Project INF442-2 (for X18)

Prediction of signal peptide cleavage site using supervised learning

difficulty **

contact: Philippe.Chassignet@polytechnique.edu

1 Introduction

Protein targeting is the mechanism by which some proteins are directed to their appropriate destinations in the cell or outside. The delivery process is carried out, based on information contained in the protein itself. This information is often identified as a short sequence of amino acids located near the N-terminal side of the protein. This sequence, known as the *signal peptide*, extends more or less toward the C-terminal side of the protein, up to a position named the *cleavage site*. Once a protein had been properly directed, its N-terminal part is generaly cleaved out at this position, while the C-terminal part leads to the protein as observed in its mature form. Correct targeting of proteins is crucial for life and the identification of signal peptids and their cleavage site is of interest to the design of new drugs.

2 Assignment

We consider the problem of predicting the position of cleavage site in proteins, relying on patterns learned from a training set. Here a protein is known as its *primary sequence*, i.e. the sequence of its amino acids, which is given starting from the N-terminal side. Each amino acid is denoted by a letter code and, as there are 20 standard amino acids, plus some peculiarities, most of upper case letters are used in this encoding. Let $\mathcal{A} = \{\texttt{A}, \ldots, \texttt{Z}\}$ be this alphabet.

For instance, the following sequence is the beginning of a protein, where the cleavage site is marked as the bond between the two underlined `AR` amino acids :

`MASKATLLLAFTLLFATCIARHQQRQQQQNQCQLQNIEALEPIEVIQAEA...`

Then, for the purpose of this project, one will simply work on such sequences of letters. An annex web page [1] gives a few sets of many sequences, for which the cleavage position is also given. One has now to program some learning algorithms, to tune them, and to evaluate their performance. A simple statistical model, will be used first to establish a reference. One will then try to improve accuracy with some specific kernel functions for Support Vector Machines. Guidelines follow.

3 Localization of a cleavage site as a classification problem

It has been shown in [2] that the cleavage site may be characterized by amino acids in its close neighborhood. Then, this neighborhood is defined as the p amino acids before and the q after. Note that the cleavage site is not an amino acid but the bond between two consecutive amino acids. So, values p and q define a neighborhood of $p + q$ amino acids in length. The work in reference suggests up to $p = 13$ and $q = 2$, but different values may also be experimented as meta-parameters of the problem.

We are facing a binary classification problem. Given any whole protein sequence $(a_i)_{i=0,\ldots,\ell-1}$, and any position j , where $p \leqslant j \leqslant \ell - q$, the word $a_{j-p}a_{j-p+1}\ldots a_{j-1}a_j\ldots a_{j+q-1} \in \mathcal{A}^{p+q}$ should be enough to decide whether bond at position j, between a_{j-1} and a_j, is a cleavage site or not.

In some cases, one may get several positive answers along a same sequence. This problem can be considered as an extension to the project

3.1 Simple statistical model using a Position Specific Scoring Matrix

Over a set of N sequences, with a known cleavage site for each, one can first count $c(a, i)$, the number of occurrences of each amino acid $a \in \mathcal{A}$, at every position $i \in \{-p, \ldots, q-1\}$, relative to the corresponding cleavage site. Then, for each a and i, let define $f(a, i) = c(a, i)/N$, the observed frequency of amino acid a at the relative position i.

In a same way, by counting over the whole length of given sequences, one can compute the observed general background frequency $g(a)$ of amino acid a in the given set, regardless of the position. However, it must be noticed that the very first amino acid at the beginning of a sequence is almost always an `M`, because it corresponds to the transcription of the *start* codon. Also, one will not count letters on this first position to avoid a bias.

These frequencies will be used as estimated probabilities to compute the probability of a given word to be located at a cleavage site, under an independent model. We rather use the logarithm of probabilities to go on additive calculations.

Then, $\forall\, a \in \mathcal{A}, \forall\, i \in \{-p, \ldots, q-1\}$, we define $s(a, i) = \log(f(a, i)) - \log(g(a))$. Also, as zero counts may occur, pseudocounts [3] must be used. Finally, for any word $w = a_0a_1\ldots a_{p+q-1}$, the score defined as $\sum_{i=-p}^{q-1} s(a_{p+i}, i)$ may tell whether w is the neighborhood of a cleavage site or not. A simple thresholding (to be tuned) is then enough to define a simple binary classifier.

3.2 Some kernels for SVM

Tutorial [4] is a good introduction. One must implement and experiment some classification algorithms with SVM around the following ideas. It should be easy to achieve better results than with the previous statistical model.

Kernels for SVM are basically positive (semi-)definite functions $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, used as a generalization of the inner product for space dimension d. Instead, we need some kernel function $K : \mathcal{A}^{p+q} \times \mathcal{A}^{p+q} \to \mathbb{R}$, defined on words of length $p + q$. First solution is to have an explicit mapping $\Phi : \mathcal{A}^{p+q} \to \mathbb{R}^d$, which allows to define a kernel as $K(u, v) = \Phi(u) \cdot \Phi(v), \forall\, u, v \in \mathcal{A}^{p+q}$, where $\cdot$ is the inner product on $\mathbb{R}^d$. As d may be large, an interesting property of some kernels is that one does not need to explicitly calculate Φ and, then, the inner product.

A direct approach for using SVM on fixed length words, consists in encoding each word as a point in a euclidian space. A simple way to encode a single letter is to form a vector of length 26, containing only *zeros*, but a *one* at the i-th position when encoding the i-th letter of the alphabet. Then the encoding of every word of n letters is the vector of length $26.n$, obtained by concatenating the n vectors for the letters of this word. Such an encoding then allows a straight linear SVM, or one of the popular polynomials and Gaussian RBF kernels.

One can observe that the dot product of two vectors, as defined by the previous encoding, simply counts the number of common letters between the two corresponding words. This leads to a dedicated implementation with a kernel defined as a similarity function of two words. Then, one should retrieve exactly the same results, as for the extensive encoding.

For a more flexible criterion, one can use the similarity defined by a substitution matrix [5]. Substitution matrices are built on the observed mutations of amino acids within proteins having a similar function. This yields a matrix of scores of similarity $M(x, y)$ between any pair (x, y) of amino acids. The annex web page [1] gives access to many examples of such matrices, with a kind of gradation in the similarity. Then, to get a similarity score $S(a, b)$ between two words,

$a = a_0 \dots a_{n-1}$ and $b = b_0 \dots b_{n-1}$, one may define $S(a,b) = \sum_{i=0}^{n-1} M(a_i, b_i)$, the sum of the corresponding letter scores along the two words. It is also a simplification of a more general alignment algorithm, but where no insertion/deletion is considered. Note that such a score is not related to a distance and that it does not satisfy Mercer's conditions. But it is safe to use it as a pseudo dot product to define a RBF kernel, for instance.

As another approach, one can use a probabilitic kernel, based on the corrected frequencies $s(a,i)$ from 3.1. See [6] for details. In short, for two words, $a = a_0 \dots a_{p+q-1}$ and $b = b_0 \dots b_{p+q-1}$, the logarithm of such a kernel can be defined as :

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

where

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

At last, since the two kernels defined above are based on very different properties, one can also try to combine them (as a weighted sum or as a product) into a single kernel.

4 Expected work

So, this project requires learning from a given sample set and then predicting whether any given word is the location of a cleavage site or not. For that, you have to write program (or many), which allows the choice of a kernel and the tuning of meta-parameters, as the size of the neighborhood around the clivage site, the C parameter of SVM and any other parameter, for example the parameter of a RBF kernel.

You have to use an existing SVM library, but implementing your own kernels. You will see the usage of `C++ libsvm` library during TD8, but a large part of the work can be started before. Note that `libsvm` also exists for `Java` and `scikit-learn` for `Python` also allows the definition of custom kernels. Then, these 3 langages are possible (and allowed) for this project. Execution time is important as tuning a model will require many consecutive runs.

One can then slide a window onto a given sequence, examining in turn each position as a potential cleavage site, to locate the real one, or all those classified as positive, as you can possibly locate more that one for some sequence. So, another program must implement your *best* classifier, already tuned, take a series of sequences as an input file, and output the predicted positions of cleavage site. It will be evaluated and discussed during the defense, on an entry that is kept secret for now, of course.

Références

[1] `https://www.enseignement.polytechnique.fr/profs/informatique/Philippe.Chassignet/19-20/CLEAVAGE/index.html`

[2] Gunnar von Heijne, *A new method for predicting signal sequence cleavage sites*, Nucleic Acids Research, Vol. 14, No. 11, pp. 4683–4690, 1986.

[3] `https://en.wikipedia.org/wiki/Additive_smoothing`

[4] Asa Ben-Hur, Cheng Soon Ong, Sören Sonnenburg, Bernhard Schölkopf and Gunnar Rätsch, *Support Vector Machines and kernels for Computational Biology*

[5] `https://en.wikipedia.org/wiki/Substitution_matrix`

[6] Jean-Philippe Vert, *Support Vector Machine prediction of signal peptide cleavage site using a new class of kernels for strings*, proc. of the 7th Pacific Symposium on Biocomputing, pp. 649-660, 2002.