Team ATCG: Kernel methods data challenge report

Armand Lacombe
ENS Paris-Saclay
armand.lacombe@polytechnique.edu

Hugo Touvron
ENS Paris-Saclay
hugo.touvron@polytechnique.edu

Abstract

In the context of the data challenge for the course "Kernel Methods for Machine Learning", we aim at tackling a sequence classification task : predicting whether or not a specific transcription factor is going to bind with a DNA sequence. For this, we focus on kernel methods such as SVM, with one special constraint : we avoid to use use any machine learning library. Hence, we design and tune all our estimators ourselves in order to maximize the algorithm accuracy.

1. Introduction

In this challenge, we aim identifying whether a DNA sequence contains a binding site for a given transcription factor or not. We have at our disposal 3 datasets, each corresponding to a transcription factor. We also have three sets of tests, each corresponding to one of the previous transcription factors. Our objective is to predict for each DNA sequence whether it contains a binding site. The metric used is accuracy.

Two things are essential to ensure a good performance on this type of problem:

First of all, it is important to have an embedding adapted to the problem. Indeed, it is necessary to be able to extract as well as possible the information from the sequences. We then need to use classifiers that will be able to extract information from our features. In a first part we will describe our embedding, and in a second part we will describe the classifiers we used.

2. Embedding

The binding site of transcription factors are pieces of non-coding DNA where the transcription factors can be fixed. These sequences will vary from one transcription factor to another and are relatively difficult to identify due to the variability that may exist in the size of the sequence and the nucleotides that compose it. Embedding is a crucial step in identifying binding sites. Indeed, many methods use only linear kernels with well adapted embedding and obtain good performance in this way. It is the case in [1] and [2]. We have implemented three types of embedding: K-mers, gapped K-mers [1] and mismatch [2].

2.1. DNA double helix

As DNA is composed of two helixes, we treat a motif and its complementary reverse in the same way. For example, we treat ATCG and CGAT in the same way.

2.2. k-mers

Embedding k-mers will correspond to identifying and counting all the motifs of a defined size in a DNA sequence. The problem with this approach is that we will have many features ($4^{motifs\ size}$). The information is not very well grouped: for example with this embedding let's suppose that we have the two binding sites of a transcription factor which are "ATTTCG" in 98% of cases and "ACTTCG" in the remaining 2% we will have difficulties to identify the second one. That's why we implemented gapped k-mers and mismatch.

2.3. Gapped k-mers

Gapped k-mers are based on the same principles as k-mers, but we also define how many nucleotides can vary in a sequence. We define the size of the motifs and the size of the gap. For example, we treat ATTCG and ACTCG in the same way because with a motif size of 5 and a gap of 1 both will add 1 to the gapped motif A_TCG. In this approach we still have many features but it has the advantage of grouping some motifs together which allows us to have some feature with more information which makes them more easily identifiable.

2.4. Mismatch

The mismatch is a mixture of the two previous approaches. We have the features of the k-mers but we will define a mismatch. And each motif will add 1 to the counter of each of its mismatches. For example, the ATC motif with a mismatch of size 1 add 1 to : ATC, TTC, CTC, GTC, ACC,..... This makes it possible to highlight the motifs that

are most likely to be binding sites and to take into account the proximity of the motifs.

2.5. Dimension reduction

The dimension reduction step is critical because of the huge number of features obtained from our embedding. We consider two options : Principal Components Analysis and Fast Independent Component Analysis.

One key concern about this step relies on the memory footprint of the matrices involved in dimension reduction. In particular, computing the whole correlation matrix, as it is routinely done, isn't possible since a 30000*30000 array would fit in the RAM of either Colab or a Kaggle kernel. Fortunately, the embedding happens to be very sparse, and invites to use sparse matrices. The linear algebra steps of both PCA and FastICA can be easily adapted to large matrices using Singular Values Decomposition.

3. Model

3.1. Model choices

Our model relies on three sub-models : an ordinary Support Vector Machine, a 2-SVM using squared hinge loss and a logistic regression.

No quadratic programming algorithm is used so as to optimize their respective objective functions since this step is very quick compared with data preprocessing. The plotting of the objective function evolution across the iterations gives a feedback about the training improvements. Experiments leads us to consider that only a small number of iterations are required so as to ensure model accuracy convergence, which accelerates the training.

In order to improve the model execution speed, the kernel is computed once for all, since this step is quite long. More generally, we aim at avoiding any computing redundancy.

The ordinary SVM happens to be a bit more efficient than 2-SVM, which is in average better than the logistic regression. The results obtained with those three models are merged using a simple weighing of their outputs, whose weights are cross-validated.

3.2. Model tuning

Due to the small number of test samlples and in a way to avoid as most as possible overfitting, we chose to minimize the number of submissions to the leaderboard. Hence, we resort largely on cross-validation methods so as to tune our hyperparameters.

A large range of parameters are to be tuned, using mainly grid-search : the kernel gamma constants, the embedding motif length, the number of gaps, the SVM margin, the PCA number of components, the models averaging weights, etc.

Despite our efforts towards algorithmic efficiency, our model execution speed keeps far behind that of *sklearn*. Thus the hyperperparameters are mostly tuned using prewriten libraries, and the best obtained values are kept for the handmade model.

It is necessary to cross-validate the results obtained with the handmade model too, in order to ensure that the results are consistent with that of the *sklearn* version before we compute predictions.

4. Results

We know as we upload it that our first attempt is suboptimal since the gamma values of both gaussian and laplacian kernel are poorly tuned, and recent experiments just showed that the motif lengths should be chosen higher. However, it leads to a prediction accuracy of 0.69733 on the public leaderboard, and 0.68466 on the private leaderboard leading to position 26.

5. Conclusion

In conclusion, we have implemented in this project different embedding methods and implement different classifiers to make the best possible use of the information contained in the features of our embedding. We have also implemented different dimension reduction methods to avoid being in the case where we have more features than observation. The combination in all these methods allowed us to reach an accuracy of 0.68466 over the private test set and thus be ranked 26/76. We note that we successfully avoided too much overfitting since we gained 12 places between the public and private leaderboard.

References

[1] Mahmoud Ghandi, Dongwon Lee, Morteza Mohammad-Noori, and Michael A. Beer. Enhanced regulatory sequence prediction using gapped k-mer features. *PLOS Computational Biology*, 10(7):1–15, 07 2014.

[2] Christina Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch string kernels for svm protein classification. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 1441–1448, Cambridge, MA, USA, 2002. MIT Press.