MACHINE LEARNING WITH KERNEL METHODS
MVA MASTER

Kaggle challenge
Team: LeSvIeUxRoUrOu
Public score: 0.658
Private score: 0.66133

Florent Bonnet
florent.bonnet@protonmail.com

Thibaut Germain
tgermain025@gmail.com

March 25, 2021

1 Introduction

1.1 Challenge Presentation

With the help of kernel methods, the aim of the challenge consists in solving a classification task: "predicting whether a DNA sequence region is binding site to a specific transcription factor".

Roughly speaking, transcription factors are proteins which aims to repress or initiate the transcription of some targets genes.

1.2 Dataset Presentation

We have access to three labeled datasets. Each dataset is associated to a specific transcription factor, it includes 2000 DNA sequences represented as strings (example: "AGCTTAA") with a corresponding binary mask (1 if it is bind, 0 else).

As data processing we have achieved a one-hot encoding mapping each character to a specific vector.

2 Method

2.1 Classifiers

We limited ourself to two kernel classifiers:

1. **Kernel Logistic Regression:** We have implemented the algorithm using the quadratic approximation as described in the lecture (pages 111-114).

2. **Support Vector Machine:** We used the dual formulation of the SVM:

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^n} \quad & \frac{1}{2}\alpha^T diag(y) K diag(y) \alpha - \mathbf{1}^T \alpha \\ \text{s.t.} \quad & 0 \leq \alpha \leq \frac{1}{2\lambda n} \\ & \alpha^T y = 0 \end{aligned}$$

where α is the dual variable, K the gram matrix, y the label vector, n the number of samples and λ the regularization factor. And where the first constraint has to be understood as a component-wise constraint.

We recover the classifier f via $f(x) = sign\left(\sum_i \alpha_i y_i K(x, x_i)\right)$.
To solve this convex optimization problem we have used the quadratic program solver of the library cvxopt.

Both algorithm have been checked on a moon shape dataset using linear and Gaussian kernels.

2.2 Kernels

We have implemented linear, gaussian and mismatch kernel (which includes the spectrum kernel). The last one is a kernel derived from large features spaces for biological sequences. The mapping function Φ counts the occurrences of predefined fixed length sub-sequences inside a target sequence with a tolerance of a given number of mismatch. The mismatched occurrences are scaled by a factor λ.

For efficiency, the mismatch kernel has been implemented using Numba library to fasten computation time. It explains the use of nested "for" loops in the code. Moreover, the kernel has been checked with the library STRKERNEL. The spectrum kernel derived from the mismatch kernel is not efficient with the current implementation, it would need its own implementation. We did not do it as we have discard it for predictions.

3 Training & Result

We have trained a classifier per train dataset. In order to estimate the performances of trained algorithms we have implemented a cross-validation methods with 5 folds. We have limited ourself to support vector machine method with Gaussian, spectrum and mistmach kernels. To tune hyperparameters, we have implemented a grid search method.

Here are the classifiers we have used for predictions:

Set	Classifier	Reg	Kernel	Length/Mismatch	Scaling	Normalized	CrossVal Score
0	SVM	1e-4	Mismatch	5/1	0.5	Yes	0.616 ± 0.017
1	SVM	1e-3	Mismatch	5/1	0.5	Yes	0.609 ± 0.044
2	SVM	5e-4	Mismatch	5/1	0.5	Yes	0.719 ± 0.022

Using those classifiers we have obtained at our last submission a public accuracy score of 0.658.
We have also implemented a kernel that is the average of a mix of spectrum, mismatch and Gaussian kernels. It leads to a more time consuming kernel without improving significantly the results on the test set besides the fact that it improved the results on the cross-validation process (especially on training set 1). We decided to only keep the mismatch kernel for the submission.

4 Further Improvements

We have approach this challenge only using kernels derived from large feature spaces. Nevertheless, they are kernels based on generative models or similarity measure which could be interesting to try as they will approach the classification task with different perspectives.

As well going toward ensemble methods might easily lead to improvements. Indeed, kernel methods are highly dependent on kernel selected and its hyper-parameters. Thus, we can easily implement different classifiers that will look at the classification task with different focus, leading to slightly different predictions which ensemble methods would be able to properly aggregate and provide more robust predictions. For instance, we could set up different SVMs with different mismatch kernels and then use a boosting algorithm such as Adaboost.
We could also, as we tried to do, do a weighted average of different kernels to allow a LASSO penalization on the choice of the kernels.
Ultimately, as we do not have infinite time and pretty limited computational resources, we did a pretty superficial hyperparameters search so it could be useful do to a real one.