Data Challenge - Kernel methods

Ali Haidar[*], **Jean-François Planchat**[†]

https://github.com/AliHaidar97/MVA-Kernel-Challenge

April 14, 2023

1. Introduction

Kernel methods have become a popular approach for graph-based data analysis, including molecule binary classification tasks. In this task, the goal is to predict whether a molecule has a particular property or not. The use of kernel methods in graph-based data analysis is based on the concept of graph kernels, which measure the similarity between graphs.

One common way to use kernel methods for molecule binary classification is to represent each molecule as a graph, where atoms are represented as nodes and bonds between atoms are represented as edges. This representation allows the use of graph kernels to measure the similarity between molecules. However, the complexity of molecular structures and the large number of features that can be extracted from them make this task challenging.

2. Graph Kernels

Here is a list of the different kernels we experimented.

2.1. Walk Kernels without tottering

These are kernels k defined by $k(G_1, G_2) = \langle (\Phi_s(G_1))_{s\in\mathcal{S}} \ , \ (\Phi_s(G_2))_{s\in\mathcal{S}} \rangle$ where:

- $\mathcal{S}$ is the set of *labelled sequences*,
- for each labelled sequence $s \in \mathcal{S}$ and graph G,

$$\Phi_s(G) = \sum_{w\in\mathcal{W}_s} \lambda_G(w)$$

with $\mathcal{W}_s$ the set of all walks in G having s as labelled sequence and $\lambda_G(w)$ a *weight* associated to the walk w in G.

The procedure to compute $k(G_1, G_2)$ involves generating all the breadth-first search (BFS) paths starting from any node in each graph, and saving each path as a string in a dictionary. Then, to compute $k(G_1, G_2)$, the frequency dictionaries are multiplied.

The time complexity of this algorithm is $\mathrm{O}(K^2 * n^2)$, where K represents the number of graphs and n is the maximum number of nodes in a graph.

n-th order Walk Kernel

Here $\lambda_G(w) = 1_{|w|=n}$.

Random Walk Kernel

Here $\lambda_G(w)$ is the probability of picking walk w in G [Kashima et al., 2003]. Its *infinite length* version is known as *geometric walk kernel* [Gärtner et al., 2003] and is defined by $\lambda_G(w) = \beta^{|w|}$ with $|\beta| < 1$ a parameter. To implement this approach, we computed the product graph using hashing.

[*] ali.haidar@polytechnique.edu
[†] jf.planchat@gmail.com

2.2. Shortest Path Kernel

Walks are replaced by shortest paths between nodes [Borgwardt and Kriegel, 2005]. More precisely, a graph G is represented as the tuple $(\Phi(G)s)s \in \mathcal{S}$, where $\Phi(G)_s$ is the number of shortest paths in G that have s as a labeled sequence. We computed the shortest path using Dijkstra's algorithm from all nodes, then hashed these paths and saved each path as a string in a dictionary. To compute $k(G_1, G_2)$, we multiply the frequency dictionaries.

3. Kernel tricks

Morgan Index

A way of enriching a graph G is, for each node, to replace its label l by (l, l') where l' is the sum of the labels of its neighbors. It makes it easier to distinguish two non-isomorphic graphs G_1 and G_2. In particular, it lowers down the size of the product graph $G_1 \times G_2$, thus speeding up the computation of $k(G_1, G_2)$ for k a walk kernel [Mahé et al., 2004].

Tanimoto Kernel

The Tanimoto normalization k^t of a kernel k is defined by $k^t(G_1, G_2) = \frac{k(G_1,G_2)}{k(G_1,G_1)+k(G_2,G_2)-k(G_1,G_2)}$.

It is a way measuring the similarity between the features that k extracts from G_1 and G_2 and shown to improve regression tasks [Ralaivola et al., 2005].

4. Experimental setting

We implemented SVM as the main model for classification.

Boostrap

To address the issue of unbalanced data, we divided the dataset into nine different subsets, each containing an equal number of samples for each label. This process created balanced datasets that could be used for training and testing the SVM model. After training an SVM model on each subset, we averaged the predictions from each model to obtain the final prediction.

Non-Weighted SVM

We performed classification directly on the full dataset without any weighting or balancing techniques.

5. Results

We used repeated k-fold cross-validation as a means to assess the performance of our algorithm.

Method	Cross-Val	Public	Private
Shortest-Path	0.912	0.89424	-
Walk-Kernel	0.92	0.89721	-
Shortest-Path + Walk-Kernel	0.922	0.89803	-
Shortest-Path + Walk-Kernel + Boostrap	0.929	0.90063	-

6. Conclusion

In conclusion, we observed an improvement in the classification score when we merged multiple kernels and employed bootstrap techniques.

It is worth to mention that Morgan index played a crucial role in enhancing the results, and we attempted to merge multiple Morgan indices. However, we found that performing the merging process only once produced the best improvement in the score.

For the final submission, we adopted a two-step approach. Firstly, we merged the shortest path and walk kernel after calculating the Morgan index, and then we normalized the resulting kernel using the Tanimoto Kernel. Secondly, we combined the solution of the full dataset with the bootstrap method to obtain the best classification score. This approach was selected as it demonstrated superior performance compared to other techniques we tested.

References

[Borgwardt and Kriegel, 2005] Borgwardt, K. M. and Kriegel, H.-P. (2005). Shortest-path kernels on graphs. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, page 74–81, USA. IEEE Computer Society.

[Gärtner et al., 2003] Gärtner, T., Flach, P., and Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In Schölkopf, B. and Warmuth, M. K., editors, *Computational Learning Theory and Kernel Machines — Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop (COLT/Kernel 2003) August 24-27, 2003, Washington, DC, USA*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143. Springer, Berlin–Heidelberg, Germany.

[Kashima et al., 2003] Kashima, H., Tsuda, K., and Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 321–328. AAAI Press.

[Mahé et al., 2004] Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., and Vert, J.-P. (2004). Extensions of marginalized graph kernels.

[Ralaivola et al., 2005] Ralaivola, L., Swamidass, S., Saigo, H., and Baldi, P. (2005). Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110.