Kernel methods for binary classification of DNA sequences

Virgile Dine *Omar Souaidi* *Antoine Van Biesbroeck*

Team Name : *Euh.*

Public Score : 0.64266 (rank 53) & Private Score : 0.67533 (rank 5)

1 Introduction

This report presents our team's work on the project of the MVA's Kernel Methods for Machine Learning course. The objective of this project was to predict whether a DNA sequence region is binding site to a specific transcription factor. More precisely the goal was to predict the labels for 3 datasets of 1000 sequences and we were provided a 2000 sequences train set for each to fit our algorithms.

All the codes we implemented and executed during this project are available at the following github address : `https://github.com/omarsou/kernel_method_kaggle_challenge`.

In the two following sections we discuss about some kernel first and classifier then we considered for our problem. Afterward we present our final model and our experiments in section 4, before providing a conclusion in section 5

2 Kernel for DNA sequences

As the main principle of kernel methods is to project the data on a space that would describe them the good way, our first task consisted into the research and the implementation of some kernel that may be adapted to DNA sequences.

The kernels we have considered are really common in the literature, we present them in the three following subsections.

2.1 Spectrum kernel

The spectrum kernel [1] extracts from sequences all their sub-sequences of a given length k. Therefore, two sequences are compared by comparing the number of sub-sequences they have in common. If x and x' denote two DNA sequences and $\phi_u(x)$ denotes the number of occurrences of sub-sequence u in x, then the spectrum kernel evaluated in x and x' is the following:

$$K^{SP}(x,x') = \sum_{u\in\mathcal{A}^k} \phi_u^{(k)}(x)\phi_u^{(k)}(x') = \langle \phi^{(k)}(x), \phi^{(k)}(x')\rangle$$

where $\mathcal{A} = \{\text{'A','T','C',G'}\}$, $\phi^{(k)}(x) = (\phi_u^{(k)}(x))_{u\in\mathcal{A}^k}$.

2.2 Mismatch kernel

The mismatch kernel [2] is a variant of the spectrum kernel defined above which compares sub-sequences having a number of mismatch less or equal to a chosen parameter m. If x and x' are two DNA sequences and $\phi_{u,m}^{(k)}(x)$ denotes the number of occurrences of sub-sequences with at most m mismatch from u in x, then the mismatch kernel evaluated in x and x' is the following:

$$K^{MS}(x,x') = \sum_{u\in\mathcal{A}^k} \phi_{u,m}^{(k)}(x)\phi_{u,m}^{(k)}(x') = \langle \phi_m^{(k)}(x), \phi_m^{(k)}(x')\rangle$$

where $\phi_m^{(k)}(x) = (\phi_{u,m}^{(k)}(x))_{u\in\mathcal{A}^k}$.

2.3 Substring kernel

The substring kernel [3] compares sub-sequences of length k of two sequences by allowing gaps between the elements of a sub-sequences. We choose a $\lambda > 0$ and denote $\psi_{u,\lambda}^{(k)}(x) = \sum_{u=x_{i_1}\dots x_{i_k}} \lambda^{i_k - i_1 + 1}$, this way the substring kernel evaluated in two DNA sequences x and x' is:

$$K^{SSK}(x,x') = \sum_{u\in\mathcal{A}^k} \psi_{u,\lambda}^{(k)}(x)\psi_{u,\lambda}^{(k)}(x') = \langle \psi_\lambda^{(k)}(x), \psi_\lambda^{(k)}(x')\rangle$$

where $\psi_\lambda^{(k)}(x) = (\psi_{u,\lambda}^{(k)}(x))_{u\in\mathcal{A}^k}$.

2.4 Local Alignment kernel

The Local Alignment kernel consist in finding the best alignment between two sequences. We introduce a score function $s(\pi(x,x'))$ for $\pi(x,x')$ an alignment between two sequences x, x'. Then for a parameter $\beta > 0$ the Local Alignment kernel is defined as :

$$K^{LA}(x,x') = \sum_\pi \exp(\beta s(\pi(x,x')))$$

2.5 Discussion and choices

We finally focused on the first kernel, the spectrum kernel. Because our implementation of the mismatch kernel was too slow, because the spectrum kernel is much faster ($O(k(|x| + |x'|))$ than the substring kernel ($O(k|x||x'|)$) even with dynamic programming and because the local alignment kernel gave poor results.

3 Classification algorithms

On the top of choosing a proper reproducing kernel, a task we had was to select a classification algorithm to label our dataset. We focused on the most popular classifiers, we have implemented them and we present them in the following sub-sections.

3.1 Principal Component Analysis

The principle is to find a low dimensional space corresponding to the principal characteristics of n pieces of data $x_i^T \in \mathbb{R}^d$. Then we got a $n \times d$ data matrix X that we must normalize w.r.t. the columns. We still write X the normalized matrix. The PCA consists in taking the SVD of X : $X = U\Sigma V^T$. We plunge the pieces of data into the order k score space by looking at $t(x)^T = x^T V_k$ where V_k is the matrix obtained by taking the first $k < d$ columns of V. For example, if x is orthogonal to the span of the k principal component v_i, $\|t(x)\| = 0$; and $\|t(x)\| = \|x\|$ if x is in it. The idea is then to work in the score space which has a lower dimension than the original space in order to be quicker with other machine learning algorithms.

3.2 Logistic Regression

Logistic Regression is the following optimization problem :

$$\hat{f} \in \underset{f\in\mathcal{H}}{\arg\min} \left\{ \frac{1}{n}\sum_i \ln(1 + e^{-y_i f(x_i)}) \right\} + \frac{\lambda}{2}\|f\|_{\mathcal{H}}$$

where x_i are the data, $y_i \in \{-1, 1\}$ the labels, and $\mathcal{H}$ the RKHS of some p.d. kernel.

3.3 Support Vector Machine (SVM)

SVM is a quadratic program looking for α in the following problem :

$$\min_{\alpha,\xi} \left\{ \frac{1}{n}\sum_i \xi_i + C\alpha^T \mathbf{K}\alpha \right\}$$

s.t. $\xi_i \leq 0 \wedge y_i[\mathbf{K}\alpha]_i + \xi_i - 1 \leq, i = 1, \dots, n$
where $\mathbf{K} = (K(x_i, x_j))_{i,j}$.

4 Experiments and results

4.1 Experimental framework

We got access to 2000 labeled training data of DNA sequences for each of the three type of unknown transcription factor binding site. And we had to predict on 1000 other testing data if the length 100 DNA sequence is a binding site for a specific unknown transcription factor. We decide to split the training data set into 80% for the experimental training data set and 20% for the validation data set. We will refer to this experimental data set by training set.

4.2 Experimentation

For each of the three types of data, we computed the spectrum kernel for $k \in \{5, 6, 7, 8, 9, 10, 11, 12\}$. For each of those kernels we tried the logistic regression for $\lambda \in \{0.01, 0.005, 0.001\}$ and $C \in \{1, 0.1, 0.01\}$. We saved the best two couples of parameters $((k_i^{(\alpha)}, \alpha_i)_{i=1,2}$ where $\alpha = |_C^\lambda)$ for each method on each data set, the ones that got the best performance on the validation set.
We obtained the following accuracy on the validation set :

	SVM	(k,C)	LoR	(k,λ)
$X_{tr,1}$	0.670	(10,0.1)	0.668	(7,0.01)
	0.665	(11,0.01)	0.675	(11,0.005)
$X_{tr,2}$	0.658	(7,0.1)	0.658	(7,0.01)
	0.658	(7,1)	0.650	(7,0.005)
$X_{tr,3}$	0.755	(8,0.1)	0.755	(9,0.001)
	0.750	(7,1)	0.755	(7,0.005)

Finally, we used some tricks. The first was to mix optimally those 4 models to obtain a better result on the validation set of each data set. Secondly, we thought that because we don't know a priori if the given sequences are read on the first or on the second branch of the DNA, we decided for the test set to look not only at the sequence but also at its reverse complementary to make our final prediction. For example, if $x = ATCCG$ we also looked at $RC(x) = ATTCG$.

5 Conclusion

During this challenge, we have tried to implement and test various kernels that could be applied to DNA sequences. We have used an ensemble method in order to get a decent score. We finally finished 5th on the private leaderboard despite the fact that we were 53th on the public leaderboard. This is because during our experiments, the results we get locally were quite good ($\sim$ 67,66,75 % accuracy) and we got the same score on the private leaderboard. The challenge was pretty exciting and interesting, it allowed us to be more familiar with machine learning tools that we had to code from scratch.

References

[1] C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for SVM protein classification. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 7:564–75, 02 2002.

[2] C. Leslie, E. Eskin, J. Weston, and W. Noble. Mismatch string kernels for SVM protein classification. *Advances in Neural Information Processing Systems*, 20:1417–1424, 01 2002.

[3] C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, 11 2004.