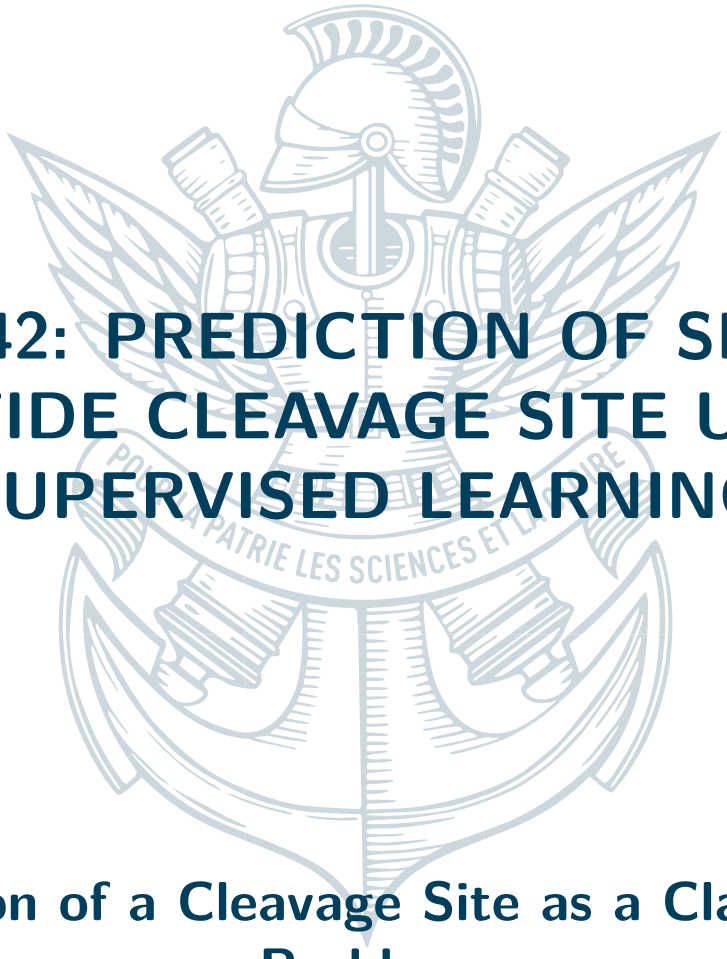


INF 442: PREDICTION OF SIGNAL PEPTIDE CLEAVAGE SITE USING SUPERVISED LEARNING

Localization of a Cleavage Site as a Classification Problem

May 12^{th}, 2023

Auguste CRABEIL, Julien GADONNEIX

ÉCOLE POLYTECHNIQUE

IP PARIS

INTRODUCTION

The goal of this report is to make a description of our works and approaches on the prediction of signal peptide cleavage site. Signal peptide is a short sequence of amino acids located near the N-terminal extremity of the protein. Its aim is to contain the information necessary to deliver the protein to its right destination in or out the cell. This step is crucial for survival. This peptide is then pulled out after being cleaved at a precise site and the location of this cleavage site is very interesting in medicinal research for instance. The rest of the protein is the mature form which is functional and ready to operate.
We will use hundreds of proteins whose cleavage sites are known in order to understand what it is made of first of all. Then, we will try to predict the cleavage sites of other proteins.
As the subject asked us, we firstly addressed the problem with a statistical approach and then, we used supervised learning throughout support vectors machine with kernels.
The datasets we used are made of hundreds of N-terminal extremities of protein. We have the portion of the proteins which begins at the N-terminal extremity with always a methionine (M) until 30 amino acids after the cleavage site.

In order to use our codes and re-carry out our tests (if you want to check them and try them :)), there is a Makefile for the compilations and you can use "./grader x", where x is the number of the test.

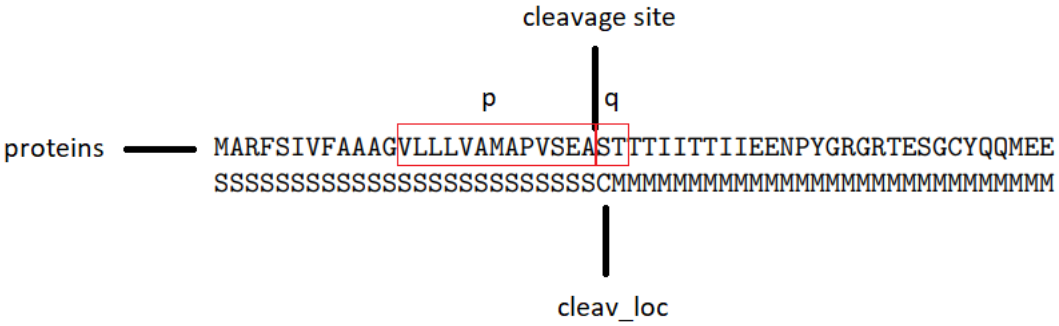
We are sorry for the length of the report due to some repetitions of the subject.

1 SIMPLE STATISTICAL MODEL USING A POSITION-SPECIFIC SCORING MATRIX

A protein is described by its sequence of amino acids, which is represented by a sequence of letters from the alphabet $\mathcal{A} = \{A, ..., Z\}$.
In previous studies, it has been demonstrated that the cleavage site of a protein can be characterized by the amino acids in its immediate vicinity. Consequently, this vicinity is defined as the p amino acids preceding and the q amino acids following the site. The work in reference suggests up to $p = 13$ and $q = 2$ but we will try to optimise these results.

We are facing a binary classification problem : knowing whether a sub-sequence of length $p + q$ is a cleavage site. In all of our subsequent work, we will test our models using cross-validation to ensure consistency and work with the same type of data.

In the data sets, there is a line under the proteins for the localization of the cleavage site. For each amino acid on the proteins, a letter indicates whether this amino acid belongs to the signal peptide (annotated with a letter S) or to the mature protein (annotated with a letter M). The first amino acid of the mature protein is annotated with a letter C and we implement it as the "cleav_loc" site.
We obtain the following figure :



The first method is just a simple statistical model. We examine whether the positioning of amino acids around the site is similar to that observed around known cleavage sites. To be more specific, let's consider $a \in \mathcal{A}$ and $i \in \{0, 1, \ldots, p + q - 1\}$. We define $f(a, i)$ as the observed frequency of amino acid a at the relative position i, given that a cleavage site is located at position p. In our implementation, we represent f using a matrix, where each amino acid is associated with a unique identifier (e.g., A for one, B for two, and so on).

Similarly, by tallying the occurrences across the entire length of the provided sequences, it is possible to calculate the observed overall background frequency, denoted as $g(a)$, of amino acid 'a' within the given dataset, irrespective of its position. However, it is important to note that the initial amino acid at the beginning of a sequence is typically an 'M' since it corresponds to the translation of the start codon. Additionally, to prevent any bias, we do not include the first

position when counting the amino acids. In our implementation, we represent g using a vector.

For all $a \in \mathcal{A}$ and $i \in \{0, \ldots, p+q-1\}$, we define $s(a,i) = \log(f(a,i)) - \log(g(a))$. Additionally, to handle the occurrence of zero counts, we introduce pseudo-counts by calculating

$$s(a,i) = \log\left(\frac{f(a,i)+\alpha}{N_1 + \alpha \cdot 26}\right) - \log\left(\frac{g(a)+\alpha}{N_2 + \alpha \cdot 26}\right)$$

where $\alpha \ll \frac{1}{N_2}$ and N_1 represents the total count of proteins ans N_2 represents the total count of amino acid. In our implementation, we represent s using a vector.

Finally, for a given word $a_0, \ldots, a_{p+q-1}$, $s(a_i, i)$ is associated with the frequencies of the amino acids at position i (normalized by the general frequencies of that amino acid), given that a cleavage site is located at position p. Subsequently, we can define a score for this word : $\sum_{i=0}^{p+q-1} s(a_i, i)$.

By applying a threshold, we can straightforwardly define a binary classifier. We implement multiple thresholds: the minimum score among all cleav_loc, the maximum score among all cleav_loc, the mean score of all cleav_loc, and finally, the best threshold (in terms of f-score) selected from a range between the minimum and maximum thresholds, with a step size of e (e is a given argument). These last selections are made in the functions "set threshold" in the "statistical model" class.

We observed that the two last described thresholds are the best ones, and they are very close in value. Applying cross-validation on the dataset "EUKSIG_13.red" using the last threshold, we obtained the following results in the form of a confusion matrix with $p = 12$ and $q = 2$.

```
                Predicted
                0       1
Actual  0       36331   502
        1       452     553

Error rate              0.0252127
False alarm rate        0.0136291
Detection rate          0.550249
F-score                 0.536893
Precision               0.524171
[allemagne projet]$
```

Figure 1: Test with the best threshold

2 SOME SVM KERNELS

We are facing a binary classification problem of two classes and it appears quite complicate and hard to understand according to what precedes. Thus, as asked in the subject, we will use a non-linear model for classification such as support vector machines using kernels. As suggested in the subject, we have an explicit mapping:

$$\phi : \left| \begin{array}{ccc} \mathcal{A} & \longrightarrow & \mathbb{R}^{26\times(p+q)} \\ a & \longmapsto & (0 \ldots 0\ 1\ 0 \ldots 0 \mid \ldots \mid 0 \ldots 0\ 1\ 0 \ldots 0)^T \end{array} \right.$$

Each segment is made of 25 "0" and 1 "1" at the place of the letter in the alphabet. The segment corresponds to the amino acids in the protein at the place of the segment in the vector. Such an encoding then allows different types of kernels which give different results. The type of kernel is a hyper-parameter and we will select it by cross-validation in order to optimize the results. However, the linear kernel simply counts the number of common letters between the two corresponding words which corresponds to part 1 and the other kernels are only slightly different. Henceforth, we introduced a substitution matrix which enables a comparison between letters more “accepting” than just binary. The BLOSUM62 matrix - here called M - represents the probabilities of mutation from one amino acid into another which thus alter the former binary comparison. A pseudo inner product can thus be defined:

$$S : \left| \begin{array}{ccc} \mathcal{A}\times\mathcal{A} & \longrightarrow & \mathbb{R} \\ (a,\ b) & \longmapsto & \sum_{j=1}^{k} M(a_i,\ b_i) \end{array} \right.$$

In the final step, we will use a probabilistic kernel based on the frequencies $s(a, i)$. More specifically, as stated in the problem, for two words $a = a_0, \ldots, a_{p+q-1}$ and $b = b_0, \ldots, b_{p+q-1}$, we define the kernel K as:

$$\log(K(a, b)) = \sum_{i=0}^{p+q-1} \phi_i(a_i, b_i)$$

where

$$\phi_i(a_i, b_i) = \begin{cases} s(a_i, i) + s(b_i, i) & \text{if } a_i \neq b_i \\ s(a_i, i) + \log(1 + e^{s(a_i,i)}) & \text{if } a_i = b_i \end{cases}$$

The construction of this kernel becomes more complex due to its dependency on a statistical model as an argument, specifically to obtain the values of $s(a, i)$. To ensure standardization and facilitate their usage as arguments in functions, we incorporated a statistical model into the construction of each kernel.

From now on, we will deal with our tests on the detection of protein cleavage sites and the choice of hyper-parameters or meta-parameters. For the SVM, we mainly drew inspiration

from the exercise session on this topic and as we had to adapt and change most of the programs, we also tried to implement our own SVM method in addition to the one from the library.

Let's have a look at the different test we carried out. Firstly, we try the statistical model as explained in the first part (test 1 in the "grading" class). Then, we used our own and SVM method and the one from the python library (tests 2 and 3) with $C = 1.0$ and $learningrate = 0.01$ (these values don't change for the others tests). The results were very close and the calculation times were quite similar.

		Predicted	
		0	1
Actual	0	31120	5525
	1	142	858
Error rate			0.150538
False alarm rate			0.150771
Detection rate			0.858
F-score			0.232426
Precision			0.13442

(a) Our own SVM

		Predicted	
		0	1
Actual	0	31175	5470
	1	146	854
Error rate			0.149183
False alarm rate			0.14927
Detection rate			0.854
F-score			0.233206
Precision			0.135041

(b) Python SVM

Figure 2: Tests with different SVM methods

For these tests, we chose a linear kernel with $p = 13$ and $q = 2$ and we implemented the cross-validation on the data set of eukaryotic proteins. However, for the rest of the tests, we used the SVM method from python as we believe it is more optimized.
With the following tests, we will try to optimize some parameters and the first one is the choice of the kernel among all the possible ones with the dot product and the pseudo-dot product (test 4). Some parameters of the kernel such as $p = 13$, $q = 2$, $gamma = 1/n$, $degree = 2$ or $coef0 = 1.0$ are fixed with common values to reduce the number of possibilities. These tests were achieved with the prokaryotic proteins data set as it is smaller and there are many tests. We will keep the best kernel for the rest and, as with the threshold determination test and future tests, the 'best' is in terms of f-score which is biased towards positives... Here is the result of the best kernel which is the polynomial:

		Predicted	
		0	1
Actual	0	6046	386
	1	33	107
Error rate			0.0637553
False alarm rate			0.0600124
Detection rate			0.764286
F-score			0.338073
Precision			0.217039

Figure 3: Test with the best kernel

Now, we want to optimize two other parameters which are p and q (tests 5 and 6) and we will keep the best values for the rest. For these tests, as the length of the words change for each values of p and q, we had to keep a fixed number of word per protein. We adopted the same strategy as for the training set and took one correct and one incorrect word per proteins. This is the reason why the results are very different. These tests and the rest are performed using the eukaryotic proteins data set as it is larger. Here are the results with the best kernel which are $q = 4$ and $p = 11$:

		Predicted	
		0	1
Actual	0	917	83
	1	180	820

Error rate	0.1315
False alarm rate	0.083
Detection rate	0.82
F-score	0.861797
Precision	0.908084

(a) Test with the best p

		Predicted	
		0	1
Actual	0	934	66
	1	174	826

Error rate	0.12
False alarm rate	0.066
Detection rate	0.826
F-score	0.87315
Precision	0.926009

(b) Test with the best q

Figure 4: Tests with some optimized hyper-parameters

This next test is a crucial one as it takes all possible words for each protein - as before tests 5 and 6 - and then, keep the more likely for each protein. Here, we used our own SVM method in order to get the result of the classifier before applying the function *sign*. It gives us the rate of good answer (test 7) and at the state of the project we got about 50%.

This last test evaluates the performance of the custom probabilistic kernel defined with the statistical computations (test 8). However, the results were not very accurate with this kernel, so we tried it with the product of the probabilistic kernel and the pseudo RBF kernel and with a weighted sum of them, but it was not clearly better. We used the prokaryotic proteins data set which is smaller because the calculations with custom kernels are very slow.

3 CODE EXPLANATIONS

3.1 Dataset class

The "dataset prot" class has only one constructor, one destructor and getters. However, the constructor sets up many useful tools for the rest of the programs.
Firstly, it reads the data and store them in a vector of strings containing the list of amino acids of each protein. Then , it creates a vector of integers containing the index of the first amino acid of the mature protein. This index is located at the same position in the vector than the string in the first vector. It also creates a vector of integers containing the length in amino acid of each sequence we have in the data. Eventually, it creates a vector containing all the integers between O and the number of sequence we have minus 1 in order to subdivide the data set for the r-fold cross-validation. This vector is shuffled to avoid the case where proteins are sorted and we can use it by dividing it into r contiguous sub-vector.
There also are methods to get the maximum value of p and q.

3.2 Kernel class

The "kernel prot" class has one constructor, one destructor, getters and many methods.
Indeed, this class implements all the kernels we used for this project including the classic ones such as the linear, the polynomial or the sigmoid, but also their versions with the pseudo inner product using the BLOSUM62 matrix. Eventually, it also implements the kernel using the statistical model.

3.3 Confusion matrix class

The "confusionmatrix prot" class is the same as usual. It includes one constructor, one destructor, getters and many methods.
This class implements all the classic calculations for the various statistical scores of our classifier and we added an option enabling us to add confusion matrix for the cross-validation.

3.4 SVM class

This class is the most important one of our project. It includes a constructor, a destructor, getters, setters and many methods.

The constructor uses the mapping defined above in order to transform the dataset into vectors of length $26 \times (p+q)$ of "0" and "1" by choosing $p+q$ consecutive amino acids in a protein. It uses the vector from the dataset class for the cross-validation in order to construct a training set and a test set. Moreover, to train properly our classifier we needed to have both correct and incorrect cleavage sites. Thus, we firstly decided to take every possible word of length $p+q$ for each protein. However, the training set became too big and the calculation times too long and we decided to choose the correct word (the one with p amino acids before the cleavage site and q after) and one incorrect word chosen at random. The boolean *all* enables us to choose which option we prefer for the training set but for the test set, we take every possibility.
Then, there are methods to compute the kernel matrix and the bias β_0 and the train method computing the coefficients α from the Representer theorem. This last method implements the projected gradient ascent on α. We adapted all these methods form the exercise session.
Eventually, the last methods are the tests. As written before, for the test sets we take every possibility for each protein as we are not supposed to know where the cleavage site is. There is one method that uses the python script for the SVM library and one that uses our own SVM method. Another method allows us to test our classifier with another data set. There also is probably the most important test method of this project which takes as input a dataset and outputs a vector of integers with the most likely cleavage sites.

3.5 SVM with sklearn

This script is written in Python language. We implemented it in order to also have a SVM method more optimized from the library.
The trick is that we have to transport the data from our C++ scripts in the python script and then transport them back in the C++ scripts. For this aim, we write the data in CSV files that we delete at the end of the operations, once we collected the results. Eventually, we also have in this class the option to scale the data but after the tests, we saw that the scaling was not useful.

3.6 Statistical model class

This class encompasses all the necessary code to implement the simple statistical model described in the first section of this report.
The constructor is responsible for constructing the matrix representing the function $s(a, i)$. Following the cross-validation approach, we train the statistical model using "r-1" parts of the dataset and evaluate its performance on the final part.
Additionally, there are methods available to set the threshold, as outlined in the first section.
Lastly, the "best_place" function determines the most probable location for the cleavage site by taking a protein as an argument.

CONCLUSION

All this demonstrates that the sequence of amino acids for the cleavage site follows indeed a pattern. However, it is quite blurry and not obvious as the classifiers that we set up struggled to get passable results. The different approaches that we explored through this project enabled us to use various statistical tools and obtain different crucial results.
Even though, many results were obtained, they are incomplete and there still is much work to do for optimizations of the tests. For instance the values of the coefficients for the kernels or for the SVM method can also be optimized.
Eventually, we carried out many tests but many others could also be achieved. Thus, in the future we could try to compare the different data sets which is possible thanks to one of the test method.

This project was really a pleasure as both of us are interested in Biology and Computer science. It helped us to master data analytic techniques in a fascinating domain.