

UNIVERSITY OF COIMBRA

DOCTORAL PROGRAM IN INFORMATION SCIENCE AND TECHNOLOGY

REAL TIME LEARNING IN INTELLIGENT SYSTEMS

---

## **Assignment #3 - Protein Classification**

---

JOAQUIM PEDRO BENTO GONÇALVES PRATAS LEITÃO - 2011150072

May 6, 2017

**Contents**

1	Introduction . . . . .	2
1.1	Objectives . . . . .	2
1.2	Methodology . . . . .	2
1.3	Document Outline . . . . .	3
2	Dataset Description . . . . .	3
3	Data Representation . . . . .	4
4	Support Vector Machines . . . . .	4
5	LASVM . . . . .	5
6	Conclusions . . . . .	5

# 1 Introduction

The protein classification problem is among the most important and fundamental problems in computational biology. In short, this problem can be defined as the task of classifying proteins into functional and structural classes based on *homology* (evolutionary similarity) of protein sequence data.

Several approaches can be taken to solve the protein classification problem, ranging from methods that analyse the protein's coding sequence in the genetic code, to more complex approaches that study the 3D structure of the proteins in order to perform such classification.

In the current assignment, a sequence similarity-based approach to this problem will be adopted. The algorithms and methods to be applied fall in the first set of practices mentioned in the previous paragraph. As such, the current work will focus on analysing the genetic sequence responsible for coding a given protein. Browsing solutions for this problem proposed in the literature Support Vector Machines appear as interesting techniques, achieving promising results [1, 2, 3].

## 1.1 Objectives

One major high-level objective can be identified in the present assignment: For any given protein determine its functional class by analysing its coding sequence.

This work will feature data from the *SCOP Database*[4], more precisely the *SCOP40* dataset<sup>1</sup>. This dataset contains genetic sequences responsible for coding proteins of 55 different families. Therefore, a more low-level objective for this work is to develop classifiers capable of identifying protein sequences belonging in each of the 55 families.

The current assignment also featured a secondary objective, consisting in comparing incremental with non-incremental classifier approaches.

## 1.2 Methodology

In light of the description of the assignment at hand, and its objectives, the following steps will be considered in the work to be performed:

1. Initial dataset preprocessing
2. Data representation
3. Classifier training
4. Results assessment

In an initial stage, a preprocessing of the *SCOP40* data was conducted, aiming to construct a training and testing dataset for each one of the 55 families in the dataset. It is worth mentioning that even though the *SCOP Database* does not provide separate training and testing datasets for each family, it proposes a list of instances to be used for training and testing, for each family. Therefore, at this stage, the preprocessing step consisted in

---

<sup>1</sup><http://pongor.itk.ppke.hu/benchmark>

analysing the provided mapping and in creating individual train and test files for each family.

Probably the most important step in this assignment, data representation can be seen as an advanced preprocessing step to transform the data to be feed to a classifier. It strongly conditions the final performance of the classifiers as an inappropriate representation does not allow for a good separation of the data, therefore leading to poor classification performance. In this work the *segmentation* was the representation technique implemented. Section 3 covers this technique in detail.

Following the representation classifiers needed to be properly trained and tested. In this work *incremental* and *non-incremental* classifiers were intended to be compared. In this sense two different but somewhat related classifiers were selected: *LASVM*, as the incremental classifier; and *Support Vector Machines (SVM)*, as the non-incremental classifier.

As each protein sequence under analysis must be assigned to one of 55 possible families, a multi-class classification problem is being considered. In this sense one classifier for each family must be develop. Considering that two different classifiers are intended to be compared, two classifiers were trained for each family (one *LASVM* and one *SVM*).

Finally, once the classifiers were properly created and trained, their performance was assessed with the testing dataset, following an analysis of the test results in order to determine which approach registered better results, if any. Considering the fact that the *SCOP40* is an highly unbalanced dataset (indeed, a fact that is very common in problems of this nature in computational biology) the different approaches were not compared with respect to their accuracy, but with respect to the *Area Under Curve* metric.

Concerning the classifiers' development and all *assignment-related* tasks, the  $R^2$  software environment was selected as the implementation and working tool.

### 1.3 Document Outline

The remainder of this document is organised as follows: Section 2 introduces and describes the dataset used in the current work. Section 3 addresses the data representation technique adopted in this assignment. Sections 4 and 5 cover the experimental results obtained with the two techniques being compared in this work. Finally, section 6 presents a reflective analysis of the obtained experimental results and identifies directions to be explored in future work.

## 2 Dataset Description

The *SCOP40* dataset contains information concerning the genetic coding sequences of different proteins and their corresponding families, being organised in two main files, as follows:

- *SCOP40mini.fasta* - Text file listing the different genetic coding sequences for several proteins. A unique identifier is also presented for each protein sequence.

---

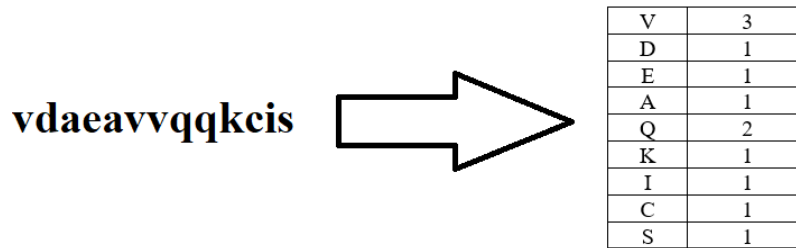
<sup>2</sup><https://www.r-project.org/>

- *SCOP40mini\_sequence\_minidatabase\_19.cast* - Text file containing a mapping between each family and the protein sequences of the *SCOP40mini.fasta* file to be used during classifier training or test.

### 3 Data Representation

Referenced in subsection 1.2, the representation technique implemented in the current work was the *segmentation* method. In short, this method consists in, for each protein sequence, count the number of occurrences of each individual nucleotide (that is, of each letter in the sequence).

For example, consider the following example protein sequence: "vdaeavvqqkcis". The segmentation method would output a total number of 3 occurrences for the "v" nucleotide, 1 occurrence for the "d" nucleotide, and so on. The only exception is for "x" nucleotides, which represent missing information about a nucleotide in the protein sequence, and therefore are discarded in this exercise. Figure 1 presents an example of the output of this method for a protein sequence.



V	3
D	1
E	1
A	1
Q	2
K	1
I	1
C	1
S	1

Figure 1: Example of the implemented segmentation representation for a protein sequence.

As it is not mandatory that each protein sequence contains all possible individual nucleotides, this method requires an initial passage for all the sequences in the dataset, in order to identify all individual nucleotides. The method's output will, therefore, present the number of occurrences of each of those nucleotides in the current protein sequence being analysed.

### 4 Support Vector Machines

*Support Vector Machines* (SVM) [5] are supervised machine learning algorithms applied in classification and regression tasks. During SVM training a model is built using a set of labelled samples (which can belong to one of two possible classes). Such model can then be used to classify new samples, that is, assign a new sample to a previously defined and learned class.

During training an hyperplane that assures the largest separation between instances of both classes is sought. The data points that lie closest to the hyperplane are called *support vectors* and determine the separation margin between the two classes. Because of their (usually) small distance to the separation hyperplane these are the most difficult data points to classify. Misclassification of some instances may or may not be allowed. Such behaviour is controlled by a *cost* parameter  $C$  that penalises misclassification.

Even though SVMs seek a linear hyperplane, and therefore perform a linear classification task, they have also been extensively applied in non-linear classification problems. When performing non-linear classification, SVMs need to map the samples into a higher dimensional space, where a linear separation hyperplane can be computed. Once such a hyperplane has been determined an inverse transformation can be performed, mapping the data back to the original space. Such mapping operation is defined as a *kernel*. For this reason, and since most *application* problems are non-linear, SVMs are considered *kernel methods*.

Several high-dimensional mappings, or *kernels*, have been proposed in the literature for SVMs, though typical kernels covered in SVM books include: *linear*, *polynomial*, *sigmoid* and *radial basis function* (RBF).

In the current work, SVMs featuring a RBF kernel are intended to be applied. RBF kernels are defined by the following mathematical expression:  $\exp(-\gamma * |x_i - x_j|^2)$ . Therefore, SVM training and classification performance will be influenced by the values defined for the cost and RBF parameters:  $C$  and  $\gamma$ , respectively.

As no prior knowledge of adequate values for these parameters has been obtained, optimal values must be searched. As suggested by Hsu *et al.* [6], a good practice to tune these parameters is to perform a grid search with exponentially growing sequences of  $C$  and  $\gamma$  (for example  $C = 2^{-5}, 2^{-4}, \dots, 2^5$  and  $\gamma = 2^{-15}, 2^{-14}, \dots, 2^3$ ).

As mentioned in 1.2, all algorithms used throughout the work were implemented in the *R* software environment. With respect to SVMs, the *e1071*<sup>3</sup> package was used, providing an interface in the R programming language for the popular SVM library, *LIBSVM* [7].

**FIXME:** Apresentar aqui os resultados! No fundo dizer quais foram as combinações de parâmetros que testámos e quais é que nos deram os melhores resultados para cada família; Apresentar valor médio de AUC das melhores abordagens nas 55 famílias

## 5 LASVM

**FIXME:** Explicação teórica + referência biblioteca, etc

**FIXME:** Apresentar aqui os resultados! No fundo dizer quais foram as combinações de parâmetros que testámos e quais é que nos deram os melhores resultados para cada família

## 6 Conclusions

## References

- [1] Christina Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch string kernels for svm protein classification. In *NIPS*, volume 15, pages 1441–1448, 2002.

---

<sup>3</sup><https://CRAN.R-project.org/package=e1071>

- [2] Nela Zavaljevski, Fred J Stevens, and Jaques Reifman. Support vector machines with selective kernel scaling for protein classification and identification of key amino acid positions. *Bioinformatics*, 18(5):689–696, 2002.
- [3] CZ Cai, LY Han, Zhi Liang Ji, X Chen, and Yu Zong Chen. Svm-prot: web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic acids research*, 31(13):3692–3697, 2003.
- [4] Alexey G Murzin, Steven E Brenner, Tim Hubbard, and Cyrus Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.
- [5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [6] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.
- [7] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.