Kernel Methods Challenge

Predicting Whether a DNA Sequence Region is Binding Site to a Specific Transcription Factor

Clément Grisi
Team Name: clems
grisi.clement@gmail.com

Abstract

In this report, I emphasize my work for the challenge organized as part of the Kernel Methods class. The goal of the challenge was to learn how to implement kernel-based machine learning algorithms, gain understanding about them and adapt them to structural data. To that end, teachers chose a DNA sequence classification task. My code is publicly available at https://github.com/clementgr/kernel-methods

1. Introduction

Transcription factors (TFs) are regulatory proteins that bind specific sequence motifs in the genome to activate or repress transcription of target genes. Genome-wide protein-DNA binding maps can be profiled using some experimental techniques and thus all genomics can be classified into two classes for a TF of interest: bound or unbound. The challenge consists in predicting whether a DNA sequence region is binding site to a specific TF.

Dataset: we had to work with three datasets corresponding to three different transcription factors. Each training set contained 2000 sequences, while each testing set contained 1000. Predictions had to be made separately for each train-test dataset pair.

Evaluation Metric: the performance is measured using the classification accuracy.

2. Classifiers

I implemented different classifiers for the sequence classification task. Each classifier aims at solving the following optimization problem, each with a different loss function ℓ:

$$\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \ell\left(f(x_i), y_i\right) + \lambda ||f||_{\mathcal{H}}^2 \qquad (\star)$$

2.1. Logistic Regression

Logistic regression is an algorithm often used in machine learning problems, which models the probability that a given input belongs to one of two different classes (binary classification). It can be viewed as solving $(\star)$ with the logistic loss $\ell\left(f(x), y\right) = \ln\left(1+e^{-yf(x)}\right)$ and the functionnal space:

$$\mathcal{H} = \{w \in \mathbb{R}^n, \beta \in \mathbb{R} \mid f(x) = \sigma(w^\top x + \beta)\}$$

Equation $(\star)$ is then a smooth convex optimization problem, efficiently solved by gradient descent.

2.2. Support Vector Machine

Support Vector Machine (SVM) is another supervised learning model, which can be viewed as solving $(\star)$ with the hinge loss: $\ell\left(f(x), y\right) = \max\left(1 - yf(x), 0\right)$. This time, the objective function of $(\star)$ is not smooth. Hence, it's common practice to introduce slack variables to reformulate $(\star)$ as a quadratic program, that is the minimization of a convex quadratic function with linear constraints. Quadratic programs are efficiently solved using any optimization package. I used `convexopt`.

2.3. Kernel Ridge Regression

If we consider $\mathcal{H}$ a RKHS associated to a p.d. kernel $\mathbf{K}$ on $\mathcal{X}$, then Kernel Ridge Regression (KRR) is obtained by regularizing the MSE criterion by the RKHS norm. By the representer theorem, KRR is equivalent to:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{n}(\mathbf{K}\boldsymbol{\alpha} - \mathbf{y})^\top(\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}) + \lambda\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}$$

which is a convex and differentiable problem in $\boldsymbol{\alpha}$ and admits the closed form solution:

$$\boldsymbol{\alpha}^* = \left(\mathbf{K} + \lambda n\mathbf{I}\right)^{-1}\mathbf{y}$$

2.4. Kernel SVM

Here again, considering $\mathcal{H}$ a RKHS associated to a p.d. kernel $\mathbf{K}$ on $\mathcal{X}$, then Kernel SVM algorithm is obtained by rewriting SVM's quadratic program using the representer theorem. This gives another QP, this time in $\boldsymbol{\alpha}$:

$$\begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad & 2\boldsymbol{\alpha}^\top \mathbf{y} - \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \preceq \boldsymbol{\alpha}^\top \mathbf{y} \preceq \frac{\mathbb{1}}{2\lambda n} \end{aligned} \tag{kSVM}$$

I used `cvxpy` to solve this problem.

3. Working with Numeric Data

3.1. Logistic Regression Baseline

The idea was to have a reference result against which I could compare the performances of the more sophisticated models I would later develop. I decided to go with a simple baseline: train a logistic regression on the numerical feature matrices provided by the teachers. This achieved 0.60466 on the public leaderboard, and 0.58666 on the private leaderboard.

3.2. Switching to Support Vector Machines

I then shifted to Support Vector Machines. Prior to fitting this model, I re-labeled inputs from $\{0, 1\}$ to $\{-1, 1\}$. I tried both plain SVM and C-SVM and didn't observe any signficant difference. In the end, C-SVM with C=10 performed just as well as the logistic regression, reaching 0.60333 on the public leaderboard, and 0.58666 on the private leaderboard.

3.3. Using the Radial-basis Function Kernel

Enough working with raw numerical feature matrices, time to use some kernels! I moved from linear to nonlinear classifiers using the Radial-basis Function Kernel (RBF), which maps data points to Gaussian functions living in a Hilbert space $\mathcal{H}$. The RBF kernel with bandwidth σ is given by:

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}$$

Once data points were mapped to $\mathcal{H}$, I solved the classification task using:

- Kernel Ridge Regression: best results were achieved for $\sigma = 100$ and regularization parameter $\lambda = 10^{-7}$. This translated into 0.60000 on the public academic, and 0.59733 on the private leaderboard,
- Kernel SVM: best results were achieved for $\sigma = 100$ and $\lambda = 10^{-8}$. This translated into 0.61000 on the public academic, and 0.59933 on the private leaderboard, slightly outperforming previous results.

Optimal hyperparameters σ and λ were found by finetuning on 30% of the training data, set aside as validation set.

4. Working with Raw DNA Sequences

Once I had finished experimenting with numerical data, I started working with the raw DNA sequences. The goal was to design kernels suited for strings. I decided to go with two kernels based on substring indexation.

4.1. Spectrum Kernel

The spectrum kernel [1] is a sequence-similarity kernel. Given an integer $k \geq 1$, the k-spectrum of an input sequence is the set of all the k-length subsequences that it contains. If we denote by $\phi_a(k)$ the number of times the subsequence a occurs in sequence x, and $\mathcal{A}^k$ all possible subsequences of length k from alphabet $\mathcal{A}$, the k-spectrum kernel is given by:

$$\mathbf{K}_k(x, y) = \langle \Phi_k(x), \Phi_k(y) \rangle$$

where $\Phi_k(x) = (\phi_a(k))_{a\in\mathcal{A}^k}$. This kernel has the advantage of being relatively simple to implememt and fast to compute. I used Kernel SVM and fine-tuned the parameters k and λ using the 30% validation set (Figure 1 and 2). The best results were achieved with $k = 6$, giving a public score of 0.65400 and a private score of 0.64066.

4.2. Mismatch Kernel

The mismatch kernel [2] is a variant of the spectrum kernel which allows subsequences of a given length to have one – or more – mismatch in the compared sub-sequences. When applied to DNA, it's comes down to accounting for small mutations to persists between sequences, hence providing a biologically motivated way to compare protein sequences. Once again, I used Kernel SVM and fine-tuned the parameters k, m (number of mismatch allowed) and λ using the 30% validation set (Figure 3 and 4). The best results were achieved with $k_0 = 8$, $k_1 = k_2 = 9$ and $m = 1$, giving a public score of 0.65200 and a private score of 0.64733, the best score I managed to get (in average).

5. Conclusion

Through iterations in model definition and data representation, I highlight the pros and cons of the different methods I tried. In the end, I got my best score using the mismatch kernel: 0.65200 classification accuracy on the public leaderboard and 0.64733 on the private leaderboard.

References

[1] Christina Leslie, Eleazar Eskin, and William Noble. The spectrum kernel: A string kernel for svm protein classification. *Pacific Symposium on Biocomputing*, 7:564–75, 02 2002. 2

[2] 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. 2

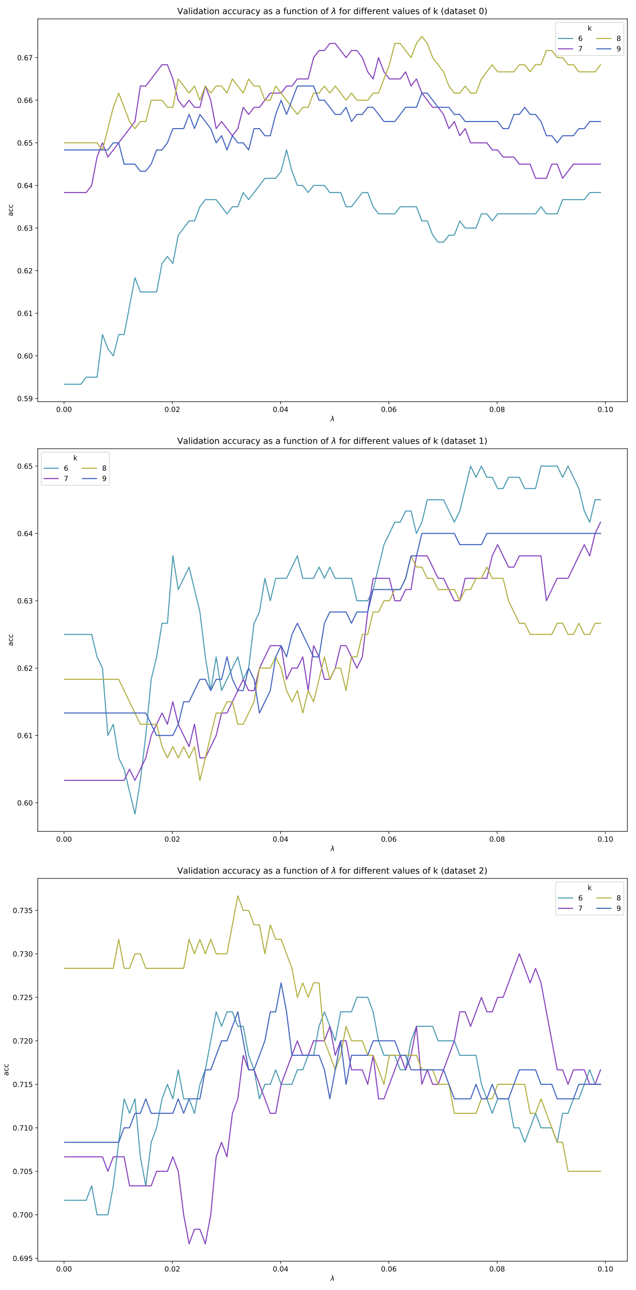

Figure 1: Validation Accuracy for Different k-Spectrum Kernels as a Function of Regularization Parameter λ

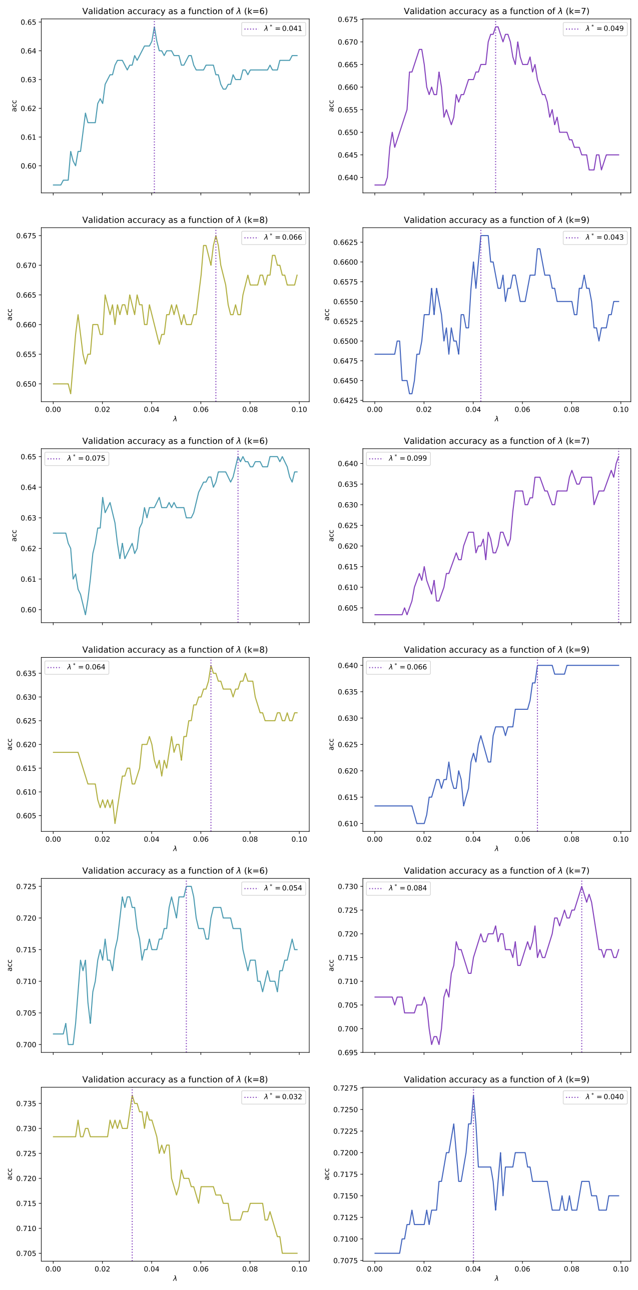

Figure 2: Optimal Regularization Parameter λ for Different k-Spectrum Kernels

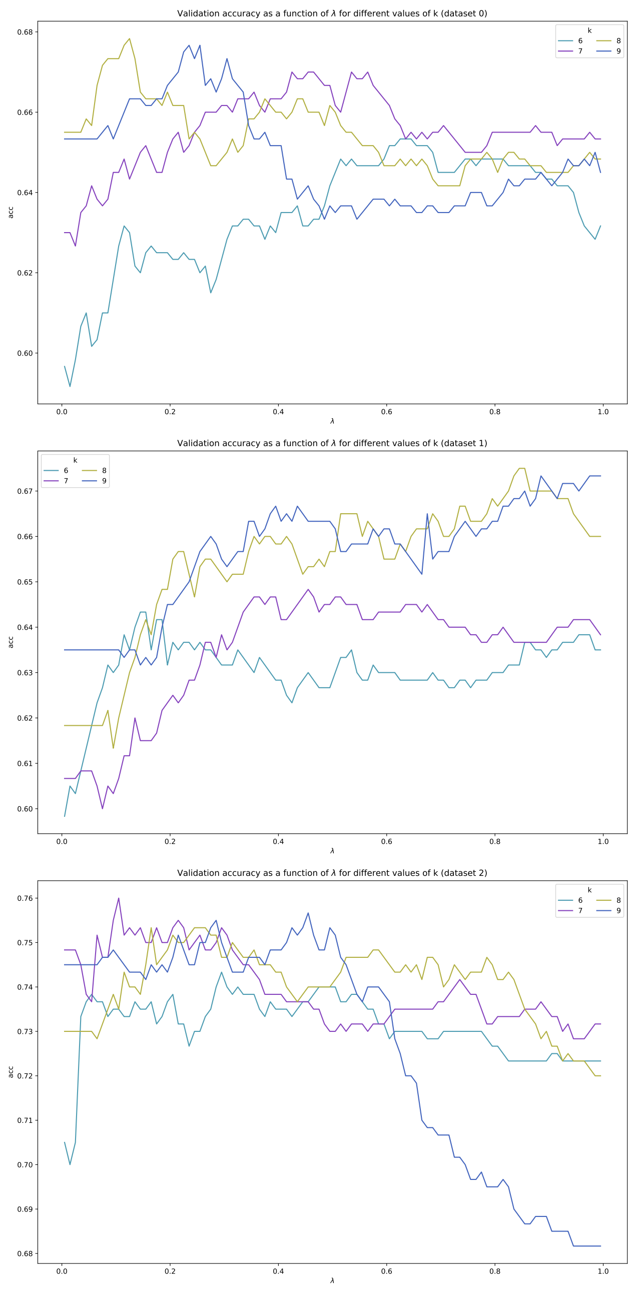

Figure 3: Validation Accuracy for Different Mismatch Kernels as a Function of Regularization Parameter λ

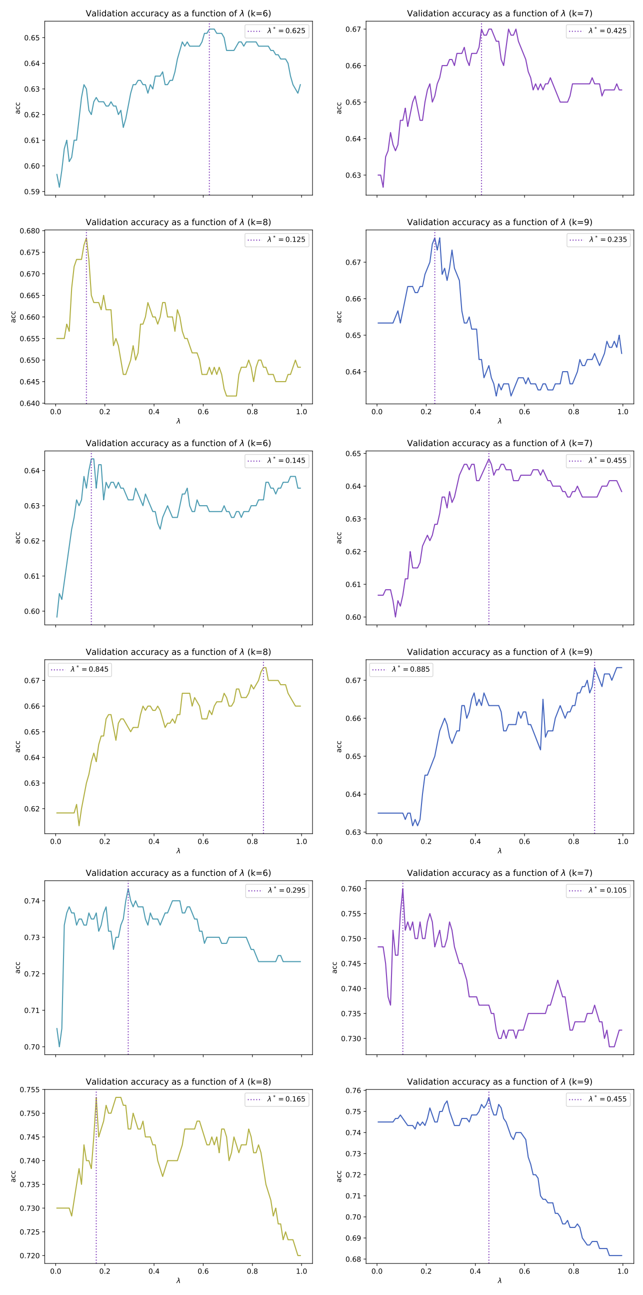

Figure 4: Optimal Regularization Parameter λ for Different Mismatch Kernels