DNA Sequence Classification with Kernel Methods

Killian Coustoulin – Killian.Coustoulin@dauphine.eu

Team: Bonsoir Public score: 0.737 (1st) Private score: 0.731 (1st)

GitHub Repository

1 Introduction

This report describes my work for the Kaggle data challenge in the course *Machine Learning with Kernel Methods*. The challenge is to implement kernel-based classification methods to predict whether a DNA sequence region is a binding site for a specific transcription factor. Predictions are performed on three datasets, each containing 2000 labeled training sequences (101 nucleotides each) and 2000 unlabeled test sequences.

2 Data Augmentation

To boost performance, I computed the complementary sequence for each DNA sample, effectively doubling the dataset size. Although this increases computation time and requires adjustments to the cross-validation function (to avoid contamination of the validation set with augmented data), it results in a performance boost of a few percentage points.

3 Kernels & Classifiers

3.1 Kernels on Vector Representations

Vectorized versions (matrices) of the DNA sequences are also provided. I initially experimented with RBF and polynomial kernels combined with Kernel Ridge Regression (KRR) or SVM. However, these representations proved suboptimal compared to kernels tailored to DNA sequences.

3.2 Kernels on String Sequences

Kernels designed for biological sequences can capture more information lost during vectorization. I focused on:

- **Spectrum Kernel** [1]: Counts occurrences $\Phi_u(x)$ of each k-mer u in a sequence x, with

$$K(x, y) = \sum_u \Phi_u(x)\Phi_u(y).$$

- **Mismatch Kernel** [2]: Similar to the spectrum kernel, but allows for m mismatches. Note that due to high computational cost, I was unable to compute mismatch kernels with $m > 2$ (insufficient RAM).

I also computed the Weighted Degree Kernel, but it did not improve performance enough to justify its own cross-validation. All kernels were pre-computed and stored as `.npy` files (totaling about 16GB).

3.3 Classifiers

I implemented several classifiers:

- **Kernel Ridge Regression (KRR):**

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

- **Kernel Logistic Regression (KLR):**

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ln\left(1 + e^{-y_i f(x_i)}\right) + C\|f\|_{\mathcal{H}}^2.$$

- **Kernel SVM:**

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

4 Simple Model & Meta Models

A grid search with cross-validation on Kernel SVM using a single kernel achieved a 0.722 score on the public test set. To further improve performance, I experimented with meta models (sum of kernels, bagging, and ensembles). Bagging with the best kernel per dataset degraded performance. Instead, I adopted a greedy strategy to build a sum of kernels: starting from the best kernel, I iteratively added the kernel combination that yielded the best improvement. This method provided the following combination per dataset:

Dataset	C	**Kernels**
1	1.19	MM(11,2), MM(20,1), MM(12,2), MM(5,1), MM(11,1), MM(17,1), MM(10,1), SP(11)
2	1.07	MM(10,1), MM(17,1), MM(5,2), MM(11,1), SP(13), MM(9,1)
3	0.98	MM(17,1), SP(12), SP(13), MM(14,2), MM(19,1), MM(15,2)

Table 1: Best kernel combinations selected by the greedy procedure.

The ensemble method with majority voting also provided a decent boost to the score (using the same greedy approach), but I decided not to use it, because the hyperparameter space was much too big to fine-tune (Each of the models needs its C parameter fine-tuned).

After fine-tuning the C parameter, this meta-model scored a final 0.736 on the public test set. Despite this improvement, the Sum of Kernel did not dramatically outperform the single-kernel model, possibly due to the similarity between the kernels. Notably, the Weighted Degree Kernel was never selected by the greedy procedure.

The meta-model's score was a bit lower on the private set (0.731), most likely due to overfitting since I was getting 99%+ on the validation set. It is interesting to note that the private score for my single kernel actually increased to 0.723, most likely because this model is less complex and less prone to overfit.

5 Conclusion

In summary, I developed several kernel-based models for DNA sequence classification. Although data augmentation and meta modeling provided modest gains, the best model (a sum of similar mismatch kernels) achieved a public test score of 0.737 and a private score of 0.731, ranking me first in both cases. Further improvements may be possible with more diverse kernel choices and deeper hyperparameter optimization.

References

[1] C. Leslie, E. Eskin, and W. S. Noble, *The spectrum kernel: A string kernel for SVM protein classification*, in *Pacific Symposium on Biocomputing*, 2002, pp. 564–575.

[2] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, *Mismatch string kernels for discriminative protein classification*, *Bioinformatics*, vol. 20, no. 4, pp. 467–476, 2004.