Kernel Methods for Machine Learning
Final Report - Kaggle team: FR-TG Présentez vous

François Remili and Théophane Gregoir

Public score : 67.866 % / Private score : 67.133% (ranked 10^{th} ex-aequo out of 74)

Downloadable link to repository

1. Introduction

The goal of this work is to develop from the ground up[1] a classification system to predict whether a DNA sequence region is a binding site to a specific transcription factor (TF) using Kernel Methods [5].

2. Methods

This work is focused on Kernel methods and according to Aronszajn's Theorem for p.d. kernel K there exists a mapping ϕ to a Hilbert space which can represent it[2]. In the case of the spectrum [4] kernel and mismatch [3] kernel, it is possible to embed the data using ϕ so that $K(x, y) = \langle \phi(x), \phi(x') \rangle_{\mathbb{R}^d}$. The embedding ϕ can be computed efficiently and makes the computations faster than direct kernel computation. Another advantage of the feature embedding approach is that several kernels can be applied on the embedded data in $\mathbb{R}^d$, while the linear kernel on the embedded data is equivalent to the actual kernel K.

2.1. Feature embedding

The spectrum kernel is treated as a particular case of the mismatch kernel with $m = 0$. The heart of the mismatch embedding processing is the hamming neighborhood algorithm which given a k-mer on the alphabet of size a and a maximum number of mismatch m computes the number $N_{k,m,a} = \sum_{i=0}^{m} \binom{k}{i}(a-1)^i$ of k-mers neighbors of the given k-mer up to the hamming distance m. The complexity of the algorithm is $O(kN_{k,m,a})$, the factor k coming from the building of each resulting k-mer.

All sequences are being processed by iterating through each of their $L-k+1$ k-mers (L is the sequence length), and updating the frequency of the k-mer and its neighbors (if mismatch) in the sequence spectrum. A hash-table is used to cache the results of the hamming neighborhood algorithm across the dataset and hence accelerate the computations.

2.2. Kernels

We implemented three p.d. kernels that we apply on the embedded data in $\mathbb{R}^d$. The **linear kernel** is the dot product in $\mathbb{R}^d$; applying it on the embedded data using the spectrum or mismatch embedding is equivalent to computing those kernels directly. The **Gaussian kernel** is also implemented. The computation of the Gaussian kernel gram matrix is made much faster by using a matrix version of the formula $||x - y||^2 = ||x||^2 + ||y||^2 - 2\langle x, y \rangle$ rather than computing pairwise squared distances. The **cosine similarity kernel** is the (p.d.) kernel obtained by normalizing the linear kernel [2], whereas the Gaussian kernel is already normalized by construction since $e^0 = 1$. We also implemented the **weighted sum kernel** by summing the other kernels (with potential repetition or omission). It is also p.d. as long as the weights are positive.

2.3. Models

We describe the algorithms we used to implement the kernelized version of three linear classification models.

The **ridge classifier** uses the closed form of the (weighted) regression problem, after mapping the two binary labels to 1 and -1. The decision function is $sgn(f(x))$ where $f(x)$ is the output of the ridge regressor for the data x. The close form solution is computed by solving a positive definite linear system using the Cholesky decomposition.

The implementation of the **logistic classifier** uses the iteratively reweighted least-square (IRLS) method to solve the standard linear logistic regression. This implementation has the advantage of reusing the code already written for the ridge classifier.

The **Support Vector Machine classifier** is implemented using the dual C-formulation and solving this quadratic program using the OSQP [6] solver through the cvxpy [1] library:

$$\max_{\boldsymbol{\alpha}} \; 2\boldsymbol{\alpha}^T \boldsymbol{y} - \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \;\text{ s.t. } 0 \leq \boldsymbol{y}_i \boldsymbol{\alpha}_i \leq C$$

2.4. Validation strategy

In order to optimize parameters without wasting Kaggle submissions, a validation pipeline was quickly built. This pipeline is decomposed into two main steps :

- First, for each of the 3 sets of DNA sequences independently, a Gridsearch method using 5-fold cross validation on 80% of the training set is operated to find the best hyperparameters.
- Then, predictions of the best model trained in the previous step are evaluated on the remaining 20% of the

[1] without the use of any machine learning library to build the models

[2] $\phi : \mathcal{X} \mapsto \mathcal{H} \;\; s.t. \; \phi(x, x') = \langle \phi(x, \phi(x') \rangle_{\mathcal{H}}$, $\mathcal{H}$ Hilbert space

training set in order to check how well the selected model generalizes.

Once, the best combination of feature embeddings, kernels, methods and hyperparameters is found for each the 3 sets, it is used to train an identical combination on the entire training set and predict on the testing set before submitting to the Kaggle Challenge.

3. Results

3.1. Regularization : key parameter

While finetuning our models with GridSearch, we discovered that the regularization parameter α (or C for SVM) appeared to be driving performances.

To illustrate this regularization impact, a fixed combination model (SVM Classifier on a Gaussian kernel with 7-mers including 1 misplacement) was run 10 times per α value on the first set of DNA sequences (see Figure 3.1). Four phases are distinguishable :

- $\alpha \in]0, 10^{-5}]$: the model is completely overfitting on the training set (with 100% train accuracy),
- $\alpha \in]10^{-5}, 10^{-4}]$: the model generalizes more until it reaches its best test accuracy,
- $\alpha \in]10^{-4}, 10^{-2}]$: the model becomes too simple and both accuracies fall,
- $\alpha \in]10^{-2}, \infty]$: the model is completely underfitting.

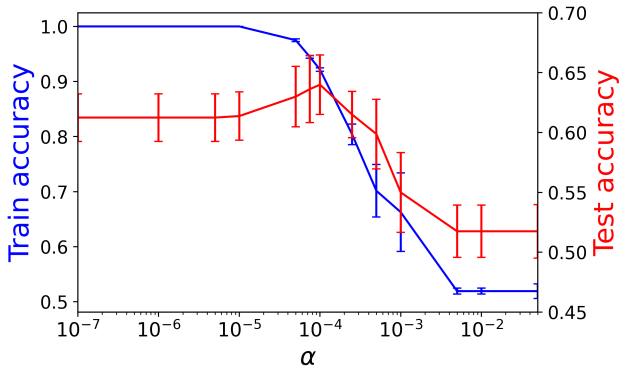

Figure 1. On the left axis, in blue, evolution of train accuracy depending on regularization factor α. On the right axis, in red, evolution of test accuracy depending on regularization factor α. Experiments were repeated 10 times for each value of α: the median is plotted and error bars represent standard deviation.

Therefore, for each model selected, the most important step is finding α^* (thanks to our GridSearch) for which the training accuracy starts dropping and the best performance is reached on testing set.

3.2. Building our best model

By comparing the labels in our best submissions, we understood that, although two different feature embeddings could enable to reach similar scores, they could capture very different insights on the DNA sequences.

We first identified the two individual embeddings which performed the best while having different predictions : 6-mers with 1 misplacement and 9-mers with 1 misplacement. Then, we decided to combine them in a sum of linear kernels in a logistic regression model as both were performing better using this combination.

Nevertheless, after observing Gram matrices composing the sum, we discovered that the mean of Gram matrix elements linked to 6-mers was around 10^2 times higher than the mean of the ones linked to 9-mers (because two random strings tend to have more 6-mers in common than 9-mers in common). Therefore, we decided to use a weighted sum of the two kernels with $\lambda_6 = 10^{-2}$ and $\lambda_9 = 1$.

	Validation Median Accuracy		
	Set 0	**Set 1**	**Set 2**
λ_6/λ_9 sum	65.5 ± 1.9	65.4 ± 2.0	75.2 ± 2.3
Sum	64.3 ± 1.8	65.9 ± 1.7	75.0 ± 1.6
9-mers 1 mis	65.8 ± 1.6	64.9 ± 1.3	74.8 ± 1.2
6-mers 1 mis	61.9 ± 1.8	64.9 ± 2.0	73.9 ± 1.9

Table 1. Ablation study focusing on the creation of our final model. All models are logistic regressions featuring linear kernels (potentially summed and weighted).Figures correspond to median accuracy over 10 different random splits : 75% training / 25% testing while uncertainty corresponds to standard deviation.

As highlighted by the table 3.2, our final model seems to benefit from both constitutive embeddings especially for sets 1 and 2. Weighting the sum enabled us to reach significantly better accuracy for set 0 which was one of the most difficult to predict. Nevertheless, as underlined by the standard deviations, performances are quite unstable (depending on the random splitting). This high variance can also be used as an argument to choose our final model because, when training for the submission, the model is confronted to the entire training set and should benefit from positive occurrences.

4. Conclusion

We were able to implement Kernel methods and analyze the influence of the hyper-parameters in order to reach good accuracy on the classification problem while applying our knowledge from the class.

References

[1] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016. 1

[2] Arnulf B.A. Graf and Silvio Borer. Normalization in support vector machines. In Bernd Radig and Stefan Florczyk, editors, *Pattern Recognition*, pages 277–282, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. 1

[3] Christina Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics (Oxford, England)*, 20:467–76, 04 2004. 1

[4] Christina Leslie, Eleazar Eskin, and William Noble. The spectrum kernel: A string kernel for svm protein classification. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 7:564–75, 02 2002. 1

[5] Vert Mairal. Machine learning with kernel methods. ENS Paris-Saclay, Master MVA Lecture, 2021. 1

[6] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. 1