Kernel methods for Machine Learning Kaggle Challenge

Team Name: *ML like it's 2005*

Ulrich GOUE
ulrich.goue@ens-paris-saclay.fr

Théo Comperot
theo.comperot@ensae.fr

Gabriel ROMON
gabriel.romon@ensae.fr

April 1, 2019

Goal & Overview

The purpose of this challenge was to implement and gain practical experience with the kernel methods seen during the class. We were challenged to predict whether a DNA sequence is a binding site to a specific transcription factor. Beside reaching the highest possible score, we explored classical kernel methods[1] along with the kernels for biological sequences described in class [3] (slides 322-361). We also tried to take into account some research topics encountered in class, namely how to combine several kernels([2] among others). In our approach we simultaneously learned the optimal parameters of SVM and the optimal polynomial combination of our available kernels like in [1].

1 Kernels

1.1 Spectrum Kernels

The classical spectrum kernel is based on $\mathcal{A}_k$ which is the set of all possible k-mers. But we propose an extension by considering $\mathcal{A}_k^{[n]}$ the set of all possible n-grams drawn from set $\mathcal{A}_k$. Therefore we embed a sequence $\mathbf{s}$ by $\psi_k^{[n]}(\mathbf{s}) = \left(\psi_{k,u}^{[n]}(\mathbf{s})\right)_{u\in\mathcal{A}_k^{[n]}}$ where $\psi_{k,u}^{[n]}(\mathbf{s})$ is the number of occurrences of u in $\mathbf{s}$. So our spectrum kernel $K_k^{[n]}$ is defined by $K_k^{[n]}(\mathbf{s},\mathbf{s}') = \langle\psi_k^{[n]}(\mathbf{s}), \psi_k^{[n]}(\mathbf{s}')\rangle$. Therefore the classical spectrum kernels are the $K_k^{[1]}$.

1.2 Mismatch Kernels

The mismatch kernel is a variant of the spectrum kernel where it can be allowed to have one or more mismatch in the compared sub-sequences. Formally, we rather use the following embedding: $\tilde{\psi}_k^{[m]}(\mathbf{s}) = \left(\tilde{\psi}_{u,k}^{[m]}(\mathbf{s})\right)_{u\in\mathcal{A}_k}$ where $\tilde{\psi}_{u,k}^{[m]}(\mathbf{s})$ is the number of occurrences of u in $\mathbf{s}$ up to m mismatches. We denote these kernels by $\tilde{K}_{k,m}$ hereafter.

1.3 *Exponentially Smoothed* Spectrum Kernels

These so-called *exponentially smoothed* spectrum kernels are based on the classical spectrum kernels $K_k^{[1]}$. Now suppose we are given sequences $\mathbf{s}_1, ..., \mathbf{s}_w$ sequences, respectively embedded by $\psi_k^{[n]}(\mathbf{s}_1), ..., \psi_k^{[n]}(\mathbf{s}_w)$, and denote by $\mathbf{X}_k = \left(\psi_k^{[n]}(\mathbf{s}_1)', ..., \psi_k^{[n]}(\mathbf{s}_w)'\right)'$. It is then clear that $K_k^{[1]} = \mathbf{X}_k\mathbf{X}_k^T$. However we can weight the elements of $K_k^{[1]}$ element-wise to account for the overall distance between the sequence now considered like words. That is, we can consider the family of kernels $\bar{K}_{\lambda,k} = \mathbf{X}_k S_\lambda \mathbf{X}_k^T$, with $S_{\lambda,ij} = \lambda^{d_{\text{Lev}}(\mathbf{s}_i,\mathbf{s}_j)}$, $0 \le \lambda < 1$, where d_{Lev} is the Levenshtein distance. Now if we use the convention $0^0 = 1$, then $\bar{K}_{0,k} = K_k^{[1]}$.

2 Classifiers

2.1 Single-Kernel-based classifiers

Now we have three types of kernels. Initially we built models with only one kernel, say K, which are **KLR**, **SVM** and a **distance-based classifier**. For the last one, imagine that we are given sets of sequences $\mathbf{S}_0$ and $\mathbf{S}_1$ with labels respectively equal to 0 and 1. For a given sequence $\mathbf{s}$ we compute $d_K(\mathbf{s}, \mathbf{S}_i)$ for $i = 0, 1$, and predict $\arg\min_{i=0,1} d_K(\mathbf{s}, \mathbf{S}_i)$.Here the definition $d_K(\mathbf{S})$ is the same as in slide 66.

2.2 Non linear kernel combinations

Unlike [2], we are not only interested in convex combination combinations of kernels, but also in non-linear

[1] like Kernel logistic regression (KLR), Kernel Ridge Regression (KRR), Support Vector Machine (SVM)

polynomial combination which is more general. That said, if we are given kernels $K_1, K_2, ..., K_p$, a vector $\mu \in \mathbb{R}^p$, and a degree d we want to learn some optimal combination $K_\mu = (\sum_{i=1}^{p} \mu_i K_i)^p$. Thus we are interested in solving problems:

$$\min_{\mu \in \mathcal{M}} \max_{\alpha \in \Gamma} 2\alpha^T y - \alpha^T (K_\mu + \lambda I)\alpha \tag{1}$$

where we chose $\mathcal{M} = \{\mu \mid \mu \succeq 0 \quad \wedge \mid\mid \mu \mid\mid_2 \leq 1\}$ for simplicity's sake. We see that (1) fits **SVM** if we proceed with its dual form. To solve it numerically we use the *projection-based gradient descent algorithm* proposed in [1].
We wrote two main python files: (i): `kernels.py` to build the kernels, (ii) `classifiers.py` to build the classifiers which has two main classes. The major class `MultiKerOpt` implements the projection-based gradient descent algorithm. But it also implemnts SVM, KLR, and KRR as methods. For SVM we use the buit-in optimization python API `cvxopt`.

3 Experiments

3.1 Kernel choices

We first built our three types of kernels.

- **Spectrum kernels**: For the spectrum kernel we just focus on 4-grams of k-mers. Initially we tried n-grams for $n \leq 8$, and we realized that the best results were achieved with $n = 4$ for every classifier (SVM and KLR). Thus we restrict ourselves to the 5 kernels $K_k^{[4]}$ for $3 \leq k \leq 7$

- **Exponentially Smoothed Spectrum Kernels**: Afterwards we build the 45 kernels $\bar{K}_{\lambda,k}$ for $3 \leq k \leq 7$ and $\lambda \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.8, 0.9\}$.

- **Mismatch Kernels**: At last we only build 3 mismatch kernels $\tilde{K}_{3,1}, \tilde{K}_{4,1}, \tilde{K}_{5,1}$. Unsurprisingly $\tilde{K}_{k,m}$ was so poor that we dropped it in the upcoming experiments.

3.2 kernels combinations

Now three hyperparameters remain to be optimized: the SVM regularization parameter, the degree[2] of kernel combinations and λ. To do so, for each one of the three datasets, we consider the following collections of 12 kernels:

$$\begin{aligned}\mathcal{T}_\lambda &= \left\{K_k^{[4]} \mid 3 \leq k \leq 7\right\} \cup \left\{\bar{K}_{\lambda,k} \mid 3 \leq k \leq 7\right\} \\ &\quad \cup \left\{\tilde{K}_{k,1} \mid 4 \leq k \leq 5\right\}\end{aligned}$$

We estimate a SVM while combining kernels $\mathcal{T}_\lambda$ and keeping the optimal regularization parameter C_λ, best degree d_λ and best accuracy g_λ using a 5-fold cross validation. The candidates degree were 1,2,3 while the proposed regularization parameters where 0.001, 0.01, 0.1, 1, and 10. We chose $\hat{\lambda} = \arg\max_\lambda g_\lambda = 0.1$. Therefore combining the kernels of $\mathcal{T}_{0.1}$, we were able to score **0.705** on the private Kaggle dataset and be ranked **8/76**.

4 Conclusion

During this challenge we tried to the best we can to use the methods presented during this class. However we were left a little bit frustrated since we felt that there was a lot of place for improvement. We encountered many other kernels during our research that we were reluctant to use since it took too much time to be computed. Considering such kernels would have been likely to improve our work. On the other hand we explored some statistical recipes that didn't work very well as suggested. It was a guideline to choose the most informative k-mers before the computation of kernels based on Kullback-Leibler distance.

References

[1] C. Cortes, M. Mohri and A. Rostamizadeh. *Learning Non-Linear Combination of kernels*, NYU Computer Science.

[2] A. Rakotomamonjy, F. Bach, S. Canu and Y. Grandvalet. *SimpleMKL*, Journal of Machine Learning Research 9 (2008) 2491-2521.

[3] Julien Mairal and Jean-Philippe Vert, *Machine Learning with Kernel Methods*, MVA slides , 2019.

[2] Recall that the kernel combining optimizer only (theoretically) works with SVM