996.

[70] Pierre Baldi and Søren Brunak. *Bioinformatics: Machine Learning Approach*. MIT Press, Cambridge, MA, 2001.

[71] János Murvai, Kristian Vlahoviek, Csaba Szepesvári, and Sándor Pongor. Prediction of protein functional domains from sequences using artificial neural networks. *Genome Research*, 11(8):1410–1417, 2001.

[72] J. Khan, J. S. Wei, M. Ringnér, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, and P. S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nat Med*, 7(6):673–679, June 2001.

[73] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pages 586–591, San Francisco, CA, 1993.

[74] Jr. J. E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations (Classics in Applied Mathematics, 16)*. Soc for Industrial & Applied Math, 1996.

[75] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.*, 6(4):525–533, 1993.

[76] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[77] S.K. Remlinger. Introduction and application of random forest on high throughput screening data from drug discovery, 2003.

[78] Tao Shi, David Seligson, Arie S. Belldegrun, Aarno Palotie, and Steve Horvath. Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma. *Mod Pathol.*, 18(4):547–557, April 2005.

[79] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[80] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, October 1999.

[81] J C Rice. Logistic regression: An introduction. In B Rhompson, editor, *Advances in social science methodology*, volume 3, pages 191–245. JAI Press, Greenwich, 1994.

[82] Ming Li and Paul M.B. Vitányi. Mathematical theory of thermodynamics of computation. Technical report, Computer Science Department, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1., Amsterdam, The Netherlands, The Netherlands, 1992.

[83] W. H. Zurek. Thermodynamic cost of computation, algorithmic complexity and the information metric. *Nature*, 341(6238):119–124, 1989.

[84] Schweizer D and Abu/mostafa Y. Kolmogorov metric spaces. *Manuscript, Compute Science*, pages 256–280, 1988.

[85] Charles H. Bennett, Peter Gács, Ming Li, Paul M. B. Vitanyi, and Wojciech H. Zurek. Information distance. *IEEETIT: IEEE Transactions on Information Theory*, 44:1407–1423, 1998.

[86] Ming Li and Paul Vitanyi. *An introduction to kolmogorov complexity and its applications*. Springer-Verlag, 2 edition, 1997. read.

[87] Ming Li, Jonathan H. Badger, Xin Chen, Sam Kwong, Paul E. Kearney, and Haoyong Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(1):149–154, 2001.

[88] Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. Zipping out relevant information. *Computing in Science and Engg.*, 5(1):80–85, 2003.

[89] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul Vitányi. The similarity metric. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 863–872, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[90] Alexander Kraskov, Harald Stögbauer, Ralph G. Andrzejak, and Peter Grassberger. Hierarchical clustering using mutual information. *CoRR*, q-bio.QM/0311037, 2003.

[91] Rudi Cilibrasi, Paul Vitányi, and Ronald De Wolf. Algorithmic clustering of music based on string compression. *Comput. Music J.*, 28(4):49–67, 2004.

[92] N. Krasnogor and D. A. Pelta. Measuring the similarity of protein structures by means of the universal similarity metric. *Bioinformatics*, 20(7):1015–1021, 2004.

[93] Paolo Ferragina, Raffaele Giancarlo, Valentina Greco, Giovanni Manzini, and Gabriel Valiente. Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment. *BMC Bioinformatics*, 8:252+, July 2007.

[94] Marco Cuturi and Jean-Philippe Vert. The context-tree kernel for strings. *Neural Netw.*, 18(8):1111–1123, 2005.

[95] C. Costa Santos, J. Bernardes, P. M. B. Vitanyi, and L. Antunes. Clustering fetal heart rate tracings by compression. In *CBMS '06: Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, pages 685–690, Washington, DC, USA, 2006. IEEE Computer Society.

[96] Ming Li. Information distance and its applications. In Oscar H. Ibarra and Hsu-Chun Yen, editors, *CIAA*, volume 4094 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2006.

[97] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149–154, Feb 2001.

[98] András Benczúr. A feltételes entrópiák összegzési tulajdonsága a shannon és a kolmogorov entrópiákra, in: Alkalmazott matematikai és informatikai szeminárium, 2005.

[99] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley and Sons, Inc., 1991.

[100] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977.

[101] Xin Chen, Sam Kwong, and Ming Li. A compression algorithm for DNA sequences and its applications in genome comparison. In *RECOMB*, page 107, 2000.

[102] J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, Apr 1984.

[103] F. M. J. Willems, Y. M. Shtarkov, and Tj. J. Tjalkens. The context-tree weighting method: basic properties. *IEEE Trans. Info. Theory*, pages 653–664, 1995.

[104] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, 130 Lytton Avenuve, Palo Alto, CA, 94301, 1994.

[105] Haixiao Cai, Sanjeev R. Kulkarni, and Sergio Verdú. Universal entropy estimation via block sorting. *IEEE Transactions on Information Theory*, 50(7):1551–1561, 2004.

[106] Juergen Abel. Improvements to the burrows-wheeler compression algorithm: After bwt stages, 2003.

[107] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of protein sequence and structure*, volume 5, pages 345–358. National Biomedical Reseaxch Foundation, Washington, D.C.,, 1978.

[108] Edward Susko and Andrew J. Roger. On reduced amino acid alphabets for phylogenetic inference. *Mol Biol Evol*, 24:2139–2150, Sep 2007.

[109] George J. Klir, Ute St. Clair, and Bo Yuan. *Fuzzy set theory: foundations and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[110] K. Karplus, C. Barrett, and R. Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

[111] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402, September 1997.

[112] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, July 1999.

[113] Craig G. Nevill-Manning and Ian H. Witten. Protein is incompressible. In *DCC '99: Proceedings of the Conference on Data Compression*, page 257, Washington, DC, USA, 1999. IEEE Computer Society.

[114] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information, 2003.

[115] Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14 - Proceedings of NIPS 2001, December 3-8, 2001, Vancouver, Canada*, pages 367–373, 2001.

[116] Ivor Tsang and James Kwok. Distance metric learning with kernels, 2003.

[117] James T. Kwok and Ivor W. Tsang. Learning with idealized kernels. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 400–407. AAAI Press, 2003.

[118] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 209–216, New York, NY, USA, 2007. ACM.

[119] Kilian Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, Cambridge, MA, 2006.

[120] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004.

[121] Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 1191–1198, New York, NY, USA, 2007. ACM.

[122] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[123] S. Amari and S. Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

[124] Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(1):38–46, 2005.

[125] Jean-Philippe Vert and Yoshihiro Yamanishi. Supervised graph inference. In *NIPS*, 2004.

[126] Y. Yamanishi, J.-P. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20(1):363–370, 2004.

[127] Y Tsuruoka, J McNaught, J Tsujii, and S Ananiadou. Learning string similarity measures for gene/protein name dictionary look-up using logistic regression. *Bioinformatics*, Aug 2007.

[128] Maricel Kann and Richard A. Goldstein. Optima: A new score function for the detection of remote homologs. In Concettina Guerra and Sorin Istrail, editors, *Mathematical Methods for Protein Structure Analysis and Design*, volume 2666 of *Lecture Notes in Computer Science*, pages 99–108. Springer, 2003.

[129] Hiroto Saigo, Jean-Philippe Vert, and Tatsuya Akutsu. Optimizing amino acid substitution matrices with a local alignment kernel. *BMC Bioinformatics*, 7:246, 2006.

[130] J. Dombi. Membership function as an evaluation. *Fuzzy Sets and Systems*, 35, 1988.

[131] J. Dombi. A general class of fuzzy operators, the demorgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy Sets and Systems*, 8:149–163, 1982.

[132] Wikipedia. http://en.wikipedia.org/wiki/dna_microarray.

[133] Pierre Baldi, G. Wesley Hatfield, and Wesley G. Hatfield. *DNA Microarrays and Gene Expression: From Experiments to Data Analysis and Modeling*. Cambridge University Press; 1 edt, 2002.

[134] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, D(82):35–45, 1960.

[135] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering : Theory and Practice Using MATLAB*. Wiley-Interscience, January 2001.

[136] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

[137] Qinghua Cui, Bing Liu, Tianzi Jiang, and Songde Ma. Characterizing the dynamic connectivity between genes by variable parameter regression and kalman filtering based on temporal gene expression data. *Bioinformatics*, 21(8):1538–1541, 2005.

[138] O. Alter, P. O. Brown, and D. Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proc Natl Acad Sci U S A*, 97(18):10101–10106, August 2000.

[139] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, October 1999.

[140] Scott A. Armstrong, Jane E. Staunton, Lewis B. Silverman, Rob Pieters, Monique L. den Boer, Mark D. Minden, Stephen E. Sallan, Todd R. Golub Eric S. Lander and, and Stanley J. Korsmeyer. Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, 30:41–47, 2001.

[141] E . Yeoh, M . Ross, S . Shurtleff, W . Williams, D . Patel, R . Mahfouz, F . Behm, S . Raimondi, M . Relling, and A . Patel. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1:133–143, 2002.

[142] S Ramaswamy, P Tamayo, R Rifkin, S Mukherjee, CH Yeang, M Angelo, C Ladd, M Reich, E Latulippe, JP Mesirov, T Poggio, W Gerald, M Loda, ES Lander, and TR. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *PNAS*, 98(26):15149–15154, 2001.

[143] GJ Gordon, RV Jensen, LL Hsiao, SR Gullans, JE Blumenstock, S Ramaswamy, WG Richards, DJ Sugarbaker, and R Bueno. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *PNAS*, 62(17):4963–4967, 2002.

[144] L. J. van 't Veer, H. Dai, M. J. van de Vijver, Y. D. He, A. A. Hart, M. Mao, H. L. Peterse, K. van der Kooy, M. J. Marton, A. T. Witteveen, G. J. Schreiber, R. M. Kerkhoven, C. Roberts, P. S. Linsley, R. Bernards, and S. H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536, January 2002.

[145] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[146] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, October 2002.