Kernel methods for protein classification

Jérémie Dentan
jeremie.dentan@polytechnique.org
École Polytechnique

ABSTRACT

Protein classification is one of the main tasks in bio-informatics and computational biology, especially in drug discovery. The structure of proteins, as 3D folded sequence of amino acids, determines their specific chemical functionalities. The aim of this project is to apply kernel methods on the graph structure of a number of proteins to classify them into 2 classes.

Our approach has been to implement five different kernel on the protein graphs, and four different kernel classifiers, and to try all the combinations between those kernels and those classifiers, using several hyperparameters for each of them.

For the kernels, we implemented: (1) a kernel based on the histograms on the edge labels, (2) a kernel based on the histograms on the vertex labels, (3) a kernel based on the concatenation of those two histograms, (4) the Pyramid Match kernel [3] with 8 different hyperparameters for the tuple dimension-level, and (5) the Shortest Path kernel [2].

For the classifiers, we implemented: (1) an unweighted K-Nearest-Neighbor (KNN) kernel, (2) a weighted KNN with a linear weight depending on the rank of the neighbor, (3) a weighted KNN with hyperbolic weight depending on the distance of the neighbor, and (4) a Kernel Support Vector Machine (K-SVM). For the three KNN, we used 58 different hyperparameters, varying the number of neighbors from 3 to 60, and for the K-SVM, we used 11 different hyperparameter for the tuple of the weight on the constraint and the number of training point.

In the end, we tested 12 different kernels and 185 different classifiers, for a total of 2220 combianison tested. This thorough study allowed us to reach satisfactory performances considering the difficulty of the challenge: we obtained an AUC score of 81% and a ranking of 14 out of 22.

1 INTRODUCTION

The code of all our implementation is available in this repository.

1.1 The challenge

The challenge consists in a binary classification task on proteins 3D configuration. There are 6000 training proteins, and 2000 proteins for which we have to do the prediction. The configuration of the proteins is given in the form of a graph representing the interractions between the amino acids. The labels of the nodes represent features on the chemical bonds between these elements, while the labels between the edges represent features on the amino acids themselves.

The challenge evaluation metric is the AUC score associated with the logits predicted on the test set. Moreover, one of the rule of the challenge was to use kernel methods only for this classification task.

A first difficulty related to these datasets is that the organizers of the challenge did not disclose the biological meaning represented by these different features or by the two classes to predict. This was done to make the task more complicated, however this omission prevents us from performing the most important part of any machine learning problem, namely the fine understanding of the data at stake via exchanges with domain experts.

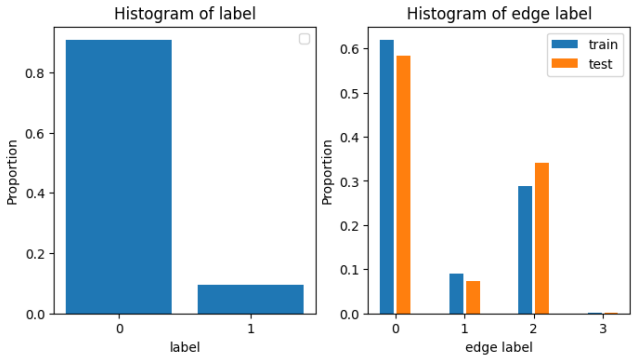

Figure 1: Left: histogram of the labels in the train set. Right: Histogram of the edge labels.

1.2 Data exploration

First, we briefly explored the data at hand. We can make several observations:

- The data are highly imbalanced, with more than 90% of the proteins belonging to the same class (cf. figure 1, left) This is a classical problem when dealing with proteins classification [4]. Given the algorithms we deployed, which are not very amenable to this problem, the only thing we did for this was to create a subset whose classes are balanced, and which we used as a training set for K-SVM.
- The distribution of the edges labels is similar between the train and test set (cf. figure 1, right). Thus, we do not have to deal with a distribution shift.

1.3 Related work

Machine Learning (ML) techniques for protein classification have received a lot of interest in the past years, mainly due to its huge industrial applications. Indeed, conducting experiments to study the properties of an unknown protein is an extremely time consuming and costly task. Thus, ML models can interestingly filter proteins in order to conduct biological experiments on promising proteins only [5].

Some surveys have been conducted on protein classification, such as [4], but they do not specifically focus on kernel methods. However, we can mention that kernel methods were already used on proteins, both on their sequences of amino-acid or on their spacial configuration [1, 7].

Finally, there exist some interesting and comprehensive survey on graph kernels, such as [6]. We used this survey to choose which kernel to implement.

2 OUR ALGORITHMS

In this section, we present the kernels we computed on the proteins, as well as the classifier we used with those kernels.

2.1 Our kernels

Histogram-based kernels. The first kernels we implemented are really simple, based on the histograms of node labels and edge labels. For example, for the node labels, there are 50 possible label for each node, so each graph G is mapped to a point $\varphi G \in \mathbb{R}^{50}$ corresponding to the histogram of each label in this graph. For this kernel, $\mathbb{R}^{50}$ is the RKHS of the kernel, and its natural dot product defines the kernel evaluation between two graphs. We did the same for edge histogram, the concatenation of those two histograms.

$$K(G_1, G_2) =< \varphi(G_1), \varphi(G_2) >$$

Pyramid Match kernel. Then, due to its very good performances announced in [6], we decided to implement the Pyramid-Match kernel [3]. This kernel first embeds each node of each graph in a d-dimension space using the eigenvalues of the adjacency matrix. Then, the kernel evaluation between two graph is computed using the histogram that count how many of the nodes in this d-dimension space fall into the same cell. The size of these cells involves an additional parameter, called the level and noted l.

We tested 8 combination of parameter, making d varying between 3 and 6, and l varying between 3 and 4.

Shortest Path kernel. Similarly, due to its good performances announced in [6], we implemented the Shortest Path kernel [2]. This kernel first computes the shortest path between all points in all graphs, and then the kernel evaluation between two graphs is computed using the histogram of shortest path length classified according to the labels of the nodes at the ends of the shortest paths.

For this kernel, we can vary how the histograms are compared, but we decided not to introduce any hyperparameter here.

2.2 Our classifiers

Kernel-KNN. First, we used three type of KNN to predict the logit of the classification on the test set:

- A simple unweighted KNN, for wich the final logit is the weighted average of the class of the neighbors
- A weighted KNN with each neighbor having weight $1-k/N$ where k is the rank of the neighbor and N the number of neighbor used.
- A weighted KNN with each neighbor having weight $1/d_k$ where d_k is the distance between neighbor k and the querry point.

For each of those classifier, we made the number of neighbor vary from 3 to 60, resulting in a total of 174 classifier tested.

Kernel-SVM. Then, we used a regular kernel SVM classifier, solving the following minimization problem:

$$\min_{f,b,\xi_i} \frac{1}{2}||f||^2 + c\sum \xi_i$$
$$\text{s. t. } y_i(f(x_i)+b) \geq 1-\xi_i \ ; \ \xi_i \geq 0$$

Where x_i are the embedded values in the RKHS (not computed), and y_i their labels. To reduce computation time, we restrained the number of points in the train set, and used a balanced subset between the two classes. Thus, there are two hyperparameters for this method: the value of c, and the number of training points. We used 11 different configuration for those hyperparameters.

2.3 Choosing the best configuration

As said above, we tested all combination of our 12 kernels and 185 classifiers, leading to 2220 possibilities. Thus, we needed a way to choose which one was the best and submit it for the challenge. To do so, we randomly extracted 1000 graphs out of the 6000, and used it as a validation set that was never used during training.

2.4 Hardware and computation time

For our experiments, we used a Intel Xeon W-1290P 3.70GHz, using a single process. For both the Pyramid Match and Shortest Path kernel, the computation of the Gram matrix for our 8000 points took about 20min per kernel. In each case, the preprocessing steps (computing shortest paths, or eigenvalue decomposition) were really fast, and the slower was then the pairwise comparison of the histograms. On the contrary, the computation of the simple histogram kernels was really fast, about few seconds.

Then, for the classifier, all classifier were really fast (bottleneck for the disk writing of the output), except the K-SVM, which took about 20sec per fitting.

Finally, all our experiments took about 2h45 of computation.

3 CONCLUSION

The approach described above enable us to reach a final AUC score of 81% and a ranking of 14 out of 22, which is quite satisfying with respect to the difficulty of the challenge.

REFERENCES

[1] Asa Ben-Hur and William Stafford Noble. 2005. Kernel methods for predicting protein-protein interactions. *Bioinformatics (Oxford, England)* 21 Suppl 1 (June 2005), i38–46. https://doi.org/10.1093/bioinformatics/bti1016

[2] K.M. Borgwardt and H.P. Kriegel. 2005. Shortest-path kernels on graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*. 8 pp.–. https://doi.org/10.1109/ICDM.2005.132 ISSN: 2374-8486.

[3] Kristen Grauman and Trevor Darrell. 2007. The Pyramid Match Kernel: Efficient Learning with Sets of Features. *Journal of Machine Learning Research* 8 (April 2007), 725–760.

[4] Chhote Lal Prasad Gupta, Anand Bihari, and Sudhakar Tripathi. 2019. Protein Classification using Machine Learning and Statistical Techniques: A Comparative Analysis. https://doi.org/10.48550/arXiv.1901.06152 arXiv:1901.06152 [cs, q-bio, stat].

[5] Armaghan W Naik, Joshua D Kangas, Devin P Sullivan, and Robert F Murphy. 2016. Active machine learning-driven experimentation to determine compound effects on protein patterns. *eLife* 5 (Feb. 2016), e10047. https://doi.org/10.7554/eLife.10047 Publisher: eLife Sciences Publications, Ltd.

[6] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. 2021. Graph Kernels: A Survey. *Journal of Artificial Intelligence Research* 72 (Nov. 2021), 943–1027. https://doi.org/10.1613/jair.1.13225 arXiv:1904.12218 [cs, stat].

[7] Jean-Philippe Vert. 2006. Classification of Biological Sequences with Kernel Methods. In *Grammatical Inference: Algorithms and Applications (Lecture Notes in Computer Science)*, Yasubumi Sakakibara, Satoshi Kobayashi, Kengo Sato, Tetsuro Nishino, and Etsuji Tomita (Eds.). Springer, Berlin, Heidelberg, 7–18. https://doi.org/10.1007/11872436_2