laSalle
UNIVERSITAT RAMON LLULL

Escola Tècnica Superior d'Enginyeria La Salle

Treball Final de Màster

Màster Universitari en Enginyeria de Xarxes i Telecomunicacions

Improved classification of genomic data by Gram-Schmidt feature selection

Alumne
Jose Maria Marco Reales

Professor Ponent
Xavier Vilasis Cardona

ACTA DE L'EXAMEN DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Jose Maria Marco Reales

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Improved classification of genomic data by Gram-Schmidt feature selection

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL

VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

ABSTRACT

This work explains some important aspects in the world of the neural networks, as the classification methods and the procedures of feature selection. Moreover, there is a practical part that consists in creating a program that provides us the useful information to do the classification. It is important to consider that in this thesis we have touched some biochemical aspects because the program has been designed for bioinformatics applications. Therefore the first part of the work consists in an introduction to genomics, namely, relations of enzymes and amino-acids. Finally all the results obtained in the work have been reported and discussed.

INDEX

INTRODUCTION

All this work is about the world of the neural networks. The neural networks have become one of the first options in many applications.

Most of these applications are related with bioinformatics and tasks of diagnosis, recognition. Therefore this work is focused in this field and, due to this, it is very important to clarify some ideas about genomes, enzymes and amino-acids.

According to this, the first part of this work is focused in explaining some ideas about genomics and bioinformatics. Later we enter more deeply in the main subject of the thesis which is related with the statistical classification.

Inside the statistical classification we have a lot of aspects to consider. This classification can be seen like a chain of various stages that are very important to obtain the final goal. We will be focused mainly in the task of ranking the features that define a problem. Inside this field we can find a lot of options to do this ranking but the one that has been chosen is based on the Gram-Schmidt orthogonalization.

We have emphasize so much in this kind of orthogonalization because it is the base of the method that we have chosen for ranking the features, in this case, the biochemical properties of some amino-acids.

Finally we present the practical part of the thesis aimed at ranking the biochemical properties of the amino-acids of each selected enzyme according to their influence in the final result.

1.- PROJECT DESCRIPTION

1.1.- EXPLANATION

The statement of the thesis is to create a program which is able to process two files, containing information about amino acid sequences and their biochemical properties, and generate from these files two new documents containing data information for svm (support vector machine) or ls-svm classifier.

The steps that have been followed are the next ones:

1. First of all it is necessary to process the text document *AAIndexes1.txt*, containing the biochemical characteristics of the amino-acids. We have skipped all the rows in this document that contains the value ‘NA’ because those rows do not contain useful information for our classification. All the wrongs detected have been reported in a text file called *log.txt.*

2. The next step has been to process the text document *Targets-lsw.txt*, that contains the information about the amino-acids. In this step it is necessary to skip all the rows containing the value ‘X’ because, as happened at the step before, it does not contain any useful information. The different classes of amino-acids detected and the errors detected in this document have been reported in the file *log.txt.*

3. In the file *Targets-lsw.txt* some redundant information appears, this is why the goal of this third step is to remove this redundant information and report them in the document *log.txt.*

4. At the fourth step, a document file called *Target-LSW-params.txt* has been generated. This document contains all the amino-acids with their respective biochemical characteristics. The letters have been substituted by their numerical values.

5. Finally, the Gram-Schmidt orthogonalization method is applied to each class as a variable selection procedure. The documents generated from these steps are the next ones:
    - *Targets-LSW-vars-classname.txt*: With information about what variables have been selected for the class name.
    - *Targets-LSW-classname.txt*: With information about the selected variables from *Target-LSW-params* which best describes class name.

2.- GOAL OF THE THESIS

2.1.- EXPLANATION

After explaining the statement of the thesis we can say that the goal of the thesis is to generate a program capable to process some files and after analysing them give some information about what biochemical properties have a stronger influence on the final results.

This task is very important because the fact of analysing all the properties that appear in the document would be computationally very complex. Then it is very important to find the variables that have more influence in the final result.

For doing this program it has been used a method based in the Gram-Schmidt orthogonalization. This method evolves some tasks of iteration and correlation and as result of this we obtain a text document containing the most important characteristics.

3.- BIOINFORMATICS

3.1.- GENERAL DESCRIPTION

Bioinformatics is the application of information technology and computer science to the field of molecular biology.

Since the beginning this branch of the computer science has been used in genomics and genetics, especially in all that areas that involve large-scale DNA sequencing.

Bioinformatics now entails the creation and advancement of databases, algorithms, computational and statistical techniques, and theory to solve formal and practical problems arising from the management and analysis of biological data.

A lot of information related to molecular biology has been developed these last years due to the developments in genomics and other molecular research technologies.

Some of the most important activities related with the bioinformatics are the following ones:

- Mapping and analyzing of DNA sequences
- Mapping and analyzing of protein sequences
- Aligning different DNA and protein sequences
- Comparing these DNA and protein sequences
- Viewing 3-D models of protein structures

Then, it can be said that the main goal of the bioinformatics is to increase our understanding of biological processes by means of developing and applying computationally intensive techniques to achieve this goal. Some of these computationally intensive techniques can be the following:

- Pattern recognition

- Data mining
- Machine learning algorithms
- Visualization

Mayor research efforts in this field include sequence alignment, gene finding, genome assembly, drug design, drug discovery, protein structure alignment, protein structure prediction, prediction of gene expression and protein-protein interactions, genome-wide association studies and the modelling of evolution.

4.- ENZYMES

4.1.- GENERAL DESCRIPTION

The enzymes are mainly proteins that catalyze chemical reactions. In enzymatic reactions, the molecules at the beginning of the process are called substrates, and the enzyme converts them into different molecules, called the products.

Enzymes are part of a group of organic proteins known as amino acids and are found in all living things. They are as old as life itself. Without enzymes, there would be no life. If enzymes were totally absent, then life would cease. When there is a partial reduction of availability of enzymes, then life is reduced. They are “activists”. All activity of life depends on them, for example the greening of leaves in spring, the ripening of foods, the digestion and absorption of food require enzymes.

Almost all processes in a biological cell (functional basic unit of life) need enzymes to occur at significant rates. Since enzymes are selective for their substrates and speed up only a few reactions from among many possibilities, the set of enzymes made in a cell determines which metabolic pathways occur in that cell.

Like all catalysts, enzymes work by lowering the activation energy for a reaction, thus dramatically increasing the rate of the reaction. As with all catalysts, enzymes are not consumed by the reactions they catalyze, nor do they alter the equilibrium of those reactions. However, enzymes do differ from most other catalysts by being much more specific. Enzymes are known to catalyze about 4000 biochemical reactions.

Then, it can be said that the enzymes perform the action, but do not become part of the action. For understanding this we can mention the fire used to cook some food. The fire cooks the food but does not become part of it.

Some examples involving the catalytic action of enzymes are shown below:

- You add fertilizer to the soil around a tomato seedling. Enzymes in the soil help supply the available nutrients to the seedling’s roots.
- You have green tomatoes that need to be ripened. You put them on a windowsill where the warmth of the sun causes the enzymes to function and turn the tomatoes into a nutritious red.
- You out food in your mouth and chew. Specific enzymes in your saliva start the digestive process, which is continued in your stomach by others. They break down the starches, fats and proteins that your body needs to live.

Enzymes can be considered as small biochemical digesters. They have the individual power to break apart vitamins, minerals, proteins, carbohydrates, fats, etc. and make them absorbable. Processed enzymes are able to exert a unique reaction that does more that dissolve accumulated toxins and prepare them for elimination.

Enzyme activity can be affected by other molecules. Inhibitors are molecules that decrease enzyme activity. For example, many drugs and poisons can be considered inhibitors. The enzyme activity can also be affected by the temperature, the chemical environment or the concentration of the substrate.

Some enzymes are used commercially, for example, in the synthesis of antibiotics. In addition, some household products use enzymes to speed up biochemical reactions. For example, enzymes in biological washing powders break down protein or fat stains on clothes, or enzymes in meat tenderizers break down proteins making the meat easier to chew.

In the pool and spa industry, enzymes are used to digest or break down oils from suntan lotions, body lotions, hair products, soaps and cosmetics. They also digest perspiration, pollen and other small organic particles. Enzymes slowly turn oils into carbon dioxide and water, without leaving behind any residue or chemical fragments. Enzymes enable pool and spa owners the ability to remove oils that form the bathtub ring, without a lot of effort.

With the use of enzymes in pools and spas, water can be used longer before draining and less chlorine. Filters do not have to be cleaned as often and the build up of scum lines on the tile and liners is reduced. Maintenance time is reduced with the use of enzymes.

In the pond industry, enzymes attack waste materials such as slime, sludge, algae, dead plants, insects, uneaten fish food, waste from pond creatures and other organic contaminants. Enzymes turn them into a form that can later be eliminated by natural bacteria. Enzymes help decompose toxic hydrogen peroxide and release healthful oxygen from poisonous waste. Enzymes hold the key to life with their powers of being able to digest and promote assimilation and reduction. When enzymes and natural indigenous bacteria are through diminishing water-born contaminants, the by-product becomes nitrogen and water, providing a naturally clean and clear environment.

We can distinct two types of bacteria in the pond industry. There is the Mother Nature's indigenous bacteria that is naturally reoccurring bacteria that produces hundreds of different enzymes. The Manmade bacteria is a hybrid bacteria that produces selected types of enzymes. While the Mother Natures' work very slowly, the Hybrid bacteria is more aggressive and reproduces more rapidly.

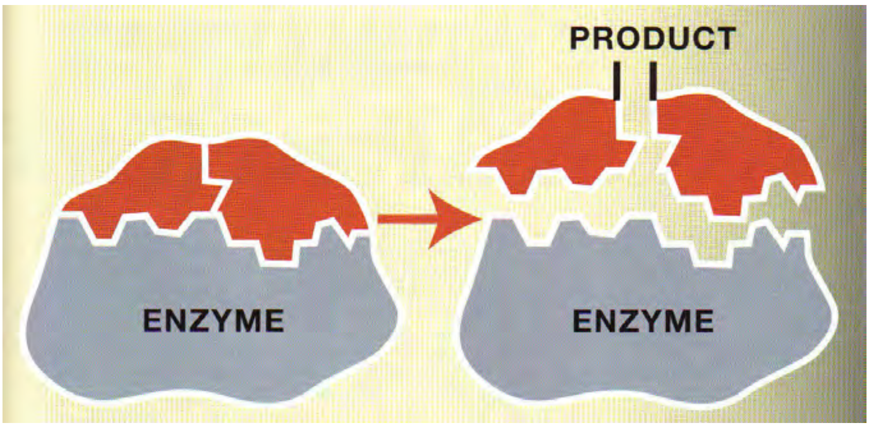


Fig.1: Enzymes bind temporarily to one or more of the reactants of the reaction they catalyze.

All bacteria can carry an enzyme. Hybrid bacteria produce selective enzymes like pectinase, cellulose, lipase and protease. Mother Nature produces these plus hundreds of others. Each type of enzyme is engineered to function differently; one specific enzyme cannot do the task of another. In a pond we have leaf litter, grass clippings, organic pond sediment, etc. Each contaminant is different and needs a specific enzyme to digest its matter. Mother Natures's bacteria functions better than that of hybrid bacteria because of the broad spectrum of enzymes she produces.

5.- BASIC INTRODUCTION TO GENOMICS

5.1.- INTRODUCTION

It is very important for understanding the goal of the thesis to have a general vision about genomics. That's why in this part of the work some important aspects of this subject are reported below.

The basic point of Darwin's theory is that every organism's dream is to become two organisms. An organism reproducing faster, or exploiting its environment more efficiently, rapidly out-competes its rivals for resources. This is reported because for understanding cells is very important to understand the power and the limitations of the natural selection.

There are two types of cells, those ones with nuclei (eukaryotic), and those ones without it (prokaryotic). These types of cells largely share the same fundamental machinery, but prokaryotes are simpler, unicellular organisms, for example a bacteria, while eukaryotes are often more complex and include both unicellular and multicellular organisms.

Unicellular organisms are the simplest free-living things. They have the ability to interact with the environment, derive the energy and materials needed to continually fabricate themselves, and then eventually to reproduce is controlled by a complex network of chemical reactions. These reactions are the essence of cellular life and need to be catalyzed by specific proteins.

A protein is a large molecule formed by a chain of amino acids, which folds into a characteristic shape. The same 20 basic amino acids are used by all known organisms. The exact composition of the chain determines its shape, and its shape determines its function.

The genetic material used by cells is formed by molecules of DNA, which have a sequential structure that enables them to act as information storage devices.

5.2.- THE WORLD OF THE GENOMES

After this brief introduction we can describe what is a genome. A genome is the set of all DNA contained in a cell, and there are a lot of organisms that can have multiples genomes in a single cell. The genome is formed by one or more long stretches of DNA strung together into chromosomes. These chromosomes can be linear or circular, and are faithfully replicated by a cell when it divides. The entire complement of chromosomes in a cell contains the DNA necessary to synthesize the proteins and other molecules needed to survive, as well as much of the information necessary for that purpose.

The most common genomes are reviewed above:

- *Prokariotic genomes:* Eubacteria and archaea are the two major groups of prokaryotes: free living organisms without nuclei, a structure within cells that is used by eukaryotes to house their genomes. Prokaryotic organisms generally have a single, circular genome between 0.5 and 13 megabases long. In addition to having relatively small genomes, prokaryote also have rather simples genes and genetic control sequences.

- *Viral genomes:* Although viruses are not free-living organisms, an examination of viral genomes can be very informative. Although these genomes are usually very short, between 5 and 50 kilobases, and contain very few genes, their sequence is a milestone for biology, and they enabled scientists to develop conceptual tools that would become essential for the analysis of the genomes of larger, free-living organisms. Because of their small size, we can analyze a large number of viral genomes simultaneously on a laptop, a task that would require a large cluster of machines in the case of longer genomic sequences.

- *Eukariotic genomes:* The nuclear genome of eukaryotes is usually considered the genome of such an organism. These nuclear genomes can be much larger than prokaryotic genomes, ranking in size from 8 Mb for some fungi to 670 gigabases for some species of the single-celled amoeba; humans come in at middling 3.5 Gb. Because of the large size of eukaryotic genomes, their sequencing is still a large effort usually undertaken by consortia of labs; these labs divide the work up by each sequencing different linear chromosomes of the same genome.

- *Organellar genomes:* In addition to these huge nuclear genomes, most eukaryotes also carry on more smaller genomes in each cell. These are contained in cellular organelles, the most common of which are the mitochondrion and the chloroplast. These genomes are usually only tens of thousands of bases long, circular, and contain a few essential genes.

DNA molecules consist of a chain of smaller molecules called nucleotides that are distinct from each other only in a chemical element called a base.

DNA is a complex molecule with three-dimensional properties but is very often modelled as a one-dimensional object, a sequence of symbols of the alphabet [A,C,G,T]. This abstraction is very powerful, as it enables us to deploy a large number of mathematical tools, but neglects all the information that can be contained in the three-dimensional structure of the molecule.

There are various elements of interest in a genome sequence. For example the structure of the genes, how we find them, the way in which they are regulated, etc. but we pay special attention in simpler statistical properties of the DNA sequences, properties like the frequency of nucleotides, dinucleotides (pair of bases), and other DNA words.

One of the most important properties in these statistical characteristics is the base composition. This property gives us information about the proportion of A, G, C y T nucleotides present in the genome.

A DNA sequence, *s*, is a finite string from an alphabet ({A, C, G, T}) of nucleotides. The genome is the set of all DNA sequences associated with an organism or organelle.

The elements of a sequence are denoted in the following way: $s=s_1s_2 \ldots s_n$, where an individual nucleotide is represented by s_i .

5.3.- PROBABILISTIC MODELS OF DNA SEQUENCES

There are some probabilistic models for describing the DNA sequences:

- *Multinomial sequence models:* The simplest model of DNA sequences assumes that the nucleotides are independent and identically distributed: the sequence has been generated by a stochastic process that produces any of the four symbols at each sequence-position *i* at random. The probability of observing any of the four nucleotides at position *i* of the sequence *s* is denoted by $p_x=p(s(i)=x)$ and does not depend on the position *i*.

- *Markov sequence models:* A more complex model of DNA sequences is provided by the theory of Markov chains. In Markov chains the probability of observing a symbol depends on the symbol preceding it in a sequence. In so doing, Markov chains are able to model local correlations among nucleotides.

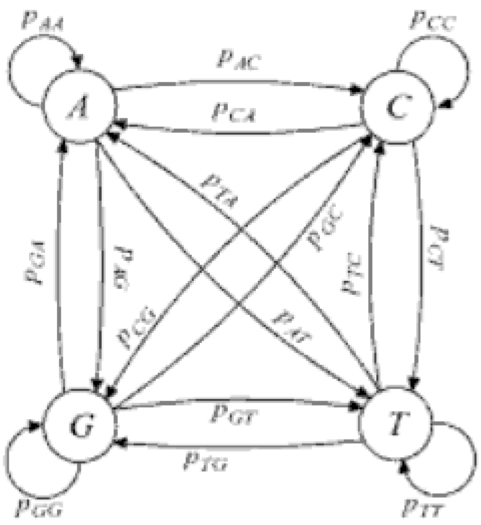


Fig.2: The trajectory of a Markov chain process generates a sequence of symbols. Starting at any one of the four nucleotides, the probability of the next nucleotide in the sequence is determines by the current state.

5.4.- IMPORTANT ASPECTS ABOUT GENOMES

Now, some elements of interest in a genome sequence will be described:

- *Base composition:* One of the most fundamental properties of a genome sequence is its base composition, the proportion of A, G, C, and T nucleotides present. It can be seen that the four nucleotides are not used at equal frequency across the genome.

- *GC content:* This content gives us information about the aggregate frequencies for G and C versus the aggregate frequencies for A and T. Given that these two quantities are required to always sum to 1, only the GC content is typically reported.

- *Change point analysis:* We would like to have a method to identify locations in the sequence where statistical properties, such as GC content, change. These locations, called change points, divide the genome into regions of approximately uniform statistical behaviour, and can help to identify important biological signals.

- *K-mer frequency and motif bias:* Another simple and important property of a genome to be measured is the frequency of all nucleotide words of length 2 (dinucleotides or dimers), or higher (trimers, k-mers). We define unusual k-mers as any words that appear more or less often than is expected. Bias in the number or position of these words can reveal important information about their function.

- *Finding unusual DNA words:* A simple statistical analysis can be used to find under-and-over-representation of motifs, and can also help us to decide when an observed bias is significant.

- *Biological relevance of unusual motifs:* Under- or over-representation of nucleotide motifs may reflect biological constraints, either mutational or selective. Frequent words may be due to repetitive elements, gene regulatory features, or sequences with other biological functions. Rare motifs including binding sites for transcription factors, words such as CTAG that have undesirable structural properties, or words that are not compatible with the internal immune system of bacterium.

All the cells need to produce the right molecules at the right time to survive. Their life is a continuous balancing act that requires them to maintain relatively constant internal conditions in the face of variation in the external environment. This constancy is achieved by constantly correcting the internal chemical environment to compensate for external change.

For multicellular organisms, it is also necessary to orchestrate the growth and development of an individual from a tiny single-celled embrio to an enormous trillion-celled adult. In order to perform all of these functions, each cell directs the production of thousands of specific proteins that each in turn control specific chemical reactions

5.5.- AMINO-ACIDS

The proteins are perhaps the most crucial piece in the cellular puzzle. They are molecules used for many tasks, from internal communication to the bricks-and-mortar of cells themselves. Some proteins, called degradases, are used to cut apart molecules no longer needed by the cell, while others, called ligases, are used to join molecules together. In humans, a protein called haemoglobin carries oxygen in red blood cells, while hormones such as insulin are proteins used to communicate between organs.

All proteins are formed from chains of simpler molecules called amino acids. Proteins are macromolecules, made by stringing together a chain of amino acids. A typical protein contains 200-300 amino-acids, but some are much smaller and some are much larger. Proteins can thus be read as one-dimensional objects, but do not have a linear form in the cell: the amino acids chain rapidly folds itself into the three-dimensional shape that ultimately determinates its function. The final shape of a protein is specified by the exact identity and order of the amino acids in the chain, and is largely the result of atomic interactions between amino acids and with the cellular medium.

There are only 20 amino acids used to form proteins. Consequently the number of possible proteins is enormous. If we assume that all proteins are just long 400 amino acids long, we can create 20^{400} different proteins. The number is huge and is more than sufficient to account for the total number of existing proteins on earth.

As with DNA sequences, proteins are modelled as strings form a finite alphabet. This representation neglects much of the crucial structural information, but also allows for powerful computational and statistical techniques to be employed.

The next alphabet is used to describe amino acid sequences:

$$\mathrm{A} = [A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V]$$

Here only a little bit of information has been given to have a smattering of genomics. This information is very useful to understand in a better way the text files that are given to us in order to do the thesis.

6.- STATISTICAL CLASSIFICATION

6.1.- NEURAL NETWORKS

6.1.1.- INTRODUCTION

A neural network is an information processing paradigm inspired by the studies of how biological nervous systems, as the brain, process information.

The key element of this paradigm is the novel structure of the information processing system. This system is composed of a large number of highly interconnected elements called neurones. These neurones work in unison to solve a specific problem.

These networks use a way of learning based in examples and each neural network has been designed for a specific application. Some of these specific applications can be pattern recognition or data classification.

The neural networks have become very important due to their remarkable ability to derive meaning from complicated or imprecise data that is why they can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

Some of the advantages that neural networks provide are the next ones:

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience
- Self-organisation: An ANN can create its own organisation or representation of the information it receives during the learning time
- Real time operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability

- Fault tolerance via redundant information coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

6.1.2.- APPLICATIONS

The tasks to which artificial neural network models are applied tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction and modelling
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making
- Data processing, including filtering, clustering, blind signal separation and compression

Application areas include:

- System identification and control (vehicle control, process control)
- Game-playing and decision making (backgammon, chess, racing)
- Pattern recognition (radar systems, face identification, object recognition, etc.)
- Sequence recognition (gesture, speech, handwritten text recognition)
- Medical diagnosis
- Financial applications
- Data mining
- Visualization
- E-mail spam filtering
- Etc.

6.1.3.- NEURAL NETWORKS VS CONVENTIONAL COMPUTERS

Neural networks take a different approach to solve problems than that used by conventional computers. The computers follows a set of instructions in order to solve a problem but unless that the computer knows what steps has to follow, the computer can not solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don’t exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements, the neurones, working in parallel to solve a specific problem. As it was said before, neural networks learn by example, they can not be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the networks finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small ambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithm approach like arithmetic operations and tasks that are more suited to neural networks.

6.2.- LINEAR DISCRIMINANT AND SUPPORT VECTOR CLASSIFIERS

6.2.1.- INTRODUCTION

Support Vector Classifiers are linear classifiers in a high dimensional space. Linear classifiers provide us a lot of advantages, between them we can emphasize three very important characteristics:

- They are computationally quite simple, both during learning and during classification
- Despite their simplicity these classifiers have had a great success in a large number of application areas.
- Another very important advantage of these classifiers is that hey are amenable to theoretical analysis.

6.2.2.- LINEAR DISCRIMINANT

For explaining this point a two classification problem is considered, in which patterns, represented as n-dimensional vectors *x*, are labelled with binary values: $y \in \{-1,+1\}$.

Then, we have the training examples with their respective labels:

$$X = \{x_1, \ldots, x_m\} \subseteq R^n$$
$$Y = \{y_1, \ldots, y_m\} \subseteq \{-1,1\}$$

But the problem now is to find a decision function, *g(x)*, which predicts accurately the class label *y* of any example *x* that may or may not belong to the training set.

The discriminant function approach uses a real valued function *f(x)*, called discriminant function, the sign of which determines the class label prediction: *g(x)=sgn(f(x))*. The discriminant function may be parameterized with some parameters $a=\{a_1,\ldots,a_p\}$ that are determined from the training examples by means of learning algorithm.

We have to emphasize the linear discriminant functions. These functions have the important characteristic that are linear in their parameters. We can find two kinds of linear discriminant functions:

$$f(x) = w \cdot \Phi(x) \rightarrow \text{Linear classifier or “Perceptron”} \qquad \textbf{Eq.1}$$

$$f(x) = \sum_{i=1}^{m} y_i \alpha_i k(x, x_i) \rightarrow \text{Kernel classifier} \qquad \textbf{Eq.2}$$

Attending to these equations we can say that there are feature-based classifiers and example-based classifiers. In the case of linear classifiers or “Perceptrons”, the basis functions $\Phi_k(x)$ of them can be understood as feature detectors.

Now if we analyse the equation of Kernel classifiers, their symmetric kernel functions *k(x,x’)* are often radial basis functions. The computation of the linear discriminant of kernel classifiers requires storing in memory the training examples, thus the name “example-based” or “memory-based” classifier.

Until here it hasn’t been considered one of the main points that appear in a lot of cases. In general, if we have a classification problem that is complex, a large number of features must be included in the classifier. However, the number of parameters to estimate rapidly becomes prohibitive. Similarly, training a large number of parameters may be infeasible, either because of high computation complexity or, more seriously, if there is insufficient training data, because the solution is undetermined.

6.3.- SUPPORT VECTOR MACHINES

6.3.1.- INTRODUCTION

A Support Vector Machine performs classification by constructing an N-dimensional hyperplane that optimally separates the data in two categories. The models developed of SVM are closely related to neural networks, in fact, a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

In other words, SVMs are a set of related supervised learning methods used for classification and regression. With some examples of a training data, each one labelled in the right way, a model is built in order to predict whether a new example falls into one category or the other.

Using a kernel function, SVMs are an alternative training method for polynomial, radial basis function and multi-layer perceptron classifiers in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.

In SVM there are attributes, that are predictor variables, and transformed attributes, features, that are used to define the hyperplane. The task of choosing the most suitable representation is known as feature selection. A set of features that describes one case is called a vector. What we want with the SVM models is to find the optimal hyperplane that separates cluster of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyperplane are in the support vectors.

6.3.2.- TWO-DIMENSIONAL EXAMPLE

Here it's considered a 2-dimensional example. A classification is wanted to be performed and the data has a categorical target variable with two categories. It is considered too that there are two predictor variables with continuous values. Below, it can be seen the example mentioned. In it, each category of the target variable is represented with crosses of different colours.

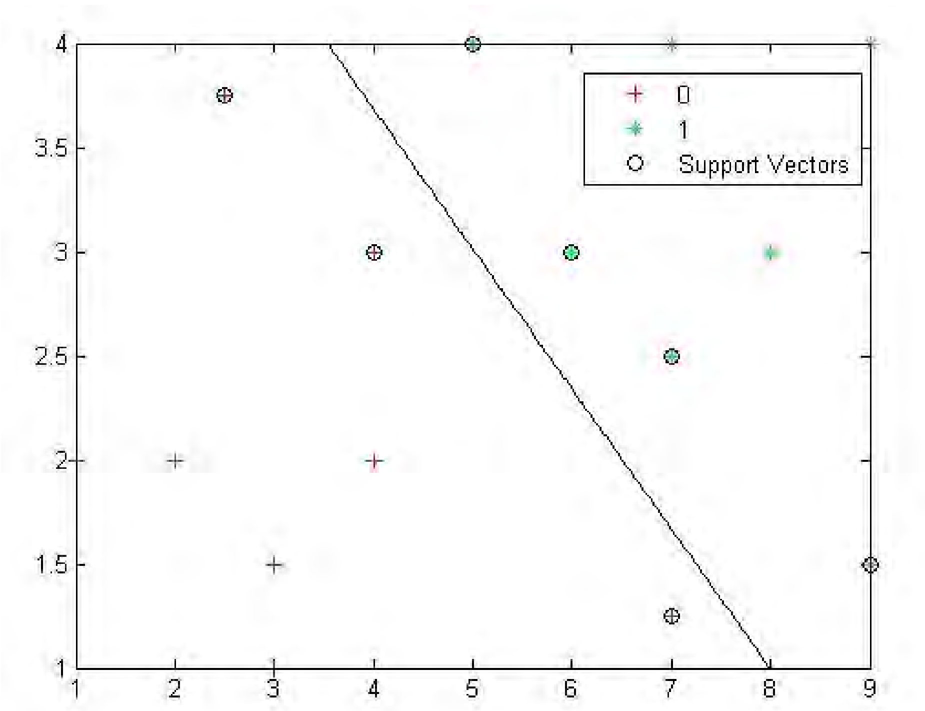


Fig.3: Classification of a two-dimensional example.

But not always we have the ideal case as the one shown in the example. In this example we have the cases of one category in the lower left corner and the cases of the other category in the upper right corner. Here the cases are completely separated. The SVM analysis attempts to find a 1-dimensional hyperplane that separates the cases based on their target categories. But there are a lot of lines that separate the two cases, the question is which one of the infinite lines that can be drawn is the optimal one.

In the picture, we could draw parallel lines to the separating line. These lines would mark the distance between the dividing line and the closest vectors to the line. The distance between these lines would be called the margin. The vectors that constrain the width of the margin are the support vectors.

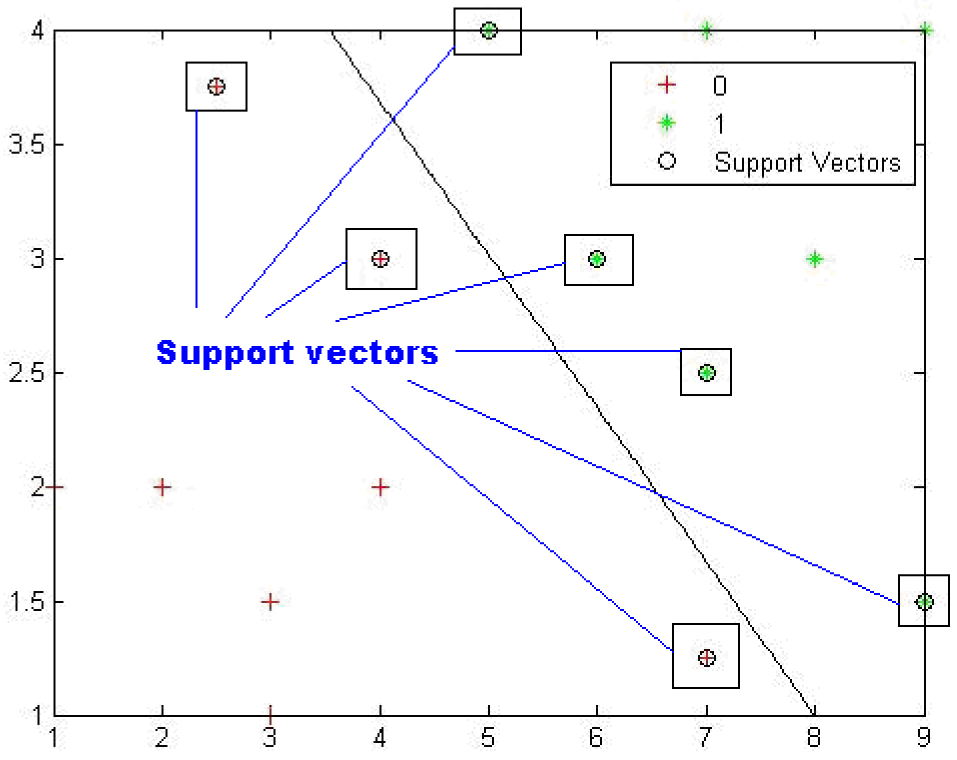


Fig.4: Support vectors used in the two-dimensional example.

Then, the goal of a SVM analysis is to find the line, or hyperplane, that is oriented so that the margin between the support vectors is maximized.

It would be very easy if all the analysis consisted of two-category target variables with two predictor variables and the cluster of points could be divided by a straight line, but unfortunately it happens not very often. Then the SVM must deal with more than two predictor variables, separating the points with non-linear curves, handling the cases where clusters cannot be completely separated, and handling classifications with more than two categories.

6.3.3.- EXPLANATION

This explain of the support vector machines is based on the studies done by Vapnik (Vapnik, 1995; Vapnik, 1998a; Vapnik 1998b).

If a set of *N* data points is considered:

$$\{y_k, x_k\}_{k=1}^{N}$$

where:

$x_k \in R^n$ is the *k*th input pattern

$y_k \in R$ is the *k*th output pattern

the support vector method approach aims at constructing a classifier of the form:

$$y(x) = sign\left[\sum_{k=1}^{N} \alpha_k y_k \Psi(x, x_k) + b\right] \qquad \textbf{Eq.3}$$

where:

α_k : positive real constant

b: real constant

At the time of analyzing the factor $\psi(\cdot,\cdot)$, it can be said that there are several choices, as the next ones:

- *Linear SVM*: $\psi(x, x_k) = x_k^T x$
- *Polynomial SVM of degree d*: $\psi(x, x_k) = (x_k^T x + 1)^d$
- *RBF SVM*: $\psi(x, x_k) = \exp\left\{-\|x - x_k\|_2^2 / \sigma^2\right\}$
- *Two layer neuronal SVM*: $\psi(x, x_k) = \tanh\left[\kappa x_k^T x + \theta\right]$

Where:

σ, κ, θ are constants

For constructing the classifier it's necessary to take some assumptions as the following ones:

$$w^T \varphi(x_k) + b \geq 1 \quad , \quad \text{if } y_k = +1 \qquad \textbf{Eq.4}$$

$$w^T \varphi(x_k) + b \leq 1 \quad , \quad \text{if } y_k = -1 \qquad \textbf{Eq.5}$$

These assumptions can be considered with this expression:

$$y_k\left[w^T \varphi(x_k) + b\right] \geq 1 \quad , \quad k = 1,...N. \qquad \textbf{Eq.6}$$

Where:

$\varphi(\cdot)$: nonlinear function which maps the input space into a higher dimensional space

However, this function is not explicitly constructed. If a separating hyperplane in this higher dimensional space does not exist we have to have the possibility of violating the Eq.6. In this case the new variables that appear in the next expression are introduced:

$$y_k\left[w^T \varphi(x_k) + b\right] \geq 1 - \xi_k \quad , \quad k = 1,...N. \qquad \textbf{Eq.7}$$

$\xi_k \geq 0 \quad , \quad k = 1,...N.$

If the optimization problem is formulated to minimize the risk bound according to the structural risk minimization principle it's obtained the following expression:

$$\min_{w,\xi_k} J_1(w,\xi_k) = \frac{1}{2} w^T w + c\sum_{k=1}^{N} \xi_k \qquad \textbf{Eq.8}$$

Now it's time to construct the Lagrangian introducing in the equation Lagrangian multipliers:

$$L_1(w,b,\xi_k;\alpha_k,v_k) = J_1(w,\xi_k) - \sum_{k=1}^{N} \alpha_k \left\{ y_k \left[w^T \varphi(x_k) + b \right] - 1 + \xi_k \right\} - \sum_{k=1}^{N} v_k \xi_k \qquad \textbf{Eq.9}$$

Where:

$\alpha_k \geq 0$: Lagrange multiplier

$v_k \geq 0$: Lagrange multiplier

The solution is given by the saddle point of the Lagrangian and this point is computed by the following way:

$$\max_{a_k,v_k} \min_{w,b,\xi_k} L_1(w,b,\xi_k;\alpha_k,v_k)$$

From this expression it is obtained:

$$\frac{\partial L_1}{\partial w} = 0 \rightarrow w = \sum_{k=1}^{N} \alpha_k y_k \varphi(x_k) \qquad \textbf{Eq.10}$$

$$\frac{\partial L_2}{\partial b} = 0 \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \qquad \textbf{Eq.11}$$

$$\frac{\partial L_1}{\partial \xi_k} = 0 \rightarrow 0 \le \alpha_k \le c, k = 1,...,N$$ **Eq.12**

The solution of this is the following quadratic programming problem:

$$\max_{\alpha_k} Q_1(\alpha_k ; \varphi(x_k)) = -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l \varphi(x_k)^T \varphi(x_l)\alpha_k \alpha_l + \sum_{k=1}^{N} a_k$$ **Eq.13**

such that

$$\sum_{k=1}^{N} \alpha_k y_k = 0 \quad , \quad 0 \le \alpha_k \le c \quad , \quad k = 1,...,N$$ **Eq.14**

6.4.- LEAST-SQUARES SUPPORT VECTOR MACHINES

6.4.1.- EXPLANATION

This method derives from the Support Vector Machine classification. We formulate the classification problem with the next equations. First of all we know that with this method we want to minimize the error, thus we define this equation:

$$\min_{w,b,e} J_3(w,b,e) = \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{k=1}^{N} e_k^2 \qquad \textbf{Eq.15}$$

But this equation is subject to equality constraints:

$$y_k \left[w^T \varphi(x_k) + b \right] = 1 - e_k, k = 1,\ldots,N \qquad \textbf{Eq.16}$$

Now with these equations we can define the Lagrangian that will be like this:

$$L_3(w,b,e;\alpha) = J_3(w,b,e) - \sum_{k=1}^{N} \alpha_k \left\{ y_k \left[w^T \varphi(x_k) + b \right] - 1 + e_k \right\} \qquad \textbf{Eq.17}$$

The α_k 's are the Lagrange multipliers. These multipliers can be positive or negative due to the equality constraints as follows from the Kuhn-Tucker conditions.

The conditions for optimality logically are based in derivatives. We obtain the derivatives of each factor that appears in the Lagrange function respect the own function:

$$\frac{\partial L_3}{\partial w} = 0 \rightarrow w = \sum_{k=1}^{N} a_k y_k \varphi(x_k) \qquad \textbf{Eq.18}$$

$$\frac{\partial L_3}{\partial b} = 0 \rightarrow \sum_{k=1}^{N} a_k y_k = 0 \qquad \textbf{Eq.19}$$

$$\frac{\partial L_3}{\partial e_k} = 0 \rightarrow a_k = \gamma e_k, k = 1,...,N$$ **Eq.20**

$$\frac{\partial L_3}{\partial a_k} = 0 \rightarrow y_k \left[w^T \varphi(x_k) + b \right] - 1 + e_k = 0, k = 1,...,N$$ **Eq.21**

To solve these equations, Fletcher in 1987, write an easier solution bases in these conditions of optimality:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & -Z^T \\ 0 & 0 & 0 & -Y^T \\ 0 & 0 & \gamma I & -I \\ \hline Z & Y & I & 0 \end{array}\right] \left[\begin{array}{c} w \\ b \\ e \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline 1 \end{array}\right]$$

Eq.22

Where:

$$\begin{aligned} Z &= \left[\varphi(x_1)^T y_1;...;\varphi(x_N)^T y_N\right] \\ Y &= \left[y_1;...;y_N\right] \\ \vec{1} &= [1;...;1] \\ e &= \left[e_1;...;e_N\right] \\ \alpha &= \left[\alpha_1;...;\alpha_N\right] \end{aligned}$$

The same solution can be given too by this way:

$$\left[\begin{array}{c|c} 0 & -Y^T \\ \hline Y & ZZ^T + \gamma^{-1} I \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 1 \end{array}\right]$$

Eq.23

The Mercer's condition is applied again to the matrix $\Omega_{kl} = y_k y_l \varphi(x_k)^T \varphi(x_l) = y_k y_l \Psi(x_k, x_l)$.

Then, it can be said that the classifier seen previously:

$$y(x) = sign\left[\sum_{k=1}^{N} \alpha_k y_k \Psi(x, x_k) + b\right] \quad \textbf{Eq.24}$$

can be solved with the linear set of equations (Eq.15 and Eq.16) instead of quadratic programming.

The parameters of the kernels such as σ for the RBF kernel can be optimally chosen by Eq.22.

The support values α_k are proportional to the errors at the data points (Eq.23), while in the case of Eq.24 most values are equal to zero.

6.4.2.- CONCLUSIONS

It has been discussed a least square version of support vector machine classifiers. Due to the equality constraints in the formulation, a set of linear equations has to be solved instead of a quadratic programming problem. Mercer’s condition is applied as in the others SVM’s.

For a complicated two-spiral classification problem a least squares SVM with RBF kernel is readily found with excellent generalization performance and low computational cost

7.- FEATURE SELECTION METHODS

7.1.- COVER'S THEOREM

The statement of Cover's theorem is the next one:

- The probability that classes are linearly separable increases when the features are nonlinearly mapped to a higher dimensional feature space.

This theorem can be explained like this:

If we have an input space of the features of *d* dimensions and the number of the samples (number of training data) is *N*, we have a relation like the next one:

$$O(N,d) = 2\sum_{i=0}^{d}\binom{N-1}{i} \qquad \textbf{Eq.25}$$

This expression gives us the number of dichotomies that we have (number of linearly separable groupings).

2^N is the total number of groupings. Knowing this, we can obtain the probability that the samples are linearly separable by the next way:

$$P_N^d = \frac{O(N,d)}{2^N} \qquad \textbf{Eq.26}$$

Then, as it was said before, this theorem says that if there is a set of training data that can't be separated in a linear way, projecting this set in a higher dimensional space (by non-linear transformation) the probability that this problem becomes linearly separable raises.

If we see the Eq. 26 we can ask ourselves what is the probability that the *N* samples in the *d*-dimensional space are linearly separable.

The answer is the next one:

- If $N \leq d+1$:

  Almost every set of points is linearly separable.

- If $N = 2(d+1)$:

  The set is linearly separable with a probability of $p = 0{,}5$.

- If $N >> d+1$:

  We have to consider that *N* is limited then *d* must be minimized.

In the publication of *Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition* in 1965 Cover presented this idea.

This statement is one of the primary theoretical motivations for the use of non-linear kernel methods in machine learning applications.

7.2.- GRAM-SCHMIDT ORTHOGONALIZATION

7.2.1.- EXPLANATION

As was talked previously, the Gram-Schmidt orthogonalization is one of the most important things in the task of classification. Below the general description and the process that must be followed for doing the Gram-Schmidt orthogonalization is described.

The Gram-Schmidt process is a method used for orthogonalization of a set of vectors in an inner product space. In other words, a set of linearly independent vectors is converted into a set of orthogonal vectors.

One of the keys of this process is the projection operator:

$$proj_u v = \frac{\langle v, u \rangle}{\langle u, u \rangle} u \qquad \textbf{Eq.27}$$

The goal of this operator is to project the vector *v* orthogonally onto the vector *u*.

For clarifying this concept an example of the Gram-Schmidt orthogonalization is shown below.

7.2.2.- EXAMPLE

Three vectors linearly independent as the next ones are considered:

$v_1 = (3,0,4)$

$v_2 = (-6,-4,1)$

$v_3 = (5,0,-3)$

Note: We know that are independent because the value of the determinant of the matrix $A = (v_1 \mid v_2 \mid v_3)$ is 116, that is different to 0.

Now we apply the Gram-Schmidt method:

$$w_1 = v_1 = (3,0,4)$$

$$w_2 = v_2 - \frac{w_1 \cdot v_2}{\|w_1\|^2} w_1 = \left(\frac{-108}{25}, -4, \frac{81}{25} \right)$$

$$w_3 = v_3 - \frac{w_1 \cdot v_3}{\|w_1\|^2} w_1 - \frac{w_2 \cdot v_3}{\|w_2\|^2} w_2 = \left(\frac{1856}{1129}, \frac{3132}{1129}, \frac{1392}{1129} \right)$$

Now we have the orthogonal set of vectors composed by w_1, w_2, w_3.

If we would like to have an orthonormal set of vectors we only have to consider $\frac{w_1}{\|w_1\|}, \frac{w_2}{\|w_2\|}, \frac{w_3}{\|w_3\|}$.

The representation of the orthogonal set of vectors of the example is the next one:

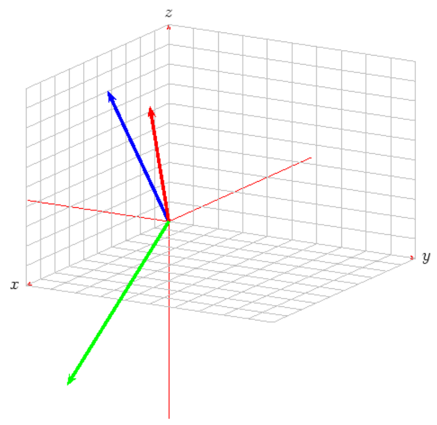


Fig.5: Example of Gram-Schmidt orthogonalization.

Then we can conclude saying that this method proceeds as follow:

To compute w_i, v_i is projected orthogonally onto the subspace W generated by $\{w_1,...,w_{i-1}\}$ which is the same as the subspace generated by $\{v_1,...,v_{i-1}\}$. The vector w_i is then defined to be the difference between v_i and this projection, guaranteed to be orthogonal to all of the vectors in the subspace W.

7.3.- APPLICATION OF THE GRAM-SCHMIDT ORTHOGONALIZATION IN THE PROCESS OF THE FEATURE RANKING SELECTION

7.3.1- INPUT FEATURES SELECTION

In many of the machine learning problems the number of input variables is very large. Given a multidimensional set of features, we cannot distinguish immediately those more relevant to the output from the irrelevant ones.

Let us consider a task of approximating the value of *y* given a set of input vectors *x*. The problem is that we do not know which of the inputs has actual influence on the value of *y*.

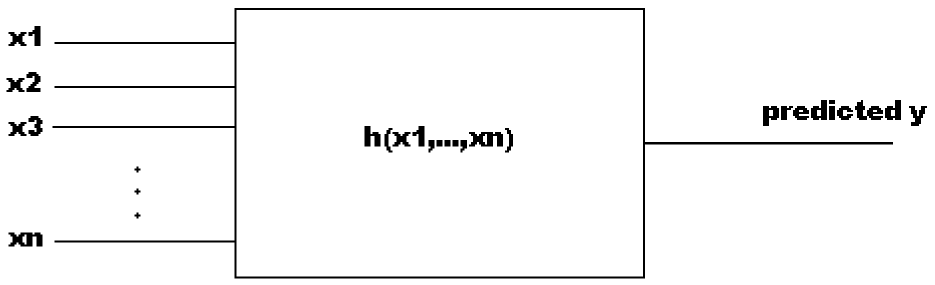


Fig.6: Simple representation of the input features selection. The value of *y* is a function of an unknown set of variables *x*.

Each of the non pertinent inputs x included in our model will only add to our model noise in the learning process. Furthermore, the complexity of such model is not optimal.

Once the most pertinent variables are chosen, model may gain some accuracy while limiting the size of the utilized input data. The purpose behind feature selection is reducing complexity of the model and thus making it more efficient.

We have used for doing the input features selection a method based on the Gram-Schmidt orthogonalization.

7.3.2.- GRAM-SCHMIDT ORTHOGONAL LEAST SQUARES ALGORITHM

The Gram-Schmidt method provides ways for orthogonalization a set of vectors in an inner product space. Thanks to this orthogonalization procedure we can examine the influence of every input feature on the output vector. Below, it is presented how to select a set of the inputs most relevant according to their influence in the output.

Given a model with Q candidate features; a data set containing N input-output pairs (measurements of the output of the process to be modelled, and of the candidature features) is available.

We denote by $x^i = \left[x_1^i, x_2^i, \ldots x_N^i\right]^T$ the vector of feature *i*, or of input *i*. We denote by y_p the *N*-vector of the classifier target values. We consider the (*N*,*Q*) matrix $X = \left[x^1, x^2, \ldots x^Q\right]$.

Ranking procedure starts with calculating correlation coefficient. The larger it is, the better the x^k feature vector explains the y^p variation.

$$\cos^2\left(x^k, y_p\right)^2 = \frac{\left(x^k y_p\right)^2}{\left\|x^k\right\|^2 \left\|y_p\right\|^2}, k = 1,\ldots,Q$$ **Eq.28**

As the first base vector we pick the one with the largest value of the Eq.. Since the angle between this input vector and the output is the smallest, we assume that it best "explains" the output.

All the remaining candidate inputs and the output vector are projected onto the null subspace (of dimension N-1) of the selected feature.

Next, we calculate Eq.28 for the projected vectors and again pick the one with the largest value of this quantity. The remaining feature vectors are projected onto the null subspace of the first two ranked vectors by the classical Gram-Schmidt orthogonalization. This procedure goes on until the x^k vectors are ranked.

The procedure for ranking the input features can be explained in a procedure based in steps:

1. Take the sets of inputs *X*.
2. Sort the inputs in *X* in descending order with respect to Eq.28.
3. Perform the Gram-Schmidt orthogonalization.
4. Update *X*: cut out the first feature vector.
5. Go back to 1; Repeat until all features are ranked.

We can explain the classical Gram-Schmidt orthogonalization by the next way:

Considering a start with a set of input vectors $\left(x^1, x^2, \ldots, x^Q\right)$:

1. $e^{(1)} = \frac{1}{\left|x^{(1)}\right|} x^{(1)}$

2. $w^{(2)} = x^{(2)} - \left(x^{(2)} e^{(1)}\right) e^{(1)}$

3. $e^{(2)} = \frac{1}{\left|w^{(2)}\right|} w^{(2)}$

4. $w^{(r)} = x^{(r)} - \left(x^{(r)} e^{(1)}\right) e^{(1)} - \ldots - \left(x^{(r)} e^{(r-1)}\right) e^{(r-1)}$

5. $e^{(r)} = \frac{1}{\left|w^{(r)}\right|} w^{(r)}$

As a result of running the above 5 step procedure on $\left(x^1, x^2, \ldots, x^Q\right)$ we acquire a set of vectors orthogonal to the first vector: x^1.

The final step of the feature selection is to reject the non relevant features. In order to do that we compare the rank value of every input feature vector with the rank of a random probe. Only the features, with rank value higher than the obtained for random probe, are considered relevant to the model.

8.- PRACTICAL PART

8.1.- EXPLANATION

This thesis has a theoretic part and a practical part. Here the practical part is going to be explained. For doing this part the Matlab program has been used.

The practical part has consisted in creating a program that can give us the useful information for doing a right classification. In other words, it has been designed an algorithm capable to rank all the features according to their influence in the output.

For doing this algorithm the concepts about ranking features and the Gram-Schmidt orthogonalization presented previously in the work have been used.

The algorithm has been divided in some functions. These functions have been designed to solve the problem step by step and finally to obtain the expected results.

Two text documents have been given to us for doing this practical part. One of them is a text file that contains a database. This database consists in numerical indices that represent various physicochemical and biochemical properties of amino acids and pairs of amino acids.

This first document is divided in some columns. The first column gives us information about the physicochemical or biochemical property name. The next ones represent the values of the different properties (Fig.7).

Aaindexes1.txt

```
#           A     C     D     E     F     G     H     I     K     …

ANDN920101  4.35  4.65  4.76  4.29  3.97  4.63  3.95  4.75  4.37  …
ARGP820101  0.61  1.07  0.46  0.47  0.07  0.61  2.22  0.06  0.    …
…
YANJ020101  NA    0.83  0.66  0.73  NA    0.92  0.88  0.76  0.59  …
```

Fig.7: Format of the document Aaindexes1.txt.

The second document is called Targets-LSW.txt and consists in a text file that contains amino acid sequences and class labels.

The format of this document is the next one: The first rows contain # and represent the class labels. The next rows that follow to the class labels are amino acid sequences belonging to the specified class. The length of the sequence is fixed but may be different in different files. The Fig.8 shows this format:

Targets-LSW.txt

```
# PKC
SKSKPKDPSQRRRSLEP
RRRRRYRRSTVARRRRR
RRRRYRRSTVARRRRRV
RRRRPRRVSRRRRARRR
...
# CDK
PFLEGCACTPERMAEAG
IPQQGFFSSPSTSRTPL
ESFKKQEKTPKTPKGPS
KKQEKTPKTPKGPSSVE
PVKKSIRDTPAKNAQKS
...
```

Fig.8: Format of the document Targets-LSW.txt

As result of the analysis of these documents we obtain some text files:

- log.txt: This file contains all the errors found in the file AAindexes1.txt, the different classes found in the document TargetsLSW.txt and the wrong strings inside this document
- "classname".txt: We obtain as many document files as classes we found. Each of these ones documents contains the different amino-acids of each class, and their properties.
- Targets-LSV-classname.txt: Here are reported the different values of the properties ordered according to their influence in the output.
- Targets-LSV-vars-classname.txt: Here are reported the number of the columns of properties according to their relevance in the output.

8.2.- PROGRAM DESCRIPTION

The designed program is clearly divided in 4 stages. The steps followed by the program are the next ones:

1. REPROCESSING OF TARGETS LSW.TXT FILE

We take the text file TargetsLSW.txt and we remove from that file all the chains of amino-acids that contain the letter ‘X’. We do this because we do not have the numerical values of the index ‘X’ in the different biochemical properties.

2. REPROCESSING OF AAINDEXES1.TXT FILE

After the first step we process the file Aaindexes1.txt. Here all the values of the indexes of the biochemical properties are reported. We must remove and report from this file all the characteristics that at least contain in one of their indexes a non-numerical value.

3. MAPPING OF AMINO-ACID SYMBOLS

After processing these two text files we generate one document file for each class detected in the file TargetsLSW.txt. These documents take the name of the corresponding class and in each of them are reported the chains of amino-acids of that class followed by all the properties that describe the amino-acid.

4. RANKING PROCEDURE

After this, the program starts to analyze each one of these files and apply the algorithm of feature selection. This algorithm follows some steps:

***4.1* –** First of all, the algorithm takes each one of the properties that describe the amino-acids and compute the correlation coefficient of the different properties with the output. The outputs are just labels that can take the values 1 or -1 depending on the class we are analyzing.

***4.2* –** Later the algorithm ranks the different properties according to their correlation coefficient.

***4.3* –** After this, the property that is ranked on the first place is removed.

4.4* –** The last step is to perform the Gram-Schmidt orthogonalization with the rest of the properties and we return again to the step ***4.1 to compute the new correlation coefficients.

The next picture shows us a brief summary of what we have reported before:

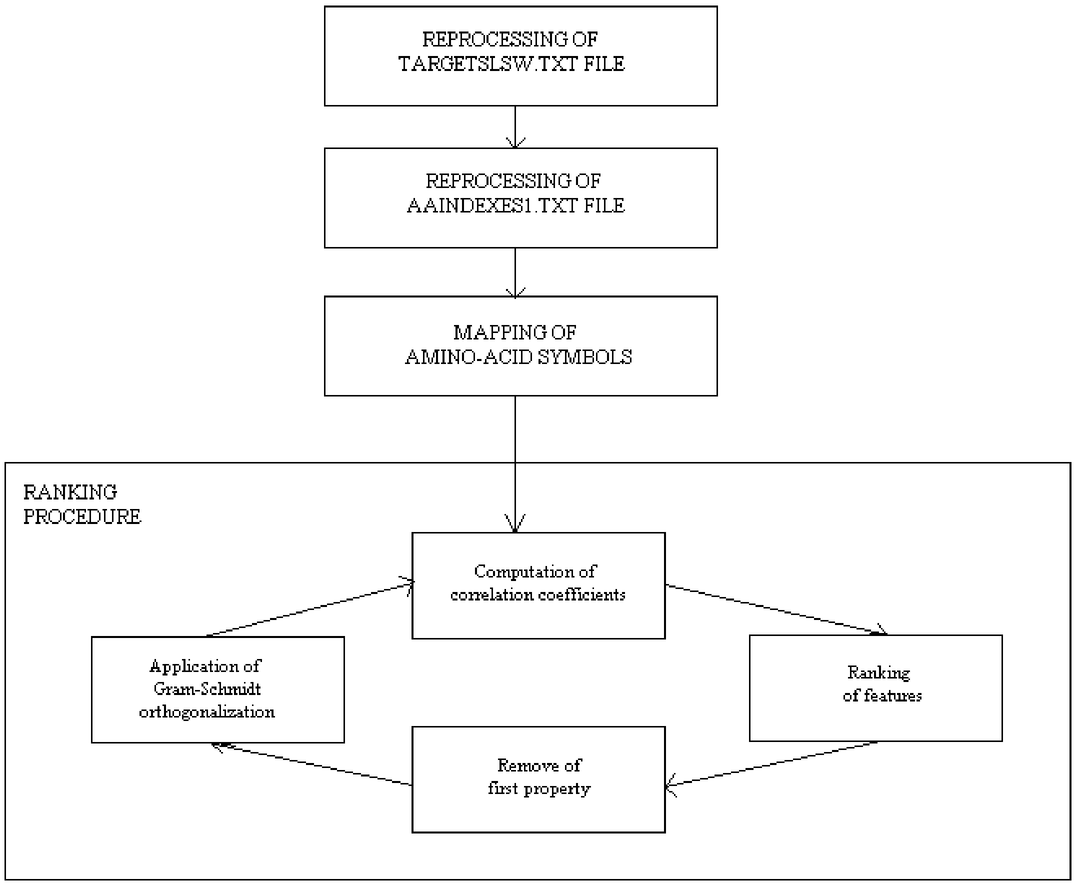

REPROCESSING OF TARGETSLSW.TXT FILE
REPROCESSING OF AAINDEXES1.TXT FILE
MAPPING OF AMINO-ACID SYMBOLS
RANKING PROCEDURE
Computation of correlation coefficients
Ranking of features
Remove of first property
Application of Gram-Schmidt orthogonalization

Fig.9: Brief summary about the program description.

9.- RESULTS

9.1.- DISCUSSION OF THE RESULTS

As we can see in this section the results obtained are the expected ones. We have obtained some text documents where the features appear ranked and thanks to this we can choose the most useful features for classification.

The sizes of the input documents were very large. Hence, we have reduced the complex problem to a simpler one. This procedure enabled us to run the computer program.

The program is prepared to run with as many classes and as many properties as we want.

Now we focus on the case study. In the case we have considered we have the following amino-acids ordered by enzyme classes:

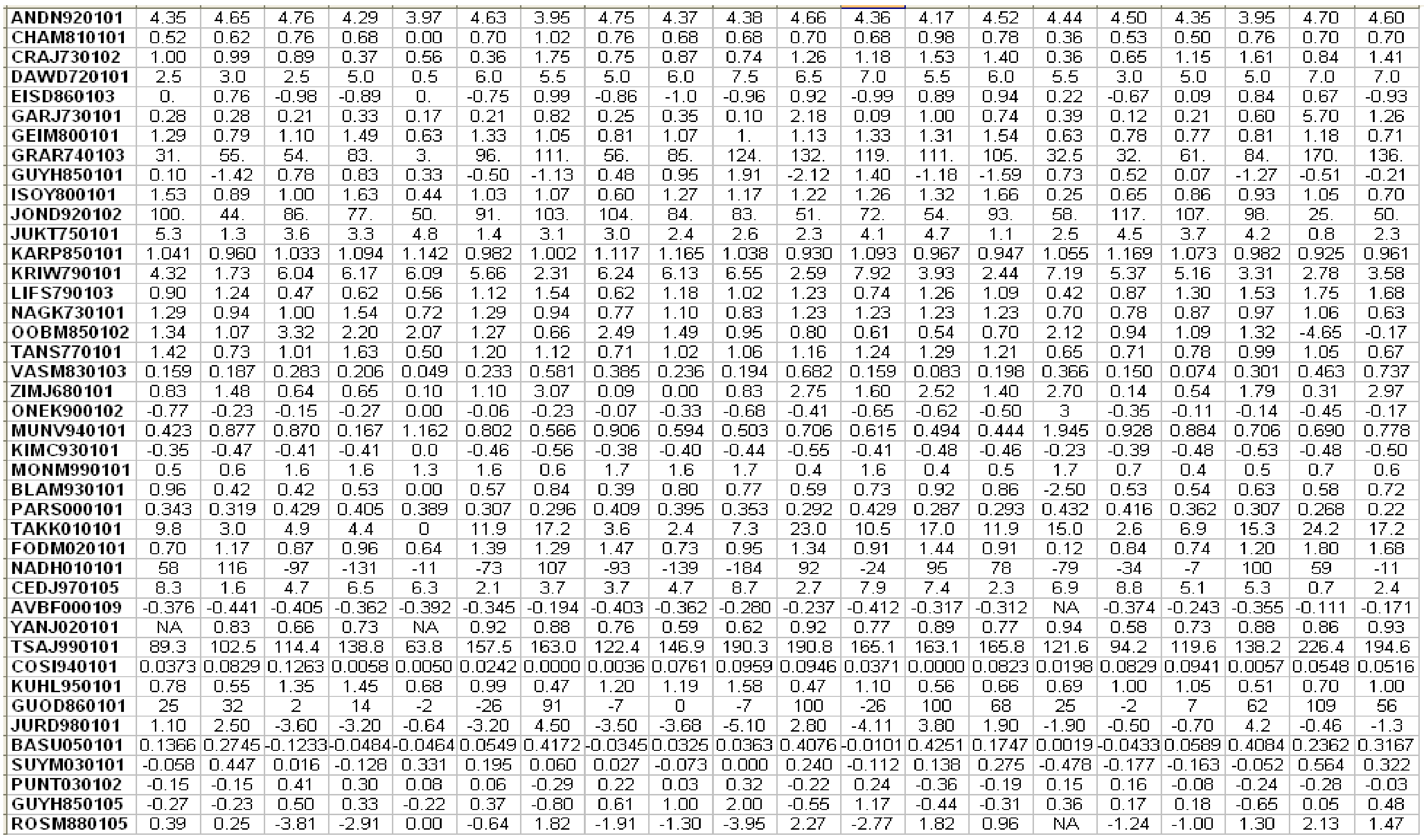
	# PKC
1	SKSKPKDPSQRRRSLEP
2	RRRRRYRRSTVARRRRR
3	VDAENRLQTMKEELDFQ
	# CDK
4	PFLEGCACTPERMAEAG
5	GGAGGYTQSPGGFGSPA
6	TQSPGGFGSPAPSQAEK
	# PKA
7	RWKALRRFSLATMRDFG
8	QCALCRRSTTDCGGPKD
9	VEPFTESQSLTLTDVEN
	# CK2
10	ADAVADSESEDEEDLDV

11	LVADAVADSESEDEEDL
12	FGFFSSSESGAPEAAEE
	# MAPK
13	ELLPTPPLSPSRRSGLC
14	TEENVSDGSPNAGSVEQ
15	MQSTKVPQTPLHTSRXX
	# PKB
16	PGARRRGGSASRSLPLP
17	HAVRIRGKSYVQCQGIP
18	KKPRHRSNSFSDEREFS

Fig.10: Amino-acids detected.

We can see that we have 17 amino-acids foe various classes and attempt to identify the pertinent properties of considered amino-acid sequences with respect to their relations to selected enzymes.

In the case of the biochemical properties we have the following 42 characteristics with the different indexes that describe the amino-acids:



ANDN920101	4.35	4.65	4.76	4.29	3.97	4.63	3.95	4.75	4.37	4.38	4.66	4.36	4.17	4.52	4.44	4.50	4.35	3.95	4.70	4.60
CHAM810101	0.52	0.62	0.76	0.68	0.00	0.70	1.02	0.76	0.68	0.68	0.70	0.68	0.98	0.78	0.36	0.53	0.50	0.76	0.70	0.70
CRAJ730102	1.00	0.99	0.89	0.37	0.56	0.36	1.75	0.75	0.87	0.74	1.26	1.18	1.53	1.40	0.36	0.65	1.15	1.61	0.84	1.41
DAWD720101	2.5	3.0	2.5	5.0	0.5	6.0	5.5	5.0	6.0	7.5	6.5	7.0	5.5	6.0	5.5	3.0	5.0	5.0	7.0	7.0
EISD860103	0.	0.76	-0.98	-0.89	0.	-0.75	0.99	-0.86	-1.0	-0.96	0.92	-0.99	0.89	0.94	0.22	-0.67	0.09	0.84	0.67	-0.93
GARJ730101	0.28	0.28	0.21	0.33	0.17	0.21	0.82	0.25	0.35	0.10	2.18	0.09	1.00	0.74	0.39	0.12	0.21	0.60	5.70	1.26
GEIM800101	1.29	0.79	1.10	1.49	0.63	1.33	1.05	0.81	1.07	1.	1.13	1.33	1.31	1.54	0.63	0.78	0.77	0.81	1.18	0.71
GRAR740103	31.	55.	54.	83.	3.	96.	111.	56.	85.	124.	132.	119.	111.	105.	32.5	32.	61.	84.	170.	136.
GUYH850101	0.10	-1.42	0.78	0.83	0.33	-0.50	-1.13	0.48	0.95	1.91	-2.12	1.40	-1.18	-1.59	0.73	0.52	0.07	-1.27	-0.51	-0.21
ISOY800101	1.53	0.89	1.00	1.63	0.44	1.03	1.07	0.60	1.27	1.17	1.22	1.26	1.32	1.66	0.25	0.65	0.86	0.93	1.05	0.70
JOND920102	100.	44.	86.	77.	50.	91.	103.	104.	84.	83.	51.	72.	54.	93.	58.	117.	107.	98.	25.	50.
JUKT750101	5.3	1.3	3.6	3.3	4.8	1.4	3.1	3.0	2.4	2.6	2.3	4.1	4.7	1.1	2.5	4.5	3.7	4.2	0.8	2.3
KARP850101	1.041	0.960	1.033	1.094	1.142	0.982	1.002	1.117	1.165	1.038	0.930	1.093	0.967	0.947	1.055	1.169	1.073	0.982	0.925	0.961
KRIW790101	4.32	1.73	6.04	6.17	6.09	5.66	2.31	6.24	6.13	6.55	2.59	7.92	3.93	2.44	7.19	5.37	5.16	3.31	2.78	3.58
LIFS790103	0.90	1.24	0.47	0.62	0.56	1.12	1.54	0.62	1.18	1.02	1.23	0.74	1.26	1.09	0.42	0.87	1.30	1.53	1.75	1.68
NAGK730101	1.29	0.94	1.00	1.54	0.72	1.29	0.94	0.77	1.10	0.83	1.23	1.23	1.23	1.23	0.70	0.78	0.87	0.97	1.06	0.63
OOBM850102	1.34	1.07	3.32	2.20	2.07	1.27	0.66	2.49	1.49	0.95	0.80	0.61	0.54	0.70	2.12	0.94	1.09	1.32	-4.65	-0.17
TANS770101	1.42	0.73	1.01	1.63	0.50	1.20	1.12	0.71	1.02	1.06	1.16	1.24	1.29	1.21	0.65	0.71	0.78	0.99	1.05	0.67
VASM830103	0.159	0.187	0.283	0.206	0.049	0.233	0.581	0.385	0.236	0.194	0.682	0.159	0.083	0.198	0.366	0.150	0.074	0.301	0.463	0.737
ZIMJ680101	0.83	1.48	0.64	0.65	0.10	1.10	3.07	0.09	0.00	0.83	2.75	1.60	2.52	1.40	2.70	0.14	0.54	1.79	0.31	2.97
ONEK900102	-0.77	-0.23	-0.15	-0.27	0.00	-0.06	-0.23	-0.07	-0.33	-0.68	-0.41	-0.65	-0.62	-0.50	3	-0.35	-0.11	-0.14	-0.45	-0.17
MUNV940101	0.423	0.877	0.870	0.167	1.162	0.802	0.566	0.906	0.594	0.503	0.706	0.615	0.494	0.444	1.945	0.928	0.884	0.706	0.690	0.778
KIMC930101	-0.35	-0.47	-0.41	-0.41	0.0	-0.46	-0.56	-0.38	-0.40	-0.44	-0.55	-0.41	-0.48	-0.46	-0.23	-0.39	-0.48	-0.53	-0.48	-0.50
MONM990101	0.5	0.6	1.6	1.6	1.3	1.6	0.6	1.7	1.6	1.7	0.4	1.6	0.4	0.5	1.7	0.7	0.4	0.5	0.7	0.6
BLAM930101	0.96	0.42	0.42	0.53	0.00	0.57	0.84	0.39	0.80	0.77	0.59	0.73	0.92	0.86	-2.50	0.53	0.54	0.63	0.58	0.72
PARS000101	0.343	0.319	0.429	0.405	0.389	0.307	0.296	0.409	0.395	0.353	0.292	0.429	0.287	0.293	0.432	0.416	0.362	0.307	0.268	0.22
TAKK010101	9.8	3.0	4.9	4.4	0	11.9	17.2	3.6	2.4	7.3	23.0	10.5	17.0	11.9	15.0	2.6	6.9	15.3	24.2	17.2
FODM020101	0.70	1.17	0.87	0.96	0.64	1.39	1.29	1.47	0.73	0.95	1.34	0.91	1.44	0.91	0.12	0.84	0.74	1.20	1.80	1.68
NADH010101	58	116	-97	-131	-11	-73	107	-93	-139	-184	92	-24	95	78	-79	-34	-7	100	59	-11
CEDJ970105	8.3	1.6	4.7	6.5	6.3	2.1	3.7	3.7	4.7	8.7	2.7	7.9	7.4	2.3	6.9	8.8	5.1	5.3	0.7	2.4
AVBF000109	-0.376	-0.441	-0.405	-0.362	-0.392	-0.345	-0.194	-0.403	-0.362	-0.280	-0.237	-0.412	-0.317	-0.312	NA	-0.374	-0.243	-0.355	-0.111	-0.171
YANJ020101	NA	0.83	0.66	0.73	NA	0.92	0.88	0.76	0.59	0.62	0.92	0.77	0.89	0.77	0.94	0.58	0.73	0.88	0.86	0.93
TSAJ990101	89.3	102.5	114.4	138.8	63.8	157.5	163.0	122.4	146.9	190.3	190.8	165.1	163.1	165.8	121.6	94.2	119.6	138.2	226.4	194.6
COSI940101	0.0373	0.0829	0.1263	0.0058	0.0050	0.0242	0.0000	0.0036	0.0761	0.0959	0.0946	0.0371	0.0000	0.0823	0.0198	0.0829	0.0941	0.0057	0.0548	0.0516
KUHL950101	0.78	0.55	1.35	1.45	0.68	0.99	0.47	1.20	1.19	1.58	0.47	1.10	0.56	0.66	0.69	1.00	1.05	0.51	0.70	1.00
GUOD860101	25	32	2	14	-2	-26	91	-7	0	-7	100	-26	100	68	25	-2	7	62	109	56
JURD980101	1.10	2.50	-3.60	-3.20	-0.64	-3.20	4.50	-3.50	-3.68	-5.10	2.80	-4.11	3.80	1.90	-1.90	-0.50	-0.70	4.2	-0.46	-1.3
BASU050101	0.1366	0.2745	-0.1233	-0.0484	-0.0464	0.0549	0.4172	-0.0345	0.0325	0.0363	0.4076	-0.0101	0.4251	0.1747	0.0019	-0.0433	0.0589	0.4084	0.2362	0.3167
SUYM030101	-0.058	0.447	0.016	-0.128	0.331	0.195	0.060	0.027	-0.073	0.000	0.240	-0.112	0.138	0.275	-0.478	-0.177	-0.163	-0.052	0.564	0.322
PUNT030102	-0.15	-0.15	0.41	0.30	0.08	0.06	-0.29	0.22	0.03	0.32	-0.22	0.24	-0.36	-0.19	0.15	0.16	-0.08	-0.24	-0.28	-0.03
GUYH850105	-0.27	-0.23	0.50	0.33	-0.22	0.37	-0.80	0.61	1.00	2.00	-0.55	1.17	-0.44	-0.31	0.36	0.17	0.18	-0.65	0.05	0.48
ROSM880105	0.39	0.25	-3.81	-2.91	0.00	-0.64	1.82	-1.91	-1.30	-3.95	2.27	-2.77	1.82	0.96	NA	-1.24	-1.00	1.30	2.13	1.47

Fig.11: Table of all the biochemical properties with their respective indexes.

Each property has specific indexes that will help us to perform the classification

After running the program we obtained some results:

The file log.text reports the wrong amino-acids, the wrong characteristics that do not provide us any information and the classes of amino-acids detected:

WRONG DATA IN THE LIST OF BIOCHEMICAL PROPERTIES
AVBF000109
YANJ020101
ROSM880105

Fig.12: Wrong data in the list of biochemical properties.

The wrong data obtained in the previous table concern to those biochemical properties that contain, in at least one of their indexes, the characters NA instead a numerical value. It is necessary to process the document Aaindexes1.txt to obtain this information. These properties can not be used to define an amino-acid because they can not provide us enough information.

WRONG DATA IN THE LIST OF AMINO-ACIDS
MQSTKVPQTPLHTSRXX

Fig.13: Wrong data in the list of amino-acids.

The chains of characters that we can find in the table of the Fig.12 concern to that chains of amino-acids that contains, at least, a letter ‘X’ in their composition. It is important to detect these amino-acids because the index ‘X’ is not defined in the biochemical properties so we do not have the numerical values that would describe this index.

CLASSES OF AMINO-ACIDS DETECTED
PKC
CDK
PKA
CK2
MAPK
PKB

Fig.14: Classes of amino-acids detected.

In the table of the Fig.14 the different classes detected in the file TargetsLSW.txt are shown. Relating each amino-acid to its corresponding class is necessary because later in order to compute all the values we must assign the value of the output to each chain. If they belong to the class the output takes the value 1, if not it takes the value -1. Thus, to perform a good classification it is necessary to relate each amino-acid to a specific class.

Finally we must extract the more influential values for the output of the amino-acids. We will show now the values of the class PKC. We have to realise that each amino-acid is followed by 663 values, so each class is described by 663 different columns, and each column takes a number of values that depend on the different indexes of the biochemical properties. We must realise that we are working with a huge amount of data so we will report the most important values according to their correlation coefficient. The rest of values have a very low correlation coefficient so they are not important for the output To clarify the results we can show a small example: Having one amino-acid for example (this amino-acid has been invented) of 4 letters and given 3 properties that define this amino-acid it is described in the next way:

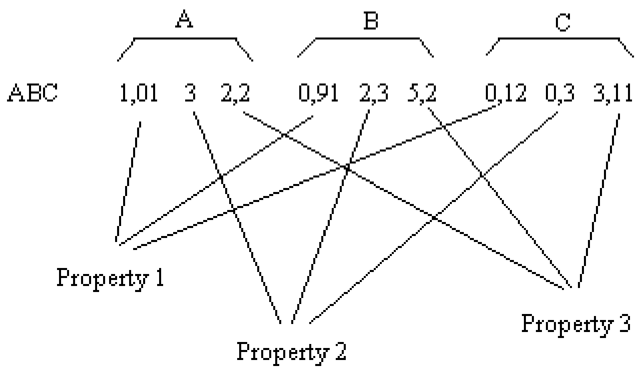


Fig.15: Simple example about how is defined an amino-acid.

Our program has the goal of ordering these 9 columns according to their influence on the system output. A possible order could be as follows:

ORDER	LOCATION	PROPERTIE
1	B	Property 2
2	A	Property 3
3	B	Property 3
4	C	Property 3
5	C	Property 2
6	A	Property 1
7	B	Property 1
8	A	Property 2
9	C	Property 1

Fig.16: Simple example about how is defined an amino-acid.

Now that it has been clarified how the results of our problem will be presented we will focus in the problem that we were threatening.

The table corresponding to the class PKC is the next one. We show the order, the location, the property and the correlation value:

ORDER	LOCATION	PROPERTY	CORRELATION COEFFICIENT
1	8	ZIMJ680101	0,697587334
2	12	SUYM030101	0,622927402
3	16	SUYM030101	0,722846761
4	6	CEDJ970105	0,627307424
5	2	MUNV940101	0,593001501
6	2	NAGK730101	0,645561259
7	7	DAWD720101	0,73165474
8	11	NADH010101	0,67647546
9	4	ZIMJ680101	0,609506438
10	14	ZIMJ680101	0,654634467
11	2	GUOD860101	0,592851885
12	13	NADH010101	0,647951595
13	14	KUHL950101	0,564617113
14	10	KIMC930101	0,570148532
15	13	KARP850101	0,672021505

16	10	BASU050101	0,684250697
17	17	PARS000101	0,68733407
18	4	KRIW790101	0,679090953
19	13	CRAJ730102	0,723366272
20	16	KARP850101	0,647703659
21	10	GARJ730101	0,665668281
22	6	NAGK730101	0,654423964
23	12	ANDN920101	0,670293594
24	2	VASM830103	0,591356083
25	1	TANS770101	0,701385114
26	3	GARJ730101	0,609539414
27	6	ZIMJ680101	0,78218442
28	1	FODM020101	0,640895409
29	16	GEIM800101	0,603560862
30	14	VASM830103	0,572063163
31	4	OOBM850102	0,617642065
32	5	FODM020101	0,730000922
33	7	JUKT750101	0,687411582
34	9	PARS000101	0,64127336
35	9	VASM830103	0,74582876
36	13	BLAM930101	0,765531816
37	10	GUYH850101	0,591573341
38	8	JUKT750101	0,634514008
39	1	GRAR740103	0,647298696
40	9	NADH010101	0,682365978
41	5	KUHL950101	0,555784303
42	9	EISD860103	0,606420487
43	8	BASU050101	0,688601306
44	12	DAWD720101	0,590281336
45	15	CEDJ970105	0,680777856
46	1	ANDN920101	0,64415254
47	2	KRIW790101	0,817130494
48	14	JUKT750101	0,692832592
49	15	LIFS790103	0,710198784
50	10	CRAJ730102	0,634167909
51	8	KIMC930101	0,63889095
52	6	DAWD720101	0,732565521
53	5	CRAJ730102	0,641455005
54	7	ZIMJ680101	0,704660452
55	11	JUKT750101	0,648536082
56	5	GARJ730101	0,600697866
57	1	ISOY800101	0,621238067
58	7	BASU050101	0,594636689
59	16	BASU050101	0,747540911
60	14	ONEK900102	0,652828791
61	10	COSI940101	0,582934623

62	18	GUYH850105	0,613428914
63	14	GUOD860101	0,74644729
64	2	TAKK010101	0,60756944
65	5	PUNT030102	0,618498537
66	16	DAWD720101	0,713372457
67	15	COSI940101	0,704499677
68	6	GUYH850101	0,638672021
69	2	GRAR740103	0,599823607
70	5	TAKK010101	0,563577439
71	12	MONM990101	0,659034886
72	3	SUYM030101	0,594220444
73	12	TAKK010101	0,740357167
74	4	LIFS790103	0,566612766
75	16	GUYH850101	0,553859649
76	14	GEIM800101	0,645721844
77	1	TSAJ990101	0,710046947
78	6	FODM020101	0,602610793
79	16	KIMC930101	0,595228764
80	8	TANS770101	0,670017298
81	1	TAKK010101	0,661176123
82	8	BLAM930101	0,603557007
83	6	KARP850101	0,638830448
84	8	NADH010101	0,511751841
85	11	DAWD720101	0,619576392
86	9	GRAR740103	0,621221624
87	6	ISOY800101	0,633262287
88	17	MUNV940101	0,706588927
89	16	KUHL950101	0,638744809
90	4	ONEK900102	0,725395841
91	16	PARS000101	0,588446307
92	12	BASU050101	0,75537508
93	3	GUYH850105	0,615344066
94	9	COSI940101	0,614232166
95	17	GUYH850101	0,553964705
96	9	GEIM800101	0,622056206
97	7	TANS770101	0,714756858
98	14	JOND920102	0,58100272
99	12	GARJ730101	0,648289519
100	13	CHAM810101	0,706694895
101	11	VASM830103	0,54540667
102	5	EISD860103	0,628446084
103	1	SUYM030101	0,632963068
104	8	PARS000101	0,777699383
105	1	BLAM930101	0,59444455
106	10	FODM020101	0,755311962
107	3	BLAM930101	0,726257153

108	13	MONM990101	0,725310492
109	17	ONEK900102	0,713144936
110	3	ANDN920101	0,599247212
111	12	FODM020101	0,63914296
112	16	ANDN920101	0,657093793
113	4	ANDN920101	0,624055281
114	15	JUKT750101	0,720565993
115	12	GUYH850101	0,652851857
116	2	ONEK900102	0,792010801
117	14	COSI940101	0,660338941
118	13	PARS000101	0,655160455
119	17	OOBM850102	0,552194836
120	3	CEDJ970105	0,639868804
121	7	VASM830103	0,733323401
122	11	KRIW790101	0,618721508
123	1	KRIW790101	0,722352024
124	13	KRIW790101	0,585579942
125	9	BLAM930101	0,694608684
126	6	EISD860103	0,728482852
127	1	MUNV940101	0,573677264
128	17	NADH010101	0,545608176
129	7	ONEK900102	0,63601141
130	10	EISD860103	0,640832338
131	12	CRAJ730102	0,588994461
132	2	OOBM850102	0,728420025
133	17	FODM020101	0,600094712
134	15	EISD860103	0,623705651
135	15	KRIW790101	0,637500998
136	3	BASU050101	0,664918371
137	16	EISD860103	0,68790261
138	11	TAKK010101	0,651528184
139	4	TSAJ990101	0,749934608
140	11	EISD860103	0,58852071
141	2	ZIMJ680101	0,55894992
142	1	NADH010101	0,64200878
143	17	BLAM930101	0,659235968
144	4	CEDJ970105	0,564242045
145	3	LIFS790103	0,618466035
146	2	TSAJ990101	0,676409901
147	5	VASM830103	0,693865686
148	14	OOBM850102	0,584713866
149	2	SUYM030101	0,640273708
150	4	NAGK730101	0,632524458
151	8	KUHL950101	0,717580203
152	8	TAKK010101	0,649060787
153	7	GEIM800101	0,5907505

154	9	ANDN920101	0,573197254
155	11	ONEK900102	0,735115926
156	2	EISD860103	0,563416661
157	6	KRIW790101	0,576648829
158	1	BASU050101	0,608228157
159	8	ISOY800101	0,579771228
160	16	CRAJ730102	0,822439732
161	13	FODM020101	0,629399073
162	4	TANS770101	0,554189014
163	8	VASM830103	0,664159553
164	14	NAGK730101	0,508369269
165	14	SUYM030101	0,666581445
166	4	JURD980101	0,546219888
167	16	CHAM810101	0,60724785
168	2	BLAM930101	0,655666529
169	9	JOND920102	0,520100406
170	3	FODM020101	0,537308258
171	17	SUYM030101	0,614389528
172	8	JOND920102	0,697224338
173	1	ZIMJ680101	0,577184165
174	9	TANS770101	0,6508223
175	14	ISOY800101	0,661918222
176	4	GARJ730101	0,559143878
177	15	ANDN920101	0,607084649
178	11	KIMC930101	0,671097215
179	4	TAKK010101	0,671735913
180	9	KARP850101	0,582961276
181	15	TAKK010101	0,640259416
182	7	CRAJ730102	0,591781511
183	7	COSI940101	0,56562391
184	3	COSI940101	0,659297993
185	10	NAGK730101	0,596523416
186	10	KRIW790101	0,51795773
187	1	EISD860103	0,647571538
188	15	SUYM030101	0,695094407
189	2	GUYH850105	0,675421432
190	6	GARJ730101	0,646567976
191	13	NAGK730101	0,710921787
192	15	GEIM800101	0,6466318
193	3	PUNT030102	0,765087275
194	15	PARS000101	0,577648851
195	14	MUNV940101	0,561460587
196	12	GEIM800101	0,54925623
197	5	DAWD720101	0,630612446
198	4	BLAM930101	0,665133207
199	9	TAKK010101	0,541020394

200	10	TAKK010101	0,627285365
201	15	FODM020101	0,524019659
202	15	OOBM850102	0,688471916
203	1	CRAJ730102	0,589214185
204	11	ISOY800101	0,534914116
205	10	ANDN920101	0,742218354
206	15	VASM830103	0,580742568
207	1	GARJ730101	0,719041775
208	13	LIFS790103	0,580751504
209	12	ONEK900102	0,601468045
210	16	MUNV940101	0,714107542
211	1	PUNT030102	0,757818142
212	16	GUOD860101	0,614466134
213	4	ISOY800101	0,563361657
214	5	PARS000101	0,617756917
215	3	GUYH850101	0,660651843
216	7	KRIW790101	0,670318361
217	7	EISD860103	0,548595661
218	12	BLAM930101	0,579696801
219	3	MUNV940101	0,634097855
220	7	CHAM810101	0,758344178
221	17	TSAJ990101	0,592082417
222	9	NAGK730101	0,527808212
223	12	VASM830103	0,527410212
224	7	GRAR740103	0,532901014
225	2	CEDJ970105	0,732723994
226	2	JOND920102	0,556307792
227	9	OOBM850102	0,523411395
228	3	GRAR740103	0,662953604
229	13	TSAJ990101	0,735330826
230	13	JOND920102	0,590548232
231	3	ZIMJ680101	0,631434679
232	12	LIFS790103	0,617107854
233	6	GUYH850105	0,627941639
234	11	KARP850101	0,559638385
235	7	PARS000101	0,684189463
236	11	PUNT030102	0,646103422
237	8	CRAJ730102	0,624896463
238	8	JURD980101	0,678078687
239	2	KIMC930101	0,753778369
240	5	GUYH850101	0,747763506
241	12	JURD980101	0,706115016
242	17	NAGK730101	0,715228457
243	10	JURD980101	0,587275598
244	14	GUYH850105	0,601211957
245	13	GRAR740103	0,666396367

246	7	KIMC930101	0,627038009
247	5	KARP850101	0,656459793
248	17	GUYH850105	0,605629963
249	1	JOND920102	0,546883763
250	17	KRIW790101	0,55281454
251	17	GEIM800101	0,675049649
252	12	NAGK730101	0,700123238
253	6	ONEK900102	0,59941061
254	7	JOND920102	0,580620354
255	3	VASM830103	0,508460801
256	11	MUNV940101	0,582038344
257	12	COSI940101	0,563871501
258	8	GRAR740103	0,565106958
259	9	KIMC930101	0,629776088
260	3	KARP850101	0,611567046
261	16	BLAM930101	0,528345499
262	13	KUHL950101	0,609255114
263	5	CEDJ970105	0,542101867
264	3	OOBM850102	0,66644871
265	17	CRAJ730102	0,604311296
266	11	COSI940101	0,662076638
267	9	PUNT030102	0,668345122
268	14	CEDJ970105	0,589120751
269	2	CHAM810101	0,5346323
270	1	GEIM800101	0,651561676
271	14	TAKK010101	0,740343788
272	3	NAGK730101	0,755733727
273	8	LIFS790103	0,77466843
274	4	PUNT030102	0,553863163
275	5	ONEK900102	0,721299713
276	12	PARS000101	0,592388601
277	5	LIFS790103	0,719280961
278	4	CHAM810101	0,551495404
279	17	ISOY800101	0,688147566
280	1	MONM990101	0,579468944
281	14	CRAJ730102	0,655859604
282	4	NADH010101	0,619353792
283	1	PARS000101	0,649746331
284	8	EISD860103	0,568305896
285	9	MONM990101	0,607212921
286	4	GUYH850101	0,559293758
287	13	GARJ730101	0,717108112
288	6	ANDN920101	0,551281347
289	16	LIFS790103	0,510228524
290	16	VASM830103	0,498215474
291	17	GARJ730101	0,601438205

292	10	BLAM930101	0,587200641
293	7	TSAJ990101	0,680372169
294	12	ZIMJ680101	0,575992931
295	2	DAWD720101	0,610507015
296	8	CEDJ970105	0,620543039
297	16	TANS770101	0,575206068
298	6	GEIM800101	0,567558304
299	8	ANDN920101	0,659205774
300	15	DAWD720101	0,609539622
301	1	OOBM850102	0,533719839
302	5	ISOY800101	0,502438066
303	14	GRAR740103	0,790143116
304	12	PUNT030102	0,658073615
305	16	PUNT030102	0,699260619
306	14	GUYH850101	0,567736209
307	12	EISD860103	0,718328759
308	6	SUYM030101	0,576877173
309	5	TSAJ990101	0,578456807
310	13	GUOD860101	0,691923665
311	12	TSAJ990101	0,674520172
312	6	COSI940101	0,541946066
313	3	TAKK010101	0,748785958
314	8	DAWD720101	0,530634551
315	9	BASU050101	0,625118759
316	10	NADH010101	0,570805388
317	3	CRAJ730102	0,623565729
318	13	COSI940101	0,630072629
319	17	KIMC930101	0,719407505
320	4	KIMC930101	0,814754846
321	10	ISOY800101	0,549964612
322	16	GARJ730101	0,577228632
323	15	TSAJ990101	0,558571499
324	5	MONM990101	0,670419168
325	5	GUOD860101	0,718528415
326	5	JURD980101	0,530016704
327	14	LIFS790103	0,646052384
328	7	NADH010101	0,51373435
329	16	TAKK010101	0,595615031
330	5	TANS770101	0,536276865
331	13	MUNV940101	0,575911346
332	4	GUOD860101	0,606120461
333	6	TSAJ990101	0,698465819
334	16	ZIMJ680101	0,554512704
335	17	JUKT750101	0,617464966
336	3	NADH010101	0,635615232
337	15	ZIMJ680101	0,478932013

338	4	JOND920102	0,644579421
339	16	CEDJ970105	0,612982214
340	7	PUNT030102	0,635514951
341	3	KUHL950101	0,618567385
342	13	VASM830103	0,534683759
343	10	GUOD860101	0,634223114
344	6	KIMC930101	0,663967124
345	13	ZIMJ680101	0,583713788
346	10	ONEK900102	0,667377157
347	17	COSI940101	0,495617725
348	8	OOBM850102	0,604692056
349	3	ISOY800101	0,585404676
350	14	KIMC930101	0,509377771
351	14	FODM020101	0,572106382
352	5	BASU050101	0,718664869
353	5	GRAR740103	0,680608533
354	8	SUYM030101	0,753814562
355	16	GRAR740103	0,670330279
356	9	GUYH850101	0,701221404
357	7	CEDJ970105	0,700040146
358	8	FODM020101	0,497034576
359	9	DAWD720101	0,612652349
360	4	DAWD720101	0,540726956
361	16	OOBM850102	0,579661985
362	6	CHAM810101	0,586336577
363	15	MUNV940101	0,548104874
364	9	CRAJ730102	0,492806279
365	14	JURD980101	0,571191838
366	16	KRIW790101	0,538539104
367	16	ISOY800101	0,594509589
368	3	TANS770101	0,603775615
369	2	COSI940101	0,596268933
370	10	MONM990101	0,623357934
371	9	KRIW790101	0,558387804
372	7	SUYM030101	0,74638851
373	14	TANS770101	0,711828963
374	11	BLAM930101	0,567296002
375	11	JOND920102	0,626695828
376	6	VASM830103	0,51795773
377	8	MUNV940101	0,51795773
378	15	GUYH850105	0,51795773
379	13	EISD860103	0,51795773
380	17	BASU050101	0,526581226
381	1	KUHL950101	0,530021334
382	1	DAWD720101	0,545152361
383	4	KARP850101	0,53191612

384	10	JUKT750101	0,51795773
385	2	MONM990101	0,51795773
386	6	PUNT030102	0,696586105
387	9	TSAJ990101	0,646969917
388	6	BLAM930101	0,51795773
389	5	NAGK730101	0,533451765
390	12	KIMC930101	0,558875484
391	2	ISOY800101	0,540559095
392	11	CEDJ970105	0,512501071
393	16	FODM020101	0,634027666
394	9	CHAM810101	0,686470946
395	8	GEIM800101	0,676211733
396	14	ANDN920101	0,519909822
397	12	GRAR740103	0,599001438
398	15	CRAJ730102	0,641217679
399	7	LIFS790103	0,558442191
400	11	MONM990101	0,508369269
401	8	KRIW790101	0,583098809
402	7	ANDN920101	0,641088149
403	6	LIFS790103	0,584741346
404	3	JUKT750101	0,508369269
405	11	ANDN920101	0,528018466
406	8	GARJ730101	0,742961588
407	10	JOND920102	0,573767695
408	2	KARP850101	0,532502454
409	2	GEIM800101	0,647393166
410	3	JURD980101	0,51795773
411	10	VASM830103	0,51795773
412	15	TANS770101	0,621598499
413	2	CRAJ730102	0,518688298
414	4	CRAJ730102	0,51795773
415	3	EISD860103	0,535688827
416	17	ZIMJ680101	0,51795773
417	13	OOBM850102	0,636027638
418	9	GUYH850105	0,721124341
419	1	COSI940101	0,558210296
420	1	GUYH850101	0,51795773
421	8	CHAM810101	0,532050743
422	17	KARP850101	0,51795773
423	10	OOBM850102	0,534935699
424	11	ZIMJ680101	0,51795773
425	16	NAGK730101	0,576838125
426	2	TANS770101	0,428864095
427	3	JOND920102	0,530627822
428	15	GARJ730101	0,391544895
429	12	NADH010101	0,496864483

430	14	GARJ730101	0,56268467
431	9	JUKT750101	0,56184746
432	14	EISD860103	0,418153538
433	4	EISD860103	0,485973247
434	12	KUHL950101	0,444404117
435	4	COSI940101	0,710180291
436	17	JOND920102	0,630257237
437	11	LIFS790103	0,494942663
438	1	VASM830103	0,48689829
439	13	TAKK010101	0,512853427
440	3	ONEK900102	0,463482111
441	17	MONM990101	0,569983986
442	12	JOND920102	0,309404218
443	10	SUYM030101	0,616063117
444	8	GUOD860101	0,51795773
445	2	NADH010101	0,51795773
446	1	KARP850101	0,51795773
447	16	JURD980101	0,51795773
448	5	KIMC930101	0,51795773
449	15	GUOD860101	0,51795773
450	11	OOBM850102	0,590288554
451	16	JOND920102	0,51795773
452	17	ANDN920101	0,51795773
453	9	ZIMJ680101	0,51795773
454	10	TANS770101	0,51795773
455	12	GUYH850105	0,51795773
456	12	JUKT750101	0,51795773
457	2	KUHL950101	0,51795773
458	13	BASU050101	0,51795773
459	2	JURD980101	0,508369269
460	13	GUYH850105	0,508369269
461	11	GUOD860101	0,508369269
462	15	BASU050101	0,576311481
463	8	NAGK730101	0,508369269
464	10	CEDJ970105	0,508369269
465	1	GUOD860101	0,508369269
466	7	GUOD860101	0,508369269
467	11	TANS770101	0,31425875
468	16	COSI940101	0,311476458
469	4	SUYM030101	0,311476458
470	11	GARJ730101	0,311476458
471	17	CHAM810101	0,311476458
472	7	GUYH850105	0,311476458
473	15	MONM990101	0,311476458
474	13	JUKT750101	0,311476458
475	5	CHAM810101	0,311476458

476	6	MUNV940101	0,311476458
477	15	NADH010101	0,311476458
478	15	ISOY800101	0,311476458

Fig.17: The most influential properties of the PKC class.

Therefore:

- The order reveals what properties are more important.
- The location gives us information about what letter of the chain of letters of the amino-acids is selected.
- Property shows us the names of the biochemical properties selected.
- Correlation coefficient shows the values of the correlation coefficient for each characteristic.

To show that the rank which we perform of the properties is the right one according to the correlation coefficient, we present below a table containing all the values of the correlation coefficient belonging to the first iteration. So if we see the first value of the correlation coefficient of Fig.17 we can see that it is the same that the first value of the next table (Fig.18) and is the largest correlation coefficient found in this specific iteration.

Values of the correlation coefficients obtained after the first computation

0.6976	0.2869	0.1630	0.0513	-0.0867	-0.1919	-0.3289
0.6194	0.2840	0.1615	0.0500	-0.0877	-0.1923	-0.3292
0.6058	0.2840	0.1572	0.0491	-0.0884	-0.1925	-0.3304
0.5902	0.2822	0.1565	0.0487	-0.0896	-0.1941	-0.3314
0.5875	0.2801	0.1545	0.0487	-0.0897	-0.1962	-0.3321
0.5868	0.2772	0.1529	0.0487	-0.0901	-0.1980	-0.3341
0.5793	0.2739	0.1529	0.0484	-0.0915	-0.1981	-0.3394
0.5785	0.2727	0.1525	0.0484	-0.0927	-0.2005	-0.3400
0.5634	0.2704	0.1502	0.0473	-0.0929	-0.2026	-0.3415
0.5615	0.2703	0.1492	0.0445	-0.0929	-0.2032	-0.3446
0.5605	0.2678	0.1477	0.0437	-0.0955	-0.2041	-0.3506
0.5479	0.2670	0.1474	0.0427	-0.0959	-0.2047	-0.3516
0.5355	0.2608	0.1467	0.0420	-0.0985	-0.2055	-0.3537
0.5330	0.2593	0.1465	0.0394	-0.0986	-0.2060	-0.3540

0.5291	0.2584	0.1453	0.0393	-0.1021	-0.2087	-0.3558
0.5257	0.2563	0.1425	0.0383	-0.1035	-0.2099	-0.3560
0.5239	0.2561	0.1423	0.0351	-0.1041	-0.2113	-0.3575
0.5117	0.2549	0.1385	0.0350	-0.1075	-0.2134	-0.3613
0.5003	0.2542	0.1355	0.0339	-0.1083	-0.2150	-0.3624
0.4914	0.2540	0.1346	0.0298	-0.1112	-0.2155	-0.3674
0.4806	0.2540	0.1338	0.0287	-0.1127	-0.2162	-0.3743
0.4774	0.2523	0.1289	0.0223	-0.1144	-0.2168	-0.3753
0.4768	0.2518	0.1272	0.0202	-0.1162	-0.2174	-0.3792
0.4726	0.2500	0.1251	0.0199	-0.1190	-0.2197	-0.3844
0.4644	0.2487	0.1238	0.0193	-0.1194	-0.2200	-0.3908
0.4643	0.2470	0.1236	0.0191	-0.1211	-0.2204	-0.3914
0.4603	0.2463	0.1234	0.0181	-0.1216	-0.2205	-0.3953
0.4547	0.2402	0.1229	0.0179	-0.1217	-0.2212	-0.3996
0.4516	0.2368	0.1213	0.0163	-0.1224	-0.2217	-0.4011
0.4455	0.2277	0.1184	0.0160	-0.1260	-0.2220	-0.4091
0.4436	0.2275	0.1177	0.0141	-0.1272	-0.2241	-0.4145
0.4395	0.2214	0.1164	0.0133	-0.1279	-0.2243	-0.4173
0.4362	0.2213	0.1159	0.0109	-0.1281	-0.2275	-0.4209
0.4324	0.2190	0.1149	0.0088	-0.1282	-0.2294	-0.4253
0.4317	0.2152	0.1144	0.0066	-0.1316	-0.2314	-0.4269
0.4267	0.2137	0.1140	-0.0009	-0.1330	-0.2327	-0.4309
0.4181	0.2123	0.1139	-0.0036	-0.1346	-0.2339	-0.4345
0.4165	0.2120	0.1133	-0.0055	-0.1363	-0.2347	-0.4383
0.4163	0.2102	0.1125	-0.0068	-0.1381	-0.2350	-0.4385
0.4159	0.2100	0.1112	-0.0105	-0.1384	-0.2360	-0.4502
0.4147	0.2090	0.1100	-0.0119	-0.1389	-0.2363	-0.4521
0.4146	0.2077	0.1087	-0.0154	-0.1392	-0.2372	-0.4556
0.4133	0.2056	0.1083	-0.0158	-0.1406	-0.2376	-0.4639
0.4047	0.2050	0.1078	-0.0169	-0.1416	-0.2415	-0.4640
0.4020	0.2049	0.1055	-0.0173	-0.1433	-0.2417	-0.4711
0.4012	0.2045	0.1031	-0.0199	-0.1451	-0.2418	-0.4740
0.3980	0.2032	0.1031	-0.0203	-0.1465	-0.2427	-0.4789
0.3922	0.2005	0.1022	-0.0217	-0.1465	-0.2443	-0.4960
0.3911	0.1998	0.1010	-0.0233	-0.1483	-0.2479	-0.4984
0.3883	0.1982	0.0994	-0.0237	-0.1501	-0.2484	-0.4995
0.3864	0.1976	0.0981	-0.0247	-0.1518	-0.2486	-0.5098
0.3863	0.1973	0.0981	-0.0251	-0.1519	-0.2493	-0.5111
0.3808	0.1958	0.0974	-0.0266	-0.1529	-0.2507	-0.5151
0.3799	0.1920	0.0972	-0.0267	-0.1541	-0.2511	-0.5196
0.3793	0.1905	0.0967	-0.0277	-0.1549	-0.2537	-0.5255
0.3786	0.1905	0.0953	-0.0299	-0.1567	-0.2546	-0.5281
0.3744	0.1905	0.0933	-0.0342	-0.1586	-0.2549	-0.5420
0.3725	0.1905	0.0929	-0.0342	-0.1603	-0.2571	-0.5427
0.3714	0.1905	0.0929	-0.0345	-0.1609	-0.2585	-0.5535
0.3682	0.1905	0.0919	-0.0349	-0.1633	-0.2587	-0.5540
0.3646	0.1905	0.0910	-0.0354	-0.1634	-0.2630	-0.5602
0.3542	0.1905	0.0910	-0.0355	-0.1637	-0.2636	-0.5861

0.3497	0.1905	0.0894	-0.0366	-0.1641	-0.2640	-0.5979
0.3465	0.1905	0.0886	-0.0369	-0.1668	-0.2665	
0.3464	0.1905	0.0856	-0.0390	-0.1670	-0.2679	
0.3454	0.1905	0.0850	-0.0407	-0.1686	-0.2679	
0.3449	0.1905	0.0824	-0.0411	-0.1690	-0.2680	
0.3448	0.1905	0.0823	-0.0418	-0.1708	-0.2690	
0.3446	0.1905	0.0822	-0.0432	-0.1733	-0.2706	
0.3416	0.1905	0.0818	-0.0445	-0.1735	-0.2709	
0.3407	0.1905	0.0810	-0.0471	-0.1736	-0.2711	
0.3347	0.1905	0.0789	-0.0479	-0.1761	-0.2754	
0.3338	0.1905	0.0753	-0.0500	-0.1826	-0.2755	
0.3336	0.1905	0.0748	-0.0509	-0.1831	-0.2758	
0.3326	0.1905	0.0746	-0.0529	-0.1835	-0.2776	
0.3323	0.1905	0.0734	-0.0533	-0.1836	-0.2784	
0.3292	0.1891	0.0729	-0.0542	-0.1845	-0.2823	
0.3276	0.1843	0.0716	-0.0560	-0.1857	-0.2851	
0.3268	0.1829	0.0714	-0.0590	-0.1858	-0.2855	
0.3267	0.1828	0.0702	-0.0606	-0.1862	-0.2865	
0.3248	0.1809	0.0696	-0.0613	-0.1872	-0.2867	
0.3237	0.1806	0.0670	-0.0618	-0.1874	-0.2928	
0.3170	0.1803	0.0659	-0.0630	-0.1879	-0.2931	
0.3165	0.1780	0.0650	-0.0681	-0.1905	-0.2957	
0.3157	0.1771	0.0646	-0.0692	-0.1905	-0.3001	
0.3143	0.1764	0.0639	-0.0713	-0.1905	-0.3042	
0.3129	0.1762	0.0634	-0.0714	-0.1905	-0.3055	
0.3127	0.1744	0.0616	-0.0719	-0.1905	-0.3056	
0.3058	0.1732	0.0602	-0.0723	-0.1905	-0.3074	
0.3056	0.1731	0.0591	-0.0723	-0.1905	-0.3116	
0.3050	0.1717	0.0587	-0.0744	-0.1905	-0.3118	
0.3043	0.1713	0.0583	-0.0750	-0.1905	-0.3130	
0.3029	0.1695	0.0576	-0.0750	-0.1905	-0.3150	
0.3011	0.1690	0.0575	-0.0753	-0.1905	-0.3156	
0.2993	0.1684	0.0572	-0.0766	-0.1905	-0.3172	
0.2985	0.1677	0.0563	-0.0801	-0.1905	-0.3177	
0.2973	0.1666	0.0561	-0.0821	-0.1905	-0.3182	
0.2899	0.1644	0.0545	-0.0828	-0.1905	-0.3239	
0.2885	0.1636	0.0539	-0.0834	-0.1905	-0.3259	
0.2875	0.1635	0.0531	-0.0844	-0.1905	-0.3264	

Fig.18: Correlation coefficients belonging to the first iteration.

For the rest of the classes the format of the results is the same but the order of values is changed.

By this way we are able to order all the biochemical properties that are given to us in any problem and we can select the properties that are pertinent for defining a problem.

10.- CONCLUSIONS

We have explained in this work the method of improved classification by selection of pertinent input features based on the correlation coefficient between the vector of a given feature and the vector of class labels.

The classification problems in bioinformatics are usually characterised by a huge number of descriptors and comparatively small sample size. Therefore the complexity of the problem increases as the number of properties increases.

We have applied this developed method to our specific problem and we can see that we have obtained some results according to what we wanted: a rank of all the features based in the correlation coefficient.

The subject that we have treated is very interesting because allows us to choose a specific number of characteristics when we are analyzing a complex problem. In this case we have realised that after computing the correlation coefficients between all the properties and the output we have obtained some very low values. Because of this we can discard some properties from the rank of features and this allows us to convert the complex problem in an easier one.

If we see the table 17 of the results we realise that the values of the correlation coefficients of the first ranked positions are not always higher than the ones ranked after. However, this does not mean that the result is wrong. It happens because after selecting the more important feature, after each iteration, we perform the Gram-Schmidt orthogonalization so the correlation coefficients change. As we can see in the table of the Fig.18 the rank of the properties is developed in the right way in each iteration and the values of the correlation coefficients are ranked in a descending order.

The program has been designed to work with as many chains of amino.acids and biochemical properties as we want. The steps that the program follows are always the same ones and as we have seen in the results of our problem finally we obtain the rank of the features, which is our main goal.

The subject of the feature selection has become very important nowadays and that is why new advances and new algorithms that allow us to perform a right rank of the features are appearing.

11.- REFERENCES

1.- BURGES, C. J. C. *Geometric Methods for Feature Extraction and Dimensional Reduction: A Guided Tour.*

2.- BURGES, C. J. C; SCHOLKOPF, B; SMOLA, A. J. *Advances in Kernel Methods: Support Vector Learning.* England: The MIT Press.

3.- CHODZYNSKA, J; DWULIT, M; JANKOWSKI, S; SZYMANSKI, Z; WYRWICZ, L. S. *Pertinent parameters selection for processing of short amino acid sequences.*

4.- CRISTIANINI, N; HAHN, M. W. *Introduction to Computational Genomics – A Case Studies Approach.*

5.- DREYFUS, G; DUBOIS, R; OUSSAR, Y. (2003). *Journal of Machine Learning Research 3, 1399-1414 – Ranking a Random Feature for Variable and Feature Selection.*

6.- DTREG — PREDICTIVE MODELING SOFTWARE [Consulted: December, 2th, 2009]. Available in Internet:
http://www.dtreg.com/

7.- DUDA, R. O; HART, P. E; STORK, D. G. *Pattern Classification.* 2nd ed. United States of America: John Wilet & Sons,Inc.

8.- DWULIT, M; JANKOWSKI, S; SZYMANSKI, Z. *Feature selection and mapping of data from short amino acid sequences.*

9.- ESPOSITO, S. (2009). *Multi Sensor Systems for Landmine Detection.* Rome: “La Sapienza” University of Rome.

10.- GRAM-SCHMIDT ORTHOGONALIZATION [Consulted: November, 20th, 2009]. Available in Internet:
http://www.ugrad.cs.ubc.ca/~cs402/handouts/handout08.pdf

11.- GUNN, S. (1998). *Support Vector Machines for Classification and Regression.*

12.- GUYON, I; STORK, D. G. *Linear Discriminant and Support Vector Classifiers.*

13.- JANKOWSKI, S. *Introduction to Support Vector Machines – New Learning Machines.*

14.- JANKOWSKI, S. *Least-Squares Support Vector Machine LS-SVM.*

15.- JANKOWSKI, S. *Neural Networks – Classification and Prediction of Financial Time Series.*

16.- JANKOWSKI, S; PIATKOWSKA-JANKO, E; RACZYK, M. *Feature selection of signal-averaged electrocardiograms by orthogonal least squares method.*

17.- PLATT, J. C. (2000). *Fast Training of Support Vector Machines using Sequential Minimal Optimization.*

18.- RACZYK, M. (2008). *Master thesis – Improving kernel methods by using adaptive kernels and feature selection: algorithms and applications.*

19.- SCHOLKOPF, B; TSUDA, K; VERT, J. *Kernel Methods in Computational Biology*. England: A Bradford book, The MIT Press.

20.- WIKIPEDIA, LA ENCICLOPEDIA LIBRE. Available in Internet: http://es.wikipedia.org/

12.- APPENDIX

12.1.- CODE OF THE ALGORITHM

The steps that have been followed are showed before in the statement of the thesis. According to this we have generated some functions, as the next ones:

The first function has the mission of read all the useful information that contain the document where appear the biochemical properties of the amino-acids. The code of this function is the next one:

```
function [colerror,colok,y] = readaaindexes1temp()
[nombres,cola,colb,colc,cold,cole,colf,colg,colh,coli,colj,colk,coll,colm,coln,colo,colp,colq,colr,cols,colt]=textread('
aaindexes1.txt','%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s');

j=1;
y=1;
fid=fopen('log.txt','w');
fprintf(fid,'%0s\n','-Wrong Data in Aindexes1');
asize=size(nombres,1);

for i=1:1:asize,
   if(strcmp(cola(i),' '))
   else
   if
((strcmp(cola(i),'NA'))|(strcmp(colb(i),'NA'))|(strcmp(colc(i),'NA'))|(strcmp(cold(i),'NA'))|(strcmp(cole(i),'NA'))|(strc
mp(colf(i),'NA'))|(strcmp(colg(i),'NA'))|(strcmp(colh(i),'NA'))|(strcmp(coli(i),'NA'))|(strcmp(colj(i),'NA'))|(strcmp(col
k(i),'NA'))|(strcmp(coll(i),'NA'))|(strcmp(colm(i),'NA'))|(strcmp(coln(i),'NA'))|(strcmp(colo(i),'NA'))|(strcmp(colp(i),'
NA'))|(strcmp(colq(i),'NA'))|(strcmp(colr(i),'NA'))|(strcmp(cols(i),'NA'))|(strcmp(colt(i),'NA'))|((strcmp(cola(i),'
'))|(strcmp(colb(i),' '))|(strcmp(colc(i),' '))|(strcmp(cold(i),' '))|(strcmp(cole(i),' '))|(strcmp(colf(i),' '))|(strcmp(colg(i),'
'))|(strcmp(colh(i),' '))|(strcmp(coli(i),' '))|(strcmp(colj(i),' '))|(strcmp(colk(i),' '))|(strcmp(coll(i),' '))|(strcmp(colm(i),'
'))|(strcmp(coln(i),' '))|(strcmp(colo(i),' '))|(strcmp(colp(i),' '))|(strcmp(colq(i),' '))|(strcmp(colr(i),' '))|(strcmp(cols(i),'
'))|(strcmp(colt(i),' '))))

       colerror{j}=nombres{i};
       fprintf(fid,'%0s\n',colerror{j});
       j=j+1;
```

```
  else
      colok(y,1)=nombres(i);
      colok(y,2)=cola(i);
      colok(y,3)=colb(i);
      colok(y,4)=colc(i);
      colok(y,5)=cold(i);
      colok(y,6)=cole(i);
      colok(y,7)=colf(i);
      colok(y,8)=colg(i);
      colok(y,9)=colh(i);
      colok(y,10)=coli(i);
      colok(y,11)=colj(i);
      colok(y,12)=colk(i);
      colok(y,13)=coll(i);
      colok(y,14)=colm(i);
      colok(y,15)=coln(i);
      colok(y,16)=colo(i);
      colok(y,17)=colp(i);
      colok(y,18)=colq(i);
      colok(y,19)=colr(i);
      colok(y,20)=cols(i);
      colok(y,21)=colt(i);
      y=y+1;
  end
end
end

fclose(fid);
y=y-1;

end
```

With the second function we try to extract all the amino-acids of each class that contain useful information. The code of this function is the next one:

```
function [arraynames,arraylabels,finalarraynames,finalarraylabels] = readtargetsLSWtemp()
[nombres]=textread('targetsLSW.txt','%s');
j=1;
```

```
y=1;
z=0;
label=1;
arraynames=[{}];
arraylabels=[{}];
stringswithX=[{}];
finalarraynames=[{}];
finalarraylabels=[{}];


fid=fopen('log.txt','a');
fprintf(fid,'%0s\n','-Classes detected and strings containing X');
fid1=fopen('targetsLSW.txt','w');

asize=size(nombres,1);

for i=1:1:asize,
    if(strcmp(nombres{i},' '))
    else
        if ((nombres{i}=='#'))
        fprintf(fid,'%0s\n',nombres{i+1});
        z=z+1;
        else
            string=nombres{i};
            ssize=size(string,2);
            for j=1:1:ssize,
                if (string(j)=='X'),
                    fprintf(fid,'%0s\n',string);
                    stringswithX=[stringswithX;{string}];
                end
            end
            arraynames=[arraynames;{string}];
            arraylabels=[arraylabels;{z}];
        end
    end
end

bsize=size(arraynames,1);
csize=size(stringswithX,1);

for i=1:1:bsize,
    r=0;
    for j=1:1:csize,
        string1=arraynames{i};
        string2=stringswithX{j};
        if strcmp(string1,string2),
           arraylabels{i}=0;
           r=1;
        end
    end
end

for h=1:1:bsize,
    stringsing=arraynames{h};
    q=arraylabels{h};
    if (q==0)
    else
        fprintf(fid1,'%0s\n',arraynames{h});
        string3=arraynames{h};
        finalarraynames=[finalarraynames;{string3}];
        string4=arraylabels{h};
        finalarraylabels=[finalarraylabels;{string4}];
        for g=(h+1):1:bsize,
            stringdob=arraynames{g};
                if (strcmp(stringsing,stringdob)),
                    arraylabels{g}=0;
                end
            end
        end
    end

fclose(fid);
fclose(fid1);

end
```

Later the next step to be followed had to be to design a function that develops the Gram-Schmidt orthogonalization. For this, we used a function that was inserted in another one (that will be explained later) whose mission is to develop the rank of the variables. The Gram-Schmidt function is the next one:

function A = mgrscho(A)

```
%MGRSHO Modified Gram-Schmidt orthogonalization procedure.
% -For a basis of fundamentals on classical Gram-Schmidt process, procedure
% and its origin. Please see the text of the m-file cgrsho you can download
% from www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=12465
% The classical Gram-Scmidt algorithm is numerically unstable, mainly
% because of all the successive subtractions in the order they appear. When
% this process is implemented on a computer, then the vectors s_n are not
% quite orthogonal because of rounding errors. This loss of orthogonality
% is particularly bad; therefore, it is said that the (naive) classical
% Gram–Schmidt process is numerically unstable. If we write an algorithm
% based on the way we developed the Gram-Schmidt iteration (in terms of
% projections), we get a better algorithm.
% The Gram–Schmidt process can be stabilized by a small modification.
% Instead of computing the vector u_n as,
%
%     u_n = v_k - proj_u_1 v_n - proj_u_2 v_n -...- proj_u_n-1 v_n
%
% it is computed as,
%
%     u_n = u_n ^n-2 - proj_u_n-1 u_n ^n-2
%
% This series of computations gives the same result as the original formula
% in exact arithmetic, but it introduces smaller errors in finite-precision
% arithmetic. A stable algorithm is one which does not suffer drastically
% from perturbations due to roundoff errors. This is called as the modified
% Gram-Schmidt orthogonalization process.
% There are several different variations of the Gram-Schmidt process
% including classical Gram-Schmidt (CGS), modified Gram-Schmidt (MGS) and
% modified Gram-Schmidt with pivoting (MGSP). MGS economizes storage and is
% generally more stable than CGS.
% The Gram-Schmidt process can be used in calculating Legendre polynomials,
% Chebyshev polynomials, curve fitting of empirical data, smoothing, and
% calculating least square methods and other functional equations.
%
% Syntax: function mgrscho(A)
%
% Input:
%    A - matrix of n linearly independent vectors of equal size. Here, them
%        must be arranged as columns.
% Output:
%    Matrix of n orthogonalized vectors.
%
% Example: Taken the problem 18, S6.3, p308, from the Mathematics 206 Solutions
% for HWK 24b. Course of Math 206 Linear Algebra by Prof. Alexia Sontag at
% Wellesley Collage, Wellesley, MA, USA. URL address:
% http://www.wellesley.edu/Math/Webpage%20Math/Old%20Math%20Site/Math206sontag/
% Homework/Pdf/hwk24b_s02_solns.pdf
%
% We are interested to orthogonalize the vectors,
%
%   v1 = [0 2 1 0], v2 = [1 -1 0 0], v3 = [1 2 0 -1] and v4 = [1 0 0 1]
%
% by the modified Gram-Schmidt process.
%
% Vector matrix must be:
%   A = [0 1 1 1;2 -1 2 0;1 0 0 0;0 0 -1 1];
%
% Calling on Matlab the function:
%   mgrscho(A)
%
% Answer is:
%
% ans =
%        0    0.9129    0.3162    0.2582
%   0.8944   -0.1826    0.3162    0.2582
%   0.4472    0.3651   -0.6325   -0.5164
%        0         0   -0.6325    0.7746
%
% NOTE.- Comparing the orthogonality of resulting vectors by both classical
%   Gram-Schmidt and modified Gram-Schmidt processes, using floating point
%   format with 15 digits for double and 7 digits for single. We found that
%   during the process, with the modified one there exists smaller errors.
%
%                         Gram-Schmidt Process
%              -------------------------------------------------------
%      Vectors        Classical                    Modified
%     ------------------------------------------------------------------
%       A1-A2   -8.326672684688674e-017        -8.326672684688674e-017
%       A1-A3    1.665334536937735e-016         1.110223024625157e-016
%       A1-A4    1.110223024625157e-016         1.110223024625157e-016
%       A2-A3    5.551115123125783e-017        -1.110223024625157e-016
%       A2-A4   -5.551115123125783e-017        -5.551115123125783e-017
%       A3-A4              0                            0
%     ------------------------------------------------------------------
%
%
% Created by A. Trujillo-Ortiz, R. Hernandez-Walls, A. Castro-Perez
%            and K. Barba-Rojo
%         Facultad de Ciencias Marinas
%         Universidad Autonoma de Baja California
%         Apdo. Postal 453
%         Ensenada, Baja California
%         Mexico.
%         atrujo@uabc.mx
%
% Copyright. September 30, 2006.
%
% To cite this file, this would be an appropriate format:
% Trujillo-Ortiz, A., R. Hernandez-Walls, A. Castro-Perez and K. Barba-Rojo. (2006).
%   mgrscho:Modified Gram-Schmidt orthogonalization procedure. A MATLAB file.
%   [WWW document]. URL http://www.mathworks.com/matlabcentral/fileexchange/
%   loadFile.do?objectId=12495
%
% References:
% Gerber, H. (1990), Elementary Linear Algebra. Brooks/Cole Pub. Co. Pacific
%     Grove, CA.
% Wong, Y.K. (1935), An Application of Orthogonalization Process to the
%     Theory of Least Squares. Annals of Mathematical Statistics, 6:53-75.
%

if nargin ~= 1,
   error('You need to imput only one argument.');
end

[m n]=size(A);

for j= 1:n
   R(j,j)=norm(A(:,j));
   A(:,j)=A(:,j)/R(j,j);
   R(j,j+1:n)=A(:,j)'*A(:,j+1:n);
   A(:,j+1:n)=A(:,j+1:n)-A(:,j)*R(j,j+1:n);
end
return,
```

Next step was to join all the useful information extracted from the documents analysed and make a relation between the amino-acids and their respective biochemical properties. The function used to accomplished this mission was the next one:

```
function
[classes,totalnames,numberofclasses]=targetsLSWparams2files(colok,finalarraynames,arraylabels,finalarrayl
abels)
%fid1=fopen('targetsLSWparamsclass1.txt','w');
%fid2=fopen('targetsLSWparamsclass2.txt','w');
classes=[{}];
onlynames=[{}];
label=0;
totalnames=0;
numberofclasses=0;
v=1;
long=length(finalarraynames);

for h=1:1:long,
   string=finalarraynames{h};
   long2=length(string);
   if (long2<15),
      classes=[classes;{string}];
      numberofclasses=numberofclasses+1;
   else
      onlynames=[onlynames;{string}];
   end
end

fid=zeros(numberofclasses,1);

for z=numberofclasses:-1:1
   d=classes(v);
   fid(v,1)=fopen(d{1},'w');
   v=v+1;
end

v=1;
asize=size(colok,1);

for h=1:1:long,
   string=finalarraynames{h};
   long3=length(string);
   if (long3<15),
      label=label+1;
      v=1;
   else
      totalnames=totalnames+1;
      v=1;
```

```
for i=1:1:long3,
  if (strcmp(string(i),'A'))
    for h=1:1:asize,
      stringp=colok(h,2);
      for z=numberofclasses:-1:1
        d=classes(v);
        fprintf(fid(v,1),'%0s ',stringp{1});
        v=v+1;
      end

      v=1;

      %fprintf(fid1,'%0s ',stringp{1});
      %fprintf(fid2,'%0s ',stringp{1});
    end
  end

  if (strcmp(string(i),'C'))
    for h=1:1:asize,
      stringp=colok(h,3);
      for z=numberofclasses:-1:1
        d=classes(v);
        fprintf(fid(v,1),'%0s ',stringp{1});
        v=v+1;
      end

      v=1;

      %fprintf(fid1,'%0s ',stringp{1});
      %fprintf(fid2,'%0s ',stringp{1});
    end
  end

  if (strcmp(string(i),'D'))
    for h=1:1:asize,
      stringp=colok(h,4);
      for z=numberofclasses:-1:1
        d=classes(v);
        fprintf(fid(v,1),'%0s ',stringp{1});
        v=v+1;
      end

      v=1;
```

```
        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'E'))
    for h=1:1:asize,
        stringp=colok(h,5);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'F'))
    for h=1:1:asize,
        stringp=colok(h,6);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'G'))
    for h=1:1:asize,
        stringp=colok(h,7);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end
if (strcmp(string(i),'H'))
    for h=1:1:asize,
        stringp=colok(h,8);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'I'))
    for h=1:1:asize,
        stringp=colok(h,9);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'K'))
    for h=1:1:asize,
        stringp=colok(h,10);
        for z=numberofclasses:-1:1
```

```
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'L'))
    for h=1:1:asize,
        stringp=colok(h,11);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'M'))
    for h=1:1:asize,
        stringp=colok(h,12);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end
```

```
if (strcmp(string(i),'N'))
   for h=1:1:asize,
      stringp=colok(h,13);
      for z=numberofclasses:-1:1
         d=classes(v);
         fprintf(fid(v,1),'%0s ',stringp{1});
         v=v+1;
      end

      v=1;

      %fprintf(fid1,'%0s ',stringp{1});
      %fprintf(fid2,'%0s ',stringp{1});
   end
end

if (strcmp(string(i),'P'))
   for h=1:1:asize,
      stringp=colok(h,14);
      for z=numberofclasses:-1:1
         d=classes(v);
         fprintf(fid(v,1),'%0s ',stringp{1});
         v=v+1;
      end

      v=1;

      %fprintf(fid1,'%0s ',stringp{1});
      %fprintf(fid2,'%0s ',stringp{1});
   end
end

if (strcmp(string(i),'Q'))
   for h=1:1:asize,
      stringp=colok(h,15);
      for z=numberofclasses:-1:1
         d=classes(v);
         fprintf(fid(v,1),'%0s ',stringp{1});
         v=v+1;
      end

      v=1;

      %fprintf(fid1,'%0s ',stringp{1});
```

```
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'R'))
    for h=1:1:asize,
        stringp=colok(h,16);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'S'))
    for h=1:1:asize,
        stringp=colok(h,17);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'T'))
    for h=1:1:asize,
        stringp=colok(h,18);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end
        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'V'))
    for h=1:1:asize,
        stringp=colok(h,19);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end
if (strcmp(string(i),'W'))
    for h=1:1:asize,
        stringp=colok(h,20);
        for z=numberofclasses:-1:1
            d=classes(v);
            fprintf(fid(v,1),'%0s ',stringp{1});
            v=v+1;
        end

        v=1;

        %fprintf(fid1,'%0s ',stringp{1});
        %fprintf(fid2,'%0s ',stringp{1});
    end
end

if (strcmp(string(i),'Y'))
    for h=1:1:asize,
        stringp=colok(h,21);
        for z=numberofclasses:-1:1
            d=classes(v);
```

```
                  fprintf(fid(v,1),'%0s ',stringp{1});
                  v=v+1;
               end

               v=1;

               %fprintf(fid1,'%0s ',stringp{1});
               %fprintf(fid2,'%0s ',stringp{1});
            end
         end

v=1;

         if (i==long3)
            for z=numberofclasses:-1:1
                if (label==v)
                  d=classes(v)
                  fprintf(fid(v,1),'%0s ','1');
                else
                  fprintf(fid(v,1),'%0s ','-1');
                end
                v=v+1;
            end

            v=1;

            for z=numberofclasses:-1:1
                  d=classes(v);
                  fprintf(fid(v,1),'#%0s;',classes{label});
                  fprintf(fid(v,1),'%0s\n',string);
                  v=v+1
            end
         end
      end
   end
end

v=1;

for z=numberofclasses:-1:1
   d=classes(v);
   fclose(fid(v,1));
   v=v+1;
end
```

```
%fclose(fid1);
%fclose(fid2);

end
end
```

Finally it had to be developed the function that ranked the variables according to their influence to the outputs. The function contains the next steps:

1. To compute the correlation coefficients of the values obtained.
2. Set the x vectors according to their correlation coefficient.
3. To apply the Gram-Schmidt orthogonalization.
4. Remove the row with the best coefficient.

We must do the necessary iterations to rank all the vectors, following each these four steps.

The code of this function is the next one:

```
function
[A,propertiesCA,namesCA,ordercoraux,ordercolaux,cormax,colmax,ordercor,ordercol,B,C,Finalmatrix,Corr
elationn,ordercolumns] = Gramschmidtort(totalnames,y,numberofclasses,classes)
%properties=[{}];
%names=[{}]
v=1;
dim=(17*y)+1;
r=1;

        propertiesCA=cell(numberofclasses,1);
        for i=1:1:numberofclasses
                propertiesCA{i,1}=zeros(totalnames,dim);
        end

        namesCA=cell(totalnames,1);
        for i=1:1:totalnames
```

```
                namesCA{i,1}=cell(1,1);
        end
        %names(1,numberofclasses)=zeros(totalnames,1);

  S=dim;

  Correlationn=cell(numberofclasses,1)
        for i=1:1:numberofclasses
                Correlationn{i,1}=zeros(1,S);
        end

  ordercolumns=zeros(numberofclasses,S);

        fid=zeros(numberofclasses,1);

        for z=numberofclasses:-1:1
                d=classes(v);
                fid(v,1)=fopen(d{1},'r');
                v=v+1;
        end

        %fid=fopen('targetsLSWparamsclass1.txt','r');

        v=1;

        for z=numberofclasses:-1:1
                %for j=1:1:dim,
                for p=1:1:totalnames
                        c=fscanf(fid(v,1),'%f\n');
                        d=fgets(fid(v,1));
                        namesCA{p,1}{25,1}=d;
                        for i=1:1:dim
                                propertiesCA{v,1}(p,i)=c(i,1);
                                %namesCA(i,1)=d(i,1);
                        end
                        %j=j+1;
                        %names(i,1)=d(1);

                        %properties=[properties;{c}];
                        %names=[names;{d}];
                        %end
                end
                v=v+1;
        end
```

```
A=0;

v=1;

for z=numberofclasses:-1:1
        d=classes(v);
        fclose(fid(v,1));
        v=v+1;
end



%Opening results files------------------------

fid8=fopen('Targets-LSW-vars-classname.txt','a');
fprintf(fid8,'%s\n\n\n','-Colums related to the variables that are selected');
if (fid8<0)
        fprintf(1,'Error fid8\n');
end

fid9=fopen('Targets-LSW-classname.txt','a');
if (fid9<0)
        fprintf(1,'Error fid9\n');
end
fprintf(fid9,'%s\n\n\n','-Values of the variables selected');



%Start of orthogonalization-------------------

S=dim;
D=propertiesCA;
ordercol=zeros(numberofclasses,S);
ordercolaux=zeros(numberofclasses,S);
ordercor=zeros(numberofclasses,S);
ordercoraux=zeros(numberofclasses,S);

for i=1:1:numberofclasses
        for j=1:1:S
                ordercolaux(i,j)=j;
        end
end


Finalmatrix=cell(numberofclasses,1);
```

```
                for i=1:1:numberofclasses
                    Finalmatrix{i,1}=zeros(totalnames,S-1);
     end

for i=1:1:numberofclasses
                        for j=1:1:S
                                      ordercol(i,j)=j;
                        end
end

u=1;
g=1;

           while (S>2)


                        S=S-1;

                        A=zeros(totalnames,2);
                        Atotal=zeros(totalnames,dim-1);
                        colmax=zeros(1,numberofclasses);
                        cormax=zeros(1,numberofclasses);
                        z=1;
                        correlationaux=-50000000;
                        colcorrelationaux=0;

    B=cell(numberofclasses,1)
                  for i=1:1:numberofclasses
                                 B{i,1}=zeros(totalnames,S);
                  end

    nansdetector=cell(numberofclasses,1);
                  for i=1:1:numberofclasses
                                 nansdetector{i,1}=zeros(totalnames,S);
                 end

    C=cell(numberofclasses,1);
                  for i=1:1:numberofclasses
                                 C{i,1}=zeros(totalnames,S-1);
                  end



                        for b=1:1:numberofclasses
                cormax(1,b)=-100000;
        end

        for c=1:1:numberofclasses
                for d=1:1:S
                        %ordercol(c,d)=-20000000;
                        ordercor(c,d)=-20000000;
                end
        end

        for i=1:1:numberofclasses
                for j=1:1:S
                        A(1:totalnames,1)=D{i,1}(1:totalnames,j);
                        A(1:totalnames,2)=D{i,1}(1:totalnames,(S+1));
                        correlation=corrcoef(A);
                        %correlation=corrcoef(A(1:totalnames,1),A(1:totalnames,2));
                        ordercoraux(i,j)=correlation(1,2);
                        %ordercolaux(i,j)=j;
                        ordercolaux(i,j)=ordercol(i,j);


                        if (correlation(1,2)>cormax(1,i))
                                cormax(1,i)=correlation(1,2);
                                 colmax(1,i)=ordercol(i,j);
                                %colmax(1,i)=j;
                        end
                end
        end

for i=1:1:numberofclasses
   Correlationn{i,1}(1,r)=cormax(1,i);
end

r=r+1;

        for i=1:1:numberofclasses
                ordercor(i,1)=cormax(1,i);
                ordercol(i,1)=colmax(1,i);
        end

ordercor=ordercoraux;
ordercol=ordercolaux;
```

```
for i=1:1:numberofclasses
      for o=1:1:S
            if ((isnan(ordercor(i,o)))==1)
               ordercor(i,o)=0;
            end
      end
end


%-----------------ORDER--------------------------------


% making (n-1) passes
   for k=1:1:numberofclasses
      for e=1:1:S-1
         % comparing each number with the next and swapping
         for w=1:1:S-1
            if ordercor(k,w)<ordercor(k,w+1)
               % temp is a variable where the numbers are kept
               % temporarily for the switch
               temp=ordercor(k,w);
               tempcol=ordercol(k,w);
               ordercor(k,w)=ordercor(k,w+1);
               ordercol(k,w)=ordercol(k,w+1);
               ordercor(k,w+1)=temp;
               ordercol(k,w+1)=tempcol;
            end
         end
      end
   end

%--------------------------------------------------------


      for i=1:1:numberofclasses
         ordercolumns(i,g)=ordercol(i,1);
      end

      g=g+1;

      for i=1:1:numberofclasses
         for j=1:1:S

            %B{i,1}(:,j)=D{i,1}(:,ordercol(i,j));
```

```
    end
  end



  for i=1:1:numberofclasses
    Finalmatrix{i,1}(:,u)=propertiesCA{i,1}(:,ordercol(i,1));
  end

  for i=1:1:numberofclasses
                U=[ordercol(i,1);i];
    fprintf(fid8,'Column %0f in the class %f\n\n', U);
    H=[i;propertiesCA{i,1}(:,ordercol(i,1))];
    fprintf(fid9,'\n');
    fprintf(fid9,'In the class %0f:\n',i);
    H=[propertiesCA{i,1}(:,ordercol(i,1))];
    fprintf(fid9,'%f ',H);
  end

  fprintf(fid8,'\n\n');
  fprintf(fid9,'\n\n\n');

  u=u+1;

  for i=1:1:numberofclasses
    for j=1:1:S
      B{i,1}(:,j)=D{i,1}(:,j);
    end
  end



  for i=1:1:numberofclasses
    B{i,1}=mgrscho(B{i,1});
  end

for i=1:1:numberofclasses
    for o=1:1:totalnames
      for d=1:1:S
        if ((isnan(B{i,1}(o,d)))==1)
          B{i,1}(o,d)=0;
        end
      end
    end
  end
```

```
D=cell(numberofclasses,1);
        for i=1:1:numberofclasses
                    D{i,1}=zeros(totalnames,S);
        end



for i=1:1:numberofclasses
   D{i,1}(:,1:(S-1))=B{i,1}(:,2:S);
end



for i=1:1:numberofclasses
   D{i,1}(:,S)=propertiesCA{i,1}(:,length(propertiesCA{1,1}));
end



%for i=1:1:numberofclasses
 %   ordercol(i,1:(S-1))=ordercol(i,2:S);
 %end



ordercolauxxx=zeros(numberofclasses,S-1);
ordercolll=zeros(numberofclasses,S-1);
ordercorrr=zeros(numberofclasses,S-1);

for i=1:1:numberofclasses
   for j=1:1:(S-1)
                ordercorrr(i,j)=ordercor(i,(j+1));
                ordercolll(i,j)=ordercol(i,(j+1));
   end
end

ordercolauxxx=ordercolll;
ordercol=zeros(numberofclasses,S-1);
ordercor=zeros(numberofclasses,S-1);
ordercolaux=zeros(numberofclasses,S-1);
ordercoraux=zeros(numberofclasses,S-1);

for i=1:1:numberofclasses
   for j=1:1:(S-1)
   ordercol(i,j)=ordercolll(i,j);
   ordercor(i,j)=ordercorrr(i,j);
   ordercolaux(i,j)=ordercolauxxx(i,j);
   end
```

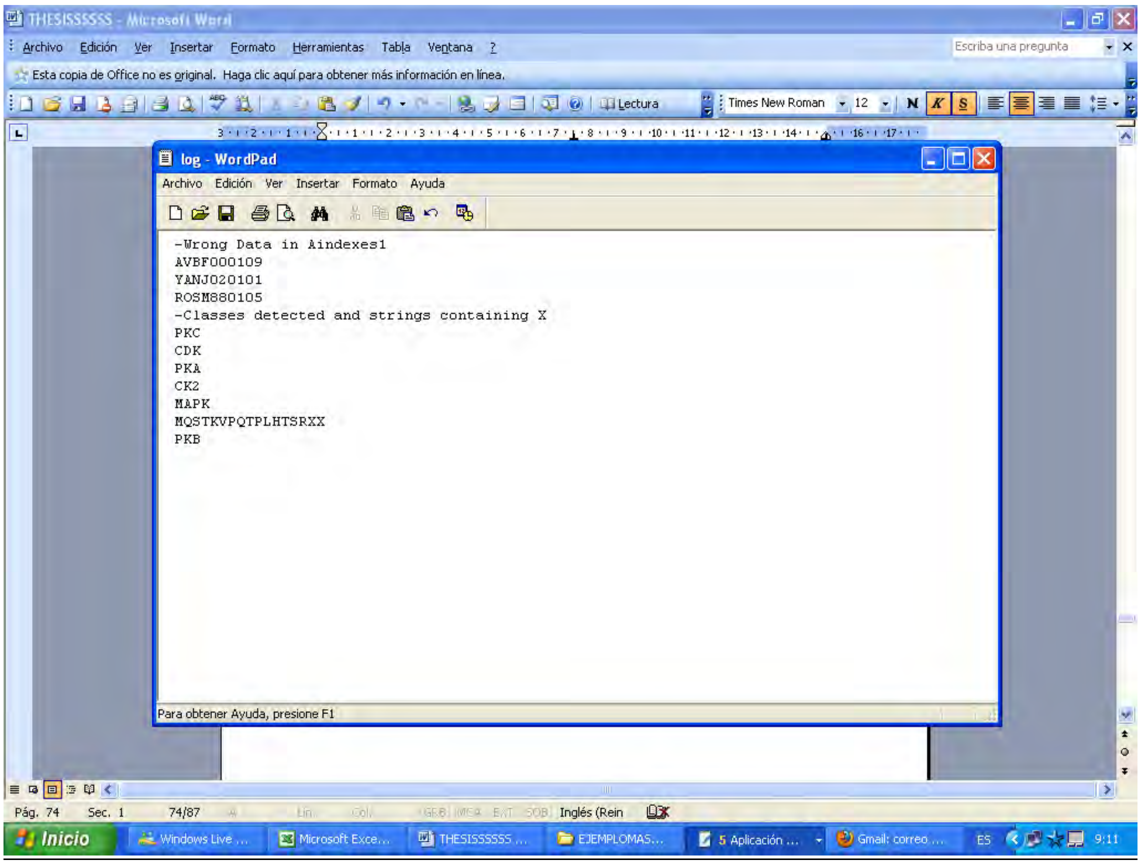
```
    end


end

for i=1:1:numberofclasses
     Finalmatrix{i,1}(:,u)=propertiesCA{i,1}(:,ordercol(1,1));
end

for i=1:1:numberofclasses
   U=[ordercol(i,1);i];
    fprintf(fid8,'Column %0f in the class %f\n\n', U);
    H=[i;propertiesCA{i,1}(:,ordercol(i,1))];
    fprintf(fid9,'\n');
    fprintf(fid9,'In the class %0f:\n',i);
    H=[propertiesCA{i,1}(:,ordercol(i,1))];
    fprintf(fid9,'%f ',H);
end


%Writing results-------------

        %E=cell(numberofclasses,1);
        %for i=1:1:numberofclasses
        %E{i,1}=zeros(totalnames,1);
        %end

        for t=1:1:numberofclasses
                    U=[ordercol(t,1);t];
                    %E=[propertiesCA{t,1}(1:totalnames,ordercol(t,1))];
                    %fprintf(fid8,'Column %0f in the class %f\n\n', U);
                    fprintf(1,'Column %0f in the class %f\n\n', U);

                    %fprintf(fid9,'\n\n');
                    %fprintf(fid9,'Values of the column %0f in the class %f\n\n', U);
                    for n=1:1:totalnames
                    %fprintf(fid9,'%f ', propertiesCA{t,1}(n,ordercol(t,1)));
                    end
        end



        %Closing files----------------
        fclose(fid9);
        fclose(fid8);

        %-----------------------------


end
```

12.2.- DOCUMENT FORMATS

The results have to be the documents mentioned before. The memory of the computer was not enough for running the full algorithm with the original input files, then, we will show the results considering an example consisted in a group of 18 amino-acids and 42 different characteristics that describe these amino-acids.

The documents obtained have the format of the next ones that appear in the following screen captures:

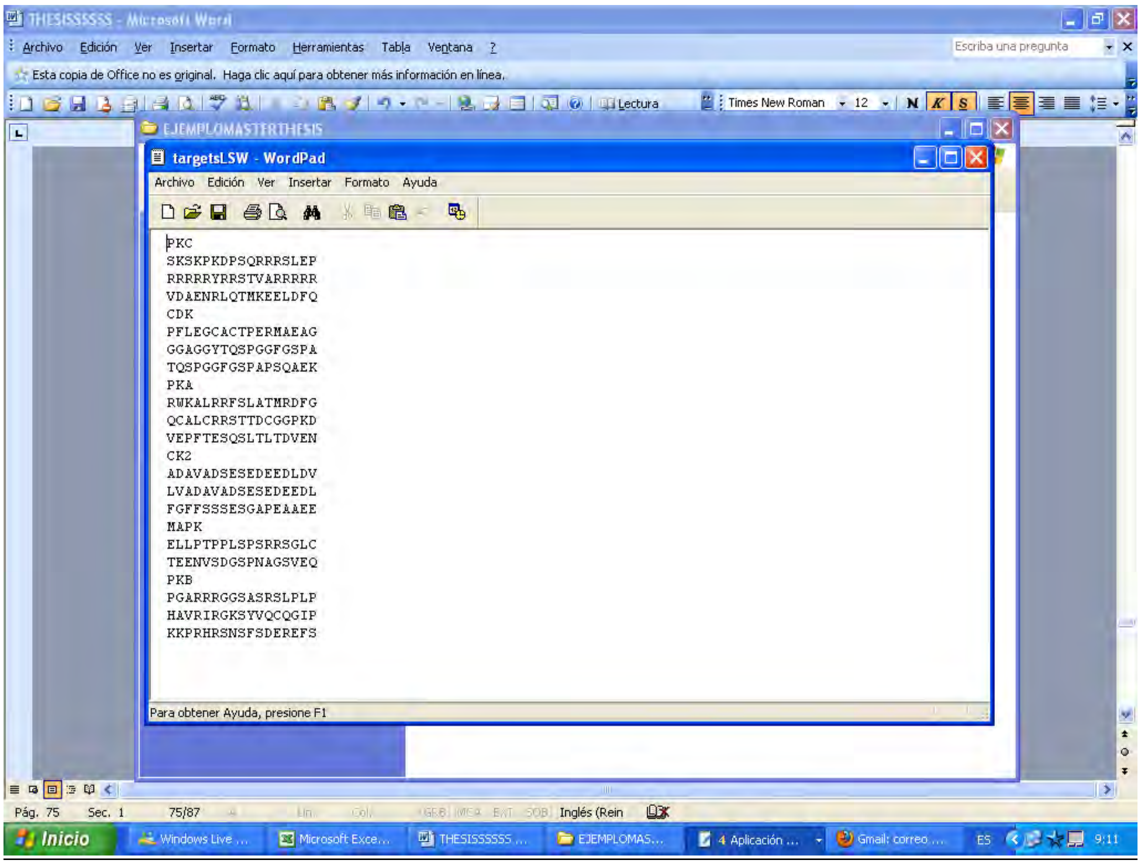
Document log.txt


THESISSSSSS - Microsoft Word
log - WordPad
Archivo Edición Ver Insertar Formato Ayuda
Para obtener Ayuda, presione F1

```
-Wrong Data in Aindexes1
AVBF000109
YANJ020101
ROSM880105
-Classes detected and strings containing X
PKC
CDK
PKA
CK2
MAPK
MQSTKVPQTPLHTSRXX
PKB
```

Fig.19: Text file log.txt.

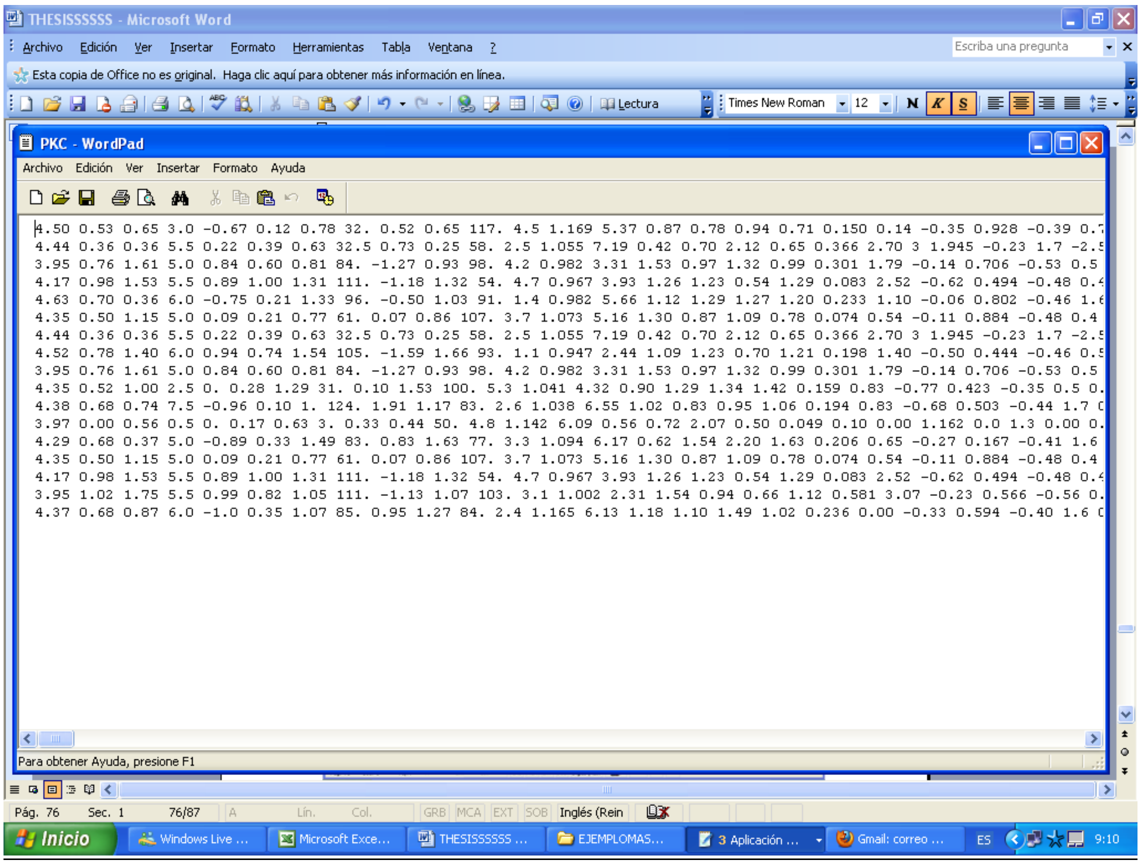
Document targetsLSW.txt



```
PKC
SKSKPKDPSQRRRSLEP
RRRRRYRRSTVARRRRR
VDAENRLQTMKEELDFQ
CDK
PFLEGCACTPERMAEAG
GGAGGYTQSPGGFGSPA
TQSPGGFGSPAPSQAEK
PKA
RWKALRRFSLATMRDFG
QCALCRRSTTDCGGPKD
VEPFTESQSLTLTDVEN
CK2
ADAVADSESEDEEDLDV
LVADAVADSESEDEEDL
FGFFSSSESGAPEAAEE
MAPK
ELLPTPPLSPSRRSGLC
TEENVSDGSPNAGSVEQ
PKB
PGARRRGGSASRSLPLP
HAVRIRGKSYVQCQGIP
KKPRHRSNSFSDEREFS
```

Fig.20: Text file targetsLSW.txt.

Document PKC.txt (as example)

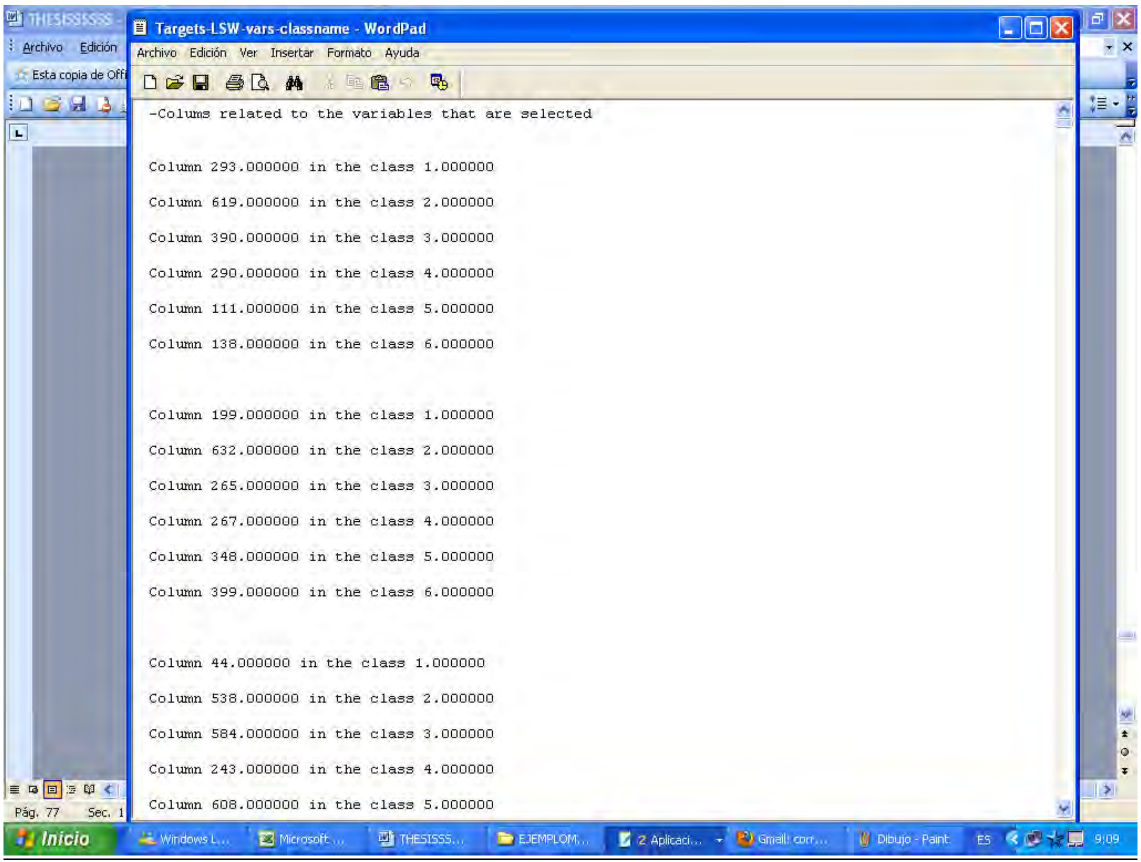


```
4.50 0.53 0.65 3.0 -0.67 0.12 0.78 32. 0.52 0.65 117. 4.5 1.169 5.37 0.87 0.78 0.94 0.71 0.150 0.14 -0.35 0.928 -0.39 0.7
4.44 0.36 0.36 5.5 0.22 0.39 0.63 32.5 0.73 0.25 58. 2.5 1.055 7.19 0.42 0.70 2.12 0.65 0.366 2.70 3 1.945 -0.23 1.7 -2.5
3.95 0.76 1.61 5.0 0.84 0.60 0.81 84. -1.27 0.93 98. 4.2 0.982 3.31 1.53 0.97 1.32 0.99 0.301 1.79 -0.14 0.706 -0.53 0.5
4.17 0.98 1.53 5.5 0.89 1.00 1.31 111. -1.18 1.32 54. 4.7 0.967 3.93 1.26 1.23 0.54 1.29 0.083 2.52 -0.62 0.494 -0.48 0.4
4.63 0.70 0.36 6.0 -0.75 0.21 1.33 96. -0.50 1.03 91. 1.4 0.982 5.66 1.12 1.29 1.27 1.20 0.233 1.10 -0.06 0.802 -0.46 1.6
4.35 0.50 1.15 5.0 0.09 0.21 0.77 61. 0.07 0.86 107. 3.7 1.073 5.16 1.30 0.87 1.09 0.78 0.074 0.54 -0.11 0.884 -0.48 0.4
4.44 0.36 0.36 5.5 0.22 0.39 0.63 32.5 0.73 0.25 58. 2.5 1.055 7.19 0.42 0.70 2.12 0.65 0.366 2.70 3 1.945 -0.23 1.7 -2.5
4.52 0.78 1.40 6.0 0.94 0.74 1.54 105. -1.59 1.66 93. 1.1 0.947 2.44 1.09 1.23 0.70 1.21 0.198 1.40 -0.50 0.444 -0.46 0.5
3.95 0.76 1.61 5.0 0.84 0.60 0.81 84. -1.27 0.93 98. 4.2 0.982 3.31 1.53 0.97 1.32 0.99 0.301 1.79 -0.14 0.706 -0.53 0.5
4.35 0.52 1.00 2.5 0. 0.28 1.29 31. 0.10 1.53 100. 5.3 1.041 4.32 0.90 1.29 1.34 1.42 0.159 0.83 -0.77 0.423 -0.35 0.5 0.
4.38 0.68 0.74 7.5 -0.96 0.10 1. 124. 1.91 1.17 83. 2.6 1.038 6.55 1.02 0.83 0.95 1.06 0.194 0.83 -0.68 0.503 -0.44 1.7
3.97 0.00 0.56 0.5 0. 0.17 0.63 3. 0.33 0.44 50. 4.8 1.142 6.09 0.56 0.72 2.07 0.50 0.049 0.10 0.00 1.162 0.0 1.3 0.00 0.
4.29 0.68 0.37 5.0 -0.89 0.33 1.49 83. 0.83 1.63 77. 3.3 1.094 6.17 0.62 1.54 2.20 1.63 0.206 0.65 -0.27 0.167 -0.41 1.6
4.35 0.50 1.15 5.0 0.09 0.21 0.77 61. 0.07 0.86 107. 3.7 1.073 5.16 1.30 0.87 1.09 0.78 0.074 0.54 -0.11 0.884 -0.48 0.4
4.17 0.98 1.53 5.5 0.89 1.00 1.31 111. -1.18 1.32 54. 4.7 0.967 3.93 1.26 1.23 0.54 1.29 0.083 2.52 -0.62 0.494 -0.48 0.4
3.95 1.02 1.75 5.5 0.99 0.82 1.05 111. -1.13 1.07 103. 3.1 1.002 2.31 1.54 0.94 0.66 1.12 0.581 3.07 -0.23 0.566 -0.56 0.
4.37 0.68 0.87 6.0 -1.0 0.35 1.07 85. 0.95 1.27 84. 2.4 1.165 6.13 1.18 1.10 1.49 1.02 0.236 0.00 -0.33 0.594 -0.40 1.6
```

THESISSSSSS - Microsoft Word
Archivo Edición Ver Insertar Formato Herramientas Tabla Ventana ?
Escriba una pregunta
Esta copia de Office no es original. Haga clic aquí para obtener más información en línea.
Lectura
Times New Roman
12
PKC - WordPad
Archivo Edición Ver Insertar Formato Ayuda
Para obtener Ayuda, presione F1
Pág. 76 Sec. 1 76/87 A Lín. Col. GRB MCA EXT SOB Inglés (Rein
Inicio
Windows Live ...
Microsoft Exce...
THESISSSSSS ...
EJEMPLOMAS...
3 Aplicación ...
Gmail: correo ...
ES
9:10

Fig.21: Text file PKC.txt.

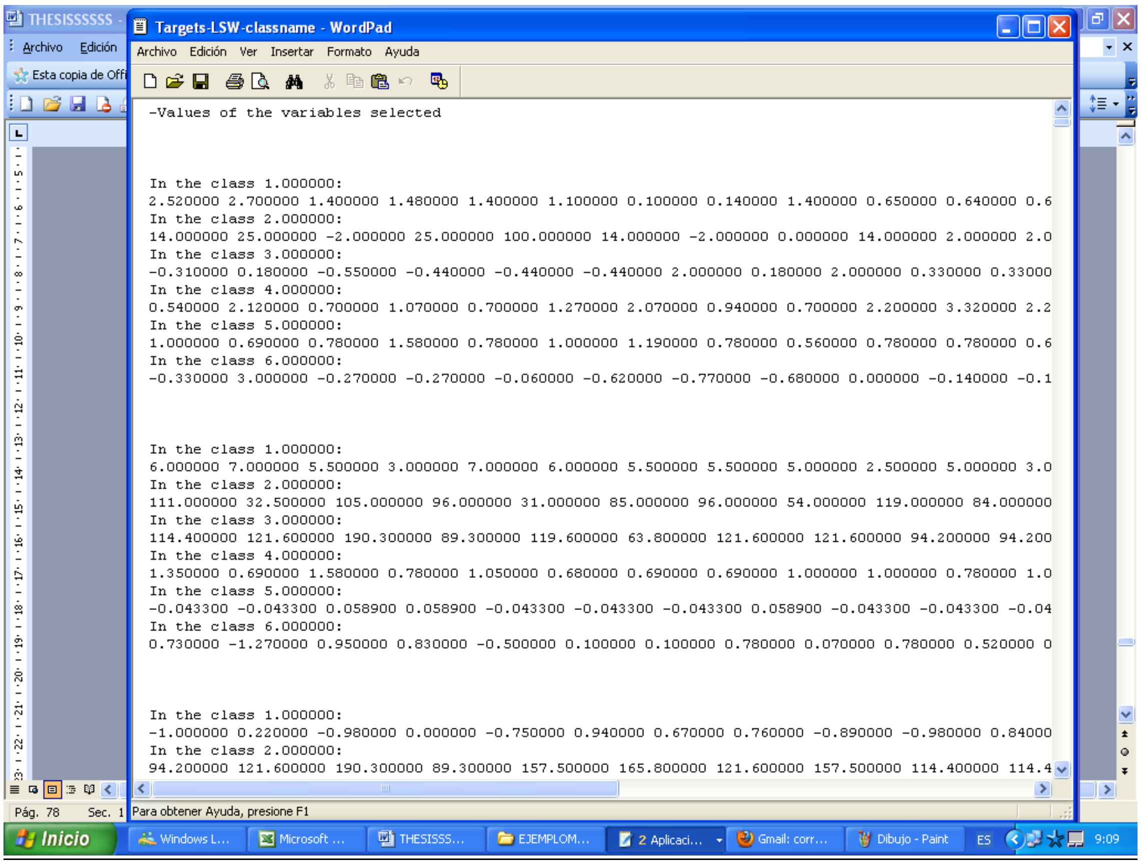
<u>*Document Targets-LSW-vars-classname*</u>


Targets-LSW-vars-classname - WordPad
Archivo Edición Ver Insertar Formato Ayuda

```
-Colums related to the variables that are selected

Column 293.000000 in the class 1.000000
Column 619.000000 in the class 2.000000
Column 390.000000 in the class 3.000000
Column 290.000000 in the class 4.000000
Column 111.000000 in the class 5.000000
Column 138.000000 in the class 6.000000

Column 199.000000 in the class 1.000000
Column 632.000000 in the class 2.000000
Column 265.000000 in the class 3.000000
Column 267.000000 in the class 4.000000
Column 348.000000 in the class 5.000000
Column 399.000000 in the class 6.000000

Column 44.000000 in the class 1.000000
Column 538.000000 in the class 2.000000
Column 584.000000 in the class 3.000000
Column 243.000000 in the class 4.000000
Column 608.000000 in the class 5.000000
```

Fig.22: Text file Targets-LSW-vars-classname.

Document Targets-LSW-classname


Targets-LSW-classname - WordPad
Archivo Edición Ver Insertar Formato Ayuda
Para obtener Ayuda, presione F1

```
-Values of the variables selected



In the class 1.000000:
2.520000 2.700000 1.400000 1.480000 1.400000 1.100000 0.100000 0.140000 1.400000 0.650000 0.640000 0.6
In the class 2.000000:
14.000000 25.000000 -2.000000 25.000000 100.000000 14.000000 -2.000000 0.000000 14.000000 2.000000 2.0
In the class 3.000000:
-0.310000 0.180000 -0.550000 -0.440000 -0.440000 -0.440000 2.000000 0.180000 2.000000 0.330000 0.33000
In the class 4.000000:
0.540000 2.120000 0.700000 1.070000 0.700000 1.270000 2.070000 0.940000 0.700000 2.200000 3.320000 2.2
In the class 5.000000:
1.000000 0.690000 0.780000 1.580000 0.780000 1.000000 1.190000 0.780000 0.560000 0.780000 0.780000 0.6
In the class 6.000000:
-0.330000 3.000000 -0.270000 -0.270000 -0.060000 -0.620000 -0.770000 -0.680000 0.000000 -0.140000 -0.1



In the class 1.000000:
6.000000 7.000000 5.500000 3.000000 7.000000 6.000000 5.500000 5.500000 5.000000 2.500000 5.000000 3.0
In the class 2.000000:
111.000000 32.500000 105.000000 96.000000 31.000000 85.000000 96.000000 54.000000 119.000000 84.000000
In the class 3.000000:
114.400000 121.600000 190.300000 89.300000 119.600000 63.800000 121.600000 121.600000 94.200000 94.200
In the class 4.000000:
1.350000 0.690000 1.580000 0.780000 1.050000 0.680000 0.690000 0.690000 1.000000 1.000000 0.780000 1.0
In the class 5.000000:
-0.043300 -0.043300 0.058900 0.058900 -0.043300 -0.043300 -0.043300 0.058900 -0.043300 -0.043300 -0.04
In the class 6.000000:
0.730000 -1.270000 0.950000 0.830000 -0.500000 0.100000 0.100000 0.780000 0.070000 0.780000 0.520000 0



In the class 1.000000:
-1.000000 0.220000 -0.980000 0.000000 -0.750000 0.940000 0.670000 0.760000 -0.890000 -0.980000 0.84000
In the class 2.000000:
94.200000 121.600000 190.300000 89.300000 157.500000 165.800000 121.600000 157.500000 114.400000 114.4
```

Fig.23: Text file Targets-LSW-classname.