Chapter 4
The Application of Data Compression-Based Distances to Biological Sequences

Attila Kertész-Farkas, András Kocsor, and Sándor Pongor

Abstract Text compressor algorithms can be used to construct metric distance measures (CBDs) suitable for character sequences. Here we review the principle of various types of compressor algorithms and describe their general behaviour with respect to the comparison of protein and DNA sequences. We employ reduced and enlarged alphabets, and model biological rearrangements like domain shuffling. In the classification experiments evaluated with ROC analysis, CBDs perform less well than substring-based methods such as the BLAST and the Smith–Waterman algorithms, but perform better than distances based on word composition. CBDs outperformed substring methods with respect to domain shuffling, and in some cases showed an increased performance when the alphabet was reduced.

4.1 Introduction

The term biological sequence denotes the most characteristic data type used in biology today. Data on the structure of genes and proteins is collected and stored in the form of long character sequences, written in a four-letter alphabet for DNA and in a 20-letter alphabet for proteins. The collection, organisation and computational analysis of sequence databanks is one of the main tasks of bioinformatics. The classification of sequence data is at the heart of this work, since the annotation of raw sequence data is primarily based on similarity searches against existing databanks,

A. Kertész-Farkas (✉) and A. Kocsor
Research Group on Artificial Intelligence, Aradi vértanúk tere 1, 6720 Szeged, Hungary
e-mail: kfa@inf.u-szeged.hu, kocsor@inf.u-szeged.hu

S. Pongor
Protein Structure and Bioinformatics Group, International Centre for Genetic Engineering and Biotechnology, Padriciano 99, 34012 Trieste, Italy and Bioinformatics Group, Biological Research Centre, Hungarian Academy of Sciences, Temesvári krt. 62, 6701 Szeged, Hungary
e-mail: pongor@icgeb.org

F. Emmert-Streib, M. Dehmer (eds.), *Information Theory and Statistical Learning,*
DOI: 10.1007/978-0-387-84816-7_4,

whereby a new sequence is assigned to one of the known classes (similarity groups). In order to accomplish this task, one needs sequence similarity measures that reflect the similarity of two sequences a realistic way.

Sequence similarity measures which are currently used fall into two main categories [2]. One of them is based on shared common substrings that are determined by approximate string matching heuristics, such as the well-known BLAST program [3], and dynamic programming methods like the Smith–Waterman algorithm [31]. The other class includes variants of n-gram distances whereby a sequence is first represented as a vector of n-gram word counts (e.g. character doublets, triplets and so on), and then compared in terms of an Euclidean distance or some other vector similarity measure [35]. Even though it is less efficient than a substring similarity search, n-gram methods are frequently included as fast screening tools into genome annotation pipelines.

The interest in compression-based methods was fostered by Vitnyi et al.'s seminal paper in which they proved that a simple index of sequence compressibility is actually better than n-gram techniques. Compression-based methods were soon employed in protein sequences as well [22], and it was also shown that different compressors perform differently on various sequence data [16].

The aim of the present work is to characterise compression algorithms with respect to two salient problems: invariance to biological changes and the resolution of sequence representations (alphabet size). In general, a similarity measure is expected to be invariant with respect to certain changes. For instance, the similarity of 3D shapes should be invariant with respect to the position of the objects; the similarity of melodies should be invariant to the pitch, the similarity of uttered words should be independent of the speaker, and so on. By the same token, we should expect that a similarity measure of biological sequences remain invariant with respect to a certain set of evolutionary events. Two of these are noteworthy: (1) Point mutations – sequences differing in only a few mutations should remain similar in the mathematical sense. This can be characterised by classification experiments followed by a ROC analysis [22]; (2) Major rearrangements can occur in the evolution of protein families. For instance, large segments of proteins can be displaced through a process called "domain shuffling". It is a moot question whether or not, or to what extent, rearranged sequences maintain similarity. This question can be studied by producing artificially rearranged sequences and comparing them with the various similarity measures [22].

The question of alphabet size is not merely one of technical interest. Reduced/enlarged alphabets are often used by bioinformaticians to represent protein and DNA sequences, but there is no systematic data telling us whether these really influence our ability to distinguish or classify biological sequences. Since the speed and memory requirements of compression algorithms depend on the algorithm's size, it would be interesting to see whether or not the classification efficiency of compression algorithms depends on this variable. And of course, all of these effects may depend on how closely the sequences are related.

In the next section we will provide a short introduction to the notion of the Kolmogorov Complexity-induced Information Metric as well as a description of the

various compressor types used in our experiments. In Sect. 4.3 we will describe the experiments with our conclusions drawn for them. Finally, in Sect. 4.4 we will summarise our findings and draw some conclusions.

4.2 Definitions and Methods

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 [27, 39] and in [30] in the context of image similarity. Later, an introduction with related definitions and theory was published in [5] in 1998. More formal definitions, theory and related details can be found in [26].

4.2.1 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.

Strings: 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 $l(\varepsilon) = 0$. The concatenation of x and y will be simply denoted by xy. A set of strings $S \subseteq \{0,1\}^*$ 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 S, we have

$$\sum_{x \in S} 2^{-l(x)} \leq 1. \tag{4.1}$$

An important consequence of this property is that in a sequence $s_1 s_2 \ldots s_n$ ($s_i \in S$) the end of string 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.

Kolmogorov Complexity: 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

$$K_F(x \mid y) = \min\{l(p) \mid F(p,x) = y\}. \tag{4.2}$$

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

$$K_U(x \mid y) \leq K_F(x \mid y) + c_F, \tag{4.3}$$

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 defined by $K(x \mid \varepsilon)$. 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 [26].

Information Distance: 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:

$$ID(x,y) = \min\{l(p) \mid U(p,x) = y,\ \ U(p,y) = x\}. \tag{4.4}$$

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:

$$E(x,y) = \max\{K(x \mid y), K(y \mid x)\}. \tag{4.5}$$

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.

Density, Universality and Metric Properties: 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 be satisfy the so-called normalization property:

$$\sum_{y:y\neq x} 2^{-D(x,y)} \leq 1. \tag{4.6}$$

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

$$log(\#B_E(x,d)) = d - K(d \mid x) \tag{4.7}$$

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" [27]. 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

$$E(x,y) \leq f(x,y) + O(log(k)/k), \tag{4.8}$$

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 [5].

Compression-Based Distances: 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

$$D(x,y) = \frac{\max\{K(x \mid y), K(y \mid x)\}}{\max\{K(x), K(y)\}}. \tag{4.9}$$

In [25] it was shown this satisfies the triangle inequality and vanishes when $x = y$ with a negligible error term. The proof of its universality was given in [24], and the proof that it obeys the normalization property is more technical (for details, see [12, 25]).

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 [12]. Thus we get

$$D(x,y) = \frac{K(xy) - \min\{K(x), K(y)\}}{\max\{K(x), K(y)\}}. \tag{4.10}$$

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:

$$CBD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \tag{4.11}$$

A CBM 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: $C(xx) = C(x)$
- Symmetry: $C(xy) = C(yx)$
- Monotonicity: $C(xy) \geq C(x)$
- Distributivity: $C(xy) + C(z) \leq C(xy) + C(xy)$

The proof can be found in Theorem 6.2 of [12]. 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.

4.2.2 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 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 [13]. 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): 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, $l(x)$ let 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

$$L(C) = \sum_{x \in D} p(x) l(x). \tag{4.12}$$

The Huffman coding is the optimal source coding; that is

$$L^* = \min_C \{L(C)\}, \tag{4.13}$$

subject to C satisfying the Kraft inequality. Solving this constrained optimization problem using Lagrange multipliers yields 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 length, we have $H(P) \leq L^* \leq H(P) + 1$ and the codes can be built by a method like Shannon–Fano coding [13].

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. For our experiments we used http://www.xcf.berkeley.edu/∼ali/K0D/Algorithms/huff/

Lempel–Ziv–Welch (LZW): It 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 $C(x)$ produces six 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 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. 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 more important so it is recommended for offline computations [11]. We downloaded this program from http://www.cs.cityu.edu.hk/∼cssamk/gencomp/GenCompress1.htm

Prediction by Partial Matching (PPM): PPM is an adaptive statistical data compression technique that uses 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 from this interval. A more likely event has a correspondingly longer interval, and thus a shorter codeword can be selected [36]. In our experiments we used the implementation from http://compression.ru/ds/.

Burrows–Wheeler Transform (BWT): While BWT 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 [8]. 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 [9]. 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 [1]. The compression speed is approximately half that of GZIP and BZIP2. This program was downloaded from http://www.data-compression.info/ABC/

4.3 Experiments and Discussion

In this section we will describe several experiments that were carried out to determine how CBDs can exploit the similarity among sequences, from different points of view. Where possible, the Smith–Waterman (SW) and the BLAST alignment-based sequence comparison methods as well as the naive distance were evaluated for comparison. The BLAST version 2.2.4 was used with a cutoff score of 25 and the SW algorithm used was implemented in MATLAB with the BLOSUM 62 matrix [19]. In both cases the gap opening and extension parameters were set to their default values. The naive distance between protein sequences was simply the Euclidean distance of a vector bi-gram counts of sequences.

4.3.1 Experiments on Compressor Properties

In our first experiments, we examined how well each compressor satisfies the properties of a normal compressor. To do this, we chose the SCOP40 (40% identity) database. The sequences were taken from the SCOP database 1.69 [4] and were downloaded from the ASTRAL site (http://astral.berkeley.edu/). Fifty-three non-contiguous domains were discarded and from the remaining 7,237 entries protein families that had at least five members, and at least 10 members outside the family but within the same superfamily were selected. Altogether, we collected 1,357 sequences. This database is available at http://net.icgeb.org/benchmark/ [32]. The average length of the sequences was 194 characters with a relatively high standard deviation of 116. Table 4.1 illustrates how each compressor satisfies the normality properties. In practice, the monotonicity is obeyed for stream-based compressors and only slightly less evident for block-coding types. The distributivity property appears to be satisfied in each case. Symmetry is not precisely obeyed for stream-based compressors and CBDs become slightly asymmetric; however, in practice, the difference $\mid CBD(x,y) - CBD(y,x) \mid$ is quite small. A compressor should search exact repetitions and obey idempotency, but in practice this seems to the most difficult condition for compressors and thus CBDs do not vanish when $x = y$.

4.3.2 Invariance Experiments on Rearranged Sequences

Protein evolution often includes rearrangements such as the gain or loss of domains and circular permutations. Therefore the question arises of whether or not CBDs can detect the differences between the products. To answer this, we used the C1S precursor (Swiss-Prot ID: C1S_HUMAN, AC:P09871), a multi-domain protein of

Table 4.1 Results of compressors' properties on SCOP40

Compressor	Length C(x)[a]		Idempontecy $\mid C(xx) - C(x) \mid$			Symmetry[b] $\mid C(xy) - C(yx) \mid$		
	avg.	std.	#viol[c]	avg.[d]	std.[e]	#viol[c]	avg.[d]	std.[e]
ABC	142	66.26	1,357	15.98	5.95	845,808	1.07	0.27
AH	121	63.01	1,357	100.84	61.55	1,240,000	1.38	0.62
LZW	154	78.99	1,357	92.31	52.52	1,570,000	2.63	1.70
GC	143	73.79	1,357	4.93	0.80	7,968	1.61	0.76
PPM	160	70.25	1,357	5.46	1.24	1,260,000	1.49	0.79

[a]The average length of the original sequences was 194 characters with a standard deviation of 116
[b]The total number of test cases was 1,841,450
[c]The number of cases when the compressor violated the condition
[d]The average bias
[e]The standard deviation of the bias
With these compressors acting on this dataset, the distributivity and monotonicity properties were not violated.

Table 4.2 The effect of sequence rearrangements on the various similarity/distance measures on C1S

	ABC		AH[c]		GC		LZW		PPM		SW	
	score	%[b]	score	%	score	%	score	%	score	%	score	%
Itself (ABCB′DD′E)[a]	0.088	0	0.939	0	0.013	0	0.646	0	0.020	0	1,895	0
Duplication of C (ABCCB′DD′E)	0.179	11	0.945	−284	0.051	4	0.661	13	0.059	5	1,804	5
Deletion of C (ABB′DD′E)	0.226	17	0.941	−107	0.115	11	0.663	14	0.121	12	1,673	12
Deletion of CB′ (ABDD′E)	0.435	43	0.939	0	0.272	28	0.701	47	0.261	28	1,305	31
Circular permut. (D′EABCB′D)	0.157	9	0.941	−107	0.062	5	0.665	16	0.072	6	966	50
Reverse order (ED′DB′CBA)	0.222	17	0.945	−249	0.082	7	0.665	16	0.075	6	679	65
Duplication (2×ABCB′DD′E)	0.142	7	0.970	−1,294	0.013	0	0.743	82	0.024	0	1,895	0
Random shuffled	0.891	100	0.963	100	0.954	100	0.763	100	0.868	100	19	100

[a]Swiss-Prot AC = P09871, A = Signal peptide (res. 1–15), B, B′ = CUB domains (res. 16–130, 175–290, respectively), C = EGF domain (res. 131–172), D,D′ = Sushi domains (res. 292–356, 357–423, respectively), E = Peptidase S1 (res. 438–688)

[b]Score of C1S with itself = 0%, score of C1S with randomly shuffled C1S = 100%

[c]AH gives abnormal percent values because the distance to itself and to its randomly shuffled sequence are similar.

688 residues consisting of a signal peptide (A), two CUB domains (B,B′), a EGF domain (C), two SUSHI domains (D,D′) and a trypsin-like catalytic domain (E) that is post-translationally cleaved from the precursor. The domain architecture of the native protein can be written as ABCB′DD′E and a hypothetical circular permutant can be written as DD′EABCB′. The results in Table 4.2 show that a reshuffling of the domains does not substantially affect the CBDs. Comparing the C1S precursor with its reshuffled counterparts in which the domain order is reversed, gives a PPM distance of 0.075, while the C1S compared to itself gives a value of 0.020 (The respective SW score values are 679 and 1,895; there is a substantial difference). In addition, the circular permutation of a long sequence has a smaller effect on the compression distance than on the SW score. This property of a CBM may be useful when looking for similarities between rearranged sequences.

4.3.3 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 a set of 131 sequences representing the essentially ubiquitous glycolytic enzyme, 3-phosphoglycerate kinase (3PGK). The sequences are available both as amino acid residues (358–505 residues

in length) and nucleotide residues (1,074–1,515) obtained from 15 archaean (consisting of 2 phyla), 83 bacterial (consisting of 3 phyla) and 33 eukaryotic species (consisting of 5 phyla). The dataset is freely available [32].

Experiment: An alphabet reduction was evaluated just on amino acids in such way that they were grouped into different groups and each amino acid residue was replaced by its group identifier. The Dayhoff classes ("AGPST, DENQ, HKR, ILMV, FWY, C") were obtained from [14] and here are referred to as Dayhoff6. Other classes were taken from Table 1 of [33] and were denoted by SB4, SB6, SB8, SB12, and SB16, respectively. The number in the 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 certain 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.

Evaluation: To evaluate how well a distance matrix reflects the groups, the ROC analysis [17] 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. Next, the ROC analysis was performed by plotting sensitivity versus 1-specificity at various threshold values, and the resulting curve was integrated to give an "area under the curve" (AUC) value. For a perfect ranking AUC = 1.0, while for a random ranking AUC = 0.5. The calculated AUC value can be interpreted as the probability of a similarity score for a pair from the same group being greater than the score for a sequence pair from a different group [18]. Here, the ROC analysis was evaluated on each distance matrix calculated by a CBD on each dataset constructed from the 3PGK by alphabet reduction, expansion and BWT, respectively. Figure 4.1a shows high correlation between the AUC-kingdom and AUC-phylum, hence in the following just the AUC-phylum is shown. In Fig. 4.1b the correlation between the compression ratio and the AUC-phylum is plotted, where the compression ratio cr is defined by

$$cr = \frac{avg.uncompr.size}{avg.compr.size} \cdot \frac{\lceil \log_2(alphabetsize) \rceil}{\log_2(256)}. \tag{4.14}$$

Here $\lceil . \rceil$ stands for the larger integer. The second constant term is a scaling factor intended to provide a fair comparison for compressors on 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 non-linear relationship was found between the compression ratio and the size of the alphabet (see Fig. 4.1c). 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

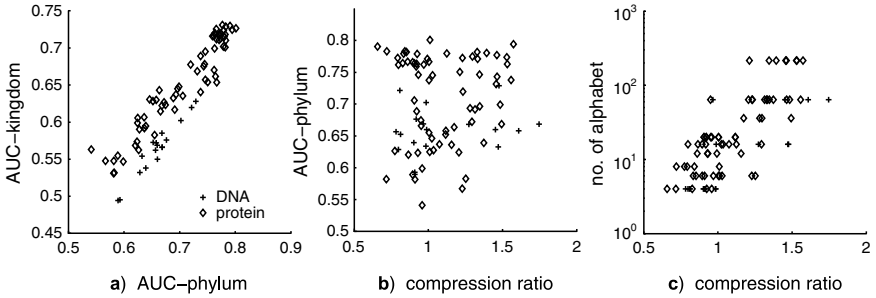

Fig. 4.1 The correlation between methods on nucleotide (*plus*) and amino acid (*diamond*) sequences from 3PGK. (**a**) The relationship between the AUC-phylum and AUC-kingdom; (**b**) the correlation between compression ratio and AUC; (**c**) the connection ratio is related to the alphabet size non-linearly

Table 4.3 AUC results for amino acid sequences with an alphabet reduction and BWT acting on 3PGK

	Original	SB16	SB12	SB8	SB6	SB4	Dayhoff6	BWT
ABC	0.738	0.766	0.771	0.783	<u>0.801</u>	0.764	0.745	0.655
AH	0.658	0.628	0.636	0.625	0.567	0.541	0.583	0.652
GC	0.763	0.759	0.746	0.773	0.762	0.779	0.767	0.689
LZW	0.666	0.620	0.623	0.626	0.589	0.582	0.581	0.599
PPM	0.764	0.762	0.766	0.783	0.779	0.790	0.782	0.706

The pairwise distance/similarity matrix also was calculated on the original amino acids sequences via naive distance and SW, and the AUC values we got were 0.685 and 0.797, respectively. The largest score is underlined.

alphabet reduced to 4–8 can improve the AUC values with some compressors, as Table 4.3 shows. Tables 4.4 and 4.5 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 perhaps because the sequence lengths were short.

4.3.4 Experiments on Protein Classification

To evaluate these CBDs for protein classification, we chose the SCOP40 database. For the train and test set division, the supervised cross-validation technique was applied, which is a selection of test and train sets that are based on known subtypes within a database [21]. A family with over five members was selected as the positive test and the rest of the superfamily was the positive train set. The negative sets were selected in a similar way. This approach provides a reliable estimation of

Table 4.4 AUC results for an alphabet expansion on 3PGK of a reduced alphabet

	SB8		SB6			SB4			Dayhoff		
	A[a]	AA[b]	A[a]	AA[b]	AAA[c]	A[a]	AA[b]	AAA[c]	A[a]	AA[b]	AAA[c]
ABC	0.783	0.746	0.801	0.668	0.794	0.764	0.672	0.699	0.745	0.738	0.774
AH	0.625	0.639	0.567	0.664	0.746	0.541	0.646	0.675	0.583	0.624	0.750
LZW	0.626	0.696	0.589	0.693	0.763	0.582	0.637	0.720	0.581	0.691	0.732
PPM	0.783	0.771	0.779	0.775	0.777	0.790	0.780	0.773	0.782	0.782	0.780

[a]The values were taken from Table 4.3
[b]Bi-gram residue compositions were used
[c]Tri-gram residue compositions were used
Here, GC compressor was not used. The largest score is underlined.

Table 4.5 AUC results for an nucleotide sequences with an alphabet expansion on 3PGK

	Original	AA[a]	AAA[b]	BWT
ABC	0.676	0.633	0.668	0.639
AH	0.589	0.660	0.669	0.593
GC	0.702	n.a.	n.a.	0.634
LZW	0.652	0.668	0.660	0.628
PPM	0.722	0.729	0.658	0.657

[a]Bi-gram residue compositions were used
[b]Tri-gram residue compositions were used
The pairwise distance/similarity matrix was also calculated on the original amino acids sequences via naive distance and SW, and the AUC values we got were 0.662 and 0.807, respectively. The largest score is underlined.

how an algorithm will generalise to a novel, distantly-related subtype of the known protein classes. Altogether, we obtained 55 classification tasks in this manner. The sequences were represented by the so-called Empirical Feature Map method, where a sequence X is represented by a feature vector $F_X = f_{x_1}, f_{x_2}, \ldots, f_{x_n}$. Here n is the total number of proteins in the training set and f_{x_i} is a similarity/distance score between sequence X and the ith sequence in the training set. For the underlying similarity measure, CBDs along with the BLAST and Smith–Waterman algorithms were applied.

Classification Methods: Nearest Neighbour (1NN) classification [15] is a technique whereby a query sequences is assigned to the priori known class of the database entry that was found most similar to it in terms of a log-likelihood ratio of the nearest positive and negative score via a distance/similarity measure [20]. Artificial Neural Networks (ANNs) were then employed, whose structure consisted of one hidden layer with 40 neurons and the output layer consisted of one neuron. For each neuron, the log-sigmoid function was used as the transfer function and the Scaled Conjugate Gradient (SCG) algorithm was used for training [6]. The package we applied was the Neural Network toolbox version 5.0 of Matlab. The Support Vector Machines (SVMs) [34] method employed was LibSVM [10]. In our experiments, the Radial

Table 4.6 The AUC results on protein classification with several classifiers and featuring method applied on SCOP40

Method name[a] (AUC[b])	1NN	SVM	RF	LogReg	ANN	Avg
ABC (0.699)	0.726	0.843	0.779	0.806	0.834	0.798
AH (0.690)	0.711	0.877	0.824	0.751	0.800	0.793
GC (0.674)	0.644	0.775	0.691	0.753	0.769	0.726
LZW (0.769)	0.751	0.856	0.821	0.718	0.794	0.788
PPM (0.700)	0.798	0.851	0.800	0.813	0.583	0.823
naive (0.597)	0.653	0.882	0.848	0.786	0.837	0.801
BLAST (0.684)	0.775	0.905	0.697	0.836	0.902	0.823
SW (0.823)	0.956	0.946	0.823	0.913	0.897	0.907

[a]Method used in the vectorization step
[b]The AUC values in parentheses were obtained via distance matrices in the way described in Sect. 4.3.3.

Basis Function kernel was used, where the width parameter σ was the median Euclidean distance from any positive training example to the nearest negative example. The Logistic Regression (LogReg) [29] method was part of Weka versions 3–4 [37]. Finally, the Random Forest (RF) technique is a combination of decision trees such that each tree is grown on a bootstrap sample of the training set. For each node, the split is chosen from $m \ll M$ variables (M being the number of dimensions) on which to base the decision [7]. In our experiments, 50 trees were used and the number of features m was set to $\log(l+1)$, where l is the number of training patterns. The RF was part of Weka versions 3–4. For an evaluation of the classifier, ROC analysis was performed by ranking the test object using their score obtained by a learned model. The results are shown in Table 4.6.

4.3.5 Sequence Length and Sequence Complexity

It is known that short sequence similarities are more difficult to locate than long ones. This tendency also appears to be valid for CBDs analysed here. For example, when we plot the classification performance measure AUC for the 55 SCOP40 families as a function of the average sequence length, the low values are predominantly found in the shorter superfamilies (Fig. 4.2). This trend is quite similar for the BLAST and Smith–Waterman algorithms as well (http://www.inf.u-szeged.hu/~kocsor/CBM05).

A dataset of high and low complexity sequences was produced by taking human proteins from the KOG database [23], processing them with the SEG program written by J. Wootton [38], and applying the parameter values of window length = 45, trigger complexity = 3.25, and extension complexity = 3.55, as recommended by the author. Segments with length in the range of 20–1,000 amino acids were chosen for further analysis. From a total of 163,473 high-complexity and 53,849

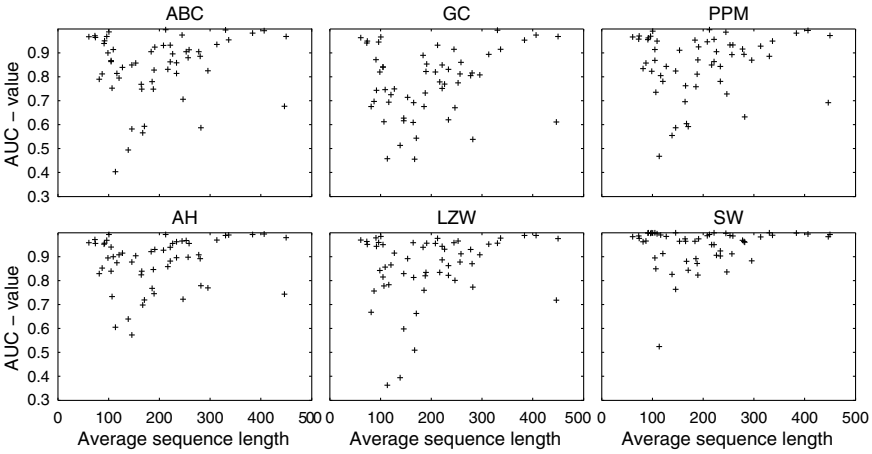

Fig. 4.2 The dependence of classification performance on the average sequence length in the SCOP40 families

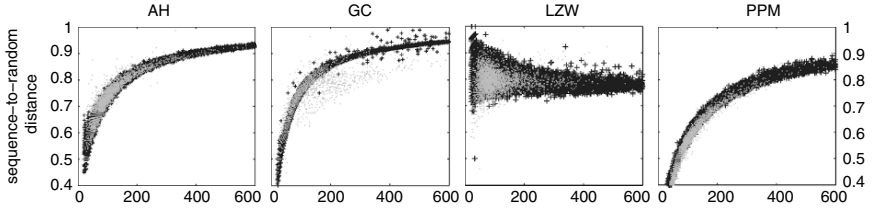

Fig. 4.3 A comparison of native sequences with their random shuffled counterparts in KOG and high- (*grey*) and low-complexity (*black*) segments of human proteins as a function of their length. The CBD distances of a sequence from its random-shuffled counterparts should give a value close to 1.00

low-complexity regions, we randomly selected 8,859 and 3,772 sequences, respectively, for an analysis, for which the results are shown in Fig. 4.3. Low-complexity segments correspond to non-globular regions in proteins, characterised by a biased amino acid composition and/or a repetitive amino acid sequence. Such regions are well known for obscuring the detection of biologically important similarities [38]. We had two particular reasons for testing CBDs on low-complexity regions: (1) repetitive character-sequences can be better compressed than average sequences; (2) our datasets of natural sequences are composed of proteins that are predominantly globular, so they are expected to contain few low complexity regions. We found that the distributions of CBDs values for low- and high-complexity proteins were not markedly different (see Fig. 4.3); however, this seems to be analogous to our earlier finding that the compression ratio is independent of the AUC measure (Fig. 4.1b).

4.4 Conclusions

One of the main problems of computational sequence analysis is the fact that sequence similarity groups are highly variable in terms of most parameters, including the number of members, the degree of within-group similarity and separation from other groups. Consequently, a method that performs well on one group may perform worse on another and vice versa, and very often there are no clear tendencies in the results. This was also true for our study, but we were able to identify a few clear tendencies.

CBDs perform less well than substructure-based comparisons such as the 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. Moreover, CBDs are less sensitive to domain rearrangement than the alignment-based BLAST or Smith–Waterman. Curiously, some combinations of CBDs approach with BLAST comparison exceed the performance of Smith–Waterman in protein classification [22]. Unfortunately CBDs and alignment-based measures depend on the length of the sequences.

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 short. Moreover, the well-tested BWT method of data compression could not be used here. Similar findings were also described in [28].

In general, in the results 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 the partial matching-based algorithms like PPM and GC improved their performance. This could mean that mutations of amino acids found in the same cluster do not really change their structure and function(s). Furthermore, a compressor applied to reduced alphabet sequences can be viewed as a lossy compressor. An alphabet expansion with bi-gram and trigram 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.

Acknowledgements A. Kocsor was supported by the Jnos Bolyai fellowship of the Hungarian Academy of Sciences. Work at ICGEB was supported in part by grants from the Ministero dell.Universita' e della Ricerca (D.D. 2187, FIRB 2003 (art. 8), "Laboratorio Internazionale di Bioinformatica").

References

1. Abel, J.: Improvements to the burrows-wheeler compression algorithm: After bwt stages (2003)
2. Ágoston, V., Kájan, L., Carugo, O., Hegedűs, Z., Vlahovicek, K., Pongor, S.: Concepts of similarity in bioinformatics. In: D. Moss, S. Jelaska, S. Pongor (eds.) Essays in Bioinformatics. IOS, Amsterdam (2005)
3. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J Mol Biol **215**(3), 403–410 (1990)
4. Andreeva, A., Howorth, D., Brenner, C.: Scop database in 2004: refinements integrate structure and sequence family data (2004)
5. Bennett, C.H., Gács, P., Li, M., Vitanyi, P.M.B., Zurek, W.H.: Information distance. IEEETIT: IEEE Trans Inform Theory **44**, 1407–1423 (1998)
6. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press, Oxford (1996)
7. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
8. Burrows, M., Wheeler, D.J.: A block-sorting lossless data compression algorithm. Tech. Rep. 124, 130 Lytton Avenuve, Palo Alto, CA, 94301 (1994)
9. Cai, H., Kulkarni, S.R., Verdú, S.: Universal entropy estimation via block sorting. IEEE Trans Inform Theory **50**(7), 1551–1561 (2004)
10. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001)
11. Chen, X., Kwong, S., Li, M.: A compression algorithm for DNA sequences and its applications in genome comparison. In: RECOMB, p 107 (2000)
12. Cilibrasi, R., Vitanyi, P.M.B.: Clustering by compression. IEEE Trans Inform Theory **51**(4), 1523–1545 (2005)
13. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley, New York (1991)
14. Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C.: A model of evolutionary change in proteins. In: M.O. Dayhoff (ed.) Atlas of protein sequence and structure, vol. 5, 345–358. National Biomedical Research Foundation, Washington, D.C., (1978)
15. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley Interscience, New York (2000)
16. Ferragina, P., Giancarlo, R., Greco, V., Manzini, G., Valiente, G.: Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment. BMC Bioinformatics **8**, 252 (2007)
17. Gribskov, M., Robinson, N.: Use of receiver operating characteristic (roc) analysis to evaluate sequence matching. Comput Chem **20**, 25–33 (1996)
18. Hanley, J.A., Mcneil, B.J.: The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology **143**(1), 29–36 (1982)
19. Henikoff, S., Henikoff, J.G., Pietrokovski, S.: Blocks: a non-redundant database of protein alignment blocks derived from multiple compilations. Bioinformatics **15**(6), 471–479 (1999)
20. Kaján, L., Kertész-Farkas, A., Franklin, D., Ivanova, N., Kocsor, A., Pongor, S.: Application of a simple likelihood ratio approximant to protein sequence classification. Bioinformatics **22**(23), 2865–2869 (2006)
21. Kertész-Farkas, A., Dhir, S., Sonego, P., Pacurar M., Netoteia, S., Nijveen, H., Kuzinar, A., Leunissen, J., Kocsor, A., Pongor, S.: Benchmarking protein classification algorithms via supervised cross-validation. J Biochem Biophys Methods **35**, 1215–1223 (2007)
22. Kocsor, A., Kertész-Farkas, A., Kaján, L., Pongor, S.: Application of compression-based distance measures to protein sequence classification: a methodological study. Bioinformatics **22**(4), 407–412 (2006)
23. Koonin, E.V., Fedorova, N.D., Jackson, J.D., Jacobs, A.R., Krylov, D.M., Makarova, K.S., Mazumder, R., Mekhedov, S.L., Nikolskaya, A.N., Rao, B.S., Rogozin, I.B., Smirnov, S., Sorokin, A.V., Sverdlov, A.V., Vasudevan, S., Wolf, Y.I., Yin, J.J., Natale, D.A.: A comprehen-

sive evolutionary classification of proteins encoded in complete eukaryotic genomes. Genome Biol **5**(2) (2004)
24. Li, M.: Information distance and its applications. In: O.H. Ibarra, H.C. Yen (eds.) CIAA, *Lecture Notes in Computer Science*, vol. 4094, 1–9. Springer, Berlin (2006)
25. Li, M., Chen, X., Li, X., Ma, B., Vitányi, P.: The similarity metric. In: SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, 863–872. Society for Industrial and Applied Mathematics, Philadelphia (2003)
26. Li, M., Vitanyi, P.: An introduction to kolmogorov complexity and its applications, 2nd edn. Springer, Berlin (1997)
27. Li, M., Vitányi, P.M.: Mathematical theory of thermodynamics of computation. Tech. rep., Centre for Mathematics and Computer Science, Amsterdam, The Netherlands (1992)
28. Nevill-Manning, C.G., Witten, I.H.: Protein is incompressible. In: DCC '99: Proceedings of the Conference on Data Compression, p. 257. IEEE Computer Society, Washington, DC, USA (1999)
29. Rice, J.C.: Logistic regression: An introduction. In: B. Rhompson (ed.) Advances in social science methodology, vol. 3, 191–245. JAI, Greenwich (1994)
30. Schweizer, D., Abu-Mostafa, Y.: Kolmogorov metric spaces. Manuscript, Computer Sciences, 256–80, California Institute of Technology, Pasadena, CA 91125 (1998)
31. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J Mol Biol **147**, 195–197 (1981)
32. Sonego, P., Pacurar, M., Dhir, S., Kertész-Farkas, A., Kocsor, A., Gáspári, Z., Leunissen, J.A.M., Pongor, S.: A protein classification benchmark collection for machine learning. Nucleic Acids Res **35**(Database-Issue), 232–236 (2007)
33. Susko, E., Roger, A.J.: On reduced amino acid alphabets for phylogenetic inference. Mol Biol Evol **24**, 2139–2150 (2007)
34. Vapnik, V.N.: The nature of statistical learning theory, 2nd edn. Springer, Berlin (1999)
35. Vinga, S., Almeida, J.: Alignment-free sequence comparison-a review. Bioinformatics **19**(4), 513–523 (2003)
36. Willems, F.M.J., Shtarkov, Y.M., Tjalkens, T.J.: The context-tree weighting method: basic properties. IEEE Trans Inform Theory, 653–664 (1995)
37. Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco (1999)
38. Wootton, J.C.: Non-globular domains in protein sequences: automated segmentation using complexity measures. Comput Chem **18**(3), 269–285 (1994)
39. Zurek, W.H.: Thermodynamic cost of computation, algorithmic complexity and the information metric. Nature **341**(6238), 119–124 (1989)