Kernel methods for machine learning (Team: Lee Hwak)

Maximilien GERMAIN
MVA ENS Paris-Saclay
maximilien.germain@ens-paris-saclay.fr

Othmane MARFOQ
MVA ENS Paris-Saclay
othmane.marfoq@ens-paris-saclay.fr

Souhaib Attaiki
MVA ENS Paris-Saclay
sohaib.attaiki@ens-paris-saclay.fr

Abstract

The use of kernel methods for biological sequences has been widely investigated since Jaakkola et al.'s seminal paper (1998). In this data challenge we explored the task of DNA binary classification: predicting whether a DNA sequence region is binding site to a specific transcription factor.

1. Introduction

Transcription factors (TFs) are regulatory proteins that bind specific sequence motifs in the genome to activate or repress transcription of target genes. Large scale binding maps between genes and proteins can be experimentally constructed. Thus, all genomics can be classified into two classes for a TF of interest: bound or unbound. In this challenge, we will work with three datasets corresponding to three different TFs and build an associated discriminative binary model.

2. Classifiers

We used two different classical kernel algorithms to perform the binary classification tasks. Both methods rely on convex optimization problems solved for instance by projected gradient. By the representer theorem, the RKHS minimization is equivalent to an easier optimization problem in finite dimension which only depends on the kernel matrix K.

2.1. Kernel SVM

For this model we use the regularized hinge loss for training:

$$\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n}\max(1-y_i f(x_i),0)) + \frac{\lambda}{2}\|f\|^2$$

$$\iff \begin{aligned} \min_{\alpha\in\mathbb{R}^n} \quad & \frac{1}{n}\sum_{i=1}^{n}\xi_i + \frac{\lambda}{2}\alpha^\top K\alpha \\ \text{subject to} \quad & y_i[K\alpha]_i + \xi_i - 1 \geq 0,\ i=1,\dots,n \\ & \xi_i \geq 0,\ i=1,\dots,n \end{aligned}$$

2.2. Kernel Logistic Regression

In this method, we wish to minimize the regularized logistic loss:

$$\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n}\log(1+\exp(-y_i f(x_i))) + \frac{\lambda}{2}\|f\|^2$$

$$\iff \min_{\alpha\in\mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n}\log(1+\exp(-y_i[K\alpha]_i)) + \frac{\lambda}{2}\alpha^\top K\alpha$$

3. Kernels

3.1. Baseline: SVM + Gaussian kernel on the embeddings

The provided embedding are obtained using an auto-encoder similar to word2vec in order to construct a 100 dimensional representation of each DNA sequence. We used a gaussian kernel with SVM classifier based on those embeddings to classify the DNA sequences. This kernel has only one hyperparameter to tune, the value of σ. Using this model, we get a validation accuracy of about 59% which is better then random guess, in the sequel we denote this model as our baseline.

3.2. Spectrum Kernel

The spectrum kernel is a quite simple kernel, which consists in comparing the distance between two DNA sequences, by computing the number of k-mers present in both the sequences. We implemented a naive version of this spectrum, by computing first for each sequence a $|\mathcal{A}|^k$ dimensional vector where each entry is the occurrence of a possible k-mers formed by the Alphabet $\{A, C, G, T\}$. This kernel has only one hyperparameter which is k, the length of the k-mers considered by our model.

3.3. Mismatch kernel

Mismatch kernel is a variant of Spectrum Kernel, which measures the distance between two DNA sequences by computing the number of k-mers present in both the sequences, up to a certain mismatch m. This kernel is more complex than the Spectrum kernel, because it has two hyper-parameters, k and m. Moreover it needs more time to be computed and the computation time grows exponentially with the parameter m. Thus we limite ourselves to use values of m lower than 3 in our experiments. Our best result on the kaggle competition is obtained for this kernel with the parameters $k = 9$ and $m = 1$.

3.4. Kernel based on tf-Idf embeddings

The idea of this kernel is widely inspired by the tf-idf used for some NLP tasks as documents classification. It consists of considering each sequence as a document, and to consider its sub-sequences as words. However the notion of word is ambiguous in our case: we first proposed to take words of fixed size k, without overlapping, then we proposed to tolerate overlappings. We also tried random words size between two bound values. Then we processed our train set, like a corpus of documents, and we considered the embedding of each sequence.

4. Comparison with simple neural networks

We used neural networks as another baseline for comparison. These techniques for sequential data are widely used in NLP. We applied them over a one-hot embedding of the letters A,C,G,T.

- **Conv1D network:** An efficient model for NLP tasks uses dimension 1 convolutional neural network.
- **LSTM:** This other architecture relies on a modification of Recurrent Neural Networks architecture performed in order to prevent gradient vanishing.

Both LSTM based and Conv1D models didn't achieve good scores: Conv1D achieved an accuracy of 63.5% and LSTM achieved an accuracy of 55%. It is to note that we didn't spend much time to fine-tune those models. We could hope for better results with better hyperparameters. We wanted to point out that it is not easy to train neural networks on this dataset. Indeed, it is rather small, and we need to perform a lot of fine-tuning of the networks hyperparameters to reach results competitive with kernel based models. They require less parameters to tune.

Model	Global Score
Gaussian Kernel	58.6
Spectrum Kernel	69.9
Mismatch Kernel*	71.1
TF-Idf based kernel	58
Conv1D Model	63.5
LSTM Model	54.83

Table 1. Accuracy results on the train set

5. Results and comparison

To evaluate our models, we have split our data set into train (80%) and validation (20%), in order to select the best hyperparametrs of each model before submitting the results. For each model we have done a grid search to find the best optimizer and the value of its regularization on each part of the dataset. We have also done grid search to find the optimal hyperparametrs of each model. The **Table 1.** resumes the results of our experiments.

6. Conclusion

During the competition we have selected mismatch kernel as our best model, in fact using this kernel put us on top of the public leaderboard with a score of 73%, however we finished 13^{th} in the end of the competition with a score of about 70%. We don't really know why we have this slight overfitting problem, because we have never tuned the models parametrs based on the scores of the test set, instead we have only used the validation score to tune the parameters. Finally, the results were satisfying for use, and we had during this challenge the opportunity to implement machine learning algorithms, and to experiment several models and strategies.