Kernel methods for machine learning - Project

Team name: *Team name*
Team members: *Eloïse Berthier, Guillaume Dalle, Clément Mantoux*

Algorithms

Classification: We implemented two base algorithms for binary classification: kernel SVM and kernel logistic regression. The estimation for kernel logistic regression was performed by IRLS, but its high computational cost deterred us from using this classifier. The estimation for kernel SVM was performed by solving a quadratic program. We initially tried to use the high-level modeling library CVXPY, but we discovered a significant performance increase when interacting directly with the underlying solver CVXOPT. Given that some of our Gram matrices were sparse, better optimizers (eg. OSQP) could have been chosen on an individual basis.

Parameter tuning: In order to improve the predictive power, we had to tune the settings of each kernel, as well as the ridge penalization λ. This was done using randomized 5-fold cross-validation. For each dataset, the best performing kernel with the optimal parameters (in terms of precision) was then retrained on the whole training set and used to predict the classes of the test set. A precise logging system was put in place to ensure reproducible results.

Multiple Kernel Learning: As an alternative to the comparison of multiple kernels, we tried to combine them by using the simpleMKL algorithm with projected gradient descent. We first faced a problem: the convex combination of kernels included only one kernel in the end, which made MKL irrelevant. Therefore we modified the optimization problem by adding an entropic regularization term to encourage diversity between the kernels. Tuning the regularization coefficient led to more balanced solutions and allowed MKL to outperform single-kernel approaches.

Boosting: We implemented an algorithm proposed by Crammer *et al.* (2003)[1]. The idea is, given a train and test set, to incrementally build a kernel using a base feature vector mapping. At each time step, a vector w_t is computed from a generalized eigenvalue problem involving the dataset and the previous kernel. The kernel is updated with the formula : $K_t(x, y) = \sum_{i=1}^{t} \alpha_i \langle x, w_i \rangle . \langle y, w_i \rangle = x^T \left(\sum_{i=1}^{t} \alpha_i w_i w_i^T \right) y$, where α_i is an adaptive learning rate. Boosting improved the performance on some feature vector kernels, but eventually did not produce the best score.

Kernels

Kernels on vectors: Our first initiative was to exploit the feature matrices provided by the challenge. We therefore defined various kernels for vector data (linear, polynomial, Gaussian, Cauchy), which were also applied to feature matrices of our own making.

Spectrum kernels: We computed spectrum kernels of various lengths, up to length 6, which performed best. We also tried the mismatch kernel, allowing a Hamming distance of one between each substring and dictionary entry. Yet it did not improve our results. To ensure a reasonable

[1] Crammer, K., Keshet, J., & Singer, Y. (2003). Kernel design using boosting. In Advances in neural information processing systems (pp. 553-560).

computation time, we used suffix trees in our implementation of substring comparisons. Besides, we tried to reweight the inner product given by the spectrum kernel by adding a TF-IDF factor before sequences, thus getting a new kernel. The resulting kernel was not better itself, but improved our results when combined with the spectrum kernel in the MKL.

Spectrum kernel corrected with protein translation: Then we tried to take advantage of prior knowledge on the way RNA sequences are translated into proteins, built from 21 different amino acids (including the STOP codon). We considered as a unique 6-mer each string of length 6 that could be translated into a unique pair of two proteins. Since the genetic code is redundant, this equivalence relation reduced the number of possible 6-mers from 4096 to 441. Yet we believe that attempting translation on short sequences extracted at random, without knowing the reading frame, nor possibly the direction 5'→3', was a bit too ambitious for this problem. Besides, it is unclear why the translated sequence would be relevant to the particular task of spotting TF binding sites, which happens upstream of the translation process.

Sequence+shape kernel: We followed an approach by Ma *et al.* (2017)[2] that specifically designs a kernel for TF binding sites identification as a sum of a spectrum and a shape kernel. There are four simple DNA shape features (MGW, Roll, ProT, HelT) that describe local structural features of the DNA and each one is defined over a pentamer of DNA. The shape kernel consists in extracting the four shape features at several positions in a small neighborhood of each k-mer of the sequence. The original article mentioned that the values of the shape features for each pentamer were derived from Monte Carlo simulations. We computed average values of the features using an extracted sequence of shape features at 100k bases from chromosome 6 of the reference human genome GRCh37 found on the GBshape database, aligned with the corresponding DNA sequence found on the NCBI database. This is certainly unsatisfactory, but we could not easily extract more shape features from GBshape. The sequence+shape kernel of length k produces an embedding of dimension $(4k+1)4^k$, which quickly makes its storage intractable, encouraging online kernel evaluations.

Substring and local alignment kernels: We tried to go beyond the spectrum kernel and compute more sophisticated similarity measures between strings. The substring kernel was coded in Python and heavily optimized using Numba, while the local alignment kernel was even coded in Julia to gain efficiency. Nonetheless, both required several hours of computation for one set of parameters, which made them impossible to tune properly.

Marginalized kernels: We made a brief foray into generative models by implementing the first- and second-order marginalized count kernels. They are based on a Hidden Markov Model for the DNA sequence, which represents the succession of exons (with 3 sub-states corresponding to codons) and introns along a gene. Because we needed to estimate the discrete HMM parameters based on a large set of quite long strings, we re-coded the Baum-Welch algorithm from scratch, granting particular importance to the efficiency of the procedure and its numerical stability.

Best kernel selection

The same kernel worked best for the three datasets with good cross-validation scores. We selected the MKL combination of Gaussian kernels applied to the feature vectors produced by the sequence+shape kernel with length 4, the translation-corrected spectrum kernel with length 6 and the TF-IDF-reweighted spectrum kernel with length 6. The combination weights depend on the dataset, but all three distributions are around $(0.45, 0.1, 0.45)$. The entropic regularization factor was set to 1.2, and the SVM regularization parameter was always chosen very small (around 10^{-4}).

[2]Ma, W., Yang, L., Rohs, R., & Noble, W. S. (2017). DNA sequence+shape kernel enables alignment-free modeling of transcription factor binding. *Bioinformatics*, 33(19), 3003-3010.