Kernel Methods for Machine Learning

Mohammed Chlieh (MVA) Romain Vial (MVA)

1. Introduction

The goal of this project is to predict whether a DNA sequence region is binding site to a specific transcription factor. Transcription factors (TFs) are regulatory proteins that bind specific sequence motifs in the genome to activate or repress transcription of target genes. Genomes can be classified into two classes for a TF of interest: bound or unbound. We will work with three datasets corresponding to three different TFs.

2. Proposed Approach

Our approach to this sequence classification task uses Support Vector Machine as presented by [1]. Let define $x = [x^{(1)} \dots x^{(p)}]$ where $x^{(i)} \in \mathcal{V} = \{A, T, C, G\}$ such DNA sequence. We call $\mathcal{X} = \mathcal{V}^p$ the space of DNA sequences of length p. Our first objective is to find a meaningful representation in a Hilbert space $\mathcal{F}$ of such sequence through a mapping $\phi : \mathcal{X} \to \mathcal{F}$. Then, we will use a positive definite kernel K such that $K(x, y) =< \phi(x), \phi(y) >_{\mathcal{F}}$, where $x, y \in \mathcal{X}$.

2.1. Features

Two classes of features have been explored: (i) spectrum and (ii) mismatch features. These two features are computationally efficient and well suited to tackle biological sequences according to [5].

2.1.1 Spectrum Features

As proposed in [3], we define the k-spectrum of a sequence $x \in \mathcal{X}$ as the vector $\phi(x) = \{\phi_u(x)\}_{u \in \mathcal{V}^k}$, where $\phi_u(x)$ denotes the number of occurrences of u in x. This can be seen as a bag-of-word representation where the "words" are all the possible subsequences of size k with the vocabulary $\mathcal{V}$.

2.1.2 Mismatch Features

Mismatch features are an extension of the spectrum features proposed by [2]. They are obtained by computing the vector $\phi(x) = \{\phi_{u,m}(x)\}_{u \in \mathcal{V}^k}$ where $\phi_{u,m}(x)$ is the number of occurrences of u in x up to m mismatches.

2.2. Kernels

Two types of widely used kernels are then presented: (i) linear and (ii) RBF kernels.

2.2.1 Linear Kernel

We define K as a simple inner product between the two feature maps:

$$K(x, y) = \phi(x)^\intercal \phi(y)$$

2.2.2 RBF Kernel

We also explored the Radial Basis Function (RBF) Kernel which is popular and commonly used:

$$K(x, y) = \exp(-\gamma ||\phi(x) - \phi(y)||^2)$$

3. Implementation

Once we extracted the raw string data $X = (x_i)_{i \in \{1,..n\}}$, where n is the number of examples, we implemented our kernel method in 2 main parts:

- A module to compute the Gram matrix $\mathbf{K} = (K(x_i, x_j))_{i,j \in \{1,..n\}}$ using either one of the two features: spectrum or mismatch features.
- A module to run a Support Vector Machine algorithm SVM given the Gram matrix as input.

3.1. Computing the Gram matrix

This is done is the python module `kernels.py`. This operation is composed of 3 main steps:

- Extracting either one of the spectrum features or the mismatch features from the raw string data.
- Computing either one of the Linear kernel or the RBF kernel using the obtained features in the previous step.
- Normalize the kernel Gram matrix obtained from above. Indeed we wanted to normalize the points to unit norm in the feature space, but we also separated those points from the origin by adding a constant as

done in [4]. This is done through the following operation:

$$K^{norm}(x,y) = \frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}} + 1$$

Indeed $K(x,x) = ||\phi(x)||^2$ where $\phi(x)$ is the mapping of x in the feature space .

3.2. SVM

The SVM algorithm is implemented in the python module `svm.py`. Implementing the SVM algorithm boils down to solving a Quadratic Program, either by solving its primal form, or its dual form.

- The primal form of the SVM can be written as follows:

$$\min_{\alpha,\xi} \frac{1}{n}\sum_i \xi_i + \lambda \alpha^T K \alpha$$

$$s.t \quad y_i[K\alpha]_i + \xi_i - 1 \geq 0 \quad \forall i \in \{1,...n\}$$

$$\xi_i \geq 0$$

- the dual form can be written as follows:

$$\max_{\alpha} 2\alpha^T y - \alpha^T K \alpha$$

$$s.t \quad 0 \leq y_i \alpha_i \leq \frac{1}{2\lambda n} \quad \forall i \in \{1,...n\}$$

We can rapidly see that solving the dual form is a more efficient strategy since we solve a problem of finding n variables instead of $2n$ variables for the primal, among other things. Many optimization libraries in python solve those QP problems, we used `cvxopt`.

4. Experiments

We ran our experiments in a cross-validation fashion on 5 folds on the training dataset. We find thanks to this approach the best parameters in terms of spectrum size (number k in the k-spectrum extracted from the letter sequence x), regularization parameter λ (we actually used the parameter $C = \frac{1}{2\lambda n}$) and the parameter γ in the RBF kernel expression. The best parameters can be found in the python main module `start.py`.

We achieved a result of 78.60% on the public test set and 77.60% on the private one using the spectrum features with the RBF kernel. We can actually see from Table 1 that the linear and RBF kernel are performing similarly, but the spectrum features outperformed the mismatch features. As the results between the private and public test are close, one can see that we didn't overfit the public test, the validation or the training set.

Features	Kernel	Split 1	Split 2	Split 3
Spectrum	Linear	74.95	87.70	63.55
Spectrum	RBF	74.75	87.55	64.50
Mismatch	Linear	73.05	86.00	63.65
Mismatch	RBF	72.90	84.15	61.65

Table 1. Mean accuracy after 5-fold cross-validation on the three training splits with different features and kernels.

5. Conclusion

Thanks to a kernel method, we succeeded in finding a better representation of the data in a feature space where we were able to classify the data with a simple linear ridge regression.

References

[1] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3), Sep 1995.

[2] E. Eskin, J. Weston, W. S. Noble, and C. S. Leslie. Mismatch string kernels for svm protein classification. In *Advances in neural information processing systems*, pages 1441–1448, 2003.

[3] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001.

[4] H. Saigo, J.-P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

[5] S. Sonnenburg, G. Rätsch, and K. Rieck. *Large scale learning with string kernels*. MIT Press, 2007.