Machine Learning with Kernel Methods
Kaggle Challenge 2022-2023

Sébastien Meyer
Team: "Attention derrière"
École Normale Supérieure Paris-Saclay, France

This report details the different approaches and strategies I have developed in order to get the best possible score I could during the Kaggle competition. It has been a really challenging project for me. In order to best describe how I tackled this problem, my report will describe both the algorithms and the kernels which I implemented by hand. Finally, I will describe the final model which brought me to a public score of 0.89480. The code is available at: `https://github.com/sebastienmeyer2/molecule-type-prediction`.

1 Data

The dataset is made up of 6,000 training and 2,000 test graphs representing molecules and our task is a binary classification task regarding a certain property of these molecules. The training set is imbalanced, with positive class accounting for only 555 of the 6,000 training points.

I noticed that the dataset was not clean, with a few molecules having no edges at all both in the training and the test sets. Moreover, many molecules are not connected. Therefore, I transformed all the input graphs to only work with their largest connected components.

2 Method

In this section, I describe both the algorithms and kernels I used. A final subsection details the label enrichment procedures.

2.1 Algorithms

For a given kernel matrix K, I solved convex optimization problems with solution α using *scipy* and *cvxopt*. The final predictions are made through the computation of $K \times \alpha$.

WKRR. Weighted Kernel Ridge Regression or WKRR solves the classical penalized weighted linear regression problem with kernels:

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{n} {}^t(K\alpha - y)W(K\alpha - y) + \lambda {}^t\alpha K\alpha,$$

where W denotes the matrix of weights and λ the regularization parameter. I set the gradient to zero and I computed α by solving the resulting linear system.

KLR. Kernel Logistic Regression or KLR solves the classical logistic regression problem with kernels:

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \ell_{logistic}(y_i [K\alpha]_i) + \frac{\lambda}{2} {}^t\alpha K\alpha.$$

I implemented both Newton's method and iterative reweighted least-square, the latter proved more efficient.

SVC. Support Vector Classifier or SVC solves the classical support vector machine classification problem with kernels:

$$\max_{\alpha \in \mathbf{R}^n} 2{}^t\alpha y - {}^t\alpha K\alpha,$$

$$s.t. \;\; 0 \leq y_i \alpha_i \leq \frac{1}{C}, \forall i = 1..n$$

where the above formula represents the dual formulation of the optimization problem and C denotes the regularization parameter. I solved it through quadratic programming.

2.2 Kernels

Let ϕ_σ denote the classical gaussian kernel in $\mathbb{R}^d$ with bandwidth parameter σ. Let us denote by $G = (V, E)$ and $G' = (V', E')$ two graphs. The following kernels are designed to operate either on the base graphs or on applied transformations.

Counting Kernel. The counting kernel is a very basic kernel which creates the vectors $f = (|V|, |E|)$ and $f' = (|V'|, |E'|)$.

Node Histogram Kernel†. Assume that there are d labels in total in both G and G' and that we have a mapping function ℓ from labels to integers. Then, the node histogram kernel is based on the vectors $f = (f_1, ..., f_d)$ where $f_i = \#\{v \in V | \ell(v) = i\}$.

Edge Histogram Kernel. Same thing as the node histogram kernel but with edge labels.

For all these three kernels, the final computation is done through gaussian kernel $K(G, G') = \phi_\sigma(f, f')$.

Order Walk Kernel†. The order walk kernel refers to the number of common walks of length k which can be extracted in both graphs. I opted for the direct computation of all walks manually. Then, I extracted the corresponding labels sequences of vertices and edges and I expressed the kernel as the number of common labels sequences in both graphs *without multiplicity*.

Geometric Walk Kernel†. The geometric walk kernel extends the order walk kernel to infinite number of walks. I opted for the computation of walks until a limit order p and I computed the kernel as the weighted sum of walks of length $k = 1..p$ times β^k where β is an hyperparameter.

Shortest Path Kernel†. The shortest path kernel is based on the transformation of the input graphs to their shortest path graphs S and S' through breadth-first search. Then, if we denote by (u, v) an edge in S with cost e and (u', v') an edge in S' with cost e', we add to the kernel the value $\mathbb{1}(e = e')(\mathbb{1}(u = u')\mathbb{1}(v = v') + \mathbb{1}(u = v')\mathbb{1}(v = u'))$. The final kernel value is the sum over all edges in both S and S'.

Sum Kernel. Let us denote by m the number of kernels. We define the sum kernel as $K = \sum_{i=1}^{m} K_i$. From a theoretical point of view, the feature map ϕ in the sum kernel is equivalent to the concatenation of the different feature maps ϕ_i from the subkernels, thus encoding more information.

Normalized Kernel. For any kernel K, I noticed that the algorithm would overfit less when using its normalized version:

$$\widetilde{K}(G, G') = \frac{K(G, G')}{\sqrt{K(G, G)K(G', G')}}.$$

2.3 Label enrichment

All kernels labeled with a dagger † are based on vertices labels. Instead of using the default vertices which represent atoms, we can perform label enrichment.

Morgan (M) indices. At each iteration (starting with all ones), labels are replaced by the sum of their neighboring labels. Kernels are then computed using final labels.

Weisfeiler-Lehman (WL) indices. At each iteration (starting with base labels), labels are replaced by a hash uniquely describing their neighboring labels. Compressed labels represent full subtree patterns. Kernels are then computed using labels generated at each iteration.

3 Results

I implemented an automated gridsearch pipeline using *optuna* package to select the parameters of any algorithm and kernels I decided to use. The cross-validation was 5 folds large, with ROC AUC as target. Typically, I would run a gridsearch for 20 to 50 trials, depending on the complexity of the model. In **Table 1**, I present my best submission model which allowed me to reach a public score of 0.89480. The algorithm I used was **KLR** with a regularization parameter of $\lambda = 1.73 \times 10^{-5}$.

Kernel	Enrichment	Parameters
Counting	$\emptyset$	$\sigma = 4.96$
Edge histogram	$\emptyset$	$\sigma = 4.96$
Node histogram	$\emptyset$	$\sigma = 4.96$
	$1 \times$ M	$\sigma_M = 4.96$
	$1 \times$ WL	$\sigma_{WL} = 4.96$
Geometric walk	$\emptyset$	$p = 4$, $\beta = 2.34$
	$1 \times$ M	$p_M = 4$, $\beta_M = 1.89$
	$1 \times$ WL	$p_{WL} = 4$, $\beta_{WL} = 0.23$
Shortest path	$\emptyset$	
	$1 \times$ M	$\emptyset$
	$1 \times$ WL	

Table 1: Description of the best submission model.

Conclusion

To conclude, I have learnt some practical skills from this project. Despite my satisfactory results, I think that things can still be done to improve my results. If I had more time, I would have tried to build more kernels, especially subtree kernels. Furthermore, there must be other models or kernels that other teams have thought about and that I did not try!