Alignment-Free Probabilistic Proteomics: Patterns to Functionality

Ewa M. Grela

Doctor of Philosophy
Aston University
December 2022

Alignment-Free Probabilistic Proteomics:

Patterns to Functionality

Ewa M. Grela

Abstract

Major Histocompatibility Complexes class I (MHC I), known as the Human Leukocyte Antigen class (HLA I) in humans, are proteins responsible for antigen presentation to T-lymphocytes. MHCs interact with T Cell Receptors (TCRs). They serve as crucial immune regulators for vertebrates. The three main sub-classes of the HLA class I proteins (HLA-A, HLA-B, HLA-C) are encoded in three different loci. Therefore (as genes within MHC I class are co-dominant), an individual has up to six different alleles of HLA class I protein present on the surface of their cells. The genetic diversity of HLA class I in the human population can be linked to the differentiated immunological response.

Based on a combination of established bioinformatic and machine learning tools, we have addressed the challenge to analyse HLA class I protein data-set in order to determine their ability to bind to specific antigens. To achieve this, we have created three dimensional models of HLA class I variants using homology modelling techniques. These have then been placed in three dimensional grids in order to calculate the electrostatic fields around the protein domains. The resultant multi-dimensional data were then analysed using the unsupervised machine learning techniques: both linear Principal Component Analysis (PCA), and nonlinear ones: the auto-encoder neural network (NLPCA) and the Gaussian Process Latent Variable Model (GPLVM). The methods used, accomplished the task of distinguishing between the HLA proteins sub-classes (A, B and C). In addition, the results obtained with the GPLVM dimensionality reduction suggested, that the electrostatic potential calculation may add information necessary to identifying HLA super-types. However, this method by itself, it is not robust enough to be independently conclusive.

The sequence alignments methods are not free from assumptions. Results they provide are influenced by the choice of a substitution matrix, as the numerical values are assigned to the differences between compared biomolecules' primary structures. The increase of the number of known sequences, related to the development of the Next Generation Sequencing techniques created additional challenge, that is a computational time required.

As an alternative to the sequence alignment, we implemented the methods from time series analysis, information and chaos theory, and statistical physics to translate information from amino acid sequences into numerical vectors, in order to predict the similarity in proteins structures and functions.

We transformed a data set of 9693 amino acid sequences belonging to 100 protein families by replacing each amino acid with numerical values representing its physicochemical and biochemical properties, and based on that, calculated multiple multidimensional vectors of non-alignment protein descriptors with measures such as approximate and sample entropy or persistence, Hurst and Lyapunov exponents. The supervised learning Linear Discriminant Analysis technique, used to assess the ability of the developed protocols to correctly assign proteins to their functional groups, showed an efficiency up to over 99%.

I would like to thank my supervisors, Dr. Amit Chattopadhyay and Dr. Darren Flower, for their invaluable insight, deep knowledge and most of all, patience and support they showed me during my PhD study. I am also grateful to my co-supervisor, Dr. Michael Stich, for his support and help. Finally, no work is possible without the proper tools. I would like to express my most sincere gratitude to Alexander Brulo, who went above and beyond to resolve multiple technical issues I have encountered.

This Ph.D. project was funded by the Krebs Memorial Scholarship awarded by Biochemical Society

Publications list

Grela E, Stich M, Chattopadhyay A K (2018), Epidemiological impact of waning immunization on a vaccinated population, European Physical Journal B 91, 267.

Grela E M, Flower D R, and Chattopadhyay A K (2022), Alignment-independent protein sequence comparison: The Entropy Descriptors. Submitted to the Physical Review Letters.

Grela E M, Flower D R, Chattopadhyay A K (2023), Evaluation of the standard non-alignment protein descriptors. In preparation.

Grela E M, Flower D R, Chattopadhyay A K (2023), Alignment-free protein mapping: a novel statistical learning gateway. In preparation.

Contents

1 Overview of the Research Problem

The genome of all known living organisms is made of DNA. Three nucleotides encode a specific amino acid. Amino acids form proteins by folding into complex three-dimensional functional structures (Figure 1). This process is complicated, not fully understood, and the prediction of the final result is still challenging, in spite of astonishing progress achieved in recent years [1].

Another approach, laboratory based techniques for visualization of proteins three dimensional structures, such as crystallography [2], are time and resources consuming, and what they provide is still imperfect. They are burdened with an error resulting even from the fact that proteins are dynamic structures, and the above methods give us a snapshot at a certain point in time. In silico protein visualization such as homology modelling, though has been proven effective in solving countless biological problem [3] requires proteins with a similar structure as template.

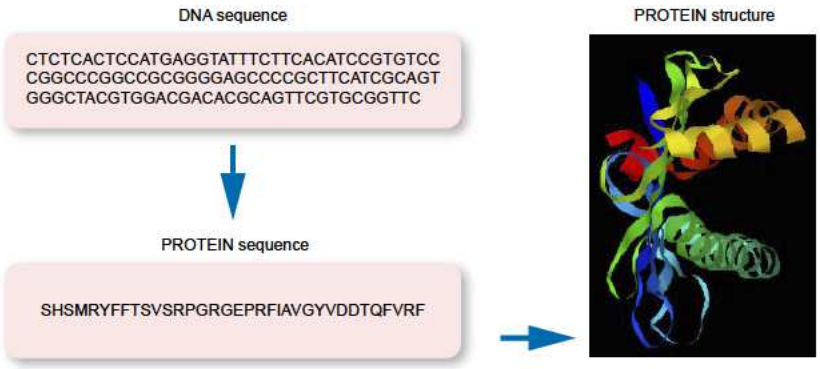

Figure 1: Transcription and translation of information contained in the genome of living organisms.

The Next Generation Sequencing techniques led to the situation when information about DNA sequence (and due to unambiguous encoding, protein sequence) is easy to obtain. The main and the most popular techniques used for understanding of the protein homology are based on sequence alignment algorithms such as BLAST (basic local alignment search tool) [4]. This algorithm focuses on the study of differences and similarities between two sequences and on that basis of the phylogenetic distance between the proteins encoded by these DNA chains is calculated. Despite being extremely powerful and successful tools, methods based on sequence alignment have their limitations.

What is worth noting, is that the sequence alignments methods are not absolutely free from assumptions. Analysis of similarities and differences between the two DNA chains not only find these differences, but also assign numerical values to them. Therefore the result is dependent on the chosen scale [5]. The increase of the number of known sequences also creates an additional challenge, that is a computational time required for comparison [6, 7].

The question about the mechanisms governing the functioning of living organisms essentially boils down to a series of questions about the functionality of individual proteins. One could risk a statement that solving all health related problems could lay in proper understanding of the gene-gene (or protein-protein) and gene-environment (or protein-compound) interaction. And though such a statement would probably still be a oversimplification, the proper knowledge of molecular mechanisms that governed those interaction cannot be overestimated. Understanding in detail protein functionality is the main question that lays before molecular biology, biotechnology and bioinformatics. The last one of those disciplines often uses techniques based on finding differences and similarities between genes and, consequently proteins.

However, differences and similarities in DNA do not translate unambiguously even differences in the amino acid sequence, let alone differences in the three-dimensional protein structure or its function. The extent to which a mutation in a nucleotide chain changes the activity of the protein encoded by this chain depends not only on the type of mutation, but also on which particular protein we are dealing with and in what specific region of the chain this mutation occurred. For example, sequence alignment-based methods are not effective when either recombination or horizontal transfer has occurred [8]. Recombination is the exchange of genetic information between two DNA (RNA) chains, that belong to either one or more organisms. It results in a new genotype without the increase of the population's genetic pool. Horizontal gene transfer is the mechanism allowing the transfer of genetic information between organisms, that does not involve reproduction. It can occur both between organisms belonging to the same, as well as to the different species. Therefore, it can and often does increase the genetic pool. The most well-known example would be the transfer of genes that are responsible for the antibiotic resistance in bacteria. Another challenge for the sequence alignment-based methods is the fact that regulatory regions of the DNA chain are, in general, not highly conserved, except for some functional regions [6, 7, 9].

There are single gene diseases that are caused by a single mutation in the DNA. The protein product of the faulty gene is faulty itself therefore its functionality is impaired. This leads to the pathology of the functioning of the whole organism. But this kind of situation is an exception rather than a rule. Gene expression can be modified by other proteins and by an environmental factors. In most cases, even if the exact location of the genome mutation is known, there is no definite way to predict its possible consequences. But it does not necessarily mean that the information is not there. Statistics show that susceptibility for countless diseases is inherited, though exact genetic mechanisms are still not known. This leaves plenty of open questions, essential for our understanding of the functioning of living organisms, as well as for finding solutions of many problems related to public health (e.g. antimicrobial resistance, vaccination design, individualized anti-cancer therapy).

Currently, methods based on sequence alignment are mainly used to solve these types

of problems. They assume that differences between examined protein (DNA, RNA) chains that share the common ancestor (therefore have functional similarities), are the results of mutations, and that these mutations accumulate with time. The main disadvantage of this approach is the need for proteins with a similar structure and known function.

The alignment-free methods can be defined as ones that do not use sequence alignment to determine sequence similarity [9]. What many, if not all, of the the alignment-free methods have in common, is that all of them are based on creating a way to assign a single or a set of numerical values to each examined protein (Figure 2). In further analysis these descriptor vectors, not sequences themselves are compared with each other. In some cases, the algorithm of finding descriptors can be computationally consuming. But even if that it is the case, it needs to be calculated just once, as opposed to alignment-based methods, where a computationally complex algorithm (one that involves a large number of operations) is used whenever comparison with a new protein is required. The number of elementary operations that need to be performed when two vectors are compared with each other depends on algorithm used, but can be as low as the vector length. For the BLAST algorithm it is larger than the product of the compared sequences lengths [14].

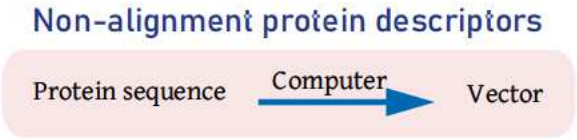

Figure 2: Illustration of alignment-free protocol.

These most popular alignment-free methods are based on word count [8, 10, 11, 12, 13, 15, 16] or length of common sub-strings [15, 17]. However in recent years we can observe the increased popularity of methods based on different mathematical and statistical concepts such as information theory [18], Markov chain [19, 20], or signal processing [21, 22].

The alignment-free approach owes its growing popularity not only to the fact that it can be much more computationally efficient but mainly because of reports of its effectiveness in areas such as: industrial biotechnology [23], cervical cancer risk prediction [24], potential drug-target interaction [25]. It was shown that alignment-independent methods work as well as the dependent ones when applied to prediction of host for the viral infection [26]. The effectiveness of the alignment-independent methods was assessed for the independent sets containing rabies, corona-viruses and influenza A viruses, and word count method was used. The k-mer method was also applied for the task of the viral whole genome classification [27]. The data set containing 3905 complete viral genomes was analysed, and the study was successful in finding the optimal k-mer length required to recreate the known evolutionary relationships. The alignment-independent methods based on the amino acid composition, amino acid physicochemical properties and proteins secondary structures were considered

as a tool for the search of the potential AHTPs (anti-hypertensive peptides), that could be obtained from natural sources [28]. Two pairs of data sets (913 AHTPs vs 913 not AHTPS and 386 AHTPs vs 386 not AHTPS) were used to validate the designed protocol, and accuracy of 85.8% and 90.4% respectively was achieved. Methods based on composition and auto-correlation were applied to the recognition of disease resistance (DR) proteins in plants [29]. The positive data set containing 400 known DR proteins was tested against 100 negative data sets (containing between 400 and 4000 non DR proteins). The achieved specificity reached 96.9±1.5% and sensitivity 86.4±4.0%. Physicochemical properties of amino acids combined with the amino acid composition were successfully applied in identifying functional similarities between the effector proteins belonging to two distinct types of plant pathogens: eucaryotic and bacterial ones [30]. This is particularly significant result, as these microorganisms are not evolutionarily closely related and the examined proteins do not show substantial sequence similarity.

1.1 The Major Histocompatibility Complex class I

The Major Histocompatibility Complex class I (MHC-I) proteins are present on the surface of most vertebrate nucleated cells. They assume a crucial regulatory role within the organism's immune system, by presenting degraded fragments of intracellular proteins (8-10 amino-acid peptide, referred to as antigen) to T lymphocytes. The T cells present antigens and subsequently identify them as either threatening or non-threatening. Thus, MHC proteins serve the role of an immune system guard, alerting the system about any possible intruder. This can be an intracellular pathogen, or a cancer cell, or even a transplanted cell [31, 32].

MHC class I proteins are widely polymorphic, when it comes to the exact structure of their active site [31, 32]. Therefore, not only any particular protein can bind with various antigens, but also antigens can bind with more than one variant of an MHC I molecule.

They are transmembrane proteins built out of three extracellular domains ($\alpha 1$, $\alpha 2$, $\alpha 3$), a transmembrane region, and a cytoplasmic tail (Figure 3). The $\alpha 3$ domain is non-covalently bonded to $\beta 2$ microglobulin (external peptide, not coded inside MHC coding DNA region). The $\alpha 1$ and $\alpha 2$ form one structural domain, where eight β-sheets support two α-helices to antigen binding active sites, that interact with the T-cell receptors (TCR) of the T lymphocytes. TCRs dock onto a surface formed from the bound peptide and the top surface of the MHC peptide-binding domain [31, 32].

The human MHC I proteins is known as the *Human Leukocyte Antigen class I* (HLA I) complex [31, 32]. This name is related to the fact that they were first discovered in the context of transplant rejection mechanism research. The three main subclasses of the HLA class I proteins (*HLA-A*, *HLA-B*, *HLA-C*) are encoded in three different alleles [32]. Therefore (as genes within MHC I loci are codominant), an individual has up to six different alleles of HLA I protein present on the surface of their cells.

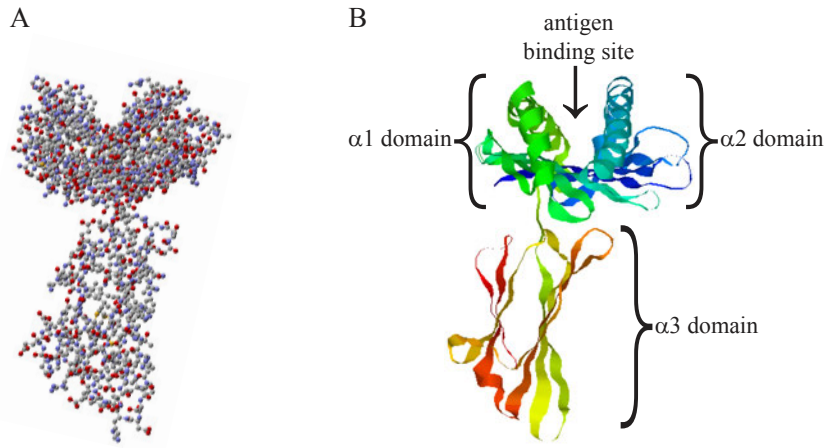

Figure 3: The three-dimensional structure of the HLA class 1 A protein. A: Presentation of individual atoms. B: Secondary structure.

The Human Leukocyte Antigen (HLA) genetic diversity in the human population can be linked to differentiated immunity to viral infections [33], vaccination effectiveness [34, 35], individual susceptibility to cancer or autoimmune diseases [36] and risk of the transplant rejection [37]. Therefore defining and understanding the functional differences and similarities between variants of HLA protein is of crucial importance.

1.2 Research goals

In our research we wanted to focus on alternative to sequence-alignment approach, to tackle problems related to the complex relationship between protein sequence and function.

- Based on a combination of established bioinformatic and machine learning tools, we addressed the challenge of analysing HLA I protein data-sets in order to determine their ability to bind to specific antigens. Our goal was to develop an effective protocol, that starts with probabilistic prediction of similarities and differences in the function of individual HLA proteins, and then classify them based on that prediction.

- Subsequently, as an alternative to the sequence alignment, we implemented the methods from time series analysis, information and chaos theory [1] and statistical physics to translate information from amino acid sequences into numerical vectors, in order to predict the similarity in proteins structure and function.

We hypothesise that in order to be able to classify a newly discovered protein, there is no need to compare it with all known and previously classified sequences using the alignment

[1] A branch of science that looks for patterns that underlie seemingly random and chaotic behavior. It focuses on systems that are extremely sensitive to even small changes in initial conditions and tries to predict the outcome of these changes.

methods. Instead, we can create a multi-dimensional vector for the newly discovered protein, and then compare it with the previously calculated and stored vectors representing known proteins, or even with a set of single vectors, each representing a whole protein family. Although the process of computing all the necessary descriptor vectors for a given protein can be time-consuming, a novelty of this approach lies in the fact that it is enough to do the calculations only once. The descriptor values can be stored in a database in a way similar to three-dimensional structures, alongside codes that calculate descriptor values. The computational cost needed to compare descriptor vectors can be substantially lower than the computational cost associated with alignment-based methods of protein classification, where each new sequence is compared with sequences of known proteins.

2 Systems modelling of peptide presentation in immunology and homeostasis

2.1 Data acquisition and preparation

The *template* (pdb) files, that represent three-dimensional proteins structures over a range of conformations, known respectively as follows: 1I4F for subclass HLA-A, 1AGD for HLA-B and 1IM9 for HLA-C, were taken from the *RCBS Protein Data Bank* [38] database. The models of 1I4F and 1AGD proteins contain 275 amino-acids. The 1IM9 subclass contains 276 amino-acids. But only the first 275 ware taken into consideration in further analysis.

The amino-acid sequences of known variations of the HLA class I proteins were downloaded from the *Immuno Polymorphism Database* [39]. The database contained 3489 sequences belonging to the subclass A, 4454 sequences belonging to the subclass B, and 3290 sequences corresponding to the subclass C. From these sets, 1163, 1537 and 1153 sequences were respectively chosen for further analysis (*target* sequences). The rest of the proteins were excluded based on results obtain from the BLAST [40] structure alignment to the sequence of the corresponding template protein. The example of the BLAST structure alignment result is given in appendix A.

The criterion was to determine if the target aligns to the entire template sequence. For the sequence, we allowed for up to one missing amino-acid at the beginning and up to two at the end of the protein sequence. This allowed detection of incomplete molecules and consequent removal from the data-set. Some of the target sequences had additional Amino acids not present in the template structure. Those Amino acids were removed from the sequences.

The next step in the data preparation process concerned Protein Homology Modelling. The process consists of predicting the structure of a protein of known sequence and unknown three dimensional structure on the basis of similarly sequenced protein and known three dimensional structure. The Modeller application was used for this part of the analysis [41]. An example of the Modeller input files can be found in Appendix A. Three different homology models were prepared.

- TYPE 1: Amino-acids from positions 2-182 of the template and corresponding amino-acids from the target were taken into account (α1 and α2 domains). Only one template was used for each model. The choice of the template was based on the affiliation of the target subclass.

- TYPE 2: Amino-acids from positions 2-182 of the template and corresponding amino-acids from the target were taken into account (α1 and α2 domains). All three templates were used for each model (multi-template homology modelling).

- TYPE 3: All 275 amino-acids of the template and corresponding amino-acids from the target were taken into account (α1, α2 and α3 domains). Only one template was used

for each model. The choice of the template was based on the target subclass affiliation.

The following three paragraphs summarise a flowchart of data processing using established, mostly open sourced, software:

1. Homology modelling was used to generate the pdb files containing three dimensional structures of target HLA class I proteins.

2. The molecular structures obtained from the Protein Data Bank often lack the hydrogen atoms' coordinates, as well as some of the parameters required to perform the electrostatic potential calculations. To address these issues the pdb2pqr application[43]] was used. Firstly, hydrogen atoms were added, then charges were assigned to the individual atoms depending on which residue they belong to and based on the PARSE force field.

3. The Adaptive Poison Boltzmann Solver (APBS) application was used to perform electrostatic potentials calculations [44]. The target proteins were placed inside a three-dimensional grid of points focusing on the target area. An example of the APBS input file is given in Appendix A. Two different grids were used for TYPE 3 models and one for TYPE 1 and TYPE 2 models. Details can be found in Table 1. APBS generates a file that contains the electrostatic potential value for each grid point, therefore a multidimensional vector of those values was assigned to each target protein. Due to the limitations of computing power, not all grid points were included in the analysis.

The two subsets for GRID I:

- Subset 1: containing points from the gap between the helices, all points in all directions, 4123 points (variables) in total.

- Subset 2: containing points around the helices and between them, every fourth point in each direction, 4250 points (variables) in total.

Unique subset for GRID II:

- Subset 3: containing every fourth point in each direction, 4913 points (variables) in total.

2.2 Dimensionality reduction methods

The collected data is multidimensional. The structure and origin of observations gives us reason to suspect, that at least some of them, could be inter-dependent. The combination of these two factors makes data visualization, coupled with appropriate multivariate analysis, imperative.

	GRID I	GRID II
Size in $\mathring{A}$: $(x \times y \times z)$:	$36 \times 64 \times 64$	$48 \times 32 \times 32$
Number of points: $(x \times y \times z = total)$:	$97 \times 65 \times 65 = 409825$	$33 \times 33 \times 33 = 35937$
Focus on HLA I domens:	$\alpha 1$ and $\alpha 2$	$\alpha 3$
Used for models TYPE:	1, 2, 3	3

Table 1: Grids used in the electrostatic potential calculation.

The statistical protocol combining these two features popularly goes under the name of *dimensionality reduction* [45, 46, 47, 48]. The target here is to project high dimensional data $Y \in R^d$ (d is the number of observed variables) into a lower dimensional subspace $X \in R^q$ (q is the number of projected variables, $q \ll d$), typically to outline a minimalist representation of a higher dimensional complex system. Those projected variables $x_i \in X$ are often called the *latent variables*.

The dependencies between observed variables and latent variables can be written as:

$$y_i = \gamma(x_i),$$

where y_i is the d-dimensional vector representing the i^th observation and x_i is the q-dimensional vector of latent variables for the i-th observation. The function γ specifies the relation between data space and latent space.

2.3 Principal Component Analysis

Principal Component Analysis (PCA) has traditionally been the most popular architecture to analyse such data structures [45]. It has been successfully applied in many scientific studies, such as face recognition algorithm [49], handwriting recognition protocols [50], genome sequences analysis [51], and in many others.

The PCA algorithm works by finding the linear projection from a latent space into its conjugal data space. This necessitates a linear relationship between the explicit and latent variables:

$$y_i = W x_i,$$

where W is a $d \times q$ matrix.

PCA transforms the observation space in such a way that the hidden variables (principal components, identified as the eigenvectors associated with the largest eigenvalues) represent the percentage variability in the data, starting with the highest. It is possible that the

number of principal components is smaller than the number of observed variables. In that case, the PCA method can be perceived as an explicit dimensionality reduction problem. More often, though, the number of principal components is equal to the number of the observed variables. But some subset of the latent variables reproduces the variability of the data to an extent sufficient for understanding the observed phenomena or perceiving patterns which were previously imperceptible [45]. We may add that while PCA is not necessarily the most accurate dimensionality reduction procedure, often it is used as the first step analysis due to its inherent simplicity and process linearity.

There are two algorithms that can perform Principal Component Analysis. One is based on eigenvalue decomposition of the covariance matrix while the other depends on a singular value decomposition of the observation matrix. In our analysis we used the second algorithm, as inbuilt within the Matlab architecture [52].

2.3.1 PCA results for TYPE 1 models: The single-template models of α1 and α2 domains

For the TYPE 1 models, amino-acids from positions 2-182 of the template and corresponding amino-acids from the target (the α1 and α2 domains) were modelled. Only one template was used for each model. The choice of the template was based on the target subclass affiliation.

Principal Component Analysis was then performed separately for the different types of points belonging to subsets 1 and 2 of GRID 1. Figure 4 shows results involving points positioned outside the Van der Waals sphere, which represents atom surface. A clear division can be observed for the subset 2 for the two highest ranked PCA components (Figure 4 top right plot). For subset 1, the clearest division occurs for components 2 and 3, but it still shows non-zero overlap (Figure 4 left panel).

The PCA visualisation of all points in the subsets, points positioned inside the Van der Waals sphere and points positioned outside the Van der Waals sphere, but within the 2Å distance from it, as well as the variance from the first three PCA components can all be found in Appendix B.

2.3.2 Quantitative measurement of The HLA subclasses separation

Due to a lack of complete separation between variables describing different subclasses of HLA protein, we measured the overlapping area. First, we divided the two-dimensional results' subspaces into $n \times n$ equal squares. We then defined a separation measure using the following three-dimensional vector:

$$K_n^{i,j} = [K_A, K_B, K_C],$$

where i, j are the components considered and

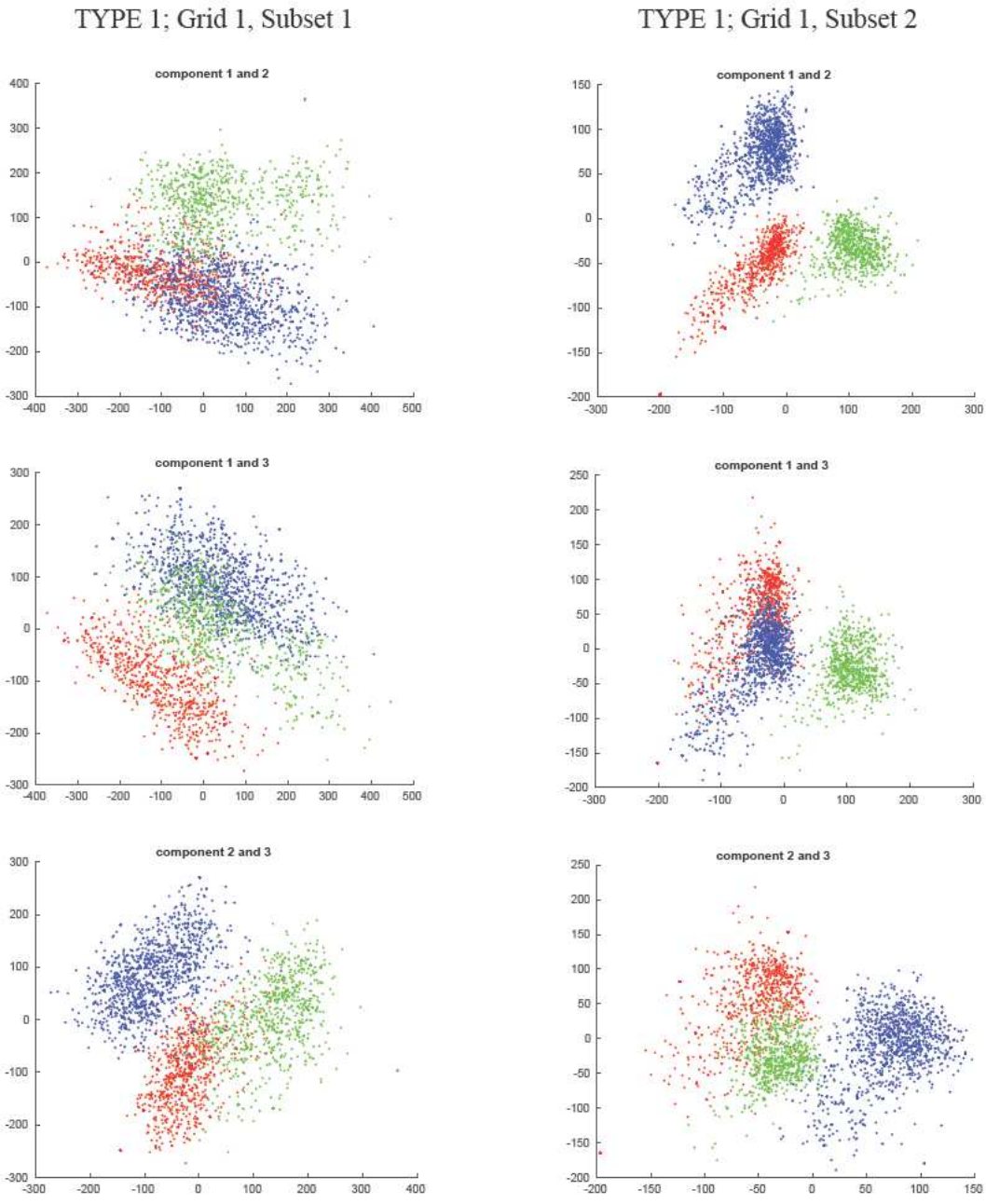

Figure 4: The PCA visualization of the TYPE 1 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere.

$$K_X = \frac{X_{\text{exclusive}}}{X_{\text{inclusive}}},$$

$X \in \{A, B, C\}$; $X_{\text{exclusive}}$ being the number of squares exclusively comprising points representing proteins of the chosen subclass (HLA-A, HLA-B, HLA-C); $X_{\text{inclusive}}$ is the number of squares with points representing the proteins of the chosen subclass.

Note that choosing too large or too small number of squares is likely to cause measurement errors. In the first case, the separation measure may not be sensitive enough and is likely to overlook situations where points from the different subclasses are mixed but the density of points distribution is small. For a too small number of squares, the separation measure may be too sensitive and may result in situations where points from different subclasses are separated but close to each other. Therefore the measure K, calculated using parameters $n = 50$ and $n = 100$, will be regarded as our optima to be used to analyse the separation of the HLA subclasses. The results for the TYPE I model data points, located outside the Van der Waals sphere (subset 1) can be found in table 2.

PCA Eigenvectors	Parameters	TYPE 1 models $[K_A, K_B, K_C]$	TYPE 2 models $[K_A, K_B, K_C]$
components 1 and 2	n=50	0.40, 0.63, 0.83	0.25, 0.47, 0.59
	n=100	0.68, 0.80, 0.94	0.59, 0.75, 0.79
components 1 and 3	n=50	0.89, 0.61, 0.42	0.47, 0.61, 0.20
	n=100	0.96, 0.81, 0.71	0.67, 0.77, 0.56
components 2 and 3	n=50	0.66, 0.92, 0.78	0.40, 0.48, 0.52
	n=100	0.89, 0.97, 0.91	0.61, 0.70, 0.71

Table 2: The separation measurement K for the subset 1; points outside the Van der Waals sphere.

2.3.3 PCA results for TYPE 2 models: The multi-template models for α1 and α2 domains

For TYPE 2 models, amino-acids from positions 2-182 of the template and corresponding amino-acids from the target (α1 and α2 domains) were modelled. All three templates were used for each model (multi-template homology modelling).

As for TYPE 1 models, PCA was performed separately for the different types of points belonging to the subsets 1 and 2 of GRID 1. The PCA visualisation of all points in the subsets, points positioned inside the Van der Waals sphere and points positioned outside the Van der Waals sphere, but within 2Å distance from it, as well as the degree of variance indicated by the first three PCA components can be found in the Appendix B.

For points lying outside the van der Waals sphere, (Figure 5) the overlap is even more pronounced than in case of the TYPE 1 model. This is an unexpected result considering there are no clear differences in sums of the variances explained by firsts PCA components

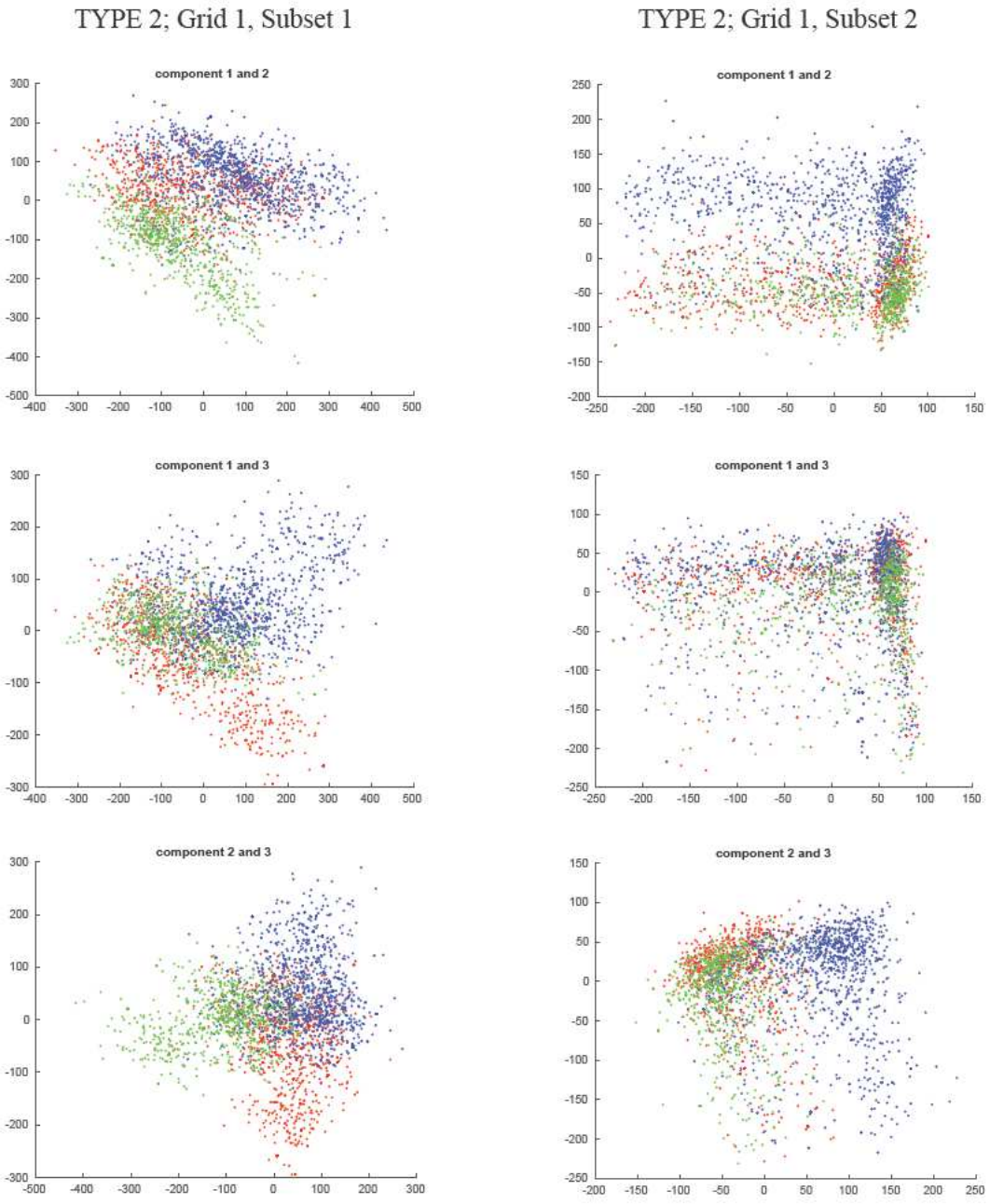

Figure 5: PCA visualization of TYPE 2 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere.

Model	Eigenvetors	Subset 1	Subset 2
TYPE 1 models	component 1	8.5	10.7
	component 2	6.5	7.2
	component 3	5.9	5.6
	sum	20.9	23.5
TYPE 2 models	component 1	12.1	10.0
	component 2	6.7	6.8
	component 3	5.0	4.7
	sum	23.8	21.5

Table 3: The percentage of the variance explained by the first three PCA components for points outside the Van der Waals sphere.

(Table 3). Table 2 shows comparison of the separation measurement K for points outside the Van der Waals sphere for subset 1 with analogous measurements for TYPE 1 models. All K values are lower for the PCA analysis results from TYPE 2 models. This compares favourably with visual comparison of Figures 4 and 5.

2.3.4 Single-template *versus* multi-template models

There is an undeniable advantage of multi template models in identical data preparation processes for all HLA class I proteins. However, we observe that the results obtained for single template models (where each subclass of proteins is modelled on a corresponding template) gives a much more pronounced separation between proteins belonging to different subclasses. The level of confidence that can be put in the accuracy of the models obtained by homology modelling depend on the similarity between the target protein and the template used.

Figure 6 shows results of the Blast sequence alignment (the number of identical amino acids) of proteins belonging to subclasses A, B, and C with reference to all three templates. For proteins belonging to subclass A, template A gives the best alignment. Templates B and C are equally inefficient. Every single HLA-A protein aligns better to the A template than to other ones. Similarly for proteins belonging to subclass C, template C gives the best alignment in general. Also template B gives better alignment than template A. It is worth noticing that for chains involving two HLA-C proteins only, template B gives better alignment than template C (167 identical amino acids versus 160 amino acids and 159 identical amino acids versus 152 amino acids).

For proteins belonging to subclass B, template B gives the best alignment in general. Also template C gives better alignment than template A. In case of 53 HLA-B proteins, template C gives equally good alignment when compared with template B. In case of just 6 HLA-B proteins, template C leads to better alignment than template B (161 identical amino acids versus 160 amino acids, 161 versus 160, 162 versus 160, 162 versus 160, 163 versus 162 and 167 versus 161).

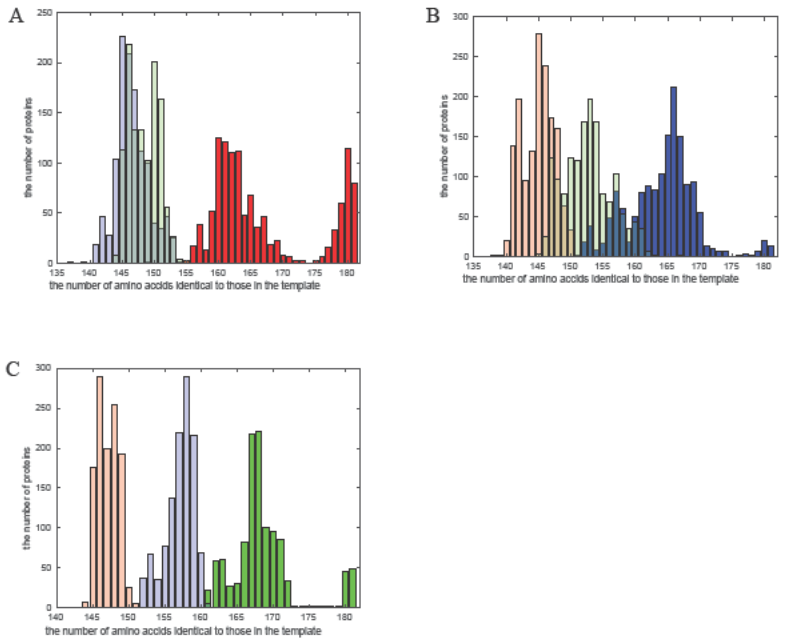

Figure 6: The Blast sequence alignment of protein belonging to the subclass A, B, and C in reference to all three templates. A: The alignment of the HLA-A proteins. B: The alignment of the HLA-B proteins. C: The alignment of the HLA-C proteins. Red bars: The alignment to the HLA-A template. Blue bars: The alignment to the HLA-B template. Green bars: The alignment to the HLA-C template.

Based on the above, we may conclude that using single-template models with different templates for each subclass of the given HLA I proteins, the homology modelling process is justified. Not only does this lead to better results with better subclass separation, but also it clearly demarcates similarities and differences between different amino acid sequences.

2.3.5 PCA results for TYPE 3 models: The single-template models of $\alpha 1$, $\alpha 2$ and $\alpha 3$ domains

For TYPE 3 models, all 275 amino-acids of the template and corresponding amino-acids from the target ($\alpha 1$, $\alpha 2$ and $\alpha 3$ domains) were modelled. Only one template was used for each model. The choice of the template was based on the target subclass affiliation.

As for TYPE 1 and TYPE 2 models, PCA was performed separately for the different types of points belonging to subsets 1 and 2 of GRID 1. The PCA visualisation of all points in the subsets, points positioned inside the Van der Waals sphere and points outside the Van der Waals sphere within 2 $\mathring{A}$ distance from it, as well as the percentage of variance explained by the first three PCA components can be found in the Appendix B.

For points localised outside the Van der Waals sphere clear separation can be observed for the subset 2, specifically for the first two PCA components (Figure 7 right panel). For subset 1, the clearest division occurs for components 2 and 3, but still with some overlapping

		TYPE 1 models $[K_A, K_B, K_C]$	TYPE 3 models $[K_A, K_B, K_C]$
components 1 and 2	n=50	0.40, 0.63, 0.83	0.42, 0.96, 0.48
	n=100	0.68, 0.80, 0.94	0.67, 0.99, 0.66
components 1 and 3	n=50	0.89, 0.61, 0.42	0.76, 0.56, 0.67
	n=100	0.96, 0.81, 0.71	0.88, 0.81, 0.86
components 2 and 3	n=50	0.66, 0.92, 0.78	0.87, 0.93, 0.91
	n=100	0.89, 0.97, 0.91	0.96, 0.98, 0.98

Table 4: The separation measurement K for the subset 1; points outside the Van der Waals sphere.

(Figure7 left panel). This overlapping looks less distinct than in case of the PCA performed for TYPE 1 models (Figure 5).

Figure 8 shows the three-dimensional plots visualizing the first three principal components for points localised outside the Van der Waals sphere for the TYPE 1 and the TYPE 3 models. The division between subclasses is more obvious here than on any of the two dimensional plots (Figures 4 and 7) and it appears more distinct for PCA results involving TYPE 3 models.

Table 4 compares the separation measurement K for points outside the Van der Waals sphere for subset 1 with analogous measurements for the TYPE 1 models. For the principal components 1 and 2 the subclass HLA-C separates better in the TYPE 1 models than in the TYPE 3 models. On the other hand the subclass HLA-B separates not only better, but almost perfectly in case of the TYPE 3 models ($K_A = 0.99$ for $n = 100$). For the principal components 1 and 3 the subclass HLA-A separates better in the TYPE 1 models than in the TYPE 3 models and the subclass HLA-C separates better in the TYPE 3 models than in the TYPE 1 models (not as good as in case of components 1 and 2 though). For the principal components 2 and 3 all subclasses separates better for the TYPE 3 models than in case of the TYPE 1 models. In fact also they separates better than in case of any pair of components for the TYPE 1 models. (With exception $K_A = 0.89$ in $K_{50}^{1,3}$ for the TYPE 1 models and $K_A = 0.87$ in $K_{50}^{2,3}$ for the TYPE 3 models).

2.3.6 PCA results for TYPE 3 models: The single-template models of α1, α2 and α3 domains; focus on the α3 domain

Principal Component Analysis was performed separately for the different types of points belonging to subset 3 of GRID 2. The PCA visualisation of all points in the subsets, points positioned inside the Van der Waals sphere and points outside the Van der Waals sphere within 2 $\mathring{A}$ distance from it, as well as the percentage of variance explained by the first three PCA components can be found in the Appendix B. Figure 9 shows that for points positioned inside the Van der Waals sphere, the principal components 1 and 2 are the complete division between HLA subclasses.

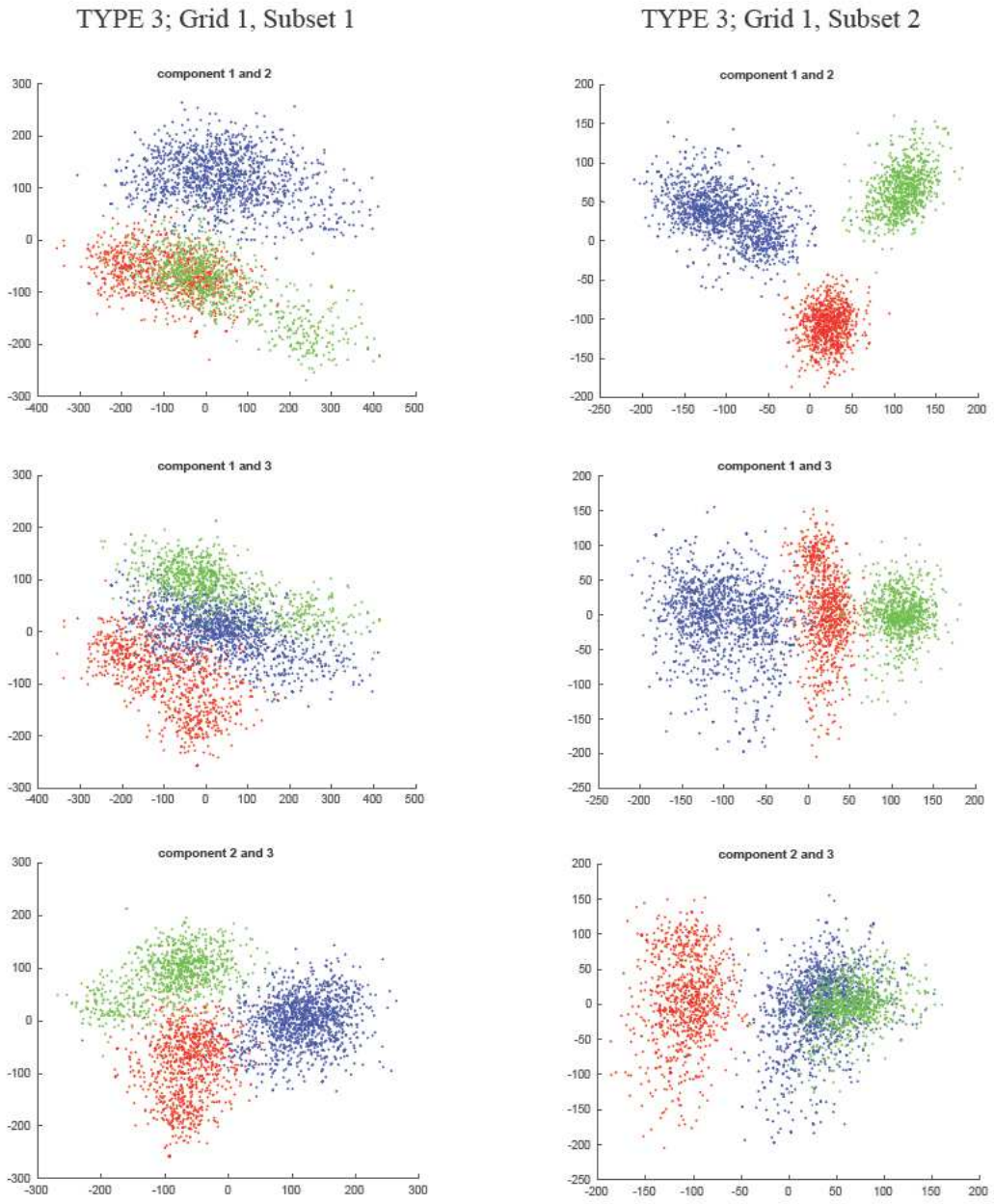

Figure 7: PCA visualization of the TYPE 3 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere.

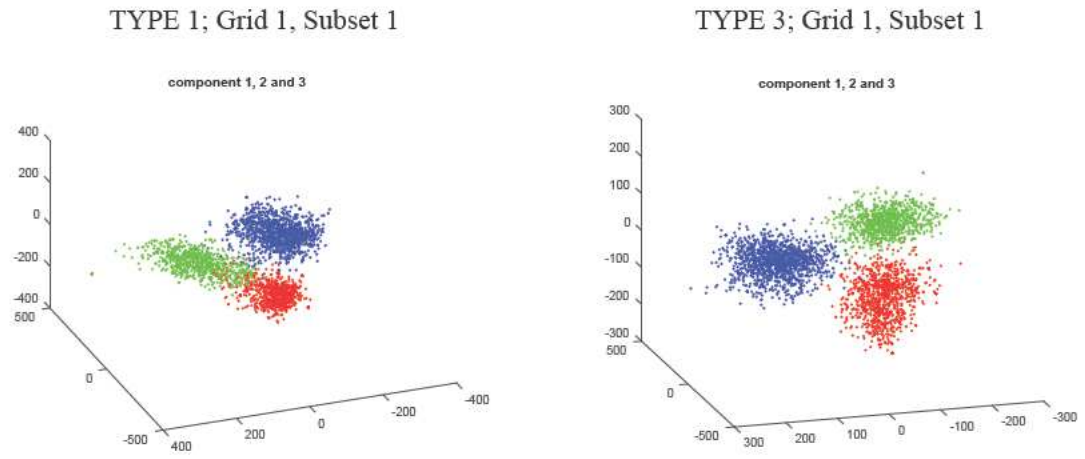

Figure 8: PCA visualization of the TYPE 1 (left panel) and the TYPE 3 models (right panel) of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere.

2.3.7 PCA results' summary

The models containing the inactive parts of the HLA class I proteins (TYPE 3 models) provide best separation between subclasses for points outside the Van der Waals sphere. This leaves the following unanswered questions:

- Does the homology modelling process and/or the electrostatic calculations work better for longer proteins?
- Are some of the observed separations caused by the differences in the structures of the inactive $\alpha 3$ domains of HLA class I proteins?

The results of the Principal Component Analysis prove that some proportion of the data variability is linear. The subclasses division, while admittedly not 100% accurate, is still clearly distinct; therefore, we can conclude that the chosen data preparation methods based on homology modelling and electrostatic potential calculations can guide us on the proteins' functionality.

For points localised outside the Van der Waals sphere, where the interactions with the obstructed peptide and with the T cell receptor are simultaneously occurring, the first three principal components can explain at most 23.8% of the variance (TYPE 2 models, the subset 1, points inside Van der Waals sphere). In that case, the first 165 principal components are required to explain over 90% of the variance.

We thus conclude that the linear PCA is insufficient for analysing dimensionality reduction for our data prototype. There is a clear need for implementing different, preferably nonlinear methods, that would not just be more potent, they will also be structurally less imperfect.

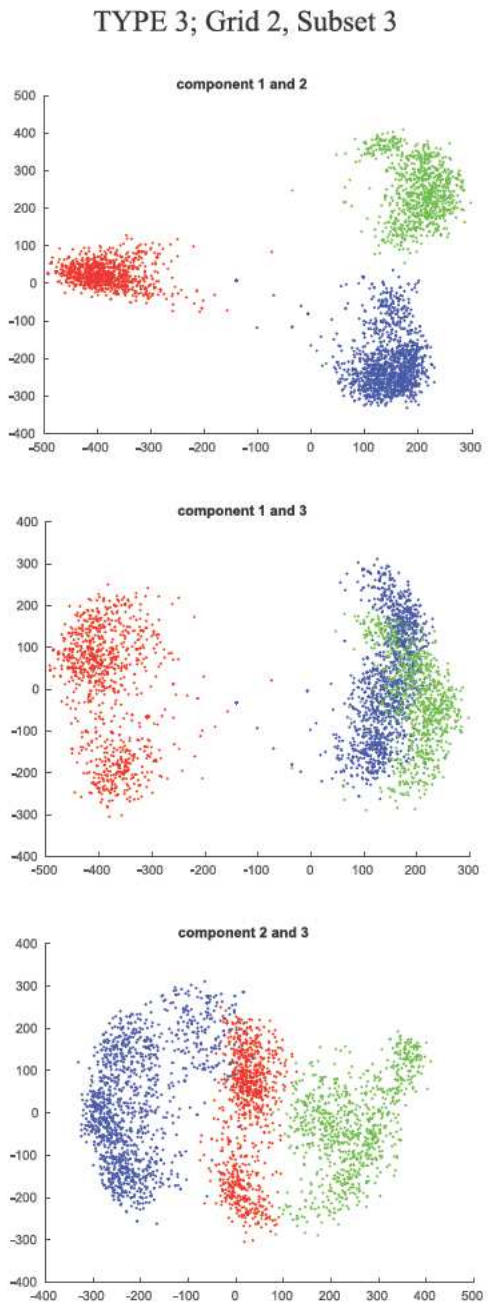

Figure 9: PCA visualization of the TYPE 3 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned inside the Van der Waals sphere.

2.4 Nonlinear Principal Component Analysis (NLPCA)

The linear dependence between all observed variables that is a strong assumption, that is mostly inaccurate. One possible approach to the problem of the nonlinear dimensionality reduction is proposed by the method called Nonlinear Principal Component Analysis (NLPCA) [47]. The name of this method suggests a close relationship with conventional linear PCA. This is due to the fact that by using a specific network architecture, results from PCA can be reproduced [47, 53]. However, the computational algorithm used is completely different.

The Nonlinear Principal Component Analysis method is based on training a specific type of neural network called the autoencoder or the bottleneck network [47, 53, 54, 55, 56, 57]. A simplified diagram is shown in the Figure 10. The neural network is trained to perform an identity mapping: the output layer is enforced to be identical to the input one. It is achieved by minimising the squared reconstruction error that measures the differences between the input and the output values of variables:

$$\|y_i - \hat{y}_i\|^2.$$

The nodes that constitute the central hidden layer can be interpreted as the principal components. If the network trains to acceptable difference between the input and output data, that means that the information encoded in the central bottleneck layer is enough to reconstruct the input data [47, 53]. As the central layer is composed of a smaller number of neurons, a smaller number of variables encode the information from the input.

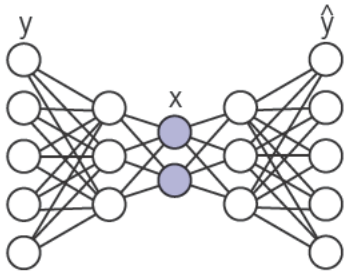

Figure 10: The schematic representation of the autoencoder neural network built with an input layer (which comprises input observed data Y), three hidden layers and an output layer (which comprised of reproduced observed data $\hat{Y}$). The number of neurons in the input layer and the output layer are equal to one another. The number of neurons in the central hidden layer X, that represents the latent variables is smaller. Other hidden layers may have larger or smaller number of neurons than the input one.

There is no definite method for deciding what should be the size of hidden layers (apart the middle one, which should be smaller than the input and output layers). For the number of network nodes below a critical threshold, accuracy is known to be limited. On the other hand, if the number is too large, there is a risk that the neural network will train the stochastic variation of the data instead of the hidden patterns [47].

2.4.1 NLPCA results

Nonlinear Principal Component Analysis was performed separately for the different types of points belonging to subsets 1 and 2 of GRID 1 for TYPE 1, TYPE 2, and TYPE 3 models. The visualisation of the results, as well as the percentage of explained variance can be found in the Appendix B.

Tables 5, 6 and 7 compare the separation measurements K for points outside the Van der Waals sphere for subset 1 against corresponding results for linear PCA. For TYPE 1 models, the maximum difference is 0.02 (e.g. for $K_{100}^{2,3}$ for the subclass HLA-C, Table 5). For TYPE 2 models (Table 6), the maximum difference is 0.07 (for $K_{100}^{1,2}$ for subclass HLA-A), second highest difference is 0.03 (e.g. for $K_{100}^{2,3}$ for subclass HLA-A). For TYPE 3 models the maximum difference is 0.03 (e.g. for $K_{100}^{1,3}$ for subclass HLA-A, Table 7).

TYPE 1 models		PCA $[K_A, K_B, K_C]$	NLPCA $[K_A, K_B, K_C]$
components 1 and 2	n=50	0.40, 0.63, 0.83	0.39, 0.61, 0.84
	n=100	0.68, 0.80, 0.94	0.68, 0.79, 0.93
components 1 and 3	n=50	0.89, 0.61, 0.42	0.89, 0.60, 0.42
	n=100	0.96, 0.81, 0.71	0.98, 0.81, 0.72
components 2 and 3	n=50	0.66, 0.92, 0.78	0.67, 0.91, 0.79
	n=100	0.89, 0.97, 0.91	0.90, 0.98, 0.93

Table 5: The separation measurement K for the subset 1; points outside the Van der Waals sphere.

TYPE 2 models		PCA $[K_A, K_B, K_C]$	NLPCA $[K_A, K_B, K_C]$
components 1 and 2	n=50	0.25, 0.47, 0.59	0.26, 0.48, 0.61
	n=100	**0.59**, 0.75, 0.79	**0.66**, 0.78, 0.82
components 1 and 3	n=50	0.47, 0.61, 0.20	0.49, 0.60, 0.21
	n=100	0.67, 0.77, 0.56	0.67, 0.75, 0.54
components 2 and 3	n=50	0.40, 0.48, 0.52	0.38, 0.48, 0.51
	n=100	0.61, 0.70, 0.71	0.64, 0.71, 0.73

Table 6: The separation measurement K for the the subset 1; points outside Van der Waals sphere.

The NLPCA does not provide any new visual information about our data when compared to the linear PCA analysis. The visualisation of the data is almost identical. For TYPE 1 and TYPE 2 models there are no substantial differences in the variances obtained by comparing the three first components between the linear and nonlinear PCA. A maximum of 0.8% variation is observed for the TYPE 1 models, typically for the second component for points inside the Van der Waals sphere in subset 2. There are some noticeable differences in the explained variances involving TYPE 3 models. For points inside the Van der Waals sphere,

TYPE 3 models		PCA $[K_A, K_B, K_C]$	NLPCA $[K_A, K_B, K_C]$
components 1 and 2	n=50	0.42, 0.96, 0.48	0.44, 0.95, 0.50
	n=100	0.67, 0.99, 0.66	0.67, 0.98, 0.65
components 1 and 3	n=50	0.76, 0.56, 0.67	0.77, 0.56, 0.67
	n=100	0.88, 0.81, 0.86	0.91, 0.83, 0.83
components 2 and 3	n=50	0.87, 0.93, 0.91	0.87, 0.92, 0.93
	n=100	0.96, 0.98, 0.98	0.97, 0.98, 0.98

Table 7: The separation measurement K for the subset 1; points outside the Van der Waals sphere.

for subset 1, the variance explained by the two first components is greater for NLPCA than for the L-PCA (29.5% versus 27.3% and 20.3% versus 16.9%). However, these differences do not affect the degree of separation between the HLA-A, HLA-B and HLA-C subclasses.

The results of the nonlinear Principal Component Analysis confirms the conclusion derived from the linear PCA analysis, that some proportion of the data variability is linear, as there is no substantial difference between results obtain from the linear and the nonlinear versions of PCA. The NLPCA does not provide any additional inside about the HLA class I proteins diversity.

2.5 Gaussian Process Latent Variable Model

Apart from its linearity assumption, PCA suffers from other limitations. It does not take into account that the observed variables, are not just linear combinations of their principal components, but also are subjected to the random distortion. To address this limitation, a new framework encapsulated under the name **Probabilistic Principal Component Analysis** (PPCA) was proposed [58]:

$$y_i = W x_i + \eta_i, \tag{1}$$

where η_i represents stochastic noise, typically drawn from a spherical Gaussian distribution.

The **Kernel Principal Component Analysis** (KPCA) is a variation of the PCA where the search for the principal components is performed in the *feature space* instead of the observation space [59]. In some cases, there is a possibility to hypothesise about nature of those unknown features, however in our research we use the kernel method as the mathematical tool that allows the search for the nonlinear interactions between variables. The *features* are represented by values of a function ψ, and the observations are the function arguments:

$$f_i = \psi(y_i), \tag{2}$$

y_i being represented by a d-dimensional vector of the i^{th} observation and f_i is an r-dimensional vector representing r features for the i^{th} observation ($r \geq d$). If ψ is a nonlinear function,

KPCA operforms nonlinear dimensionality reduction.

The classical PCA can be carried out through the eigenvalue decomposition of the covariance matrix C (data is scaled so that it has a zero mean) [59], that is represented as an inner product involving the variable vectors:

$$C = y_i \cdot y_j. \tag{3}$$

The kernel PCA require the eigenvalue decomposition of the covariance matrix $\widehat{C}$:

$$\widehat{C} = \psi(y_i) \cdot \psi(y_j) = k(y_i, y_j). \tag{4}$$

The function k is called the *kernel function* [45, 59].

In order to perform the kernel PCA there is no need to extract the mapping function ψ. The knowledge of values of kernel function is sufficient enough [45, 48, 59]. The advantage of this approach is the possibility to significantly reduce the computational power required.

Similarly to the traditional linear PCA, the nonlinear kernel PCA is easy to implement and the calculations performed gives unambiguous results. There is, however, a substantial difference between these methods. Conventional PCA does not require any assumption *a priori* while for KPCA, the selection of the kernel function has to be done prior to analysis.

The **Gaussian Process Latent Variable Model** (GPLVM) is a nonlinear, probabilistic dimensionality reduction method [46, 60], that can be understood as the Probabilistic Kernel Principal Component Analysis (see Figure 11). As in the case of the kernel PCA, the matrix of covariance is replaced by the matrix of the kernel function. The observed variables are not just a combination of latent variables, but the error in the form of the normally distributed noise is added to the model. That approach acknowledges the probabilistic nature of the real-life data.

$$y_i = \gamma(x_i) + \eta_i. \tag{5}$$

η_i is the noise term, taken as an independent sample from a spherical Gaussian distribution [46, 48, 61].

A key target of dimensionality reduction methods is to determine how the latent space representation of the data preserves the distances between data points. Unlike most nonlinear methods (such as NLPCA) or the classical linear PCA, the GPLVM focuses on the preservation of the long distances (dissimilarities), rather than short ones. This means that points that are close together in the data space may be not close in the latent space, and this potentially introduces an error by overlooking some similarities between data points. It is considered an imperfection of the GPLVM method. However, there is a modification that allows for a compromise. This modification was proposed by the same author who invented the GPLVM methods [62]. The pure GPLVM algorithm performs mapping from the latent space

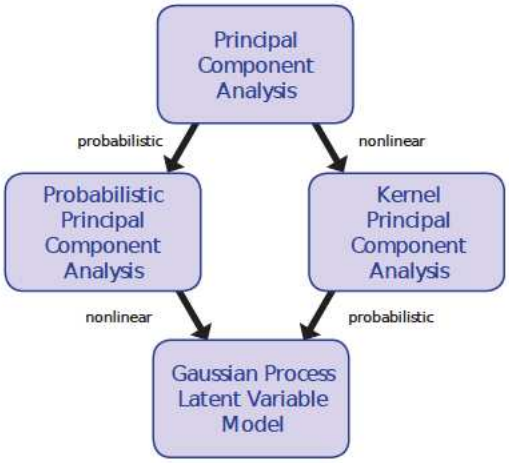

Figure 11: The schematic representation of the relations between the different dimensionality reduction methods.

to the data space. By adding synchronously working parametric mapping (back constraints) from data space to latent space, the short distances (similarities) from data space are also preserved in the latent space. We used this GPLVM algorithm with MLP back constrains [62] in our analysis.

2.5.1 The GPLVM results: Analysis of the latent variables

The Gaussian Process Latent Variable Model was performed separately for the different types of points belonging to the subset 1 and 2 of GRID 1 and subset 3 of GRID 2. Figure 12 shows results for points positioned outside the Van der Waals sphere and Figure 13 results for points positioned outside the Van der Waals sphere, but within a 2 $\mathring{A}$ radius from it. The results for all points in the subsets and points positioned inside the Van der Waals sphere, as well as all results for TYPE 3 models (focusing on the inactive part of HLA proteins) can be found in the Appendix B.

Table 8 shows values of the separation measurement K calculated for results obtained for subsets 1 and 2 of TYPE 1, TYPE 2 and TYPE 3 models.

As with both, the linear and non-linear PCA, the least clear separation between subclasses can be seen for TYPE 2 models (multi template homology modelling). However, GPLVM provides us with information that PCA does not. For some analyses, we see a clear division into clusters within each subclass (e.g. Figure 13 top left and bottom left plot and middle plots). However, two issues are important here. Firstly, the visual representation of the GPLVM results may be misleading. Sometimes what looks like one point in our results' space is actually a group of points lying very close or even on top of each other (e.g. Figure 14 left panel, yellow circle marks what looks like one point, but is in fact eighty points). It is also important that where there is complete or almost complete separation between

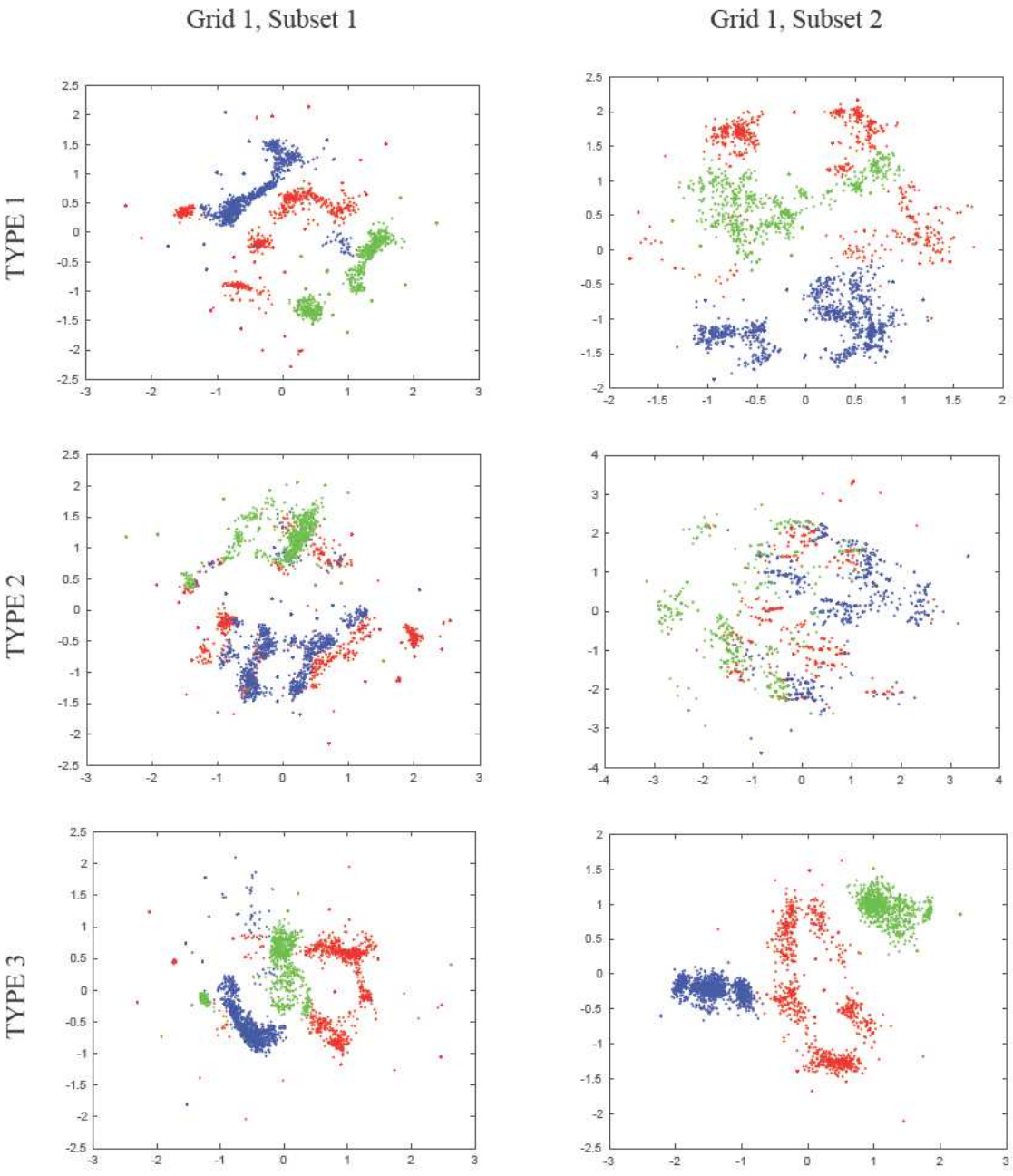

Figure 12: GPLVM visualization of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere.

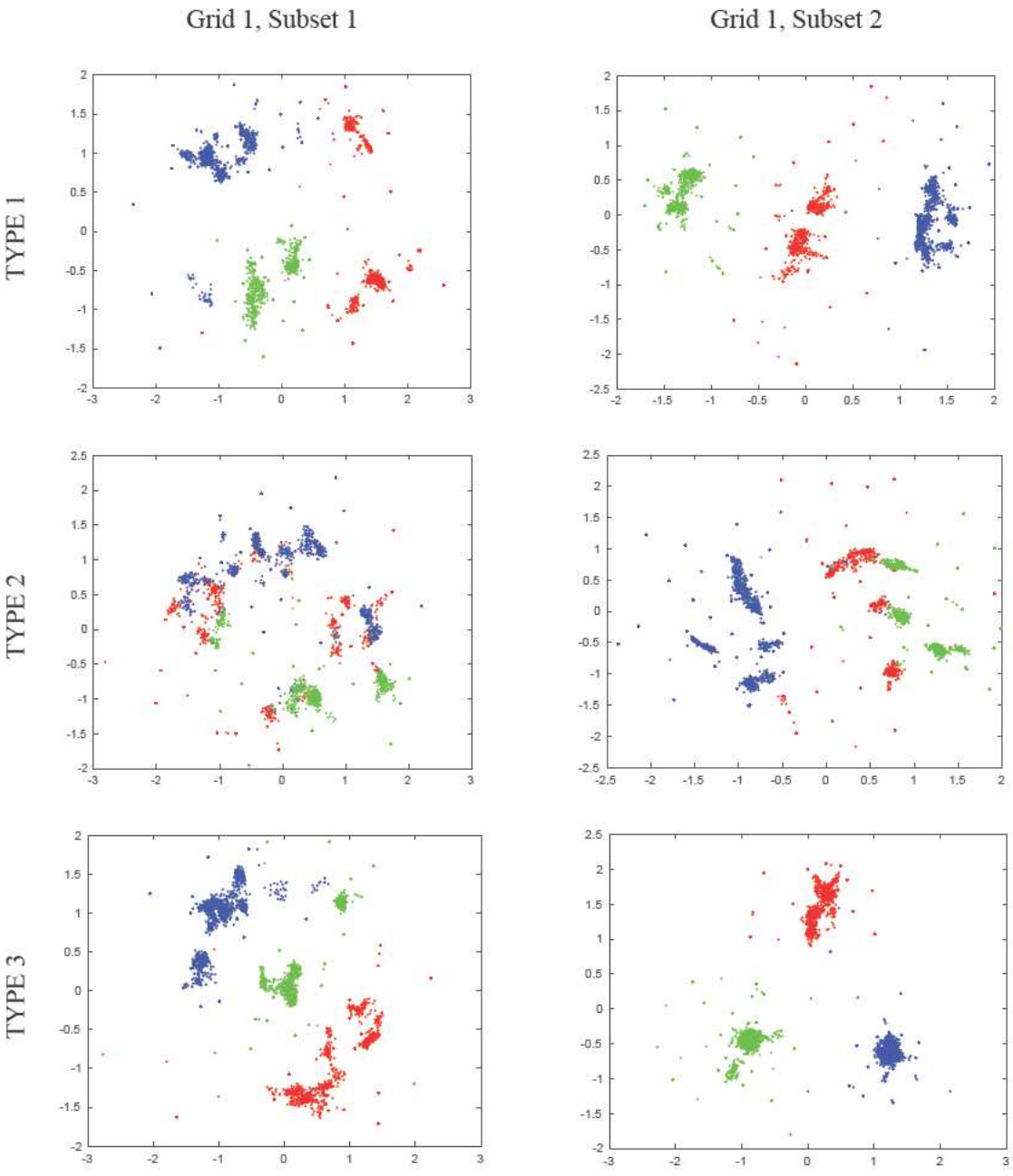

Figure 13: GPLVM visualization of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere, but within the 2 $\mathring{A}$ distance from it.

		TYPE 1 models $[K_A, K_B, K_C]$	TYPE 2 models $[K_A, K_B, K_C]$	TYPE 3 models $[K_A, K_B, K_C]$
Points outside the Van der Waals sphere				
Subset 1	n=50	0.95, 0.91, 0.94	0.61, 0.58, 0.69	0.95, 0.84, 0.85
	n=100	0.97, 0.97, 0.97	0.77, 0.79, 0.83	0.99, 0.96, 0.96
Subset 2	n=50	0.96, 0.98, 0.97	0.56, 0.74, 0.62	0.98, 0.96, 0.99
	n=100	0.99, 0.99, 0.99	0.74, 0.86, 0.78	≈1, 0.99, ≈1
Points outside the Van der Waals sphere within 2 $\mathring{A}$ distance from it				
Subset 1	n=50	0.98, 0.96, 0.93	0.54, 0.59, 0.58	0.98, 0.96, 0.91
	n=100	1, 0.98, 0.98	0.69, 0.74, 0.70	0.99, 0.98, 0.96
Subset 2	n=50	0.98, 0.99, 0.99	0.68, 0.79, 0.68	0.99, 0.98, 1
	n=100	0.99, ≈1, 0.99	0.77, 0.86, 0.75	1, 1, 1

Table 8: The separation measurement K for the GPLVM analysis.

subclasses, clusters are difficult to distinguish (e.g. Figure 13 right top and bottom plots). Therefore, the cluster analysis was performed for each subclass separately.

2.6 HLA-A subclass cluster analysis

In this section we will discuss HLA-A subclass cluster division based on the GPLVM analysis for points positioned outside the Van der Waals sphere, but within a $2\mathring{A}$ distance from it based on three different GPLVM analysis results. (Figure 14, this choice was based on the fact that for those results, the cluster division is distinctly visible). Isolated single points and points in groups of 10 or less are considered not to have a cluster affiliation. Table 9 shows the number of proteins belonging to each cluster and symbols used for their graphical representation. As was already stated, the visual representation of the GPLVM results may be misleading, because in some cases what looks like one point in our results' space, is actually a group of points lying on top of each other. Therefore table 10 shows the number of proteins assigned to clusters from one division belong to a cluster from another division.

Red colour indicates cases for which all proteins belonging to a cluster from a given division belong to one cluster from another division. For example, all twenty two proteins from cluster 10 from the cluster division based on TYPE 2 models belong to cluster 13 from the cluster division based on TYPE 1 models.

Blue colour indicates cases in which almost all proteins (except one or two) belonging to a cluster of a given division belong to one cluster of another division. Proteins of unidentified clusters affiliation are not taken into account. For example, twenty four out of twenty five proteins from cluster 16 from the cluster division based on TYPE 2 models belong to cluster 13 from the cluster division based on TYPE 1 models.

Based on Table 10, we may conclude that proteins belonging to clusters created on the basis of GPLVM analysis do not spread randomly between clusters derived from another GPLVM analysis. However, cluster division from two different GPLVM analyses are not

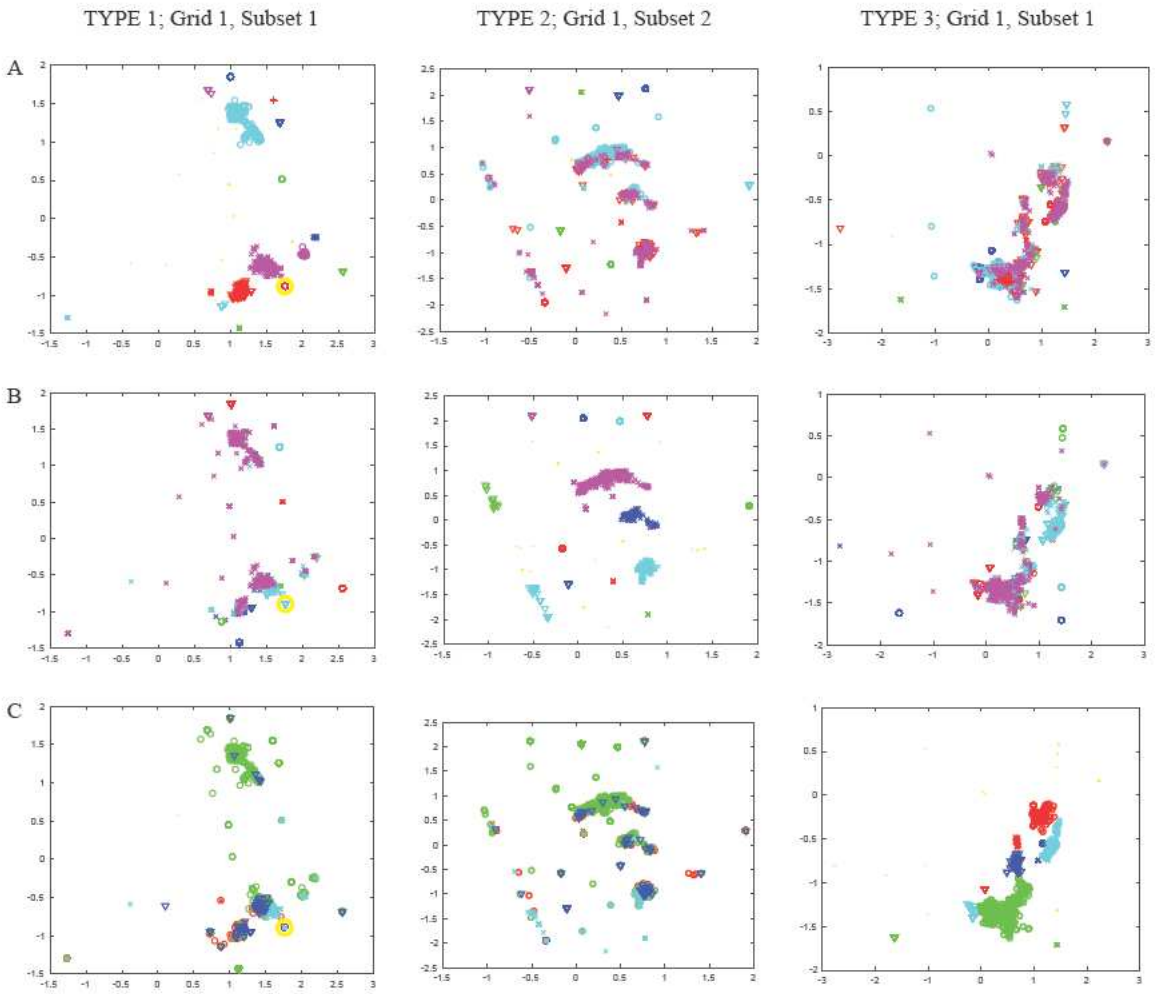

Figure 14: Visual representation of HLA-A subclass cluster division based on the GPLVM analysis for points positioned outside the Van der Waals sphere, but within the 2 $\AA$ distance from it. A: Cluster division based on the TYPE 2 models, subset 2. B: Cluster division based on the TYPE 2 models, subset 2. C: Cluster division based on the TYPE 3 models, subset 1.

identical. Therefore the GPLVM dimensionality reduction method of *in silico* electrostatic potential values created by HLA class I protein may add information necessary to identifying HLA super-types, but by itself it is not robust enough to be independently conclusive.

We believe that the success of the used methodology can be linked to both: the use of the Poisson- Boltzmann electrostatic calculations, as well as to the sophisticated multi-dimensional data analysis approach. The PCA has been proven to be effective tool for a dimensionality reduction task when dealing with data characterized by a certain degree of linearity. GPLVM, developed for the nonlinear data analysis is also, through its probabilistic nature, well suited to deal with the measurement errors related to the real data collection. The Poisson-Boltzmann electrostatic potential calculations are a good predictor of the molecular interactions, as electrostatic potential depends not only on the charges of individual atoms,

Cluster division based on the TYPE 1 models, subset 1															
number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
population	31	16	110	80	11	32	79	54	28	12	22	32	229	367	13
symbol	.	×	▽	○	×	▽	○	×	▽	○	×	▽	○	×	▽
number	15	16													
population	22	25													
symbol	○	+													
Cluster division based on the TYPE 2 models, subset 2															
number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
population	37	12	79	28	105	14	53	11	11	31	244	112	32	382	12
symbol	.	×	▽	○	×	▽	○	×	▽	○	×	▽	○	×	▽
Cluster division based on the TYPE 3 models, subset 1															
number	0	1	2	3	4	5	6	7	8	9	10	11			
population	39	23	28	134	21	77	24	16	16	518	229	38			
symbol	.	×	▽	○	×	▽	○	×	▽	○	×	▽			

Table 9: Clusters population for points positioned outside the Van der Waals sphere, but within the 2 $\AA$ distance from it and symbols used for its graphical representation. 0 denotes points of the unidentified clusters affiliation.

but also on the shape of the examined molecule.

2.7 Conclusions

The challenge we have addressed in this part of research, was to probabilistically analyse HLA protein data set in order to determine their ability to bind to specific antigens.

1. Principal Component Analysis results strongly suggest that large proportion of the analysed data variability is linear. The subclasses division, even if not 100% accurate, indicate that the chosen data preparation methods based on homology modelling and electrostatic potential calculations can provide us with vital information about protein functionality.

2. Although the nonlinear Principal Component Analysis results do not provide any additional insight into the HLA class I proteins diversity when compared to linear PCA results, they tend to affirm the linear structure of the analysed data.

3. The GPLVM dimensionality reduction method performed for electrostatic potential values data set provide distinct HLA-A, HLA-B and HLA-C subclass divisions. As to identifying super-types inside separate subclasses, the results are more ambiguous. Although this method may provide partial information necessary to identify HLA super-types, it is not robust enough to be independently conclusive.

Cluster division based on the TYPE 1 models, subset 1	Cluster division based on the TYPE 2 models, subset 2														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0					1						1			29	
1	10										6				
2	4				26	14					35	4		27	
3												80			
4									1		2			8	
5													32		
6			79												
7					1		53								
8				28											
9		12													
10														22	
11					1					31					
12	9				27				5		18	1		169	
13	14				45			11	4		172	27		94	
14														1	12
15					4				1		9			8	
16											1			24	

Cluster division based on the TYPE 3 models, subset 1	Cluster division based on the TYPE 2 models, subset 2														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0				1	1	1				9	6	3	10	8	
1					1				1					21	
2			28												
3	5			10	22		6		1	20	22	3		45	
4												21			
5	7		1	2	11	13			1	1	24			17	
6												24			
7							16								
8							16								
9	19	1	21	15	67		15		7	1	56	1	22	284	9
10	6	11			2			11	1		135	60		3	
11			29		1						1			4	3

Cluster division based on the TYPE 3 models, subset 1	Cluster division based on the TYPE 1 models, subset 1																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1		4	2		10			1			9	4	7		1	
1	1		5								4		1	12			
2							28										
3	2	5	34	1				6	10		16	21	1	37		1	
4				21													
5	1	4	32				1		2			1	5	31			
6				24													
7								16									
8								16									
9	25	7	27		9	22	21	16	15	1	1	1	207	117	10	14	25
10	1		8	32	2					11	1		6	162		6	
11							29						5	1	3		

Table 10: The number of proteins assigned to clusters from one division belong to cluster from another division.

3 Standard non-alignment protein descriptors

3.1 Descriptors based on composition

The **Amino Acid Composition** $AAC(i)$ is the proportion of amino acids of each type in the protein sequence of a N length. The **Dipeptide Composition** $DC(j)$ is the proportion of each pair of amino acids and the **Tripeptide Composition** $TC(k)$ is the proportion of each triplet.

$$p(i) = AAC(i) = \frac{n_i}{N}$$
$$p(j) = DC(j) = \frac{n_j}{N-1}$$
$$p(k) = TC(k) = \frac{n_k}{N-2}$$

where n_i represents the number of the amino acid of type i in the sequence. $i \in \{1, 2, ..., 20\}$;
n_j is the number of the amino acid pairs of type j in the sequence. $j \in \{1, 2, ..., 400\}$;
n_k is the number of the amino acid triplets of type k in the sequence. $k \in \{1, 2, ..., 8000\}$.

Descriptor values calculated using the above definitions were obtained with the protr package [63], software generating numerical descriptors for amino acid sequences, successfully used among others, in the research of the potential drug-target interactions [25] and population genomics [64].

3.2 Descriptors based on Shannon Entropy

The concept of Information Entropy was first introduced in 1948 as a measurement of the "information content" of a signal [65]. It is defined as the function of the probability distribution of random variable X, where p(x) is its probably mass function [18, 66].

$$H[X] = -\sum_x p(x) \log p(x)$$

We propose set of 6 descriptors based on entropy and conditional entropy definitions. Firstly, to calculate the entropy of a protein sequence, we treat the descriptors based on composition, as random variables:

$$H[AAC] = -\sum_i^{n_i} p(i) \log \big(p(i)\big)$$
$$H[DC] = -\sum_i^{n_j} p(j) \log \big(p(j)\big)$$
$$H[TC] = -\sum_i^{n_k} p(k) \log \big(p(k)\big)$$

where $n_i = 20$, $n_j = 400$, $n_k = 8000$. For our purpose, $p(i)$ is the proportion of i^{th} amino acid, $p(j)$ is the proportion of j^{th} dipeptide and $p(i)$ is the proportion of i^{th} tripeptide in the examined protein sequence.

Conditional entropy [66] is calculated as:

$$H[DC|AAC] = -\sum_{i}^{n_i} p(i) \sum_{j}^{n_j} p(j|i) \log\big(p(j|i)\big)$$

$$H[[TC|AAC] = -\sum_{i}^{n_i} p(i) \sum_{j}^{n_k} p(k|i) \log\big(p(k|i)\big)$$

$$H[TC|DC] = -\sum_{i}^{n_j} p(j) \sum_{j}^{n_k} p(k|j) \log\big(p(k|j)\big)$$

where $p(j|i)$ is the proportion of j^{th} dipeptide given the i^{th} amino acid,
$p(j|i)$ is the proportion of k^{th} tripeptide given the i^{th} amino acid,
$p(j|i)$ is the proportion of k^{th} tripeptide given the j^{th} dipeptide.

In order to add possible protein descriptors we proposed calculating the **partial conditional entropy** which we define as:

$$H[DC_{AAC}] = -p(i) \sum_{j}^{n_j} p(j|i) \log\big(p(j|i)\big)$$

$$H[[TC_{AAC}] = -p(i) \sum_{j}^{n_k} p(k|i) \log\big(p(k|i)\big)$$

$$H[TC_{DC}] = -p(j) \sum_{j}^{n_k} p(k|j) \log\big(p(k|j)\big)$$

It is calculated analogously to contingent entropy, but omits the first summation, and results in vectors instead of single values.

3.3 Proximity between amino acids

To assess the proximity between amino acids, we adopted the simplified approach based on the one proposed in [67]. The proximity is understood as the distance between the amino acid of type i and the amino acid of type j that is nearest to it, assuming that between the considered pair of neighbours (i and j), no other amino acid type j exists (see Figure 15). For each type of ij pair, the average and the standard deviation of proximity was calculated. The calculations were performed twice. In the first case, the neighbour j of the amino acid i was searched in both directions; in the second case, only the direction in which the protein is synthesized was studied.

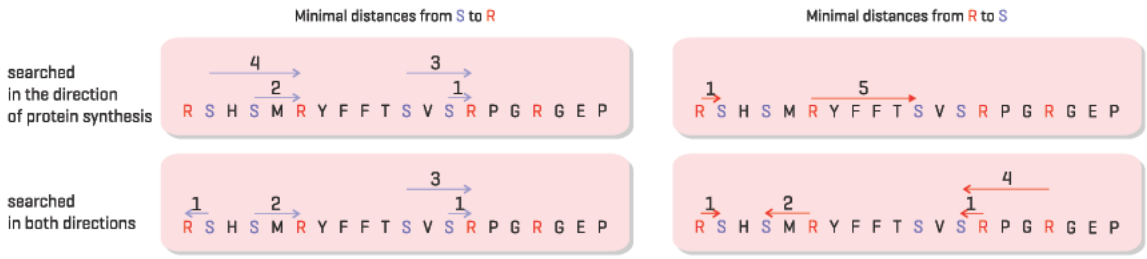

Figure 15: The Graphical representation of the search of the minimal proximity between amino acids. Left panel shows the search result for each Arginine (R) located in the closest proximity of each Serine (S), the right panel shows search results for each Serine (S) located in the closest proximity of each Arginine (R). Top panels shows search in the direction of the process of protein synthesis, bottom panels shows search in both directions.

3.4 Composition/Transition/Distribution

For the composition, transition and distribution [68, 69] calculations, amino acids were divided into three classes, depending on their properties (7 different attributes were taken into account). The amino acid allocation can be found in the Table 11 below. Descriptor values calculated with definitions presented in this section were obtained with the protr package [63].

The **Composition** $C(r)$ is the proportion of amino acids of each classes n_r in the protein sequence of a N length.

$$C(r) = \frac{N_r}{N}$$

$r \in \{1, 2, 3\}$. For each attributes listed in table 1 the three Composition descriptorswerecalculated giving $3 \times 7 = 21$ dimensional vector.

The **Transition** $T(r_k/r_l)$ is the proportion of amino acids from the group r_k directly preceded by the amino acid from the group r_l (denoted n_{r_k/r_l}) or from the group r_l directly preceded by the amino acid from the group r_k (denoted n_{r_l/r_k}). If composition can be seen as the Amino Acid Composition equivalents for the seven attributes defined in Table 11, then the Transition would be Dipeptide Composition equivalent.

$$T(r_k/r_l) = \frac{n_{r_k/r_l} + n_{r_l/r_k}}{N - 1}$$

There are five **Distribution** descriptors for each attribute and class within that attribute (5 descriptors $\times$ 7 attributes $\times$ 3 classes = 105 descriptor values for each amino acid in total). They are the position of chosen amino acids from each class. Those chosen amino acids are the first one from each class, the $\frac{1}{4}^{th}$ one, the middle one, the $\frac{3}{4}^{th}$ one and the last one. Figure 16 illustrate the process of constructing distribution descriptors.

	Group 1	Group 2	Group 3
Hydrophobicity	Polar	Neutral	Hydrophobicity
	R,K,E,D,Q,N	G,A,S,T,P,H,Y	C,L,V,I,M,F,W
Normalized	0-2.78	2.95-4.0	4.03-8.08
van der Waals Volume	G,A,S,T,P,D,C	N,V,E,Q,I, L	M,H,K,F,R,Y,W
Polarity	4.9-6.2	8.0-9.2	10.4-13.0
	L,I,F,W,C,M,V,Y	P,A,T,G,S	H,Q,R,K,N,E,D
Polarizability	0-1.08	0.128-0.186	0.219-0.409
	G,A,S,D,T	C,P,N,V,E,Q,I,L	K,M,H,F,R,Y,W
Charge	Positive	Neutral	Negative
	K,R	A,N,C,Q,G,H,I,L, M,F,P,S,T,W,Y,V	D,E
Secondary Structure	Helix	Strand	Coil
	E,A,L,M,Q,K,R,H	V,I,Y,C,W,F,T	G,N,P,S,D
Solvent Accessibility	Buried	Exposed	Intermediate
	A,L,F,C,G,I,V,W	R,K,Q,E,N,D	M,S,P,T,H,Y

Table 11: The amino acid allocation to the classes based on seven attributes. Source [63].

3.5 Conjoint Triad calculation

The 20 amino acids are clustered into seven classes (see Table 12) [70]. Like in case of Tripeptide Composition the conjoint triad descriptors regarded any three following amino acids as a unit. But the different amino acids from any class were treated as identical. As a result, instead of 8000 descriptors we had we had to deal with only $7 \times 7 \times 7 = 343$. Descriptor values calculated with definitions presented in this section were obtained with the protr package [63].

$$f(i) = \frac{n_{fi}}{N-2}$$

n_{fi} is the number of the amino acid triad of type fi in the sequence. N is the protein sequence length. $i \in \{1, 2, ..., 343\}$. Then $f(i)$ is normalised to become independent of N.

$$CT(i) = \frac{f(i) - min\{f(1), f(2), ...f(343)\}}{max\{f(1), f(2), ...f(343)\}}$$

3.6 Quasi Sequence Order descriptors

The **Sequence Order Coupling Numbers** [71] is defined as:

$$SOCN(d) = \sum_{i=1}^{N-d} (dist_{i,i+d})^2,$$

	1	2	3	4		5	6	7	8	9	10
Amino acid	S	H	S	M	R	Y	F	F	T	S	V
Position in sequence	1	2	3	4	5	6	7	8	9	10	11
	11		12	13		14		15		16	17
Amino acid	S	R	P	G	R	G	E	P	R	F	I
Position in sequence	12	13	14	15	16	17	18	19	20	21	22
	18	19	20	21	22			23	24		
Amino acid	A	V	G	Y	V	D	D	T	Q	R	
Position in sequence	23	24	25	26	27	28	29	30	31	32	

Figure 16: Construction of distribution descriptors. Amino acids belonging to group 2 (Neutral) based on attribute Charge (see Table 11) are marked red. There are 24 of them. First is on position 1 in amino acid sequence, sixth ($\frac{1}{4}^{th}$) one in position 7, twelfth (middle) one on position 14, eighteenth ($\frac{3}{4}^{th}$) one on position 23 and the last one on position 31. Therefore the subset of distribution descriptors for group Neutral in the attribute Charge will be $\{1, 7, 14, 23, 31\}$.

$dist_{i,i+d}$ is the distance between the two amino acids taken from chosen distance matrix. i and $i + d$ are amino acid positions in the protein sequence separated by $d \in \{1, 2, ..., 30\}$. N is the protein sequence length.

First twenty **Quasi Sequence Order descriptors** [71] are defined as follows:

$$QSOD(i = 1, 2, ..., 20) = \frac{f_i}{\sum_{i=1}^{20} f_i + w \sum_{d=1}^{30} SOCN(d)},$$

The next thirty abide the following updating rule:

$$QSOD(i = 21, 22, ..., 50) = \frac{wSOCN(i - 20)}{\sum_{i=1}^{20} f_i + w \sum_{d=1}^{30} SOCN(d)}$$

where f_i is the frequency of the amino acid of type i, the weight factor being $w = 0.1$.

Descriptor values calculated with above definitions were obtained with the protr package [63]. Two different distance matrices were used: Schneider-Wrede physicochemical distance matrix was used as in original work [71, 72] and chemical distance matrix proposed by Grantham [73].

3.7 Descriptors based on Pseudo Amino Acid Composition

The **Pseudo Amino Acid Composition** (PAAC, also known as type 1 pseudo-amino acid composition) is a descriptor vector that is calculated based on the values of hydrophobicity

	Dipole Scale	Volume Scale	Amino acids
1	< 1.0	< 50Å	A,G,V
2	< 1.0	> 50Å	I,L,F,P
3	> 1.0 and < 2.0	> 50Å	Y,M,T,S
4	> 2.0 and < 3.0	> 50Å	H,N,Q,W
5	> 3.0	> 50Å	R,K
6	> 3.0[1]	> 50Å	D,E
7	> 1.0 and < 2.0 [2]	> 50Å	C

[1]with opposite orientation

[2] Cysteine (C) is separated from class 3 because of its ability to form disulfide bonds

Table 12: Classification of amino acids based on dipoles and volumes of the side chains. Source [63].

(property considered to have a crucial role in the process of proteins folding), hydrophilicity and side chain masses of the 20 amino acids. Firstly these values were normalised and then correlation function $\Theta(i, i+d)$ between the amino acid values at the positions i an $i+d$ were calculated. d is the distance between these two amino acids in the protein chain. The detailed protocol can be found in [74]. As a next step, the the order-correlated factors of the sequence were calculated:

$$\delta_1(d) = \frac{\sum_{i=1}^{N-d} \Theta(i, i+d)}{N-d}$$

$d \in \{1, 2, ..., 30\}$. N is the protein sequence length.

Based on this, the first 20 descriptors are calculated sequentially:

$$PAAC(j = 1, 2, ..., 20) = \frac{f_j}{\sum_{j=1}^{20} f_j + \omega \sum_{d=1}^{30} \delta_1(d)}$$

followed by the next 30 descriptors:

$$PAAC(j = 21, 22, ..., 50) = \frac{\omega \delta_1(j-20)}{\sum_{j=1}^{20} f_j + \omega \sum_{d=1}^{30} \delta_1(d)}$$

where f_j is the frequency of amino acid of type j and ω is the weight factor that we set at $w = 0.05$ [74].

The **Amphiphilic Pseudo Amino Acid Composition** [74] (APAAC, also known as type 2 pseudo-amino acid composition) is descriptor vector that was calculated based on the normalised values of hydrophobicity $H_1(j)$ and hydrophilicity $H_2(j)$ of the amino acids ($j \in \{1, 2, 3, ..., 20\}$ are 20 amino acid types). Firstly the correlation functions $H_1(i, i+d) = H_1(i)H_1(i+d)$ for hydrophobicity and $H_2(i, i+d) = H_2(i)H_2(i+d)$ hydrophilicity were calculated. i an $i+d$ are amino acids on position i and $i+d$ in the protein chain, while d is the distance between them. Next step provides values for the order-correlated factors of the

sequence:

$$\delta_2(2d-1) = \frac{\sum_{i=1}^{N-d} H_1(i, i+d)}{(N-d)}$$
$$\delta_2(2d) = \frac{\sum_{i=1}^{N-d} H_2(i, i+d)}{(N-d)}$$

$d \in \{1, 2, ..., 30\}$. N is the protein sequence length.

Based on that the first twenty descriptors are calculated respectively:

$$APAAC(j = 1, 2, ..., 20) = \frac{f_j}{\sum_{j=1}^{20} f_j + \omega \sum_{d=1}^{30} \delta_2(d)}$$

The next sixty descriptors:

$$APAAC(j = 21, 22, ..., 80) = \frac{\omega \delta_2(j-20)}{\sum_{j=1}^{20} f_j + \omega \sum_{d=1}^{30} \delta_2(d)}$$

where f_j is the frequency of amino acid of type j; ω is the weighting factor set to $w = 0.05$ [74].

Descriptor values calculated for the type 1 and type 2 Pseudo Amino Acid Composition were obtained with the protr package [63].

3.8 AAIndex scales

As a first step, the amino acid sequence was transformed into a numerical sequence by replacing each amino acid with a numerical value (see Figure 17). Those values were taken from AAIndex database [75], that contains numerical representations of various physicochemical and biochemical properties of amino acids, derived from published literature. The 553 different scales (each providing numerical values for all amino-acids, each representing a different property such as hydrophobicity or charge etc.) were used. Therefore each protein is now represented by 553 different numerical sequences.

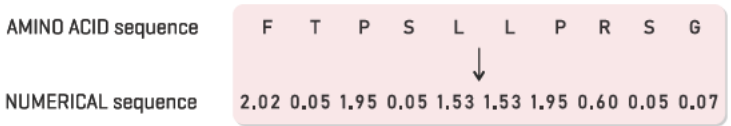

Figure 17: The transforming of amino acid sequence into a numerical sequence. Hydrophobicity scale is used in this example. Whenever amino acid Arginine (R) appears in the protein chain we substitute it with value of 0.6; amino acid Serine (S) with value 0.05; etc.

3.9 AAIndex derived descriptors based on Pseudo Amino Acid Composition

For the **AAIndex derived Pseudo Amino Acid Composition** [76] (type I) the order-correlated factors $\delta_1(d)$ of the sequence were calculated:

$$\delta_1(d) = \frac{\sum_{i=1}^{N-d}(P_i - P_{i+d})^2}{N-d}$$

For the **AAIndex derived Amphiphilic Pseudo Amino Acid Composition** (type II) the order-correlated factors $\delta_2(d)$ of the sequence were calculated:

$$\delta_2(d) = \frac{\sum_{i=1}^{N-d} P_i P_{i+d}}{(N-d)}$$

$d \in \{1, 2, ..., 30\}$, P_i and P_{i+d} are the values of i and $i+d$ entry in the examined numerical sequence. N defines the length of the sequence.

Based on the above formulation, the first 20 descriptors were calculated respectively:

$$PAAC_{AAIndex}(j = 1, 2, ..., 20) = \frac{f_j}{\sum_{j=1}^{20} f_j + \sum_{d=1}^{30} \delta_1(d)}$$
$$APAAC_{AAIndex}(j = 1, 2, ..., 20) = \frac{f_j}{\sum_{j=1}^{20} f_j + \sum_{d=1}^{30} \delta_2(d)}$$

The next 30 sequences can be calculated as follows:

$$PAAC_{AAIndex}(j = 21, 22, ..., 50) = \frac{\delta_1(j-20)}{\sum_{j=1}^{20} f_j + \sum_{d=1}^{30} \delta_1(d)}$$
$$APAAC_{AAIndex}(j = 21, 22, ..., 50) = \frac{\delta_2(j-20)}{\sum_{j=1}^{20} f_j + \sum_{d=1}^{30} \delta_2(d)}$$

where f_j is the frequency of amino acid of type j.

For both the type I and II PAAC AAIndex derived descriptors, sets of 50 descriptors for each one of 553 AAIndex scales were calculated.

3.10 Auto-correlation descriptors

Moreau-Broto, Moran and Geary autocorrelation are well established measures commonly used for a sequence analysis [63]. Calculated descriptor values depend not only on the values of the physicochemical properties of amino acids, but also on positions of specific amino acids in the protein chain.

In order to calculate the auto-correlation descriptors all original numerical $AAIndex_k(j)$ (k represents one of 553 scales and j represents one of 20 amino acids) values were centralized and standardized:

$$\overline{AAIndex_k} = \frac{\sum_{j=1}^{20} AAIndex_k(j)}{20}$$

$$\sigma_k = \sqrt{\frac{1}{2}\sum_{j=1}^{20}(AAIndex_k(j) - \overline{AAIndex_k})^2}$$

Then the original values of AAIndex derived numerical sequences $AAIndex_k(j)$ were replaced with the normalised ones $Z_k(j)$ and based on these new values the auto-correlation descriptors are calculated.

$$Z_k(j) = \frac{AAIndex_k(j) - \overline{AAIndex_k}}{\sigma}$$

Moreau-Broto Autocorrelation [63]:

$$MBA(d) = \sum_{i=1}^{N-d} P_i P_{i+d}$$

Normalized Moreau-Broto Autocorrelation [63]:

$$nMBA(d) = \frac{1}{N-d}\sum_{i=1}^{N-d} P_i P_{i+d}$$

Moran Autocorrelation Descriptors [63]:

$$MA(d) = \frac{\frac{1}{N-d}\sum_{i=1}^{N-d}(P_i - \overline{P})(P_{i+d} - \overline{P})}{\frac{1}{N}\sum_{i=1}^{N}(P_i - \overline{P'})^2}$$

Geary Autocorrelation Descriptors [63]:

$$GA(d) = \frac{\frac{1}{2(N-d)}\sum_{i=1}^{N-d}(P_i - P_{i+d})^2}{\frac{1}{N-1}\sum_{i=1}^{N-d}(P_i - \overline{P})^2}$$

P_i and P_{i+d} are the values of i^{th} and $i+d^{th}$ entry in the examined numerical sequence studied. $d \in \{1, 2, ...30\}$. N is a sequence length. $\overline{P}$ is the average value of the entries in that sequence: $\overline{P} = \frac{\sum_{i=1}^{N} P_i}{N}$

3.11 Results and Discussion

The data set containing 3872 proteins belonging to 40 protein families (full list can be found in Appendix C) was analysed in this chapter. They were taken from the well established Pfam database, containing a large selection of protein families [77]. The descriptors values were calculated for each of the examined proteins, and multi-dimensional vectors, comprising descriptor values type were created.

The **comparison of the accuracy** of calculated statistical descriptor vectors were performed with **Linear Discriminant Analysis** (LDA) classifier (supervised machine learning technique) inbuilt in the Matlab software [52]. LDA finds the linear combination of independent continuous variables, and in that aspect is similar to Principal Component Analysis. However, instead of maximising the variance of the data set as a whole, LDA maximises the differences between data points belonging to the separate classes and minimises the intra-class variance.

The Linear Discriminant analysis technique is a popular choice in many research fields, including heart rate analysis[78] and DNA functional motif recognition [79] and cancer research [80, 81]. It was proven effective in the process of differentiation between the cancer and the healthy tissues based on the gene expression data. Apart from being known for its effectiveness as a linear technique, LDA requires very little computational power, a key reason for our choice in dealing with BIG data-sets.

3.11.1 Misclassification Error

Here we propose an algorithm to compute the misclassification error of a given descriptor vectors set as follows:

1. k (number of errors) to be set equal to 0;
2. first protein from the K element data set to be selected;
3. the LDA classifier inbuilt in the Matlab software to be trained using
   a) protein data set as training data,
   b) family affiliation as the label;
4. if the classification is deemed correct, the selected protein would be classified as one belonging to its parent family, the value of k remaining unchanged;
5. if the classification is deemed incorrect, the selected protein would be classified as one belonging to family that is different to its parent one, then k would be upgraded by a unit: $k = k + 1$;
6. steps 2-5 to be repeated across all proteins from the K-elements data set;

7. the **misclassification error** would be calculated by dividing the number of errors k by the number of examined proteins K.

Table 13 presents the results of the misclassification error of various descriptor vectors described in this chapter.

Descriptor	n_f	misclassification error	threshold value
Amino Acid Composition (AAC)*	20	1004 (25.93%)	800
Dipeptide Composition (DC)*	400	53 (1.37%)	40
AAC* and DC*	420	49 (1.27%)	38
Entropy and Conditional entropy	6	2128 (54.96%)	2668
$H[DC_{AAC}]$[1]	20	711 (18.36%)	800
$H[TC_{AAC}]$[1]	20	639 (16.50%)	800
$H[TC_{DC}]$[1]	400	300 (7.75%)	40
$H[DC_{AAC}]$, $H[TC_{AAC}]$, $H[TC_{DC}]$	440	142 (3.67%)	36
Proximity between amino acids:			
Mean (one direction)	400	244 (6.30%)	40
SD (one direction)	400	341 (8.81%)	40
Mean (both directions)	400	307 (7.93%)	40
SD (both directions)	400	400 (10.33%)	40
Mean and SD (one direction)	800	22 (0.57%)	20
Mean and SD (both directions)	800	49 (1.27%)	20
Composition*	21	1472 (38.01%)	762
Transition*	21	1461 (37.73%)	762
Distribution*	105	1746 (45.09%)	152
Composition/Transition/Distribution*	147	521 (13.46%)	108
Conjoint Triad (CT)*	343	233 (6.02%)	46
Sequence Order Coupling Numbers ($SOCN$)*	60	513 (13.25%)	266
Quasi sequence order descriptors ($QSOD$)*	100	214 (5.52%)	160
Pseudo Amino Acid Composition I ($PAAC$)*	50	361 (9.32%)	320
Pseudo Amino Acid Composition II ($APAAC$)*	80	260 (6.71%)	200
$SOCN$*, $QSOD$*, $PAAC$*, $APAAC$*	290	90 (2.32%)	55
CT*, $SOCN$*, $QSOD$*, $PAAC$*, $APAAC$*	633	18 (0.46%)	25

[1] partial conditional entropy
$H[DC_{AAC}]$ based on proportions of dipeptides starting with specific amino acid
$H[TC_{AAC}]$ based on proportions of triprptides starting with specific amino acid
$H[TC_{AAC}]$ based on proportions of triprptides starting with specific dipeptide

Table 13: The descriptor vectors misclassification error. n_f denotes the dimension of the descriptor vector. Descriptor vectors marked with $*$ had their values calculated using the protr package[63]. The combinations of the descriptor vectors are marked in gray. The values of misclassification error that fall below corresponding threshold values are marked in red.

The number of misclassified protein varied from 18 (0.46%) for a combination of five different descriptor vectors (CT, $SOCN$, $QSOD$, $PAAC$ and $APAAC$) or 53 (1.37%) for single descriptor vector (Dipeptide Composition) to 2128 (25.93%) for entropy and conditional entropy. However comparing those results seemed unreasonable as the dimensions of

the descriptor vectors varied significantly (6 vs. 400 vs. 633). Even a superficial analysis of the results presented in the Table 13 suggests a negative correlation between the dimension of the descriptor vector and the misclassification error. That implies that in order to achieve protein classification closer to the correct one, we need to use the descriptor vectors with higher dimension. We will also investigate whether considering the larger number of physicochemical properties of amino acids (using the AAIndex scales) would increase the classification accuracy.

Figure 18 shows the dependence between the dimension of the single descriptor vector and the misclassification error for the standard non-alignment descriptors (red crosses). As expected, the misclassification error decreased with increasing number of descriptors. Even for the best performing descriptors, this decrease was not linear. We have extrapolated the results with $y = \frac{a}{x}$ function, for the value of parameter $a \approx 16011.62$. The fitted line (Figure 18) coincides best with the results obtained for the descriptors that give lowest misclassification error for any given descriptors number. We propose that the descriptor vectors can be considered to be better performing than the standard ones if the pairs $\{x_i, y_i\}$ (where x_i is a dimension of the descriptor vector, and y_i is the number of misclassified proteins) lay below the fit line. For example, for $x_i = 50$, y_i must be lower or equal than the threshold value of 320. The threshold values corresponding to the dimension of the examined descriptor vectors are presented in the last column of Table 30.

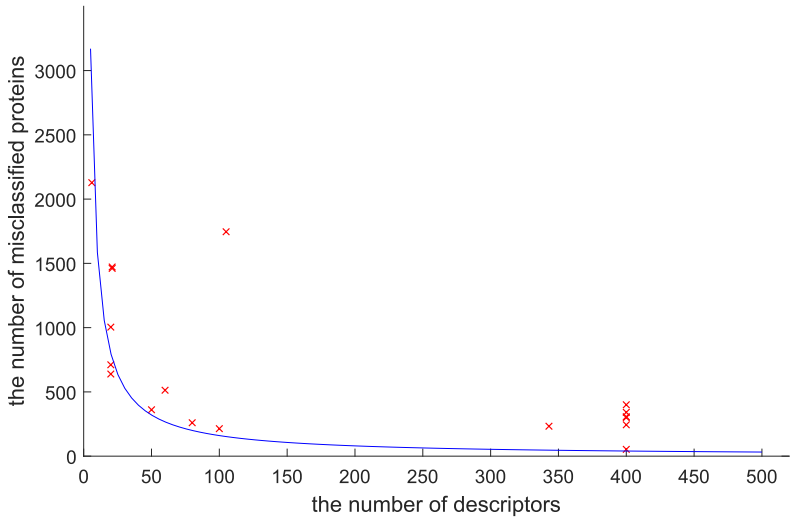

Figure 18: The dependence between the number of used descriptors and misclassification error shown in Table 13. The combination of the descriptor vectors are not taken into account

For the combinations of descriptor vectors (marked in gray in Table 13), the threshold values are given in an ascending order: 20 (for $x_i = 800$), 25 (for $x_i = 633$), 36 (for $x_i = 440$), 38 (for $x_i = 420$), 55 (for $x_i = 290$) and 108 (for $x_i = 147$). Only one out of seven examined combinations of standard descriptors, have the misclassification error that falls below its

dimension threshold value: CT with $SOCN$ with $QSOD$ with $PAAC$ and $APAAC$ (number if misclassified protein $18 < 22$), while the combination of mean and standard deviation of proximity between amino acids in one direction is marginally greater (22 vs. 20).

However our expectation for the misclassification error fitting the line $y = \frac{a}{x}$ may be overly optimistic, especially for the descriptor vectors characterized by large dimensions. The fit line consistently lay above the actual misclassification error results in most cases (the exceptions are marked red in Table 13,). This observation established that results, obtained for the combination of CT with $SOCN$ with $QSOD$ with $PAAC$ and $APAAC$, and the one obtained for he combination of mean and standard deviation of proximity between amino acids (the number of misclassified proteins: 22 and 49) are exceptional.

The Figure 19 shows the variation of the proportion of misclassification error divided by corresponding threshold value against the dimensions of the descriptor vectors. Blue dashed lines represents $y = 1$, $y = 1.5$, $y = 2$ and $y = 4$ to highlight the misclassification error results smaller than corresponding threshold value, greater than corresponding threshold value but smaller than corresponding threshold value multiplied by 1.5 etc.

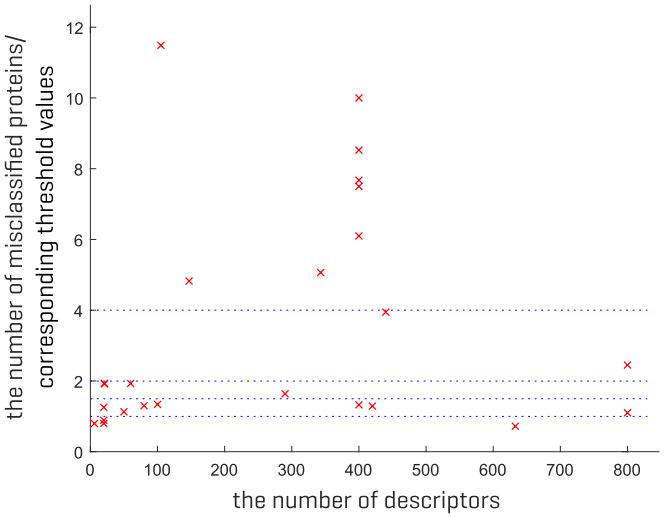

Figure 19: The proportion of misclassification error divided by corresponding threshold value plotted against the dimension of the descriptor vectors. Blue dashed lines represents plots $y = 1$, $y = 1.5$, $y = 2$ and $y = 4$.

We used the threshold values obtained from the $y = \frac{a}{x}$ fit and its re-scaled values to determine the relative quality of performance of the $PAAC_{AAIndex}$ and the Auto-correlation descriptor vectors. However, we do not suggest that the threshold values can be treated as an objective measure of any kind.

For all previously defined auto-correlation types, 30 sets of 553-dimensional descriptor vectors were constructed, and the misclassification errors were calculated (Table 14). The smallest number of the misclassified proteins was obtained for $d = 1$ when the auto-correlation was

d	$MBA(d)$	$nMBA(d)$	$MA(d)$	$GA(d)$
1	270 (6.97%)	307 (7.93%)	150 (3.87%)	149 (3.85%)
2	307 (7.93%)	371 (9.58%)	179 (4.62%)	169 (4.36%)
3	326 (8.42%)	350 (9.04%)	186 (4.80%)	163 (4.21%)
4	341 (8.81%)	379 (9.79%)	206 (5.32%)	173 (4.47%)
5	413 (10.67%)	455 (11.75%)	230 (5.94%)	197 (5.09%)
6	389 (10.05%)	457 (11.80%)	264 (6.82%)	236 (6.10%)
7	434 (11.21%)	491 (12.68%)	241 (6.22%)	225 (5.81%)
8	405 (10.46%)	482 (12.45%)	307 (7.93%)	259 (6.69%)
9	435 (11.23%)	472 (12.19%)	285 (7.36%)	242 (6.25%)
10	467 (12.06%)	530 (13.69%)	310 (8.01%)	265 (6.84%)
11	474 (12.24%)	520 (13.43%)	337 (8.70%)	258 (6.66%)
12	483 (12.47%)	555 (14.33%)	373 (9.63%)	309 (7.98%)
13	527 (13.61%)	578 (14.93%)	373 (9.63%)	281 (7.26%)
14	513 (13.25%)	583 (15.06%)	340 (8.78%)	255 (6.59%)
15	550 (14.20%)	618 (15.96%)	397 (10.25%)	298 (7.70%)
16	550 (14.20%)	628 (16.22%)	377 (9.74%)	290 (7.49%)
17	560 (14.46%)	636 (16.43%)	387 (9.99%)	289 (7.46%)
18	533 (13.77%)	618 (15.96%)	382 (9.87%)	280 (7.23%)
19	624 (16.12%)	668 (17.25%)	439 (11.34%)	313 (8.08%)
20	570 (14.72%)	677 (17.48%)	453 (11.70%)	314 (8.11%)
21	543 (14.02%)	644 (16.63%)	410 (10.59%)	303 (7.83%)
22	585 (15.11%)	672 (17.36%)	431 (11.13%)	321 (8.29%)
23	563 (14.54%)	643 (16.61%)	468 (12.09%)	329 (8.50%)
24	631 (16.30%)	702 (18.13%)	485 (12.53%)	356 (9.19%)
25	592 (15.29%)	684 (17.67%)	471 (12.16%)	338 (8.73%)
26	645 (16.66%)	707 (18.26%)	484 (12.50%)	314 (8.11%)
27	627 (16.19%)	710 (18.34%)	468 (12.09%)	310 (8.01%)
28	663 (17.12%)	751 (19.40%)	509 (13.15%)	341 (8.81%)
29	651 (16.81%)	785 (20.27%)	512 (13.22%)	372 (9.61%)
30	630 (16.27%)	716 (18.49%)	508 (13.12%)	321 (8.29%)

Table 14: The Auto-correlation Descriptors misclassification error. d is the proximity between amino acids, $MBA(d)$ denotes Moreau-Broto auto-correlation, $nMBA(d)$ normalized Moreau-Broto auto-correlation, $MA(d)$ Moran auto-correlation and $GA(d)$ Geary auto-correlation.

calculated for two consecutive values in the numerical sequence. But even the best results 3.87% and 3.85% (150 and 149 misclassified proteins) were still way above 28, the threshold value for the 553-dimensional descriptor vector. For the larger values of d, the misclassification error becomes even larger.

For all values of the parameter d, the Geary auto-correlation descriptors performed better than the Moran auto-correlation descriptors, and the second ones performed better than the Moreau-Broto auto-correlation descriptors with normalised Moreau-Broto giving the highest misclassification error of all.

The 50-dimensional vectors of AAIndex derived Pseudo Amino Acid Composition descriptors (Type I and Type II) were analysed separately for each of the AAIndex derived protein

numerical sequence. That means that each one of the examined protein is represented by 2 × 553 descriptor vectors. For the $PAAC_{AAIndex}$ Type I the lowest misclassification error is 10.15% (393 misclassified proteins) and the highest one is 23.40% (906 misclassified proteins). For the Type II $PAAC_{AAIndex}$ the lowest misclassification error is 5.71% (221 misclassified proteins) and the highest one is 14.88% (576 misclassified proteins). Figure 20 shows the misclassification error distribution histogram for the Type I and Type II $PAAC_{AAIndex}$ descriptors.

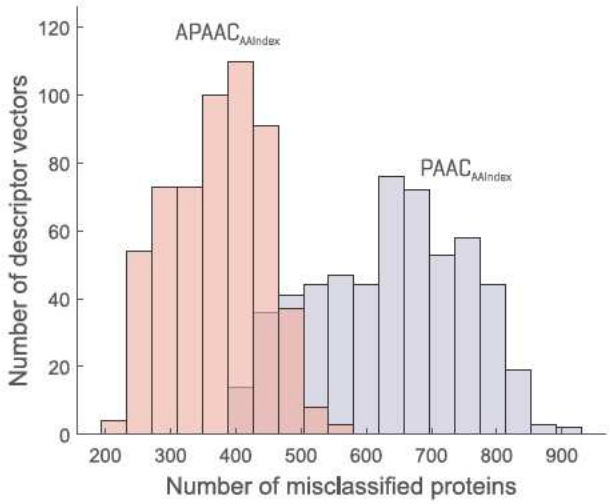

Figure 20: The misclassification error distribution histogram for the Type I and Type II $PAAC_{AAIndex}$ descriptors.

As mentioned before, the threshold value for the 50–dimensional descriptor vector as marked at 320 (8.26% misclassified proteins). For the Type I Pseudo Amino Acid Composition, none of the 553 AA Index based descriptor vectors roduced misclassification error below that value. All the descriptor vectors however produced the misclassification error below (960 = 3× the threshold value), and for most of the 50-elements descriptors sets the observed misclassification error lay between 2 and 3× the threshold value (Table 15).

Misclassification error interval	Number of descriptor vectors	
	$PAAC_{AAIndex}$	$APAAC_{AAIndex}$
(0, 320]	0	155
(320, 480]	65	369
(480, 640]	198	29
(640, 960]	290	0

Table 15: The number of AAIndex derived $PAAC_{AAIndex}$ and $APAAC_{AAIndex}$ 50-dimensional descriptor vectors giving misclassification errors in ranges whose edges are defined by multiples of threshold values (0, 1, 1.5, 2 and 3).

Around 28% (155 out of 553) of the Amphiphilic Pseudo Amino Acid Composition AAIndex based ($PAAC_{AAIndex}$ Type II), descriptor vectors, produced misclassification error lower

than or equal to the threshold value, while the following 369 descriptor vectors gave misclassification errors greater than the threshold value, but lower than or equal to the threshold value multiplied by the scaling factor of 1.5 (480 misclassification protein). The remaining 29 descriptor vectors produced misclassification error lower than or equal to 1.8 $\times$ the threshold value (Table 15) .

From the sets of descriptors calculated for Type II $PAAC_{AAIndex}$, the 11 best performing were chosen (each having a misclassification error lower than 6.18%; 239 misclassified protein at most), and then the misclassification error was calculated for the combined vector of $11 \times 50 = 550$ descriptors. The obtained misclassification error was 0.88% (34 misclassified proteins). This result is around 17% greater than 29, the threshold value for the 550-dimensional descriptor vector. These results lead us to conclude that AAIndex derived Pseudo Amino Acid Composition descriptors perform well on average in terms of estimating the discrimination between protein families, with the Type II performing significantly better.

4 Amino acids into time series transformation

4.1 Data set

For the main part of the research presented in this thesis, the extended set containing 100 protein families (9693 proteins, full list can be found in Appendix D) were taken from the established Pfam database, containing a large selection of protein families [77]. The protein families were chosen randomly, in a way that they either belonged to different clans (Pfam generated higher-level groupings of protein families), or without any clan affiliation. Initial analyses shown in the previous chapter were performed for the relatively smaller dataset containing 3872 proteins belonging to 40 protein families while the clan affiliation was not taken into account at that preliminary stage of the research.

4.2 Time series prepossessing

As a first step, the amino acid sequence was transformed into a time series by replacing each amino acid with a numerical value (see Figure 25). Those values were taken from AAIndex database [75], that contains numerical representations of various physicochemical and biochemical properties of amino acids, derived from published literature. The 553 different scales (each providing numerical values for all Amino acids, representing different property such as hydrophobicity or charge) of have been used, therefore each protein is now represented by 553 different time series.

To "smooth out" short term fluctuations, the **moving average** of the time series was calculated with Matlab software [52] with the averaging window size 7, the value that was successfully used in protein-derived time series analysis [82] (the averaging window size is a parameter, and can be changed in future applications). Three types of moving averages were calculated (see Figure 22). Firstly, the simple moving average, where each moving average value depends only on the values of seven consecutive entries of original time series. Secondly, the modified moving average, where each moving average value depends on all preceding values from the original time series, with weights decreasing with time. And finally, the exponential moving average, yet another type of weighted moving average that places even greater weight to the most recent data point.

To calculate the simple moving average, the arithmetic mean of the first seven values of the time series was calculated and the time series entry at position one was replaced with the calculated value. Then the mean of the next seven values was calculated, starting from the entry at the position two. The result replaced the second entry of original time series. The calculations continued until the end of the time series.

$$SM_i = \frac{1}{7}\sum_{k=i}^{i+6} x_k$$

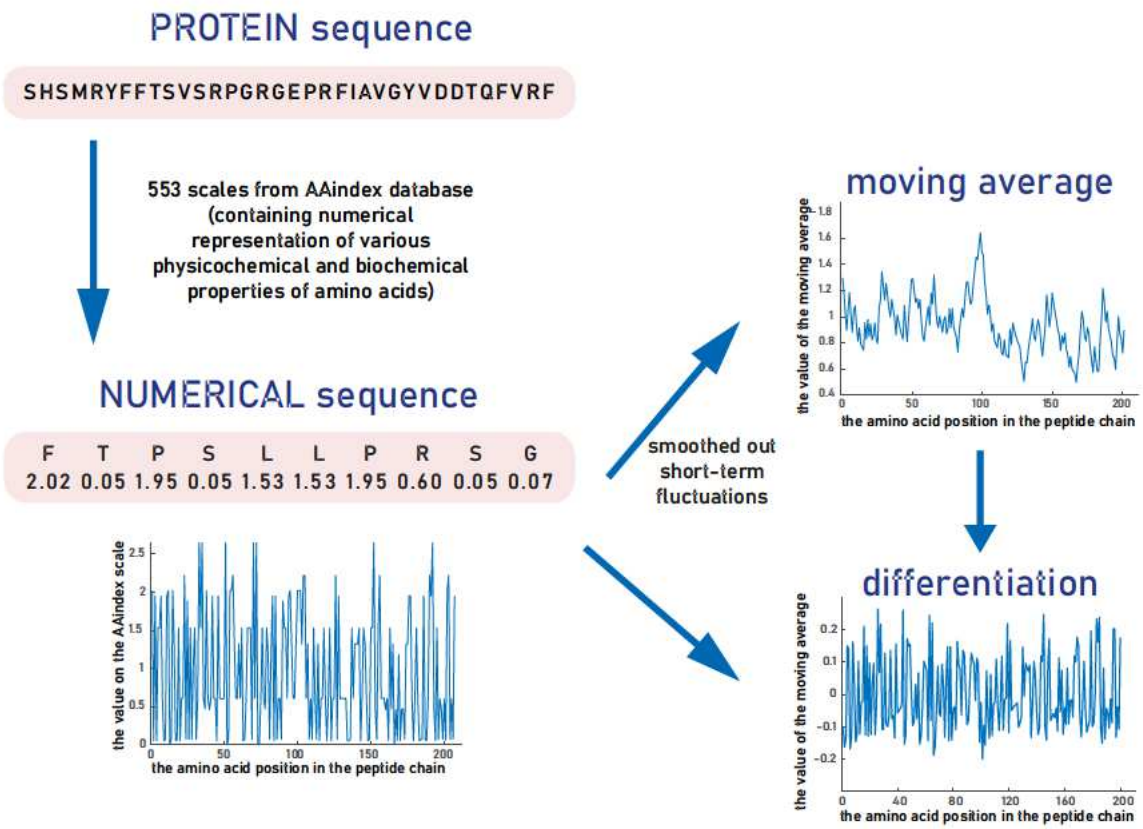

Figure 21: The summary of the protocol that transforms amino acid sequence into moving average time series.

SM_i is the i^{th} entry of the simple moving average time series and x_i is the i^{th} entry of the original time series.

The illustration of simple moving average calculations can be found in the Figure 22 (top right panel). The SM_i value of the simple moving average can be also calculated based on moving average previous $SM_{i-1}h$ value and the current value of the original time series.

$$SM_i = SM_{i-1} + \frac{x_{i+6}}{7} - \frac{x_{i-1}}{7}$$

However the values of simple moving average entries do not contain any information about the past values of original time series (see Figure 23).

For the modified moving average, the first entry was calculated exactly as for the simple moving average as the arithmetic mean of the first 7 values of the time series. For the second entry, the first value of moving average was taken and the next entry from the original time series divided by seven was added. Then the value of previous moving average, also divided by seven was subtracted.

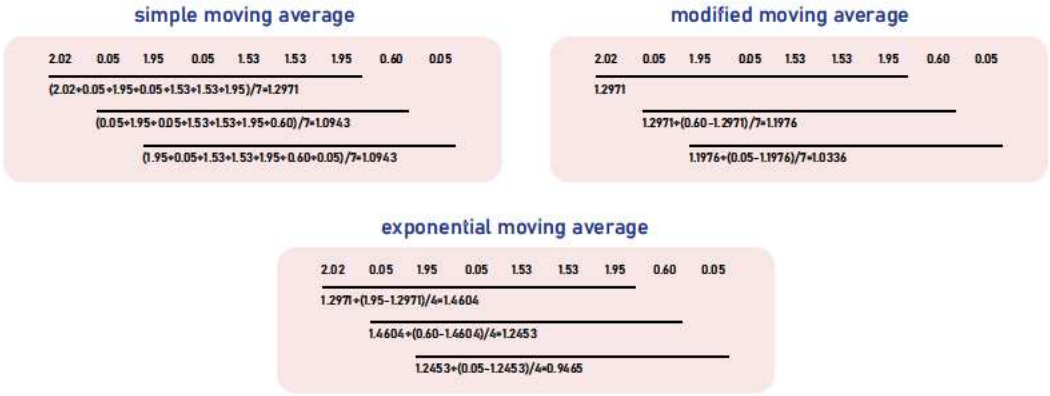

Figure 22: The moving average calculation.

$$MM_1 = SM_1 = \frac{1}{7}\sum_{k=1}^{7} x_k$$
$$MM_i = MM_{i-1} + \frac{x_{i+6}}{7} - \frac{MM_{i-1}}{7}$$

MM_i is the i^{th} entry of the modified moving average. The illustration of modified moving average calculations can be found in the Figure 22 (top left panel). The current modified moving average value depends on all previous values of the original time series, with weights decreasing with time (see Figure 23). The highest weight, the one used for the most recent entry x_{i+6}, is equal $\frac{1}{7}$

$$MM_i = \frac{6^{i-1}}{7^i}\sum_{k=1}^{7} x_k + \sum_{k=8}^{i+6}\left(\frac{6^{i+6-k}}{7^{i+7-k}}\right) x_k$$

The exponential moving average is yet another type of weighted moving average. It places even greater weight on the most recent data point than the modified or simple moving average. In case of window size 7, this weight was equal to $\frac{1}{4}$, and calculated using formula $w = \frac{2}{windowsize+1}$. The calculations of the value of the first entry started with the arithmetic mean of the first seven values of the time series multiplied by factor $(1-w)$. Then the most recent value of time series x_7 was multiplied by the factor w. The next entries were calculated based on the most recent x_{i+6} value of the original time series and the value of exponential moving average for the previous entry.

$$EM_1 = \frac{3}{4}SM_1 + \frac{1}{4}x_7 = \frac{1}{7}\sum_{k=1}^{7} x_k + \frac{x_7}{4} - \frac{\frac{1}{7}\sum_{k=1}^{7} x_k}{4}$$
$$EM_i = EM_{i-1} + \frac{x_{i+6}}{4} - \frac{EM_{i-1}}{4}$$

EM_i is the i^{th} entry of the exponential moving average time series and x_i is the i^{th} entry of the original time series. The illustration of the exponential moving average calculations can be found in Figure 22 (bottom panel). As in the case of the modified moving average, the current value of the exponential moving average depends on all of previous values of the original time series, but their impact is smaller than the one applied in the modified moving average calculations (see Figure 23).

$$EM_i = \frac{3^i}{7 * 4^i} \sum_{k=1}^{7} x_k + \sum_{k=7}^{i+6} \left(\frac{3^{i+6-k}}{4^{i+7-k}} \right) x_k$$

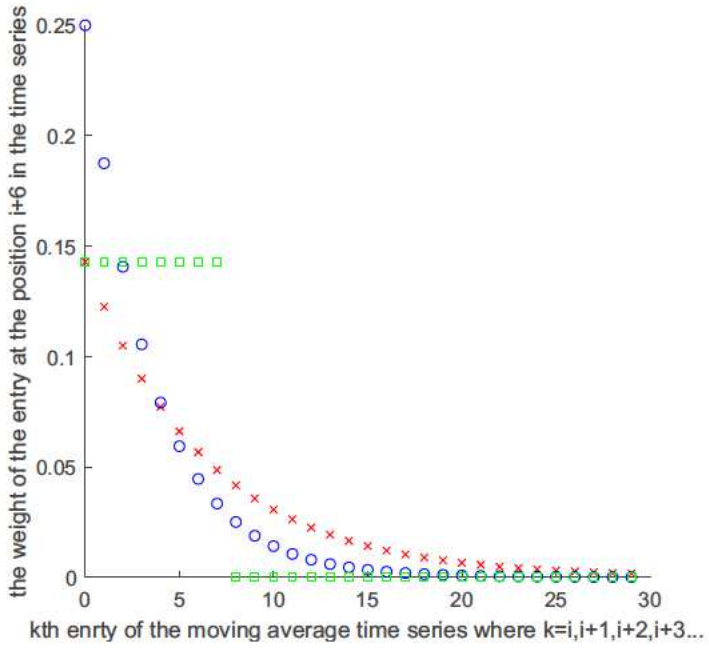

Figure 23: The comparison of the weights of the consecutive entries of original time series in the moving average calculation (simple moving average is represented by green circles, modified by red crosses and exponential by blue circles). In case of modified and exponential moving average the values of i^{th} entry depends on all previous entries, but the dependence decrease with distance between entries.

The **differentiation of time series** was calculated as:

$$y_i = x_{i+1} - x_i$$

$$z_i = y_{i+1} - y_i$$

where x_{i+1} and x_i are two consecutive entries of time series. Therefore, with the step size $\Delta x = 1$, $y(t) = \frac{dx(t)}{dt}$ and $z(t) = \frac{d^2x(t)}{dt^2}$

We hypothesise that, in some cases, instead of the actual value of various physicochemical and biochemical properties of amino acids, the changes of these values can be crucial in discriminating between protein families. These differences can be used to calculate the

differentiated time series. Mapping from mathematics and physics, time series differentiation is equivalent to calculating derivatives of functions. The first derivative $y(t)$ analyses the changes in values while the second derivative $z(t)$ defines the rate of such changes.

Based on this remodelled time series, twelve 553-dimensional vectors of numerical descriptors were calculated for each protein, as is summarised below and in Figure 24. It was done by applying an established method from information theory and statistical physics, with algorithms developed for time series analysis, as will be described in detail in the next chapters.

In summary,:

- Each protein (amino acid sequence) was transformed into a 553 time series based on 553 propensity scales, and then for each descriptor type, 553-dimensional vector was calculated.
- Each protein was transformed into a 3×553 moving average time series based on already created 553 time series, and then for each descriptor type, three 553-dimensional vectors was calculated.
- Each protein was transformed into a 2×553 differentiated time series based on already created 553 time series, and then for each descriptor type, two 553-dimensional vectors was calculated.
- Each protein was transformed into a $2 \times 3 \times 553$ differentiated moving average time series based on already created 3×553 moving average time series, and then for each descriptor type, six 553-dimensional vectors was calculated.

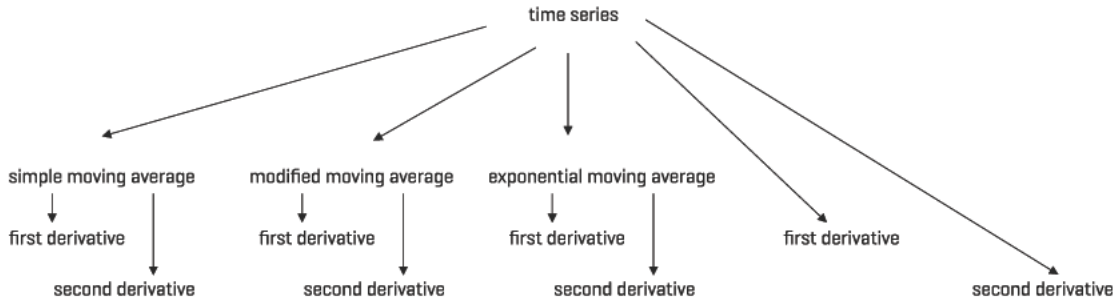

Figure 24: The assignment of twelve 533-dimensional vectors to a single amino acid sequence.

5 Descriptors based on information theory

5.1 Approximate entropy

Approximate entropy is a statistical technique that allows us to quantify the "regularity" of the examined time series. [83, 84, 85]. As such this provides the means to discriminate between a "regular" time series, that is characterised by a large proportion of the repeated segments, and an "irregular" time series, that is characterised with larger unpredictability.

Let's compare two time series:
A: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
B: 0 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 1 0

They cannot be distinguished by comparison of their means or variances, as the values 0 and 1 occur in both with the same probability: $p = 0.5$. Shannon entropy calculated for both of them will also have the same value: $H = 0.5 \times \log 0.5$. Hence the need for a new statistic, that allows us not only discriminate between time series with different levels of repetition, but also reliably quantifies the repetition levels.

The value of Approximate Entropy ($ApEn$) for a given time series depends on the chosen parameters: tolerance r and the length of a template vector m [83, 84, 85].

Let's consider time series $\{x_1, x_2, x_3, ..., x_N\}$ of a length of N. The template vector of a length m is defined as $x_m(i) = \{x_i, x_{i+1}, ..., x_{i+m-1}\}$. All possible template vectors of the length m, that can be created from a given time series:

$$
\begin{aligned}
&x_m(1) = \{x_1, x_2, ..., x_m\} \\
&x_m(2) = \{x_2, x_3, ..., x_{m+1}\} \\
&x_m(3) = \{x_3, x_4, ..., x_{m+2}\} \\
&... \\
&x_m(i) = \{x_i, x_{i+1}, ..., x_{x+m-1}\} \\
&... \\
&x_m(N-m+1) = \{x_{N-m+1}, x_{N-m+2}, ..., x_N\}
\end{aligned}
$$

We calculate the Chebyshev distance between each pairs of template vectors:

$$d[x_m(i), x_m(j)] = \max_k |u(k) - v(k)|$$

where $u(k)$ and $v(k)$ are components of vectors $x_m(i)$ and $x_m(j)$ respectively; $k \in \{1, 2, .., m\}$.

For each template vector $x_m(i)$ we calculate $B_m(i)$ defined as the number of template vectors $x_m(j)$ such that Chebyshev distance $d[x_m(i), x_m(j)] < r$. In this case i can be equal to j, so the value of $B_m(i)$ is always equal at least 1 ($d[x_m(i), x_m(i)] = 0$). Then we normalize

the value of $B_m(i)$ by the number of all template vectors of the length of m.

$$C_m(i) = \frac{B_m(i)}{N - m + 1}$$

Now we define Φ^m:

$$\Phi^m = \frac{1}{N - m + 1} \sum_{i=1}^{N-m+1} \log C_m(i)$$

Approximate entropy ($ApEn$) is defined as:

$$ApEn = \Phi^m - \Phi^{m+1}$$

From the definition, the value of $\sum B_m$ will always be larger or equal than the value of B_{m+1}, and the value of $ApEn$ will always be greater than 0. The smaller the value of the $ApEn$, the more regular and predictable the examined data is.

It is worth mentioning a disadvantage of the approximate entropy is that it does not guarantee the relative relative consistency [85, 86]. This means that if the value of the approximate entropy calculated for one time series is greater than for another, it may not necessarily remain larger for different values of the parameters m and r.

Based on 553 protein sequence derived time series (and moving average and/or differentiated time series), the 553-dimensional vectors of $ApEn$ values (later called descriptors) were assigned to each protein. As in previous chapters, the comparison of the accuracy of the performance of calculated descriptor vectors was performed with the Linear Discriminant Analysis classifier inbuilt in the Matlab software [52] and is shown in Table 16.

The recommended value of the parameter m is low (m=2 or m=3) [85]. This is particularly important for relatively short time series (the length of protein sequence derived time series in the examined test set vary from 105 to 770 entries). The reason behind it is that for larger m we may not be able to find enough template vector pairs such that Chebyshev distance $d[x_m(i), x_m(j)] < r$.

The usual recommendation for the value of the parameter r is in the range between 0.1 to 0.25 of the time series standard division (Std) [85]. However in the case of some protein-derived time series for the $r = 0.25 \times Std$ the template vectors were similar only to themselves (similarity understood as Chebyshev distance between vectors lower than value of $r4$). Results influenced by this problem are marked * in Table 16. Therefore we have decided to use also value of parameter $r = 1 \times Std$.

For non-averaged protein-derived time series we see smaller misclassification error for single time and twice differentiated time series, than for time series without differentiation. Apart from the values of parameter $m = 2$ and $r = 0.25 \times Std$ double differentiation provides

	$m = 2$		$m = 3$	
	$r = 0.25 \times Std$	$r = 1 \times Std$	$r = 0.25 \times Std$	$r = 1 \times Std$
time series without differentiation				
time series	355 (3.66%)	663 (6.84%)	447 (4.61%)*	504 (5.20%)
simple moving average	310 (3.20%)	536 (5.53%)	353 (3.64%)*	519 (5.35%)
modified moving average	237 (2.45%)	509 (5.25%)	274 (2.83%)*	501 (5.17%)
exponential moving average	289 (2.98%)	351 (3.62%)	251 (2.59%)*	312 (3.22%)
once differentiated time series				
time series	269 (2.78%)*	313 (3.23%)	295 (3.04%)*	276 (2.85%)
simple moving average	389 (4.01%)*	432 (4.46%)	372 (3.84%)*	376 (3.88%)
modified moving average	221 (2.28%)	225 (2.32%)	211 (2.18%)*	226 (2.33%)
exponential moving average	270 (2.79%)	238 (2.46%)	222 (2.29%)*	261 (2.69%)
twice differentiated time series				
time series	275 (2.84%)*	293 (3.02%)	264 (2.72%)*	264 (2.72%)
simple moving average	399 (4.12%)*	427 (4.41%)	372 (3.84%)*	377 (3.89%)
modified moving average	217 (2.24%)	227 (2.34%)	203 (2.09%)*	230 (2.37%)
exponential moving average	259 (2.67%)	249 (2.57%)	221 (2.28%)*	239 (2.47%)

Table 16: The approximate entropy descriptors misclassification error defined as the number of misclassified proteins for 553-dimensional vectors (each dimension corresponds with one scale obtained from AAIndex database). * indicates the results influenced by the fact in case of some protein-derived time series the template vectors were similar only to them self.

slightly better result than single one. The best result for the undifferentiated simple moving average time series is achieved for the value of parameters $m = 2$ and $r = 0.25 \times Std$ (310, 3.20%). But still all but one ($m = 2$, $r = 1 \times Std$, the number of misclassified proteins: 313, the percentage 3.23%) results for non-averaged single and double differentiated time series are better than the best of simple moving average time series.

For exponential and modified moving average in all cases apart from when $m = 2$ and $r = 0.25 \times Std$, the single and double differentiation decrease the misclassification error in comparison with the undifferentiated time series. This improvement is generally more pronounced in the case of modified moving average than the exponential one.

The value of the parameter $r = 0.25 \times Std$ gives generally better results than the value of the parameter $r = 1 \times Std$ in most cases of the single and double differentiated time series (averaged and non-averaged). In the case of time series without differentiation the misclassification error improvement is significant.

Approximate entropy is bias statistic. In general it is heavily dependent on the length N of the examined time series [85] (See Figure 25 A). Our results shows that this dependence is not pronounced for relatively (compered to more "traditional" physical time series) short protein-derived time series, in case of larger value of parameter $r = 1 \times Std$, that insures relatively low values of the approximate entropy denoting high self similarity of the time series (see Figure 25 B).

This result is consistent with what could be expected from the way the approximate entropy is calculated, and led us to propose an Approximate-entropy-based-descriptor derived

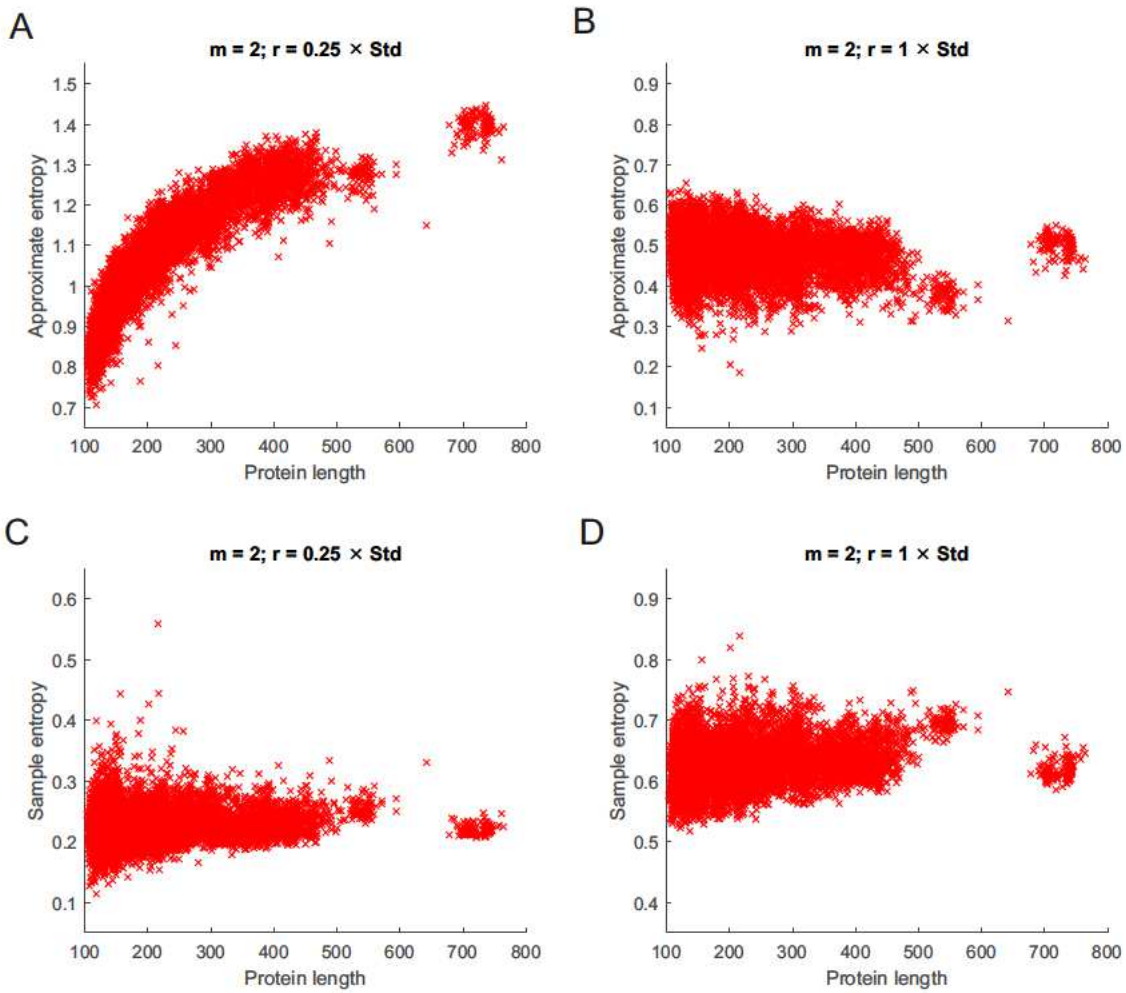

Figure 25: The example of the dependence of the value of approximate entropy (A and B) and the value of the sample entropy (C and D) on protein length. The approximate and sample entropy calculated for the exponential moving average time series without differentiation for hydrophobicidy scale taken from AAIndex database.

with the linear regression formula.

5.2 Linear regression application for the approximate entropy

From the definition:

$$\Phi^m = \frac{1}{k}\sum_{i=1}^{k} \log \frac{B_m(i)}{k}$$

where $k = N + m - 1$. Therefore:

$$\Phi^m = \frac{1}{k}\sum_{i=1}^{k} \log \frac{B_m(i)}{k} = \frac{1}{k}\sum_{i=1}^{k-1} \log B_m(i) + \frac{1}{k}\log B_m(k) - \log k =$$

Let's put in $C_1 = \frac{1}{k}\log B_m(k) - \log k$:

$$\begin{aligned}\Phi^m =& \frac{k-1}{k} * \frac{1}{k-1}\sum_{i=1}^{k-1}\log B_m(i) + C_1 \\ =& \frac{k-1}{k}\left(\frac{1}{k-1}\sum_{i=1}^{k-1}\log B_m(i) - \log(k-1) + \log(k-1)\right) + C_1\end{aligned}$$

Let's put in $C_2 = C_1 + (1-\frac{1}{k})\log(k-1) = \frac{1}{k}\log\frac{B_m(k)}{k-1} + \log\frac{k-1}{k}$:

$$\Phi^m = \frac{k-1}{k}\left(\frac{1}{k-1}\sum_{i=1}^{k-1}\log\frac{B_m(i)}{(k-1)}\right) + C_2$$

In case of time series characterised by a large proportion of the repeated segments (high value of the conditional probability $p(B_{m+1}(i)|B_m(i))$:

$$B_m(i) \approx B_{m+1}(i)$$

From the definition:

$$\Phi^{m+1} = \frac{1}{k-1}\sum_{i=1}^{k-1}\log\frac{B_{m+1}(i)}{k-1}$$

Therefore:

$$\Phi^m \approx \frac{k-1}{k}\left(\frac{1}{k-1}\sum_{i=1}^{k-1}\log\frac{B_{+1}m(i)}{(k-1)}\right) + C_2 = \frac{k-1}{k}\Phi^{m+1} + C_2$$

That leads to:

$$\Phi^{m+1} \approx \frac{N-m+1}{N-m}\Phi^m + C$$

where $C = \frac{1}{N-m}\log\frac{B_m(N-m+1)}{N-m} + \frac{N-m+1}{N-m}\log\frac{N-m}{N-m+1}$, for time series characterized by low value of the approximate entropy.

Let's calculate the value of Φ^m for $m \in \{1,2,3,...,\}$, according to the definition given for the approximate entropy calculation. X axis represent $\{\Phi^1,\Phi^2,\Phi^3,\Phi^4,.....,\Phi^{m-1}\}$ and Y axis represent $\{\Phi^2,\Phi^3,\Phi^4,\Phi^5,.....,\Phi^m\}$. Now we have points with coordinates $(x_i = \Phi^i, y_i = \Phi^{i+1})$. As can be seen in the Figure 26 A and B, points (Φ^i,Φ^{i+1}) approximate the straight line $(y = ax + b)$ more closely in case of $r = 0.25 \times Std$ than in case of $r = 1 \times Std$. As has been already stated, the larger value of parameter r, insures relatively high self similarity of the time series, resulting in the values $B_m(i) \approx B_{m+1}(i)$, and $a = \frac{N-m+1}{N-m} \approx 1$.

Even in cases, where pairs (Φ^i,Φ^{i+1}) cannot be sufficiently approximated by a linear re-

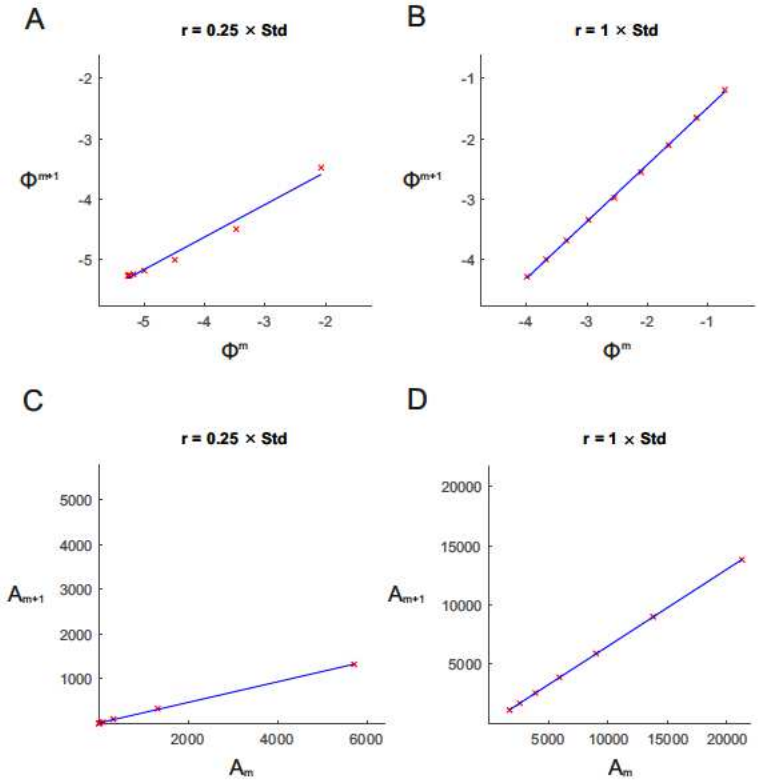

Figure 26: The graphical illustration of the linear regression application for the approximate entropy (A and B) and the sample entropy exponent (C and D).

gression line, still the calculations can be performed. Therefore we define the **Approximate-entropy-based-descriptor** as the slope of the best fit first degree polynomial $y = ax + b$ calculated for points $(x_i = \Phi^i, y_i = \Phi^{i+1})$.

The approximate-entropy-based-descriptor calculated with linear regression formula (Table 17) gives better discrimination between protein groups for single and double differentiated time series and for undifferentiated non-averaged time series in case of the value of the parameter $r = 1 \times Std$ compared to the value of $r = 0.25 \times Std$. For all three undifferentiated moving average time series the value of $r = 0.25 \times Std$ provides better protein group discrimination.

For the value of the parameter $r = 0.25 \times Std$ using linear regression is not obviously beneficial to decrease misclassification error while compared to the traditionally calculated approximate entropy (Table 16 vs. Table 17). It does that in some cases, but in majority the the effect is opposite: less accuracy in protein classification for linear regression derived descriptor vectors.

In case of the value of the parameter $r = 1 \times Std$, the results are much more consistent.The approximate-entropy-based-descriptor calculated with linear regression gives consistently better results than traditionally calculated approximate entropy with both examined values of the parameter m (with one exception, with equal number of misclassified protein for corresponding descriptor vectors).

	$r = 0.25 \times Std$	$r = 1 \times Std$
time series without differentiation		
time series	462 (4.77%)	424 (4.37%)
simple moving average	292 (3.01%)	437 (4.51%)
modified moving average	258 (2.66%)	316 (3.26%)
exponential moving average	229 (2.36%)	294 (3.03%)
differentiated time series		
time series	263 (2.71%)	200 (2.06%)
simple moving average	409 (4.22%)	355 (3.66%)
modified moving average	227 (2.34%)	166 (1.71%)
exponential moving average	270 (2.79%)	182 (1.88%)
twice differentiated time series		
time series	261 (2.69%)	193 (1.99%)
simple moving average	437 (4.51%)	377 (3.89%)
modified moving average	229 (2.36%)	163 (1.68%)
exponential moving average	283 (2.92%)	168 (1.73%)

Table 17: The approximate-entropy-based-descriptor vectors misclassification error defined as the number of misclassified proteins for 553-dimensional vectors (each dimension corresponds with one scale obtained from AAIndex database).

Figure 27 shows the example of the dependence of the value of the approximate-entropy-based-descriptor calculated with linear regression formula on the protein length. Not only the dependence remains (though seems to be slightly weaken) for the value of parameter $r = 0.25 \times Std$ (see Figure 25A and Figure 27A), but also the dependence is much more obvious in the case of the value of $r = 1 \times Std$ (see Figure 25B and Figure 27B). Even though we have chosen an example of protein-derived time series presenting typical behaviour, there is no guarantee that observed dependence will reappear for all examined proteins.

5.3 Sample entropy

The Sample entropy $SampEn$ [86] is a modification of the Approximate entropy. As in case of its predecessor, the $SampEn$ value for a given time series, depends on the chosen parameters: tolerance r and the length of a template vector m [85, 86].

Let's consider time series $\{x_1, x_2, x_3, ..., x_N\}$ of a length of N and all possible template vectors of the length m, that can be created from a given time series.

Again we calculate the Chebyshev distance between each pair of template vectors, but this time $i \neq j$:

$$d[x_m(i), x_m(j)] = \max_k |u(k) - v(k)|$$

where $u(k)$ and $v(k)$ are components of vectors $x_m(i)$ and $x_m(j)$ respectively; $k \in \{1, 2, .., m\}$.

Then we define A_m as the number of template vector pairs having $d[x_m(i), x_m(j)] < r$. A_m is a positive integer or zero. For that reason we have to make sure to choose the value of

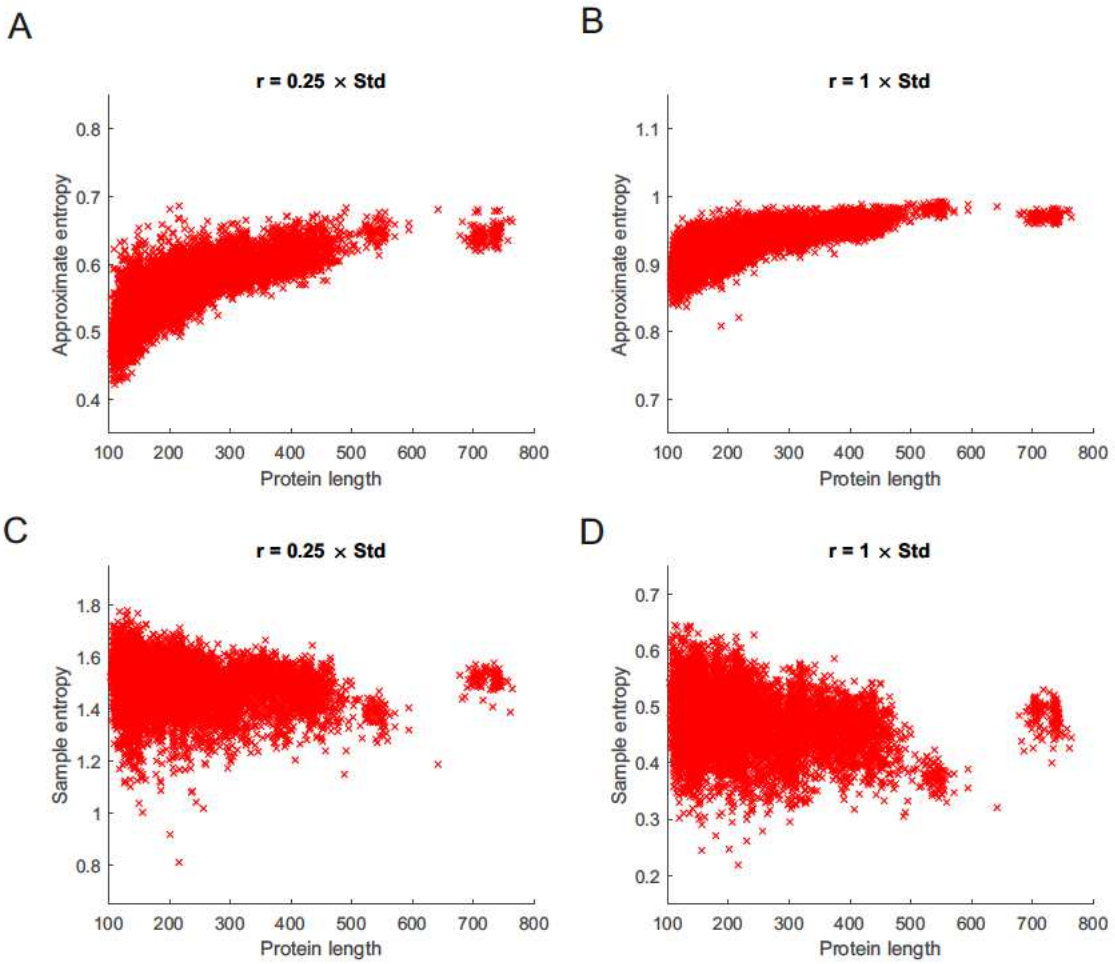

Figure 27: The example of the dependence of the value the approximate-entropy-based-descriptor calculated with linear regression formula (A and B) and the value of the sample-entropy-linear-descriptor (C and D) on the protein length. The protein descriptors were calculated for the exponential moving average time series without differentiation for the hydrophobicity scale taken from AAIndex database.

parameter m such as the value of A_m is greater than 0 to be able to calculate finite value for the $SampEn$ calculations. This limitation results in the fact that in case of some protein-derived time series for the $r = 0.25 \times Std$ we were not able to calculate the value of sample entropy. (See Table 18 and Table 19)

Sample entropy is define as:

$$SampEn = -\log \frac{A_{m+1}}{A_m}$$

From A_m definition, the value of A_m will always be larger or equal to the value of A_{m+1}, therefore the value of the fraction $\frac{A_{m+1}}{A_m} \in (0, 1]$. The fraction $\frac{A_{m+1}}{A_m}$ is the conditional proba-

$m = 2$	Without added length		With added length	
	$r = 0.25 \times Std$	$r = 1 \times Std$	$r = 0.25 \times Std$	$r = 1 \times Std$
time series without differentiation				
time series	567 (5.85%)	535 (5.52%)	305 (3.15%)	326 (3.36%)
simple moving average	446 (4.60%)	412 (4.25%)	252 (2.60%)	242 (2.50%)
modified moving average	364 (3.76%)	397 (4.10%)	195 (2.01%)	241 (2.49%)
exponential moving average	400 (4.13%)	310 (3.20%)	227 (2.34%)	175 (1.81%)
once differentiated time series				
time series	*	288 (2.97%)	*	172 (1.77%)
simple moving average	*	460 (4.75%)	*	278 (2.87%)
modified moving average	551 (5.68%)	272 (2.81%)	273 (2.82%)	172 (1.77%)
exponential moving average	550 (5.67%)	250 (2.58%)	303 (3.13%)	159 (1.64%)
twice differentiated time series				
time series	*	288 (2.97%)	*	171 (1.76%)
simple moving average	*	459 (4.74%)	*	270 (2.79%)
modified moving average	548 (5.65%)	268 (2.76%)	275 (2.84%)	170 (1.75%)
exponential moving average	567 (5.85%)	250 (2.58%)	301 (3.11%)	160 (1.65%)

Table 18: The sample entropy descriptors misclassification error obtained for 553-dimensional vectors (each dimension corresponds with one scale obtained from AAIndex database). * indicates the missing results that were impossible to obtain due to the sample entropy algorithm limitation.

bility that two template vectors of the $m+1$ length have the distance $d[x_m(i), x_m(j)]$ between each other lower than tolerance r given that two template vectors of the m length have also the distance $d[x_m(i), x_m(j)]$ between each other lower than r.

$SampEn$ will always be greater or equal 0. Smaller value of $SampEn$ means more self similarity and less noise in examined time series. The value of sample entropy does not depend on the length of the examined time series [85] (see Figure 25 C and D).

Tables 18 and 19 show the misclassification error results for sample entropy descriptor calculated for the protein-derived time series. For some of the sample entropy descriptor vectors results were impossible to obtain due to the sample entropy algorithm limitation. For the value of parameters $m = 2$ and $r = 0.25 \times Std$ the sample entropy set of descriptors (that were possible to obtain) give much larger misclassification error than corresponding approximate entropy sets of descriptors (see Table 16 and Table 18). We can speculate that the approximate entropy length dependence and sample entropy length independence (see Figure 25 A and C) may have contributed to this result. For $m = 2$ and $r = 1 \times Std$ in case of most protein-derived time series the opposite effect (the approximate entropy larger misclassification error) was observed. For these parameters, again opposite to the case of $r = 0.25 \times Std$ and apart from simple moving average time series, single and double differentiation improve the misclassification error result compared to the undifferentiated time series.

For $m = 3$ and $r = 1 \times Std$ in most cases (except undifferentiated simple and modified moving average time series) the approximate entropy sets of descriptors give lower misclassifi-

$m = 3$; $r = 1 \times Std$	Without added length	With added length
time series without differentiation		
time series	534 (5.51%)	312 (3.22%)
simple moving average	434 (4.48%)	248 (2.56%)
modified moving average	415 (4.28%)	230 (2.37%)
exponential moving average	322 (3.32%)	203 (2.09%)
once differentiated time series		
time series	309 (3.19%)	190 (1.96%)
simple moving average	516 (5.32%)	310 (3.20%)
modified moving average	305 (3.15%)	192 (1.98%)
exponential moving average	306 (3.16%)	183 (1.89%)
twice differentiated time series		
time series	314 (3.24%)	184 (1.90%)
simple moving average	507 (5.23%)	309 (3.19%)
modified moving average	305 (3.15%)	185 (1.91%)
exponential moving average	304 (3.14%)	186 (1.92%)

Table 19: The sample entropy descriptors misclassification error obtained for 553-dimensional vectors (each dimension corresponds with one scale obtained from AAIndex database). Results for the value of parameter $r = Std$. (Results for $r = 0.25 \times Std$ were impossible to obtain due to the sample entropy algorithm limitation).

cation error the corresponding sample entropy sets of descriptors (see Table 16 and Table 19).

Results for $m = 3$ and $r = 0.25 \times Std$ were impossible to obtain due to the sample entropy algorithm limitation.

Because, as mentioned before the sample entropy, unlike the approximate is an unbiased statistic, with values not depending on the time series length, we decided to calculate misclassification error of the 553-dimensional sets of sample entropy descriptors, with the added vector of protein lengths as the 554th dimension. As the result, we observe the increase of protein classification accuracy, Apart from that in most cases (with the exception of single and double differentiated exponential and modified moving average for values of $m = 2$ and $r = 0.25 \times Std$) the results of combining the sample entropy with protein length outperform the results of the approximate entropy for the corresponding descriptor vectors.

5.4 Linear regression application for the sample entropy

Let's calculate the value of A_m for $m \in 1, 2, 3, ..., n$, according to the definition given for the sample entropy calculation. X axis represent $\{A_1, A_2, A_3, A_4, ...A_{n-1}\}$ and Y axis represent $\{A_2, A_3, A_4, A_5, ...A_n\}$. For finite time series, the maximum value of n depends on the examined time series, because A_n has to be greater than 0. For infinite time series, there could be theoretically no maximum possible value of n.

If $Y = aX + b$ (see Figure 26 C and D) then $SampEn$ can be redefined as:

$$SampEn = -\log\frac{aX+b}{X}$$

Let's consider the limit value of the sample entropy in case of infinite time series:

$$\lim_{X\to\infty} SampEn = -\lim_{X\to\infty}\log\frac{aX+b}{X} = -\lim_{X\to\infty}\log(a+\frac{b}{X}) = -\log(a)$$

Therefore we define the **Sample-entropy-based-descriptor** as the negative logarithm of a (a being the slope of the linear regression fit line and can be understood as the limit of $SampEn$).

When we compare the performance of the sample-entropy-linear-descriptor (for $r = 0.25\times Std$, Table 20) with the traditionally calculated sample entropy descriptors (for $r = 0.25\times Std$ and $m = 2$, in cases where was possible to obtain, Table 18) we see that using linear regression is justified by definite decrease of the misclassification error for single and double differentiated protein-derived time series. Decrease can be also seen for all three types of moving average for undifferentiated time series, though improvement is less pronounced. In case of the non-averaged non-differentiated protein-derived time series sample-entropy-linear-descriptors performs slightly worse than the traditionally calculated one.

	Without added length		With added length	
	$r = 0.25 \times Std$	$r = 1 \times Std$	$r = 0.25 \times Std$	$r = 1 \times Std$
time series without differentiation				
time series	581 (5.99%)	605 (6.24%)	297 (3.06%)	362 (3.73%)
simple moving average	371 (3.83%)	423 (4.36%)	204 (2.10%)	241 (2.49%)
modified moving average	348 (3.59%)	401 (4.14%)	203 (2.09%)	261 (2.69%)
exponential moving average	367 (3.79%)	310 (3.20%)	204 (2.10%)	176 (1.82%)
differentiated time series				
time series	420 (4.33%)	318 (3.28%)	232 (2.39%)	166 (1.71%)
simple moving average	569 (5.87%)	440 (4.54%)	319 (3.29%)	289 (2.98%)
modified moving average	355 (3.66%)	228 (2.35%)	181 (1.87%)	153 (1.58%)
exponential moving average	362 (3.73%)	232 (2.39%)	185 (1.91%)	155 (1.60%)
twice differentiated time series				
time series	415 (4.28%)	305 (3.15%)	223 (2.30%)	178 (1.84%)
simple moving average	603 (6.22%)	478 (4.93%)	333 (3.44%)	301 (3.11%)
modified moving average	358 (3.69%)	224 (2.31%)	193 (1.99%)	155 (1.60%)
exponential moving average	363 (3.74%)	226 (2.33%)	191 (1.97%)	154 (1.59%)

Table 20: The sample-entropy-based-descriptors misclassification error defined as the number of misclassified proteins for 553-dimensional vectors (each dimension corresponds with one scale obtained from AAIndex database).

For sample-entropy-linear-descriptors calculated with $r = 1 \times Std$ and the traditionally calculated sample entropy descriptors with $r = 1\times Std$ and $m = 2$, we do not see a consistent pattern. For some protein-derived time series the former gives better discrimination between

protein groups, for other the latter. The differences are not extreme in any of the examined time series (see Table 18 and Table 20). The sample-entropy-linear-descriptors improve misclassification error results (with two exceptions: non differentiated and single differentiated non-averaged time series) when compared with the traditionally calculated sample entropy descriptors calculated for $r = 1 \times Std$ and $m = 3$ (Table 19 vs. Table 20).

Similarly like in case of the sample entropy, we decided to calculate misclassification errors of the 553-dimensional sets of sample entropy descriptors, with the added vector of protein lengths as the 554 dimension. As a result, we observe the definite increase of protein classification accuracy in all examined cases.

Using the sample-entropy-linear-descriptors is not beneficial while compared to using the approximate-entropy-based-descriptor (Table 17 vs. Table 20). On the contrary, the latter proves to be more effective in the task of discrimination between protein groups. However combination of the sample-entropy-linear-descriptors with the vector of protein lengths improves results beyond the level achieved by the approximate-entropy-based-descriptor.

6 Power spectra analysis

Mean (median) frequency of the protein-derived time series, was calculated as the weighted mean (median) of the frequency of the signal, where the weights are the values of the power spectrum. Power spectra were estimated based on the discrete Fourier transform (f_s) of the signal:

$$f_k = \sum_{i=0}^{N-1} P_i e^{-\frac{2\pi i}{N} kn}$$

The Fast Fourier Transform (FFA) algorithm inbuilt in Matlab software [52] was used with basic rectangular window of the length of the signal. Two approaches to calculate mean/median frequency were used:

- Type 1: number of points used to FFT calculation was equal to the maximum of the 256 and next power of 2 greater than the length of the protein
- Type 2: number of points used to FFT calculation was equal to the length of the protein

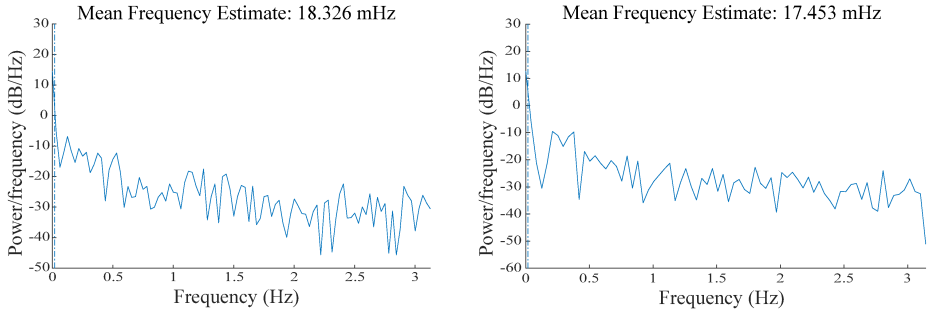

Figure 28: The example of the power spectrum and the mean frequency of the time series (modified moving average time series based on one of the AAIndex scales).

Mean (median) period was calculated as the inverse of mean (median) frequency:

$$period = \frac{1}{frequency}$$

For both frequency and period, all 553-dimensional descriptor vectors (computed for the non-differentiated non-averaged time series and for the non-differentiated moving average time series) yield a significantly lower misclassification error than their single or doubly differentiated counterparts (see tables 21 and 22). The highest misclassification error calculated for the descriptors derived from time series without differentiation is the one for the Type 1 median period for non-averaged time series, 7.05%. This is still lower than the lowest misclassification error calculated for the descriptors derived from the differentiated time series

	Mean Frequency		Median Frequency	
	Type I	Type II	Type I	Type II
time series without differentiation				
time series	251 (2.59%)	530 (5.47%)	494 (5.10%)	436 (4.50%)
simple moving average	175 (1.81%)	380 (3.92%)	499 (5.15%)	318 (3.28%)
modified moving average	232 (2.39%)	468 (4.83%)	534 (5.51%)	324 (3.34%)
exponential moving average	199 (2.05%)	418 (4.31%)	514 (5.30%)	335 (3.46%)
once differentiated time series				
time series	870 (8.98%)	864 (8.91%)	1024 (10.56%)	1028 (10.61%)
simple moving average	870 (8.98%)	864 (8.91%)	1648 (17.00%)	1663 (17.16%)
modified moving average	796 (8.21%)	824 (8.50%)	1007 (10.39%)	1005 (10.37%)
exponential moving average	838 (8.65%)	845 (8.72%)	1005 (10.37%)	1007 (10.39%)
twice differentiated time series				
time series	905 (9.34%)	924 (9.53%)	1125 (11.61%)	1137 (11.73%)
simple moving average	1136 (11.72%)	1143 (11.79%)	1691 (17.45%)	1692 (17.46%)
modified moving average	908 (9.37%)	910 (9.39%)	1063 (10.97%)	1071 (11.05%)
exponential moving average	908 (9.37%)	906 (9.35%)	1066 (11.00%)	1099 (11.34%)

Table 21: The mean and median frequency descriptors misclassification error obtained for 553-dimensional vectors (each dimension corresponds with one scale obtain from AAIndex database).

.

	Mean Period		Median Period	
	Type I	Type II	Type I	Type II
time series without differentiation				
time series	253 (2.61%)	571 (5.89%)	683 (7.05%)	649 (6.70%)
simple moving average	189 (1.95%)	413 (4.26%)	500 (5.16%)	370 (3.82%)
modified moving average	212 (2.19%)	462 (4.77%)	514 (5.30%)	391 (4.03%)
exponential moving average	225 (2.32%)	406 (4.19%)	558 (5.76%)	398 (4.11%)
once differentiated time series				
time series	928 (9.57%)	929 (9.58%)	1047 (10.80%)	1073 (11.07%)
simple moving average	969 (10.00%)	985 (10.16%)	1449 (14.95%)	1463 (15.09%)
modified moving average	881 (9.09%)	897 (9.25%)	1110 (11.45%)	1066 (11.00%)
exponential moving average	935 (9.65%)	915 (9.44%)	1099 (11.34%)	1098 (11.33%)
twice differentiated time series				
time series	967 (9.98%)	971 (10.02%)	1169 (12.06%)	1166 (12.03%)
simple moving average	1206 (12.44%)	1213 (12.51%)	1614 (16.65%)	1618 (16.69%)
modified moving average	969 (10.00%)	968 (9.99%)	1122 (11.58%)	1099 (11.34%)
exponential moving average	967 (9.98%)	956 (9.86%)	1110 (11.45%)	1123 (11.59%)

Table 22: The mean and median periods descriptors misclassification error obtained for 553-dimensional vectors (each dimension corresponds with one scale obtain from AAIndex database)

8.21% (Type 1 mean frequency, once differentiated modified moving average time series). Even though similar statement cannot be given, when comparing maximum error observed for the once differentiated time series with minimum error observed for the double differentiated time series, still there is certain regularity. With one to one comparison, in all cases, the misclassification error calculated for the once differentiated time series is lower than the one for its double differentiated counterparts.

For the single and double differentiated time series the misclassification error is lower when calculated for frequency descriptor than when calculated for its counterpart: the period descriptor in almost all cases. The exception is median calculated for simple moving average time series. In those exceptional cases median frequency vs median period errors are: for Type 1: 17.00% vs. 14.95% in case of single and 17.45% vs. 16.65% in case of double differentiation and for Type 2: 17.16% vs. 15.09% in case of single and 17.46% vs. 16.69% in case of double differentiation. In typical cases, where the misclassification error is lower for frequency descriptors than for its period counterpart, the maximum difference between corresponding errors (1.25%) can be observed for Type II Mean descriptors calculated for the once differentiated simple moving average time series: 8.91% vs. 10.16%

As a summary we can state, that for the differentiated time series period and frequency descriptors, are outperformed, not only by the same descriptors calculated for the non differentiated time series, but also by the many of descriptors analysed in previous part of this thesis. For example the maximum misclassification error observed for the approximate entropy descriptor was 6.84% and for sample entropy descriptor 5.85% (see Chapter 3.2).

For the non-differentiated time series, the largest misclassification error difference for the corresponding frequency and period descriptors can be observed in case of Type 1 (5.10% vs. 7.05%) and Type 2 (4.50% vs. 6.70%) medians calculated for the non-averaged time series. For the rest of the cases the differences are much lower and do not exceed 0.69%.

For both frequency and period, the best results are obtained for the Type 1 mean calculated for non-averaged and all three types of averaged time series. The largest misclassification error in that category is 2.61% (Type 1 mean period, non-averaged), is lower than the misclassification error calculated for all Type 2 mean and Type 1 and 2 median descriptor vectors.

The lowest misclassification error for all calculated frequency and period descriptors 1.81% was obtained in case of Type 1 mean frequency non-differentiated simple moving average.

7 Lyapunov exponent – descriptor based on the chaos theory

In the chaos theory the Lyapunov exponent (λ) is a quantitative way to measure the differences between two paths taken by an object in motion, that both start with almost identical initial conditions, and after time t are separated by the distance $d(t)$ [87].

$$e^{\lambda t} \propto d(t)$$

The concept, initially applied in dynamical systems calculations, was later adapted for time series analysis, with the algorithm described below [87]. It is used to compare time series with itself in order to find a level of self-similarity.

As in case of the approximate and sample entropy calculations, let's consider time series $\{x_1, x_2, x_3, ..., x_N\}$ of a length of N. The template vector of a length m is define as $x_m(i) = \{x_i, x_{i+1}, ..., x_{i+m-1}\}$. We calculate the Euclidean distance between each pairs of template vectors:

$$d[x_m(i), x_m(j)] = \sqrt{\left(\sum_{k=1}^{m} u(k) - \sum_{k=1}^{m} u(k)\right)^2}$$

where $u(k)$ and $v(k)$ are components of vectors $x_m(i)$ and $x_m(j)$ respectively; $k \in \{1, 2, .., m\}$.

For each template vector $x_m(i)$, we find its **nearest neighbour** $x'_m(i) = x_m(j)$ such us the distance d between them is minimal:

$$d[x_m(i), x'_m(i)] = \min_d d[x_m(i), x_m(j)],$$

given that the proximity between i^{th} and j^{th} entries of time series has to be greater than the mean (or median) frequency of the time series.

The nearest neighbour is searched in both directions. If two values $d[x_m(i), x_m(j_1)]$ and $d[x_m(i), x_m(j_2)]$ are equal, the template vector that lays in closer proximity to the beginning of sequence (N-terminus of protein) is chosen as the nearest neighbour. If the mean/median frequency is larger than $N + m - 1$, than the first template vector, becomes the nearest neighbour for all template vectors.

When possible, for pairs containing the template vectors $x_m(i)$ and its nearest neighbour $x'_m(i)$ we find the n-dimensional vectors of the Euclidean distances d (the trajectory vectors)

$$dist^n_m(i) = \{d[x_m(i+1), x'_m(i+1)], d[x_m(i+2), x'_m(i+2)], ..., d[x_m(i+n), x'_m(i+n)]\}$$

Neither template vectors $x_m(i)$ nor its nearest neighbour $x'_m(i) = x_m(j)$ can be one of the n last template vectors: $(i < (N - m + 1) - n)$ and $j < (N - m + 1) - n)$.

We calculate the the average values of the logarithm of the distance between template vectors

$x_m(i+t)$ and $x'_m(i+t)$, for every $t \in \{1, 2, ..., n\}$:

$$L_m(t) = \frac{1}{F} \sum_{i=1}^{F} \log(d[x_m(i+t), x'_m(i+t)])$$

F is the number of template vectors, for which the trajectory vectors were possible to calculate.

The value of **Lyapunov exponent** can be estimated by the slope of the linear regression line fitting points $\{t, L_m(t)\}$ (Figure 29).

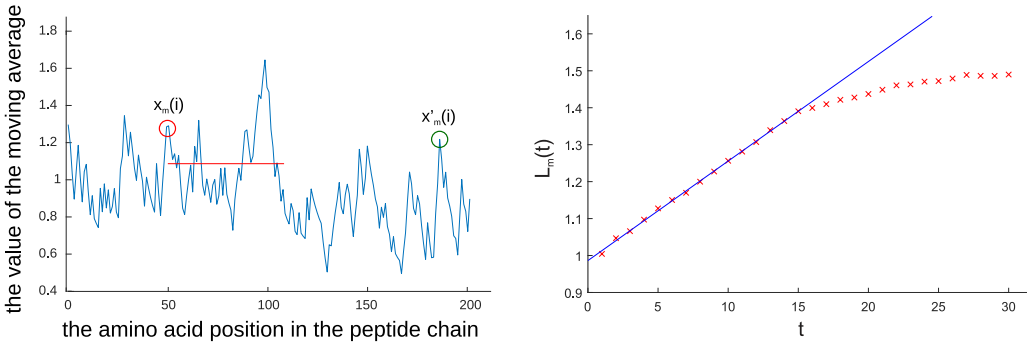

Figure 29: The graphical representation of the Lyapunov exponent calculation. Left panel show the position template vector $x_m(i)$ and its nearest neighbour $x'_m(i)$ in the time series. The proximity between them have to be greater than the mean (or median) frequency of the time series, represented by the red line. In the left panel, the y axis represent $L_m(t)$ the average value of the logarithm of the distance between template vectors $x_m(i+t)$ and $x'_m(i+t)$ and the x axis represents the proximity t between template vectors $x_m(i)$ and $x_m(i+t)$. The linear regression line fitting points $\{t, L_m(t)\}$ is marked blue.

For each protein-derived time series, the Lyapunov exponent values were calculated with the Type I and Type II mean and median frequency values calculated with the protocols described in the Chapter 6, providing 4 separate 553-dimensional persistent exponent vectors for each of the twelve protein-derived time series type (Table 23), with the Matlab code [88]. We have used value of $n = 30$, but for the linear regression calculations we only considered $t \in \{1, 15\}$. The values that protein derived time series can take are limited by the maximum and minimum AAIndex scales values, therefore the differences between "trajectories" will stop increasing at some point (Figure 29, right panel).

For the non differentiated time series, when the values of Type I and Type II mean period and Type II median period were used, to calculate the Lyapunov exponent values, the non-averaged time series based descriptor performs better in the task of discriminating between the protein families than any of moving average ones. This is no longer true for the Lyapunov exponent descriptors calculated with Type I median period, however the differences between non-averaged and moving averaged time series are, in this case, much less clear.

For Lyapunov exponents calculated with median period Type I and II, and mean period

Lyapunov based on	Mean Period		Median Period	
	Type I	Type II	Type I	Type II
time series without differentiation				
time series	472 (4.87%)	258 (2.66%)	355 (3.66%)	217 (2.24%)
simple moving average	751 (7.75%)	410 (4.23%)	345 (3.56%)	423 (4.36%)
modified moving average	689 (7.11%)	424 (4.37%)	394 (4.06%)	404 (4.17%)
exponential moving average	635 (6.55%)	396 (4.09%)	332 (3.43%)	386 (3.98%)
once differentiated time series				
time series*	512 (5.28%)	512 (5.28%)	512 (5.28%)	512 (5.28%)
simple moving average	744 (7.68%)	741 (7.64%)	743 (7.67%)	740 (7.63%)
modified moving average	495 (5.11%)	494 (5.10%)	493 (5.09%)	495 (5.11%)
exponential moving average	518 (5.34%)	517 (5.33%)	518 (5.34%)	518 (5.34%)
twice differentiated time series				
time series*	596 (6.15%)	596 (6.15%)	596 (6.15%)	596 (6.15%)
simple moving average*	856 (8.83%)	856 (8.83%)	856 (8.83%)	856 (8.83%)
modified moving average*	593 (6.12%)	593 (6.12%)	593 (6.12%)	593 (6.12%)
exponential moving average*	620 (6.40%)	620 (6.40%)	620 (6.40%)	620 (6.40%)

Table 23: The Lyapunov exponent descriptor vectors misclassification error obtained for 553-dimensional sets (each dimension corresponds with one scale obtain from AAIndex database). *the value of mean/median period used for the Lyapunov exponents calculations are so low for all protein-derived time series in the examined set, that the choice of the nearest neighbour is the same irrespectively of the chosen period calculation protocol.

Type II, the single and double differentiated time series provides greater misclassification error, than corresponding non differentiated ones. The same is not always true in case of Lyapunov exponent descriptors calculated with Type I mean period. What is worth noting however, is the fact, that in this case, all descriptors derived based on the time series without differentiation give much higher misclassification error than all other Lyapunov exponent descriptors derived based on the same time series.

8 Statistical Mechanics Based Descriptors

8.1 Persistence exponent: the first passage probability distribution

Persistence exponent analysis is a concept derived from statistical physics [89, 90] and has been successfully applied in many research areas, such as modelling interactions between lymphocyte T and antigen-presenting cells [91, 92], population dynamic models [93], C-G distribution in DNA [94], protein sequence analysis [82], as well as enzyme kinetics [95].

The first passage probability, of a discrete time series can be simply define as the probability that the value of set time series reaches predefined threshold after k time intervals [82]. In other words it determines how long on average, starting from initial moment, it takes for a process to reach or return to the predefined state.

In our research we have used the arithmetical mean of the examined protein-derived time series as a threshold value (the red line in the Figure 30 left panel). Based on that value, the time series (blue line) is divided into segments. Then the numbers of positive (with values above the mean) and negative (values below the mean) segments of consecutive length $(1, 2, 3,$ etc) are calculated, providing set of points on a plane (x_i, y_i). x_i is the logarithm of the positive (or negative) segment length and y_i is the logarithm of the non-zero number of segments of such length (Figure 30 left panel). Then the first degree polynomial $y = ax + b$ is fitted to the data, and the **Persistence Exponent** is understood as the slope of the linear regression line [82].

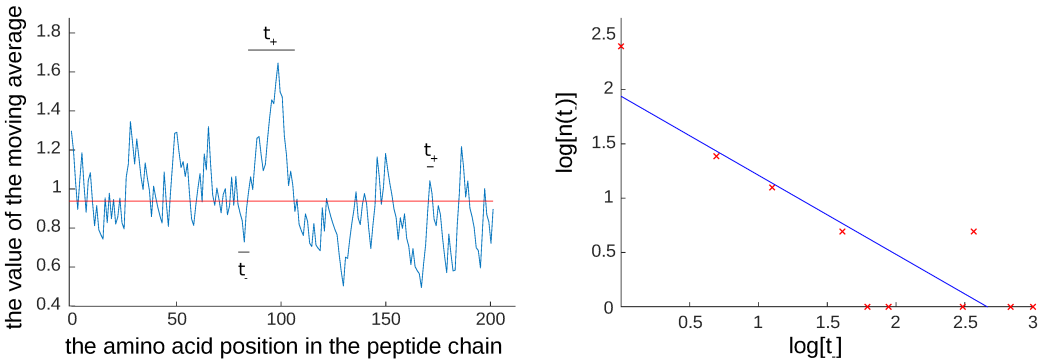

Figure 30: The graphical illustration of the Persistence exponent calculation algorithm. t_+ marks examples of positive segments, and t_- the negative one. $n(t_-)$ is the number of negative segments of (t_-) length.

For each protein-derived time series, the positive and negative persistence exponent values were calculated, and than averaged, providing 3 separate 553-dimensional persistent exponent vectors for each of the twelve protein-derived time series type (Table 24).

For the persistence exponent descriptor vectors the lowest misclassification error is obtained for the exponential moving average time series without differentiation: 2292 misclassification protein (3.01%) for the positive exponent, 290 (2.99%) for the negative, and 301

Persistence exponent	Positive	Negative	Average
time series without differentiation			
time series	554 (5.72%)	543 (5.60%)	922 (9.15%)
simple moving average	610 (6.29%)	587 (6.06%)	556 (5.74%)
modified moving average	391 (4.03%)	425 (4.38%)	417 (4.30%)
exponential moving average	292 (3.01%)	290 (2.99%)	301 (3.11%)
once differentiated time series			
time series	1017 (10.49%)	1084 (11.18%)	947 (9.77%)
simple moving average	586 (6.05%)	565 (5.83%)	836 (8.62%)
modified moving average	400 (4.13%)	323 (3.33%)	576 (5.94%)
exponential moving average	440 (4.54%)	353 (3.64%)	548 (5.65%)
twice differentiated time series			
time series	639 (6.59%)	711 (7.34%)	869 (8.97%)
simple moving average	1121 (11.57%)	1115 (11.50%)	1203 (12.41%)
modified moving average	549 (5.66%)	610 (6.29%)	770 (7.94%)
exponential moving average	520 (5.36%)	596 (6.15%)	679 (7.01%)

Table 24: The persistence exponent descriptors misclassification error obtained for 553-dimensional sets (each dimension corresponds with one scale obtain from AAIndex database)

(3.11%) for the average one. In all cases, for non differentiated and both, once and twice differentiated time series, the exponential and modified moving average based descriptors outperform the descriptors calculated for simple moving average and for non-averaged time series. There is no obvious advantage of using positive or negative persistence exponent descriptors, though the negative slightly outperforms the positive one more often than the other way around. In almost all cases, both positive and negative exponents outperform the average one (with exception of simple moving average without differentiation, and once differentiated non averaged time series).

It is not beneficial to use differentiated time series, for the persistence exponent analysis.

Descriptors calculated for modified and exponential moving average, where the values of each moving average entry depends on all previous entries, provides better division between protein families, than descriptors calculated for simple moving average.

In three cases, the descriptor vectors calculated for the once and twice differentiated time series slightly improve the misclassification error results, while compared to time series without differentiation. It happens for the positive and negative persistence exponent calculated for once differentiated simple moving average and for average persistence exponent in case of twice differentiated non averaged time series. In all other cases, differentiation increase the obtained misclassification error. Therefore we can conclude that the persistence exponent, in general, do not outperform the results obtained with the information theory based descriptors, calculated for the protein derived time series.

8.2 Sum total of the pick

The relatively short length of the examined protein-derived time series has a consequence of limited number of points used for the persistence exponent linear regression calculation. Therefore we have proposed an alternative routines for calculating statistical descriptors, with algorithm that shares some steps with the previous, and a collective name **Sum total of the pick**.

We start by centralizing the values of the time series with respect to its arithmetic mean. Then, exactly like in case of the persistence exponent, we consider the segments below and above the mean. This time however sums of the values of positives and negative segments is also calculated, not just the segments lengths (see Figure 30). Firstly we look at the segment with the maximum absolute value of the sum, separately in case of positives segments and the negative ones. If there are two positive (or two negative) segments with the same sums, we consider the one, that appears earlier in the protein-derived time series (the one that is closer to the N-terminus of the examined protein sequence). Then we divide the chosen segment by its length. The value calculated in this way from now on will be called the **Normalized-maximum-sum**.

Second type of the Sum total of the pick descriptors: **Normalised-grand-total** is calculated by summing up all values above (below) the mean. Then absolute value of that sum is taken and divided by the sum of the lengths of all positive and all negative segments.

Both types of the sum total of the pick descriptor values were calculated separately for each scale, therefore for each protein sequences the 553-dimensional vectors were constructed. The misclassification error obtained for the positive and negative Normalized Maximum Sum vectors can be found in the Table 25 and for the positive and negative Normalised Grand Total Sum in the Table 26.

The positive and negative normalized-maximum-sum descriptor vectors provides the minimal misclassification error for the non averaged time series without differentiation: 6.09% (590 misclassified protein) and and 6.24% (590 misclassified protein). These values are substantially greater than minimum ones obtained for the positive and persistence exponent descriptor vectors (3.01% and 290 2.99%, see Table 24). In fact, with exception of the twice differentiated simple moving average time series, all normalized-maximum-sum descriptors perform worse than corresponding persistence exponent descriptors. And the mentioned exception provides misclassification error above 11% for all normalized-maximum-sum and persistence exponent descriptors.

The normalised-grand-total descriptor vectors give smaller misclassification errors for the descriptors calculated based on the double differentiated time series than corresponding ones calculated for once differentiated time series, or time series without differentiation. What is more, the best results, meaning the lowest misclassification error, for the normalised-grand-total descriptor vectors, are obtained, in case of positive descriptor for modified moving

	Normalized-maximum-sum	
	positive	negative
time series without differentiation		
time series	590 (6.09%)	605 (6.24%)
simple moving average	639 (6.59%)	613 (6.32%)
modified moving average	628 (6.48%)	585 (6.08%)
exponential moving average	627 (6.47%)	557 (5.75%)
once differentiated time series		
time series	1170 (12.07%)	1171 (12.08%)
simple moving average	927 (9.56%)	940 (9.68%)
modified moving average	797 (8.22%)	779 (8.04%)
exponential moving average	865 (8.92%)	874 (9.02%)
twice differentiated time series		
time series	670 (6.91%)	650 (6.71%)
simple moving average	1079 (11.13%)	1085 (11.19%)
modified moving average	638 (6.58%)	626 (6.46%)
exponential moving average	619 (6.38%)	637 (6.57%)

Table 25: The misclassification error obtained with Linear Discriminant Analysis classifier for the 553-dimensional normalized-maximum-sum descriptor vectors. Each dimension corresponds with one scale obtain from AAIndex database.

	Normalised-grand-total	
	positive	negative
time series without differentiation		
time series	1904 (19.64%)	1943 (19.95%)
simple moving average	394 (4.06%)	360 (3.71%)
modified moving average	379 (3.91%)	401 (4.14%)
exponential moving average	309 (3.19%)	321 (3.31%)
once differentiated time series		
differentiated time series	643 (6.63%)	685 (7.07%)
simple moving average	419 (4.32%)	406 (4.19%)
modified moving average	382 (3.94%)	345 (3.56%)
exponential moving average	327 (3.37%)	303 (3.13%)
twice differentiated time series		
time series	234 (2.41%)	242 (2.50%)
simple moving average	356 (3.67%)	372 (3.84%)
modified moving average	231 (2.38%)	236 (2.43%)
exponential moving average	244 (2.52%)	224 (2.31%)

Table 26: The misclassification error obtained with Linear Discriminant Analysis classifier for 553-dimensional vectors of the normalised-grand-total descriptors. Each dimension corresponds with one scale obtain from AAIndex database.

average: 2.38% (231 misclassified proteins), and for exponential moving average 2.31% (224 misclassified protein) in the case of the negative normalised-grand-total.

Therefore we can conclude that persistence exponent descriptor is more efficient in protein sequence classification than normalized-maximum-sum descriptor. However, when we

compare the performance of the normalised-grand-total with persistence exponent, we cannot derive a similar conclusion. Not only normalised-grand-sum performs better than normalised-maximum-sum, in all cases except non averaged time series without differentiation, it also, in most cases, outperforms the corresponding persistence exponent (Table 24 vs Table 26).

8.3 Hurst exponent

Hurst exponent H provides quantified information about the persistence of the examined time series. Its value for the random walk is 0.5, and t allows us to distinguish between ones that have a tendency to fluctuate between small and large values ($H < 0.5$) and the ones in which the large values are usually followed by a succession of large values, and small values by a succession of small ones ($H > 0.5$) [96]. Hurst exponent analysis has been successfully applied, among others, to protein sequence analysis [97] and logistic map model for DNA sequence [98].

The Hurst exponent calculations algorithm using the re-scaled range [R/S] [99] method is presented below.

Let's divide time series $\{x_1, x_2, x_3, ..., x_N\}$ into a k non overlapping segments of approximately equal lengths $n_k \approx \frac{N}{k}$:

$$\{x_1^1, x_2^1, x_3^1, ..., x_{n_1}^1\}$$
$$\{x_1^2, x_2^2, x_3^2, ..., x_{n_2}^2\}$$
$$...$$
$$\{x_1^j, x_2^j, x_3^j, ..., x_{n_j}^2\}$$
$$...$$
$$\{x_1^k, x_2^k, x_3^k, ..., x_{n_k=N}^k\}$$

For all segments mean M and standard deviation S are obtained ($j \in \{1, 2, .., k\}$).

$$M_k(j) = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i^j$$
$$S_k(j) = \sqrt{\frac{1}{n_j} \sum_{i=1}^{n_j} (x_i^j - M_k(j))^2}$$

The values of each segment are centralised with respect to the segment arithmetic mean

$$y_i^j = x_i^j - M_k^j,$$

and the cumulative deviate series for each segment is calculated

$$z_i^j = \sum_{l=1}^{i} y_l^j.$$

the range $R_k(j)$ is defined as

$$R_k(j) = max(z_i^j) - min(z_i^j),$$

divided by standard deviation $S_k(j)$ and averaged over all k segment.

$$\frac{R(n)}{S(n)} = \frac{1}{k}\sum_{i=1}^{k}\frac{R_k(j)}{S_k(j)}$$

Then the whole process is repeated for the next value of k, $k \in \{1, 2, 3, ..., 10\}$ and $n \in \{N, \frac{N}{2}, \frac{N}{3}, ..., \frac{N}{10}\}$

The **Hurst Exponent** H is defined as:

$$E\left[\frac{R}{S}\right] = Cn^H,$$

where $n \to \infty$. Therefore, it can be understood as the slope a of the linear regression line $y = ax + b$, (Figure 31, blue line) fitting points (x_k, y_k) where $x_k = \log n$, where n=$\frac{N}{k}$ and $y_k = \frac{R(n)}{S(n)}$.

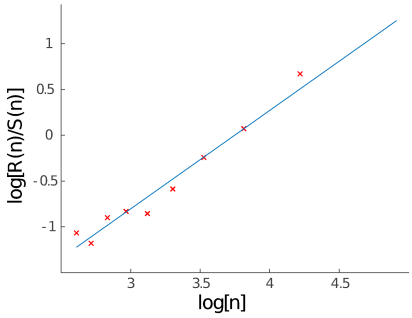

Figure 31: The graphical illustration of the linear line fitting of the Hurst exponent calculations.

The Hurst exponent was calculated separately for each protein-derived time series. The misclassification errors obtained for the 553-dimensional descriptor vectors can be found in the Table 27.

Normalised-grand-total descriptor vectors, outperform most of the corresponding Hurst exponent ones (the only exceptions being: no differentiated and once differentiated none averaged time series), with the best result (meaning the lowest misclassification error) 5.66% (549 misclassified protein), being almost 2.5 times larger than the lowest negative normalised-

	without differentiation	single differentiation	double differentiation
time series	1049 (10.82%)	598 (6.17%)	854 (8.81%)
simple moving average	1998 (20.61%)	1708 (17.62%)	1290 (13.31%)
modified moving average	1213 (12.51%)	796 (8.21%)	585 (6.04%)
exponential moving average	1220 (12.59%)	773 (7.97%)	549 (5.66%)

Table 27: The misclassification error obtained with Linear Discriminant Analysis classifier for 553-dimensional vectors of the Hurst exponent descriptors.

grand-total result: 2.31% (244 misclassified protein, both results obtained for the double differentiated exponential moving average, Table 27 vs Table 26). It also can be seen that the Hurst exponent, does not perform as well as the Persistence exponent, for the majority of cases (see Table 24).

For most protein-derived time series, the value of the Hurst exponent descriptor, calculated with re-scaled range [R/S], is greater than 1 (Table 28). This result suggests a high proportion of non-stationary time series [100] within the examined set, with a single differentiation process significantly decreasing that proportion, and double differentiation decreasing it even more (only the non averaged time series presents partially different behaviour).

	without differentiation	single differentiation	double differentiation
time series	0.001%	17.65%	15.39%
simple moving average	0%	0.17%	8.15%
modified moving average	0%	0.08%	9.82%
exponential moving average	0%	0.37%	8.15%

Table 28: The proportion of smaller than 1 values for Hurst exponent in the (553x9693)-elements sets. (553 id the number of used AAIndex scales being the number of time series calculated for each protein and 9396 is the number of protein in the examined test set).

8.4 Box-counting-algorithm-based descriptors

Fractal dimension D is a measure of self-similarity of a geometric object. While the dimension of a straight line is 1, and the dimension of square is 2, the dimension of object that has fractal properties can lay between those two values. To understand the concept of fractal dimension, firstly we need to divide an object into N identical parts (see Figure 32), and then increase any chosen part by a scaling factor r to make it the size of the original object. Fractal dimension is defined as exponent D such that $r^D = N$.

One of the well established methods to calculate the **Fractal Dimension** of time series is based on the Hurst exponent value between $(0, 1)$, corresponding with the value of the fractal dimension D between $(0, 1)$, as $D = 2 - H$. In case of the examined set of protein-derived time series this method was unsuccessful, therefore we decided to try a different

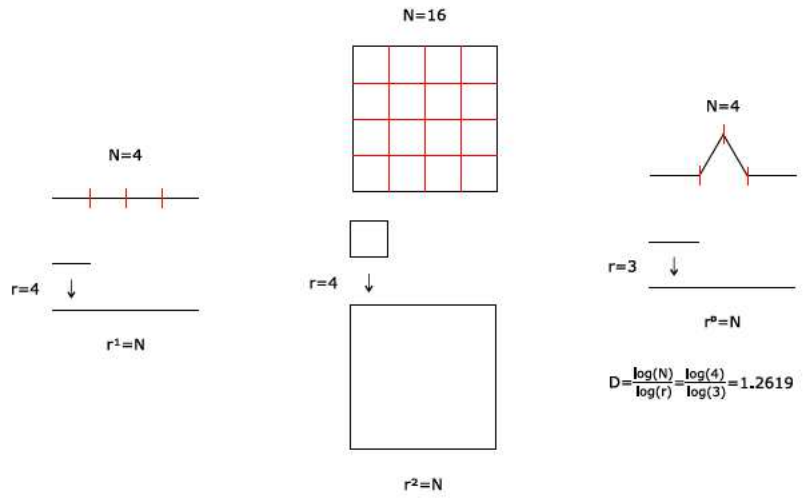

Figure 32: The graphical illustration of the concept of fractal dimension.

approach as "Box counting algorithm for the fractal dimension estimation" or, in short, the **Box Counting Dimension** [101, 102]. This technique was applied, among others, to DNA sequence comparison [103, 104] and phylogenetic analysis [105]. Firstly, a two-dimensional histogram of time series values is created. The graphical representation of that would be to "cover" the time series with boxes of equal sizes r (Figure 33, left panel). Next the number of "boxes" $N(r)$ that contain any data points were calculated (hence the algorithm name).

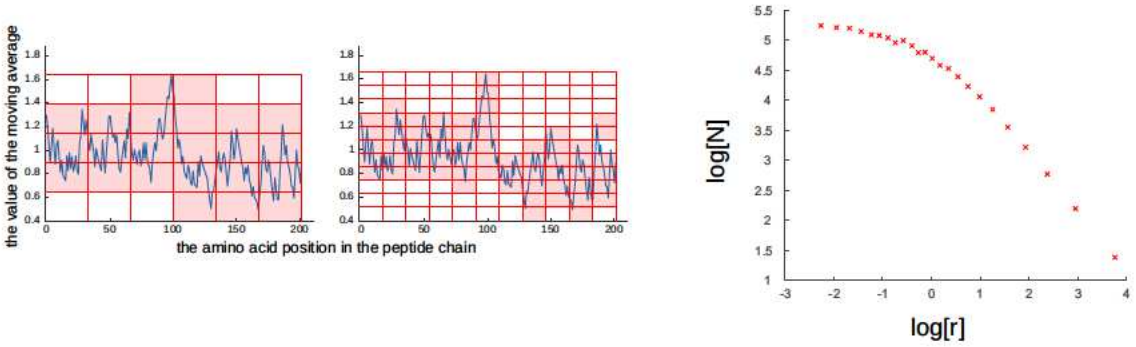

Figure 33: The graphical illustration of the Box-counting-algorithm-based descriptors calculations.

With the decreasing boxes size r:

$$D_{box} = \lim_{t \to 0} \frac{\log N(r)}{\log \frac{1}{r}}$$

By "covering" the examined time series with boxes of a different sizes r_i, the box counting algorithm provides pairs (x_i, y_i), where x_i is the logarithm of the box size r_i and y_i is the logarithm of $N(r_i)$, the number of boxes of r_i size containing non zero data points. The value

	first degree polynomials	second degree polynomials	
	a	$a2$	$a1$
time series without differentiation			
time series	374 (3.86%)	498 (5.14%)	389 (4.01%)
simple moving average	326 (3.36%)	443 (4.57%)	228 (2.35%)
modified moving average	278 (2.87%)	597 (6.16%)	237 (2.45%)
exponential moving average	332 (3.43%)	433 (4.47%)	225 (2.32%)
once differentiated time series			
time series	408 (4.21%)	241 (2.49%)	211 (2.18%)
simple moving average	431 (5.44%)	342 (3.53%)	311 (3.21%)
modified moving average	306 (3.16%)	219 (2.26%)	182 (1.88%)
exponential moving average	356 (3.68%)	209 (2.16%)	211 (2.18%)
twice differentiated time series			
time series	319 (3.29%)	236 (2.43%)	205 (2.11%)
simple moving average	611 (6.30%)	418 (4.31%)	316 (3.26%)
modified moving average	252 (2.60%)	246 (2.54%)	237 (2.45%)
exponential moving average	254 (2.62%)	240 (2.48%)	207 (2.14%)

Table 29: The misclassification error obtained with Linear Discriminant Analysis classifier for 553-dimensional sets of the Fractal dimension like descriptor. Each dimension corresponds to one scale obtain from AAIndex database. Moving average calculations were performed before differentiation.

of the **Box Counting Dimension** D_{box} can be estimated by the negative slope of the linear regression line $y = ax + b$.

$$\log N(r) \propto -D_{box} \log r$$

As can be seen (Figure 33, right panel), for the protein-derived time series, points $(\log r_i, \log N(r_i))$ they do not approximate to a straight line, therefore the box counting algorithm is unable to provide the estimate of fractal dimension of the examined numerical sequence [102]. This observation was repeated for other examined protein-derived time series.

The first $y = ax + b$ and the second degree polynomials $y = a_2x^2 + a_1x + b$ were fitted to the box-counting-algorithm pairs $(\log r_i, \log N(r_i))$, and the coefficients a, a_2 and a_1 become the **box-counting-derived descriptors**. They were calculated separately for each protein-derived time series (Table 29), therefore the 553-dimensional vectors were created. As different proteins have different length, the number of repetition of the algorithm (i), as well as exact box sizes r_i differ for each protein from the examined test set. This modification was necessary to decrease the correlation between the box-counting algorithm-descriptor values and the protein length (Figure 34). This problem should not exist if the protein derived time series had an actual fractal dimension.

The box-counting-algorithm-based second degree polynomial descriptors: $a2$ and $a1$ perform better than the corresponding first degree polynomial descriptor a, when misclassification errors are calculated for all single and double differentiated time series. For all non

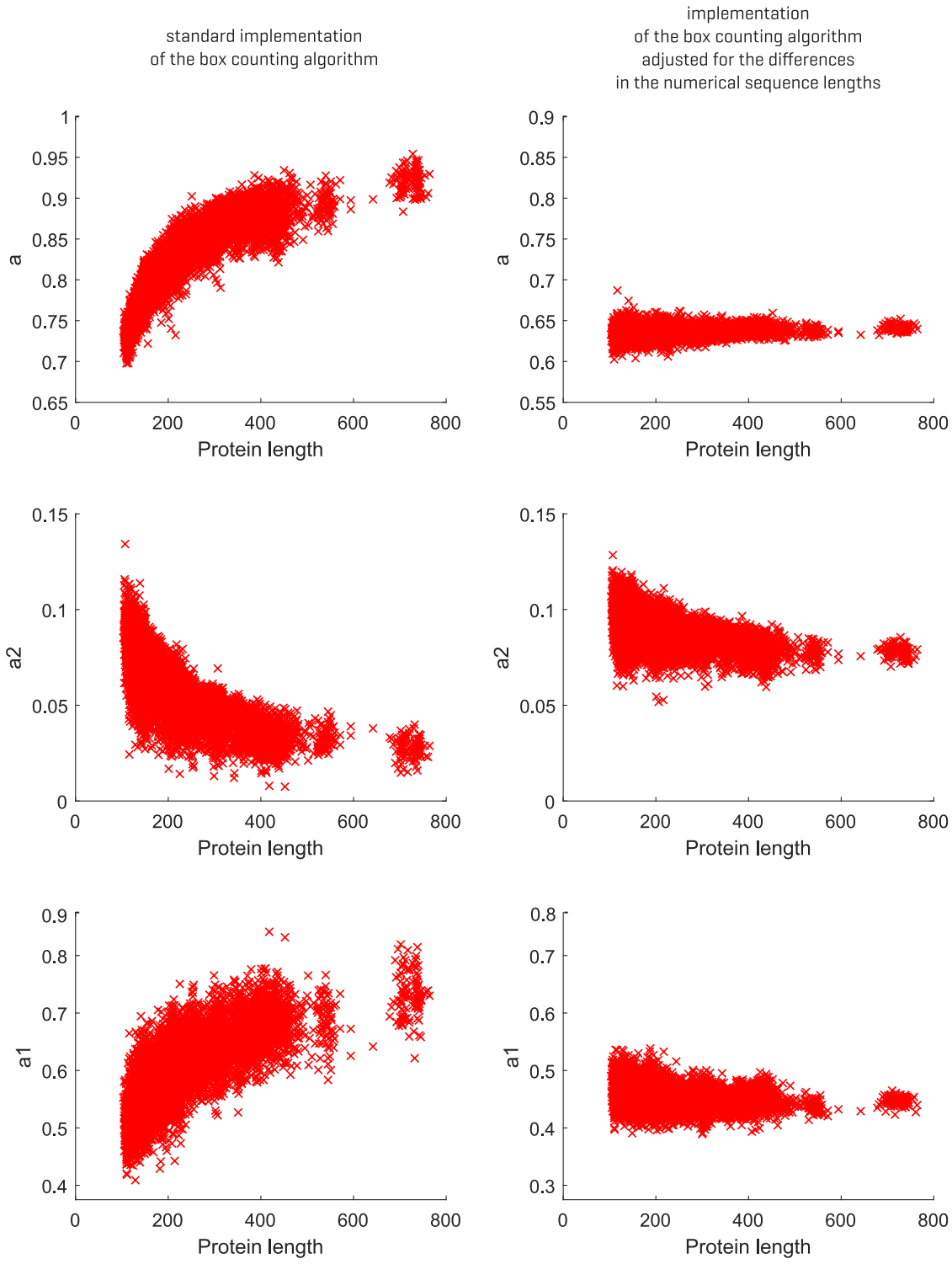

Figure 34: The dependence between box-counting algorithm-descriptor values and the protein length for the standard implementation of the box counting algorithm, and the implementation adjusted for the differences in the numerical sequence lengths.

differentiated moving average time series, $a1$ provides smaller and $a2$ greater misclassification error than a. For the non averaged time series without differentiation, a provides better discrimination between protein families, than both $a2$ and $a1$.

In case of almost all examined protein-derived time series, the box-counting-algorithm-based second degree polynomial $a1$ provides lower values of the misclassification error, wile compared to $a2$, with difference from around 0.09% (2.45% vs. 2.54%, for the twice differentiated modified moving average) up to around 3.22% (2.45% vs 6.16%, for the modified moving average without differentiation). The only exception, where $a2$ performs better, compared to $a1$ is once differentiated exponential moving average time series. Here the observed difference is around 0.02% (211 vs. 209 misclassified proteins).

9 Discussion

In this thesis, a set of new algorithms for calculating alignment-independent descriptors was introduced. Methods based on information theory, chaos theory and statistical physics, although fairly new to the bioinformatics researched area, have a well established theoretical background, and have been successfully used in many different fields, including life science research [91, 92, 93, 94, 98, 105]. We postulate that our newly established framework can be used in combination with existing methods, or as an alternative to them.

In order to compare the effectiveness of the new developed protocols with the existing ones, we have calculated the values of the standard non-alignment protein descriptors, introduced in Chapter 3 [63], for the data set used to evaluate the methods described in Chapters 4-8.

Firstly, we are going to compare the numbers of misclassification errors calculated for the **standard non-alignment descriptor vectors** for the two data sets, described in detail in sections 3.11 and 4.1:

- 3872 proteins belonging to 40 protein families, results can be found in the Table 13.
- 9693 proteins belonging to 100 protein families, results can be found in the Table 30.

Figure 35 shows the dependence between the misclassification errors calculated for the 3872-elements data set and the corresponding ones, calculated for the 9693-elements data set for the same descriptor type. The dependence seems to be linear. The best fit regression line $y \approx 2.95x - 7.94$ is marked blue.

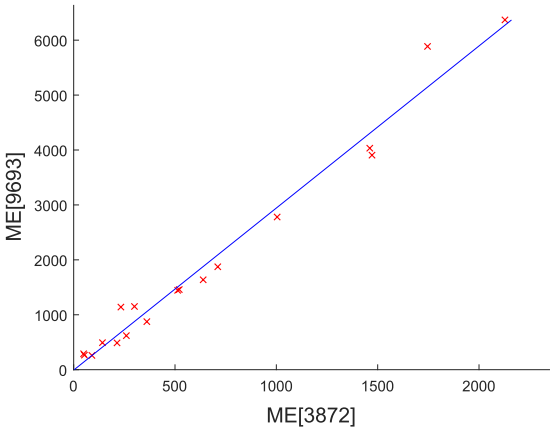

Figure 35: ME (3872) denotes the number of misclassified proteins for 3872 protein data set and ME (9693) denotes the number of misclassified proteins for 9693-elements protein data set.

Based on that observation we propose a **threshold value y^a** that can be used to evaluate the performance of descriptor vectors by looking at the misclassification error calculated for

Descriptor	n_f	misclassification error	threshold value y^a	y^b	Y^A	Y^B
Amino Acid Composition (AAC)*	20	2780 (28.68%)	2353	2289		
Dipeptide Composition (DC)*	400	265 (2.73%)	110	114	375	382
AAC* and DC*	420	290 (2.99%)	104	109	354	366
Entropy and Conditional entropy	6	6369 (65.71%)	7866	7632		
$H[DC_{AAC}]$[1]	20	1874 (19.33%)	2353	2289		
$H[TC_{AAC}]$[1]	20	1637 (16.89%)	2353	2289		
$H[TC_{DC}]$[1]	400	1151 (11.87%)	110	114	375	382
$H[DC_{AAC}]$, $H[TC_{AAC}]$, $H[TC_{DC}]$	440	492 (5.08%)	98	104	334	349
Composition*	21	3906 (40.30%)	2241	2180		
Transition*	21	4033 (41.61%)	2241	2180		
Distribution*	105	5886 (60.72%)	440	436	1500	1464
Composition/Transition/Distribution*	147	1459 (15.05%)	310	311	1056	1044
Conjoint Triad (CT)*	343	1140 (11.76%)	127	133	432	446
Sequence Order Coupling Numbers ($SOCN$)*	60	1445 (14.91%)	777	763		
Quasi sequence order descriptors ($QSOD$)*	100	488 (5.03%)	464	457		
Pseudo Amino Acid Composition I ($PAAC$)*	50	875 (9.03%)	936	915		
Pseudo Amino Acid Composition II ($APAAC$)*	80	620 (6.40%)	582	572		
$SOCN$*, $QSOD$*, $PAAC$*, $APAAC$*	290	260 (2.68%)	154	157	525	527

[1] partial conditional entropy

$H[DC_{AAC}]$ based on proportions of dipeptides starting with specific amino acid

$H[TC_{AAC}]$ based on proportions of triprptides starting with specific amino acid

$H[TC_{AAC}]$ based on proportions of triprptides starting with specific dipeptide

Table 30: The descriptor vectors misclassification error obtained for protein data set containing 9693 proteins belonging to 100 protein families. The n_f denotes the dimension of descriptor vector. Descriptor vectors marked with * had their values calculated with protr package[63]. The combinations of the descriptor vectors are marked gray.

the 9693-elements data set:

$$y^a(n_f) \approx 2.95 \times y(n_f)$$

where n_f are the dimensions of descriptor vectors, and **y** are the threshold values calculated in the Chapter 3, section 3.11 (Table 13) for the 3872-elements data set.

Figure 36 shows the dependence between the dimensions of the descriptor vector, and the misclassification error values calculated for the standard non-alignment protein descriptors for the 9693-element data set (the values can be found in the table Table 30). The trend we observe is analogous to the one seen for the 3872-elements data set (Figure 18). There is a definite decrease of misclassification error with increased number of descriptors, and again, this decrease is not linear. It extrapolates with $y = \frac{a_1}{x}$, and the rectangular hyperbola (for $a_1 \approx 45797.68$) was fitted to the data.

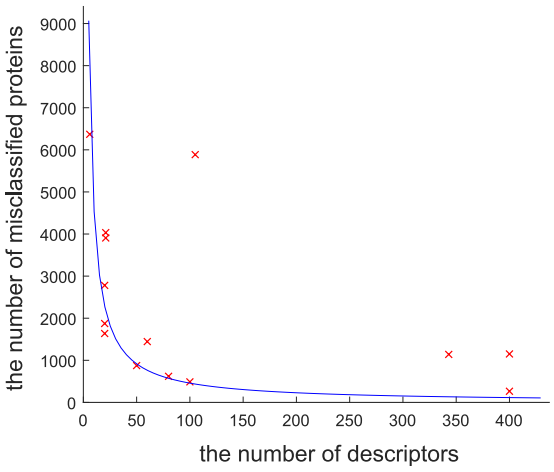

Figure 36: The dependence between the number of used descriptors and misclassification error.

The threshold values $y(n_f)$, that were used for the y^a calculations, were obtained by fitting the $y=\frac{a}{x}$ hyperbola to the misclassification error results for the 3872-element data set (Chapter 3, Figure 18) and $a_1 \approx 2.86 \times a$.

Therefore we propose the a **threshold value $\mathbf{y^b}$**, that can be used to evaluate the performance of descriptor vectors for the 9693-elements data set, to be calculated analogically as the threshold values **y** calculated for the 3872-elements data set. The descriptors can be considered to be better performing than the standard ones, if the pair $\{x_i, y_i\}$; (where $x_i = n_f(i)$ is a dimension of the descriptor vector, and y_i is the number of misclassified proteins) lays below the $y = \frac{a_1}{x}$ fit line: $y_i \leq y^b(n_f(i))$

The threshold values y^a and y^b can be found in Table 30. y^a values are between 94.23% and 103.07% of the y^b values.

Figure 37 shows the misclassification error calculated for the standard non-alignment protein descriptors, divided by corresponding threshold values plotted against the dimension of the descriptor vectors n_f (compare with the Figure 19). Blue dashed lines represents the averaged values:

$$\left\langle \frac{y}{y^a} \right\rangle = \frac{1}{18} \sum_{i=1}^{18} \frac{y_i}{y^a(n_f(i))} \approx 3.36$$
$$\left\langle \frac{y}{y^b} \right\rangle = \frac{1}{18} \sum_{i=1}^{18} \frac{y_i}{y^b(n_f(i))} \approx 3.41$$

to highlight the misclassification error results smaller than average ones.

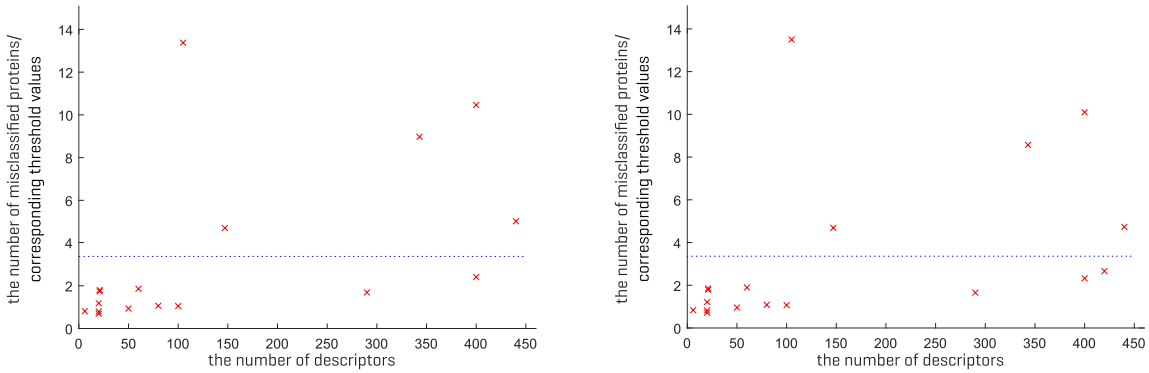

Figure 37: The proportion of misclassification error divided by corresponding threshold value y^a (left panel) and y^b (right panel), plotted against the dimension of the descriptor vectors.

As discussed in the Chapter 3, the expectation that the misclassification error, for the descriptor vector, will be lower than the corresponding threshold value, may be too optimistic. The risk of that is particularly strong in case of the high-dimension-descriptor vectors (Figure 37), possibly because all of the standard non-alignment ones, used for the threshold values calculations, have dimension at most 400, and around half of them lower or equal 60.

We therefore propose more realistic **threshold values $\mathbf{Y^A}$** and **$\mathbf{Y^B}$** to evaluate the performance of the descriptor vectors characterised by high dimensions:

$$Y^A = \left\langle \frac{y}{y^a} \right\rangle \times y^a \approx 3.36 \times y^a$$
$$Y^B = \left\langle \frac{y}{y^b} \right\rangle \times y^b \approx 3.41 \times y^b$$

The threshold values Y^A and Y^B for the descriptor vectors with the dimension higher than can be found in the Table 30. Y^A are between 95.70% and 103.28% of the Y^B values. For the 553 dimensional vectors the threshold values are: Y^A=264 misclassified proteins and Y^B=278 misclassified proteins.

When we compared, the results of the misclassification errors for the newly derived protein

descriptors (Chapter 4-8) with the Y^A and Y^B threshold values values calculated for the 553-dimensional vectors ($n_f = 553$), we can see some regularities:

- Most but not all results that are lower or equal the Y^A and/or Y^B thresholds are obtained for the information based theory descriptors, calculated in Chapter 4. and second degree polynomials box-counting-algorithm-based-descriptors calculated in section 8.4 of Chapter 8.

- The misclassification error lower or equal to Y^B=278, can be observed for all cases of once and twice different time series for the following:

  - the approximate entropy descriptors calculated for the modified and exponential moving average time series;
  - the approximate entropy descriptors calculated for the non-averaged time series for the combined values of parameters:
    - $r = 0.25 \times Std$ and $m = 2$,
    - $r = 1 \times Std$ and $m = 3$;
  - the approximate-entropy-based-descriptors calculated for the non-averaged and modified moving average time series;
  - the approximate-entropy-based-descriptors calculated for the exponential moving average time series for the value of parameter $r = 1 \times Std$;
  - the sample entropy descriptors calculated for the modified and exponential moving average time series for the value of parameters $r = 1 \times Std$ and $m = 2$, even when protein length data are not considered;
  - the sample-entropy-based-descriptors calculated for the modified and exponential moving average time series for the value of parameters $r = 1 \times Std$ even when protein length data are not considered;
  - the second degree polynomials box-counting-algorithm-based-descriptors calculated for the non-averaged, modified and exponential moving average time series.

- The misclassification error lower or equal to Y^B=278, can be observed for all cases of twice (but not once) differentiated time series for the following:

  - the approximate entropy descriptors calculated for the non-averaged time series for the value of parameters $r = 0.25 \times Std$ and $m = 3$;
  - the first degree polynomials box-counting-algorithm-based-descriptors calculated for the modified and exponential moving average time series;
  - the positive and negative normalised-grand-total descriptors calculated for the non-averaged, modified and exponential moving average time series.

- The misclassification error lower or equal to Y^B=278, can be observed for all cases of once (but not twice) differentiated time series for the following:
  - the approximate-entropy-based-descriptors calculated for the exponential moving average time series for the value of parameter $r = 0.25 \times Std$.
- The misclassification error lower or equal to Y^B=278, can be observed for non differentiated moving average time series for the following:
  - the approximate-entropy-based-descriptors calculated for the exponential and modified moving average time series for the value of parameter $r = 0.25 \times Std$;
  - the first degree polynomials box-counting-algorithm-based-descriptors calculated for the modified moving average time series;
  - one of the second degree polynomials box-counting-algorithm-based-descriptors ($a1$) calculated for the simple, modified and exponential moving average time series;
  - the Type I mean period and Type I mean frequency descriptors calculated for the simple, modified and exponential moving average time series.

The obvious conclusion to be drawn from these observations, is that the process of calculating both first and second order derivatives of the examined protein-derived time series contributes significantly to the efficiency with which descriptors differentiate between proteins belonging to different families. Additionally, a large proportion of best performing descriptors are calculated for the modified end exponential moving average time series, where the current value of averaged numerical sequence depends on all previous values of the original ones, with weights decreasing with time.

The Lyapunov exponent calculated with the Type II mean and median period for the non-averaged and non-differentiated time series provides misclassification errors lower than the threshold value Y^A=264. The same is true for non-differentiated and non-averaged Type I mean period and Type I mean frequency descriptors calculated for the non-averaged and non-differentiated time series.

Calculating the misclassification error, for the combination of more than one 553-dimensional descriptors, for the large (9693-elements) data-set provides a challenge, even with the LDA low computational costs related to its linearity. Hence the need to apply the dimensionality reduction, and the LDA technique can be also used as such [106].

Unlike PCA it is a supervised learning technique, and this difference is the reason why we used it for this part of our research. For the calculations of the LDA components we used the Matlab code [107]. In order to determine the misclassification error for the combination of two descriptors vectors, we took the first 2x50 LDA features for the combination of three vectors we took the first, 3x50 LDA features etc. (Table 31).

Descriptor	The number of LDA components	The number of misclassified proteins
PER	100	67 (0.69%)
SUM		93 (0.96%)
FD2		64 (0.66%)
PER_ pos + HURST		62 (0.64%)
PER_ neg + HURST		71 (0.73%)
PER_ pos + FD1		76 (0.78%)
PER_ neg + FD1		80 (0.82%)
FD1 + HURST		71 (0.73%)
FD2 + HURST	150	26 (0.27%)
PER + HURST		25 (0.26%)
PER + MEAN		18 (0.19%)
PER_ pos + HURST + MEAN		12 (0.12%)
PER_ neg + HURST + MEAN		15 (0.15%)
PER_ pos + FD2		30 (0.31%)
PER_ neg + FD2		25 (0.26%)
FD2 + MEAN		16 (0.17%)
PER + FD2	200	10 (0.10%)
PER + HURST + FD1		9 (0.09%)
PER + FD2 + HURST	250	2 (0.02%)
PER + FD2 + MEAN		4 (0.04%)
PER + SUM + HURST		2 (0.02%)
PER + SUM + FD1		2 (0.02%)
PER + FD2 + SUM	300	0 (0.00%)
PER + FD2 + MEAN + HURST		2 (0.02%)

PER – positive and negative persistence exponent; exponential moving average; time series without differentiation
SUM – positive and negative sum total of the pick; exponential moving average; second derivative
HURST – Hurst exponent; exponential moving average; second derivative
FD1 –Box-counting-algorithm-based descriptor; first degree polynomials; exponential moving average; first derivative
FD2 – Box-counting-algorithm-based descriptor; second degree polynomials; exponential moving average; first derivative
MEAN –Type I mean frequency; simple moving average; time series without differentiation

Table 31: The classification error obtained with Linear Discriminant Analysis classifier for different combinations of 553-dimensional sets of descriptors.

The combination of two descriptor vectors provides the misclassification error lower than 1%, and the combination of three lower than 0.5%. For the combination of six descriptor vectors we were able to achieve the complete reproduction of the protein classification.

Figure 38 shows the average misclassification error calculated for the protein sequences belonging to a different family. For each comparison the set containing 8 descriptors is considered: positive and negative persistence exponent calculated for the time series without differentiation, positive and negative sum total of the pick calculated for the twice differ-

entiated time series, type I mean frequency and mean period calculated for the time series without differentiation and second degree polynomials coefficients calculated with the box counting algorithm for the twice differentiated time series.

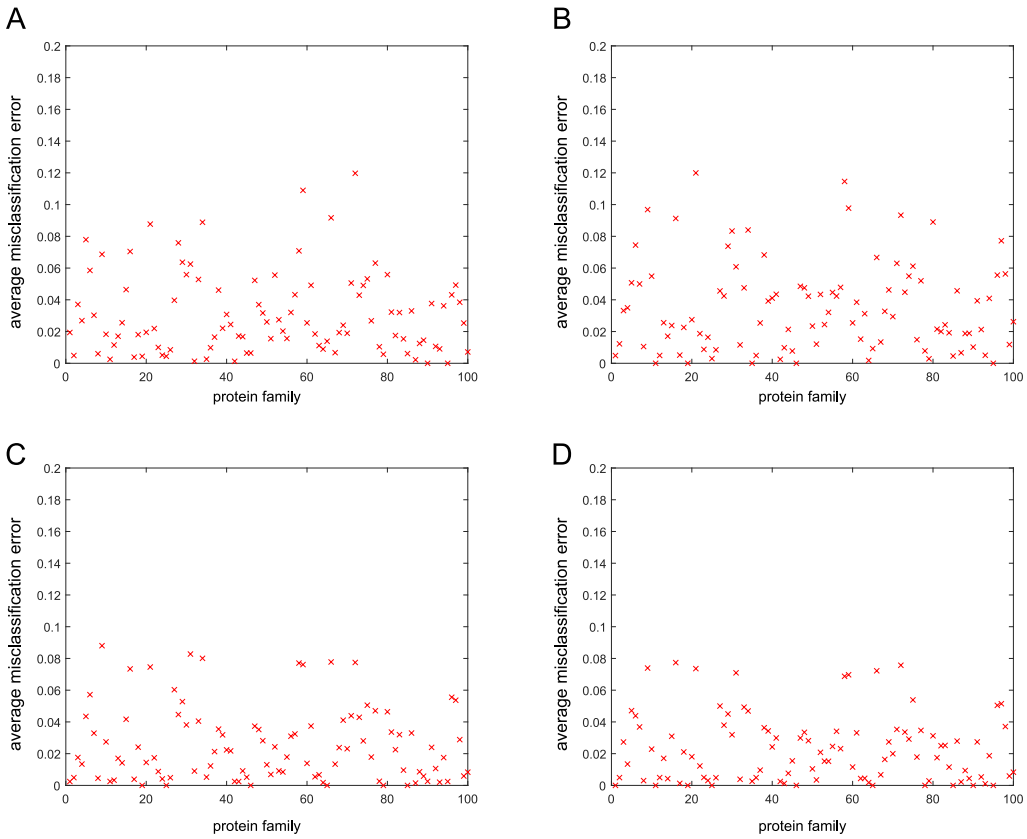

Figure 38: The average misclassification error calculated for the protein sequences belonging to a different families: (A) calculated for the non-averaged time series; (B) calculated for the simple moving average; (C) calculated for the modified moving average; (D) calculated for the exponential moving average. The X axis represents the sum of all misclassification errors obtained for all proteins belonging to an examined family divided by 8 (the number of descriptors' types taken into account) and by the number of proteins belonging to the family in question.

The average misclassification error calculated separately for each protein family varies from 0 to 11.97% for the descriptors calculated for the non-averaged time series; from 0 to 12.00% for the descriptors calculated for the simple moving average; from 0 to 8.80% for the descriptors calculated for the modified moving average; from 0 to 7.74% for the descriptors calculated for the exponential moving average. The expected value of the average misclassification error is respectively 3.07±2.56%, 3.47±2.80%, 2.53±2.32% and 2.27±2.11%.

Table 32 shows protein families (see also Appendix D) that are characterised by the largest and smallest average misclassification error. An undeniable tendency can be seen, as the protein families with relatively large (small) misclassification error calculated for one type of moving average tends to have also relatively large(small) misclassification error calculated

for another type. Biological explanation for this type of behaviour is not obvious at this stage of research and requires further studies.

	protein families with the average misclassification error		
	equel 0	between 2 and 3 Std above the expected value	more then 3 Std above the expected value
time series	90, 95	21, 34, 66	59, 72
simple moving average	11, 19, 35, 46, 95	9, 16, 58, 59, 72	21
modified moving average	19, 25, 46, 65, 79, 85	9, 16, 21, 31, 34, 58, 59, 66, 72	
exponential moving average	1, 11, 19 25, 46, 65, 78, 85, 90, 95	9, 16, 21 31, 58, 59 66, 72	

Table 32: Protein families that are characterised by relatively largest and smallest average misclassification error.

The physicochemical properties of amino acids are common [28, 29, 30] and well justified choice for the search of the alignment independent protocol of protein classification. Many of previous studies focused on relatively small protein test sets and tailored their approach for the specific task related to chosen protein group. Examples of such research [26, 28, 29, 30] have been undeniably successful. That led us to believe in the possibility of creating more universal protocol that would allow to recreate phylogenetic classification of random and unrelated proteins. We were able to achieve that goal for a large data set with accuracy reaching 100% and with the approach that could easily be automatised and applied for any protein (or DNA/RNA) data. Unfortunately, it was achieved with quite large computational cost. The number of necessary dimensions of descriptor vector exceed 3000, and the decrease of accuracy with the increase of the size of the data is a considerable problem, not unlike the one faced by alignment methods for sequence classification. However, there is a fundamental difference between alignment-based methods and our approach. Translation of a protein sequence into a multidimensional vector opens countless possibilities. Firstly, it opens a door for a search of similarities between proteins having similar function that are not close in the evolutionary sense [30]. For more general application, it allows the development of machine learning techniques that would effectively perform the comparison of the newly discovered protein with a whole group at once, instead of with each element in this group separately. That could immensely decrease the computation cost related to the classification task. Also, in some cases, the alignment independent methods could be used as an addition, not a replacement, for the alignment ones. They could become the first step, highlighting the possible candidates for phylogenetic relation of the sequence in question.

9.1 Conclusions

We proposed a new, alignment-independent protocol to discriminate between proteins families. The common feature of all of the used methods is that, in a complex statistical way, they look for patterns in sequences. Their effectiveness varies to a degree, but even the least successful ones, indicate promising directions in the search for the ultimate alignment-independent protein descriptors for function dependent protein classification.

The noticeable increase of dimensions needed to maintain classification accuracy, with the data size, remains a problem. A possible solution could be the development of a protocol that allows the use of dimensionality reduction methods that are in no way dependent on the input data. However, with the increasing number of available protein sequences, this would still most likely not be enough. Therefore, we postulate the need to select or to develop an algorithm to be used in place of the LDA, that allows for the hierarchical alignment-independent classification, and thus limits the size of the data tested at each step.

The alignment independent protocol allowed us to reproduce the already existing classification based on evolutionary relationship. The next step is to test the methods against the data, where there is functional similarity without the sequence one; and in parallel, once again address the question of HLA class I protein super-types.

Ever tried. Ever failed. No matter.
Try Again. Fail again.
Fail better.

Samuel Beckett

References

[1] J. Jumper, R. Evans, A. Pritzel, et al. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596:583–589. https://doi.org/10.1038/s41586-021-03819-2

[2] A. McPherson, J.A. Gavira (2014). Introduction to protein crystallization. Acta Crystallographica Section F: Structural Biology Communications, 70(Pt 1):2–20.

[3] V.K. Vyas, R.D. Ukawala, M. Ghate, C. Chintha (2012). Homology modeling a fast tool for drug discovery: current perspectives. Indian Journal of Pharmaceutical Sciences, 74(1):1–17.

[4] S. Rani, O.P. Gupta (2017). Study and Analysis of Various Bioinformatics Applications using Protein BLAST: An Overview. Advances in Computational Sciences and Technology, 10:2587-2601.

[5] M. Davies, A. Secker, A. Freitas, J. Timmis, E. Clark, D. Flower (2008). Alignment-Independent Techniques for Protein Classification. Current Proteomics, 5:217-223.

[6] A. Zielezinski, H.Z. Girgis, G. Bernard, et al. (2019). Benchmarking of alignment-free sequence comparison methods. Genome Biology, 20:144. https://doi.org/10.1186/s13059-019-1755-7

[7] J. Ren, X. Bai, Y.Y. Lu, K. Tang, Y. Wang, G. Reinert, F. Sun (2018). Alignment-Free Sequence Analysis and Applications. Annual Review of Biomedical Data Science, 1:93–114.

[8] G.A. Wu, S.R. Jun, G.E. Sims, S.H. Kim (2009). Whole-proteome phylogeny of large dsDNA virus families by an alignment-free method. Proceedings of the National Academy of Sciences of the United States of America, 106(31):12826-12831. https://doi.org/10.1073/pnas.0905115106

[9] A. Zielezinski, S. Vinga, J. Almeida, W.M. Karlowski (2017). Alignment-free sequence comparison: benefits, applications, and tools. Genome Biology, 18:186. https://doi.org/10.1186/s13059-017-1319-7

[10] G.E. Sims, S.R. Jun, G.A. Wu, S.H. Kim (2009). Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. Proceedings of the National Academy of Sciences of the United States of America, 106:2677–2682.

[11] Y. Wang, L. Liu, L. Chen, T. Chen, F. Sun (2014). Comparison of metatranscriptomic samples based on k-tuple frequencies. PLOS ONE, 9(1):e84348.

[12] G. Reinert, D. Chew, F. Sun, M.S. Waterman (2009). Alignment-free sequence comparison (I): statistics and power. Journal of Computational Biology, 16(12):1615-34.

[13] K.D. Murray, C. Webers, C.S. Ong, J. Borevitz, N. Warthmann (2017). kWIP: The k-mer weighted inner product, a de novo estimator of genetic similarity. PLOS Computational Biology, 13(9):e1005727. https://doi.org/10.1371/journal.pcbi.1005727

[14] A. Nag, S Karforma (2015). A Space-Efficient Approach towards Distantly Homologous Protein Similarity Searches. International Journal of Advanced Research in Computer Science (IJARCS). Vol.6(2):19-22.

[15] G. Bernard, et al. (2016). Alignment-free microbial phylogenomics under scenarios of sequence divergence, genome rearrangement and lateral genetic transfer. Scientific Reports, 6:28970.

[16] C.R. De Pierri, R. Voyceik, L.G.C. Santos de Mattos, et al. (2020). SWeeP: representing large biological sequences datasets in compact vectors. Scientific Reports, 10:91. https://doi.org/10.1038/s41598-019-55627-4

[17] C.A. Leimeister, B. Morgenstern (2014). Kmacs: the k-mismatch average common substring approach to alignment-free sequence comparison. Bioinformatics, 30(14):2000–8.

[18] S. Vinga (2014). Information theory applications for biological sequence analysis. Briefings in Bioinformatics, 215(3):376-389.

[19] A.K. Saw, G. Raj, M. Das, et al. (2019). Alignment-free method for DNA sequence clustering using Fuzzy integral similarity. Scientific Reports, 9:3753. https://doi.org/10.1038/s41598-019-40452-6

[20] A.K. Saw, B.C. Tripathy, S. Nandi (2019). Alignment-free similarity analysis for protein sequences based on fuzzy integral. Scientific Reports, 9:2775. https://doi.org/10.1038/s41598-019-39477-8

[21] S. Kouchaki, A. Tapinos, D.L. Robertson (2019). A signal processing method for alignment-free metagenomic binning: multi-resolution genomic binary patterns. Scientific Reports, 9:2159. https://doi.org/10.1038/s41598-018-38197-9

[22] E. Borrayo, E.G. Mendizabal-Ruiz, H. Vélez-Pérez, R. Romo-Vázquez, A.P. Mendizabal, J.A. Morales (2014). Genomic Signal Processing Methods for Computation of Alignment-Free Distances from DNA Sequences. PLOS One, 9(11):e110954. https://doi.org/10.1371/journal.pone.0110954

[23] M.B. Fiamenghi, J.G.R. Bueno, A.P. Camargo, G. Borelli, M.F. Carazzolle, G.A.G. Pereira, L.V. Dos Santos, J. Jose (2022). Machine learning and comparative genomics approaches for the discovery of xylose transporters in yeast. Biotechnology for Biofuels and Bioproducts, 15(1):57.

[24] J. Zhang, K. Wang (2022). Mathematical Modeling and Computational Prediction of High-Risk Types of Human Papillomaviruses. Computational and Mathematical Methods in Medicine, 2022:1515810.

[25] A. Raies, E. Tulodziecka, J. Stainer, L. Middleton, R.S. Dhindsa, P. Hill, O. Engkvist, A.R.Harper, S. Petrovski, D. Vitsios (2022). DrugnomeAI is an ensemble machine-learning framework for predicting druggability of candidate drug targets. Communications Biology, 5(1):1291.

[26] H. Li, F. Sun (2018). Comparative studies of alignment, alignment-free and SVM based approaches for predicting the hosts of viruses based on viral sequences. Scientific Reports, 8:10032 https://doi.org/10.1038/s41598-018-28308-x

[27] Q. Zhang, S. Jun, M. Leuze, et al. (2017). Viral Phylogenomics Using an Alignment-Free Method: A Three-Step Approach to Determine Optimal Length of k-mer. Scientific Reports, 7:40712. https://doi.org/10.1038/srep40712

[28] S. Lertampaiporn, A. Hongsthong, W. Wattanapornprom, C. Thammarongtham (2022). Ensemble-AHTPpred: A Robust Ensemble Machine Learning Model Integrated With a New Composite Feature for Identifying Antihypertensive Peptides. Frontiers in Genetics, 13:883766.

[29] D. Simon, O. Borsani, C.V. Filippi (2022). RFPDR: a random forest approach for plant disease resistance protein prediction. PeerJ, 10:e11683.

[30] D. Deb, D. Mackey, S.O. Opiyo, J.M. McDowell (2018). Application of alignment-free bioinformatics methods to identify an oomycete protein with structural and functional similarity to the bacterial AvrE effector protein. PLOS ONE, 13(4):e0195559. https://doi.org/10.1371/journal.pone.0195559

[31] J.A. Owen, J. Punt, S.A. Stranford, P.P. Jones, J. Kuby (2013). Kuby immunology. New York: W.H. Freeman, 261-298.

[32] D.K. Male, J. Brostoff, D.B. Roth, I.M. Roitt (2013). Immunology. Elsevier/Saunders, 89-106.

[33] S. Vejbaesya, R. Thongpradit, S. Kalayanarooj, K. Luangtrakool, P. Luangtrakool, R.V. Gibbons, D. Srinak, S. Ngammthaworn, K. Apisawes, I. Yoon, S.J. Thomas, R.G. Jarman, A. Srikiakthachorn, S. Green, D. Chandanayingyong, S. Park, J. Friedman, A.L. Rothman, H.A.F. Stephens (2015). HLA Class I Supertype Associations With Clinical Outcome of Secondary Dengue Virus Infections in Ethnic Thais. The Journal of Infectious Diseases, 212(6):939–947. https://doi.org/10.1093/infdis/jiv127

[34] I.A. Doytchinova, P. Guan, D.R. Flower (2004). Identifying human MHC supertypes using bioinformatics methods. The Journal of Immunology, 172:4314-4323.

[35] M. Wang, M.H. Claesson (2014). Classification of human leukocyte antigen (HLA) supertypes. Methods in Molecular Biology, 1184:309-17.

[36] S. Muniz-Castrillo, A. Vogrig, J. Honnorat (2020). Associations between HLA and autoimmune neurological diseases with autoantibodies. Autoimmun Highlights, 11:2 `https://doi.org/10.1186/s13317-019-0124-6`

[37] A. Lazaryan, T. Wang, S.R. Spellman, H.L. Wang, J. Pidala, T. Nishihori, M. Askar, R. Olsson, M. Oudshoorn, H. Abdel-Azim, A. Yong, M. Gandhi, C. Dandoy, B. Savani, G. Hale, K. Page, M. Bitan, R. Reshef, W Drobyski, S.G. Marsh, K. Schultz, C.R. Muller, M.A. Fernandez-Vina, M.R. Verneris, M.M. Horowitz, M. Arora, D.J. Weisdorf, S.J. Lee (2016). Human leukocyte antigen supertype matching after myeloablative hematopoietic cell transplantation with 7/8 matched unrelated donor allografts: a report from the Center for International Blood and Marrow Transplant Research. Haematologica, 101(10):1267-1274.

[38] `http://www.rcsb.org/pdb/home/home.do`

[39] `ftp://ftp.ebi.ac.uk/pub/databases/ipd/imgt/hla`

[40] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, D.J. Lipman (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research, 25:3389-3402.

[41] `https://salilab.org/modeller`

[42] A.S. Konagurthu, J.C. Whisstock, P.J. Stuckey, A.M. Lesk (2006). MUSTANG: a multiple structural alignment algorithm. Proteins, 15;64(3):559-74.

[43] T.J. Dolinsky, J.E. Nielsen, J.A. McCammon, N.A. Baker (2004). PDB2PQR: an automated pipeline for the setup, execution, and analysis of Poisson-Boltzmann electrostatics calculations. Nucleic Acids Research, 32:W665-W667

[44] N.A. Baker, D. Sept, S. Joseph, M.J. Holst, J.A. McCammon (2001). Electrostatics of nanosystems: application to microtubules and the ribosome. Proceedings of the National Academy of Sciences of the United States of America, 98:10037-10041.

[45] L.J.P. van der Maaten, E.O. Postma, H.J. van den Herik (2010). Dimensionality Reduction: A Comparative Review.
`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.112.5472`.

[46] N. Lawrence (2005). Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. Journal of Machine Learning Research, 6:1783-1816.

[47] M.A. Kramer (1991). Nonlinear Principal Component Analysis Using Autoassociative Neural Networks. American Institute of Chemical Engineers Journal, 37:233-243.

[48] P. Li, S. Chen (2016). A review on Gaussian Process Latent Variable Models. Transactions on Intelligence Technology, 4:366-376.

[49] H. Moon, P.J. Phillips (2016). Computational and Performance Aspects of PCA-Based Face-Recognition Algorithms. Perception, 30(3):303-321.

[50] V. Deepu, S. Madhvanath, A.G. Ramakrishnan (2004). Principal component analysis for online handwritten character recognition. Proceedings of the 17th International Conference on Pattern Recognition.

[51] B. Wang, M.A. Kennedy (2014). Principal components analysis of protein sequence clusters. Journal of Structural and Functional Genomics, 15(1):1-11.

[52] MATLAB and Statistics Toolbox Release 2017b, The MathWorks, Inc., Natick, Massachusetts, United States.

[53] M. Scholz, M. Fraunholz, J. Selbig (2007). Nonlinear principal component analysis: neural network models and applications. In Principal Manifolds for Data Visualization and Dimension Reduction, edited by A.N. Gorban, B. Kegl, D.C. Wunsch, A.Y. Zinovyev. Volume 58 of LNCSE, pp 44-67.

[54] M. Scholz, R. Vigario (2002). Nonlinear PCA: a new hierarchical approach. M. Verleysen. Proceedings European Symposium on Artificial Neural Networks, 439-444.

[55] M. Scholz, F. Kaplan, C.L. Guy, J. Kopka, J. Selbig (2005). Non-linear PCA: a missing data approach. Bioinformatics, 21(20):3887-3895.

[56] M. Scholz (2012). Validation of nonlinear PCA. Neural Processing Letters, Volume 36, 1:21-30.

[57] M. Scholz (2007). Analysing periodic phenomena by circular PCA. S. Hochreiter, R. Wagner, editors, Proceedings of the Conference on Bioinformatics Research and Development BIRD'07, LNCS/LNBI 4414:38-47.

[58] M.E. Tipping, C.M. Bishop (1999). Probabilistic principal component analysis. Journal of the Royal Statistical Society, Series B, 21(3):611-622.

[59] B. Scholkopf, A. Smola, K.R. Muller (1998). Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, 10(5):1299-1319.

[60] M.K. Titsias, N.D. Lawrence (2010). Bayesian Gaussian Process Latent Variable Model. Artificial Intelligence, 9:844-851.

[61] N.D. Lawrence (2004). Gaussian process models for visualization of high dimensional data. Advances in Neural Information Processing Systems, 16:329-336.

[62] N.D. Lawrence, J. Quinonero-Candela (2006). Local distance preservation in the GP-LVM through back constraints. Proceedings of the 23rd international conference on Machine learning, 513-520.

[63] X. Nan, C. Dong-Sheng, Z. Min-Feng, X. Qing-Song (2015). protr/ProtrWeb: R package and web server for generating various numerical representation schemes of protein sequences. Bioinformatics, 31(11):1857-1859.

[64] C. Nef, M.A. Madoui, E. Pelletier, C. Bowler (2022). Whole-genome scanning reveals environmental selection mechanisms that shape diversity in populations of the epipelagic diatom Chaetoceros. PLOS Biology, 20(11):e3001893.

[65] C.E. Shannon (1948). A Mathematical Theory of Communication. Bell System Technical Journal, 27(3):379–423.

[66] T.M. Cover, J.A. Thomas (1991). Elements of Information Theory. A Wiley-Interscience Publication John Wiley & Sons, Inc.

[67] D. Santoni, G. Felici, D. Vergni (2016). Natural vs. random protein sequences: Discovering combinatorics properties on amino acid words. Journal of Theoretical Biology, 391:13-20.

[68] I. Dubchak, I. Muchink, S.R. Holbrook, S.H. Kim (1995). Prediction of Protein Folding Class Using Global Description of Amino Acid Sequence. Proceedings of the National Academy of Sciences, 92:8700-8704.

[69] I. Dubchak, I. Muchink, C. Mayor, I. Dralyuk, S.H. Kim (1999). Recognition of a Protein Fold in the Context of the SCOP Classification. Proteins: Structure, Function and Genetics, 35:401-407.

[70] J. Shen, J. Zhang, X. Luo, W. Zhu, K. Yu, K. Chen, Y. Li, H. Jiang (2007). Predicting Protein-protein Interactions Based Only on Sequences Information. Proceedings of the National Academy of Sciences, 104:4337-4341.

[71] K.C. Chou (2000). Prediction of Protein Subcellar Locations by Incorporating Quasi-Sequence-Order Effect. Biochemical and Biophysical Research Communications, 278:477-483.

[72] G. Schneider, P. Wrede (1994). The rational design of amino acid sequences by artificial neural networks and simulated molecular evolution: Do novo design of an idealized leader cleavage site. Biophysical Journal, 66:335–344.

[73] R. Grantham (1974). Amino acid difference formula to help explain protein evolution. Science, 185:862–864.

[74] K.C. Chou (2001). Prediction of Protein Cellular Attributes Using Pseudo-Amino Acid Composition. PROTEINS: Structure, Function, and Genetics, 43:246-255.

[75] https://www.genome.jp/aaindex

[76] B. Liu, X. Wang (2013). Using Chou's Pseudo Amino Acid Composition for Protein Remote Homology Detection. Engineering, 5:149-153.

[77] http://pfam.xfam.org

[78] T. Shinba, K. Murotsu, Y. Usui, Y. Andow, H. Terada, N. Kariya, Y. Tatebayashi, Y. Matsuda, G. Mugishima, Y. Shinba, G. Sun, T. Matsui (2021). Return-to-Work Screening by Linear Discriminant Analysis of Heart Rate Variability Indices in Depressed Subjects. 21:51-77. https://doi.org/10.3390/s2115517

[79] M.Q. Zhang (2000). Discriminant analysis and its application in DNA sequence motif recognition. Briefings in Bioinformatics, 1(4):331–342. https://doi.org/10.1093/bib/1.4.331

[80] Q. Mai (2013). A review of discriminant analysis in high dimensions WIREs Computational Statistics. 5(3):81-266

[81] D. Huang, Y. Quan, M. He, B. Zhou (2009). Comparison of linear discriminant analysis methods for the classification of cancer based on gene expression data. J Exp Clin Cancer Res. 28(1):149. https://doi:10.1186/1756-9966-28-149.

[82] A.K. Chattopadhyay, D. Nasiev, D.R. Flower (2015). A statistical physics perspective on alignment-independent protein sequence comparison. Bioinformatics, 31(15):2469-2474.

[83] S.M. Pincus, I.M. Gladstone, R.A. Ehrenkranz (1991). A regularity statistic for medical data analysis. Journal of Clinical Monitoring and Computing, 7(4):335-345.

[84] S.M. Pincus (1991). Approximate entropy as a measure of system complexity. Proceedings of the National Academy of Sciences, 2188(6):2297–2301.

[85] A. Delgado-Bonal, A. Marshak (2019). Approximate Entropy and Sample Entropy: A Comprehensive Tutorial. Entropy, 21(6):541.

[86] J.S. Richman, J.R. Moorman (2000). Physiological time-series analysis using approximate entropy and sample entropy. American Journal of Physiology - Heart and Circulatory Physiology, 278(6):H2039–49.

[87] M.T. Rosenstein, J.J. Collins, C.J. De Luca (1993). A practical method for calculating largest Lyapunov exponents from small data sets. Physica D: Nonlinear Phenomena, 65(1-2):117-134.

[88] Largest Lyapunov Exponent with Rosenstein's Algorithm, https://www.mathworks.com/matlabcentral/fileexchange/38424-largest-lyapunov-exponent-with-rosenstein-s-algorithm, MATLAB Central File Exchange.

[89] B. Derrida, V. Hakim, R. Zeitak (1996). Persistent Spins in the Linear Diffusion Approximation of Phase Ordering and Zeros of Stationary Gaussian Processes. Physical Review Letters, 77(14):2871-2874.

[90] B. Derrida, et al. (1995). Exact first-passage exponents of 1D domain growth: relation to a reaction-diffusion model. Physical Review Letters, 75:751–754

[91] A.K. Chattopadhyay, N. Burroughs (2007). Close contact fluctuations: the seeding of signaling domains in immunological synapse. Europhysics Letters, 77:48003.

[92] D.J. Bush, A.K. Chattopadhyay (2014). Contact time periods in immunological synapse. Physical Review E, 90:042706.

[93] H.C. Tuckwell, F.Y.M. Wan (2000). First passage time to detection in stochastic population dynamical models for HIV-1. Applied Mathematics Letters, 13:79–83.

[94] D. Poland (2004). The persistence exponent of DNA. Biophysical Chemistry, 110:59–72.

[95] J.J. Sines, D..D Hackney (1987). A residence-time analysis of enzyme kinetics. Biochemical Jurnal 243(1):159-164. https://doi.org/10.1042/bj2430159

[96] F. Cervantes-De la Torre, J.I. González-Trejo, C.A. Real-Ramírez, L.F Hoyos-Reyes (2013). Fractal dimension algorithms and their application to time series associated with natural phenomena. Journal of Physics: Conference Series, 475:012002

[97] K.S. Nikhila, V. Vrinda (2018). Nair Protein Sequence Similarity Analysis Using Computational Techniques Materials Today. Proceedings, 5:724–731.

[98] X. Deng, J. Meng (2017). A Non-Linear Analogy Procedure for Gene Repair. Proceedings 1(3):128. https://doi.org/10.3390/IS4SI-2017-03999

[99] M.S. Raimundo, J. Okamoto Jr (2018). Application of Hurst Exponent (H) and the R/S Analysis in the Classification of FOREX Securities. International Journal of Modeling and Optimization Vol.8, No.2.

[100] S. Chandrasekaran, S. Poomalai, B. Saminathan, S. Suthanthiravel, K. Sundaram, F.F. Abdul Hakkim (2019). An investigation on the relationship between the Hurst exponent and the predictability of a rainfall time series. Meteorol, 26:511–519. https://doi.org/10.1002/met.1784

[101] G. Gonzato, F. Mulargia, W. Marzocchi (1998). Practical application of fractal analysis: problems and solutions. Geophysical Journal International, 132(2):275–282. https://doi.org/10.1046/j.1365-246x.1998.00461.x

[102] B.S. Raghavendra, D.N. Dutt (2010). Computing fractal dimension of signals using multiresolution box-counting method. World Academy of Science, Engineering and Technology International Journal of Electronics and Communication Engineering, 4(1):183-198.

[103] M. Valla, J. Nedved, D. Pavlik, V. Adam, J. Hubalek, L. Trnkova, R. Kizek, I. Provaznik (2010). Box counting method in recording, processing and evaluation of genomic signals. Journal Of Biochemical Technology, 2(5):S91-S93

[104] L. Athanasopoulou, D. Sellis, Y. Almirantis (2014). A Study of Fractality and Long-Range Order in the Distribution of Transposable Elements in Eukaryotic Genomes Using the Scaling Properties of Block Entropy and Box-Counting. Entropy, 16:1860-1882.

[105] X. Wang, G.F. Weber (2021). Quantitative Analysis of Protein Evolution: The Phylogeny of Osteopontin. Frontiers in Genetics, 12:700789.

[106] J. Ye, S. Ji (2010). Discriminant Analysis for Dimensionality Reduction: An Overview of Recent Developments. Biometrics: Theory, Methods, and Applications. Edited by Boulgouris, Plataniotis, and Micheli-Tzanakou.

[107] Yarpiz (2022). Linear Discriminant Analysis (LDA) aka. Fisher Discriminant Analysis (FDA). https://www.mathworks.com/matlabcentral/fileexchange/53151-linear-discriminant-analysis-lda-aka-fisher-discriminant-analysis-fda, MATLAB Central File Exchange.

Appendix A

Score = 519 bits (1337), Expect = 0.0, Method: Compositional matrix adjust.
Identities = 246/275 (89%), Positives = 253/275 (92%), Gaps = 0/275 (0%)

Query	25	GSHSMRYFFTSVSRPGRGEPRFIAVGYVDDTQFVRFDSDAASQKMEPRAPWIEQEGPEYW	84
		GSHSMRYFFTSVSRPGRGEPRFIAVGYVDDTQFVRFDSDAASQ+MEPRAPWIEQEGPEYW	
Sbjct	1	GSHSMRYFFTSVSRPGRGEPRFIAVGYVDDTQFVRFDSDAASQRMEPRAPWIEQEGPEYW	60
Query	85	DQETRNMKAHSQTDRANLGTLRGYYNQSEDGSHTIQIMYGCDVGPDGRFLRGYRQDAYDG	144
		D ETR +KAHSQT R +LGTLRGYYNQSE GSHT+Q MYGCDVG D RFLRGY Q AYDG	
Sbjct	61	DGETRKVKAHSQTHRVDLGTLRGYYNQSEAGSHTVQRMYGCDVGSDWRFLRGYHQYAYDG	120
Query	145	KDYIALNEDLRSWTAADMAAQITKRKWEAVHAAEQRRVYLEGRCVDGLRRYLENGKETLQ	204
		KDYIAL EDLRSWTAADMAAQ TK KWEA H AEQ R YLEG CV+ LRRYLENGKETLQ	
	121	KDYIALKEDLRSWTAADMAAQTTKHKWEAAHVAEQLRAYLEGTCVEWLRRYLENGKETLQ	180
Query	205	RTDPPKTHMTHHPISDHEATLRCWALGFYPAEITLTWQRDGEDQTQDTELVETRPAGDGT	264
		RTD PKTHMTHH +SDHEATLRCWAL FYPAEITLTWQRDGEDQTQDTELVETRPAGDGT	
Sbjct	181	RTDAPKTHMTHHAVSDHEATLRCWALSFYPAEITLTWQRDGEDQTQDTELVETRPAGDGT	240
Query	265	FQKWAAVVVPSGEEQRYTCHVQHEGLPKPLTLRWE	299
		FQKWAAVVVPSG+EQRYTCHVQHEGLPKPLTLRWE	
Sbjct	241	FQKWAAVVVPSGQEQRYTCHVQHEGLPKPLTLRWE	275

The example of the BLAST alignment.

```
>P1;1I4F
structureX:1I4F.pdb:   1 :A:+275 :A:MOL_ID  1; MOLECULE  HLA CLASS I HISTOCOMPATIBILITY ANTIGEN, A-2
ALPHA CHAIN; CH$
GSHSMRYFFTSVSRPGRGEPRFIAVGYVDDTQFVRFDSDAASQRMEPRAPWIEQEGPEYWDGETRKVKAHSQTHR
VDLGTLRGYYNQSEAGSHTVQRMYGCDVGSDWRFLRGYHQYAYDGKDYIALKEDLRSWTAADMAAQTTKHKWEAA
HVAEQLRAYLEGTCVEWLRRYLENGKETLQRTDAPKTHMTHHAVSDHEATLRCWALSFYPAEITLTWQRDGEDQT
QDTELVETRPAGDGTFQKWAAVVVPSGQEQRYTCHVQHEGLPKPLTLRWE*

>P1;HLA00001
sequence:HLA00001:FIRST:@:     : :::-1.00:-1.00
GSHSMRYFFTSVSRPGRGEPRFIAVGYVDDTQFVRFDSDAASQKMEPRAPWIEQEGPEYWDQETRNMKAHSQTDR
ANLGTLRGYYNQSEDGSHTIQIMYGCDVGPDGRFLRGYRQDAYDGKDYIALNEDLRSWTAADMAAQITKRKWEAV
HAAEQRRVYLEGRCVDGLRRYLENGKETLQRTDPPKTHMTHHPISDHEATLRCWALGFYPAEITLTWQRDGEDQT
QDTELVETRPAGDGTFQKWAAVVVPSGEEQRYTCHVQHEGLPKPLTLRWE*
```

```
from modeller import *
from modeller.automodel import *

log.verbose()
env = environ()

a = automodel(env, alnfile  = '../modeller_inputA/A_prot1.ali', knowns   = '1I4F', sequence = 'HLA00001')
a.starting_model= 1
a.ending_model  = 1

a.make()
```

The example of the Modeller input files.

```
read
   mol pqr HLA00001.pqr
end
elec
   mg-auto
   dime 97 65 65
   cglen 210 210 210
   fglen 36 64 64
   cgcent 25.137 12.035 13.87
   fgcent 25.137 12.035 13.87
   mol 1
   lpbe
   bcfl sdh
   pdie 2.0000
   sdie 78.5400
   srfm smol
   chgm spl2
   sdens 10.00
   srad 1.40
   swin 0.30
   temp 298.15
   calcenergy total
   calcforce no
   write pot dx HLA00001.pqr
end
print elecEnergy 1 end
quit
```

The example of the APBS input files

		Subset 1	Subset 2	Subset 3
TYPE 1 models	All points	4123	4250	
	Points inside the Van der Waals sphere	899	370	
	Points outside the Van der Waals sphere	1989	3370	
	Points outside the Van der Waals sphere within 2 $\AA$ distance from it	879	459	
TYPE 2 models	All points	4123	4250	
	Points inside the Van der Waals sphere	1057	441	
	Points outside the Van der Waals sphere	2082	3452	
	Points outside the Van der Waals sphere within 2 $\AA$ distance from it	1051	577	
TYPE 3 models	All points	4123	4250	4913
	Points inside the Van der Waals sphere	814	344	627
	Points outside the Van der Waals sphere	1947	3319	3338
	Points outside the Van der Waals sphere within 2 $\AA$ distance from it	761	448	721

Table 33: Supplementary Table 1: The number of points in subsets.

Appendix B

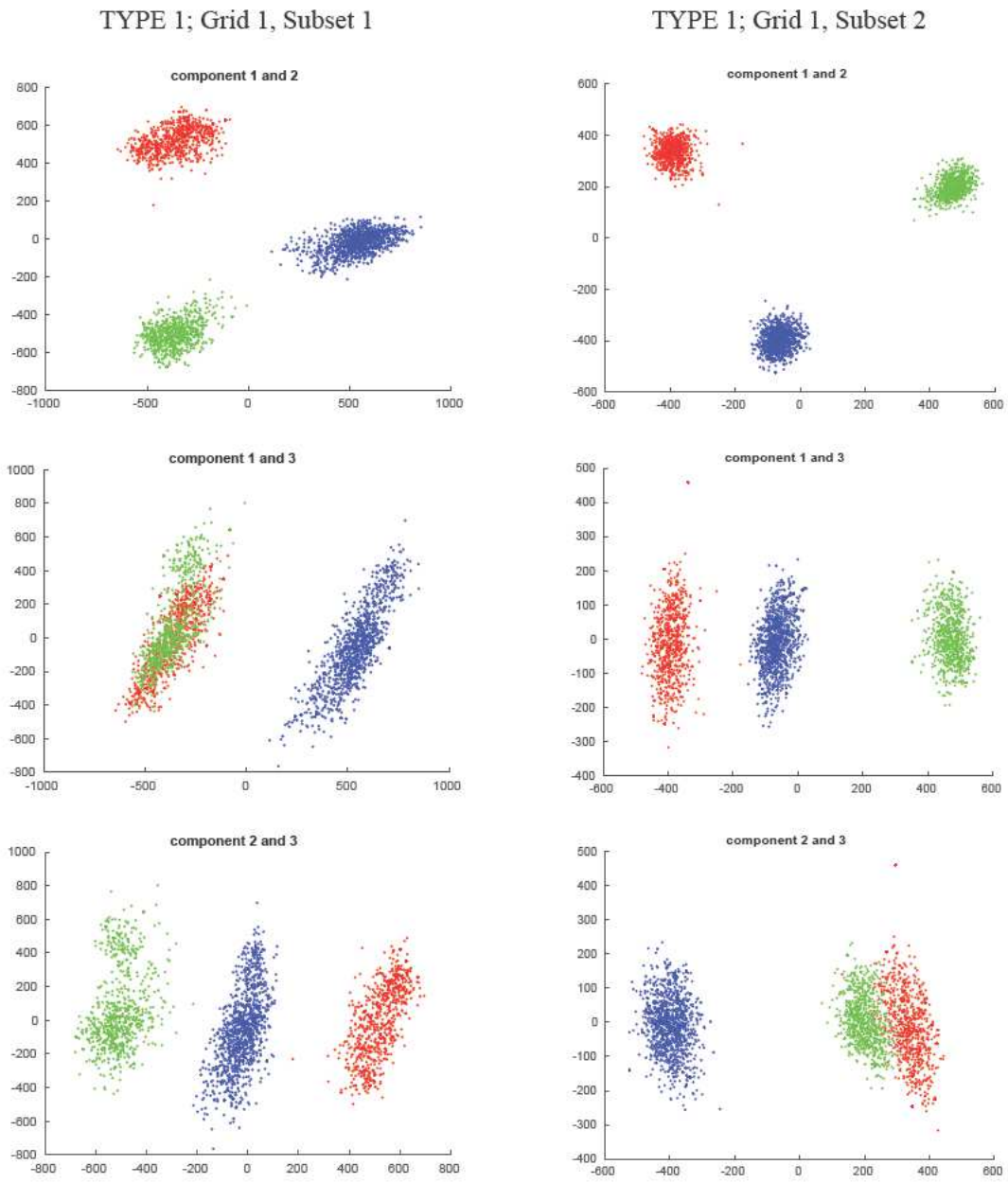

Figure 39: The PCA visualization of the TYPE 1 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for all points in the subsets.

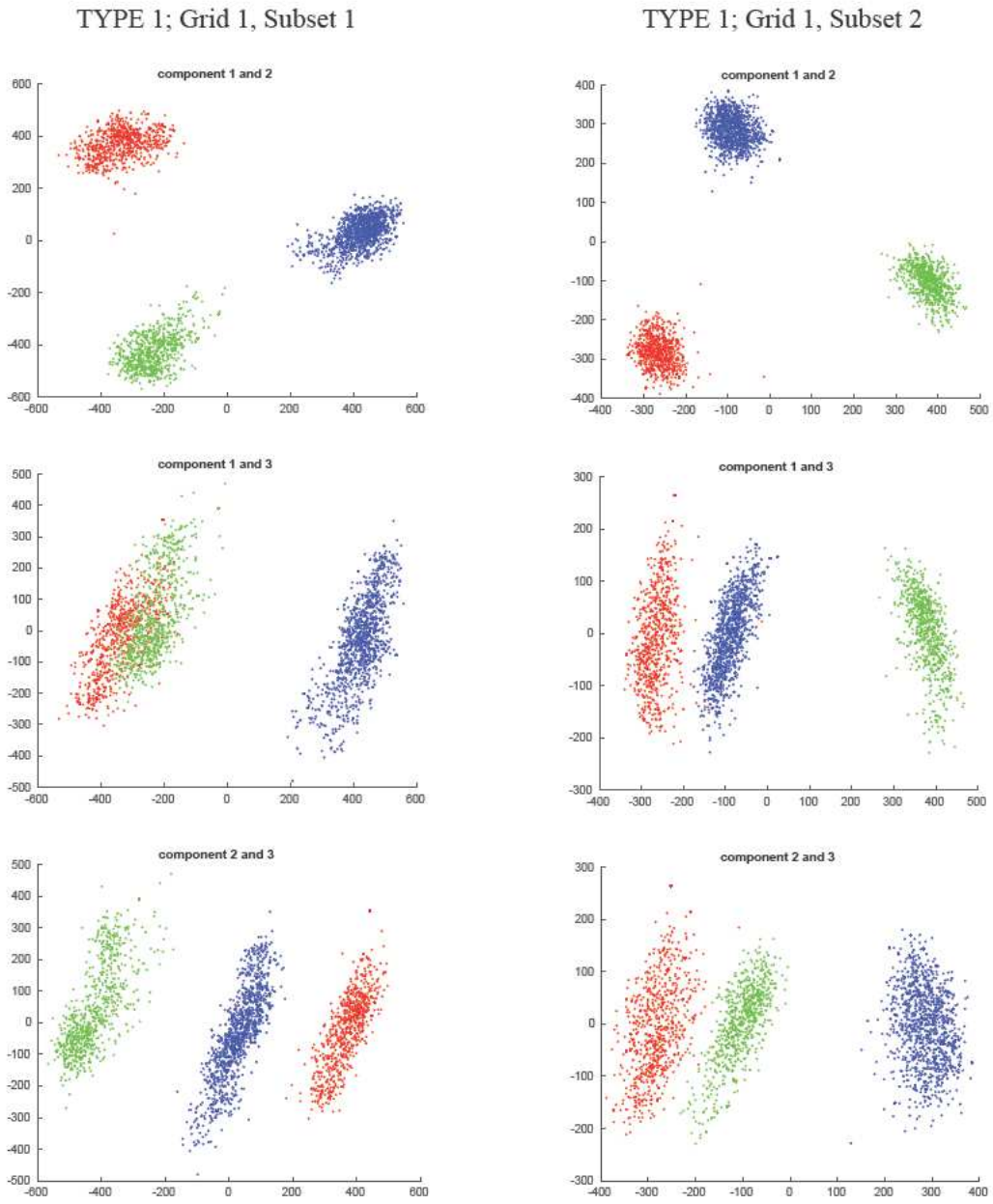

Figure 40: The PCA visualization of the TYPE 1 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned inside the Van der Waals sphere.

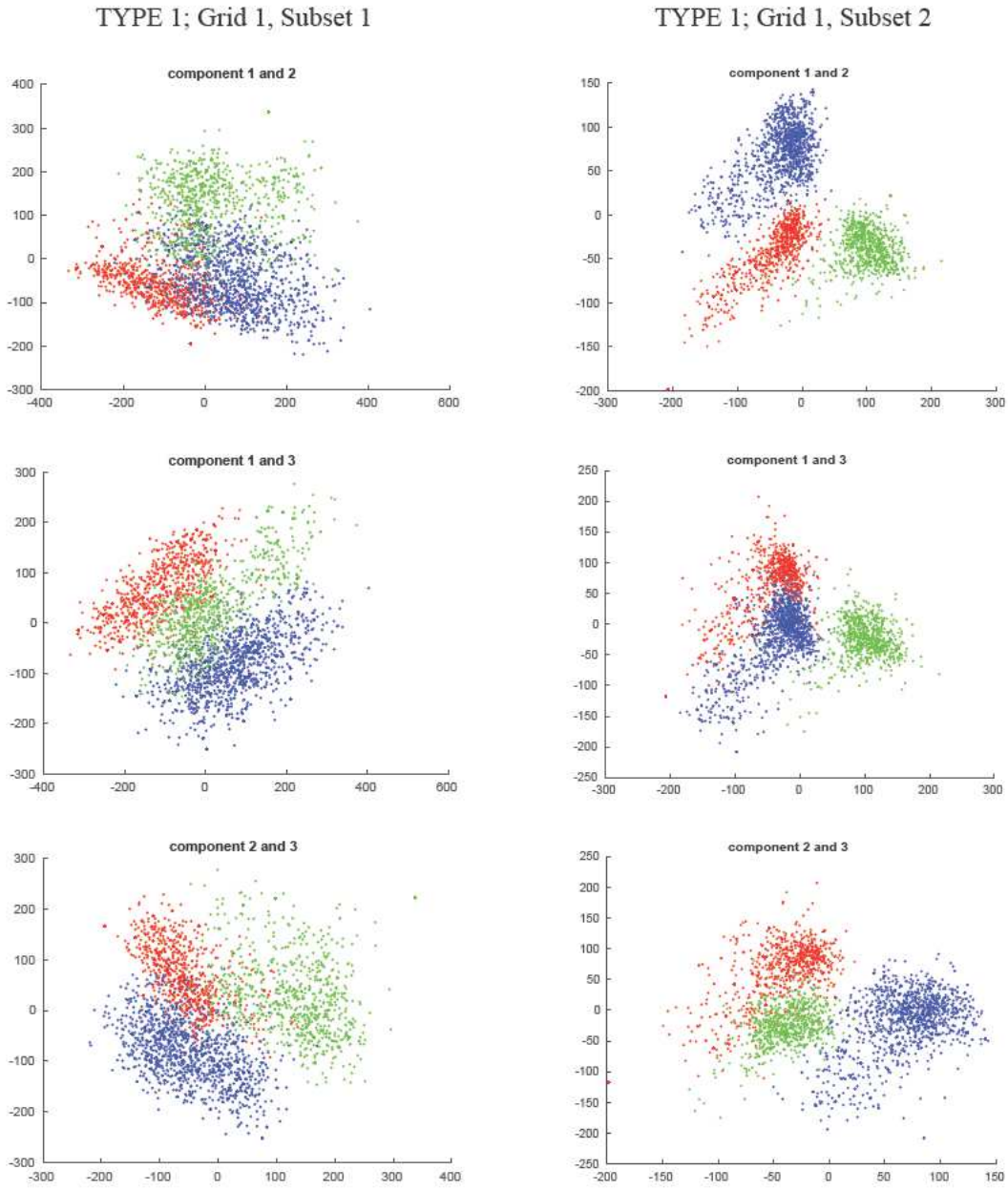

Figure 41: The PCA visualization of the TYPE 1 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere, but within the 2 $\mathring{A}$ distance from it.

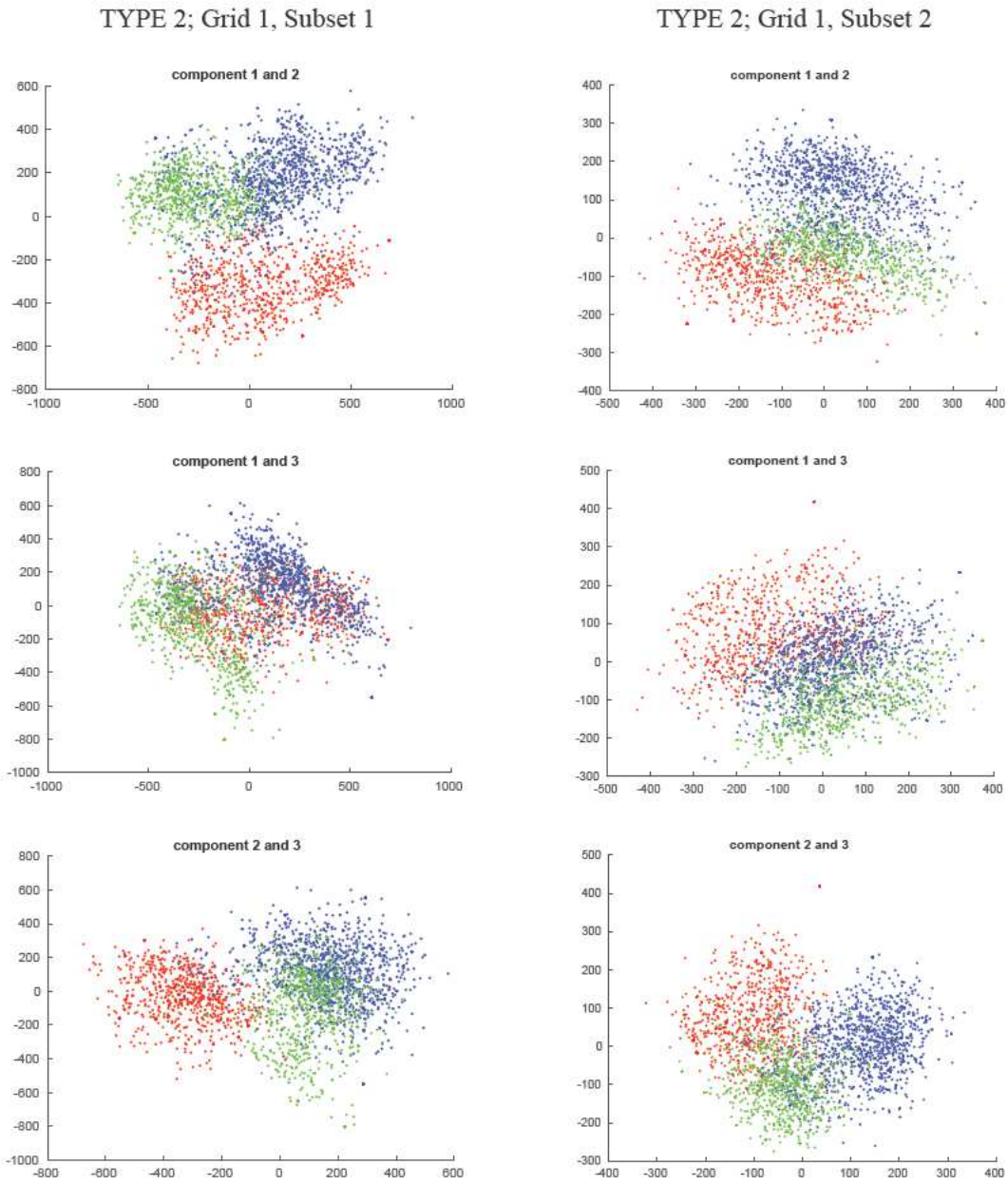

Figure 42: PCA visualization of TYPE 2 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for all points in the subsets.

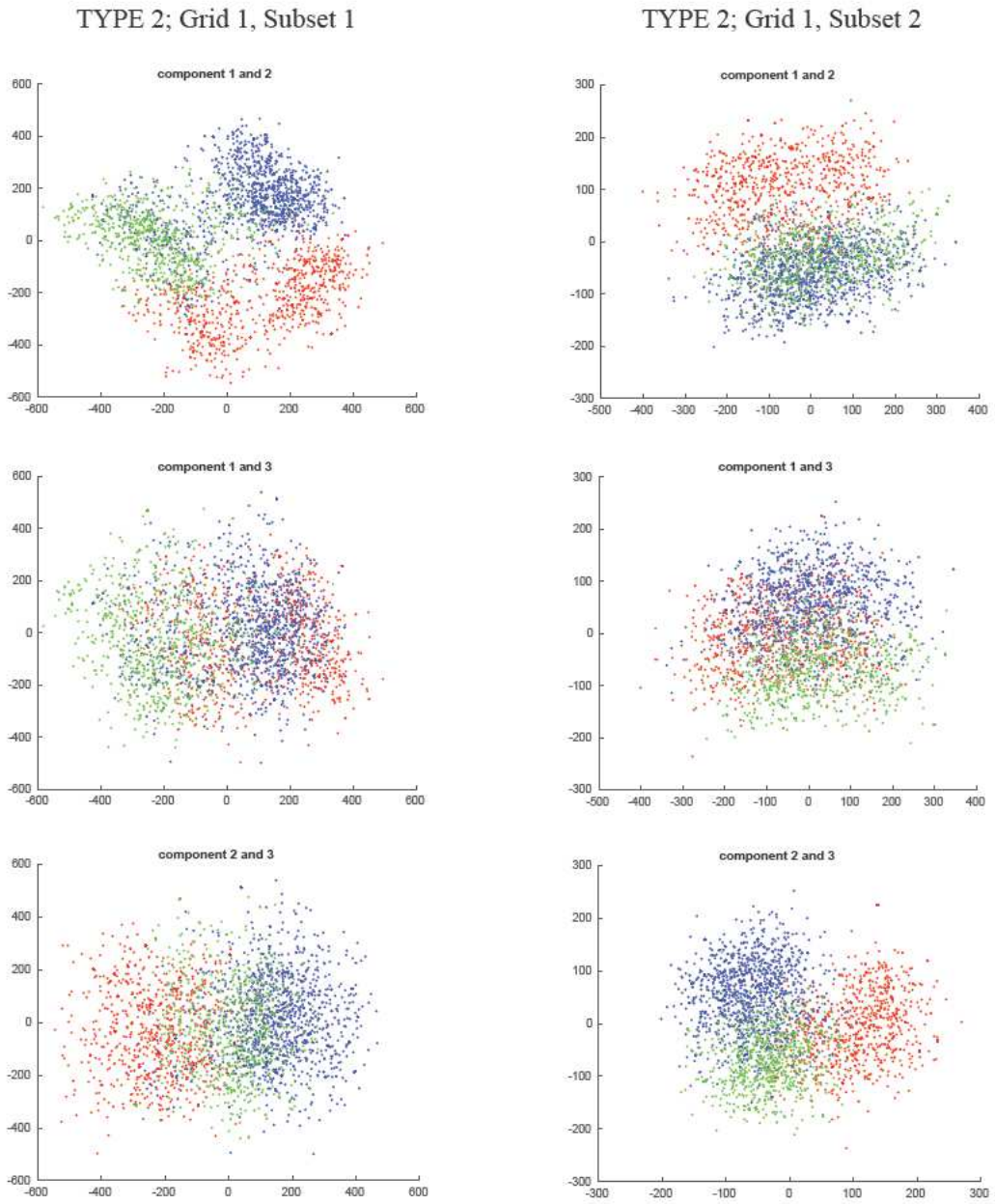

Figure 43: PCA visualization of TYPE 2 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned inside the Van der Waals sphere.

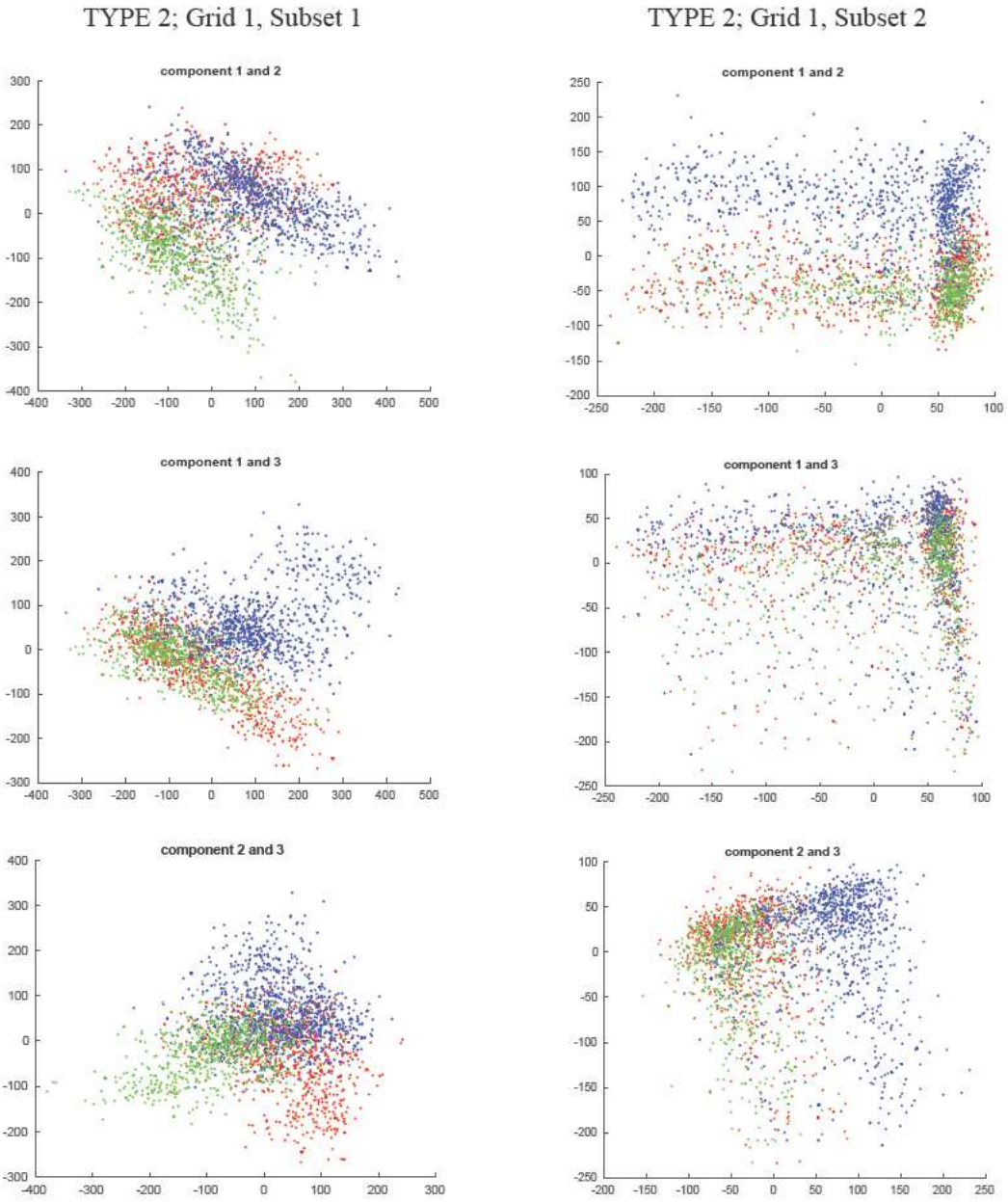

Figure 44: PCA visualization of TYPE 2 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere, but within the 2 $\mathring{A}$ distance from it.

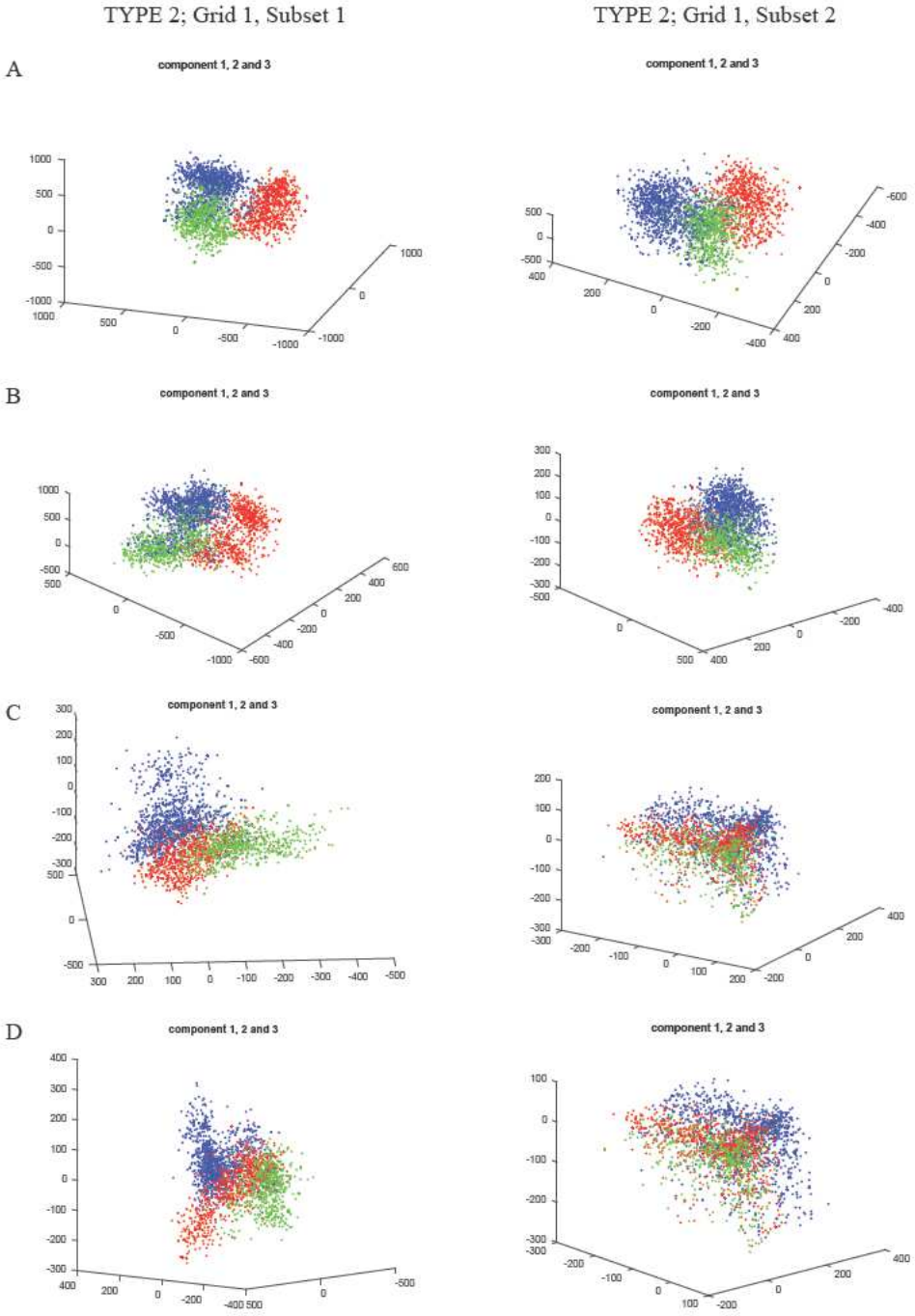

Figure 45: PCA visualization of TYPE 2 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. A: Results for all points in the subsets. B: Results for points positioned inside the Van der Waals sphere. C: Results for points positioned outside the Van der Waals sphere. D: Results for points positioned outside the Van der Waals sphere, but within the 2 $\mathring{A}$ distance from it.

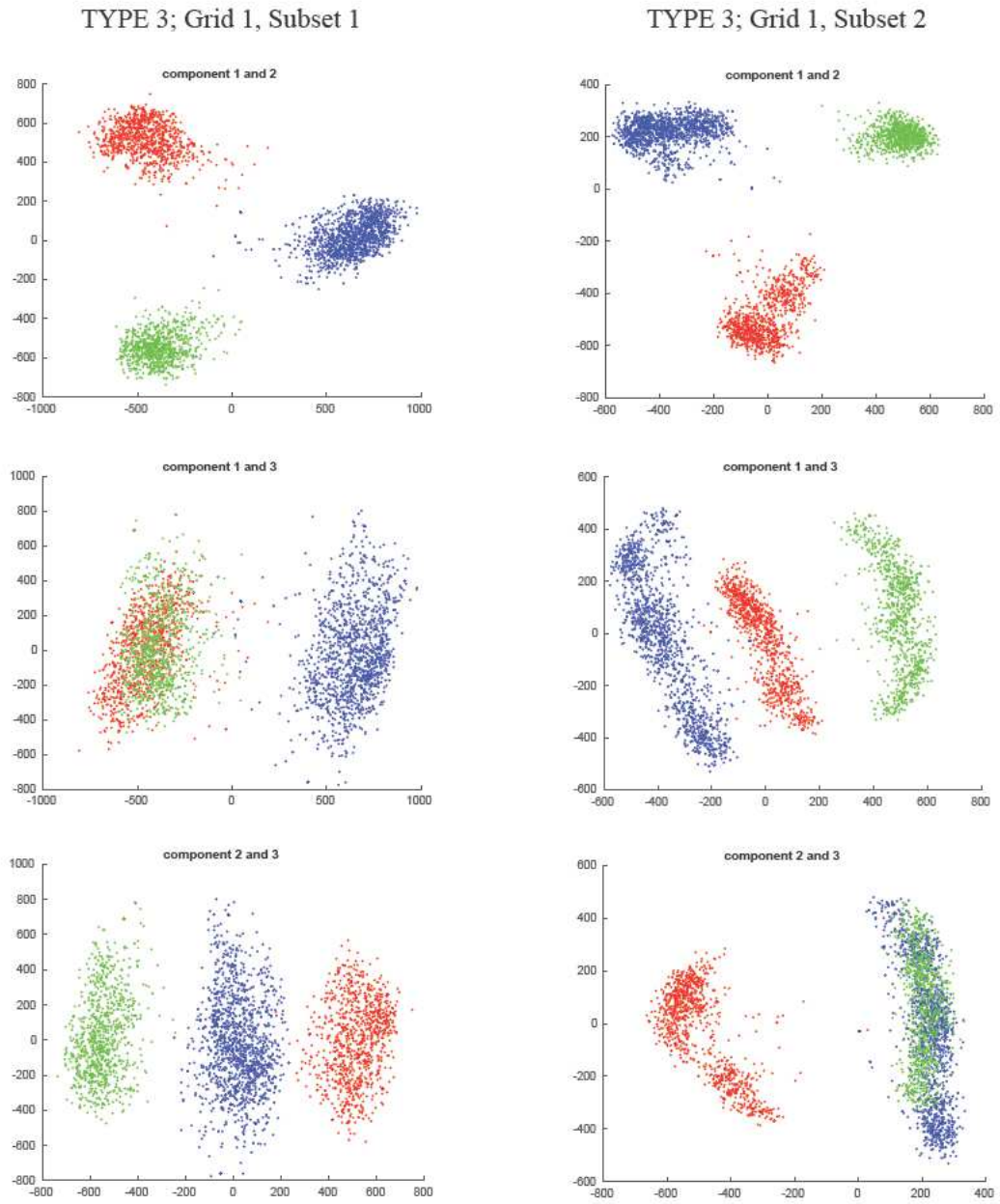

Figure 46: PCA visualization of the TYPE 3 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for all points in the subsets.

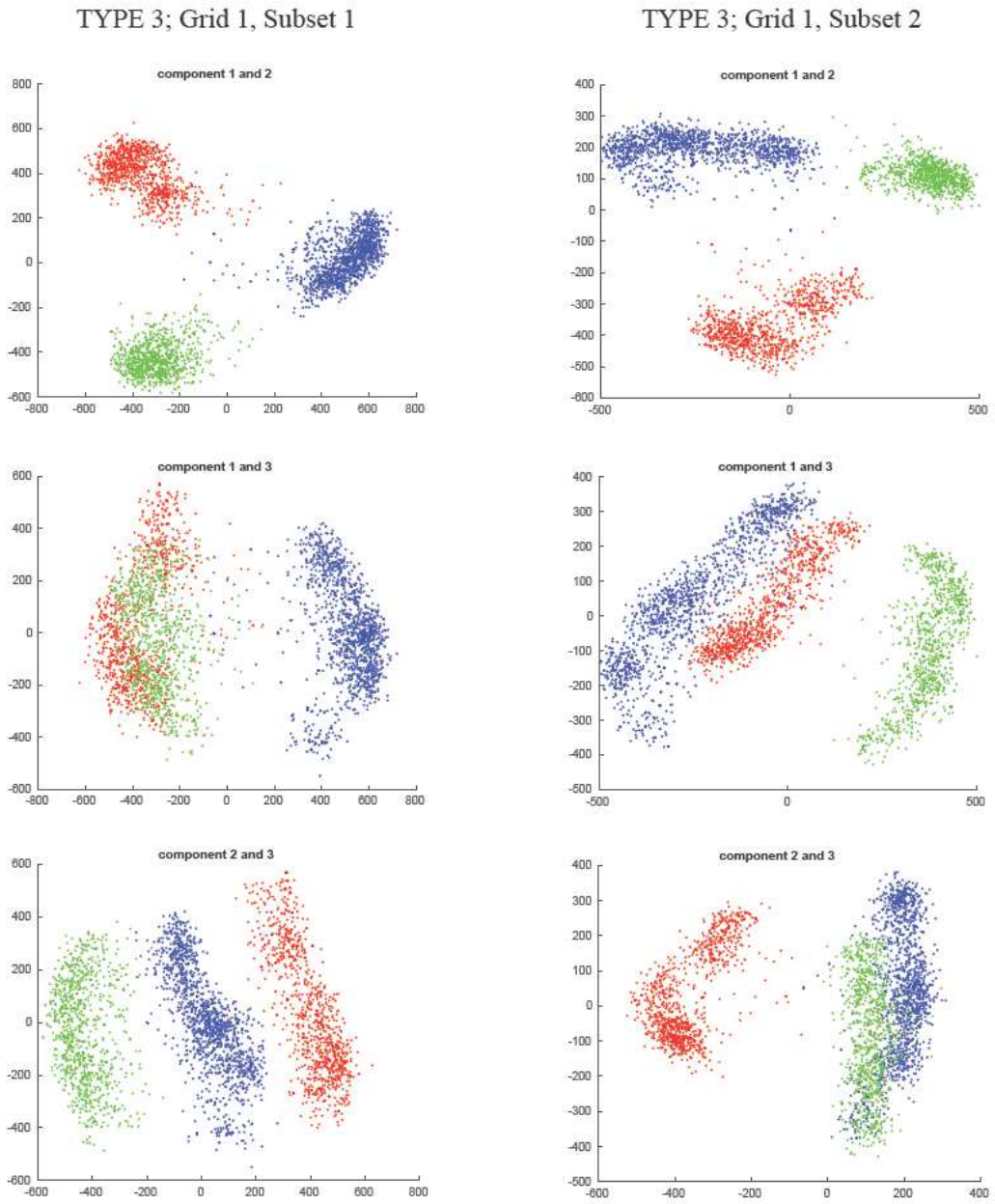

Figure 47: PCA visualization of the TYPE 3 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned inside the Van der Waals sphere.

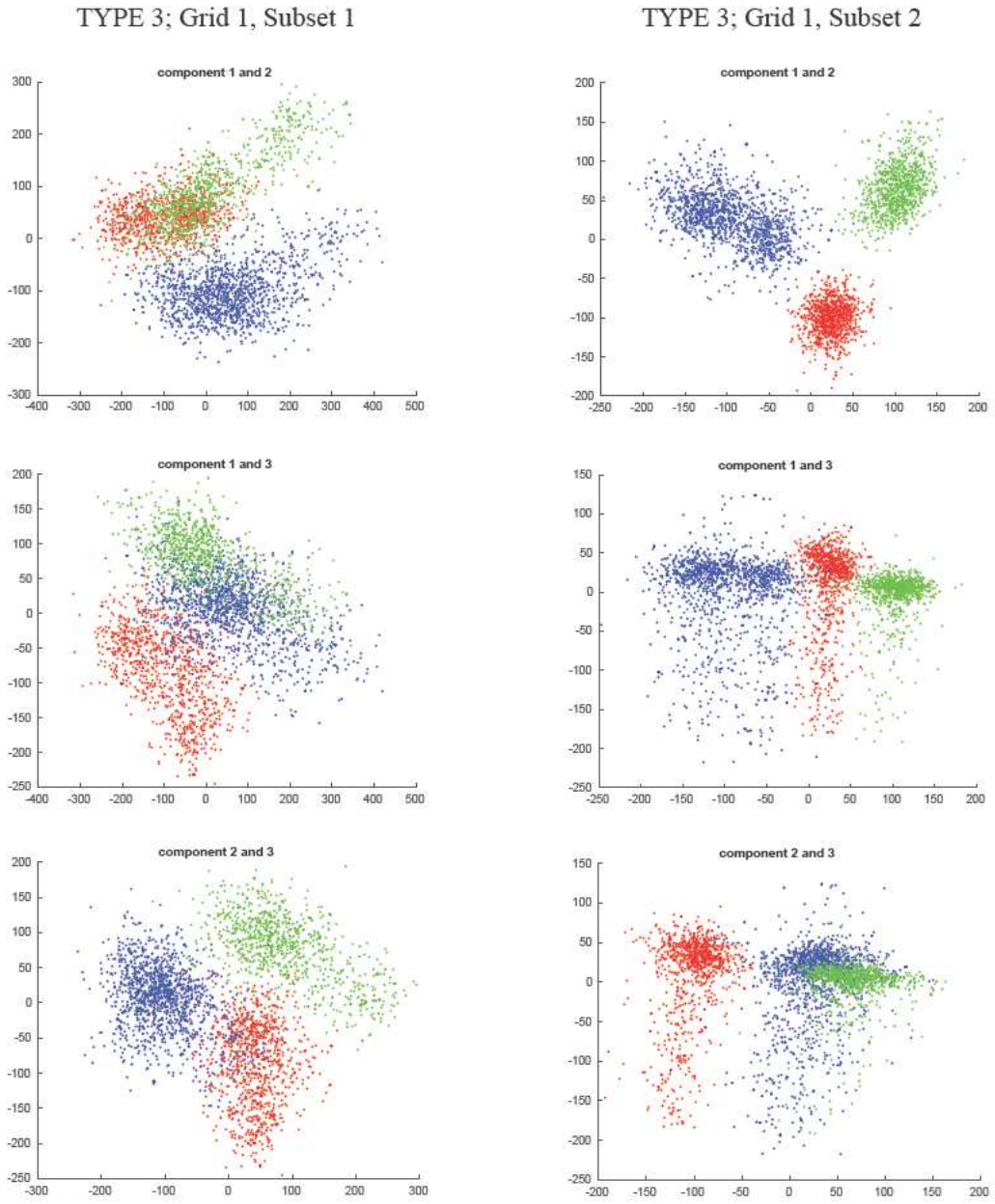

Figure 48: PCA visualization of the TYPE 3 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere, but within the 2 $\mathring{A}$ distance from it.

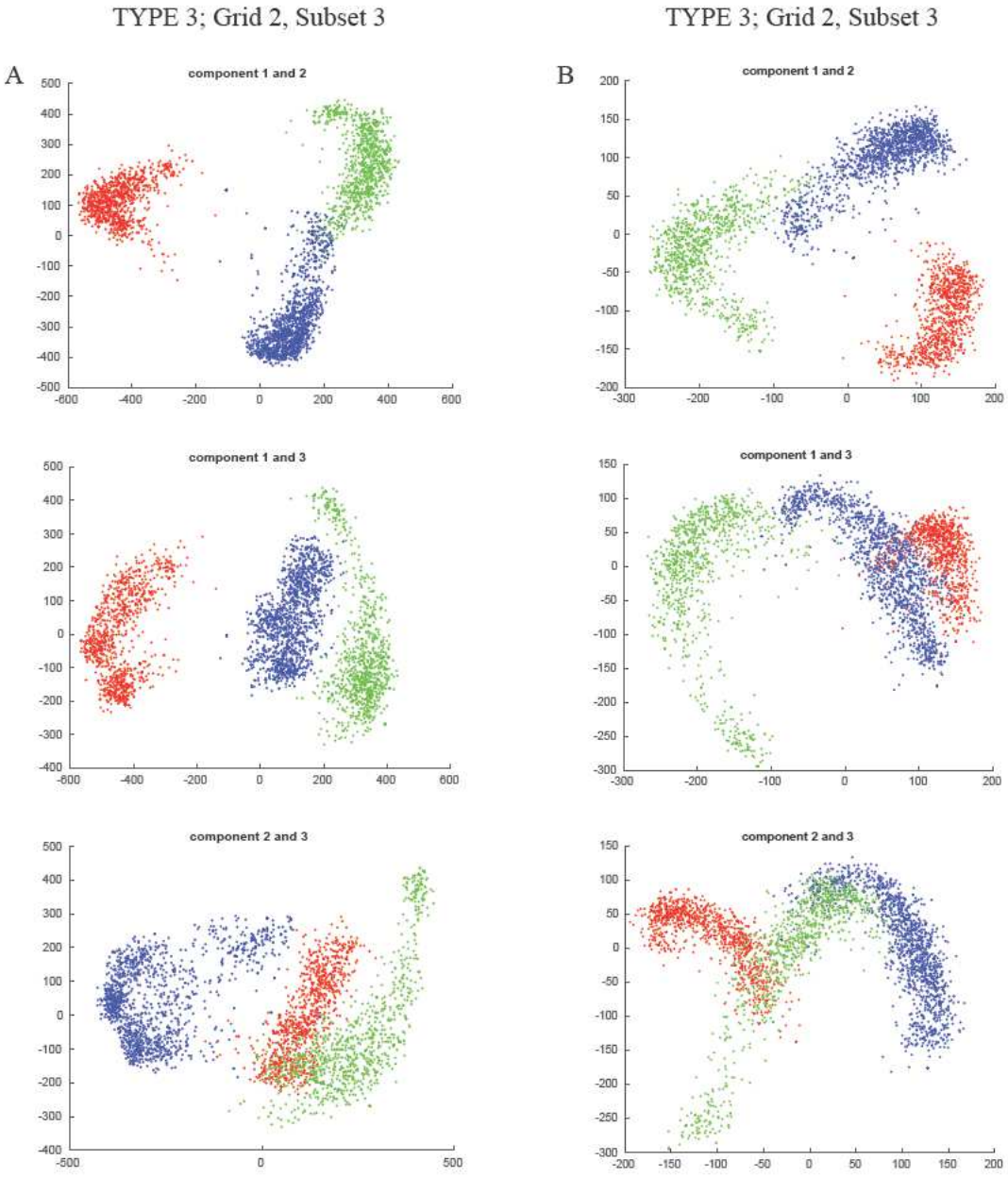

Figure 49: PCA visualization of the TYPE 3 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. A: Results for all points in the subsets. B: Results for points positioned inside the Van der Waals sphere.

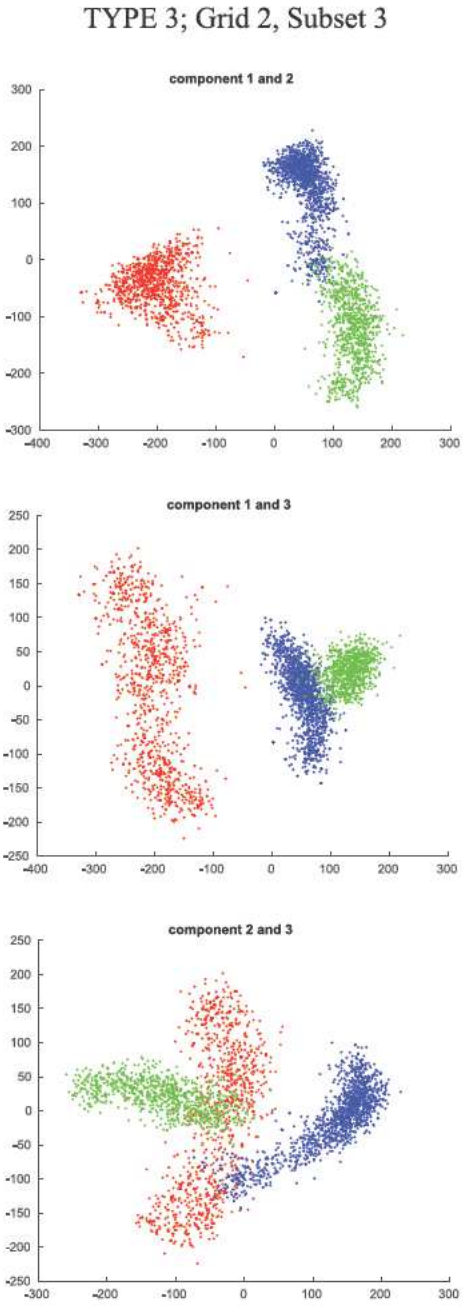

Figure 50: PCA visualization of the TYPE 3 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere, but within the 2 $\mathring{A}$ distance from it.

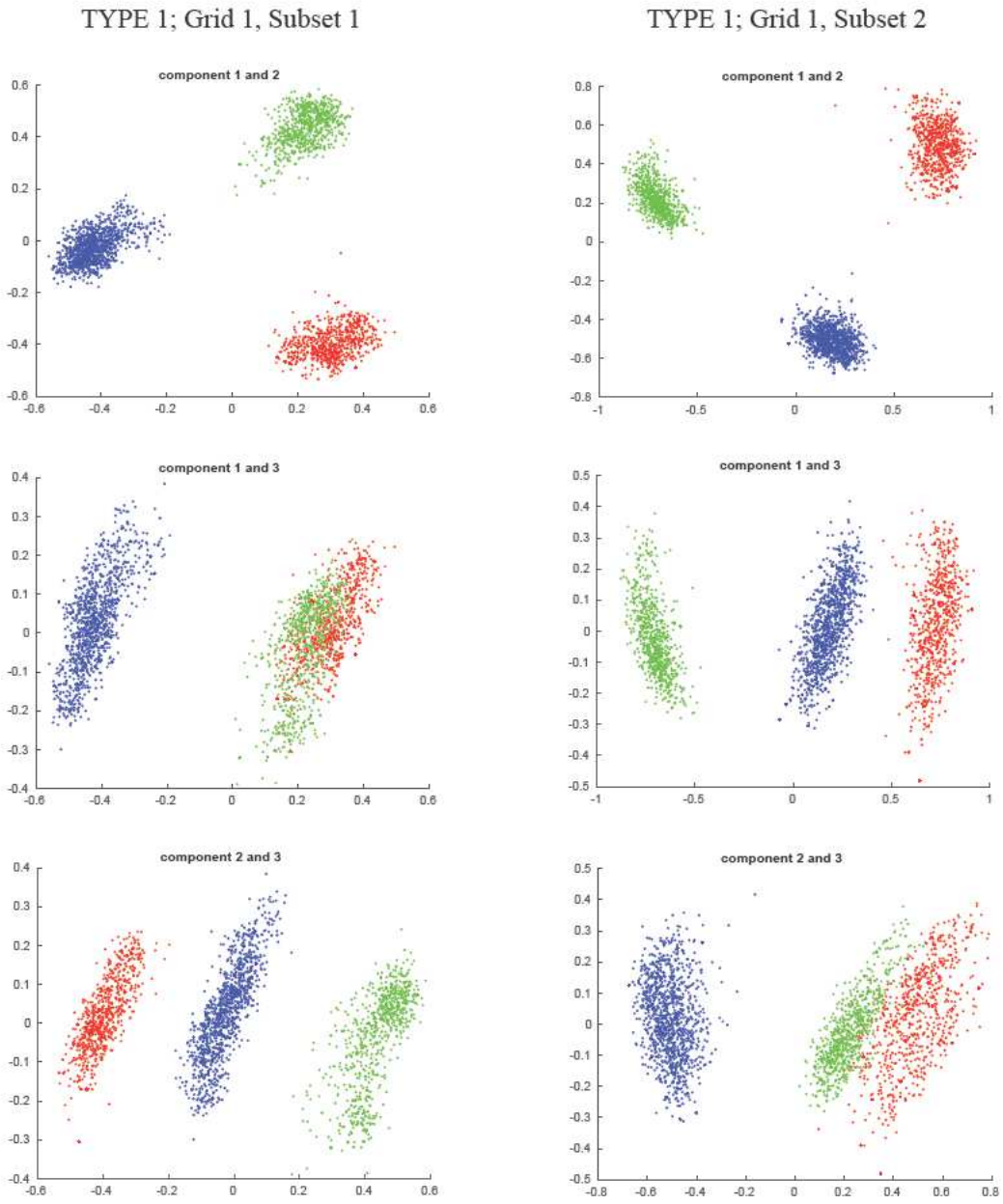

Figure 51: The NLPCA visualization of the TYPE 1 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned inside the Van der Waals sphere.

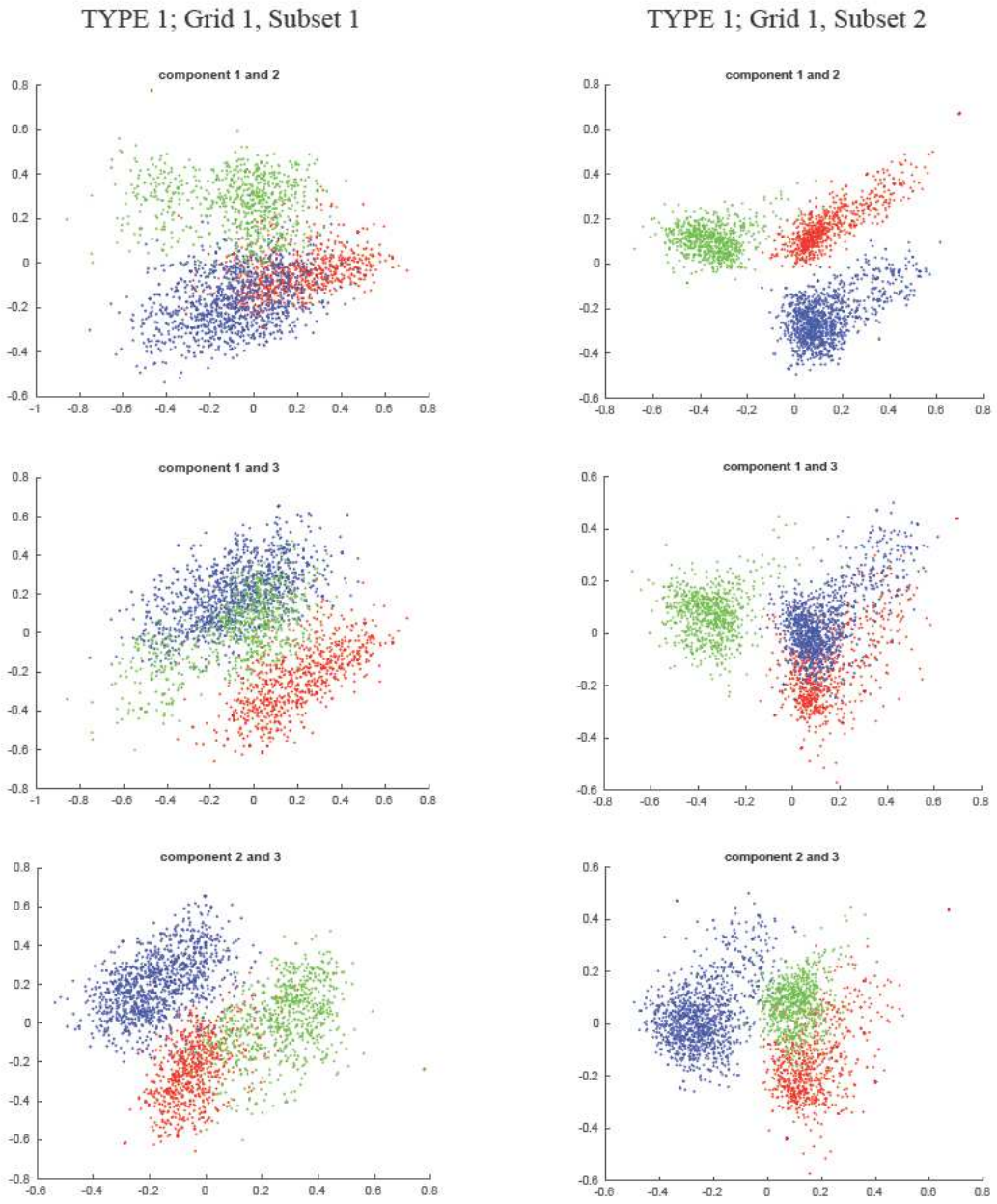

Figure 52: The NLPCA visualization of the TYPE 1 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere.

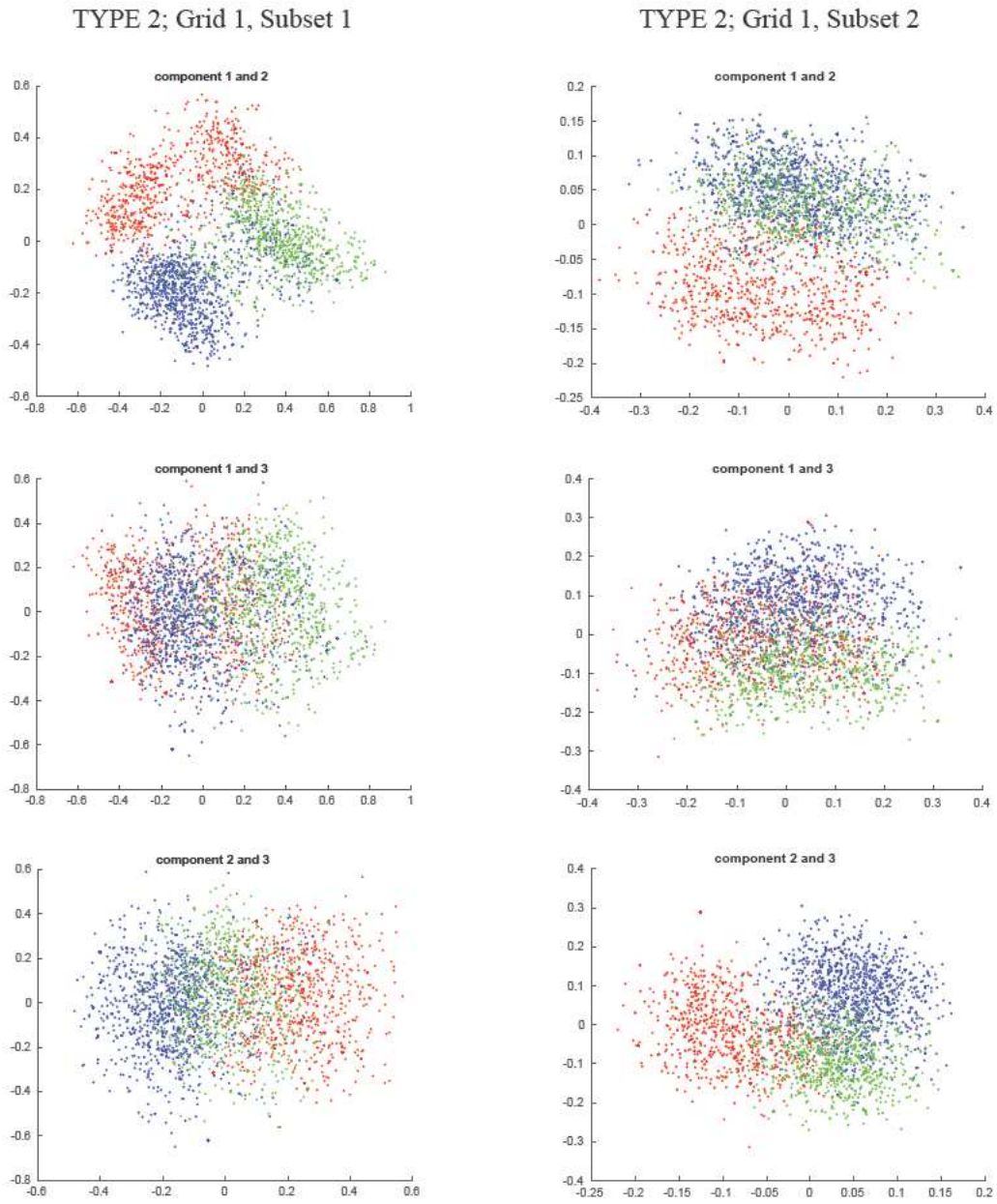

Figure 53: The NLPCA visualization of the TYPE 2 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned inside the Van der Waals sphere.

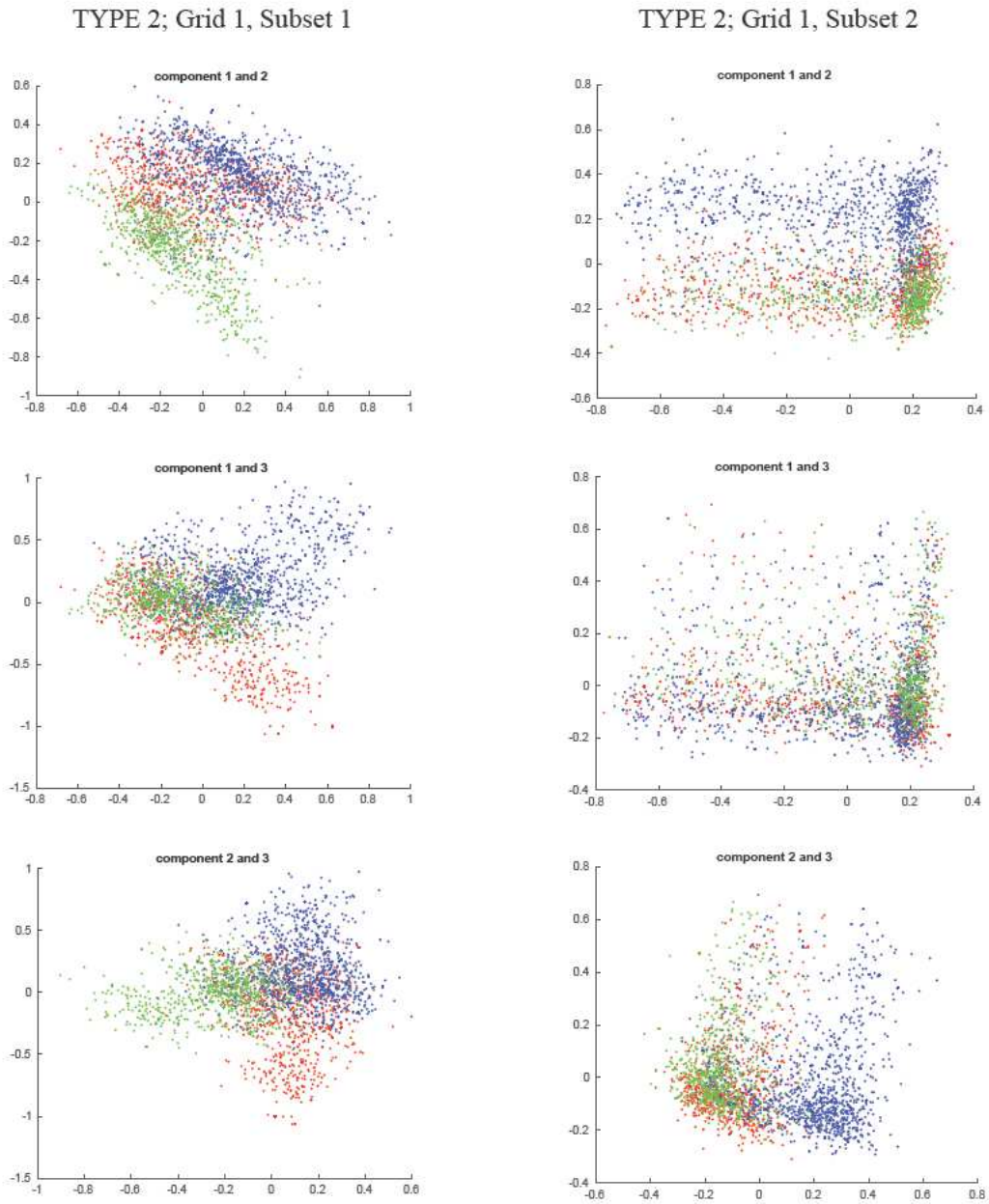

Figure 54: The NLPCA visualization of the TYPE 2 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere.

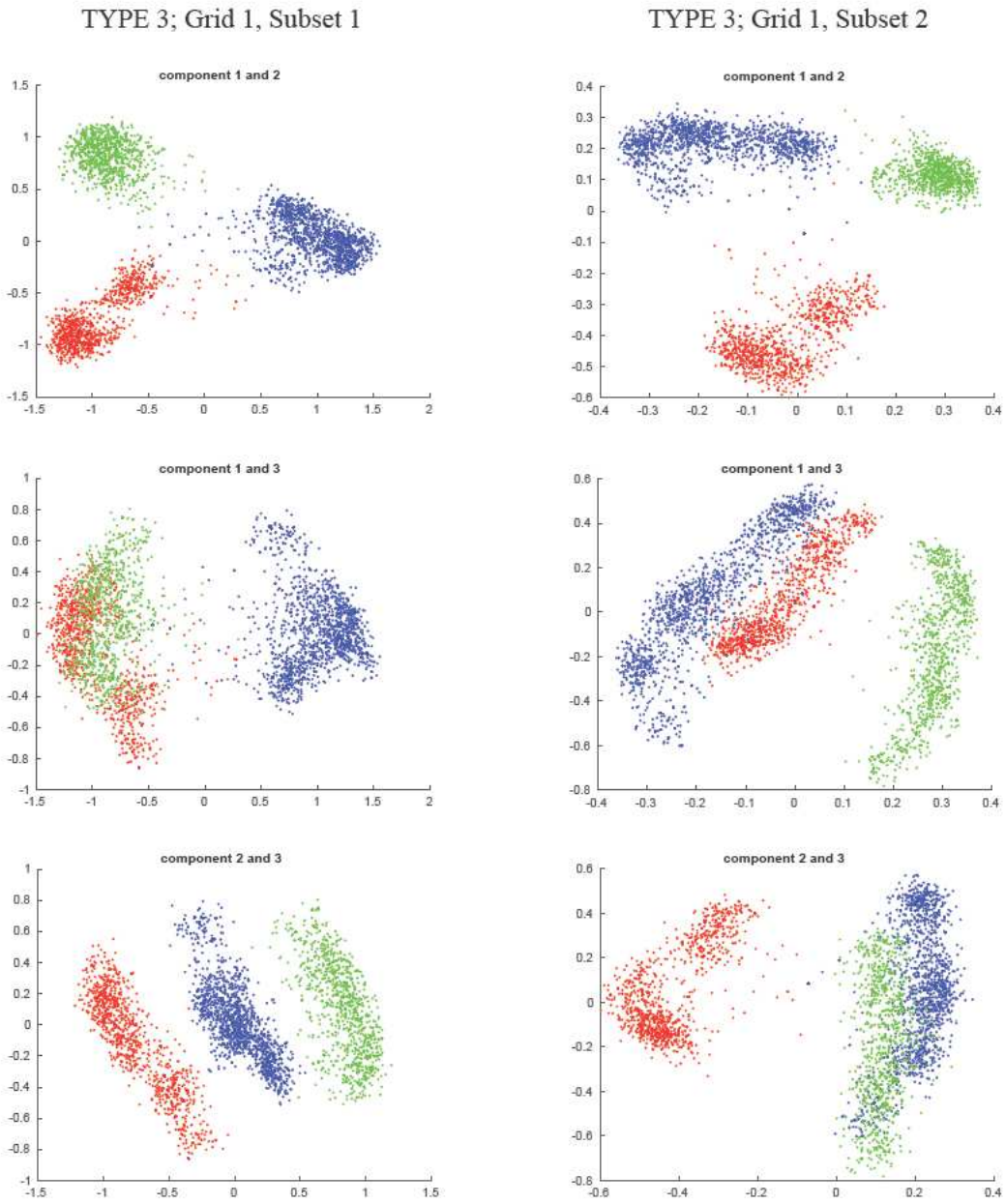

Figure 55: The NLPCA visualization of the TYPE 3 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned inside the Van der Waals sphere.

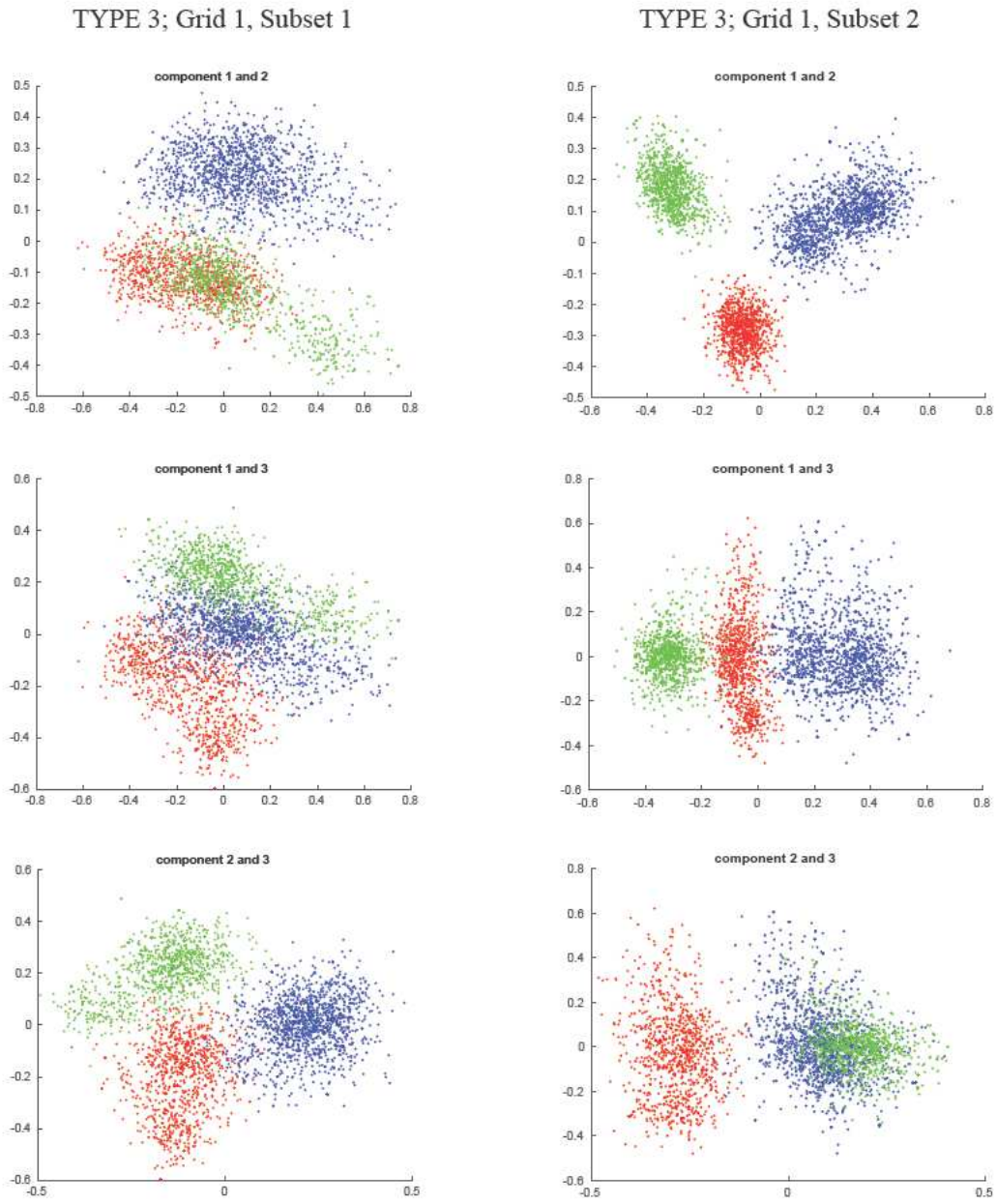

Figure 56: The NLPCA visualization of the TYPE 3 models of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned outside the Van der Waals sphere.

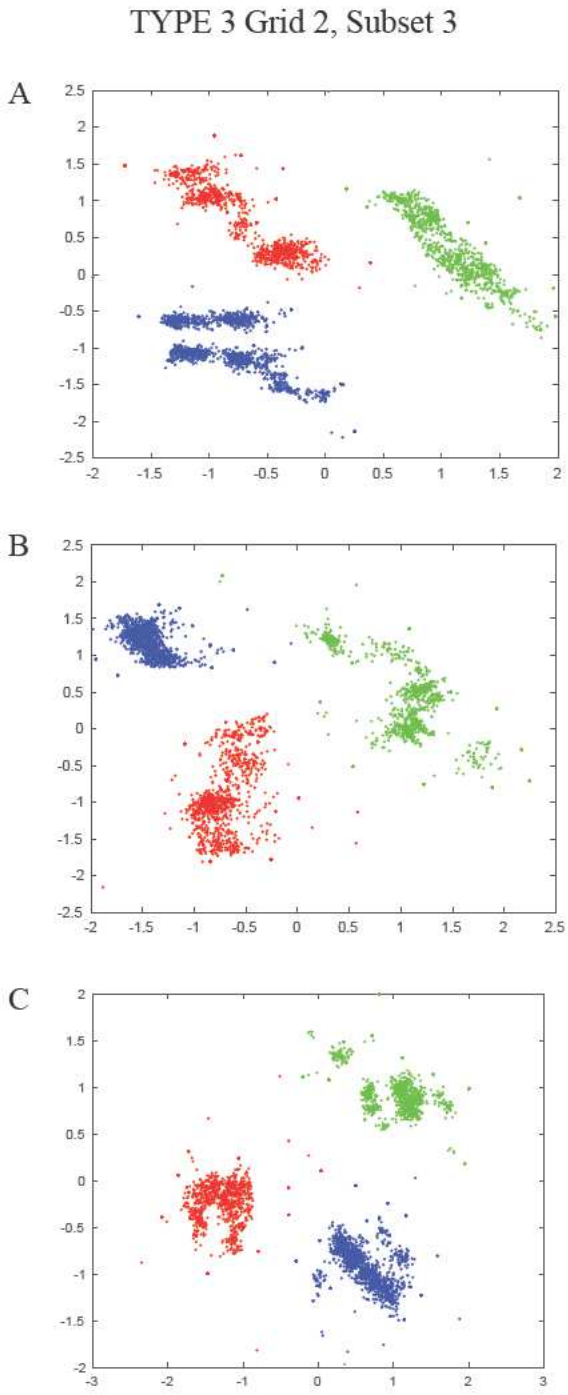

Figure 57: GPLVM visualization of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. A: Results for points positioned inside the Van der Waals sphere. B: Results for points positioned outside the Van der Waals sphere. C: Results for points positioned outside the Van der Waals sphere, but within the 2 $\AA$ distance from it.

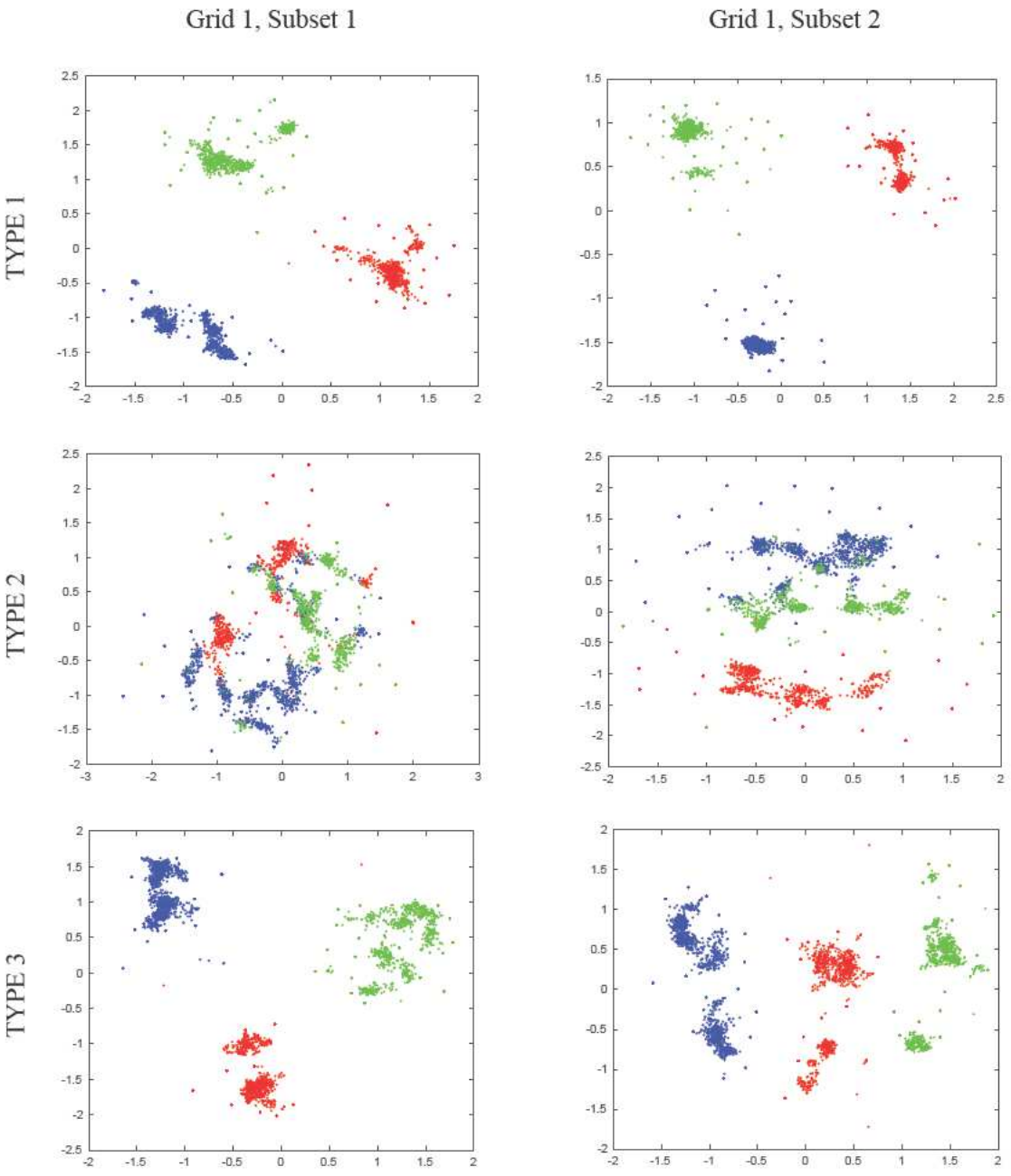

Figure 58: GPLVM visualization of the HLA class 1 proteins: red dots for the HLA-A subclass, blue dots for the HLA-B subclass and green dots for the HLA-C subclass. Results for points positioned inside the Van der Waals sphere.

			Subset 1	Subset 2
TYPE 1 models	All points	component 1	16.1	19.7
		component 2	13.1	18.8
		component 3	4.6	2.2
		sum	33.8	40.7
	Points inside the	component 1	23.8	26.7
	Van der Waals sphere	component 2	19.0	24.2
		component 3	3.9	3.0
		sum	46.7	53.9
	Points outside the	component 1	8.5	10.7
	Van der Waals sphere	component 2	6.5	7.2
		component 3	5.9	5.6
		sum	20.9	23.5
	Points outside the	component 1	8.7	11.7
	Van der Waals sphere	component 2	6.9	7.7
	within 2 $\mathring{A}$ distance from it	component 3	5.4	6.1
		sum	21.0	25.5
TYPE 2 models	All points	component 1	8.7	5.5
		component 2	6.2	4.0
		component 3	5.2	3.7
		sum	20.1	13.2
	Points inside the	component 1	10.7	9.2
	Van der Waals sphere	component 2	8.5	4.5
		component 3	7.7	4.0
		sum	26.9	17.7
	Points outside the	component 1	12.1	10.0
	Van der Waals sphere	component 2	6.7	6.8
		component 3	5.0	4.7
		sum	23.8	21.5
	Points outside the	component 1	12.7	10.7
	Van der Waals sphere	component 2	6.0	7.1
	within 2 $\mathring{A}$ distance from it	component 3	5.0	5.0
		sum	23.7	22.8
TYPE 3 models	All points	component 1	19.1	18.3
		component 2	12.0	14.6
		component 3	4.3	6.8
		sum	35.4	39.7
	Points inside the	component 1	27.3	24.1
	Van der Waals sphere	component 2	16.9	18.4
		component 3	6.9	9.1
		sum	51.1	51.6
	Points outside the	component 1	9.3	10.2
	Van der Waals sphere	component 2	6.4	7.3
		component 3	4.1	3.2
		sum	19.8	20.7
	Points outside the	component 1	9.7	11.2
	Van der Waals sphere	component 2	6.5	7.8
	within 2 $\mathring{A}$ distance from it	component 3	4.4	3.6
		sum	20.6	22.6

Table 34: The percentage of the variance explained by the first three PCA components.

			Subset 3
TYPE 3 models	All points	component 1	21.6
		component 2	13.9
		component 3	5.2
		sum	40.7
	Points inside the	component 1	23.1
	Van der Waals sphere	component 2	11.4
		component 3	7.2
		sum	41.7
	Points outside the	component 1	22.7
	Van der Waals sphere	component 2	14.1
		component 3	5.8
		sum	42.6
	Points outside the	component 1	21.1
	Van der Waals sphere	component 2	15.9
	within 2 $\mathring{A}$ distance from it	component 3	5.4
		sum	42.4

Table 35: The percentage of variance explained by the first three PCA components.

			Subset 1	Subset 2
TYPE 1 models	Points inside the	component 1	24.0	26.5
	Van der Waals sphere	component 2	18.7	23.4
		component 3	4.0	3.2
		sum	46.7	53.1
	Points outside the	component 1	8.6	10.8
	Van der Waals sphere	component 2	6.6	7.4
		component 3	5.9	5.7
		sum	21.1	23.9
TYPE 2 models	Points inside the	component 1	10.8	9.3
	Van der Waals sphere	component 2	8.6	4.5
		component 3	7.8	3.9
		sum	27.2	17.7
	Points outside the	component 1	12.4	10.0
	Van der Waals sphere	component 2	6.8	6.8
		component 3	5.2	4.7
		sum	24.4	21.5
TYPE 3 models	Points inside the	component 1	29.5	24.4
	Van der Waals sphere	component 2	20.3	18.4
		component 3	7.0	9.4
		sum	56.8	52.2
	Points outside the	component 1	9.3	10.2
	Van der Waals sphere	component 2	6.4	7.4
		component 3	4.1	3.2
		sum	19.8	20.8

Table 36: The percentage of the variance explained by the first three NLPCA components.

Points inside the Van der Waals sphere		TYPE 1 models $[K_A, K_B, K_C]$	TYPE 2 models $[K_A, K_B, K_C]$	TYPE 3 models $[K_A, K_B, K_C]$
Subset 1	n=50	1, 1, 1	0.64, 0.59, 0.50	1, 1, 1
	n=100	1, 1, 1	0.80, 0.76, 0.68	1, 1, 1
Subset 2	n=50	1, 1, 1	0.91, 0.63, 056	1, 1, 1
	n=100	1, 1, 1	0.96, 0.81, 0.76	1, 1, 1

Table 37: The separation measurement K for the GPLVM analysis.

Appendix D

	Pfam ID	Protein family	the number of proteins
1	Abhydrolase_9_N	Alpha/beta-hydrolase family N-terminus	103
2	AMNp_N	Bacterial AMP nucleoside phosphorylase N-terminus	102
3	Aldose_epim	Aldose 1-epimerase	64
4	Acetyltransf_8	Acetyltransferase (GNAT) domain	196
5	AAA_22	AAA domain	69
6	AhpC-TSA	AhpC/TSA family	96
7	ATG27	Autophagy-related protein 27	93
8	Aminotran_4	Amino-transferase class IV	69
9	2_5_RNA_ligase2	2'-5' RNA ligase superfamily	94
10	ADH_zinc_N	Zinc-binding dehydrogenase	87
11	ATG2_CAD	Autophagy-related protein 2 CAD motif	95
12	ATG17_like	Autophagy protein ATG17-like domain	83
13	ADC	Acetoacetate decarboxylase (ADC)	71
14	Alkyl_sulf_C	Alkyl sulfatase C-terminal	82
15	AmiS_UreI	AmiS/UreI family transporter	50
16	ADH_zinc_N_2	Zinc-binding dehydrogenase	142
17	2HCT	2-hydroxycarboxylate transporter family	76
18	Asp_Glu_race	Asp/Glu/Hydantoin racemase	44
19	2-Hacid_dh	D-isomer specific 2-hydroxyacid dehydrogenase, catalytic domain	88
20	AAA_35	AAA-like domain	59
21	Arabinose_bd	Arabinose-binding domain of AraC transcription regulator, N-term	105
22	Antibiotic_NAT	Aminoglycoside 3-N-acetyltransferase	126
23	Adenosine_kin	Adenosine specific kinase	97
24	AAL_decarboxy	Alpha-acetolactate decarboxylase	166
25	AAA_21	AAA domain, putative AbiEii toxin, Type IV TA system	78
26	7tm_1	7 transmembrane receptor (rhodopsin family)	63
27	AAA_31	AAA domain	57
28	AAA_26	AAA domain	42
29	AstB	Succinylarginine dihydrolase	57
30	Aconitase_B_N	Aconitate B N-terminal domain	173
31	AAA_16	AAA ATPase domain	124
32	ALIX_LYPXL_bnd	ALIX V-shaped domain binding to HIV	194
33	AbfB	Alpha-L-arabinofuranosidase B (ABFB) domain	100
34	AraC_binding_2	AraC-binding-like domain	115
35	ATG9	Autophagy protein ATG9	122
36	7TM_GPCR_Srw	Serpentine type 7TM GPCR chemoreceptor Srw	85
37	Autophagy_act_C	Autophagocytosis associated protein, active-site domain	103
38	ApbA_C	Ketopantoate reductase PanE/ApbA C terminal	85
39	Amidohydro_1	Amidohydrolase family	56
40	Anticodon_1	Anticodon-binding domain of tRNA ligase	161

Appendix E

	Pfam ID	Protein family	the number of proteins
1	Abhydrolase_9_N	Alpha/beta-hydrolase family N-terminus	103
2	AMNp_N	Bacterial AMP nucleoside phosphorylase N-terminus	102
3	Aldose_epim	Aldose 1-epimerase	64
4	ATG27	Autophagy-related protein 27	93
5	Aminotran_4	Amino-transferase class IV	69
6	2_5_RNA_ligase2	2'-5' RNA ligase superfamily	94
7	ATG2_CAD	Autophagy-related protein 2 CAD motif	95
8	ATG17_like	Autophagy protein ATG17-like domain	83
9	ADC	Acetoacetate decarboxylase (ADC)	71
10	Alkyl_sulf_C	Alkyl sulfatase C-terminal	82
11	AmiS_UreI	AmiS/UreI family transporter	50
12	2HCT	2-hydroxycarboxylate transporter family	76
13	Asp_Glu_race	Asp/Glu/Hydantoin racemase	44
14	2-Hacid_dh	D-isomer specific 2-hydroxyacid dehydrogenase, catalytic domain	88
15	Arabinose_bd	Arabinose-binding domain of AraC transcription regulator, N-term	105
16	Antibiotic_NAT	Aminoglycoside 3-N-acetyltransferase	126
17	Adenosine_kin	Adenosine specific kinase	97
18	AAL_decarboxy	Alpha-acetolactate decarboxylase	166
19	AstB	Succinylarginine dihydrolase	57
20	Aconitase_B_N	Aconitate B N-terminal domain	173
21	AAA_16	AAA ATPase domain	124
22	ALIX_LYPXL_bnd	ALIX V-shaped domain binding to HIV	194
23	AbfB	Alpha-L-arabinofuranosidase B (ABFB) domain	100
24	ATG9	Autophagy protein ATG9	122
25	7TM_GPCR_Srw	Serpentine type 7TM GPCR chemoreceptor Srw	85
26	Autophagy_act_C	Autophagocytosis associated protein, active-site domain	103
27	ApbA_C	Ketopantoate reductase PanE/ApbA C terminal	85
28	Amidohydro_1	Amidohydrolase family	56
29	Anticodon_1	Anticodon-binding domain of tRNA ligase	161
30	2-ph_phosp	2-phosphosulpholactate phosphatase	141
31	4HBT_2	Thioesterase-like superfamily	74
32	5-nucleotidase	5'-nucleotidase	97
33	5TM-5TMR_LYT	5TMR of 5TMR-LYT	71
34	AA_kinase	Amino acid kinase family	128
35	ABC2_membrane_4	ABC-2 family transporter protein	48
36	AbLIM_anchor	Putative adherens-junction anchoring region of AbLIM	51
37	AcetylCoA_hyd_C	Acetyl-CoA hydrolase/transferase C-terminal domain	182
38	Acetyltransf_6	Acetyltransferase (GNAT) domain	141
39	Aconitase_2_N	Aconitate hydratase 2 N-terminus	51
40	Acyl-CoA_ox_N	Acyl-coenzyme A oxidase N-terminal	134
41	Adap_comp_sub	Adaptor complexes medium subunit family	46
42	A_deamin	Adenosine-deaminase (editase) domain	100
43	AdoMet_Synthase	S-adenosylmethionine synthetase (AdoMet synthetase)	102
44	ADP_PFK_GK	ADP-specific Phosphofructokinase/Glucokinase conserved region	82
45	AIP3	Actin interacting protein 3	97
46	ALG3	ALG3 protein	155
47	Alginate_lyase2	Alginate lyase	67
48	Alph_Pro_TM	Putative transmembrane protein (Alph_Pro_TM)	71
49	Amidase_3	N-acetylmuramoyl-L-alanine amidase	71
50	Aminotran_1_2	Aminotransferase class I and II	48

	Pfam ID	Protein family	the number of proteins
51	AMP-binding	AMP-binding enzyme	145
52	ANAPC8	Anaphase promoting complex subunit 8 / Cdc23	72
53	AnfO_nitrog	Iron only nitrogenase protein AnfO (AnfO_nitrog)	41
54	ANF_receptor	Receptor family ligand binding region	74
55	Anth_synt_I_N	Anthranilate synthase component I, N terminal region	56
56	AOX	Alternative oxidase	121
57	ApbA	Ketopantoate reductase PanE/ApbA	81
58	AP_endonuc_2	Xylose isomerase-like TIM barrel	180
59	APH	Phosphotransferase enzyme family	156
60	Aph-1	Aph-1 protein	54
61	Apolipoprotein	Apolipoprotein A1/A4/E domain	117
62	ApoO	Apolipoprotein O	115
63	Apyrase	Apyrase	56
64	Arabino_trans_C	EmbC C-terminal domain	70
65	Arb1	Argonaute siRNA chaperone (ARC) complex subunit Arb1	54
66	ARPC4	ARP2/3 complex 20 kDa subunit (ARPC4)	45
67	ArsD	Arsenical resistance operon protein ArsD	56
68	ASF1_hist_chap	ASF1 like histone chaperone	84
69	AsmA_2	AsmA-like C-terminal region	73
70	Asparaginase	Asparaginase, N-terminal	119
71	Asp_Arg_Hydrox	Aspartyl/Asparaginyl beta-hydroxylase	131
72	Aspzincin_M35	Lysine-specific metallo-endopeptidase	71
73	ASXH	Asx homology domain	67
74	ATPgrasp_YheCD	YheC/D like ATP-grasp	107
75	ATP-sulfurylase	ATP-sulfurylase	188
76	ATP-synt_10	ATP10 protein	42
77	Autoind_bind	Autoinducer binding domain	101
78	Auxin_canalis	Auxin canalisation	48
79	ApoL	Apolipoprotein L	44
80	AIRS_C	AIR synthase related protein, C-terminal domain	132
81	AHSA1	Activator of Hsp90 ATPase homolog 1-like protein	93
82	AXH	Ataxin-1 and HBP1 module (AXH)	50
83	ATP11	ATP11 protein	129
84	AIPR	AIPR protein	65
85	Arv1	Arv1-like family	82
86	Abi_2	Abi-like protein	197
87	AsnA	Aspartate-ammonia ligase	170
88	Apt1	Golgi-body localisation protein domain	160
89	Ada3	Histone acetyltransferases subunit 3	86
90	AMOP	AMOP domain	49
91	ArdA	Antirestriction protein (ArdA)	73
92	ASD1	Apx/Shroom domain ASD1	47
93	ACC_central	Acetyl-CoA carboxylase, central region	124
94	AAR2	AAR2 protein	107
95	Amj	Alternate to MurJ	51
96	14-3-3	14-3-3 protein	171
97	AbiEii	Nucleotidyl transferase AbiEii toxin, Type IV TA system	170
98	Anillin	Cell division protein anillin	91
99	AbrB	Transition state regulatory protein AbrB	148
100	AUX_IAA	AUX/IAA family	105

Appendix E

Persistence exponent	Positive	Negative	Average
simple moving average	584 (6.03%)	580 (5.98%)	854 (8.81%)
modified moving average	341 (3.52%)	301 (3.11%)	530 (5.47%)
exponential moving average	419 (4.30%)	360 (3.71%)	567 (5.85%)

Table 38: The persistence exponent descriptors misclassification error obtained for 553-dimensional sets (each dimension corresponds with one scale obtain from AAIndex database). Once differentiated time series, moving average calculated after differentiation.

	sum of the maximum positive pick / maximum segment length	sum of the maximum negative pick / maximum segment length
once differentiated time series		
simple moving average	923 (9.52%)	949 (9.79%)
modified moving average	900 (9.29%)	1194 (12.32%)
exponential moving average	1211 (12.49%)	1312 (13.54%)

Table 39: The misclassification error obtained with Linear Discriminant Analysis classifier for 553-dimensional sets of the sum total of the pick descriptors. Each dimension corresponds with one scale obtain from AAIndex database. Moving average calculation were performed after differentiation.

Sum total of the pick	sum of all positive picks / sum of the segment lengths	sum of all negative picks / sum of the segment lengths
once differentiated time series		
simple moving average	405 (4.18%)	411 (4.24%)
modified moving average	341 (3.52%)	342 (3.53%)
exponential moving average	340 (3.51%)	318 (3.28%)

Table 40: The misclassification error obtained with Linear Discriminant Analysis classifier for 553-dimensional sets of the sum total of the pick descriptors. Each dimension corresponds with one scale obtain from AAIndex database. Moving average calculation were performed after differentiation.